MANIPAL INSTITUTE OF TECHNOLOGY

Manipal Academy of Higher Education

Manipal – 576 104

DEPARTMENT OF COMPUTER SCIENCE & ENGG.

CERTIFICATE

This is to certify that Ms./Mr. ……..…… Reg. No. …..……has satisfactorily completed the lab exercises prescribed for Computing Lab1/Information Systems Lab1 [CSE 5111/CSE5112] of First Year M. Tech. Degree at MIT, Manipal, in the academic year …..

Date: …….....

Signature
Faculty in Charge

CONTENTS

Course Objectives

- To implement the various routing algorithms.
- To implement a mini project using a network simulator.

Course Outcomes

At the end of this course, students will have the

- Ability to write code along the lines of the algorithm.
- Ability to carry out a mini project to solve a given problem.

Evaluation plan

- Internal Assessment Marks : 60%
  - Comprises of lab exercises and miniproject.
    - Continuous evaluation component (for each evaluation):5 marks
    - The assessment will depend on punctuality, program execution, maintaining the observation note and answering the questions in viva voce.
- Total marks for the 9 weekly lab exercises is marks out of 45, Mini project is evaluated for 15 marks. Both are added to finalize the internal assessment for 60 marks.

- End semester assessment of 2-hour duration: 40 %

INSTRUCTIONS TO THE STUDENTS

Pre- Lab Session Instructions

1. Students should carry the Lab Manual Book to every lab session
2. Be in time and adhere to the institution rules and maintain the decorum
3. Must Sign in the log register provided
4. Make sure to occupy the allotted system and answer the attendance

In- Lab Session Instructions

- Follow the instructions on the allotted exercises
- Show the program and results to the instructors on completion of experiments
- Copy the program and results in the Lab record.
- Prescribed textbooks and class notes can be kept ready for reference if required

General Instructions for the exercises and mini project in the Lab

- Academic honesty is required in all your work. You must solve all programming assignments entirely on your own, except where group work is explicitly authorized.
- The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Programs should perform input validation (Data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
    - Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs and outputs.
    - Statements within the program should be properly indented.
    - Use meaningful names for variables and functions.

- Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.

- Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.

- In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition class with the permission of the faculty concerned but credit will be given only to one day's experiment(s).

THE STUDENTS SHOULD NOT

- Bring mobile phones or any other electronic gadgets to the lab.
- Go out of the lab without permission.

Lab No. 1 **Date:**

IMPLEMENT SHORTEST PATH ALGORITHM

Finding the shortest path from a given source node to all other destination nodes in a given network:

One algorithm for finding the shortest path from a starting node to a target node in a weighted graph is Dijkstra's algorithm. The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph. Dijkstra's algorithm, published in 1959 and named after its creator Dutch computer scientist Edsger Dijkstra, can be applied on a weighted graph. The graph can either be directed or undirected. One stipulation to using the algorithm is that the graph needs to have a nonnegative weight on every edge.

Algorithm:

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.

5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

Exercises:

i. WAP to find the shortest path from a given source node to all others node in a given network and display the same. Assume an undirected network.

ii. WAP to find the shortest path from a given source node to a given destination node and also find the link with the smallest cost in a given undirected network. Display the corresponding outputs.

Additional Exercise:

iii. Repeat the above exercises for a directed network.

Lab No. 2 **Date:**

Implement BGP Routing Algorithm

Border Gateway Protocol

The Border Gateway Protocol (BGP) is an inter-autonomous system routing protocol. An autonomous system (AS) is a network or group of networks under a common administration and with common routing policies. BGP is used to exchange routing information for the Internet and is the protocol used between Internet service providers (ISP), which are different ASs. One of the most important characteristics of BGP is its flexibility. The protocol can connect together any internetwork of autonomous systems using an arbitrary topology. The only requirement is that each AS have at least one router that is able to run BGP and that this router connect to at least one other AS's BGP router. Beyond that, "the sky's the limit," as they say. BGP can handle a set of ASs connected in a full mesh topology (each AS to each other AS), a partial mesh, a chain of ASs linked one to the next, or any other configuration. It also handles changes to topology that may occur over time. The primary function of a BGP speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of Autonomous Systems (ASs) that reachability information traverses. BGP constructs a graph of autonomous systems based on the information exchanged between BGP routers. As far as BGP is concerned, whole Internet is a graph of ASs, with each AS identified by a Unique AS number. Connections between two ASs together form a path and the collection of path information forms a route to reach a specific destination. BGP uses the path information to ensure the loop-free inter domain routing. Another important assumption that BGP makes is that it doesn't know anything about what happens within the AS. This is of course an important prerequisite to the notion of an AS being autonomous - it has its own internal topology and uses its own choice of routing protocols to determine routes. BGP only takes the information conveyed to it from the AS and shares it with other ASs. When a pair of autonomous systems agrees to exchange routing information, each must designate a router that will speak BGP on its behalf; the two routers are said to become BGP peers of one another. As a router speaking BGP must communicate with a peer in another autonomous system, usually a machine, which is near to the edge

(Border) of the autonomous system is selected for this. Hence, BGP terminology calls the machine a Border Gateway Router.

