Module 7

Routing and Congestion Control

Lesson 5

Congestion Control

Specific Instructional Objectives

On completion of this lesson, the students will be able to:

- Explain the cause for congestion
- Understand the effects of congestion
- Understand various open-loop and close-loop congestion control techniques:
    - The leaky bucket algorithm
    - The token bucket algorithm
    - Admission Control
    - Choke packets
    - Weighted fair queuing
    - Load shedding
    - Resource reservation
- Distinguish between flow and congestion control

7.5.1 Introduction

As Internet can be considered as a *Queue of packets*, where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue. So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them. This degrades the performance, and such a situation is termed as *Congestion*. Main reason of congestion is more number of packets into the network than it can handle. So, the objective of congestion control can be summarized as to maintain the number of packets in the network below the level at which performance falls off dramatically. The nature of a Packet switching network can be summarized in following points:

- A network of queues
- At each node, there is a queue of packets for each outgoing channel
- If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound
- When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly

When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have too be rejected due to transmission errors). And then the number delivered is proportional to the number of packets sent. However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets. This tends to make matter worse. At very high traffic, performance collapse completely, and almost no packet is delivered. In the following sections, the causes of congestion, the effects of congestion and various congestion control techniques are discussed in detail.

7.5.2 Causes Of Congestion

Congestion can occur due to several reasons. For example, if all of a sudden a stream of packets arrive on several input lines and need to be out on the same output line, then a long queue will be build up for that output. If there is *insufficient memory* to hold these packets, then packets will be lost (dropped). Adding more memory also may not help in certain situations. If router have an infinite amount of memory even then instead of congestion being reduced, it gets worse; because by the time packets gets at the head of the queue, to be dispatched out to the output line, they have already timed-out (repeatedly), and duplicates may also be present. All the packets will be forwarded to next router up to the destination, all the way only increasing the load to the network more and more. Finally when it arrives at the destination, the packet will be discarded, due to time out, so instead of been dropped at any intermediate router (in case memory is restricted) such a packet goes all the way up to the destination, increasing the network load throughout and then finally gets dropped there.

Slow processors also cause Congestion. If the router CPU is slow at performing the task required for them (Queuing buffers, updating tables, reporting any exceptions etc.), queue can build up even if there is excess of line capacity. Similarly, *Low-Bandwidth* lines can also cause congestion. Upgrading lines but not changing slow processors, or vice-versa, often helps a little; these can just shift the bottleneck to some other point. The real problem is the mismatch between different parts of the system.

Congestion tends to feed upon itself to get even worse. Routers respond to overloading by dropping packets. When these packets contain TCP segments, the segments don't reach their destination, and they are therefore left unacknowledged, which eventually leads to timeout and retransmission.

So, the major cause of congestion is often the *bursty* nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion because other causes just act as an enzyme which boosts up the congestion when the traffic is bursty (i.e., other causes just add on to make the problem more serious, main cause is the bursty traffic).

This means that when a device sends a packet and does not receive an acknowledgment from the receiver, in most the cases it can be assumed that the packets have been dropped by intermediate devices due to congestion. By detecting the rate at which segments are sent and not acknowledged, the source or an intermediate router can infer the level of congestion on the network. In the following section we shall discuss the ill effects of congestion.

7.5.3 Effects of Congestion

Congestion affects two vital parameters of the network performance, namely *throughput* and *delay*. In simple terms, the throughput can be defined as the percentage utilization of the network capacity. Figure 7.5.1(a) shows how throughput is affected as offered load

increases. Initially throughput increases linearly with offered load, because utilization of the network increases. However, as the offered load increases beyond certain limit, say 60% of the capacity of the network, the throughput drops. If the offered load increases further, a point is reached when not a single packet is delivered to any destination, which is commonly known as *deadlock* situation. There are three curves in Fig. 7.5.1(a), the ideal one corresponds to the situation when all the packets introduced are delivered to their destination up to the maximum capacity of the network. The second one corresponds to the situation when there is no congestion control. The third one is the case when some congestion control technique is used. This prevents the throughput collapse, but provides lesser throughput than the ideal condition due to overhead of the congestion control technique.

