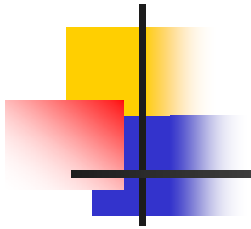
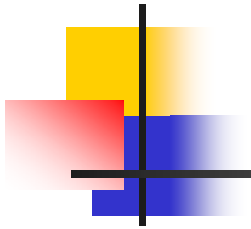


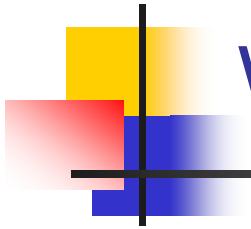
Outline

- What is scheduling, why we need it?
- Requirements of a scheduling discipline
- Fundamental choices
- Scheduling disciplines
- Buffer management and packet drop strategies



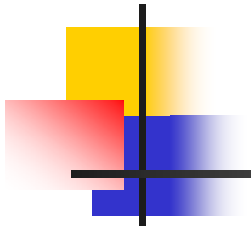
Buffer Management

- Packets that cannot be served immediately are buffered
- How should the buffer be shared among flows/connections?
  - **When buffers is full, a *packet drop strategy is needed***
- Packet losses happen almost always from best-effort connections (why?)
- Shouldn’t drop packets unless imperative
  - packet drop wastes resources (why?)



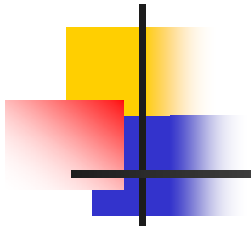
Why is buffer management important?

- Consider the case where there are 2 flows, flow 1 has strict priority over flow 2.
  - Let both flows share the same buffer, of size N, with no differentiation.
  - Let the buffer be empty initially
  - Assume >N packets from flow 2 arrives, occupying all the buffer space
  - Packets from flow 1 arrives later and are dropped (since buffer is full)
  - With sufficiently large difference in arrival rates between flow 2 and flow 1, packets from flow 1 may never be (buffered and) scheduled even though it has higher scheduling priority!!!



Classification of drop strategies

1. Degree of aggregation
2. Drop priorities
3. Drop position
4. When to drop?

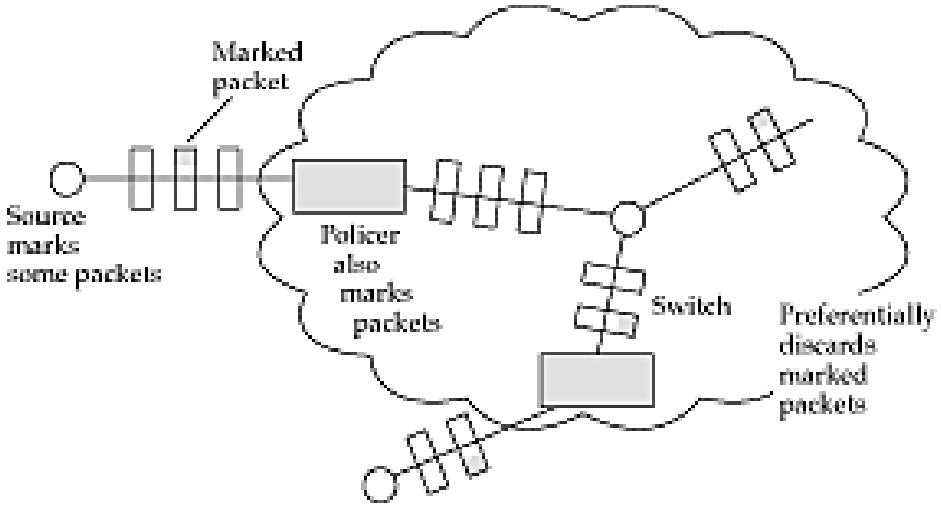


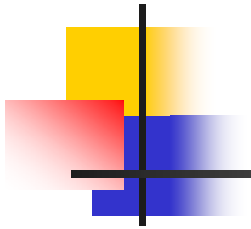
1. Degree of aggregation

- 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
- Issues:
  - Who decides the aggregation: router or another element?
  - If another element decides, how's the aggregation indicated to the router?
  - How many aggregations are needed?
  - The finer the classification the better the protection but more work/overhead for the network elements

