K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONTENTS

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109

DEPARTMENT OF COMPUTER SCIENCEAND ENGINEERING

Course File

21CS52 – Computer Networks

V Sem 'B' sec 2023-24

Faculty In-charge

Mrs. Sushmitha Suresh

Asst. Professor

Dept of Computer Science and Engineering

KS School of Engineering & Management, Bangalore

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT

VISION

To impart quality education in engineering and management to meet technological, business and societal needs through holistic education and research.

MISSION

K.S. School of Engineering and Management shall,

- Establish state-of-art infrastructure to facilitate effective dissemination of technical and Managerial knowledge.
- Provide comprehensive educational experience through a combination of curricular and Experiential learning, strengthened by industry-institute-interaction.
- Pursuesocially relevant research and disseminate knowledge.
- Inculcate leadership skills and foster entrepreneurial spirit among students.

Department of Computer Science and Engineering

VISION

To produce quality Computer Science professional, possessing excellent technical knowledge, skills, personality through education and research.

MISSION

Department of Computer Science and Engineering shall,

- Provide good infrastructure and facilitate learning to become competent engineers who meet global challenges.
- Encourages industry instituteinteraction to give an edge to the students.
- Facilitates experimental learning through interdisciplinary projects.
- Strengthen soft skill to address global challenges.

COMPUTER NETWORKS

Course Code:	21CS52	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40T + 20P	Total Marks	100
Credits	04	Exam Hours	03

Course Objectives:

CLO 1. Fundamentals of data communication networks.
CLO 2. Software and hardware interfaces
CLO 3. Application of various physical components and protocols
CLO 4. Communication challenges and remedies in the networks.

Teaching-Learning Process (General Instructions)

These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes.

1. Lecturer method (L) need not to be only traditional lecture method, but alternative effective teaching methods could be adopted to attain the outcomes.
2. Use of Video/Animation to explain functioning of various concepts.
3. Encourage collaborative (Group Learning) Learning in the class.
4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.
5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop design thinking skills such as the ability to design, evaluate, generalize, and analyze information rather than simply recall it.
6. Introduce Topics in manifold representations.
7. Show the different ways to solve the same problem and encourage the students to come up with their own creative ways to solve them.
8. Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students' understanding.

Module-1

Introduction to networks: Network hardware, Network software, Reference models,

Physical Layer: Guided transmission media, Wireless transmission

Textbook 1: Ch.1.2 to 1.4, Ch.2.2 to 2.3

Laboratory Component:

1. Implement Three nodes point – to – point network with duplex links between them for different topologies. 1Set the queue size, vary the bandwidth, and find the number of packets dropped for various iterations.

Teaching-Learning Process	Chalk and board, Problem based learning, Demonstration

Module-2

The Data link layer: Design issues of DLL, Error detection and correction, Elementary data link protocols, Sliding window protocols.

The medium access control sublayer: The channel allocation problem, Multiple access protocols.

Textbook 1: Ch.3.1 to 3.4, Ch.4.1 and 4.2

Laboratory Component:

1. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the throughput with respect to transmission of packets
2. Write a program for error detecting code using CRC-CCITT (16- bits).

Teaching-LearningProcess	Chalkandboard,Problembasedlearning,Demonstration

Module-3

TheNetworkLayer:
NetworkLayerDesignIssues,RoutingAlgorithms,CongestionControlAlgorithms,QoS.

Textbook1:Ch5.1to5.4

LaboratoryComponent:

1. Implementtransmissionofpingmessages/tracerouteoveranetworktopologyconsistingof6nodesandfindthenumberofpacketsdroppedduetocongestioninthenetwork.
2. Writeaprogramtofindtheshortestpathbetweenverticesusingbellman-fordalgorithm.

Teaching-LearningProcess	Chalkandboard,Problembasedlearning,Demonstration

Module-4

TheTransportLayer: TheTransportService,Elementsoftransportprotocols,Congestioncontrol,Theinternettransportprotocols.

Textbook1:Ch6.1to6.4and6.5.1to6.5.7

LaboratoryComponent:

1. ImplementanEthernetLANusingnnodesandsetmultipletrafficnodesandplotcongestionwindowfordifferentsource/destination.
2. Writeaprogramforcongestioncontrolusingleakybucketalgorithm.

Teaching-LearningProcess	Chalkandboard,Problembasedlearning,Demonstration

Module-5

ApplicationLayer: PrinciplesofNetworkApplications,TheWebandHTTP,ElectronicMailintheInternet,DNS—TheInternet'sDirectoryService.

Textbook2:Ch2.1to2.4

Teaching-LearningProcess	Chalkandboard,Problembasedlearning,Demonstration

CourseOutcomes(CourseSkillSet)

Attheendofthecoursethestudentwillbeableto:

CO1.Learnthebasicneedsofcommunicationsystem.
CO2.Interpretthecommunicationchallengesanditssolution.
CO3.IdentifyandorganizethecommunicationsystemnetworkcomponentsCO4.Designcommunicationnetworksforuserrequirements.

AssessmentDetails(bothCIEandSEE)

TheweightageofContinuousInternalEvaluation(CIE)is50%andforSemesterEndExam(SEE)is50%.TheminimumpassingmarkfortheCIEis40%ofthemaximummarks(20marks).Astudentshallbedeemedtohavesatisfiedtheacademicrequirementsandearnedthecreditsallottedtoeachsubject/courseifthestudentsecuresnotlessthan35%(18Marksoutof50)inthesemester-endexamination(SEE),andaminimumof40%(40marksoutof100)inthesumtotaloftheCIE(ContinuousInternalEvaluation)andSEE(SemesterEndExamination)takentogether

ContinuousInternalEvaluation:

ThreeUnitTestseachof**20Marks(duration01hour)**

1. Firsttestattheendof5thweekofthesemester
2. Secondtestattheendofthe10thweekofthesemester
3. Thirdtestattheendofthe15thweekofthesemesterTwoassignmentseachof**10Marks**
4. Firstassignmentattheendof4thweekofthesemester
5. Secondassignmentattheendof9thweekofthesemester

PracticalSessionsneedtobeassessedbyappropriaterubricsandviva-vocemethod.Thiswillcontributeto**20marks**.

Note:Minimumof80%ofthelaboratorycomponentshavetobecovered.

- RubricsforeachExperimenttakenaverageforallLabcomponents–15Marks.
- Viva-Voce–5Marks(moreemphasizedondemonstrationtopics)

Thesumofthreetests,twoassignments,andpracticalsessionswillbeoutof100marksandwillbe **scaleddownto50marks**
(tohavealessstressedCIE,theportionofthesyllabusshouldnotbecommon/repeatedforanyofthemethodsoftheCIE.EachmethodofCIEshouldhaveadifferentsyllabusportionofthecourse).
CIEmethods/questionpaperhastobedesignedtoattainthedifferentlevelsofBloom'staxonomyaspertheoutcomedefinedforthecourse.

SemesterEndExamination:

TheorySEEwillbeconductedbyUniversityasperthescheduledtimetable,withcommonquestionpapersforthesubject(**duration03hours)**

1. Thequestionpaperwillhavetenquestions.Eachquestionissetfor20marks.
2. Therewillbe2questionsfromeachmodule.Eachofthetwoquestionsunderamodule(withamaximumof3sub-questions),**shouldhaveamixoftopics**underthatmodule.

Thestudentshavetoanswer5fullquestions,selectingonefullquestionfromeachmodule

SuggestedLearningResources:

Textbooks:

1. Computer-Networks-AndrewS.TanenbaumandDavidJ.Wetherall,PearsonEducation,5th-Edition.(www.pearsonhighered.com/tanenbaum)
2. ComputerNetworkingATop-DownApproach-JamesF.KuroseandKeithW.RossPearsonEducation7thEdition.

ReferenceBooks:

1. BehrouzAForouzan,DataandCommunicationsandNetworking,FifthEdition,McGrawHill,IndianEdition
2. LarryLPetersonandBrusceSDavie,ComputerNetworks,fifthedition,ELSEVIER

WeblinksandVideoLectures(e-Resources):

1. https://www.digimat.in/nptel/courses/video/106105183/L01.html
2. http://www.digimat.in/nptel/courses/video/106105081/L25.html
3. https://nptel.ac.in/courses/106105081
4. VTUe-ShikshanaProgram

ActivityBasedLearning(SuggestedActivitiesinClass)/PracticalBasedlearning

SimulationofPersonalareanetwork,Homeareanetwork,achieveQoSetc.

Note: FortheSimulationexperimentsmodifythetopologyandparameterssetfortheexperimentandtakemultipleroundsofreadingandanalyzetheresultsavailableinlogfiles.PlotnecessarygraphsandconcludeusingNS2.Installationprocedureoftherequiredsoftwaremustbedemonstrated,carriedout ingroups,anddocumentedinthereport.NonsimulationprogramscanbeimplementedusingJava

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT
BENGALURU-560109

TENTATIVE CALENDAR OF EVENTS: V ODD SEMESTER (2023-2024)

SESSION: OCT 2023 TO MAR 2024

Week No.	Month	Day						Days	Activities
		Mon	Tue	Wed	Thu	Fri	Sat		
1	NOV						**25***	1	25*-Commencement of V sem 25- Wednesday Time Table
2	NOV/DEC	27	28	29	30H	1	2DH	4	30- Kanakadasa Jayanti
3	DEC	4	5	6	7	**8**	9	6	9- Tuesday Time Table
4	DEC	11	12	13	14	15	16DH	5	
5	DEC	18	19	**20**	21	22	23TA	6	23- Monday Time Table
6	DEC	25 H	**26 T1**	**27 T1**	**28 T1**	29	30	5	25- Christmas 30 - Monday Time Table
7	JAN	1 BV	2 ASD	**3 *FFB1**	4	5	6DH	5	3 - First Faculty Feed Back
8	JAN	**8**	9	10	11	12	13	6	13- Tuesday Time Table
9	JAN	15 H	16	17	18	19	20DH	4	15- Makara Sankranti
10	JAN	22	23	24	25	26 H	27 TA	5	26 - Republic Day 27- Monday Time Table
11	JAN / FEB	**29T2**	**30 T2**	**31 T2**	1	2 BV	3DH	5	
12	FEB	5 ASD	**6 *FFB2**	7	8	9	10	6	6 - Second Faculty Feed Back 10 - Monday Time Tabie
13	FEB	12	13	**14**	15	16	17DH	5	
14	FEB	19	20	21	22	23	24 TA	6	24- Thursday Time Table
15	FEB /MAR	**26 T3**	**27 T3**	**28 T3**	29 LT	1 LT	2DH	5	
16	MAR	4 LT	5 LT	6	7	8 H	**9***	5	8- Maha Shivratri 9- Friday Time Table **9* - Last working Day**

Total No of Working Days : 79

Total Number of working days (Excluding holidays and Tests)=66

H	Holiday
BV	Blue Book Verification
T1,T2	Tests 1,2
ASD	Attendance & Sessional Display
DH	Declared Holiday
LT	Lab Test
TA	Test attendance

Monday	14
Tuesday	13
Wednesday	13
Thursday	13
Friday	13
Total	**66**

SIGNATURE OF PRINCIPAL

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

Note: INTERNSHIP : 25th October 2023 to 23rd November 2023

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SESSION: 2023-2024 (ODD SEMESTER)
CLASS TIME TABLE
(w.e.f. 27/11/2023)

Class: V CSE 'B'
Lecture Hall: LH- 414
Class Teacher: Mrs. Sushmitha Suresh

DAY	8.40-9.35	9.35-10.30	10.30-10.45	10.45 -11.40	11.40-12.35	12.35-1.20	1.20 -2.10	2.10-3.00	3.00-3.50
MONDAY	AIML (21CS54)	ATC (21CS51)	TEA BREAK	CNS (21CS52)	RMIP (21XX56)	LUNCH BREAK	Internship / Skill Lab		
TUESDAY	RMIP (21XX56)	ATC (21CS51)		DBMS (21CS53)	CNS (21CS52)		DBMS (21CS53)	AIML (21CS54)	Tutorial
WEDNESDAY	DBMS Lab Batch - B1 CNS Lab Batch - B2 AJS Lab Batch - B3				RMIP * (21XX56)		CNS (21CS52)	ATC (21CS51)	Tutorial
THURSDAY	DBMS (21CS53)	ATC (21CS51)	TEA BREAK	RMIP (21XX56)	AIML (21CS54)		DBMS Lab Batch - B2 CNS Lab Batch - B3 AJS Lab Batch - B1		
FRIDAY	DBMS Lab Batch - B3 CNS Lab Batch - B1 AJS Lab Batch - B2				EVS (21CIV57)		AIML (21CS54)	DBMS * (21CS53)	CNS (21CS52)
SATURDAY	AS PER CALENDAR OF EVENTS								

CODE	SUBJECT	HOURS /WEEK	STAFF
21CS51	Automata Theory and compiler Design	4	Dr. K Venkata Rao
21CS52	Computer Networks	4	Mrs. Sushmitha Suresh
21CS53	Database Management Systems	4	Mrs. Sougandhika Narayan
21CS54	Artificial Intelligence and Machine Learning	4	Mrs. Deepashree N
21XX56	Research Methodology & Intellectual Property Rights	4	Mrs. Meena G
21CIV57	Environmental Studies	1	Mrs. Ashvini C
21CS52	Computer Networks Laboratory	3	Mrs. Sushmitha Suresh & Mrs. Chandana V S
21CSL55	Database Management Systems Laboratory with Mini Project	3	Mrs. Sougandhika Narayan & Mrs. Deepashree N
21CSL581	Angular JS and Node JS	3	Mrs. Nita Meshram & NS1

Time-table Coordinator

Head of the Department

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Dr. K. RAMA NARASIMHA
Principal
Principal/Director
K S School of Engineering and Manage
Bengaluru - 560 109

KSSEM

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SESSION: 2023-2024(ODD SEMESTER)

(w. e. f 27/11/2023)

INDIVIDUAL TIME TABLE

Class: V A & B

Faculty Name: Mrs. Sushmitha Suresh

DAY	8.40-9.35	9.35-10.30	10.30 -10.45	10.45 -11.40	11.40-12.35	12.35-1.20	1.20 -2.10	2.10-3.00	3.00-3.50
MONDAY	RMIP (V A)		TEA BREAK	CNS (VB)		LUNCH BREAK	CN Lab Batch - A2		
TUESDAY	CN Lab Batch - A3				CNS (VB)				
WEDNESDAY	CN Lab Batch - B2	RMIP (V A)	CN Lab Batch - B2				CNS (VB)		
THURSDAY	CN Lab Batch - A1				RMIP (V A)		CN Lab Batch - B3		
FRIDAY	CN Lab Batch - B1						RMIP (V A)		CNS (VB)
SATURDAY	AS PER CALENDAR OF EVENTS								

CODE	SUBJECT	Hours /Week	
21CS52	Computer Networks	4	Mrs. Sushmitha Suresh
21XX56	Research Methodology & Intellectual Property Rights	4	
21CS52	Computer Networks Laboratory	9	
18CSP77	Project Work Phase -1	1.5	

Time-table Coordinator

Head of the Department

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management

K. Rama

Dr. K. RAMA NARASIMHA
Principal
Principal/Director
K S School of Engineering and Managem
Bengaluru - 560 109

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU -560 019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

V Semester - B Student List

Sl. No.	USN	Name of the Student
1	1KG21CS064	MANASVI K M
2	1KG21CS065	MANYA R
3	1KG21CS066	MEGHANA G
4	1KG21CS067	MOHITH KUMAR Y V
5	1KG21CS068	MYTHRI Y
6	1KG21CS069	NAMITH U
7	1KG21CS070	NANDANA M
8	1KG21CS071	NANNURU JASWANTH
9	1KG21CS072	NAVACHANDU N
10	1KG21CS073	NAVEENKANTH S
11	1KG21CS074	NEHA U
12	1KG21CS075	NIKITHA L
13	1KG21CS076	NIKITHA N M
14	1KG21CS077	NILESH RAJAN
15	1KG21CS078	NITHIN B
16	1KG21CS079	NITHIN NAIDU S P
17	1KG21CS080	P HEMANTH KUMAR
18	1KG21CS081	PACHA BHARADWAZ
19	1KG21CS082	PATIL ADESH WAMANRAO
20	1KG21CS083	PAVAN R
21	1KG21CS084	POSINA LIKHITHA
22	1KG21CS085	PRAKRIT R ARITAS
23	1KG21CS086	PRANAVA S BHAT
24	1KG21CS087	PRERANA B
25	1KG21CS088	R H HEMANTH KUMAR
26	1KG21CS089	RACHANA R
27	1KG21CS090	RAMKI G K L
28	1KG21CS091	RAMU T N
29	1KG21CS092	REKHA RATHOD
30	1KG21CS093	ROHIT L

31	1KG21CS094	S SAI KIRAN
32	1KG21CS095	SAGAR B
33	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND
34	1KG21CS097	SAHIL CHALWA
35	1KG21CS098	SAIRAMA NAIDU MALLARAPU
36	1KG21CS099	SANJANA S VASISTA
37	1KG21CS100	SANJANA U REVANKAR
38	1KG21CS101	SHARMILA K P
39	1KG21CS102	SINCHANA S
40	1KG21CS103	SIRISHA M
41	1KG21CS104	SIRISHA T
42	1KG21CS106	SUDARSHAN S KAKALWAR
43	1KG21CS108	SURAVI H U
44	1KG21CS109	SWATHI S
45	1KG21CS110	T CHAITANYA KRISHNA
46	1KG21CS111	TANISH JAIN
47	1KG21CS112	TATHI REDDY GIRIDHAR REDDY
48	1KG21CS113	UDAY L
49	1KG21CS114	VAIBHAV ES
50	1KG21CS115	VAISHNAVI U KULKARNI
51	1KG21CS116	VAMSHI KRISHNA R
52	1KG21CS117	VARSHA K R
53	1KG21CS118	VEEKSHITH V
54	1KG21CS119	VINAY M
55	1KG21CS120	VINEETHA N
56	1KG21CS121	VINUTHA C M
57	1KG21CS122	VISHWAS K R
58	1KG21CS123	YASHU V D
59	1KG21CS124	YASHWANTH SAIBABA H
60	1KG21CS125	YASHWIN KUMAR R
61	1KG21CS126	YUVARAJ S
62	1KG22CS406	KIRAN KUMAR. N
63	1KG22CS407	MANOJ M K
64	1KG22CS408	NABEEL BAIG
65	1KG22CS409	RAKESH K S
66	1KG22CS410	RAKESH M
67	1KG22CS411	SHASHANK D D

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT,BENGALURU-560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
V SEMSTER B SECTION BATCH LIST 2023-2024

S.No	USN	Name of the Student	BATCH
1	1KG21CS064	MANASVI K M	Batch B1
2	1KG21CS065	MANYA R	
3	1KG21CS066	MEGHANA G	
4	1KG21CS067	MOHITH KUMAR Y V	
5	1KG21CS068	MYTHRI Y	
6	1KG21CS069	NAMITH U	
7	1KG21CS070	NANDANA M	
8	1KG21CS071	NANNURU JASWANTH	
9	1KG21CS072	NAVACHANDU N	
10	1KG21CS073	NAVEENKANTH S	
11	1KG21CS074	NEHA U	
12	1KG21CS075	NIKITHA L	
13	1KG21CS076	NIKITHA N M	
14	1KG21CS077	NILESH RAJAN	
15	1KG21CS078	NITHIN B	
16	1KG21CS079	NITHIN NAIDU S P	
17	1KG21CS080	P HEMANTH KUMAR	
18	1KG21CS081	PACHA BHARADWAZ	
19	1KG21CS082	PATIL ADESH WAMANRAO	
20	1KG21CS083	PAVAN R	
21	1KG21CS084	POSINA LIKHITHA	
22	1KG21CS085	PRAKRIT R ARITAS	
23	1KG21CS086	PRANAVA S BHAT	
24	1KG21CS087	PRERANA B	
25	1KG21CS088	R H HEMANTH KUMAR	
26	1KG21CS089	RACHANA R	
27	1KG21CS090	RAMKI G K L	
28	1KG21CS091	RAMU T N	
29	1KG21CS092	REKHA RATHOD	
30	1KG21CS093	ROHIT L	
31	1KG21CS094	S SAI KIRAN	
32	1KG21CS095	SAGAR B	
33	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT,BENGALURU-560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
V SEMSTER B SECTION BATCH LIST 2023-2024

S.No	USN	Name of the Student	BATCH
1	1KG21CS097	SAHIL CHALWA	Batch B2
2	1KG21CS098	SAIRAMA NAIDU MALLARAPU	
3	1KG21CS099	SANJANA S VASISTA	
4	1KG21CS100	SANJANA U REVANKAR	
5	1KG21CS101	SHARMILA K P	
6	1KG21CS102	SINCHANA S	
7	1KG21CS103	SIRISHA M	
8	1KG21CS104	SIRISHA T	
9	1KG21CS106	SUDARSHAN S KAKALWAR	
10	1KG21CS108	SURAVI H U	
11	1KG21CS109	SWATHI S	
12	1KG21CS110	T CHAITANYA KRISHNA	
13	1KG21CS111	TANISH JAIN	
14	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	Batch B3
15	1KG21CS113	UDAY L	
16	1KG21CS114	VAIBHAV ES	
17	1KG21CS115	VAISHNAVI U KULKARNI	
18	1KG21CS116	VAMSHI KRISHNA R	
19	1KG21CS117	VARSHA K R	
20	1KG21CS118	VEEKSHITH V	
21	1KG21CS119	VINAY M	
22	1KG21CS120	VINEETHA N	
23	1KG21CS121	VINUTHA C M	
24	1KG21CS122	VISHWAS K R	
25	1KG21CS123	YASHU V D	
26	1KG21CS124	YASHWANTH SAIBABA H	
27	1KG21CS125	YASHWIN KUMAR R	
28	1KG21CS126	YUVARAJ S	
29	1KG22CS406	KIRAN KUMAR. N	
30	1KG22CS407	MANOJ M K	
31	1KG22CS408	NABEEL BAIG	
32	1KG22CS409	RAKESAH K S	
33	1KG22CS410	RAKESH M	
34	1KG22CS411	SHASHANK D D	

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

LESSON PLAN

NAME OF THE STAFF : Sushmitha Suresh

COURSE CODE/TITLE : 21CS52/COMPUTER NETWORKS

SEMESTER/YEAR : V/III – B Sec

Sl. No.	Topic to be covered	Mode of Delivery	Teaching Aid	No. of Periods	Cumulative No. of Periods	Proposed Date	Delivery Date
	MODULE 1						
1	**Introduction to networks:** Network hardware	L+D	BB+LCD	1	1	27/11/2023	27/11/23
2	Network hardware: PAN, LAN,MAN,WAN	L+D	BB+LCD	1	2	28/11/2023	27/11/23
3	Network software	L+D	BB+LCD	1	3	29/11/2023	28/11/23
4	Network software(continued)	L+D	BB+LCD	1	4	01/12/2023	29/11/23
5	Reference models	L+D	BB+LCD	1	5	04/12/2023	29/11/23
6	Physical Layer: Guided transmission media	L+D	BB+LCD	1	6	05/12/2023	4/12/23
7	Guided transmission media(continued)	L+D	BB+LCD	1	7	06/12/2023	5/12/23
8	Wireless transmission	L+D	BB+LCD	1	8	08/12/2023	6/12/23
9	**Practical:** 1. Implement Three nodes point – to – point network with duplex links between them for different topologies. 1Set the queue size, vary	Practical	D	3	3	**B1**-01/12/2023 **B2**-29/11/2023 **B3**-07/12/2023	B1-1/12/23 8/12/23 B2-29/11/23 6/12/23 B3-7/12/23 28/12/23

	the bandwidth, and find the number of packets dropped for various iterations						
10	Tutorial	L+D	BB+LCD	1	-	09/12/2023	7/12/23
	MODULE 2						
11	**The Data link layer:** Design issues of DLL	L+D	BB+LCD	1	9	11/12/2023	8/12/23
12	Error detection and correction	L+D	BB+LCD	1	10	12/12/2023	9/12/23
13	Error detection and correction(Problems)	L+D	BB+LCD	1	11	13/12/2023	18/12/23
14	Elementary data link protocols	L+D	BB+LCD	1	12	15/12/2023	18/12/23
15	Sliding window protocols.	L+D	BB+LCD	1	13	18/12/2023	18/12/23
16	The medium access control sublayer: The channel allocation problem	L+D	BB+LCD	1	14	19/12/2023	19/12/23
17	The channel allocation problem(continued)	L+D	BB+LCD	1	15	20/12/2023	20/12/23
18	Multiple access protocols	L+D	BB+LCD	1	16	22/12/2023	20/12/23
19	**Practical:** 1. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the throughput with respect to transmission of packets. 2. Write a program for error detecting code using CRC-CCITT (16- bits).	Practical	D	3	6	**B1**-08/12/2023 15/12/2023 22/12/2023 29/12/2023 **B2**-06/12/2023 13/12/2023 20/12/2023 03/01/2024 **B3**-14/11/2023 21/12/2023 04/01/2024 11/01/2024	B1-29/12/23 12/1/23 B2-27/12/23 10/1/24 B3-30/12/23 11/1/24
20	Tutorial	L+D	BB+LCD	1	-	23/12/2023	21/12/23
	MODULE 3						

21	**The Network Layer:** Network Layer Design Issues	L+D	BB+LCD	1	17	29/12/2023	22/12/23
22	Network Layer Design Issues(continued)	L+D	BB+LCD	1	18	30/12/2023	23/12/23
23	Routing Algorithms	L+D	BB+LCD	1	19	01/01/2024	27/12/23
24	Routing Algorithms(problems)	L+D	BB+LCD	1	20	02/01/2024	29/12/23
25	Congestion Control Algorithms	L+D	BB+LCD	1	21	03/01/2024	01/1/24
26	Congestion Control Algorithms(continued)	L+D	BB+LCD	1	22	05/01/2024	08/1/24
27	QoS [Quality of Service]	L+D	BB+LCD	1	23	08/01/2024	09/1/24
28	QoS [Quality of Service] (continued)	L+D	BB+LCD	1	24	09/01/2024	10/1/24
29	**Practical:** 1. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion in the network. 2. Write a program to find the shortest path between vertices using bellman-ford algorithm	Practical	D	3	9	**B1**-05/01/2024 12/01/2024 19/01/2024 02/02/2024 **B2**-10/01/2024 17/01/2024 24/01/2024 07/02/2024 **B3**-18/01/2024 25/01/2024 01/02/2024 08/02/2024	B1- 19/1/24 16/2/24 B2- 17/1/24 31/1/24 B3- 18/1/24 15/2/24
30	Tutorial	L+D	BB+LCD	1	-	10/01/2024	13/1/24
MODULE 4							
31	**The Transport Layer:** The Transport Service	L+D	BB+LCD	1	25	12/01/2024	13/1/24
32	The Transport Service(continued)	L+D	BB+LCD	1	26	13/01/2024	16/1/24
33	Elements of transport protocols	L+D	BB+LCD	1	27	16/01/2024	17/1/24
34	Elements of transport	L+D	BB+LCD	1	28	17/01/2024	19/1/24

	protocols(continued)						
35	Congestion control	L+D	BB+LCD	1	29	19/01/2024	22/1/24
36	Congestion control(continued)	L+D	BB+LCD	1	30	22/01/2024	23/1/24
37	The internet transport protocols.	L+D	BB+LCD	1	31	23/01/2024	25/1/24
38	The internet transport protocols(continued)	L+D	BB+LCD	1	32	24/01/2024	29/1/24
39	**Practical:** 1. 1. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination. 2. Write a program for congestion control using leaky bucket algorithm	Practical	D	3	12	**B1**-09/02/2024 16/02/2024 23/02/2024 **B2**-14/02/2024 21/02/2024 **B3**-15/02/2024 22/02/2024 24/02/2024	B_1-23/2/24 1/3/24 B_2-14/2/24 21/2/24 28/2/24 B_3 22/2/24 29/2/24
40	Tutorial	L+D	BB+LCD	1	-	27/01/2024	30/1/24
	MODULE 5						
41	**Application Layer:** Principles of Network Applications	L+D	BB+LCD	1	33	02/02/2024	31/1/24
42	Principles of Network Applications(continued)	L+D	BB+LCD	1	34	05/02/2024	5/2/24
43	The Web and HTTP	L+D	BB+LCD	1	35	06/02/2024	6/2/24
44	The Web and HTTP (continued)	L+D	BB+LCD	1	36	07/02/2024	12/2/24
45	Electronic Mail in the Internet	L+D	BB+LCD	1	37	09/02/2024	13/2/24
46	Electronic Mail in the Internet(continued)	L+D	BB+LCD	1	38	10/02/2024	15/2/24
47	DNS—The Internet's Directory Service.	L+D	BB+LCD	1	39	12/02/2024	16/2/24
48	DNS—The Internet's Directory Service(continued)	L+D	BB+LCD	1	40	13/02/2024	20/2/24

49	**Practical:** Revision					**B1**-09/03/2024 **B2**-06/03/2024 **B3**-07/03/2024	
50	Tutorial	L+D	BB+LCD	1	-	14/02/2024	2|2|24
51	Revision	L+D	BB+LCD	7	-	16/02/2024 19/02/2024 20/02/2024 21/02/2024 23/02/2024 06/03/2024 09/03/2024	23|2|24 26|2|24

	Week	Remarks
Assignment 1	**4th Week –18/12/2023**	**Mode of Assignment – Written Assignment**
Assignment 2	**9th Week- 22/01/2024**	

Total No. of Lecture Hours = 40

Total No. of Tutorial Hours =12

Total No. of Practical Hours = 20

Sushmitha
Course In charge

Head of Dept

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

IQAC Coordinator

Principal

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

QUESTION BANK - 1

Computer Networks (21CS52)

Module-1

1. **Explain** network hardware based on two dimensions.
2. **Discuss** network software with relevant diagram.
3. **Summarize** OSI reference model.
4. **Explain** TCP/IP reference model.
5. **Differentiate** between OSI and TCP/IP models.
6. **Discuss** critique of OSI model and protocol.
7. **Discuss** critique of TCP/IP model and protocol.
8. **Demonstrate** various guided transmission media with a neat diagram.
9. **Explain** various wireless transmissions.

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

QUESTION BANK - 2

Computer Networks (21CS52)

Module-2

1. **Outline** data link layer design issues with relevant diagram.
2. **Illustrate** various error - correcting codes with a suitable example.
3. **Illustrate** various error - detecting codes with a suitable example.
4. **Discuss** various definitions needed in elementary data link protocols.
5. **Develop** a code for utopian simplex protocol and explain its working.
6. **Design** a code for simplex stop and wait protocol for a noisy channel and explain its working.
7. **Discuss** simplex stop and wait protocol for an error free channel.
8. **Interpret** one bit sliding window protocol with an example.
9. **Explain** Go –back –N protocol with a neat diagram.
10. **Illustrate** selective repeat protocol with a suitable example.
11. **Explain** static channel allocation problem.
12. **Discuss** assumptions for dynamic channel allocation.
13. **List** various types of aloha and **explain** each.
14. **Describe** carrier sense multiple access protocol (CSMA) and its types.
15. **Discuss** CSMA with collision detection.
16. **Summarize** the following
    i. Bit – map protocol
    ii. Token passing
    iii. Binary count down
17. **Interpret** limited – contention protocols.
18. **Discuss** wireless LAN protocol with a neat diagram.

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

QUESTION BANK - 3

Computer Networks (21CS52)

Module-3

1. **Outline** Network layer design issues with relevant diagram.
2. **Compare** virtual circuit network and datagram network.
3. **Illustrate** shortest path routing algorithm with example.
4. **Demonstrate** Dijkstra's algorithm to compute shortest path.
5. **Summarize** the concept of flooding.
6. **Interpret** distance vector routing algorithm with a suitable example.
7. **Outline** the concept of count to infinity problem.
8. **Illustrate** Link state routing algorithm with a suitable example.
9. **Illustrate** hierarchical routing with an example.
10. **Figure out** the following with an example for each:
    I. Broadcast routing
    II. Multicast routing
    III. Any cast routing
11. **Explain** routing for mobile host with a neat diagram.
12. **Discuss** routing Ad hoc networks.
13. **Interpret** various approaches to congestion control with suitable diagram.
14. **Summarize** Quality Of Services.
15. **Demonstrate** the concept of leaky bucket and token bucket algorithm.
16. **Discuss** the concept of Packet scheduling.
17. **Compare** the working of integrated services and differentiated services.

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

QUESTION BANK - 4

Computer Networks (21CS52)

Module-4

1. **Discuss** the services provided by transport layer.
2. **Summarize** the concept of transport service primitives with a neat diagram.
3. **Develop** a client and server program for transferring a whole file.
4. **Demonstrate** elements of transport protocols with a suitable diagram.
5. **Outline** connection establishment phase with a neat diagram.
6. **Outline** connection release phase with a neat diagram.
7. **Explain** the following congestion control mechanisms:
   I. Desirable bandwidth allocation.
   II. Regulating the sending rate.
   III. Wireless issue.
8. **Discuss** UDP in detail.
9. **Discuss** TCP in detail.

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

QUESTION BANK-5

COMPUTER NETWORKS(21CS52)

Module-5

1. **Explain** the client server and P2P architecture.
2. With a block diagram, **explain** how application process communicates through a socket.
3. **List** and **describe** 4 transport services available to applications.
4. **Discuss** transport services provided by the internet.
5. **Illustrate** with diagram, the working of Web and HTTP.
6. **Explain** the HTTP non-persistent and persistent connections.
7. **Discover** the HTTP request and response messages.
8. **Illustrate** how cookies are used in user server interactions.
9. With a neat diagram **describe** web caching.
10. With a neat diagram, **explain** the http request and response message formats.
11. **Illustrate** the working of FTP with a neat diagram.
12. **Illustrate** the working of SMTP with a neat diagram.
13. **Summarize** on how DNS works.
14. With a message format, **explain** the DNS messages.
15. **Compare** HTTP and SMTP.

K.S. SCHOOL OF ENGINEERING AND MANAGEMENTBANGALORE - 560109

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

CO-PO Mapping

Course: Computer Networks	
Type: Integrated Professional Core Course	**Course Code: 21CS52**

No of Hours

Theory (Lecture Class)	Tutorials	Practical/Field Work/Allied Activities	Total/Week	Total hours of Pedagogy
4	0	3	7	40 T + 20 P

Marks

CIE	SEE	Total	Credits
50	50	100	4

Aim/Objectives of the Course

1. Understand the fundamentals of data communication networks and discuss Physical layer services.
2. Discuss Data link layer services, data link protocols and software and hardware interfaces.
3. Understand Communication Challenges and remedies in the network and application of various Routing algorithms.
4. Identify various Transport layer services and Application of various physical components and protocols.
5. Demonstration of application layer protocols.

Course Learning Outcomes
After completing the course, the students will be able to

CO1	**Outline** the basic needs of communication system.	**Understanding (K2)**
CO2	**Discover** design issues in data link layer and Interpret the communication challenges and its solution.	**Applying (K3)**
CO3	**Demonstrate** how routing algorithm can be used. Identify and organize the communication system network components.	**Applying (K3)**
CO4	**Determine** transport layer services and interpret transport layer protocols.	**Applying (K3)**
CO5	**Apply** principles of application layer protocols and demonstrate knowledge in designing and implementing network protocols.	**Applying (K3)**

Syllabus Content

Module 1: Introduction to networks: Network hardware, Network software, Reference models, **Physical Layer:** Guided transmission media, Wireless transmission.	CO1 8hrs

Laboratory Experiments - 1. Implement Three nodes point – to – point network with duplex links between them for different topologies. Set the queue size, vary the bandwidth, and find the number of packets dropped for various iterations. **LO:** At the end of this session the student will be able to 1. Understand network software and hardware 2. Understand different reference models 3. Identify various guided and wireless transmission media.	PO1-3 PO3-1 PO5-1 PO10-2 PO12-1 PSO1-3 PSO2-1
Module 2: The Data link layer: Design issues of DLL, Error detection and correction, Elementary data link protocols, Sliding window protocols. **The medium access control sublayer:** The channel allocation problem, Multiple access protocols. **Laboratory Experiments**: 1. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the throughput with respect to transmission of packets. 2. Write a program for error detecting code using CRC-CCITT (16- bits). **LO:**At the end of this session the student will be able to 1. Determine data link layer design issues. 2. Understand concepts of Error correction and error detection. 3. Understanding the medium access control layer.	**CO2** 8 hrs. PO1-3 PO2-3 PO4-2 PO10-2 PO12-1 PSO1-3 PSO2-2
Module 3: The Network Layer: Network Layer Design Issues, Routing Algorithms, Congestion Control Algorithms, QoS. **Laboratory Experiments**: 1. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion in the network. 2. Write a program to find the shortest path between vertices using bellman-ford algorithm. **LO:** At the end of this session the student will be able to 1. Determine network layer design issues. 2. Determine various routing algorithms and its application. 3. Understanding various congestion control algorithms.	**CO3** 8hrs PO1-3 PO2-3 PO3-2 PO4-1 PO10-2 PO12-1 PSO1-3 PSO2-1
Module 4: The Transport Layer: The Transport Service, Elements of transport protocols, Congestion control, the internet transport protocols. **Laboratory Experiments**:	**CO4**

1. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination. 2. Write a program for congestion control using leaky bucket algorithm. **LO:** At the end of this session the student will be able to 1. Identify different transport layer services. 2. Understand transport protocols 3. Understand various congestion control mechanisms.	8 hrs PO1-3 PO2-2 PO3-1 PO10-2 PO12-1 PSO1-3 PSO2-1
Module 5:Application Layer: Principles of Network Applications, The Web and HTTP, Electronic Mail in the Internet, DNS—The Internet's Directory Service. **LO:** At the end of this session the student will be able to 1. Understand Principles of Network Application. 2. Discussion of Web, HTTP, FTP protocols.	**CO5** 8 hrs PO1-3 PO2-1 PO3-2 PO5-2 PO10-2 PO12-1 PSO1-3 PSO2-1

Text Books

1. Computer-Networks- Andrew S. Tanenbaum and David J. Wetherall, Pearson Education, 5thEdition. (www.pearsonhighered.com/tanenbaum).
2. Computer Networking A Top-Down Approach -James F. Kurose and Keith W. RossPearson Education 7th Edition.

Reference Books

1. Behrouz A Forouzan, Data and Communications and Networking, Fifth Edition, McGraw Hill,Indian Edition
2. Larry L Peterson and Brusce S Davie, Computer Networks, fifth edition, ELSEVIER

Useful Website:

1. https://www.digimat.in/nptel/courses/video/106105183/L01.html
2. http://www.digimat.in/nptel/courses/video/106105081/L25.html
3. https://nptel.ac.in/courses/106105081
4. VTU e-Shikshana Program

Useful Journals

- elsevier
- airccse
- scimag
- springer

Teaching and Learning Methods

1. Lecture class: 40 hrs
2. Tutorial classes: 12hrs
3. Practical: 20 hrs

Assessment

Type of test/examination: Written examination

Continuous Internal Evaluation(CIE) : 1) Three Tests each of 20 marks (duration 01 hour)

2) Two assignments each of 10 Marks

3) Practical Sessions for 20 Marks Rubrics for each Experiment taken average for all Lab components – 15 Marks. • Viva-Voce– 5 Marks (more emphasized on demonstration topics)
The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be scaled down to 50 marks

Total CIE: 50 Marks

Semester End Exam (SEE): 100 marks (students have to answer all main questions) which will be reduced to 50 Marks.

Test duration: 1 hrs

Examination duration: 3 hrs

CO to PO Mapping

PO1: Science and engineering Knowledge **PO2:** Problem Analysis **PO3:** Design & Development **PO4:**Investigations of Complex Problems **PO5:** Modern Tool Usage **PO6:** Engineer & Society	**PO7:**Environment and Society **PO8:**Ethics **PO9:**Individual & Team Work **PO10:** Communication **PO11:**Project Management & Finance **PO12:**Lifelong Learning

PSO1: Understand fundamental and advanced concepts in the core areas of Computer Science and Engineering to analyze, design and implement the solutions for the real world problems.

PSO2: Utilize modern technological innovations efficiently in various applications to work towards the betterment of society and solve engineering problems.

CO	PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
21CS52	K-level														
CO1	K2	3	-	1	-	1	-	-	-	-	2	-	1	3	1
CO2	K3	3	3	-	2	-	-	-	-	-	2	-	1	3	2
CO3	K3	3	3	2	1	-	-	-	-	-	2	-	1	3	1
CO4	K3	3	2	1	-	-	-	-	-	-	2	-	1	3	1
CO5	K3	3	1	2	-	2	-	-	-	-	2	-	1	3	1

Course In charge

Head of the Department

IQAC Coordinator

K. Rama

Principal

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Dr. K. RAMA NARASIMHA
Principal/Director
K.S School of Engineering and Management

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

FIRST ASSIGNMENT

Degree	: B.E	Semester	: V
Branch	: CSE	Course Code	: 21CS52
Course Title	: Computer Networks	Max Marks	: 10
Date	: 12/12/2023	Last Date for submission	: 20/12/2023

Q No.	Questions	Marks	K-Level	CO mapping
1	**Explain** network hardware based on two dimensions.	1	Understanding K2	CO1
2	**Discuss** network software with relevant diagram.	1	Understanding K2	CO1
3	**Summarize** OSI reference model with neat diagram.	1	Understanding K2	CO1
4	**Differentiate** between OSI and TCP/IP models and **Discuss** critique of OSI model and TCP/IP model.	1	Understanding K2	CO1
5	**Explain** TCP/IP reference model with neat diagram.	1	Understanding K2	CO1
6	**Demonstrate** various guided transmission media with a neat diagram.	1	Understanding K2	CO1
7	**Explain** various wireless transmissions.	1	Understanding K2	CO1
8	**List** various data link layer design issues and **Explain** various types of framing methods.	1	Understanding K2	CO2
9	**Illustrate** various error - correcting codes with a suitable example.	1	Applying K3	CO2
10	**Illustrate** various error - detecting codes with a suitable example.	1	Applying K3	CO2

Course In charge

HOD

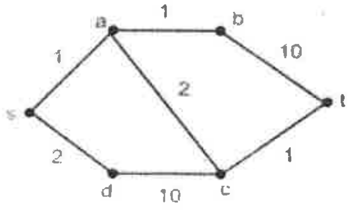
HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

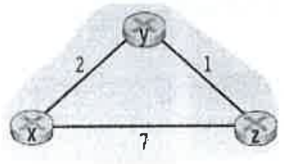
K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

SECOND ASSIGNMENT

Degree	:	**B.E**	**Semester**	:	**V**
Branch	:	**CSE**	**Course Code**	:	**21CS52**
Course Title	:	**Computer Networks**	**Max Marks**	:	**10**
Date	:	**19/1/2024**	**Last Date for submission**	:	**25/1/2024**

Q No.	Questions	Marks	K-Level	CO mapping
1	**Develop** a code for utopian simplex protocol and stop and wait protocol (Error-Free Channel).	1	Applying K3	CO2
2	**Discuss** static channel allocation problem and assumptions for dynamic channel allocation.	1	Understanding K2	CO2
3	a) **Discuss** carrier sense multiple access protocol (CSMA), ALOHA and its types with neat diagram. b) **Summarize** the following i. Bit – map protocol ii. Token passing iii. Binary count down	1	Understanding K2	CO2
4	**Illustrate** Go-Back N, selective repeat and one-bit sliding window protocols with a suitable example.	1	Applying K3	CO2
	Outline the network layer design issues with a relevant diagram	1	Understanding K2	CO3
6	**Generalize** the following a) The optimality principle b) Count to infinity problem c) Flooding	1	Understanding K2	CO3
7	a) **Apply** Dijkstra's algorithm for the following graph to find the shortest path from 's'  a b s t d c 1 1 10 2 2 1 10	1	Applying K3	CO3

	b) **Apply** Bellman ford algorithm for the following graph to find the shortest path from all nodes to destination. 			
8	**Illustrate** link state, hierarchical and broadcast routing algorithms with suitable example.	1	**Applying K3**	**CO3**
9	**Interpret** various congestion control approaches with suitable example.	1	**Applying K3**	**CO3**
10	**Explain** quality of services with respect to network layer	1	**Understanding K2**	**CO3**

Course In charge

HOD

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION: 2023-2024 (ODD SEMESTER)
I SESSIONAL TEST QUESTION PAPER
SET-A

USN										

Degree	: B.E	Semester	: V
Branch	: Computer Science & Engineering	Course Code	: 21CN52
Course Title	: Computer Networks	Date	: 03/01/2024
Duration	: 60 minutes	Max Marks	: 20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Explain** network hardware based on scale.	5	Understanding K2	CO1
(b)	**Differentiate** between OSI and TCP/IP models.	5	Understanding K2	CO1
	OR			
2(a)	**Explain** network software with relevant diagram.	5	Understanding K2	CO1
(b)	**Discuss** critique of OSI model and protocol.	5	Understanding K2	CO1
	PART-B			
3(a)	**Outline** the following i. Twisted pairs ii. Coaxial Cable iii. Magnetic Media	5	Understanding K2	CO1
(b)	**List** and **illustrate** various types of framing methods with relevant examples.	5	Applying K3	CO2
	OR			
4(a)	**Generalize** the following i. Electromagnetic Spectrum ii. Radio Transmission	5	Understanding K2	CO1
(b)	**Calculate** Checksum for the given problem. 10011001111000100010010010000100 where k=4 and m=8	5	Applying K3	CO2

(S)

Course Incharge   HOD   IQAC- Coordinator   Principal

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Manageme
Bengaluru - 560 109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109

DEPARTMENT OF CIVIL ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

I SESSIONAL TEST QUESTION PAPER

SET-B

USN										

Degree	:	B.E			
Branch	:	Computer Science & Engineering	Semester	:	V
Course Title	:	Computer Networks	Course Code	:	21CN52
Duration	:	60 Minutes	Date	:	03/01/2024
			Max Marks	:	20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Summarize** OSI reference model.	5	Understanding K2	CO1
(b)	**Discuss** critique of TCP/IP model and protocol.	5	Understanding K2	CO1
	OR			
2(a)	**Explain** TCP/IP reference model.	5	Understanding K2	CO1
(b)	**Outline** the following i. Fiber Optics ii. Fiber Cables iii. Power Lines	5	Understanding K2	CO1
	PART-B			
3(a)	**Discuss** network hardware based on transmission technology.	5	Understanding K2	CO1
(b)	**Calculate** hamming code for the pattern (11,7) at the sender side where m=1000001 and **find out** value of "t" for Reed-Solomon code (255,223).	5	Applying K3	CO2
	OR			
4(a)	**Generalize** the following i. Microwave Transmission ii. Infrared Transmission iii. Light Transmission	5	Understanding K2	CO1
(b)	**Calculate** cyclic redundancy check for the given data. Frame=1001, Generator=1011 assume no errors.	5	Applying K3	CO2

Course Incharge HOD IQAC- Coordinator Principal

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

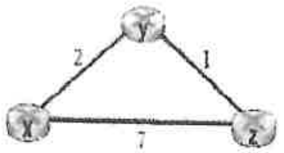
Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Managemen
Bengaluru - 560 10

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION: 2023-2024 (ODD SEMESTER)
II SESSIONAL TEST QUESTION PAPER
SET-A

USN □□□□□□□□□□

Degree	:	B.E			
Branch	:	Computer Science & Engineering	Semester	:	V
Course Title	:	Computer Networks	Course Code	:	21CN52
Duration	:	60 minutes	Date	:	08/02/2024
			Max Marks	:	20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Discuss** carrier sense multiple access protocol (CSMA), ALOHA and its types with neat diagram.	5	Understanding K2	CO2
(b)	**Outline** the network layer design issues with a relevant diagram.	5	Understanding K2	CO3
	OR			
2(a)	**Summarize** one-bit sliding window protocol with a relevant diagram.	5	Understanding K2	CO2
(b)	**Generalize** the following a) The optimality principle b) Count to infinity problem c) Flooding	5	Understanding K2	CO3
	PART-B			
3(a)	**Interpret** Go-Back N sliding window protocol with suitable example.	5	Applying K3	CO2
(b)	**Illustrate** link state and hierarchical algorithms with suitable example.	5	Applying K3	CO3
	OR			
4(a)	**Design** a TCL code to implement simple ESS with transmitting nodes in wireless LAN.	5	Applying K3	CO2
(b)	**Apply** Bellman ford algorithm for the following graph to find the shortest path from all nodes to destination. 	5	Applying K3	CO3

Course Incharge

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management

HOD

IQAC- Coordinator

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and M

Principal

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109

DEPARTMENT OF CIVIL ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

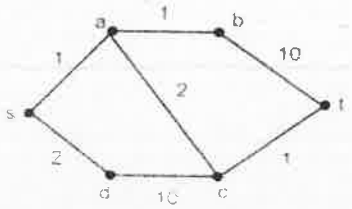
II SESSIONAL TEST QUESTION PAPER

SET-B

USN										

Degree	:	B.E	Semester	:	V
Branch	:	Computer Science & Engineering	Course Code	:	21CN52
Course Title	:	Computer Networks	Date	:	08/02/2024
Duration	:	60 Minutes	Max Marks	:	20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Discuss** static channel allocation problem and assumptions for dynamic channel allocation.	5	Understanding K2	CO2
(b)	**Explain** quality of services with respect to network layer.	5	Understanding K2	CO3
	OR			
2(a)	**Summarize** the following i. Bit – map protocol ii. Token passing iii. Binary count down	5	Understanding K2	CO2
(b)	**Describe** various congestion control approaches.	5	Understanding K2	CO3
	PART-B			
3(a)	**Illustrate** selective repeat sliding window protocol with suitable example.	5	Applying K3	CO2
(b)	**Design** a program to find the shortest path between vertices using bellman-ford algorithm.	5	Applying K3	CO3
	OR			
4(a)	**Develop** a code for utopian simplex protocol and stop and wait protocol (Error-Free Channel).	5	Applying K3	CO2
(b)	**Apply** Dijkstra's algorithm for the following graph to find the shortest path from 's'  a 1 b 1 10 2 s t 2 1 d 10 c	5	Applying K3	CO3

S

Course Incharge

HOD

Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

IQAC- Coordinator

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Management
Bengaluru - 560 109

Principal

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION: 2023-2024 (ODD SEMESTER)
III SESSIONAL TEST QUESTION PAPER
SET-A

USN										

Degree : B.E
Branch : Computer Science & Engineering
Course Title : Computer Networks
Duration : 60 minutes

Semester : V
Course Code : 21CN52
Date : 13/03/2024
Max Marks : 20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Explain** the concept of transport service primitives and Berkeley sockets with a neat diagram.	5	Understanding K2	CO4
(b)	**Discuss** client server and P2P network architecture.	5	Understanding K2	CO5
	OR			
2(a)	**Discuss** connection establishment phase with a neat diagram.	5	Understanding K2	CO4
(b)	**Explain** the HTTP non-persistent and persistent connections.	5	Understanding K2	CO5
	PART-B			
3(a)	**Design** a TCL code to implement an ethernet LAN using n nodes and set multiple traffic nodes.	5	Applying K3	CO4
(b)	**Illustrate** the working of FTP with a neat diagram.	5	Applying K3	CO5
	OR			
4(a)	**Demonstrate** UDP and RTP header with suitable diagram.	5	Applying K3	CO4
(b)	**Illustrate** http request and response message formats with a suitable example.	5	Applying K3	CO5

Course Incharge

HOD

IQAC- Coordinator

Principal

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Manageme
Bengaluru - 560 109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESSION: 2023-2024 (ODD SEMESTER)

III SESSIONAL TEST QUESTION PAPER

SET-B

USN										

Degree	:	B.E	Semester	:	V
Branch	:	Computer Science & Engineering	Course Code	:	21CN52
Course Title	:	Computer Networks	Date	:	13/03/2024
Duration	:	60 Minutes	Max Marks	:	20

Note: Answer ONE full question from each part.

Q No.	Question	Marks	K-Level	CO mapping
	PART-A			
1(a)	**Discuss** connection release phase with a neat diagram.	5	Understanding K2	CO4
(b)	**Explain** web caching and how application process communicates through a socket.	5	Understanding K2	CO5
	OR			
2(a)	**Outline** TCP header and states used in TCP connection management finite state machine with suitable diagram.	5	Understanding K2	CO4
(b)	**Explain** DNS records and messages with a message format.	5	Understanding K2	CO5
	PART-B			
3(a)	**Illustrate** the following with suitable example i. Desirable bandwidth allocation ii. Regulating the sending rate congestion	5	Applying K3	CO4
(b)	**Illustrate** the working of SMTP with a neat diagram.	5	Applying K3	CO5
	OR			
4(a)	**Design** a program to control congestion using leaky bucket.	5	Applying K3	CO4
(b)	**Illustrate** how cookies are used in user server interactions.	5	Applying K3	CO5

S

Course Incharge

HOD

IQAC- Coordinator

Principal

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Dr. K. RAMA NARASIMHA
Principal/Director
K S School of Engineering and Mana
Bengaluru - 560 109

CBCS SCHEME

USN □□□□□□□□□□

18CS52

Fifth Semester B.E. Degree Examination, July/August 2022
Computer Networks and Security

Time: 3 hrs.

Max. Marks: 100

Note: ***Answer any FIVE full questions, choosing ONE full question from each module.***

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-1

1 a. Explain the steps involved in transferring a web page from server to client in case of HTTP with non – persistent connection. Also brief the Back of the Envelope calculation for time needed to request and receive the file. **(10 Marks)**

b. Consider an e – commerce site that wants to keep a purchase record for each of its customers. Describe with neat diagram how this can be done with cookies. **(10 Marks)**

OR

2 a. Explain with neat diagram, the socket related activity of client – server communication over the TCP along with client and server code. **(10 Marks)**

b. Explain FTP with its Commands and Replies. **(10 Marks)**

Module-2

3 a. Describe the various fields of UDP segment structure. Suppose you have the following three 16 – bit words 0110011001100000 , 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 , 1000111100001100. Find the checksum. How does the receiver detect errors? Is it possible that 1 – bit errors will go undetected? **(10 Marks)**

b. Explain Sender and Receiver side Finite State Machine (FSM) representation for rdt 2.1 protocol. **(10 Marks)**

OR

4 a. Draw TCP Segment structure. Describe the various fields of TCP segment structure. **(10 Marks)**

b. Explain with neat diagram, the causes and costs of congestion considering the following scenarios.
Scenario 1 : Two sender , A Router , with Infinite Buffer.
Scenario 2 : Two sender , A Router , with Finite Buffer. **(10 Marks)**

Module-3

5 a. Write Link state Routing Algorithm. Apply it to the following graph [Refer Fig. Q5(a)] with source node as "U". Draw the least cost path tree and the forwarding table for node "U". **(10 Marks)**

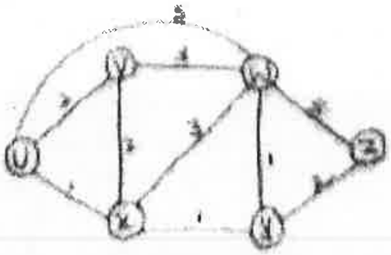


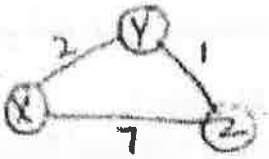
Fig. Q5(a)

b. Draw IPV4 datagram format. Mention the significance of each field. **(10 Marks)**

OR

6 a. Write distance Vector Routing Algorithm and apply it to the following graph. [Refer Fig. Q6(a)]. (10 Marks)

Fig. Q6(a)



b. Draw IPV6 datagram format. Mention the significance of each field. (10 Marks)

Module-4

7 a. Explain Diffie – Hellman Key Exchange Protocol. Suppose two parties A and B wish to set up a common secret key between themselves using Diffie Hellman Protocol selecting generator as 3 and prime number as 7. Party A chooses 2 and Party B chooses 5 as their respective secret. Find the Diffie Hellman Key. (10 Marks)

b. Explain Data Encryption Standard (DES) algorithm. (10 Marks)

OR

8 a. Explain three phases of RSA Algorithm. For an encryption of a 4 – bit message "1000" or M = 9 we choose a = 3 and b = 11. Find the Public and Private keys for this security action and show the Cipher text. (10 Marks)

b. Write short notes on :

i) Security Implementation in wireless IEEE 802.11.

ii) Firewalls.

