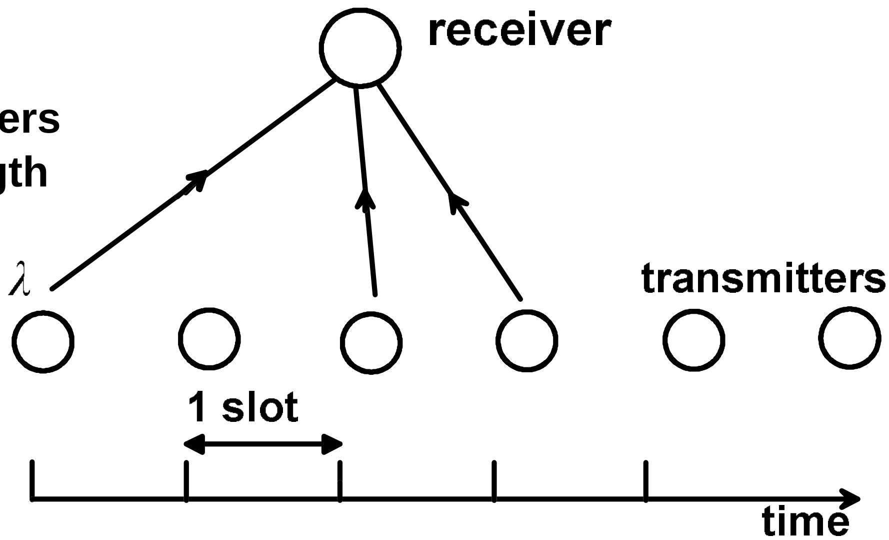
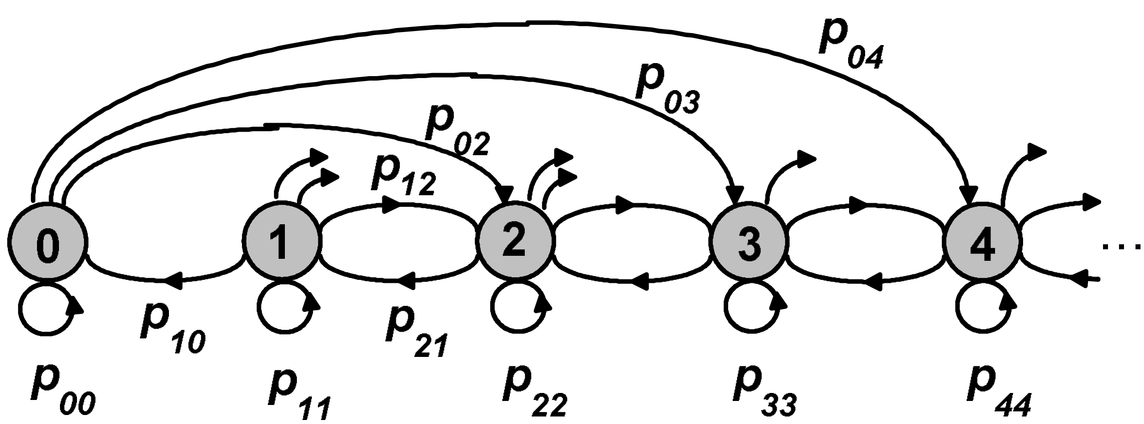
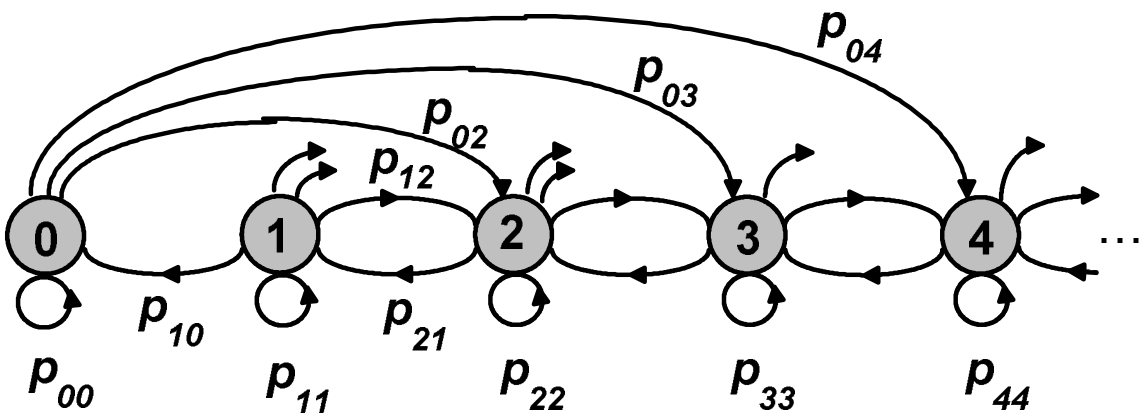
Slotted Aloha