2. Drop priorities

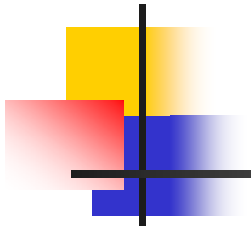
- Drop lower-priority packets first
- How to choose?
  - endpoint marks packets
  - regulator marks packets
  - congestion loss priority (CLP) bit in packet header





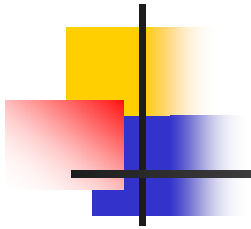
CLP bit: pros and cons

- 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
  - what prevents all packets being marked as high priority?



2. Drop priority (contd.)

- Special case of AAL5
  - want to drop an entire frame, not individual cells
  - 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

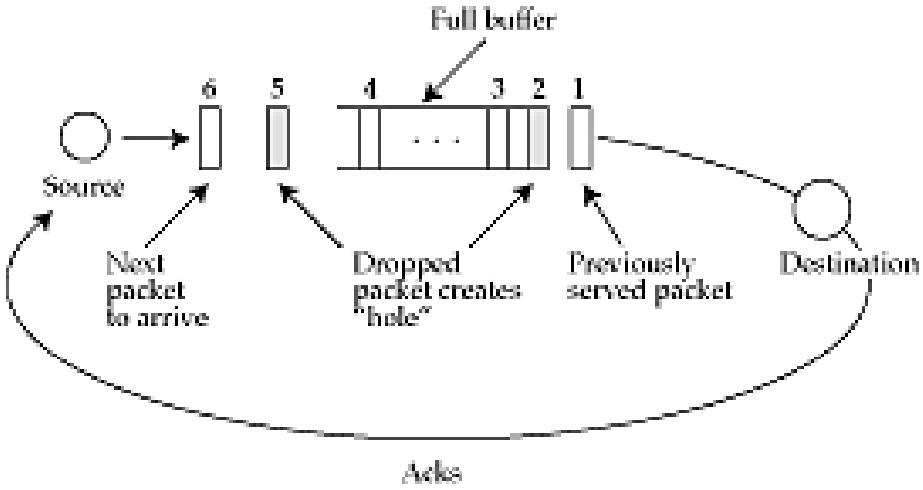


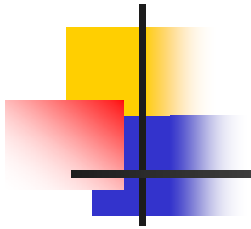
2. Drop priority (contd.)

- Given a set of aggregates of the same “weight”, which aggregate to drop from?
- Drop packet from class with the longest queue
  - Why?
  - Max-min fair allocation of buffers to aggregates

3. Drop position

- Can drop a packet from head, tail, or random position in the queue
- Tail
  - easy
  - default approach
- Head
  - harder
  - lets source detect loss earlier (useful for TCP)





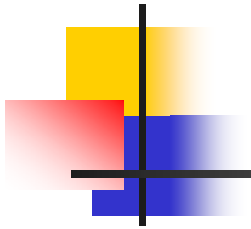
3. Drop position (contd.)

- Random
  - harder to implement
  - Useful?
- Drop entire longest queue
  - easy
  - almost as effective as drop tail from longest queue

How much buffer space per flow?

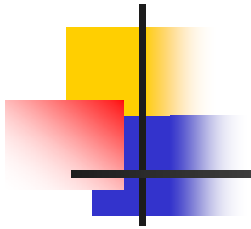
- One way is to define a maximum queue length threshold
  - How should the maximum queue length be set ?
  - Static threshold is inflexible
    - If the thresholds are too small, does not support statistical multiplexing efficiently
    - If the thresholds are too large, does not provide isolation
    - Number of flows/connections can change
  - With dynamic thresholds, the maximum permissible length at any instant is proportional to the amount of unused buffer
    - **$T(t) = \alpha (B - Q(t))$, $\alpha = 2,4,...$**
    - Thresholds are sensitive to load and number of flows
    - Some spare capacity is left to handle transit load

- A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds for Shared-Memory Packet Switches," IEEE/ACM Trans. Commun., vol. 6, no. 2, Apr. 1998, pp. 130--40.



