Congestion Control

Outline

- What is scheduling
- Why we need it
- Requirements of a scheduling discipline
- Fundamental choices
- Scheduling best effort connections
- Scheduling guaranteed-service connections
- Packet drop strategies
- Policing

Congestion Control

- Any buffering point has to implement congestion control schemes
  - Maximize or make efficient use of the buffering resources
  - Distribute the buffering resources fairly between the contending connections
  - Prevent connections from affecting the QoS of each other
- Two Approaches
  - Reactive Congestion Control (Passive Queue Management)
    - Make decisions on the onset of congestion
    - e.g., packet dropping
  - Proactive Congestion Control (Active Queue Management)
    - Minimize occurrence of Congestion (Avoidance)
    - Early drop packets
    - Throttle sources by feedback

Congestion Control Elements

- ***Buffer Partitioning***, which defines the amount of buffer space available to a given queue and the ways in which total available buffering resource is shared among a set of queues.
- ***Occupancy Measure***, which defines how the occupancy of the queue is measured. The occupancy measure along with the buffer partitioning defines the *congestion level* of a queue.
- ***Discard Policy***, which determines whether to discard or queue the packets based on the congestion level.

Buffer Partitioning

- **Complete Partitioning**
  - The available buffering resource is divided among the queues such that each queue gets its own buffer space
  - Since each queue gets its own buffer space, controlling the QoS for a queue becomes easy.
  - It is not efficient on buffer usage. If a queue does not use its allocated buffer space, it cannot be used by any other queue, which needs i

$$B_1 + B_2 + \cdots + B_N = B$$

$$B_j \cap B_k = \Phi \quad \forall (j,k) \in \{1,2,\cdots,N\} \text{ and } j \neq k$$

$$q_1 + q_2 + \cdots + q_N \leq B$$

B_i is the max allocation and q_i is the current occupancy

Buffer Partitioning (contd ..)

- **Complete Sharing**
  - The available buffering resource is fully shared among the queues such that no queue gets its own buffer space
  - Since each queue does not get its own buffer space, controlling the QoS for a queue becomes difficult.
  - The QoS of each queue is affected by the traffic in other queues.
  - This method is very efficient on buffer usage since any queue has access to the complete available buffer resource.

$$B_1 = B_2 = \cdots = B_N = B$$

$$B_j \cap B_k = B \quad \forall (j,k) \in \{1,2,\cdots,N\} \text{ and } j \neq k$$

$$q_1 + q_2 + \cdots + q_N \leq B$$

Buffer Partitioning (contd ..)

- **Sharing with Minimum Allocation**
  - Allocates a minimum buffer space to each queue at the congestion point.
  - Remaining buffering resource is completely shared among the connections, similar to complete buffer sharing policy.
  - Suffers from an isolation problem since the shared buffer space can be unfairly utilized by the connections.
  - However, since this scheme allocates a minimum buffering space to each connection, the unfairness may not be as severe as in complete sharing policy.
  - Let *Li* is the minimum guaranteed buffer space for each queue

$$B_1 = B_2 = \cdots = B_N = \left(B - \sum_{i=1}^{N} L_i\right)$$

$$B_j \cap B_k = \left(B - \sum_{i=1}^{N} L_i\right) \quad \forall (j,k) \in \{1,2,\cdots,N\} \text{ and } j \neq k$$

$$q_1 + q_2 + \cdots + q_N \leq B$$

Buffer Partitioning (contd ..)

- **Sharing with Maximum Queue Length**
  - Allows complete sharing of the buffer space between the queues at a congestion point.
  - Each queue is limited by a maximum buffer size.
  - Limiting the maximum queue length, a connection cannot take more than this share from the common pool of memory.
  - Thus, this scheme protects from unfair use of buffer space by connections

$$B_1 = L_1; B_2 = L_2; \cdots; B_N = L_N$$

$$B_1 + B_2 + \cdots + B_N \geq B$$

$$B_j \cap B_k \neq \Phi \quad \forall (j,k) \in \{1,2,\cdots,N\} \text{ and } j \neq k$$

$$q_1 + q_2 + \cdots + q_N \leq B$$

- Sharing with a Maximum Queue Length and Minimum Allocation

Occupancy Measure

- The occupancy measure is a metric that provides an instantaneous or filtered estimation of the utilization of the buffer pool for each queue
- The occupancy measures can include one or more of the following:
  - Instantaneous queue Size
    - Current queue Size
  - Average Queue Size
    - Periodic samples of the queue averaged
  - Queue Growth
    - Queue growth is an alternative occupancy measure, which measures the rate of increase and decrease of the queue size
    - Regular periodic estimation of this measure is necessary and the length of the measurement interval (defined as a monitoring period) is the key in providing a stable and meaningful measure

Packet Discard Policy (Dropping)

- Packets that cannot be served immediately are buffered
- Full buffers => *packet drop strategy (Discard Policy)*
- Packet losses happen almost always from best-effort connections (why?)
- Shouldn't drop packets unless imperative
  - ◆ packet drop wastes resources (why?)
- Packet Dropping on the onset of congestion is ***Reactive Congestion Control*** – Passive Queue Management Technique
- Early Packet Dropping (without an onset of congestion) is ***Proactive Congestion Control*** – Active Queue Management Technique

Classification of drop strategies

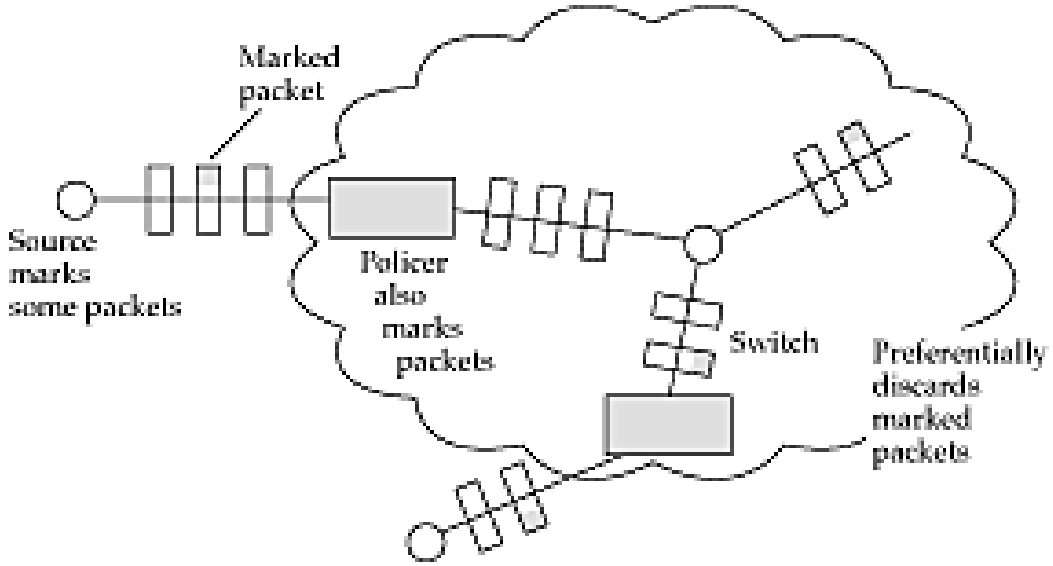
1. Degree of aggregation
2. Drop priorities
3. Early or late
4. Drop position

