3. FLOW, ERROR and CONGESTION CONTROL
Copyrights on Figures

• Figures presented in these notes with the following captions are extracted from the textbook and copyrighted by the publisher as indicated

• DISCLAIMER
  – Whilst every attempt is made to maintain the accuracy and correctness of these notes, the author, Dr. Roberto Togneri, makes no warranty or guarantee or promise express or implied concerning the content.
Flow, Congestion and Error Control

• Flow and Congestion Control
  – Two types of control to avoid dropped/lost packets/frames
    • **Flow control** is concerned with *end-to-end control* of packet rates to avoid overflowing receiver buffers resulting in dropped packets/frames at the receiver
    • **Congestion control** is concerned with *(inter)network control* of packet rates and routes to avoid lengthening or overflowing router queues resulting in long delays or dropped packets/frames respectively
  – Send node must not transmit data faster than can be handled by the receive node
    • *Received data has to be processed*: headers read, checksum checked, data stored, etc.
    • *Received data has to be buffered* for retrieval by user (application end-user)
    • *Received data has to be queued* for retransmission on another link (router / bridge)
Flow, Congestion and Error Control

- **Error Control**
  - Allows recovery from lost or damaged packets/frames
    - *Damaged*: bit errors due to channel noise
    - *Lost*: over multiple links (e.g. end-to-end error control) packets can be “dropped”
  - **Automatic-Repeat-Request (ARQ) systems**
    - Relies on error detecting codes (e.g. CRC-32) stored as the frame/packet header/trailer checksum (FCS in LAPB/Ethernet, Checksum in TCP/UDP)
    - Requires timers, acknowledgments and retransmissions of packets/frames
    - Operation:
      - In full-duplex transmission acknowledgments for outgoing data frames can be “piggy-backed’ on the incoming data frames
      - If sender does not receive a positive ACK before the timer expires or receives a negative ACK (NACK) the frame is re-transmitted
  - **Forward-Error-Correction (FEC) systems**
    - Relies on error correcting codes (e.g. Hamming codes) to correct any bits in an error at the receiver
    - Due to lower information rates, additional processing at the receiver and lower error protection (fewer bits corrected then could be detected) FEC is not used.
Flow, Congestion and Error Control

- Different scope for flow / congestion control
  - **Hop scope**: between directly connected systems (e.g. router to next router)
    - LAPB flow control between adjacent packet-switched nodes
    - Frame Relay, ATM and Internet congestion avoidance and rate control mechanisms
  - **Network Interface**: between end system and entry into network
    - the D-bit in the X.25 header
  - **Entry to Exit**: between entry and exit points to network of a logical connection
  - **End to End**: between the end systems (source and destination) of a connection
    - TCP end-to-end flow and congestion avoidance control mechanisms

Figure 9.2 (hsn1e)
The \( a \) parameter

**What is \( a \)?**
- Important performance parameter for link, MAC and transport protocols
- Different networks exhibit different ranges for \( a \)
  - Performance for one protocol may be excellent for one range of \( a \) but fail miserably for another range
- \( a = \) (Propagation time / Transmission Time)

Let:

| \( d \) | Distance between source and destination stations in m. |
| \( V \) | Propagation velocity. This is \( c = 3 \times 10^8 \) m/s for wireless, satellite and optic fibre transmission media and 0.67\( c \) for copper media. |
| \( L \) | Length of the frame in bits. |
| \( R \) | Data rate on the link, in bits per second. |
| \( T_{frame} \) | Time to transmit the complete frame onto the link or network. This is a function of the data encoding technology (e.g. 100 Mbps Fast Ethernet will take a \((1/10)^{th}\) of the time to transmit the same frame as a 10 Mbps Ethernet) = \( L / R \) |
| \( T_{prop} \) | Propagation time from source to destination (or destination to source). Since in most cases the propagation velocity is close to the speed of light the time is usually directly proportional to the distance = \( d / V \) |

Then we have:

\[
a = \frac{T_{prop}}{T_{frame}} = \frac{(d/V)}{L/R} = \frac{Rd}{LV}
\]
The \( a \) parameter

