Unit 3

Packet-Switching Networks and Traffic management: Datagram Networks, Virtual Circuit Networks, Shortest-path routing, Traffic management at the packet level; Traffic management at the flow level.

Introduction

Traditional telephone networks operate on the basis of circuit switching. A call setup process reserves resources (time slots) along a path so that the stream of voice samples can be transmitted with very low delay across the network.

The resources allocated to a call cannot be used by other users for the duration of the call. This approach is inefficient when the amount of information transferred is small or if information is produced in bursts, as is the case in many computer applications. Networks that transfer blocks of information called packets. Packet-switching networks are better matched to computer applications and can also be designed to support real-time applications such as telephony.

One perspective involves an E**xternal View** of the network and is concerned with the services that the network provides to the transport layer that operates above it at the end systems.

A second perspective on packet networks is concerned with the **Internal** operation of a network, physical topology of a network, the interconnection of links, switches, and routers. Concerned with the approach that is used to direct information across the network,, datagrams, or virtual circuits.

DATAGRAMS AND VIRTUAL CIRCUITS

- A network is usually represented as a cloud with multiple input sources and output destinations as shown in Figure 7.9.
- A network can be viewed as a generalization of a physical cable

in the sense of providing connectivity between multiple users.

- Unlike a cable, a switched network is geographically distributed and consists of a graph of transmission lines (i.e., links) interconnected by switches (nodes).
- These transmission and switching resources are configured to enable the flow of information among users.

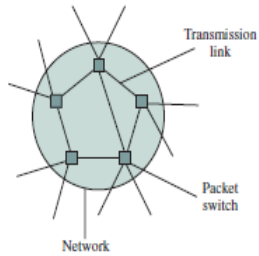


FIGURE 7.9 Switched network

- Networks provide for the interconnection of sources to destinations on a dynamic basis.
- Resources are typically allocated to an information flow only when needed.
- In this manner the resources are shared among the community of users resulting in efficiency and lower costs.
- There are two fundamental approaches to transferring information over a packet-switched network.
- The first approach, called **connection-oriented,** involves setting up a connection across the network before information can be transferred. The setup procedure typically involves the exchange of signaling messages and the allocation of resources along the path from the input to the output for the duration of the connection.
- The second approach is **connectionless** and does not involve a prior allocation of resources. Packet of information is routed independently from switch to switch until the packet arrives at its destination. Both approaches involve the use of switches or routers to direct packets across the network.

Connectionless Packet Switching

1. Packet switching has its origin in message switching, where a message is relayed from one station to another until the message arrives at its destination.

2. At the source each message has a header attached to it to provide source and destination addresses.

3. CRC Checkbits are attached to detect errors. As shown in Figure 7.13, the message is transmitted in a store-and-forward fashion. The message is transmitted in its entirety from one switch to the next switch.

4. Each switch performs an error checks, and if no errors are found, the switch examines the header to determine the next hop in the path to the destination.

5. If errors are detected, a retransmission may be requested. After the next hop is determined, the message waits for transmission over the corresponding transmission link.

6. Because the transmission links are shared, the message may have to wait until previously queued messages are transmitted.

7. Message switching does not involve a call setup. Message switching can achieve a high utilization of the transmission line. This increased utilization is achieved at the expense of queuing delays.

8. Loss of messages may occur when a switch has insufficient buffering to store the arriving message.2 End-to-end mechanisms are required to recover from these losses.

Figure 7.14 shows the total delay that is incurred when a message is transmitted over a path that involves two intermediate switches. The message must first traverse the link that connects the source to the first switch. Assume that this link has a propagation delay of p seconds and the message has a transmission time of T seconds. The message must next traverse the link connecting the two switches, and then it must traverse the link connecting the second switch and the destination. For simplicity we assume that the the propagation delay is the time that elapses from when a bit enters a transmission line to when it exits the line at the other end.

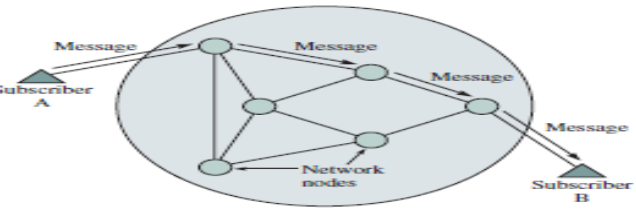


FIGURE 7.13 Message switching

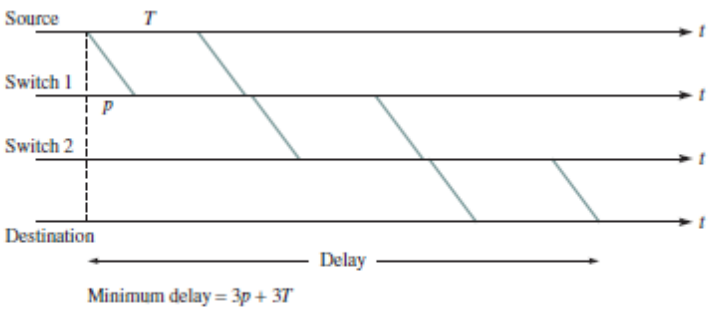


FIGURE 7.14 Delays in message switching

FIGURE 7.14 Delays in message switching

propagation delay and the bit rate of the transmission lines are the same. It then follows that the minimum end-to-end message delay is 3p+ 3Z.

Example—Long Messages versus Packets

Suppose that we wish to transmit a large message ($L = 10^6$ bits) over two hops. Suppose that the transmission line in each hop has an error rate of $p = 10^{-6}$ and that each hop does error checking and retransmission. How many bits need to be transmitted using message switching?

If we transmit the message in its entirety, the probability that the message arrives correctly after the first hop is

$$P_c = (1-p)^L = (1-10^{-6})^{1000000} \approx e^{-Lp} = e^{-1} \approx 1/3$$

Therefore, on the average it will take three tries to get the message over the first hop. Similarly, the second hop will require another three full message transmissions on the average. Thus 6 Mbits will need to be transmitted to get the 1 Mbit message across.

Now suppose that the message is broken up into ten 10^5-bit packets. The probability that a packet arrives correctly after the first hop is

$$P'_c = (1-10^{-6})^{100000} \approx e^{-1/10} \approx 0.90$$

Thus each packet needs to be transmitted 1/0.90 = 1.1 times on the average. The message gets transmitted over each hop by transmitting an average of 1.1 Mbit. The total number of bits transmitted over the two hops is then 2.2 Mbits.

Disadvantages

1.Very long messages are not desirable if the transmission lines are noisy because they lead to a larger

rate of message retransmissions. Message switching is also not suitable for interactive applications because it allows the transmission of very long messages that can impose very long waiting delays on other messages. By placing a maximum length on the size of the blocks that

are transmitted, packet switching limits the maximum delay that can be imposed by a single packet on other packets.

Solution

This situation is one reason that it is desirable to place a limit on the maximum size of the blocks that can be transmitted by the network. Thus long messages should be broken into smaller blocks of information, or packets. packet switching is more suitable than message switching for interactive applications.

Datagram Packet switching networks

1. In the datagram, or Connectionless packet-switching approach, each packet is routed independently through the network.
2. Each packet has an attached header that provides all of the information required to route the packet to its destination.
3. When a packet arrives at a packet switch, the destination address (and possibly other fields) in the header are examined to determine the next hop in the path to the destination.
4. The packet is then placed in a queue to wait until the given transmission line becomes available. By sharing the transmission line among multiple packets, packet switching can achieve high utilization at the expense of packet queuing delays.
5. The routers in the Internet are packet switches that operate in datagram mode.
6. Because each packet is routed independently, packets from the same source to the same destination may traverse different paths through the network as shown in Figure 7.15.

For example, the routes may change in response to a network fault. Thus packets may arrive out of order, and resequencing may be required at the destination.

Figure 7.16 shows the delay that is incurred by transmitting a message that is broken into three separate packets. Here we assume that the three packets follow two switches. In the

same path and are transmitted in succession. Ue neglect the overhead due to headers and suppose that each packet requires P=T/3 seconds to transmit. The three packets are

transmitted successively from the source to the first packet switch

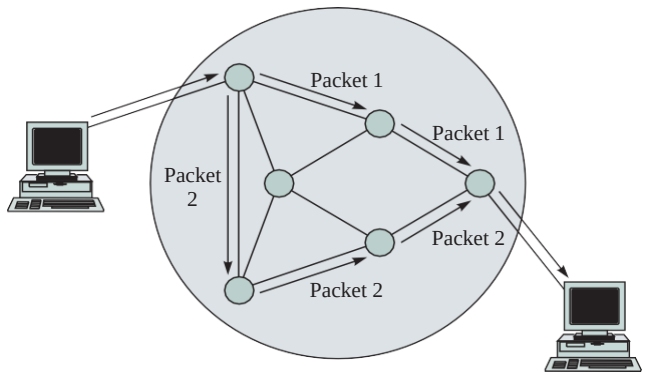


FIGURE 7.15 Datagram packet switching

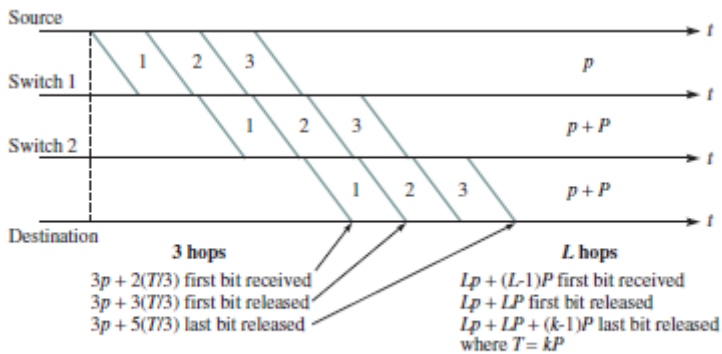


FIGURE 7.16 Delays in packet switching

The first packet in Figure 7.16 arrives at the first switch after p+ P seconds. Assuming that the packet arrives correctly, it can begin

transmission over the next hop after a brief processing time. The first packet is received at the second packet switch at time 2p+ 2P. Again we assume that the packet begins transmission over the final hop after a brief processing time. The first packet then arrives at the final link at time 3p+3P. As the first packet traverses the network, the subsequent packets follow immediately, as shown in the figure. In the absence of transmission errors, the final packet will arrive at the destination at time 3p+3P+2P which is less than the delay incurred in the message switching example in Figure 7.14. In general, if the path followed by a sequence of packets consists of L hops with identical propagation delays and transmission speeds, then the total delay incurred by a message that consists of k packets is given by

$$Lp + LP + (k-1)P$$

In contrast, the delay incurred using message switching is

$$Lp + LT = Lp + L(kP)$$

Figure 7.17 shows a routing table that contains an entry for each possible destination for a small network. This entry specifies the next hop that is to be taken by packets with the given destination. When a packet arrives, the destination address in the header is used to perform a table lookup. The result of the lookup is the number of the output port to which the packet must be forwarded.