BGP Characteristics: BGP is different from other routing protocols in several ways. Most important being that BGP is neither a pure distance vector protocol nor a pure link state protocol.

Let's have a look at some of the characteristics that stands BGP apart from other protocols.

• Inter-Autonomous System Configuration: BGP's primary role is to provide communication between two autonomous systems.

• Next-Hop paradigm: Like RIP, BGP supplies next hop information for each destination.

• Coordination among multiple BGP speakers within the autonomous system:

If an Autonomous system has multiple routers each communicating with a peer in other autonomous system, BGP can be used to coordinate among these routers, in order to ensure that they all propagate consistent information.

• Path information: BGP advertisements also include path information, along with the reachable destination and next destination pair, which allows a receiver to learn a series of autonomous system along the path to the destination.

• Policy support: Unlike most of the distance-vector based routing, BGP can implement policies that can be configured by the administrator. For Example, a router running BGP can be configured to distinguish between the routes that are known from within the Autonomous system and that which are known from outside the autonomous system.

• Runs over TCP: BGP uses TCP for all communication. So the reliability issues are taken care by TCP.

• Conserve network bandwidth: BGP doesn't pass full information in each update message. Instead full information is just passed on once and thereafter successive messages only carries the incremental changes called deltas. By doing so a lot of network Bandwidth is saved. BGP also conserves bandwidth by allowing sender to

aggregate route information and send single entry to represent multiple, related destinations.

• Support for CIDR: BGP supports classless addressing (CIDR). That it supports a way to send the network mask along with the addresses.

• Security: BGP allows a receiver to authenticate messages, so that the identity of the sender can be verified.

BGP Functionality and Route Information Management:

The job of the Border Gateway Protocol is to facilitate the exchange of route information between BGP devices, so that each router can determine efficient routes to each of the networks on an IP internetwork. This means that descriptions of routes are the key data that BGP devices work with. But in a broader aspect, BGP peers perform three basic functions. The First function consists of initial peer acquisition and authentication. Both the peers establish a TCP connection and perform message exchange that guarantees both sides have agreed to communicate. The second function primarily focuses on sending of negative or positive reachability information, this step is of major concern. The Third function provides ongoing verification that the peers and the network connection between them are functioning correctly. Every BGP speaker is responsible for managing route descriptions according to specific guidelines established in the BGP standards.

BGP Route Information Management Functions:

Conceptually, the overall activity of route information management can be considered to encompass four main tasks:

• Route Storage: Each BGP stores information about how to reach networks in a set of special databases. It also uses databases to hold routing information received from other devices.

• Route Update: When a BGP device receives an Update from one of its peers, it must decide how to use this information. Special techniques are applied to determine when and how to use the information received from peers to properly update the device's knowledge of routes.

• Route Selection: Each BGP uses the information in its route databases to select good routes to each network on the internetwork.

• Route Advertisement: Each BGP speaker regularly tells its peers what it knows about various networks and methods to reach them. This is called route advertisement and is accomplished using BGP Update messages.

Exercise:

i. WAP in which a BGP router receives routing tables from its peers and updates its own routing table. Display all the routing tables. Display the path information for a given intra domain packet/inter domain packet. Assume a completely connected network.

Additional Exercise:

ii. Repeat the above exercise for a non-completely connected network.

Lab No. 3 **Date:**

Implement Proactive Routing Technique-DSDV

Destination Sequenced Distance-Vector Routing Protocol (DSDV)

Here each node maintains a table that contains the shortest distance and the first node on the shortest path to every other node in the network. It incorporates table updates with increasing sequence number tags to prevent loops, to counter the count-to-infinity problem, and for faster convergence. As it is a table-driven routing protocol, routes to all destinations are readily available at every node at all times. The tables are exchanged between neighbors at regular intervals to keep an up-to-date view of the network topology. The tables are also forwarded if a node observes a significant change in local topology. The table updates are of two types: incremental updates and full dumps. An incremental update takes a single network data packet unit (NDPU), while a full dump may take multiple NDPUs. Incremental updates are used when a node does not observe significant changes in the local topology. A full dump is done either when the local topology changes significantly or when an incremental update requires more than a single NDPU. Table updates are initiated by a destination with a new sequence number which is always greater than the previous one. Upon receiving an updated table, a node either updates its tables based on the received information or holds it for some time to select the best metric (which may be the lowest number of hops) received from multiple versions of the same update table from different neighboring nodes. Based on the sequence number of the table update, it may forward or reject the table.

Exercises:

i. From a given adjacency matrix and cost matrix, WAP to construct a routing table and display the same.
ii. WAP that updates a given routing table of a given node upon receiving broken link information from one of the neighbors.

