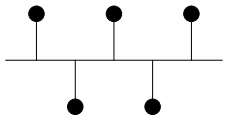
Data Communication Networks

Lecture 9

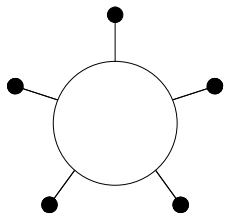
Saad Mneimneh
Computer Science
Hunter College of CUNY
New York

Multiaccess networks

- Nodes are not connected by separate point-to-point links
- Nodes share a communication medium
- All nodes transmit using the shared medium and all nodes see all messages
- Need to coordinate the use of the shared medium (e.g. collisions)
- Examples: Ethernet and Token Ring



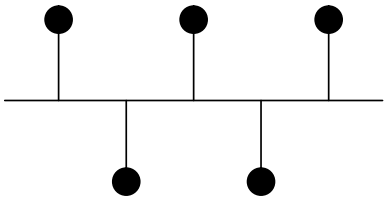
Ethernet



Token Ring

- Ethernet (IEEE 802.3), developed in the 1970s, uses an algorithm that has roots in the Aloha protocol developed by the University of Hawai
- Token Right (IEEE 802.5) uses an algorithm developed by IBM in the 1980s

Aloha



- m nodes, each with a Poisson arrival of rate λ/m
- Time is divided into slots, each slot corresponds to the time of transmitting a packet (packet have same size)
- Feedback at the end of each slot
  - **0**: slot was idle
  - **1**: transmission was successful
  - **e**: error, e.g. collision

Aloha (simplified version): each node with a packet at the beginning of a slot (backlogged node), transmits with probability p

Probability of successful transmission

- Assme n backlogged nodes
- The probability of a successful transmission is

$$P_{succ} = \binom{n}{1} p(1-p)^{n-1}$$

- Using the approximation $(1-p)^n \approx e^{-np}$ for small p, we get

$$P_{succ} \approx \frac{np}{1-p} e^{-np} \approx npe^{-np} = g(n)e^{-g(n)}$$

  where $g(n) = np$ is the expected number of transmission attempts
- P_{succ} is also the expected number of departures per time slot

Expected number of arrivals

- Let $N(t)$ be the number of arrivals at a node during time interval t

$$Pr[N(t) = x] = \frac{(\frac{\lambda}{m}t)^x e^{-\frac{\lambda}{m}t}}{x!} \quad \text{(Poisson)}$$

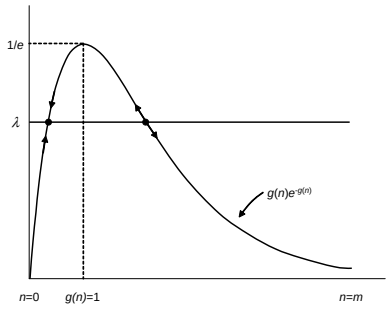
- The expected number of arrivals per time slot is

$$\begin{aligned} mPr[N(1) > 0] &= m(1 - Pr[N(1) = 0]) \\ &= m(1 - e^{-\lambda/m}) \\ &\approx \lambda \quad (\text{if } m >> \lambda) \end{aligned}$$

- Therefore, for stability we need

$$\lambda < P_{succ} = g(n)e^{-g(n)}$$

Stability



- Maximum throughput $= 1/e$ when $g(n) = 1$
- Aloha is not stable
  - there is one stable point with throughput $\lambda < 1/e$
  - there is one unstable point, in the event of crossing that point (n considerable fraction of m), the system goes unbounded, and throughput goes to zero

Stabilizing Aloha

We will look at two ways for stabilizing Aloha:

- Blocking future packets on backlogged nodes
- Changing p dynamically to maintain $g(n) = 1$