(10 Marks)

Module-5

9 a. Explain how DNS Redirects a User's request to a CDN Server. (10 Marks)

b. Explain RTP Basics and RTP packet Header fields. (10 Marks)

OR

10 a. Explain the properties of Audio and Video. Also mention the three key distinguishing features of Streaming Stored Video. (10 Marks)

b. With neat diagram, explain Session Initiation Protocol (SIP) Call establishment. (10 Marks)

CBCS SCHEME

USN [illegible] 18CS52

Fifth Semester B.E. Degree Examination, Feb./Mar. 2022

Computer Networks and Security

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-1

1 a. Differentiate between :
(i) HTTP and FTP (ii) SMTP and HTTP (iii) UDP and TCP **(10 Marks)**

b. Explain Cookies and Web Caching with diagram. **(10 Marks)**

OR

2 a. Describe in detail the services offered by DNS and explain DNS message format. **(08 Marks)**

b. Compare HTTP and SMTP. **(04 Marks)**

c. Define Socket. Demonstrate the working of TCP-Socket. **(08 Marks)**

Module-2

3 a. With the help of FSM, describe the two states of the sender side and one state of the receiver side of rdt2.0 **(10 Marks)**

b. With a neat diagram, demonstrate the working of Go-BACK-N protocol. **(10 Marks)**

OR

4 a. Describe TCP connection management with help of diagram. **(10 Marks)**

b. Interpret the FSM to TCP congestion control. **(10 Marks)**

Module-3

5 a. Explain the Implementation of virtual circuit services in Computer Network. **(07 Marks)**

b. Explain the three Switching Techniques. **(06 Marks)**

c Explain Distance vector algorithm using three nodes network. **(07 Marks)**

OR

6 a. Explain Dijkstra's algorithm with example. **(10 Marks)**

b. Explain various broadcast routing algorithms. **(10 Marks)**

Module-4

7 a. Explain Feistel structure of DES Algorithm. **(10 Marks)**

b Explain RSA Algorithm with an example. **(10 Marks)**

OR

8 a. In the Diffie - Hellman key exchange protocol prove that the two keys k_1 and k_2 are equal. **(10 Marks)**

b. Discuss the following :
(i) Secure Hash Algorithm (ii) Firewalls. **(10 Marks)**

Module-5

9 a. Explain briefly how DNS redirects a users request to a CDN server. **(10 Marks)**

b. With neat diagram explain the naïve-architecture for audio/video streaming. **(10 Marks)**

OR

10 a. Write a short notes on :
(i) Netflix video streaming platform (ii) VOIP with Skype. **(10 Marks)**

b. With neat diagram explain the RTP header fields. **(10 Marks)**

* * * * *

CBCS SCHEME

USN | 1 | K | 4 | 2 | 1 | C | S | | | **21CS52**

Fifth Semester B.E. Degree Examination, Dec.2023/Jan.2024

Computer Networks

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-1

1 a. Define Computer Networks. Explain local area network in detail with a neat diagram. **(06 Marks)**

b. Explain MAN with a neat labelled diagram. **(06 Marks)**

c. List and explain design issues for layer. **(08 Marks)**

OR

2 a. What are guided transmission media? Explain twisted pair cable in detail. **(06 Marks)**

b. Explain TCP/IP reference model with a neat labelled diagram. **(10 Marks)**

c. Briefly discuss virtual private networks. **(04 Marks)**

Module-2

3 a. List and explain any two data link layer design issues. 1001101 **(10 Marks)**

b. A bit stream transmitted using standard CRC method. The generator polynomial is $X^3 + 1$.

i) What is actual bit string transmitted

ii) Suppose 3rd bit from the left is inverted during transmission, how will receiver detect this error? **(10 Marks)**

OR

4 a. Explain Go-Back-N protocol working. **(10 Marks)**

b. Briefly explain static channel and dynamic channel allocation problem. **(10 Marks)**

Module-3

5 a. Write an Dijkstra's algorithm to compute shortest path through graph. Explain with example. **(10 Marks)**

b. Illustrate working of OSPF and BGP. **(10 Marks)**

OR

6 a. What is congestion control? List and explain various approaches to congestion control. **(12 Marks)**

b. What is packet scheduling algorithm? Explain FIFO algorithm. **(08 Marks)**

Module-4

7 a. Write a program for congestion control using leaky bucket algorithm. **(10 Marks)**

b. Briefly explain about transport service primitives. **(10 Marks)**

OR

8 a. With a neat labelled diagram, explain TCP segment structure. **(10 Marks)**

b. Explain TCP connection management with TCP connection management FSM diagram. **(10 Marks)**

Module-5

9 a. Explain client/server and P-P architecture with a neat labelled diagram. (10 Marks)

b. Explain use and server interaction with a neat diagram. (10 Marks)

OR

10 a. Explain persistant and non persistant http in details. (10 Marks)

b. Write notes on:

(i) E-mail in the internet

(ii) Distributed DNS architecture (10 Marks)

* * * * *

CBCS SCHEME

USN □□□□□□□□□□

21CS52

Fifth Semester B.E. Degree Examination, Dec.2023/Jan.2024
Computer Networks

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-1

1 a. Define Computer Networks. Explain local area network in detail with a neat diagram. **(06 Marks)**

b. Explain MAN with a neat labelled diagram. **(06 Marks)**

c. List and explain design issues for layer. **(08 Marks)**

OR

2 a. What are guided transmission media? Explain twisted pair cable in detail. **(06 Marks)**

b. Explain TCP/IP reference model with a neat labelled diagram. **(10 Marks)**

c. Briefly discuss virtual private networks. **(04 Marks)**

Module-2

3 a. List and explain any two data link layer design issues. **(10 Marks)**

b. A bit stream transmitted using standard CRC method. The generator polynomial is $X^3 + 1$.

i) What is actual bit string transmitted

ii) Suppose 3rd bit from the left is inverted during transmission, how will receiver detect this error? **(10 Marks)**

OR

4 a. Explain Go-Back-N protocol working. **(10 Marks)**

b. Briefly explain static channel and dynamic channel allocation problem. **(10 Marks)**

Module-3

5 a. Write an Dijkstra's algorithm to compute shortest path through graph. Explain with example. **(10 Marks)**

b. Illustrate working of OSPF and BGP. **(10 Marks)**

OR

6 a. What is congestion control? List and explain various approaches to congestion control. **(12 Marks)**

b. What is packet scheduling algorithm? Explain FIFO algorithm. **(08 Marks)**

Module-4

7 a. Write a program for congestion control using leaky bucket algorithm. **(10 Marks)**

b. Briefly explain about transport service primitives. **(10 Marks)**

OR

8 a. With a neat labelled diagram, explain TCP segment structure. **(10 Marks)**

b. Explain TCP connection management with TCP connection management FSM diagram. **(10 Marks)**

Module-5

9 a. Explain client/server and P-P architecture with a neat labelled diagram. (10 Marks)

b. Explain use and server interaction with a neat diagram. (10 Marks)

OR

10 a. Explain persistant and non persistant http in details. (10 Marks)

b. Write notes on:

(i) E-mail in the internet

(ii) Distributed DNS architecture (10 Marks)

* * * * *

Module-I

Contents:

INTRODUCTION

- Each of the past three centuries was dominated by a single new technology.
- The 18th century was the era of the great mechanical systems accompanying the Industrial Revolution.
- The 19th century was the age of the steam engine.
- During the 20th century, the technology was information gathering, processing, and distribution. Ex: Telephone networks, invention of radio and computer industry, communication satellites, Internet.
- The 21st century Networks are Human-to-Human, Machine-to-Machine.
- **Computer Network:** A collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection need not be via a copper wire; fiber optics, microwaves, infrared and communication satellites can also be used. Networks come in many sizes, shapes and forms, as we will see later. They are usually connected together to make larger networks, for example **Internet.**

Uses of Computer Networks

A) **Business Applications:** Resource Sharing, Client-Server-mail, video conferencing

- Many companies have a substantial number of computers. For example, a company may have separate computers to monitor production, keep track of inventories, and do the payroll. Initially, each of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to extract and correlate information about the entire company.

1. **Resource sharing:** The main task of the connectivity of resources is resource sharing. For example, a high-volume networked printer may be installed instead of large collection of individual printers.

2. **Information Sharing:** large and medium-sized company and many small companies are vitally dependent on computerized information. This can be done by a simple client server model connected by network as illustrated in Fig.1.4.

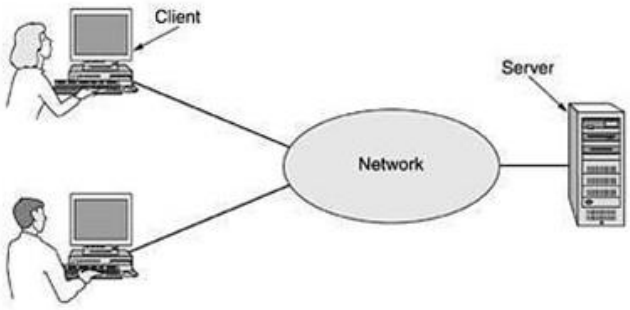


Figure 1.4 A network with two clients and one server

In client-server model in detail, two processes are involved, one on the client machine and one on the server machine. Communication takes the form of the client process sending a message over the network to the server process. The client process then waits for a reply message. When the server process gets the request, it performs the requested work or looks up the requested data and sends back a reply. These messagesare shown in Fig. 1.5.

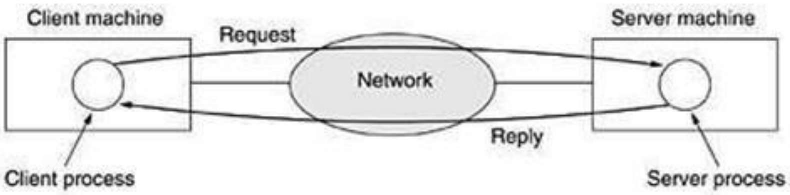


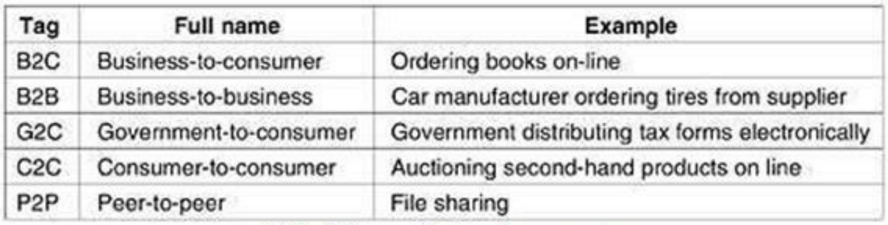
Figure 1.5 Client-server model involves requests and replies

3. **Connecting People:** another use of setting up a computer network has to do with people rather than information or even computers. It is achieved through Email, Video Conferencing.
4. **E-commerce:** many companies is doing business electronically with other companies, especially suppliers and customers, and doing business with consumers over the Internet.

B) **Home Applications:** Shopping, Digital library, email, game playing, TV,Twitter, Instagram

The computer network provides better connectivity for home applications via desktop computers, laptops, iPads, iPhones. Some of the more popular uses of the Internet for home users are as follows:

1. Access to remote information.
2. Person-to-person communication (peer-to-peer).
   i. Peer-to-peer - there are no fixed clients and servers.
   ii. Audio and Video sharing
3. Interactive entertainment.
4. Electronic commerce.



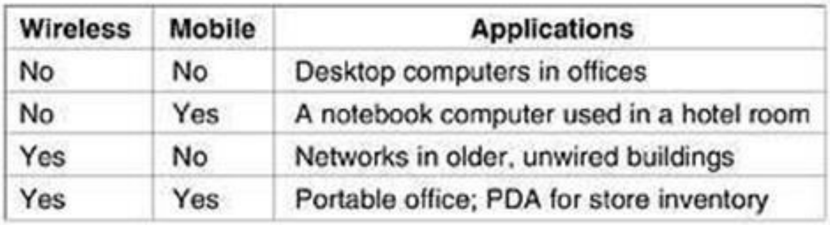
Tag	Full name	Example
B2C	Business-to-consumer	Ordering books on-line
B2B	Business-to-business	Car manufacturer ordering tires from supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products on line
P2P	Peer-to-peer	File sharing

Table 1.2 some forms of e-commerce

C) **Mobile Users:** Notebook Computer,Hotspots,Text Messaging,GPS

As wireless technology becomes more widespread, numerous other applications are likely to emerge. Wireless networks are of great value to fleets of trucks, taxis, delivery vehicles, and repairpersons for keeping in contact with home. Wireless networks are also important to the military.

Although wireless networking and mobile computing are often related, they are not identical, as Table 1.3 shows. Here we see a distinction between fixed wireless and mobile wireless. Even notebook computers are sometimes wired. For example, if a traveler plugs a notebook computer into the telephone jack in a hotel room, he has mobility without a wirelessnetwork.



Wireless	Mobile	Applications
No	No	Desktop computers in offices
No	Yes	A notebook computer used in a hotel room
Yes	No	Networks in older, unwired buildings
Yes	Yes	Portable office; PDA for store inventory

Table 1.3 Combinations of wireless networks and mobile computing

Another area in which wireless could save money is utility meter reading. If electricity, gas, water, and other meters in people's homes were to report usage over a wireless network, there would be no need to send out meter readers.

D) **Social Issues:**Phishing,network neutrality(Traffic treated as equal)

The widespread introduction of networking has introduced new social, ethical, and political problems. A popular feature of many networks is newsgroups or bulletin boards whereby people can exchange messages with like-minded individuals. As long as the subjects are restricted to technical topics or hobbies like gardening, not too many problems will arise.

The following are the issues in society due to the misbehave or misconduct of computer networks.

1. Network neutrality
2. Digital Millennium Copyright Act
3. Profiling users
4. Phishing

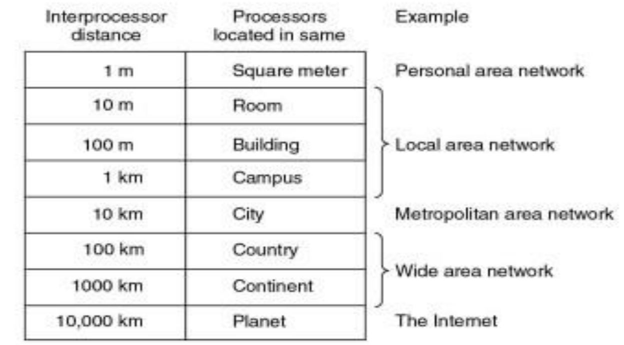
NETWORK HARDWARE

- There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important: transmission technology and scale.
- There are two transmission technologies: Broadcast links and Point-to-Point links.

- **Point-to-point** links connect individual pairs of machines. To go from the source to the destination on a network made up of point-to-point links, short messages, called **packets** in certain contexts, may have to first visit one or more intermediate machines.
- Often multiple routes of different lengths are possible, so finding good ones is important in point-to-point networks.
- Point-to-point transmission with exactly one sender and exactly one receiver is sometimes called **unicasting**. Example browsing a website.

- **Broadcast links**: single communication channel shared by all machines, example wireless network.

- An address field within each packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that machine processes the packet; if the packet is intended for some other machine, it is just ignored.
- Some broadcast systems also support transmission to a subset of the machines, which known as **multicasting**.

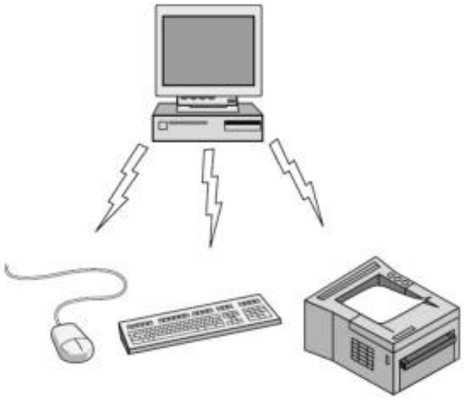


Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

Classification of interconnected processors by scale.

PANs (Personal Area Networks):

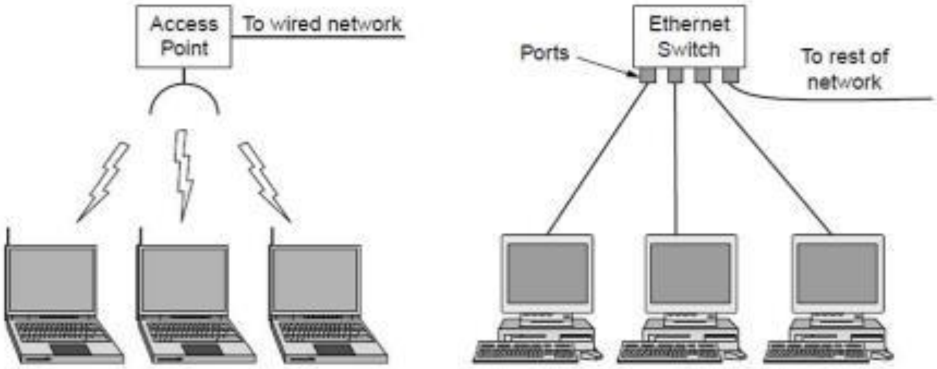
- Let devices communicate over the range of a person.
- For example a wireless network that connects a computer with its peripherals, without using wireless, this connection must be done with cables.
- So many new users have a hard time finding the right cables and plugging them into the right place.
- To help these users, some companies got together to design a short-range wireless network called Bluetooth to connect these components without wires.
- The idea is that if your devices have Bluetooth, then you need no cables. You just put them down, turn them on, and they work together. For many people, this ease of operation is a big plus.
- In the simplest form, Bluetooth networks use the **master-slave paradigm** of
- Fig. The system unit (the PC) is normally the master, talking to the mouse, keyboard, etc., as slaves. The master tells the slaves what addresses to use, when they can broadcast, how long they can transmit, what frequencies they can use, and so on.
- It connects a headset to a mobile phone without cords and it can allow your digital music player



Bluetooth PAN configuration

LAN (Local Area Network):

- It is a privately owned network that operates within and nearby a single building like a home, office or factory.
- When LANs are used by companies, they are called enterprise networks.
- Wireless LANs(IEEE 802.11) /Wireless Fidelity (WiFi): It is used in homes, older office buildings, cafeterias, and other places where it is too much trouble to install cables.
- In these systems, every computer has a radio modem and an antenna that it uses to communicate with other computers.



Wireless and wired LANs. (a) 802.11. (b) Switched Ethernet.

- An AP (Access Point), wireless router, or base station, relays packets between the wireless computers and also between them and the Internet.
- Wireless LAN operates at a speed of 11 to 100’s Mbps.
- Wired LANs use a range of different transmission technologies. Most of them use copper wires, but some use optical fiber. LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance.

- Wired LANs run at speeds of 100 Mbps to 1 Gbps, have low delay and few errors. Newer LANs can operate at up to 10 Gbps. It is just easier to send signals over a wire or through a fiber than through the air.
- The topology wired LANs is built from point-to-point links. IEEE 802.3, called **Ethernet**, is, by far, the most common type of wired LAN. Fig. (b) **Switched Ethernet**. Each computer speaks the Ethernet protocol and connects to a box called a **switch** with a point-to-point link. A switch has multiple **ports**, each of which can connect to one computer. The job of the switch is to relay packets between computers that are attached to it, using the address in each packet to determine which computer to send it to.
- Both wireless and wired broadcast networks can be divided into static and dynamic designs, depending on how the channel is allocated.

- Static allocation would be to divide time into discrete intervals and use a round-robin algorithm, allowing each machine to broadcast only when its time slot comes up.
- Static allocation wastes channel capacity when a machine has nothing to say during its allocated slot, so most systems attempt to allocate the channel dynamically (i.e., on demand).
- Dynamic allocation methods for a common channel are either centralized or decentralized.
- **Centralized channel allocation: T**here is a single entity, for example, the base station in cellular networks, which determines who goes next. It might do this by accepting multiple packets and prioritizing them according to some internal algorithm.
- **Decentralized channel allocation:** there is no central entity; each machine must decide for itself whether to transmit.

MAN (Metropolitan Area Network):

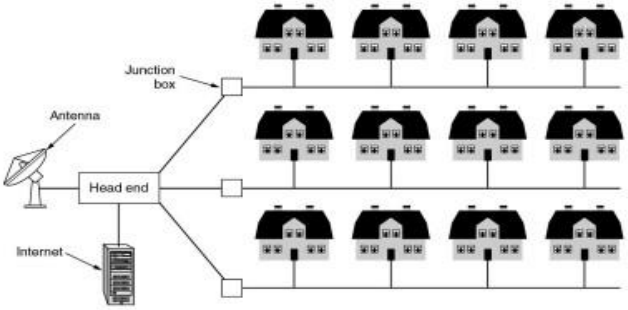
- It covers a city and best-known example is cable television networks available in many cities. These systems grew from earlier community antenna systems used in areas with poor over-the-air television reception. In those early systems, a large antenna was placed on top of a nearby hill and a signal was then piped to the subscribers' houses.

- At first, these were locally designed, ad hoc systems. The next step was television programming and even entire channels designed for cable only. Often these channels were highly specialized, such as all news, all sports, all cooking, all gardening, and so on.

- When the Internet began attracting a mass audience, the cable TV network operators began to realize that with some changes to the system, they could pro- vide two-way Internet service in unused parts of the spectrum.

- In figure, both television signals and Internet being fed into the centralized **cable headend** for subsequent distribution to people's homes.

- Recent developments in high- speed wireless Internet access have resulted in another MAN, which has been standardized as IEEE 802.16 and is popularly known as **WiMAX**



A metropolitan area network based on cable TV.

WAN (Wide Area Network):

- ❖ It Spans a large geographical area, often a country or continent.
- ❖ In most WANs, the subnet consists of two distinct components: transmission lines and switching elements.
- ❖ **Transmission lines** (copper wire, optical fiber, or even radio links) move bits between machines.
- ❖ Switching elements (switches), are specialized computers that connect two or more transmission lines.
- ❖ When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them.
- ❖ The routers will usually connect different kinds of networking technology. The networks inside the offices may be switched Ethernet, for example, while the long-distance transmission lines may be SONET links.

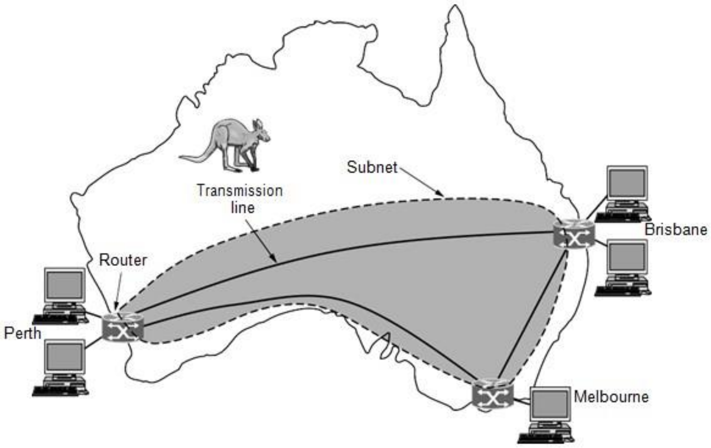


Figure 1-10. WAN that connects three branch offices in Australia.

- ❖ VPN (Virtual Private Network): it provides flexible reuse of a resource (Internet connectivity).
- ❖ It has disadvantage, which is a lack of control over the underlying resources and mileage may vary with internet speed.

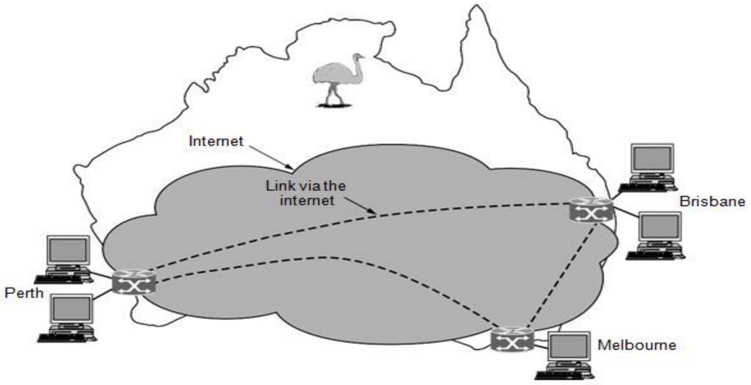


Figure 1-11. WAN using a virtual private network.

- The subnet operator is known as a **network service provider** and the offices are its customers. The subnet operator will connect to other customers too, as long as they can pay and it can provide service.
- A subnet operator is called an **ISP** (**Internet Service Provider**) and the subnet is an **ISP network**. Its customers who connect to the ISP receive Internet service.

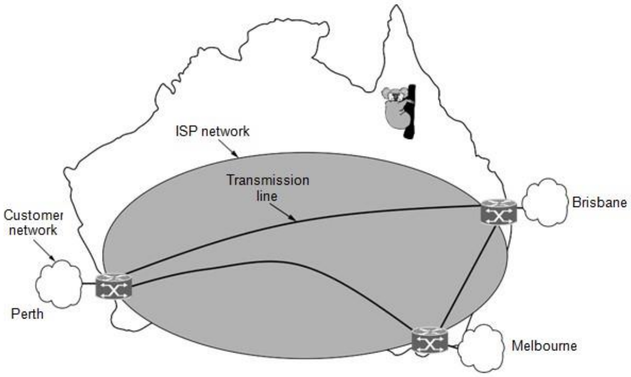


Figure 1-12. WAN using an ISP network.

- How the network makes the decision as to which path to use is called the **routing algorithm**.
- How each router makes the decision as to where to send a packet next is called the **forwarding algorithm**.
- Examples of WAN make heavy use of wireless technologies i.e. satellite systems.
- The cellular telephone network is another example of a WAN that uses wireless technology.
- The first generation was analog and for voice only. The second generation was digital and for voice only. The third generation is digital and is for both voice and data.
- Each cellular base station covers a distance much larger than a wireless LAN, with a range measured in kilometers rather than tens of meters.

Internetworks:

- A collection of interconnected networks is called an **internetwork** or **internet**.
- The Internet uses ISP networks to connect enterprise networks, home networks, and many other networks.
- There are two rules of thumb that are useful. First, if different organizations have paid to construct different parts of the network and each maintains its part, we have an internetwork rather than a single

network.

- ❖ Second, if the underlying technology is different in different parts (e.g., broadcast versus point-to-point and wired versus wireless), we probably have an internetwork.
- ❖ Gateways are distinguished by the layer at which they operate in the protocol hierarchy.

NETWORK SOFTWARE:

Protocol Hierarchies:

❖ To reduce their design complexity, most networks are organized as a stack of **layers** or **levels**, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network.

❖ The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

❖ When layer *n* on one machine carries on a conversation with layer *n* on another machine, the rules and conventions used in this conversation are collectively known as the layer *n* protocol. Basically, a **protocol** is an agreement between the communicating parties on how communication is to proceed.

❖ **A five-layer network is illustrated:**

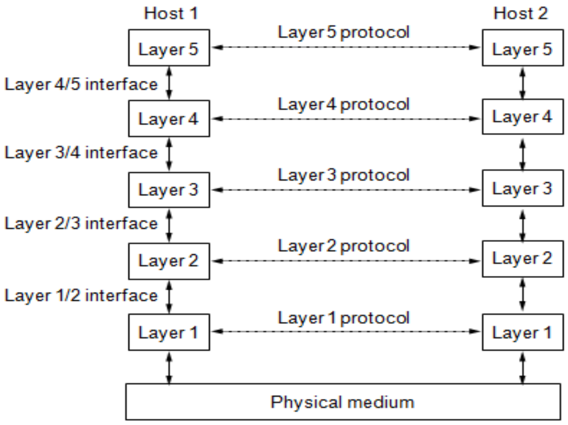


Figure 1-13. Layers, protocols, and interfaces.

- ❖ In reality, no data are directly transferred from layer *n* on one machine to layer *n* on another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached.
- ❖ Below layer 1 is the **physical medium** through which actual communication occurs. Virtual communication is shown by dotted lines and physical communication by solid lines.
- ❖ **Interface:** defines which primitive operations and services the lower layer makes available to the

upper one.

- Clear- cut interfaces also make it simpler to replace one layer with a completely different protocol or implementation.
- A set of layers and protocols is called network **architecture.** The specification of architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol.
- A list of the protocols used by a certain system, one protocol per layer, is called a **protocol stack**.

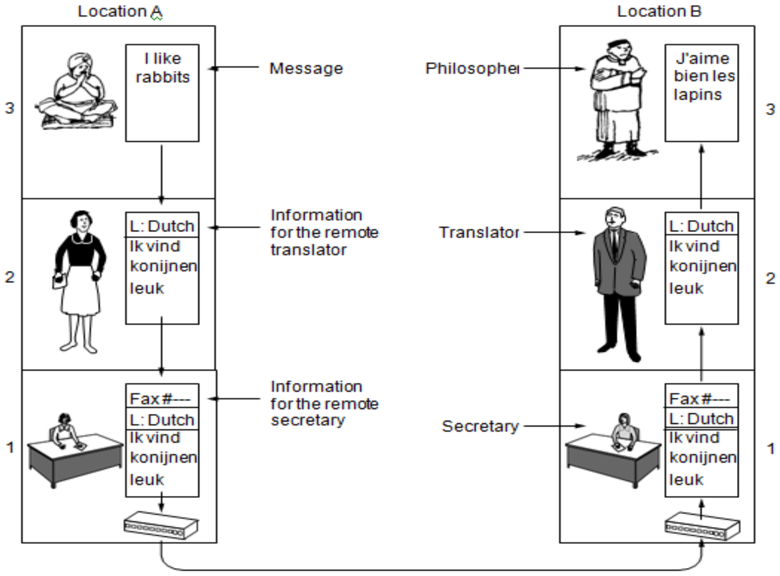


Figure 1-14. The philosopher-translator-secretary architecture.

- In this example, M is split into two parts, M_1 and M_2, that will be transmitted separately. Layer 3 decides which of the outgoing lines to use and passes the packets to layer 2. Layer 2 adds to each piece not only a header but also a trailer, and gives the resulting unit to layer 1 for physical transmission.
- At the receiving machine the message moves upward, from layer to layer, with headers being stripped off as it progresses. None of the headers for layers below *n* are passed up to layer *n.*

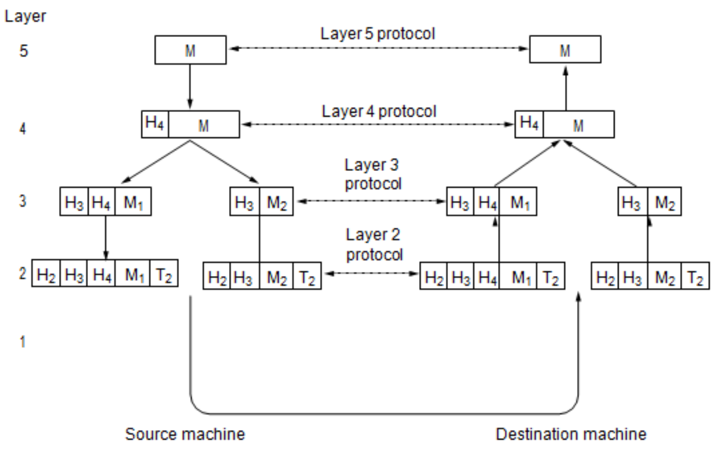


Figure 1-15. Example information flow supporting virtual communication in layer 5.

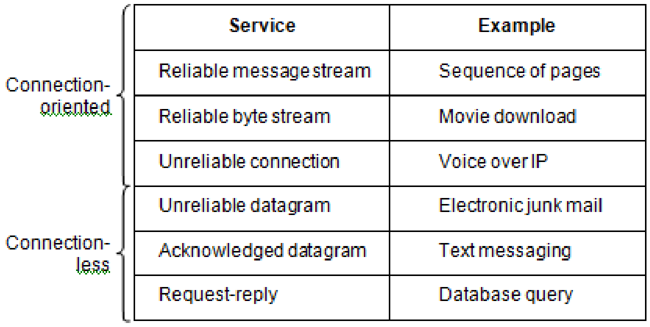
Design Issues for the Layers:

Some of the key design issues that occur in computer networks are present in several layers. The following are briefly mention some of the more important ones.

- ***Identifying senders and receivers*** - some form of addressing is needed in order to specify a specific source and destination.
- ***Rules for data transfer*** - The protocol must also determine the direction of data flow, how many logical channels the connection corresponds to and what their priorities are. Many networks provide at least two logical channels per connection, one for normal data and one for urgent data.
- ***Error control*** – when circuits are not perfect, both ends of the connection must agree on which error-detecting and error-correcting codes is being used.
- ***Sequencing*** - protocol must make explicit provision for the receiver to allow the pieces to be reassembled properly.
- ***Flow Control*** - how to keep a fast sender from swamping a slow receiver with data. This is done by feedback-based (receiver to sender) or agreed-on transmission rate.
- ***Segmentation and reassembly*** - several levels are the inability of all processes to accept arbitrarily long messages. It leads to mechanisms for disassembling, transmitting, and then reassembling messages.
- ***Multiplexing and demultiplexing*** – to share the communication medium by severalusers.
- ***Routing*** - When there are multiple paths between source and destination, a route must be chosen.

Connection-Oriented Versus Connectionless Service:

- Connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection.
- Connection acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end. In most cases the order is preserved so that the bits arrive in the order they were sent.
- In some cases when a connection is established, the sender, receiver, and subnet conduct a **negotiation** about the parameters to be used, such as max message size, QoS required etc. Typically, one side makes a proposal and the other side can accept it, reject it, or make a counter- proposal.
- **Connectionless** service is modeled after the postal system. Each message (letter) carries the full destination address, and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages.
- There are different names for messages in different contexts; a **packet** is a message at the network layer. When the intermediate nodes receive a message in full before sending it on to the next node, this is called **store-and-forward switching**.
- The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called **cut-through switching**.
- Unreliable (meaning not acknowledged) connectionless service is often called **datagram** service, in analogy with telegram service, which also does not return an acknowledgement to the sender.



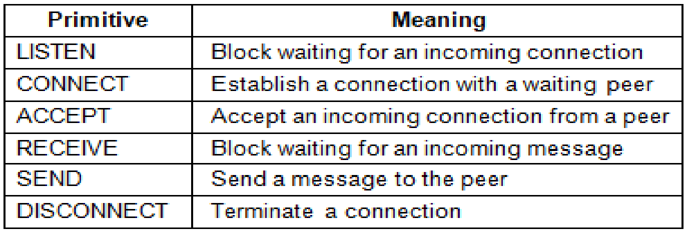
	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Movie download
	Unreliable connection	Voice over IP
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Text messaging
	Request-reply	Database query

Figure 1-16. Six different types of service.

Service Primitives (Operations):

- These primitives tell the service to perform some action or report on an action taken by peer entity.

The primitives for connection-oriented service are different from those of connectionless service.



Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Figure 1-17. Six service primitives that provide a simple connection-oriented service.

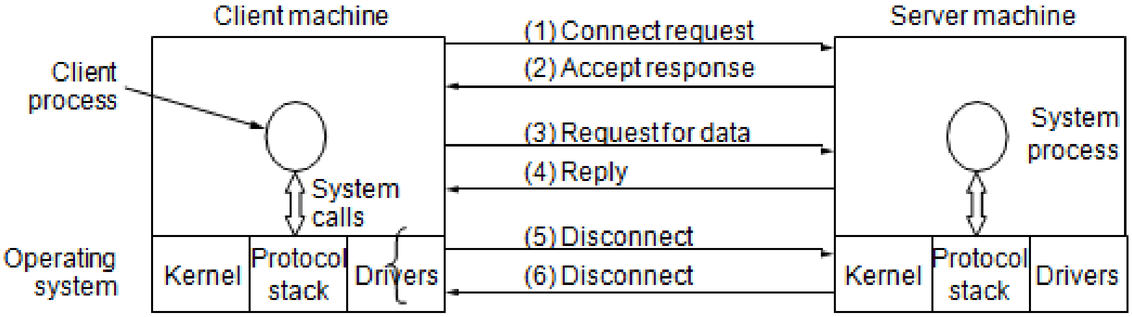


Figure 1-18. A simple client-server interaction using acknowledged datagram's.

- ❖ First, the server executes LISTEN to indicate that it is prepared to accept in coming connections. A common way to implement LISTEN is to make it a blocking system call. After executing the primitive, the server process is blocked untila request for connection appears.
- ❖ Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server's address. Client is suspended until there is a response.

- ❖ When the packet arrives at the server, the operating system sees that the packet is requesting a connection. It checks to see if there is a listener, and if so it unblocks the listener. The server process can then establish the connection with the ACCEPT call.
- ❖ The next step is for the server to execute RECEIVE to prepare to accept the first request. Normally, the server does this immediately upon being released from the LISTEN, before the acknowledgement can get back to the client. The RECEIVE call blocks the server.
- ❖ Then the client executes SEND to transmit its request followed by the execution of RECEIVE to get the reply.
- ❖ When the client is done, it executes DISCONNECT to terminate the connection.

The Relationship of Services to Protocols

- ❖ **A *service*** is a set of primitives (operations) that a layer provides to the layer above it. I t defines what operations the layer is prepared to perform on behalf of its users. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.
- ❖ A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are

implemented.

- **A *protocol*,** in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer.
- In contrast, a protocol relates to the *implementation* of the ser- vice and as such is not visible to the user of the service.

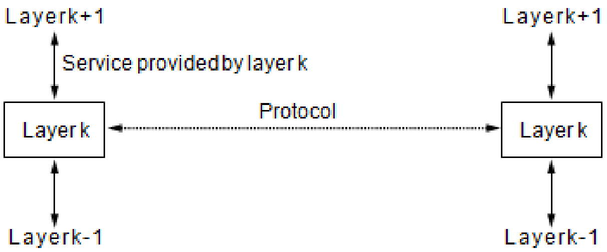


Figure 1-19. The relationship between a service and a protocol.

REFERENCE MODELS

The OSI Reference Model

- This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983).
- It was revised in 1995 (Day, 1995). The model is called the ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems.
- The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.
5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

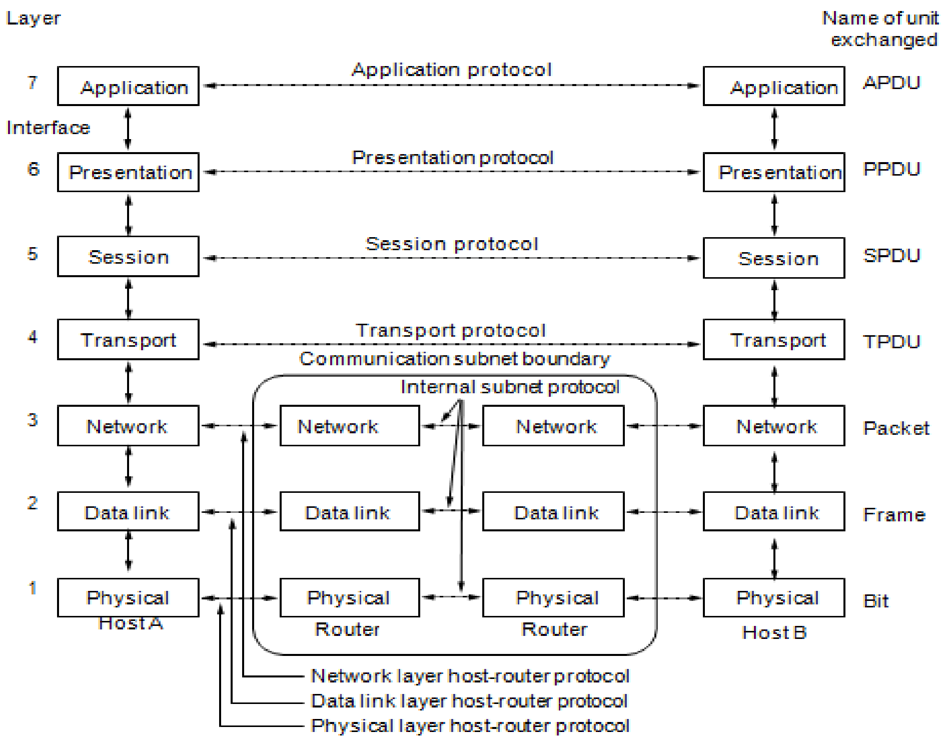


Figure 1-20. The OSI reference model.

The Physical Layer

- This **layer** is concerned with transmitting raw bits sequence of 0's and 1's over a communication channel.
- The design issues are deal with mechanical, electrical, and timing interfaces, as well as the physical transmission medium, which lies below the physical layer.

The Data Link Layer

- This layer transforms a raw transmission facility into a line that appears free of undetected transmission errors.
- It accomplishes this task by having the sender break up the input data into **data frames** (typically a few hundred or a few thousand bytes) and transmits the frames sequentially.
- If the service is reliable, the receiver confirms correct receipt of each frame by send- ing back an **acknowledgement frame**.
- Another issue in the data link layer is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanisms are used.
- **Medium access control** sub layer deals with how to control access to the shared channel.

The Network Layer

- Controls the operation of the subnet.
- A key design issue is determining how packets are routed from source to destination.
- Routes can be based on static tables that are ''wired into'' the network and rarely changed, or more often they can be updated automatically to avoid failed components.

- If too many packets are present in the subnet at the same time, they will get in one another’s way, forming bottlenecks. Handling congestion is also a responsibility of the network layer.
- Heterogeneous networks to be interconnected.

The Transport Layer

- It accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end.
- It also determines what type of service to provide to the session layer, ultimately, to the users of the network.
- The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent.
- It also provides the service of transporting of isolated messages with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations.
- The transport layer is a true end-to-end layer; it carries data all the way from the source to the destination.

The Session Layer

- It allows users on different machines to establish sessions between them.
- Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management(preventing two parties from attempting the same critical operation simultaneously), and synchronization (check pointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery).

The Presentation Layer

- **This layer** is concerned with the syntax and semantics of the information transmitted.
- In order to make it possible for computers with different internal data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used ‘‘on the wire.’’
- The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records) to be defined and exchanged.

The Application Layer

- **It c**ontains a variety of protocols that are commonly needed by users.
- One widely used application protocol is **HTTP** (**HyperText Transfer Protocol**), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server hosting the page using HTTP. The server then sends the page back.
- Other application protocols are used for file transfer, electronic mail, and network news.

The TCP/IP Reference Model

- This reference Model is a four-layered suite of communication protocols, developed by the DoD (Department of Defence) in the 1960s. It is named after the two main protocols that are used in the model, namely, TCP and IP.

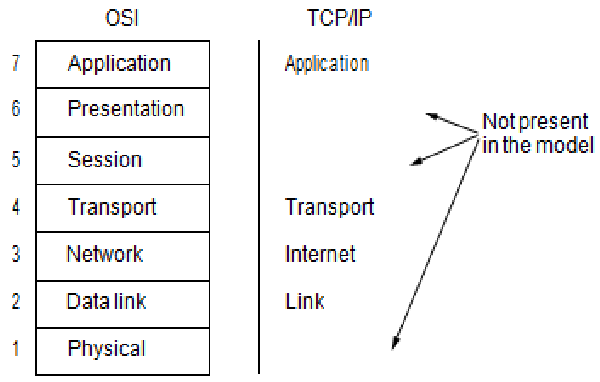


Figure 1-21. The TCP/IP reference model.

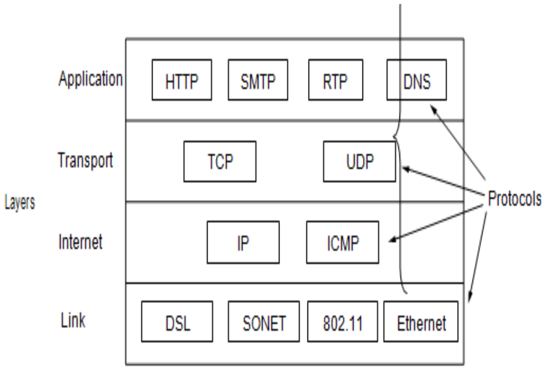


Figure 1-22. The TCP/IP model with some protocols we will study.

The Link Layer

- It describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer.
- It is not really a layer at all, in the normal sense of the term, but rather an interface between hosts and transmission links.

The Internet Layer

- Its job is to permit hosts to inject packets into any network and have they travel independently to the destination.
- Packet may arrive in a completely different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired.
- This layer defines an official packet format and protocol called **IP** (**Internet Protocol**), plus a companion protocol called **ICMP** (**Internet Control Message Protocol**) that helps it function.
- The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly a major issue here, as is congestion (though IP has not proven effective at avoiding congestion).

The Transport Layer

- It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer.
- Two end-to-end transport protocols have been defined here TCP,UDP.
- **TCP** (**Transmission Control Protocol**), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet.
- It segments the incoming byte stream into discrete messages and passes each one on to the internet

layer. At the destination, the receiving TCP process reassembles the received messages into the output stream.

- ❖ TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.
- ❖ UDP (User Datagram Protocol), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own.
- ❖ It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video.

The Application Layer

- ❖ It contains all the higher-level protocols. The file transfer (FTP), and electronic mail (SMTP). Domain Name System (DNS), for mapping host names onto their net- work addresses, HTTP, the protocol for fetching pages on the World Wide Web, RTP, the protocol for delivering real-time media such as voice or movies.

A Comparison of the OSI and TCP/IP Reference Models

- ❖ Three concepts are central to the OSI model:

1. **Services:** It tells layer's semantics, what the layer does, not how entities above it access it or how the layer works.
2. **Interfaces:** It specifies what the parameters are and what results to expect.
3. **Protocols:** provides the offered services.

- ❖ The TCP/IP model did not originally clearly distinguish between services, interfaces, and protocols, although people have tried to retrofit it after the fact to make it more OSI-like.
- ❖ The protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes.
- ❖ As a consequence, the proto- cols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes.
- ❖ With TCP/IP the reverse was true: the protocols came first, and the model was really just a description of the existing protocols. There was no problem with the protocols fitting the model.
- ❖ The OSI model has seven layers and the TCP/IP model has four. Both have (inter)network, transport, and application layers, but the other layers are different.
- ❖ The OSI model supports both connectionless and connection- oriented communication in the network layer, but only connection-oriented communication in the transport layer, where it counts
- ❖ The TCP/IP model supports only one mode in the network layer (connectionless) but both in the transport layer, giving the users a choice.

PHYSICAL LAYER: GUIDED TRANSMISSION MEDIA

1. Magnetic Media :

- One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs), physically transport the tape or disks to the destination machine, and read them back in again.
- Although this method is not as sophisticated as using a geosynchronous communication satellite, it is often more cost effective, especially for applications in which high bandwidth or cost per bit transported is the key factor.
- An industry-standard Ultrium tape can hold 800 gigabytes. A box 60 × 60 × 60 cm can hold about 1000 of these tapes, for a total capacity of 800 terabytes, or 6400 terabits (6.4 petabits).
- A box of tapes can be delivered anywhere in the United States in 24 hours by Federal Express and other companies. The effective bandwidth of this transmission is 6400 terabits/86,400 sec, or a bit over 70 Gbps.
- If the destination is only an hour away by road, the bandwidth is increased to over 1700 Gbps. No computer net- work can even approach this. Of course, networks are getting faster, but tape den- sities are increasing, too.
- The cost of an Ultrium tape is around $40 when bought in bulk. A tape can be reused at least 10 times , So the tape cost is maybe $4000 per box per usage.

2. Twisted Pairs

- A twisted pair consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together in a helical form, just like a DNA molecule.
- Twisting is done because two parallel wires constitute a fine antenna. When the wires are twisted, the waves from different twists cancel out, so the wire radiates less effectively.
- A signal is usually carried as the difference in voltage between the two wires in the pair. This provides better immunity to external noise because the noise tends to affect both wires the same, leaving the differential unchanged.
- The most common application of the twisted pair is the telephone system.
- Twisted pairs can run several kilometers without amplification, but for longer distances the signal becomes too attenuated and repeaters are needed.
- The bandwidth depends on the thickness of the wire and the distance traveled, but several megabits/sec can be achieved for a few kilometers in many cases.

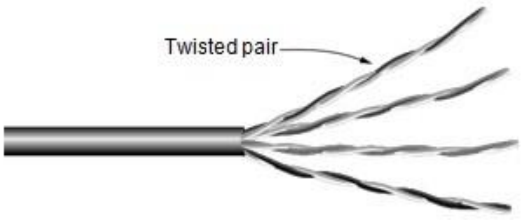


Figure 2-3. Category 5 UTP cable with four twisted pairs.

- Twisted-pair cabling comes in several varieties. A ‘‘Cat 5”category 5 twisted pair consists of two insulated wires gently twisted together. Four such pairs are typically grouped in a plastic sheath to protect the wires and keep them together.
- Links that can be used in both directions at the same time, like a two-lane road, are called **full-duplex** links.
- links that can be used in either direction, but only one way at a time, like a single-track railroad line are called **half-duplex** links.

- Links that allow traffic in only one direction, like a one-way street. They are called **simplex** links.
- Cat 5 replaced earlier Category 3 cables, but has more twists per meter. More twists result in less crosstalk and a better-quality signal over longer distances, making the cables more suitable for high-speed computer communication, especially 100-Mbps and 1-Gbps Ethernet LANs.

- Category 6 or even Category 7 has more stringent specifications to handle signals with greater band-widths.

3. Coaxial Cable

- It has better shielding and greater bandwidth than unshielded twisted pairs, so it can span longer distances at higher speed.
- There are two kinds, one kind, 50-ohm cable, is commonly used when it is intended for digital transmission from the start. The other kind, 75-ohm cable, is commonly used for analog transmission and cable television.
- A coaxial cable consists of a stiff copper wire as the core, surrounded by an insulating material. The insulator is encased by a cylindrical conductor, often as a closely woven braided mesh.
- The outer conductor is covered in a protective plastic sheath.

- The construction and shielding of the coaxial cable give it a good combination of high bandwidth and excellent noise immunity. The bandwidth possible depends on the cable quality and length.
- Coaxial cables used to be widely used within the telephone system for long-distance lines but have now largely been replaced by fiber optics on long- haul routes. Coax is still widely used for cable television and metropolitan area networks, however.

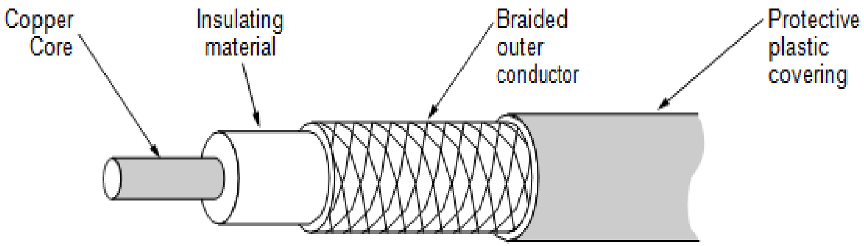


Figure 2-4. A coaxial cable.

4. Power Lines

- ❖ Power lines deliver electrical power to houses, and electrical wiring within houses distributes the power to electrical outlets. Its been used by electricity companies for low-rate communication such as re- mote metering for many years, as well in the home to control devices.
- ❖ Simply plug a TV and a receiver into the wall, which you must do anyway because they need power, and they can send and receive movies over the electrical wiring.
- ❖ The difficulty with using household electrical wiring for a network is that it was designed to distribute power signals.
- ❖ Electrical signals are sent at 50–60 Hz and the wiring attenuates the much higher frequency (MHz) signals needed for high-rate data communication.
- ❖ Transient currents when appliances switch on and off create electrical noise over a wide range of frequencies.
- ❖ Despite these difficulties, it is practical to send at least 100 Mbps over typical household electrical wiring by using communication schemes that resist impaired frequencies and bursts of errors.

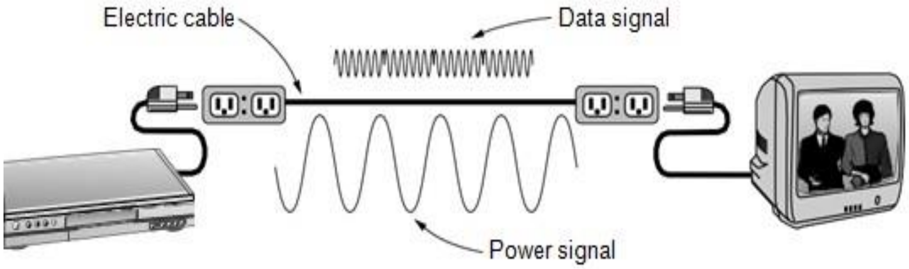


Figure 2-5. A network that uses household electrical wiring.

5. Fiber Optics

- ❖ In contrast, the achievable bandwidth with fiber technology is in excess of 50,000 Gbps (50 Tbps) and we are nowhere near reaching these limits.
- ❖ The current practical limit of around 100 Gbps is due to our inability to convert between electrical and opti- cal signals any faster.
- ❖ Fiber optics are used for long-haul transmission in network backbones, high-speed LANs and high-speed Internet access such as **FttH** (**Fiber to the Home**).
- ❖ An optical transmission system has three key components: the light source, the transmission medium,

and the detector.

- Conventionally, a pulse of light indicates a 1 bit and the absence of light indicates a 0 bit. The transmission medium is an ultra-thin fiber of glass. The detector generates an electrical pulse when light falls on it.

Transmission of Light through Fiber:

Optical fibers are made of glass, which, in turn, is made from sand, an in expensive raw material available in unlimited amounts.

Fiber Cables:

- Fiber optic cables are similar to coax, except without the braid. At the center is the glass core through which the light propagates. In multimode fibers, the core is typically 50 microns in diameter, about the thickness of a human hair. In single-mode fibers, the coreis 8 to 10 microns.

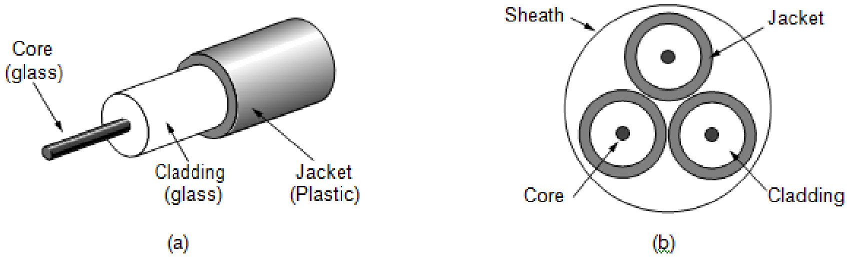


Figure 2-8. (a) Side view of a single fiber. (b) End view of a sheath with three fibers.

- The core is surrounded by a glass cladding with a lower index of refraction than the core, to keep all the light in the core. Next comes a thin plastic jacket to protect the cladding. Fibers are typically grouped in bundles, protected by an outer sheath.

WIRELESS TRANSMISSION

The Electromagnetic Spectrum

- When electrons move, they create electromagnetic waves that can propagate through space. These waves were predicted by the British physicist James Clerk Maxwell in 1865 and first observed by the German physicist Heinrich Hertz in 1887.
- The number of oscillations per second of a wave is called its **frequency**, *f*, and is measured in **Hz.** The distance between two consecutive maxima (or minima) is called the **wavelength** λ (lambda).
- When an antenna of the appropriate size is attached to an electrical circuit, the electromagnetic waves can be broadcast efficiently and received by a receiver some distance away.
- In a vacuum, all electromagnetic waves travel at the same speed, no matter what their frequency as called **speed of light**, *c*, is approximately 3×10^8 m/sec, or about 1 foot (30 cm) per nanosecond.
- The fundamental relation between *f*, λ, and *c* (in a vacuum) is $\lambda f = c$

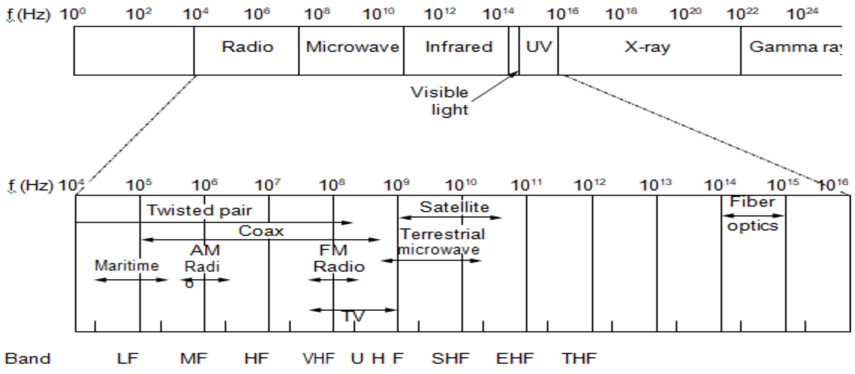

f (Hz) 10^{0} 10^{2} 10^{4} 10^{6} 10^{8} 10^{10} 10^{12} 10^{14} 10^{16} 10^{18} 10^{20} 10^{22} 10^{24}
Radio
Microwave
Infrared
UV
X-ray
Gamma ray
Visible light
f (Hz) 10^{4} 10^{5} 10^{6} 10^{7} 10^{8} 10^{9} 10^{10} 10^{11} 10^{12} 10^{13} 10^{14} 10^{15} 10^{16}
Twisted pair
Coax
Satellite
Terrestrial microwave
Fiber optics
Maritime
AM Radio
FM Radio
TV
Band LF MF HF VHF UHF SHF EHF THF

Figure 2-10. The electromagnetic spectrum and its uses for communication.

- In **frequency hopping spread spectrum**, the transmitter hops from frequency to frequency hundreds of times per second. It is popular for military communication because it makes transmissions hard to detect and next to impossible to jam.
- **Direct sequence spread spectrum** uses a code sequence to spread the data signal over a wider frequency band. It is widely used commercially as a spectrally efficient way to let multiple signals share the same frequency band.

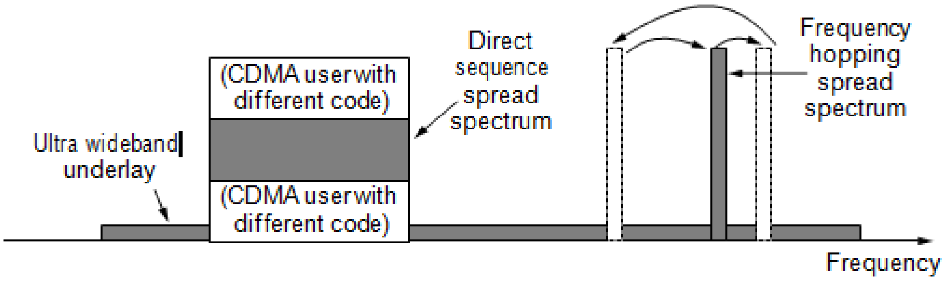

(CDMA user with different code)
(CDMA user with different code)
Direct sequence spread spectrum
Frequency hopping spread spectrum
Ultra wideband underlay
Frequency

Figure 2-11. Spread spectrum and ultra-wideband (UWB) communication.

Radio Transmission

- Radio frequency (RF) waves are easy to generate, can travel long distances, and can penetrate buildings easily, so they are widely used for communication, both indoors and outdoors.
- Radio waves also are omnidirectional, meaning that they travel in all directions from the source, so the transmitter and receiver do not have to be carefully aligned physically.
- The properties of radio waves are frequency dependent. At low frequencies, radio waves pass through obstacles well, but the power falls off sharply with distance from the source—at least as fast as $1/r^{2}$ in air—as the signal energy is spread more thinly over a larger surface. This attenuation is called **path loss**.

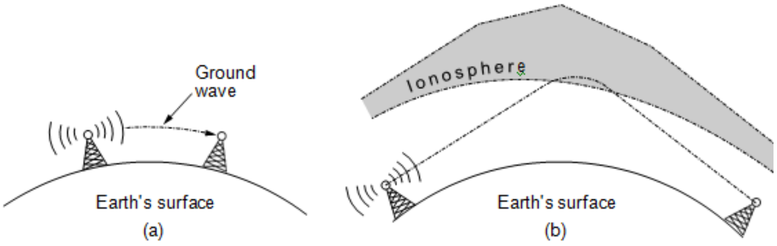


Figure 2-12. (a) In the VLF, LF, and MF bands, radio waves follow the curvature of the earth. (b) In the HF band, they bounce off the ionosphere.

Microwave Transmission

- Before fiber optics, for decades these microwaves formed the heart of the long-distance tele- phone transmission system.
- In fact, MCI, one of AT&T's first competitors after it was deregulated, built its entire system with microwave communications passing between towers tens of kilometers apart. Even the company's name reflected this (MCI stood for Microwave Communications, Inc.).
- Microwaves travel in a straight line, so if the towers are too far apart, the earth will get in the way.
- Unlike radio waves at lower frequencies, microwaves do not pass through buildings well. In addition, even though the beam may be well focused at the transmitter, there is still some divergence in space.
- The demand for more and more spectrum drives operators to yet higher frequencies. Bands up to 10 GHz are now in routine use, but at about 4 GHz a new problem sets in: absorption by water.
- Microwave communication is so widely used for long-distance telephone communication, mobile phones, television distribution, and other pur- poses that a severe shortage of spectrum has developed.
- Microwave is also relatively inexpensive. Putting up two simple towers and putting antennas on each one may be cheaper than burying 50 km of fiber through a congested urban area or up over a mountain, and it may also be cheaper than leasing the telephone company's fiber,

Infrared Transmission

- Unguided infrared waves are widely used for short-range communication. The remote controls used for televisions, VCRs, and stereos all use infrared communication.
- On the other hand, the fact that infrared waves do not pass through solid walls well is also a plus. It means that an infrared system in one room of a building will not interfere with a similar system in adjacent rooms or buildings: you cannot control your neighbor's television with your remote control.
- Furthermore, security of infrared systems against eavesdropping is better than that of radio systems precisely for this reason.

- Infrared communication has a limited use on the desktop, for ex- ample, to connect notebook computers and printers with the **IrDA** (**Infrared Data Association**) standard, but it is not a major player in the communication game.

Light Transmission

- Unguided optical signaling or free-space optics has been in use for centuries.
- Optical signaling using lasers is inherently unidirectional, so each end needs its own laser and its own Photodetector. This scheme offers very high bandwidth at very low cost and is relatively secure because it is difficult to tap a narrow laser beam.
- The laser’s strength, a very narrow beam, is also its weakness here. Aiming a laser beam 1 mm wide at a target the size of a pin head 500 meters away requires the marksmanship of a latter-day Annie Oakley.
- To add to the difficulty, wind and temperature changes can distort the beam and laser beams also cannot penetrate rain or thick fog, although they normally work well on sunny days.
- Unguided optical communication may seem like an exotic networking technology today, but it might soon become much more prevalent.
- Communicating with visible light in this way is inherently safe and creates a low-speed network in the immediate vicinity of the display.