4. Early vs. late drop

- Early drop => drop even if space is available
  - signals endpoints to reduce rate in early stages of congestion
  - 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
  - Does it work?



RED

- Random early detection (RED) makes three improvements
- Metric is moving average of queue lengths
  - 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

RED Algorithm

- For each packet arrival
  - Calculate the average queue size ave Q
- If min <= Q < <= max
  - Calculate probability P_a
  - With probability P_a, mark the packet
- Else if Q > max
  - Mark the packet

- Q is the smoothed version of the queue defined by

$$Q_{k+1}=(1-w) \times Q_k + w \times q$$

  where q is the current queue size.
- The smaller the w, the slower Q reflects changes in the queue size.

Packet Drop Probability (P_a)

- $\mathbf{P_b \leftarrow max_p\,(avg - min_{th})}$

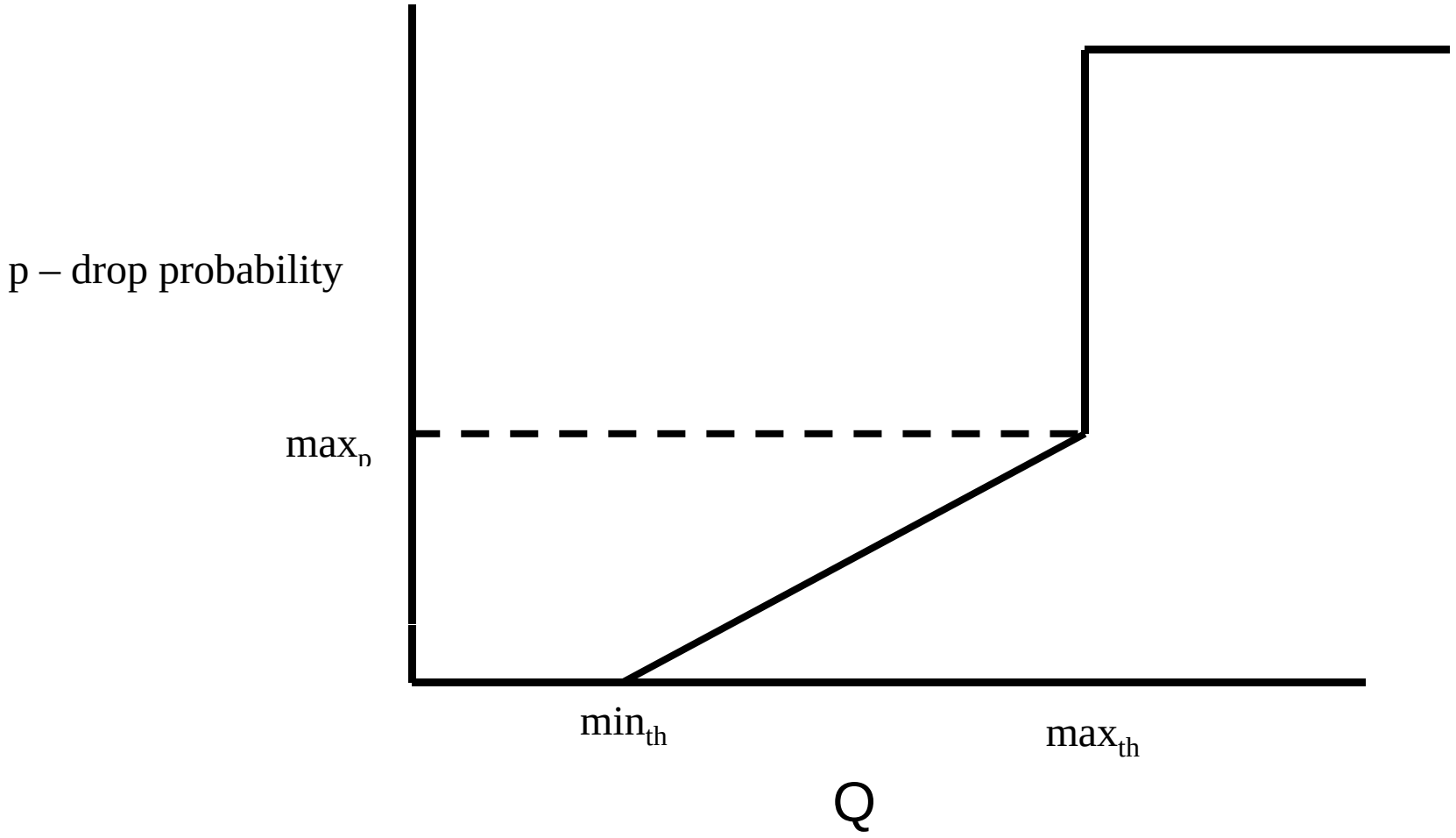
$$\frac{}{\mathbf{(max_{th} - min_{th})}}$$

