IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 3, June-July, 2013
ISSN: 2320 - 8791
www.ijreat.org

RATE CONTROL SCHEME IN DATA NETWORK

Joy Eneh[1], Harris Orah[2]

[1]Department of Electrical and Electronic Engineering, Nnamdi Azikiwe University
Awka, Anambra state, **Nigeria.**

[2]Department of Electronic Engineering, Projects Development Institute (PRODA),
Emene, Enugu state, **Nigeria.**

Abstract

Network resources are shared as needed by a community of users. Without effective traffic controls, networks are vulnerable to possible congestion when the offered traffic exceeds the network capacity, leading to serious deterioration of network performance. This paper looks at two access control schemes for network traffic, which are based on limiting the rates at which traffic are admitted into the network, and also gives an overview of the underlying principles for these schemes.

1. Introduction

Computer networks are designed to handle a certain amount of traffic with an acceptable level of network performance. Network congestion will increase as network speed increases and the performance of such networks will deteriorate if the offered traffic exceeds the given network capacity. Packets will suffer long queuing delays at congested nodes and possibly packet loss if buffers overflow.

New control methods are needed, especially for handling "bursty" traffic expected in very high speed networks found in present day applications.

Traffic control methods seek to achieve a balance between isolation of network resources to reduce congestion and statistical sharing of the resources to accommodate the varying requirements of the sessions in the network while still keeping delay low and maintaining a reasonable throughput.

Under traffic management,. The primary means of protecting a network from congestion are admission control and access regulation (or policing) which limits the rate of traffic entering the network. By restricting the total amount of carried traffic, congestion will be avoided, and acceptable network performance will thus be sustained at the expense of blocking some amount of ingress traffic.

Access control or policing refers to regulation of ingress traffic at the user-network interface. There are reasons for regulating traffic at the network edge and not internally within the network. At the network boundary, excessive traffic can be blocked before consuming any network resources. Moreover, it is natural to verify conformance of the traffic as close to the source as possible before the traffic shape is effected by other packet flows. Finally, individual packet flows at the network edge are slower than multiplexed flows in the core network

1.1 Network congestion problems

Any network has bottlenecks or congestion points, i.e. locations where more data may arrive than the network can carry. A common cause for congestion is a mismatch in speed between networks. For example, a typical high-speed performance local area network (LAN) environment in the next several years may have the Ethernet architecture as shown in fig. 1, while the server will use new high speed ATM

connections at a rate of 155 Mbps. Many clients will still depend on the old, inexpensive but slower 10-Mbps Ethernet connections. Data flowing from the server at 155Mbps to the clients at 10-Mbps will experience congestion at the interface between the ATM and Ethernet networks.

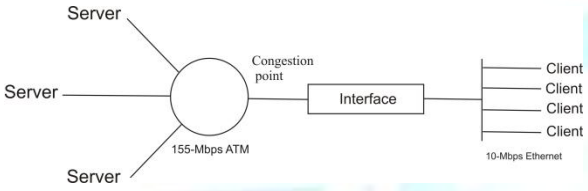

Fig. 1 congestion due to a mismatch in speed between 155-Mbps ATM network and 10-Mbps Ethernet

Congestion also occurs inside a network node that has multiple ports. Such a node can be a switch such as an ATM switch or a gateway such as a router. As shown in fig 2, congestion arises when data, destined for a single output port, arrive at many different input ports. The faster and more numerous these input ports are, the severer the congestion will be.

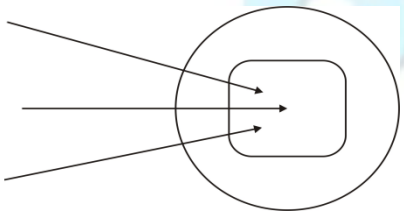

Fig 2 A network node showing congestion in a switch our gateway due to multiple arrival at the same output port

1.2 Consequences of network congestion

A consequence of congestion is the loss of data due to buffer overflow. For data communications in which every bit must be transmitted correctly, lost data will have to be retransmitted, and will result in degraded network utilization and increased communications delay for end users, and also cause network instability.

1.3 Flow control objectives

Consider a case of file transfer:

I. Sender sends a stream of packets representing fragments of a file
II. Sender should try to match rate at which the receiver network can process data
III. Can't send too slow or too fast
IV. Too slow wastes time
V. Too fast can lead to buffer overflow
VI. How to find the correct rate?