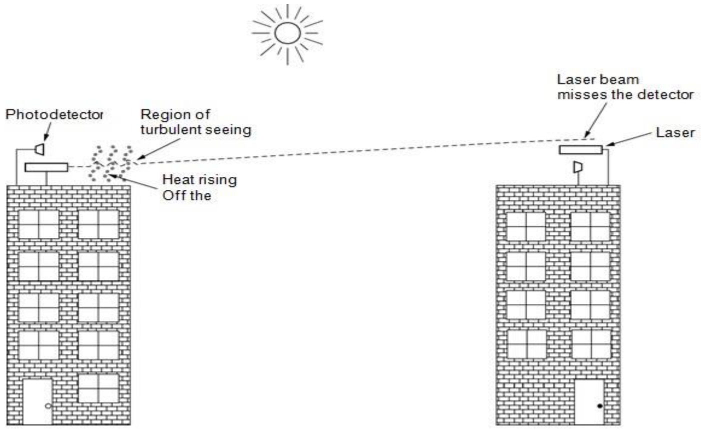


Figure 2-14. Convection currents can interfere with laser communication systems. A bidirectional system with two lasers is pictured here.

MODULE – 2

Topics:

- **The Data link layer:**
  - **Design issues of DLL**
  - **Error detection and correction**
  - **Elementary data link protocols**
  - **Sliding window protocols.**
  - **The medium access control sublayer:**
    - **The channel allocation problem**
    - **Multiple access protocols.**

DATA LINK LAYER DESIGN ISSUES

- The data link layer uses the services of the physical layer to send and receivebits over communication channels. Functions of data link layer include:
  - Providing a well-defined service interface to the network layer.
  - Dealing with transmission errors.
  - Regulating the flow of data so that slow receivers are not swampedby fast senders.
- To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into **frames** for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig. 3-1. Frame management forms the heart of what the data link layer does.

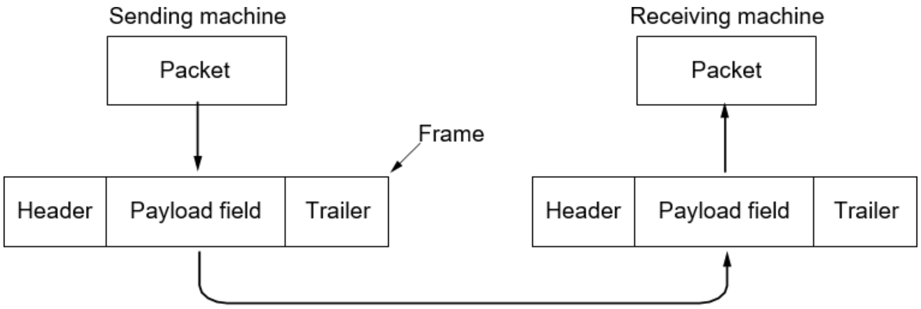


Figure 3-1. Relationship between packets and frames.

The following are the data link layer design issues

1. Services Provided to the Network Layer

The network layer wants to be able to send packets to its neighbors without worrying about the details of getting it there in one piece.

2. Framing

Group the physical layer bit stream into units called frames. Frames are nothing more than "packets" or "messages". By convention, we use the term "frames" when discussing DLL.

3. Error Control

Sender checksums the frame and transmits checksum together with data. Receiver re-computes the checksum and compares it with the received value.

4. Flow Control

Prevent a fast sender from overwhelming a slower receiver

Services Provided to the Network Layer

- ➢ The function of the data link layer is to provide services to the network layer. The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine.

- ➢ On the source machine is an entity(a process), in the network layer that hands some bits to the data link layer for transmission to the destination.

- ➢ The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there, as shown in Fig. 3-2(a). The actual transmission follows the path of Fig. 3-2(b)

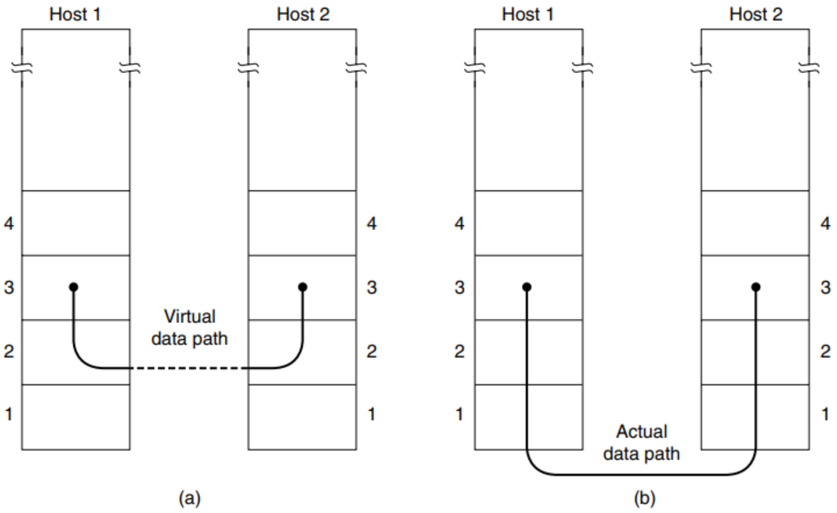


Figure 3-2. (a) Virtual communication. (b) Actual communication.

- The data link layer can be designed to offer various services:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

- Unacknowledged connectionless service consists of having the source ma- chine send independent frames to the destination machine without having the destination machine acknowledge them. Ethernet is a good example of a data link layer that provides this class of service. No logical connection is established beforehand or released afterward. If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer. This class of service is appropriate when the error rate is very low, so recovery is left to higher layers. It is also appropriate for real-time traffic, such as voice, in which late data are worse than bad data.

- The next step up in terms of reliability is acknowledged connectionless service. When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. In this way, the sender knows whether a frame has arrived correctly or been lost. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems. Eg:
  802.11 (WiFi)

- The most sophisticated service the data link layer can provide to the network layer is connection-oriented service. With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

- When connection-oriented service is used, transfers go through three distinct phases.

- First, connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not.
- Second, one or more frames are actually transmitted.
- Third, connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

FRAMING

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. If the channel is noisy, as it is for most wireless and some wired links, the physical layer will add some redundancy to its signals to reduce the bit error rate to a tolerable level. However, the bit stream received by the data link layer is not guaranteed to be error free. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the data link layer to detect and, if necessary, correct errors.

The data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted.

When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

DLL translates the physical layer's raw bit stream into discrete units (messages) called **frames.**

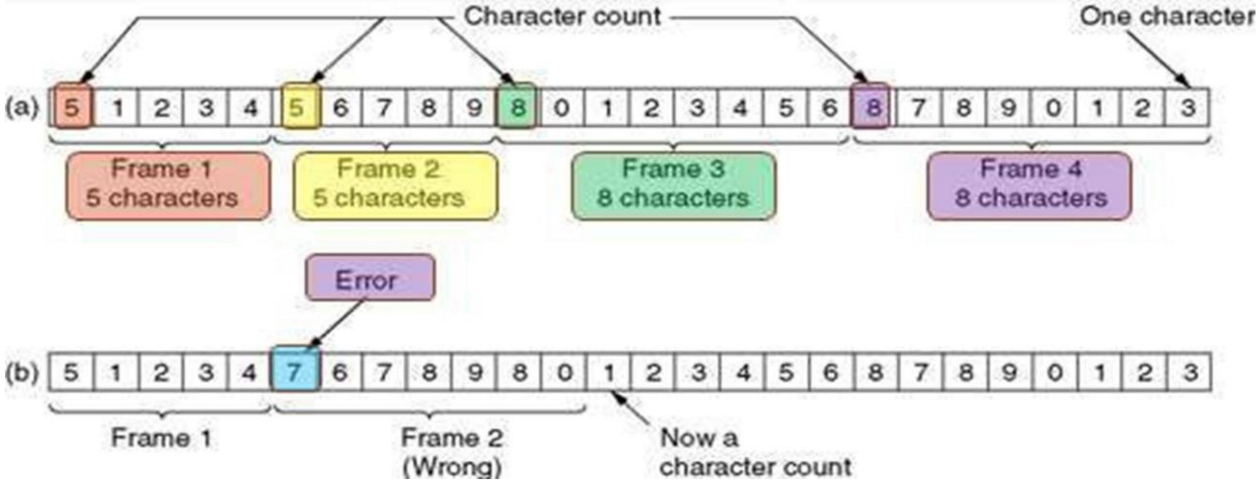
A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at **four methods:**

1. Byte count.
2. Flag bytes with byte stuffing.
3. Flag bits with bit stuffing.
4. Physical layer coding violations.

1. Byte count (Character Count) :

This framing method uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is. This technique is shown in Fig.(a) For four small example frames of sizes 5, 5, 8, and 8 bytes, respectively.

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the byte count of 5 in the second frame of Fig.(b) becomes a 7 due to a single bit flip, the destination will get out of synchronization. It will then be unable to locate the correct start of the next frame.

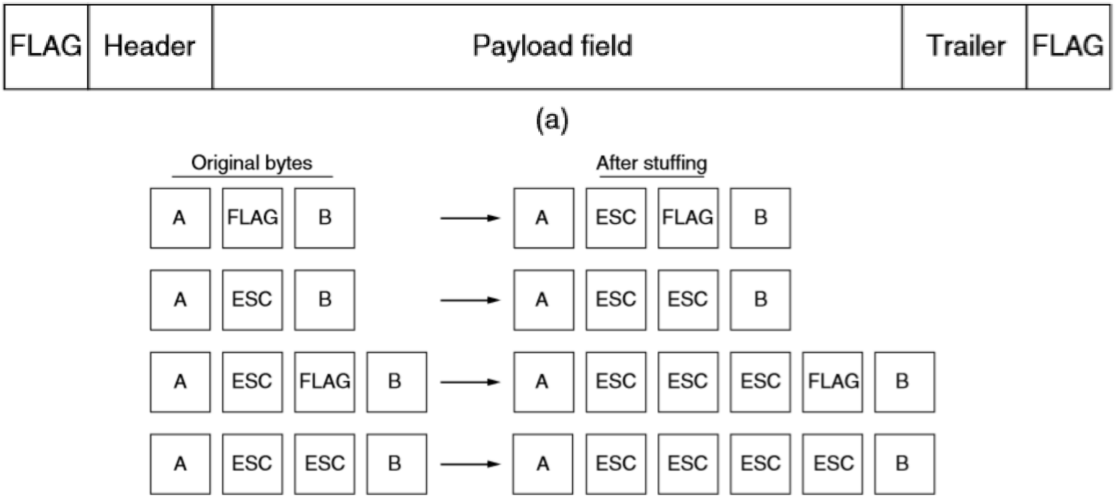


2. Flag bytes with byte stuffing:

This framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. Often the same byte, called a **flag byte**, is used as both the starting and ending delimiter. This byte is shown in Fig.(a) as **FLAG.** Two consecutive flag bytes indicate the end of one frame and the start of the next. Thus, if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame.

However, there is a still a problem we have to solve. It may happen that the flag byte occurs in the data, especially when binary data such as photographs or songs are being transmitted. This situation would interfere with the framing. One way to solve this problem is to have the sender’s data link layer insert a special escape byte (ESC) just before each ‘‘accidental’’ flag byte in the data.

The data link layer on the receiving end removes the escape bytes before giving the data to the network layer. This technique is called **byte stuffing.**

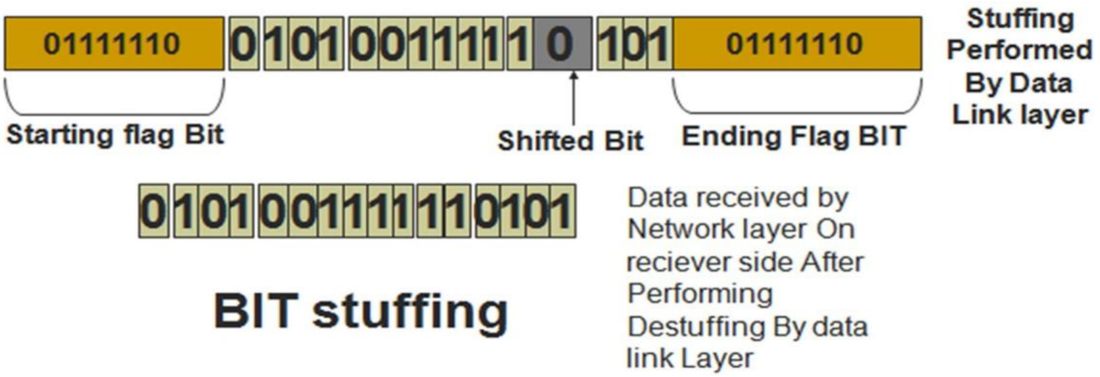


Four examples of byte sequences before and after byte stuffing.

3. Flag bits with bit stuffing:

Framing can be also be done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size. It was developed for the once very popular **HDLC (Highlevel Data Link Control)** protocol. Each frame begins and ends with a special bit pattern, 01111110 or 0x7E in hexadecimal. This pattern is a flag byte. Whenever the sender’s data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This **bit stuffing** is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data. It also ensures a minimum density of transitions that help the physical layer maintain synchronization. USB (Universal Serial Bus) uses bit stuffing for this reason.

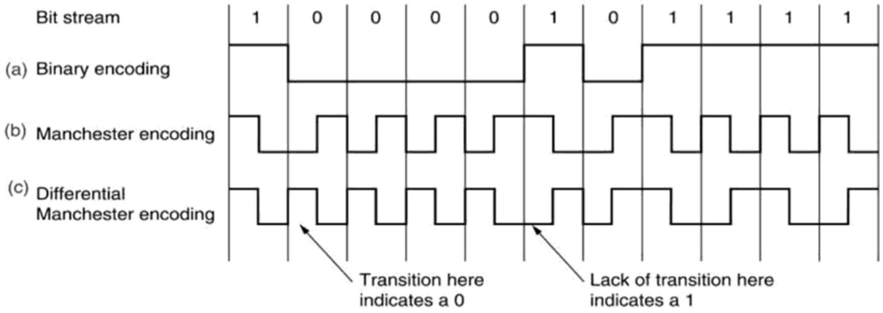
When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver’s memory as 01111110. Figure gives an example of bit stuffing.


01111110
0101001111101 0 101
01111110
Stuffing Performed By Data Link layer
Starting flag Bit
Shifted Bit
Ending Flag BIT
0101001111110101
Data received by Network layer On reciever side After Performing Destuffing By data link Layer
BIT stuffing

4. Physical layer coding violations:

- This Framing Method is used only in those networks in which Encoding on the Physical Medium contains some redundancy.
- Some LANs encode each bit of data by using two Physical Bits i.e. Manchester coding is used. Here, Bit 1 is encoded into high- low (10) pair and Bit 0 is encoded into low-high (01) pair.

- The scheme means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries. The combinations high-high and low-low are not used for data but are used for delimiting frames in some protocols.


Bit stream
1
0
0
0
0
1
0
1
1
1
1
(a) Binary encoding
(b) Manchester encoding
(c) Differential Manchester encoding
Transition here indicates a 0
Lack of transition here indicates a 1

ERROR CONTROL:

Error control is concerned with ensuring that all frames are eventually delivered (possibly in order) to a destination. How? Three items are required.

- **Acknowledgements:** Typically, reliable delivery is achieved using the “acknowledgments with retransmission" paradigm, whereby the receiver. returns a special acknowledgment (ACK) frame to the sender indicating the correct receipt of a frame. In some systems, the receiver also returns a negative acknowledgment (NACK) for incorrectly-received frames. This is nothing more than a hint to the sender so that it can retransmit a frame right away without waiting for a timer to expire.
- **Timers:** One problem that simple ACK/NACK schemes fail to address is recovering from a frame that is lost? Retransmission timers are used to resend frames that don't produce an ACK. When sending a frame, schedule a timer to expire at some time after the ACK should have been returned. If the timer goes o, retransmit the frame.
- **Sequence Numbers:** Retransmissions introduce the possibility of duplicate frames. To suppress duplicates, add sequence numbers to each frame, so tha1t 9a receiver can distinguish between new frames and old copies.

FLOW CONTROL:

Flow control deals with controlling the speed of the sender to match that of the receiver.

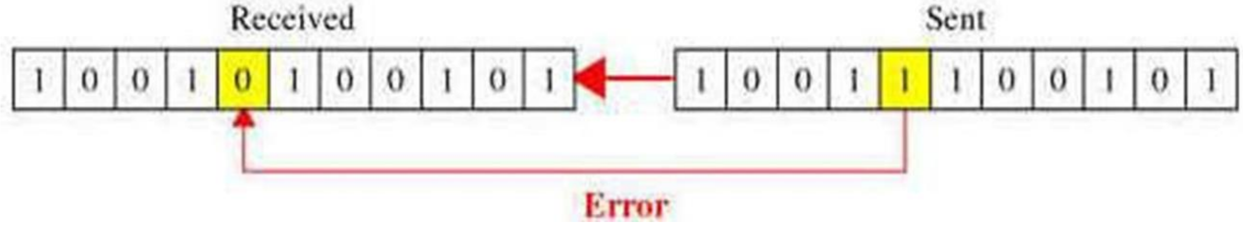
Two Approaches:

- **feedback-based flow control**, the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing
- **rate-based flow control**, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

TYPES OF ERRORS:

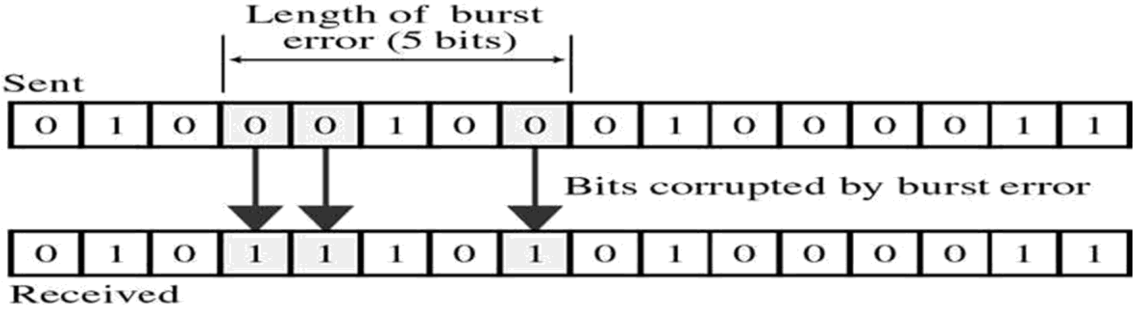
There are two main types of errors in transmissions:

1. **Single bit error:** It means only one bit of data unit is changed from 1 to 0 or from 0 to 1.



2. **Burst error:** It means two or more bits in data unit are changed from 1 to 0 from 0 to 1. In burst error, it is not necessary that only consecutive bits are changed. The

length of burst error is measured from first changed bit to last changed bit

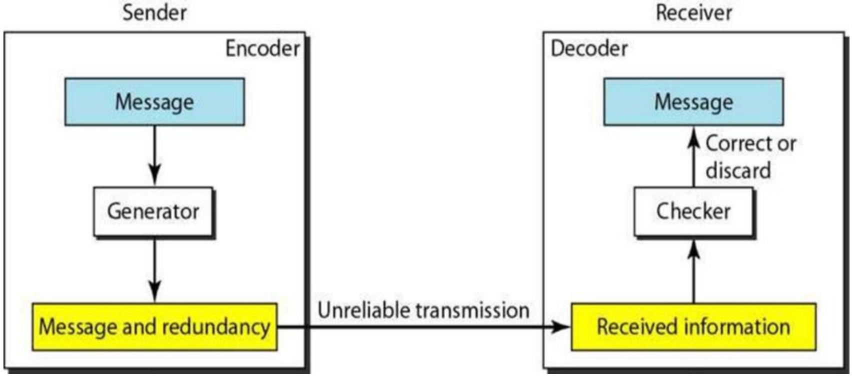


ERROR DETECTION AND CORRECTION:

Network designers have developed two basic strategies for dealing with errors. Both add redundant information to the data that is sent. One strategy is to include enough redundant information to enable the receiver to deduce what the transmitted data must have been. The other is to include only enough redundancy to allow the receiver to deduce that an error has occurred and have it request a retransmission. The former strategy uses **error-correcting codes** and the latter uses **error-detecting codes**.

Error Detecting Codes: Include enough redundancy bits to *detect* errors and use ACKs and retransmissions to recover from the errors.

Error Correcting Codes: Include enough redundancy to detect and correct errors.



ERROR-DETECTING CODES:

Error detection means to decide whether the received data is correct or not without having a copy of the original message. Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.

1. **Parity Check/ Vertical Redundancy Check(VRC)**

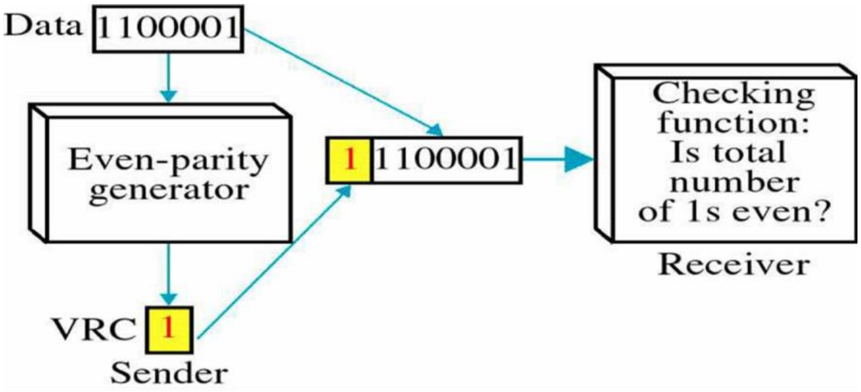
   Append a single bit at the end of data block such that the number of one's is even

   →À Even Parity (odd parity is similar)

   0110011 →01100110

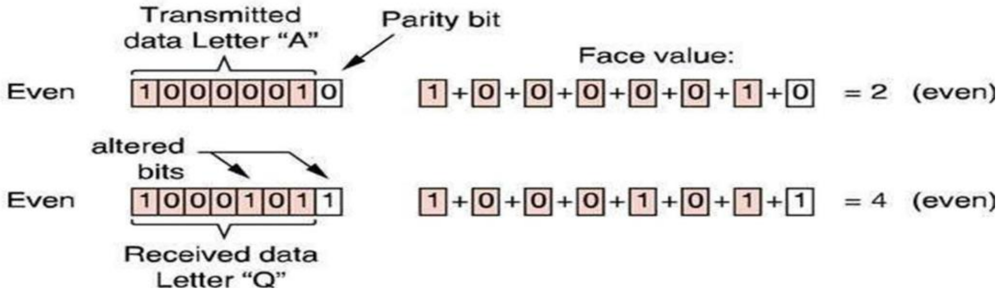
   0110001 → 01100011

   VRC is also known as **Parity Check**. Detects all odd-number errors in a data block.



The problem with parity is that it can only detect odd numbers of bit substitution errors, i.e. 1 bit, 3bit, 5, bit, etc. Errors. If there two, four, six, etc. bits which are transmitted in error, using VRC will not be able to detect the error.

Multiple bit errors/even parity



2. Cyclic Redundancy Check(CRC):

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. The CRC (**Cyclic Redundancy Check**), also known as a **polynomial code**

Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients 1, 1, 0, 0, 0, and 1: **1x5 + 1x4 + 0x3 + 0x2 + 0x1 + 1x0.**

Polynomial arithmetic is done modulo 2, according to the rules of algebraic field theory. It does not have carries for addition or borrows for subtraction. Both addition and subtraction are identical to **exclusive OR.** For example:

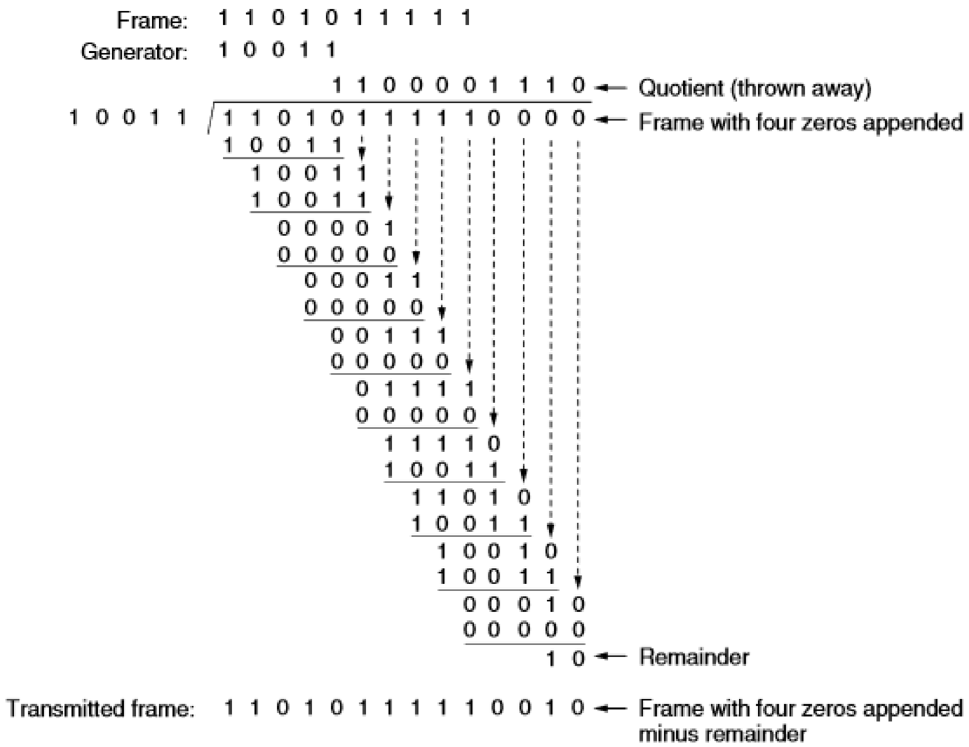
10011011	**00110011**	**11110000**	**01010101**
+ 11001010	**+ 11001101**	**− 10100110**	**– 10101111**
01010001	**11111110**	**01010110**	**11111010**

When the polynomial code method is employed, the sender and receiver must agree upon a **generator polynomial, G(x),** in advance. Both the high- and low order bits of the generator must be 1. To compute the CRC for some frame with m bits corresponding to the polynomial M(x), the frame must be longer than the generator polynomial. The idea is to append a CRC to the end of the frame in such a way that the polynomial represented by the check summed frame is divisible by G(x). When the receiver gets the check summed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error.

The **algorithm for computing the CRC is as follows:**

1. Let r be the degree of G(x). Append r zero bits to the low-order end of the frame so it now contains m + r bits and corresponds to the polynomial $x^rM(x)$.
2. Divide the bit string corresponding to G(x) into the bit string corresponding to $x^rM(x)$, using modulo 2 divisions.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^rM(x)$ using modulo 2 subtractions. The result is the check summed frame to be transmitted. Call its polynomial T(x).

Below figure illustrates the calculation for a frame 1101011111 using the generator G(x) = x4 + x + 1.


Frame: 1 1 0 1 0 1 1 1 1 1
Generator: 1 0 0 1 1
1 1 0 0 0 0 1 1 1 0 ← Quotient (thrown away)
1 0 0 1 1 / 1 1 0 1 0 1 1 1 1 1 0 0 0 0 ← Frame with four zeros appended
1 0 ← Remainder
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0 ← Frame with four zeros appended minus remainder

Example calculation of the CRC.

It should be clear that T(x) is divisible (modulo 2) by G(x). In any division problem, if you diminish the dividend by the remainder, what is left over is divisible by the divisor.

Example of CRC:

- Consider a message 110010 represented by the polynomial M(x) = $x^5 + x^4 + x$
  Consider a *generating polynomial* G(x) = $x^3 + x^2 + 1$ (1101)

This is used to generate a 3 bit CRC = C(x) to be appended to M(x).

Steps:

1. Multiply M(x) by x^3 (highest power in G(x)). i.e. Add 3 zeros. 110010000
2. Divide the result by G(x). The remainder = C(x).

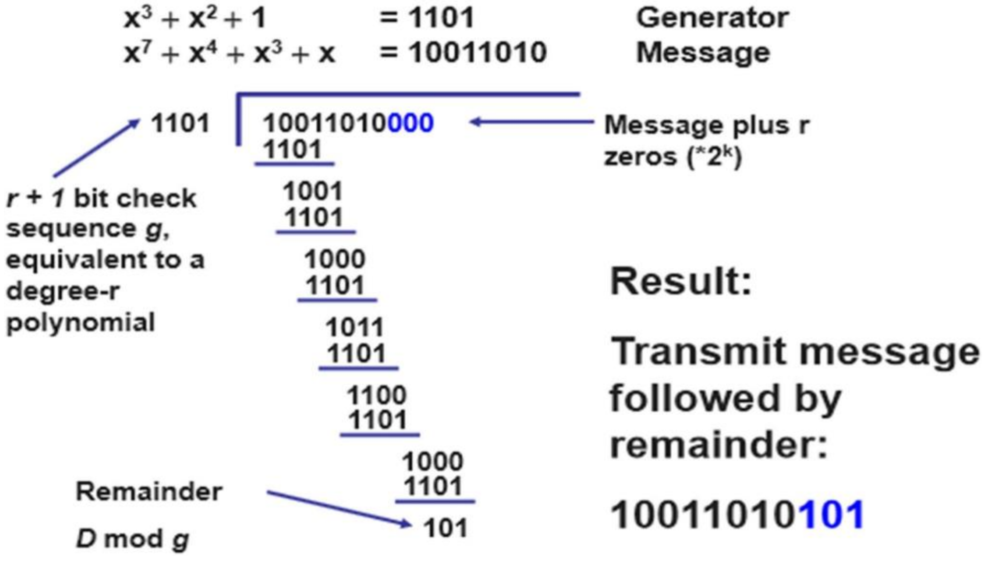
1101 long division into 110010000 (with subtraction mod 2)
= 100100 remainder **100**

3. Transmit 110010000 + 100

To be precise, transmit: T(x) = x^3M(x) + C(x)
= 110010100

4. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

At sender side calculation of CRC:



$x^3 + x^2 + 1$ = 1101 Generator

$x^7 + x^4 + x^3 + x$ = 10011010 Message

```
1101 | 10011010000
       1101
        1001
        1101
         1000
         1101
          1011
          1101
           1100
           1101
             1000
             1101
              101
```

Message plus r zeros (*2^k)

r + 1 bit check sequence *g*, equivalent to a degree-r polynomial

Remainder

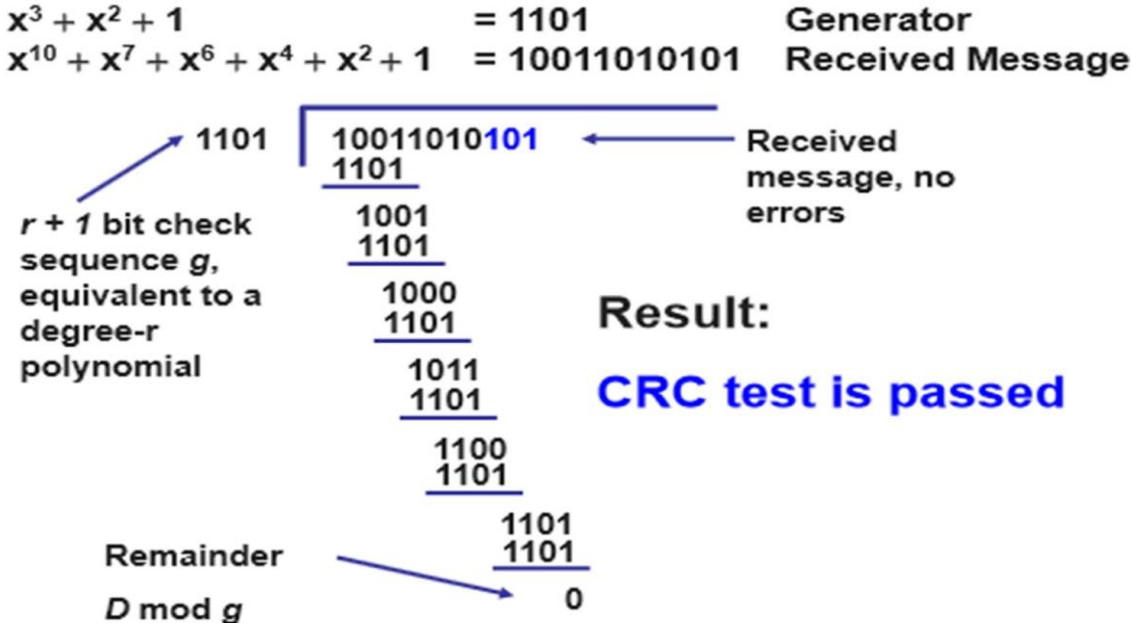
D mod *g*

Result:

Transmit message followed by remainder:

10011010101

At receiver side calculation of CRC:



$x^3 + x^2 + 1$ = 1101 Generator

$x^{10} + x^7 + x^6 + x^4 + x^2 + 1$ = 10011010101 Received Message

```
1101 | 10011010101
       1101
        1001
        1101
         1000
         1101
          1011
          1101
           1100
           1101
             1101
             1101
                0
```

Received message, no errors

r + 1 bit check sequence *g*, equivalent to a degree-r polynomial

Remainder

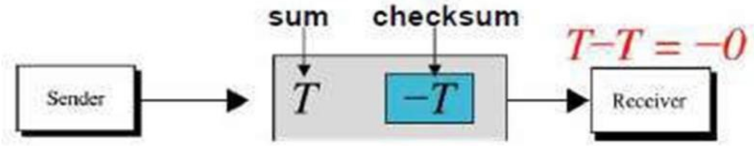
D mod *g*

Result:

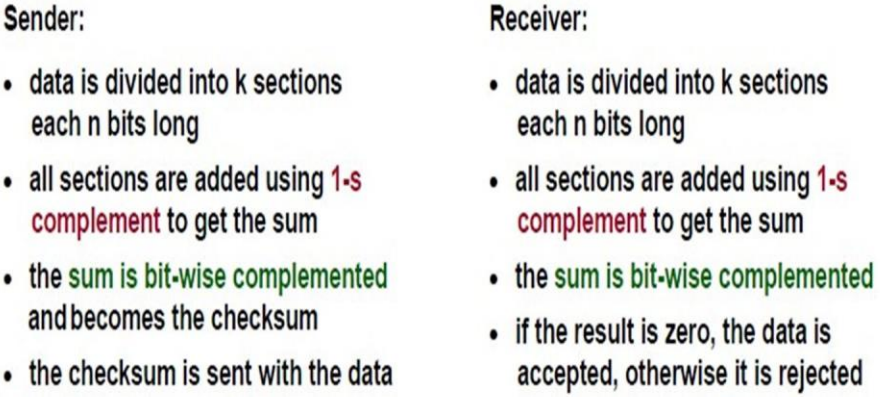
CRC test is passed

3. CHECKSUM:

- Checksum is the error detection scheme used in IP, TCP & UDP.
- Here, the data is divided into k segments each of n bits. In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum. The checksum segment is sent along with the data segments
- At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented. If the result is zero, the received data is accepted; otherwise discarded
- The checksum detects all errors involving an odd number of bits. It also detects most errors involving even number of bits.



Checksum procedure at sender and receiver end:



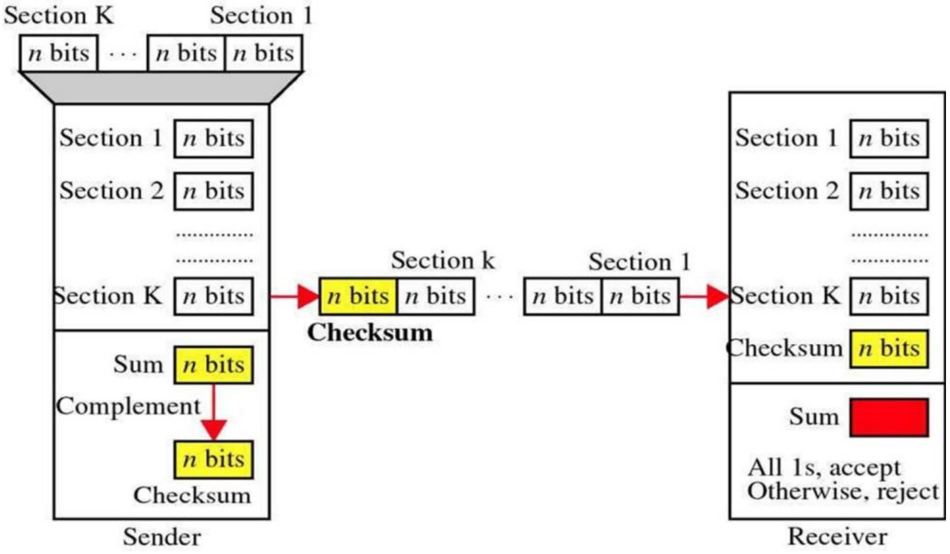
Sender:

- data is divided into k sections each n bits long
- all sections are added using 1-s complement to get the sum
- the sum is bit-wise complemented and becomes the checksum
- the checksum is sent with the data

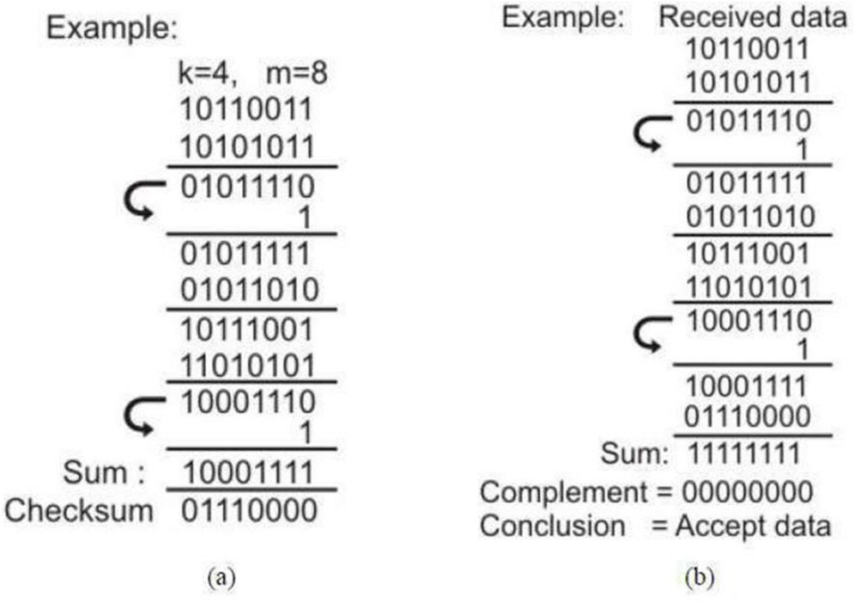
Receiver:

- data is divided into k sections each n bits long
- all sections are added using 1-s complement to get the sum
- the sum is bit-wise complemented
- if the result is zero, the data is accepted, otherwise it is rejected

Diagrammatic approach:



Example for Checksum:



```
Example:
            k=4,   m=8
            10110011
            10101011
          ----------
         ↺  01011110
                   1
          ----------
            01011111
            01011010
          ----------
            10111001
            11010101
          ----------
         ↺  10001110
                   1
          ----------
   Sum :    10001111
          ----------
Checksum    01110000
```

(a)

```
Example:   Received data
            10110011
            10101011
          ----------
         ↺  01011110
                   1
          ----------
            01011111
            01011010
          ----------
            10111001
            11010101
         ↺  10001110
                   1
          ----------
            10001111
            01110000
          ----------
      Sum:  11111111
Complement = 00000000
Conclusion   = Accept data
```

(b)

(a) Sender's end for the calculation of the checksum. (b) Receiving end for checking the checksum

ERROR-CORRECTING CODES

We will examine four different error-correcting codes:

1. Hamming codes.
2. Binary convolutional codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

All of these codes add redundancy to the information that is sent. A frame consists of *m* data (i.e., message) bits and *r* redundant (i.e. check) bits. In a systematic code, the *m* data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent. In a linear code, the *r* check bits are computed as a linear function of the *m* data bits. Exclusive OR (XOR) or modulo 2 addition is a popular choice.The total length of a block be *n* (i.e., *n* = *m* + *r*). We will describe this as an (*n*,*m*) code. An *n*-bit unit containing data and check bits is referred to as an *n* bit **codeword.** To understand how errors can be handled, it is necessary to first look closely at what an error really is. Given any two codewords that may be transmitted or received—say, 10001001 and 10110001—it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just XOR the two codewords and count the number of 1 bits in the result.The number of bit positions in which two codewords differ is called the **Hamming distance**

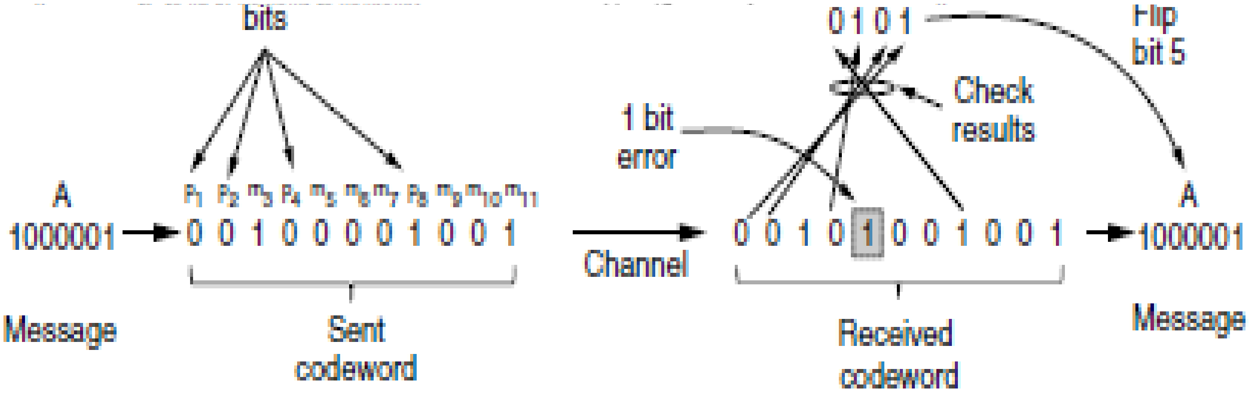


Figure 3-6. Example of an (11, 7) Hamming code correcting a single-bit error.

In Fig. 3-6, a single-bit error occurred on the channel so the check results are 0, 1, 0, and 1 for k = 8, 4, 2, and 1, respectively. This gives a syndrome of 0101 or 4 + 1=5.Hamming distances are valuable for understanding block codes, and Hamming codes are used in error-correcting memory

- The second code we will look at is a **convolutional code**. In a convolutional code, an encoder processes a sequence of input bits and generates a sequence of output bits.The output depends on the current and previous input bits. The number of previous bits on which the output depends is called the **constraint length** of the code. Convolutional codes are specified in terms of their rate and constraint length.Convolutional codes are widely used in deployed networks, for example, as part of the GSM mobile phone system, in satellite communications.As an example, a popular convolutional code is shown in Fig. 3-7. This code is known as the NASA convolutional code of r = 1⁄2 and k = 7.

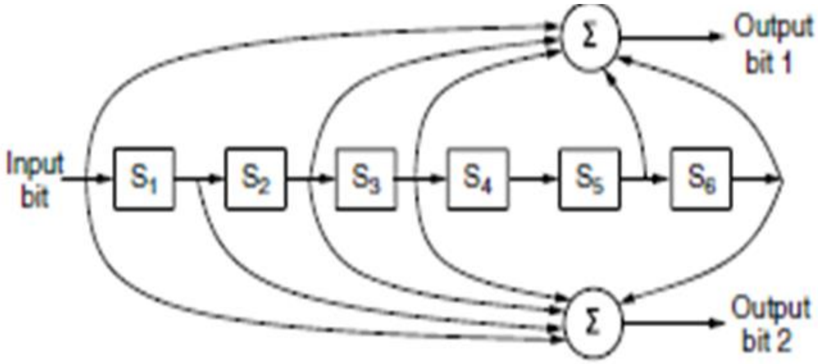


Figure 3-7. The NASA binary convolutional code used in 802.11.

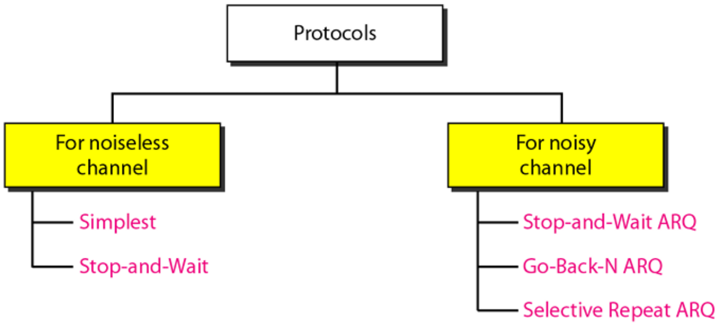
In Fig. 3-7, each input bit on the left-hand side produces two output bits on the right-hand side that are XOR sums of the input and internal state. Since it deals with bits and performs linear operations, this is a binary, linear convolutional code. Since 1 input bit produces 2 output bits, the code rate is ½. The internal state is kept in six memory registers. Each time another bit is input the values in the registers are shifted to the right. For example, if 111 is input and the initial state is all zeros, the internal state, written left to right, will become 100000, 110000, and 111000 after the first, second, and third bits have been input.It takes seven shifts to flush an input completely so that it does not affect the output. The constraint length of this code is thus k = 7.

- The third kind of error-correcting code is the **Reed-Solomon code**. Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too. Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on m bit symbols. Reed-Solomon codes are based on the fact that every n degree polynomial is uniquely determined by n + 1 points. Reed-Solomon codes are actually defined as polynomials that operate over finite fields, but they work in a similar manner. For m bit symbols, the code words are 2m−1 symbols long. Reed-Solomon codes are widely used in practice because of their strong error-correction properties, particularly for burst errors.
- The final error-correcting code is the **LDPC** (**Low-Density Parity Check**) code.LDPC codes are linear block codes that were invented by Robert Gallagher in his doctoral thesis.In an LDPC code, each output bit is formed from only a fraction of the input bits. This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code. The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. LDPC codes are practical for large block sizes and have excellent error-correction abilities that outperform many other codes (including the ones we have looked at) in practice

Elementary Data Link Layer protocols:

Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages.

We divide the discussion of protocols into those that can be used for noiseless (error-free) channels and those that can be used for noisy (error-creating) channels. The protocols in the first category cannot be used in real life, but they serve as a basis for understanding the protocols of noisy channels.



NOISELESS CHANNELS:

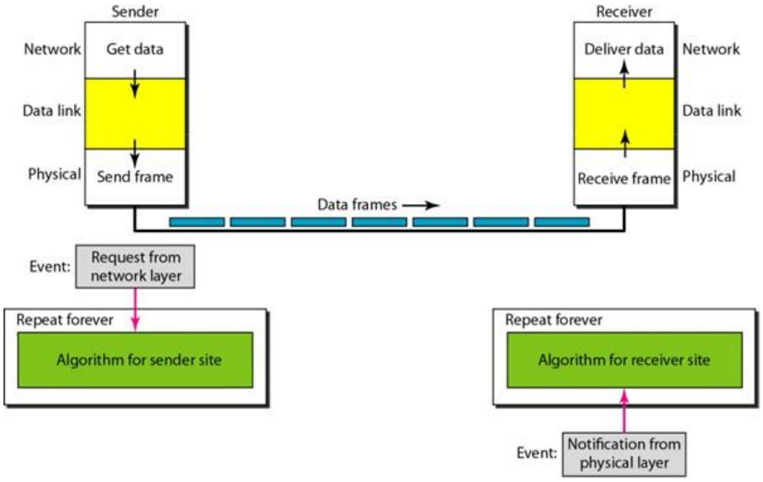
Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel. The first is a protocol that does not use flow control; the second is the one that does. Of course, neither has error control because we have assumed that the channel is a perfect noiseless channel.

Simplest Protocol:

Our first protocol, which we call the Simplest Protocol for lack of any other name, is one that has no flow or error control. Like other protocols we will discuss in this chapter, it is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver. We assume that the receiver can immediately handle any frame it receives with a processing time that is small enough to be negligible. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately. In other words, the receiver can never be overwhelmed with incoming frames.

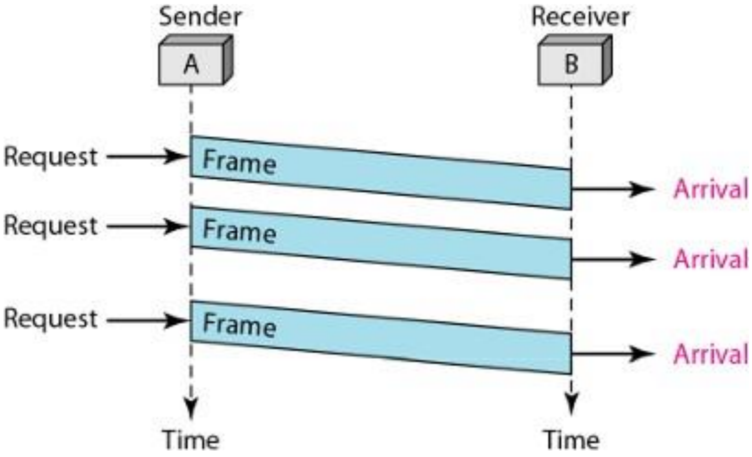
Design

There is no need for flow control in this scheme. The data link layer at the sender site gets data from its network layer, makes a frame out of the data, and sends it. The data link layer at the receiver site receives a frame from its physical layer, extracts data from the frame, and delivers the data to its network layer. The data link layers of the sender and receiver provide transmission services for their network layers.



Example:

Below Figure shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.



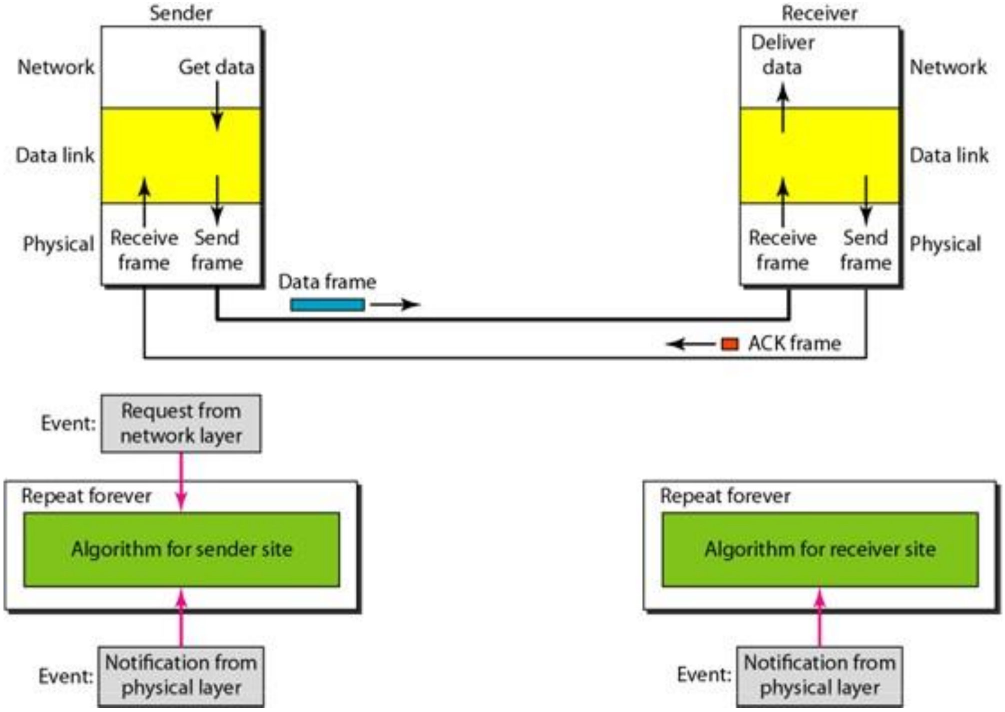
Stop-and-Wait Protocol:

If data frames arrive at the receiver site faster than they can be processed, the frames must be stored until their use. Normally, the receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service. To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

The protocol we discuss now is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

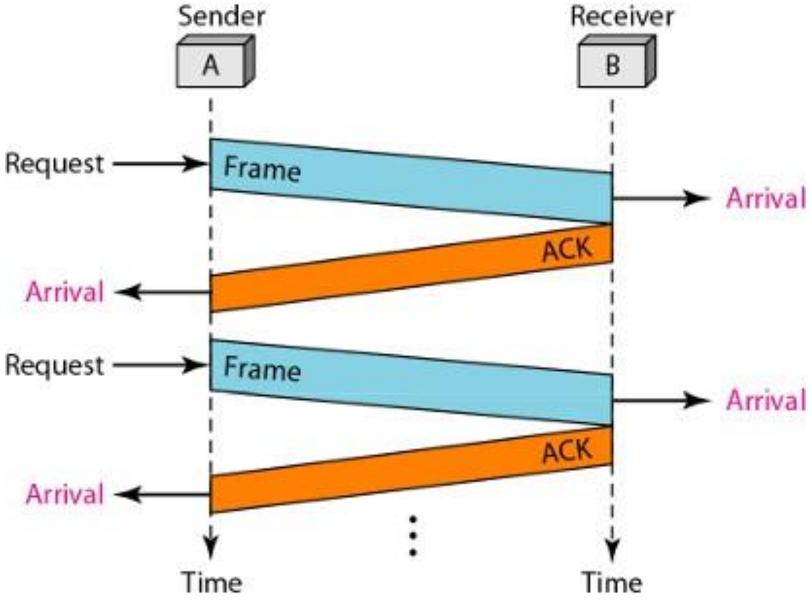
Design

We can see the traffic on the forward channel (from sender to receiver) and the reverse channel. At any time, there is either one data frame on the forward channel or one ACK frame on the reverse channel. We therefore need a half-duplex link.



Example:

Below Figure shows an example of communication using this protocol. It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.


Sender
A
Receiver
B
Request
Frame
Arrival
ACK
Arrival
Request
Frame
Arrival
ACK
Arrival
Time
Time

NOISY CHANNELS:

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request:

Our first protocol, called the Stop-and-Wait Automatic Repeat Request (Stop-and Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated.

The lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK even though several copies of the same frame can be in the network.

Sequence Numbers

As we discussed, the protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.

One important consideration is the range of the sequence numbers. Since we want to minimize the frame size, we look for the smallest range that provides unambiguous communication. The sequence numbers of course can wrap around. For example, if we decide that the field is *m* bits long, the sequence numbers start from 0, go to 2^m - 1, and then are repeated.

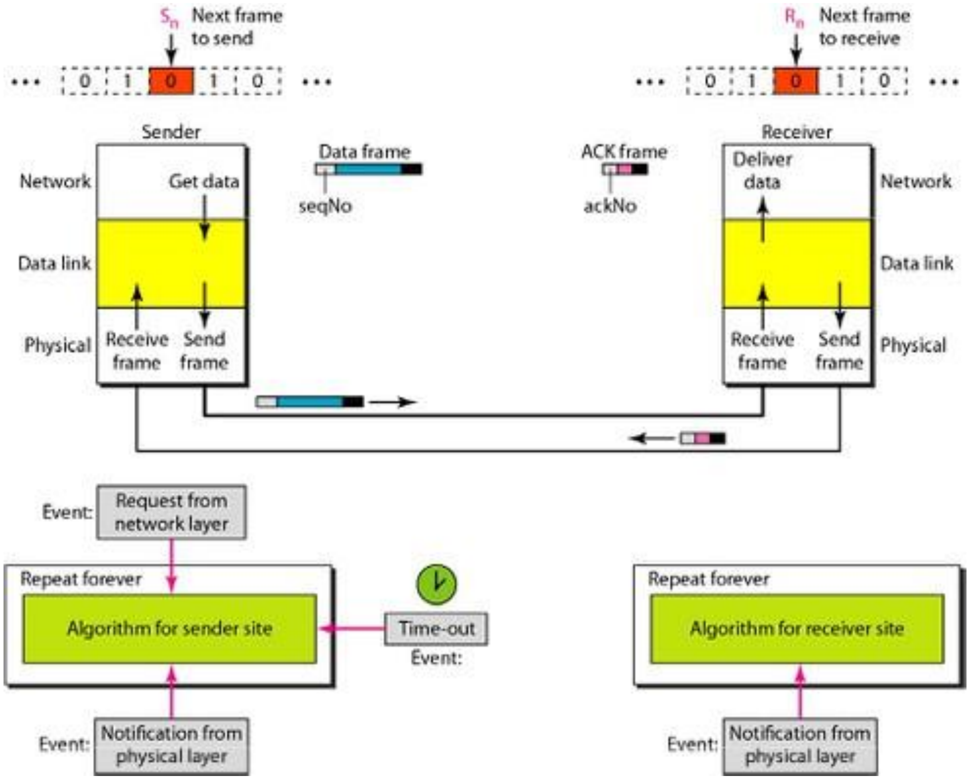
Acknowledgment Numbers

Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1 (meaning frame 1 is expected next). If frame 1 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 0 (meaning frame 0 is expected).

Design

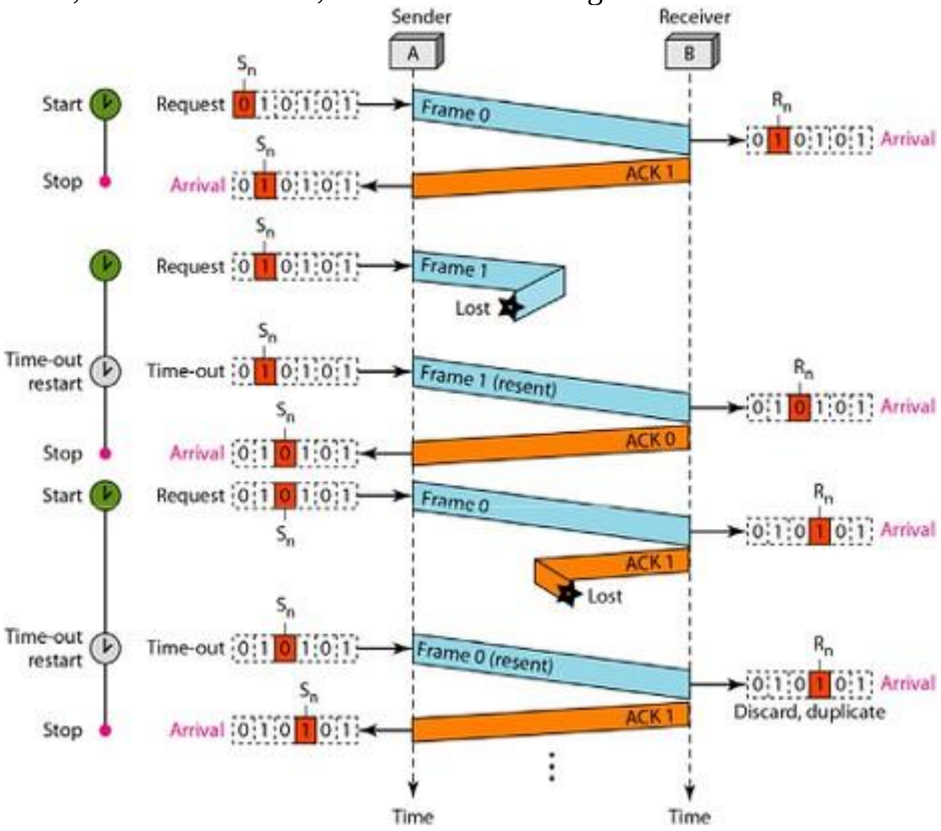
Below Figure shows the design of the Stop-and-WaitARQ Protocol. The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frames uses a seqNo (sequence number); an ACK frame uses an ackNo (acknowledgment number). The sender has a control variable, which we call *Sn* (sender, next frame to send), that holds the sequence number for the next frame to be sent (0 or 1).

The receiver has a control variable, which we call *Rn* (receiver, next frame expected), that holds the number of the next frame expected. When a frame is sent, the value of *Sn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. When a frame is received, the value of *Rn* is incremented (modulo-2), which means if it is 0, it becomes 1 and vice versa. Three events can happen at the sender site; one event can happen at the receiver site. Variable *Sn* points to the slot that matches the sequence number of the frame that has been sent, but not acknowledged; *Rn* points to the slot that matches the sequence number of the expected frame.