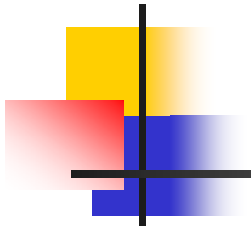
The final packet marking probability

- $P_a \leftarrow \dfrac{P_b}{(1 - count \,.\, P_b)}$

- *count*: # of unmarked packets that have arrived since the last marked packet
- Ensures that the gateway does not wait too long before marking a packet

RED (cont'd)





Issues with RED

- RED is extremely sensitive to # sources and parameter settings
  - Static values of min, max and max_p are not good when network conditions change
- Many variants of RED are proposed:
  - ARED - Adaptive RED
  - FRED - Flow Random Early Drop
  - SRED - Stabilized RED
- Other Active Queue Management (AQM)
  - BLUE
  - REM and many, many more

- T.V. Lakshman and U. Madhow, “The performance of TCP/IP for networks with high bandwidth-delay products and random loss,” IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 5, NO. 3, JUNE 1997.
  - Sections 1 to 3

Model

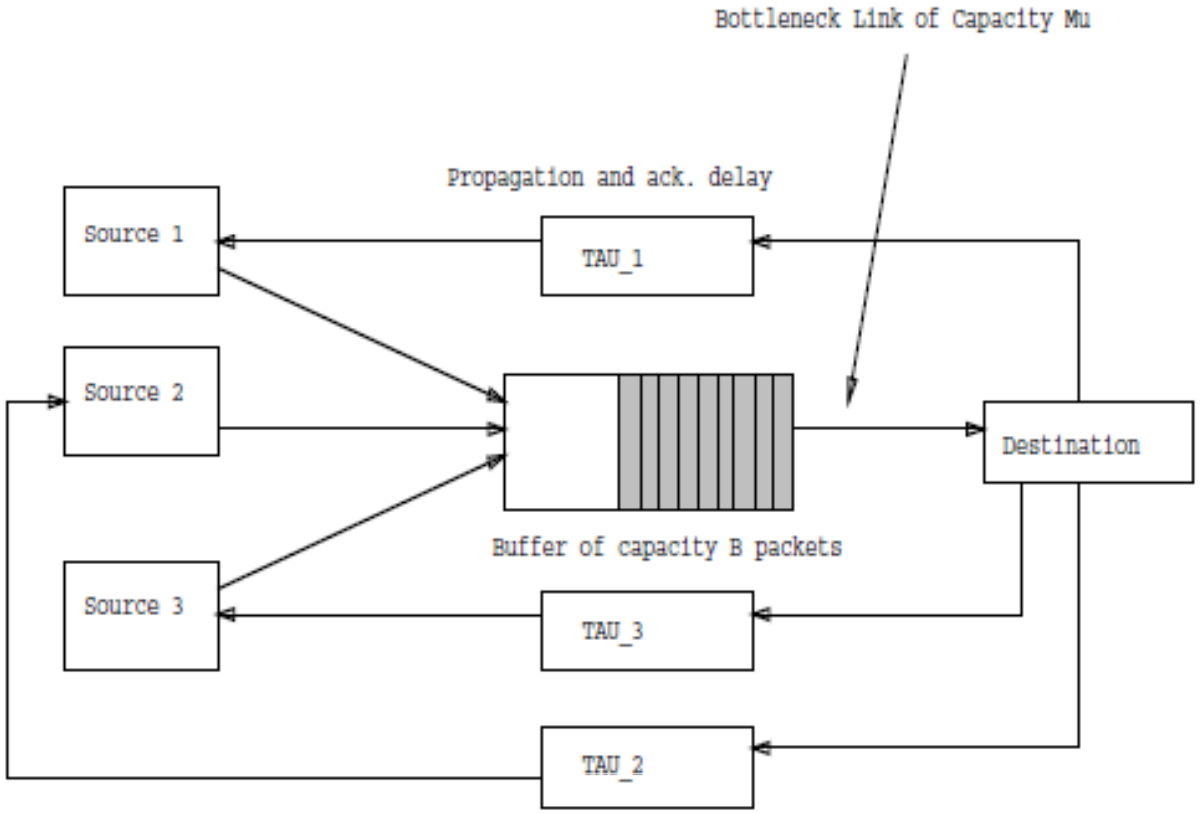


Figure 1: System model showing three connections

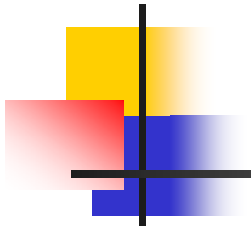
TCP Reno Behavior Recap

(1) After every non-repeated acknowledgement, the algorithm works as before:
if $W < W_t$, set $W = W + 1$; **Slow Start Phase**
else set $W = 1 + 1/[W]$. **Congestion Avoidance Phase**

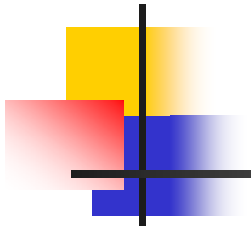
(2) When the duplicate acknowledgements exceeds a threshold,
retransmit "next expected" packet;
set $W_t = W/2$, then set $W = W_t$ (i.e. halve the window);
resume congestion avoidance using new window once retransmission is acknowledged.

(3) Upon timer expiry, the algorithm goes into slow start as before:
set $W_t = W/2$;
set $W = 1$.

Simple Loss Model



- Let $\beta = B / (\mu\tau + 1) = B/T$
  - Where $T = \tau + 1/\mu$
- Each connection uses its allowable window to te fullest extent
- Any packet arriving when the buffer is full is lost (no random loss)
- Maximum window size in steady state
  - $Wpipe = \mu T + B$
- When buffer loss occurs, $Wb \sim 2B$
- Focus on TCP Reno



Slow Start

- Wt = slow start threshold (Wb > Wt)

$$t_{ss} = T log_2 W_t \qquad (6)$$

$$n_{ss} = W_t \qquad (7)$$

Congestion Avoidance

$$W_0(reno) = W_{max}/2, \tag{15}$$

$$\frac{dW}{da} = 1/W. \tag{16}$$

$$\frac{da}{dt} = \lambda = \min\ \{W/T, \mu\} \tag{17}$$

$$\frac{dW}{dt} = \begin{cases} 1/T, & W \leq \mu T, \\ \mu/W, & W \geq \mu T. \end{cases} \tag{18}$$

Phase A

Phase B

Phase A

- Link not fully utilized

$$t_A = T(\mu T - W_0), \tag{19}$$

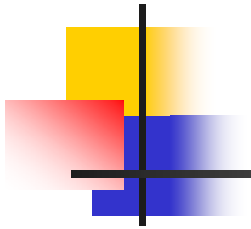
$$n_A = \int_0^{t_A} W(t + t_{ss})/T dt = \int_0^{t_A} (W_0 + t/T)/T dt$$
$$= [W_0 t_A + t_A^2/(2T)]/T \tag{20}$$

Phase B

- Link is “fully” utilized
- For W > μT, W^2 grows as 2μt

$$t_B = [W_{max}^2 - (\mu T)^2]/(2\mu) \qquad (21)$$

$$n_B = \mu t_B, \qquad (22)$$



Questions

- Consider TCP Reno
  - Why is the impact of buffer size?
  - How big should the buffer be for a TCP Reno flow?

β	Link Utilization (Analysis/Simulation)	
	Tahoe	Reno
.1	.604/.604	.818/.818
.2	.660/.664	.871/.870
.31	.708/.718	.915/.911
.32	.856/.858	.919/.916
.8	.953/.954	.996/.994

Table 2: Link Utilization as a function of normalized buffer size ($\mu = 100, \tau = 1$)