The delay also increases with offered load, as shown in Fig. 7.5.1(b). And no matter what technique is used for congestion control, the delay grows without bound as the load approaches the capacity of the system. It may be noted that initially there is longer delay when congestion control policy is applied. However, the network without any congestion control will saturate at a lower offered load.

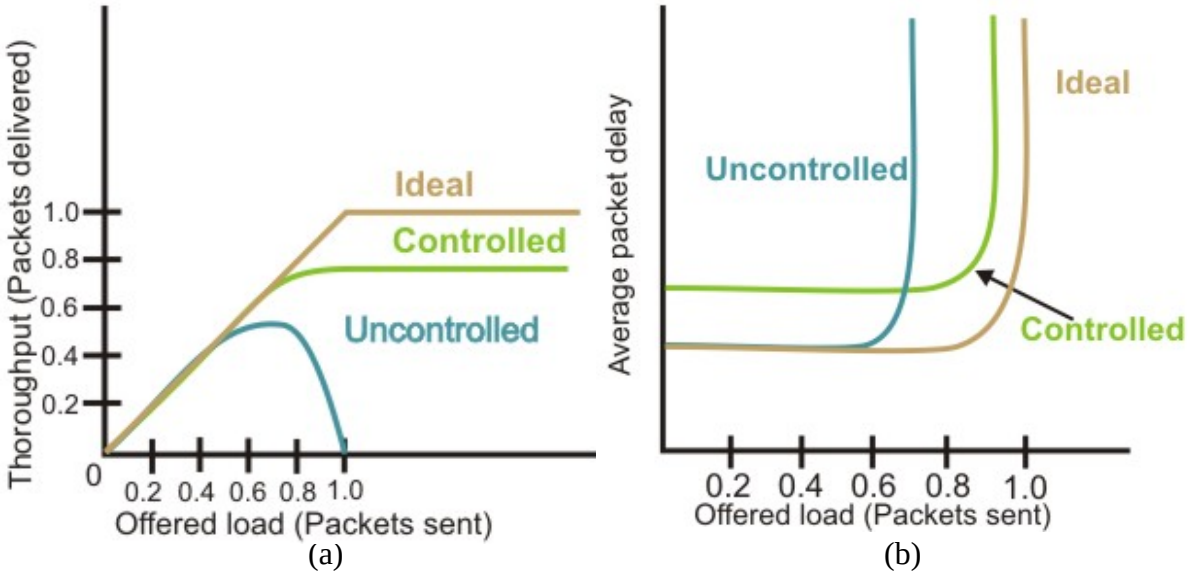


Figure 7.5.1 (a) Effect of congestion on throughput (b) Effect of congestion on delay

7.5.4 Congestion Control Techniques

Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. As shown in Fig. 7.5.2, the congestion control techniques can be broadly classified two broad categories:

- **Open loop:** Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State.
- **Close loop:** Protocols that allow system to enter congested state, detect it, and remove it.