- **Typical values of \( a \)**
  - **ATM WAN**
    - \( d = 1000 \text{ km} \); \( V = 3 \times 10^8 \text{ m/s} \); \( L = 424 \text{ bits} \); \( R = 155.52 \text{ Mbps} \)
    - \( a = 1222 \)
  - **Satellite WAN**
    - \( d = 35,863 \text{ km} \); \( V = 3 \times 10^8 \text{ m/s} \); \( L = 10000 \text{ bits} \); \( R = 1 \text{ Mbps} \)
    - \( a = 11.95 \)
  - **100 Mbps Ethernet LAN**
    - \( d = 100 \text{ m} \); \( V = 2 \times 10^8 \text{ m/s} \); \( L = 1000 \text{ bits} \); \( R = 100 \text{ Mbps} \)
    - \( a = 0.05 \)
  - **Analogue Modem MAN**
    - \( d = 10 \text{ km} \); \( V = 2 \times 10^8 \text{ m/s} \); \( L = 1000 \text{ bits} \); \( R = 28.8 \text{ Mbps} \)
    - \( a = 0.00144 \)

- **Properties of \( a \)**
  - If \( a > 1 \) then \( T_{\text{prop}} > T_{\text{frame}} \)
    - More than one frame can be in transit between source and destination
  - If \( a < 1 \) then \( T_{\text{frame}} > T_{\text{prop}} \)
    - Only one frame is in transit; destination receives the first bit of the frame before the sender finishes transmitting the last bit
Stop-and-Wait ARQ

• **Operation**
  
  – **Error-free operation:**
    • Source sends frame and waits for ACK from receiver before sending next frame
  
  – **If frame is lost/damaged:**
    • Source sets timer (usually a little over the round-trip-time for the ACK to arrive)
    • If timer expires source retransmits the same frame
  
  – **If ACK is lost/damaged**
    • Source timer expires and frame is retransmitted
    • Receiver discards duplicate frame

• **Properties**
  
  – ACK frame length $\ll$ data frame length
  
  – Poor link utilisation
    • link under-utilised if $a > 1$
Stop-and-Wait Error-Free Throughput

Net time to send one frame, $T$, involves:

1. Source transmits the complete frame ($T_{\text{frame}}$)
2. Frame has to reach destination (+ $T_{\text{prop}}$)
3. Destination has to process frame (+ $T_{\text{proc}}$)
4. Destination transmits the complete ACK frame (+ $T_{\text{ack}}$)
5. ACK frame has to reach source (+ $T_{\text{prop}}$)
6. Source has to process ACK frame (+ $T_{\text{proc}}$)

Thus:

$$T = T_{\text{frame}} + T_{\text{prop}} + T_{\text{proc}} + T_{\text{ack}} + T_{\text{prop}} + T_{\text{proc}}$$

For most cases one can assume: $T_{\text{proc}} \approx 0$, and $T_{\text{frame}} \gg T_{\text{ack}}$

Hence:

$$T = T_{\text{frame}} + 2T_{\text{prop}}$$

This means that a frame is sent at the rate of one frame every $T$ seconds. However the actual rate on the link must be $1 / T_{\text{frame}}$, so the normalised throughput or utilisation $U$ can be expressed as:

$$U = \frac{1}{1 / (T_{\text{frame}} + 2T_{\text{prop}})} = \frac{T_{\text{frame}}}{T_{\text{frame}} + 2T_{\text{prop}}} = \frac{1}{1 + \alpha}$$
Stop-and-Wait ARQ Throughput