Destination address	Output port
0785	7
1345	12
1566	6
2458	12

FIGURE 7.17 Routing table in connectionless packet switching

Routing table in datagram networks

1. When the size of the network becomes very large, this simple table lookup is not feasible, and the switch/router processor needs to execute a route lookup algorithm for each arriving packet.
2. In datagram packet switching, the packet switches have no knowledge of a ‘‘connection’’ even when a source and destination exchange a sequence of pack- ets. This feature makes datagram packet switching robust with respect to faults in the network.
3. If a link or packet switch fails, the neighboring packet switches react by routing packets along different links and by sharing the fault information with other switches.
4. This process results in the setting up of a new set of routing tables.
5. Because no connections are set up, the sources and destinations need not be aware of the occurrence of a failure in the network.
6. The processors in the switch /routers execute a distributed algorithm for sharing network state information and for synthesizing routing tables.
7. The design of the routing table is a key issue in the proper operation of a packet-switching network.
8. This design requires knowledge about the topology of the network as well as of the levels of traffic in various parts of the network.
9. Another issue is that the size of the tables can become very large as the size of the network increases.

7.3.3 Virtual-Circuit Packet Switching

Features of Virtual-Circuit packet switching

1. It involves the establishment of a fixed path between a source and a destination prior to the transfer of packets, as shown in Figure 7.18.

2. As in circuit switching, the call setup procedure usually takes place before any packets can flow through the network as shown in Figure 7.19.

3. The connection setup procedure establishes a path through the network and then sets parameters in the switches along the path as shown in Figure 7.20.

4. The controller/processor in every switch is involved in the exchange of signaling messages to set up the path.

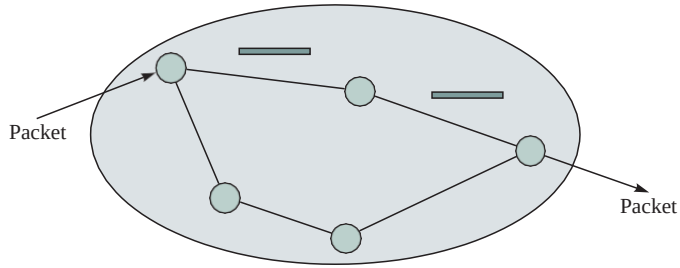
5. As in the datagram approach, the transmission links are shared by packets from many flows.

6. In general, in virtual-circuit packet switching, buffer and transmission resources need not be dedicated explicitly for the use of the connection, but the number of flows admitted may be limited to control the load on certain links.

7. All packets for the connection then follow the same path.

8. In datagram packet switching each packet must contain the full address of the source and destination.

9. In large networks these addresses can require a large number of bits and result in significant packet overhead and hence wasted transmission bandwidth.



FIGYRE 7.18 Virtual-circuit packet switching

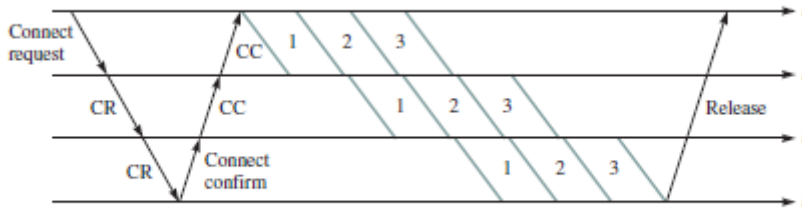


FIGURE 7.19 Delays in virtual-circuit packet switching

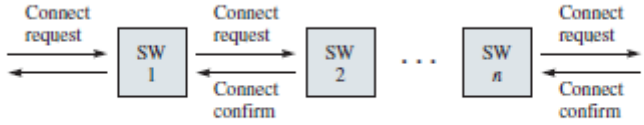


FIGURE 7.20 Signaling message exchanges in virtual-circuit setup

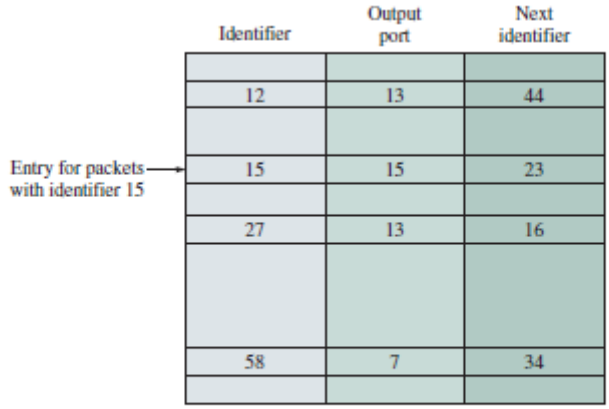


FIGURE 7.21 Example of virtual-circuit routing table for an input port

10.Abbreviated headers can be used.

11. The connection setup procedure establishes a number of entries in forwarding tables located in the various switches along the path.

12. At the input to every switch, the connection is identified by a Virtual-circuit identifier (VCI).

13. When a packet arrives at an input port, the VCI in the header is used to access the table, as shown in the example in Figure 7.21.

14. The table lookup provides the output port to which the packet is to be forwarded and the VCI that is to be used at the input port of the next switch.

15. The call setup procedure sets up a chain of pointers across the network that direct the flow of packets in a connection.

16. The table entry for a VCI can also specify the type of priority that is to be given to the packet by the scheduler that controls the transmissions in the next output port.

X

Madhu B

17. The number of bits required in the header in virtual-circuit switching is reduced to the number required to represent the maximum number of simultaneous connections over an input port.

18. This number is much smaller than the number required to provide full destination network addresses.

19. This factor is one of the advantages of virtual-circuit switching

relative to datagram switching.

20. In addition, the use of abbreviated headers and hardware-based table lookup allows fast processing and forwarding of packets.

21. Virtual-circuit switching does a table lookup and immediately forwards the packet to the output port; connectionless packet switching traditionally was much slower because it required software processing of the header before the next hop in the route could be determined. (This situation has changed with the development of hardware-based routing techniques.)

22. Another advantage of virtual-circuit packet switching is that resources can be allocated during connection setup. For example, a certain number of buffers may be reserved for a connection at every switch along the path, and a certain amount of bandwidth can be allocated at each link in the path.

23. In addition, the connection setup process ensures that a switch is able to handle the volume of traffic that is allowed over every transmission link.

24. In particular, a switch may refuse a connection over a certain link when the delays or link utilization exceed certain thresholds.

Disadvantages of Virtual-circuit packet switching

1. The switches in the network need to maintain information about the flows they are handling.

2. The amount of required ‘‘state’’ information grows very quickly with the number of flows.

3. Another disadvantage is evident when failures occur. In the case of virtual-circuit packet switching, when a fault occurs in the network all affected connections must be set up again.

4. If virtual-switching packet switching is used then the minimum delay for transmitting a message that consists of k packets is the same as in Figure 7.16, in addition to the time required to set up the connection.

5. A modified form of virtual-circuit packet switching, called cut-through packet switching, can be used when retransmissions are not used in the underlying data link control.

6. It is then possible for a packet to be forwarded as soon as the header is received and the table lookup is carried out.

As shown in Figure 7.22, the minimum delay in transmitting the message is then reduced to approximately the sum of the propagation delays in the various hops plus the one-message transmission time. (This scenario assumes that all lines are available to transmit the packet immediately.)

Cut-through packet switching may be desirable for applications such as speech transmission, which has a delay requirement but can tolerate some errors. Cut-through packet switching is also appropriate when the transmission is virtually error free, as in the case of optical fiber transmission, so that hop-by-hop error checking is unnecessary

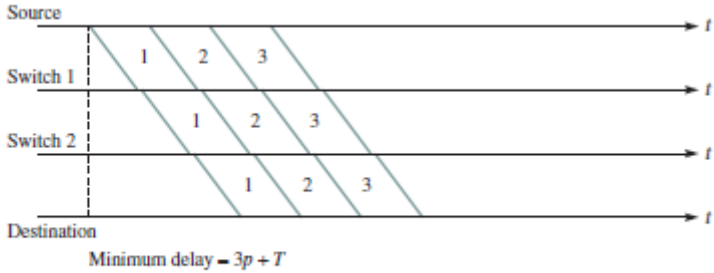


FIGURE 7.22 Cut-through packet switching

Comparison of Datagram and Virtual Circuit

Issue	Datagram	Virtual circuit
Connection setup	Not needed	Required
Addressing	Each packet contains the full destination address	Each packet contains a short VC identifier
State information	Switch doesn't hold state information	Switch holds VC state information in a table
Routing	Each packet is forwarded independently	All packets follow the same route
Effect of switch failures	None, except for packets lost during the crash	All VCs passing through the failed switch are broken
QoS support	Difficult	Easy
Congestion control	Difficult	Easy

7.4 ROUTING IN PACKET NETWORKS

1. A packet-switched network consists of nodes (routers or switches) interconnected by communication links in an arbitrary mesh like fashion as shown in Figure 7.23.

2 As suggested by the figure, a packet could take several possible paths from host A to host B. For example, three possible paths are 1-3-6, 1-4-5-6, and 1-2-5-6. However, which path is the ‘‘best’’one? Here the meaning of the term best depends on the objective function that the network operator tries to optimize.