Additional Exercise:

iii. WAP that updates a given routing table of a given node upon receiving full dumps from its neighbors.

Lab No. 4 **Date:**

Implement Proactive Routing Technique-WRP

Wireless Routing Protocol (WRP)

WRP uses a set of tables to maintain more accurate information. The tables that are maintained by a node are the following: distance table (DT), routing table (RT), link cost table (LCT), and a message retransmission list (MRL).
The DT contains the network view of the neighbors of a node. It contains a matrix where each element contains the distance and the penultimate node reported by a neighbor for a particular destination.The RT contains the up-to date view of the network for all known destinations. It keeps the shortest distance, the *predecessor* node (penultimate node), the *successor* node (the next node to reach the destination), and a flag indicating the status of the path. The path status may be a simple path (correct), or a loop (error), or the destination node not marked (null). The LCT contains the cost (*e.g,*the number of hops to reach the destination) of relaying messages through each link. The cost of a broken link is ∞. It also contains the number of update periods (intervals between two successive periodic updates) passed since the last successful update was received from that link. This is done to detect link breaks. The MRL contains an entry for every update message that is to be retransmitted and maintains a counter for each entry. This counter is decremented after every retransmission of an update message. Each update message contains a list of updates. A node also marks each node in the RT that has to acknowledge the update message it transmitted. Once the counter reaches zero, the entries in the update message for which no acknowledgments have been received are to be retransmitted and the update message is deleted. Thus, a node detects a link break by the number of update periods missed since the last successful transmission. After receiving an update message, a node not only updates the distance for transmitted neighbors but also checks the other neighbors' distance, hence convergence is much faster than DSDV.

Exercises:

i. WAP to construct the initial DT at each node for a given network.
ii. WAP to update the DT at a node after receiving the DTs from the neighbors.

iii. Given a RT WAP to find the cost between a given destination node to each of the source nodes.

Additional Exercises:

iv. Given a MRL, WAP to detect the link breaks in a given network.
v. WAP to build a RT for a given network.

Lab No. 5 **Date:**

Implement On-demand Routing Technique-DSR

Dynamic Source Routing Protocol (DSR)

It is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The basic approach of this protocol during the route construction phase is to establish a route by flooding RouteRequest packets in the network. The destination node, on receiving a RouteRequest packet, responds by sending a RouteReply packet back to the source, which carries the route traversed by the RouteRequest packet received.

Consider a source node that does not have a route to the destination. When it has data packets to be sent to that destination, it initiates a RouteRequest packet. This RouteRequest is flooded throughout the network. Each node, upon receiving a RouteRequest packet, rebroadcasts the packet to its neighbors if it has not forwarded already or if the node is not the destination node, provided the packet's time to live (TTL) counter has not exceeded. Each RouteRequest carries a sequence number generated by the source node and the path it has traversed. A node, upon receiving a RouteRequest packet, checks the sequence number on the packet before forwarding it. The packet is forwarded only if it is not a duplicate RouteRequest. The sequence number on the packet is used to prevent loop formations and to avoid multiple transmissions of the same RouteRequest by an intermediate node that receives it through multiple paths. Thus, all nodes except the destination forward a RouteRequest packet during the route construction phase. A destination node, after receiving the first RouteRequest packet, replies to the source node through the reverse path the RouteRequest packet had traversed.

Exercises:

i. WAP to construct a RouteRequest and a RouteReply packets for a given network and display the same. Also, WAP that checks whether a receiving node must forward the RouteRequest packet or reply with a RouteReply packet.
ii. WAP to implement exponential backoff algorithm to avoid frequent RouteRequest flooding in the network when the destination is in another disjoint set.

Additional Exercise:

iii. Given the RouteReply packets at a source node WAP to construct a path to a given destination node.

Lab No. 6 **Date:**

Implement On Demand Routing Technique-AODV

Ad Hoc On-demand Distance Vector (AODV) Routing Protocol

It uses an on demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and DSR stems out from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on demand routing protocol, the source node floods the RouteRequest packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single RouteRequest. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node. A RouteRequest carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a RouteRequest, it either forwards it or prepares a RouteReply if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the RouteRequest packet. If a RouteRequest is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to sendRouteReply packets to the source. Every intermediate node, while forwarding a RouteRequest, enters the previous node address and its BcastID. A timer is used to delete this entry in case a RouteReply is not received before the timer expires. This helps in storing an active path at the intermediate node asAODV does not employ source routing of data packets. When a node receives a RouteReply packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

Exercises:

i. WAP to construct a RouteRequest and a RouteReply packets for a given network and display the same.
ii. WAP to illustrate the behavior of an intermediate node when it receives a RouteRequest packet from a neighbor and display the output for a given network.