Time to send one frame with consecutive failures, $T$
If a data frame or ACK frame is lost/damaged the source retransmits the frame. The retransmitted frame (or its ACK) can in turn be lost/damaged, resulting in a 2nd retransmission of the same frame. And so on ….
If a frame suffers $x$ consecutive failures it must be retransmitted $x$ times for successful delivery, the time required for $x$ retransmissions is $x(T_{\text{frame}} + 2T_{\text{prop}})$. Let $N_x = x + 1$ be the average number of times each frame is transmitted (original transmission + $x$ retransmissions), then:

$$T = N_x (T_{\text{frame}} + 2T_{\text{prop}})$$

Deriving the expected number of transmissions, $N_x$
Let $P$ be the probability that a single frame is in error. Let $Pr(k)$ represent the probability of exactly $k$ attempts to transmit a frame successfully:

$$Pr(k) = Pr[(k - 1) \text{ unsuccessful attempts} + \text{the successful attempt}]$$
$$= P^{k-1} \times (1 - P)$$

Therefore:

$$N_x = E[\text{transmissions}] = \sum_{i=1}^{\infty} (i \times Pr[i \text{ transmissions}])$$
$$= \sum_{i=1}^{\infty} (iP^{i-1}(1 - P)) = \frac{1}{1 - P} \quad \text{using the identity} \quad \sum_{i=1}^{\infty} (iX^{i-1}) = \frac{1}{(1 - X)^2} \quad \text{for} \ (-1 < X < 1)$$

Stop-and-Wait ARQ Throughput

$$U = \frac{1 - P}{1 + 2a}$$
For error-free transmission $P = 0$. 

18-Jan-99 IND426, Roberto Togneri, E&E Eng, Univ. of Western Australia
Sliding-Window Techniques

• Source and Destination agree on the number of frames W
  – Operation
    • Source can send W data frames immediately, and each frame has a sequence number (SEQ)
    • Destination sends a RR (Receive Ready) or ACK frame m to acknowledge that all frames up to
      and including (m-1) have been received (i.e. last frame received had SEQ = (m-1))
    • Source receives RR m and can then transmit up to m+W frames
  – Destination / Source need to allocate a receive / send circular buffer (respectively)
    Data and RR frames need to store the value m in a k-bit frame header field
    • Size of buffer is 2^k - 1 and sequence number wraps-around at 2^k - 1 (i.e. SEQ = 0 … 2^k - 1 )
    • Max imum value of W is 2^k - 1
  – Properties
    • More than one frame can be in transit → improved link utilisation for a > 1
    • Destination can either acknowledge each frame received or send cumulative acknowledgments
      (i.e. “ack each frame” or “ack each window”)
  • ARQ variants (with errors)
    – If destination receives an out-of-order frame after m (e.g. frame m-2, m-1, m+1, m+2 →
      frame m is lost!) the destination sends a REJ m (Go-Back-N ARQ) or SREJ m (Selective-
      Reject ARQ) frame back to the sender
      • In Go-Back-N ARQ the destination will discard frames until frame m is received;
        the sender will retransmit frame m and all subsequent frames
      • In Selective-Reject ARQ the destination will buffer out-of-order frames received after m;
        the sender will only retransmit frame m
    – If a RR, REJ or SREJ frame is lost the source may time-out after exhausting the window
      and not having received an ACK to recommence transmission
      • Source will send an RR (P=1) frame to the destination to force it to resend the last/pending
        RR, REJ or SREJ frame
    – Go-Back-N ARQ is more commonly used (simpler to implement)
Sliding-Window Depiction

(a) Sender's perspective

(b) Receiver's perspective

Figure 9.5 (hsn1e)
Sliding-Window Example

Source System A

Destination System B

Figure 9.6 (hsn1e)
Sliding-Window ARQ Protocols

(a) Go-back-N ARQ

(b) Selective-reject ARQ

Figure 9.7 (hn1e)
Let \( W \) be the window size. Assume the source sends \( W \) frames.

Consider frame 1. If the destination immediately sends RR 2 upon receiving frame 1 to acknowledge it, then the source has to wait \( T = T_{\text{frame}} + 2T_{\text{prop}} \) since sending frame 1 for it to be acknowledged.

