CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 3 – May 2017

Sub:	Advances In Computer Networks						**Code:**	16SCS22
Date:	31-05-2017	Duration:	90 mins	Max Marks:	50	**Sem:** II	**Branch:**	M.Tech(CSE)

NOTE: Answer any five questions. Total marks: 50

		Marks	OBE CO	OBE RBT
1.	List the BGP characteristics and give the BGP path attributes.	[10]	CO3	L1
2.	a) Describe the format of UDP header and UDP message queue. b) Explain different end-to-end issues in TCP protocol.	[5] [5]	CO4	L2, L4
3.	Explain TCP header format along with significance of each field.	[10]	CO4	L4
4.	Explain with a neat diagram 3 way handshake algorithm of TCP Connection.	[10]	CO4	L4
5.	Explain the following congestion avoidance mechanisms: i)DEC ii) RED	[10]	CO5	L4
6.	Explain the following queuing disciplines: FIFO , Fair Queuing.	[10]	CO4	L4
7.	Explain adaptive retransmission mechanism with original algorithm for computing a timeout value between a pair of hosts.	[10]	CO4	L4

1

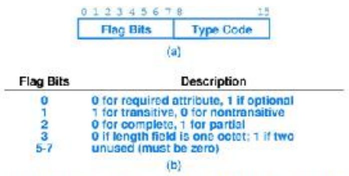
BGP Characteristics:

- Inter-Autonomous System Communication
- Coordination Among Multiple BGP Speakers
- Propagation of Reachability Information
- Next-hop Paradigm
- Policy Support
- Reliable Transport
- Path Information
- Incremental Updates
- Support for both IPV4 and IPv6
- Route Aggregation
- Authentication

$\}$ **5 Marks**

- ***path attributes* field**
  - Can specify:
    - Next hop for advertised destinations
    - List of autonomous systems along the destination path
    - Whether path info learned from other AS or derived from within sender's AS
- Path attributes important for 3 reasons
  - Allows receiver to check for routing loops

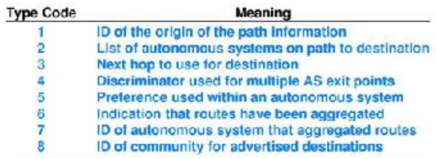
  - Allows receiver to implement policy constraints
  - Allows receiver to know source of all routes
- Path attributes field contains list of items
  - Each item consists of triple:
    (*type*, *length*, *value*)
    - *Type* is 2 octets
    - *Length* field 1 or 2 octets
    - Type field used to get size of length field; length field used to get size of *value* field



0 1 2 3 4 5 6 7 8 15
Flag Bits
Type Code
(a)
(b)

Flag Bits	Description
0	0 for required attribute, 1 if optional
1	1 for transitive, 0 for nontransitive
2	0 for complete, 1 for partial
3	0 if length field is one octet; 1 if two
5-7	unused (must be zero)

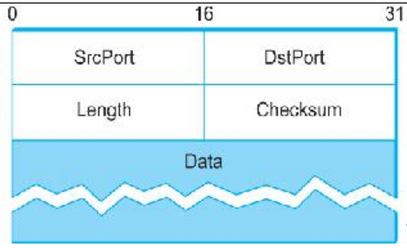
Figure 14.8 (a) The two octet type field that appears before each BGP attribute path item, and (b) the meaning of each flag bit.



Type Code	Meaning
1	ID of the origin of the path information
2	List of autonomous systems on path to destination
3	Next hop to use for destination
4	Discriminator used for multiple AS exit points
5	Preference used within an autonomous system
6	Indication that routes have been aggregated
7	ID of autonomous system that aggregated routes
8	ID of community for advertised destinations

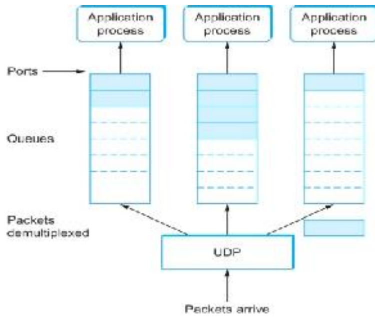
Figure 14.9 The BGP attribute type codes and the meaning of each. ------------------- (5 Marks)

