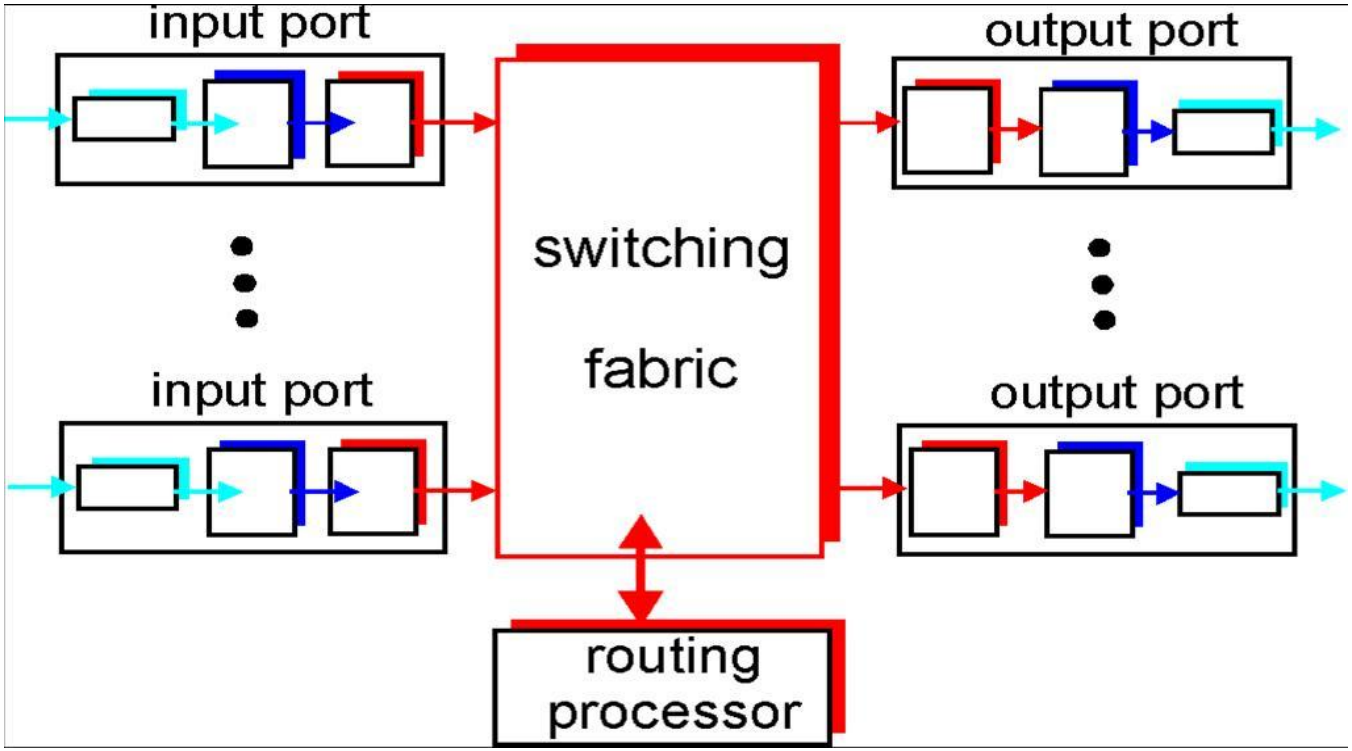
Congestion Control Techniques and Algorithms

CSIT 560

Brahim Bensaou

- Internet Routers fulfill essentially two functions
  - Routing
  - Forwarding



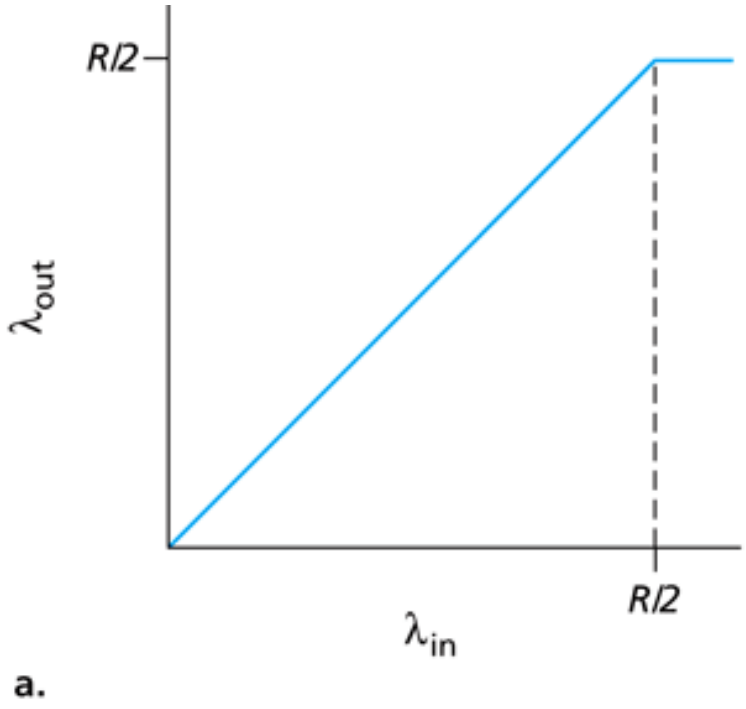
Congestion

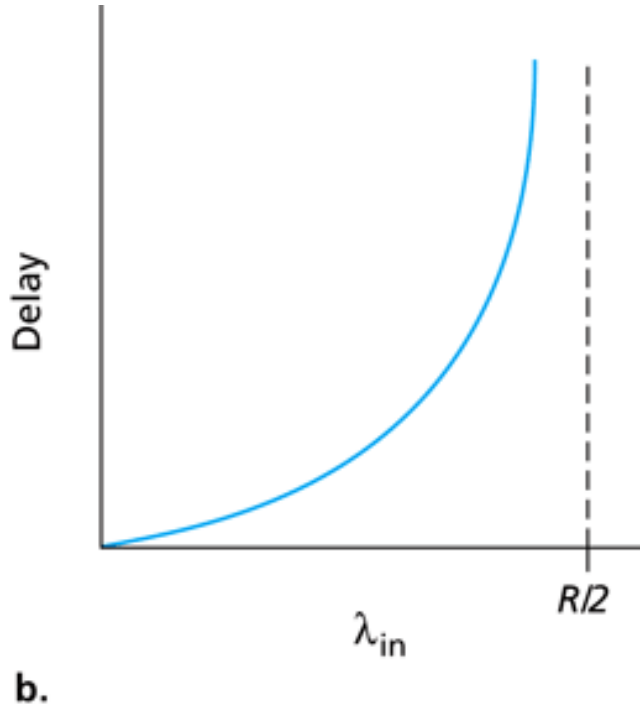
- Input port queueing:
  - HOL blocking induces queueing with long delays and possibly packet losses
- Output port queueing (typically with large speed up factor)
  - The switching fabric may overrun the output port leading to long delays and possibly packet losses
- Both types of switches are subject to Congestion
- Congestion manifest itself via:
  - Long delays: when the queues build up in the routers
  - Packet losses: when the queue size exceeds the physical buffer size in the router

Manifestations of Congestion

- Consider two identical sources sharing a buffer of infinite capacity in a router of switching capacity R bits/s

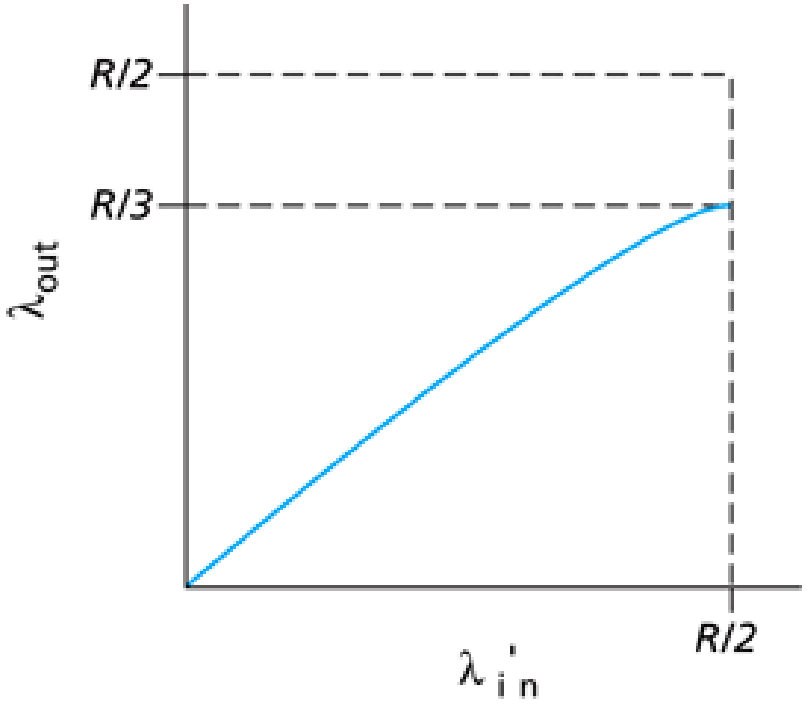
- If there is **no retransmission and no dropped packets:**



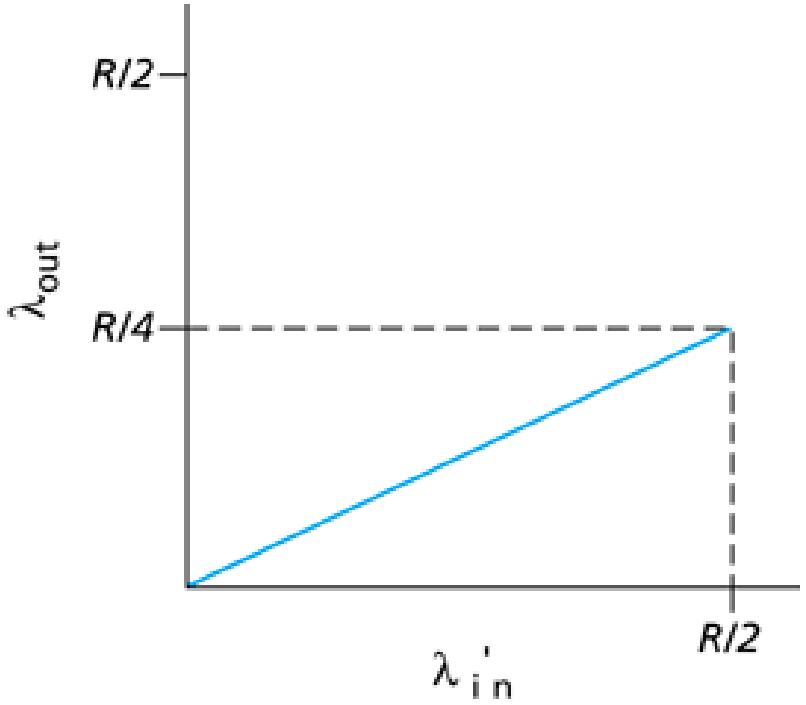


Manifestations of Congestion

- Consider now the two identical sources sharing a buffer of finite capacity in a router of switching capacity R bits/s
- When the buffer is full there are **packet drops and retransmissions**
- The throughput drops further due to the retransmissions (more work is done to achieve the same outcome)



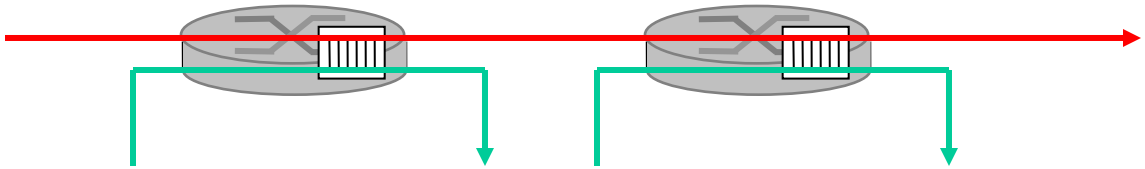
With perfect retransmissions



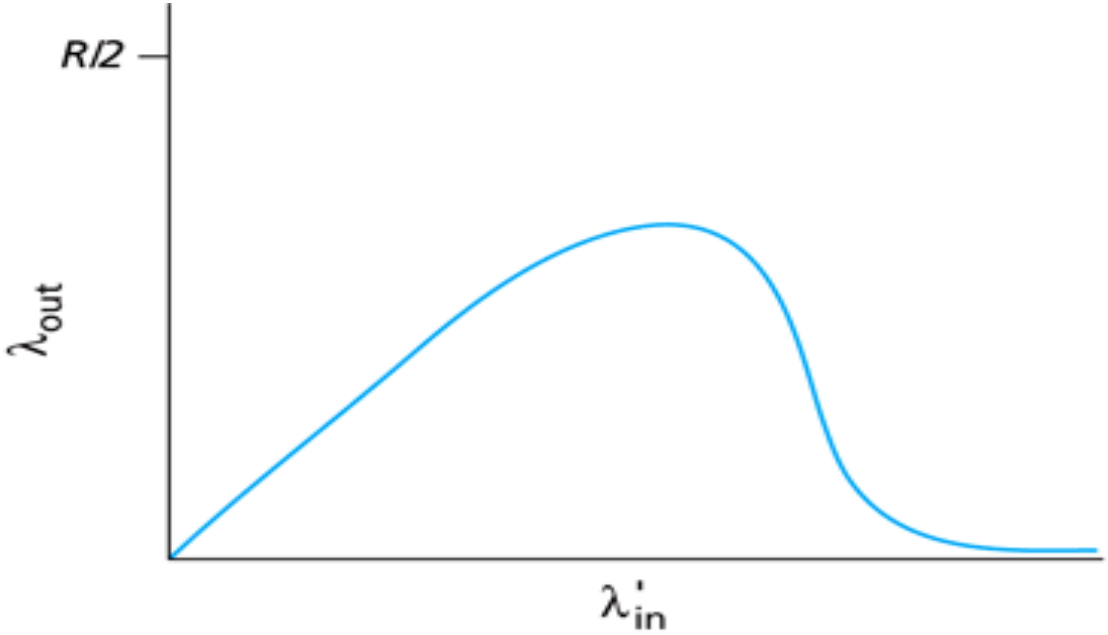
with imperfect retransmissions

Manifestations of Congestion

- Consider now the three identical sources one crossing two routers and sharing each router's buffer with one other source



- As the source rate increases we can observe **congestion collapse** (a lot of work is done but very little is useful)



Solutions to the Problems of Congestion

- There are two broad categories of solutions to the problems of congestion

- Preventive measures: do not allow congestion to take place
  - Dimension the network to sustain the overall long term offered load
  - Perform admission control at the network ingress on a contract basis to limit the instantaneous load
  - Police/shape the traffic of each source to ensure the source does not violate its contract
- Reactive measures: Allow congestion to occur and react to it
  - End-to-end techniques address the problem at the sources by adapting the sending rate in response to congestion
  - Network-assisted congestion control relies on some help from the routers to react to congestion

Prevention vs. Reaction?