The last question highlights the need for flow control in any network. Generally, a need for flow control arises whenever there is a constraint on the communication rate between two points due to limited capacity of the communication lines or the processing hardware. Thus, a flow control scheme may be required between two users at the transport layer, between a user and an entry point of the subnet (network layer), between two nodes of the subnet (network layer), or between two gateways of an interconnected network (internet layer). Flow control prevents network instability by keeping packets waiting outside the network rather than in queues inside the network and avoids wasting of network resources.

Flow control for data entering into a network is done to achieve the following objectives:

- To make sure that data are not sent faster than they can be processed
- To optimize channel utilization
- To avoid data clogging transmission links to cause congestion of the link

- Maximize network throughput
- Reduce network delays
- Maintain quality-of-service parameters such as fairness, delay and throughput.

Any algorithm designed to achieve the above flow control objectives must first, *strike a good compromise between throttling sessions (subject to minimum data rate requirements) and keeping average delay and buffer overflow at a reasonable level.* Second, *maintain fairness between sessions in providing the requisite quality of service.*

1.4 Means of flow control

1. *Call blocking.* Here a session is simply blocked from entering the network (its access request is denied).
2. *Packet discarding.* When a node with no available buffer space receives a packet, it has no alternative but to discard the packet.
3. *Packet blocking.* When a packet is discarded at some node, the network resources that were used to get the packet to that node are wasted. It is thus preferable to restrict a session's packets from entering the network if after entering they are to be discarded. If the packets carry essential information, they must wait in a queue outside the network; otherwise, they are discarded at the source.
4. *Packet scheduling*. In addition to discarding packets, a sub-network node can exercise flow control by selectively expediting or delaying the transmission of the packets of various sessions.

In our analysis of the rate control schemes, the Asynchronous Transmission Mode (ATM) shown in fig. 3 is our network of interest because such networks support very high speed connections and multimedia services. An ATM network can simultaneously support multiple types of services for voice and other fixed-rate guaranteed traffic; Variable bit rate (VBR) services for video, and available bit rate (ABR) services for data. Under ABR services, users can have instant access to available network bandwidth when they need it. These services are exactly what many computer users desire.

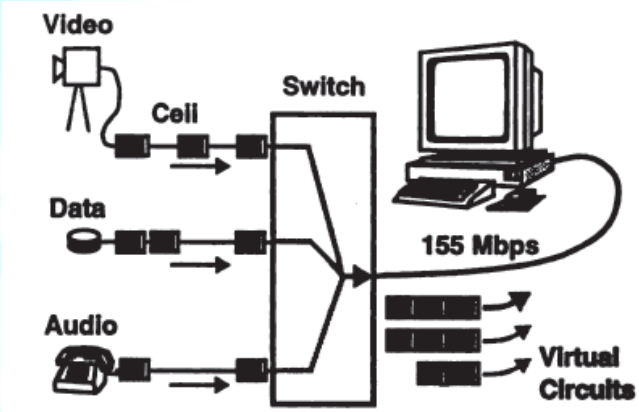

Fig. 3 a block diagram of an ATM Network

Flow control mechanisms designed to support ATM ABR services should meet a variety of technical goals, including the following:

1. Data should rarely, if ever, be discarded due to exhaustion of node buffer memory. Such data may have to be retransmitted after a possibly lengthy time-out period, further contributing to network congestion and the delay experienced by the user
2. Network links should be used at full capacity whenever possible. For instance, if one connection sharing a link reduces the rate at which it sends, the other should increase their rates as soon as possible. The flow control mechanism should allow ABR traffic to fill in, instantly, unused bandwidth

left on the link after guaranteed traffic is served

3. All the connections that are constrained by a bottleneck link should get fair shares of that link
4. The flow control mechanism should be robust. Loss or delay of control messages, and admission of additional connections while maintaining the total traffic load, for instance , should not cause increased congestion
5. Network administrator should not have to adjust any complex parameters to achieve high performance
6. The flow control mechanism should have a cost commensurate with the benefits it provides.