3. If the objective is to minimize the number of hops, then path 1-3-6 is the best. If each link incurs a certain delay and the objective function is

to minimize the end-to-end delay, then the best path is the one that gives the end-to-end minimum delay.

4. A third objective function involves selecting the path with the greatest available bandwidth.

5. The purpose of the routing algorithm is to identify the set of paths that are best in a sense defined by the network operator.

6. Note that a routing algorithm must have global knowledge about the network state in order to perform its task.

7. The main ingredients of a good routing algorithm depend on the objective function that one is trying to optimize.

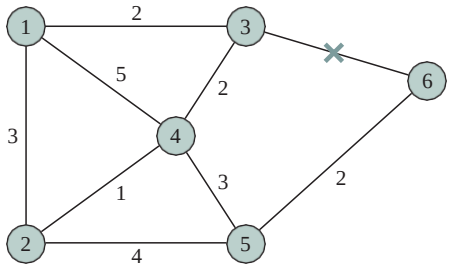
SHORTEST-PATH ALGORITHMS

1. Network routing is a major component at the network layer and is concerned with the problem of determining feasible paths (or routes) from each source to each destination.
2. A router or a packet-switched node performs two main functions,, routing and forwarding. In the routing function an algorithm finds an optimal path to each destination and stores the result in a routing table.
3. In the forwarding function a router forwards each packet from an input port to the appropriate output port based on the information stored in the routing table.
4. To commonly implemented shortest-path routing algorithms,, the Bellman-Ford algorithm and Dijkstra's algorithm.
5. Most routing algorithms are based on variants of shortest-path algorithms, which try to determine the shortest path for a packet according to some cost criterion.
6. To better understand the purpose of these algorithms, consider a communication network as a graph consisting of a set of nodes (or vertices) and a set of links (or edges), where each node represents a router or a packet switch and each link represents a communication channel between two routers.
7. Figure 7.28 shows such an example. Associated with each link is a value that represents the cost (or metric) of using that link.
8. For simplicity, it is assumed that each link is nondirected. If a link is directed, then the cost must be assigned to each direction.
9. If we define the path cost to be the sum of the link costs along the path, then the shortest path between a pair of nodes is the path with the least cost. For example, the shortest path from node 2 to node 6 is 2-4-3-6, and the path cost is 4.
10. Many metrics can be used to assign a cost to each link, depending on which function is to be optimized.

FIGURE 7.28 A sample network with associated link costs

Bellman-Ford Algorithm

The Bellman-Ford algorithm (also called the Ford-Fulkerson algorithm) is based on a principle that is intuitively easy to understand,, If a node is in the shortest path between A and B, then the path from the node to A must be the shortest path and the path from the node to B must also be the shortest path. As an example, suppose that we want to find the shortest path from node 2 to node 6 (the destination) in Figure 7.28. To reach the destination, a packet from node 2 must first go through node 1, node 4, or node 5. Suppose that someone tells us that the shortest paths from nodes 1, 4, and 5 to the destination (node 6) are 3, 3, and 2, respectively. If the packet first goes through node 1, the total distance (also called total cost) is 3 , which is equal to 6. Through node 4, the total distance is 1 3, equal to 4. Through node 5, the total distance is 4 2, equal to 6. Thus the shortest path from node 2 to the destination node is achieved if the packet first goes through node 4.



- Consider computations for one destination d
- Initialization
  - Each node table has 1 row for destination d
  - Distance of node d to itself is zero: $D_d=0$
  - Distance of other node j to d is infinite: $D_j=\infty$, for $j\neq d$
  - Next hop node n_j = -1 to indicate not yet defined for $j \neq d$
- Send Step
  - Send new distance vector to immediate neighbors across local link
- Receive Step
  - At node j, find the next hop that gives the minimum distance to d,
    - $Min_j \{ C_{ij} + D_j \}$
  - Go to send step

- wait for (change in local link cost of msg from neighbor)
- recompute distance table
- if least cost path to any dest has changed, notify neighbors

Example—Minimum Cost

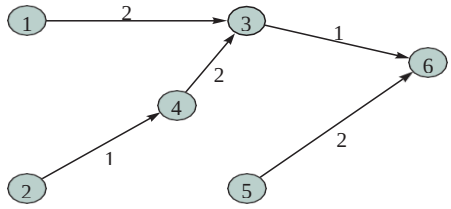
Iteration	Node 1	Node 2	Node 3	Node 4	Node 5
Initial	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)	(-1, ∞)
1	(-1, ∞)	(-1, ∞)	(6, 1)	(-1, ∞)	(6,2)
2	(3,3)	(5,6)	(6, 1)	(3,3)	(6,2)
3	(3,3)	(4,4)	(6, 1)	(3,3)	(6,2)

TABLE 7.1 Sample processing of Bellman-Ford algorithm. Each entry for node j represents the next node and cost of the current shortest path to destination 6.

Example—Shortest-Path tree

From the preceding example, draw the shortest path from each node to the destination node. From the last row of Table 7.1, we see the next node of node 1 is node 3, the next node of node 2 is node 4, the next node of node 3 is node 6, and so forth. Figure 7.29 shows the shortest-path tree rooted at node 6.

FIGURE 7.29 Shortest-path tree to node 6



One nice feature of the Bellman-Ford algorithm is that it lends itself readily to a distributed implementation.

1. The process involves having each node independently compute its minimum cost to each destination and periodically broadcast the vector of minimum costs to its neighbors.

2. Changes in the routing table can also trigger a node to broadcast the minimum costs to its neighbors to speed up convergence. This mechanism is called triggered updates.
3. It turns out that the distributed algorithm would also converge to the correct minimum costs under mild assumptions.
4. Upon convergence, each node would know the minimum cost to each destination and the corresponding next node along the shortest path.
5. Because only cost vectors (or distance vectors) are exchanged among neighbors, the distributed Bellman-Ford algorithm is often referred to as a distance Vector algorithm.
6. Each node i participating in the distance vector algorithm computes the following equation,,

$$Dii = 0$$

$$Dij = \min\{Cik \ddagger Dkj\},$$

$$\begin{aligned} D_2 &= \min\{C_{21} + D_1, C_{24} + D_4, C_{25} + D_5\} \\ &= \min\{3 + 3, 1 + 3, 4 + 2\} \\ &= 4 \end{aligned}$$

Where Dij is the minimum cost from node i to the destination node j. Upon updating, node i broadcast { } the vector Di1 Di2 Di3.. . to its neighbors. The distributed version can adapt to changes in link costs or topology as the next example shows.

Example—Recomputing Minimum Cost

Suppose that after the distributed algorithm stabilizes for the network shown in Figure 7.28, the link connecting node 3 and node 6 breaks. Compute the mini- mum cost from each node to the destination node (node 6), assuming that each node immediately recomputes its cost

after detecting changes and broadcasts its routing updates to its neighbors. The new network topology is shown in Figure 7.30.

For simplicity assume that the computation and transmission are synchronous. As soon as node 3 detects that link (3,6) breaks, node 3 recomputes the minimum cost to node 6 and finds that the new minimum cost is 5 via node 4 (as indicated in the first update in Table 7.2). It then sends the new routing update to its neighbors, which are nodes 1 and 4. These nodes then recompute their mini- mum costs (update 2). Node 1 transmits its routing table to nodes 2, 3, and 4, and node 4 transmits its routing table to nodes 1, 2, 3, and 5. After the messages from nodes 1 and 4 are received, nodes 2 and 3 will update their minimum costs (update 3).

FIGYRE 7.3Đ New network topology following break from node 3 to 6

Update	Node 1	Node 2	Node 3	Node 4	Node 5
Before break	(3, 3)	(4, 4)	(6, 1)	(3, 3)	(6, 2)
1	(3, 3)	(4, 4)	(4, 5)	(3, 3)	(6, 2)
2	(3, 7)	(4, 4)	(4, 5)	(2, 5)	(6, 2)
3	(3, 7)	(4, 6)	(4, 7)	(2, 5)	(6, 2)
4	(2, 9)	(4, 6)	(4, 7)	(5, 5)	(6, 2)
5	(2, 9)	(4, 6)	(4, 7)	(5, 5)	(6, 2)

TABLE 7.2 Next node and cost of current shortest path to node 6

Next nodes 1 and 4 update their minimum costs and send the update messages to their neighbors (update 4). In the last row no more changes are detected, and the algorithm converges. Note that during the calculation, packets already in transit may loop among nodes but eventually find the destination.

Example—Reaction to Link Failure

This example shows that the distributed Bellman−Ford algorithm may react slowly to a link failure. To see this, consider the topology shown in Figure 7.31a with node 4 as the destination. Suppose that after the algorithm stabilizes, link (3,4) breaks, as shown in Figure 7.31b. Recompute the minimum cost from each node to the destination node (node 4).

The computation of minimum costs is shown in Table 7.3. As the table shows, each node keeps updating its cost (in increments of 2 units). At each

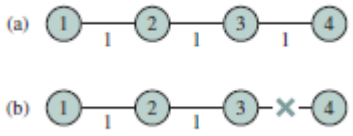


FIGURE 7.31 Topology before and after link failure

Update	Node 1	Node 2	Node 3
Before break	(2, 3)	(3, 2)	(4, 1)
After break	(2, 3)	(3, 2)	(3, 3)
1	(2, 3)	(3, 4)	(3, 3)
2	(2, 5)	(3, 4)	(3, 5)
3	(2, 5)	(3, 6)	(3, 5)
4	(2, 7)	(3, 6)	(3, 7)
5	(2, 7)	(3, 8)	(3, 7)
...	...	...	...

Note,, Dots in the last row indicate that the table continues to infinity

TABLE 7.3 Routing table for Figure 7.31

Advantages of DVA

- Stable and proven method (distance vector was the original routing algorithm)
- 2 Easy to implement and administer.
- Bandwidth requirements negligible for a typical LAN environment.
- Requires less hardware and processing power than other routing methods.

Disadvantages of DVA

- Relatively long time to reach convergence (updates sent at specified intervals)
- Routers must recalculate their routing tables before forwarding changes.
- Susceptible to routing loops (count-to-infinity).
- Bandwidth requirements can be too great for WAN or complex LAN environments.

