Transport Layer: Congestion Control and Leaky Bucket

1 Congestion in Transport Layer

Transport layer provides application multiplexing and demultiplexing, error detection, reliable data transfer, and congestion control services to the application layer. Application multiplexing and demultiplexing allow multiple applications to share and utilize both network and system resources. Error detection and reliable data transfer ensure correct and reliable data transformation respectively for application layer over unreliable network links. Unlike these services, congestion control mechanism balances the network traffic as well as improves end-to-end Quality of Services (QoS). Congestion causes packets to be dropped on the network due to buffer overflow, and therefore leads to data loss and unreliable connection. Therefore effective congestion control is an important issue in transport layer.

There are several causes of network congestion. When buffers become overflow, routers will drop any newly arriving packets without informing the original sender. When multiple hosts send data packets at different rates, the host sending packets at a significantly high rate, comparing with rates of other hosts, will eventually win the contention and fully occupy the buffer. The router becomes inaccessible to other hosts, and drops their packets. As a consequence, network performance will degrade dramatically, and application performance will be affected. Therefore, it is crucial to have a proper congestion control algorithm to avoid and react to congestion accordingly. Since TCP provides reliable and controlled network service at transport layer, it is desirable and appropriate to implement congestion control within TCP.

TCP congestion control can improve the performance of a single application, provide effective interaction and fairness among applications, and resolve network congestion collapse. This note discusses the different causes of network congestion and their effects to network performance. Section 2 describes and analyzes three congestion cases in detail, in order to provide insights on the causes of network congestion. Section 3 provides an overview of two major approaches to congestion control, open-loop control and close-loop control. This note is concluded with a detailed study of leaky bucket, an open-loop control mechanism for transmission rate policing.

2 Congestion Problems and Examples

Major factors relating to the cause of network congestion include number of senders, arrival rate of new packets, buffer size, and network structure. This section studies three representative network congestion cases to provide insights on how these factors affect network congestion.

2.1 One Sender, a Router with Finite Buffer

In this scenarios, there is only one sender and one router with a finite buffer. Let the arrival rate be λ, buffer size be B, and service rate be μ. As shown in Figure 1, the system is exactly the same as an M/M/1/B queue.

From previous lectures, the probability of having n packets in an M/M/1/B queue, p_n, can be easily calculated as $\frac{1-\rho}{1-\rho^{B+1}}\rho^n$, for $n = 0, ..., B$, where $\rho = \lambda/\mu$.

Throughput and average delay are:

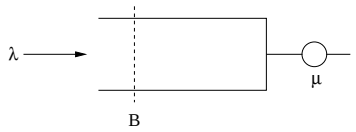

Figure 1: One Sender, a Router with Finite Buffer

Throughput is equal to $(1 - p_0)\mu = \mu\rho\frac{(1-\rho^B)}{(1-\rho^{B+1})}$

Average delay is equal to $\frac{1}{\mu} + \sum_{n=1}^{B-1} n\frac{1}{\mu}p_n$

The system throughput, as in Figure 2, increases as λ approches μ, and reaches and remains at the maximum value, μ, when λ is equal or greater than μ. Once λ is greater than μ, any additional packets will be dropped due to buffer overflow. This observation shows that system throughput cannot exceed the service rate, μ, regardless of λ, since the system can only serve the packets at rate μ at maximum.

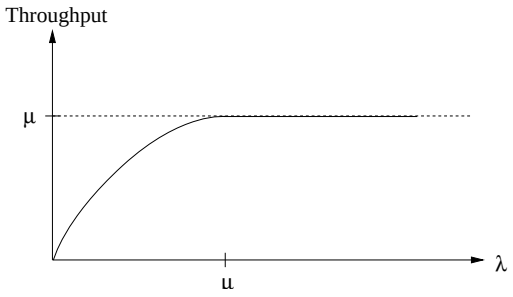

Figure 2: Throughtput

When the queue is empty, new packets are served in $1/\mu$ time. As arrival rate, λ, increases, the size of queue is growing accordingly. The system delay, as in Figure 3, increases exponentially as λ approaches μ, which leads to bad performance. The packets being dropped will have infinite delay.