Lab No. 7 **Date:**

IMPLEMENT VOIP PACKET LOSS RECOVERY TECHNIQUES

Recovering from Packet Loss in VoIP

In VoIP there are several schemes that attempt to preserve acceptable audio quality in the presence of packet loss. Such schemes are called loss recovery schemes. A packet is lost either if it never arrives at the receiver or if it arrives after its scheduled playout time.

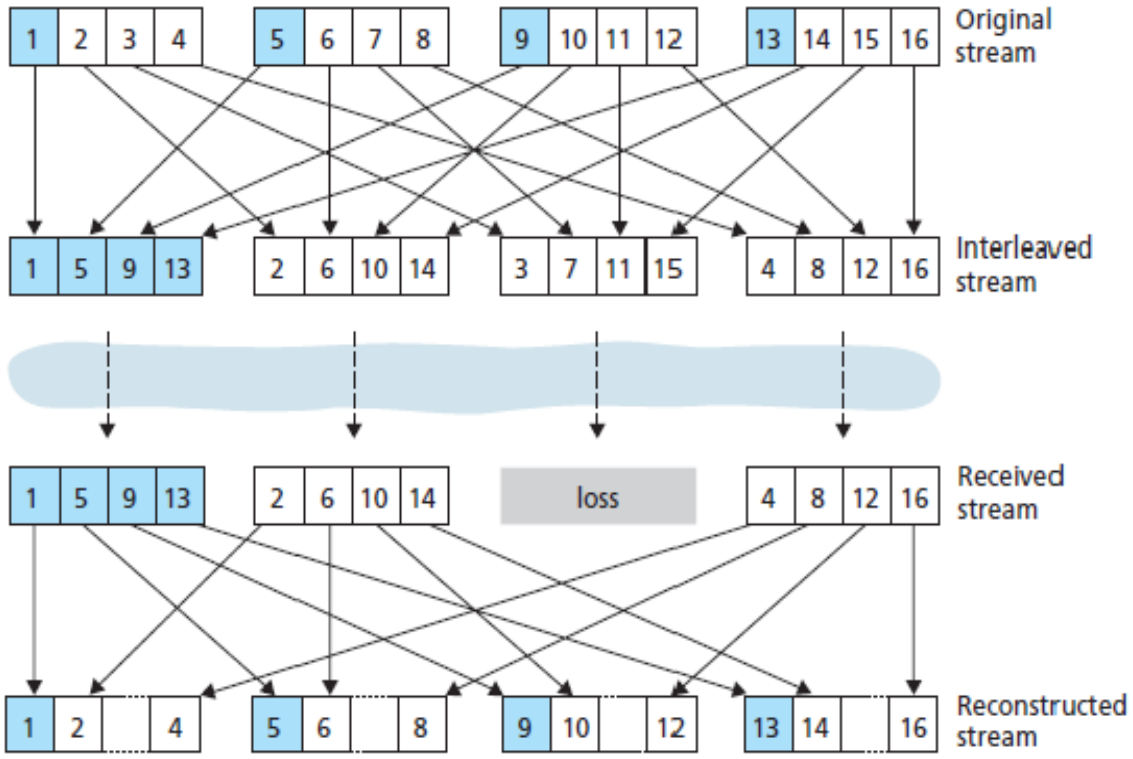
Retransmitting lost packets may not be feasible in a real-time conversational application such as VoIP. Indeed, retransmitting a packet that has missed its playout deadline serves absolutely no purpose. And retransmitting a packet that overflowed a router queue cannot normally be accomplished quickly enough. Because of these considerations, VoIP applications often use some type of loss anticipation scheme. Two types of loss anticipation schemes are forward error correction (FEC) and interleaving.

Simple Forward Error Correction (FEC)

The basic idea of FEC is to add redundant information to the original packet stream. For the cost of marginally increasing the transmission rate, the redundant information can be used to reconstruct approximations or exact versions of some of the lost packets. The mechanism sends a redundant encoded chunk after every n chunks. The redundant chunk is obtained by exclusive OR-ing the n original chunks. In this manner if any one packet of the group of n + 1 packets is lost, the receiver can fully reconstruct the lost packet. But if two or more packets in a group are lost, the receiver cannot reconstruct the lost packets. By keeping n + 1, the group size, small, a large fraction of the lost packets can be recovered when loss is not excessive. However, the smaller the group size, the greater the relative increase of the transmission rate. In particular, the transmission rate increases by a factor of 1/n, so that, if n = 3, then the transmission rate increases by 33 percent. Furthermore, this simple scheme increases the playout delay, as the receiver must wait to receive the entire group of packets before it can begin playout.