There are 2 cases to consider:

1. If \( WT_{\text{frame}} \geq T \) \((W \geq 1 + 2a)\), then source is still sending frames by the time the first frame is acknowledged. Thus the source can transmit continuously without pause and the normalised throughout is 1

2. If \( WT_{\text{frame}} < T \) \((W < 1 + 2a)\), the actual data rate is \( 1/T_{\text{frame}} \) but the net data rate is \( W \) frames in \( T \) seconds, which gives a normalised throughput of: 

\[
U = \frac{W/(T_{\text{frame}} + 2T_{\text{prop}})}{1/T_{\text{frame}}} = \frac{WT_{\text{frame}}}{T_{\text{frame}} + 2T_{\text{prop}}} = \frac{W}{1 + 2a}
\]

**Sliding-Window Throughput:**

\[
U = \begin{cases} 
1 & W \geq 1 + 2a \\
\frac{W}{(1 + 2a)} & W < 1 + 2a
\end{cases}
\]
Selective-Reject ARQ Throughput

The same reasoning used for Stop-and-Wait ARQ yields the following expression for the normalised throughput:

\[
U = \begin{cases} 
\frac{1}{W} & \text{if } W \geq 1 + 2a \\
\frac{N_x}{W} & \text{if } W < 1 + 2a \\
\end{cases}
\]

where \( N_x \) is the average number of times each frame must be transmitted.

Since the analysis of Selective-Reject ARQ with regards to frame retransmissions is identical to Stop-and-Wait ARQ (i.e. \( N_x = 1/(1 - P) \)) we have:

\[
U = \begin{cases} 
1 - P & \text{if } W \geq 1 + 2a \\
\frac{W(1 - P)}{1 + 2a} & \text{if } W < 1 + 2a \\
\end{cases}
\]

Selective-reject ARQ Throughput
Go-Back-N ARQ Throughput

The same reasoning used for Stop-and-Wait ARQ yields the following expression for the normalised throughput:

\[
U = \begin{cases} 
\frac{1}{W} & \text{if } W \geq 1 + 2a \\
\frac{N_x}{W} & \text{if } W < 1 + 2a \\
\end{cases}
\]

where \(N_x\) is the average number of times each frame must be transmitted.

Deriving the expected number of transmissions, \(N_x\)

Let \(P\) be the probability that a single frame is in error. Let \(\text{Pr}(k)\) represent the probability of exactly \(k\) attempts to transmit a frame successfully:

\[
\text{Pr}(k) = \text{Pr}[(k - 1) \text{ unsuccessful attempts}] + \text{Pr}[\text{the successful attempt}] = p^{k-1} \times (1 - P)
\]

The cost function, \(f(i)\), is the total number of frames transmitted if \(i\) attempts are needed to transmit the original frame \(m\). For Selective-Reject ARQ (and Stop-and-Wait ARQ) the obvious result \(f(i) = i\) follows.

For Go-Back-N ARQ however the \((i-1)\) retransmissions require retransmitting not only frame \(m\) but the \(K\) frames transmitted after frame \(m\) and before the source received the REJ \(m\). Thus \(f(i) = 1 + (i - 1)K = (1 - K) + Ki\)
Go-Back-N ARQ Throughput

We derive \( N_x \) as follows:

\[
N_x = E[\text{transmissions}] = \sum_{i=1}^{\infty} (f(i) \times \text{Pr}[i \text{ transmissions}])
\]

\[
= 1 - K + \frac{K}{1-P} = \frac{1-P+KP}{1-P}
\]

There are 2 cases to consider in deriving K:

1. If \( W \geq (1+2a) \), then \( K = \frac{\text{Time taken to receive REJ m}}{\text{Time taken to transmit 1 frame}} = \frac{T_{\text{frame}} + 2T_{\text{prop}}}{T_{\text{frame}}} = 1+2a \)

2. If \( W < (1+2a) \), then \( K = W \)

Go-Back-N ARQ Throughput:

\[
U = \begin{cases} 
  \frac{1-P}{1+2aP} & \text{if } W \geq 1+2a \\
  \frac{W(1-P)}{(1+2a)(1-P+WP)} & \text{if } W < 1+2a 
\end{cases}
\]
ARQ Throughput as a function of $a$