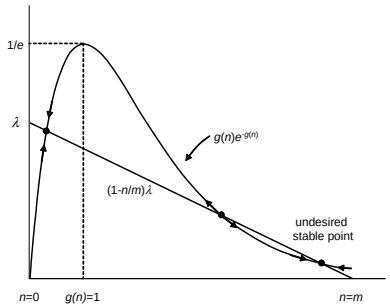
Blocking packets

- Assume backlogged nodes are not allowed to receive more packets
  - a packet arriving at a backlogged node simply disappears
  - each node keeps at most one packet
- The expected number of arrivals is now

$$\begin{aligned}(m-n)(1-e^{\lambda/m}) &= (1-n/m)m(1-e^{\lambda/m}) \\ &\approx (1-n/m)\lambda \ \ (\text{if } m >> \lambda)\end{aligned}$$

  - $n = 0 \Rightarrow \lambda$
  - $n = m \Rightarrow 0$
  - horizontal line in previous figure becomes oblique
- The oblique line will possibly cross the curve at another stable point

Stability after blocking



- Crossing the unstable point will now lead to another stable point
- The second stable point, however, is not desired
  - throughput is close to zero
  - high fraction of nodes are backlogged (and now cannot receive new packets)
- The undesired stable point can be eliminated by making p small enough for the line to move inside the curve

p small enough...

- When is p small enough?
  - when $(1 - n/m)\lambda$ is tangent to $g(n)e^{-g(n)}$
  - $\frac{d}{dn}g(n)e^{-g(n)} = pe^{-np} - p^2ne^{-np}$
  - this must be equal to $-\frac{\lambda}{m}$

$$p(np-1)e^{-np} = \frac{\lambda}{m}$$

$$\log p + \log(np-1) - np = \log\frac{\lambda}{m}$$

$$np \approx \log\frac{mp}{\lambda}$$

Now

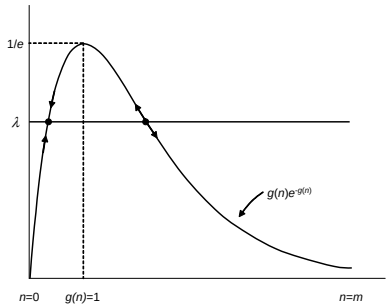
$$\log\frac{mp}{\lambda}e^{-\log\frac{mp}{\lambda}} = (1 - \frac{\log\frac{mp}{\lambda}}{mp})\lambda$$

$$\log\frac{mp}{\lambda}\frac{\lambda}{mp} = (\frac{mp - \log\frac{mp}{\lambda}}{mp})\lambda$$

$$mp = 2\log\frac{mp}{\lambda}$$

- p is in the order of $\frac{\log m}{m}$
- this means a packet experiences a delay proportional to $\frac{m}{\log m}$ $(1/p)$

Maintaining $g(n) = 1$



- p is changed dynamically to keep $g(n) = np = 1$
- Since n is not known at a given node, let $\hat{n}$ be the estimate of n

$$p = \min(1, \frac{1}{\hat{n}})$$

- The $\min$ operation ensures that p will never exceed 1 when the estimate $\hat{n}$ is too small

Estimating n

The estimated backlog $\hat{n}$ is updated at the beginning of each slot k

$$\hat{n}_k = \begin{cases} \max(\lambda, \hat{n}_{k-1} + \lambda - 1) & \text{for } \mathbf{0} \text{ or } \mathbf{1} \text{ feedback (idle or success)} \\ \hat{n}_{k-1} + \lambda + \frac{1}{e-2} & \text{for } \mathbf{e} \text{ feedback (collision)} \end{cases}$$

- The addition of λ accounts for new arrivals
- The $\max$ operation ensures that the estimate is never less than the contribution from new arrivals (needed only when subtracting)
- On successful transmission, 1 is subtracted from the previous backlog
- Subtracting 1 on idle slots and adding $\frac{1}{e-2}$ on collisions is also appropriate
  - When estimate is good $\hat{n} \approx n$
    - $P_{succ} \approx g(n)e^{-g(n)} \approx 1/e = 0.368$
    - $P_{idle} \approx e^{-g(n)} \approx 1/e = 0.368$
    - $P_{collison} \approx 1 - 2/e = 0.264$
  - $E[n_k - n_{k-1}] = (\lambda - 1)2/e + (\lambda + \frac{1}{e-2})(1 - 2/e) = \lambda - 1/e$
  - this is the correct expected increment in backlog
- But λ is unknown $\Rightarrow$ use fixed $\lambda = 1/e$ for all actual (unknown) $\lambda < 1/e$