Link State Routing(Dijkstra's Algorithm)

- Initial state : similar to distance vector i.e., state of link to neighbors known (up/down).
- goal: To find the path of least cost to destination.
- Basic Idea -- Every node knows how to reach its neighbors. If this info is dissemination to every node, every node ultimately has the info.
- To build the complete map of the network.
- Reliable flooding
- Process of making sure that all the nodes participating in the link state routing protocol get a copy of the link-state info from all other nodes.
- Each node sends out link-state info on its directly connected links.
- Each node that receives this, forwards it.
- Each node creates a link-state packet (LSP) that contains: ID of the node that created LSP, A list of directly connected nodes and the cost to each node,sequence number ,TTL , Eg: X receives LSP from some node Y.
- X checks to see if it already has an update from Y. If it does, it compares the sequence number in the new LSP to the one stored.

- ~~If New seq no < Old sequence number, then, discard LSP.~~
- Else -- store LSP and send the LSP to all neighbors except the one that sent the LSP.
- If no update from Y, keep it.
- Graph abstraction
- Used for computation of shortest path using Dijkstra's.
- Let N denote the set of nodes in the graph.
- l(i,j) denotes the non-negative cost or weight associated with the edge between nodes i and j.
- l(i,j) = ∞ if there is no edge between i and j.
- Remember -- each node has entire map of network.

Link state algorithm

Basic idea: two step procedure

Each source node gets a map of all nodes and link metrics (link state) of the entire network , find the shortest path on the map from the source node to all destination nodes , Broadcast of link-state information to every node i in the network broadcasts to every other node in the network: ID's of its neighbours: Ni=set of neighbours of i Distances to its neighbours: {Cij | j ∈Ni} Flooding is a popular method of broadcasting packets.

Dijkstra's algorithm

Initialization

- N: set of nodes for which shortest path already found
- Initialization: (Start with source node s)
- N = {s}, Ds = 0, "s is distance zero from itself"
- $D_j=C_{sj}$ for all j ≠s, distances of directly-connected neighbors

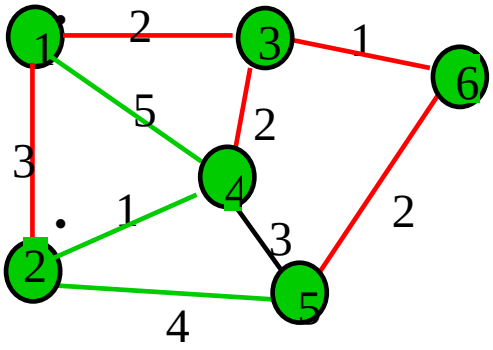
Step A: (Find next closest node i)

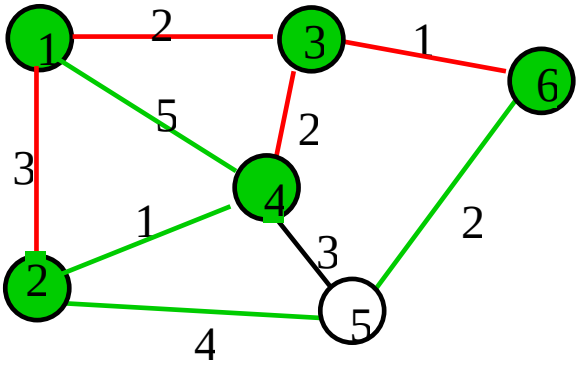
- Find i∉N such that
- D_i = min D_j for j∉ N, Add i to N
- If N contains all the nodes, stop

Step B: (update minimum costs)

- For each node j ∉ N
- D_j = min (D_j, Di+C_{ij})

- ~~Go to Step A.~~
- Execution of Dijkstra's algorithm





Iteration	N	D_2	D_3	D_4	D_5	D_6
Initial	{1}	3	2	5	∞	∞
1	{1,3}	3	2	4	∞	3
2	{1,2,3}	3	2	4	7	3
3	{1,2,3,6}	3	2	4	5	3
4	{1,2,3,4,6}	3	2	4	5	3
5	{1,2,3,4,5,6}	3	2	4	5	3

- Reaction to failure, If a link fails, Router sets link distance to infinity & floods the network with an update packet.
- All routers immediately update their link database & recalculate their shortest paths, Recovery very quick, but watch out for old update messages
- Add time stamp or sequence # to each update message
- Check whether each received update message is new
- If new, add it to database and broadcast
- If older, send update message on arriving link.

Advantages of link state over distance vector algorithm.

- Fast, loop less convergence
- Support for precise metrics, and multiple metrics if necessary (throughput, delay, cost, reliability)
- Support for multiple paths to a destination
- Algorithm can be modified to find best two paths.

Fig:Advantages of a Link-State Routing Protocol.

Routing protocol	Builds Topological map	Router can independently determine the shortest path to every network.	Convergence	A periodic/ event driven routing updates	Use of LSP
Distance vector	No	No	Slow	Generally No	No
Link State	Yes	Yes	Fast	Generally Yes	Yes

Requirements for using a link state routing protocol

- **Memory requirements**

  Typically link state routing protocols use more memory

- **Processing Requirements**

  More CPU processing is required of link state routing protocols

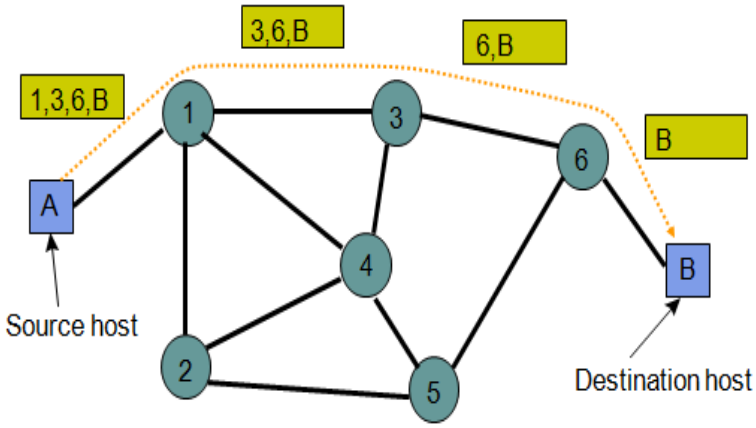
- **Bandwidth Requirements**

  Initial startup of link state routing protocols can consume lots of bandwidth.

Source routing

- Source host selects path that is to be followed by a packet determined at source.
- Explicit Routing – Not necessarily the source node
- Source routing can be
  - Strict: sequence of nodes in path inserted into header(1,3,6,B)
  - Loose: subsequence of nodes in path specified(1,6,B)
- Can use Datagram or Virtual Ckt packet switching
- Intermediate switches read next-hop address and remove address
- Source host needs link state information or access to a route server

- ~~Source routing allows the host to control the paths that its in~~formation traverses in the network.



Traffic Management

Vehicular traffic management

- Traffic lights & signals control flow of traffic in city street system
- Objective is to maximize flow with tolerable delays
- **Priority Services**
    - Police sirens
    - Cavalcade for dignitaries
    - Bus & High-usage lanes
    - Trucks allowed only at night

Packet traffic management

- Multiplexing & access mechanisms to control flow of packet traffic
- Objective is make efficient use of network resources & deliver QoS
- Priority
    - Fault-recovery packets
    - Real-time traffic

- Enterprise (high-revenue) traffic
- High bandwidth traffic.

Time Scales & Granularities

- Packet Level
  - Queuing & scheduling at multiplexing points
  - Determines relative performance offered to packets over a short time scale (microseconds)
- Flow Level
  - Management of traffic flows & resource allocation to ensure delivery of QoS (milliseconds to seconds)
  - Matching traffic flows to resources available; congestion control
- Flow-Aggregate Level
  - Routing of aggregate traffic flows across the network for efficient utilization of resources and meeting of service levels
  - "Traffic Engineering", at scale of minutes to days.

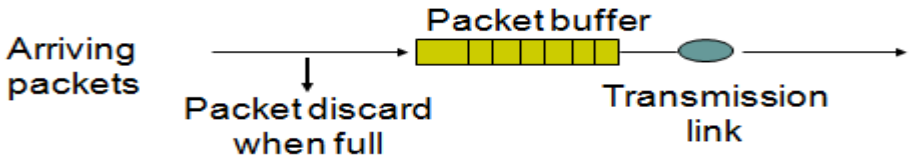
End to End QOS (Packet management at packet Level)

- Packet-level traffic management operates in the small scale on the order of packet transmission time.
- Dashed line interfere with the packet
- A packet traversing network encounters delay and possible loss at various multiplexing points
- End-to-end performance is accumulation of per-hop performances
- The performance experienced by a packet along the path is the accumulation of the performance at N queuing systems .
- Jitter

  Variability in the packet delays(difference in min and max delay)
- Packet Loss

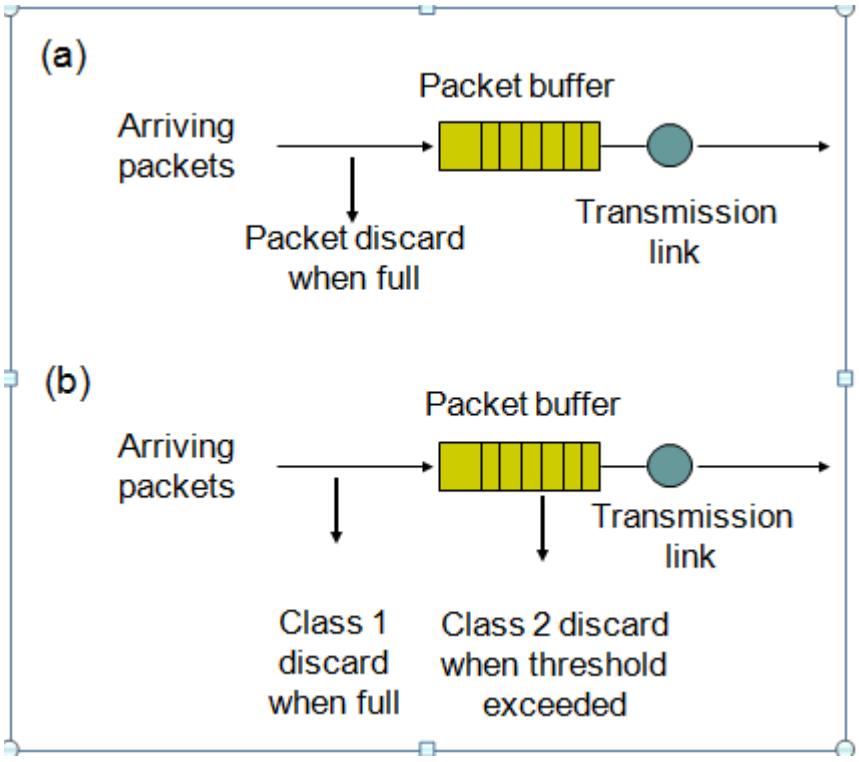
  When a packet arrives at a queuing systems that has no buffer space .