- When is it preferable to invoke preventive congestion control versus reactive congestion control?
- Preventive congestion control is more appropriate when we are interested in building a network whose behaviour with respect to application performance is predictable
  - VOIP: Speech is encoded at a given bitrate. The resulting stream is cut into small chunks of data sent individually over the network in UDP datagrams. We want to minimize packet losses in the network, and experience a small delay jitter.
  - Circuit emulation: providing services such as leased lines on a packet switched network. We want to be able to deliver data without delay or jitter
- Reactive congestion control is more appropriate when data is highly bursty and the traffic is elastic (reliability is more important that timeliness)
  - File transfer

1. Preventive Congestion Control

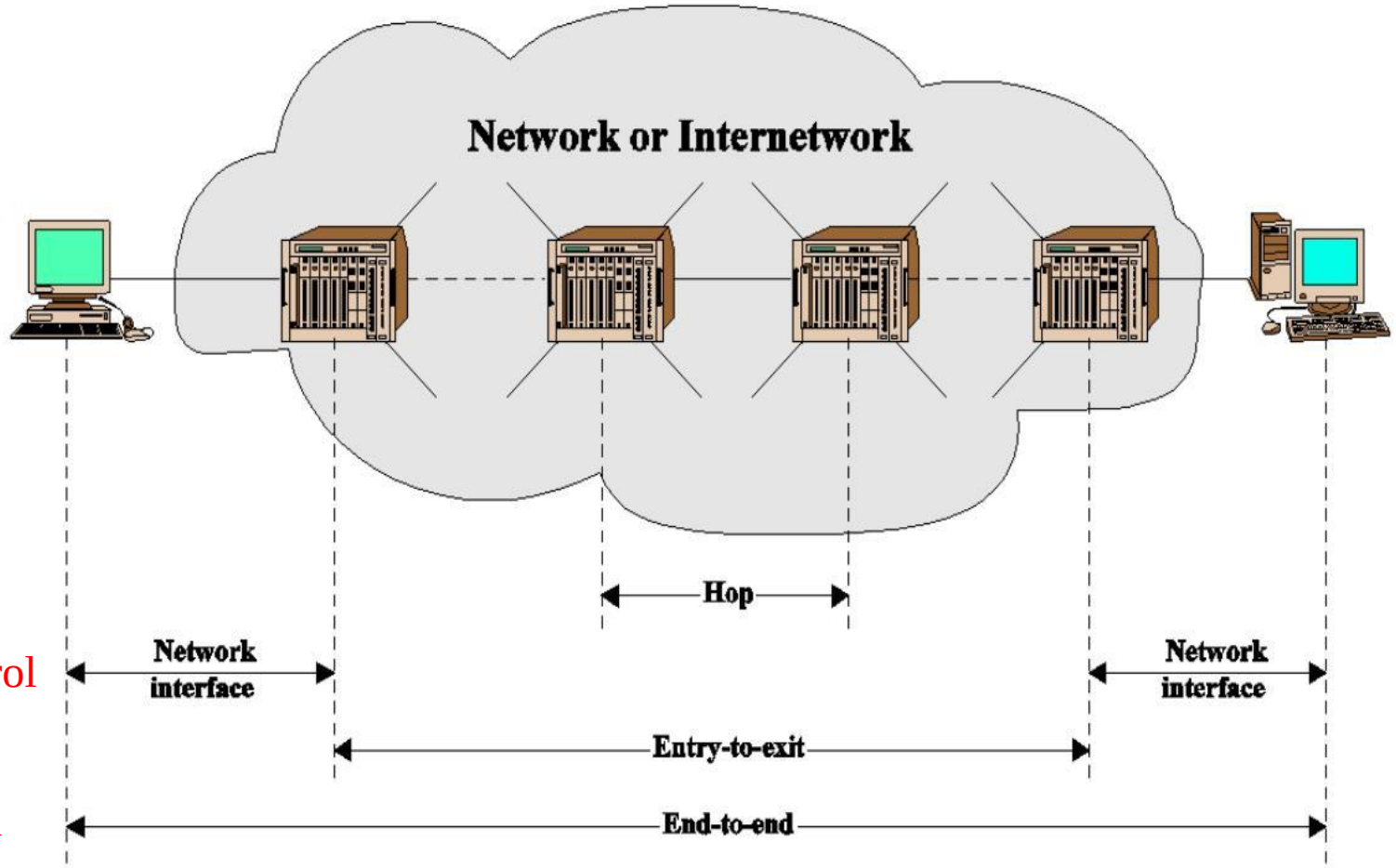
1. Preventive Congestion Control

Traffic Policing

Preventive Congestion Control

- It is originally thought that reactive traffic control is not appropriate for high-bandwidth delay product networks.
- In such networks, the amount of traffic that can be in transit in the network can be huge and results in a high inertia in the realization of traffic rate changes and leads to huge buffer requirements in the network
- This led to the definition of preventive traffic control based on the notion of a **traffic-contract** between the source and the network
- The Traffic-Contract involves:
  - Fixing agreed QoS levels to be maintained by the network
  - Fixing traffic parameters to be respected by the source
- The mechanisms involved in achieving this:
  - Admission control: to limit the total traffic in the network
  - Traffic Policing: to ensure the source abides by the traffic descriptor
  - Resource sharing principles: to share the resources among competing sources.

Preventive Congestion Control



Preventive Control

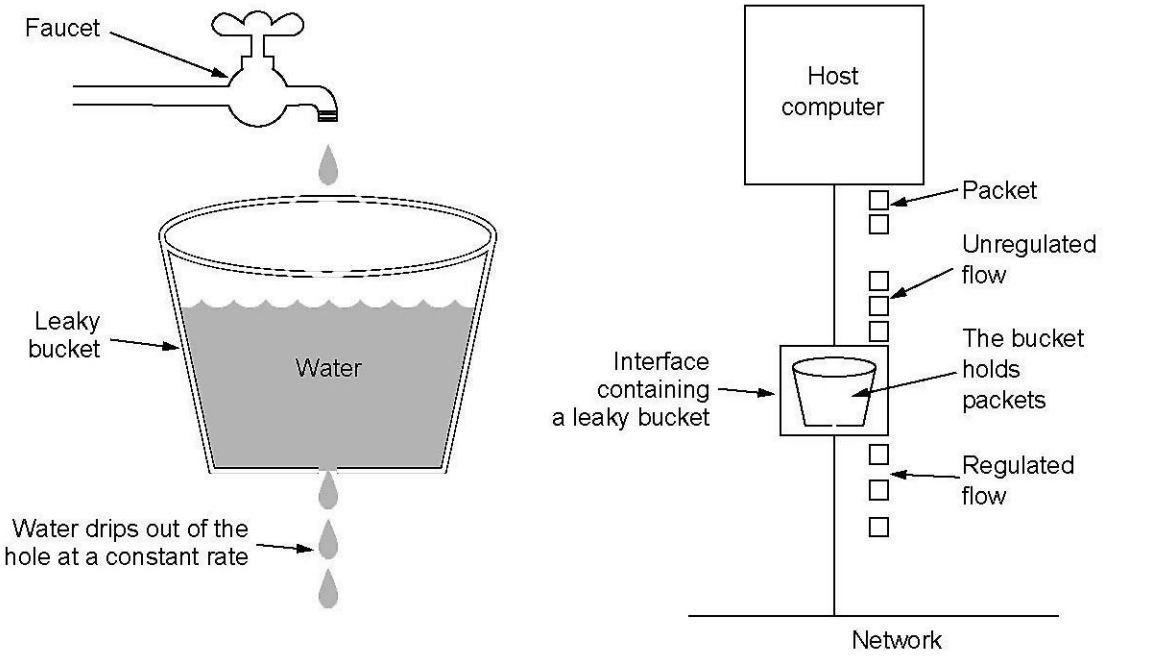
Reactive Control

Traffic Policing

- Traffic policing is the process by which a network ensures that a traffic source abides by its declared traffic parameters. Can take many forms and be deployed by many mechanisms
  - Traffic shaping: modify the traffic profile to make it follow a given profile
  - Peak rate policing: let the traffic enter the network at a rate no larger than a given rate
  - Sustained rate policing: let the traffic enter the network in a bursty manner for a maximum burst then shape the traffic to not exceed a given sustained rate
- Mechanisms to achieve this:
  - Leaky bucket: traffic shaper
  - Token bucket: sustained rate/peak rate policer

Traffic Policing: The Leaky Bucket

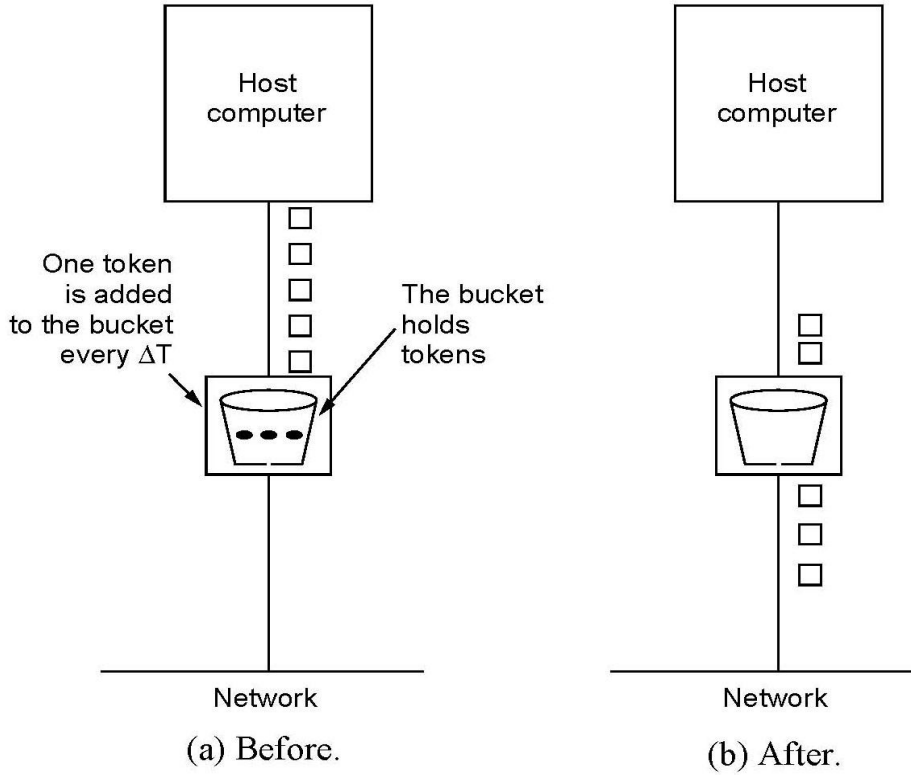
- Leaky bucket: traffic shaper for each source *i*
  - Buffer of size b_i
  - Deterministic server of rate r_i
  - Traffic admitted to the network at a constant rate
- Its goal is to regulate the traffic: it takes an unpredictable traffic pattern from a source and shape it into a well known traffic pattern



(a) A leaky bucket with water. (b) A leaky bucket with packets.

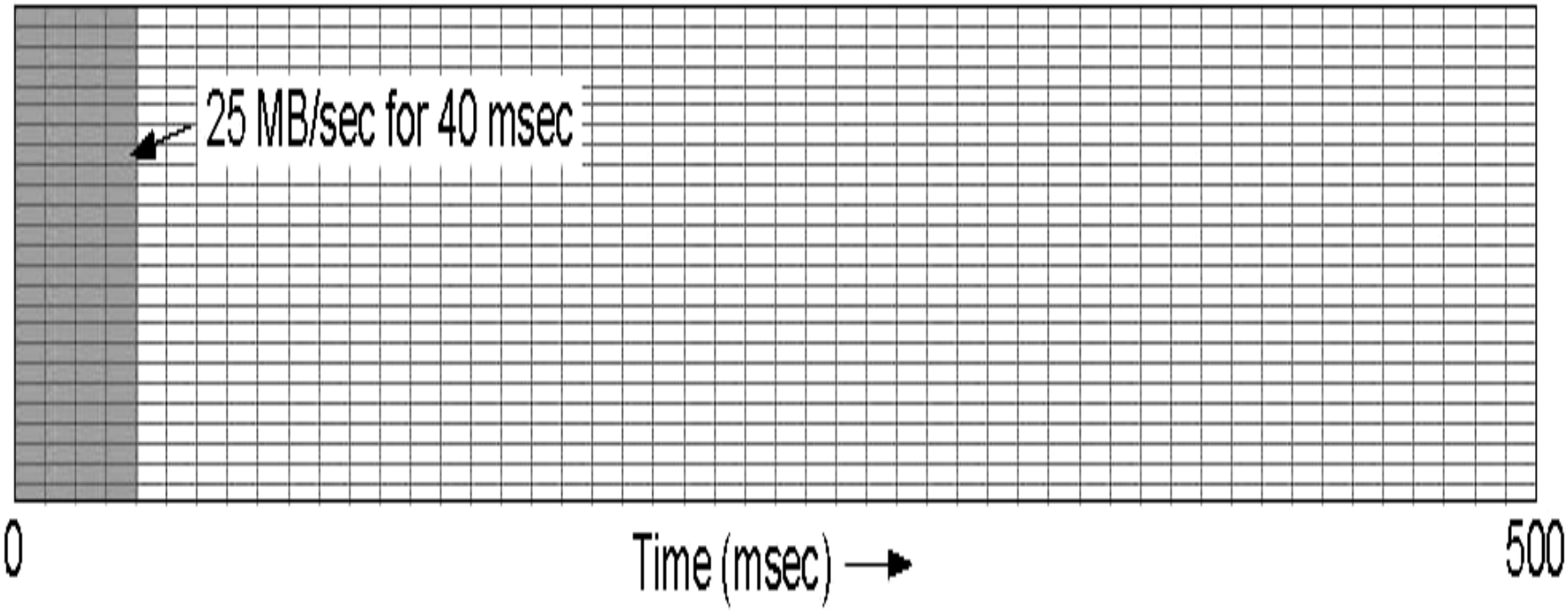
Traffic Policing: The Token Bucket

- Token bucket: traffic policer for each source *i*
  - Bucket of size b_i
  - Deterministic token generator of rate r_i
  - May or may not have buffer (typically provided by the source)
  - Traffic admitted in the network only if there are enough tokens available in the bucket
- Its goal is to regulate the traffic only if it violates its traffic profile agreed in the traffic-contract
- Token bucket is more versatile than the leaky bucket
- Token bucket can be thought of as a conditional shaper



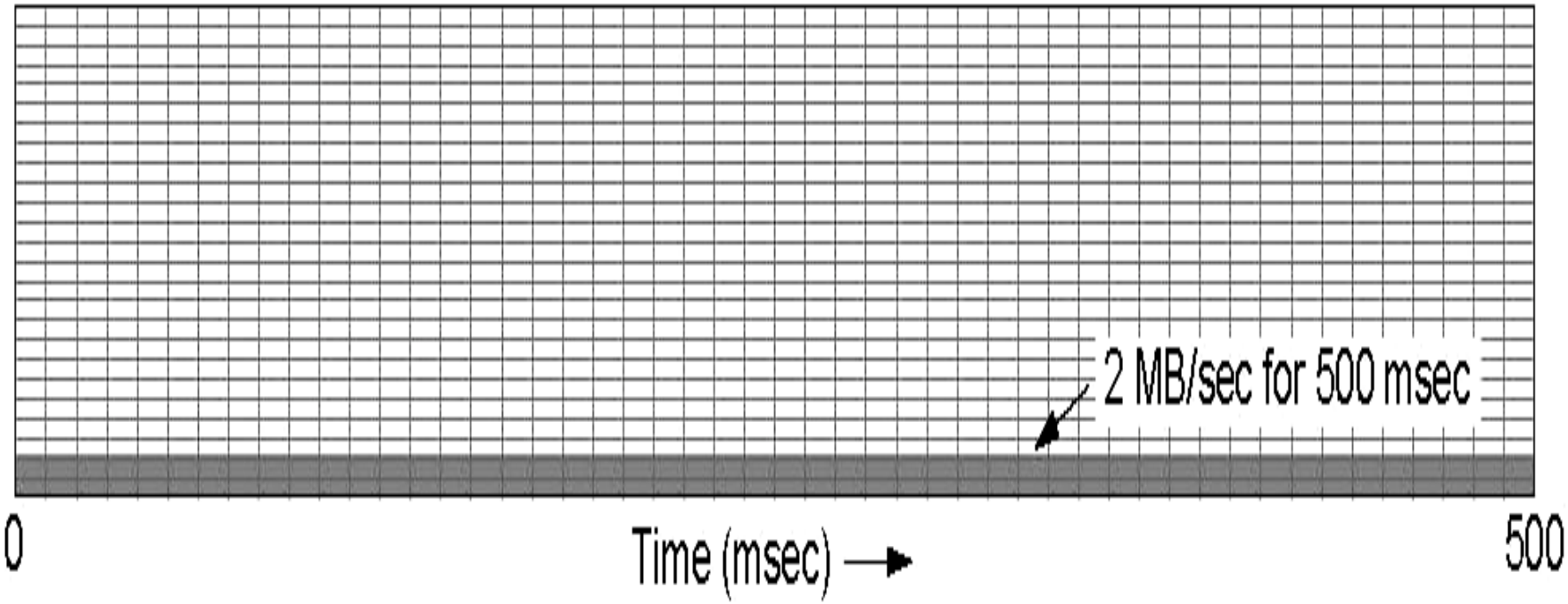
Traffic Policing: Example

- Consider a traffic source that sends a burst of data of 1MB at the peak bitrate of 25MBps for 40 ms