Delay

(assuming you know M/G/1)

- A packet is successfully transmitted with $P_{succ} = 1/e$
- Assuming packets are served in order (this assumption does not affect the average performance), we can model Aloha as an M/G/1 queue with vacations where

$$Pr[X = i] = 1/e(1 - 1/e)^{i-1} \quad i \geq 1$$

$$\overline{X} = e, \overline{X^2} = 2e^2 - e, \overline{V} = 1, \overline{V^2} = 1$$

$$W = \frac{\lambda \overline{X^2}}{2(1 - \lambda \overline{X})} + \frac{\overline{V^2}}{2\overline{V}} = \frac{\lambda(2e^2 - e)}{2(1 - \lambda e)} + \frac{1}{2}$$

$$T = W + \overline{X} = \frac{\lambda(2e^2 - e)}{2(1 - \lambda e)} + \frac{1}{2} + e$$

- Average delay is independent of m (better than SFDM)

CSMA Aloha

- In many systems, a node can hear whether other nodes are transmitting, thus the name Carrier Sensing Multiple Access (CSMA)
- This can happen after a time equal to the propagation delay of the channel, say $\beta << 1$ slot
- Therefore, the feedback **0** (for idle) is obtained after a slot of length β (rather than at the end of a complete slot)

CSMA
  sense the channel for an idle slot
  if idle after $\beta << 1$
    then transmit your packet with probability p (Aloha)
      CSMA
    else continue the current slot (busy slot)
      CSMA

- A busy slot has length 1
- An idle slot has length $\beta << 1$
- A busy slot is always followed by an idle slot
- An idle slot is followed by either an idle slot or a busy slot

Expected slot length

The length of a slot is

- β with probability $(1-p)^n \approx e^{-g(n)}$ (no transmission)
- $1+\beta$ with probability $1-e^{-g(n)}$
- Therefore, the expected slot length is

$$\beta e^{-g(n)} + (1+\beta)(1-e^{-g(n)}) = \beta + 1 - e^{-g(n)}$$

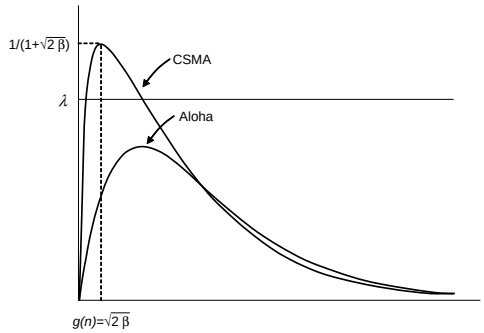
- The expected number of arrivals per slot is

$$m(1-e^{-\frac{\lambda}{m}[\beta+1+e^{-g(n)}]}) \approx \lambda(\beta+1-e^{-g(n)}) \quad (\text{if } m >> \lambda)$$

- Therefore, for stability we need

$$\lambda < \frac{g(n)e^{-g(n)}}{\beta+1-e^{-g(n)}}$$

Stability of CSMA



- The maximum throughput is $\frac{1}{1+\sqrt{2\beta}}$, when $g(n) \approx \sqrt{2\beta}$
- CSMA can be stabilized using the same technique of estimating n described previously for pure Aloha (but a bit more involved)

CSMA/CD Aloha $\approx$ Ethernet

- In addition to carrier sensing, it may be possible for a node to listen to the channel while transmitting
- If two nodes transmit almost simultaneously, they can detect a collision after the propagation delay of the channel $\beta << 1$ slot
- This justifies the name CSMA with Collision Detection, CSMA/CD, e.g. the Ethernet
- Therefore, a **e** feedback is obtained after two slots of length $\beta << 1$ each (rather than at the end of a complete slot)
- A **1** feedback is now implicit