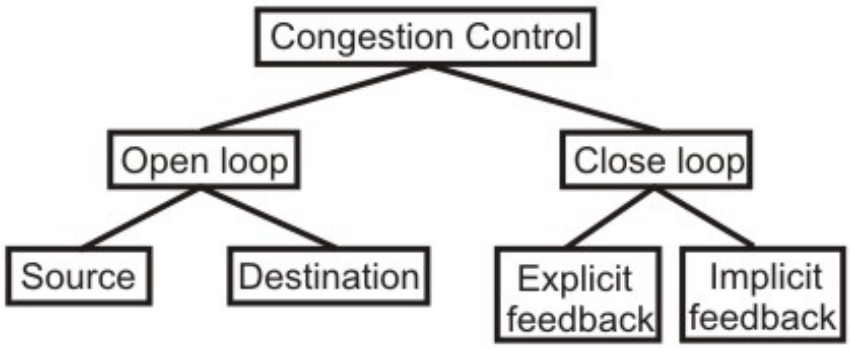


Figure 7.5.2 Congestion control categories

The first category of solutions or protocols attempt to solve the problem by a good design, at first, to make sure that it doesn't occur at all. Once system is up and running midcourse corrections are not made. These solutions are somewhat static in nature, as the policies to control congestion don't change much according to the current state of the system. Such Protocols are also known as *Open Loop* solutions. These rules or policies include deciding upon when to accept traffic, when to discard it, making scheduling decisions and so on. Main point here is that they make decision without taking into consideration the current state of the network. The open loop algorithms are further divided on the basis of whether these acts on source versus that act upon destination.

The second category is based on the concept of feedback. During operation, some system parameters are measured and feed back to portions of the subnet that can take action to reduce the congestion. This approach can be divided into 3 steps:

- Monitor the system (network) to detect whether the network is congested or not and what's the actual location and devices involved.
- To pass this information to the places where actions can be taken
- Adjust the system operation to correct the problem.

These solutions are known as *Closed Loop* solutions**.** Various Metrics can be used to monitor the network for congestion. Some of them are: the average queue length, number of packets that are timed-out, average packet delay, number of packets discarded due to lack of buffer space, etc. A general feedback step would be, say a router, which detects the congestion send special packets to the source (responsible for the congestion) announcing the problem. These extra packets increase the load at that moment of time, but are necessary to bring down the congestion at a later time. Other approaches are also used at times to curtail down the congestion. For example, hosts or routers send out probe packets at regular intervals to explicitly ask about the congestion and source itself regulate its transmission rate, if congestion is detected in the network. This kind of approach is a *pro-active* one, as source tries to get knowledge about congestion in the network and act accordingly.

Yet another approach may be where instead of sending information back to the source an intermediate router which detects the congestion send the information about the congestion to rest of the network, piggy backed to the outgoing packets. This approach will in no way put an extra load on the network (by not sending any kind of special packet for feedback). Once the congestion has been detected and this information has been passed to a place where the action needed to be done, then there are two basic approaches that can overcome the problem. These are: either to increase the resources or to decrease the load. For example, separate dial-up lines or alternate links can be used to increase the bandwidth between two points, where congestion occurs. Another example could be to decrease the rate at which a particular sender in transmitting packets out into the network.

The closed loop algorithms can also be divided into two categories, namely *explicit feedback* and *implicit feedback* algorithms. In the explicit approach, special packets are sent back to the sources to curtail down the congestion. While in implicit approach, the source itself acts pro-actively and tries to deduce the existence of congestion by making local observations.

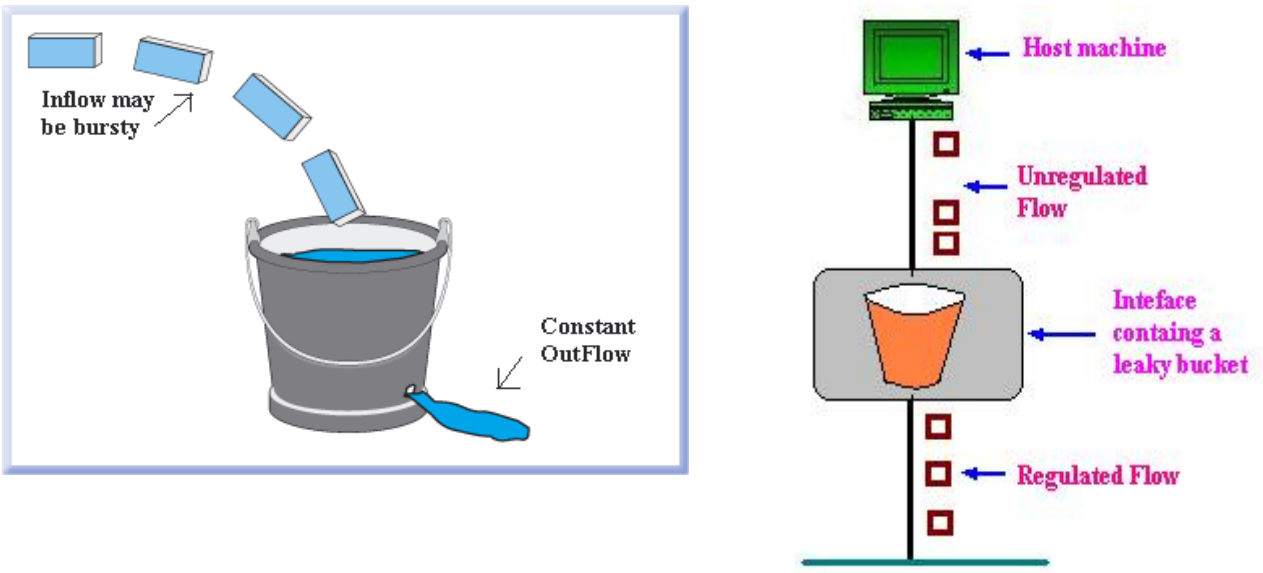
In the following sections we shall discuss about some of the popular algorithms from the above categories.