Example

Below Figure shows an example of Stop-and-Wait ARQ. Frame **a** is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame **a** is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.



Pipelining

In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining. There is no pipelining in Stop-and-Wait ARQ because we need to wait for a frame to reach the destination and be acknowledged before the next frame can be sent. However, pipelining does apply to our next two protocols because several frames can be sent before we receive news about the previous frames. Pipelining improves the efficiency of the transmission if the number of bits in transition is large with respect to the bandwidth-delay product.

Go-Back-N Automatic Repeat Request:

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment.

Go-Back-N Automatic Repeat Request protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows *m* bits for the sequence number, the sequence numbers range from 0 to *2m* - 1. For example, if *m* is 4, the only sequence numbers are 0 through 15 inclusive. However, we can repeat the sequence. So the sequence numbers are

0, 1,2,3,4,5,6, 7,8,9, 10, 11, 12, 13, 14, 15,0, 1,2,3,4,5,6,7,8,9,10, 11, ...

In other words, the sequence numbers are modulo-2^m

.

Sliding Window

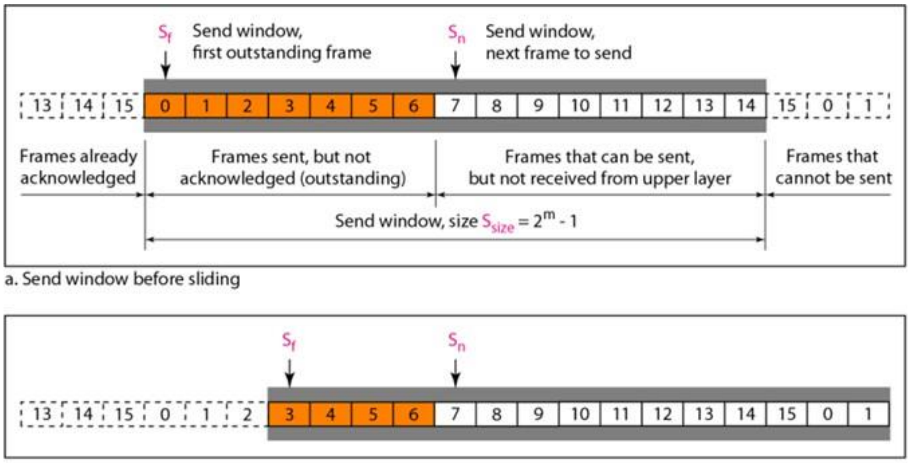
In this protocol the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers. The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window.

The send window is an imaginary box covering the sequence numbers of the data frames which can be in transmit. In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is 2^m - 1 we let the size be fixed and set to the maximum value, below figure **a** shows a sliding window of size 15 *(m* =4).

The window at any time divides the possible sequence numbers into four regions. The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them. The second region, colored in Figure **a**, defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames. The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides, as we see next.

Below Figure **b** shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. As we will see shortly, the acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure **b**, frames 0, I, and 2 are acknowledged, so the window has slid to the right three slots. Note that the value of *Sf* is 3 because frame 3 is now the first outstanding frame.

The window itself is an abstraction; three variables define its size and location at any time. We call these variables *Sf(send* window, the first outstanding frame), *Sn* (send window, the next frame to be sent), and Ssize (send window, size). The variable *Sf* defines the sequence number of the first (oldest) outstanding frame. The variable *Sn* holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window, which is fixed in our protocol.

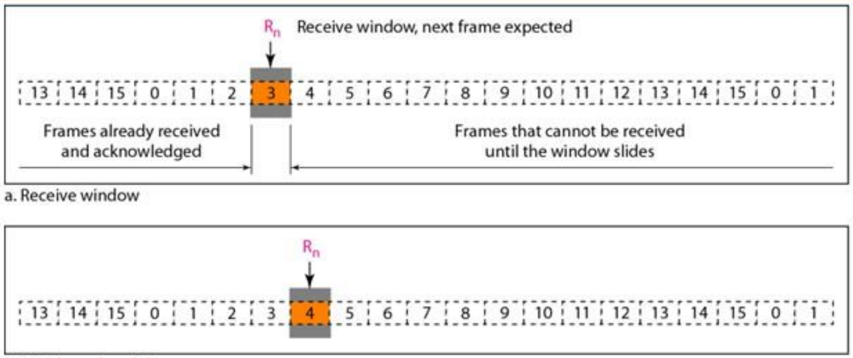

S_f Send window, first outstanding frame
S_n Send window, next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged (outstanding)
Frames that can be sent, but not received from upper layer
Frames that cannot be sent
Send window, size $S_{size} = 2^m - 1$
S_f
S_n
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1

a. Send window before sliding

b. Send window after sliding

The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent. The size of the receive window is always 1. The receiver is always looking for the arrival of a specific frame. Any frame arriving out of order is discarded and needs to be resent. Below figure shows the receive window.

We need only one variable *Rn* (receive window, next frame expected) to define this abstraction. The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of *Rn* is accepted and acknowledged.


R_n Receive window, next frame expected
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already received and acknowledged
Frames that cannot be received until the window slides
R_n
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1

a. Receive window

b. Window after sliding

Timers
Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

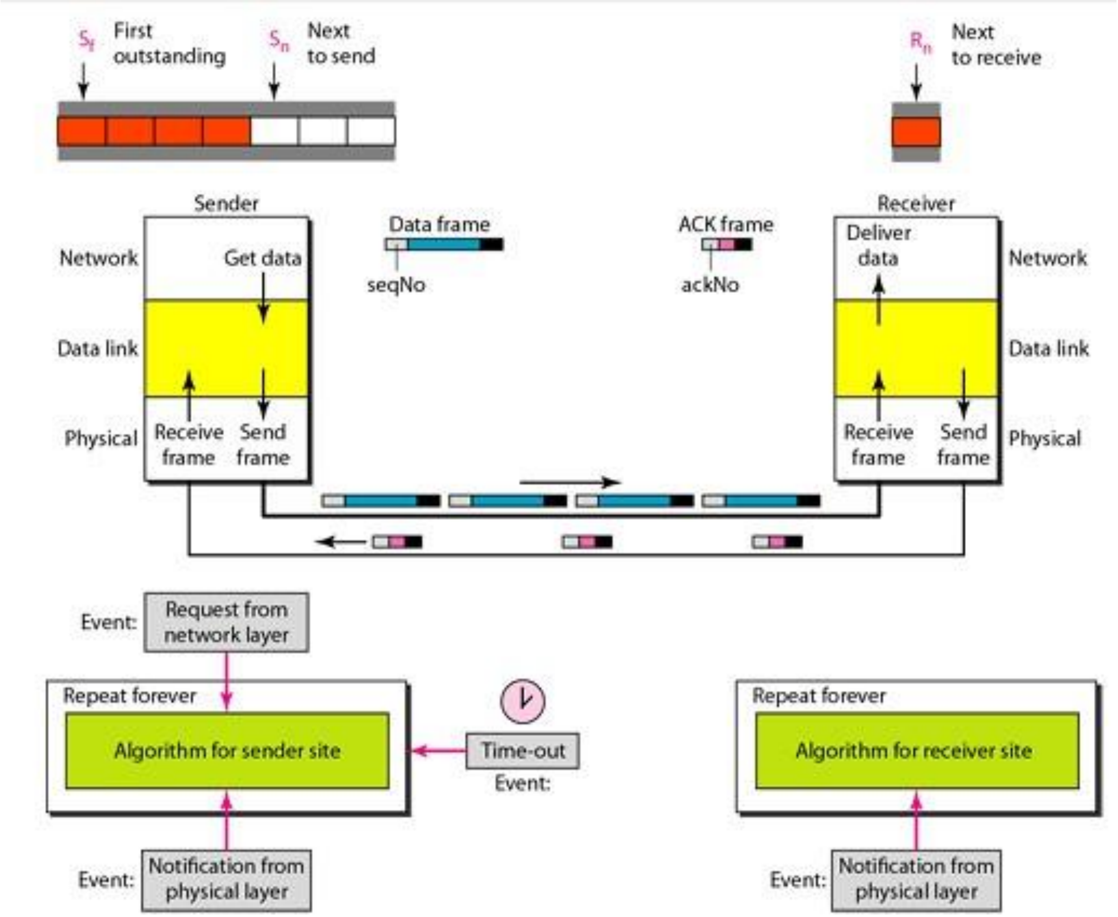
Acknowledgment
The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender site to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

Resending a Frame
When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender

goes back and sends frames 3, 4,5, and 6 again. That is why the protocol is called *Go-Back-N* ARQ.

Design

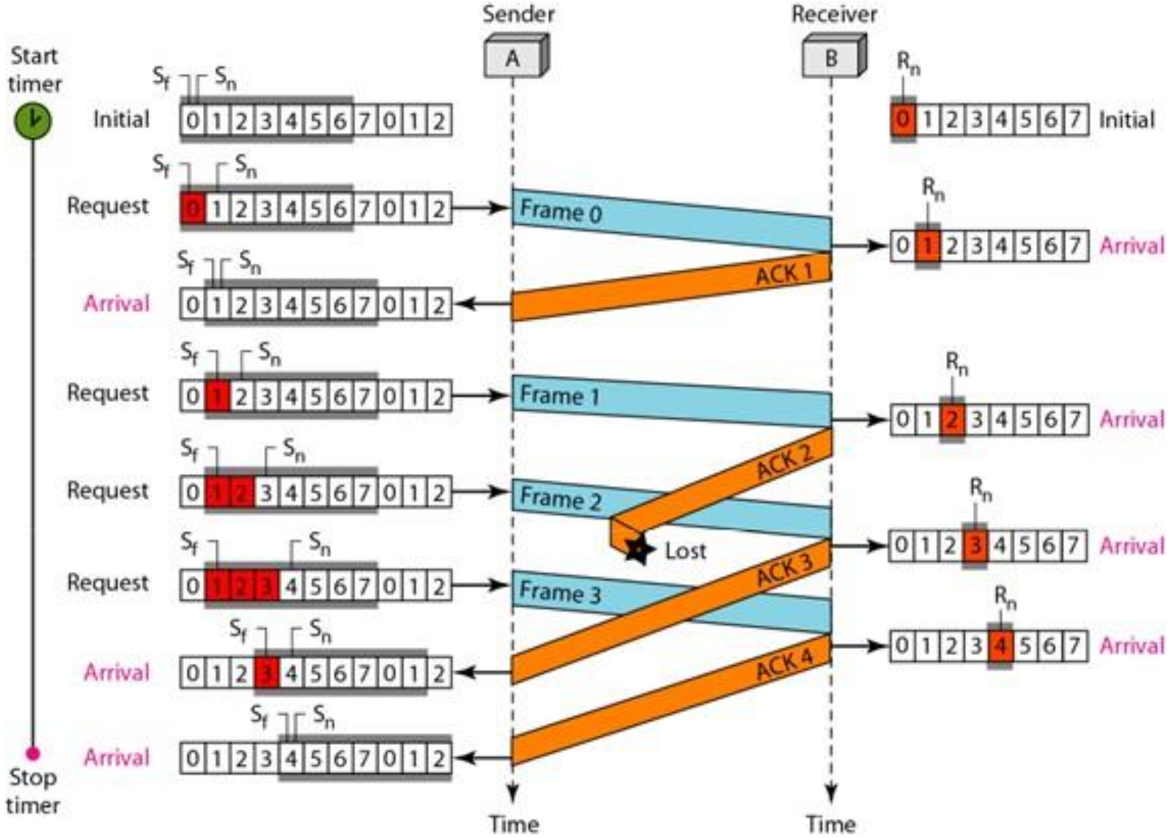
Below Figure shows the design for this protocol. As we can see, multiple frames can be in transit in the forward direction, and multiple acknowledgments in the reverse direction. The idea is similar to Stop-and-Wait ARQ; the difference is that the send window allows us to have as many frames in transition as there are slots in the send window.



Example

Below Figure shows an example of Go-Back-*N*. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost.

After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK3.

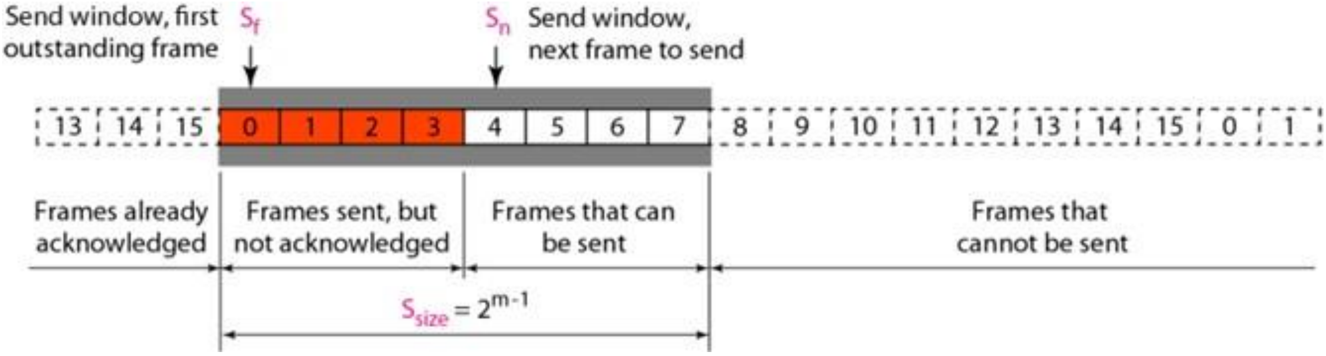

Sender
Receiver
A
B
Start timer
Stop timer
Initial
Request
Arrival
Request
Request
Request
Arrival
Arrival
S_f
S_n
R_n
Initial
Arrival
Arrival
Arrival
Arrival
Frame 0
ACK 1
Frame 1
ACK 2
Frame 2
Lost
ACK 3
Frame 3
ACK 4
0 1 2 3 4 5 6 7 0 1 2
0 1 2 3 4 5 6 7
Time
Time

Selective Repeat Automatic Repeat Request:

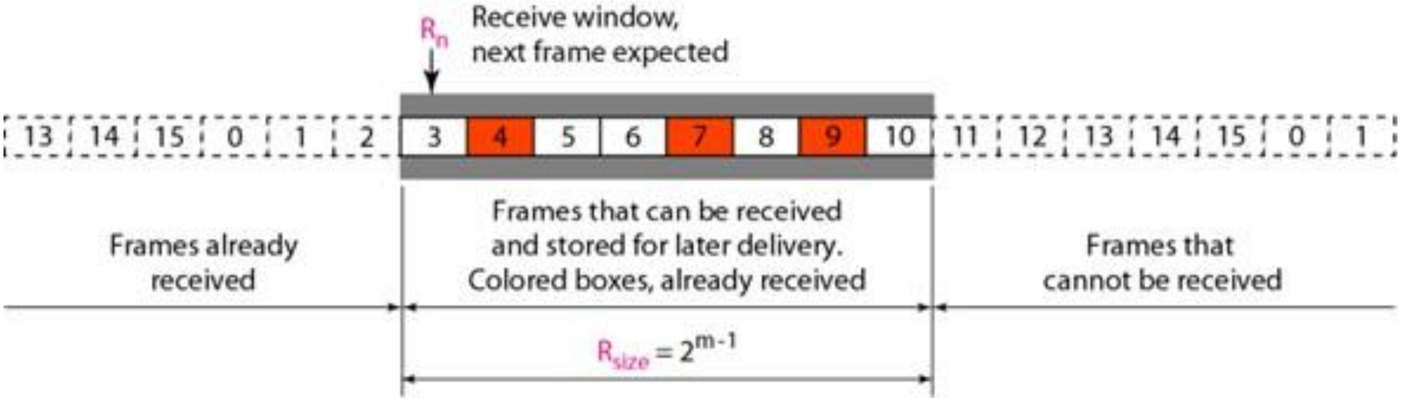
Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective RepeatARQ.

Windows

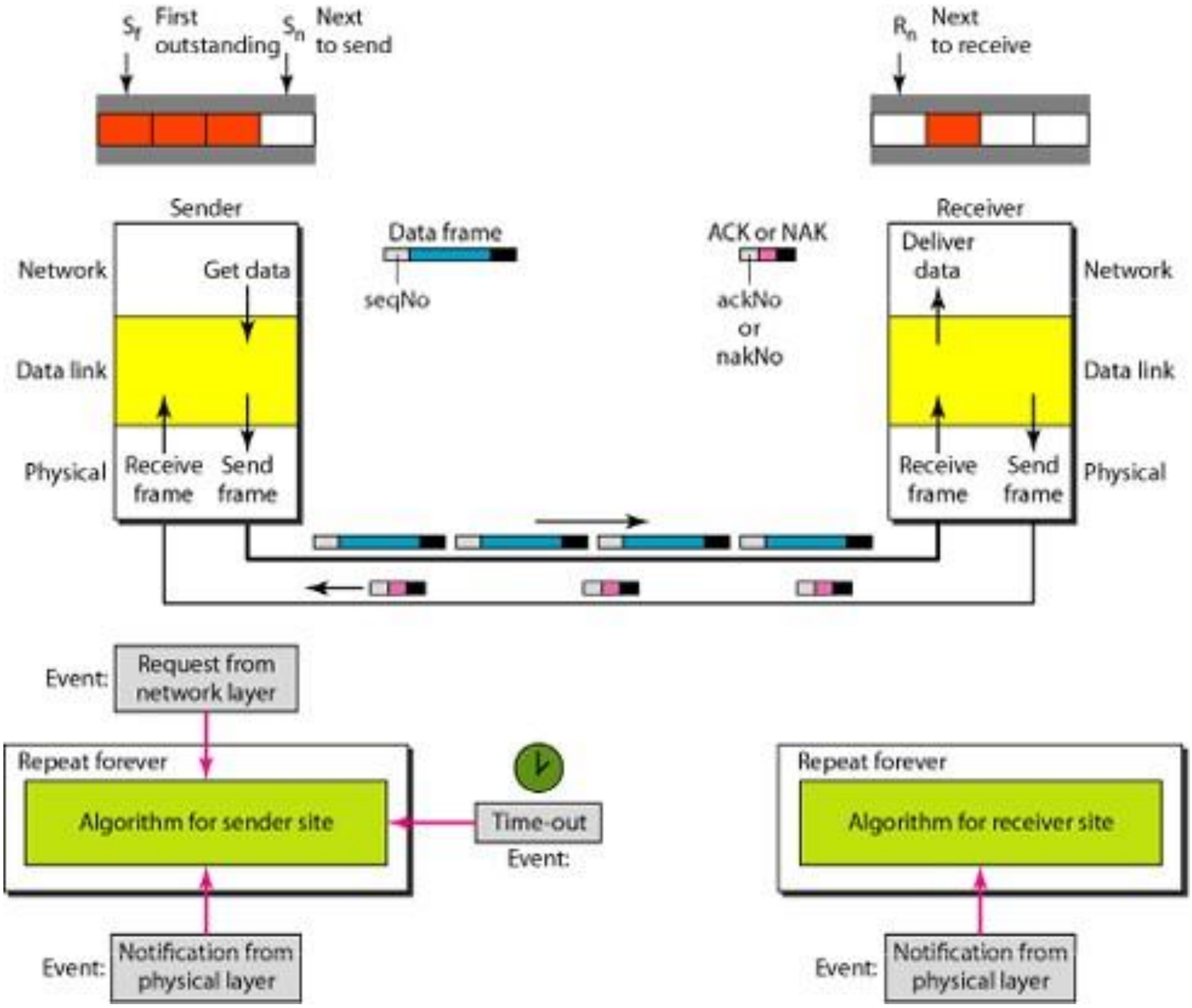
The Selective Repeat Protocol also uses two windows: a send window and a receive window. However, there are differences between the windows in this protocol and the ones in Go-Back-N. First, the size of the send window is much smaller; it is $2^{m}- I$. The reason for this will be discussed later. Second, the receive window is the same size as the send window. The send window maximum size can be $2^{m}- I$. For example, if $m = 4$, the sequence numbers go from 0 to 15, but the size of the window is just 8


Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
13
14
15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
Frames already acknowledged
Frames sent, but not acknowledged
Frames that can be sent
Frames that cannot be sent
$S_{size} = 2^{m-1}$

The receive window in Selective Repeat is totally different from the one in GoBack-N. First, the size of the receive window is the same as the size of the send window *(2^{m}- I)*. The Selective Repeat Protocol allows as many frames as the size of the receive window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same.

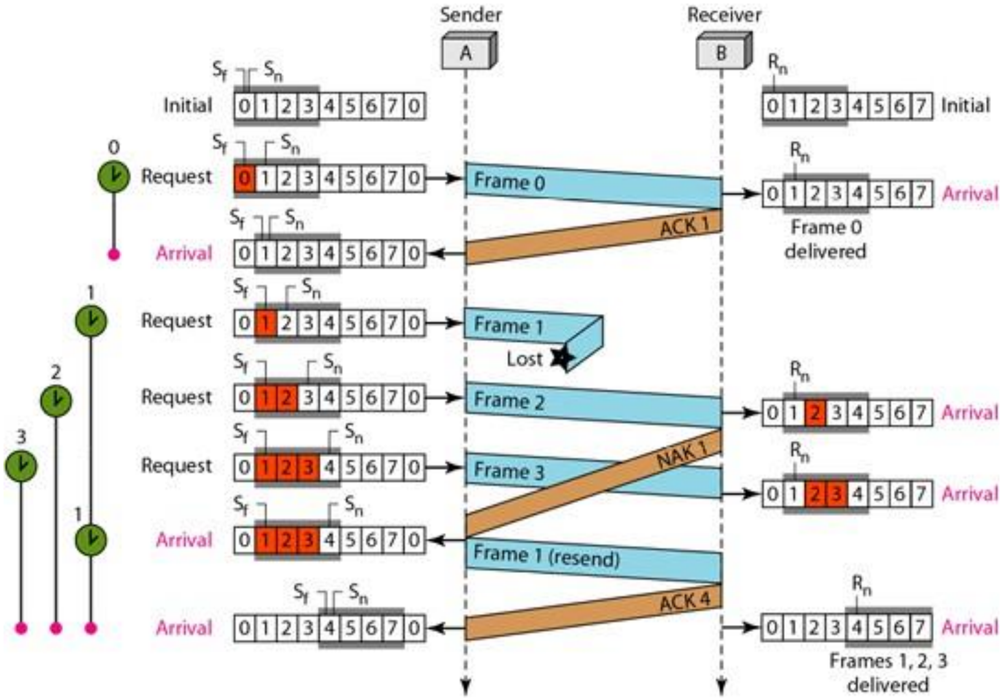

R_n
Receive window, next frame expected
13
14
15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

Design



Example

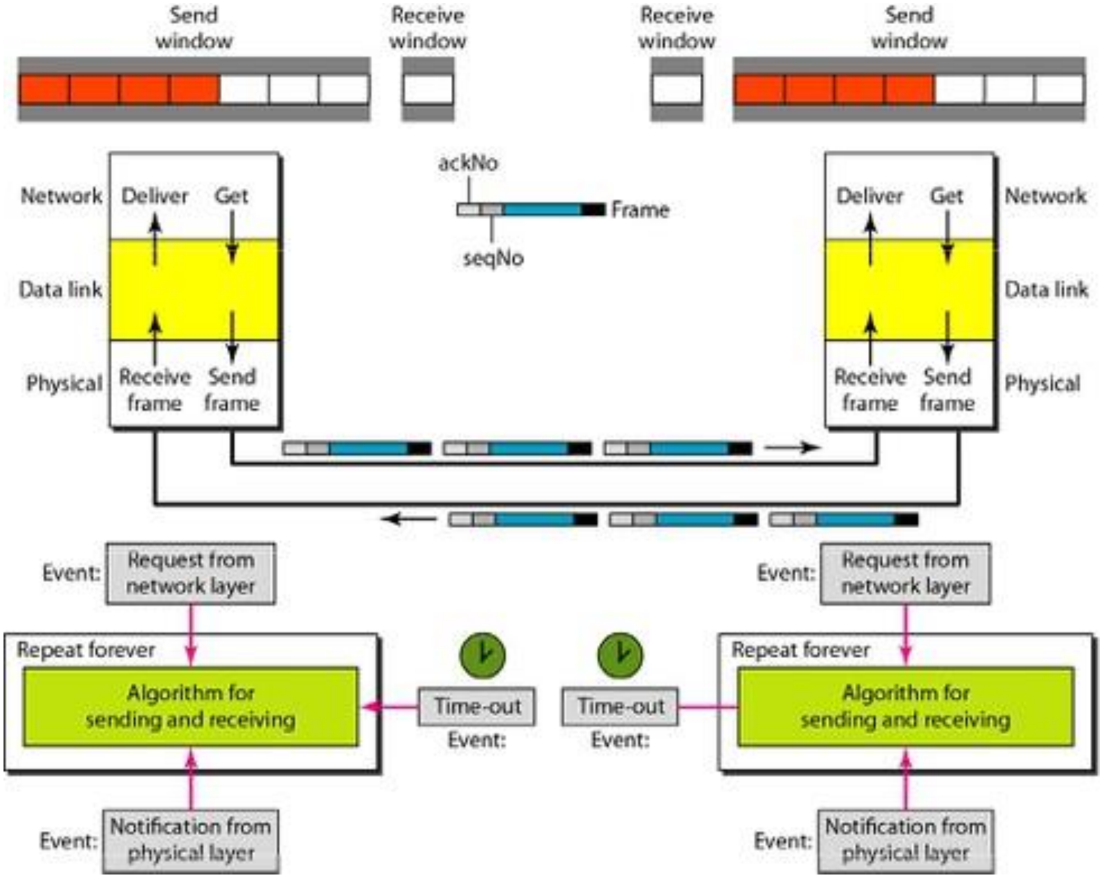
This example is similar to go back N Example in which frame 1 is lost. We show how Selective Repeat behaves in this case. Below Figure shows the situation.



Piggybacking:

The three protocols we discussed in this section are all unidirectional: data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction. In real life, data frames are normally flowing in both directions: from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions. A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

We show the design for a Go-Back-N ARQ using piggybacking in below Figure. Note that each node now has two windows: one send window and one receive window. Both also need to use a timer. Both are involved in three types of events: request, arrival, and time-out. However, the arrival event here is complicated; when a frame arrives, the site needs to handle control information as well as the frame itself. Both of these concerns must be taken care of in one event, the arrival event. The request event uses only the send window at each site; the arrival event needs to use both windows.



The Channel Allocation problem:

a) **Static channel allocation** in LANs & MANs

i) **FDM** ii) **TDM**

Drawbacks: -1) Channel is wasted if one or more stations do not send data.

2) If users increases this will not support.

b) **Dynamic channel allocation**

i) **Pure ALOHA & Slotted ALOHA**

ii) *CSMA*

- CSMA/CA
- CSMA/CD

Pure ALOHA

-1970's Norman Abramson end his colleagues devised this method, used ground –basedradio broad costing. This is called the **ALOHA** system.

-The basic idea, many users are competing for the use of a single shared channel.

-There are two versions of ALOHA: **Pure and Slotted**.

-Pure ALOHA does not require global time synchronization, where as in slotted ALOHAthe time is divided into discrete slots into which all frames must fit.

-Let users transmit whenever they have data to be sent.

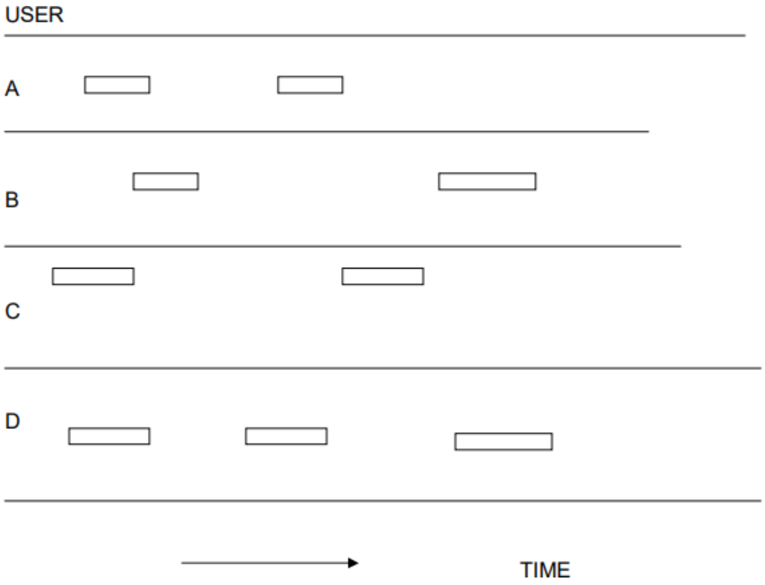
-There will be collisions and all collided frames will be damaged.

-Senders will know through feedback property whether the frame is destroyed or not bylistening

channel. [-With a LAN it is immediate, with a satellite, it will take 270m sec.]

-If the frame was destroyed, the sender waits random amount of time and again sendsthe frame.

-The waiting time must be random otherwise the same frame will collide over and over.



Frames are transmitted at completely arbitrary times

-Whenever two frames try to occupy the channel at the same time, there will be a collisionand both will be destroyed.

-We have to find out what is the efficiency of an ALOHA channel?

-Let us consider an infinite collection of interactive users sitting at their systems (stations).

-A user will always in two states **typing or waiting**.

-Let the ‘Frame time’ denotes the time required to transmit one fixed length frame.

-Assume that infinite populations of users are generating new frames according topossion distribution with mean N frames per frame time.

-If N>1 users are generating frames at a higher rate than the channel can handle.

-For reasonable throughput 0<N<1.

-In addition to new frames, the station also generates retransmission of frames.

-Old and new frames are G per frame time.

-G$\geq$ N

-At low load there will be few collisions, so G ~ N

-Under all loads, the throughput S = GP$_{o,}$ where P$_o$ is the probability that a frame does notsuffer a collision.

-A frame will not suffer a collision if no other frames are sent with one frame time of itsstart.

-Let ‘t’ be the time required to send a frame.

-If any other user has generated a frame between time t_o and t_o+t, the end of that framewill collide with the beginning of the shaded frame.

-Similarly, any other frame started b/w t_o+t and t_o+2t will bump into the end of the shadedframe.

-The probability that ‘k’ frames are generated during a given frame time is given by thepossion distribution:

$P_r[k]$ =

$G^k e^{-G}$ k!

-The probability of zero frames is just e^{-G}

-In an interval two frame times long, the mean number at frames generated is 2G.

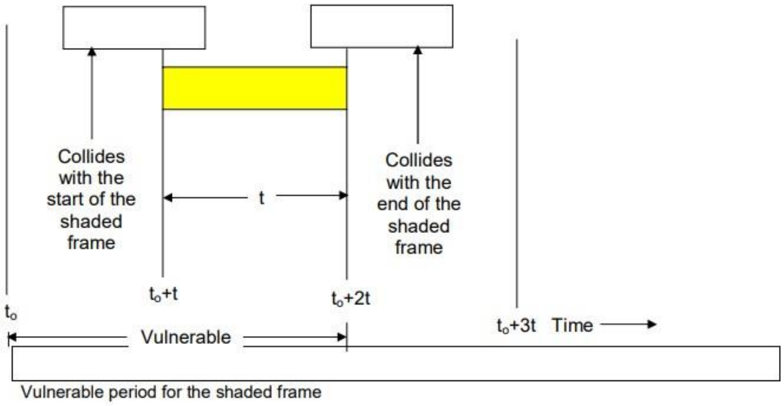
-The probability at no other traffic being initiated during the entire vulnerable period

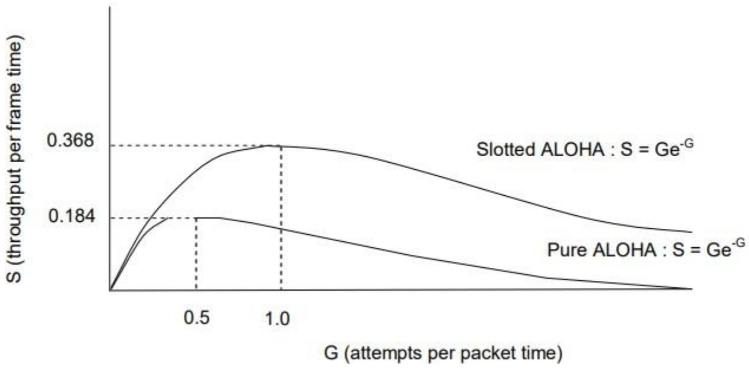
isgiven by $P_O = e^{-2G}$

$S = Ge^{-2G}$ $[S=GP_o]$

The Maximum through put occurs at G=0.5 with S=1/2e = 0.184

The channel utilization at pure ALOHA =18%.


Collides with the start of the shaded frame
Collides with the end of the shaded frame
t
t_o
t_o+t
t_o+2t
t_o+3t
Time
Vulnerable
Vulnerable period for the shaded frame



Throughput versus offered traffic for ALOHA systems

Slotted ALOHA

-In 1972, Roberts' devised a method for doubling the capacity of ALOHA system.

-In this system the time is divided into discrete intervals, each interval corresponding toone frame.

-One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

-In Roberts' method, which has come to be known as slotted ALOHA, in contrast to Abramson's pure ALOHA; a computer is not permitted to send whenever a carriage returnis typed.

-Instead, it is required to wait for the beginning of the next slot.

-Thus the continuous pure ALOHA is turned into a discrete one.

-Since the vulnerable period is now halved, the of no other traffic during the same slot as our test frame is e^{-G} which leads to

$S = Ge^{-G}$

- At G=1, slotted ALOHA will have maximum throughput.
- So S=1/e or about 0.368, twice that of pure ALOHA.
- The channel utilization is 37% in slotted ALOHA.

Carrier Sense Multiple Access Protocols

Protocols in which stations listen for a carrier (transmission) and act accordingly are called carries sense protocols.

Persistent CSMA

When a station has data to send, it first listens to the channel to see if any one else is transmitting at that moment. If the channel is busy, the station waits until it become idle. When the station detects an idle channel, it transmits a frame. If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1- persistent also because the station transmits with a probability of 1 when it finds the channel idle.

The propagation delay has an important effect on the performance of the protocol. The longer the propagation delay the worse the performance of the protocol.

Even if the propagation delay is zero, there will be collisions. If two stations listen the channel, that is idle at the same, both will send frame and there will be collision.

Non persistent CSMA

In this, before sending, a station sense the channel. If no one else is sending, the station begins doing so it self. However, if the channel is busy, the station does not continually sense it but it waits a random amount of time and repeats the process.

This algorithms leads to better channel utilization but longer delays then 1-persistent CSMA.

With persistent CSMA, what happens if two stations become active when a third station is busy? Both wait for the active station to finish, then simultaneously launch a packet, resulting a collision. There are two ways to handle this problem.

a) P-persistent CSMA b) exponential backoff.

P-persistent CSMA

The first technique is for a waiting station not to launch a packet immediately when the channel becomes idle, but first toss a coin, and send a packet only if the coin comes up heads. If the coin comes up tails, the station waits for some time (one slot for slotted CSMA), then repeats the process. The idea is that if two stations are both waiting for the medium, this reduces the chance of a collision from 100% to 25%. A simple generalization of the scheme is to use a biased coin, so that the probability of sending a packet when the medium becomes idle is not 0.5, but p, where $0< p < 1$. We call such a scheme **P-persistent CSMA**. The original scheme, where $p=1$, is thus called 1-persitent CSMA.

Exponential backoff

The key idea is that each station, after transmitting a packet, checks whether the packet transmission was successful. Successful transmission is indicated either by an explicit acknowledgement from the receiver or the absence of a signal from a collision detection circuit. If the transmission is successful, the station is done. Otherwise, the station retransmits the packet, simultaneously realizing that at least one other station is also contending for the medium. To prevent its retransmission from colliding with the other station's retransmission, each station backs off (that is, idles) for a random time chosen from the interval [0,2*max-propagation_delay] before retransmitting its packet. If the retransmission also fails, then the station backs off for a random time in the interval [0,4* max_propagation_delay], and tries again. Each subsequent collision doubles the backoff interval length, until the retransmission finally succeeds. On a successful transmission,the backoff interval is reset to the initial value. We call this type of backoff exponential backoff.

CSMA/CA

In many wireless LANS, unlike wired LANS, the station has no idea whether the packet collided with another packet or not until it receives an acknowledgement from receiver. In this situation, collisions have a greater effect on performance than with CSMA/CD, where colliding packets can be quickly detected and aborted. Thus, it makes sense to try to avoid collisions, if possible. CSMA/CA is basically p-persistence, with the twist that when

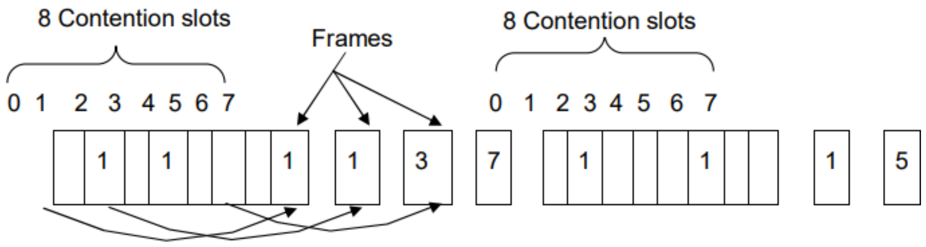
the medium becomes idle, a station must wait for a time called the interframe spacing or IFS before contending for a slot. A station gets a higher priority if it is allocated smaller inter frame spacing.

When a station wants to transmit data, it first checks if the medium is busy. If it is, it continuously senses the medium, waiting for it to become idle. When the medium becomes idle, the station first waits for an interframe spacing corresponding to its priority level, then sets a contention timer to a time interval randomly selected in the range [0,CW], where CW is a predefined contention window length. When this timer expires, it transmits a packet and waits for the receiver to send an ack. If no ack is received, the packet is assumed lost to collision, and the source tries again, choosing a contention timer at random from an interval twice as long as the one before(binary exponential backoff). If the station senses that another station has begun transmission while it was waiting for the expiration of the contention timer, it does not reset its timer, but merely freezer it, and restarts the countdown when the packet completes transmission. In this way, stations that happen to choose a longer timer value get higher priority in the next round of contention.

Collision-Free Protocols

A Bit-Map Protocol

In the basic bit-map method, each contention period consists of exactly N slots. If station0 has a frame to send, it transmits a 1 bit during the zeroth slot. No other station is allowed to transmit during this slot. Regardless of what station 0 does, station 1 gets the opportunity to transmit a 1during slot 1, but only if it has a frame queued. In general, station j may announce the fact that it has a frame to send by inserting a 1 bit into slot j. after all N slots have passed by, each station has complete knowledge of which stations with to transmit.



The basic bit-map protocol

The basic bit-map protocol

Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easily monitor, another Nbit contention period is begun. If a station becomes ready just after its bit slot has passed by, it is out of luck and must remain silent until every station has had a chance and the bit map has come around again. Protocols like this in which the desire to transmit is broadcast before the actual transmission are called reservation protocols.

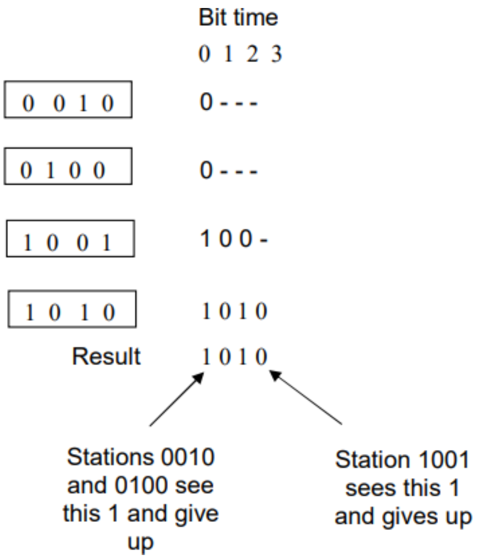
Binary Countdown

A problem with the basic bit-map protocol is that the overhead is 1 bit per station. A station wanting to use the channel now broadcasts its address as a binary bit string, starting with the high-order bit. All addresses are assumed to be the same length. Thebits in each address position from different stations are BOOLEAN ORed together. We will call this protocol binary countdown. It is used in Datakit.

As soon as a station sees that a high-order bit position that is 0 in its address has been overwritten with a 1, it gives up. For example, if station 0010,0100,1001, and 1010 are all trying to get the channel, in the first bit time the stations transmit 0,0,1, and 1, respectively. Stations 0010 and 0100 see the 1 and know that a higher- numbered station is competing for the channel, so they give up for the current round. Stations 1001 and 1010 continue.

The next bit is 0, and both stations continue. The next bit is 1, so station 1001 gives up. The winner is station 1010, because it has the highest address. After winning the bidding, it may now transmit a frame, after which another bidding cycle starts.

The binary countdown protocol. A dash indicates silence


Bit time
0 1 2 3
0 0 1 0
0 - - -
0 1 0 0
0 - - -
1 0 0 1
1 0 0 -
1 0 1 0
1 0 1 0
Result
1 0 1 0
Stations 0010 and 0100 see this 1 and give up
Station 1001 sees this 1 and gives up

COMPUTER NETWORKS

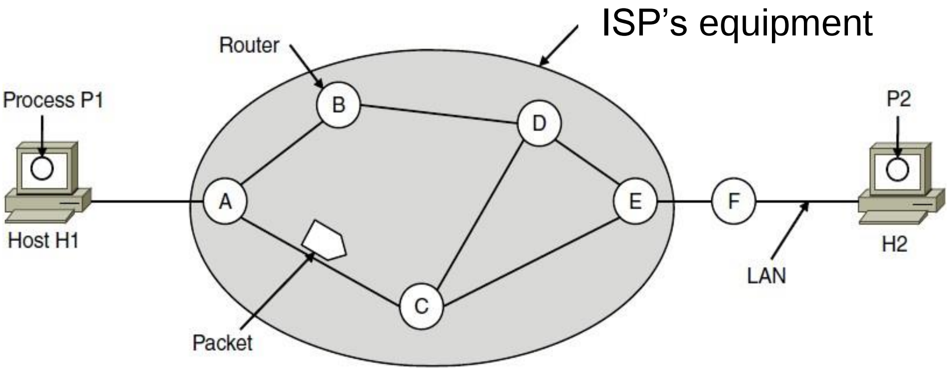
Module 3

The Network Layer

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other.

4.1 NETWORK LAYER DESIGN ISSUES:

4.1.1 Store-and-Forward Packet Switching:



The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host H1 is directly connected to one of the carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer.

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is stored there until it has fully arrived so the checksum can be verified. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is **store-and-forward packet switching.**

4.1.2 Services Provided to the Transport Layer:

The network layer provides services to the transport layer at the network layer/transport layer interface. The network layer services have been designed with the following goals in mind.

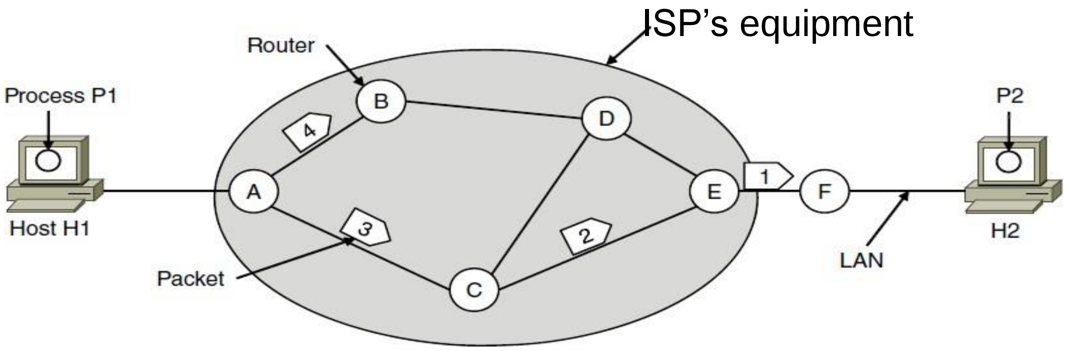
1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

The network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway, and there is usually little to be gained by doing it twice. Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.

4.1.3 Implementation of Connectionless Service:

In connectionless service, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** and the subnet is called a **datagram subnet.**

Let us now see how a datagram subnet works,



A's table (initially)

A	⊠
B	B
C	C
D	B
E	C
F	C

Dest. Line

A's table (later)

A	⊠
B	B
C	C
D	B
E	D
F	D

C's Table

A	A
B	A
C	⊠
D	E
E	E
F	E

E's Table

A	C
B	D
C	C
D	D
E	⊠
F	F

The process P1 on host H1 has a long message for P2 on host H2. The network layer has to break a message into four packets, 1, 2, 3, and 4 and sends each of them in turn to router A. A has only two outgoing lines to B and C so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. A's initial routing table is shown in the figure under the label "initially."

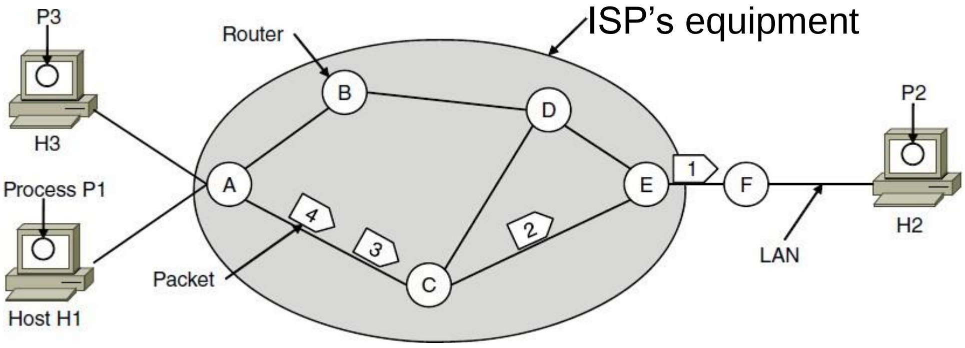
As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to C according to A's table. Packet 1 was then forwarded to E and then to F. When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN. Packets 2 and 3 follow the same route.

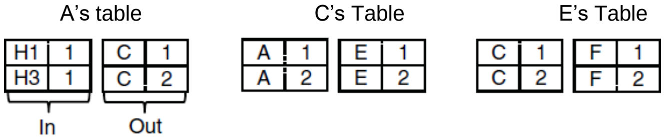
However, something different happened to packet 4. When it got to A it was sent to router B, even though it is also destined for F. For some reason, perhaps it learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label "later." The algorithm that manages the tables and makes the routing decisions is called the routing algorithm.

4.1.4 Implementation of Connection-Oriented Service:

In connection-oriented service, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a **VC (virtual circuit),** and the subnet is called a **virtual-circuit subnet.**

The idea behind virtual circuits is to avoid having to choose a new route for every packet sent. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.





A’s table

In		Out	
H1	1	C	1
H3	1	C	2

C’s Table

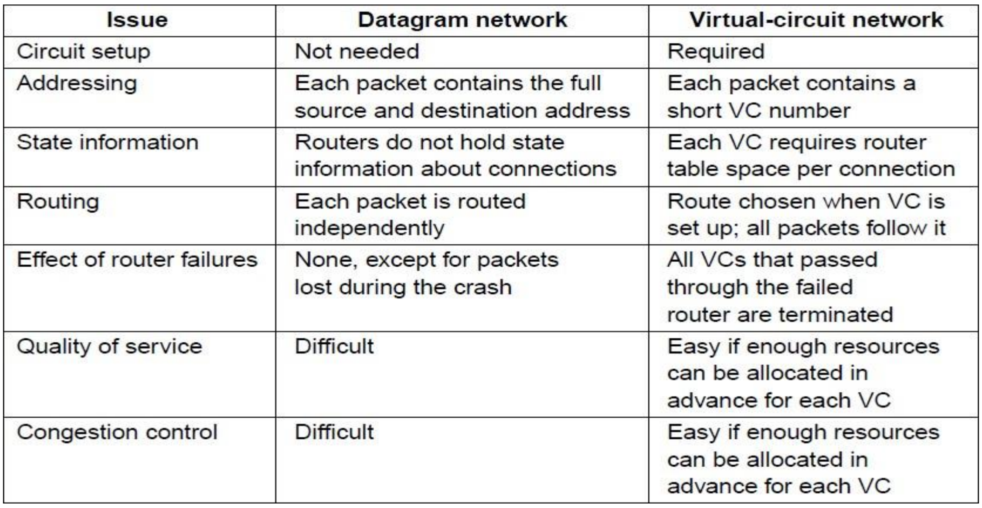
A	1	E	1
A	2	E	2

E’s Table

C	1	F	1
C	2	F	2

Here, host H1 has established connection 1 with host H2. It is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1. Similarly if H3 wants to connect to H2 they have to do the same procedure and it has to use different connection identifier in above example host H3 uses connection identifier as one.

4.1.5 Comparison of Virtual-Circuit and Datagram Subnets:



Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

4.2 ROUTING ALGORITHMS:

The main function of the network layer is routing packets from the source machine to the destination machine. In most subnets, packets will require multiple hops to make the journey. The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.

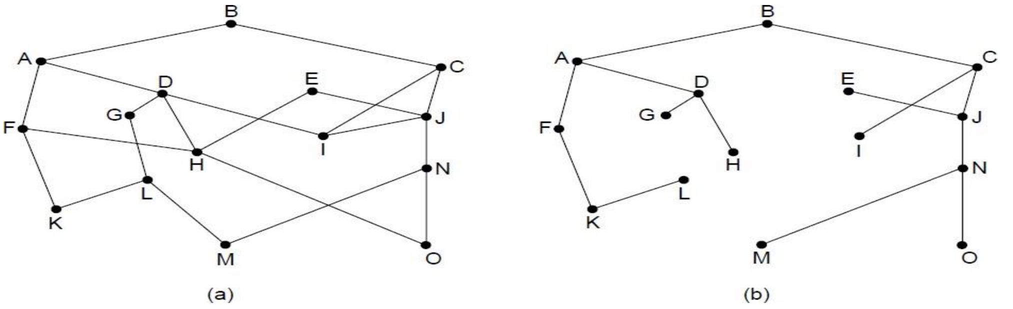
Routing algorithms can be grouped into two major classes: **nonadaptive** and **adaptive**. **Nonadaptive algorithms** do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, off-line, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes. This procedure is sometimes called **Dynamic routing.**

4.2.1 The Optimality Principle:

“**It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route**.” To see this, call the part of the route from I to Jr1 and the rest of the route r2. If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the route from I to K, contradicting our statement that r1r2 is optimal.
the set of optimal routes from all sources to a given destination form a tree rooted at the destination.

Such a tree is called a sink tree and is illustrated in below Fig. where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops.



4.2.2 Shortest Path Routing:

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABC and ABE in below Fig. are equally long. Another metric is the geographic distance in kilometers, in which case ABC is clearly much longer than ABE (assuming the figure is drawn to scale).

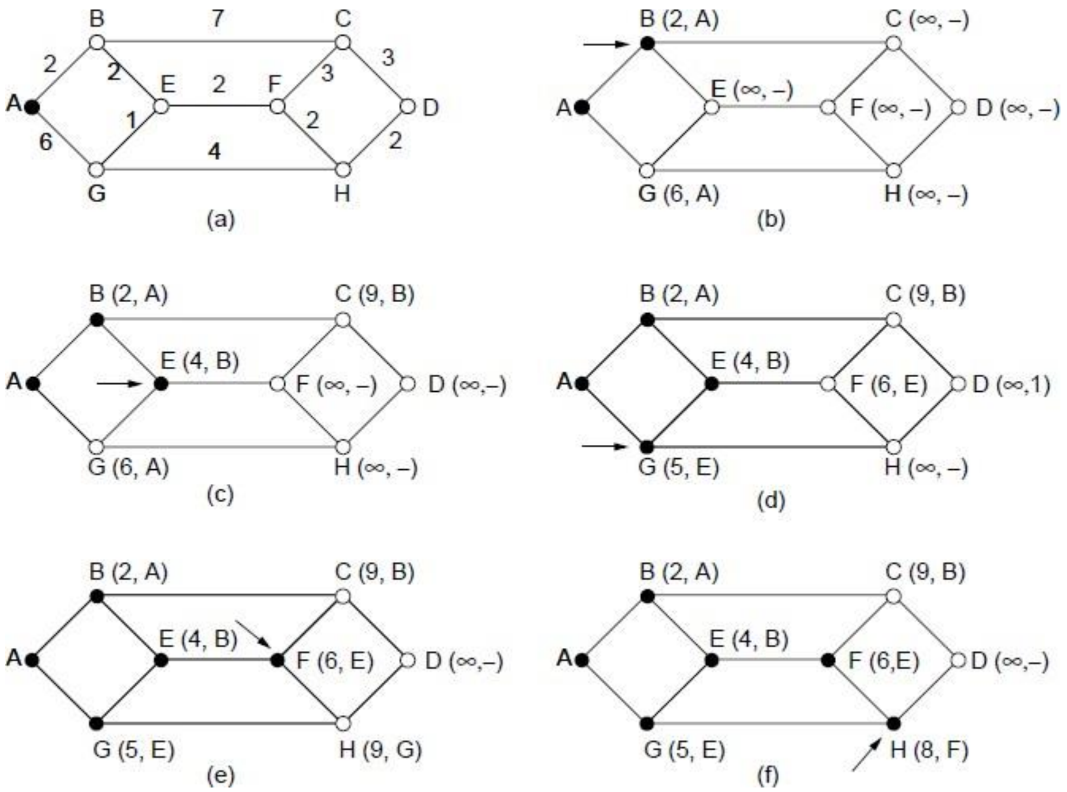
Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to **Dijkstra (1959).** Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

To illustrate how the labeling algorithm works, look at the weighted, undirected graph of **Fig.(a),** where the weights represent, for example, distance. We want to find the **shortest path from A to D**. We start out by marking node **A as permanent,** indicated by a filled-in circle. Then we examine, in turn, each of the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the **smallest label permanent**, as shown in **Fig.(b).** This one becomes the **new working node**.

We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled. The entire graph is searched for the tentatively-labeled node with the smallest value. This node is made permanent and becomes the working node for the next round.

Look at **Fig.(c)**. At that point we have just made E permanent. Suppose that there were a shorter path than ABE, say AXYZE. There are two possibilities: either node Z has already been made permanent, or it has not been. If it has, then E has already been probed (on the round following the one when Z was made permanent), so the AXYZE path has not escaped our attention and thus cannot be a shorter path.

Now consider the case where Z is still tentatively labeled. Either the label at Z is greater than or equal to that at E, in which case AXYZE cannot be a shorter path than ABE, or it is less than that of E, in which case Z and not E will become permanent first, allowing E to be probed from Z.



4.2.3 Flooding:

Another static algorithm is **flooding**, in which **every incoming packet is sent out on every outgoing line except the one it arrived on**. Flooding obviously generates vast numbers of **duplicate packets**, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a **hop counter** contained in the header of each packet, which is **decremented at each hop**, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.

An alternative technique for damming the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. Achieve this goal is to have the source router put **a sequence number** in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen.

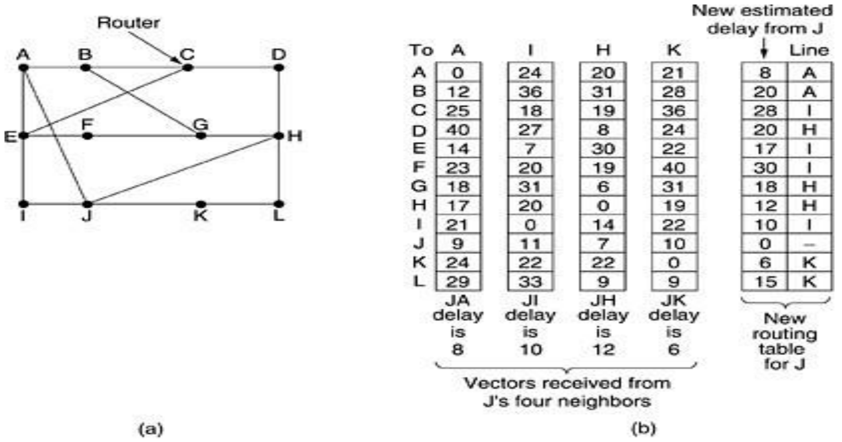
4.2.4 Intra- and Interdomain Routing:

Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems. An **autonomous system (AS)** is a group of networks and routers under the authority of a single administration. Routing inside an autonomous system is referred to as **intradomain** routing. Routing between autonomous systems is referred to as **interdomain** routing.

4.2.5 Distance Vector Routing:

- **Distance vector routing** algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to get there.
- These tables are updated by exchanging information with the neighbors.
- The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm and the **Ford-Fulkerson** algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962).

- It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

***(a) A subnet. (b) Input from* A, I, H, K, *and the new routing table for* J.**

- Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbours of router *J*.
- *A* claims to have a 12-msec delay to *B*, a 25-msec delay to *C*, a 40-msec delay to *D*, etc. Suppose that *J* has measured or estimated its delay to its neighbours, *A*, *I*, *H*, and *K* as 8, 10, 12, and 6 msec, respectively.

Each node constructs a one-dimensional array containing the "distances"(costs) to all other nodes and distributes that vector to its immediate neighbors.

1. The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.
2. A link that is down is assigned an infinite cost.

Example.

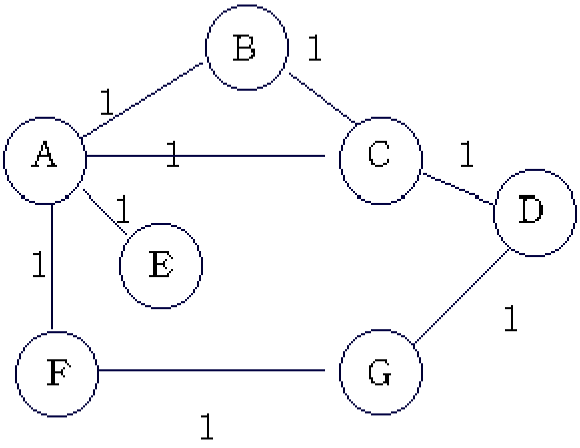


Table 1. Initial distances stored at each node(global view).

Information	Distance to Reach Node						
Stored at Node	**A**	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	∞	1	1	∞

B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

We can represent each node's knowledge about the distances to all other nodes as a table like the one given in Table 1.

Note that each node only knows the information in one row of the table.

1. Every node sends a message to its directly connected neighbors containing its personal list of distance. (for example, **A** sends its information to its neighbors **B,C,E**, and **F**.)
2. If any of the recipients of the information from **A** find that **A** is advertising a path shorter than the one they currently know about, they update their list to give the new path length and note that they should send packets for that destination through **A**. (node **B** learns from **A** that node **E** can be reached at a cost of 1; **B** also knows it can reach **A** at a cost of 1, so it adds these to get the cost of reaching **E** by means of **A**. **B** records that it can reach **E** at a cost of 2 by going through **A**.)
3. After every node has exchanged a few updates with its directly connected neighbors, all nodes will know the least-cost path to all the other nodes.
4. In addition to updating their list of distances when they receive updates, the nodes need to keep track of which node told them about the path that they used to calculate the cost, so that they can create their forwarding table. (for example, **B** knows that it was **A** who said " I can reach **E** in one hop" and so **B** puts an entry in its table that says " To reach **E**, use the link to **A**.)

Table 2. final distances stored at each node (global view).

Information	**Distance to Reach Node**						
Stored at Node	**A**	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	▯	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	▯	3	2	1	3	1	0

In practice, each node's forwarding table consists of a set of triples of the form:

(Destination, Cost, NextHop).