CSMA/CD
  sense the channel
  if idle after $\beta << 1$ (idle slot)
    then transmit your packet with probability p (Aloha)
      CSMA/CD
    else if collision after $2\beta << 1$ (collision slot)
      then CSMA/CD
      else continue the current slot (busy slot)
        CSMA/CD

Expected slot length

The length of a slot is

- β with probability $(1-p)^n \approx e^{-g(n)}$ (no transmission)
- $1+\beta$ with probability $g(n)e^{g(n)}$ (success)
- 2β with probability $1 - g(n)e^{-g(n)} - e^{-g(n)}$ (collision)
- Therefore, the expected slot length is

$$\beta e^{-g(n)} + (1+\beta)(g(n)e^{-g(n)}) + 2\beta(1 - g(n)e^{-g(n)} - e^{-g(n)})$$

$$= \beta + g(n)e^{-g(n)} + \beta(1 - e^{-g(n)} - g(n)e^{-g(n)})$$

- The expected number of arrivals per slot is

$$m[\beta + g(n)e^{-g(n)} + \beta(1 - e^{-g(n)} - g(n)e^{-g(n)})]$$

- Therefore, for stability we need

$$\lambda < \frac{g(n)e^{-g(n)}}{\beta + g(n)e^{-g(n)} + \beta(1 - e^{-g(n)} - g(n)e^{-g(n)})}$$

Throughput of CSMA/CD

$$\frac{g(n)e^{-g(n)}}{\beta + g(n)e^{-g(n)} + \beta(1 - e^{-g(n)} - g(n)e^{-g(n)})}$$

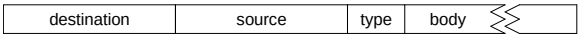
is maximized when $g(n) \approx 0.77$ resulting in

$$\lambda < \frac{1}{1 + 3.31\beta}$$

- Better than CSMA if $3.31\beta < \sqrt{2\beta} \Rightarrow \beta < 0.183$ (e.g. in Ethernet, $\beta \approx 0.021$)
- Packets may be of different lengths, as long as the length is multiple of β
- The analysis of CSMA did not account for variable length packets
  - one can show that the expected length of a collison slot is relatively large
  - must wait until all packets finish to detect a collision
  - with variable size, this is more than just one slot

Ethernet

- Ethernet is a CSMA/CD network similar to the CSMA/CD Aloha
- Most famous network over the past 20 years
- Developed in 1970s by Xeros Palo Alto Research
- Frame format
  - 6 byte destination address
  - 6 byte source address
  - 2 byte type field (length field in IEEE 802.5)
  - $46 <$ bytes of data < 1500 (we will see why)
  - the channel adds a preamble of up to 64 bytes, and a 4 byte CRC



- The address of an Ethernet node is hard coded into the Ethernet network adaptor (network card), e.g. 8:0:2b:e4:b1:2

Ethernet algorithm

- Transmitter algorithm: variant of CSMA/CD Aloha, but unslotted
  - after a collision wait for some time using exponential backoff, i.e. doubling delay
  - 1^{st} attempt: immediate after sensing
  - 2^{nd} attempt: wait 0 or 2β uniformly, then start sensing
  - 3^{rd} attempt: wait 0, 2β, 4β, 6β uniformly, then start sensing
  - k^{th} attempt: wait $2k\beta$, k chosen uniformly in $[0, 2^{k-1} - 1]$, then start sensing
- Receiver algorithm: Ethernet adaptor receives all frames (CSMA/CD) and accepts
  - frames addressed to its own address
  - frames addressed to the broadcast address (all 1's)
  - frames address to a multicast address (1 followed by something) if configured to accept that
  - all frames if it has been placed in promiscuous mode

Why 2β ?