7.5.5 Leaky Bucket Algorithm

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. This scenario is depicted in figure 7.5.3(a). Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn't appear in the output stream through the hole underneath).
The same idea of leaky bucket can be applied to packets, as shown in Fig. 7.5.3(b). Conceptually each network interface contains a *leaky bucket*. And the following steps are performed:

- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- In practice the bucket is a finite queue that outputs at a finite rate.

This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.



(a) (b)

Figure 7.5.3(a) Leaky bucket (b) Leaky bucket implementation

7.5.6 Token Bucket Algorithm

The leaky bucket algorithm described above, enforces a rigid pattern at the output stream, irrespective of the pattern of the input. For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data. Token Bucket algorithm provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals. Main steps of this algorithm can be described as follows:

- In regular intervals tokens are thrown into the bucket.
- The bucket has a maximum capacity.
- If there is a ready packet, a token is removed from the bucket, and the packet is send.
- If there is no token in the bucket, the packet cannot be send.

Figure 7.5.4 shows the two scenarios before and after the tokens present in the bucket have been consumed. In Fig. 7.5.4(a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Fig. 7.5.4(b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

The token bucket algorithm is less restrictive than the leaky bucket algorithm, in a sense that it allows bursty traffic. However, the limit of burst is restricted by the number of tokens available in the bucket at a particular instant of time.

The implementation of basic token bucket algorithm is simple; a variable is used just to count the tokens. This counter is incremented every *t seconds* and is decremented whenever a packet is sent. Whenever this counter reaches zero, no further packet is sent out as shown in Fig. 7.5.5.