Interleaving

As an alternative to redundant transmission, a VoIP application can send interleaved audio. As shown in figure below, the sender resequences units of audio data before transmission, so that originally adjacent units are separated by a certain distance in the transmitted stream. Interleaving can mitigate the effect of packet losses. If, for example, units are 5 msec in length and chunks are 20 msec (that is, four units per chunk), then the first chunk could contain units 1, 5, 9, and 13; the second chunk could contain units 2, 6, 10, and 14; and so on. The figure shows that the loss of a single packet from an interleaved stream results in multiple small gaps in the reconstructed stream, as opposed to the single large gap that would occur in a non interleaved stream. Interleaving can significantly improve the perceived quality of an audio stream. It also has low overhead. The obvious disadvantage of interleaving is that it increases latency. This limits its use for conversational applications such as VoIP, although it can perform well for streaming stored audio. A major advantage of interleaving is that it does not increase the bandwidth requirements of a stream.



Exercises:

i. Apply simple FEC on a given number of data chunks. The transmitter program must accept the data chunks from the user and add the redundant chunk. The original and redundant chunks must be displayed. The receiver program must receive the data chunks with/without error, apply error correction if required and display the original data chunks.
ii. Apply interleaving mechanism on a given number of data chunks. The transmitter program must accept the original stream and display the interleaved stream. The receiver program must accept the received stream with/without loss and display the reconstructed stream.
iii. Apply Hamming code on a given number of data chunks. The transmitter program must accept the data chunks from the user and add the Hamming parity bits. The original and redundant chunks must be displayed. The receiver program must receive the data chunks with/without error, apply error correction if required and display the original data chunks.

Additional Exercises:

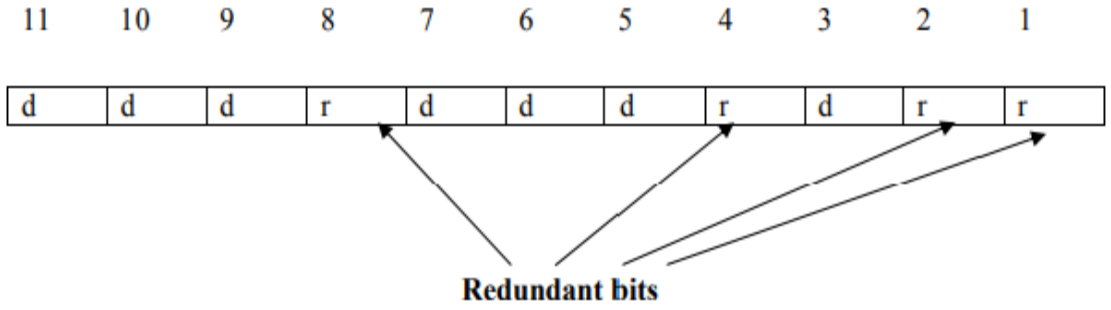
iv. Repeat the above exercises by taking an audio file, converting it into binary bits and grouping them into data chunks which go as input to each of the techniques.
v. Repeat the above exercises by taking a video file, converting it into binary bits and grouping them into data chunks which go as input to each of the techniques.

Lab No. 8 **Date:**

IMPLEMENT VOIP PACKET LOSS RECOVERY TECHNIQUE-HAMMING ERROR CORRECTING CODE.
IMPLEMENT VOIP PACKET SCHEDULING MECHANISM-FIFO

Hamming error correcting code

Hamming code can be applied to data units of any length and uses the relationship between the data bits and redundant bits. To understand how error correcting codes work we must define the *Hamming distance* between two binary strings. Suppose that we have two six-bit strings such as 1 1 1 0 1 0 and 1 0 1 1 1 1. Then the Hamming distance is the number of digits which are different. So in this case the Hamming distance is 3. If one bit in a symbol is changed then it has a Hamming distance of one from its original. This change might be due to the action of noise, which has corrupted the message. If two strings are separated by a large Hamming distance, say 3, then they can still be distinguished even if one bit is changed by noise. The idea of the simplest error correcting codes is to exploit this fact by adding extra digits to a binary symbol, so that the symbols are a large Hamming distance apart.

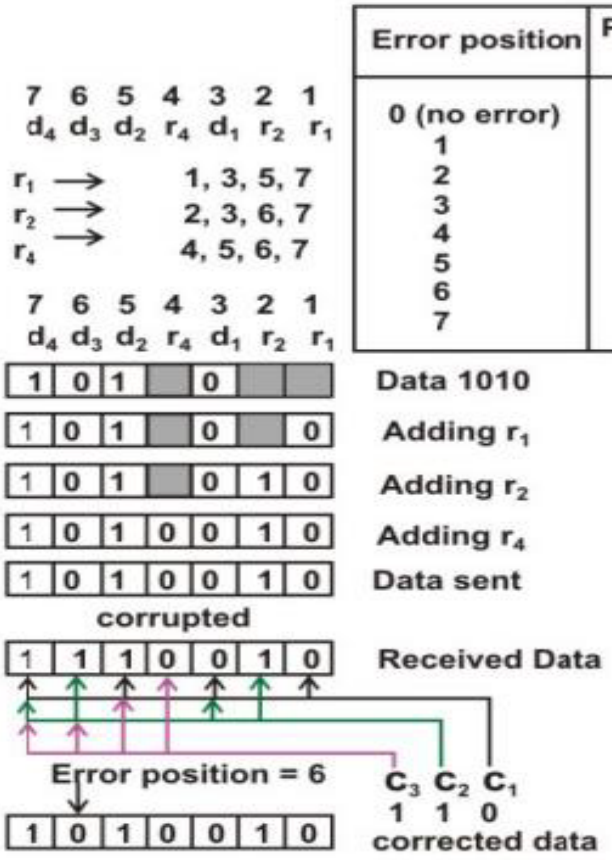


Positions of redundancy bits in hamming code

Basic approach for error detection by using Hamming code is as follows:

• To each group of m information bits k parity bits are added to form (m+k) bit code as shown above.