1. **FIFO Queuing.**



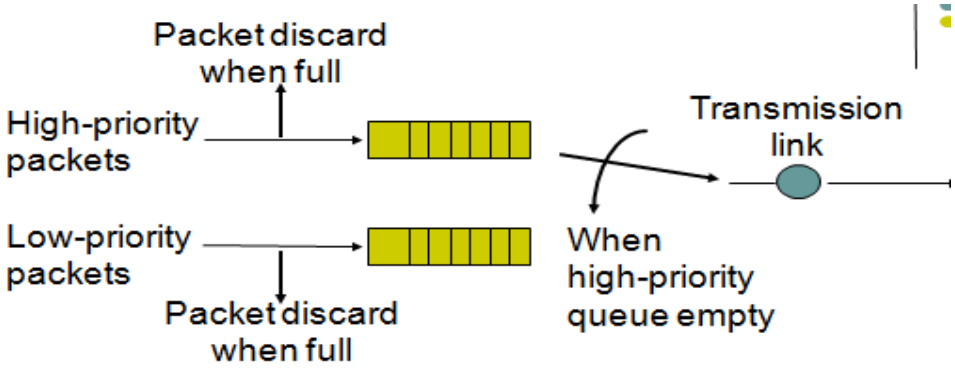
- All packet flows share the same buffer
- Transmission Discipline: First-In, First-Out
- Buffering Discipline: Discard arriving packets if buffer is full (Alternative: random discard; push out head-of-line, i.e. oldest, packet).
- The delay and loss experienced by packets in a FIFO queuing systems depend on the packet interarrival time and packet length.
- "Hogging" When a user sends a packet at a high rate and fills it will not allow other packets to enter.

1 a) FIFO Queueing with Discard Priority



- Two classes of traffic .
- When the number of packets in buffer reaches threshold ,arrival at lower access priority **(class 2)** not allowed into the system.
- Arrivals of higher access priority (**class 1**)are allowed as long as the buffer if not full.
- Packets of lower access priority will experience packet loss probability.

1.b)Head Of Line Priority Queueing

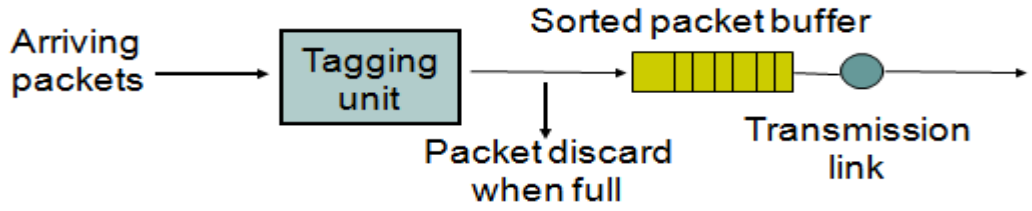


- Second Queue scheduling approach defining priority classes.
- Separate buffer for each priority class.
- High priority queue serviced until empty
- High priority queue has lower waiting time
- Buffers can be dimensioned for different loss probabilities
- Surge in high priority queue can cause low priority queue to saturate
- Fairness problem.

Features:

- Provides differential QoS
- Pre-emptive priority: lower classes invisible
- on-preemptive priority: lower classes impact higher classes through residual service times
- High-priority classes can hog all of the bandwidth & starve lower priority classes
- Need to provide some isolation between classes.

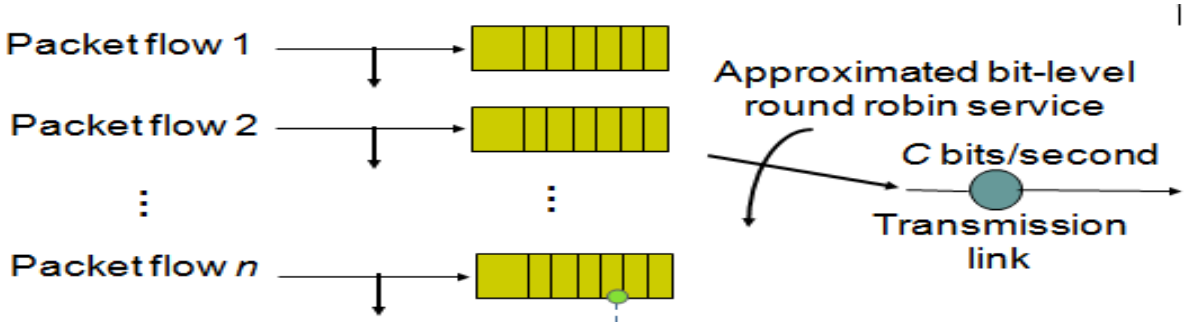
1.c)Priority Scheduling



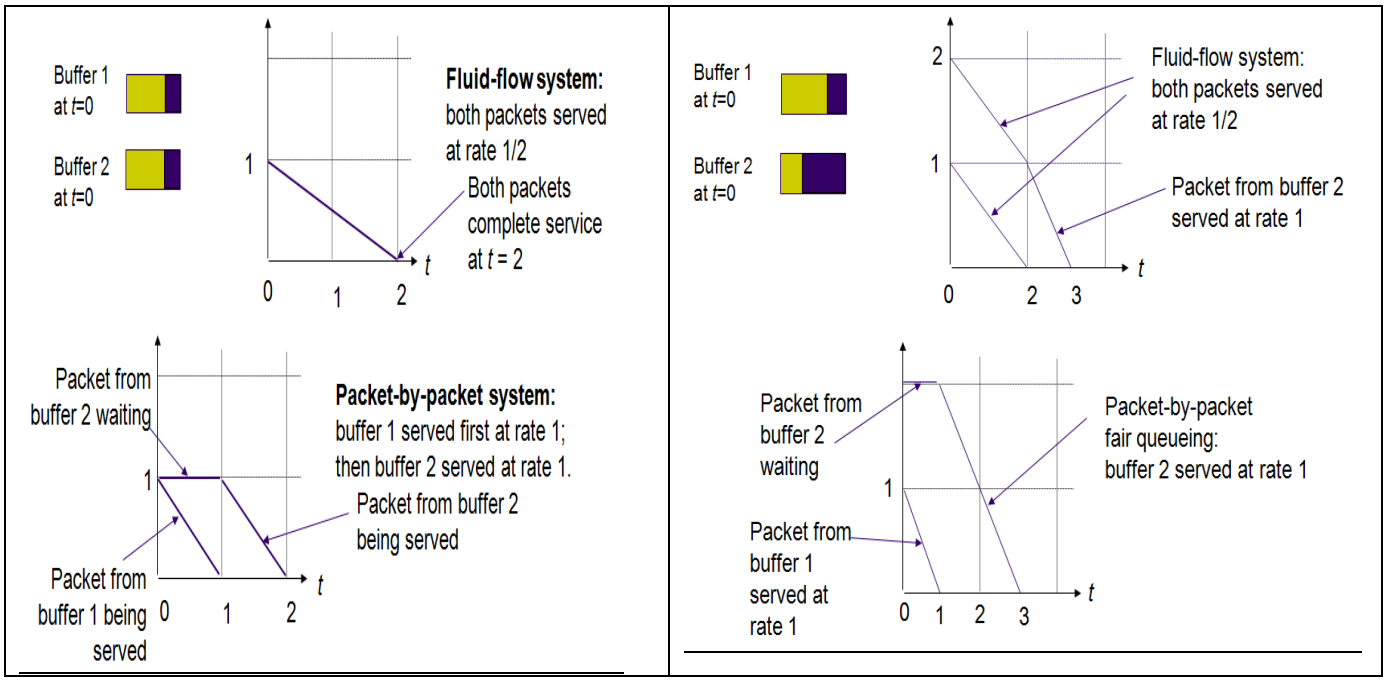
- Method for defining priority and dynamic.
- Priority tag have arrival time of the packet.
- Queue in order of "due date"
  - packets requiring low delay get earlier due date
  - packets without delay get indefinite or very long due dates.

2. Fair Queueing / Generalized Processor Sharing

- Fair Queuing provide equitable access to transmission bandwidth.
- Each user flow has its own logical buffer(fluid). prevents hogging;
- The Transmission bandwidth C bits/Sec is divided equally among the buffers that have packets to transmit.
- C bits/sec allocated equally among non-empty queues
  - transmission rate = C / n(t), where n(t)= number of non-empty queues



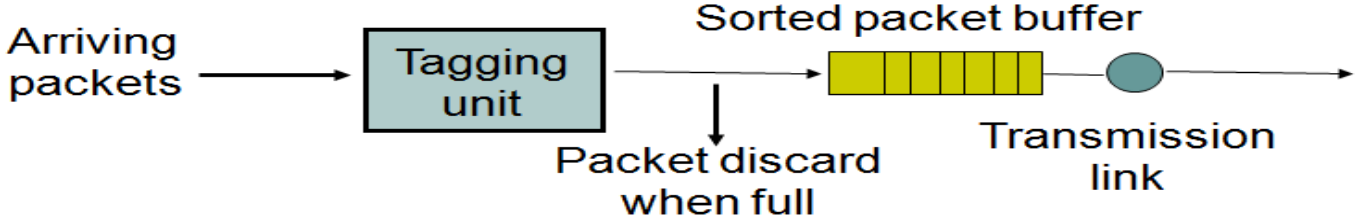
- The size of the buffer for each user flow can be selected to meet specific loss probability, so to discard the user when the buffer is full.
- Service to each non empty buffer one bit at a time in round robin fashion, decomposing the resultant bit stream into the component packets would require the introduction of frames and processing at the output .