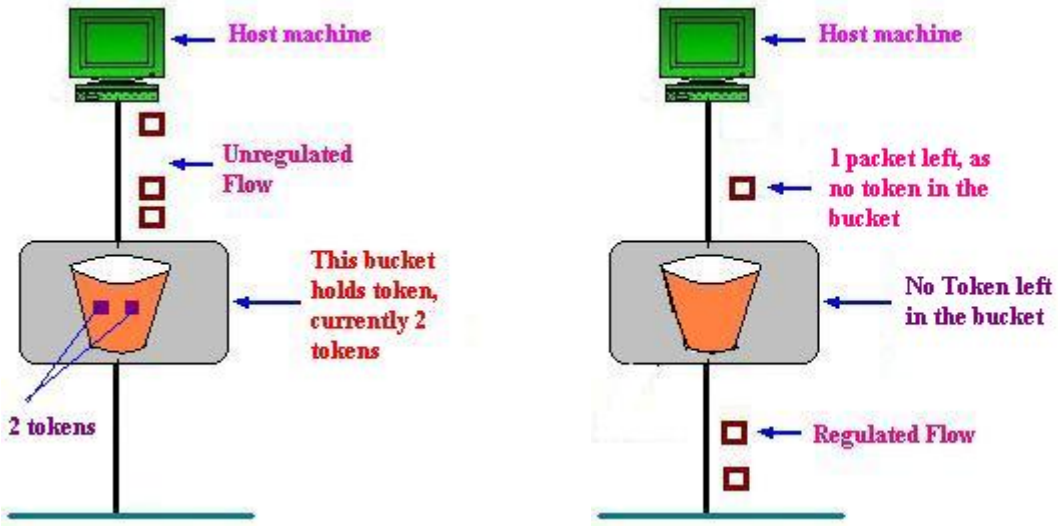


Figure 7.5.4(a) Token bucket holding two tokens, before packets are send out, (b) Token bucket after two packets are send, one packet still remains as no token is left

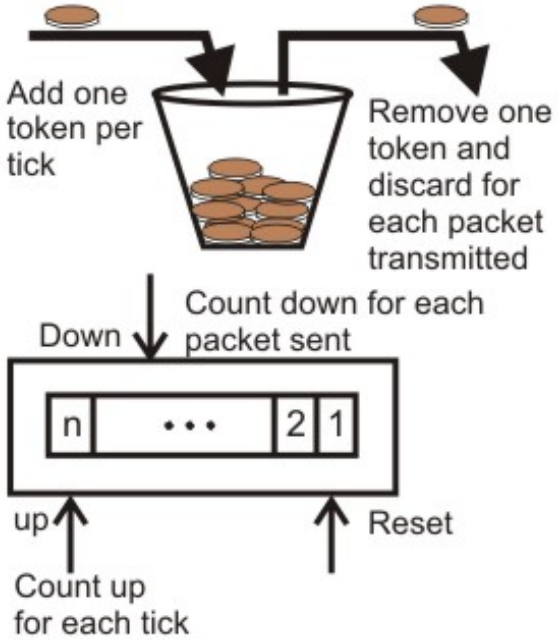


Figure 7.5.5 Implementation of the Token bucket algorithm

7.5.7 Congestion control in virtual Circuit

Till now we have discussed two open loop algorithms, where the policy decisions are made in the beginning, irrespective of the current state. Both leaky bucket algorithm and token bucket algorithm are open loop algorithms.

In this section we shall have a look at how the congestion is tackled in a virtual-circuit network. *Admission control* is one such closed-loop technique, where action is taken once congestion is detected in the network. Different approaches can be followed:

- Simpler one being: do not set-up new connections, once the congestion is signaled. This type of approach is often used in normal telephone networks. When the exchange is overloaded, then no new calls are established.
- Another approach, which can be followed is: to allow new virtual connections, but route these carefully so that none of the congested router (or none of the problem area) is a part of this route.
- Yet another approach can be: To negotiate different parameters between the host and the network, when the connection is setup. During the setup time itself, Host specifies the volume and shape of traffic, quality of service, maximum delay and other parameters, related to the traffic it would be offering to the network. Once the host specifies its requirement, the resources needed are reserved along the path, before the actual packet follows.

7.5.8 Choke Packet Technique

The *choke packet* technique, a closed loop control technique**,** can be applied in both virtual circuit and datagram subnets. Each router monitors its resources and the utilization at each of its output line. There is a threshold set by the administrator, and whenever any of the resource utilization crosses this threshold and action is taken to curtail down this. Actually each output line has a utilization associated with it, and whenever this utilization crosses the threshold, the output line enters a “warning” state. If so, the router sends a *choke packet* back to the source, giving it a feedback to reduce the traffic. And the original packet is tagged (a bit is manipulated in the header field) so that it will not generate other choke packets by other intermediate router, which comes in place and is forwarded in usual way. It means that the first router (along the way of a packet), which detects any kind of congestion, is the only one that sends the choke packets.