Traffic Policing: Example

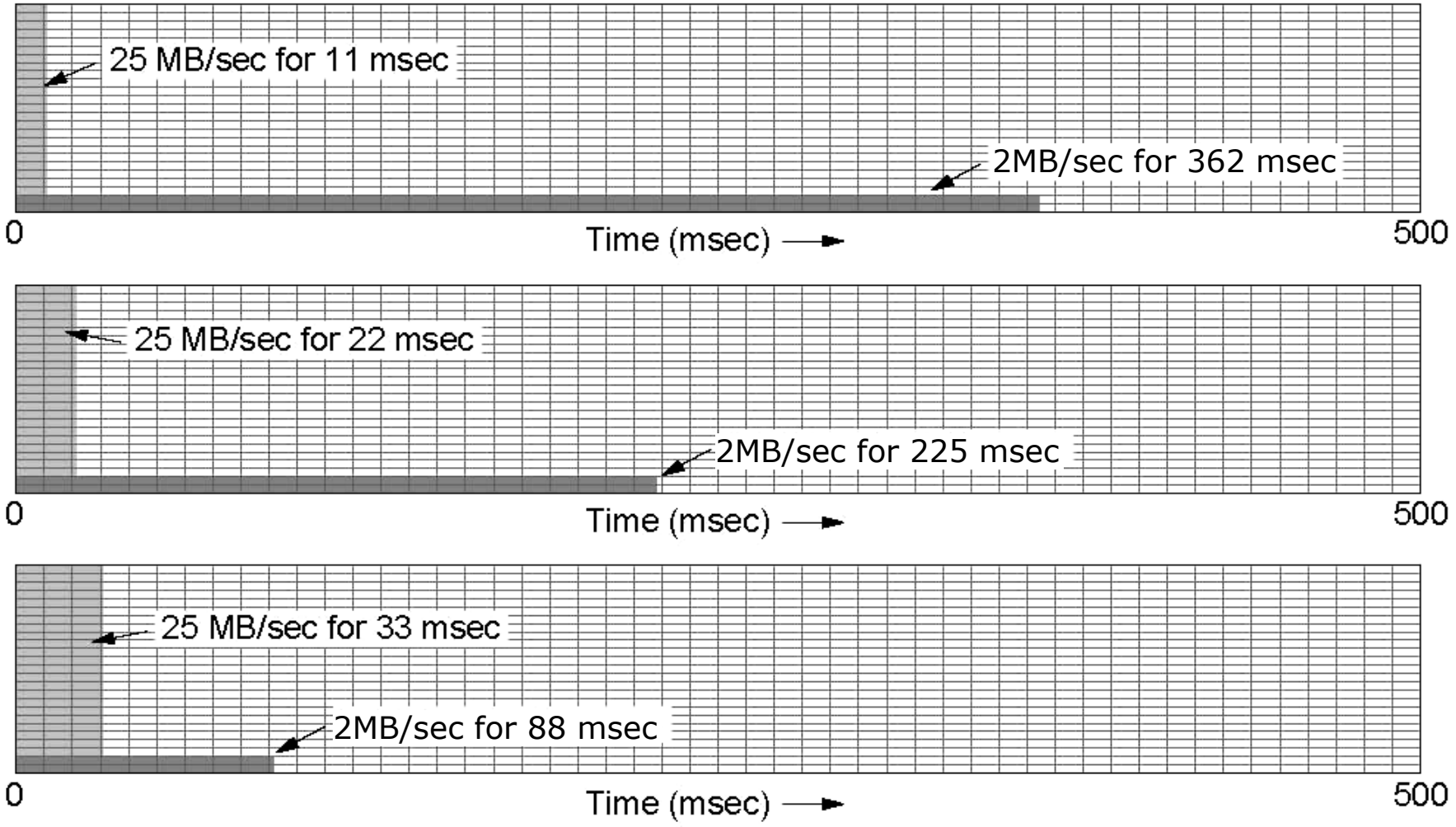
- If we submit to to a leaky bucket with a rate of 2MBps and enough buffer to avoid loss, the output is a long stream of 500ms at 2MBps



- The network access delay is long (the last byte of the raw burst arrive to the LB at 40ms, and will leave the LB and enter the network at 500ms so the worst case access delay 450ms

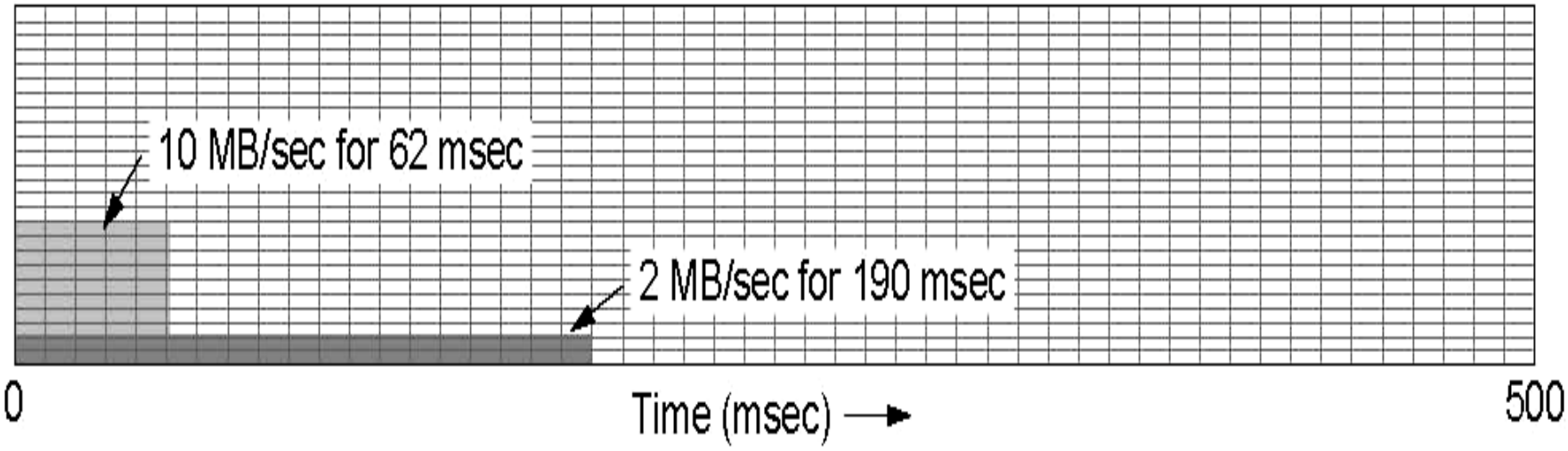
Traffic Policing: Example

- Using the token bucket with the same rate 2MBps and different bucket sizes (250KB, 500KB, 750KB) we have



Traffic Policing: Example

- Typically the token bucket is used to police the maximum burst size, while guaranteeing a sustained rate equal to the token rate
- Using more than one TB allows us to police more than one parameter.
- Example: a (500KB, 2MBps) token bucket followed by a 10MBps leaky bucket can police the maximum burst size and the peak rate.



- Leaky bucket is a particular case of a token bucket with bucket size 1

- One good characteristic of the token bucket as a traffic policer is that it provides a means to make the traffic more manageable
- Notably, the token bucket ensures that the workload, $\nu_i(s,t)$, entering the network in the interval of time (s,t) is bounded by

$$\nu_i(s,t) \leq b_i + r_i(t-s)$$

- If the network element is receiving data directly from a number of such sources then the total workload is bounded by

$$\nu(s,t) \leq \sum_i b_i + \sum_i r_i.(t-s)$$

1. Preventive Congestion Control

Admission Control

Admission Control and Resource Sharing

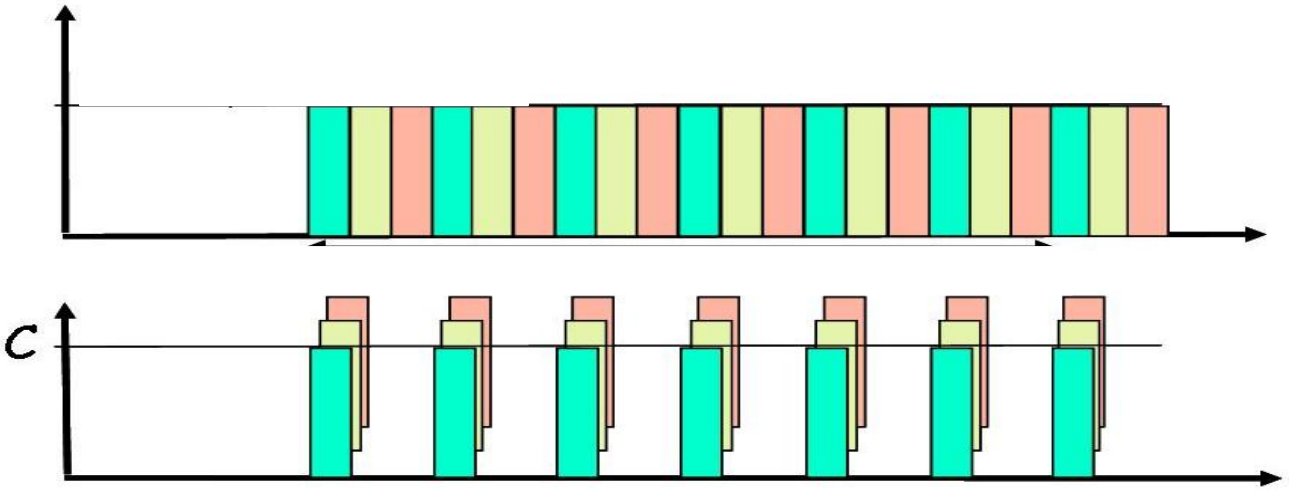
- Two possible approaches to resource sharing
- Example:
  - VOIP source encodes speech at 64Kbps
  - To avoid echo the voice-encoder must frame and send data every 20 ms at most. Packet generation rate is thus 1/0.02 = 50 packets per second (with 160 bytes each)
  - Silence detection and suppression: packets are sent during talk-spurts that last on average 0.4s interspersed with silence periods that last 0.6s on average
- How many such VOIP sources can be multiplexed on a link (or virtual link) of C Mbps?
- Two Possible solutions based on the type of guarantees and the associated admission control process we adopt
  - Rate Envelope Multiplexing, relies on peak rate admission control and achieves deterministic guarantees but possibly low utilization
  - Rate sharing, relies on sustained-rate-based admission control and can achieve statistical guarantees with good bandwidth utilization

Rate Envelope multiplexing

Rate Envelope Multiplexing:

- Useful when multiplexing a large number of lightly bursty sources who require strict QoS guarantees.
- To avoid packet losses, it is sufficient to ensure that the combined overall arrival rate never exceeds the outgoing service rate for more than a packet transmissions time duration

- The only queueing that appears is due to the phase among the sources in the short term



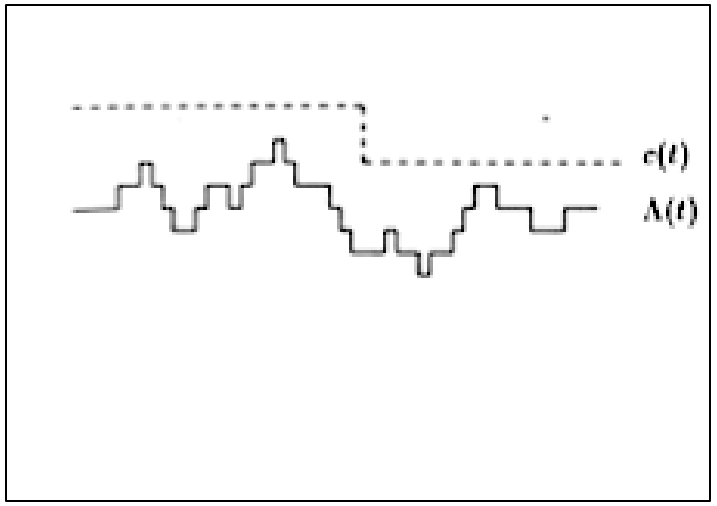
Rate Envelope multiplexing

- Queues can be absorbed with very small buffers (typically a few hundred packets). This is why it is also called buffer-less multiplexing
- Rate Envelope Multiplexing can be achieved by
  - performing admission control to limit the aggregate instantaneous input rate or by
  - increasing the capacity to adjust the rate envelope
- The negligible overload objective is expressed as:

$$\Pr\{\Lambda(t) > C(t)\} < \epsilon$$

- The Packet loss rate can be expressed as

$$PLR \approx \frac{\mathbb{E}\left(\Lambda(t) - C(t)\right)^{+}}{\mathbb{E}(\Lambda(t))}$$



Rate Envelope multiplexing

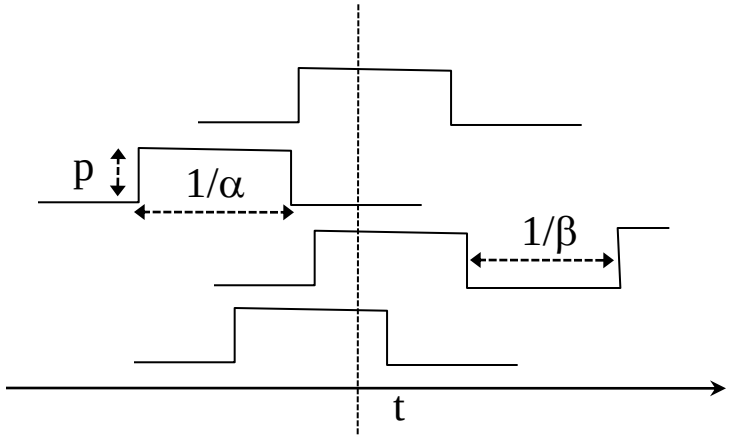
Application:

- N identical Voice sources, each with mean rate *m* and peak rate *p*
- If average talk-spurt duration $1/\alpha$ and average silence $1/\beta$ then we have:

$$m = \frac{p\beta}{\beta + \alpha}$$

- Packet loss rate can be written as

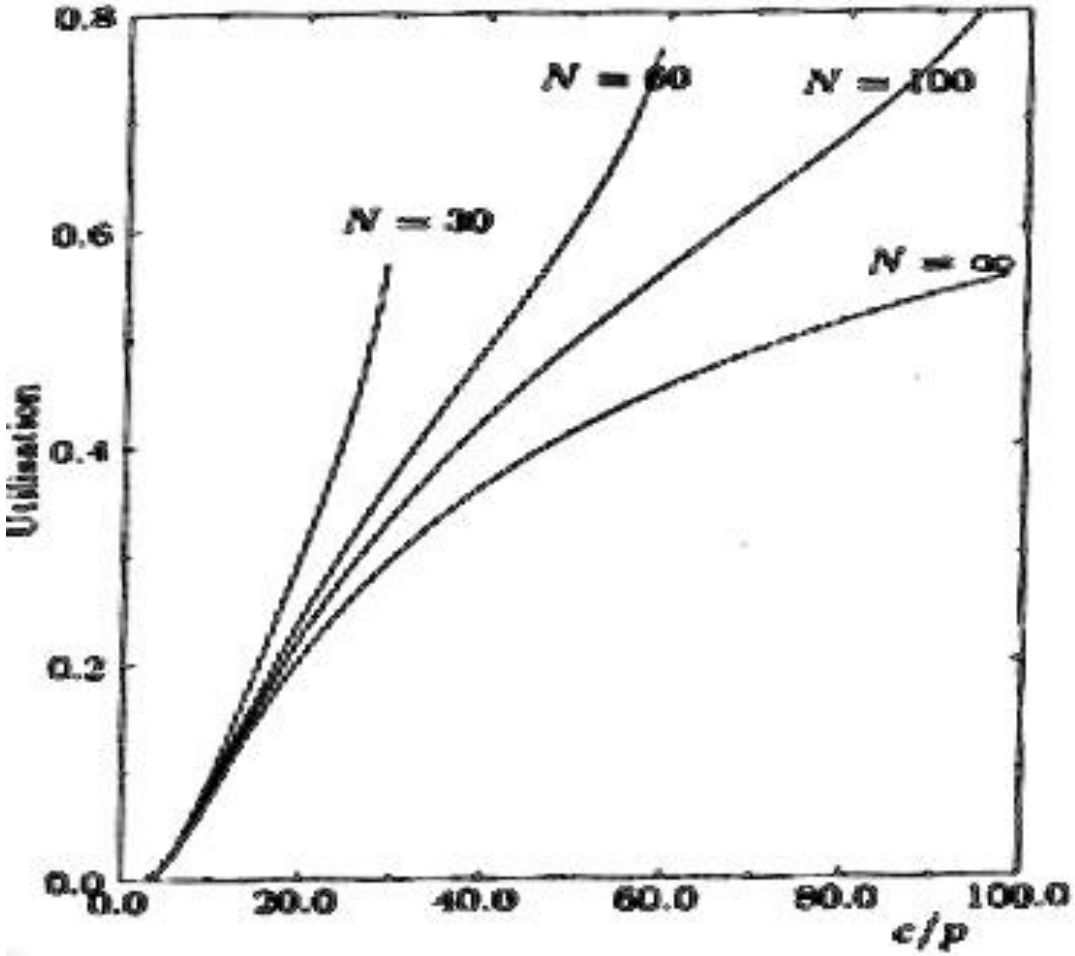
$$PLR = \sum_{i>C/p} (ip - C) \binom{N}{i} \left(\frac{m}{p}\right)^i \left(1 - \frac{m}{p}\right)^{N-i} \times \frac{1}{Nm}$$



Rate Envelope multiplexing

Application:

- For a PLR of 10^{-9}, the utilization Nm/C of the bandwidth is large only if the peak rate p is a small fraction of the capacity C as shown in the figure