Packet completion time is 1 and 2 second for packet and for fluid flow fair queuing 2 seconds as a whole

- Approximating fair Queuing for variable length packets is difficult .
- For ex if packet of one flow is twice than the second ,then in the long run first flow takes twice the bandwidth of the second .
- So usage of buffer is preferable for the packet transmission.
- Each packet arrives at user buffer ,completion time is calculated using fluid fair queuing system
- "Finish Tag" each time for the packet transmission check the smallest finish tag.
- Packet by packet fair queuing system.

Packet by Packet Fair Queuing



- Compute packet completion time in ideal system
  - add tag to packet(finish tag)

- sort packet in queue according to tag
- serve according to HOL

- Assume n flows, n queues
- 1 round = 1 cycle serving all n queues
- If each queue gets 1 bit per cycle, then 1 round = # active queues
- Round number = number of cycles of service that have been completed
- Assume that there are n flows, each with its own queue. Suppose for now that each queue is served one bit at a time. Let a Round consist of a cycle in which all n queues are offered service as shown in Figure.
- The actual duration of a given round is the actual number of queues n_{active} (t)that have information to transmit.
- When the number of active queues is large, the duration of a round is large; when the number of active queues is small, the rounds are short in duration.
- > Now suppose that the queues are served as in a fluid-flow system. Also suppose that the system is started at t= 0.
- > Let R t be the number of the rounds at time t, that is, the number of cycles of service to all n queues.
- > However, we let R t be a continuous function that increases at a rate that is inversely proportional to the number of active queues; $dR(t)/dt = C/n_{active}(t)$
- C=transmission capacity.R(t) changes each time as the no of active buffer changes

Computing the Finishing Time for Fluid flow fair Queueing

- Suppose that kth packet from flow i arrives at empty buffer at time (tki)and packet has length P(i,k).This packet will complete its transmission when P(I,k) elapses ,one round for each bit in the packet.
- The packet completion time will be the value of time when R(t*) reaches the value
- If packet arrives to idle queue:

  Finishing time = round number + packet size in bits

$F(i, k) = R(t_k i) + P(i, k)$ [F(i,k) as the finish tag of the packet]

- If packet arrives to active queue:

Finishing time = finishing time of last packet in queue + packet size

- On the other hand, if the k^{th} packet from the i^{th} flow arrives at a nonempty queue, then the packet will have a finish tag F(i, k) equal to the finish tag of the previous packet in its queue F(i, k — 1) plus its own packet length P(i, k); that is,,
- **$F(i, k) = F(i, k — 1) + P(i, k)$**

- The two preceding equations can be combined into the following compact equation
- **$F(i, k) = \max\{F(i, k - 1), R(t_k^i)\} + P(i, k)$ for fair queueing**
- > The actual packet completion time for the k^{th} packet in flow i in a fluid-flow fair-queueing system is the time t when R(t) reaches the value F(i, k) .
- > The relation between the actual completion time and the finish tag is not straightforward because the time required to transmit each bit varies according to the number of active queues.
- suppose that at time t =0 queue 1 has one packet of length one unit and queue 2 has one packet of length two units. A fluid-flow system services each queue at rate 1/2 as long as both queues remain nonempty. As shown in Figure 7.48, queue 1 empties at time t= 2.
- Thereafter queue 2 is served at rate 1 until it empties at time t 3.
- In the packet-by-packet fair-queueing system, the finish tag of the packet of queue the finish tag of the packet of buffer 1 is
- F(1,1)=R(0)+1=1.
- the finish tag of the packet of buffer 2 is
- F(2,1)=R(0)+2=2

3. Weighted Fair Queuing(WFQ)

- Weighted fair queueing addresses the situation in which different users have different requirements.
- As before, each user flow has its own queue, but each user flow also has a weight that determines its relative share of the bandwidth.
- Thus if buffer 1 has weight 1 and buffer 2 has weight 3, then when both buffers are nonempty, buffer 1 will receive 1/ (1+ 3)= 1/4 of the bandwidth and buffer 2 will receive 3/4 of the bandwidth.
- The transmission of the packet from buffer 2 is now completed at time t= 4/3, and the packet from buffer 1 is completed at t= 2.
- The bit-by-bit approximation to weighted fair queueing would operate by allotting each queue a different number of bits/round.
- In the preceding example, buffer 1 would receive 1 bit/ round and buffer 2 would receive 3 bits/round.
- Packet-by-packet weighted fair queueing is also easily generalized from fair queueing.

Suppose that there are n packet flows and that flow i has weight wi, then the packet-by-packet system calculates its finish tag as follows,,

- $F(i, k)= \max\{F(i, k — 1), R(t_k^i)\} + P(i, k)/wi$
  for weighted fair queueing.

  if flow i has a weight that is twice that of flow j, then the finish tag for a packet from flow i will be calculated assuming a depletion rate that is twice that of a packet from flow j.
- The finish tag of the packet from buffer 1 is
- $F(1, 1)= R(0) + 1/1=1.$
- The finish tag of the packet from buffer 2 is $F(2, 1) = R(0)+ 1/3=1/3.$
- Therefore the packet from buffer 2 is served first. The packet for buffer 2 is now completed at time t =1, and the packet from buffer 1 at time t =2.

4. Random Early Detection

- Packets produced by TCP will reduce input rate in response to network congestion
- Early drop: discard packets before buffers are full
- Random drop causes some sources to reduce rate before others, causing gradual reduction in aggregate input rate
- Algorithm:

  Maintain running average of queue length

  If Q_{avg}<minthreshold, do nothing

  If Q_{avg}>maxthreshold, drop packet

  If in between, drop packet according to probability

  Flows that send more packets are more likely to have packets dropped.

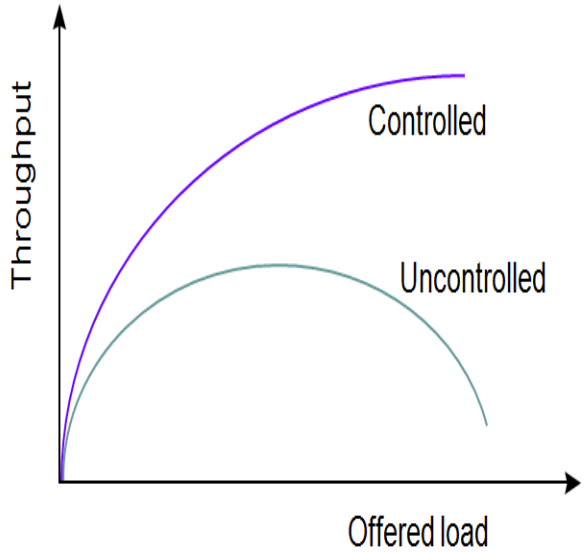
Traffic Management at the Flow Level

- At the flow level traffic management is concerned with managing the individual traffic flow to ensure that the QOS (Jitter,delay,loss) requested by the user is satisfied.
- Operates in milliseconds to seconds. The purpose of traffic management
- Control the flows of traffic and maintain performance in traffic overload–Controlled curve.

- The performance of managing the traffic flow to control congestion is congestion control.
- Larger buffer may be a temporary solution but may leads to delay.
- Congestion occurs when a surge of traffic overloads network resources
- Approaches to Congestion Control:

  Preventive Approaches: Scheduling & Reservations

  Reactive Approaches: Detect & Throttle/Discard
- Ideal effect of congestion control:
- Resources used efficiently up to capacity available



Two methods to control the congestion

1. Open loop control
2. Closed loop control
3.

1. Open loop control

- Network performance is guaranteed to all traffic flows that have been admitted into the network
- If QOS is not satisfied network rejects the traffic flow
- Initially for connection-oriented networks
- The function that makes the decision to accept or reject the traffic flow is usually called an admission control.
- Open-loop congestion control does not rely on feedback information to regulate the traffic flow.

- Key Mechanisms
  - Admission Control
  - Policing.
  - Traffic Shaping
  - QoS Guarantees and Service Scheduling

Admission control

- Admission control is an open-loop preventive congestion control scheme.
- virtual-circuit &datagram networks as well.
- Admission control typically works at the connection level but can also work at the burst level.
- A connection in datagram networks is a flow.
- At the connection level the function is called a connection admission control (CAC).
- At the burst level, it is called a burst admission control.
- When a source requests a connection setup, CAC has to decide whether to accept or reject the connection.
- If the QoS of all the sources (including the new one) that share the same path can be satisfied, the connection is accepted;
- otherwise, the connection is rejected.
- Specify requirements:

  Peak, Avg., Min Bit rate,Maximum burst size,Delay, Loss requirement

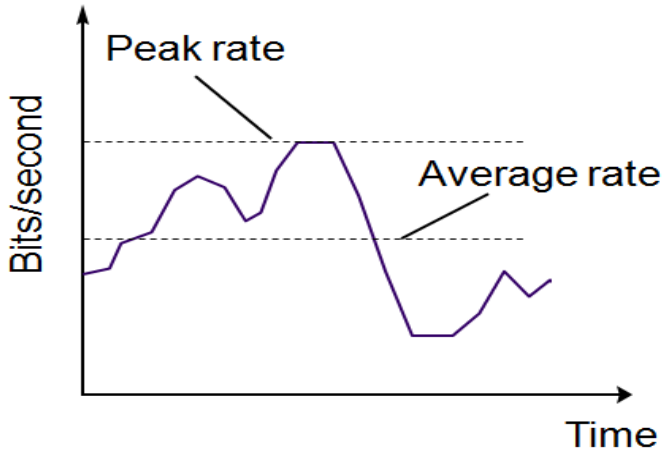
Peak Rate :Maximum rate that the source will generate its packets.

Average Rate: Average rate that the source will generate its packets .

