PARUL UNIVERSITY - Faculty of IT & Computer Science

Department of Computer Application

SYLLABUS FOR 2nd Sem MCA, M.Sc. (IT), MCA, 8th Sem IMCA PROGRAMME

Data Communications and Networking (05201253)

Type of Course: IMCA, M.Sc. (IT), MCA

Prerequisite: Basic concept of computer organization and knowledge of programming skills.

Rationale: To provide thorough understanding of networking concepts and knowledge of OSI layer's functionality.

Teaching and Examination Scheme:

Teaching Scheme			Credit	Examination Scheme					Total
				External		Internal			
Lect Hrs/ Week	Tut Hrs/ Week	Lab Hrs/ Week		T	P	T	CE	P	
4	0	2	5	60	30	20	20	20	150

Lect - Lecture, **Tut** - Tutorial, **Lab** - Lab, **T** - Theory, **P** - Practical, **CE** - CE, **T** - Theory, **P** - Practical

Contents:

Sr.	Topic	Weightage	Teaching Hrs.
1	**Introduction**: Components of data communication, Reference model - Open System Interconnection (OSI), Transmission Control Protocol/ Internet Protocol (TCP/IP) and comparisons, Types of networks-Local Area Network, Metropolitan Area Network, Wide Area Network, Wireless networks, Internetworks, Data communication fundamentals - introduction, frequency, band, analog signals, digital signals and transmission, Multiplexing and de-multiplexing - Time Division Multiplexing, Frequency Division Multiplexing, Wavelength Division Multiplexing, Orthogonal Frequency Division Multiplexing, Transmission and errors.	15%	10
2	**Physical Layer**: Transmission media - twisted pair, coaxial cable, fiber optics, Wireless transmission - radio, microwave, infrared, Switching - circuit switching, packet switching, message switching, Analog modulation - Amplitude Modulation, Frequency Modulation, Phase Modulation, Digital modulation - Amplitude Shift Keying, Frequency Shift Keying, Phase Shift Keying.	15%	9
3	**Data Link Layer**: Design issues, Error detection and correction, link protocols, Sliding window protocols, Medium access sub layer - channel allocations, ALOHA Protocols, Carrier Sense Multiple Access Protocols (CSMA), CSMA with Collision Detection, Collision free protocols, IEEE 802 standards, Data link layer switching.	20%	12

4	**The Network Layer**: Design issues, Routing algorithms - shortest path, flooding, Distance Vector Routing, hierarchical, broadcast, multicast, Congestion control algorithms - leaky bucket, token bucket, load shedding, Error detection and correction, Internetworking, IP protocol, Address Resolution Protocol, Reverse Address Resolution Protocol, IP address and classifications.	25%	12
5	**Transport Layer**: Design issue, Connection management, Transmission Control Protocol (TCP) - introduction, window management, User Datagram Protocol, Performance issues.	15%	6
6	**Application Layer**: Domain Name Services, Electronic mail, WWW, Hypertext Transfer Protocol, Network security - basics of cryptography and compression techniques.	10%	5

***Continuous Evaluation:**

It consists of Assignments/Seminars/Presentations/Quizzes/Surprise Tests (Summative/MCQ) etc.

Reference Books:

1. Computer Networks (TextBook)
   Andrew. S. Tanenbaum; Prentice Hall Publication, 2004,; 4th edition
2. Data and Computer Communication
   W. Stallings; McMillan
3. Data Communication and Networking
   Behrouz Forouzan; TMH
4. Computer Networks and Internet
   Comer; PHI

Useful Links:

https://sites.google.com/a/paruluniversity.ac.in/pu-dcn/

Course Outcome:

After Learning the course the students shall be able to:

1. define and explain basics of data communications and computer network architecture.
2. describe OSI reference model and basic functionalities of DNS, WWW and HTTP.
3. identify relevant data transmission techniques and media.
4. implement framing, error handling and congestion control techniques.
5. describe need for computer network security.

List of Practical:

1. Introduction to LINUX environment and related system programming

Linux provides a complete UNIX programming environment which includes all of the standard libraries, programming tools, compilers, and debuggers which you would expect of other UNIX systems.

With Linux, you have access to the complete set of libraries and programming utilities and the complete kernel and library source code.

Within the UNIX software world, systems and applications are often programmed in C or C++. The standard C and C++ compiler for Linux is GNU gcc.

2. Introduction of various networking equipments and Configuration of Computer Network

3. Introduction to pipes and related system calls for pipe management

4. Framing Protocol: Character Count

This method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow, and hence where the end of the frame is.

5. Framing Protocol: Byte Stufing

Implement a Program in C which demonstrates byte-stuffing framing technique, where sender reads data from a text file, encapsulates the read data in a frame, applies byte stuffing to the frame and sends it to receiver. Assume appropriate character for flag byte and stuff byte. Sender should display the frame data before stuffing and after stuffing for each frame transmitted. Receiver should display the received frame data before de-stuffing and after de-stuffing for each frame. At the end, the data file contents of the sender and the receiver should match. For sender/receiver communication use IPC mechanism FIFO/Named Pipes in Linux/Unix environment. Use Bit-Wise operators in C wherever applicable. Consider a simplistic frame structure consisting of only flag bytes and frame data.