When the source host gets the choke packet, it is required to reduce down the traffic send out to that particular destination (choke packet contains the destination to which the original packet was send out). After receiving the choke packet the source reduces the traffic by a particular fixed percentage, and this percentage decreases as the subsequent choke packets are received. Figure 7.5.6 depicts the functioning of choke packets.

For Example, when source A receives a choke packet with destination B at first, it will curtail down the traffic to destination B by 50%, and if again after affixed duration of time interval it receives the choke packet again for the same destination, it will further curtail down the traffic by 25% more and so on. As stated above that a source will entertain another subsequent choke packet only after a fixed interval of time, not before that. The reason for this is that when the first choke packet arrives at that point of time other packets destined to the same destination would also be there in the network and

they will generate other choke packets too, the host should ignore these choke packets which refer to the same destination for a fixed time interval.

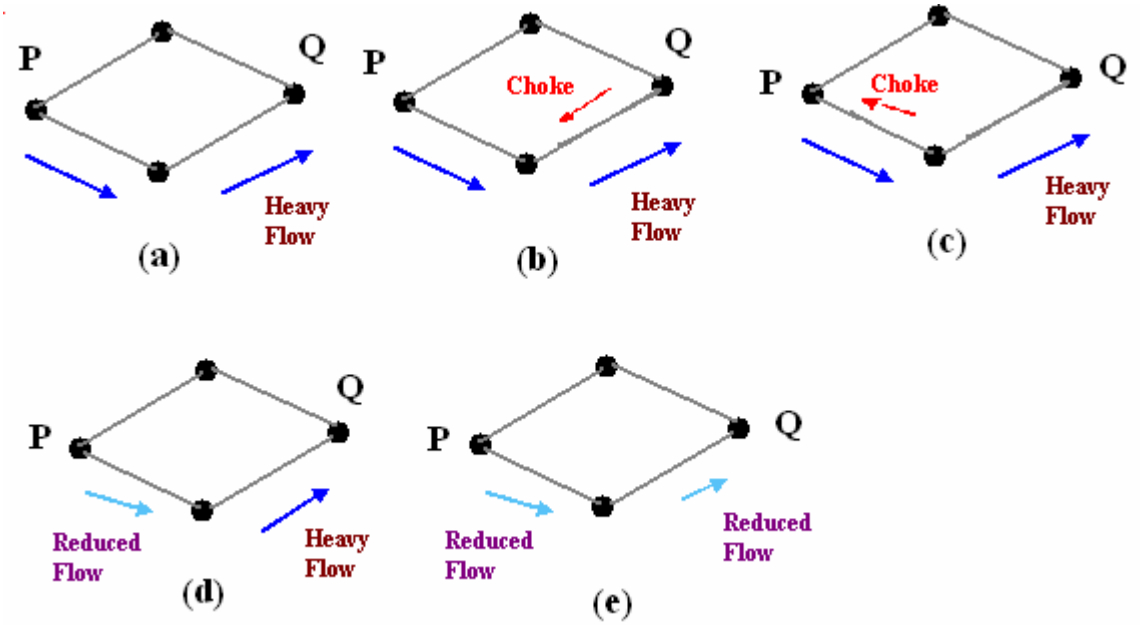


Figure 7.5.6 Depicts the functioning of choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches P, (d) P reduces the flow and send a reduced flow out, (e) Reduced flow reaches node Q

7.5.9 Hop-by Hop Choke Packets

This technique is an advancement over Choked packet method. At high speed over long distances, sending a packet all the way back to the source doesn't help much, because by the time choke packet reach the source, already a lot of packets destined to the same original destination would be out from the source. So to help this, Hop-by-Hop Choke packets are used. In this approach, the choke packet affects each and every intermediate router through which it passes by. Here, as soon as choke packet reaches a router back to its path to the source, it curtails down the traffic between those intermediate routers. In this scenario, intermediate nodes must dedicate few more buffers for the incoming traffic as the outflow through that node will be curtailed down immediately as choke packet arrives it, but the input traffic flow will only be curtailed down when choke packet reaches the node which is before it in the original path. This method is illustrated in Fig. 7.5.7.