Figure 9.12 (hsn1e) [$P = 10^{-3}$]
The Need for Congestion Control

• Network Congestion
  – Router/node input queue fills and arriving packets are either discarded or experience a long delay in the queue
    • Sender has a time-out and retransmits un-acknowledged packet
    • Receiver may also time-out and send a special request packet
  – Retransmissions cause more packets to enter network and more queues to fill
    • The *offered load* of packets injected into the network increases but the net end-to-end *throughput* does not, and may level off or even drop-off
  – With any control the network collapses and throughput drops to zero

• Congestion Control
  – Router/node exercises a control on the flow of packets that pass through it
    • If queues begin to fill packets are deliberately dropped before congestion starts
    • The rate of packets through the node is restricted or “reserve” capacity is used
    • The route of packets is changed to avoid congested nodes
    • The sender is explicitly or implicitly triggered to reduce the transmission rate
The Need for Congestion Control

Figure 9.25 (dccn4)
1. Homogenous nodes with identical characteristics.
2. Packets are either in transit (node to node) or input (injected into network at a node)
   a) \( N \): size of finite buffer at each node
   b) \( P_B \): probability of getting blocked by a full buffer
   c) \( \lambda_T \): arrival rate at node of transit packets
   d) \( y_T \): throughput at node of transit packets = \( \lambda_T(1-P_B) \)
   e) \( \lambda_I, y_I \): as above but for input packets
   f) \( p_T y_T \): fraction of throughput transit packets that are destined for this node and depart from network
   g) \( h \): average number of hops from source to destination = \( 1/p_T \)
Congestion Analysis: Flow Control Effects

• Point-to-point flow control only affects destination host
  – Maximum throughout happens when $W \geq 1 + 2a$
  – Larger $W$ implies larger send and receiver buffers (more expensive)
  – Thus $W = 1 + 2a$ seems like a good choice
    • If too small ($W < 1 + 2a$) then utilisation is low
    • If too large ($W >> 1 + 2a$) then buffers are bigger

• End-to-end flow control affects congestion of intermediate hosts
  – Maximum throughout happens when $W \geq 1 + 2a$
  – The number of frames in transit within the network is $\propto W$
    • If the network is viewed as one huge queuing system then $q \equiv W$, $T_q = (q/\lambda)$ and the delay in the network increases proportionally with $W$
    • Longer end-to-end delays are symptomatic of congestion
  – Thus $W = 1 + 2a$ seems like a good choice
    • If too small ($W < 1 + 2a$) then utilisation is low
    • If too large ($W >> 1 + 2a$) then end-to-end delays increase and congestion is likely
Congestion Control: Causes and Solutions

• Causes
  – Fast sender
  – Slow or low capacity router
    • Slow CPU to process headers and calculate route
    • Not enough memory to queue packets
  – Low link rates
• Solutions
  – Open loop (Avoidance and Prevention Measures)
    • Optimal scheduling and routing of packets
    • Selected discard of packets
    • Input buffer limiting
    • Traffic shaping
  – Closed loop (Monitor, Notify and Take action)
    • Monitor for congestion onset
      – Maintain running estimates of average delay, average queue length, delay variation, retransmissions, dropped packets, line utilisation, etc.
    • Notify other nodes and end hosts that congestion is occurring
      – Congestion detected in one node can be relieved by action from other nodes
      – Use special control packets or special bits in normal packet headers to notify other nodes of congestion
    • Take action!
Congestion Control: Monitoring

- **Parameters to monitor**
  - Internal state of router
    - Length of input and output queues
    - Number of packets discarded
  - End host
    - Mean Round-Trip Time
    - Round-Trip Time Variance
  - External state of router
    - Utilisation of output links

Model output link queue as a M/D/1 system

\[ T_s = \text{service time} = \frac{\text{packet length in bits}}{\text{link data rate in bps}} \]

\[ T_q = \text{measured delay (time packet spends in system)} \]

Then:

\[ T_q = \frac{T_s (2 - \rho)}{2(1 - \rho)} \Rightarrow \rho = \frac{2(T_s - T_q)}{T_s - 2T_q} \]