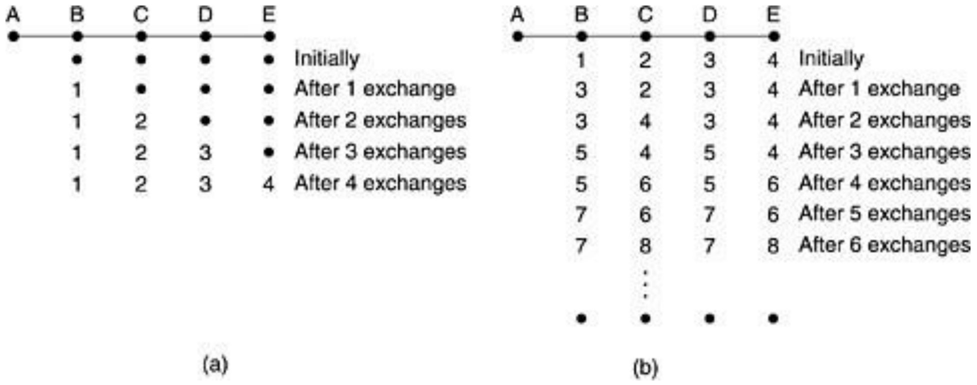
For example, Table 3 shows the complete routing table maintained at node B for the network in figure1.

Table 3. Routing table maintained at node B.

Destination	Cost	NextHop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

THE COUNT-TO-INFINITY PROBLEM

The count-to-infinity problem.



A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)

A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

(b)

- Consider the five-node (linear) subnet of above fig, where the delay metric is the number of hops. Suppose *A* is down initially and all the other routers know this. In other words, they have all recorded the delay to *A* as infinity.
- Now let us consider the situation of Fig (b), in which all the lines and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4, respectively. Suddenly *A* goes down, or alternatively, the line between *A* and *B* is cut, which is effectively the same thing from *B*'s point of view.

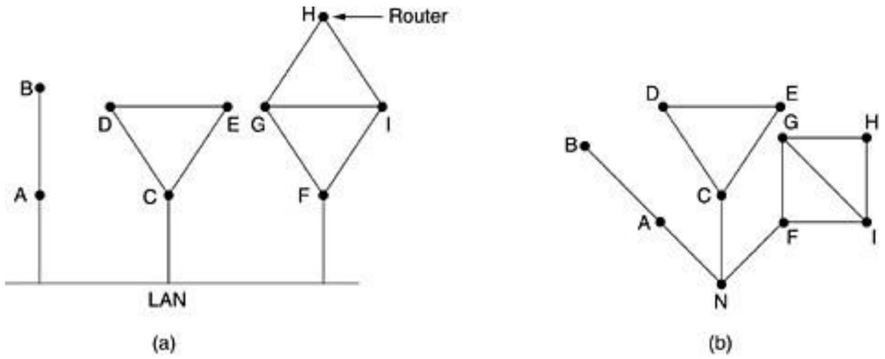
4.2.6 LINK STATE ROUTING:

The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router

Learning about the Neighbours

When a router is booted, its first task is to learn who its neighbours are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply telling who it is.



(a) Nine routers and a LAN. (b) A graph model of (a).

Measuring Line Cost

- The link state routing algorithm requires each router to know, or at least have a reasonable estimate of, the delay to each of its neighbors. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately.
- By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.
- For even better results, the test can be conducted several times, and the average used. Of course, this method implicitly assumes the delays are symmetric, which may not always be the case.

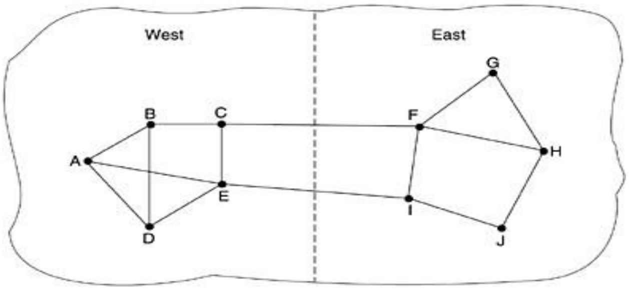
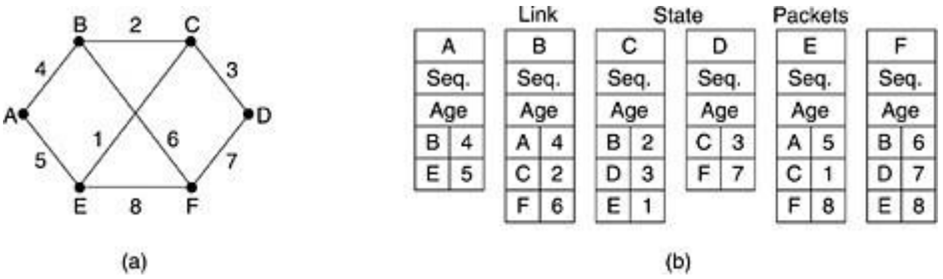


Figure: A subnet in which the East and West parts are connected by two lines.

- Unfortunately, there is also an argument against including the load in the delay calculation. Consider the subnet of above Fig. which is divided into two parts, East and West, connected by two lines, *CF* and *EI*.

Building Link State Packets

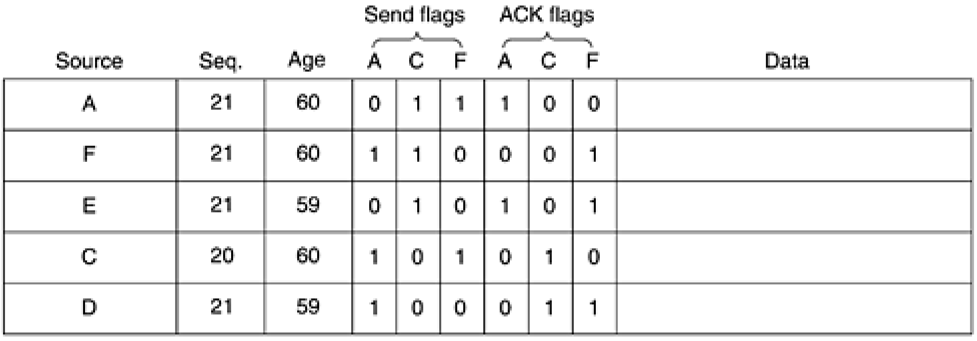


A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

(a) A subnet. (b) The link state packets for this subnet.

- Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data.
- The packet starts with the identity of the sender, followed by a sequence number and age (to be described later), and a list of neighbours.
- For each neighbour, the delay to that neighbour is given.
- An example subnet is given in Fig.(a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig. (b).

Distributing the Link State Packets



			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

The packet buffer for router **B.**

- In above Fig. the link state packet from *A* arrives directly, so it must be sent to *C* and *F* and acknowledged to *A*, as indicated by the flag bits.
- Similarly, the packet from *F* has to be forwarded to *A* and *C* and acknowledged to *F*.

Computing the New Routes:

In link state routing, once a router has collected all link state packets representing the entire network graph, it can construct a comprehensive view of the network, considering links in both directions with potential different costs. Dijkstra's algorithm is then employed locally to compute the shortest paths to all destinations, informing the router of the preferred link for each destination, which is added to the routing tables.

Compared to distance vector routing, link state routing demands more memory and computation, with memory needs proportional to k*n (n = number of routers, k = average neighbors), potentially exceeding the size of a routing table. However, link state routing offers faster convergence, making it practical for many scenarios. Actual networks often use link state routing protocols like IS-IS (Intermediate System-Intermediate System) and OSPF (Open Shortest Path First).

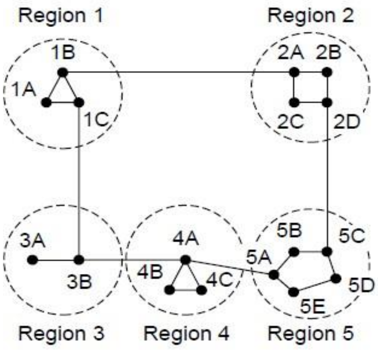
4.2.7 Hierarchical Routing:

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them.

When hierarchical routing is used, the routers are divided into what we will call **regions**, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.

For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the **regions** into **clusters**, the **clusters** into **zones**, the **zones** into **groups**, and so on, until we run out of names for aggregations.

Below Fig. (a) Gives a quantitative example of routing in a two-level hierarchy with five regions. The full routing table for router 1A has 17 entries, as shown in Fig. (b). When routing is done hierarchically, as in Fig. (c), there are entries for all the local routers as before, but all other regions have been condensed into a single router, so all traffic for region 2 goes via the 1B -2A line, but the rest of the remote traffic goes via the 1C -3B line. Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

4.3 CONGESTION CONTROL ALGORITHMS

- When too many packets are present in (a part of) the subnet, performance degrades. This situation is called **congestion**.
- Below Figure depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent.
- However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered.

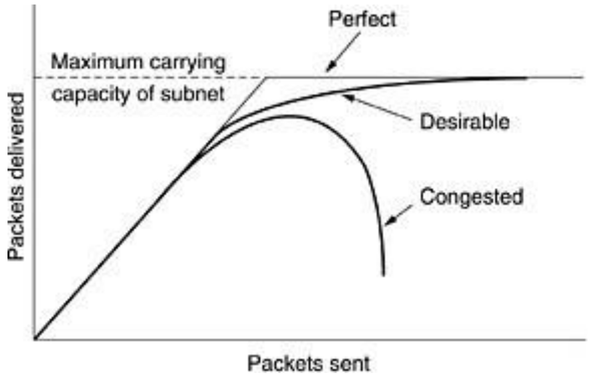


Fig:. When too much traffic is offered, congestion sets in and performance degrades sharply.

- Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up.
- If there is insufficient memory to hold all of them, packets will be lost.
- Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queuing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion.

4.3.1 APPROACHES TO CONGESTION CONTROL

- Many problems in complex systems, such as computer networks, can be viewed from a control theory point of view. This approach leads to dividing all solutions into two groups: open loop and closed loop.

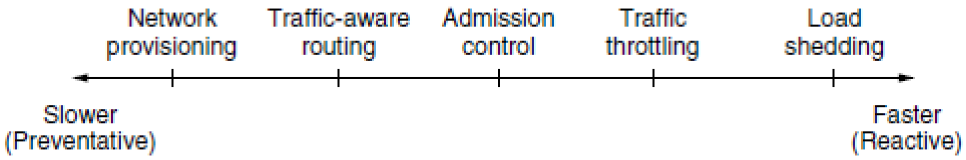


Fig: Timescales Of Approaches To Congestion Control

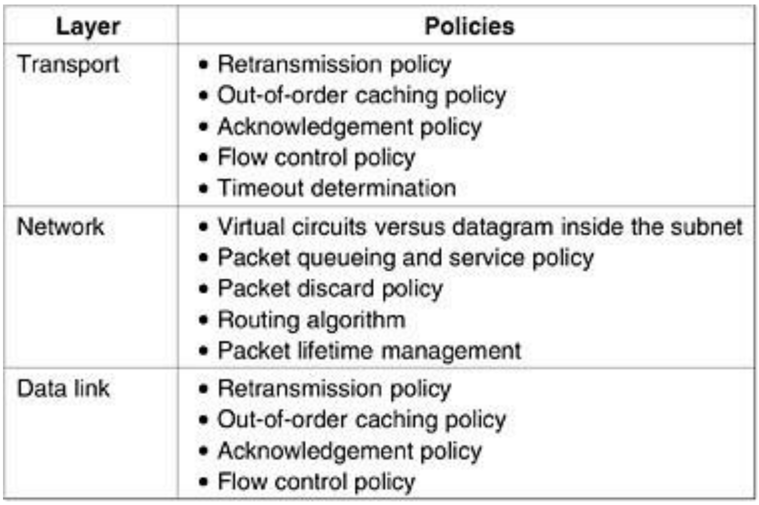
- Open loop solutions attempt to solve the problem by good design.
- Tools for doing open-loop control include deciding when to accept new traffic, deciding when to discard packets and which ones, and making scheduling decisions at various points in the network.
- Closed loop solutions are based on the concept of a feedback loop.
- This approach has three parts when applied to congestion control:
  1. Monitor the system to detect when and where congestion occurs.
  2. Pass this information to places where action can be taken.
  3. Adjust system operation to correct the problem.
- A variety of metrics can be used to monitor the subnet for congestion. Chief among these are the percentage of all packets discarded for lack of buffer space, the average queue lengths, the number of packets that time out and are retransmitted, the average packet delay,

and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion.

- The second step in the feedback loop is to transfer the information about the congestion from the point where it is detected to the point where something can be done about it.
- In all feedback schemes, the hope is that knowledge of congestion will cause the hosts to take appropriate action to reduce the congestion.
- The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. Two solutions come to mind: increase the resources or decrease the load.

4.3.2 CONGESTION PREVENTION POLICIES

The methods to control congestion by looking at open loop systems. These systems are designed to minimize congestion in the first place, rather than letting it happen and reacting after the fact. They try to achieve their goal by using appropriate policies at various levels. In Fig. 5-26 we see different data link, network, and transport policies that can affect congestion (Jain, 1990).



Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Figure 5-26. Policies that affect congestion.

The **data link layer Policies.**

- The **retransmission policy** is concerned with how fast a sender times out and what it transmits upon timeout. A jumpy sender that times out quickly and retransmits all outstanding packets using go back n will put a heavier load on the system than will a leisurely sender that uses selective repeat.
- Closely related to this is the **buffering policy**. If receivers routinely discard all out-of- order packets, these packets will have to be transmitted again later, creating extra load. With respect to congestion control, selective repeat is clearly better than go back n.
- **Acknowledgement policy** also affects congestion. If each packet is acknowledged immediately, the acknowledgement packets generate extra traffic. However, if acknowledgements are saved up to piggyback onto reverse traffic, extra timeouts and

retransmissions may result. A tight flow control scheme (e.g., a small window) reduces the data rate and thus helps fight congestion.

The **network layer Policies.**

- The choice between using **virtual circuits and using datagrams** affects congestion since many congestion control algorithms work only with virtual-circuit subnets.
- **Packet queueing and service policy** relates to whether routers have one queue per input line, one queue per output line, or both. It also relates to the order in which packets are processed (e.g., round robin or priority based).
- **Discard policy** is the rule telling which packet is dropped when there is no space.
- A good **routing algorithm** can help avoid congestion by spreading the traffic over all the lines, whereas a bad one can send too much traffic over already congested lines.
- **Packet lifetime management** deals with how long a packet may live before being discarded. If it is too long, lost packets may clog up the works for a long time, but if it is too short, packets may sometimes time out before reaching their destination, thus inducing retransmissions.

The **transport layer Policies**,

- The **same issues occur as in the data link layer**, but in addition, determining the **timeout interval** is harder because the transit time across the network is less predictable than the transit time over a wire between two routers. If the timeout interval is too short, extra packets will be sent unnecessarily. If it is too long, congestion will be reduced but the response time will suffer whenever a packet is lost.

4.3.3 ADMISSION CONTROL

- One technique that is widely used to keep congestion that has already started from getting worse is **admission control**.
- Once congestion has been signaled, no more virtual circuits are set up until the problem has gone away.
- An alternative approach is to allow new virtual circuits but carefully route all new virtual circuits around problem areas. For example, consider the subnet of Fig. 5-27(a), in which two routers are congested, as indicated.

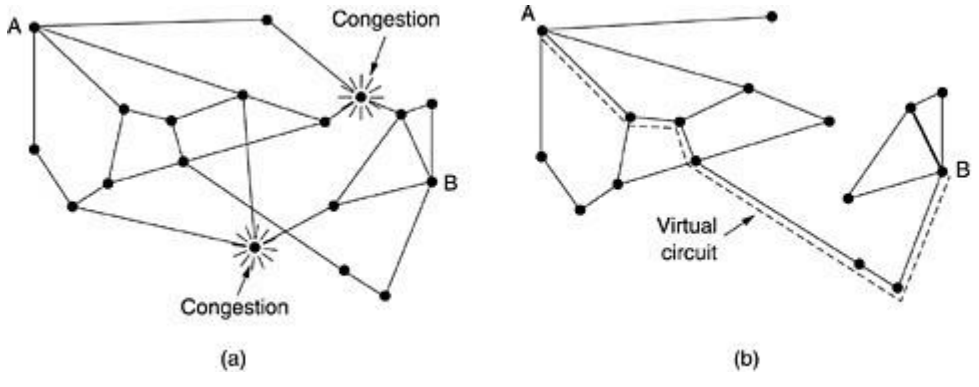


Figure 5-27. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion. A virtual circuit from* A *to* B *is also shown.

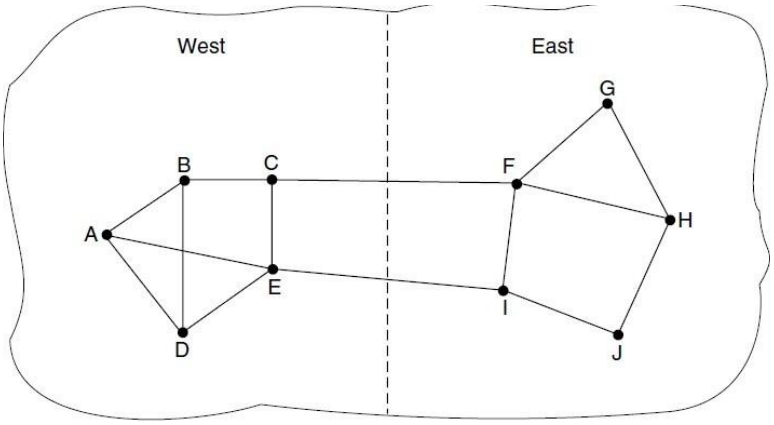
Suppose that a host attached to router *A* wants to set up a connection to a host attached to router *B*. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the subnet as shown in Fig. 5-27(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

4.3.4 TRAFFIC AWARE ROUTING

These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion.

The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal.

Consider the network of Fig. 5-23, which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queueing delay in the weight used for the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, loading this link. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.



If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multipath routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge.

4.3.5 TRAFFIC THROTTLING

- Each router can easily monitor the utilization of its output lines and other resources. For example, it can associate with each line a real variable, *u*, whose value, between 0.0 and 1.0, reflects the recent utilization of that line. To maintain a good estimate of *u*, a sample of the instantaneous line utilization, *f* (either 0 or 1), can be made periodically and *u* updated according to

$$u_{\text{new}} = au_{\text{old}} + (1 - a)f$$

where the constant *a* determines how fast the router forgets recent history.

Whenever *u* moves above the threshold, the output line enters a "warning" state. Each newly-arriving packet is checked to see if its output line is in warning state. If it is, some action is taken. The action taken can be one of several alternatives, which we will now discuss.

4.3.6 CHOKE PACKETS

- In this approach, the router sends a **choke packet** back to the source host, giving it the destination found in the packet.
- The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and is then forwarded in the usual way.
- When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by *X* percent. Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval. After that period has expired, the host listens for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again. If no choke packets arrive during the listening period, the host may increase the flow again.
- The feedback implicit in this protocol can help prevent congestion yet not throttle any flow unless trouble occurs.
- Hosts can reduce traffic by adjusting their policy parameters.
- Increases are done in smaller increments to prevent congestion from reoccurring quickly.
- Routers can maintain several thresholds. Depending on which threshold has been crossed, the choke packet can contain a mild warning, a stern warning, or an ultimatum.

4.3.7 HOP-BY-HOP BACK PRESSURE

- At high speeds or over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow.

Consider, for example, a host in San Francisco (router *A* in Fig. 5-28) that is sending trafficto a host in New York (router *D* in Fig. 5-28) at 155 Mbps. If the New York host begins to run out of buffers, it will take about 30 msec for a choke packet to get back to San Francisco to tellit to slow down. The choke packet propagation is shown as the second, third, and fourth steps in Fig. 5-28(a). In those 30 msec, another 4.6 megabits will have been sent. Even if the host in San Francisco completely shuts down immediately, the 4.6 megabits in the pipe will continue to pour in and have to be dealt with. Only in the seventh diagram in Fig. 5- 28(a) will the New York router notice a slower flow.

An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. 5-28(b). Here, as soon as the choke packet reaches *F*, *F* is required to reduce the flow to *D*. Doing so will require *F* to devote more buffers to the flow, since the source is still sending away at full blast, but it gives *D* immediate relief, like a headache remedy in a television commercial. In the next step, the choke packet reaches *E*, which tells *E* to reduce the flow to *F*. This action puts a greater demand on *E*'s buffers but gives *F* immediate relief. Finally, the choke packet reaches *A* and the flow genuinely slows down.

The net effect of this hop-by-hop scheme is to provide quick relief at the point of congestion at the price of using up more buffers upstream. In this way, congestion can be nipped in the bud without losing any packets.

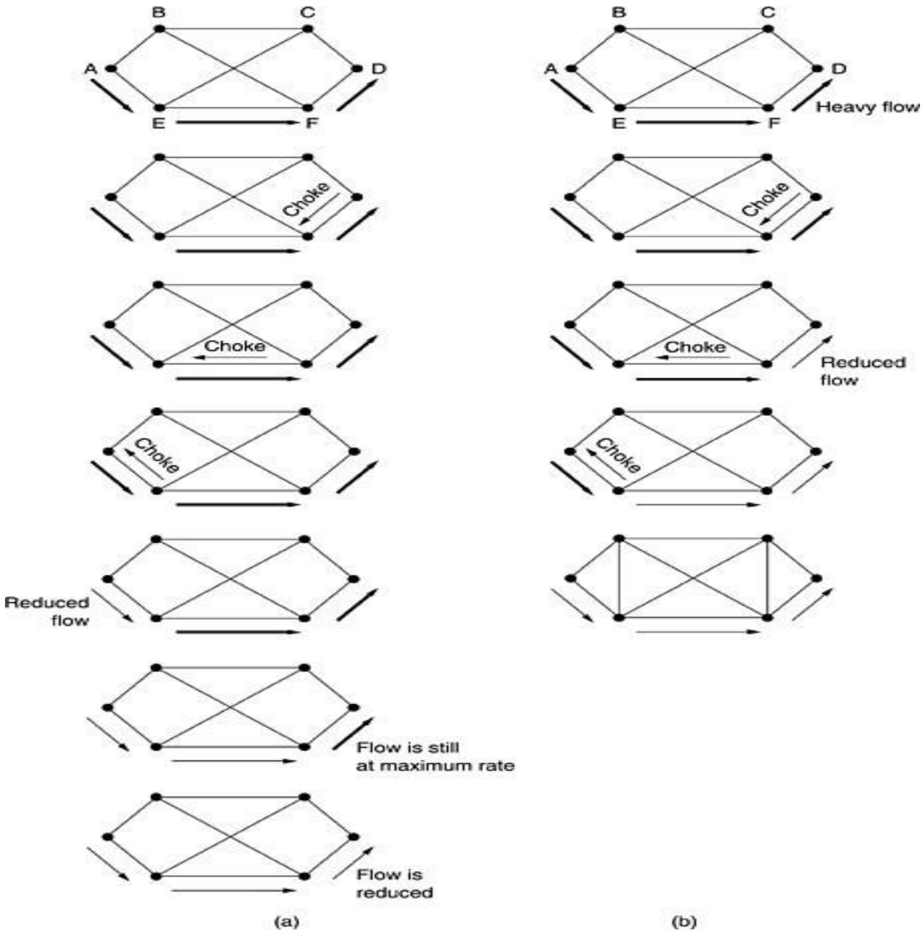


Figure 5-28. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.

4.3.8 LOAD SHEDDING

- When none of the above methods make the congestion disappear, routers can bring out the heavy artillery: load shedding.
- **Load shedding** is a fancy way of saying that when routers are being in undated by packets that they cannot handle, they just throw them away.
- A router drowning in packets can just pick packets at random to drop, but usually it can do better than that.
- Which packet to discard may depend on the applications running.
- To implement an intelligent discard policy, applications must mark their packets in priority classes to indicate how important they are. If they do this, then when packets have to be discarded, routers can first drop packets from the lowest class, then the next lowest class, and so on.

4.3.9 RANDOM EARLY DETECTION

- It is well known that dealing with congestion after it is first detected is more effective than letting it gum up the works and then trying to deal with it. This observation leads to the idea of discarding packets before all the buffer space is really exhausted. A popular algorithm for doing this is called **RED** (**Random Early Detection)**.
- In some transport protocols (including TCP), the response to lost packets is for the source to slow down. The reasoning behind this logic is that TCP was designed for wired networks and wired networks are very reliable, so lost packets are mostly due to buffer overruns

rather than transmission errors. This fact can be exploited to help **reduce congestion**.

- By having routers drop packets before the situation has become hopeless (hence the "early" in the name), the idea is that there is time for action to be taken before it is too late. To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some line exceeds a threshold, the line is said to be congested and action is taken.

4.4 QUALITY OF SERVICES

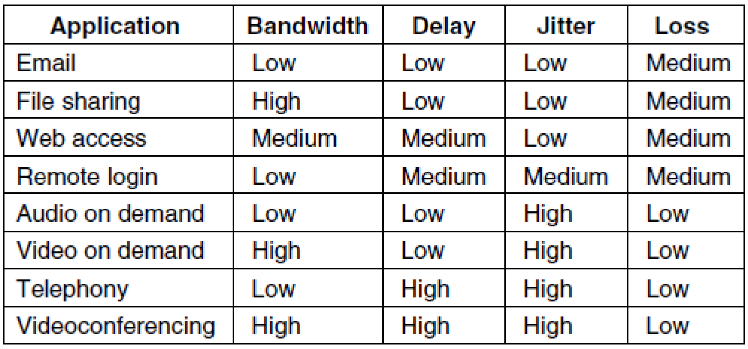
- ➢ An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it.
- ➢ The name for this solution is overprovisioning.
- ➢ The trouble with this solution is that it is expensive.
- ➢ With quality-of-service mechanisms, the network can honor the performance guarantees that it makes even when traffic spikes, at the cost of turning down some requests.

- ➢ Four issues must be addressed to ensure quality of service:
  1. What applications need from the network.
  2. How to regulate the traffic that enters the network.
  3. How to reserve resources at routers to guarantee performance.
  4. Whether the network can safely accept more traffic**.**

Application Requirements

- ➢ A stream of packets from a source to a destination is called a flow.
- ➢ A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network.
- ➢ The needs of each flow can be characterized by four primary parameters: bandwidth, delay, jitter, and loss. Together, these determine the QoS (Quality of Service) the flow requires.

Several common applications and the stringency of their network requirements are listed below



Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Figure 5-27. Stringency of applications' quality-of-service requirements.

Traffic Shaping

- ➢ **Traffic shaping** is a technique for regulating the average rate and burstiness of a flow of data that enters the network.
- ➢ The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network.

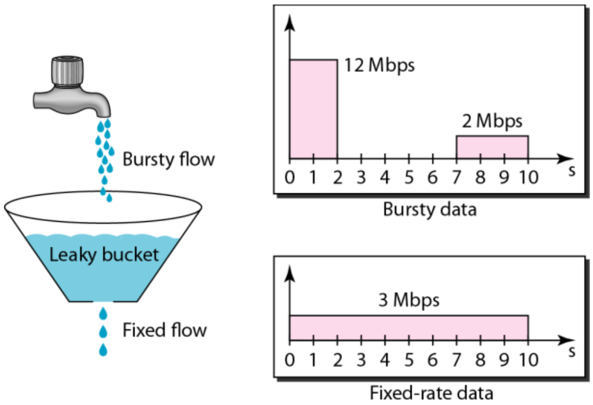
- When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow**.**
- Sometimes this agreement is called an **SLA** (**Service Level Agreement**), especially when it is made over aggregate flows and long periods of time, such as all of the traffic for a given customer**.**
- Traffic shaping reduces congestion and thus helps the network live up to its Promise**.**

Monitoring a traffic flow is called **traffic policing**

Traffic shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Two techniques can shape traffic: leaky bucket and token bucket.

1. Leaky Bucket

If a bucket has a small hole at the bottom, the water leaks from the bucket at a constant rate as long as there is water in the bucket. The rate at which the water leaks does not depend on the rate at which the water is input to the bucket unless the bucket is empty. The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate. Below Figure shows a leaky bucket and its effects.

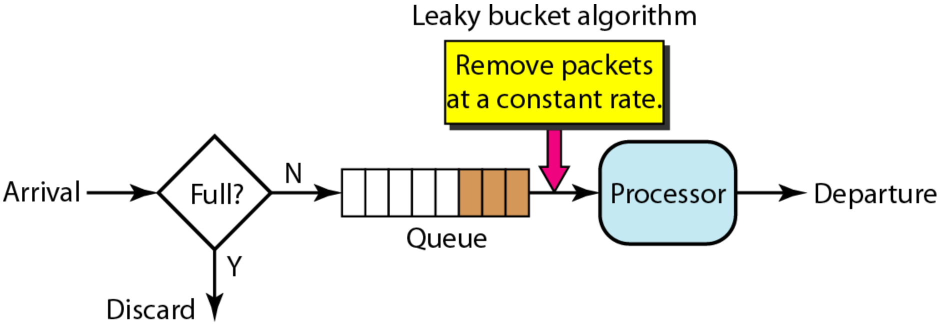


In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure 24.19 the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in l0s. The leaky bucket smooth's the traffic by sending out data at a rate of 3 Mbps during the same 10 s. Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

A simple leaky bucket implementation is shown in below Figure. A FIFO queue holds the packets. If the traffic consists of fixed-size packets, the process removes a fixed number of packets from the queue at each tick of the clock. If the traffic consists of variable-length packets, the fixed output rate must be based on the number of bytes or bits.

The following is an algorithm for variable-length packets:

1. Initialize a counter to *n* at the tick of the clock.
2. If *n* is greater than the size of the packet, send the packet and decrement the counter by the packet size. Repeat this step until *n* is smaller than the packet size.
3. Reset the counter and go to step 1.

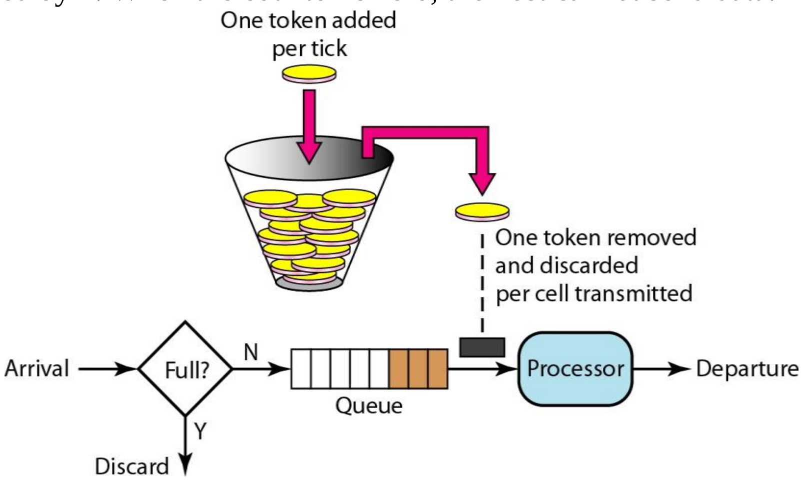


2. Token Bucket

The leaky bucket is very restrictive. It does not credit an idle host. For example, if a host is not sending for a while, its bucket becomes empty. Now if the host has bursty data, the leaky bucket allows only an average rate. The time when the host was idle is not taken into account. On the other hand, the token bucket algorithm allows idle hosts to accumulate credit for the future in the form of tokens. For each tick of the clock, the system sends *n* tokens to the bucket. The system removes one token for every cell (or byte) of data sent. For example, if *n* is 100 and the host is idle for 100 ticks,

the bucket collects 10,000 tokens. Now the host can consume all these tokens in one tick with 10,000 cells, or the host takes 1000 ticks with 10 cells per tick. In other words, the host can send bursty data as long as the bucket is not empty. Below Figure shows the idea.

The token bucket can easily be implemented with a counter. The token is initialized to zero. Each time a token is added, the counter is incremented by 1. Each time a unit of data is sent, the counter is decremented by 1. When the counter is zero, the host cannot send data.



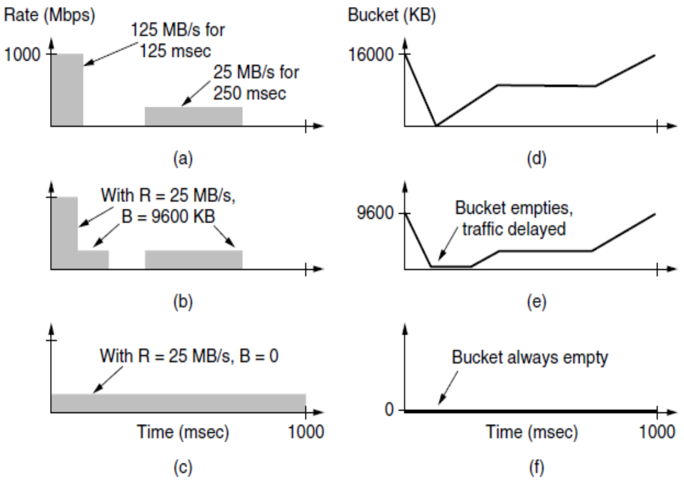


Figure 5-29. (a) Traffic from a host. Output shaped by a token bucket of rate 200 Mbps and capacity (b) 9600 KB and (c) 0 KB. Token bucket level for shaping with rate 200 Mbps and capacity (d) 16,000 KB, (e) 9600 KB, and (f) 0 KB.

Packet Scheduling

- Being able to regulate the shape of the offered traffic is a good start.
- However, to provide a performance guarantee, we must reserve sufficient resources along the route that the packets take through the network.
- Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**.
- Three different kinds of resources can potentially be reserved for different flows:

1. Bandwidth.
2. Buffer space.

3. CPU cycles.

- The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work.
- A second resource that is often in short supply is buffer space. When a packet arrives, it is buffered inside the router until it can be transmitted on the chosen outgoing line.
- The purpose of the buffer is to absorb small bursts of traffic as the flows contend with each other.
- If no buffer is available, the packet has to be discarded since there is no place to put it.
- Finally, CPU cycles may also be a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second.
- While modern routers are able to process most packets quickly, some kinds of packets require greater CPU processing, such as the ICMP packets.
- Packet scheduling algorithms allocate bandwidth and other router resources by determining which of the buffered packets to send on the output line next.
- Each router buffers packets in a queue for each output line until they can be sent, and they are sent in the same order that they arrived. This algorithm is known as **FIFO** (**First-In First-Out**), or equivalently **FCFS** (**First-Come First-Serve**).
- FIFO routers usually drop newly arriving packets when the queue is full.
- Since the newly arrived packet would have been placed at the end of the queue, this behavior is called **tail drop.**
- FIFO scheduling is simple to implement, but it is not suited to providing good quality of service because when there are multiple flows, one flow can easily affect the performance of the other flows**.**
- Many packet scheduling algorithms have been devised that provide stronger isolation between flows and thwart attempts at interference.
- One of the first ones was the **fair queueing** algorithm devised by Nagle.
- The essence of this algorithm is that routers have separate queues, one for each flow for a given output line.
- When the line becomes idle, the router scans the queues round-robin, as shown in Fig. 5-30.
- It then takes the first packet on the next queue. In this way, with *n* hosts competing for the output line, each host gets to send one out of every *n* packets.

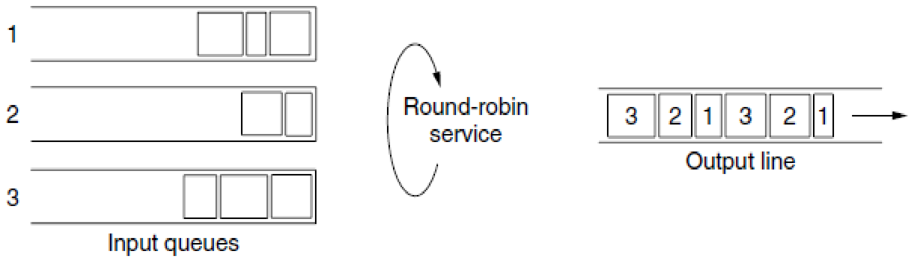
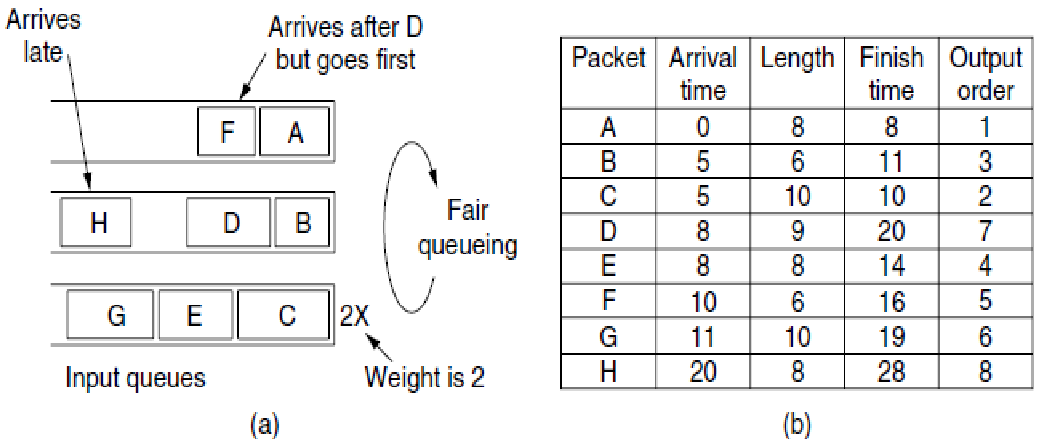


Figure 5-30. Round-robin fair queueing.



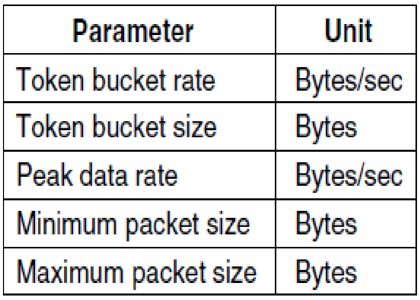
Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

Figure 5-31. (a) Weighted Fair Queueing. (b) Finishing times for the packets.

- This algorithm and an example of finish times for packets arriving in three flows are illustrated in Fig. 5-31.
- If a packet has length *L*, the round at which it will finish is simply *L* rounds after the start time. The start time is either the finish time of the previous packet, or the arrival time of the packet, if the queue is empty when it arrives
- From the table in Fig. 5-32(b), and looking only at the first two packets in the top two queues, packets arrive in the order *A*, *B*, *D*, and *F*. Packet *A* arrives at round 0 and is 8 bytes long, so its finish time is round 8.

Admission Control

- We first saw admission control used to control congestion, which is a performance guarantee, albeit a weak one.
- The guarantees we are considering now are stronger, but the model is the same.
- The user offers a flow with an accompanying QoS requirement to the network.
- The network then decides whether to accept or reject the flow based on its capacity and the commitments it has made to other flows.
- Because many parties may be involved in the flow negotiation (the sender, the receiver, and all the routers along the path between them), flows must be described accurately in terms of specific parameters that can be negotiated.
- A set of such parameters is called a **flow specification**



Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

Figure 5-32. An example flow specification.

- The first two parameters, the *token bucket rate* and *token bucket size*, use a token bucket to give the maximum sustained rate the sender may transmit, averaged over a long time interval, and the largest burst it can send over a short time interval.
- The third parameter, the *peak data rate*, is the maximum transmission rate tolerated, even for brief time intervals.
- The last two parameters specify the minimum and maximum packet sizes, including the transport and

network layer headers.

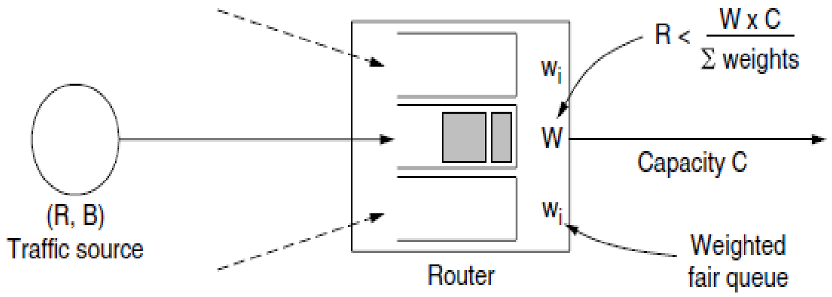


Figure 5-33. Bandwidth and delay guarantees with token buckets and WFQ.

Integrated Services

- A lot of effort into devising an architecture for streaming multimedia. This work resulted in over two dozen RFCs, starting with RFCs 2205–2212.
- The generic name for this work is **integrated services.**
- It was aimed at both unicast and multicast applications.

RSVP—The Resource reservation Protocol

- The main part of the integrated services architecture that is visible to the users of the network is **RSVP**.
- This protocol is used for making the reservations; other protocols are used for sending the data.
- RSVP allows multiple senders to transmit to multiple groups of receivers, permits individual receivers to switch channels freely, and optimizes bandwidth use while at the same time eliminating congestion.
- In its simplest form, the protocol uses multicast routing using spanning trees.
- Each group is assigned a group address. To send to a group, a sender puts the group's address in its packets
- As an example, consider the network of Fig. 5-34(a). Hosts 1 and 2 are multicast senders, and hosts 3, 4, and 5 are multicast receivers. In this example, the senders and receivers are disjoint, but in general, the two sets may overlap. The multicast trees for hosts 1 and 2 are shown in Fig. 5-34(b) and Fig. 5-34(c), respectively.

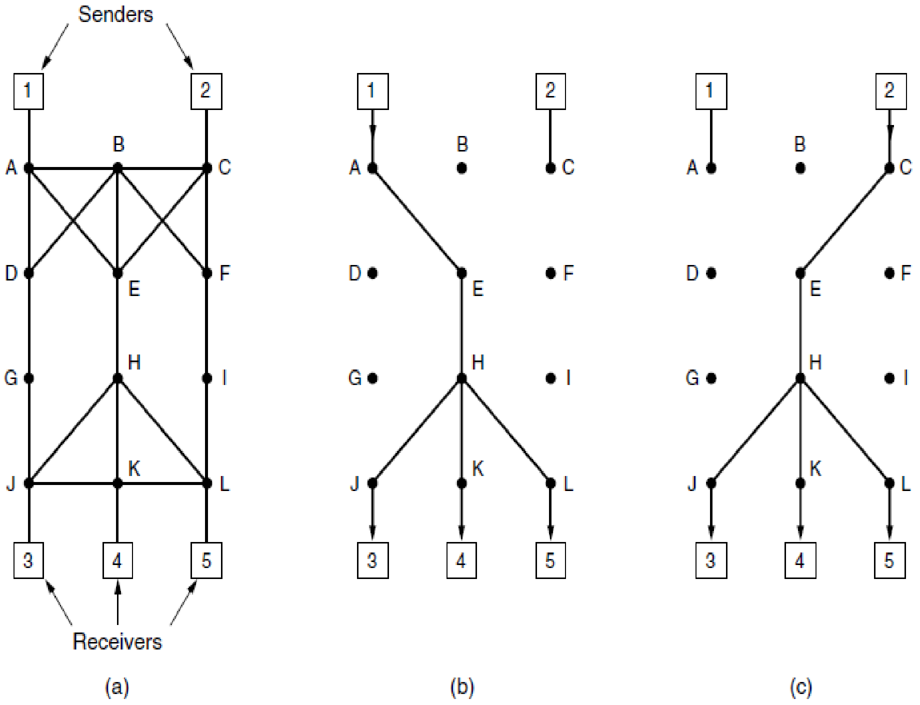


Figure 5-34. (a) A network. (b) The multicast spanning tree for host 1. (c) The multicast spanning tree for host 2.

- To get better reception and eliminate congestion, any of the receivers in a group can send a reservation message up the tree to the sender.
- The message is propagated using the reverse path forwarding algorithm discussed.
- An example of such a reservation is shown in Fig. 5-35(a).
- Here host 3 has requested a channel to host 1. Once it has been established, packets can flow from 1 to 3 without congestion.
- Now consider what happens if host 3 next reserves a channel to the other sender, host 2. A second path is reserved, which is illustrated in Fig. 5-35(b).
- Finally, in Fig. 5-35(c), host 5 decides to watch the program being transmitted by host 1 and also makes a reservation

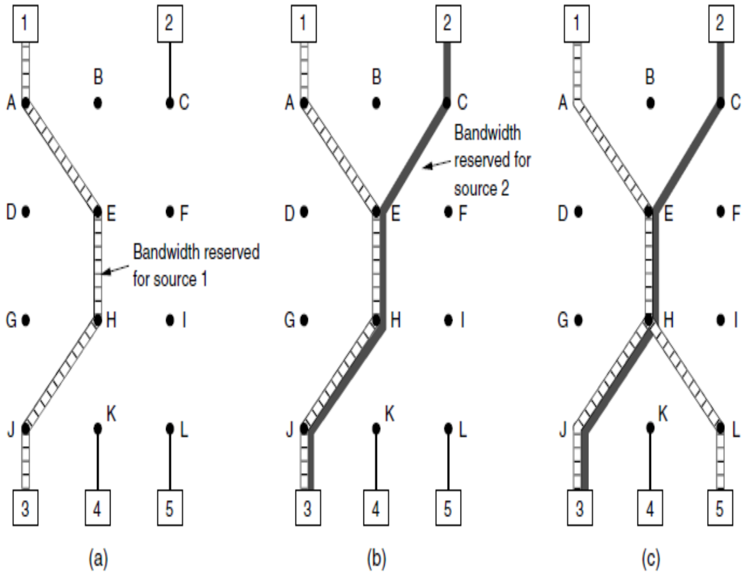

1
2
A
B
C
D
E
F
G
H
I
J
K
L
3
4
5
Bandwidth reserved for source 1
Bandwidth reserved for source 2
(a)
(b)
(c)

Figure 5-35. (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

Differentiated Services

- Differentiated services can be offered by a set of routers forming an administrative domain (e.g., an ISP or a telco).
- The administration defines a set of service classes with corresponding forwarding rules.
- If a customer subscribes to differentiated services, customer packets entering the domain are marked with the class to which they belong.
- This information is carried in the *Differentiated services* field of IPv4 and IPv6 packets. The classes are defined as **per hop behaviors** because they correspond to the treatment the packet will receive at each router, not a guarantee across the network.
- To make the difference between flow-based quality of service and class-based quality of service clearer, consider an example: Internet telephony.
- With a flow based scheme, each telephone call gets its own resources and guarantees. With a class-based scheme, all the telephone calls together get the resources reserved for the class telephony

Expedited Forwarding

- The choice of service classes is up to each operator, but since packets are often forwarded between networks run by different operators, IETF has defined some network-independent service classes.
- The simplest class is **expedited forwarding.**
- The idea behind expedited forwarding is very simple. Two classes of service are available: **regular and expedited.**
- The vast majority of the traffic is expected to be regular, but a limited fraction of the packets are expedited.
- The expedited packets should be able to transit the network as though no other packets were present.
- One way to implement this strategy is as follows. Packets are classified as expedited or regular and marked accordingly.
- This step might be done on the sending host or in the ingress (first) router. The advantage of doing classification on the sending host is that more information is available about which packets belong to which flows.

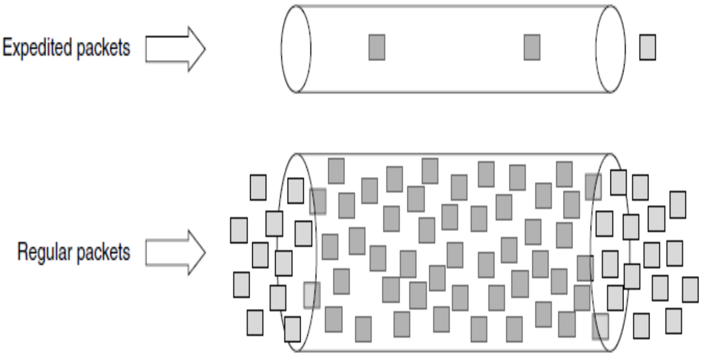


Figure 5-36. Expedited packets experience a traffic-free network.

Assured Forwarding

- A somewhat more elaborate scheme for managing the service classes is called **assured forwarding**.
- Assured forwarding specifies that there shall be four priority classes, each class having its own resources.
- The top three classes might be called gold, silver, and bronze.
- In addition, it defines three discard classes for packets that are experiencing congestion: low, medium, and high. Taken together, these two factors define 12 service classes.
- Figure 5-37 shows one-way packets might be processed under assured forwarding.

- The first step is to classify the packets into one of the four priority classes.
- The next step is to determine the discard class for each packet.
- This is done by passing the packets of each priority class through a traffic policer such as a token bucket.
- Finally, the packets are processed by routers in the network with a packet scheduler that distinguishes the different classes.
- A common choice is to use weighted fair queueing for the four priority classes, with higher classes given higher weights.
- In this way, the higher classes will get most of the bandwidth, but the lower classes will not be starved of bandwidth entirely.

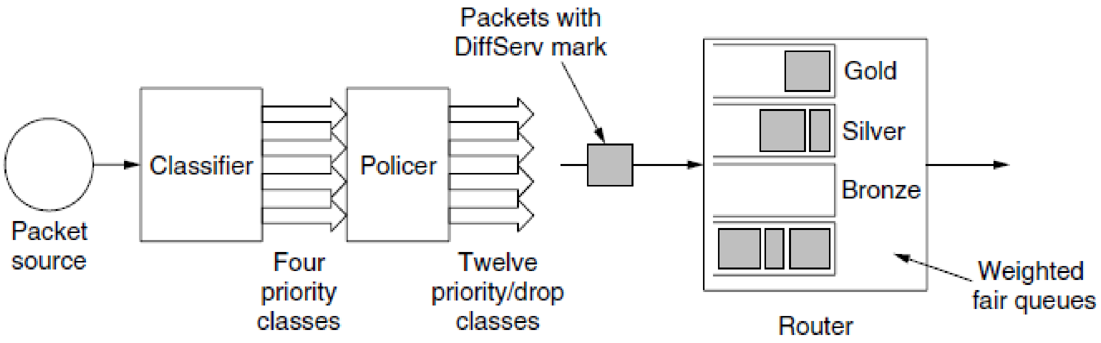


Figure 5-37. A possible implementation of assured forwarding.

THE TRANSPORT LAYER

- The network layer provides end-to-end packet delivery using data- grams or virtual circuits.
- The transport layer builds on the network layer to pro- vide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use.
- It provides the abstractions that applications need to use the network. Without the transport layer, the whole concept of layered proto- cols would make little sense.

6.1 THE TRANSPORT SERVICE

➢ **we will examine two sets of transport layer primitives:**

- First comes a simple (but hypothetical) one to show the basic ideas.
- Then comes the interface commonly used in the Internet.

6.1.1 Services Provided to the Upper Layers

- The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer.
- The software and/or hardware within the transport layer that does the work is called the **transport entity**.
- The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.
- The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 6-1.

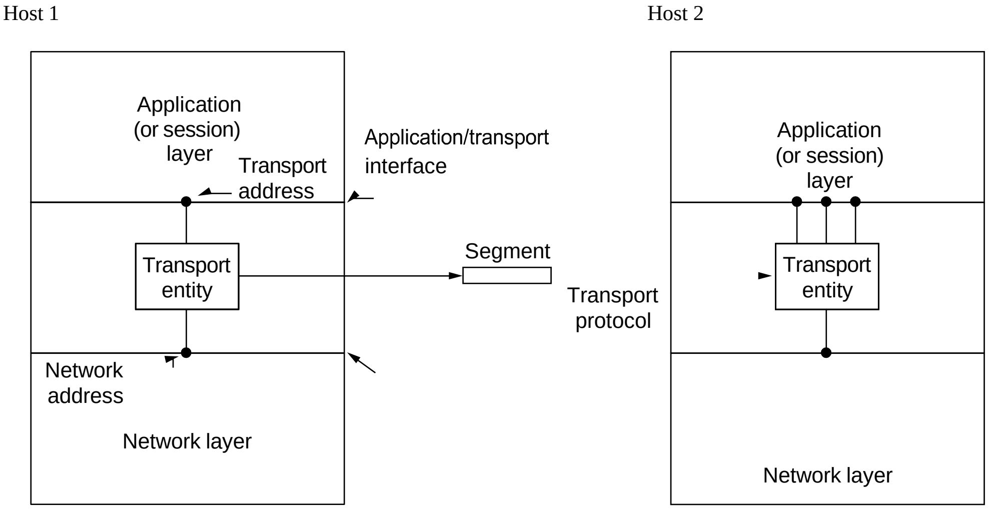


Figure 6-1. The network, transport, and application layers

- Connection-oriented and connectionless, there are also two types of transport service.
- The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.

- The connectionless transport service is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards.

6.1.2 Transport Service Primitives

- To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.
- The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks, warts and all.

- The connection-oriented transport service, in contrast, is reliable. Of course, real networks are not error-free, but that is precisely the purpose of the transport layer—to provide a reliable service on top of an unreliable network.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

Figure 6-2. The primitives for a simple transport service.

- To see how these primitives might be used, consider an application with a ser- ver and a number of remote clients.
- To start with, the server executes a LISTEN primitive, typically by calling a library procedure that makes a system call that blocks the server until a client turns up. When a client wants to talk to the server,it executes a CONNECT primitive.
- The transport entity carries out this primitive by blocking the caller and sending a packet to the server.

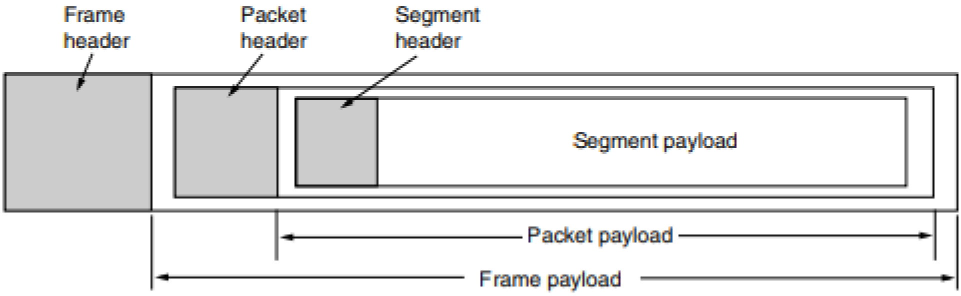


Figure 6-3. Nesting of segments, packets, and frames.

- The term **segment** for messages sent from transport entity to transport entity. TCP, UDP and other Internet protocols use this term.
- Some older protocols used the ungainly name **TPDU** (**Transport Protocol Data Unit**). That term isnot used much any more now but you may see it in older papers and books.
- When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local deliv- ery, passes the contents of the frame payload field up to the network entity.

6.1.3 Berkeley Sockets

- Inspect another set of transport primitives, the socket prim- itives as they are used for TCP.

- Sockets were first released as part of the Berke- ley UNIX 4.2BSD software distribution in 1983.
- They quickly became popular. The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called ‘‘winsock.’

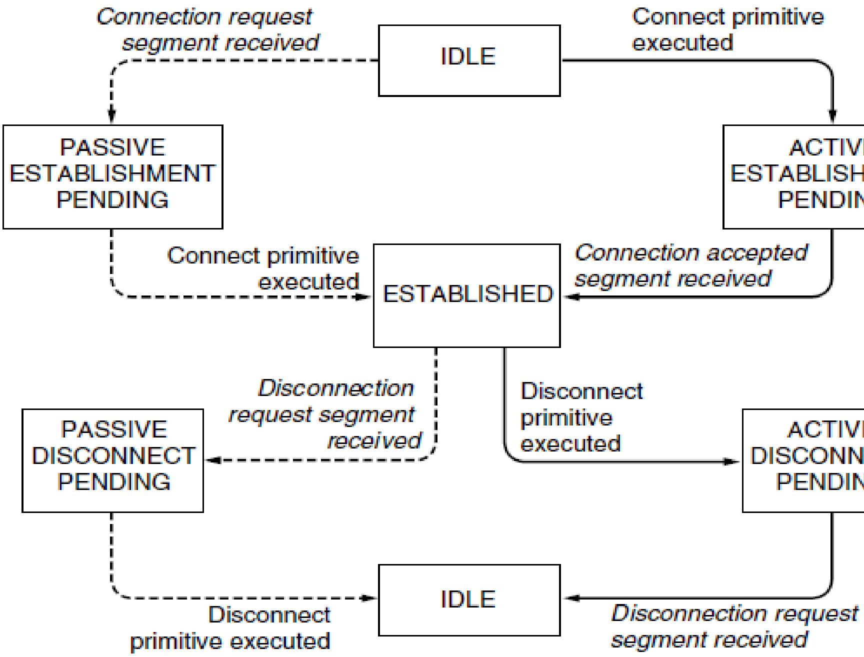

Connection request segment received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted segment received
Disconnection request segment received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request segment received

Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client’s state sequence. The dashed lines show the server’s state sequence.

Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Figure 6-5. The socket primitives for TCP.

The primitives are listed in Fig. 6-5. Roughly speaking, they follow the model of our first example but offer more features and flexibility.

- The first four primitives in the list are executed in that order by servers. The SOCKET primitive creates a new endpoint and allocates table space for it within the transport entity.
- The parameters of the call specify the addressing format tobe used, the type of service desired (e.g., reliable byte stream), and the protocol.
- The socket API is often used with the TCP protocol to provide a connection-oriented service called a **reliable byte stream**, which is simply the reliable bit pipe.
- **DCCP** (**Datagram Congestion Controlled Protocol**) is a version of UDP with congestion control (Kohler et al., 2006). It is up to the tran- sport users to understand what service they are getting.
- Newer protocols and interfaces have been devised that support groups of related streams more effectively and simply for the application.
- Two examples are **SCTP** (**Stream Control Transmission Protocol**) defined in RFC 4960 and **SST** (**Structured Stream Transport**) (Ford, 2007).
- These protocols must change the socket API slightly to get the benefits of groups of relatedstreams, and they also support features such as a mix of connection-oriented and connectionless traffic and even multiple network paths.