1. Degree of aggregation

- Degree of discrimination in selecting a packet to drop
- E.g. in vanilla FIFO, all packets are in the same class
- Instead, can classify packets and drop packets selectively based on class
- The finer the classification the better the protection
- Max-min fair allocation of buffers to classes
  - drop packet from class with the longest queue (why?)
  - Fair share buffer allocation
    - Allocate buffers based on the weights
    - Drop packets based on this fair-share buffer allocation

2. Drop priorities

- Drop lower-priority packets first (Service Class based discard)
  - e.g, AF11 (Green), AF12 (Yellow), AF13 (Red)
- How to choose?
  - endpoint marks packets
  - regulator marks packets
  - congestion loss priority (CLP) bit in packet header



ATM CLP bit: pros and cons

- Pros
  - if network has spare capacity, all traffic is carried
  - during congestion, load is automatically shed
- Cons
  - separating priorities within a single connection is hard
    - Is CBR with CLP=1 higher priority than VBR with CLP=0?
  - what prevents all packets being marked as high priority?
    - Policing

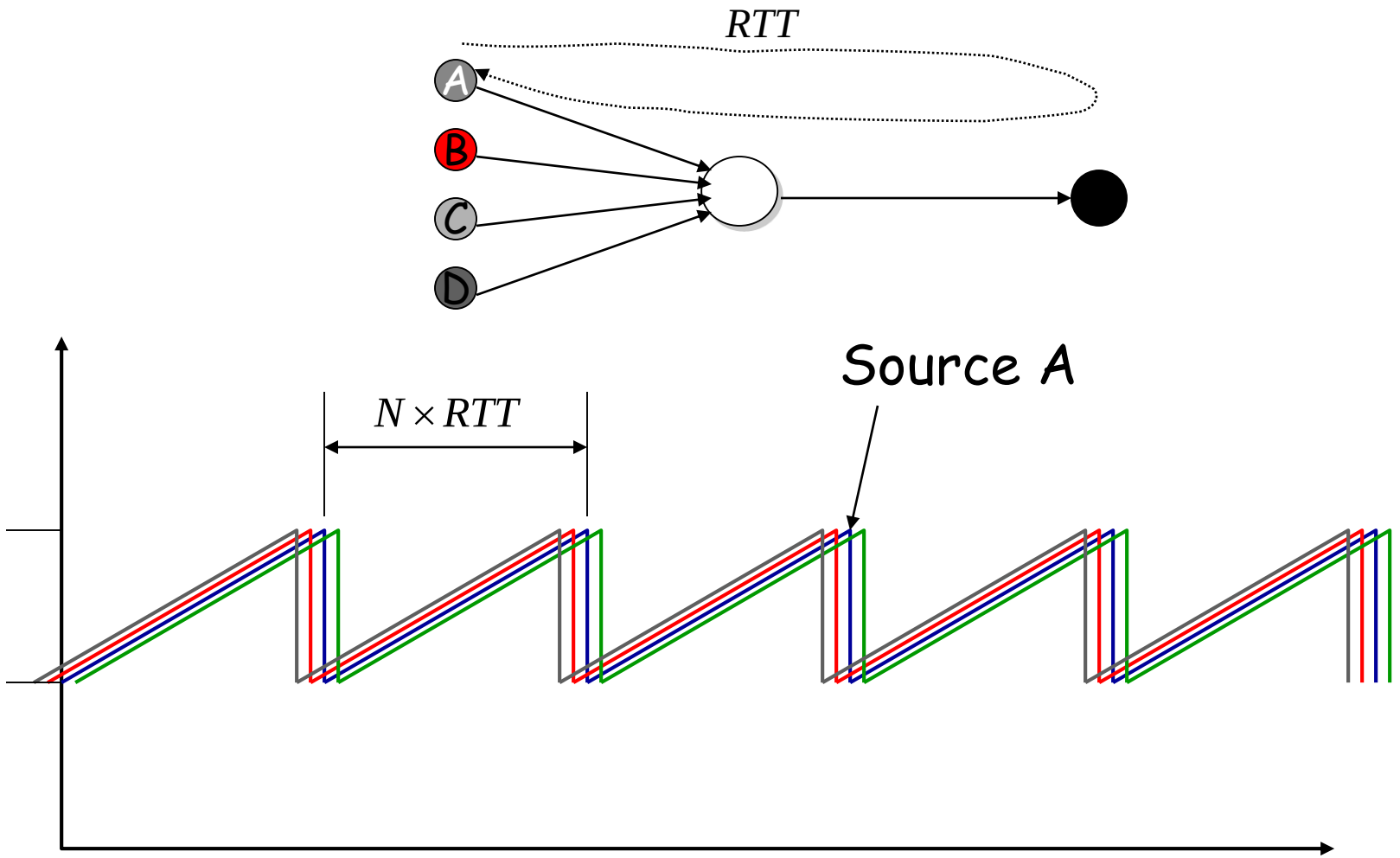
2. Drop priority (contd.)

- Special case of AAL5 (ATM Networks)
  - AAL5 is a framing strategy
  - want to drop an entire frame, not individual cells
    - ☞ Partial Packet Discard
  - cells belonging to the selected frame are preferentially dropped
- Drop packets from ‘nearby’ hosts first
  - because they have used the least network resources
  - can’t do it on Internet because hop count (TTL) decreases