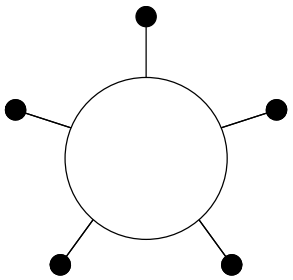
- With an unslotted system, collision requires a time of 2β (i.e. RTT) after transmission to be detected
  - Suppose A begins transmitting a frame at time t
  - The first bit of A's frame arrives at B at $t + \beta$
  - An instant before, B still sees an idle channel, and begins to transmit
  - B's frame will immediately collide with A's frame, and B will stop transmission immediately
  - But B sends a special frame (runt frame), which will reach A at time $t + 2\beta$
  - This is when A detects the collision
- As a result, A must transmit for 2β to be sure to detect all collisions
- On Ethernet, 2β is 51.2 μs (experimentally)
  - at most 4 repeaters connecting 5 segments (CSMA/CD lines or buses)
  - each segment is at most 500 m
- At 10 Mbps, this corresponds to 512 bits, or 64 bytes
- 14 bytes (header) + body + (4 bytes) CRC $>= 64 \Rightarrow$ body $>= 46$ bytes

Ethernet technologies

coax	10Base5	500 m
thin coax	10Base2	200 m
twisted pair	10BaseT	100 m

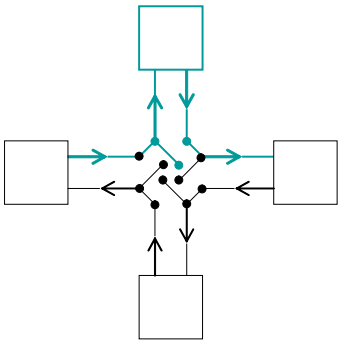
- 10 Mbps (twisted pair can support 100 Mpbs and 1000 Mbps)
- at most 4 repeaters between any two nodes (10Base5)
- up to 1024 nodes

Token Ring



- Nodes are on a ring and data flows in one direction
- Like Ethernet, the ring is a shared medium (not a collection of point-to-point links
  - all nodes see all frames
  - must control when each node can transmit
- A token circulates
- A node with a frame holds the token and transmits its frame instead
- The node then releases the token
- Round robin service, fair if all framess have the same size

Ring as a Star



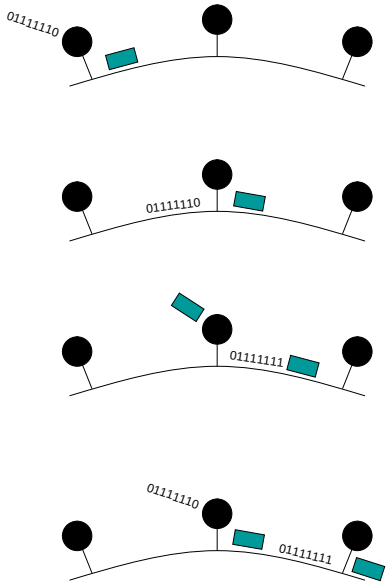
- 4 Mbps or 16 Mbps
- $\approx$ 250 nodes
- twisted pair

- A node failure would render the whole ring useless
- Each node is connected into the ring using an electromagnetic relay
  - if the node is up, the relay is open
  - if the node is down, the relay closes and the ring bypasses the node
- Several relays are grouped together in a box making the ring look like a star

Token implementation

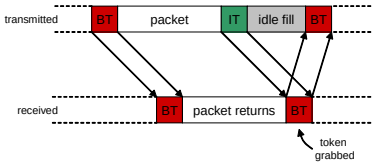
- The ring represents a continuous bit stream
- A node
  - either relays the bit stream received from the previous node, with at least a one bit delay
  - or transmits its own frame and discards what is being received
    - must make sure that what is being received and discarded has already reached its destination
  - or simply transmits idle fill and discards what is being received
    - e.g. the node can be receiving its own frame if it is long enough
- Token: let token denote beginning or end of frame (e.g. flag 01111110 if bit framing is used)
  - when a node finishes transmisson, it places 01111110 (idle token) at the end of the frame (as usual)
  - when next node reads the token
    - it simply passes the token if it has no frame to send
    - or converts it to 01111111 (busy token) and follows it by its frame and the idle token