As compared to choke packet technique, hop-by-hop choke packet algorithm is able to restrict the flow rapidly. As can been seen from Figures 7.5.6 and 7.5.7, one-step reduction is seen in controlling the traffic, this single step advantage is because in our example there is only one intermediate router. Hence, in a more complicated network, one can achieve a significant advantage by using hop-by-hop choke packet method.

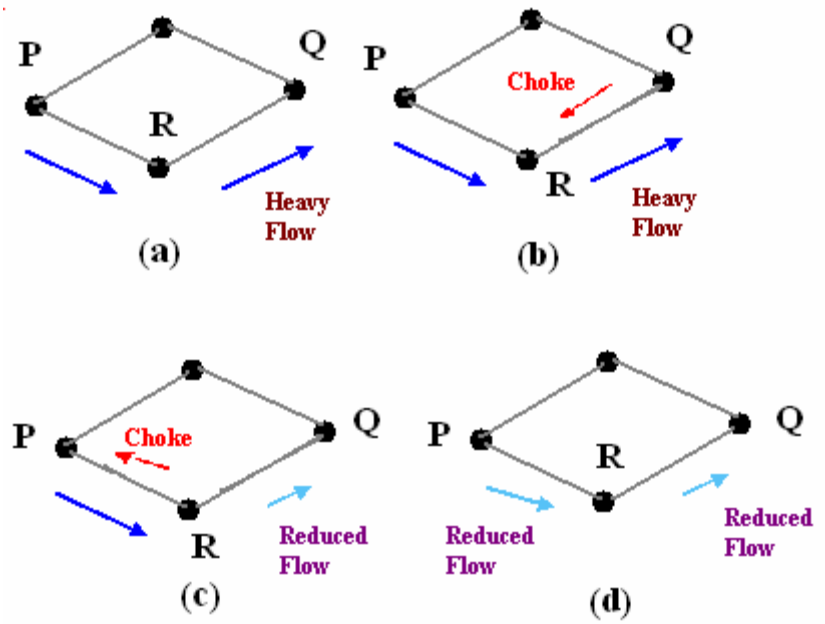


Figure 7.5.7 Depicts the functioning of Hop-by-Hop choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches R, and the flow between R and Q is curtail down, Choke packer reaches P, and P reduces the flow out

7.5.10 Load Shedding

Another simple closed loop technique is *Load Shedding***;** it is one of the simplest and more effective techniques. In this method, whenever a router finds that there is congestion in the network, it simply starts dropping out the packets. There are different methods by which a host can find out which packets to drop. Simplest way can be just choose the packets randomly which has to be dropped. More effective ways are there but they require some kind of cooperation from the sender too. For many applications, some packets are more important than others. So, sender can mark the packets in priority classes to indicate how important they are. If such a priority policy is implemented than intermediate nodes can drop packets from the lower priority classes and use the available bandwidth for the more important packets.

7.5.11 Slow Start - a Pro-active technique

This is one of the pro-active techniques, which is used to avoid congestion. In the original implementation of TCP, as soon as a connection was established between two devices, they could each go “hog wild”, sending segments as fast as they liked as long as there was room in the other devices receive window. In a busy internet, the sudden appearance of a large amount of new traffic could aggravate any existing congestion.

To alleviate this, modern TCP devices are restrained in the rate at which they initially send segments. Each sender is at first restricted to sending only an amount of

data equal to one “full-sized” segment—that is, equal to the MSS (maximum segment size) value for the connection. Each time an acknowledgment is received, the amount of data the device can send is increased by the size of another full-sized segment. Thus, the device “starts slow” in terms of how much data it can send, with the amount it sends increasing until either the full window size is reached or congestion is detected on the link. In the latter case, the congestion avoidance feature is used.