- **Average Parameter**
  - \( f = \) current value of parameter
  - \( u(n) = \) average value of parameter at sampling time \( n \)

\[ u(n) = 0.5u(n-1) + 0.5f \]
Congestion Control: Notification

- **Choke Packet**
  - If output line has $u(n) > U_{\text{threshold}}$ a choke packet is sent back to the source of the corresponding outgoing packets on that line
  - **Actions**
    - Source reduces rate to destination by X percent when choke packet is received
    - Intermediate router nodes that see choke packet will reduce corresponding outgoing flow to immediately alleviate congestion
  - **Results**
    - Offered load to network is reduced, thereby allowing queue lengths to decrease and potential congestion is avoided
  - **Issues**
    - Choke packets should be sent before the onset of congestion
    - Choke packets add to network congestion

- **Notification Examples**
  - *Frame Relay*: BECN and FECN bits in Address header field
  - *ATM*: 3-bit Payload Type in cell header
  - *TCP*: delayed or lost ACKs
Congestion Control: Actions

• Admission Control and Variations
  – Network does not allow new connections / virtual circuits
  – Routers “reroute” new and old connections around congested nodes
  – Source host reduces the transmission rate (e.g. choke packet actions)

• Load Shedding / Selective Discard
  – End-to-end hosts terminate connections
  – Network routers discard rather than queue packets
  – Which packets to discard and which connections to terminate?
    • Most recent packets of a file transfer should be discarded in favour of older ones
      – If older packets discarded the receiver may later discard resulting “out-of-order” packets
    • Older packets of a real-time transaction should be discarded in favour of recent ones
      – To retain real-time or “live” transmission out-dated packets should be discarded
    • Idle connections should be terminated in favour of active connections
      – a form of admission control to prevent new active connections
  – Low-priority packets are discarded in favour of high-priority packets
    • Assumes network provides for different levels of service
    • Assumes application can differentiate between low and higher priority data

• Congestion Control Examples (for congestion avoidance)
  – Frame Relay CIR, Internet RED
Congestion Avoidance: RED

- Random Early Detection
  - Congestion Avoidance Design Goals
    - Bound on average queue length (control average queue size and average delay);
      Prevent bias against bursty traffic (allow queue length to grow in controlled fashion)

```
calculate the average queue size avg
if avg < TH_min
  queue packet, set count to -1
else if TH_min ≤ avg < TH_max
  increment count and calculate the probability P_a
  with probability P_a
    discard the packet, set count to 0
  else with probability (1 - P_a)
    queue packet
else if avg ≥ TH_max
  discard packet, set count to 0
```
Congestion Avoidance: RED

\[ P_a = \frac{F \times P_{\text{max}}}{1 - \text{count} \times F \times P_{\text{max}}} \]

where \( F = \frac{\text{avg} - \text{TH}_{\text{min}}}{\text{TH}_{\text{max}} - \text{TH}_{\text{min}}} \)

- **Features**
  - The closer \( \text{avg} \) is to \( \text{TH}_{\text{max}} \) (\( F \) closer to 1), the higher the probability of discard
  - As long as \( \text{avg} \) is in the critical region, a count of how many consecutive packets escape discard is kept; the higher the value of count, the higher the probability of discard and when count = \( 1 / (F \times P_{\text{max}}) - 1 \) a packet is discarded with \( P_a = 1 \)
    - Using count forces a more or less uniform spacing of discards
  - Better performance to drop-tail policy (packets dropped only when queue is full)
Congestion Avoidance: Rate Control

• Types of traffic
  – Bursty traffic
    • Highly variable packet arrival rates
      → peak rate may cause queues to fill although average rate would not
  – Smooth traffic
    • Low variable packet arrival rate → higher average rate of traffic can be sustained

• Rate Control
  – Transform bursty traffic to smooth traffic
  – Token-based control
    • *Token*: allows a waiting packet to be transmitted
      – A token is used up if it causes a packet to be transmitted
      – Tokens are generated at a regular rate, *r*
    • *Bucket*: arriving packets are placed in a bucket until a token becomes available
    • *count*: Number of tokens in the system
    • *Capacity* (C): Maximum number of packets that can be in the bucket
    • *Token Credit* (W): Maximum number of unclaimed tokens (*count* ≤ W)
  – Rate Control Example: ATM GCRA