6.1.4 An Example of Socket Programming: An Internet File Server

- As an example of the nitty-gritty of how real socket calls are made, consider the client and server code of Fig. 6-6. Here we have a very primitive Internet file server along with an example client that uses it.
- The code has many limitations (discussed below), but in principle the server code can be compiled and run on any UNIX system connected to the Internet.
- It starts out by including some standard headers, the last three of which contain the main Internet-related definitions and data structures.
- Next comes a definition of *SERVER PORT* as 12345. This num- ber was chosen arbitrarily. Any number between 1024 and 65535 will work justas well, as long as it is not in use by some other process; ports below 1023 are re- served for privileged users.
- The next two lines in the server define two constants needed. The first one determines the chunk size in bytes used for the file transfer.

```
/* This page contains a client program that can request a file from the server program
 * on the next page. The server responds by sending the whole file.
 */

#include      <sys/types.h>
#include     <sys/socket.h>
#include      <netinet/in.h>
#include <netdb.h>

#define SERVER  PORT 12345                       /* arbitrary, but client & server must agree */
#define BUF  SIZE¯4096                           /* block transfer size */

int main(int argc, char **argv)

{

  int c, s, bytes;

  char buf[BUF  SIZE];                           /* buffer for incoming file */

  struct hostent *h;                             /* info about server */

  struct sockaddr  in channel;                   /* holds IP address */

  if (argc != 3) fatal("Usage: client server-name file-name");

  h = gethostbyname(argv[1]);                    /* look up host's IP address */if
  (!h) fatal("gethostbyname failed");

  s = socket(PF  INET, SOCK  STREAM, IPPROTO  TCP);
  if (s <0) fatal("socket"); memset(&channel,
  0,  sizeof(channel));  channel.sin   family=
  AF  INET;   _                  _

  memcpy(&channel.sin addr.s addr, h->h addr, h->h length);
  channel.sin  port= htons(SERVER  PORT);


  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));if (c <
  0) fatal("connect failed");

  /* Connection is now established. Send file name including 0 byte at end. */write(s,
  argv[2], strlen(argv[2])+1);

  /* Go get the file and write it to standard output. */while
  (1) {

       bytes = read(s, buf, BUF  SIZE);                /* read from socket */

       if (bytes <= 0) exit(0);                        /* check for end of file */

       write(1, buf, bytes);                           /* write to standard output */

  }

}


fatal(char *string)

{

  printf("%s\n", string);
  exit(1);

}
```

Figure 6-6. Client code using sockets. The server code is on the next page.

```
#include <sys/types.h>                                /*  This is the server code */
#include <sys/fcntl.h>

#include     <sys/socket.h>
#include      <netinet/in.h>
#include <netdb.h>

#define SERVER  PORT 12345                            /* arbitrary, but client & server must agree */
#define BUF  SIZE 4096                                /* block transfer size */

#define QUEUE  SIZE 10

int main(int argc, char *argv[])

{

  int s, b, l, fd, sa, bytes, on = 1;

  char buf[BUF  SIZE];                                /* buffer for outgoing file */
  struct sockaddr  in channel;                       /* holds IP address */

  /* Build address structure to bind to socket. */

  memset(&channel, 0, sizeof(channel));              /* zero channel */
  channel.sin  family = AF  INET;

  channel.sin_addr.s_addr = htonl(INADDR  ANY);
  channel.sin  port = htons(SERVER  PORT);

  /* Passive open. Wait for connection. */

  s = socket(AF  INET, SOCK  STREAM, IPPROTO  TCP);          /* create socket */if
  (s < 0) fatal("socket failed");

  setsockopt(s, SOL  SOCKET, SO  REUSEADDR, (char *) &on, sizeof(on));

  b = bind(s, (struct sockaddr *) &channel, sizeof(channel));if (b
  < 0) fatal("bind failed");

  l = listen(s, QUEUE  SIZE);                        /* specify queue size */if
  (l < 0) fatal("listen failed");

  /* Socket is now set up and bound. Wait for connection and process it. */while (1)
  {

      sa = accept(s, 0, 0);                          /* block for connection request  */if
      (sa < 0) fatal("accept failed");

      read(sa, buf, BUF  SIZE);                      /* read file name from socket */

      /* Get and return the file. */

      fd = open(buf, O  RDONLY);                     /* open the file to be sent back */if
      (fd < 0) fatal("open failed");

      while (1) {

            bytes = read(fd, buf, BUF  SIZE); /* read from file */
            if (bytes <= 0) break;                   /* check for end of file */

            write(sa, buf, bytes);                   /* write bytes to socket */

      }

      close(fd);                                     /* close file */

      close(sa);                                     /* close connection */

  }

}
```

6.2 ELEMENTS OF TRANSPORT PROTOCOLS

- The transport service is implemented by a transport protocol used between the two transport entities.
- However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 6-7.

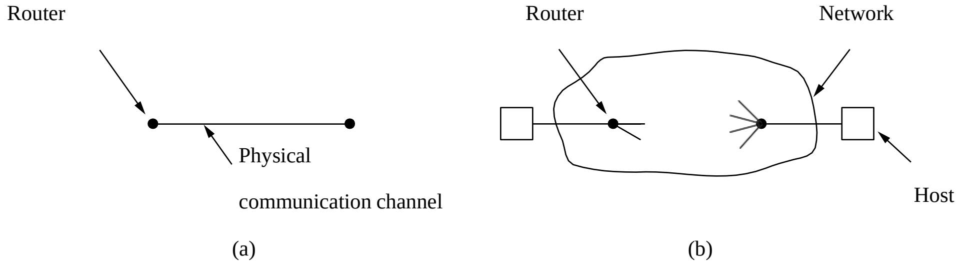


Figure 6-7. (a) Environment of the data link layer. (b) Environment of the transport layer.

- At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.
- For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router.
- Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations.
- When a router sends a packet over a link, it may arrive or be lost, but it cannot bounce around for a while, go into hiding in a far corner of the world, and sudden- ly emerge after other packets that were sent much later.

6.2.1 Addressing

- When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to.

- The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**.
- We will use the generic term **TSAP** (**Transport Service Access Point**) to mean a specific endpoint in the transport layer.
- The analogous endpoints in the network layer (i.e., network layer addresses) are not-surprisingly called **NSAPs** (**Network Service Access Points**).
- Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP.

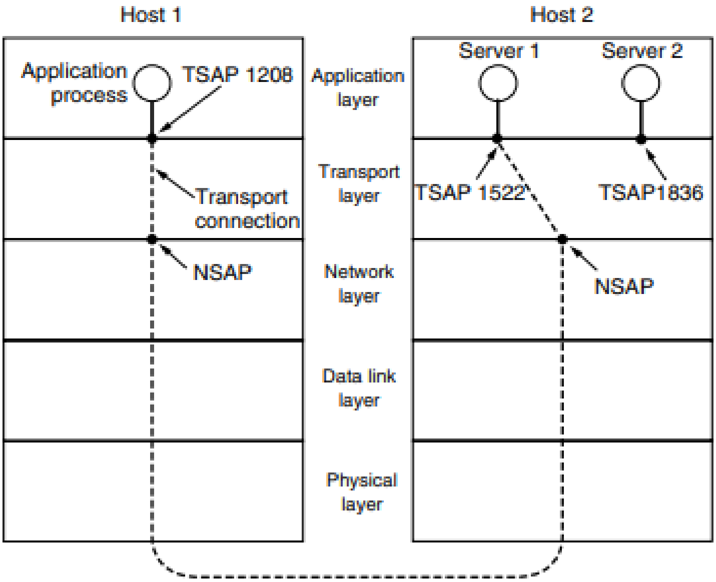


Figure 6-8. TSAPs, NSAPs, and transport connections.

➢ A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.

3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

- To handle this situation, an alternative scheme can be used. In this scheme, there exists a special process called a **portmapper**.
- The user then sends a message specifying the service name, and the portmapper sends back the TSAP address.
- The function of the portmapper is analogous to that of a directory assistance operator in the telephone system—it provides a mapping of names onto numbers.
- It is wasteful to have each of them active and listening to a stable TSAP address all day long. An alternative scheme is shown in Fig. 6-9 in a simplified form. It is known as the **initial connection protocol**.
- Instead of every conceivable server listening at a well-known TSAP, each machine that wishes to offer services to remote users has a special **process server** that acts as a proxy for less heavily used servers.
- This server is called *inetd* on UNIX systems. It listens to a set of ports at the same time, waiting for a connection request. Potential users of a service begin by doing a CONNECT request, specifying the TSAP address of the service they want.

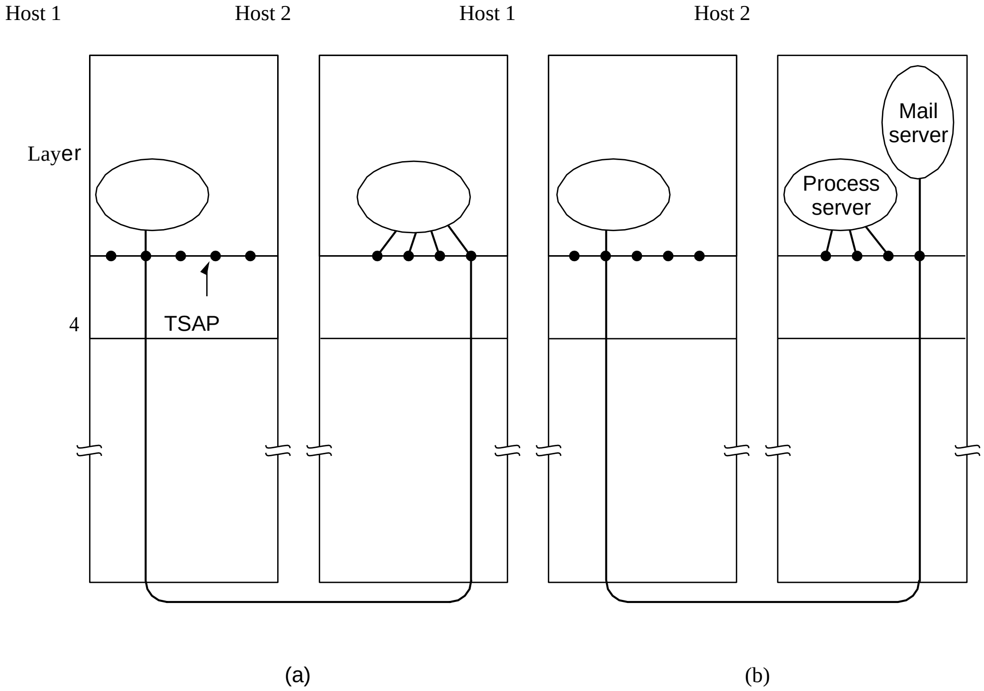


Figure 6-9. How a user process in host 1 establishes a connection with a mailserver in host 2 via a process server.

6.2.2 Connection Establishment

- At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply.
- The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person.

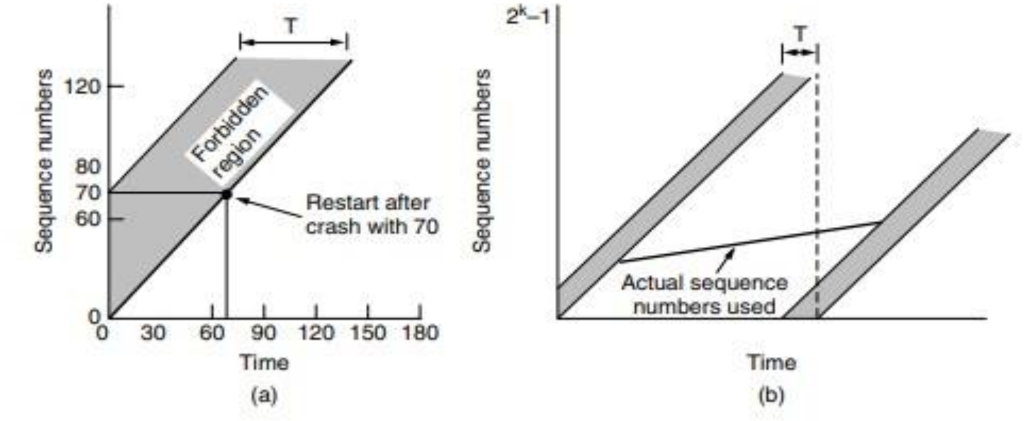


Figure 6-10. (a) Segments may not enter the forbidden region. (b) The resynchronization problem.

➢ Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

- Once both transport entities have agreed on the initial sequence number, any sliding window protocol can be used for data flow control.
- . Within a connection, a timestamp is used to extend the 32-bit sequence number so that it will not wrap within the maximum packet lifetime, even for gigabit-per-second connections.

- This mechanism is a fix to TCP that was needed as it was used on faster and faster links. It is described in RFC 1323 and called **PAWS** (**ProtectionAgainst Wrapped Sequence numbers**).
- Across connections, for the initial se- quence numbers and before PAWS can come into play, TCP originally used the clock-based scheme just described.

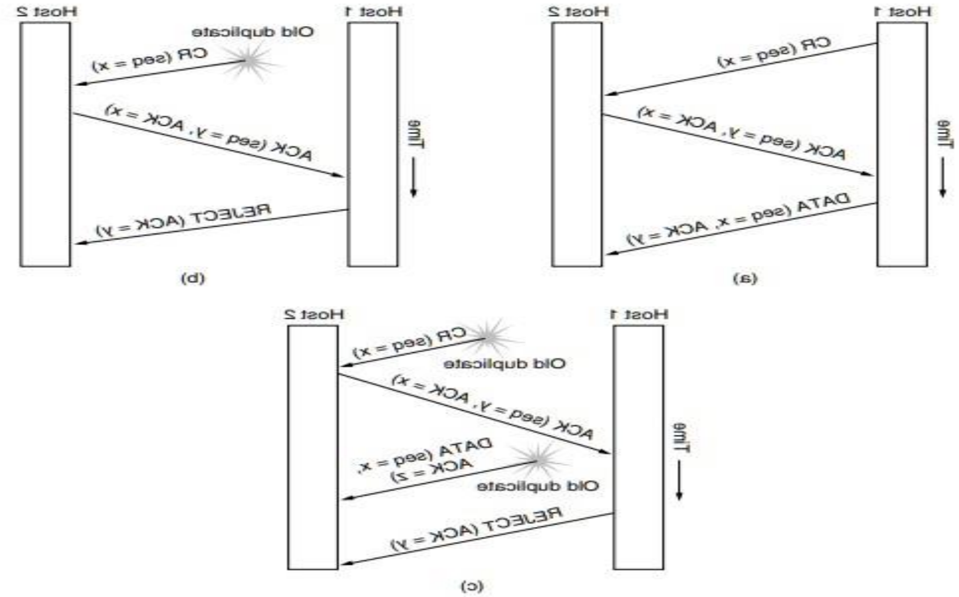


Figure 6-11. Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal opera- tion. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere.

(c) Duplicate CONNECTION REQUEST and duplicate ACK.

- However, it remains important that the initial sequence numbers not repeat for an interval even though they appearrandom to an observer.

6.2.3 Connection Release

- Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here. As we mentioned earlier, there are two styles of terminating a connection: asymmetric release and symmetric release.
- Asymmetric release is the way the telephone system works: when one

party hangs up, the connection is broken. Symmetric release treats the connection as two sep- arate unidirectional connections and requires each one to be released separately.

- Asymmetric release is abrupt and may result in data loss. Consider the scen- ario of Fig. 6-12. After the connection is established, host 1 sends a segment that arrives properly at host 2.

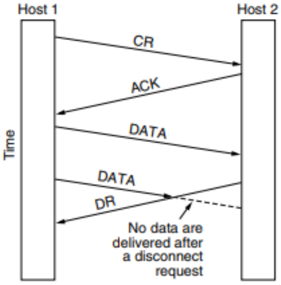


Figure 6-12. Abrupt disconnection with loss of data.

    - Then host 1 sends another segment. Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

- There is a famous prob- lem that illustrates this issue. It is called the **two-army problem**.

- Imagine that a white army is encamped in a valley, as shown in Fig. 6-13. On both of the sur- rounding hillsides are blue armies.

- The white army is larger than either of the blue armies alone, but together the blue armies are larger than the white army

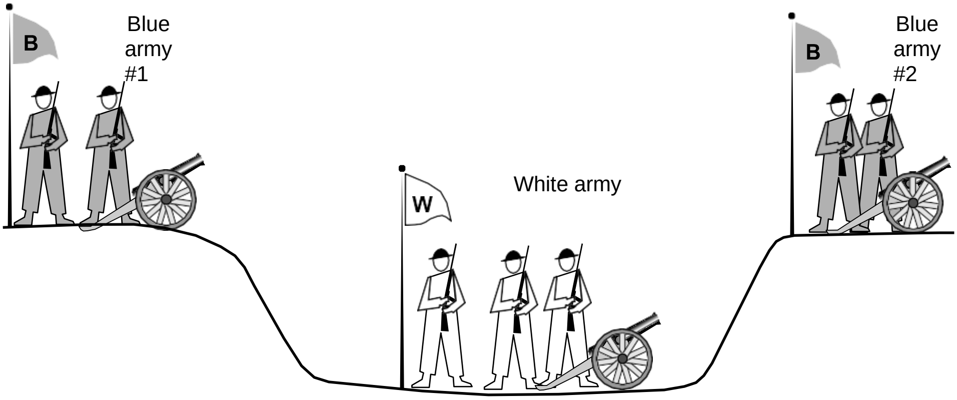


Figure 6-13. The two-army problem

- If either blue army attacks by itself, it will be defeated, but if the two blue armies at- tack simultaneously, they will be victorious.
- The blue armies want to synchronize their attacks. However, their only communication medium is to send messengers on foot down into the valley, where they might be captured and the message lost (i.e., they have to use an unreliable communication channel).

- In Fig. 6-14(a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release. When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost.
- When this DR arrives, the original sender sends back an ACK segment and releases the connection. Finally, when the ACK segment arrives, the receiver also releases the connection.
- . Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection’s owner (the transport user) somehow.
- If the final ACK segment is lost, as shown in Fig. 6-14(b), the situation is saved by the timer. When the timer expires, the connection is released anyway.
- In Fig. 6-14(c), we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.
- Our last scenario, Fig. 6-14(d), is the same as Fig. 6-14(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost seg- ments.

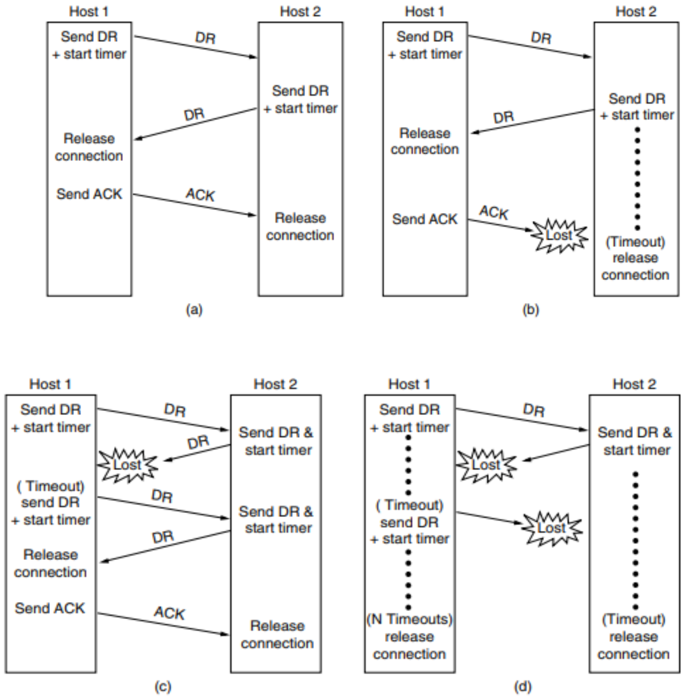


Figure 6-14. Four protocol scenarios for releasing a connection. (a) Normal case of three-way handshake. (b) Final ACK lost. (c) Response lost. (d) Re- sponse lost and subsequent DRs lost.

6.2.4 Error Control and Flow Control

- Error control is ensuring that the data is deliv- ered with the desired level of reliability, usually that all of the data is delivered without any errors.

- Flow control is keeping a fast transmitter from overrunning a slow receiver.

Given that these mechanisms are used on frames at the link layer, it is natural to wonder why they would be used on segments at the transport layer as well.

1 A frame carries an error-detecting code (e.g., a CRC or checksum) that is used to check if the information was correctly received.
2 A frame carries a sequence number to identify itself and is retransmitted by the sender until it receives an acknowledgement of successful receipt from the receiver. This is called **ARQ** (**Automatic Repeat reQuest**).
3 There is a maximum number of frames that the sender will allow to be outstanding at any time, pausing if the receiver is not acknowledg-ing frames quickly enough. If this maximum is one packet the proto- col

is called **stop-and-wait**. Larger windows enable pipelining and improve performance on long, fast links.

4 The **sliding window** protocol combines these features and is also used to support bidirectional data transfer.

- In Fig. 6-15(a). Howev- er, if there is wide variation in segment size, from short requests for Web pages to large packets in peer-to-peer file transfers, a pool of fixed-sized buffers presents problems.
- Another approach to the buffer size problem is to use variable-sized buffers,as in Fig. 6-15(b).
- A third possibility is to dedicate a single large circular buffer per connection, as in Fig. 6-15(c).

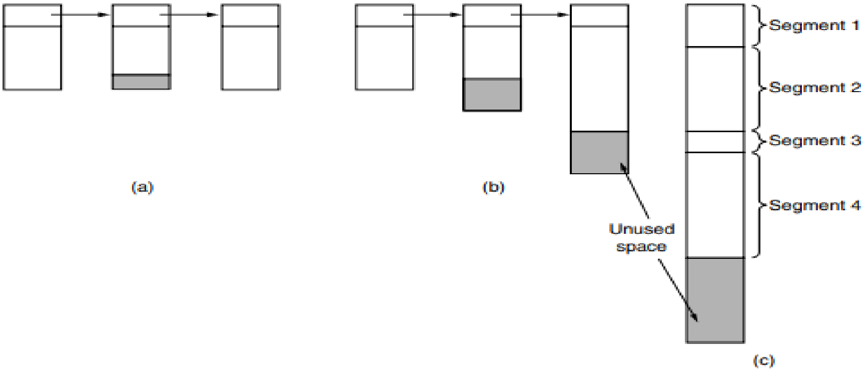


Figure 6-15. (a) Chained fixed-size buffers. (b) Chained variable-sized buffers. (c) One large circular buffer per connection.

- Figure 6-16 shows an example of how dynamic window management might work in a datagram network with 4-bit sequence numbers.
- Initially, *A* wants eight buffers, but it is granted only four of these. It then sends three segments, of which the third is lost.
- Segment 6 acknowledges receipt of all segments up to and including sequence number 1, thus allowing *A* to release those buffers, and furthermore informs *A* that it has permission to send three more segments starting beyond 1 (i.e., segments 2, 3, and 4).
- The next segment from *B* to *A* allocates another buffer and allows *A* to continue. This will happen when *B* has bufferspace, likely because the transport user has accepted more segment data.

	A	Message	B	Comments
1	►	< request 8 buffers>	►	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 onlyA
3	►	<seq = 0, data = m0>	►	has 3 buffers left now
4	►	<seq = 1, data = m1>	►	A has 2 buffers left now
5	►	<seq = 2, data = m2>		
6	←	<ack = 1, buf = 3>		Message lost but A thinks it has 1 leftB
7		<seq = 3, data = m3>		acknowledges 0 and 1, permits 2-4A has
8		<seq = 4, data = m4>		1 buffer left
9		<seq = 2, data = m2>		A has 0 buffers left, and must stopA
10		<ack = 4, buf = 0>		times out and retransmits
11		<ack = 4, buf = 1>		Everything acknowledged, but A still blockedA
12		<ack = 4, buf = 2>		may now send 5
13		<seq = 5, data = m5>		B found a new buffer somewhereA
14		<seq = 6, data = m6>		has 1 buffer left
15		<ack = 6, buf = 0>		A is now blocked againA
16		<ack = 6, buf = 4>		is still blocked Potential
				deadlock

Figure 6-16. Dynamic buffer allocation. The arrows show the direction oftransmission. An ellipsis (...) indicates a lost segment.

- Lookat line 16. *B* has now allocated more buffers to *A*, but the allocation segment was lost. Oops. Since control segments are not sequenced or timed out, *A* is now deadlocked.
- When buffer space no longer limits the maximum flow, another bottleneckwill appear: the carrying capacity of the network.

6.2.5 Multiplexing

- When a segment comes in, some way is needed to tell which process to give it to. This situation, called **multiplexing**, is shown in Fig. 6-17(a).
- If a user needs more bandwidth or more reliability than one of the network paths can pro- vide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in Fig. 6-17(b). This modus operandi is called **inverse multiplexing**.
- **SCTP** (**Stream Control Transmission Protocol**), which can runa connection using multiple network interfaces.

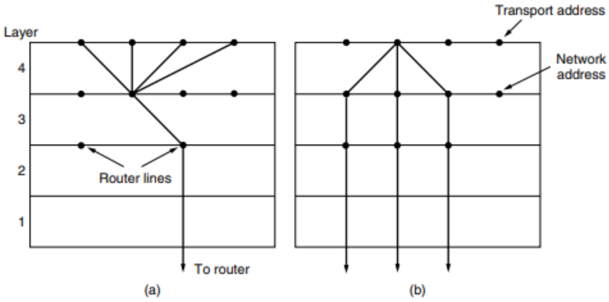
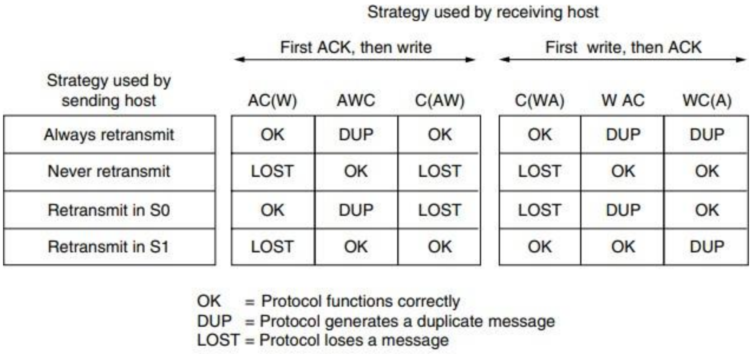


Figure 6-17. (a) Multiplexing. (b) Inverse multiplexing.

6.2.6 Crash Recovery

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads), recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The transport entities expect lost segments all the time and know how to cope with them by using retransmissions.



	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
Strategy used by sending host	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

Figure 6-18. Different combinations of client and server strategies.

6.3 CONGESTION CONTROL

- If the transport entities on many machines send too many packets into the net- work too quickly, the network will become congested, with performance degraded as packets are delayed and lost.
- Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers.
- Congestion oc- curs at routers, so it is detected at the network layer. However, congestion is ulti- mately caused by traffic sent into the network by the transport layer.

- The Internet relies heavily on the transport layer for congestion control, and specific algorithms are built into TCP and other protocols.

6.3.1 Desirable Bandwidth Allocation

- we must specify the state in which a good congestion control algorithm will operate the network
- The goal is more than to simply avoid congestion. It is to find a good al- location of bandwidth to the transport entities that are using the network

Efficiency and Power

- An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, five transport entities should get 20 Mbps each.

- The **goodput** (or rate of useful packets arriving at the receiver) as a function of the offered load. This curve and a matching curve for the delay as a function of the offered load are given in Fig. 6-19.

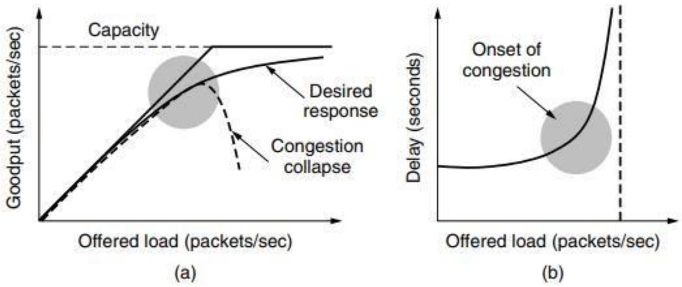


Figure 6-19. (a) Goodput and (b) delay as a function of offered load.

- As the load increases in Fig. 6-19(a) goodput initially increases at the same rate, but as the load approaches the capacity, goodput rises more gradually.
- This falloff is because bursts of traffic can occasionally mount up and cause some losses at buffers inside the network.
- In this state, senders are furiously sending packets, but in- creasingly little useful work is being accomplished.
- The corresponding delay is given in Fig. 6-19(b). Initially the delay is fixed, representing the propagation delay across the network.
- As the load approaches the capacity, the delay rises, slowly at first and then much more rapidly
- The delay cannot really go to infinity, except in a model in which the routers have infinite buffers. Instead, packets will be lost after experiencing the maximum buffering delay
- For both goodput and delay, performance begins to degrade at the onset of congestion.
- This point is be- low the capacity. To identify it, Kleinrock (1979) proposed the metric of **power**, where $power = \frac{load}{delay}$

Max-Min Fairness

- Perhaps the first consideration is to ask what this problem has to do with con- gestion control. After all, if the network gives a sender some amount of bandwidth to use, the sender should just use that much bandwidth.
- They may for some flows if quality of service is supported, but many connections will seek to use whatever bandwidth is available or be lumped together by the network under a common allocation.
- A second consideration is what a fair portion means for flows in a network. Itis simple enough if N flows use a single link, in which case they can all have $1/N$ of the bandwidth.
- An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an alloca- tion that is no larger.
- A third consideration is the level over which to consider fairness. A network could be fair at the level of connections, connections between a pair of hosts, orall connections per host.

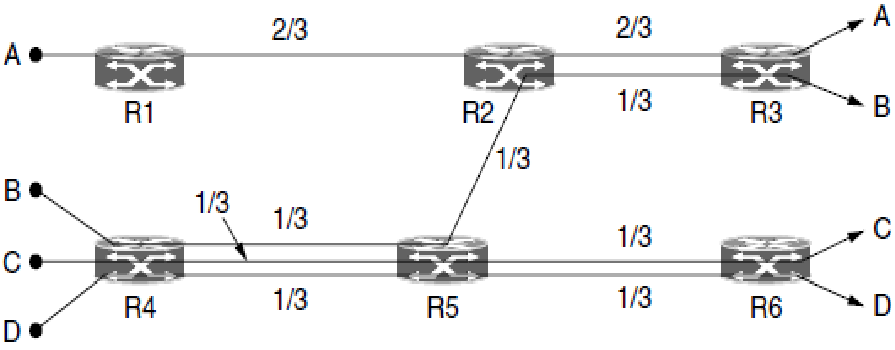


Figure 6-20. Max-min bandwidth allocation for four flows.

- ○ Let us see an example. A max-min fair allocation is shown for a network with four flows, *A*, *B*, *C*, and *D*, in Fig. 6-20. Each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the links will have different capacities.
- ○ Notice that all of the other links have spare capacity. However, this capacity cannot be given to any of the flows without decreasing the capacity of another, lower flow.

Convergence

- A final criterion is that the congestion control algorithm converges quickly to a fair and efficient allocation of bandwidth.
- A good congestion control algorithm should rapidly converge to the ideal operating point, and it should track that point as it changes over time.

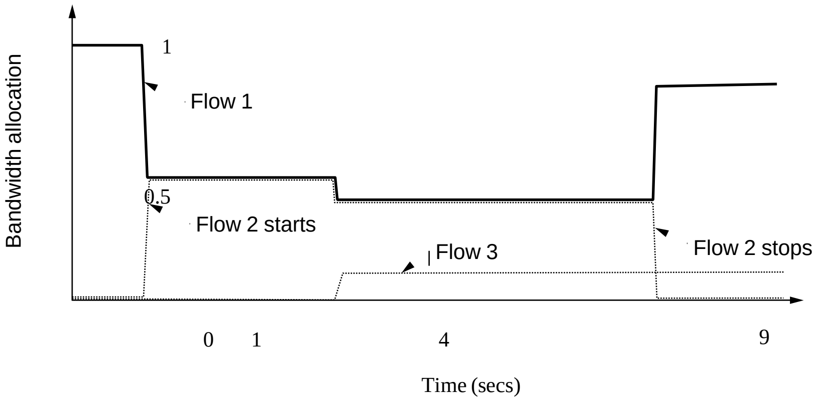


Figure 6-21. Changing bandwidth allocation over time.

- ○ An example of a bandwidth allocation that changes over time and converges quickly is shown in Fig. 6-21,
- ○ Initially, flow 1 has all of the bandwidth. One sec- ond later, flow 2 starts. It needs bandwidth as well.
- ○ The first flow quickly captures 80% of the bandwidth. At all times, the total allocated bandwidth is approximately 100%, so that the network is fully used, and competing

flows get equal treatment.

6.3.2 Regulating the Sending Rate

- The first is flow control, in the case that there is insufficient buffering at the receiver.

- The second is congestion, in the case that there is insufficient capaci- ty in the network

- In Fig. 6-22, we see this problem illustrated hydraulically. In Fig. 6-22(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation.
- . In Fig. 6-22(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network.

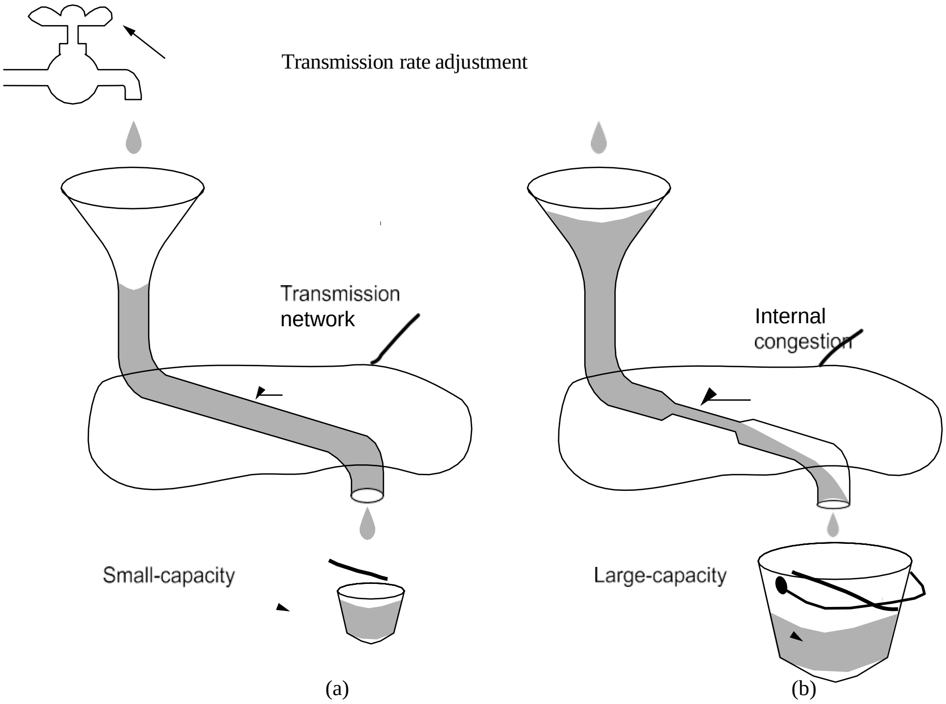


Figure 6-22. (a) A fast network feeding a low-capacity receiver. (b) A slownetwork feeding a high-capacity receiver.

- Combinationsare also possible. For example, Windows includes Compound TCP that uses both packet loss and delay as feedback signals (Tan et al., 2006). These designs are summarized in Fig. 6-23.
- . The way in which the rates are increased or decreased is given by a **control law**. These laws have a major effect on performance.

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
Compound TCP	Packet loss & end-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

Figure 6-23. Signals of some congestion control protocols.

- Chiu and Jain (1989) studied the case of binary congestion feedback and con- cluded that **AIMD** (**Additive Increase Multiplicative Decrease**) is the appropr- iate control law to arrive at the efficient and fair operating point
  - The graph in Fig. 6-24 shows the bandwidth allocated to user 1 on the x-axis and to user 2 on the y-axis.
  - This is shown by the dotted efficiency line. A congestion signal is given by the network to both users when the sum of their allocations crosses this line.
  - The intersection of these lines is the de- sired operating point, when both users have the same bandwidth and all of the net- work bandwidth is used.

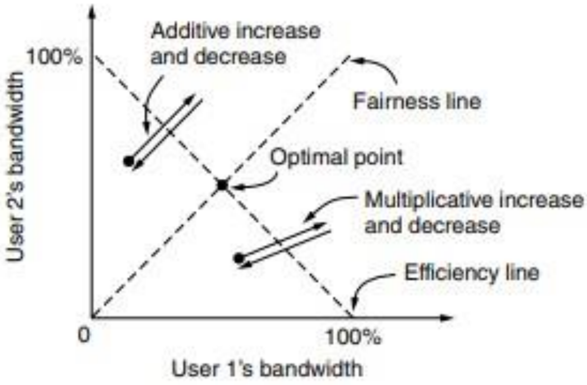


Figure 6-24. Additive and multiplicative bandwidth adjustments.

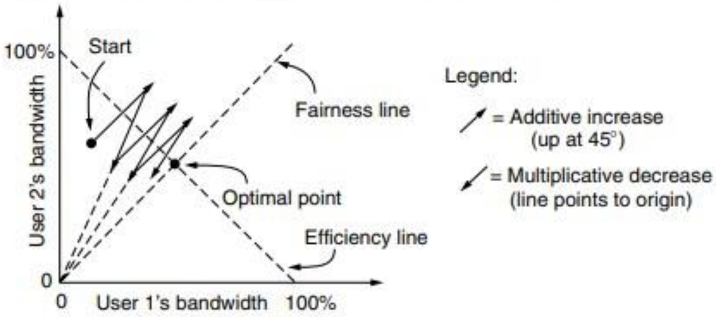


Figure 6-25. Additive Increase Multiplicative Decrease (AIMD) control law.

  - This behavior is the AIMD control law, and it is shown in Fig. 6-25. It can beseen that the path traced by this behavior does converge to the optimal point thatis both fair and efficient.
  - AIMD is the control law that is used by TCP, based on this argument and an- other stability argument.
  - In Sec. 6.5, we will describe in detail how TCP implements an AIMD control law to adjust the sending rate and provide congestion control.
  - .TCP uses this strategy. If the window size is *W* and the round-trip time is *RTT*, the equivalent rate is *W/ RTT*.

6.3.3 Wireless Issues

- The main issue is that packet loss is often used as a congestion signal, including by TCP as we have just discussed.

- Wireless networks lose packets all the time due to transmission er- rors.
- To function well, the only packet losses that the congestion control algorithm should observe are losses due to insufficient bandwidth, not losses due to trans- mission errors.
- One solution to this problem is to mask the wireless losses by using retransmissions over the wireless link.

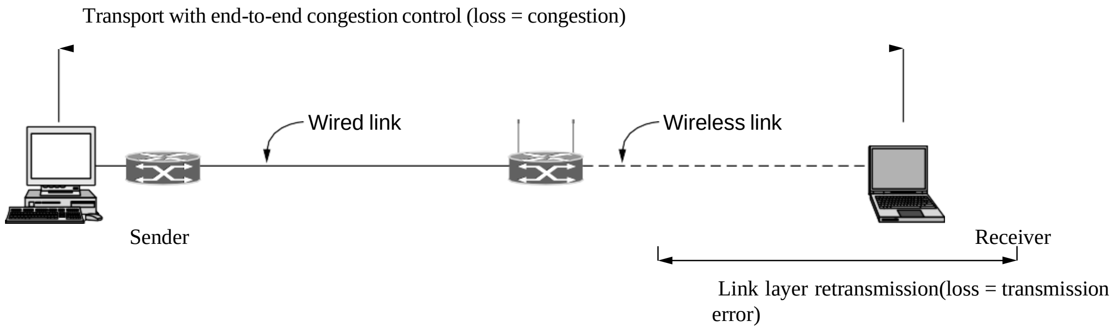

Transport with end-to-end congestion control (loss = congestion)
Wired link
Wireless link
Sender
Receiver
Link layer retransmission(loss = transmission error)

Figure 6-26. Congestion control over a path with a wireless link.

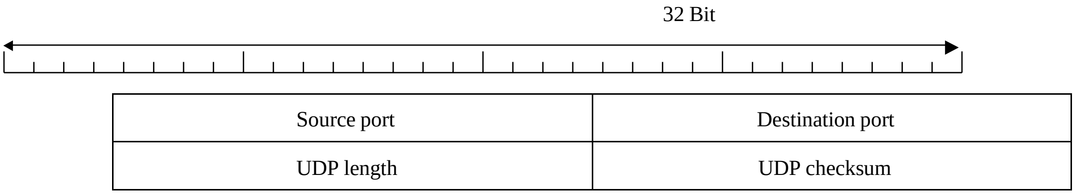
  - Fig. 6-26 shows a path with a wired and wireless link for which the masking strategy is used.
  - There are two aspects to note. First, the sender does not necessarily know that the path includes a wireless link, since all it sees is the wired link to which it is attached.
  - Internet paths are heterogeneous and there is no general method for the sender to tell what kind of links comprise the path.
  - This complicates the congestion control problem, as there is no easy way to use one protocol for wireless links and another protocol for wired links.

6.4 THE INTERNET TRANSPORT PROTOCOLS: UDP

- The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. The protocols complement each other.
- The connectionless protocol is UDP. It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed. The connection-oriented protocol is TCP.
- Since UDP isa transport layer protocol that typically runs in the operating system and protocols that use UDP typically run in user space, these uses might be considered applications.

6.4.1 Introduction to UDP

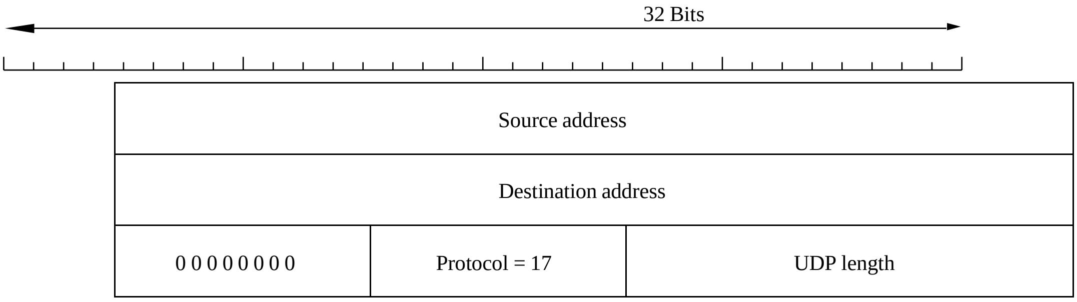
- The Internet protocol suite supports a connectionless transport protocol called UDP (User Datagram Protocol).
- UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- UDP transmits **segments** consisting of an 8-byte header followed by the pay- load. The header is shown in Fig. 6-27.
- The two **ports** serve to identify the end- points within the source and destination machines.
- By copying the *Source port* field from the incoming segment into the *Destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.



Source port	Destination port
UDP length	UDP checksum

Figure 6-27. The UDP header.

- The pseudoheader for the case of IPv4 is shown in Fig. 6-28.
- It contains the 32-bit IPv4 addresses of the source and destination machines, the protocol number for UDP (17), and the byte count for the UDP segment (including the header).



Source address		
Destination address		
0 0 0 0 0 0 0 0	Protocol = 17	UDP length

Figure 6-28. The IPv4 pseudoheader included in the UDP checksum.

- It does not do flow control, congestion control, or retransmission upon receipt of a bad segment.

6.4.2 Remote Procedure Call

- In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language.

- This observation has led people to try to arrange request-reply interactions on networks to be cast in the form of procedure calls.
- When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called pro- cedure takes place on 2.
- Information can be transported from the caller to the cal- lee in the parameters and can come back in the procedure result.

- No message pas- sing is visible to the application programmer. This technique is known as **RPC** (**Remote Procedure Call**)
- In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub.**
- The represent the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**.

    - The actual steps in making an RPC are shown in Fig. 6-29. Step 1 is the cli- ent calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
    - Step 2 is the client stub packing the pa- rameters into a message and making a system call to send the message. Packing the parameters is called **marshaling.**
    - Step 3 is the operating system sending the message from the client machine to the server machine.
    - Step 4 is the operating system passing the incoming packet to the server stub.
    - Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

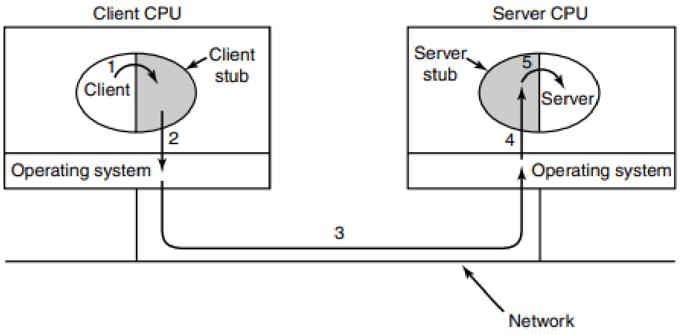


Figure 6-29. Steps in making a remote procedure call. The stubs are shaded.

6.4.3 Real-Time Transport Protocols

- Thus was **RTP** (**Real-time Transport Protocol**) born. It is described in RFC 3550 and is now in widespread use for multimedia applications.
- The first is the RTP protocol for transporting audio and video data in packets.
- The second is the processing that takes place, mostly at the receiver, to play out the audio and video at the right time.

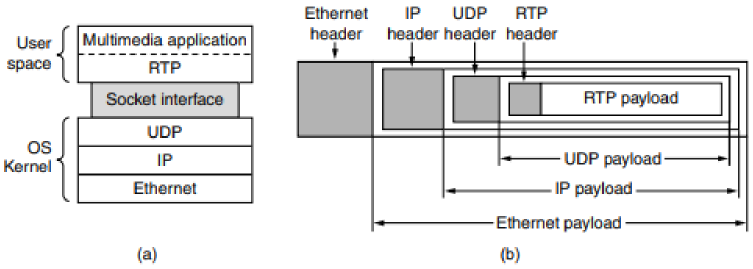


Figure 6-30. (a) The position of RTP in the protocol stack. (b) Packet nesting.

➢ RTP—The Real-time Transport Protocol

- The basic function of RTP is to multiplex several real-time data streams ontoa single stream of UDP packets.
- The UDP stream can be sent to a single destina- tion (unicasting) or to multiple destinations (multicasting).
- The RTP header is illustrated in Fig. 6-31. It consists of three 32-bit words and potentially some extensions.
- The first word contains the *Version* field, which is already at 2. Let us hope this version is very close to the ultimate version since there is only one code point left.

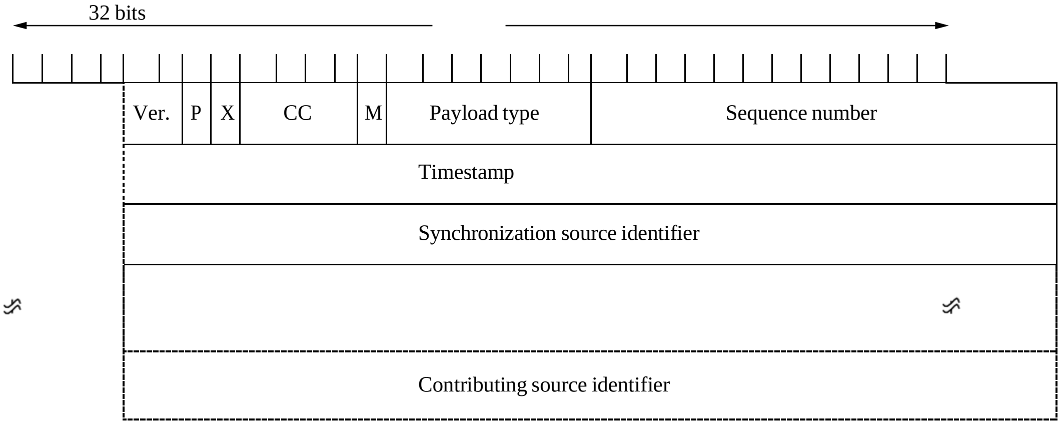

32 bits
Ver.
P
X
CC
M
Payload type
Sequence number
Timestamp
Synchronization source identifier
Contributing source identifier

Figure 6-31. The RTP header.

- The *M* bit is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or some- thing else that the application understands.

➢ RTCP—The Real-time Transport Control Protocol

- RTP has a little sister protocol (little sibling protocol?) called **RTCP** (**Real- time Transport Control Protocol**).
- The first function can be used to provide feedback on delay, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources
- The *Payload type* field is used to tell the destination what encoding algorithm is used for the current packet, making it possible to vary it on demand.
- RTCP also handles interstream synchronization. The problem is that different streams may use different clocks, with different granularities and different drift rates.

➢ Playout with Buffering and Jitter Control

- The packets are injected with exactly the right intervals be- tween them at the sender, they will reach the receiver with different relative times. This variation in delay is called **jitter**.
- The solution to this problem is to **buffer** packets at the receiver before they are played out to reduce the jitter.
  - As an example, in Fig. 6-32 we see a stream of packets being delivered with a substantial amount of jitter.
  - Packet 1 is sent from the server at *t* _ 0 sec and arrives at the client at *t* _ 1 sec.

- As the packets arrive, they are buffered on the client machine.

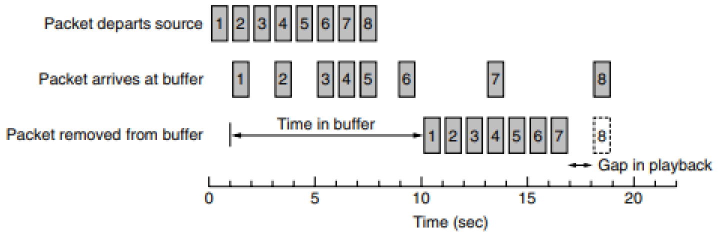


Figure 6-32. Smoothing the output stream by buffering packets.

- A key consideration for smooth playout is the **playback point**, or how long to wait at the receiver for media before playing it out. Deciding how long to wait depends on the jitter.
    - The difference between a low-jitter and high-jitter con- nection is shown in Fig. 6-33.
    - The average delay may not differ greatly between the two, but if there is high jitter the playback point may need to be much further out to capture 99% of the packets than if there is low jitter.
    - One way to avoid this problem for audio isto adapt the playback point between **talkspurts**, in the gaps in a conversation.

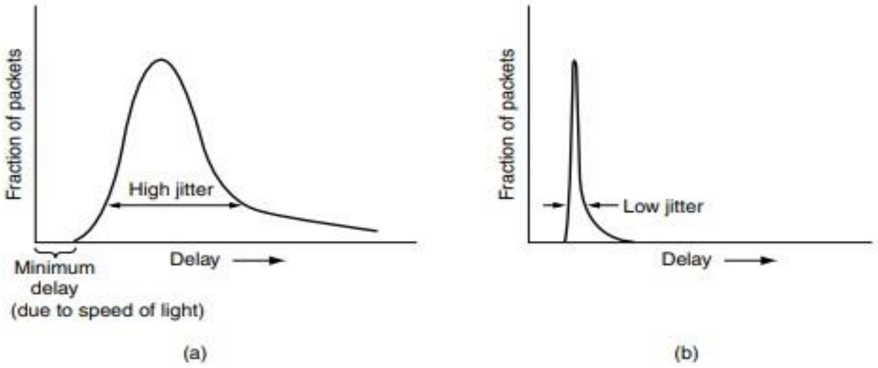


Figure 6-33. (a) High jitter. (b) Low jitter.

6.5 THE INTERNET TRANSPORT PROTOCOLS: TCP

- UDP is a simple protocol and it has some very important uses, such as client- server interactions and multimedia, but for most Internet applications, reliable, se- quenced delivery is needed.

- UDP cannot provide this, so another protocol is re- quired. It is called TCP and is the main workhorse of the Internet.

6.5.1 Introduction to TCP

- **TCP** (**Transmission Control Protocol**) was specifically designed to providea reliable end-to-end byte stream over an unreliable internetwork.
- An internet- work differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters.

- TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures.

6.5.2 The TCP Service Model

- TCP service is obtained by both the sender and the receiver creating end points, called **sockets.**
- Each socket has a socket num- ber (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**.
- Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users (e.g., root in UNIX systems). They are called **well-known ports**.

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Figure 6-34. Some assigned ports.

- ○ The list of well-known ports is given at *www.iana.org*. Over 700 havebeen assigned. A few of the better-known ones are listed in Fig. 6-34.

  - Doing so would clutter up memory with daemons that were idle most of the time. Instead, what is commonly done is to have a single daemon, called **inetd** (**Internet daemon**) in UNIX, attach itself to multiple ports and wait for the first incoming connection.
  - All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points.

- ○ if the sending process does four 512-byte writes to a TCP stream, these data may be delivered to the receiving process as four 512-byte chunks, two 1024-byte chunks, one 2048-byte chunk (seeFig. 6-35), or some other way.

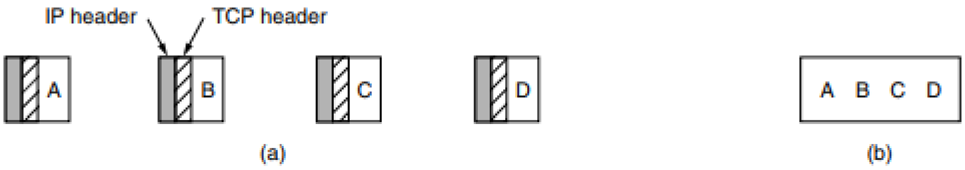


Figure 6-35. (a) Four 512-byte segments sent as separate IP datagrams. (b) The 2048 bytes of data delivered to the application in a single READ call.

- ○ The reader of a file cannot tell whether the file was written a block at a time, a byte at a time, or all in one blow.

- For Internet archaeologists, we will also mention one interesting feature of TCP service that remains in the protocol but is rarely used: **urgent data**.

6.5.3 The TCP Protocol

- A **TCP segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes.
- The TCP software decides how big segments should be. It can accumulate data from several writes into one segment or can split data from one write over multiple segments.
- Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515- byte IP payload. Second, each link has an **MTU** (**Maximum**
- The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size.
- When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment.

6.5.4 The TCP Segment Header

- Figure 6-36 shows the layout of a TCP segment. Every segment begins with a fixed-format, 20-byte header.
- . The fixed header may be followed by header options. After the options, if any, up to 65,535 □ 20 □ 20 □ 65, 495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header.

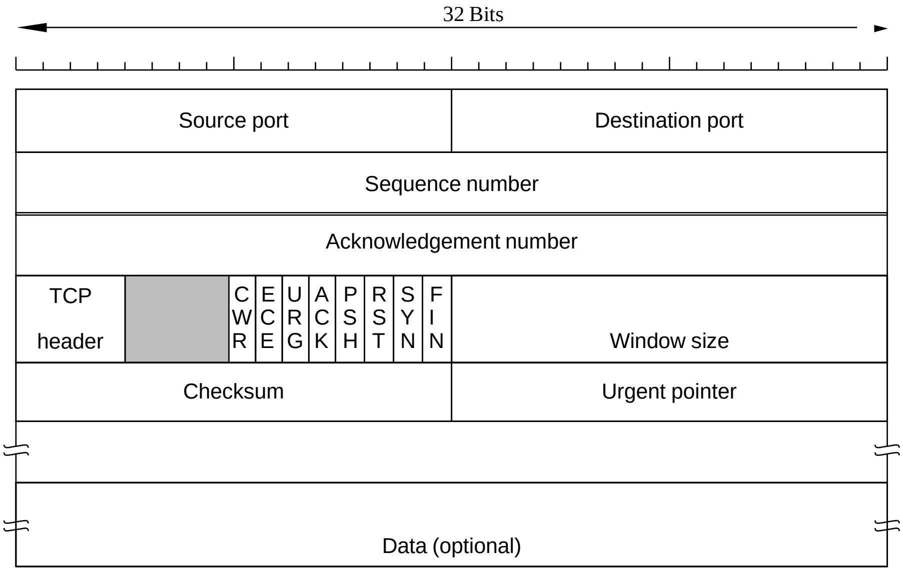


Figure 6-36. The TCP header.

- This connection identifier is called a **5 tuple** because it consists of five pieces of information: the protocol (TCP), source IP and source port, and destination IP and destination port.

- It is a **cumulative acknowledgement** because it summarizes the received data with a single number.
- The *TCP header length* tells how many 32-bit words are contained in the TCP header.
- The *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
- A widely used option is the one that allows each host to specify the **MSS** (**Maximum Segment Size**) it is willing to accept.
- The **window scale** option allows the sender and receiver to negotiate a window scale factor at the start of a connection.
- The **timestamp** option carries a timestamp sent by the sender and echoed by the receiver.
- The **PAWS** (**Protection Against Wrapped Sequence numbers**) scheme discards ar- riving segments with old timestamps to prevent this problem.
- Finally, the **SACK** (**Selective ACKnowledgement**) option lets a receiver tell a sender the ranges of sequence numbers that it has received.

6.5.5 TCP Connection Establishment

- When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port given in the *Destination port* field.
  - The sequence of TCP segments sent in the normal case is shown in Fig. 6-37(a).

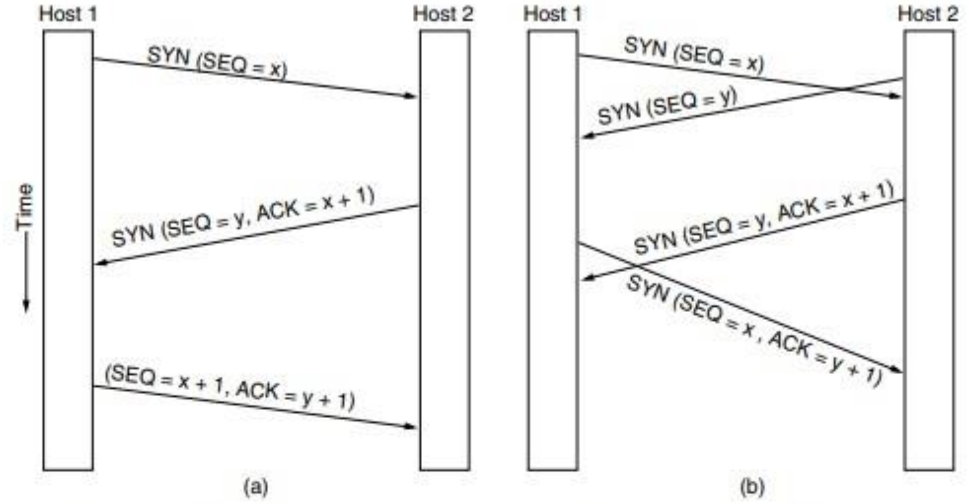

Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
Time
(a)
Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x, ACK = y + 1)
(b)

Figure 6-37. (a) TCP connection establishment in the normal case. (b) Simultaneous connection establishment on both sides.

  - In the event that two hosts simultaneously attempt to establish a connection between the same two sockets, the sequence of events is as illustrated in Fig. 6- 37(b).
- A vulnerability with implementing the three-way handshake is that the listening process must remember its sequence number as soon it responds with its own *SYN* segment.
- This means that a malicious sender can tie up resources ona host by sending a stream of *SYN* segments and never following through to com- plete the connection. This attack is called a **SYN flood.**
- One way to defend against this attack is to use **SYN cookies**. Instead of remembering the sequence number, a host chooses a cryptographically generated sequence number, puts it on the outgoing segment, and forgets it.

- There are some caveats, such as the inability to handle TCP options, so SYN cookies may be used only when the hostis subject to a SYN flood.

6.5.6 TCP Connection Release

- To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit.
- Data may continue to flow indefinitely in the other direction, how- ever. When both directions have been shut down, the connection is released.
- There is, in fact, no essential difference between the two hosts releasing sequentially or simultaneously.

6.5.7 TCP Connection Management Modeling

- The steps required to establish and release connections can be represented in a finite state machine with the 11 states listed in Fig. 6-38.
- In each state, certain events are legal. When a legal event happens, some action may be taken.
- Each connection starts in the *CLOSED* state. It leaves that state when it does either a passive open (LISTEN) or an active open (CONNECT).
- If the other side does the opposite one, a connection is established and the state becomes *ESTABLISHED.*
- Connection release can be initiated by either side. When it is com- plete, the state returns to *CLOSED.*

State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIME WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

Figure 6-38. The states used in the TCP connection management finite state machine.

- The finite state machine itself is shown in Fig. 6-39. The common case of a client actively connecting to a passive server is shown with heavy lines—solid for the client, dotted for the server.

- Each line in Fig. 6-39 is marked by an *event/action* pair. The event can either bea user-initiated system call (CONNECT, LISTEN, SEND, or CLOSE), a segment arrival (*SYN*, *FIN*, *ACK*, or *RST*), or, in one case, a timeout of twice the maximum packet lifetime.

- When an application program on the client machine issues a CONNECT re- quest, the local TCP entity creates a connection record, marks it as being in the *SYN SENT* state, and shoots off a *SYN* segment.

- When the *ACK* arrives, a transition is made to the state *FIN WAIT 2* and one direction of the connection is closed. When the other side closes, too, a *FIN* comes in, which is acknowledged.

- When a *SYN* comes in, it is acknowledged and the server goes to the *SYN RCVD* state. When the server's *SYN* is itself acknowledged, the three-way handshake is complete and the server goes to the *ESTABLISHED* state. Data transfer can now occur.

- When the client is done transmitting its data, it does a CLOSE, which causes a *FIN* to arrive at the server (dashed box marked ''passive close'').

- The server is then signaled. When it, too, does a CLOSE, a *FIN* is sent to the client.