2.a)



UDP Header

Explain the fig (5 Marks)

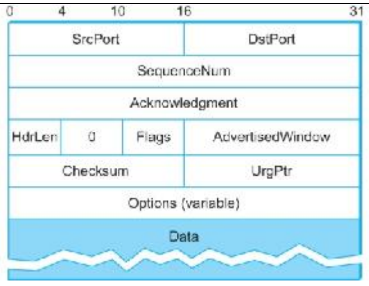


UDP Message Queue : Explain the mechanism

b) Explain the following points: ---------------------------------- (5 marks)

- TCP supports logical connections between processes that are running on two different computers in the Internet
- TCP connections are likely to have widely different RTT times
- Packets may get reordered in the Internet
- TCP needs a mechanism using which each side of a connection will learn what resources the other side is able to apply to the connection
- TCP needs a mechanism using which the sending side will learn the capacity of the network

3.



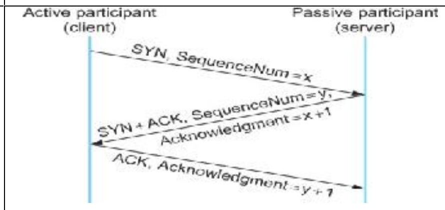
--------------------------------- **fig with explanation(4+6)**

- The SrcPort and DstPort fields identify the source and destination ports, respectively.
- The Acknowledgment, SequenceNum, and AdvertisedWindow fields are all involved in

TCP’s sliding window algorithm.

- Because TCP is a byte-oriented protocol, each byte of data has a sequence number; the SequenceNum field contains the sequence number for the first byte of data carried in that segment.
- The Acknowledgment and AdvertisedWindow fields carry information about the flow of data going in the other direction.
- The 6-bit Flags field is used to relay control information between TCP peers.
- The possible flags include SYN, FIN, RESET, PUSH, URG, and ACK.
- The SYN and FIN flags are used when establishing and terminating a TCP connection, respectively.
- The ACK flag is set any time the Acknowledgment field is valid, implying that the receiver should pay attention to it.
- The URG flag signifies that this segment contains urgent data. When this flag is set, the UrgPtr field indicates where the nonurgent data contained in this segment begins.
- The urgent data is contained at the front of the segment body, up to and including a value of UrgPtr bytes into the segment.
- The PUSH flag signifies that the sender invoked the push operation, which indicates to the receiving side of TCP that it should notify the receiving process of this fact.
- Finally, the RESET flag signifies that the receiver has become confused
- Finally, the RESET flag signifies that the receiver has become confused, it received a segment it did not expect to receive—and so wants to abort the connection.
- Finally, the Checksum field is used in exactly the same way as for UDP—it is computed over the TCP header, the TCP data, and the pseudoheader, which is made up of the source address, destination address, and length fields from the IP header.

4.


Active participant (client)
Passive participant (server)
SYN, SequenceNum = x
SYN + ACK, SequenceNum = y, Acknowledgment = x + 1
ACK, Acknowledgment = y + 1

Explain the mechanism in detail. ------------ (4 for fig+6 for explanation)

5.

DEC Bit: --- (5 Marks)

- Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur.
- This notification is implemented by setting a binary congestion bit in the packets that flow through the router; hence the name DECbit.
- The destination host then copies this congestion bit into the ACK it sends back to the source.
- Finally, the source adjusts its sending rate so as to avoid congestion
- A single congestion bit is added to the packet header. A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives.

This average queue length is measured over a time interval that spans the last busy+idle cycle, plus the current busy cycle

- Essentially, the router calculates the area under the curve and divides this value by the time interval to compute the average queue length.
- Using a queue length of 1 as the trigger for setting the congestion bit is a trade-off between significant queuing (and hence higher throughput) and increased idle time (and hence lower delay).

In other words, a queue length of 1 seems to optimize the power function

- The source records how many of its packets resulted in some router setting the congestion bit.

In particular, the source maintains a congestion window, just as in TCP, and watches to see what fraction of the last window’s worth of packets resulted in the bit being set

- If less than 50% of the packets had the bit set, then the source increases its congestion window by one packet.
- If 50% or more of the last window’s worth of packets had the congestion bit set, then the source decreases its congestion window to 0.875 times the previous value.