Burst size: The maximum burst size is the maximum length of time the traffic can be generated at the peak rate.

- Flows negotiate contract with network
- Each source must specify its traffic flow, described by a set of traffic parameters called the traffic descriptor during the connection setup.
- Network computes resources needed
- “Effective” bandwidth

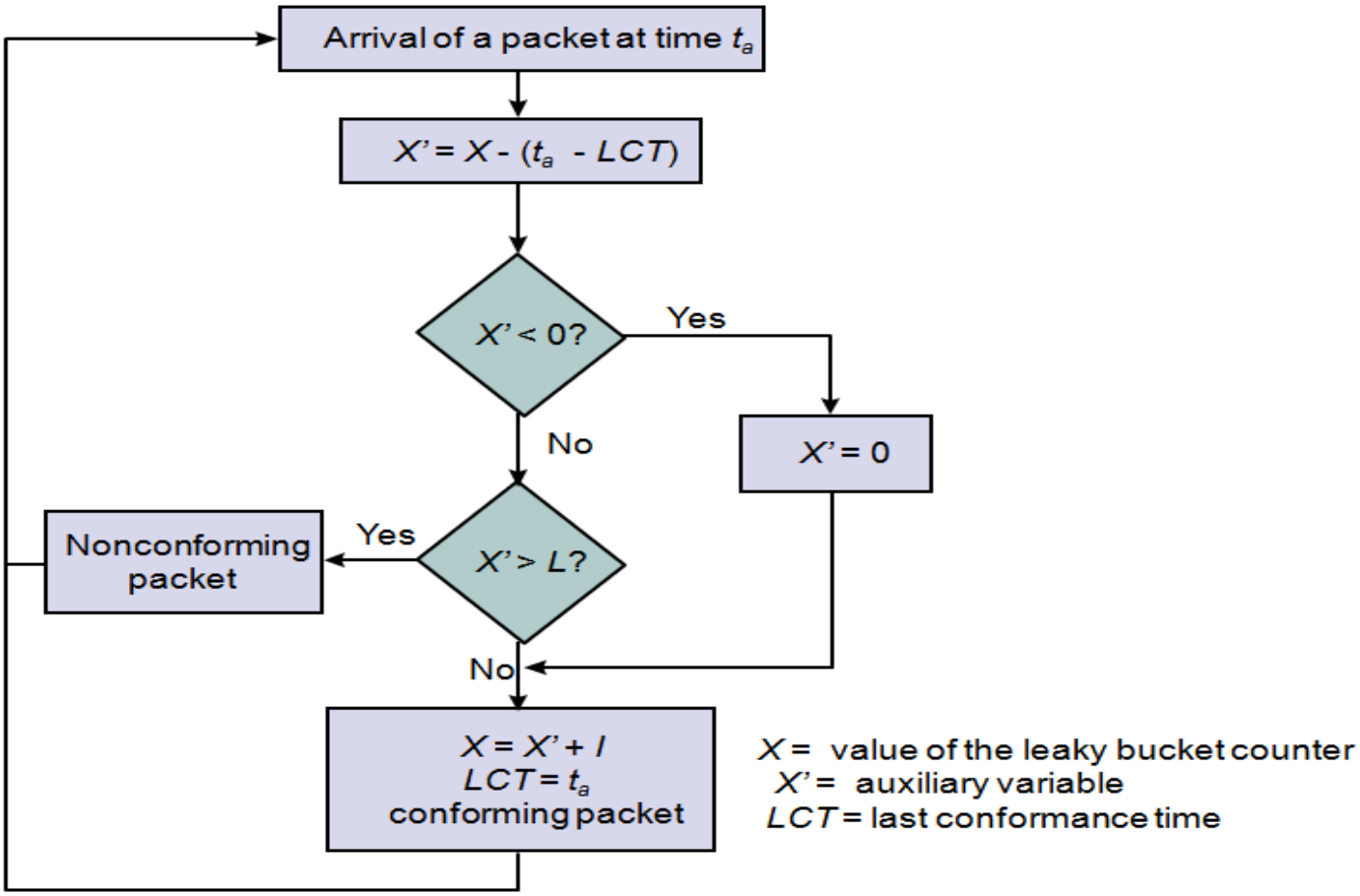
- If flow accepted, network allocates resources to ensure QoS delivered as long as source conforms to contract



Policing

- Once a connection is accepted by a CAC, the QOS will be satisfied as long as the source obeys the traffic descriptor that it specified during the connection setup.
- To prevent the source from violating its contract, the network may want to monitor the traffic flow during the connection period.
- The process of monitoring and enforcing the traffic flow is called traffic policing.
- When the traffic violates the agreed-upon contract, the network may choose to discard or tag the nonconforming traffic giving it lower priority.
- If congestion occurs, tagged packets are discarded first
- Leaky Bucket Algorithm is the most commonly used policing mechanism
- Bucket has specified leak rate
- Bucket has specified depth to accommodate variations in arrival rate
- Arriving packet is conforming if it does not result in overflow

Leaky bucket algorithm



- Leaky Bucket algorithm can be used to police arrival rate of a packet stream
- Leak rate corresponds to long-term rate
- Bucket depth corresponds to maximum allowable burst arrival 1 packet per unit time
- Assume constant-length packet. Depletion rate: 1 packet per unit time
- L+I = Bucket Depth

- I = increment per arrival, nominal interarrival time
- Let X = bucket content at last conforming packet arrival
- Let t_a – last conforming packet arrival time = depletion in bucket algorithm
- At the arrival of the first packet, the content of the bucket X is set to zero and the last conforming time (LCT) is set to the arrival time of the first packet.
- The depth of the bucket is L + I, where L depends on the maximum burst size.
- If the traffic is expected to be bursty, then the value of L should be made large.
- At the arrival of the kth packet, the auxiliary variable Xj records the difference between the bucket content at the arrival of the last conforming packet and the interarrival time between the last conforming packet and the kth packet.
- The auxiliary variable is constrained to be nonnegative. If the auxiliary variable is greater than L, the packet is considered nonconforming.

- ~~Otherwise, the packet is conforming. The bucket content and~~ the arrival time of the packet are then updated.

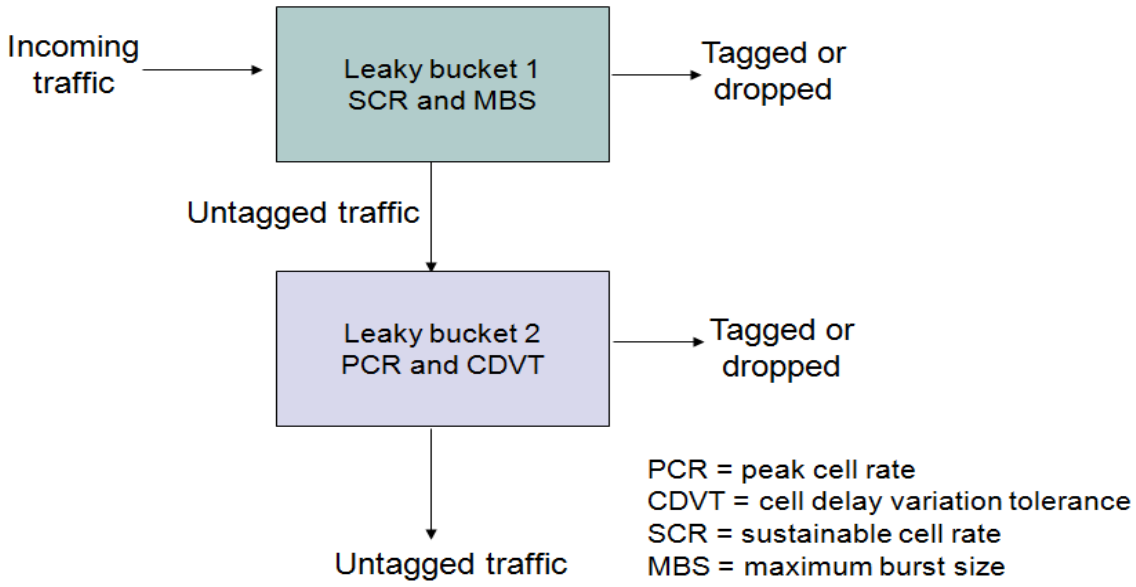
Policing parameters

- Inverse of I is the Sustainable rate which is the long-term average rate allowed for the conforming traffic. Suppose the peak rate of a given traffic is denoted by R and its inverse is T; that is, T=1/R. Then the maximum burst size is given by
- T = 1 / peak rate
- MBS = maximum burst size
- I = nominal interarrival time
- [x] gives the greatest integer less than or equal to x.
- The first packet increases the bucket content to I.
- After the first packet the bucket content increases by the amount of (I — T) for each packet arrival at the peak rate.
- Thus we can have approximately L/(I —T) additional conforming packets.[MBS approximates the bursty traffic].

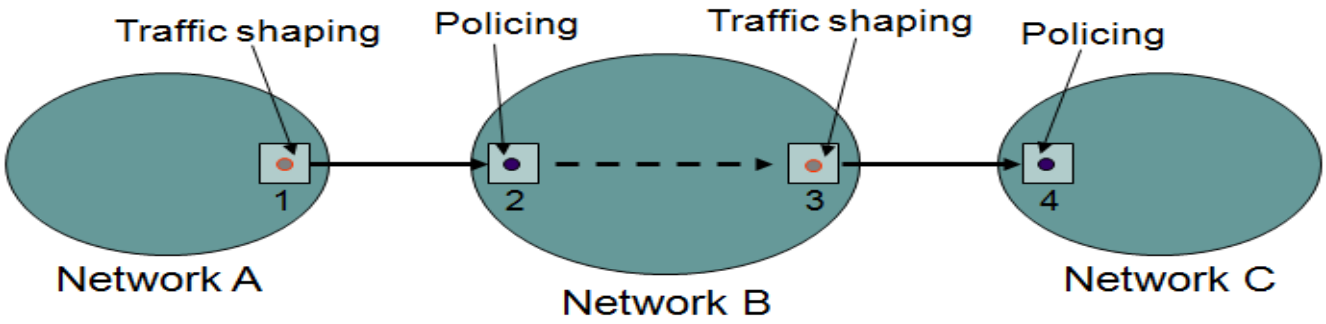
Dual Leaky Bucket

A combination of leacky buckets can be used to police multiple traffic parameters-Dual Leaky bucket .