When potential congestion is detected on a TCP link, a device responds by throttling back the rate at which it sends segments. A special algorithm is used that allows the device to drop the rate at which segments are sent quickly when congestion occurs. The device then uses the *Slow Start* algorithm just above to gradually increase the transmission rate back up again to try to maximize throughput without congestion occurring again.

7.5.12 Flow Control Versus Congestion control

Let’s have a look at the difference between *Flow Control* and *Congestion Control*, which are mixed up at times.

Flow control is a very important part of regulating the transmission of data between devices, but it is limited in a way that it only considers what is going on within each of the devices on the connection, and not what is happening in devices between them. It relates to the point-point traffic between a given sender and a receiver. Flow control always involves some kind of feedback from receiver to sender to tell sender how things are at other end of the network. Since we are dealing with how TCP works between a typical server and client at layer four, we don't worry about how data gets between them; that's the job of the Internet Protocol at layer three.

In practice, what is going on at layer three can be quite important. Considered from an abstract point of view, our server and client may be connected “directly” using TCP, but all the packets we transmit are carried across an internet and routers between different networks. These networks and routers are also carrying data from many other connections and higher-layer protocols. If the internet becomes very busy, the speed at which segments are carried between the endpoints of our connection will be reduced, and they could even be dropped. This is called *congestion control*. Congestion control has to do with making sure that subnet carry the offered traffic. It is the global issue, involving the behavior of all the hosts, router, link, store and forward mechanism between them in the entire subnet or internet.

Review Questions

What is congestion? Why congestion occurs?

Ans : In a packet switching network, packets are introduced in the nodes (i.e. *offered load*), and the nodes in-turn forward the packets (i.e. *throughput*) into the network. When the "offered load" crosses certain limit, then there is a sharp fall in the throughput. This phenomenon is known as **congestion**.
In every node of a packet switching network, queues (or buffers) are maintained to receive and transmit packets (store/forward network). Due to busty nature of the network traffic there may be situations where there is overflow of the queues. As a result there will be re-transmission of several packets, which further increases the network traffic. This finally leads to **congestion**.

1. **What are the two basic mechanisms of congestion control?**

Ans : The two basic mechanisms of congestion control are:

- One is preventive, where precautions are taken so that congestion can not occur.
- Another is recovery from congestion, when congestion has already taken place

2. **How congestion control is performed by leaky bucket algorithm?**

Ans : In **leaky bucket algorithm,** a buffering mechanism is introduced between the host computer and the network in order to regulate the flow of traffic. Busty traffic are generated by the host computer and introduced in the network by leaky bucket mechanism in the following manner

- Packets are introduced in the network in one per tick
- In case of buffer overflow packets are discarded

3. **In what way token bucket algorithm is superior to leaky bucket algorithm?**

Ans : The leaky bucket algorithm controls the rate at which the packets are introduced in the network, but it is very conservative in nature. Some flexibility is introduced in token bucket algorithm. In token bucket algorithm tokens are generated at each tick (up to certain limit). For an incoming packet to be transmitted, it must capture a token and the transmission takes place at the same rate. Hence some of the busty packets are transmitted at the same rate if tokens are available and thus introduces some amount of flexibility in the system. This also improves the performance.

4. **What is choke packet? How is it used for congestion control?**

Ans : Choke packet scheme is a close loop mechanism where each link is monitored to examine how much utilization is taking place. If the utilization goes beyond a certain threshold limit, the link goes to a warning and a special packet, called **choke packet** is sent to the source. On receiving the choke packet, the source reduced the traffic in order to avoid congestion.

The congestion control in the choke packet scheme can be monitored in the following manner.

- Each link is monitored to estimate the level of utilization.
- If the utilization crosses a certain threshold limit, the link goes to a warning state and a choke packet is send to the source.
- On receiving the choke packet, the source reduces the transmitting limit to a certain level (say, by 50%).
- If still warning state persists, more choke packets are sent further reducing the traffic. This continues until the link recovers from the warning state.
- If no further choke packet is received by the source within a time interval, the traffic is increased gradually so that the system doesn't go to congestion state again.