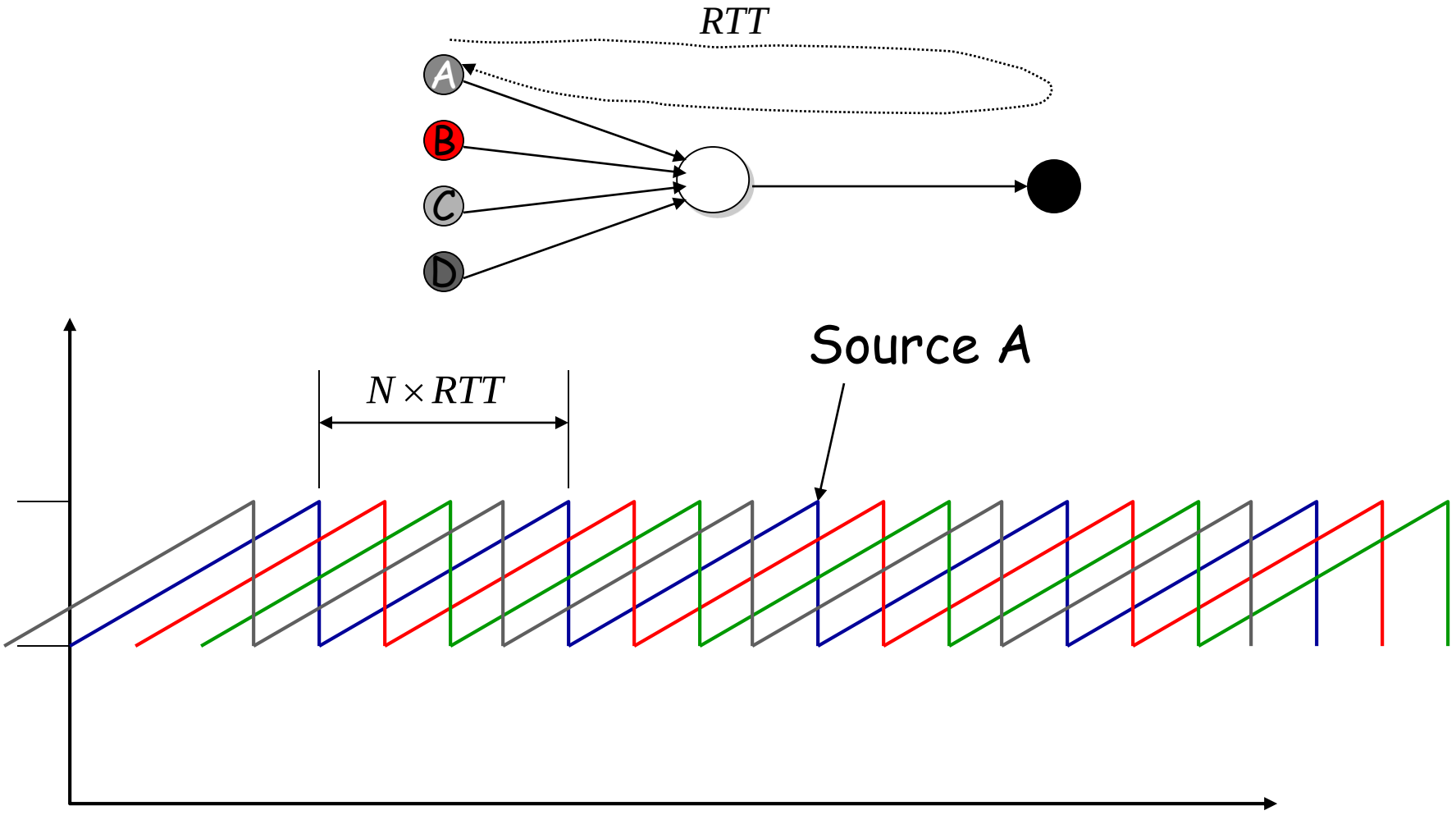
3. Early vs. late drop

- Early drop => drop even if space is available
  - signals endpoints to reduce rate
  - cooperative sources get lower overall delays, uncooperative sources get severe packet loss
- Early random drop
  - drop arriving packet with fixed drop probability if queue length exceeds threshold
  - intuition: misbehaving sources more likely to send packets and see packet losses
- Late Drop (Drop Tail)
  - when the output buffer is full, simply drop an arriving packet
  - TCP sources send packets in bursts (due to ack compression). So, drops occur in bursts, causing TCP sources to Slow-Start
  - Synchronization between various sources could occur: they end up using bandwidth in periodic phases, leading to in-effeciences

Synchronization of sources



Desynchronized sources



3. Early vs. late drop: RED

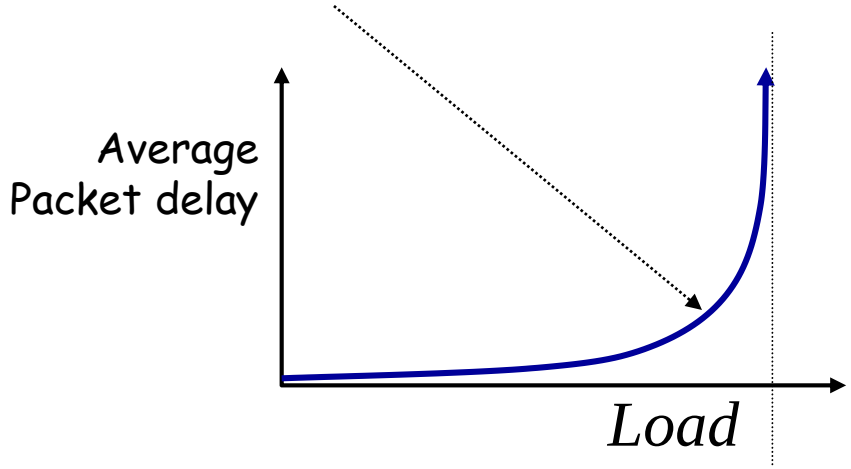
- Random early detection (RED) makes three improvements
- Metric is moving average of queue lengths and not current queue size
  - small bursts pass through unharmed
  - only affects sustained overloads
- Packet drop probability is a function of mean queue length
  - prevents severe reaction to mild overload
- Can mark packets instead of dropping them
  - allows sources to detect network state without losses
- RED improves performance of a network of cooperating TCP sources
- No bias against bursty sources
- Controls queue length regardless of endpoint cooperation
  - drops incoming packets randomly based on congestion level
  - this signals the onset of congestion to the sources who will back-off (if they are responsive)
  - thus, RED avoided bursty dropping, synchronization, etc

4. Drop position

- Can drop a packet from head, tail, or random position in the queue
- Tail
  - easy
  - default approach
- Head
  - harder
- Random
  - hardest
  - unlikely to make it to real routers
- Drop entire longest queue
  - easy
  - almost as effective as drop tail from longest queue

Congestion Avoidance

- TCP reacts to congestion ***after*** it takes place. The data rate changes rapidly and the system is barely stable (or is even unstable).
- Can we *predict* when congestion is about to happen and avoid it? E.g. by detecting the knee of the curve.

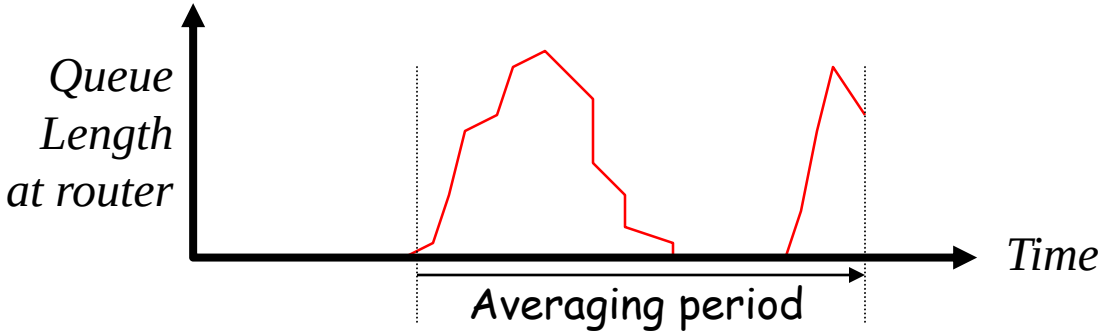


Congestion Avoidance Schemes

- Router-based Congestion Avoidance:
  - DECbit:
    - Routers explicitly notify sources about congestion.
  - Random Early Detection (RED):
    - Routers implicitly notify sources by dropping packets.
    - RED drops packets at random, and as a function of the level of congestion.
- Host-based Congestion Avoidance
  - Source monitors changes in RTT to detect onset of congestion.
  - Variety of algorithms possible

DECbit

- Each packet has a “Congestion Notification” bit called the DECbit in its header.
- If any router on the path is congested, it sets the DECbit.
  - Set if average queue length >= 1 packet, averaged since the start of the previous busy cycle.
- To notify the source, the destination copies DECbit into ACK packets.
- Source adjusts rate to avoid congestion.
  - Counts fraction of DECbits set in each window.
  - If <50% set, increase rate additively.
  - If >=50% set, decrease rate multiplicatively.