Rate Envelope multiplexing

- Rate envelope multiplexing makes sense only for a large number of sources with low traffic rate variation and some deterministic QoS requirements

- Typically admission control for rate envelope multiplexing can be done using peak rate allocation:
- For each source
  - Admit a source if its peak rate does not exceed the available bandwidth
  - Subtract the peak rate from the available bandwidth

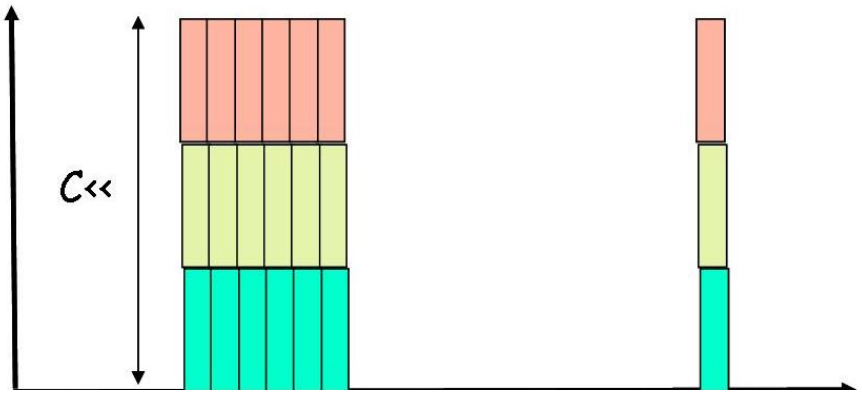
Rate Sharing

Rate Sharing:

- Useful when multiplexing number of highly bursty sources who require statistical QoS guarantees
- The individual source peak rate is of the same order of magnitude as the link bandwidth. To avoid packet losses yet achieve a high utilization, we need large buffers to absorb the traffic that arrives during long overload periods

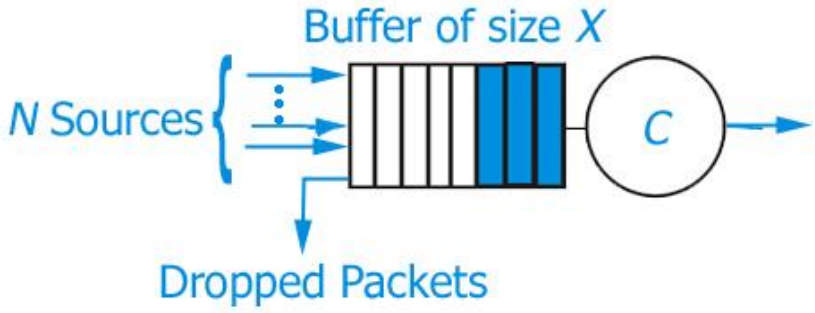
$$\Pr\{\Lambda(t) > C(t)\} >> \epsilon$$

$$\bar{\Lambda} < \bar{C}$$



Rate Sharing

- For rate envelope multiplexing admission control does not take into account the buffer performance, as the queue does not affect the performance much in terms of the QoS guarantees.
- For Rate Sharing since the queues are large and can be busy for large periods of time, it is necessary to take into account the performance of such queues.
  - Deterministic performance guarantees
    - Can be obtained by investigating the worst case traffic scenario which is facilitated by the traffic policing
  - Statistical performance guarantees
    - Can be obtained by characterizing the statistical performance of a queueing system by accurately modelling the traffic behavior



Rate Sharing: Deterministic Guarantees

- Deterministic performance guarantees can be obtained by assuming the source to be controlled with traffic policers such as the token bucket
- Recall that a token bucket ensures that $\nu_i(s,t)$ the work entering the network in the interval (s,t) is bounded by

$$\nu_i(s,t) \leq b_i + r_i(t-s)$$

- If the network element is receiving data directly from a number of such sources then the total workload is bounded by

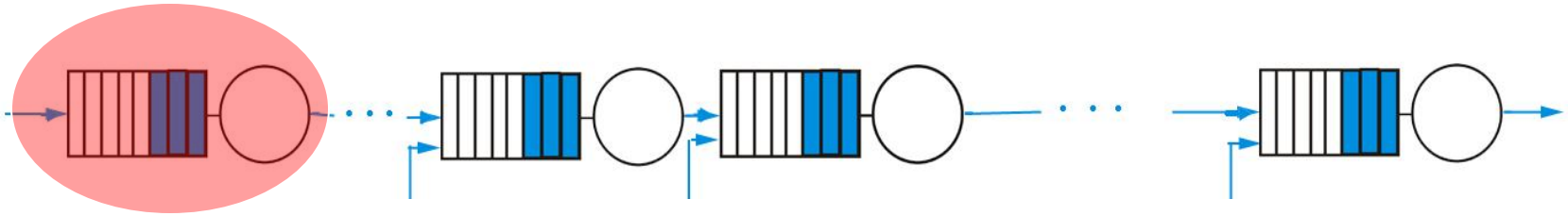
$$\nu(s,t) \leq \sum_i b_i + \sum_i r_i.(t-s)$$

Rate Sharing: Deterministic Guarantees

- By Reich Theorem the work in the queue at time t satisfies

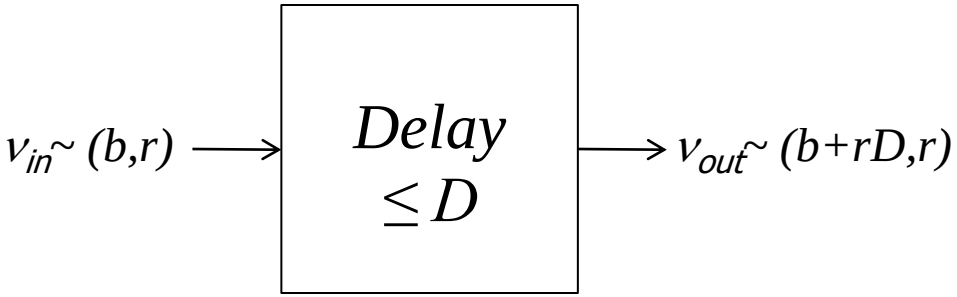
$$\begin{aligned} W_t &= \sup_{s \le t} \{\nu(s,t) - C(t-s)\} \\ &\le \sup_{s \le t} \left\{ \sum b_i + \left(\sum r_i - C \right) .(t-s) \right\} \\ &\le \sum b_i \end{aligned}$$

- This states that: If the sources are constrained with token buckets of parameters (b_i, r_i), the in the worst case:
  - assigning a rate $C \geq \sum r_i$ and a buffer of capacity $B \geq \sum b_i$ avoids packet losses and the delay in a FIFO queue is bounded by $\sum b_i / C$
- One problem with this though is that the result is only valid if the router is fed directly by the Token Buckets. How to estimate then the end-to-end delay?



Rate Sharing: Deterministic Guarantees

- Cruz has shown in [1] that a traffic that satisfies the token bucket burstiness bound and that is subjected to a network element can be characterized as it moves through the network (in a multi-hop network)
- He studied
  - Delay elements: is an element that introduces a delay bounded by D, then the output verifies

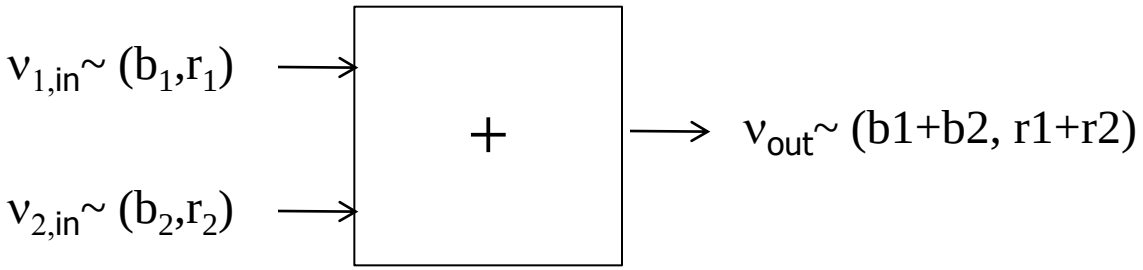


$$\nu_{out}(s,t) \leq b + rD + r(t-s)$$

[1] R. Cruz, Calculus for network Delay, Transactions on Information Theory

Rate Sharing: Deterministic Guarantees

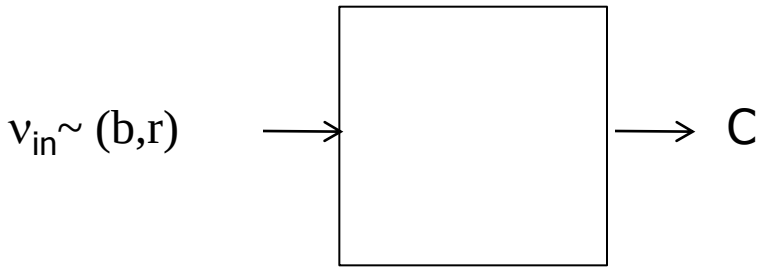
- Adder: an adder is an element that mixes two or more traffic streams without delaying them



$$\nu_{out}(s,t) \leq (b_1 + b_2) + (r_1 + r_2)(t - s)$$

Rate Sharing: Deterministic Guarantees

- Work conserving element: is an element which typically represents a router's output queue. Traffic that comes in is delivered to the output at rate C. The server is never idle when there is traffic.

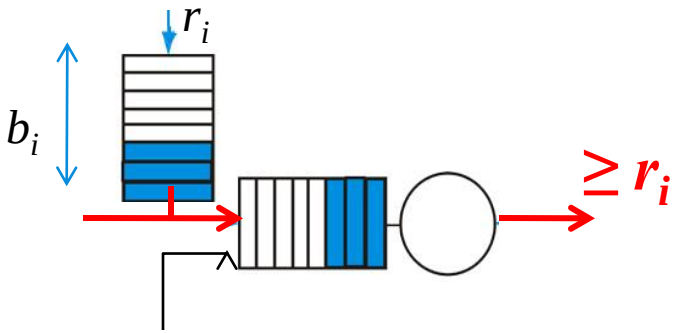


- A token bucket policed traffic stream submitted to a work conserving element of rate C experiences a delay of no more than

$$D = \frac{b}{C - r}$$

Rate Sharing: Deterministic Guarantees

- Using these basic properties, we can show [2] for a class of servers that guarantee a minimum rate r_i for source i along the end-to-end path that the token bucket burstiness bounds are preserved



- Property is highly desirable because if every router along the end-to-end path guarantees a rate no less than r_i for source i then:
  - Packet loss is avoided by simply reserving buffer size b_i for each source
  - The delay of any packet is bounded by b_i / r_i at each router

[2] A. Parekh and R. Gallager, A generalized processor sharing approach to flow control in integrated services networks IEEE Transactions on Networking June 1993

- Since the service rate is no less than r_i, *by Reich Theorem,* V_{it}, the amount of work of flow *i* in the output queue of the router at time t is:

$$\begin{aligned} V_{it} &\leq \sup_{s<t} \{\nu_i(s,t) - r_i(t-s)\} \\ &\leq b_i \end{aligned}$$

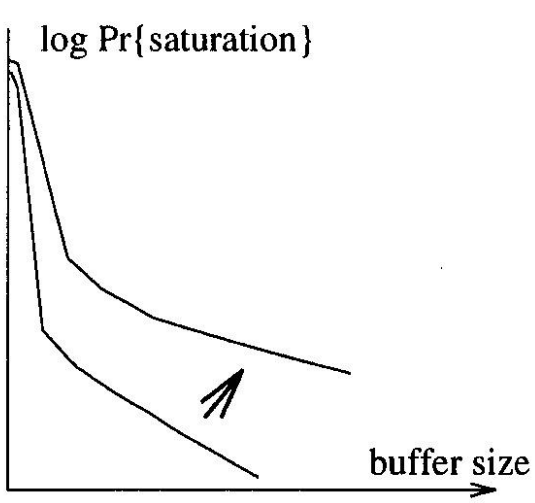
- If $\xi_i(t,u)$ the amount of work from flow *i* leaving the router queue in (t,u)

$$\begin{aligned} \xi_i(t,u) &\leq V_{it} + \nu_i(t,u) \\ &\leq \sup_{s<t} \{\nu_i(s,t) - r_i(t-s)\} + \nu_i(t,u) \\ &= \sup_{s<t} \{\nu_i(s,u) - r_i(t-s)\} \\ &\leq \sup_{s<t} \{r_i(u-s) + b_i - r_i(t-s)\} \\ &= r_i(u-t) + b_i \end{aligned}$$

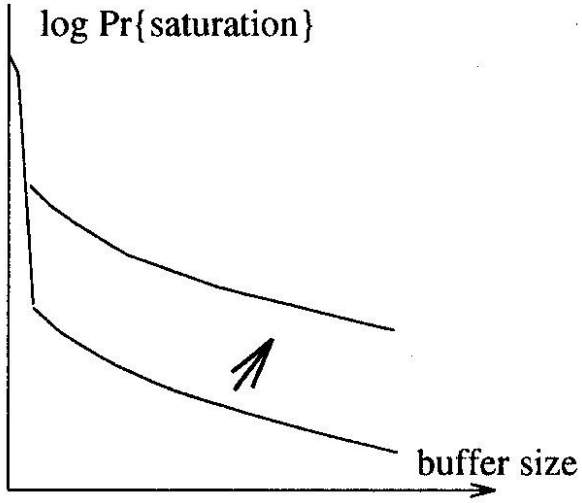
Rate Sharing: Stochastic Guarantees

- One drawback with deterministic guarantees is that they ignore statistical multiplexing gains and result in possibly very large buffers
- Also, for many applications it is sometimes sufficient to only guarantee probabilistic packet loss rate so as to accept more than the strict minimum number of flows
- The difficulty however lies in the very complex relation between performance (probability of saturation), traffic characteristics, and network capacity. For example for on/off sources
  - Non-linear relation
  - Depends on the load
  - Depends on the Peak rate
  - Depends on the mean burst size
  - Depends on the burst variance

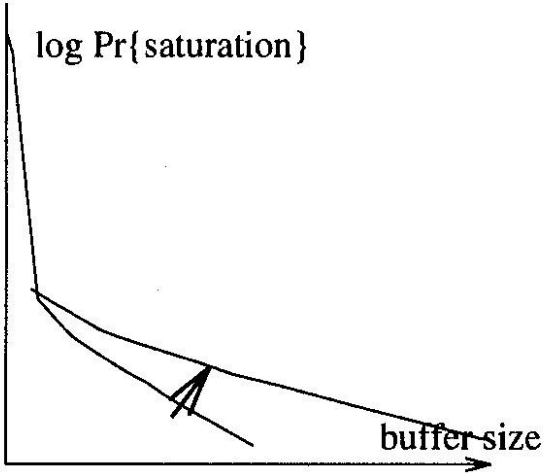
Rate Sharing: Stochastic Guarantees



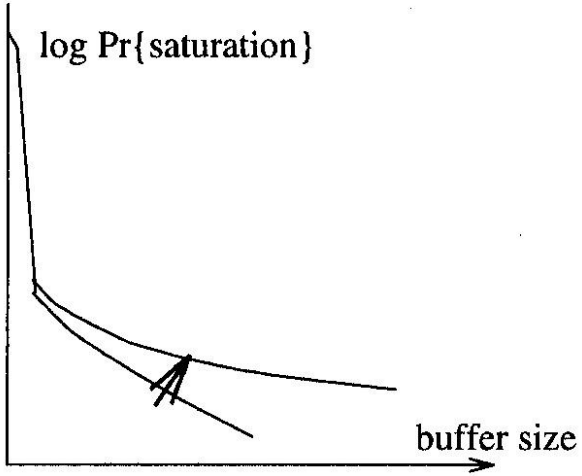
a) Effect of increasing load



b) Effect of increasing peak rate



c) Effect of increasing mean burst

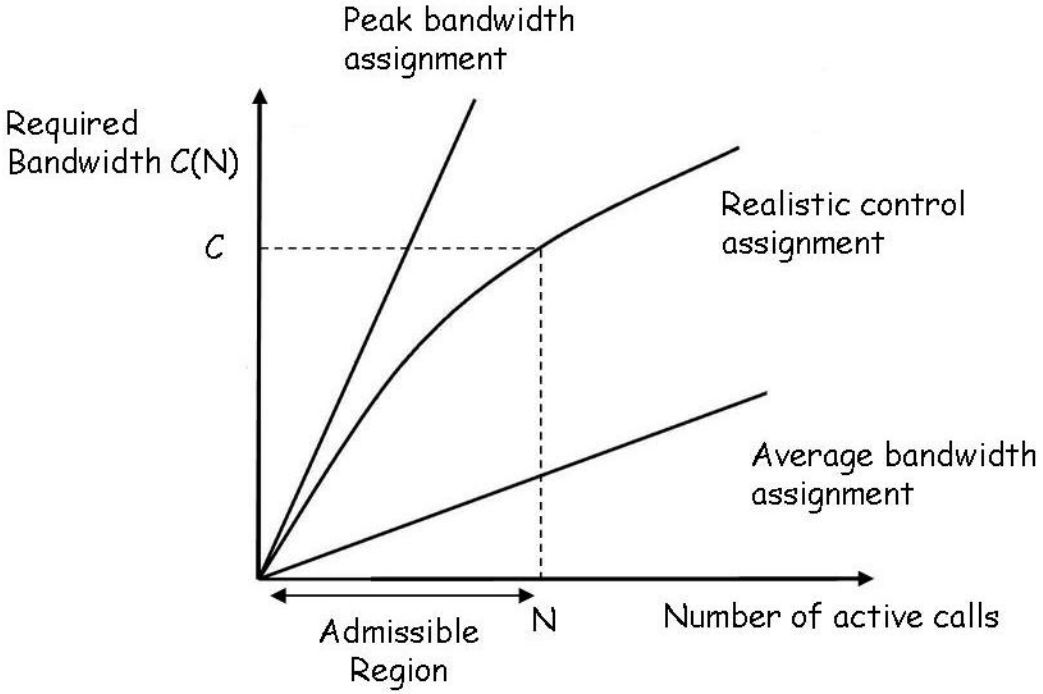


d) Effect of increasing burst variance

Rate Sharing: Stochastic Guarantees

- To guarantee stochastic bounds in rate sharing, the admission control process must answer the following question
- Given the traffic characteristics, and the Buffer size, how large should the capacity C of a router be so that the Probability of overload is less than a predetermined threshold

$$\Pr\{X_t > B\} \leq \epsilon$$



2. Reactive Congestion Control

Internet Design Principles

- The Internet has been designed with the following underlying principles
- **Scale**: Protocols should work in networks of all sizes and distances
- **Incremental deployment:** New protocols need to be deployed gradually
- **Heterogeneity**: Different technologies make up the infrastructure and such infrastructure is controlled by autonomous organizations
- As a result, the **end-to-end argument** became a cornerstone of the Internet design
- **end-to-end argument**: Networking functions should be relegated to the edge of the network (end-hosts)