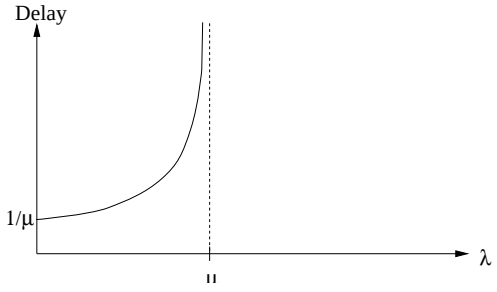

Figure 3: Delay

These two figures illustrate the tradeoff between system utilization and performance (delay). When the system reaches the maximum throughput, packets will suffer from large delay. Network congestion occurs when λ is greater than μ. The application may experience longer network delay

while the system is being fully utilized. Since there is no clear cut on what is the best throughput and delay combination, congestion control problem is difficult to address.

2.2 Two Senders, a Router with Finite Buffer

As shown in Figure 4, there are two streams and one router with finite buffer. Let the arrival rate from host 1 (stream 1), λ_1, be a fixed value 0.8, while arrival rate from stream 2, λ_2 varies. Router buffer size is B, and service rate $\mu = 1$.

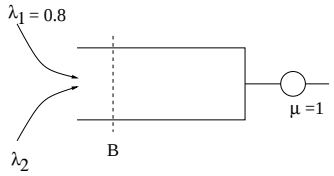

Figure 4: Two Senders, a Router with Finite Buffer

This system is again an M/M/1/B queue with total arrival rate $\lambda = \lambda_1 + \lambda_2$. From previous lectures, the probability of packet from each server is

Probability that the arriving packet is from stream 1 is $P_1 = \frac{\lambda_1}{\lambda} = \frac{\lambda_1}{\lambda_1+\lambda_2}$

Probability that the arriving packet is from stream 2 is $P_2 = \frac{\lambda_2}{\lambda} = \frac{\lambda_2}{\lambda_1+\lambda_2}$

Figure 5 shows the total throughput with respect to arrival rate from host 2. Since the arrival rate from host 1 is fixed at 0.8, the total throughput is 0.8 when λ_2 is zero and is increasing as λ_2 approaches 0.2. Then the system reaches its maximum throughput rate μ; it can not server more packets regardless of the arrival rate from stream 2. When λ is greater than μ, any additional packets will be dropped due to buffer overflow.

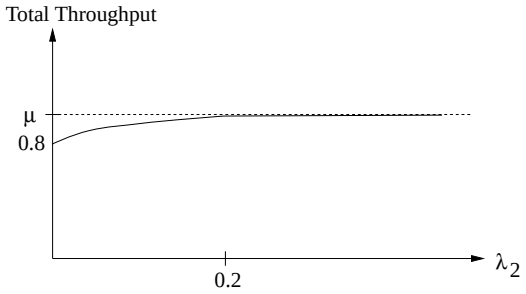

Figure 5: Total Throughput

Throughput for stream 1 as shown in Figure 6 remains constant as long as λ_2 is less than 0.2, *i.e.*, the total arrival rate is less than μ. Once λ_2 is greater than 0.2, the throughput for stream 1 gradually decreases as λ_2 increases. As λ_2 increases, the number of packets from stream 1 being served is decreasing. When λ_2 is significantly larger than λ_1, the system serves packets from stream 2 only. Therefore the throughput of stream 1 is approximately equal to zero.

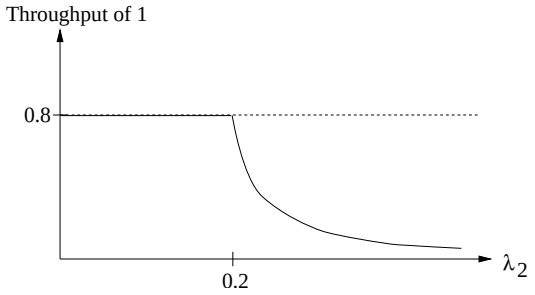

Figure 6: Throughput for Stream 1

Figure 7 agrees with Figure 6. Throughput for stream 2 approaches 1 as λ_2 approaches infinity. The higher λ_2 is, the more packets from stream 2 is being served. When λ_2 is large enough, the router serves stream 2 only. In other words, as arrival rate from host 2 increases, throughput for stream 1 will eventually approach zero and stream 2 will take over the total capacity of the system.