2. The Rate control schemes

The rate control scheme is an access control method. It merely adjusts the rate at which a sender produces data to the rate at which the receiver absorbs data**.**

The main considerations in setting input session rates are:

1. *Delay-throughput trade-off.* Increasing throughput by setting the rates too high runs the risk of buffer overflow and excessive delay.

2. *Fairness.* If session rates must be reduced to accommodate some new sessions, the rate reduction must be done fairly, while obeying the minimum rate requirement of each session.

This access control scheme is an alternative to the window flow control scheme on which the most widely used form of flow control is based. This alternative was developed to correct the shortcomings of the window flow control scheme. In the window flow control, the rate at which a source sends packets depends on both the size of a window and on the rate of feedback from the network to the source. Thus, windows do not perform well on high-speed networks, largely due to the relatively long round trip delay before the source receives feedback. This is relatively large compared with packet transmission times, so that the windowing method does not react quickly enough to prevent congestion and large delay.

With end-to-end windows, the source depends on feedback from the destination, whereas with node-by-node windows, the source receives feedback from the intermediate node to which it directly transmits. The delay in waiting for a response will obviously be longer for end-to-end windows. Thus, the issues of congestion that arise due to delayed feedback for node-by-node windows are an even more serious problem with end-to-end windows.

Additionally, high-speed networks will need to accommodate a wide range of source transmission speeds and windows do not perform well under these conditions.

Another problem with window is that it does not guarantee a minimum data rate. Voice, video, and an increasing variety of data sessions require upper bounds on delay and lower bounds on rate. High-speed wide area networks increasingly carry such traffic, and many lower-speed networks also carry such traffic, making windows inappropriate.

Furthermore, when queuing delays arise, the window mechanisms will often time-out waiting for an ACK. A packet will be interpreted as being lost if an ACK has not been received by a certain time, and will be retransmitted. The value of *W* limits the number of different unacknowledged packets there can be, not the number of 'copies' of these packets. It is likely

that several sources will simultaneously experience the delay in feedback, and resend packets unnecessarily. This will worsen the backlog at the intermediate node and cause further queuing delays. This could result in low throughput and high delay.

The intrinsic problem with window is that the same size window that is used under light load conditions is also being used in the case of heavy congestion. Thus, by choosing the window size to accommodate the sources under light traffic, the sources are not throttled enough when congestion develops. Too many packets are still permitted to be sent by the source, and the queue at the intermediate node is not given a chance to fully dissipate. The problem is exacerbated on high-speed networks where the window size may be very large, and hence the queue length and associated queuing delay may be very large.

Ideally, the window size should be dynamically adjusted according to the level of congestion in the network. This is a difficult control problem due to the relatively large delay before the source receives feedback from the intermediate node. The source would have to adjust its windows based on 'old' information, which may result in unnecessary oscillations in window size. Additionally, changing the window size of one source affects the traffic level of all of the intermediate nodes in its path, which in turn affects the congestion experienced by other sources that send data through these nodes. Therefore, the window sizes of the sources and intermediate nodes should be adjusted collectively; otherwise, the adjustments may counteract each other or result in overcompensation, and the congestion will be shifted rather than lessened. This is difficult to accomplish in practice due to the large scale of the problem.

An alternative form of flow control is to assign a maximum transmission rate and an average transmission rate to each source. The network, with knowledge of the burstiness and desired rates of all nodes that feed into any link, assigns the rates such that the probability of congestion developing is very small. This method essentially provides a guaranteed rate without dependence on feedback from the intermediate nodes, and thus is more appropriate than windows for use on high-speed networks. This is what gave rise to the rate-based flow control.

Rate control schemes offer an alternative form of flow control which is based on giving each session a guaranteed data rate, which is commensurate to its needs. This rate should lie within certain limits that depend on the session type. For example, for a voice session, the rate should lie between the minimum needed for language intelligibility and a maximum beyond which the quality of voice cannot be further improved.

When determining the input rate session, the following must be taken into consideration:

1. Delay-throughput trade-off: increasing the throughput by setting the rates too high runs the risk of buffer overflow and excessive delay
2. Fairness: if session rates must be reduced to accommodate some new sessions, the rate reduction must be done fairly, while obeying the minimum rate requirement of each session.