• Location of each of the (m+k) digits is assigned a decimal value.
• The k parity bits are placed in positions 1, 2, …, 2k-1 positions. K parity checks are performed on selected digits of each codeword.
• At the receiving end the parity bits are recalculated. The decimal value of the k parity bits provides the bit-position in error, if any.



Error position	Position number c3 c2 c1
0 (no error)	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Use of Hamming code for error correction for a 4-bit data

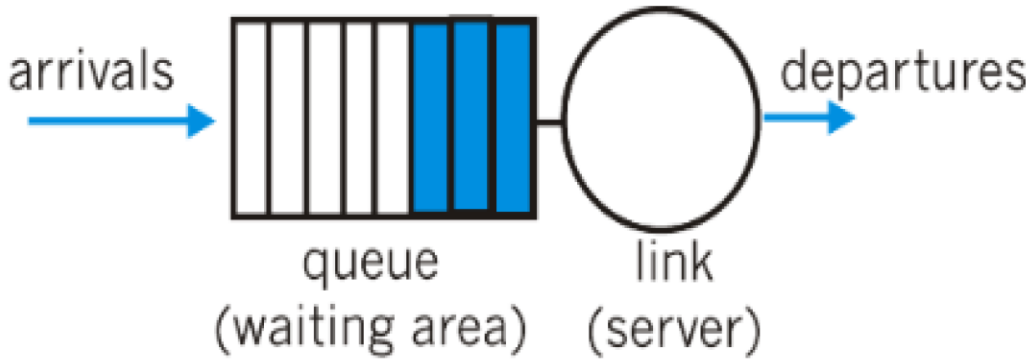
Figure above shows how hamming code is used for correction for 4-bit numbers (d4d3 d2 dl) with the help of three redundant bits (r3r2r1). For the example data 1010, first r1 (0) is calculated considering the parity of the bit positions, 1, 3, 5 and 7. Then the parity bit r2 is calculated considering bit positions 2, 3, 6 and 7. Finally, the parity bit r4 is calculated considering bit positions 4, 5, 6 and 7 as shown. If any corruption occurs in any of the transmitted code 1010010, the bit position in error can be found out by calculating r3r2r1 at the receiving end. For example, if the received code word is

1110010, the recalculated value of r3r2r1 is 110, which indicates that bit position in error is 6, the decimal value of 110.

FIFO Packet Scheduling Mechanism

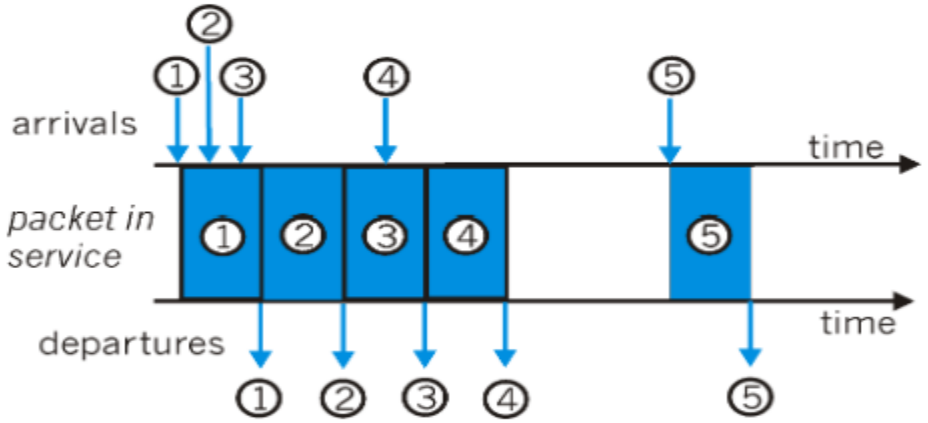
Packets belonging to various network flows are multiplexed together and queued for transmission at the output buffers associated with a link. The manner in which queued packets are selected for transmission on the link is known as the link scheduling discipline it plays an important role in providing QoS guarantees.

Figure below shows the queuing model abstractions for the First-in-First-Out (FIFO) link scheduling discipline. Packets arriving to the link output queue are queued for transmission if the link is currently busy transmitting another packet. If there is not sufficient buffering space to hold the arriving packet, the queue's packet discarding policy then determines whether the packet will be dropped ("lost") or whether other packets will be removed from the queue to make space for the arriving packet. When a packet is completely transmitted over the outgoing link (i.e., receives service) it is removed from the queue.