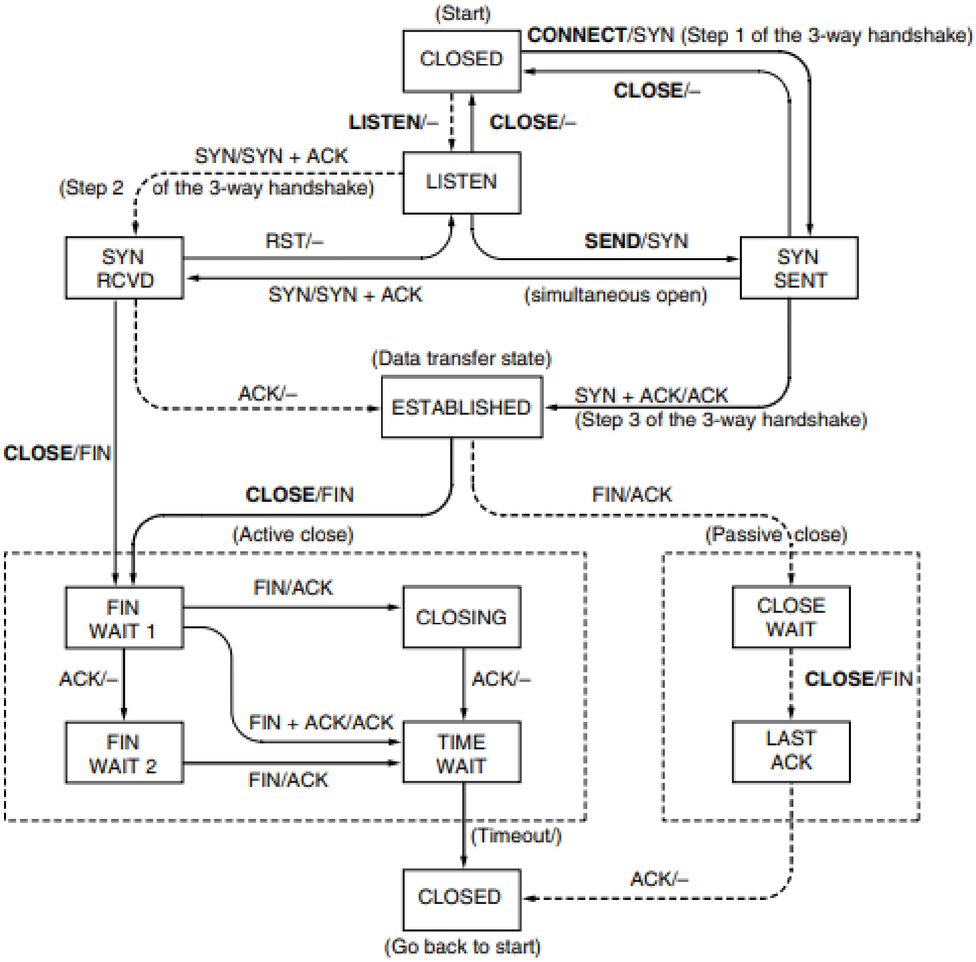


Figure 6-39. TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled with the event causing it and the action resulting from it, separated by a slash.

Module – 5
APPLICATION LAYER

Principles of Network Applications

Network application development is writing programs that run on different end systems and communicate with each other over the network.

For example, in the Web application there are two distinct programs that communicate with each other: the browser program running in the user's host and the Web server program running in the Web server host.

Network Application Architectures.

There are two different network application architecture, they are

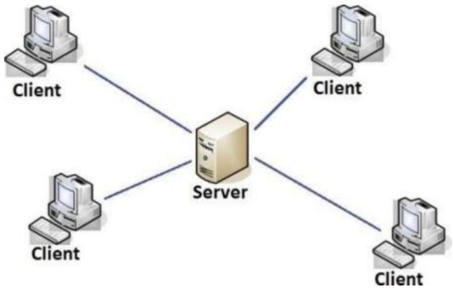
1) Client Server Architecture

2) P2P Architecture

Client Server Architecture:

- In client-server architecture, there is an always-on host, called the server, which provides services when it receives requests from many other hosts, called clients.

  Example: In Web application Web server services requests from browsers running on client hosts. When a Web server receives a request for an object from a client host, it responds by sending the requested object to the client host.
- In client-server architecture, clients do not directly communicate with each other.
- The server has a fixed, well-known address, called an IP address. Because the server has a fixed, well-known address, and because the server is always on, a client can always contact the server by sending a packet to the server's IP address.
- Some of the better-known applications with a client-server architecture include the Web, FTP, Telnet, and e-mail.



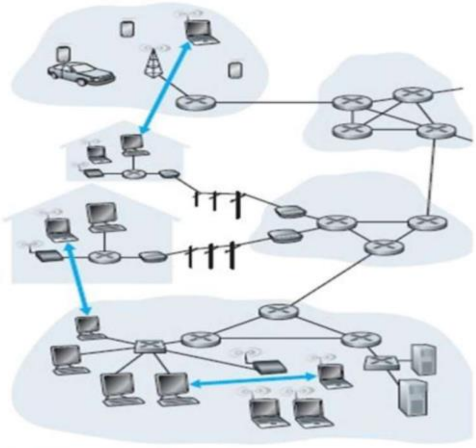
Client Server Architecture

- In a client-server application, a single-server host is incapable of keeping up with all the requests from clients. For this reason, a data center, housing a large number of hosts, is often used to create a powerful virtual server.
- The most popular Internet services—such as search engines (e.g., Google and Bing), Internet commerce (e.g., Amazon and e-Bay), Web-based email (e.g., Gmail and Yahoo Mail), social networking (e.g., Facebook and Twitter) —employ one or more data centers.

Peer-to-peer (P2P) Architecture:

- In a P2P architecture, there is minimal dependence on dedicated servers in data centers.
- The application employs direct communication between pairs of intermittently connected hosts, called peers.
- The peers are not owned by the service provider, but are instead desktops and laptops controlled by users, with most of the peers residing in homes, universities, and offices.
- Many of today's most popular and traffic-intensive applications are based on P2P architectures. These applications include file sharing (e.g., BitTorrent), Internet Telephony (e.g., Skype), and IPTV (e.g., Kankan and PPstream).
- **Features:**
  - **Self-scalability:**
    For example, in a P2P file-sharing application, although each peer generates workload by requesting files, each peer also adds service capacity to the system by distributing files to other peers.
  - **Cost effective:**

P2P architectures are also cost effective, since they normally don't require significant server infrastructure and server bandwidth



P2P Architecture

Future P2P applications face three major challenges:

1. **ISP Friendly.** Most residential ISPs have been dimensioned for "asymmetrical" bandwidth usage, that is, for much more downstream than upstream traffic. But P2P video streaming and file distribution applications shift upstream traffic from servers to residential ISPs, thereby putting significant stress on the ISPs. Future P2P applications need to be designed so that they are friendly to ISPs
2. **Security.** Because of their highly distributed and open nature, P2P applications can be a challenge to secure
3. **Incentives.** The success of future P2P applications also depends on convincing users to volunteer bandwidth, storage, and computation resources to the applications, which is the challenge of incentive design.

Processes Communicating

- A Process is a program or application under execution.
- When processes are running on the same or different end system, they can communicate with each other with inter process communication, using rules that are governed by the end system's operating system.

- Processes on two different end systems communicate with each other by exchanging messages across the computer network. A sending process creates and sends messages into the network; a receiving process receives these messages and possibly responds by sending messages back.

Client and Server Processes

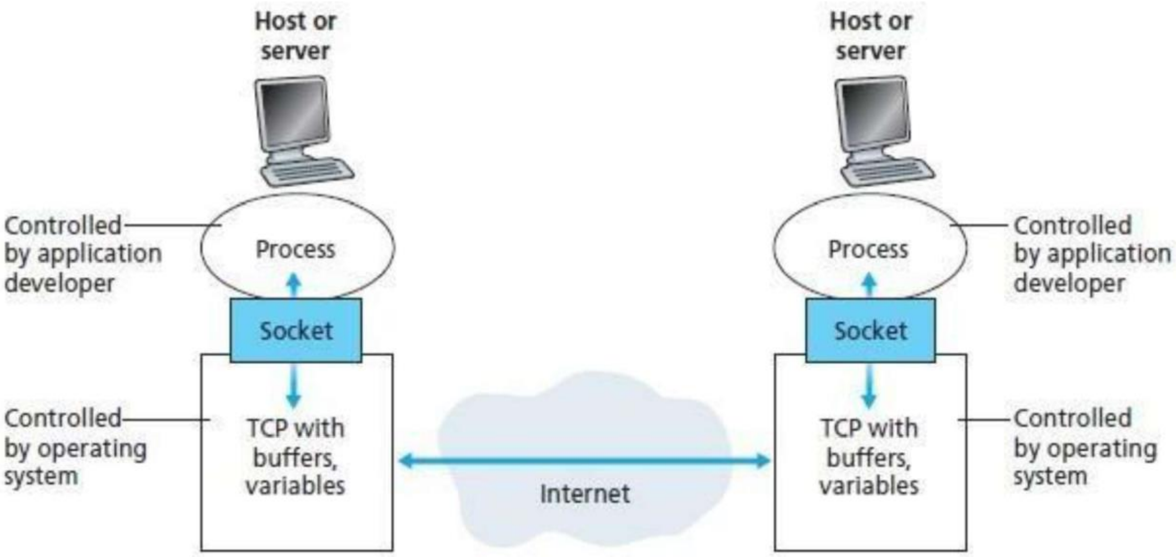
- A network application consists of pairs of processes that send messages to each other over a network.

  For example, in the Web application a client browser process exchanges messages with a Web server process.

- In the context of a communication session between a pair of processes, the process that initiates the communication is labeled as the client. The process that waits to be contacted to begin the session is the server.

The Interface between the Process and the Computer Network

- A process sends messages into, and receives messages from, the network through a software interface called a socket.
- It is also referred to as the Application Programming Interface (API) between the application and the network, since the socket is the programming interface with which network applications are built.
- The application at the sending side pushes messages through the socket. At the other side of the socket, the transport-layer protocol has the responsibility of getting the messages to the socket of the receiving process.


Host or server
Host or server
Controlled by application developer
Process
Socket
Controlled by operating system
TCP with buffers, variables
Internet
Process
Controlled by application developer
Socket
TCP with buffers, variables
Controlled by operating system

Application processes, sockets, and underlying transport protocol

Addressing Processes

- For a process running on one host to send packets to a process running on another host, the receiving process needs to have an address.
- To identify the receiving process, two pieces of information need to be specified:
  (1) The address of the host
  (2) An identifier that specifies the receiving process in the destination host.
- In the Internet, the host is identified by its IP address.
- In addition to knowing the address of the host to which a message is destined, the sending process must also identify the receiving process running in the host. A destination port number serves this purpose. Popular applications have been assigned specific port numbers. For example, a Web server is identified by port number 80. A mail server process (using the SMTP protocol) is identified by port number 25.

Transport Services Available to Applications

1) Reliable Data Transfer

- Packets can get lost within a computer network. For example, a packet can overflow a buffer in a router, or can be discarded by a host or router after having some of its bits corrupted.

- For many applications—such as electronic mail, file transfer, remote host access, Web document transfers, and financial applications —data loss can have devastating consequences.
- Thus, to support these applications, something has to be done to guarantee that the data sent by one end of the application is delivered correctly and completely to the other end of the application.
- If a protocol provides such a guaranteed data delivery service, it is said to provide reliable data transfer. One important service that a transport-layer protocol can potentially provide to an application is process-to-process reliable data transfer.
- When a transport protocol provides this service, the sending process can just pass its data into the socket and know with complete confidence that the data will arrive without errors at the receiving process.
- When a transport-layer protocol doesn't provide reliable data transfer, some of the data sent by the sending process may never arrive at the receiving process. This may be acceptable for loss-tolerant applications, most notably multimedia applications such as conversational audio/video that can tolerate some amount of data loss.

2) Throughput

- Transport-layer protocol could provide guaranteed available throughput at some specified rate.
- With such a service, the application could request a guaranteed throughput of r bits/sec, and the transport protocol would then ensure that the available throughput is always at least r bits/sec. Such a guaranteed throughput service would appeal to many applications.

  For example, if an Internet telephony application encodes voice at 32 kbps, it needs to send data into the network and have data delivered to the receiving application at this rate.
- If the transport protocol cannot provide this throughput, the application would need to encode at a lower rate or may have to give up.
- Applications that have throughput requirements are said to be bandwidth-sensitive applications. Many current multimedia applications are bandwidth sensitive
- Elastic applications can make use of as much, or as little, throughput as happens to be available. Electronic mail, file transfer, and Web transfers are all elastic applications.

3) Timing

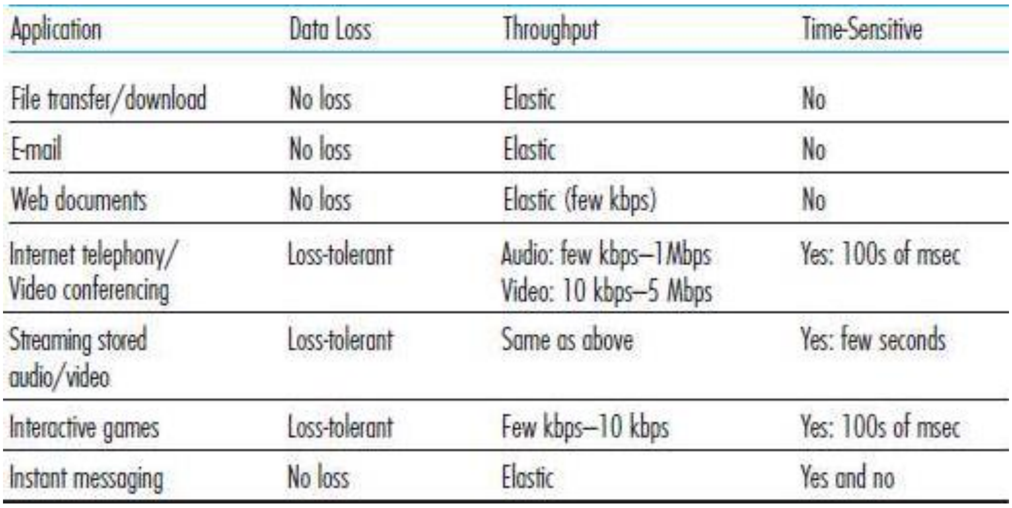
- A transport-layer protocol can also provide timing guarantees.
- Interactive real-time applications, such as Internet telephony, virtual environments, teleconferencing, and multiplayer games require tight timing constraints on data delivery in order to be effective.

4) Security

- Transport protocol can provide an application with one or more security services.
  For example, in the sending host, a transport protocol can encrypt all data transmitted by the sending process, and in the receiving host, the transport-layer protocol can decrypt the data before delivering the data to the receiving process.
- A transport protocol can provide security services like confidentiality, data integrity and end-point authentication.

Transport Services Provided by the Internet

The Internet makes two transport protocols available to applications, UDP and TCP.



Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Requirements of selected network applications

TCP Services

The TCP service model includes a connection-oriented service and a reliable data transfer service.

1) Connection-oriented service:

- In TCP the client and server exchange transport layer control information with each other before the application-level messages begin to flow.
- This handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets.
- After the handshaking phase, a TCP connection is said to exist between the sockets of the two processes.
- The connection is a full-duplex connection in that the two processes can send messages to each other over the connection at the same time.
- When the application finishes sending messages, it must tear down the connection.

2) Reliable data transfer service:

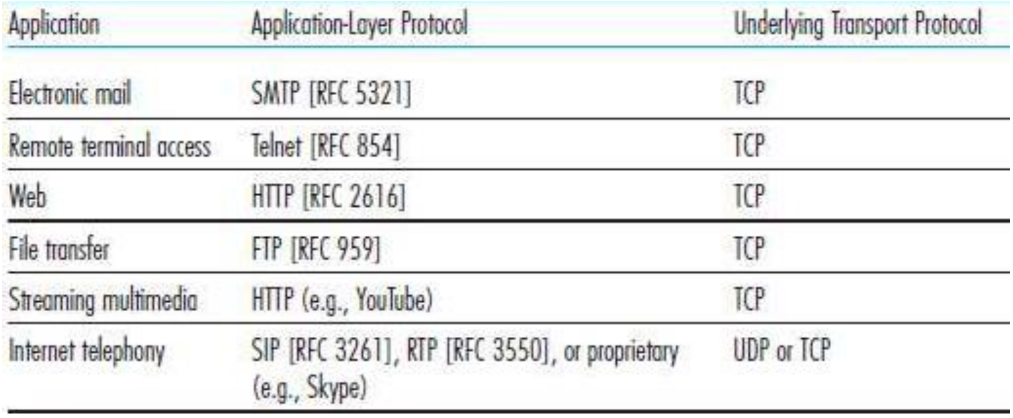
- The communicating processes can rely on TCP to deliver all data sent without error and in the proper order.
- When one side of the application passes a stream of bytes into a socket, it can count on TCP to deliver the same stream of bytes to the receiving socket, with no missing or duplicate bytes.

TCP also includes a congestion-control mechanism.

UDP Services

- UDP is connectionless, so there is no handshaking before the two processes start to communicate.
- UDP provides an unreliable data transfer service—that is, when a process sends a message into a UDP socket, UDP provides no guarantee that the message will ever reach the receiving process.
- UDP does not include a congestion-control mechanism, so the sending side of UDP can

pump data into the layer below (the network layer) at any rate it pleases.



Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Popular Internet applications, their application-layer protocols, and their underlying transport protocols

Application-Layer Protocols

An application-layer protocol defines:

- The types of messages exchanged, for example, request messages and response messages
- The syntax of the various message types, such as the fields in the message and how the fields are delineated
- The semantics of the fields, that is, the meaning of the information in the fields
- Rules for determining when and how a process sends messages and responds to messages.

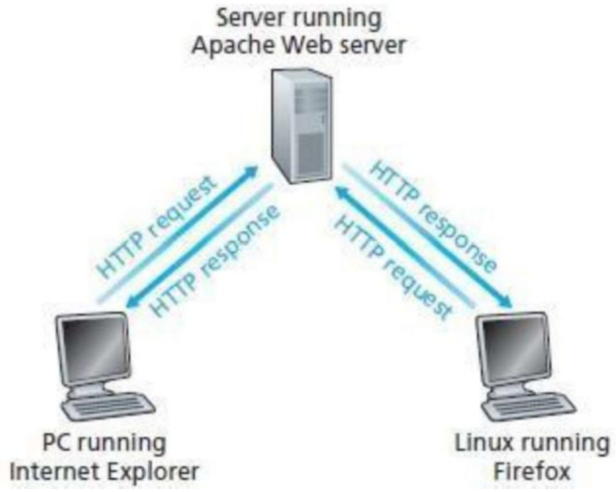
The Web and HTTP

Overview of HTTP

- The Hyper Text Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web.
- HTTP is implemented in two programs: a client program and a server program. The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages. HTTP defines the structure of these messages and how the

client and server exchange the messages.

- A Web page consists of objects. An object is simply a file like HTML file, a JPEG image, a Java applet, or a video clip—that is addressable by a single URL.
- Most Web pages consist of a base HTML file and several referenced objects. For example, if a Web page contains HTML text and five JPEG images, then the Web page has six objects: the base HTML file plus the five images.
- The base HTML file references the other objects in the page with the objects' URLs. Each URL has two components: the hostname of the server that houses the object and the object's path name.
- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients.
- When a user requests a Web page (for example, clicks on a hyperlink), the browser sends HTTP request messages for the objects in the page to the server. The server receives the requests and responds with HTTP response messages that contain the objects.
- HTTP uses TCP as its underlying transport protocol. The HTTP client first initiates a TCP connection with the server. Once the connection is established, the browser and the server processes access TCP through their socket interfaces.
- It is important to note that the server sends requested files to clients without storing any state information about the client. If a particular client asks for the same object twice in a period of a few seconds, the server does not respond by saying that it just served the object to the client; instead, the server resends the object, as it has completely forgotten what it did earlier. Because an HTTP server maintains no information about the clients, HTTP is said to be a stateless protocol.



Non-Persistent and Persistent Connections

If Separate TCP connection is used for each request and response, then the connection is said to be non persistent. If same TCP connection is used for series of related request and response, then the connection is said to be persistent.

HTTP with Non-Persistent Connections

Let's suppose the page consists of a base HTML file and 10 JPEG images, and that all 11 of these objects reside on the same server.

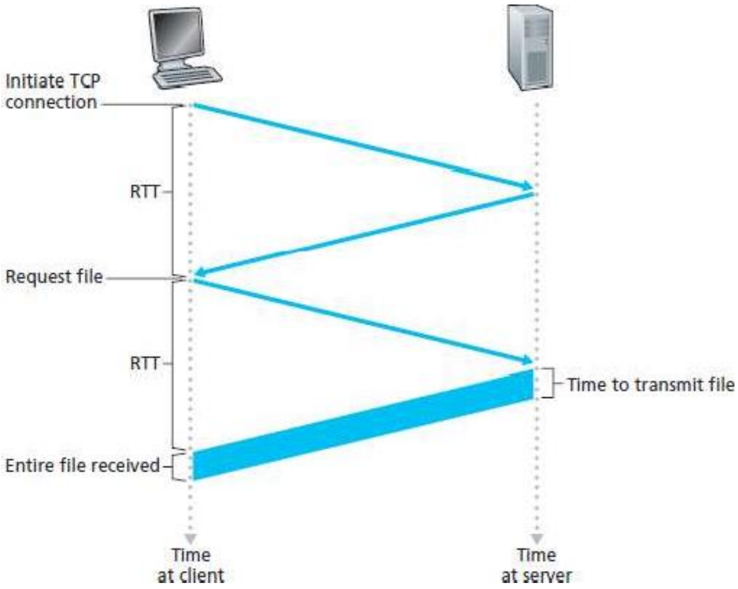
Further suppose the URL for the base HTML file is

http://www.someSchool.edu/someDepartment/home.index

Here is what happens:

1. The HTTP client process initiates a TCP connection to the server www.someSchool.edu on port number 80, which is the default port number for HTTP. Associated with the TCP connection, there will be a socket at the client and a socket at the server.
2. The HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name /someDepartment/home.index.
3. The HTTP server process receives the request message via its socket, retrieves the object /someDepartment/home.index from its storage (RAM or disk), encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.
4. The HTTP server process tells TCP to close the TCP connection.

5. The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
6. The first four steps are then repeated for each of the referenced JPEG objects.



- Round-trip time (RTT) is the time it takes for a small packet to travel from client to server and then back to the client.
- The RTT includes packet-propagation delays, packet queuing delays in intermediate routers and switches, and packet-processing delays.
- When a user clicks on a hyperlink, the browser initiate a TCP connection between the browser and the Web server; this invo lves a "three-way handshake"—the client sends a small TCP segment to the server, the server acknowledges and responds with a small TCP segment, and, finally, the client acknowledges back to the server.
- The first two parts of the three way handshake take one RTT.
- After completing the first two parts of the handshake, the client sends the HTTP request message combined with the third part of the three-way handshake (the acknowledgment) into the TCP connection.

- Once the request message arrives at the server, the server sends the HTML file into the TCP connection. This HTTP request/response eats up another RTT. Thus, roughly, the total response time is two RTTs plus the transmission time at the server of the HTML file.

Non-persistent connections have some shortcomings(disadvantages) they are.

1. A brand-new connection must be established and maintained for each requested object. For each of these connections, TCP buffers must be allocated and TCP variables must be kept in both the client and server. This can place a significant burden on the Web server, which may be serving requests from hundreds of different clients simultaneously.
2. Each object suffers a delivery delay of two RTTs— one RTT to establish the TCP connection and one RTT to request and receive an object.

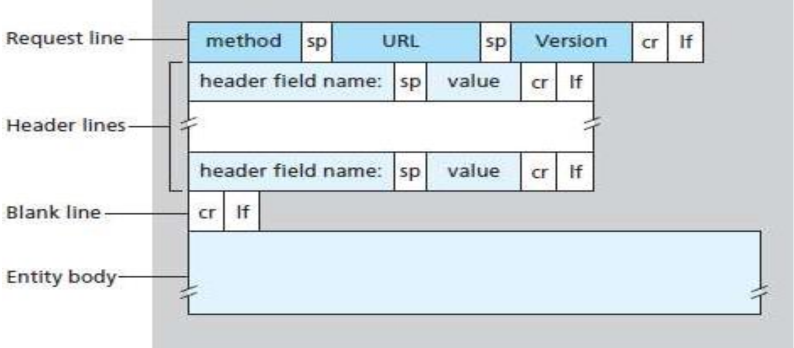
HTTP with Persistent Connections

With persistent connections, the server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connection. In particular, an entire Web page can be sent over a single persistent TCP connection. Moreover, multiple Web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection.

HTTP Message Format Two types of HTTP messages:

- **Request messages and**
- **Response messages**

HTTP Request Message:



Where sp – space, cr – carriage return and lf – line feed.

Method:

There are five HTTP methods:

- **GET:** The GET method is used when the browser requests an object, with the requested object identified in the URL field.
- **POST:** With a POST message, the user is still requesting a Web page from the server, but the specific contents of the Web page depend on what the user entered into the form fields. If the value of the method field is POST, then the entity body contains what the user entered into the form fields.
- **PUT:** The PUT method is also used by applications that need to upload objects to Web servers.
- **HEAD:** Used to retrieve header information. It is used for debugging purpose.
- **DELETE:** The DELETE method allows a user, or an application, to delete an object on a Web server.

URL: Specifies URL of the requested object

Version: This field represents HTTP version, usually HTTP/1.1

Header line:

Ex:
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: french

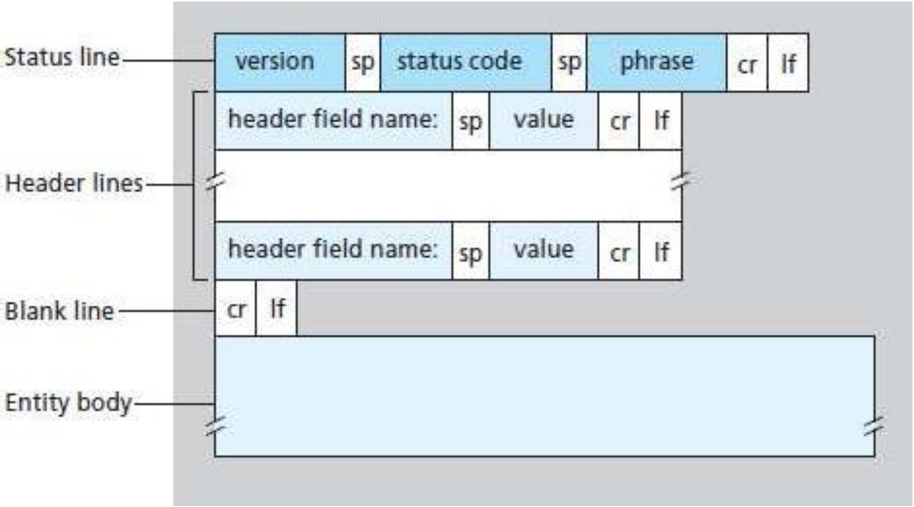
The header line **Host:www.someschool.edu** specifies the host on which the object resides.

By including the **Connection:close** header line, the browser is telling the server that it doesn't want to bother with persistent connections; it wants the server to close the connection after sending the requested object.

The **User-agent:** header line specifies the user agent, that is, the browser type that is making the request to the server. Here the user agent is Mozilla/5.0, a Firefox browser.

The **Accept-language:** header indicates that the user prefers to receive a French version of the object, if such an object exists on the server; otherwise, the server should send its default version.

HTTP Response Message



Ex:

HTTP/1.1 200 OK

Connection: close

Date: Tue, 09 Aug 2011 15:44:04 GMT

Server: Apache/2.2.3 (CentOS)

Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT

Content-Length: 6821

Content-Type: text/html

(data data data data data ...)

The **status line** has three fields: the protocol version field, a status code, and a corresponding status message.

Version is HTTP/1.1

The status code and associated phrase indicate the result of the request. Some common status codes and associated phrases include:

• 200 OK: Request succeeded and the information is returned in the response.

• 301 Moved Permanently: Requested object has been permanently moved; the new URL is specified in Location: header of the response message. The client software will automatically retrieve the new URL.

• 400 Bad Request: This is a generic error code indicating that the request could not be understood by the server.

• 404 Not Found: The requested document does not exist on this server.

• 505 HTTP Version Not Supported: The requested HTTP protocol version is not supported by the server.

Header fields:

- The server uses the **Connection: close** header line to tell the client that it is going to close the TCP connection after sending the message.
- The **Date:** header line indicates the time and date when the HTTP response was created and sent by the server.
- The **Server:** header line indicates that the message was generated by an Apache Web server; it is analogous to the User-agent: header line in the HTTP request message.
- The **Last-Modified:** header line indicates the time and date when the object was created or last modified.
- The **Content-Length:** header line indicates the number of bytes in the object being sent.
- The **Content-Type:** header line indicates that the object in the entity body is HTML text.

User-Server Interaction: Cookies

It is often desirable for a Web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity. For these purposes, HTTP uses cookies.

Cookie technology has four components:

(1) A cookie header line in the HTTP response message;

(2) A cookie header line in the HTTP request message;

(3) A cookie file kept on the user's end system and managed by the user's browser;

(4) A back-end database at the Web site.

Ex:

Suppose a user, who always accesses the Web using Internet Explorer from her home PC, contacts Amazon.com for the first time. Let us suppose that in the past he has already visited the eBay site. When the request comes into the Amazon Web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number. The Amazon Web server then responds to Susan's browser, including in the HTTP response a Set-cookie: header, which contains the identification number.

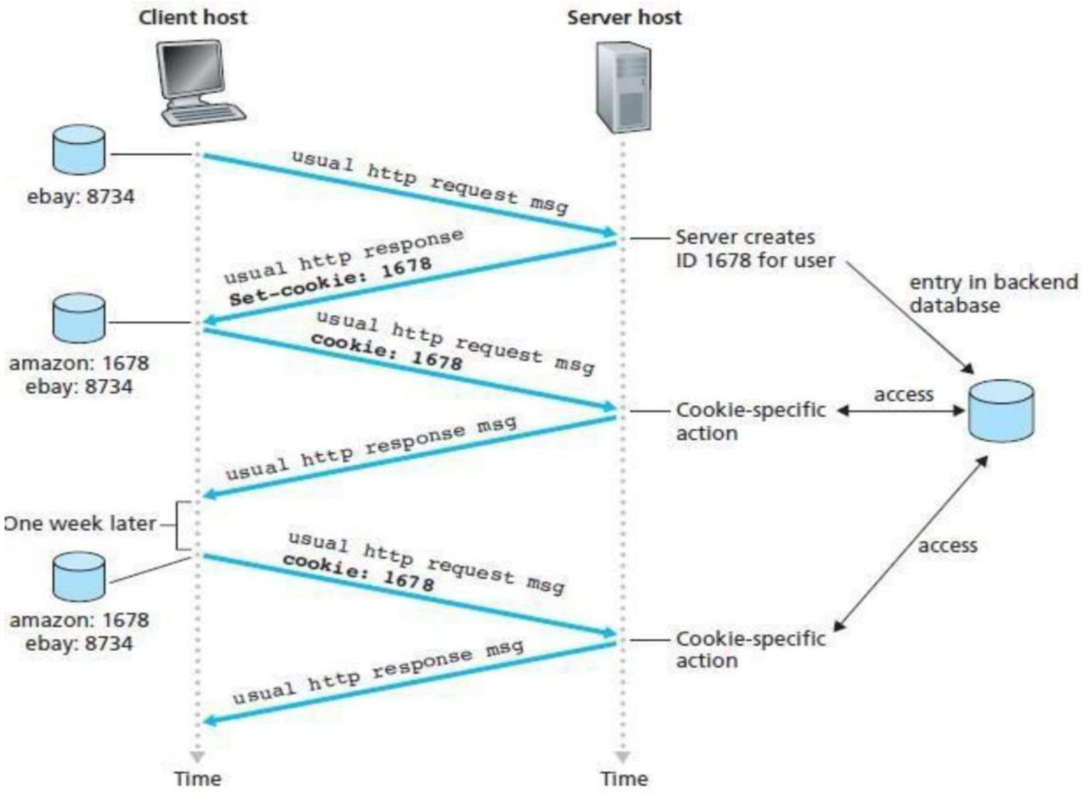
For example, the header line might be:

Set-cookie: 1678

When users browser receives the HTTP response message, it sees the Set-cookie: header. The browser then appends a line to the special cookie file that it manages. This line includes the hostname of the server and the identification number in the Set-cookie: header.

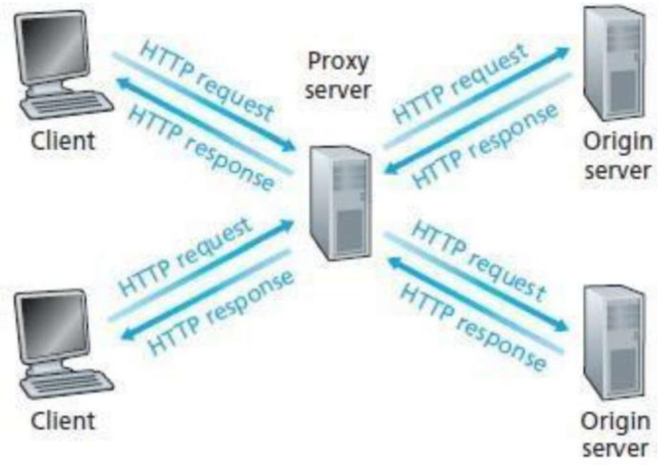
As user continues to browse the Amazon site, each time he requests a Web page, his browser consults his cookie file, extracts his identification number for this site, and puts a cookie header line that includes the identification number in the HTTP request. Specifically, each of his HTTP requests to the Amazon server includes the header line:

Cookie: 1678



Web Caching

- A Web cache—also called a proxy server—is a network entity that satisfies HTTP requests on the behalf of an origin Webserver.
- The Web cache has its own disk storage and keeps copies of recently requested objects in this storage.
- A user's browser can be configured so that all of the user's HTTP requests are first directed to the Web cache.



Ex: Suppose a browser is requesting the object http://www.someschool.edu/campus.gif. Here is what happens:

1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.someschool.edu. The Web cache then sends an HTTP request for the object into the cache-to-server TCP connection.
4. After receiving this request, the origin server sends the object within an HTTP response to the Web cache.
5. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser (over the existing TCP connection between the client browser and the Web cache).

- When web cache receives requests from and sends responses to a browser, it is a server. When it sends requests to and receives responses from an origin server, it is a client.
- Typically a Web cache is purchased and installed by an ISP. For example, a university might install a cache on its campus network and configure all of the campus browsers to point to the cache. Or a major residential ISP (such as AOL) might install one or more caches in its network and pre configure its shipped browsers to point to the installed caches.

- Web caching has seen deployment in the Internet for two reasons. First, a Web cache can substantially reduce the response time for a client request. Second, Web caches can substantially reduce traffic on an institution's access link to the Internet.

The Conditional GET

- Although caching can reduce user-perceived response times, it introduces a new problem—the copy of an object residing in the cache may be stale. In other words, the object housed in the Web server may have been modified since the copy was cached at the client.
- HTTP has a mechanism that allows a cache to verify that its objects are up to date. This mechanism is called the conditional GET.
- An HTTP request message is a so-called conditional GET message if (1) the request message uses the GET method and (2) the request message includes an If-Modified- Since: header line.

Ex: First, on the behalf of a requesting browser, a proxy cache sends a request message to a Web server:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

Second, the Web server sends a response message with the requested object to the cache:

```
HTTP/1.1 200 OK
Date: Sat, 8 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 7 Sep 2011 09:23:24
Content-Type: image/gif
(data data data data data ...)
```

The cache forwards the object to the requesting browser but also caches the object locally. Importantly, the cache also stores the last-modified date along with the object.

Third, one week later, another browser requests the same object via the cache, and the object is still in the cache. Since this object may have been modified at the Web server in the past week, the cache performs an up-to-date check by issuing a conditional GET. Specifically, the cache sends:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

This conditional GET is telling the server to send the object only if the object has been modified since the specified date.

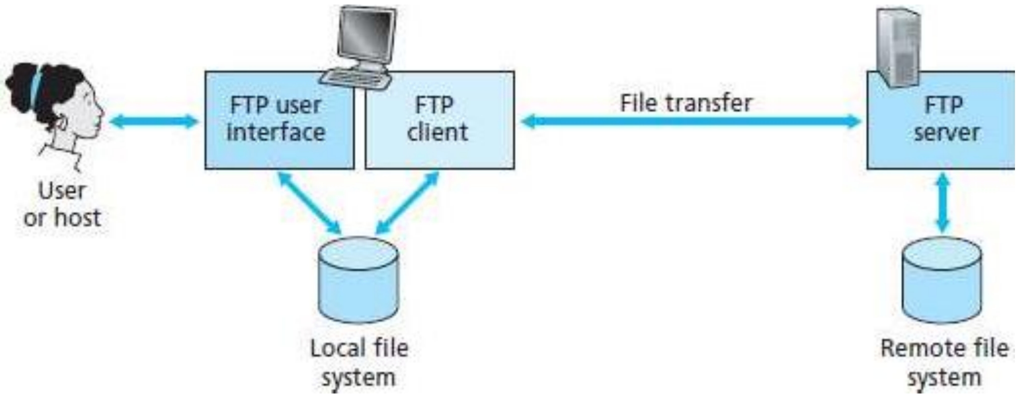
Suppose the object has not been modified since 7 Sep 2011 09:23:24. Then, fourth, the Web server sends a response message to the cache:

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)
(empty entity body)
```

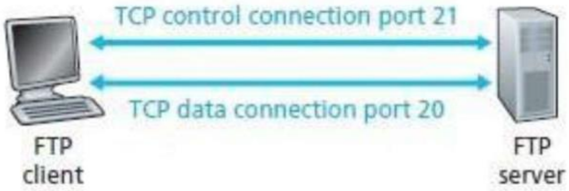
We see that in response to the conditional GET, the Web server still sends a response message but does not include the requested object in the response message.

File Transfer: FTP

- FTP is used for transferring file from one host to another host.
- In order for the user to access the remote account, the user must provide user identification and a password. After providing this authorization information, the user can transfer files from the local file system to the remote file system and vice versa.
- The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish a TCP connection with the FTP server process in the remote host.
- The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands.
- Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice versa).



- FTP uses two parallel TCP connections to transfer a file, a control connection and a data connection.
- The control connection is used for sending control information between the two hosts — information such as user identification, password, commands to change remote directory, and commands to "put" and "get" files.
- The data connection is used to actually send a file.



- When a user starts an FTP session with a remote host, the client side of FTP (user) first initiates a control TCP connection with the server side (remote host) on server port number 21.
- The client side of FTP sends the user identification and password over this control connection. The client side of FTP also sends, over the control connection, commands to change the remote directory.
- When the server side receives a command for a file transfer over the control connection (either to, or from, the remote host), the server side initiates a TCP data connection to the client side.

- FTP sends exactly one file over the data connection and then closes the data connection. If, during the same session, the user wants to transfer another file, FTP opens another data connection.
- Thus, with FTP, the control connection remains open throughout the duration of the user session, but a new data connection is created for each file transferred within a session (that is, the data connections are non-persistent).
- Throughout a session, the FTP server must maintain state about the user. In particular, the server must associate the control connection with a specific user account, and the server must keep track of the user's current directory as the user wanders about the remote directorytree.

FTP Commands and Replies

Some of the more common commands are given below:

• USER username: Used to send the user identification to the server.

• PASS password: Used to send the user password to the server.

• LIST: Used to ask the server to send back a list of all the files in the current remote directory. The list of files is sent over a (new and non-persistent) data connection rather than the control TCP connection.

• RETR filename: Used to retrieve (that is, get) a file from the current directory of the remote host. This command causes the remote host to initiate a data connection and to send the requested file over the data connection.

• STOR filename: Used to store (that is, put) a file into the current directory of the remote host.

Each command is followed by a reply, sent from server to client. The replies are three-digit numbers, with an optional message following the number.

• 331 Username OK, password required

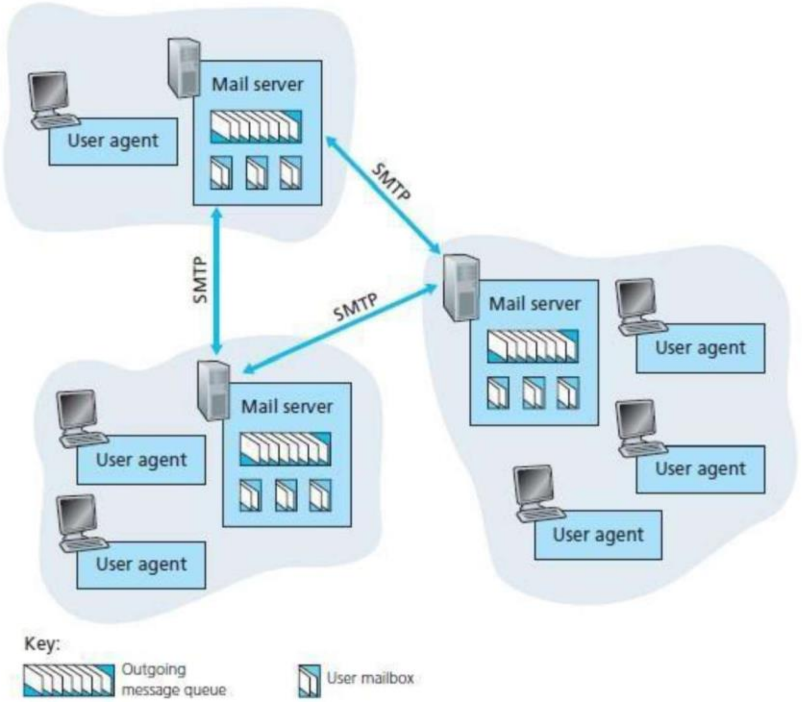
• 125 Data connection already open; transfer starting

• 425 Can't open data connection

• 452 Error writing file

Electronic Mail in the Internet

E-mail has three major components: user agents, mail servers, and the Simple Mail Transfer Protocol (SMTP).



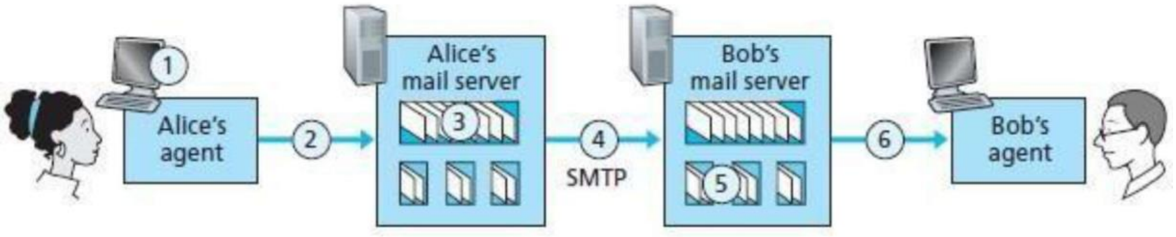
- **User agents** allow users to read, reply to, forward, save, and compose messages.
- **Mail servers** form the core of the e-mail infrastructure. Each recipient has a mailbox located in one of the mail servers. A typical message starts its journey in the sender's user agent, travels to the sender's mail server, and travels to the recipient's mail server, where it is deposited in the recipient's mailbox.
- **SMTP** is the principal application-layer protocol for Internet electronic mail. It uses the reliable data transfer service of TCP to transfer mail from the sender's mail server to the recipient's mail server. As with most application-layer protocols, SMTP has two sides: a client side, which executes on the sender's mail server, and a server side, which executes on the recipient's mail server.

SMTP

SMTP transfers messages from senders' mail servers to the recipients' mail servers. It restricts the body (not just the headers) of all mail messages to simple 7-bit ASCII.

Suppose Alice wants to send Bob a simple ASCII message.

1. Alice invokes her user agent for e-mail, provides Bob's e-mail address (for example, bob@someschool.edu), composes a message, and instructs the user agent to send the message.
2. Alice's user agent sends the message to her mail server, where it is placed in a message queue.
3. The client side of SMTP, running on Alice's mail server, sees the message in the message queue. It opens a TCP connection to an SMTP server, running on Bob's mail server.
4. After some initial SMTP handshaking, the SMTP client sends Alice's message into the TCP connection.
5. At Bob's mail server, the server side of SMTP receives the message. Bob's mail server then places the message in Bob's mailbox.
6. Bob invokes his user agent to read the message at his convenience.



An example transcript of messages exchanged between an SMTP client (C) and an SMTP server (S).

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Comparison with HTTP

HTTP	SMTP
Pull Protocol- someone loads information on a Web server and users use HTTP to pull the information from the server at their convenience.	Push Protocol- the sending mail server pushes the file to the receiving mail server.
HTTP does not mandates data to be in 7-bit ASCII format.	SMTP requires each message, including the body of each message, to be in 7-bit ASCII format.
HTTP encapsulates each object in its own HTTP response message.	Internet mail places all of the message's objects into one message.

Mail Message Formats

When an e-mail message is sent from one person to another, a header containing peripheral information precedes the body of the message.

The header lines and the body of the message are separated by a blank line.

Every header must have a From: header line and a To: header line; a header may include a Subject: header line as well as other optional header lines.

A typical message header looks like this:

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

Mail Access Protocols

SMTP protocol delivers the mail to the mail server. To fetch the mail from mail server receiver used mail access protocols.

There are currently a number of popular mail access protocols, including Post Office Protocol — Version 3 (POP3), Internet Mail Access Protocol (IMAP), and HTTP.

POP3

- POP3 is an extremely simple mail access protocol.
- POP3 begins when the user agent (the client) opens a TCP connection to the mail server (the server) on port 110.
- With the TCP connection established, POP3 progresses through three phases: authorization, transaction, and update.
- During the **authorization phase**, the user agent sends a username and a password to authenticate the user.
- During the **transaction phase**, the user agent retrieves messages; also during this phase, the user agent can mark messages for deletion, remove deletion marks, and obtain mail statistics.
- The update phase occurs after the client has issued the quit command, ending the POP3 session; at this time, the mail server deletes the messages that were marked for deletion.

- In a POP3 transaction, the user agent issues commands, and the server responds to each command with a reply. There are two possible responses: +OK used by the server to indicate that the previous command was fine; and -ERR, used by the server to indicate that something was wrong with the previous command.
- The authorization phase has two principal commands: user <username> and pass <password>.

```
user bob
+OK
pass hungry
+OK user successfully logged on
```

- A user agent using POP3 can often be configured (by the user) to "**download and delete**" or to "**download and keep** "
- In the download-and-delete mode, the user agent will issue the list, retr, and dele commands.

Ex:

```
C: list
S: 1 498
S: 2 912
S: .
```

```
C: retr 1
S: (blah blah ...
S: .................
S............blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: .................
S............blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

- A problem with this download-and-delete mode is that the recipient cannot access mail messages from multiple machines.
- In the download-and keep mode, the user agent leaves the messages on the mail server after downloading them. In this case, user can reread messages from different machines.

IMAP

- With POP3 access, once user has downloaded his messages to the local machine, he can create mail folders and move the downloaded messages into the folders. User can then delete messages, move messages across folders, and search for messages (by sender name or subject).
- But this paradigm—namely, folders and messages in the local machine—poses a problem for the nomadic user, who would prefer to maintain a folder hierarchy on a remote server that can be accessed from any computer. This is not possible with POP3—the POP3 protocol does not provide any means for a user to create remote folders and assign messages to folders.
- To solve this and other problems, the IMAP protocol was invented. Like POP3, IMAP is a mail access protocol. It has many more features than POP3, but it is also significantly more complex.
- An IMAP server will associate each message with a folder; when a message first arrives at the server, it is associated with the recipient's INBOX folder.
- The recipient can then move the message into a new, user-created folder, read the message, delete the message, and so on.
- The IMAP protocol provides commands to allow users to create folders and move messages from one folder to another.
- IMAP also provides commands that allow users to search remote folders for messages matching specific criteria.
- Another important feature of IMAP is that it has commands that permit a user agent to obtain components of messages. For example, a user agent can obtain just the message header of a message or just one part of a multipart MIME message. This feature is useful when there is a

low-bandwidth connection (for example, a slow-speed modem link) between the user agent and its mail server. With a low bandwidth connection, the user may not want to download all of the messages in its mailbox, particularly avoiding long messages that might contain, for example, an audio or video clip.

Web-Based E-Mail

More and more users today are sending and accessing their e-mail through their Web browsers. In this case user communicates with its remote mailbox via HTTP.

DNS—The Internet's Directory Service

- All the hosts connected to network is identified by IP address. But it is difficult for human beings to remember these IP address to access a particular host. Hence hosts are identified by hostnames. Ex: google.com
- But the routers require IP address to forward the packet.
- In order to map hostname with the IP address DNS is used.

Services Provided by DNS

- The DNS is (1) a distributed database implemented in a hierarchy of DNS servers, and (2) an application-layer protocol that allows hosts to query the distributed database.
- DNS is commonly employed by other application-layer protocols —including HTTP, SMTP, and FTP—to translate user-supplied hostnames to IP addresses.

Example:

Consider what happens when a browser running on some user's host, requests the URL www.someschool.edu/index.html.

In order for the user`s host to be able to send an HTTP request message to the Web server www.someschool.edu, the user's host must first obtain the IP address of www.someschool.edu. This is done as follows.

1. The same user machine runs the client side of the DNS application.

2. The browser extracts the hostname, www.someschool.edu, from the URL and passes the hostname to the client side of the DNS application.
3. The DNS client sends a query containing the hostname to a DNS server.
4. The DNS client eventually receives a reply, which includes the IP address for the hostname. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.

DNS provides a few other important services in addition to translating hostnames to IP addresses:

- **Host aliasing:** A host with a complicated hostname can have one or more alias names. For example, a hostname such as relay1.west-coast.enterprise.com could have, say, two aliases such as enterprise.com and www.enterprise.com. In this case, the hostname relay1.westcoast. enterprise.com is said to be a **canonical hostname.** Alias hostnames, when present, are typically more mnemonic than canonical hostnames. DNS can be invoked by an application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
- **Mail server aliasing:** For obvious reasons, it is highly desirable that e-mail addresses be mnemonic. For example, if Bob has an account with Hotmail, Bob's e-mail address might be as simple as bob@hotmail.com. However, the hostname of the Hotmail mail server is more complicated and much less mnemonic than simply hotmail.com (for example, the canonical

  Host name might be something like relay1.west-coast.hotmail.com). DNS can be invoked by a mail application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
- **Load distribution:** DNS is also used to perform load distribution among replicated servers, such as replicated Web servers. Busy sites, such as cnn.com, are replicated over multiple servers, with each server running on a different end system and each having a different IP address. For replicated Web servers, a set of IP addresses is thus associated with one canonical hostname. The DNS database contains this set of IP addresses. When clients make a DNS query for a name mapped to a set of addresses, the server responds with the entire set

of IP addresses, but rotates the ordering of the addresses within each reply. Because a client typically sends its HTTP request message to the IP address that is listed first in the set, DNS rotation distributes the traffic among the replicated servers.

Overview of How DNS Works

- Suppose that some application running in a user's host needs to translate a hostname to an IP address. The application will invoke the client side of DNS, specifying the hostname that needs to be translated.
- DNS in the user's host then takes over, sending a query message into the network.
- All DNS query and reply messages are sent within UDP datagrams to port 53. After a delay, ranging from milliseconds to seconds, DNS in the user's host receives a DNS reply message that provides the desired mapping. This mapping is then passed to the invoking application.

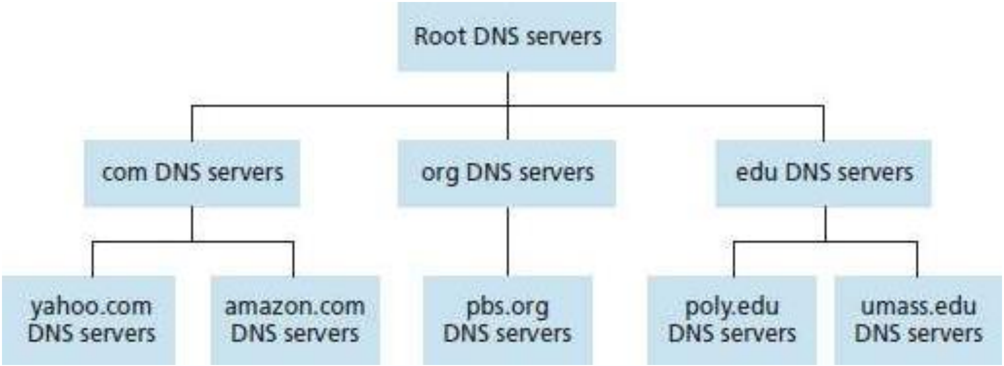
In this centralized design, clients simply direct all queries to the single DNS server, and the DNS server responds directly to the querying clients. Although the simplicity of this design is attractive, it is inappropriate for today's Internet, with its vast (and growing) number of hosts. The problems with a centralized design include:

- A single point of failure. If the DNS server crashes, so does the entire Internet!
- Traffic volume. A single DNS server would have to handle all DNS queries.
- Distant centralized database. A single DNS server cannot be "close to"all the querying clients. If we put the single DNS server in New York City, then all queries from Australia must travel to the other side of the globe, perhaps over slow and congested links. This can lead to significant delays.
- Maintenance. The single DNS server would have to keep records for all Internet hosts. Not only would this centralized database be huge, but it would have to be updated frequently to account for every new host.

A Distributed, Hierarchical Database

- In order to deal with the issue of scale, the DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world.

- There are three classes of DNS servers—root DNS servers, top-level domain (TLD) DNS servers, and authoritative DNS servers—organized in a hierarchy.

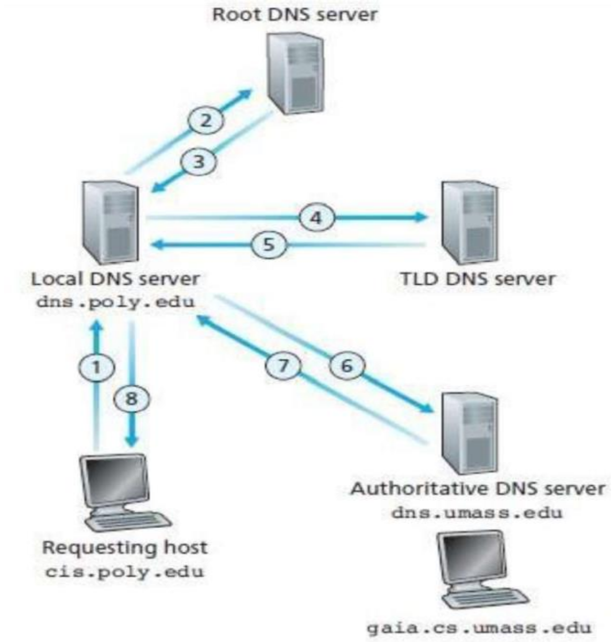


- **Root DNS servers.** In the Internet there are 13 root DNS servers (labeled A through M), most of which are located in North America.

  Although we have referred to each of the 13 root DNS servers as if it were a single server, each "server" is actually a network of replicated servers, for both security and reliability purposes. All together, there are 247 root servers.
- **Top-level domain (TLD) servers:** These servers are responsible for top-level domains such as com, org, net, edu, and gov, and all of the country top-level domains such as in,uk, fr, ca.
- **Authoritative DNS servers:** Every organization with publicly accessible hosts on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses. An organization's authoritative DNS server houses these DNS records.
- There is another important type of DNS server called the **local DNS server**. A local DNS server does not strictly belong to the hierarchy of servers but is nevertheless central to the DNS architecture. Each ISP —such as a university, an academic department, an employee's company, or a residential ISP—has a local DNS server.

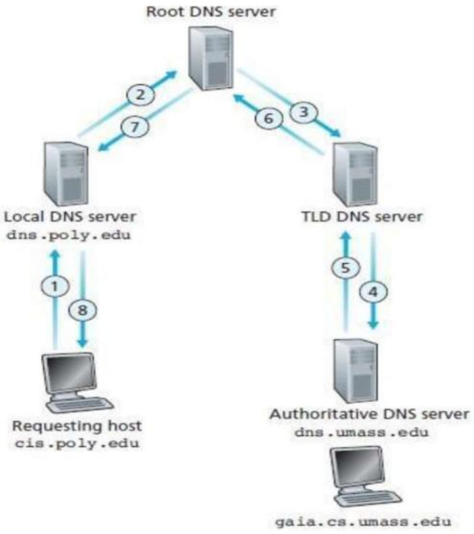
Two type of Interaction:

1) Recursive Queries:



Here DNS query is sent to local DNS server then to root server, then to TLD server and finally to authoritative DNS server. DNS response arrives in the reverse order.

2) Iterative Queries:



Here DNS query will be sent to Local DNS server, then to root server. Root server sends the IP address of TLD server. Now local DNS server sends query to TLD DNS server. TLD DNS server sends the IP address of authoritative DNS server to local DNS server. Now Local DNS server sends query to authoritative DNS server. Authoritative DNS server sends the IP address of host to local DNS server. Local DNS server sends it to the host.

DNS Caching

In a query chain, when a DNS server receives a DNS reply it can cache the mapping in its local memory.

If a hostname/IP address pair is cached in a DNS server and another query arrives to the

DNS server for the same hostname, the DNS server can provide the desired IP address, even if it is not authoritative for the hostname. Because hosts and mappings between hostnames and IP addresses are by no means permanent, DNS servers discard cached information after a period of time (often set to two days).

DNS Records and Messages

The DNS servers that together implement the DNS distributed database store **resource records** (RRs).

A resource record is a four-tuple that contains the following fields:

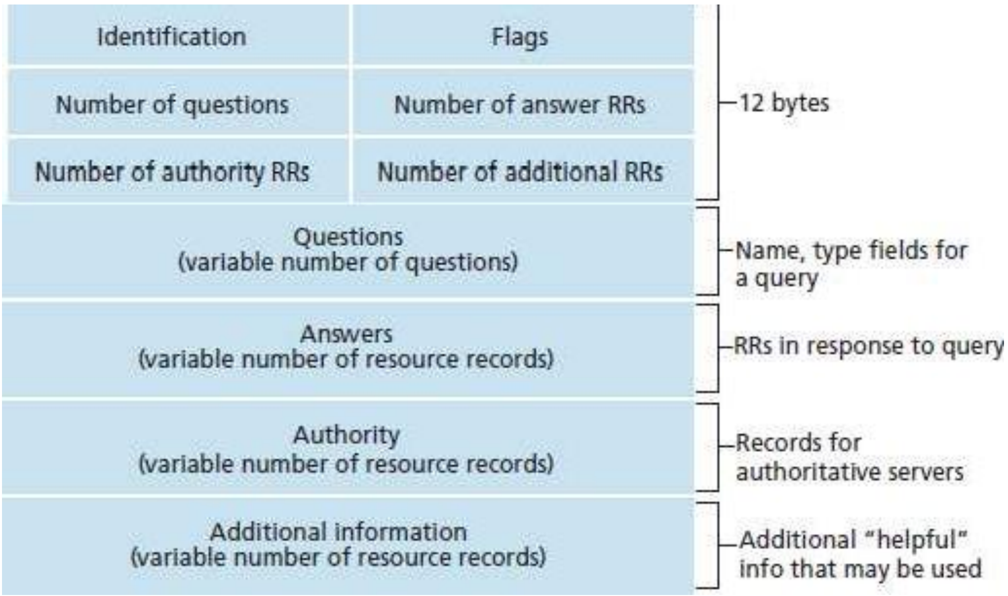
(Name, Value, Type, TTL)

TTL is the time to live of the resource record; it determines when a resource should be removed from a cache.

The meaning of Name and Value depend on Type:

- If Type=A, then Name is a hostname and Value is the IP address for the hostname.
- If Type=NS, then Name is a domain (such as foo.com) and Value is the hostname of an authoritative DNS server that knows how to obtain the IP addresses for hosts in thedomain.
- If Type=CNAME, then Value is a canonical hostname for the alias hostname Name. This record can provide querying hosts the canonical name for a hostname.
- If Type=MX, then Value is the canonical name of a mail server that has an aliashostname Name.