6. Framing Protocol: Bit Stuffing

Implement a Program in C which demonstrates bit-stuffing framing technique, where sender reads data from a text file, encapsulates the read data in a frame, applies bit stuffing to the frame and sends it to receiver. Consider the character 01111110 (binary) for flag byte. Sender should display the frame data before bit stuffing and after bit stuffing for each frame transmitted. Receiver should display the received frame data before de-stuffing and after de-stuffing for each frame. Receiver should keep on storing the de-stuffed frame data in an external file. At the end, the data file contents of the sender and the receiver files should match. For sender/receiver communication use IPC mechanism FIFO/Named Pipes in Linux/Unix environment. Use Bit-Wise operators in C wherever applicable. Consider a simplistic frame structure consisting of only flag bytes and frame data and neglect all other fields like FCS, etc. Use appropriate data-types for various variables/frame structure. Take appropriate frame size and file sizes. Test the program for arbitrary bit sequences in the sender side data file.

7. Error Detection : LRC and Checksum

Implement a Program in GNU C which demonstrates the Block Parity (LRC) Method (for Even Parity). The sender reads an ASCII character string from K/B, applies the block parity logic as applicable, encapsulates the character string in a frame and transmits the frame to the receiver. Sender should display the ASCII character string on screen before and after applying the block parity logic. Receiver should apply the necessary block parity logic and subsequently decide whether there is error or not. If, as per the receiver, there is an error, it should display an appropriate error message on screen else it should display the original character string (as per the receiver). For sender/receiver communication use IPC mechanism FIFO/Named Pipes in Linux/Unix environment. Use Bit-Wise operators in C wherever applicable. Assume a simplistic frame structure consisting of only frame data (ASCII character string). All other fields of frame are to be neglected. Use appropriate data-types for various variables/frame structure. Take appropriate frame size.

8. Error Detection : VRC

Implement a Program in GNU C which demonstrates the Single Bit Even Parity Method (VRC). The sender reads a single ASCII character from K/B, applies the parity bit logic as applicable, encapsulates the character in a frame and transmits the frame to the receiver. Sender should display the ASCII character on screen before and after applying the parity bit logic. Receiver should apply the necessary parity logic and subsequently decide whether there is error or not. If, as per the receiver, there is an error, it should display an appropriate error message on screen else it should display the original character (as per the receiver). Create Bit corruption routines on the sender side in the program and accordingly demonstrate those bit error scenarios for which this method works and also those bit error scenarios for which this method fails. All other fields of frame are to be neglected. Use appropriate data-types for various variables/frame structure. Take appropriate frame size.

9. Error Detection : CRC

Implement a Program in GNU C which determines whether a given Divisor is valid for CRC. The Divisor is to be entered in binary format from K/B. The sender encapsulates this data in a frame and transmits it to the receiver. The receiver applies the validation logic for CRC divisor and accordingly displays appropriate validation message on screen. For sender/receiver communication use IPC mechanism FIFO/Named Pipes in Linux/Unix environment. Use Bit-Wise operators in C wherever applicable. Consider a simplistic frame structure which encapsulates only the divisor information. All other fields of frame are to be neglected. Use appropriate data-types for various variables/frame structure. Take appropriate frame size.

10. Error Correction : Hamming Code

Implement a Program in GNU C which demonstrates the Hamming Code method. The sender reads an ASCII character from K/B, computes the corresponding Hamming Code and encapsulates the Hamming codeword into a frame and transmits the frame to the receiver. Sender should display the bit pattern of the codeword on screen before transmission. Receiver should apply the necessary Hamming code logic and identify whether there is an error or not. In case of no error, it should display the original character on screen. In case of error, it should display appropriate error message and also the bit position of the corrupted bit. Receiver should correct the error and then display the corrected character on screen. Create single Bit corruption routines on the sender side in the program and accordingly test the program for arbitrary single bit error positions and record your observations. For sender/receiver communication use IPC mechanism FIFO/Named Pipes in Linux/Unix environment. Use Bit-Wise operators in C wherever applicable. Consider a simplistic frame structure consisting of only the Hamming Code Word for the given ASCII character

11. Congestion control protocols : Leaky Bucket

The leaky bucket is an algorithm used in packet switched computer networks and telecommunications networks. It can be used to check that data transmissions, in the form of packets, conform to defined limits on bandwidth and burstiness (a measure of the unevenness or variations in the traffic flow). It can also be used as a scheduling algorithm to determine the timing of transmissions that will comply with the limits set for the bandwidth and burstiness: see network scheduler. The leaky bucket algorithm is also used in leaky bucket counters, e.g. to detect when the average or peak rate of random or stochastic events or stochastic processes exceed defined limits.

12. Congestion control protocols : Token Bucket

Unlike leaky bucket, token bucket allows saving, up to maximum size of bucket n. This means that bursts of up to n packets can be sent at once, giving faster response to sudden bursts of input. An important difference between two algorithms: token bucket throws away tokens when the bucket is full but never discards packets while leaky bucket discards packets when the bucket is full. Let token bucket capacity be C (bits), token arrival rate ρ (bps), maximum output rate M (bps), and burst length S (s) – During burst length of S (s), tokens generated are ρS (bits), and output burst contains a maximum of C + ρS (bits) – Also output in a maximum burst of length S (s) is M · S (bits), thus

$C + \rho S = MS$ or $S = C\ M - \rho$

Token bucket still allows large bursts, even though the maximum burst length S can be regulated by careful selection of ρ and M. One way to reduce the peak rate is to put a leaky bucket of a larger rate (to avoid discarding packets) after the token bucket.