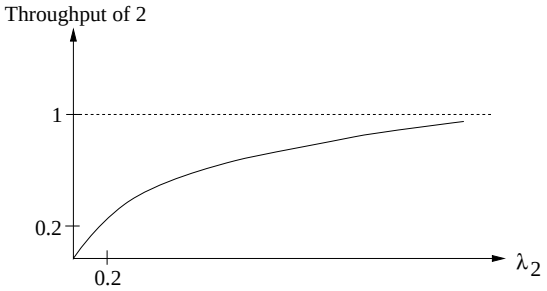

Figure 7: Throughput for Stream 2

This result could be further verified by the probability formulas defined above. As λ_2 approaches infinity and λ_1 remains fixed, total arrival rate λ will approach infinity as well. Therefore, the probability that the arriving packet is from host 1 will approach 0, and the probability the the arriving packet is from host 2 will approach 1. This result further demonstrates the lack of cooperation between two streams competing for shared network resources. Streams without proper rate control can congest the network easily.

Probability that the arriving packet is from host 1 is $P_1 = \frac{\lambda_1}{\lambda_1+\lambda_2} \rightarrow_{\lambda_2 \rightarrow \infty} 0$

Probability that the arriving packet is from host 2 is $P_2 = \frac{\lambda_2}{\lambda_1+\lambda_2} \rightarrow_{\lambda_2 \rightarrow \infty} 1$

Intuitively, since there is no control/agreement between the two incoming streams, stream 2 takes advantage of the fact that no congestion control is imposed and will increase λ_2, and eventually blow out the system. Then the router becomes inaccessible for stream 1. Applications running on host 1 experiences infinite delay, and applications on host 2 fully utilizes the network capacity. The total throughput of the system shows no difference since the system is being fully utilized. Finding the optimal trade-off point between throughput and delay for both streams is an issue need to be addressed. The other issue to be addressed is cooperation among streams competing for the same network resource. Multiple streams may not be aware of each other's transmission

rate before or during sessions. There also may not be a central control mechanism to manage all streams. Therefore, cooperation among multiple streams becomes a difficult problem due to the lack of global information.

2.3 Four Senders, Routers with Finite Buffers, and Multihop Paths

Now let's consider a more complicated scenario as in Figure 8, where data is exchanged in multihop paths between four senders via four routers with finite buffers. There are four data streams, $A \rightarrow C$, $D \rightarrow B$, $B \rightarrow D$, and $C \rightarrow A$; each stream is generated at same rate λ.

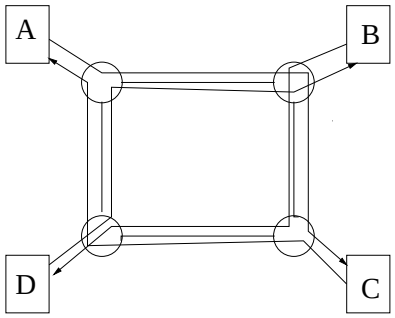

Figure 8: Four Senders, Routers with Finite Buffers, and Multihop Paths

Figure 9 illustrates the data flow in each router and host. It shows that packet rate leaving any router is μ at maximum since the service rate of each router is μ. When this data stream arrives at the next node, it is split into two streams, one is directed to the end host, and the other enters the buffer of the successive router. Each router also accepts new data packets generated by its local host. The total arrival rate at each router must be equal to or less than μ in order to keep the system stable.

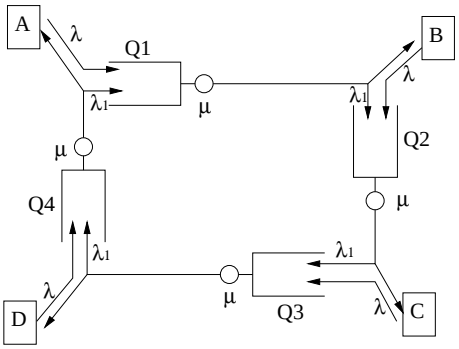

Figure 9: Buffer Flow Diagram

Each queue is exactly the same as the M/M/1/B queue described in previous section. There are two streams competing at each queue. One stream arrives at the queue with an upper bounded rate, λ_1, where $\lambda_1 \leq \mu$, and the other stream arrives at the queue with an unbounded rate, λ. The total arrival rate at each queue is $\lambda + \lambda_1$.