The “increase by 1, decrease by 0.875” rule was selected because additive increase/multiplicative decrease makes the mechanism stable.

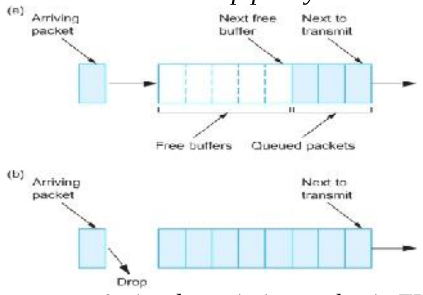
RED: -- (5 Marks)

- rather than explicitly sending a congestion notification message to the source, RED is most commonly implemented such that it *implicitly notifies* the source of congestion by dropping one of its packets.
- The source is, therefore, effectively notified by the subsequent timeout or duplicate ACK.
- RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts (or some other means of detecting packet loss such as duplicate ACKs).
- As the “early” part of the RED acronym suggests, the gateway drops the packet earlier than it would have to, so as to notify the source that it should decrease its congestion window sooner than it would normally have.
- In other words, the router drops a few packets before it has exhausted its buffer space completely, so as to cause the source to slow down, with the hope that this will mean it does not have to drop lots of packets later on.
- The second difference between RED and DECbit is in the details of how RED decides when to drop a packet and what packet it decides to drop.
- To understand the basic idea, consider a simple FIFO queue. Rather than wait for the queue to become completely full and then be forced to drop each arriving packet, we could decide to drop each arriving packet with some *drop probability* whenever the queue length exceeds some *drop level.*
- This idea is *called early random drop. The RED algorithm defines the details of how to monitor the queue length and* when to drop a packet.
- First, RED computes an average queue length using a weighted running average similar to the one used in the original TCP timeout computation. That is, AvgLen is computed as
  - AvgLen = (1 − Weight) × AvgLen + Weight × SampleLen
  - where 0 < Weight < 1 and SampleLen is the length of the queue when a sample measurement is made.
- In most software implementations, the queue length is measured every time a new packet arrives at the gateway.
- In hardware, it might be calculated at some fixed sampling interval.
- Second, RED has two queue length thresholds that trigger certain activity: MinThreshold and MaxThreshold.
- When a packet arrives at the gateway, RED compares the current AvgLen with these two thresholds, according to the following rules:
  - if AvgLen ≤ MinThreshold
    - → queue the packet
  - if MinThreshold < AvgLen < MaxThreshold
    - → calculate probability P
    - → drop the arriving packet with probability P
  - if MaxThreshold ≤ AvgLen
    - → drop the arriving packet
- P is a function of both AvgLen and how long it has been since the last packet was dropped.
- Specifically, it is computed as follows:
  - TempP = MaxP × (AvgLen − MinThreshold)/(MaxThreshold − MinThreshold)
  - P = TempP/(1 − count × TempP)

6.

FIFO: -- (5 Marks)

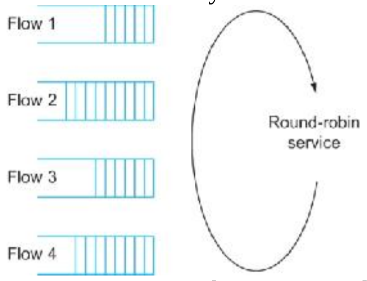
- The first packet that arrives at a router is the first packet to be transmitted
- Given that the amount of buffer space at each router is finite, if a packet arrives and the queue (buffer space) is full, then the router discards that packet
- This is done without regard to which flow the packet belongs to or how important the packet is. This is sometimes called *tail drop, since packets that arrive at* the tail end of the FIFO are dropped
- Note that tail drop and FIFO are two separable ideas. FIFO is a *scheduling discipline*—it determines the order in which packets are transmitted. Tail drop is a *drop policy—it determines which packets get dropped*



- A simple variation on basic FIFO queuing is priority queuing. The idea is to mark each packet with a priority; the mark could be carried, for example, in the IP header.
- The routers then implement multiple FIFO queues, one for each priority class. The router always transmits packets out of the highest-priority queue if that queue is nonempty before moving on to the next priority queue.
- Within each priority, packets are still managed in a FIFO manner.