Recall assumptions:
- infinite # of transmitters
- packets of equal length
- immediate feedback
- Poisson arrivals rate λ

If a new packet arrives during a slot, transmit in next slot .
If a transmission has collision, node becomes backlogged.
While backlogged, transmit in each slot with probability q_r until successful (i.e. not backlogged)
State (n) of system is number of backlogged nodes.
State of system can be modeled by Markov chain with transition probability P_{ij}





$P_{l,l-1} =$ **prob. of one backlogged attempt and no new arrival**

$$= l \cdot q_r(1-q_r)^{l-1} \cdot e^{-\lambda} \ , \ l > 0$$

$P_{l,l} =$ **prob. of one new arrival and no backlogged attempts or no new arrival and no success**

$$= (1-q_r)^l \cdot \lambda e^{-\lambda} + [1 - lq_r(1-q_r)^{l-1}] \cdot e^{-\lambda} \ , \ l > 0$$

$P_{l,l+1} =$ **prob. of one new arrival and one or more backlogged attemp**

$$= [1-(1-q_r)^l] \cdot \lambda e^{-\lambda}, l > 0$$

$P_{l,l+j} =$ **prob. of** $j > 1$ **new arrivals** $= \frac{\lambda^j e^{-\lambda}}{j!}, \ j > 1$

Steady state probabilities do not exist!
Problem is that $P_{l,l-1} \to 0$ **as** $l \to \infty$ **.**
That is, if state *l* **gets large, it tends to get larger and larger and never return to** *0***.**

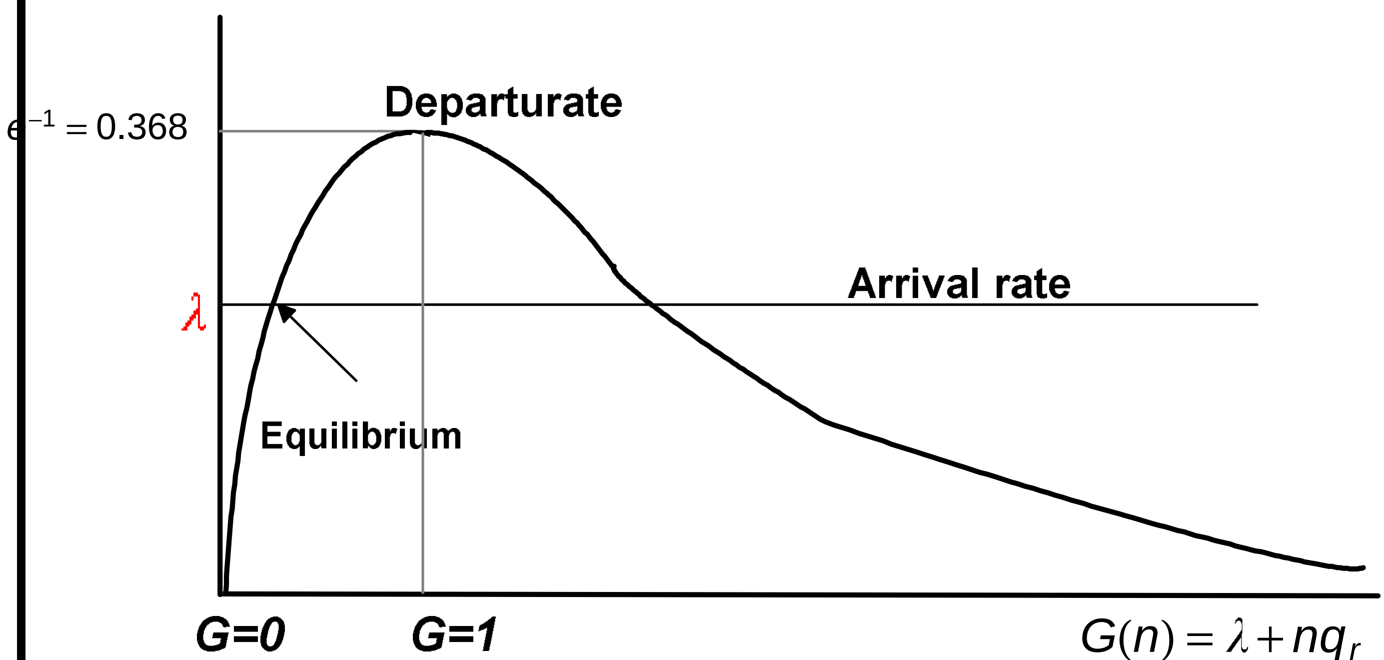
Another (slightly more approximate) Approach

Let *G(n)* **be the attempt rate (the expected number of packets transmitted in a slot) in state** *n***:**

$$G(n) = \lambda + nq_r$$

The number of attempted packets per slot in state *n* **is approximately a Poisson r.v of mean** *G(n)*

$$P_{\text{success}} = G(n) \cdot e^{-G(n)}$$ **= departure rate instate** *n*



If backlog increases beyond unstable point, then tends to increase without limit.
Choosing q_r **small increases the backlog at which instability accrue (since** $G(n) = \lambda + nq_r$**), but increases delay (since mean retry time is** $\frac{1}{q_r}$**)**
$G(n) = \lambda + nq_r = 1$ **results in highest** P_{success} **(but we don't know n).**