Probability that the arriving packet is from the host is $P_s = \frac{\lambda}{\lambda_1 + \lambda} \rightarrow_{\lambda \rightarrow \infty} 1$

Probability that the arriving packet is from previous router is $P_r = \frac{\lambda_1}{\lambda_1+\lambda} \rightarrow_{\lambda\rightarrow\infty} 0$

The above formula shows that as the arrival from each host increases, the router will serve no by-pass packets but only packets generated by its local host. The following throughput diagrams further show that all packets will be dropped at the second hop. The system eventually can not deliver any packet to the destination.

Figure 10 is the throughput diagram for flow $A \rightarrow C$. Throughput is increased from zero to $\mu/2$ as λ approaches $\mu/2$, and it decreases due to packet loss at each router. For sufficiently large λ, packets from A wins the contention and enter Q1. However, since the arrival rate from host A at router 2 is bounded by the service rate of router 1, μ, B will win in queue Q2 at router 2. As λ approaches infinity, there will eventually be no packets delivered to the destination, since each flow is dropped at the second hop due to contention with new packets generated by the hosts. Since the data flow in this scenario is symmetric on each router and host, the throughput diagram is identical for all four flows.

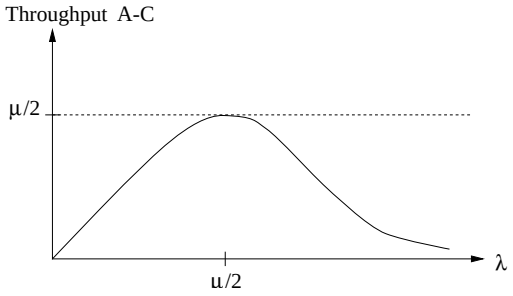

Figure 10: Throughput A-C

The total throughput of this system can be easily obtained by adding the the throughput of the four individual flows, as shown in Figure 11. Overall, as λ increases, eventually the system will serve no packets, since A, B, C, D win the contention at their corresponding queue and the by-pass packets will be dropped. The rapid decreasing of total system throughput as λ increases will eventually blow out network traffic since the network is not able to carry out any success data transmission. This phenomenon is known as *congestion collapse*.

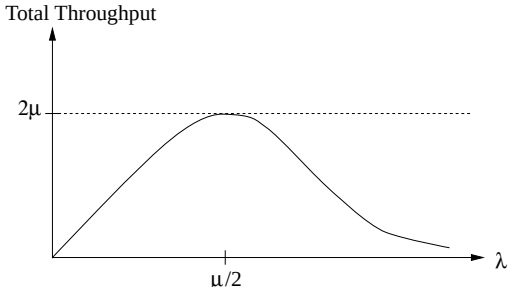

Figure 11: Total Throughput

One analogy of this situation is traffic in the city. Routers can be seen as intersections. Network links are roads between intersections. The maximum traffic each intersection allows is μ cars. New

cars enter each intersection from small streets between two intersection. As the number of new cars gets larger, the intersection will be busy serving new cars instead of by-pass cars, the cars from previous intersection. The by-pass cars stuck between two intersections and is going nowhere. If the same thing happens at each intersection in the city, every car will get stuck at the next intersection it arrives at. Eventually no one can get to their destination. This is referred as congestion collapse since the traffic system collapses due to congestion at each intersection.

When λ is significantly large, the application running on all four hosts may experience infinite packet delay while the network system is fully utilized. The actual throughput of the system is zero since no packets can be delivered to their destination. The cause of congestion collapse is again the lack of control/agreement and cooperation among multiple hosts and routers.

2.4 Summary

The three case studies presented above have provided insights on the causes of network congestion. One is having arrival rate greater than the service rate, which causes the newly arrived packets being dropped. The other is the lack of control and agreement among the flows competing for network resources. As a consequence of congestion, some or all applications might experience unbounded delay, which results in unfair resource allocation among multiple streams. From the system point of view, throughput is maximized during congestion. In resolving network congestion, congestion control is required to balance the throughput-performance tradeoff. In addition, congestion control is required to minimize the total delay as well as to cooperate the competing streams. However, congestion control is not an simple issue to resolve. Since there is no clear cut on best performance and throughput combination, the decision to make is not obvious. Without application-dependent QoS specification and relatively accurate system information, such as capacity and service rate, congestion control cannot properly allocate network resources. Even QoS specification can be formalized using middleware applications, system must be able to detect and react to congestion immediately to keep loss rate as low as possible. The last issue congestion control must resolve is congestion collapse. In order to solve this issue, prediction and proper avoidance must exist to overcome the possible congestion collapse. So far, there is no efficient algorithm providing sufficient prediction and proper avoidance. As a conclusion, congestion control mechanisms are difficult to design and implement due to various uncertainties.