TCP and Congestion Control

- Originally, TCP was designed as a reliable data transfer protocol over unreliable IP
- Before 1988, it only implemented a Go-Back-N protocol with a flow control window
- It did not have a congestion control mechanism
- As traffic demands increased, congestion collapse appeared in the network, bringing with it the necessity of an additional mechanism to control congestion
- Typically, reactive congestion control needs two entities
  - A set of rules for the source to adapt its rate in response to congestion to limit the amount of data in flight in the network
  - A mechanism for the network to send the source implicit or explicit congestion signals

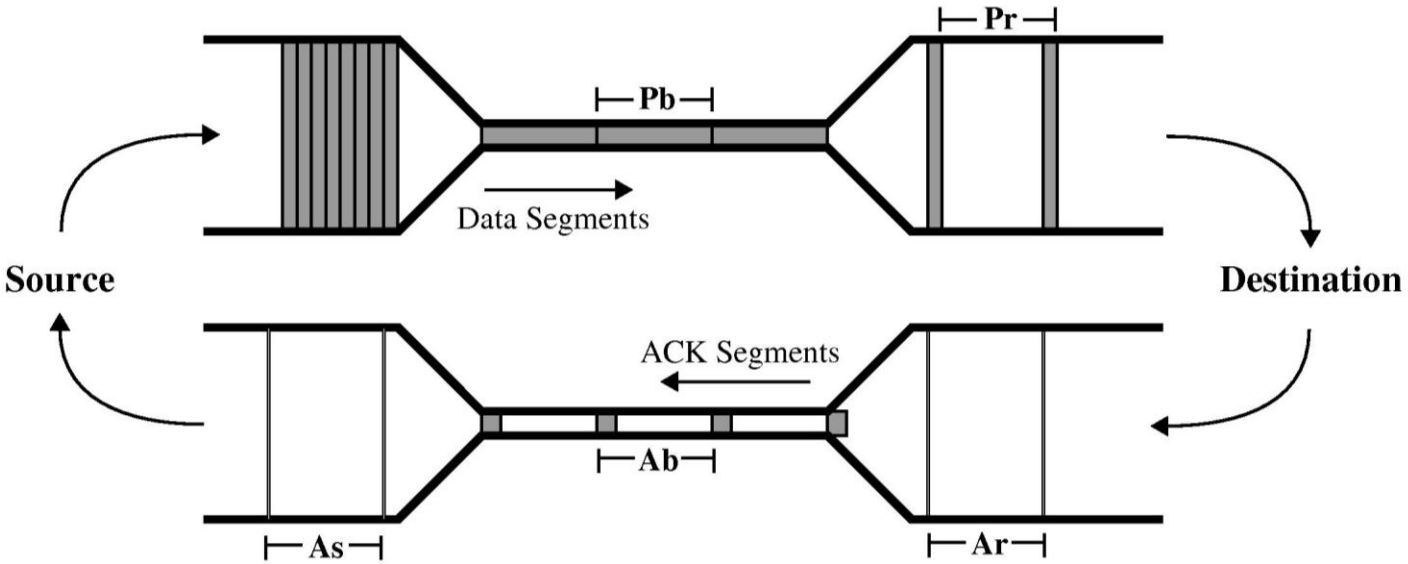
- Typically a TCP source's data rate can be written as

$$\text{Source rate} = \frac{N \times MSS}{RTT}$$

- To adapt the source rate only N, the window size, can be changed for each round trip time
  - A too large value of N may lead to severe congestion
  - A too small value of N may lead to inefficient use of bandwidth
  - We need an algorithm to adapt N while probing for the available bandwidth
- It is desirable that the new algorithm
  - Achieves a very high bandwidth utilization
  - Avoids congestion as fast as possible
  - Shares the available bandwidth fairly among the flows
  - Adhere to the design principles of the Internet

TCP Self-Clocking

- The first problem to solve is the need for an accurate clock to adapt N
  - OS clocks rely on non-real time clock (jiffy) with a coarse accuracy that can be larger than the typical RTT (100ms across an ocean)
    - 200-500ms in the early days of the Internet
    - Today the jiffy 1-10ms (in Linux Kernel)
- Fortunately, TCP is self-clocked (arriving ACKs are the clock ticks)
  - Instead of increasing or decreasing the window on a per-RTT basis, we can increase or decrease it fractionally for each ACK



TCP: Tahoe 1988

- TCP Tahoe uses the lack of ACK packets as an implicit indication of congestion
- The algorithm is divided in two phases
  - Slow-start: A phase where the source tries to obtain a coarse estimate of the available bandwidth by doubling the window in each RTT (up to a predefined threshold)

```
Initially: set cwnd = 1MSS ;
For Each ACK Do
        cwnd = cwnd + MSS ;
        If (cwnd > ssthreshold )
                Set state to "Congestion Avoidance"
        End if
End for
```

TCP: Tahoe 1988

- Congestion Avoidance is a state where the source probes for the available bandwidth slowly by increasing the window by one segment in each RTT

For Each ACK Do

cwnd = cwnd + MSS x MSS/cwnd ;

End for

- Losses are detected by timeout or by three duplicate ACKs (4 ACKs for the same sequence number)

For Each Loss Do

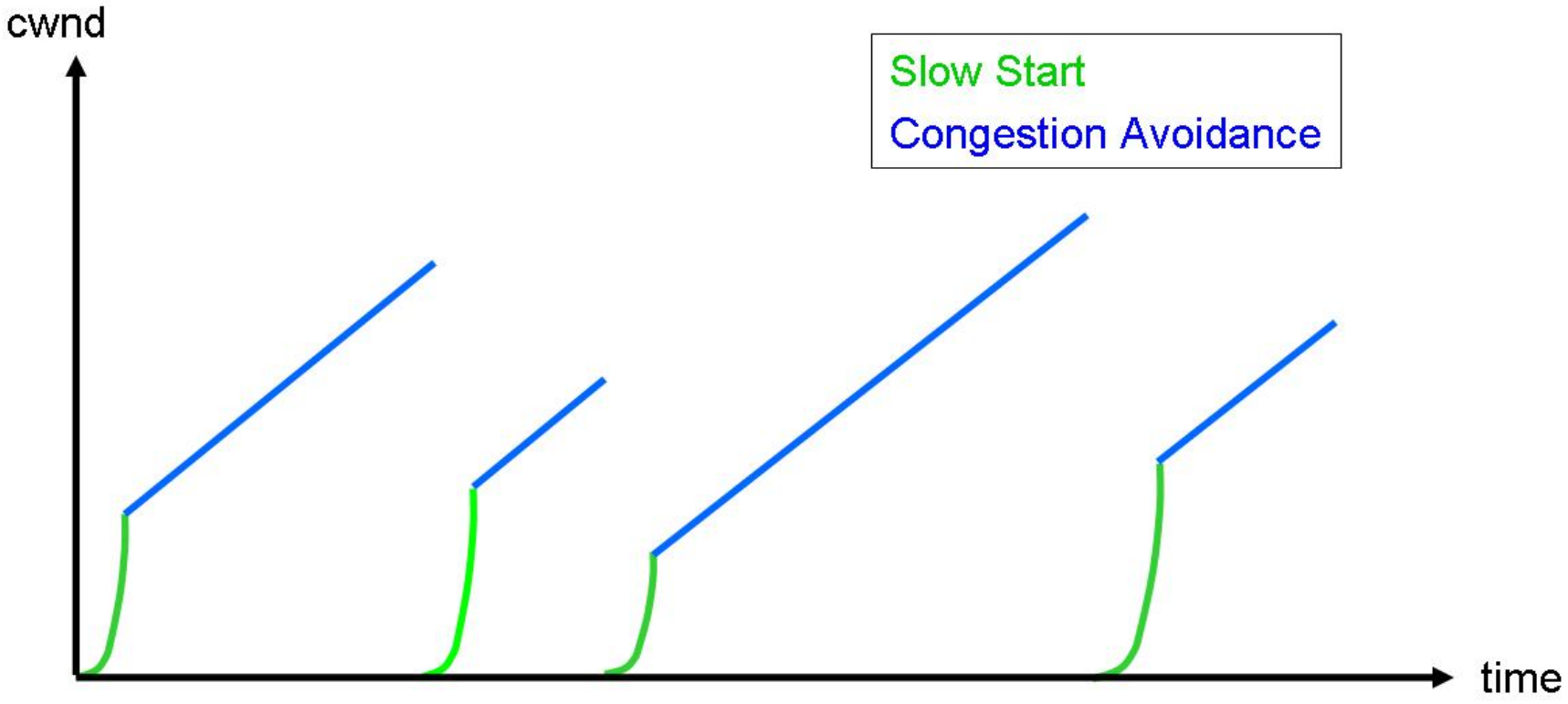
ssthreshold = cwnd/2;

cwnd = 1 MSS ;

Set state to "Slow Start"

End for

TCP: Tahoe 1988



TCP: Reno 1990

- Three duplicate ACKs indicate a loss, but also that congestion is mild (three segments sent after the lost one left the network successfully)
- Because congestion is mild no need to go to slow start state

For Each Loss detected by 3 Dup ACKs Do

ssthreshold = cwnd/2;

cwnd = ssthreshold;

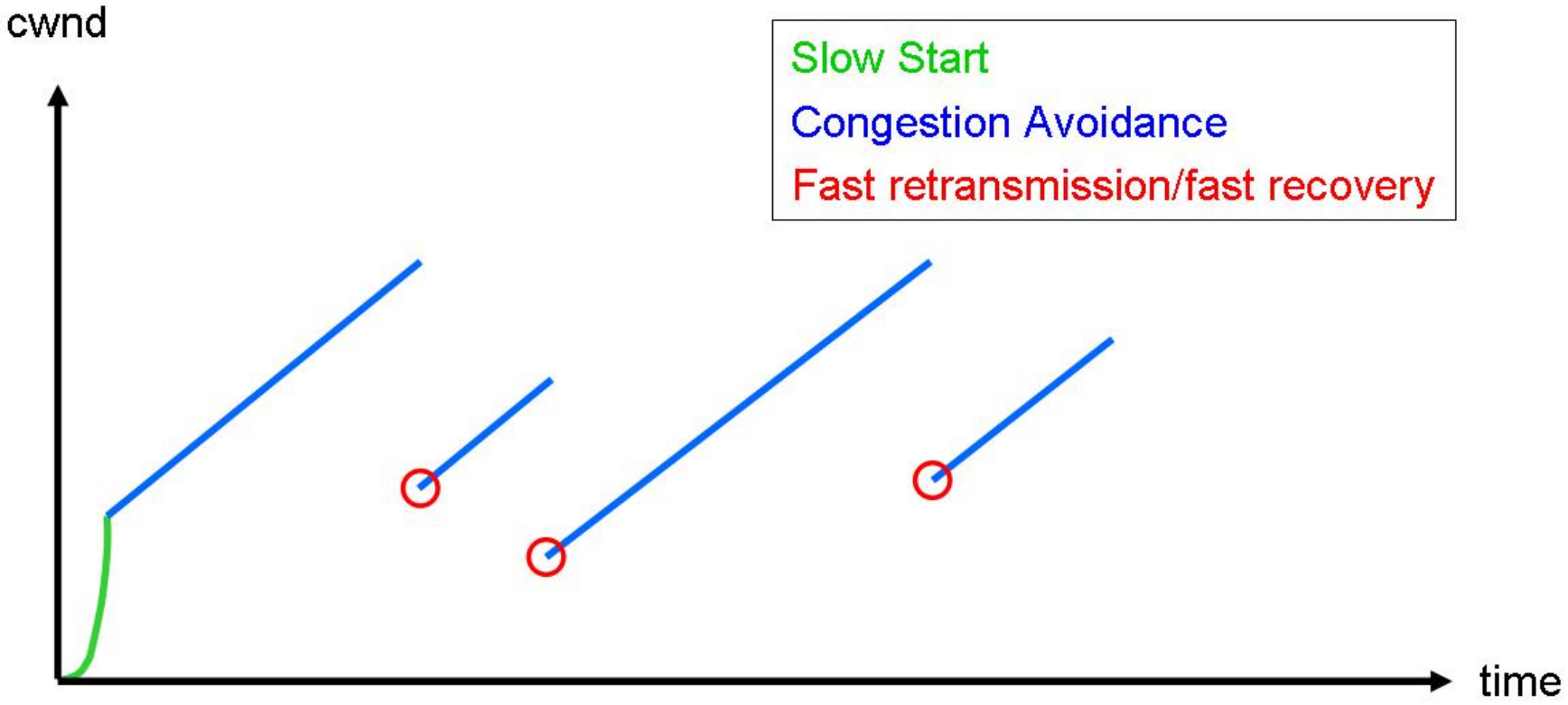
retransmit the lost segment immediately

Set state to "Fast Recovery"

End for

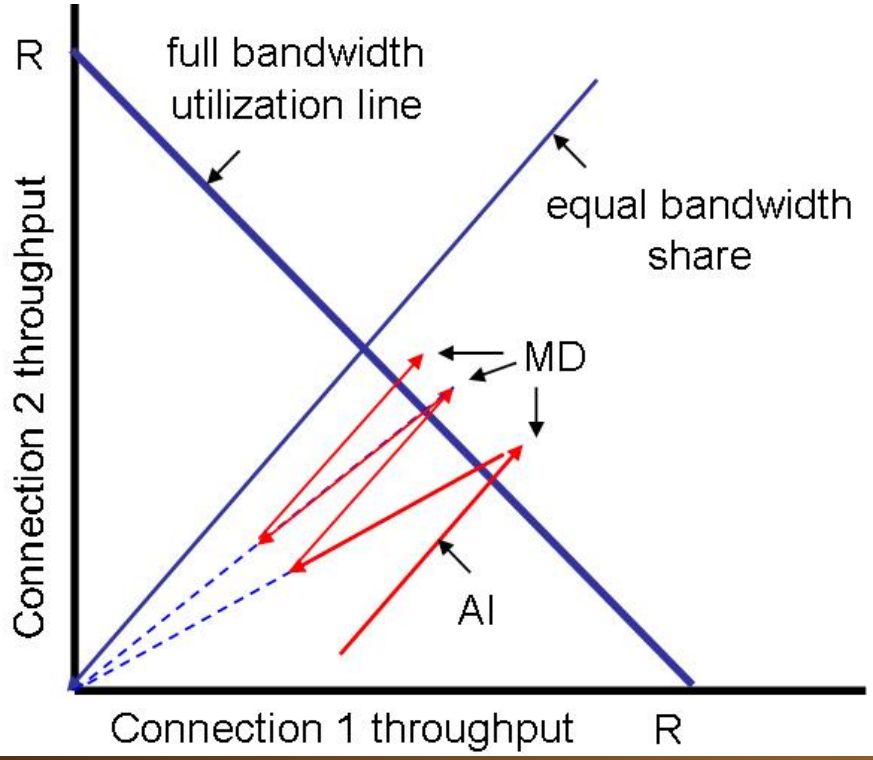
- In Fast recovery
  - Inflate the window artificially by 1 segment for each dup Ack
  - Send one segment if the window allows
  - If ACK for new data arrives (not dup ACK)
    - Deflate the window
    - Set state to "congestion avoidance"