DNS Messages



Identification	Flags
Number of questions	Number of answer RRs
Number of authority RRs	Number of additional RRs
Questions (variable number of questions)	
Answers (variable number of resource records)	
Authority (variable number of resource records)	
Additional information (variable number of resource records)	

12 bytes
Name, type fields for a query
RRs in response to query
Records for authoritative servers
Additional "helpful" info that may be used

- The first 12 bytes is the header section, which has a number of fields.
- The first field is a 16-bit number that identifies the query. This identifier is copied into the reply message to a query, allowing the client to match received replies with sent queries.
- There are a number of flags in the flag field.

  A 1-bit query/reply flag indicates whether the message is a query (0) or a reply (1). A1-bit authoritative flag is set in a reply message when a DNS server is an authoritative server for a queried name.

  A 1-bit recursion-desired flag is set when a client (host or DNS server) desires that the DNS server perform recursion when it doesn't have the record.

  A 1-bit recursion available field is set in a reply if the DNS server supports recursion.
- In the header, there are also four number-of fields. These fields indicate the number of occurrences of the four types of data sections that follow the header.
- The **question** section contains information about the query that is being made. This section includes (1) a name field that contains the name that is being queried, and (2) a type field that indicates the type of question being asked about the name

- In a reply from a DNS server, the **answer** section contains the resource records for the name that was originally queried.
- The **authority** section contains records of other authoritative servers.
- The **additional** section contains other helpful records.

Inserting Records into the DNS Database

Suppose you have just created an exciting new startup company called Network Utopia. The first thing you'll surely want to do is register the domain name networkutopia.com at a registrar. A registrar is a commercial entity that verifies the uniqueness of the domain name, enters the domain name into the DNS database (as discussed below), and collects a small fee from you for its services.

For the primary authoritative server for networkutopia.com, the registrar would insert the following two resource records into the DNS system:

(networkutopia.com, dns1.networkutopia.com, NS)

(dns1.networkutopia.com, 212.212.212.1, A)

Peer-to-Peer Applications

In P2P architecture, there is minimal (or no) reliance on always-on infrastructure servers. Instead, pairs of intermittently connected hosts, called peers, communicate directly with each other.

P2P File Distribution

- In P2P file distribution, each peer can redistribute any portion of the file it has received to any other peers, thereby assisting the server in the distribution process.
- The most popular P2P file distribution protocol is BitTorrent.

Scalability of P2P Architectures

As shown in below Figure the server and the peers are connected to the Internet with access links. Denote the upload rate of the server's access link by u_s, the upload rate of the ith peer's

K.S SCHOOL OF ENGINEERING AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Attendance for Remedial Class

YEAR / SEMESTER	III / V B
COURSE TITLE	Computer Networks
COURSE CODE	21CS52

S.NO	USN	NAME	Signature
1	1KG21CS065	MANYA R	Manya
2	1KG21CS066	MEGHANA G	Meghana.G
3	1KG21CS071	NANNURU JASWANTH	Jaswanth
4	1KG21CS073	NAVEENKANTH S	
5	1KG21CS079	NITHIN NAIDU S P	NithinNaidu
6	1KG21CS080	P HEMANTH KUMAR	
7	1KG21CS082	PATIL ADESH WAMANRAO	
8	1KG21CS090	RAMKI G K L	
9	1KG21CS094	S SAI KIRAN	Sai
10	1KG21CS097	SAHIL CHALWA	
11	1KG21CS098	SAIRAMA NAIDU MALLARAPU	
12	1KG21CS100	SANJANA U REVANKAR	Sanjana.U.R
13	1KG21CS104	SIRISHA T	
14	1KG21CS110	T CHAITANYA KRISHNA	
15	1KG21CS111	TANISH JAIN	
16	1KG21CS113	UDAY L	Uday
17	1KG21CS116	VAMSHI KRISHNA R	
18	1KG21CS117	VARSHA K R	Varsha KR
19	1KG21CS122	VISHWAS K R	
20	1KG21CS126	YUVARAJ S	
21	1KG22CS406	KIRAN KUMAR. N	Kiran Kumar
22	1KG22CS407	MANOJ M K	ManoJMK
23	1KG22CS408	NABEEL BAIG	

Sushmitha
Faculty Signature

HOD Signature

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K.S SCHOOL OF ENGINEERING AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE & ENGG.
Attendance for Remedial Class

YEAR / SEMESTER	III / V B
COURSE TITLE	Computer Networks
COURSE CODE	21CS52

S.NO	USN	NAME	Signature
1	1KG21CS065	MANYA R	Manya
2	1KG21CS066	MEGHANA G	Meghana.G
3	1KG21CS071	NANNURU JASWANTH	Jaswanth
4	1KG21CS074	NEHA U	U.Neha
5	1KG21CS080	P HEMANTH KUMAR	
6	1KG21CS088	R H HEMANTH KUMAR	R.H.Hemanth Kumar.
7	1KG21CS091	RAMU T N	
8	1KG21CS092	REKHA RATHOD	Rathod
9	1KG21CS098	SAIRAMA NAIDU MALLARAPU	
10	1KG21CS099	SANJANA S VASISTA	Sanjana
11	1KG21CS104	SIRISHA T	Sirisha.
12	1KG21CS111	TANISH JAIN	
13	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	T.Giridhar
14	1KG21CS113	UDAY L	Uday
15	1KG21CS114	VAIBHAV ES	
16	1KG21CS116	VAMSHI KRISHNA R	Vamsi
17	1KG21CS117	VARSHA K R	Varsha KR
18	1KG21CS119	VINAY M	Vinay
19	1KG21CS122	VISHWAS K R	
20	1KG21CS123	YASHU V D	Yashu
21	1KG22CS407	MANOJ M K	
22	1KG22CS411	SHASHANK D D	

Sushmitha
Faculty Signature

HOD Signature

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K.S SCHOOL OF ENGINEERING AND MANAGEMENT

KSSEM

DEPARTMENT OF COMPUTER SCIENCE & ENGG.

Advanced Learners List

YEAR / SEMESTER - V - 'B'

COURSE TITLE -| COMPUTER NETWORKS

COURSE CODE -21CS52

S.NO	USN	NAME	Signature
1	1KG21CS064	MANASVI K M	Manasvi KM
2	1KG21CS067	MOHITH KUMAR Y V	Mohith
3	1KG21CS068	MYTHRI Y	MythriY
4	1KG21CS072	NAVACHANDU N	NavachanduN
5	1KG21CS075	NIKITHA L	Nikitha.L
6	1KG21CS085	PRAKRIT R ARITAS	R Arit
7	1KG21CS086	PRANAVA S BHAT	Prana S
8	1KG21CS102	SINCHANA S	Sinchana.S
9	1KG21CS108	SURAVI H U	Suv
10	1KG21CS125	YASHWIN KUMAR R	Yashwin Kumar

Sushmitha
Faculty Signature

HOD Signature

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

KSSEM

K.S SCHOOL OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER SCIENCE & ENGG.

Advanced Learners List

YEAR / SEMESTER - V - 'B'

COURSE TITLE - COMPUTER NETWORKS

COURSE CODE -21CS52

S.NO	USN	NAME	Signature
1	1KG21CS064	MANASVI K M	Manasvi K M
2	1KG21CS067	MOHITH KUMAR Y V	MohithKYV
3	1KG21CS068	MYTHRI Y	Mythri Y
4	1KG21CS072	NAVACHANDU N	Navchandu N
5	1KG21CS074	NEHA U	Neha
6	1KG21CS075	NIKITHA L	Nikitha L
7	1KG21CS077	NILESH RAJAN	Rajan
8	1KG21CS081	PACHA BHARADWAZ	
9	1KG21CS084	POSINA LIKHITHA	likitha
10	1KG21CS086	PRANAVA S BHAT	
11	1KG21CS089	RACHANA R	Rachana R
12	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	Sahebbg
13	1KG21CS101	SHARMILA K P	
14	1KG21CS108	SURAVI H U	
15	1KG21CS115	VAISHNAVI U KULKARNI	Vaishnavi
16	1KG21CS125	YASHWIN KUMAR R	

Sushmitha
Faculty Signature

HOD Signature

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU-560109
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
V SEM B sec Assignment Marks List - CN

SL.NO	USN	NAME	Assignment 1	Assignment 2	Total	Signature
1	1KG21CS064	MANASVI K M	10	10	20	Manasvi
2	1KG21CS065	MANYA R	10	10	20	Manya
3	1KG21CS066	MEGHANA G	10	10	20	
4	1KG21CS067	MOHITH KUMAR Y V	10	10	20	
5	1KG21CS068	MYTHRI Y	10	10	20	mythri
6	1KG21CS069	NAMITH U	10	10	20	Namith
7	1KG21CS070	NANDANA M	10	10	20	Nandana M
8	1KG21CS071	NANNURU JASWANTH	10	10	20	Jaswanth
9	1KG21CS072	NAVACHANDU N	10	10	20	Navachandu N
10	1KG21CS073	NAVEENKANTH S	10	10	20	
11	1KG21CS074	NEHA U	10	10	20	U Neha
12	1KG21CS075	NIKITHA L	10	10	20	Nikitha L
13	1KG21CS076	NIKITHA N M	10	10	20	Nikitha
14	1KG21CS077	NILESH RAJAN	10	10	20	Rajan
15	1KG21CS078	NITHIN B	10	10	20	
16	1KG21CS079	NITHIN NAIDU S P	10	10	20	Nith
17	1KG21CS080	P HEMANTH KUMAR	10	10	20	
18	1KG21CS081	PACHA BHARADWAZ	10	10	20	
19	1KG21CS082	PATIL ADESH WAMANRAO	10	10	20	
20	1KG21CS083	PAVAN R	10	10	20	Pavan.R
21	1KG21CS084	POSINA LIKHITHA	10	10	20	Likhitha
22	1KG21CS085	PRAKRIT R ARITAS	10	10	20	
23	1KG21CS086	PRANAVA S BHAT	10	10	20	
24	1KG21CS087	PRERANA B	10	10	20	
25	1KG21CS088	R H HEMANTH KUMAR	10	10	20	R.H Hemanth Kumar.
26	1KG21CS089	RACHANA R	10	10	20	Rachana R
27	1KG21CS090	RAMKI G K L	10	10	20	
28	1KG21CS091	RAMU T N	10	10	20	
29	1KG21CS092	REKHA RATHOD	10	10	20	
30	1KG21CS093	ROHIT L	10	10	20	Rohit
31	1KG21CS094	S SAI KIRAN	10	10	20	
32	1KG21CS095	SAGAR B	10	10	20	
33	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	10	10	20	
34	1KG21CS097	SAHIL CHALWA	10	10	20	
35	1KG21CS098	SAIRAMA NAIDU MALLARAPU	10	10	20	
36	1KG21CS099	SANJANA S VASISTA	10	10	20	
37	1KG21CS100	SANJANA U REVANKAR	10	10	20	Sanjana UR

38	1KG21CS101	SHARMILA K P	10	10	20	
39	1KG21CS102	SINCHANA S	10	10	20	
40	1KG21CS103	SIRISHA M	10	10	20	S.M
41	1KG21CS104	SIRISHA T	10	10	20	
42	1KG21CS106	SUDARSHAN S KAKALWAR	10	10	20	
43	1KG21CS108	SURAVI H U	10	10	20	
44	1KG21CS109	SWATHI S	10	10	20	Swathi.S.
45	1KG21CS110	T CHAITANYA KRISHNA	10	10	20	
46	1KG21CS111	TANISH JAIN	10	10	20	
47	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	10	10	20	T.Giridhar
48	1KG21CS113	UDAY L	10	10	20	
49	1KG21CS114	VAIBHAV ES	10	10	20	
50	1KG21CS115	VAISHNAVI U KULKARNI	10	10	20	Vaishnavi
51	1KG21CS116	VAMSHI KRISHNA R	10	10	20	
52	1KG21CS117	VARSHA K R	10	10	20	VarshaKR
53	1KG21CS118	VEEKSHITH V	10	10	20	
54	1KG21CS119	VINAY M	10	10	20	Vinay
55	1KG21CS120	VINEETHA N	10	10	20	
56	1KG21CS121	VINUTHA C M	10	10	20	
57	1KG21CS122	VISHWAS K R	10	10	20	
58	1KG21CS123	YASHU V D	10	10	20	Yashu
59	1KG21CS124	YASHWANTH SAIBABA H	10	10	20	
60	1KG21CS125	YASHWIN KUMAR R	10	10	20	Yash
61	1KG21CS126	YUVARAJ S	10	10	20	Yuvaraj
62	1KG22CS406	KIRAN KUMAR N	10	10	20	Kiran Kumar
63	1KG22CS407	MANOJ M K	10	10	20	ManojMK
64	1KG22CS408	NABEEL BAIG	10	10	20	Nab
65	1KG22CS409	RAKESH KS	10	10	20	
66	1KG22CS410	RAKESH M	10	10	20	
67	1KG22CS411	SHASHANK D D	10	10	20	

Sushmitha
Faculty Signature

HOD Signature

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

KSSEM

K.S SCHOOL OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Consolidated IA Marks - CN 5th 'B' Sec

Sl. No.	USN	STUDENT NAME	IA1 (20)	IA2 (20)	IA3 (20)	ASSIGNMENT (20)	TOTAL (80)	TOTAL (30)	LAB MARKS (20)	TOTAL (50)	Signature
1	1KG21CS064	MANASVI K M	19	17	17	20	73	28	19	47	Manasvi
2	1KG21CS065	MANYA R	9	8	14	20	51	20	17	37	Manya
3	1KG21CS066	MEGHANA G	3	3	6	20	32	12	16	28	
4	1KG21CS067	MOHITH KUMAR Y V	16	16	7	20	59	23	20	43	
5	1KG21CS068	MYTHRI Y	17	17	19	20	73	28	19	47	Mythri
6	1KG21CS069	NAMITH U	6	0	4	20	30	12	15	27	Namith
7	1KG21CS070	NANDANA M	12	13	2	20	47	18	18	36	Nandana M
8	1KG21CS071	NANNURU JASWANTH	3	7	0	20	30	12	16	28	Jaswanth
9	1KG21CS072	NAVACHANDU N	17	20	17	20	74	28	19	47	Navachandu N
10	1KG21CS073	NAVEENKANTH S	11	9	9	20	49	19	18	37	
11	1KG21CS074	NEHA U	7	15	11	20	53	20	18	38	U-Neha
12	1KG21CS075	NIKITHA L	15	18	11	20	64	24	20	44	Nikitha.L
13	1KG21CS076	NIKITHA N M	14	14	9	20	57	22	18	40	Nikitha
14	1KG21CS077	NILESH RAJAN	13	16	10	20	59	23	20	43	
15	1KG21CS078	NITHIN B	14	10	8	20	52	20	19	39	NithinB
16	1KG21CS079	NITHIN NAIDU S P	6	4	0	20	30	12	16	28	
17	1KG21CS080	P HEMANTH KUMAR	6	8	2	20	36	14	18	32	
18	1KG21CS081	PACHA BHARADWAZ	14	16	8	20	58	22	18	40	

19	1KG21CS082	PATIL ADESH WAMANRAO	0	8	4	20	32	12	17	29	
20	1KG21CS083	PAVAN R	14	10	8	20	52	20	19	39	Pavan.R
21	1KG21CS084	POSINA LIKHITHA	12	16	15	20	63	24	19	43	
22	1KG21CS085	PRAKRIT R ARITAS	17	0	10	20	47	18	19	37	
23	1KG21CS086	PRANAVA S BHAT	17	15	11	20	63	24	20	44	
24	1KG21CS087	PRERANA B	0	10	10	20	40	15	18	33	
25	1KG21CS088	R H HEMANTH KUMAR	3	11	8	20	42	16	19	35	
26	1KG21CS089	RACHANA R	11	17	12	20	60	23	20	43	Rachana R
27	1KG21CS090	RAMKI G K L	12	8	8	20	48	18	20	38	
28	1KG21CS091	RAMU T N	6	14	8	20	48	18	17	35	
29	1KG21CS092	REKHA RATHOD	7	11	8	20	46	18	17	35	Rathod
30	1KG21CS093	ROHIT L	14	10	13	20	57	22	19	41	Rohit
31	1KG21CS094	S SAI KIRAN	4	2	5	20	31	12	16	28	
32	1KG21CS095	SAGAR B	10	11	13	20	54	21	18	39	
33	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	12	15	7	20	54	21	18	39	
34	1KG21CS097	SAHIL CHALWA	4	1	6	20	31	12	12	24	
35	1KG21CS098	SAIRAMA NAIDU MALLARAPU	3	1	9	20	33	13	19	32	
36	1KG21CS099	SANJANA S VASISTA	6	13	10	20	49	19	17	36	
37	1KG21CS100	SANJANA U REVANKAR	10	6	3	20	39	15	17	32	Sanjana.U.R

38	1KG21CS101	SHARMILA K P	14	15	13	20	62	24	17	41	
39	1KG21CS102	SINCHANA S	15	12	9	20	56	21	18	39	
40	1KG21CS103	SIRISHA M	12	12	8	20	52	20	18	38	M
41	1KG21CS104	SIRISHA T	7	9	11	20	47	18	19	37	
42	1KG21CS106	SUDARSHAN S KAKALWAR	0	10	8	20	38	15	17	32	
43	1KG21CS108	SURAVI H U	17	18	17	20	72	27	20	47	
44	1KG21CS109	SWATHI S	10	11	15	20	56	21	18	39	Swathi S
45	1KG21CS110	T CHAITANYA KRISHNA	4	2	4	20	30	12	17	29	
46	1KG21CS111	TANISH JAIN	8	5	7	20	40	15	17	32	
47	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	6	10	6	20	42	16	16	32	
48	1KG21CS113	UDAY L	6	7	9	20	42	16	18	34	Uday
49	1KG21CS114	VAIBHAV ES	5	13	3	20	41	16	18	34	
50	1KG21CS115	VAISHNAVI U KULKARNI	14	16	10	20	60	23	20	43	Vaishnavi
51	1KG21CS116	VAMSHI KRISHNA R	6	5	3	20	34	13	18	31	
52	1KG21CS117	VARSHA K R	5	1	5	20	31	12	17	29	Varsha
53	1KG21CS118	VEEKSHITH V	12	13	11	20	56	21	19	40	
54	1KG21CS119	VINAY M	8	12	6	20	46	18	16	34	Vinay
55	1KG21CS120	VINEETHA N	12	13	7	20	52	20	18	38	
56	1KG21CS121	VINUTHA C M	14	12	4	20	50	19	18	37	
57	1KG21CS122	VISHWAS K R	1	7	5	20	33	13	18	31	
58	1KG21CS123	YASHU V D	7	11	5	20	43	17	18	35	

59	1KG21CS124	YASHWANTH SAIBABA H	11	0	7	20	38	15	18	33	Yashwanth
60	1KG21CS125	YASHWIN KUMAR R	18	19	0	20	57	22	19	41	
61	1KG21CS126	YUVARAJ S	11	7	4	20	42	16	18	34	
62	1KG22CS406	KIRAN KUMAR. N	0	9	5	20	34	13	17	30	Kiran Kumar
63	1KG22CS407	MANOJ M K	3	5	2	20	30	12	17	29	
64	1KG22CS408	NABEEL BAIG	0	8	5	20	33	13	18	31	
65	1KG22CS409	RAKESH K S	14	10	15	20	59	23	19	42	
66	1KG22CS410	RAKESH M	0	13	8	20	41	16	18	34	
67	1KG22CS411	SHASHANK D D	8	12	7	20	47	18	18	36	

Sushmitha
Faculty Incharge

HOD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT- 560 109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Computer NetworkS Laboaratory (CN LAB)

SUBJECT CODE : 21CS52

SEMESTER V / SECTION B

BATCH B1

SL NO.	USN	NAME	SIGNATURE
1	1KG21CS064	MANASVI K M	Manasvi
2	1KG21CS065	MANYA R	Manya
3	1KG21CS066	MEGHANA G	Meghana.G
4	1KG21CS067	MOHITH KUMAR Y V	Mohit K
5	1KG21CS068	MYTHRI Y	Mythri.Y
6	1KG21CS069	NAMITH U	Namith.U
7	1KG21CS070	NANDANA M	Nandana M
8	1KG21CS071	NANNURU JASWANTH	Jaswantha
9	1KG21CS072	NAVACHANDU N	Navachandu.N
10	1KG21CS073	NAVEENKANTH S	
11	1KG21CS074	NEHA U	U·Neha
12	1KG21CS075	NIKITHA L	Nikitha.L
13	1KG21CS076	NIKITHA N M	Nikitha
14	1KG21CS077	NILESH RAJAN	
15	1KG21CS078	NITHIN B	Nithin B
16	1KG21CS079	NITHIN NAIDU S P	
17	1KG21CS080	P HEMANTH KUMAR	
18	1KG21CS081	PACHA BHARADWAZ	
19	1KG21CS082	PATIL ADESH WAMANRAO	
20	1KG21CS083	PAVAN R	Pavan, R
21	1KG21CS084	POSINA LIKHITHA	Likhitha
22	1KG21CS085	PRAKRIT R ARITAS	
23	1KG21CS086	PRANAVA S BHAT	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT- 560 109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Computer NetworkS Laboaratory (CN LAB)
SUBJECT CODE : 21CS52
SEMESTER V / SECTION B
BATCH B2

SL NO.	USN	NAME	SIGNATURE
1	1KG21CS087	PRERANA B	
2	1KG21CS088	R H HEMANTH KUMAR	R H Hemanth Kumar.
3	1KG21CS089	RACHANA R	RachanaR.
4	1KG21CS090	RAMKI G K L	GRKL
5	1KG21CS091	RAMU T N	Ramu TN
6	1KG21CS092	REKHA RATHOD	
7	1KG21CS093	ROHIT L	Rohit L
8	1KG21CS094	S SAI KIRAN	
9	1KG21CS095	SAGAR B	
10	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	
11	1KG21CS097	SAHIL CHALWA	— AB —
12	1KG21CS098	SAIRAMA NAIDU MALLARAPU	
13	1KG21CS099	SANJANA S VASISTA	
14	1KG21CS100	SANJANA U REVANKAR	
15	1KG21CS101	SHARMILA K P	
16	1KG21CS102	SINCHANA S	
17	1KG21CS103	SIRISHA M	
18	1KG21CS104	SIRISHA T	
19	1KG21CS106	SUDARSHAN S KAKALWAR	
20	1KG21CS108	SURAVI H U	Suravi
21	1KG21CS109	SWATHI S	Swathi S
22	1KG21CS110	T CHAITANYA KRISHNA	
23	1KG21CS111	TANISH JAIN	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT- 560 109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Computer NetworkS Laboaratory (CN LAB)
SUBJECT CODE : 21CS52

SEMESTER V / SECTION B

BATCH B3

SL NO.	USN	NAME	SIGNATURE
1	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	T-Giridhar
2	1KG21CS113	UDAY L	Uday
3	1KG21CS114	VAIBHAV ES	
4	1KG21CS115	VAISHNAVI U KULKARNI	Vaishnavi
5	1KG21CS116	VAMSHI KRISHNA R	
6	1KG21CS117	VARSHA K R	Varsha KR
7	1KG21CS118	VEEKSHITH V	
8	1KG21CS119	VINAY M	Vinay
9	1KG21CS120	VINEETHA N	Vin N
10	1KG21CS121	VINUTHA C M	Vinutha
11	1KG21CS122	VISHWAS K R	
12	1KG21CS123	YASHU V D	
13	1KG21CS124	YASHWANTH SAIBABA H	
14	1KG21CS125	YASHWIN KUMAR R	
15	1KG21CS126	YUVARAJ S	Yuvaraj
16	1KG22CS406	KIRAN KUMAR. N	Kiran Kumar
17	1KG22CS407	MANOJ M K	Manoj MK
18	1KG22CS408	NABEEL BAIG	
19	1KG22CS409	RAKESAH K S	
20	1KG22CS410	RAKESH M	R.M
21	1KG22CS411	SHASHANK D D	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K. S. SCHOOL OF ENGINEERING AND MANAGEMENT- 560 109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Computer Networks Laboaratory (CN LAB)
SUBJECT CODE : 21CS52
SEMESTER V / SECTION B

BATCH - B1

DATE : 05:03:2024
TIME : 12:30 PM TO 3:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
1	1KG21CS064	MANASVI K M	5	4	5	14	4	30	15	19	Manasvi
2	1KG21CS065	MANYA R	0	4	2	6	2	30	15	17	Manya
3	1KG21CS066	MEGHANA G	2	2	1	5	1	30	15	16	Meghana. G.
4	1KG21CS067	MOHITH KUMAR Y V	5	10	4	19	5	30	15	20	
5	1KG21CS068	MYTHRI Y	0	10	4	14	4	30	15	19	Mythri. Y..
6	1KG21CS069	NAMITH U	0	0	0	0	0	30	15	15	Namith
7	1KG21CS070	NANDANA M	5	5	0	10	3	30	15	18	Nandana. M
8	1KG21CS071	NANNURU JASWANTH	0	4	1	5	1	30	15	16	Jaswanth.
9	1KG21CS072	NAVACHANDU N	5	4	5	14	4	30	15	19	Navachandu. N.
10	1KG21CS073	NAVEENKANTH S	0	10	2	12	3	30	15	18	
11	1KG21CS074	NEHA U	0	10	2	12	3	30	15	18	U.Neha
12	1KG21CS075	NIKITHA L	5	10	3	18	5	30	15	20	Nikitha.L
13	1KG21CS076	NIKITHA N M	4	4	3	11	3	30	15	18	Nikitha
14	1KG21CS077	NILESH RAJAN	5	10	4	19	5	30	15	20	NRajan
15	1KG21CS078	NITHIN B	5	10	2	17	4	30	15	19	
16	1KG21CS079	NITHIN NAIDU S P	3	0	2	5	1	30	15	16	Nith
17	1KG21CS080	P HEMANTH KUMAR	4	4	2	10	3	30	15	18	
18	1KG21CS081	PACHA BHARADWAZ	4	4	3	11	3	30	15	18	
19	1KG21CS082	PATIL ADESH WAMANRAO	0	8	0	8	2	30	15	17	

BATCH - B1

DATE : 05:03:2024
TIME : 12:30 PM TO 3:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
20	1KG21CS083	PAVAN R	5	10	2	17	4	30	15	19	Pavan. R
21	1KG21CS084	POSINA LIKHITHA	5	10	2	17	4	30	15	19	Likhitha
22	1KG21CS085	PRAKRIT R ARITAS	5	10	2	17	4	30	15	19	
23	1KG21CS086	PRANAVA S BHAT	5	10	3	18	5	30	15	20	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

S. SCHOOL OF ENGINEERING AND MANAGEMENT- 109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Computer Networks Laboaratory (CN LAB)
SUBJECT CODE : 21CS52
SEMESTER V / SECTION B

BATCH - B2

DATE : 09:03:2024
TIME : 09:00 AM TO 12:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
1	1KG21CS087	PRERANA B	2	4	4	10	3	30	15	18	
2	1KG21CS088	R H HEMANTH KUMAR	5	10	1	16	4	30	15	19	R.H Hemanth Kumar.
3	1KG21CS089	RACHANA R	5	10	4	19	5	30	15	20	Rachana.R
4	1KG21CS090	RAMKI G K L	5	10	5	20	5	30	15	20	GKL
5	1KG21CS091	RAMU T N	0	4	2	6	2	30	15	17	Ramu
6	1KG21CS092	REKHA RATHOD	0	4	2	6	2	30	15	17	
7	1KG21CS093	ROHIT L	0	10	4	14	4	30	15	19	Rohit
8	1KG21CS094	S SAI KIRAN	0	10	0	10	3	26	13	16	
9	1KG21CS095	SAGAR B	0	10	3	13	3	30	15	18	
10	1KG21CS096	SAHEBAGOUD MALLAPPA DANAGOND	0	10	3	13	3	30	15	18	
11	1KG21CS097	SAHIL CHALWA	0	0	0	0	0	24	12	12	
12	1KG21CS098	SAIRAMA NAIDU MALLARAPU	5	10	0	15	4	30	15	19	
13	1KG21CS099	SANJANA S VASISTA	2	4	2	8	2	30	15	17	Sanjana
14	1KG21CS100	SANJANA U REVANKAR	2	4	2	8	2	30	15	17	Sanjana.U.R
15	1KG21CS101	SHARMILA K P	0	5	3	8	2	30	15	17	
16	1KG21CS102	SINCHANA S	0	10	1	11	3	30	15	18	
17	1KG21CS103	SIRISHA M	0	10	1	11	3	30	15	18	Sirisha.M
18	1KG21CS104	SIRISHA T	5	10	0	15	4	30	15	19	
19	1KG21CS106	SUDARSHAN S KAKALWAR	0	5	1	6	2	30	15	17	

BATCH - B2

DATE : 09:03:2024
TIME : 09:00 AM TO 12:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
20	1KG21CS108	SURAVI H U	5	10	4	19	5	30	15	20	
21	1KG21CS109	SWATHI S	3	4	3	10	3	30	15	18	Swathi S
22	1KG21CS110	T CHAITANYA KRISHNA	3	3	0	6	2	30	15	17	
23	1KG21CS111	TANISH JAIN	0	5	4	9	2	30	15	17	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K SCHOOL OF ENGINEERING AND MANAGEMENT- 109
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Computer Networks Laboaratory (CN LAB)
SUBJECT CODE : 21CS52
SEMESTER V / SECTION B

BATCH - B3

DATE : 07:03:2024
TIME : 12:30 PM TO 3:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
1	1KG21CS112	TATHI REDDY GIRIDHAR REDDY	0	4	1	5	1	30	15	16	T. Giridhar
2	1KG21CS113	UDAY L	4	4	3	11	3	30	15	18	Uday
3	1KG21CS114	VAIBHAV ES	4	4	2	10	3	30	15	18	
4	1KG21CS115	VAISHNAVI U KULKARNI	5	10	4	19	5	30	15	20	Vaishnavi
5	1KG21CS116	VAMSHI KRISHNA R	0	10	3	13	3	30	15	18	Vamsi
6	1KG21CS117	VARSHA K R	3	4	2	9	2	30	15	17	VARSHA KR
7	1KG21CS118	VEEKSHITH V	5	10	1	16	4	30	15	19	
8	1KG21CS119	VINAY M	0	4	1	5	1	30	15	16	Vinay
9	1KG21CS120	VINEETHA N	0	10	3	13	3	30	15	18	
10	1KG21CS121	VINUTHA C M	4	4	3	11	3	30	15	18	
11	1KG21CS122	VISHWAS K R	0	10	1	11	3	30	15	18	
12	1KG21CS123	YASHU V D	0	10	2	12	3	30	15	18	Yashu
13	1KG21CS124	YASHWANTH SAIBABA H	0	10	2	12	3	30	15	18	
14	1KG21CS125	YASHWIN KUMAR R	0	10	4	14	4	30	15	19	Yash
15	1KG21CS126	YUVARAJ S	0	10	2	12	3	30	15	18	
16	1KG22CS406	KIRAN KUMAR N	0	5	3	8	2	30	15	17	Kiran Kumar
17	1KG22CS407	MANOJ M K	0	4	3	7	2	30	15	17	
18	1KG22CS408	NABEEL BAIG	0	10	2	12	3	30	15	18	

BATCH - B3

DATE : 07:03:2024
TIME : 12:30 PM TO 3:30 PM

SL NO.	USN	NAME	Procedure (5)	Execution (10)	Viva (5)	Total (20)	Scale down (5)	Record + Observation (15+15)	Average (15)	Final Total (20)	Signature
19	1KG22CS409	RAKESAH K S	0	10	4	14	4	30	15	19	Rakesh.K.S
20	1KG22CS410	RAKESH M	0	10	1	11	3	30	15	18	Rakesh.M
21	1KG22CS411	SHASHANK D D	0	10	0	10	3	30	15	18	

Faculty Incharge

HoD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BANGALORE

Branch : CS Semester : 5

Sl NO.	USN	21CS52
1	1KG21CS001	21 (**TH**) , 19 (**PR**)
2	1KG21CS002	15 (**TH**) , 19 (**PR**)
3	1KG21CS003	23 (**TH**) , 18 (**PR**)
4	1KG21CS004	19 (**TH**) , 19 (**PR**)
5	1KG21CS005	29 (**TH**) , 19 (**PR**)
6	1KG21CS006	12 (**TH**) , 16 (**PR**)
7	1KG21CS007	16 (**TH**) , 16 (**PR**)
8	1KG21CS008	25 (**TH**) , 19 (**PR**)
9	1KG21CS009	20 (**TH**) , 19 (**PR**)
10	1KG21CS010	24 (**TH**) , 18 (**PR**)
11	1KG21CS011	16 (**TH**) , 18 (**PR**)
12	1KG21CS012	19 (**TH**) , 18 (**PR**)
13	1KG21CS013	23 (**TH**) , 18 (**PR**)
14	1KG21CS014	24 (**TH**) , 18 (**PR**)
15	1KG21CS015	12 (**TH**) , 16 (**PR**)
16	1KG21CS016	16 (**TH**) , 18 (**PR**)
17	1KG21CS017	24 (**TH**) , 19 (**PR**)
18	1KG21CS018	19 (**TH**) , 18 (**PR**)
19	1KG21CS019	20 (**TH**) , 17 (**PR**)
20	1KG21CS021	17 (**TH**) , 18 (**PR**)
21	1KG21CS022	20 (**TH**) , 18 (**PR**)
22	1KG21CS023	21 (**TH**) , 17 (**PR**)
23	1KG21CS024	22 (**TH**) , 19 (**PR**)
24	1KG21CS026	18 (**TH**) , 16 (**PR**)
25	1KG21CS027	30 (**TH**) , 19 (**PR**)
26	1KG21CS028	24 (**TH**) , 17 (**PR**)
27	1KG21CS029	18 (**TH**) , 19 (**PR**)
28	1KG21CS030	27 (**TH**) , 20 (**PR**)
29	1KG21CS031	18 (**TH**) , 18 (**PR**)
30	1KG21CS032	20 (**TH**) , 17 (**PR**)
31	1KG21CS033	23 (**TH**) , 17 (**PR**)
32	1KG21CS034	20 (**TH**) , 19 (**PR**)
33	1KG21CS035	14 (**TH**) , 16 (**PR**)
34	1KG21CS036	20 (**TH**) , 17 (**PR**)
35	1KG21CS037	28 (**TH**) , 20 (**PR**)
36	1KG21CS038	21 (**TH**) , 16 (**PR**)

Sl NO.	USN	21CS52
37	1KG21CS039	17 (**TH**) , 17 (**PR**)
38	1KG21CS040	15 (**TH**) , 18 (**PR**)
39	1KG21CS041	19 (**TH**) , 17 (**PR**)
40	1KG21CS042	14 (**TH**) , 16 (**PR**)
41	1KG21CS043	20 (**TH**) , 18 (**PR**)
42	1KG21CS044	19 (**TH**) , 19 (**PR**)
43	1KG21CS045	29 (**TH**) , 20 (**PR**)
44	1KG21CS046	24 (**TH**) , 20 (**PR**)
45	1KG21CS047	12 (**TH**) , 16 (**PR**)
46	1KG21CS048	22 (**TH**) , 18 (**PR**)
47	1KG21CS050	15 (**TH**) , 16 (**PR**)
48	1KG21CS051	13 (**TH**) , 18 (**PR**)
49	1KG21CS052	15 (**TH**) , 19 (**PR**)
50	1KG21CS053	26 (**TH**) , 20 (**PR**)
51	1KG21CS054	27 (**TH**) , 20 (**PR**)
52	1KG21CS055	24 (**TH**) , 19 (**PR**)
53	1KG21CS056	12 (**TH**) , 17 (**PR**)
54	1KG21CS057	26 (**TH**) , 20 (**PR**)
55	1KG21CS058	24 (**TH**) , 18 (**PR**)
56	1KG21CS059	21 (**TH**) , 20 (**PR**)
57	1KG21CS060	26 (**TH**) , 20 (**PR**)
58	1KG21CS061	29 (**TH**) , 20 (**PR**)
59	1KG21CS062	21 (**TH**) , 18 (**PR**)
60	1KG21CS063	14 (**TH**) , 16 (**PR**)
61	1KG21CS064	28 (**TH**) , 19 (**PR**)
62	1KG21CS065	20 (**TH**) , 17 (**PR**)
63	1KG21CS066	12 (**TH**) , 16 (**PR**)
64	1KG21CS067	23 (**TH**) , 20 (**PR**)
65	1KG21CS068	28 (**TH**) , 19 (**PR**)
66	1KG21CS069	12 (**TH**) , 15 (**PR**)
67	1KG21CS070	18 (**TH**) , 18 (**PR**)
68	1KG21CS071	12 (**TH**) , 16 (**PR**)
69	1KG21CS072	28 (**TH**) , 19 (**PR**)
70	1KG21CS073	19 (**TH**) , 18 (**PR**)
71	1KG21CS074	20 (**TH**) , 18 (**PR**)
72	1KG21CS075	24 (**TH**) , 20 (**PR**)
73	1KG21CS076	22 (**TH**) , 18 (**PR**)
74	1KG21CS077	23 (**TH**) , 20 (**PR**)
75	1KG21CS078	20 (**TH**) , 19 (**PR**)

Sl NO.	USN	21CS52
76	1KG21CS079	12 **(TH)** , 16 **(PR)**
77	1KG21CS080	14 **(TH)** , 18 **(PR)**
78	1KG21CS081	22 **(TH)** , 18 **(PR)**
79	1KG21CS082	12 **(TH)** , 17 **(PR)**
80	1KG21CS083	20 **(TH)** , 19 **(PR)**
81	1KG21CS084	24 **(TH)** , 19 **(PR)**
82	1KG21CS085	18 **(TH)** , 19 **(PR)**
83	1KG21CS086	24 **(TH)** , 20 **(PR)**
84	1KG21CS087	15 **(TH)** , 18 **(PR)**
85	1KG21CS088	16 **(TH)** , 19 **(PR)**
86	1KG21CS089	23 **(TH)** , 20 **(PR)**
87	1KG21CS090	18 **(TH)** , 20 **(PR)**
88	1KG21CS091	18 **(TH)** , 17 **(PR)**
89	1KG21CS092	18 **(TH)** , 17 **(PR)**
90	1KG21CS093	22 **(TH)** , 19 **(PR)**
91	1KG21CS094	12 **(TH)** , 16 **(PR)**
92	1KG21CS095	21 **(TH)** , 18 **(PR)**
93	1KG21CS096	21 **(TH)** , 18 **(PR)**
94	1KG21CS097	12 **(TH)** , 12 **(PR)**
95	1KG21CS098	13 **(TH)** , 19 **(PR)**
96	1KG21CS099	19 **(TH)** , 17 **(PR)**
97	1KG21CS100	15 **(TH)** , 17 **(PR)**
98	1KG21CS101	24 **(TH)** , 17 **(PR)**
99	1KG21CS102	21 **(TH)** , 18 **(PR)**
100	1KG21CS103	20 **(TH)** , 18 **(PR)**
101	1KG21CS104	18 **(TH)** , 19 **(PR)**
102	1KG21CS106	15 **(TH)** , 17 **(PR)**
103	1KG21CS108	27 **(TH)** , 20 **(PR)**
104	1KG21CS109	21 **(TH)** , 18 **(PR)**
105	1KG21CS110	12 **(TH)** , 17 **(PR)**
106	1KG21CS111	15 **(TH)** , 17 **(PR)**
107	1KG21CS112	16 **(TH)** , 16 **(PR)**
108	1KG21CS113	16 **(TH)** , 18 **(PR)**
109	1KG21CS114	16 **(TH)** , 18 **(PR)**
110	1KG21CS115	23 **(TH)** , 20 **(PR)**
111	1KG21CS116	13 **(TH)** , 18 **(PR)**
112	1KG21CS117	12 **(TH)** , 17 **(PR)**
113	1KG21CS118	21 **(TH)** , 19 **(PR)**
114	1KG21CS119	18 **(TH)** , 16 **(PR)**

Sl NO.	USN	21CS52
115	1KG21CS120	20 (**TH**) , 18 (**PR**)
116	1KG21CS121	19 (**TH**) , 18 (**PR**)
117	1KG21CS122	13 (**TH**) , 18 (**PR**)
118	1KG21CS123	17 (**TH**) , 18 (**PR**)
119	1KG21CS124	15 (**TH**) , 18 (**PR**)
120	1KG21CS125	22 (**TH**) , 19 (**PR**)
121	1KG21CS126	16 (**TH**) , 18 (**PR**)
122	1KG22CS400	16 (**TH**) , 16 (**PR**)
123	1KG22CS401	15 (**TH**) , 16 (**PR**)
124	1KG22CS402	15 (**TH**) , 16 (**PR**)
125	1KG22CS403	26 (**TH**) , 20 (**PR**)
126	1KG22CS404	14 (**TH**) , 17 (**PR**)
127	1KG22CS405	13 (**TH**) , 16 (**PR**)
128	1KG22CS406	13 (**TH**) , 17 (**PR**)
129	1KG22CS407	12 (**TH**) , 17 (**PR**)
130	1KG22CS408	13 (**TH**) , 18 (**PR**)
131	1KG22CS409	23 (**TH**) , 19 (**PR**)
132	1KG22CS410	16 (**TH**) , 18 (**PR**)
133	1KG22CS411	18 (**TH**) , 18 (**PR**)

KSSEM

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course End Survey V SEM B SEC CN

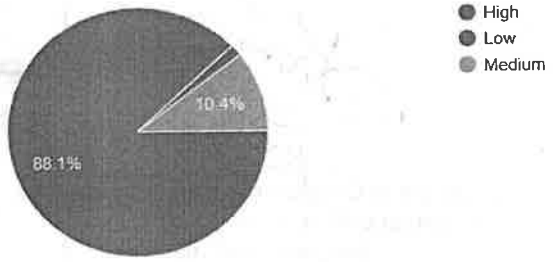
Timestamp	Email Address	USN	NAME	1.The course followed the syllabus prescribed by the university.	2.The course was organized in a structured manner that helped me to understand underlying concepts.	3.The course assignments /test assesses what I have learned in this course.	4.The course developed my ability to apply the concepts of computer networks to practice.	5.The course has improved my knowledge on the subjects.	6. The course has given enough knowledge to take next level courses.	Signature
2-29-2024 19:22:49	kmmanasvi@gmail.com	1KG21CS064	Manasvi K M	High	High	High	Medium	Medium	Medium	Manasvi
2-29-2024 10:56:59	mmanyar535@gmail.com	1KG21CS065	Manya. R	Medium	Medium	Medium	Medium	Medium	Medium	Manya
2-29-2024 12:06:20	mouryamegha21@gmail.com	1KG21CS066	Meghana G	High	High	High	High	High	High	
2-29-2024 10:38:49	mohithkumar657@gmail.com	1KG21CS067	Mohith Kumar Y V	High	High	High	High	High	High	
2-29-2024 11:04:29	mythri783@gmail.com	1KG21CS068	Mythri Y	High	High	High	High	High	High	
3-4-2024 17:09:59	namithumesh28@gmail.com	1KG21CS069	Namith	High	High	High	High	High	High	
3-3-2024 19:23:23	nandanamurthy6203@gmail.com	1KG21CS070	Nandana M	High	High	High	Medium	High	High	NANDANA M
2-29-2024 19:23:46	jashwanthnannuri@gmail.com	1KG21CS071	N Jaswanth	High	High	High	High	High	High	
3-3-2024 19:25:12	nchandu2003@gmail.com	1KG21CS072	Navachandu N	High	High	High	High	High	High	
3-4-2024 21:01:15	srinivasaranga.1978@gmail.com	1KG21CS073	NAVEENKANT H S	High	High	High	High	High	High	
2-29-2024 13:01:15	neha.udaya31@gmail.com	1KG21CS074	Neha u	High	High	High	High	High	High	
2-29-2024 19:48:41	nikitha.lakshmana26@gmail.com	1KG21CS075	Nikitha L	High	High	High	High	High	High	Nikitha L
3-3-2024 19:35:32	nikithanm01@gmail.com	1KG21CS076	Nikitha NM	High	Medium	High	Medium	Medium	Medium	
3-4-2024 17:12:24	nileshdimitri911@gmail.com	1KG21CS077	Nilesh Rajan	Medium	Medium	Medium	Low	Low	Medium	
3-3-2024 19:25:37	nithinb013@gmail.com	1KG21CS078	Nithin B	High	High	High	High	High	High	NithinB
2-29-2024 11:00:43	nithinnaidu85@gmail.com	1KG21CS079	Nithin Naidu S.P	High	High	High	High	High	High	
2-29-2024 16:00:34	phemanthkumar90@gmail.com	1KG21CS080	P hemanth kumar	High	High	High	High	High	High	
2-29-2024 19:24:33	bharadwazpacha@gmail.com	1KG21CS081	Pacha Bharadwaz	High	High	High	High	High	High	
2-29-2024 21:14:50	adeshp.7760@gmail.com	1KG21CS082	Patil Adesh	High	High	High	High	High	High	
2-29-2024 20:10:02	pavancse12345@gmail.com	1KG21CS083	PAVAN R	High	High	High	High	High	High	Pavan. R
2-29-2024 10:54:41	likhithasudhakar14@gmail.com	1KG21CS084	Posina likhitha	High	High	High	High	High	High	
3-4-2024 17:24:16	prakrit2002@gmail.com	1KG21CS085	Prakrit R Aritas	High	Medium	High	Medium	Medium	Medium	
2-29-2024 19:36:46	sbhatpranava1@gmail.com	1KG21CS086	Pranava S Bhat	High	High	High	High	High	High	
3-4-2024 13:06:46	balakrishnaprerana93@gmail.com	1KG21CS087	Prerana B	Medium	Medium	Medium	Medium	Medium	Medium	

2-29-2024 19:23:38	rhhemanthkumar.07@gmail.com	1KG21CS088	R H Hemanth Kumar	High	High	High	High	High	High	
3-3-2024 20:16:48	rachanarajkumar.rorach@gmail.com	1KG21CS089	Rachana R	High	High	High	High	High	High	Rachana.R
2-29-2024 19:31:51	ramkigkl@gmail.com	1KG21CS090	Ramki G K L	High	High	High	High	High	High	
2-29-2024 21:21:17	ramunagaraj15@gmail.com	1KG21CS091	Ramu T N	High	High	High	High	High	High	
2-29-2024 10:28:23	rekharathodrathod74@gmail.com	1KG21CS092	Rekha Rathod	High	High	High	High	High	High	
3-3-2024 19:19:58	rohitl_2003@icloud.com	1KG21CS093	Rohit.L	High	High	High	High	High	High	
3-5-2024 16:03:41	saikirz12@gmail.com	1KG21CS094	S SAI KIRAN	High	High	High	High	High	High	
3-4-2024 19:35:31	sagarb4696@gmail.com	1KG21CS095	Sagar B	High	High	High	High	High	High	
2-29-2024 11:08:14	sahebdanagond17@gmail.com	1KG21CS096	Sahebgoud Danagond	High	High	High	High	High	High	
3-3-2024 19:21:14	sahilchalwa7760@gmail.com	1KG21CS097	Sahil chalwa	High	High	High	High	High	High	
2-29-2024 11:04:22	naidusairam30@gmail.com	1KG21CS098	Sairama naidu.M	High	High	High	High	High	High	
3-4-2024 11:49:59	sanjanavasista03@gmail.com	1KG21CS099	Sanjana S Vasista	Medium	Medium	Medium	Medium	Medium	Medium	
3-4-2024 17:28:35	vidyasanjana123@gmail.com	1KG21CS100	Sanjana U Revankar	High	High	High	High	High	High	
3-4-2024 10:54:27	sharmilakp2003@gmail.com	1KG21CS101	Sharmila k p	Low	Medium	Medium	High	High	Medium	
3-4-2024 17:49:19	ssinchana085@gmail.com	1KG21CS102	Sinchana	Medium	Medium	Medium	Medium	Medium	Medium	
2-29-2024 19:27:31	sirisham15082003@gmail.com	1KG21CS103	Sirisha M	High	High	High	High	High	High	
2-29-2024 10:42:34	sirithippareddy@gmail.com	1KG21CS104	Sirisha T	High	High	High	High	High	High	
3-4-2024 13:12:39	kakalwarsudarshan2004@gmail.com	1KG21CS106	SUDARSHAN S KAKALWAR	High	High	High	High	High	High	
2-29-2024 12:34:35	husuravi@gmail.com	1KG21CS108	Suravi H U	High	High	High	High	High	High	
2-29-2024 10:56:12	swathisuresha29@gmail.com	1KG21CS109	Swathi.s	High	Medium	High	Medium	Medium	Medium	
2-29-2024 11:04:58	t.chaitanyakrishna2003@gmail.com	1KG21CS110	T Chaitanya Krishna	High	High	High	High	High	High	
3-5-2024 0:03:12	tanishjain5803@gmail.com	1KG21CS111	TANISH JAIN	High	Medium	Medium	High	High	Medium	
2-29-2024 19:26:29	giridharr55@gmail.com	1KG21CS112	Tathi Reddy Giridhar Reddy	High	High	High	High	High	High	
3-4-2024 17:50:49	udayuda8@gmail.com	1KG21CS113	Uday L	High	High	High	High	High	High	
2-29-2024 19:33:38	vaibhavmj13@gmail.com	1KG21CS114	Vaibhav ES	High	High	High	High	High	High	
2-29-2024 13:37:23	ukulkarnivaishnavi@gmail.com	1KG21CS115	Vaishnavi u Kulkarni	High	Medium	High	Medium	Medium	High	Vaishnavi
2-29-2024 12:39:51	vamkri04@gmail.com	1KG21CS116	Vamshi Krishna.R	High	High	High	High	High	High	
2-29-2024 10:58:42	varshakrvarshakr4@gmail.com	1KG21CS117	Varsha.K.R	High	High	High	High	High	High	
2-29-2024 21:09:33	veekshithgowda00@gmail.com	1KG21CS118	Veekshith V	High	High	High	High	High	High	
3-3-2024 19:26:50	vinayvianvinay@gmail.com	1KG21CS119	VINAY M	High	Medium	High	High	Medium	High	Vinay
2-29-2024 21:03:58	vineethanarayan011@gmail.com	1KG21CS120	Vineetha N	Medium	Medium	Medium	Low	Medium	Medium	
2-29-2024 20:44:15	vinu2003cm@gmail.com	1KG21CS121	VINUTHA CM	Medium	Medium	Medium	Medium	Medium	Medium	
2-29-2024 19:23:36	vishwasramappa61@gmail.com	1KG21CS122	Vishwas.k.r	High	High	High	High	High	High	
2-29-2024 19:22:39	yashuvd52@gmail.com	1KG21CS123	Yashu vd	High	High	Medium	High	High	High	

2-29-2024 16:49:16	yashkrish21042003@gmail.com	1KG21CS124	YASHWANTH SAIBABA H	High	High	High	High	High	High	
3-4-2024 13:21:47	yashwinkumar.ravikumar@gmail.com	1KG21CS125	Yashwin Kumar R	High	High	High	High	High	High	
2-29-2024 12:17:59	yuvarajjust@gmail.com	1KG21CS126	Yuvaraj S	High	High	High	High	High	High	
2-29-2024 17:23:57	kirankumar90083@gmail.com	1KG22CS406	Kiran Kumar N	High	High	High	High	High	High	
3-1-2024 12:18:31	manumanu808849@gmail.com	1KG22CS407	Manoj mk	High	High	High	High	High	High	
2-29-2024 10:34:38	nabeelbaig074@gmail.com	1KG22CS408	Nabeel baig	High	High	High	High	High	High	
2-29-2024 11:04:54	rakeshrakesh48434@gmail.com	1KG22CS409	Rakesh KS	High	High	Medium	Medium	High	Medium	
3-3-2024 19:20:12	rakeshgm10@gmail.com	1KG22CS410	Rakesh M	High	High	High	High	High	High	
3-4-2024 12:45:17	shashankgowda1144@gmail.com	1KG22CS411	Shashank	High	High	High	High	Medium	Medium	

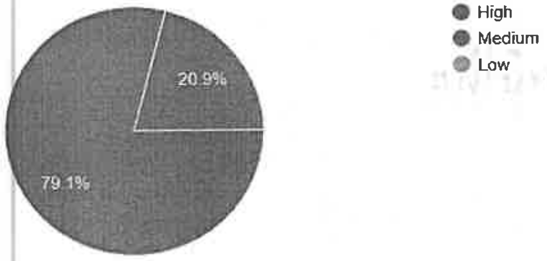
1.The course followed the syllabus prescribed by the university.

67 responses



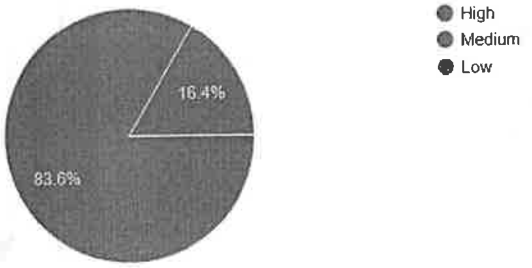
2.The course was organized in a structured manner that helped me to understand underlying concepts.

67 responses



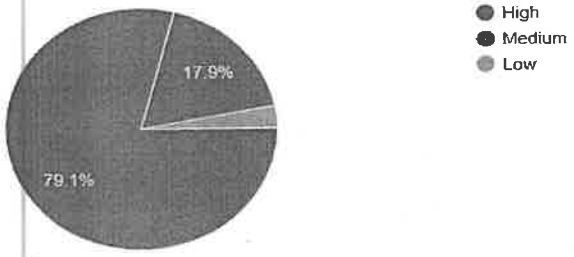
3.The course assignments/test assesses what I have learned in this course.

67 responses



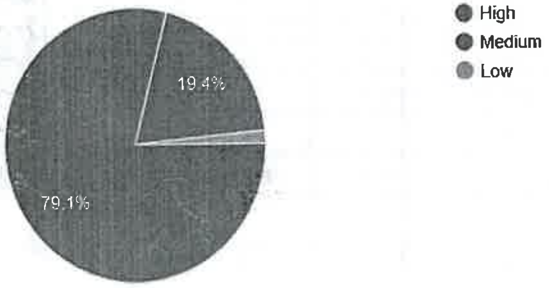
4.The course developed my ability to apply the concepts of computer networks to practice.

67 responses



5.The course has improved my knowledge on the subjects.

67 responses

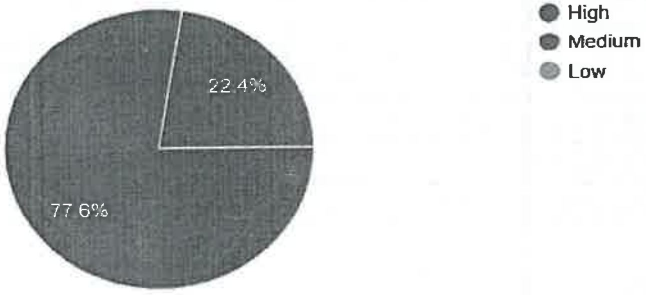


Sushmitha

Faculty Signature

6. The course has given enough knowledge to take next level courses.

67 responses



HOD

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

K. S. School of Engineering & Management, Bangalore - 560109

Department of Computer Science Engineering

Staff Feedback (2023-24) Odd Sem

Fifth Sem 'B' Section

Faculty Name: Mrs.Sushmitha Suresh

Course Name & Code: Computer Networs /21CS52

Class Strength:66

Sl. No.	1. Effective Planning & organisation	2. Punctuality / Class time Utilization	3. Ability to teach /explain / effective use of board	4. Interaction / Motivating students	5. Subject knowledge	6. Presentation of the subject / communication	7. Linking subject with practical application	8. Syllabus covearage / exam point of view	9. Evaluation of test / counselling	10. Attitude towards students
1	5	5	5	5	5	3	3	5	1	3
2	5	5	5	5	5	5	5	5	5	5
3	5	5	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5	5	5
5	3	3	3	3	3	3	3	3	3	3
6	5	5	5	5	5	5	5	5	5	5
7	5	5	5	5	5	5	5	5	5	5
8	5	5	5	5	5	5	5	5	5	5
9	4	4	4	4	4	4	4	4	4	4
10	5	5	5	5	5	5	5	5	5	5
11	5	5	5	5	5	5	5	5	5	5
12	5	5	5	5	5	5	5	5	5	5
13	5	5	5	5	5	5	4	5	5	5
14	4	4	4	4	4	4	4	4	4	3
15	5	5	5	5	5	5	5	5	5	5
16	5	5	5	5	5	5	5	5	5	5
17	5	5	5	5	5	5	5	5	5	5
18	5	5	5	5	5	5	5	5	5	5
19	5	5	4	4	4	5	3	5	3	4
20	5	5	5	5	5	5	5	5	5	5
21	4	5	3	3	2	3	3	5	5	4
22	5	5	4	4	4	4	4	5	4	4
23	5	5	5	5	5	5	5	5	5	5
24	5	5	5	5	5	5	5	5	5	5
25	4	4	4	4	4	4	4	4	4	4
26	5	5	5	5	5	5	5	5	5	5
27	5	5	5	5	5	5	5	5	5	5
28	5	5	5	5	5	5	5	5	5	5
29	5	5	5	5	5	5	5	5	5	5
30	5	5	4	4	4	4	4	5	4	5
31	5	5	5	5	5	5	5	5	5	5
32	4	4	4	4	4	4	4	4	4	4
33	5	5	4	5	5	5	5	5	5	5
34	5	5	5	5	5	5	5	5	5	5
35	5	5	5	5	5	5	5	5	5	5
36	5	5	5	5	5	5	5	5	5	5
37	5	5	5	5	5	5	5	5	5	5

38	5	5	5	5	5	5	5	5	5	5
39	4	5	4	3	4	4	4	5	5	4
40	5	5	5	5	5	5	5	5	5	5
41	5	5	5	5	5	5	5	5	5	5
42	5	5	5	5	5	5	5	5	5	5
43	5	5	5	5	5	5	5	5	5	5
44	5	5	5	5	5	5	5	5	5	5
45	5	5	5	5	5	5	5	5	5	5
46	5	5	5	5	5	5	5	5	5	5
47	5	5	5	5	5	5	5	5	5	5
48	4	4	4	4	4	4	4	4	4	4
49	5	5	5	5	5	5	5	5	5	5
50	4	4	4	4	4	4	4	4	4	4
51	5	5	5	5	5	5	5	5	5	5
52	5	5	5	5	5	4	5	5	4	5
53	5	5	5	5	5	5	5	5	5	5
54	4	5	4	4	4	4	4	4	4	4
55	5	5	5	5	5	5	5	5	5	5
56	5	5	5	4	5	4	5	5	5	5
57	4	4	3	3	4	4	4	5	3	4
58	4	4	4	4	4	4	4	4	4	4
59	5	5	5	5	5	5	5	5	5	5
60	4	5	4	4	4	4	4	4	4	4
61	5	5	5	5	5	5	5	5	5	5
62	5	5	5	5	5	5	5	5	5	4
63	5	5	5	5	5	5	5	5	5	5
64	5	5	5	5	5	5	5	5	5	5
65	5	5	5	5	5	5	5	5	5	5
66	5	5	5	5	5	5	5	5	5	5
67	5	5	5	5	5	5	5	5	5	5
Col. Total	321	325	315	314	316	314	313	324	313	315
Col. Avg.	4.79	4.85	4.70	4.69	4.72	4.69	4.67	4.84	4.67	4.70
Over all %	94.63									

Head of Department

HOD
Department of Computer Science Engineering
K.S School of Engineering & Management
Bangalore-560109

Principal