Idle token and busy token

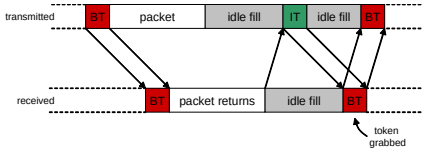


Releasing the token

- When a node inserts its frame in the bit stream it can
  - (early release) either transmit the IT immediately, followed by idle fill until BT is seen



  - (delayed release) or transmit idle fill until packet completely returns, then IT, followed by idle fill until BT is seen

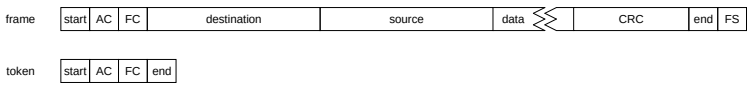


- BT is always followed by a frame
- IT is always followed by enough idle fill to make up the round trip

Issues to be considered

- Token length: round trip delay must be at least as large as token length; otherwise, a node will discard part of the token as it returns
  - nodes may add delay (not just one bit) $\Rightarrow$ nodes must have buffers
  - i.e. ring must have enough storage to hold the token
- Release strategy: when should a node release the token?
  - early release: better utilization of the ring
  - delayed release: better ARQ
    - destination node can set a special status bit in the frame
    - a node can check the status of a frame when it returns and decide whether to retransmit or not
- Priority: priority field can be added in a fixed position after the (idle or busy) token (or as part of the token)
  - a node with high priority traffic would set the field
  - a node with a low priority traffic relays the idle token instead of using it
  - when a node transmits its high priority frames, it must reset the priority field

Frame format



- start delimiter: 8 bit start of frame (or token)
- AC: 8 bit access control, for setting priority
- FC: 8 bit frame control, distinguishes between data frames and token
  - when a nodes reads the token (idle token), it can simply change the frame control field and then add the rest of its frame
  - what we previosuly called busy token is now just the beginning of the frame header
- destination: 48 bit destination address
- source: 48 bit source address
- CRC: 32 bit CRC
- end delimiter: 8 bit end of frame (or token)
- FS: 8 bit frame status
  - the A bit: destination node sets this when it sees the frame
  - the C bit: destination node sets this if it successfully stores the frame

FDDI

- FDDI stands for Fiber Distributed Data Interface
- 100 Mbps token ring using fiber optics with the following two main features:
  - nodes use early release to increase network throughput
  - handling priority: high priority traffic receives guaranteed throughput and delay
- FDDI does not use priority reservation as described earlier (therefore, no AC field in the frame format)
- When ring is initialized, a parameter $TTRT$ (Target Token Rotation Time) is set
- Each node i is allocated a time α_i to send its high priority traffic, $\sum_j \alpha_j \leq TTRT$
  - node i times the interval $\Delta_i t$ between successive idle token arrivals
  - node i can send high priority traffic for a time α_i
  - node i can send low priority traffic for a time $\max(0, TTRT - \Delta t_i)$
  - when congestion builds up (token is late $\Delta_i t > TTRT$), only high priority traffic is sent
- FYI: Maximum frame size is 4500 bytes

Analysis of FDDI

- Let $T = \sum_{j=0}^{m-1} \alpha_j \leq TTRT$
- Let t_i be the time at which the token reaches node $i \bmod m$ for the $(\lfloor i/m \rfloor + 1)^{th}$ time
- Based on the FDDI algorithm:

$$t_{i+1} \leq \begin{cases} t_{i-m} + TTRT + \alpha_i & \Delta t_i = t_i - t_{i-m} < TTRT \\ t_i + \alpha_i & \Delta t_i = t_i - t_{i-m} \geq TTRT \end{cases}$$

$$t_{i+1} \leq \max(t_i, t_{i-m} + TTRT) + \alpha_i$$

$$t_{i+1} - \sum_{j=0}^{i} \alpha_j \leq \max(t_i - \sum_{j=0}^{i-1} \alpha_j, t_{i-m} - \sum_{j=0}^{i-m-1} \alpha_j + TTRT - \sum_{j=i-m}^{i-1} \alpha_j)$$

But $\sum_{j=i-m}^{i-1} \alpha_j = \sum_{j=0}^{m-1} \alpha_j = T$, let $t'_i = t_i - \sum_{j=0}^{i-1} \forall i \geq 0$

$$t'_{i+1} \leq \max(t'_i, t'_{i-m} + TTRT - T)$$

Analysis of FDDI (cont.)