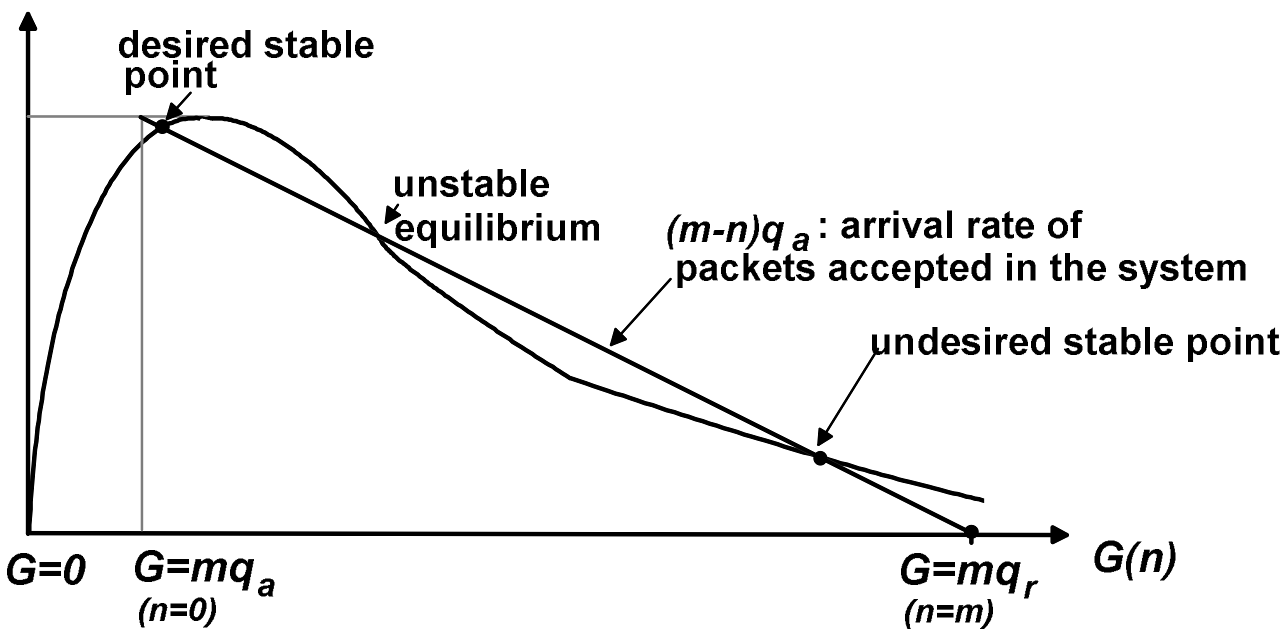
Slotted Aloha-case where there are *m* users

Backlogged users cannot accept new packets

q_a = **prob. of new arrival at a node during a slot**
q_r = **prob. of retransmission of a backlogged node**
n = **number of backlogged nodes (state)**

Attempt rate = $G(n) = (m-n)q_a + nq_r$

If $q_a, q_r << 1,$ **then** $P_{success} \approx G(n) \cdot e^{-G(n)}$



At undesired stable point, throughput is small and most new packets are discarded.

Pure Aloha (unslotted)

New arrivals are transmitted immediately (no slots)

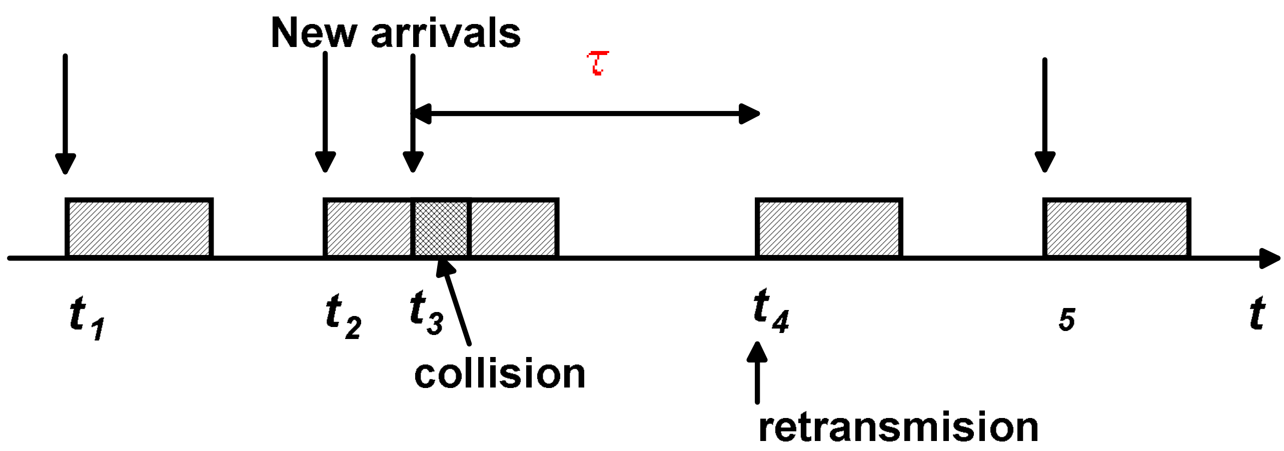
Packets may have variable lengths; however for the analysis we will assume that all packet lengths are equal and require 1 unit of time to be transmitted.