FIFO queuing abstraction

Figure below shows an example of the FIFO queue in operation.



FIFO Queue in Operation

The FIFO scheduling discipline (also known as First-Come-First-Served - FCFS) selects packets for link transmission in the same order in which they arrived at the output link queue. Packet arrivals are indicated by numbered arrows above the upper timeline, with the number indicating the order in which the packet arrived. Individual packet departures are shown below the lower timeline. The time that a packet spends in service (being transmitted) is indicated by the shaded rectangle between the two timelines. Because of the FIFO discipline, packets leave in the same order in which they arrived. Note that after the departure of packet 4, the link remains idle (since packets 1 through 4 have been transmitted and removed from the queue) until the arrival of packet 5.

Discard policy: if a packet arrives at a full queue: which one to discard?

- **tail drop**: drop arriving packet
- **priority:** drop/remove on priority basis
- **random:** drop/remove randomly

Exercise:

i. WAP to illustrate FIFO in operation. The program accepts the arrival time and burst time, and calculates the packet waiting time and turnaround time. Display all the timings for each packet.

Additional Exercise:

ii. WAP to illustrate shortest job first in operation. The program accepts the arrival time and burst time, and calculates the packet waiting time and turnaround time. Display all the timings for each packet.

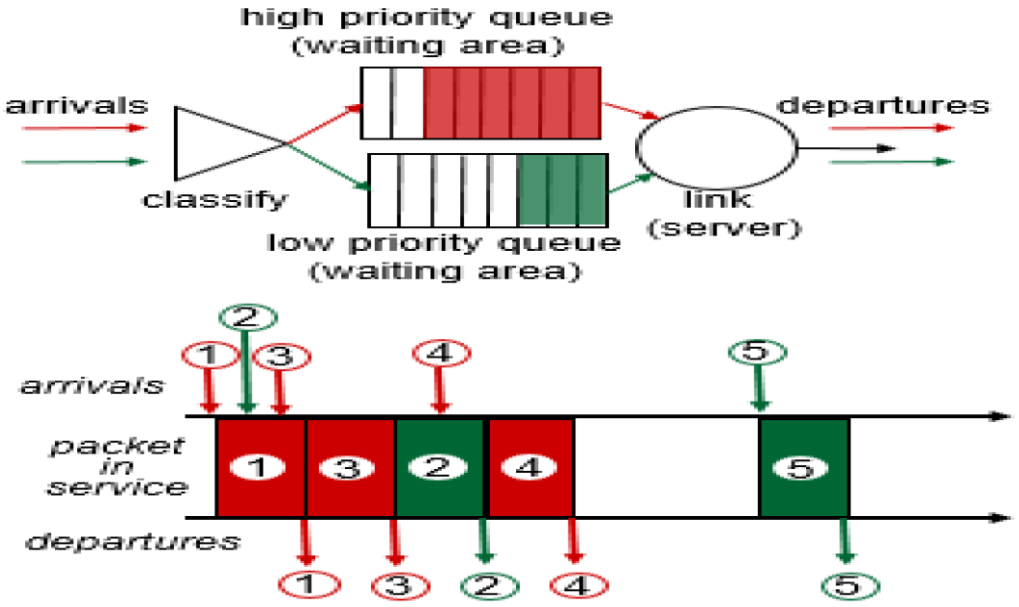
iii. WAP to design a suitable discard policy when the queue is full.

Lab No. 9 **Date:**

IMPLEMENT VOIP PACKET SCHEDULING MECHANISMS- PRIORITY, ROUND ROBIN

Priority Scheduling

Under priority queuing, packets arriving to the output link are classified into one of two or more priority classes at the output queue, as shown in figure below. As discussed in the previous section, a packet's priority class may depend on an explicit marking that it carries in its packet header (e.g., the value of the Type of Service (ToS) bits in an IPv4 packet), its source or destination IP address, its destination port number, or other criteria. Each priority class typically has its own waiting area (queue). When choosing a packet to transmit, the priority queuing discipline will transmit a packet from the highest priority class that has a non-empty queue (i.e., has packets waiting for transmission). The choice among packets in the same priority class is typically done in a FIFO manner.