Also, assuming each node always has high priority traffic,

$$t_i \geq t_{i+j-m} + \sum_{k=i+j-m}^{i-1} \alpha_k, j \leq m$$

We can show that $t'_i \geq t'_{i-j+m}$ for $j \leq m$

$$t'_i = t_i - \sum_{k=0}^{i-1} \alpha_k \geq t_{i+j-m} + \sum_{k=i+j-m}^{i-1} \alpha_k - \sum_{k=0}^{i-1} \alpha_k$$

$$t'_i \geq t_{i+j-m} - \sum_{k=0}^{i+j+m-1} \alpha_k - \sum_{k=i+j-m}^{i-1} \alpha_k + \sum_{k=i+j-m}^{i-1} \alpha_k$$

$$t'_i \geq t'_{i+j-m}$$

Analysis of FDDI (cont.)

$$t'_{i+1} \leq \max(t'_i, t'_{i-m} + TTRT - T)$$

$$t'_i \geq t'_{i+j-m}, j \leq m$$

By induction on $i \geq m$, we can show that

$$t'_{i+m+1} \leq t'_i + TTRT - T$$

Example: consider $m = 3$, the following tables show the times the token arrives to the nodes (assuming each node i holds the token as long as possible, i.e. $\max(0, TTRT - \Delta t_i) + \alpha_i$)

t

0	1	2
0 *	$TTRT + \alpha_0$	$TTRT + \alpha_0 + \alpha_1$
$TTRT + T$	$TTRT + T + \alpha_0$ *	$2TTRT + \alpha_0 + \alpha_1$
$2TTRT + T$	$2TTRT + T + \alpha_0$	$2TTRT + T + \alpha_0 + \alpha_1$ *
$3TTRT + T$	$3TTRT + T + \alpha_0$	$3TTRT + T + \alpha_0 + \alpha_1$
$3TTRT + 2T$ *	$4TTRT + T + \alpha_0$	$4TTRT + T + \alpha_0 + \alpha_1$

t'

0	1	2
0 *	$TTRT$	$TTRT$
$TTRT$	$TTRT$ *	$2TTRT - T$
$2TTRT - T$	$2TTRT - T$	$2TTRT - T$ *
$3TTRT - 2T$	$3TTRT - 2T$	$3TTRT - 2T$
$3TTRT - 2T$ *	$4TTRT - 3T$	$4TTRT - 3T$

(* indicates low priority traffic)

Analysis of FDDI (cont.)

- (Average rotation time) Consider token makes $m + 1$ cycles around the ring

$$t'_{i+m(m+1)} \leq t'_i + m(TTRT - T)$$

$$t_{i+m(m+1)} - \sum_{j=0}^{i+m(m+1)-1} \alpha_j - t_i + \sum_{j=0}^{i-1} \alpha_j \leq m(TTRT - T)$$

$$t_{i+m(m+1)} - t_i \leq m(TTRT - T) + (m+1)T$$

$$\frac{t_{i+m(m+1)} - t_i}{m+1} \leq \frac{m}{m+1} TTRT + \frac{T}{m+1} \leq TTRT$$

- (Low priority traffic) For each of the $m + 1$ cycles, each node i can send low priority traffic for a at least $TTRT - \Delta t_i$; therefore, each node is given at least

$$(m+1)TTRT - [m(TTRT - T) + (m+1)T] = TTRT - T$$

  for its low priority traffic, an average time of $\frac{TTRT-T}{m+1}$ per cycle

- (fairness) The time for low priority traffic per cycle is

$$\frac{m}{m+1} TTRT + \frac{T}{m+1} - T = \frac{m(TTRT - T)}{m+1}$$