A backlogged packet is retried at a random time τ of density $x \cdot e^{-\tau x}$ (x = retransmission rate)

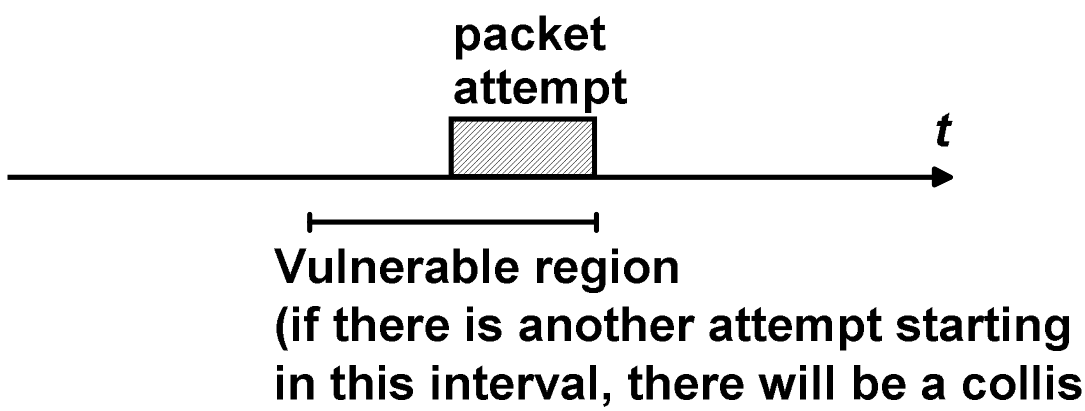
The times at which transmissions start is a time-varying Poisson process of rate

$$G(n) = \lambda + nx$$

Where n = # of backlogged nodes



An attempt suffers a collision if the previous attempt is not yet finished, or if the next attempt starts too soon.

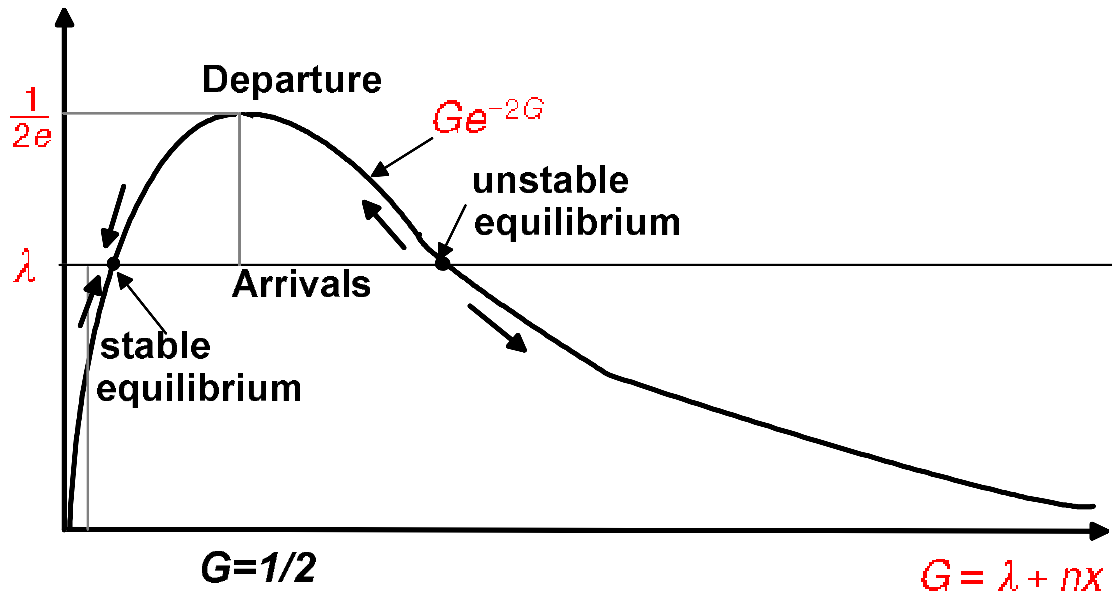


P_r (**no other attempt in 2 units of time**) $= e^{-2G(n)}$

Throughput (# of successful transmissions per slot)

$$= G(n) \cdot e^{-2G(n)}$$

Max throughput = $\frac{1}{2e} (\approx 0.18)$ **at** $G(n) = \frac{1}{2}$



Pure (unslotted) Aloha with
Infinite number of users ($m = \infty$)

Stabilization issues are similar to slotted Aloha

Advantages of pure Aloha are simplicity and possibility of unequal lengths of packets.