In rate-based flow control, there are three parameters that are specified to control the data flow, as

compared to the single parameter (i.e., window size) that is used in the window-based scheme.

i. First, each source is assigned an average transmission rate. These rates should be assigned such that the aggregate average arrival rate into any intermediate node is less than the capacity of the outgoing link of the node. By this means the sources are essentially guaranteed of achieving their assigned average data rate.

ii. Each source is also assigned a maximum rate of transmission (which is often simply the capacity of the access link)

iii. Each source is finally assigned a limit on the length of time it can send data at that rate. These parameters control the ability of the source to send a burst of data. In window-based flow control, the burst size is essentially the window size, which is determined by the rate of transmission. In the rate-based method, the maximum burst size can be set independently of the rates, thus allowing more control of the data flow.

However, with proper choice of the bucket parameters, buffer overflow and maximum packet delay can be reduced. In particular, the bucket size W is an important parameter for the performance of the leaky bucket scheme, which will be discussed shortly.

If W is small, bursty traffic is delayed, waiting for permits to become available (W= 1 resembles time division multiplexing). If W is too large, long bursts of packets will be allowed into the network; these bursts may accumulate at a congested node downstream and causes buffer overflow. These facts lead to the idea of dynamic adjustment of the bucket size. In particular, a congested node may send a special control message to the corresponding sources instructing them to shrink their bucket sizes.

A session is a fixed-path connection established between a source and the destination in a virtual circuit network.

2.1 Ways of implementing the rate-based control schemes.

In general, when a call is set up, the source specifies the average traffic rate it expects to have. The network decides whether a call with this average rate can be accepted, or negotiates with the source to determine a more acceptable rate. It then assigns the source a maximum transmission rate and a maximum burst size that is appropriate for the type of data the source is sending. The parameters should be assigned in such a way that there is only a small probability of the instantaneous aggregate arrival rate at any intermediate node being larger than the capacity of its outgoing link. This keeps queuing delay small and minimizes the chance of packets being dropped due to buffer overflow.

If retransmissions are necessary, they are sent using the same rate-based scheme, and thus they are not really 'extra' traffic to the system as they are in the window-based scheme. This should prevent the system from entering a situation in which high queuing delays and retransmissions feed off of each other. Additionally, fine tuning of the rates can be done through feedback from the intermediate node, but as with windows, there will be a delay before such changes take effect. Given an algorithm that generates desired rates for various sessions, the

question of implementing these rates arises. A strict implementation of a session rate of *r* packets/sec would be to admit 1 packet each *1/r* seconds. This, however, amounts to a form of time-division multiplexing and tends to introduce large delays when the offered load of the sessions is bursty.

A more appropriate implementation is to admit as many as *W packets (W>1)* every W/r seconds,

This allows a burst of as many as *W* packets into the network without delay, and is better suited for a dynamically changing load. There are several variations of this scheme. A prominent variation of this scheme is the leaky bucket scheme. It is patterned after the window flow control.

Here an allocation of W packets (a window) is given to each session, and a count *x* of the unused portion of this allocation (i.e. the remainder of the allocation) is kept at the session origin. Packets from the session are admitted into the network as long as x>0. Each time a packet is admitted, the count is decremented by 1, and W/r seconds later (r is the rate assigned to the session), the count is incremented by 1.

2.2 The leaky bucket rate control scheme

A related method that regulates the burstiness of the transmitted traffic somewhat better is the leaky bucket scheme illustrated by the diagram in fig 4.

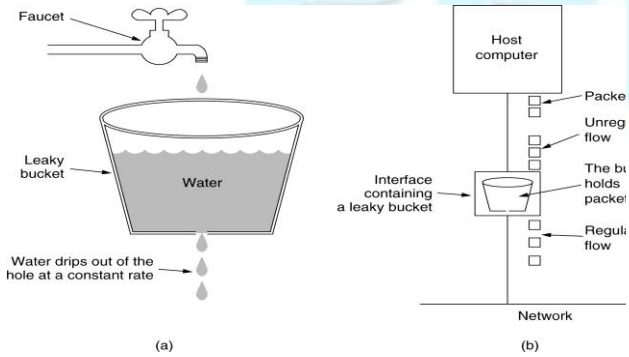

Fig 4: Illustration of the leaky bucket scheme
(a) A leaky bucket with water. (b) a leaky bucket with packets

Here the count is incremented periodically, every 1/r seconds, up to a maximum of W packets. Another way of viewing this scheme is to imagine that for each session, there is a queue of packets without a permit and a bucket of permits at the session's source. The packet at the head of the packet queue obtains a permit once one is available in the permit bucket and then joins the set of packets with permits waiting to be transmitted. The permits are generated at the desired input rate r of the session (one permit each 1/r seconds) as long as the number in the permit bucket does not exceed a certain threshold W.