Fair Queuing: -- (5 Marks)

- The main problem with FIFO queuing is that it does not discriminate between different traffic sources, or it does not separate packets according to the flow to which they belong.
- Fair queuing (FQ) is an algorithm that has been proposed to address this problem. The idea of FQ is to maintain a separate queue for each flow currently being handled by the router. The router then services these queues in a sort of round-robin,

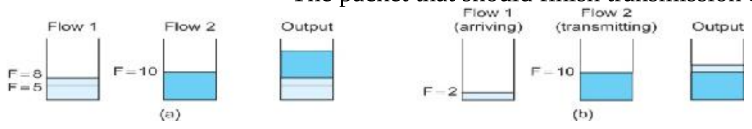


- The main complication with Fair Queuing is that the packets being processed at a router are not necessarily the same length.
- To truly allocate the bandwidth of the outgoing link in a fair manner, it is necessary to take packet length into consideration.
    - For example, if a router is managing two flows, one with 1000-byte packets and the other with 500-byte packets (perhaps because of fragmentation upstream from this router), then a simple round-robin servicing of packets from each flow’s queue will give the first flow two thirds of the link’s bandwidth and the second flow only one-third of its bandwidth.
- What we really want is bit-by-bit round-robin; that is, the router transmits a bit from flow 1, then a bit from flow 2, and so on.
- Clearly, it is not feasible to interleave the bits from different packets.

The FQ mechanism therefore simulates this behavior by first determining when a given packet would finish being transmitted if it were being sent using bit-by-bit round-robin, and then using this finishing time to sequence the packets for transmission

- To understand the algorithm for approximating bit-by-bit round robin, consider the behavior of a single flow
- For this flow, let
    - P_i : denote the length of packet *i*
    - S_i: time when the router starts to transmit packet *i*

    - F_i: time when router finishes transmitting packet i
    - Clearly, $F_i = S_i + P_i$
  - When do we start transmitting packet i?
    - Depends on whether packet i arrived before or after the router finishes transmitting packet i-1 for the flow
  - Let A_i denote the time that packet i arrives at the router
  - Then $S_i = \max(F_{i-1}, A_i)$
  - $F_i = \max(F_{i-1}, A_i) + P_i$
  - Now for every flow, we calculate F_i for each packet that arrives using our formula
  - We then treat all the F_i as timestamps
  - Next packet to transmit is always the packet that has the lowest timestamp
    - The packet that should finish transmission before all others

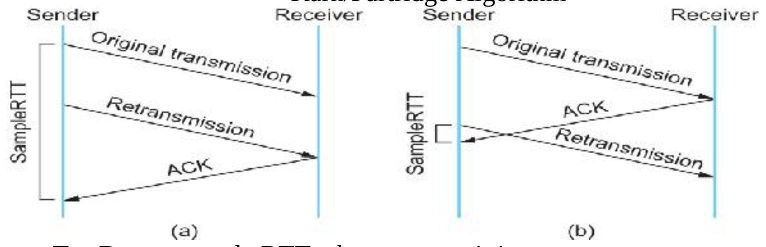


7.

- Original Algorithm
  - Measure **SampleRTT** for each segment/ ACK pair
  - Compute weighted average of RTT
    - **EstRTT** = *a* x **EstRTT** + (1 - *a*)x **SampleRTT**
    - *a* between 0.8 and 0.9
  - Set timeout based on **EstRTT**
    - **TimeOut = 2** x **EstRTT**

3 Marks

- Problem **(4 Marks)**
  - ACK does not really acknowledge a transmission
    - It actually acknowledges the receipt of data
  - When a segment is retransmitted and then an ACK arrives at the sender
    - It is impossible to decide if this ACK should be associated with the first or the second transmission for calculating RTTs
    - Karn/Partridge Algorithm



- Do not sample RTT when retransmitting
- Double timeout after each retransmission

Also, mention Jacbson/Karels Algorithm --(3 **Marks)**

-