TCP Reno 1990



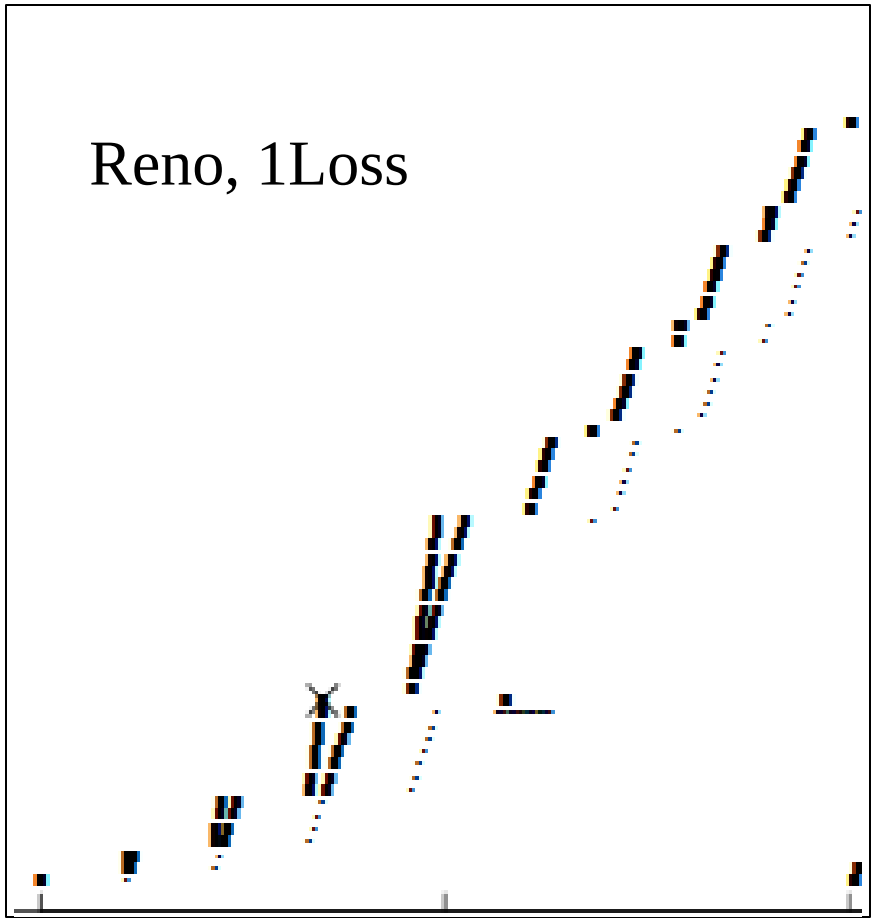
TCP: Reno Characteristics

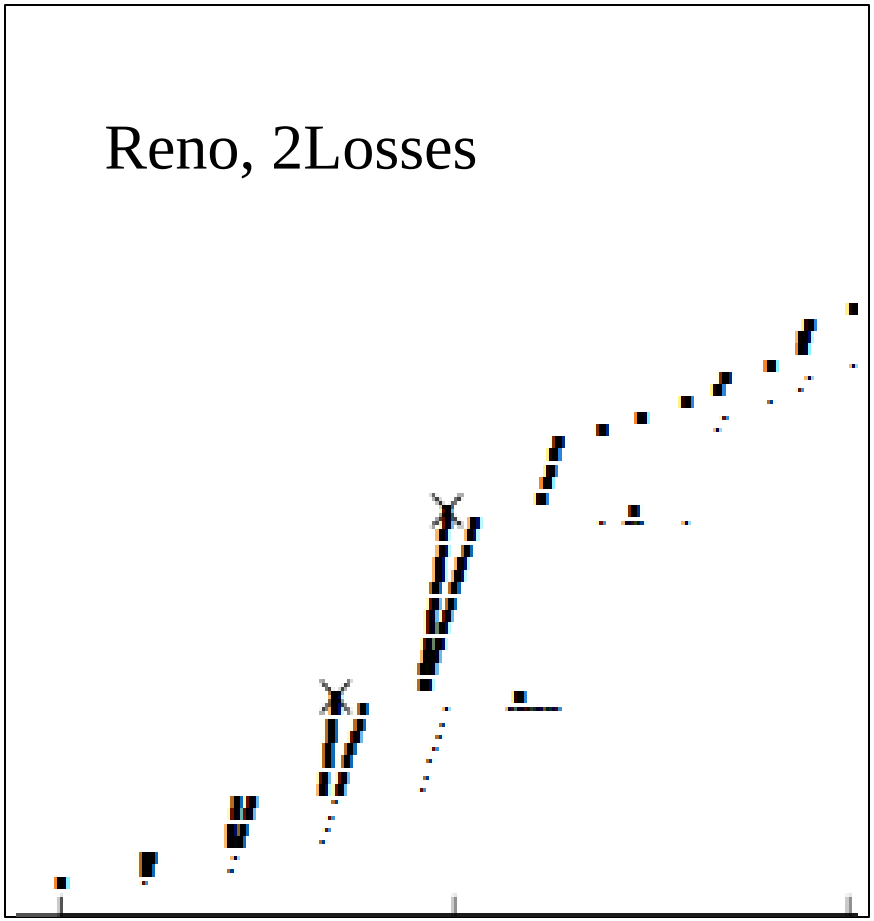
- TCP Reno (and its variations) makes up most TCP implementations in today's Internet because it fulfills all the requirements
  - Efficient, because in steady state it operates in congestion avoidance and it is conservative so no risk of congestion collapse
  - End-to-end and does not rely on network assistance
  - Achieves fairness (relatively) among flows sharing a bottleneck



TCP Reno Problem

- Low throughput when more than one loss occurs per flight of data [3]





[3] K. Fall and S. Floyd Simulation-based Comparisons of Tahoe, Reno, and SACK TCP

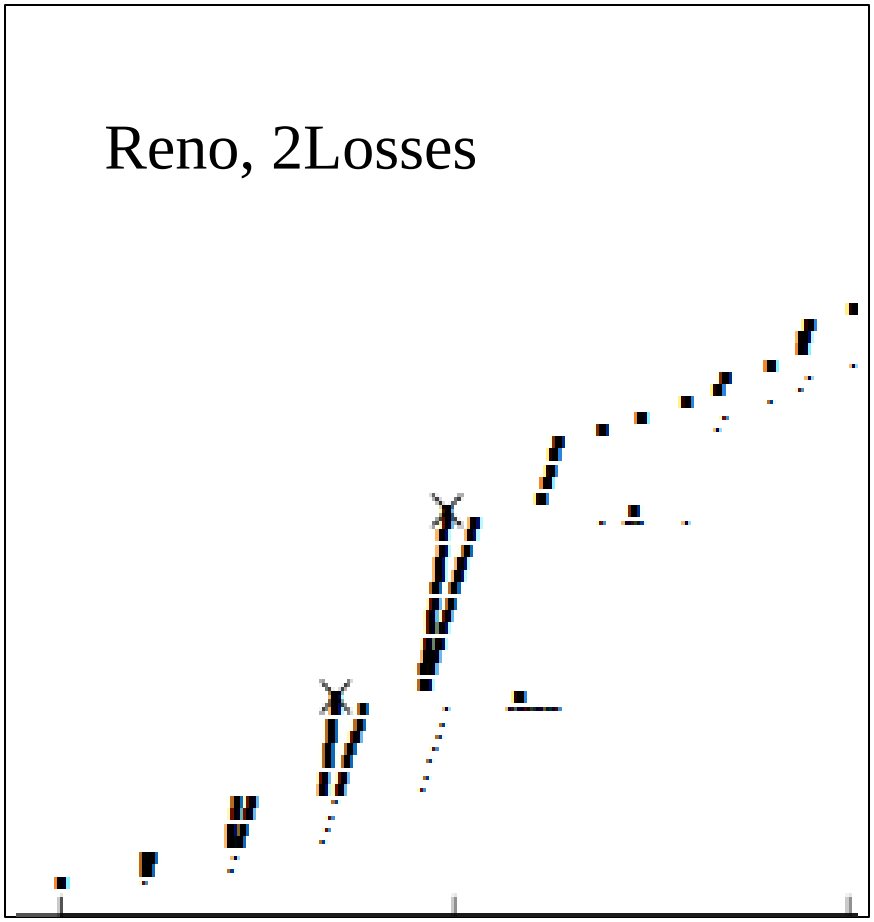
TCP NewReno 1999

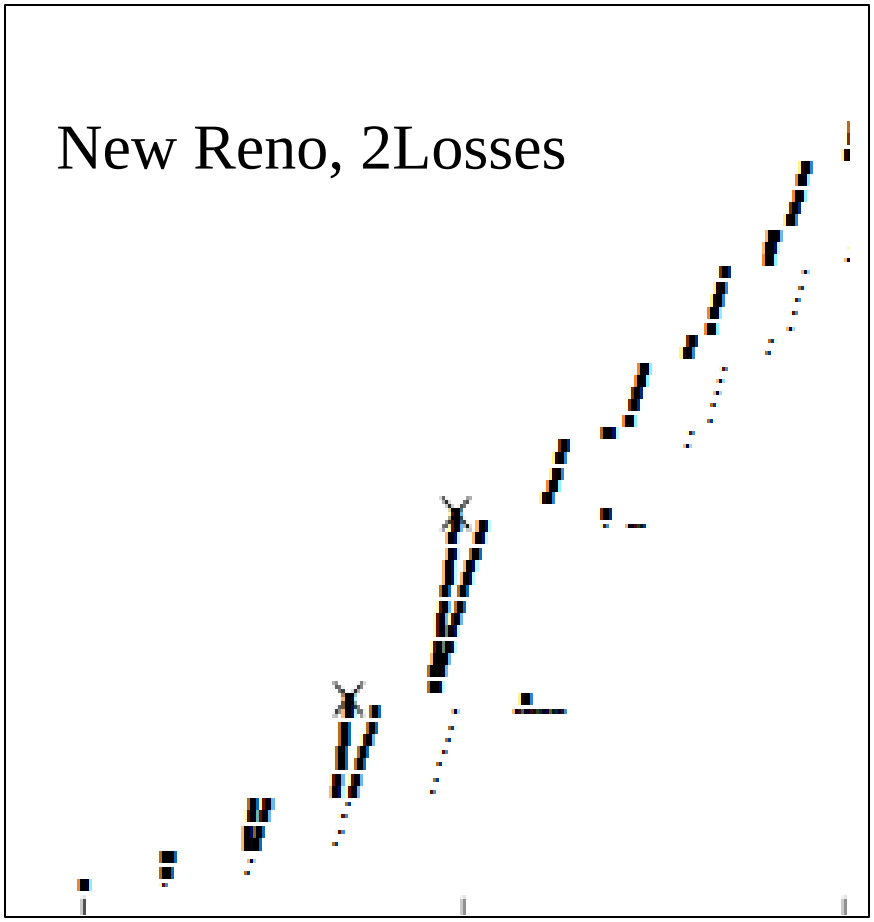
- TCP new Reno is proposed [4] to overcome this problem by avoiding to exit fast recovery prematurely
- New Reno:
  - Upon 3 Dup ACKs record sequence number of the last segment sent before receiving the third dup ACK in *recover* and enter fast recovery state
  - If a new ACK is received
    - It ACK sequence number = recover then
      - Exit from Fast recovery to Congestion Avoidance
      - Deflate the window
    - If ACK sequence number < recover then (this is a partial ACK) one or more packets in flight are lost
      - Retransmit lost packet
      - Deflate the window
      - Reset the timer to avoid RTO
      - Stay in Fast Recovery

[4] S. Floyd T. Henderson The NewReno Modification to TCP's Fast Recovery Algorithm RFC 2582 and 3782

TCP New Reno vs. TCP Reno, 2 Losses

- New Reno recovers well when more than one loss occurs per flight of data [3]





[3] K. Fall and S. Floyd Simulation-based Comparisons of Tahoe, Reno, and SACK TCP

TCP Vegas

- TCP Vegas uses the delay variation as an indication of the queue size in the bottleneck router to detect congestion
- At the bottleneck router
  - Delay increases fast when load increases
  - Delay remains quasi constant when load is light
- Vegas keeps track of the best achieved delay in RTT_{min} which in the best case reflects the delay when there is no queueing at all, and uses it as a benchmark

For Each RTT Do

If (cwnd/RTTmin – cwnd/RTT < a) cwnd = cwnd + 1;

If (cwnd/RTTmin – cwnd/RTT > a) cwnd = cwnd - 1;

Otherwise leave cwnd unchanged

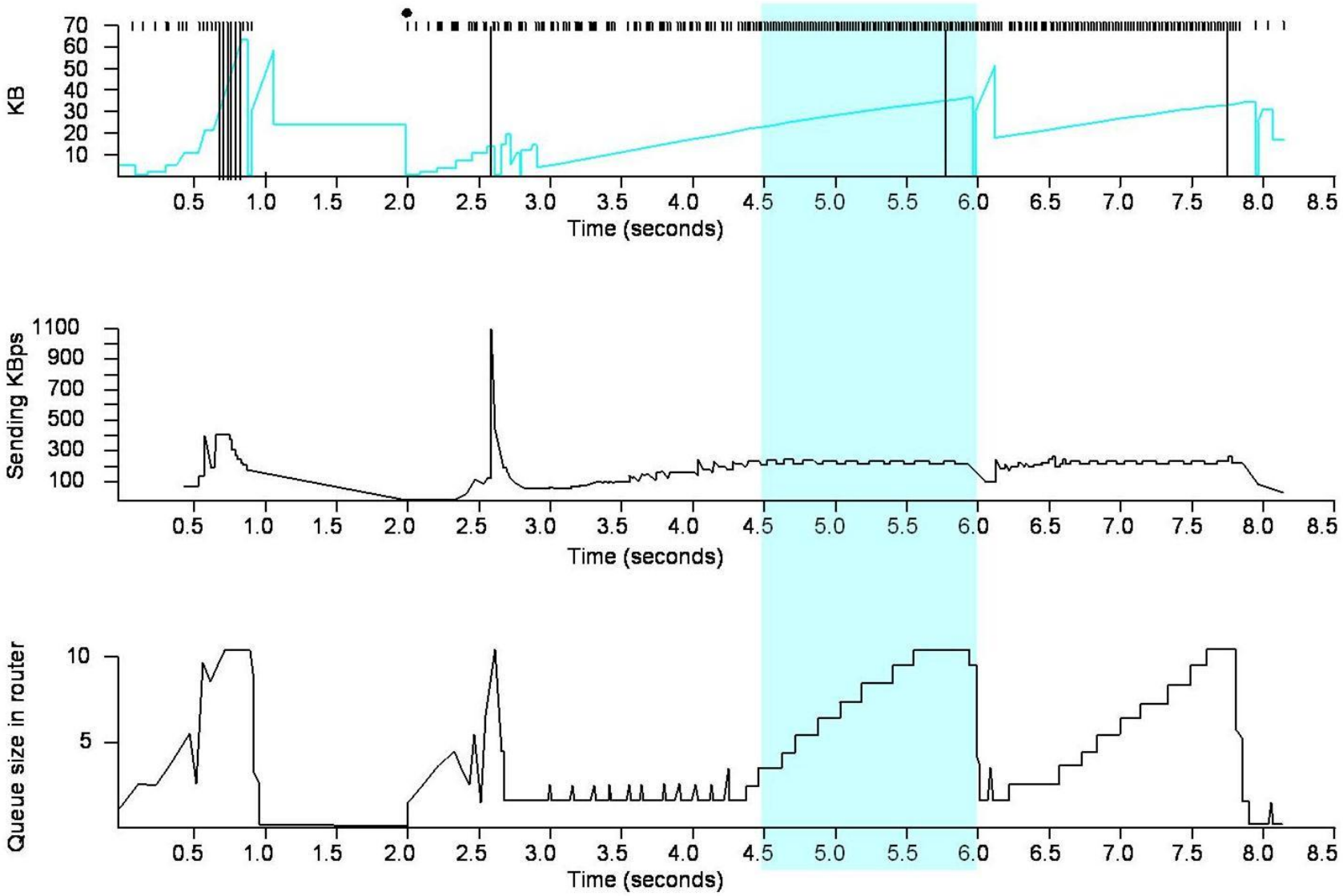
End for

For Each Loss Do

cwnd = cwnd/2;