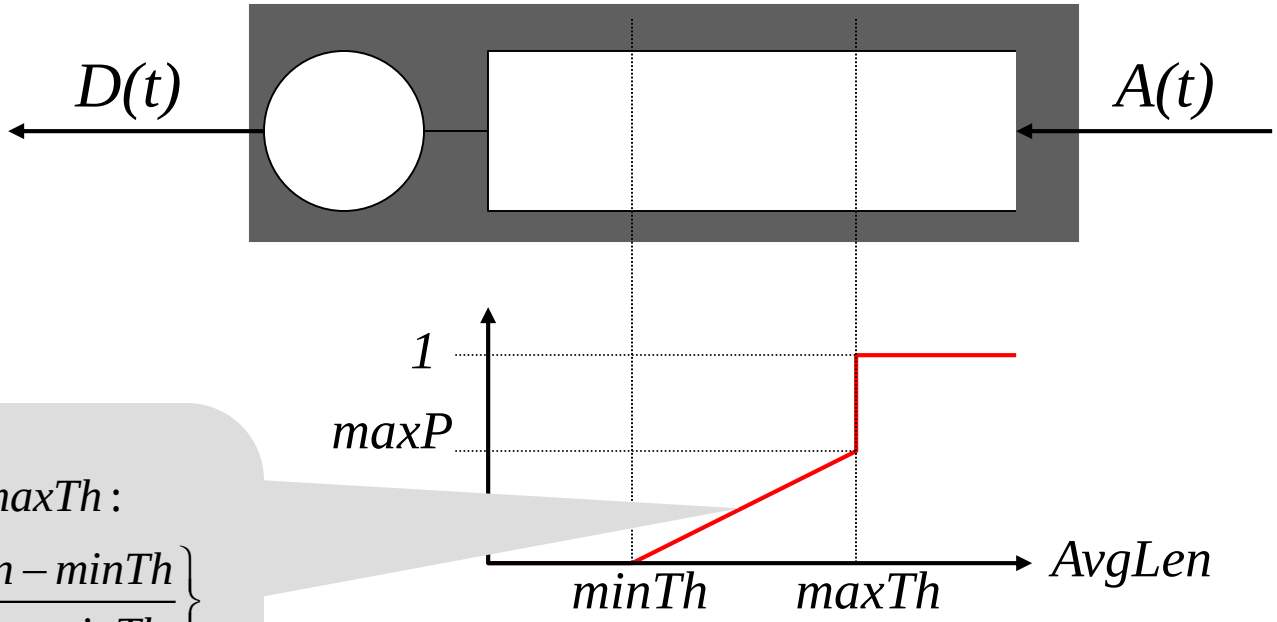
Random Early Detection (RED)

- RED is based on DECbit, and was designed to work well with TCP.
- RED **implicitly** notifies sender by dropping packets.
- Drop probability is increased as the *average* queue length increases.
- (Geometric) moving average of the queue length is used so as to detect long term congestion, yet allow short term bursts to arrive.

$$AvgLen_{n+1} = (1-\alpha) \times AvgLen_n + \alpha \times Length_n$$

i.e. $$AvgLen_{n+1} = \sum_{i=1}^{n} Length_i (\alpha)(1-\alpha)^{n-i}$$

RED Drop Probabilities



If $minTh < AvgLen < maxTh$:

$$\hat{p}_{AvgLen} = maxP\left\{\frac{AvgLen - minTh}{maxTh - minTh}\right\}$$

$$\Pr(\text{Drop Packet}) = \frac{\hat{p}_{AvgLen}}{1 - count \times \hat{p}_{AvgLen}}$$

count counts how long we've been in $minTh < AvgLen < maxTh$ since we last dropped a packet. i.e. drops are spaced out in time, reducing likelihood of re-entering slow-start.

Properties of RED

- Drops packets before queue is full, in the hope of reducing the rates of some flows.
- Drops packet for each flow *roughly* in proportion to its rate.
- Drops are spaced out in time to avoid synchronization
- Because it uses average queue length, RED is tolerant of bursts.
- Random drops hopefully desynchronize TCP sources.
- WRED (Weighted RED) preferentially drops packets for classes!!

TCP Congestion Control

- TCP implements host-based, feedback-based, window-based congestion control.
- TCP sources attempts to determine how much capacity is available
- TCP sends packets, then reacts to observable events (loss).

TCP Congestion Control (contd ..)

- TCP sources change the sending rate by modifying the window size:

  Window = min{Advertized window, Congestion Window}

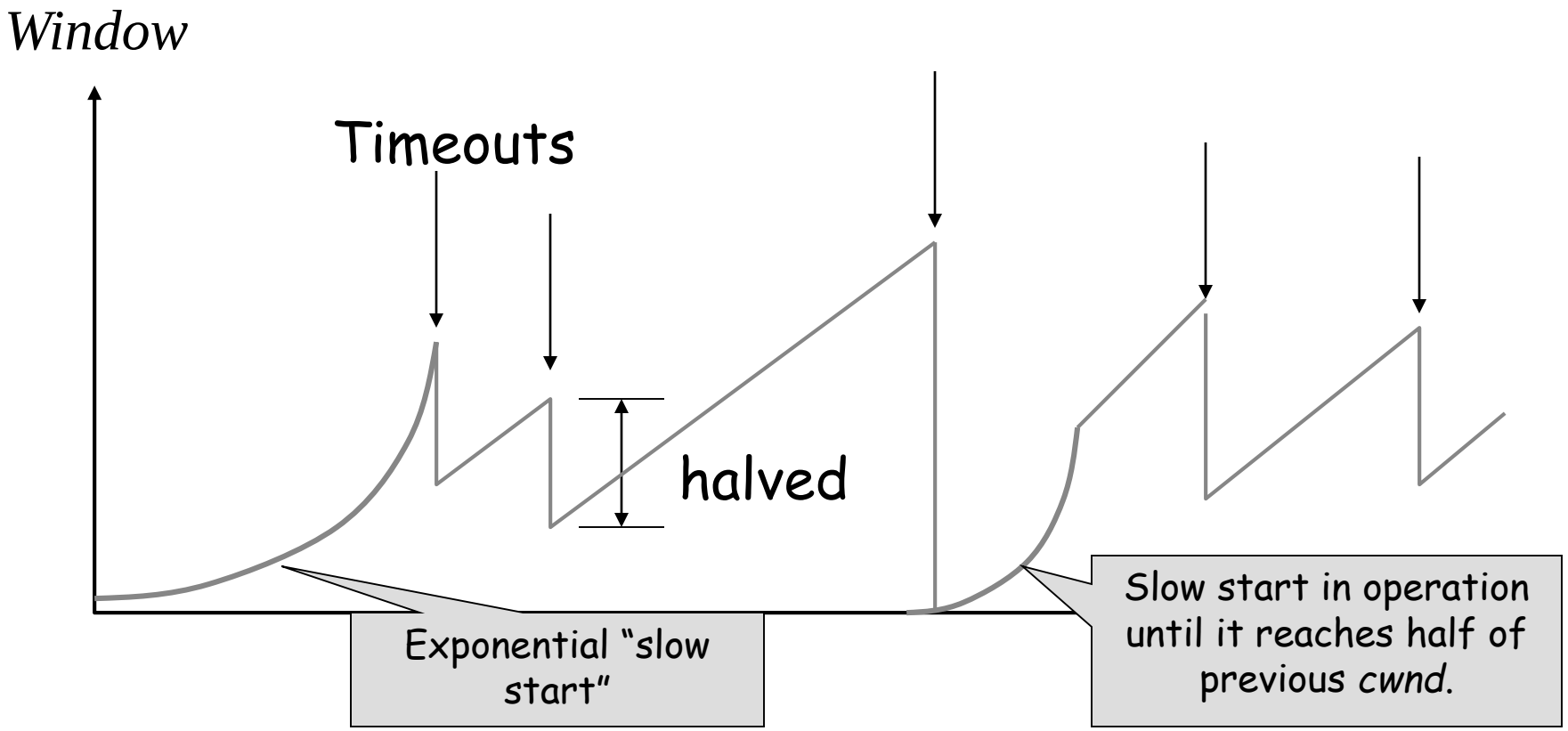
  Receiver — Transmitter ("cwnd")

- In other words, send at the rate of the slowest component: *network or receiver*.
- "cwnd" follows additive *increase/multiplicative decrease*
  - On receipt of Ack: cwnd += 1
  - On packet loss (timeout): cwnd *= 0.5

TCP details

- Window starts at 1
- Increases exponentially for a while, then linearly
- Exponentially => doubles every RTT
- Linearly => increases by 1 every RTT
- During exponential phase, every *ack* results in window increase by 1
- During linear phase, window increases by 1 when # acks = window size
- Exponential phase is called *slow start*
- Linear phase is called *congestion avoidance*

Slow Start



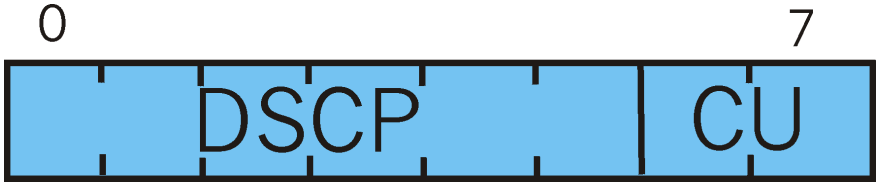
Why is it called slow-start? Because TCP originally had no congestion control mechanism. The source would just start by sending a whole window's worth of data.

TCP Sending Rate

- What is the sending rate of TCP?
- Acknowledgement for sent packet is received after one RTT
- Amount of data sent until ACK is received is the current window size W
- For TCP over a network link with enough buffer, RTT and W are proportional to each other
- Therefore the sending rate R = W/RTT is constant
- TCP rate can be controlled in two ways:
  1. Buffering packets and increasing the RTT
  2. Dropping packets to decrease TCP's window size

Explicit Congestion Notification (ECN)

- RFC2481
- Adds two bits to indicate network congestion, the two unused bits on ToS byte
- Bits 6 and 7 are designated as the ECN field.
- Bit 6 is designated as the ECT bit, and bit 7 is designated as the CE bit



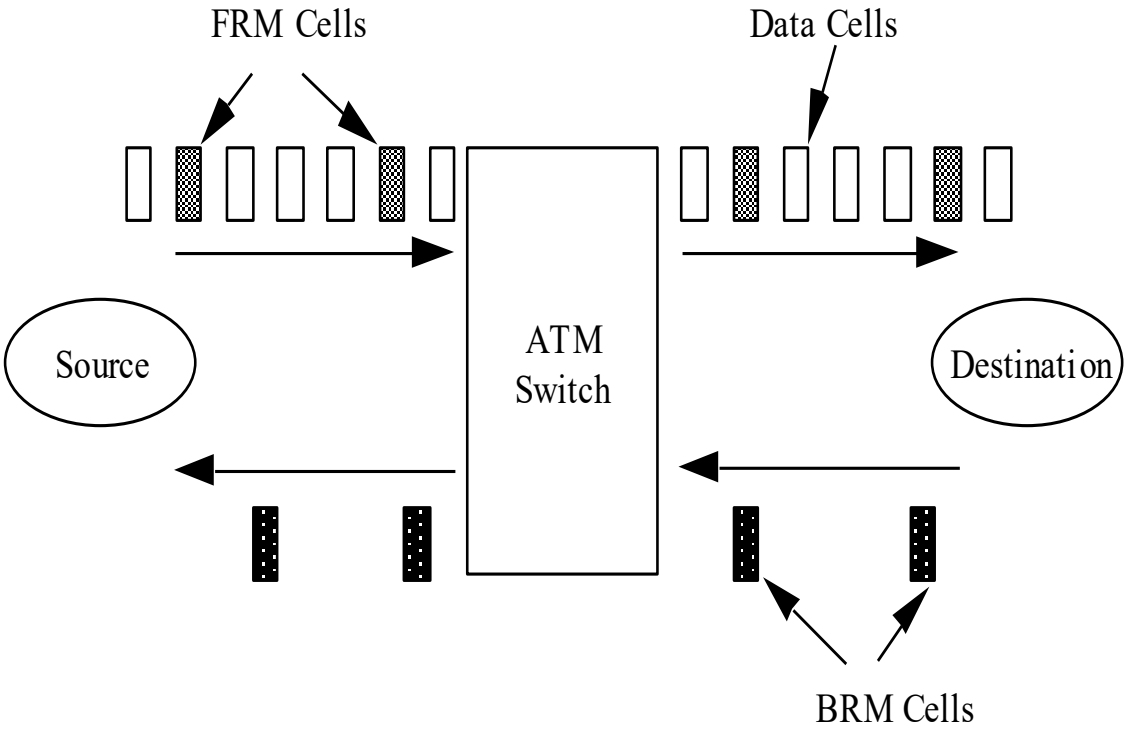
- ECT bit – ECN Capable Transport is set by the end points (Sources and Receivers must be capable) to indicate that they are ECN capable
- The CE (Congestion Experienced) bit would be set by the router to indicate congestion to the end nodes.
- Perform RED and mark the CE bit instead of dropping. Routers that have a packet arriving at a full queue would drop the packet, just as they do now

Problems still remain!!