2.3 The token bucket rate control scheme

In this scheme, a source generates tokens at a fixed rate and can only send a packet when there is a token available. If the source temporarily has no data to send, the tokens are allowed to build up, up to the burst size limit. The source can then send a burst of packets at its maximum rate.

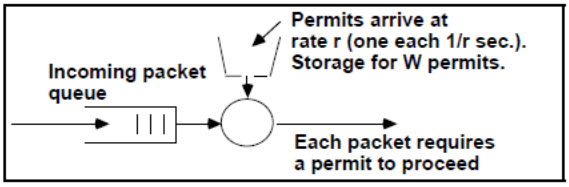

Fig 5 the operation of the token bucket scheme

The token bucket scheme is based on an analogy of a fixed capacity bucket into which tokens, normally representing a unit of bytes or a single packet of predetermined size, are added at a fixed rate. When a packet is to be checked for conformance to the defined limits, the bucket is inspected to see if it contains sufficient tokens at that time. If so, the appropriate number of tokens, e.g. equivalent to the length of the packet in bytes, are removed ("cashed in"), and the packet is passed, e.g., for transmission.

If there are insufficient tokens in the bucket the packet does not conform and the contents of the bucket are not changed.

Non-conformant packets can be treated in various ways:

- They may be dropped.
- They may be queued up for subsequent transmission when sufficient tokens have accumulated in the bucket.
- They may be transmitted, but marked as being non-conformant, possibly to be dropped subsequently if the network is overloaded.

A conforming flow can thus contain traffic with an average rate up to the rate at which tokens are added to the bucket, and have a burstiness determined by the depth of the bucket.

2.4 Uses of the token bucket

The token bucket can be used in either traffic shaping or traffic policing. In traffic policing, nonconforming packets may be discarded (dropped) or may be reduced in priority (for downstream traffic management functions to drop if there is congestion). In traffic shaping, packets are delayed until they conform. Traffic policing and traffic shaping are commonly used to protect the network against excess or excessively bursty traffic. Traffic shaping is commonly used in the network interfaces in hosts to prevent transmissions being discarded by traffic management functions in the network.

Table 1 leaky bucket Vs token bucket

Leaky bucket	Token bucket
The Leaky Bucket Algorithm used to control rate in a network. It is implemented as a single-server queue with constant service time. If the bucket (buffer) overflows then packets are discarded	The Token Bucket (TB) algorithm, allows the output rate to vary, depending on the size of the burst.
The leaky bucket enforces a constant output rate regardless of the burstiness of the input. Does nothing when input is idle	In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must capture and destroy one token.
The host injects one packet per clock tick onto the network. This results in a uniform flow of packets, smoothing out bursts and reducing congestion	Tokens are generated by a clock at the rate of one token every Δt sec.
When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick.	Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.

2.5 Queuing analysis of leaky buckets

This analysis provides an insight into the behavior of the leaky bucket scheme. For the purpose of the analysis, the following assumptions are made.

1. Packets arrive according to a Poisson process with rate of λ. $\mathbf{a_i}$ = Probability(i arrivals) = $\mathbf{(\lambda/r)^i e^{-\lambda/r} / i!}$
2. A permit arrives every 1/r seconds, but if the permit pool contains W permits, the arriving permit is discarded.

The system is then represented as a discrete-time Markov chain with states 0, 1, W. The states represent the "permit deficit" and are equal to the number of permits needed in order to refill the bucket. The states i = 0, 1 …W corresponds to W-i permits available and no packets without permits waiting.