- The traffic is first checked for the sustainable rate at first leaky bucket .
- The nonconforming packets at the first bucket are dropped or tagged.
- Confirming packets are then checked for the peak rate at the second bucket, nonconfirm-drop.

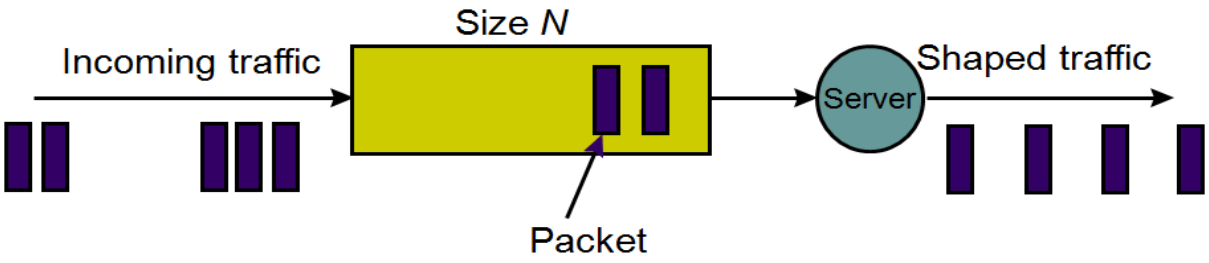


○ Traffic shaping



- When a source tries to send packets, it may not know what its traffic looks like.
- If the source wants to ensure that the traffic conforms to the parameters specified in the leaky bucket policing device, it should first alter the traffic.
- The process of altering a traffic flow to another flow is called traffic shaping(used to make the traffic smoother).
- Networks police the incoming traffic flow
- Traffic shaping is used to ensure that a packet stream conforms to specific parameters
- Networks can shape their traffic prior to passing it to another network

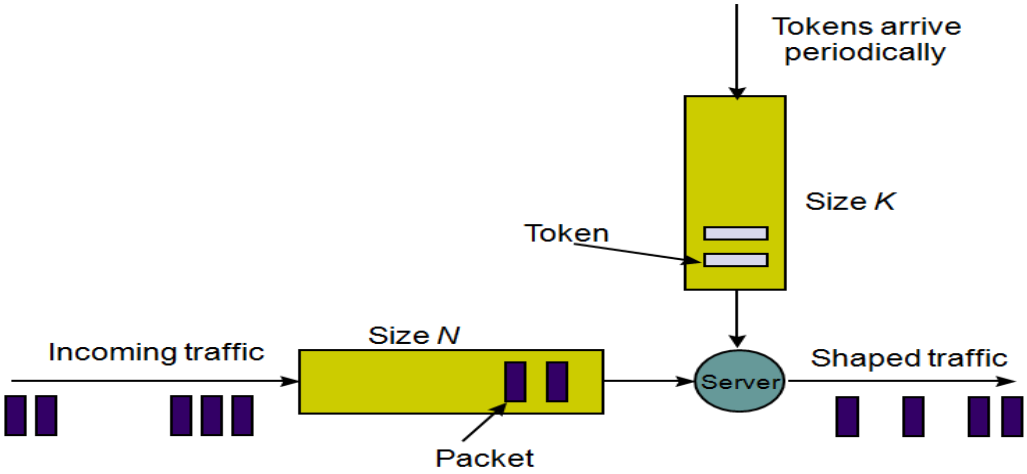
Leaky Bucket Traffic Shaper



- A leaky bucket traffic shaper is a very simple device.
- Incoming packets are first stored in a buffer(fixed length) and served periodically ,so that stream of packets at the output is smooth.
- Buffer stores momentary bursts.
- Buffer size defines the maximum burst that can be accommodated and redetection once the buffer if full.
- A policing device checks and passes each packet on the fly and introduce certain delays for those packets who will arrive earlier than the scheduled departures.
- Buffer incoming packets
- Play out periodically to conform to parameters

- ~~Surges in arrivals are buffered & smoothed out~~
- Possible packet loss due to buffer overflow
- Too restrictive, since conforming traffic does not need to be completely smooth, who generates variable rate traffic

Token Bucket Traffic Shaper



- Regulates only the packets that are not conforming.
- Packets that are deemed conforming are passed through without further delay.
- The token bucket is a simple extension of the leaky bucket.
- Tokens are generated periodically at a constant rate and are stored in a token bucket. If the token bucket is full, arriving tokens are discarded.
- A packet from the buffer can be taken out only if a token in the token bucket can be drawn. If the token bucket is empty, arriving packets have to wait in the packet buffer.
- Token rate regulates transfer of packets
- If sufficient tokens available, packets enter network without delay
- K determines how much burstiness allowed into the network
- When the buffer has a backlog of packets when the token bucket is empty. These backlogged packets have to wait for new tokens to be generated before they can be transmitted out.
- When the token bucket is not empty. Packets are transmitted out as soon as they arrive without having to wait in the buffer, since there is a token to draw for an arriving packet. Thus the burstiness of the traffic is preserved in this case.

DIFFERENCE BETWEEN LEAKY BUCKET AND TOKEN BUCKET ALGORITHM

TOKEN BUCKET	LEAKY BUCKET
Token dependent.	Token independent.
If bucket is full token are discarded, but not the packet.	If bucket is full packet or data is discarded.
Packets can only transmitted when there are enough token	Packets are transmitted continuously.
It allows large bursts to be sent faster rate after that constant rate	It sends the packet at constant rate
It saves token to send large bursts.	It does not save token.

Qos guarantees and service scheduling

- packet delay across a network can be guaranteed to be less than a given value. The technique makes use of a token bucket shaper and weighted fair-queueing scheduling.
- Let b be the bucket size in bytes and let r be the token rate in bytes/second. Then in a time period T, the maximum traffic that can exit the traffic shaper is b+rT bytes.
- Token Bucket Shaping Effect
- The token bucket constrains the traffic from a source to be limited to b + r t bits in an interval of length.
- Suppose we apply this traffic to two buffers in tandem each served by transmission lines of speed R bytes/second with R >r. [Assume that the two multiplexers are empty and not serving any other flows.]
- Assume that the token bucket allows an immediate burst of b bytes to exit and appear at the first multiplexer at t= 0, so the multiplexer buffer surges to b bytes at that instant.
- Immediately after t= 0, the token bucket allows information to flow to the multiplexer at a rate of r bytes/second, and the transmission line drains the multiplexer at a rate of R bytes/second.
- Thus the buffer occupancy falls at a rate of R-r bytes/second
- Packet transfer with Delay Guarantees.

- Assume fluid flow for information,Token bucket allows burst of b bytes 1 & then r bytes/second, Since R>r, buffer content @ 1 never greater than b byte
- Thus delay @ mux <b/R , Rate into second mux is r<R, so bytes are never delayed.
- Assume traffic shaped to parameters b & r.Schedulers give flow at least rate R>r ,H hop path m is maximum packet size for the given flow,M maximum packet size in the network Rj transmission rate in jth hop
- Maximum end-to-end delay that can be experienced by a packet from flow i is:

$$D \leq \frac{b}{R} + \frac{(H-1)m}{R} + \sum_{j=1}^{H} \frac{M}{R_j}$$

- Scheduling for Guaranteed Service for IP networks

2. Closed-Loop Flow Control

- Congestion control feedback information to regulate flow from sources into network
- Based on buffer content, link utilization, etc.
- Examples: TCP at transport layer; congestion control at ATM level

1. End-to-end vs. Hop-by-hop

- Delay in effecting control

2. Implicit vs. Explicit Feedback

- Source deduces congestion from observed behavior
- Routers/switches generate messages alerting to congestion.

1a . End- End Approach

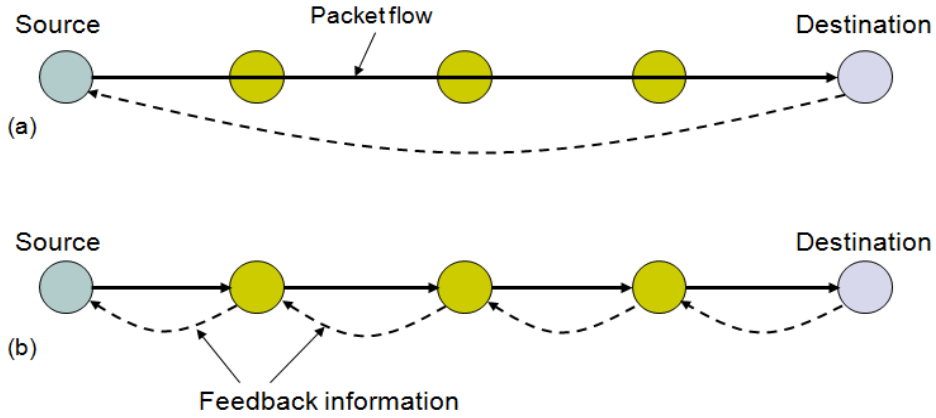
1.E-E approach ,the feedback information about the state of the network is propagated back to the source that can regulate the traffic flow rate (a).

2.The feedback information may be forwarded directly by a node that detects congestion, or may be forwarded to the destination first which then forwards it to the source.

1.b. Hop-Hop approach

- React faster than End -End (b)
- Shorter propagation delay

- ~~The state of the information is sent to the upstream node.~~
- This back –pressure signal will be sent from the detected node to the source.



2. **Implicit vs explicit feedback**

2 a.Explicit feedback:

- Feedback info can be implicit or explicit
- With explicit the node detects congestion initiates an explicit message arrives at the source for notification.-choke packet or piggyback.
- Use one bit to tell weather congestion or not.
- Other way is to specify the “desired rate” expected by the receiver.

2.b Implicit feedback:

- No explicit message is forwarded.
- Source has to depend on surrogate information to reduce congestion.-timeout.