End for

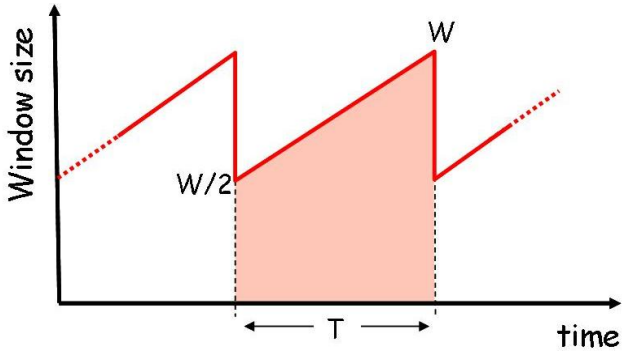
TCP Vegas


KB
70
60
50
40
30
20
10
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
Time (seconds)
Sending KBps
1100
900
700
500
300
100
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
Time (seconds)
Queue size in router
10
5
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
Time (seconds)

TCP Friendly Rate Control

- Dramatic increase of applications that UDP instead of TCP (e.g., Video streaming, VoIP, …)
- Such applications are often insensitive to loss and more sensitive to delay and delay jitter therefore they cannot work well with TCP's elastic rate
- When there is excessive UDP traffic in the network TCP flows may face starvation
- There are two approaches to deal with this problem
  - Upon congestion, penalize non-responsive (UDP) traffic severely. This requires the intervention of the routers (e.g., Active Queue Management with penalties; flow-aware service at the routers, ...)
  - Keep the network unchanged, and **make the end applications TCP-friendly**

- What is TCP friendliness?
- A flow is said to be TCP-Friendly if given a level of congestion it does not try to take more bandwidth than a TCP compliant flow
- To develop a TCP-Friendly Rate Control (TFRC) we need to characterize the TCP throughput first.
- Many studies have been conducted for this purpose, the essential conclusion of most is that TCP throughput is inversely proportional to the square root of the loss probability [5]
- Simple Approximation:
  - Steady state -> AIMD
  - No change in the load -> window when loss occurs is constant



[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, Modeling TCP Throughput: a Simple Model and its Empirical Validation

- A loss occurs when the window is equal to W. As a result the window evolves linearly between W/2 and W in a round between two losses
- So if N is the number of packets sent within one round the loss probability p is 1/N

$$N = \frac{1}{2}\frac{T}{RTT}\left(\frac{W}{2} + W\right)$$

- T is the time it takes to ramp up the window from W/2 to W linearly

$$T = RTT\left(W - \frac{W}{2}\right)$$

- The loss probability is then

$$p = 1 \Big/ \frac{W}{4}\left(\frac{W}{2} + W\right)$$

- Or

$$W \approx \sqrt{\frac{8}{3p}}$$

TCP Friendly Rate Control

- Basic Idea:
- The Server (multimedia source)
  - Measures the RTT and RTO
  - Use the TCP response function (square root equation), to estimate a TCP-friendly (acceptable) sending rate

$$\text{Rate estimate} = \frac{K.MSS}{RTT\sqrt{p}}$$

- The Client (multimedia player)
  - Estimate p
  - Send feedback to the source at least once per RTT to advertise the value of p

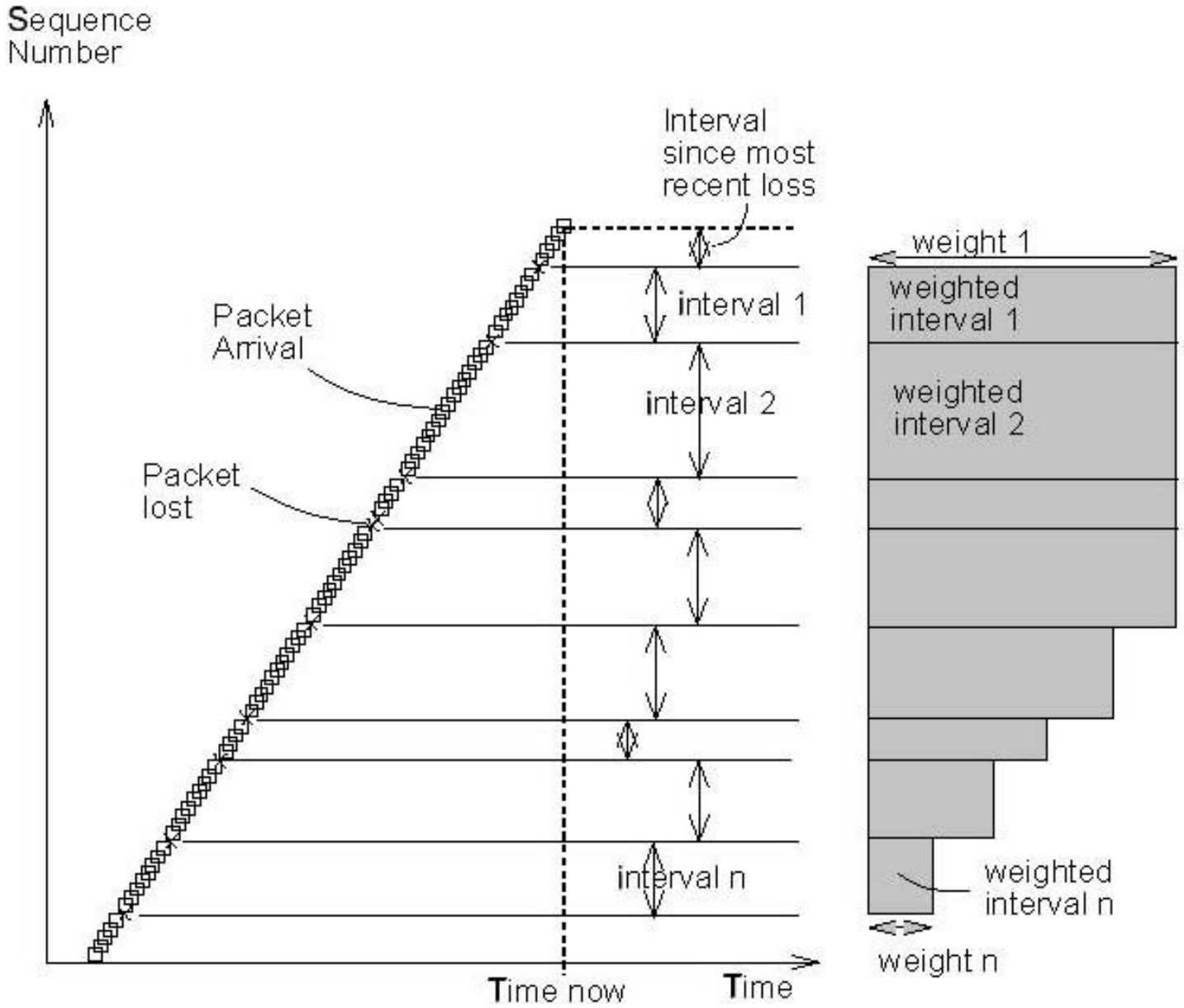
- TFRC accuracy depends on the accuracy of estimating p

- P is the loss event probability not the packet loss probability (i.e., clustered packet losses within the same RTT count as 1)
- The loss event rate increases only in response to a new loss event; it decreases only if the new loss event interval (the interval between two loss events) is larger
- The average loss interval is estimated as a smooth average of n previous intervals with the most recent ones having a larger weight

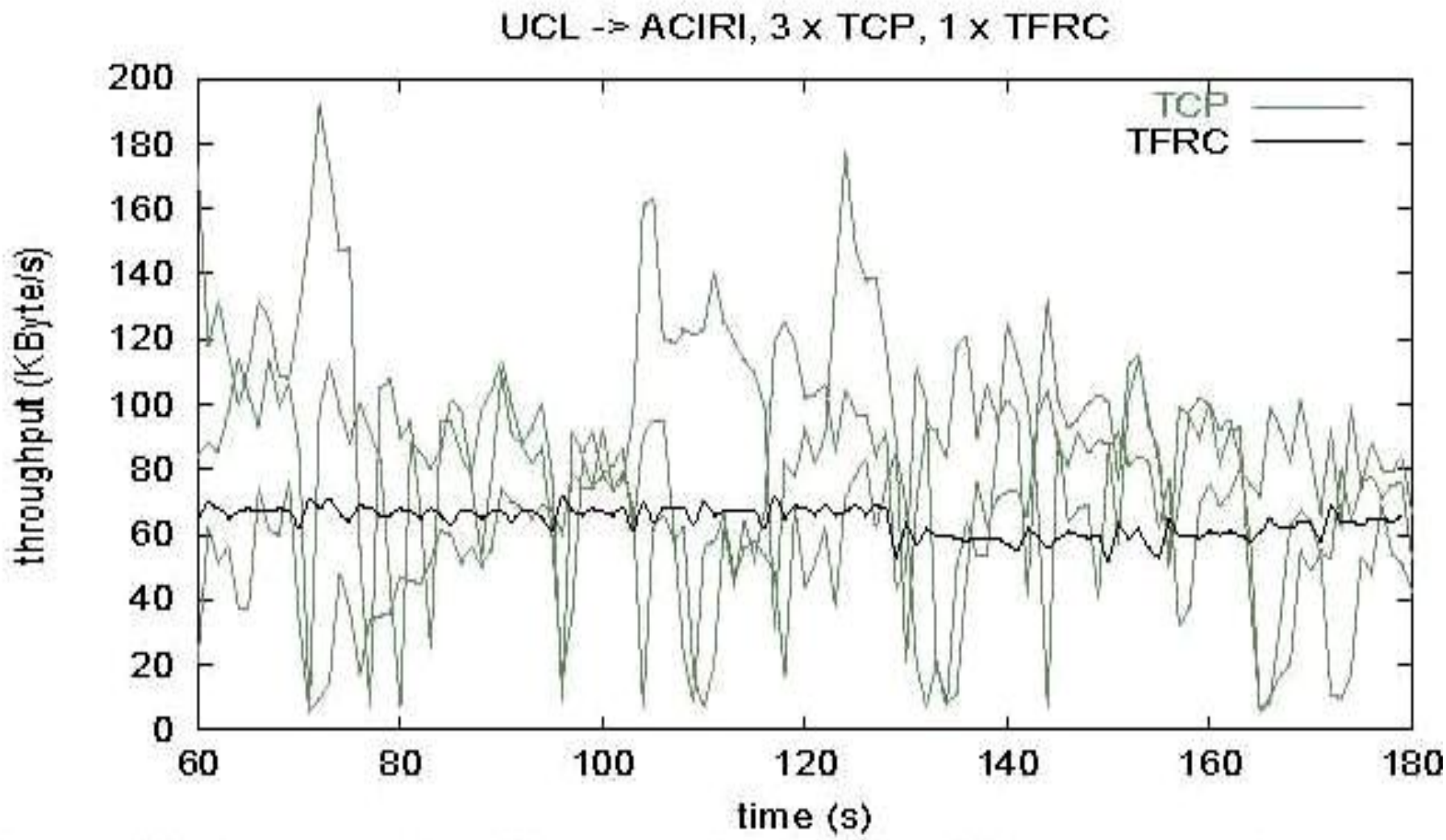
$$\hat{s}_{1,n} = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i}$$

$$w_i = \begin{cases} 1 & \text{for} \quad 1 \le i \le n/2 \\ 1 - \frac{i - n/2}{n/2+1} & \text{for} \quad n/2 \le i \le n \end{cases}$$

TCP Friendly Rate Control



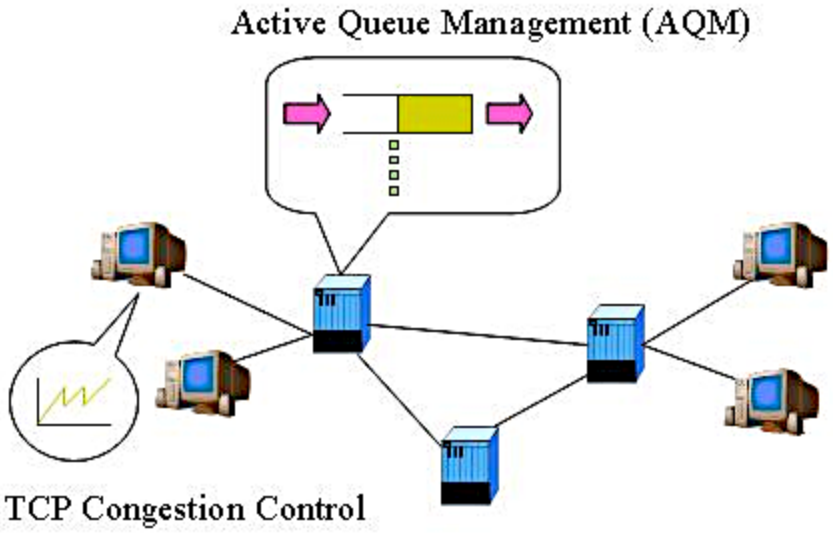
TCP Friendly Rate Control



2. Router Assisted Reactive Congestion Control

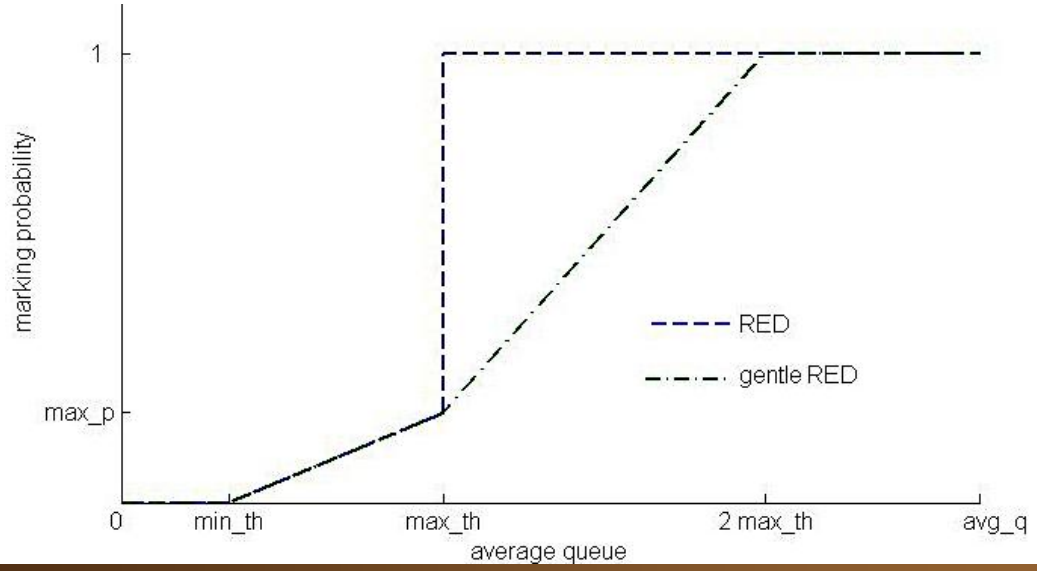
Active queue management

- Basic Idea: Instead of waiting until congestion occurs to reduce the source rate, generate congestion signals selectively to maintain the queue length at a moderate level
- With high-speed networks, TCP losses tend to be **synchronized**
  - All sources decrease rate at the same time and increase at the same time
  - Buffer occupancy is low -> less efficiency



AQM: RED

- Random Early Detection is the first AQM scheme proposed for the Internet (other variation Gentle RED)
- Upon packet arrival,
  - If the average queue length is below a minimum threshold admit the packet to the queue
  - If the average queue length is above a maximum threshold drop the packet
  - If the average queue length is between the two thresholds mark/drop the packet with a probability p