- RED has been shown to be difficult to tune.
- RED can be unfair to heterogeneous flows (e.g., mix of TCP and UDP traffic)
- Researchers believe ECN is better after a few RED versus ECN comparison studies.
- The differences between RED and ECN behavior is not well understood.
- Is ECN also unfair to heterogeneous flows?
- What happens when there are many flows?
- Can ECN be adapted to perform better?
- Instead of fixing the thresholds, can they be varied dynamically for better tuning??

ATM Available Bit Rate (ABR) Service

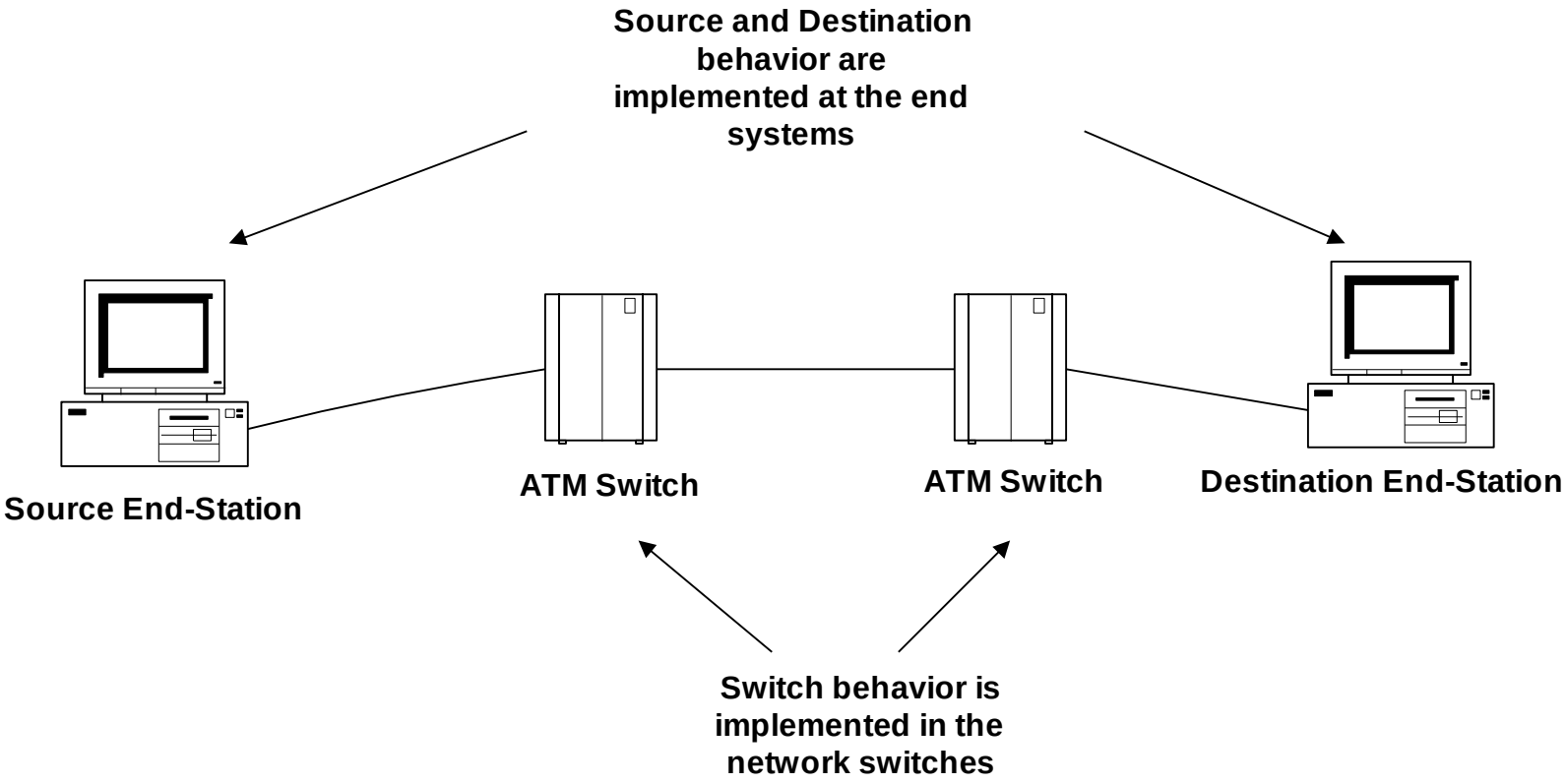
- Based on **Explicit Notification** of Congestion (Binary Mode) or Rates (Explicit Rate mode)
- Resource Management (RM) Cells are used to indicate congestion and Rate



ABR – Basic Concepts

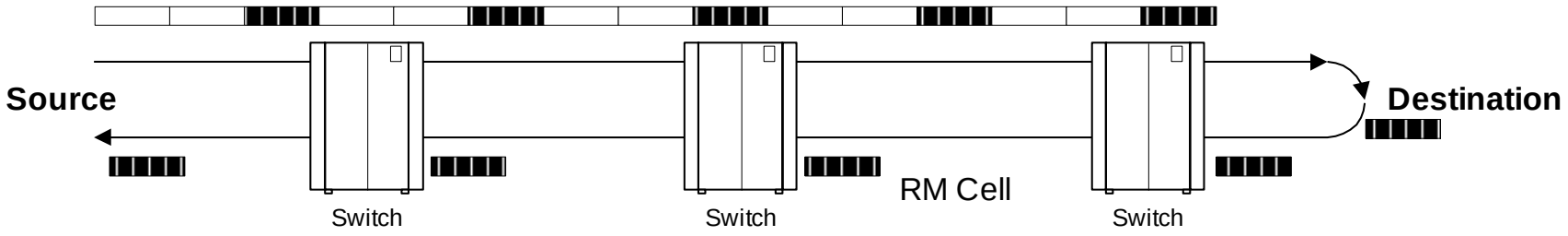
- The source end-station follows a set of rules called the "***Source Behavior"*** which determine the sending rate based on network feedback
- Each network element follows a set of rules called the "***Switch Behavior"*** which define how the feedback may be provided to control the source rate
- The destination end-station follows a set of rules called the "***Destination Behavior"*** which combine two functions: 1) reflect back the network congestion indication to the source, 2) provide destination's own feedback to the source

ABR – Concepts (contd ..)