3 Congestion Control

As concluded in the previous section, proper congestion control requires application-dependent specification and system information. Transport layer is the right layer to implement congestion control since it resides between application layer and network layer. There are three ways to deal with congestion, depending on the QoS requirements for each session. By default, routers discard overflow packets without informing the sender. However, this method may incur congestion collapse as described in Section **??**. The other two ways of congestion control are packet blocking and call (session) blocking, which are instances of two general congestion control classes, open-loop control and close-loop control respectively, described in this section.

3.1 Open-Loop Control

During connection setup, the transmission rate of a connection is determined. Once the connection is set up, packets are sent and processed by the system without feedback information from either the

network or the destination. For this reason, this control scheme is referred to as open-loop control. There are two main open-loop control mechanisms, call admission control and leaky bucket, which are discussed below.

- Call Admission Control: During session initialization, both ends of the connection exchange session control information and QoS parameters, such as maximum rate, minimum rate, buffer size and maximum allowed delay. The control information and QoS parameters are together referred as traffic descriptor. The network admits a data stream to access the network only if enough resources are available in the network. Traffic flow can also be enforced by assigning maximum allowed arrival rate in order to avoid buffer overflow at the routers. In other words, each session is assigned with an arrival rate, which is the upper bound of this session. Circuit switch network employed in telephony is an example of call admission control. During call setup, a particular path with certain bandwidth and transmission rate is reserved on the network to ensure the QoS during a session. The drawback of this approach is that network resources can not be fully utilized and may be wasted.

- Policing (Leaky Bucket): The policing method ensures that the transmission rate does not exceed the allowable rate specified in traffic descriptor during a session. If the traffic descriptor is violated, the packets are either dropped or assigned lower priority. Leaky bucket scheme is a widely used policing technique, which can monitor average rate, peak rate and burstiness of the traffic. Number of packets sent into the network depends on the number of tokens assigned to each queue. The use of tokens ensures that the connection does not transmit packets with rate higher than certain rate, r packets per second, that is specified in the traffic descriptor. Next section will discuss leaky bucket scheme in detail.

Open-loop control provides source host no feedback from either the network or the destination host regarding packet loss or network congestion. From the end hosts point of view, a session may not always be able to successfully initialize a session due to the burstiness of the network traffic. However, if a session is successfully initialized, the service quality is guaranteed.

3.2 Closed-Loop Control / Feed Back Control:

Connections are informed dynamically about the congestion state of the network, and are asked to adapted their rates accordingly. Source host, network, and destination host form a closed loop since network or destination host always provide feedback to source host so that the source host can adjust its transmission rate dynamically to fit current network condition. The well-known and widely used close-loop control mechanism is TCP congestion control, which is implemented at transport layer.

- TCP Congestion Control: TCP provides a reliable end-to-end transport services. TCP congestion control adjusts the transmission rate dynamically by controlling the window size, which specifies the number of unacknowledged packets allowed by the source process. The source may use a time-out as implicit feedback from the network to determine network congestion state. The threshold value provides congestion avoidance on top of congestion detection provided by the time-out mechanism. This algorithm controls the number of packets entering the network, and allows multiple TCP connections to share the bandwidth of a link. The two phases of TCP congestion control are slow start phase and congestion avoidance phase, which will be discussed in the coming lecture.

In contrast to open-loop control, close-loop control does provide feedback from the network or destination to the source regarding the packet loss or network congestion. Each session does not reserve resources and may adjust QoS parameters dynamically according to the feedback information. From the end hosts point of view, a data stream can always be admitted into the network without performance and transmission rate guarantees.

4 Leaky Bucket

As mentioned in the previous section, leaky bucket is a widely used policing technique in open-loop control. The leaky bucket scheme monitors and controls average rate, peak rate and burstiness of traffic by using tokens to release packets into network. The advantage of using leaky bucket is that it ensures the transmission rate does not exceed the allowable rate specified in traffic descriptor, which in turn is specified during call setup.