Splitting algorithms

A more efficient way to use the idle/success/ collision feedback

Assume only two packets are involved in a collision.

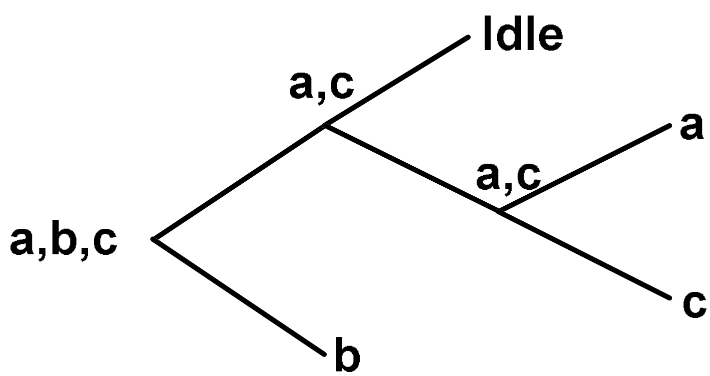
Suppose all other nodes remain quiet unit collision is resolved, and nodes in the collision each transmit with probability ½ until one successful. On the next slot after this success, the other node transmits.

The expected number of slots for the first successful attempt is 2, so the expected number of slots to transmit both packets is 3 slots.

Tree Algorithms

After a collision all new arrivals and all backlogged packets not in the collision wait.

Each collision packets joints either a transmitting set or a waiting set



b waits for the resolution of the (a, c) collision.
In general, each waiting node is on a stack. The node goes down one on each collision and up one on each success or idle.

The second collision between packets a and c is unnecessary. In general, when a collision is followed by an idle, the waiting subset from the first collision should be split again.

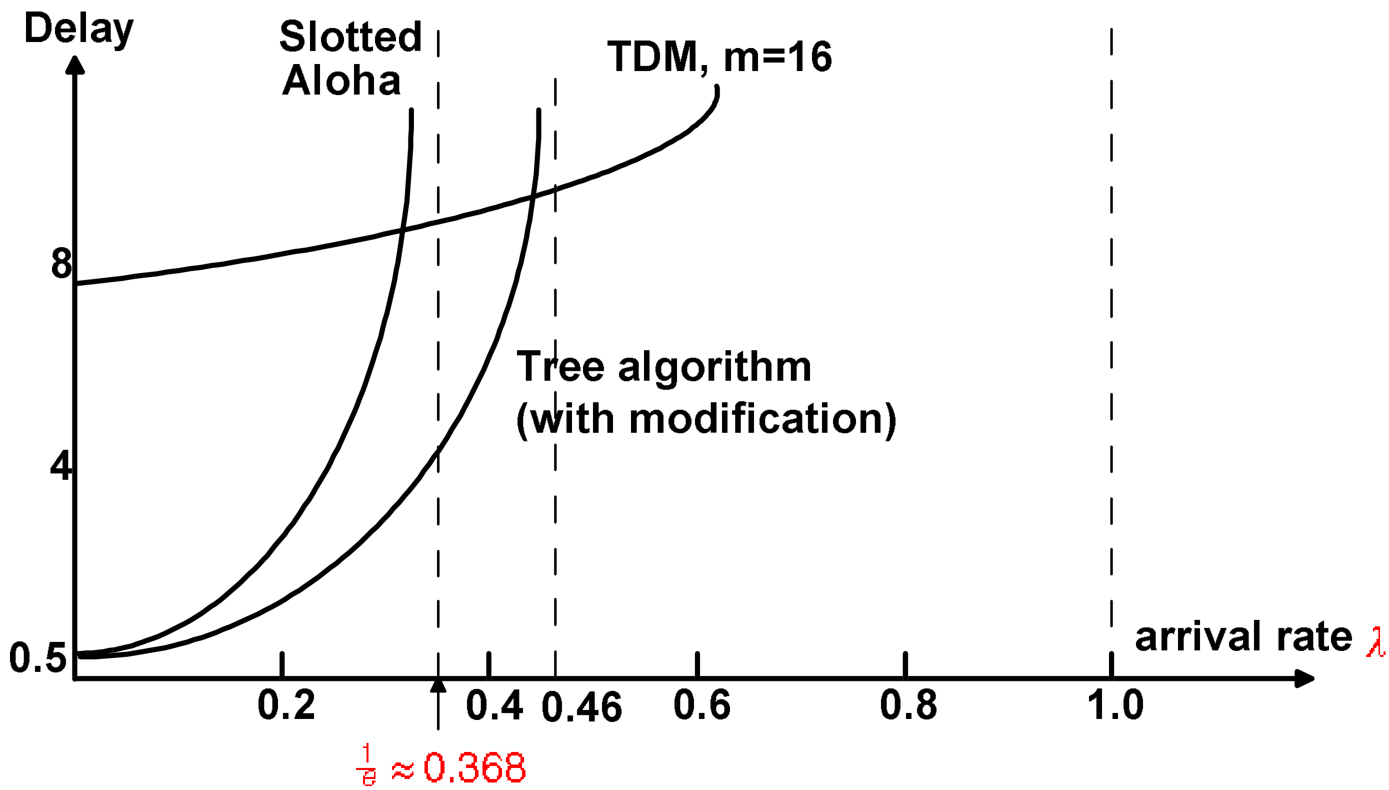
What to do after a collision is resolved?