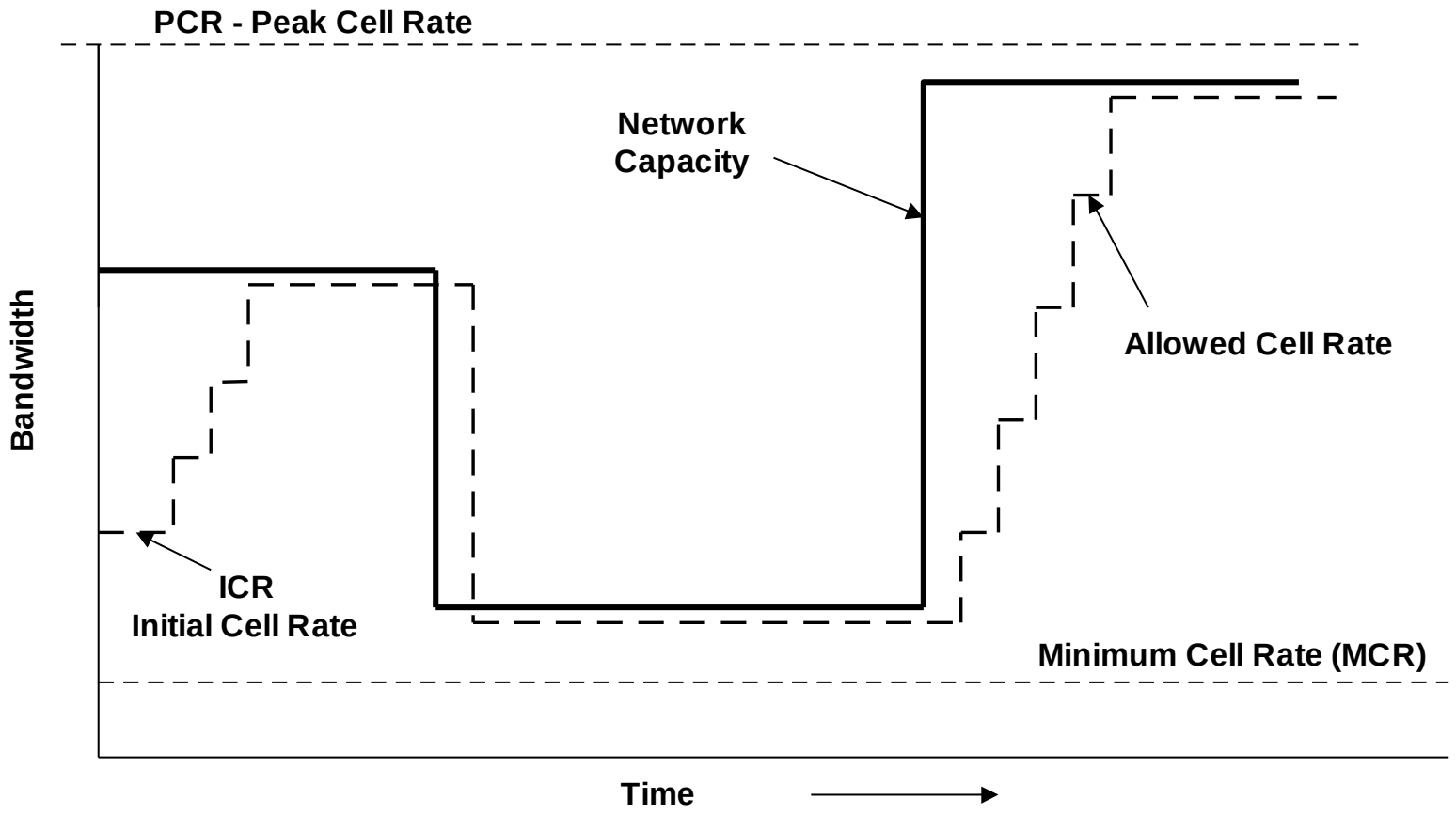


ABR – Basic Concepts (contd ..)

- ABR sources insert Resource Management cells periodically in the data stream
- The RM cells are turned around by the destination
- Each RM cell contains an Explicit Rate (ER) at which the source wishes to operate. This rate may be reduced by the network elements to the rate they can support. RM cells also contain binary fields which are used by non-ER switches (EFCI Binary Switches)



ABR – ER (Explicit Rate) Operation

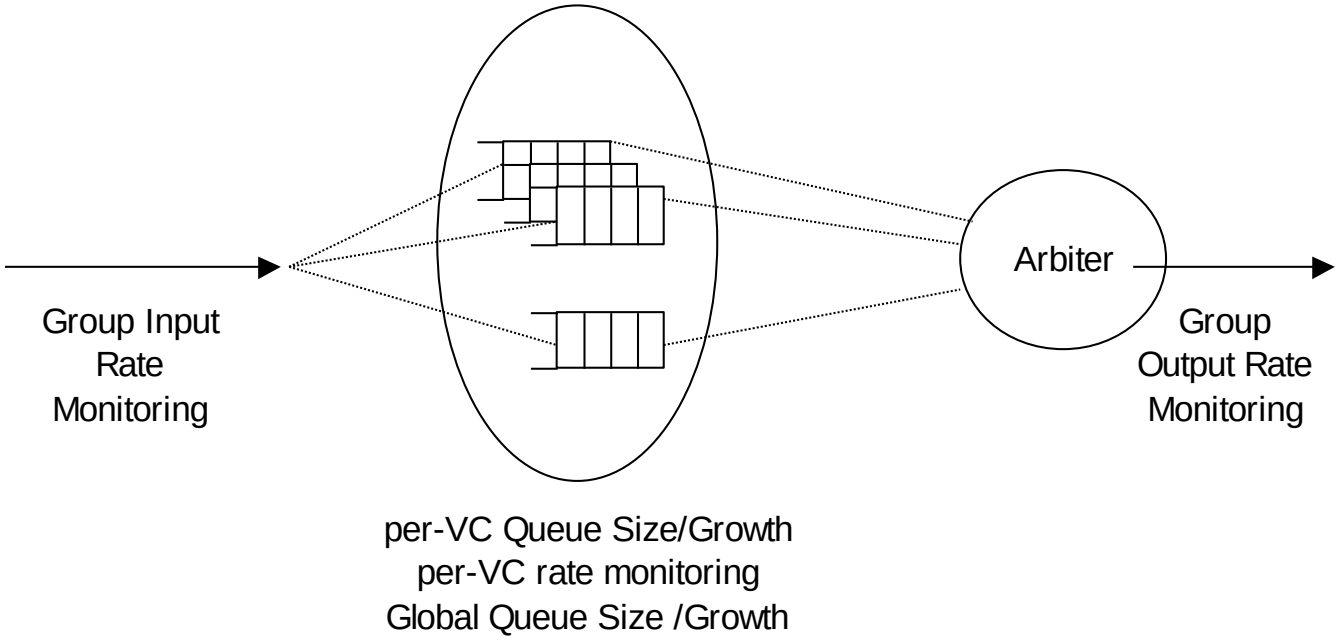


ABR – ER Operation

- Switches should compute the ER at each queuing point for each connection (VC)
- Switches will estimate the available bandwidth for ABR traffic periodically
- This bandwidth is then divided to all the ABR connections going through that queuing point according to *a fairness criteria*
- Possible fairness schemes: *Equal Share, MCR Proportional, Weight Proportional, MCR+Weight Proportional etc*