States are determined using equation (1)

State of system: **K = W + P –B** …… (1)

Where:

P = number of packets waiting in the buffer for a permit

B = number of permits in the buffer

W = bucket size

State 0 = bucket full of permits

State W = no permits in buffer

State W + j = j packets waiting for a permit

The state transition occur at times 0, 1/r, 2/r…, just after a permit arrival. The probability of k packet arrivals in 1/r seconds is given by

$\mathbf{P_k = (\lambda/r)^k e^{-\lambda/r} / k!}$

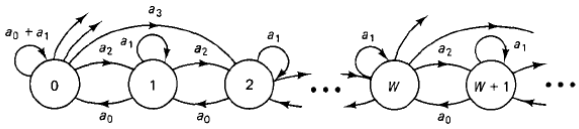

Fig 6 the transition probabilities of a discrete Markov chain model for the leaky bucket scheme. Here a_k is the probability of k packets arrivals in 1/r seconds.

It can be seen that the transition probabilities of the chain are:

$$\mathbf{P_{0i}} = \begin{cases} \mathbf{a_{i+1}, \ if\ i \geq 1} \\ \\ - \\ \\ \mathbf{a_0 + a_1, \ if\ i = 0} \end{cases} \qquad \mathbf{(2)}$$

for j ≥ 1,

$$\mathbf{P_{ji}} = \begin{cases} \mathbf{a_{i-j+1}, \ if\ j \leq i-1} \\ \\ - \\ \\ \mathbf{0, \ \ otherwise} \end{cases} \qquad \mathbf{(3)}$$

The global balance equation for the transition states is given as:

$\mathbf{P_0 = a_0 P_1 + (a_0 + a_1) P_0,}$ - **(4)**

$\mathbf{P_i = \sum_{j=0}^{i+1} a(i-j+1) Pj, \ i \geq 1}$ - **(5)**

Equations (4) and (5) can be solved recursively to obtain the transition probability for each state. We have

$\mathbf{P_1 = a_2 P_0 + a_1 P_1 + a_0 P_2}$ - **(6)**

Now using the equation $\mathbf{P_1 = (1-a_0-a_1)P_0/a_0}$, we obtain

$$p_2 = \frac{p_0}{a_0}\left(\frac{(1-a_0-a_1)(1-a_1)}{a_0} - a_2\right) \quad \text{- (7)}$$

In a similar manner, we can use the global balance equation for P_2, and the computed expression for P_1 and P_2, to express p_3 in terms of P_0, and so on.

The average delay to obtain a permit is given by

$$T = \left[\sum_{j=W+1}^{\infty} (j - W) P(j) \right] \frac{1}{r} \quad - \quad (8)$$

3. Conclusion

The rate control scheme is a better and more widely used means of controlling access to a high-speed network. It has shown a better performance than the window control scheme, while overcoming all the limitations of the window flow control methods. The issue of major interest in the use of the rate-based control schemes is evaluating how well a given set of source rates will perform in preventing congestion in a selected network. Works in this area only evaluates the data flow at one intermediate node, rather than the network as a whole and provide insight into the possible queuing delay which might develop and the probability of a buffer overrun. The output flow is always less bursty than the input flow and the burstiness of the output flow increases with the size of the bucket.

References

[1] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall PTR, 2003. Page 401.

[2] ITU-T, Traffic control and congestion control in B ISDN, Recommendation I.371, International Telecommunication Union, , Annex A, 2004, page 87.

[3]http://www.en.wikipedia.org/w/index.php?title=Token_bucket&oldid=540867580

[4] H. T. Kung, Traffic management for high-speed networks, presented at the National Research Council (U.S.), United States. Office of Naval Research, United States Air Force. Office of Scientific Research.

[5] Dimitri Bertsekas, and Robert Gallager Data networks, Second Edition, Prentice-Hall 1992.

[6] Wang, P-C., Chan, C-T., Hu, S-C., Lee, C-L., and Tseng, W-C. High-speed packet classification for differentiated services in next-generation networks, IEEE Trans. On Multimedia, 6, 925-935. 2004

[7] Baboescu, F., and Varghese, G. ATM Forum (1999). Traffic Management Specification Version 4.1.. Scalable packet classification. IEEE/ACM Trans. On Networking, 13, 2-14. 2005

[8] Elwalid, A., and Mitra, D. Traffic shaping at a network node: theory, optimum design, admission control. In proc. Of IEEE INFOCOM'97, 444-454. 1997

[9] H.T Kung, and Gordon Mckay traffic management for high speed networks Fourth lecture. International Science lecture series, National academy press, Washington D.C, 1997.