The best splitting algorithms have a maximum throughput of 0.4878

	throughput
Stabilized Pure Aloha	**0.184** $(= \frac{1}{2e})$
Stabilized Slotted Aloha	**0.368** $(= \frac{1}{e})$
Tree algorithm (as originally described)	**0.434**
Tree algorithm (after modification)	**0.46**
Best splitting algorithm	**<u>0.4878</u>**
Upper bound (under assumptions of infinite nodes)	**0.568**

TDM can achieve throughputs up to 1 packet per slot, but the delay increases linearly with the # of nodes.

the delay for stabilized Aloha and for splitting algorithms is essentially independent of the # of nodes (for given total arrival rate)


Delay
Slotted
Aloha
TDM, m=16
8
Tree algorithm
(with modification)
4
0.5
arrival rate λ
0.2
0.4
0.46
0.6
0.8
1.0
1/e ≈ 0.368

Slotted Aloha with Carrier Sensing

(CSMA Slotted Aloha)

Assume all nodes hear each other and can determine if a channel is busy (with some delay)

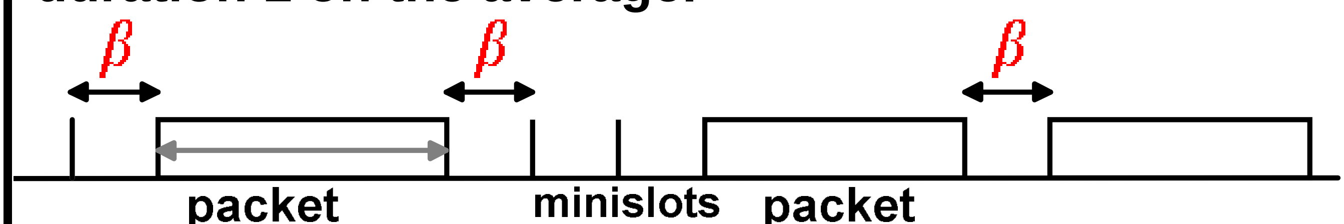
Nodes should be able to initiate a packet transmission when the line is detected idle.

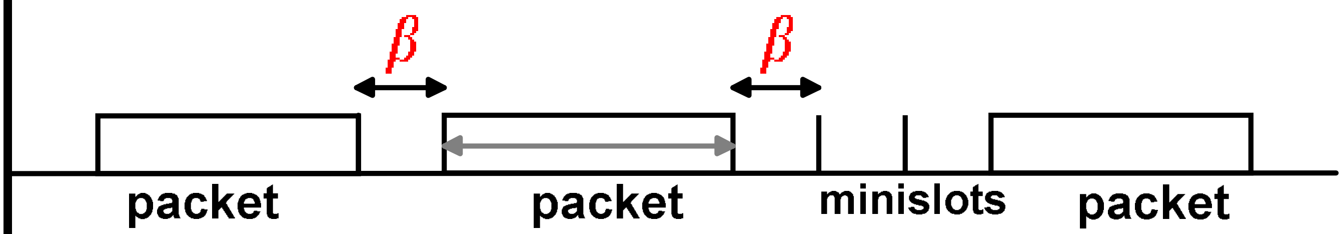
Let $\beta =$ time needed to recognize that a slot is idle (β as a fraction of a slot)

τ : propagation and detection delay (secs)
C: capacity (bits/sec)
L: average packet length in bits
($\frac{L}{C} =$1 unit of time = 1 slot)

$$\beta \approx \frac{\tau \cdot C}{L}$$

For initial understanding, view the system as slotted with idle "minislots" of duration β and packet slots of duration 1 on the average.