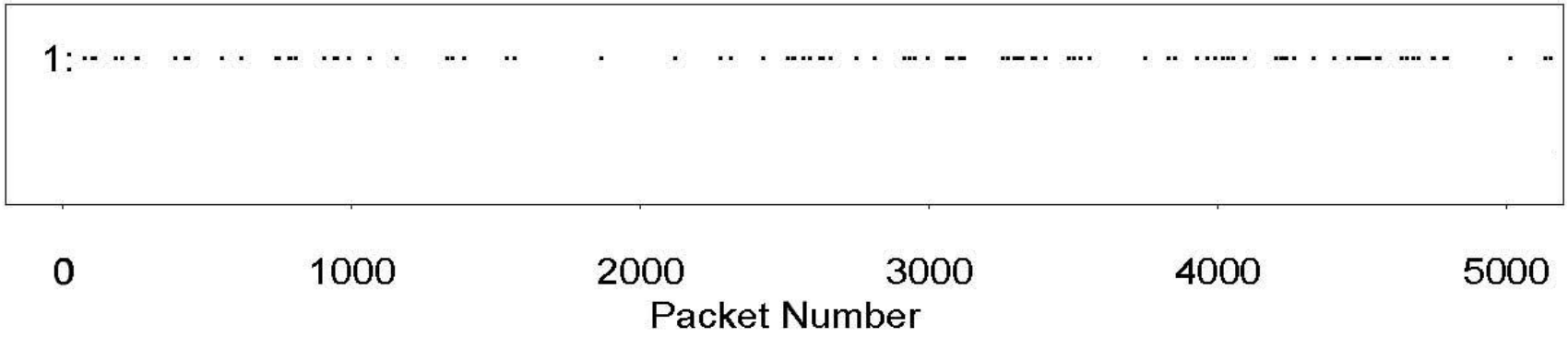


AQM: RED

- One problem is that p is linear between the two thresholds,
- X, the number of packets accepted between two markings, is geometrically distributed:

$$\Pr\{X = n\} = (1-p)^{n-1}p$$

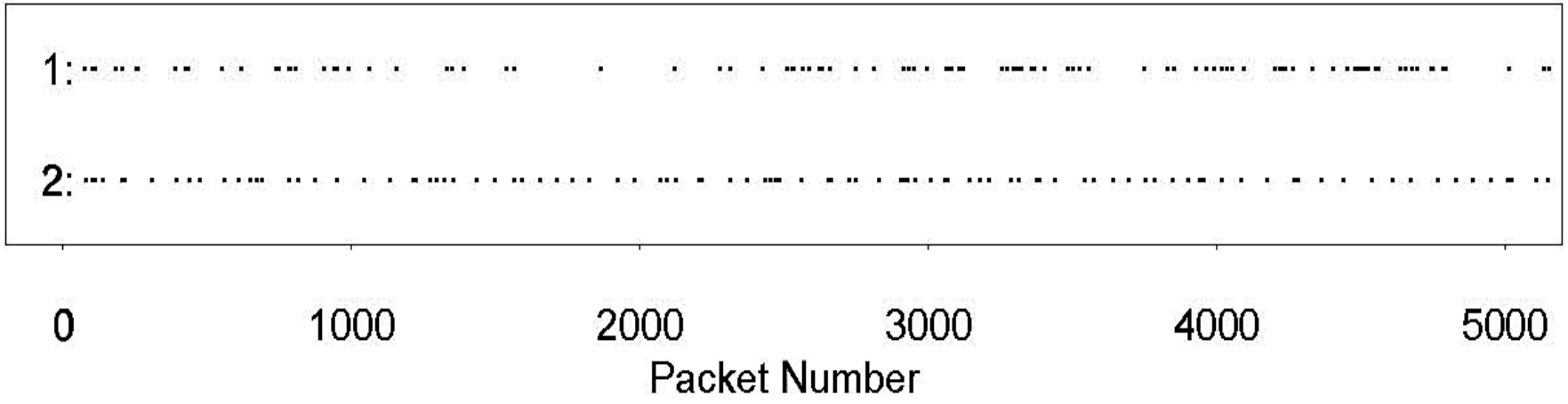
- So the probability that the interval is long is non negligible and the probability that too many marked/dropped packets are clustered is non negligible
- The two events can result in TCP synchronization



- So, we want *X* to be uniform in the set {1,2,3,…, 1/p}
- We can reach this goal by using the following marking/dropping probability

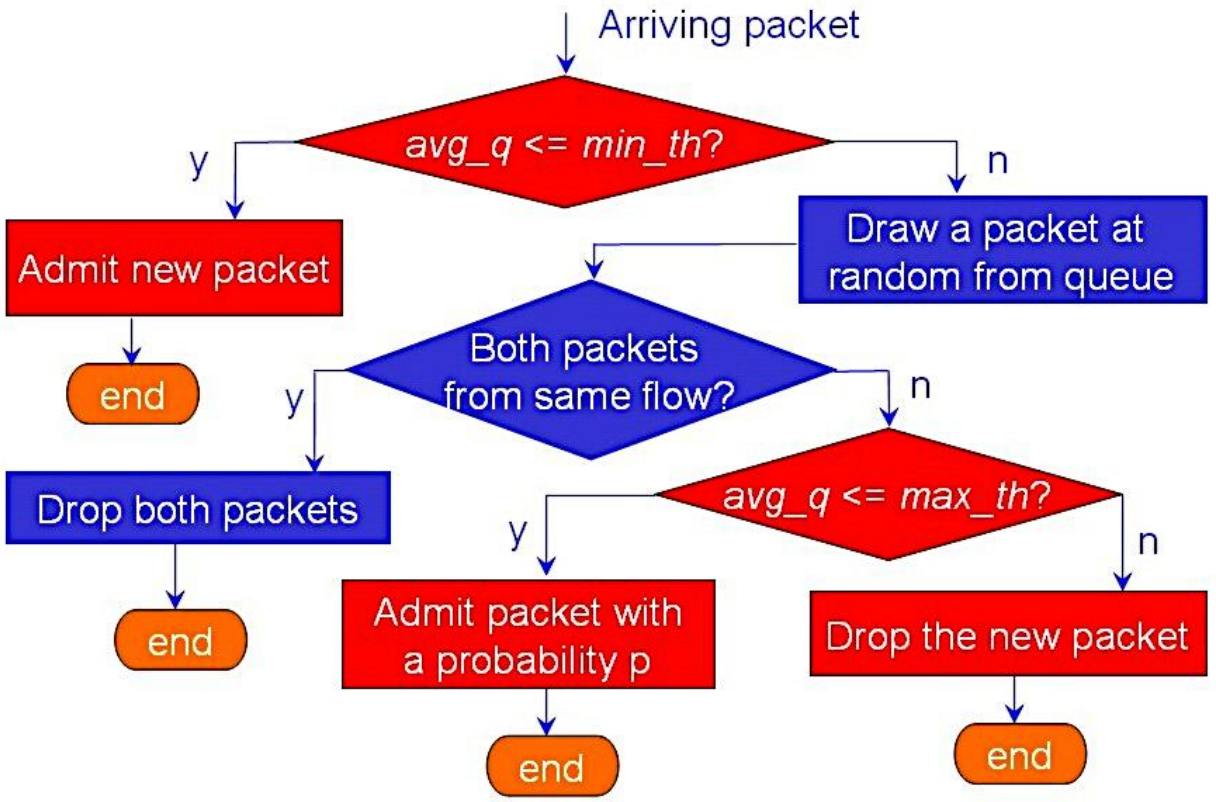
$$p_b = \frac{p}{(1 - count.p)}$$

count is the number of packets admitted since the last drop/marking



AQM: Choke

- ChoKe (Choose and Kill) tries to improve fairness among flows by marking/dropping flows in proportion to their buffer occupancy with a very simple mechanism

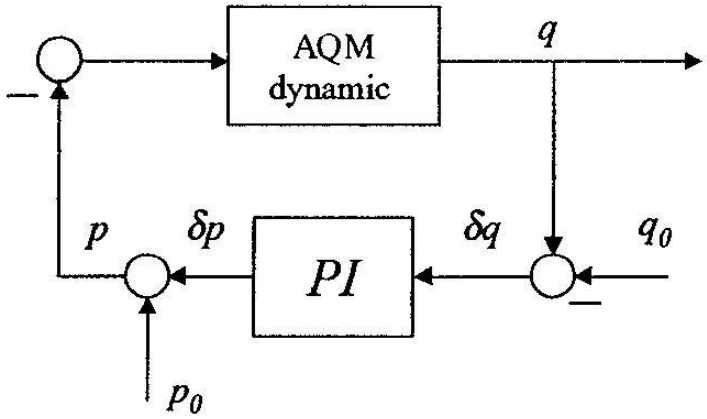


AQM: PI

- Uses a control theoretic approach to AQM
- Tries to convert the difference between the actual queue length and a given target queue length into a marking/dropping probability
- PID (Proportional, Integral, Derivative) controller
  - Proportional controller: calculate the feedback based on the size of the error term by simply multiplying it by a gain factor
  - Integral controller: compensates for the error in the proportional component to eliminate the steady state error - the longer the target queue is not reached the greater is the feedback
  - Derivative controller: adjusts the feedback based on how quickly we are closing on the target queue length

AQM: PI

- The queue length slope (as a function of time) determines the packet marking probability
- q_{ref} is the reference queue length
- T length of the sampling interval



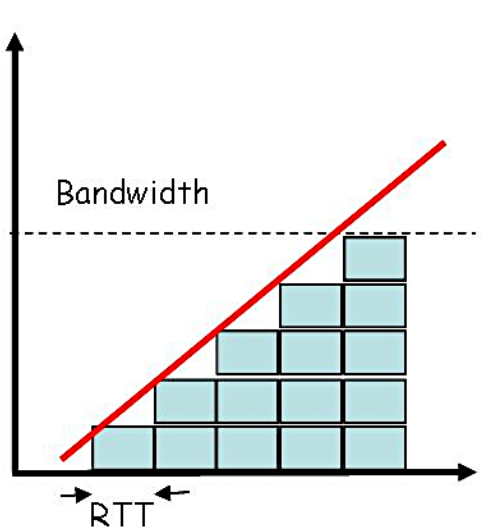
$$p(kT) = a.[q(kT) - q_{ref}] - b[q((k-1)T) - q_{ref}] + p((k-1)T)$$

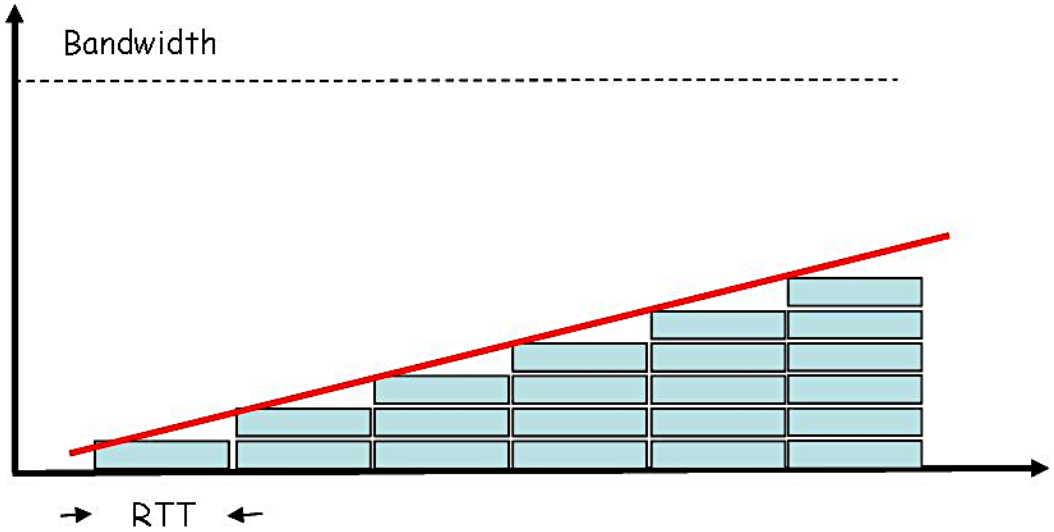
3. Evolution of Congestion Control

Challenges to TCP

- New application requirements and new technologies, create a major challenge to TCP Reno
- In wireless network
  - High bit error rate (BER) in wireless network and interference among wireless nodes lead to some random losses
  - TCP Reno interprets packet losses due to such impairments as congestion indication
- In high bandwidth-delay product networks
  - Additive Increase (AI) is not fast enough to probe the available bandwidth
  - Multiplicative decrease (MD) is too aggressive.
  - Queue occupancy oscillates dramatically at equilibrium leading to unpredictable delays
- Short lived Web traffic flows are usually small but require optimal completion time for users' experience terminate in the slow start phase, thus can not compete for bandwidth with long TCP flows

TCP in High Bandwidth Delay Product networks





New TCP Proposals

- Many exciting proposals exist nowadays in the literature to cope with these problems

- Snoop [Balakrishn, 1995]
- TCP Fast Start [Padmanabhan, 1998]
- HighSpeed TCP (RFC 3649) [Floyd, 2003]
- TCP Westwood [Casetti, 2001]
- XCP [Katabi, 2002]
- TCP Veno [Fu, 2003]
- FAST TCP [Jin, 2004]
- …

TCP Veno

- VeNo (Vegas-Reno) is designed to reduce the drawbacks of TCP Reno in random loss prone networks (wireless networks)
- End-to-end protocol that requires only the modification of the sender
- Underlying assumption: Random losses have no effect on the RTT estimation
  - When queueing delay is small and there is a loss -> random loss
  - When the queueing delay is large and there is a loss-> congestion
- BaseRTT= minimum measured RTT
- ActualRate – cwnd/RTT
- N= ActualRate (RTT – BaseRTT)
- β: congestion estimation threshold

When 3 Dup Acks received

If (N≤ β) / random loss */*

ssthresh = 0.8 cwnd;

cwnd = ssthresh;

else / congestion loss */*

ssthresh = 0.5 cwnd;

cwnd = ssthresh;

EndIF

For each ACK (during congestion avoidance)

If (N≤ β) / random loss */*

cwnd = cwnd + MSS x MSS/cwnd

else / congestion loss */*

cwnd = cwnd + 0.5 MSS x MSS/cwnd

EndIF

High Speed TCP

- Goal: Improve performance of TCP in high bandwidth-delay product networks
- Rationale:
  - Current TCP Reno has a problem in such networks
  - According to the square root formula on a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps requires cwnd of 83,333 segments, and a packet drop rate of at most one congestion event every 5,000,000,000 packets (or equivalently, at most one congestion event every 100 minutes)
- Idea: If window is above a minimum threshold change AI term and MD factor
  - AI -> from 1 in Reno to a(cwnd) an increasing function of cwnd
  - MD -> from 0.5 in Reno to b(cwnd) a decreasing function of cwnd

$$a(cwnd) = cwnd^2 \times p(cwnd) \times 2 \times \frac{b(cwnd)}{2 - b(cwnd)}$$

$$b(cwnd) = (D_{high} - 0.5)\frac{\log(cwnd) - \log(cwnd_{low})}{\log(cwnd_{high}) - \log(cwnd_{low})} + 0.5$$

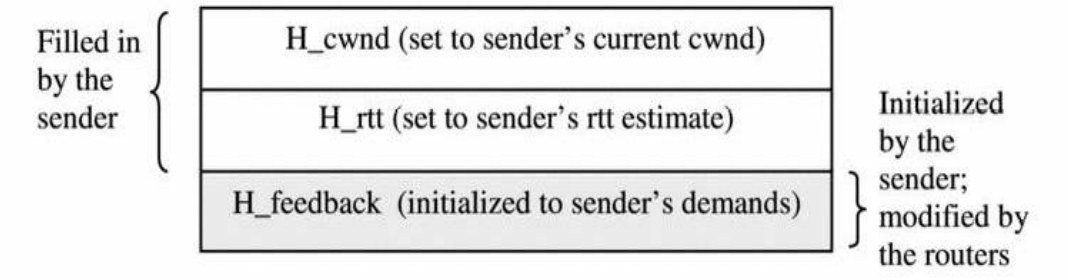
$cwnd_{low}$: Threshold to switch from TCP to HSTCP

$cwnd_{high}$: High window threshold typically 83000

D_{high} : Target MD for a high window threshold 10%

$p(cwnd) = 0.078/cwnd^{1.2}$: HSTCP response function

- XCP (eXplicit Congestion control Protocol) relies on the cooperation of all the routers along the end-to-end path to control congestion
- Uses a rate feedback mechanism from the bottleneck router to the sources instead of packet losses as congestion indication
- Basic Idea:
  - XCP sender estimates and forwards its connection information to the routers (RTT, cwnd) in an option in the TCP header
  - XCP router uses the information from all flows to compute a feasible rate for each connection
  - Bottleneck router sets an explicit feedback in the TCP header
  - The receiver copies the feedback field to the ACK header
  - The sources increase/decrease the window based on the feedback



Congestion controller

- Goal: Match the input traffic load to the link capacity and drain the queue
- Looks only at aggregate traffic and queue
- Uses MIMD

$$\Phi = \alpha \times T \times S - \beta \times Q$$

Φ : Change on aggregate traffic

T: smapling period

S: spare capacity

Q: persistent queue length

Fairness controller

- Goal: divide Φ among the flows fairly
- Looks at the flow information in the congestion header of each packet
- Uses AIMD

- If Φ >0 divide P equally among flows
- If Φ <0 divide P among flows proportionally to their congestion window
- If Φ = 0 redistribute 10% of the traffic among flows every T according to AIMD

We can show that the system converges to a stable optimal allocation for any number of sources, any bandwidth and any delay if:

$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad \text{and} \quad \beta = \alpha^2\sqrt{2}$$