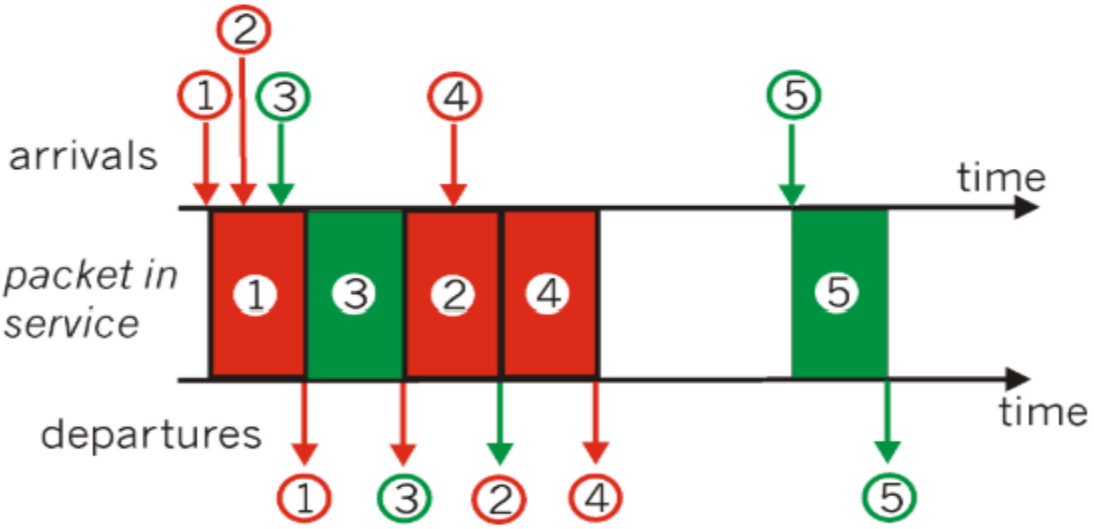
Operation of the priority queue

Figure above illustrates the operation of a priority queue with two priority classes. Packets 1,3 and 4 belong the the high priority class and packets 2 and 5 belong to the low priority class. Packet 1 arrives and, finding the link idle, begins transmission. During the transmission of packet 1, packets 2 and 3 arrive and are queued in the low and high priority queues, respectively. After the transmission of packet 1, packet 3 (a high priority packet) is selected for transmission over packet 2 (which, even though it arrived earlier, is a low priority packet). At the end of the transmission of packet 3, packet 2 then begins transmission. Packet 4 (a high priority packet) arrives during the transmission of packet 3 (a low priority packet). Under a so-called non-preemptive priority queuing discipline, the transmission of a packet is not interrupted once it has begun. In this case, packet 4 queues for transmission and begins being transmitted after the transmission of packet 2 is completed.

Round Robin Scheduling

Under the round robin queuing discipline, packets are again sorted into classes, as with priority queuing. However, rather than there being a strict priority of service among classes, a round robin scheduler alternates service among the classes. In the simplest form of round robin scheduling, a class 1 packet is transmitted, followed by a class 2 packet, followed by a class 1 packet, followed by a class 2 packet, etc. A so-called work-conserving queuing discipline will never allow the link to remain idle whenever there are packets (of any class) queued for transmission. A work-conserving round robin discipline that looks for a packet of a given class but finds none will immediately check the next class in the round robin sequence.

Figure below illustrates the operating of a two-class round robin queue. In this example, packets 1, 2 and 4 belong to class one, and packets 3 and 5 belong to the second class. Packet 1 begins transmission immediately upon arrival at the output queue. Packets 2 and 3 arrive during the transmission of packet 1 and thus queue for transmission. After the transmission of packet 1, the link scheduler looks for a class-two packet and thus transmits packet 3. After the transmission of packet 3, the scheduler looks for a class-one packet and thus transmits packet 2. After the transmission of packet 2, packet 4 is the only queued packet; it is thus transmitted immediately after packet 2.



Operation of the two-class round robin queue

Exercise:

i. WAP to illustrate priority scheduling in operation. The program accepts the total number of priorities, along with each packet priority, arrival time and burst time, and calculates the packet waiting time and turnaround time. Display all the timings and priority for each packet. Assume a non-preemptive priority queuing.

ii. WAP to illustrate round robin scheduling in operation. The program accepts the total number of classes, along with each packet class, arrival time and burst time. The program calculates the packet departure time, delay between arrival and departure time, and the average delay for all packets and displays all these timings along with the class for each packet. Assume a work conserving policy.

Additional Exercise:

iii. WAP to illustrate priority scheduling in operation. The program accepts the total number of priorities, along with each packet priority, arrival time and burst time, and calculates the packet waiting time and turnaround time. Display all the timings and priority for each packet. Assume a preemptive priority queuing.

iv. WAP to illustrate round robin scheduling in operation. The program accepts the total number of classes, along with each packet class, arrival time and burst time. The program calculates the packet departure time, delay between arrival and departure time, and the average delay for all packets and displays all these timings along with the class for each packet. Assume a non-work conserving policy.

Lab No. 10-12 **Date:**

Mini Project using UAV simulator/SDN simulator/MANET simulator

Students need to define the problem and submit the synopsis stating the objectives and expected outcomes in lab 10 and demonstrate the working of the mini project in lab 12.

Useful resources:

i. **mininet.org**

ii. https://github.com/microsoft/airsim

iii. https://pypi.org/project/sim2net/