• Jitter Control
  – Minimise delay variations
    • Timestamp each packet with the transit time
      – If (packet behind schedule) then promote to head of queue
      – If (packet ahead of schedule) then place at tail of queue
Congestion Avoidance: Rate Control

• Time Window rate control
  – Operation:
    • Packet ready to be transmitted:
      – **If** \( (\text{count} > 0) \) **then** packet transmitted and \( \text{count} = \text{count} - 1; \) **else wait** (until \( \text{count} > 0 \))
    • At \( t = 0 \) and every \( (W/r) \) seconds:
      – Set \( \text{count} = W; \) thus up to \( W \) packets waiting will be allowed to be transmitted
      – This happens every \( (W/r) \) seconds → rate is controlled
    • Tokens act as an “ack at end of window” which arrives every \( (W/r) \) seconds

• Token rate control
  – Operation:
    • Packet ready to be transmitted (arrives):
      – **If** \( (\text{count} > 0) \) **then** transmit and \( \text{count} = \text{count} - 1; \)
        **else if** (number of packets waiting = \( C \)) **then** discard;
        **else** add to bucket and **wait** (until \( \text{count} > 0 \))
    • Every \( (1/r) \) seconds:
      – **If** \( (\text{count} \leq W) \) **then** \( \text{count} = \text{count} + 1; \) **else** continue
  – Leaky Bucket Algorithm
    • Restricted to the case \( W = 0 \) → Strictly constant bit rate traffic
  – Token Bucket Algorithm
    • Allows \( W \geq 0 \) → Permits bursty traffic but still limits average rate to \( r \)
Leaky Bucket Example:

Network has a transmission rate of 25 MB/s.

Source data rate is bursty with 1 MB bursts of data every second (burst lasts 1/25 sec = 40 ms).

*Bucket Input:* 25 MB/s for 40 ms then nothing for the next 960 ms. Average source rate of 1MB/s.

*But* intermediate nodes can only sustain a rate of 2 MB/s.

Use leaky bucket with \( r = 2 \) MB/s and \( C = 1 \) MB.

*Bucket Output:* 1 MB at \( r = 2 \) MB/s \( \Rightarrow \) at 2 MB/s lasts 1/2 sec = 500 ms, then nothing for the next 500 ms.
**Token Bucket Example:**

Assume $W = 250$ kB, $C = 1$ MB, $r = 2$ MB/s and a 1 MB burst at 25 MB/s (duration 40 ms) every second (as in the Leaky Bucket example).

If the bucket is full of tokens and a 1 MB burst arrives, then the token bucket will drain at 25 MB/s for 10.6 msec, (in 10.6 msec the 250 kB of tokens in the bucket plus the extra tokens that arrive in 10.6 msec will drain at the full burst rate of 25 MB/s)

After 10.6 msec the remaining packets in the bucket will drain at 2 MB/s for 367 ms and then no output for the next 622 ms until the next burst arrives (except that tokens will accumulate at the rate of 2 MB/s for 122 ms until there is $W = 250$ kB of token credit).
Congestion Avoidance: Queuing Discipline

• Rationale
  – Sources which send more packets will on average grab more of the bandwidth → great if the source pays, unfair otherwise.

• Fair Queuing
  – Multiple queues for each output link of switch/router, one for each source:
    • Select packet from each queue per tick by Round-Robin (RR) scheduling
    • Bandwidth allocated fairly to all sources
    • Use byte-by-byte RR to avoid sources with larger packets using more bandwidth
      – Bytes are counted RR and the a packet is selected when all its bytes have been counted

    – Figure 5.29 (cne3e)

• Weighted Fair Queuing
  – If source “pays” → may demand different levels of service
    • Use priority-based RR by weighting each queue, $w_i$.
      Selects $w_i$ bytes from queue i per tick.