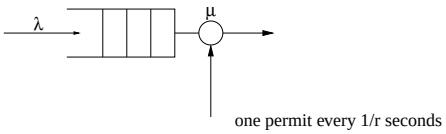

Figure 12: First Approach

In any network, packets can arrive faster than the rate μ for a long period, and cause buffer overflow and packet loss. In leaky bucket scheme, λ is employed to enforce the long-term transmission rate while L specifies the length of a packet. As show in Figure 12, the buffer is allowed to release one packet to the network every $1/r$ seconds in order to generate the desired input rate r of the session.

As shown in Figure 13, all packets from the buffer are sending at the rate of Lr into the network. However, the drawback of this approach is that bandwidth is wasted when there is no packets in the buffer to be sent. Therefore, the average transmission rate is less than Lr. In other words, the token (permission) is wasted, and the network is left idled.

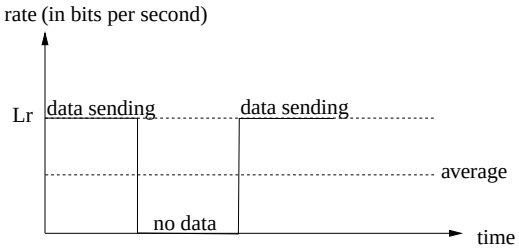

Figure 13: Transmission Rate

The leaky bucket scheme, Figure 14, improves bandwidth usage by saving unused tokens in a bucket, which is controlled by a threshold value. At a session's source, a buffer contains packets without permit and another bucket contains permits. The packet at the head of queue can be released into the network once a permit is available in the bucket. In order to keep the input rate below r for each session, a permit is generated every $1/r$ seconds. If no packets in the queue are ready to be transmitted, permits are saved in the bucket as long as the number of permits in

the bucket does not exceed a certain threshold β. An alternative approach is initially assigning β permits in the bucket and restore the permits to β as necessary every β/r seconds.

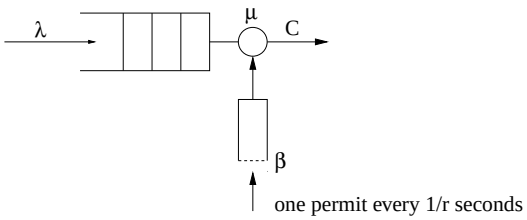

Figure 14: Leaky Bucket Scheme

Figure 15 shows the improved transmission rate in bits per second. When the bucket is empty, the transmission rate of the system the is same as the first approach. When there is no data to be sent, the network is still left idled as before. However, the tokens are saved in the bucket instead of being discarded. The saved tokens allow later transmission to achieve peak rate of C bits per seconds, where C is the link capacity. The range of peak rate depends on the number of tokens saved in the bucket, and is at most $\frac{\beta L}{C}$ seconds. The average transmission rate achieved by leaky bucket scheme is Lr bits per second.

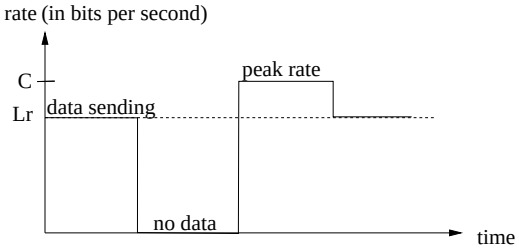

Figure 15: Transmission Rate

The choice of bucket parameter β, bucket size, is crucial in controlling buffer overflow and reducing maximum packet delay. With small β, packets will have longer delayed due to long wait for available permits. With large β, packets will be released into the network at a higher rate, which might cause buffer overflow at a congested downstream node. Dynamic adjustment of bucket size, β, allows the source node to release packets into the network according to the burstiness of the network. Source node may decrease the value of β upon receiving a control message from a congested downstream node in order to avoid buffer overflow. Dynamic adjustment of both bucket parameters, bucket size and permit rate, can achieve better buffer overflow avoidance and smaller packet delay. Nonetheless, the control message may not reach the corresponding source node in time due to large propagation delay. Some predictive and probabilistic mechanism may be employed to issue control messages before actual congestion take place in the network.

Leaky bucket scheme can be used to provide QoS in terms of delay and performance. However, as discussed above, the bucket parameters are very tricky to tune, and the large propagation delay limits the correctness of the control scheme. Due to its complexity, leaky bucket scheme is conservative and unscalable, which are its major drawbacks.