ER Block

- **Rate Control through Rate monitoring**
- **Rate Control through Congestion Monitoring**



Rate Control By Rate Monitoring

- Periodic estimation of Output bandwidth
- Aggregate arrival rate (group) measurement
- No built-in queue congestion
- Achieves Max-Min fairness

$$BW_k = \left(\frac{BW_{ABR}}{AR} \right) \times BW_{k-1}$$

Example

- Let BW_{ABR}=100Mbps; N = 2 connections; Equal-Share
- Say one connection is bottlenecked at 25Mbps

	AR	BW_k	Share	C1	C2
k=0	-	100	50	50	25
k=1	75.00	133.33	66.67	66.67	25
k=2	91.67	145.44	72.72	72.72	25
k=3	97.72	148.83	74.41	74.41	25
k=4	99.41	149.71	74.85	74.85	25

Rate Control By Congestion Monitoring

$$BW_k = f(Q) \times BW_{k-1}$$

- Rate is controlled implicitly through queue depth (congestion indicator) monitoring
- Increase in Queue length means Arrival Rate > Available Bandwidth. Reduce the rate!!
- Various functions can be implemented: step, linear etc.
- The general form of *f(Q)* is:

$$f(Q) = \begin{cases} > 1 \text{ if } Q < \text{Threshold} \\ < 1 \text{ if } Q > \text{Threshold} \end{cases}$$

Some Bits in RM Cells

- *CI (Congestion Indication):* allows a network element to indicate that there is congestion in the network. When a source receives a backward RM-cell with CI=1, it decreases its ACR (Allowed Cell Rate). When turning around a forward RM-cell, a destination will set CI=1 to indicate that the previous received data cell had the EFCI state set.
- *No Increase (NI):* prevents a source from increasing its ACR
- *Explicit Rate (ER):* Limits the source ACR. For each RM-cell ER is set by the source to a requested rate (such as PCR). It may be subsequently reduced by any network element in the path to a value that the element can sustain
- *Current Cell Rate (CCR):* Set by the source to its current ACR. It may be useful to network elements in computing a value to place in ER.
- *Minimum Cell Rate (MCR):* Carries the connection's Minimum Cell Rate. It may be useful to network elements in allocating bandwidth among connections

Explicit Rate Calculations

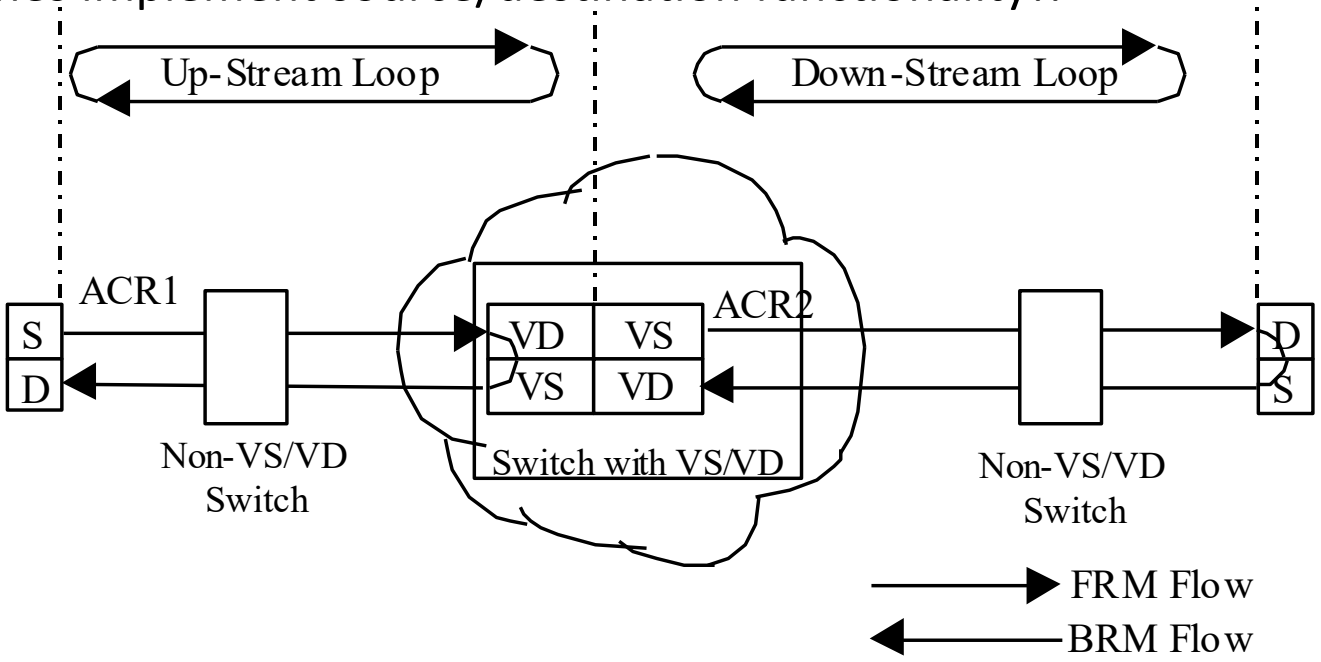
- Each switch along the path computes a fair-share of bandwidth for each connection (e.g., max-min)
- Source sets the Explicit Rate in the RM cells to be Peak Cell Rate (desired transmission rate by the source)
- Each switch in the forward path, based on the fair-share calculation may reduce this desired rate of PCR
  - If switch allocation is lower than what's currently marked in the ER, the ER value is replaced by the lower value
  - Each node along the path performs a similar operation
- The RM cells are turned back by the destination and sent back to the source
- Source gets the latest, updated value from network
  - Adjusts rate. Decrease must be immediate. Increase is additive.

ER Algorithms

- Many algorithms were developed that are evaluated against
  - Fairness (bandwidth allocation)
  - Efficiency (network resource maximization)
  - Stability (convergence to steady state)
  - Robustness (under variety of traffic scenarios)

Virtual Source/Virtual Destination (VS/VD)

- By incorporating the Virtual Source (VS) or Virtual Destination (VD) behavior into the intermediate switching elements, the end-to-end ABR flow control loop may be segmented into several smaller loops thereby reducing feedback delays
- Switches implement source/destination functionality!!



Future Work on Congestion Control??

- One area of future direction for the congestion control can be based on “prediction”.
- The *Prediction based policies* try to predict whether the congestion is about to occur, or not.
- The congestion function reacts to the congestion level based on this prediction in preventing services or connections from loosing their QoS
  - Dynamically changing the buffer allocation or threshold
  - Dynamic adjusting of service rate (e.g., changing the weights of a weighted-fair scheduling)
  - Dynamically changing the service priority of the queues.