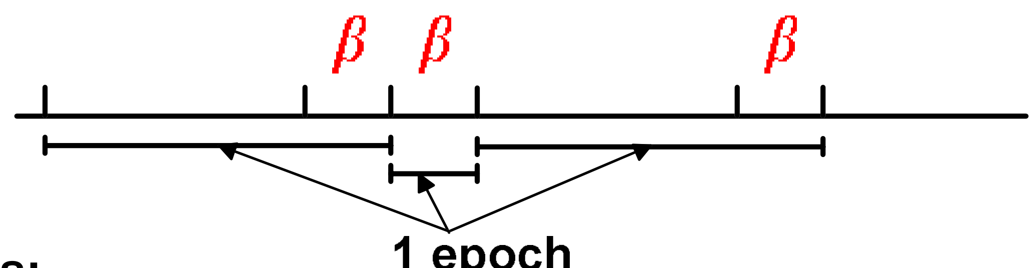


Each node with a packet listens to ensure that one minislot is idle before transmitting in the next slot.

When a node starts to transmit at the beginning of the next slot, that slot automatically becomes a unit duration slot (rather than a minislot).

Two kinds of epochs:

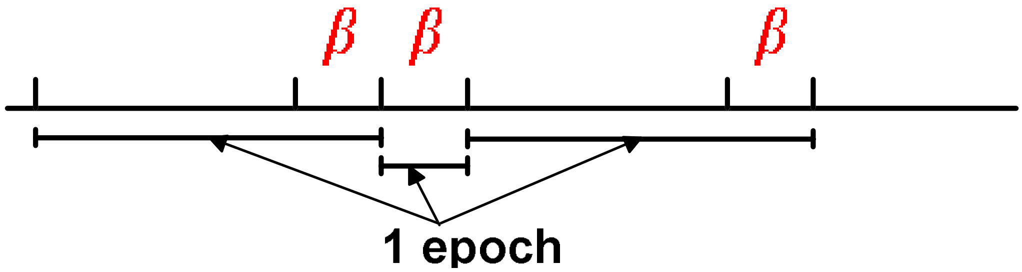
- **Full epoch consisting of a packet slot followed by a minislot (successful or collision)**
- **Mini epoch of length β (idle)**



Rules:

- **On a new arrival, if current minislot is idle, start transmission at the end of minislot**
- **On a new arrival when channel is busy (i.e., during a packet slot)**
  - ❖ **Nonpersistent CSMA: join the backlog**
  - ❖ **Persistent CSMA: transmitted after the epoch**
  - ❖ **P-persistent CSMA: transmitted after the epoch with prob. P**
- **If collision occurs, nodes involved join the backlog.**
- **While backlogged, transmit after an idle minislot with prob. q_r .**

Analysis of CSMA slotted Aloha



Let n be the # of backlogged nodes at the end of an epoch. The number of packets that attempt transmission at the next epochs

$$g(n) = \underbrace{\lambda\beta}_{\text{new packets}} + \underbrace{nq_r}_{\text{backlogged packets}}$$

(Recall: packets that arrive during a busy period are assumed backlogged for nonpersistent CSMA)

The prob. of success (per epoch) is:

$$P_{success} = (\lambda\beta + \frac{q_r n}{1-q_r})e^{-\lambda\beta}(1-q_r)^{n}$$

$$P_{success} \approx g(n) \cdot e^{-g(n)}$$ **for small q_r .**

(# of attempts can be approximated by a Poisson process with rate $g(n)$)

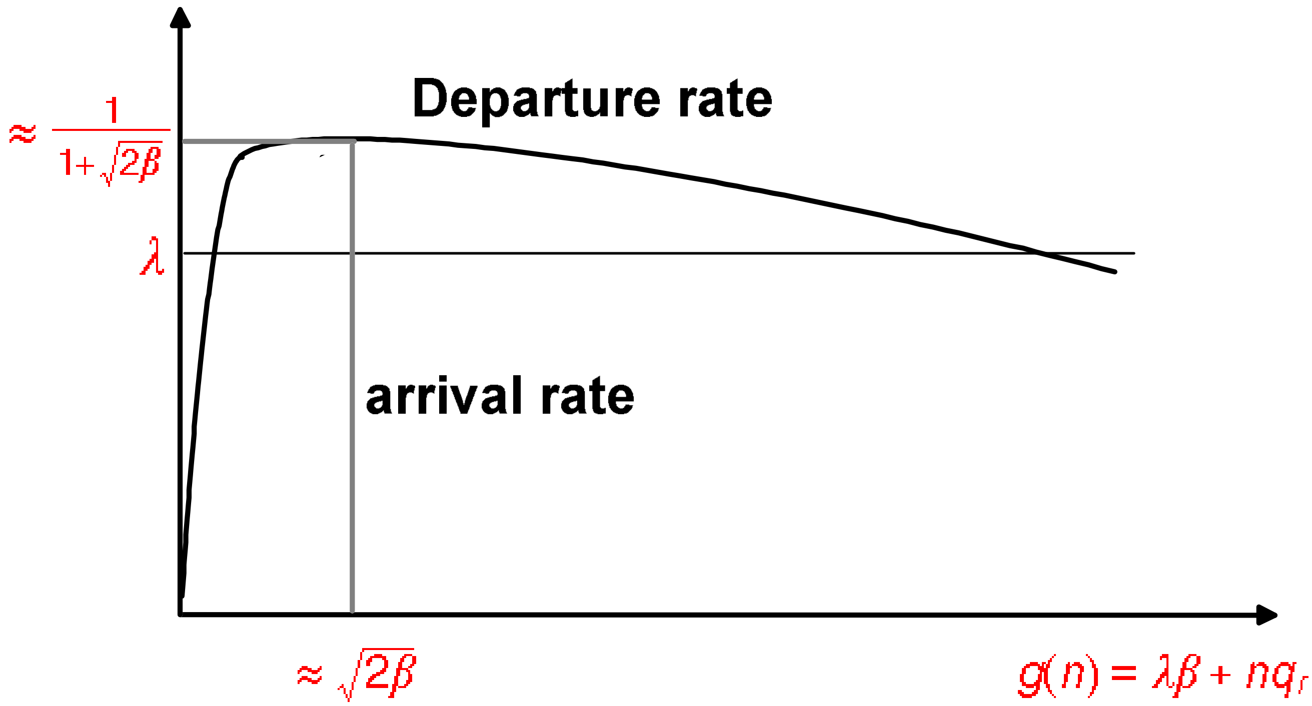
The expected duration of an epoch is:

$$\beta + 1 - e^{-\lambda\beta}(1-q_r)^n \approx \beta + 1 - e^{-g(n)}$$

Thus the success rate per unit of time is:

$$\textbf{Departure rate} \approx \frac{g(n) \cdot e^{-g(n)}}{\beta + 1 - e^{-g(n)}}$$

$$\textbf{Departure rate} \approx \frac{g(n)\cdot e^{-g(n)}}{\beta + 1 - e^{-g(n)}}$$



- **For small β, the optimal $g(n)$ is $\approx \sqrt{2\beta}$ and the corresponding departure rate is $\approx \frac{1}{1+\sqrt{2\beta}}$**
- **Note that throughputs very close to 1 are possible, and optimization is non-critical**
- **Stability problem is less serious for CSMA than for Aloha**

CSMA Unslotted Aloha

- **In CSMA slotted Aloha all nodes were synchronized to start transmission at end of a minislot.**

- **Here we assume that when a packet arrive, its transmission starts immediately if the channel is sensed to be idle.**

- **Unslotted CSMA is the natural choice for CSMA**

- **Unslotted CSMA increases the probability of a collision somewhat for the same β, causing the maximum throughput to drop from $\frac{1}{1+\sqrt{2\beta_{max}}}$ to $\frac{1}{1+2\sqrt{\beta_{eff}}}$ (for small β)**

- **Unslotted Aloha CSMA has a smaller effective value of β than slotted CSMA. (Including the average instead of maximum propagation delay.)**

- **Also the synchronization required for minislots is difficult with multiple receivers.**

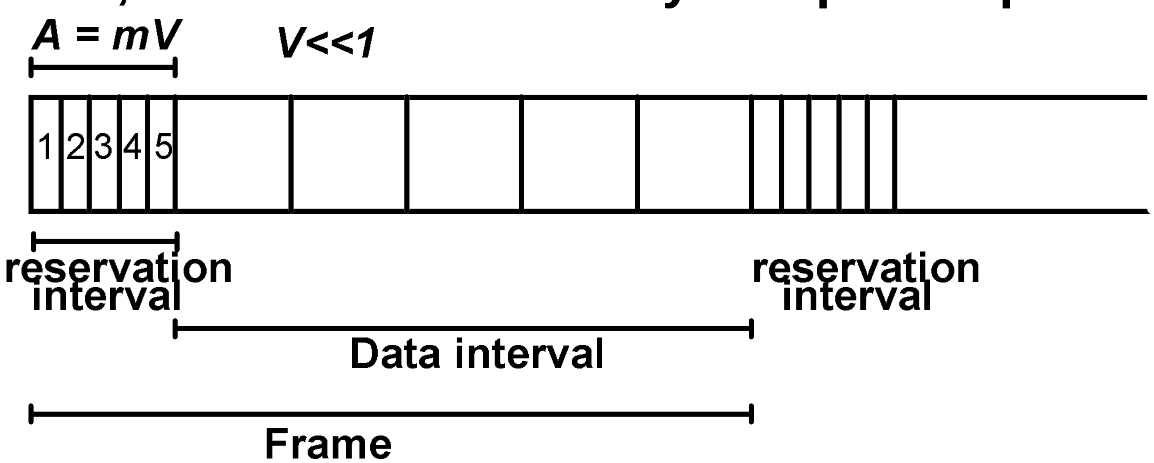
Reservation Systems

TDM wastes time when nodes have no data.

Collision resolution strategies waste time both for idle periods and collision periods

The general idea of reservation is to use mini packets to reserve the channel and then send the data at ideal efficiency.

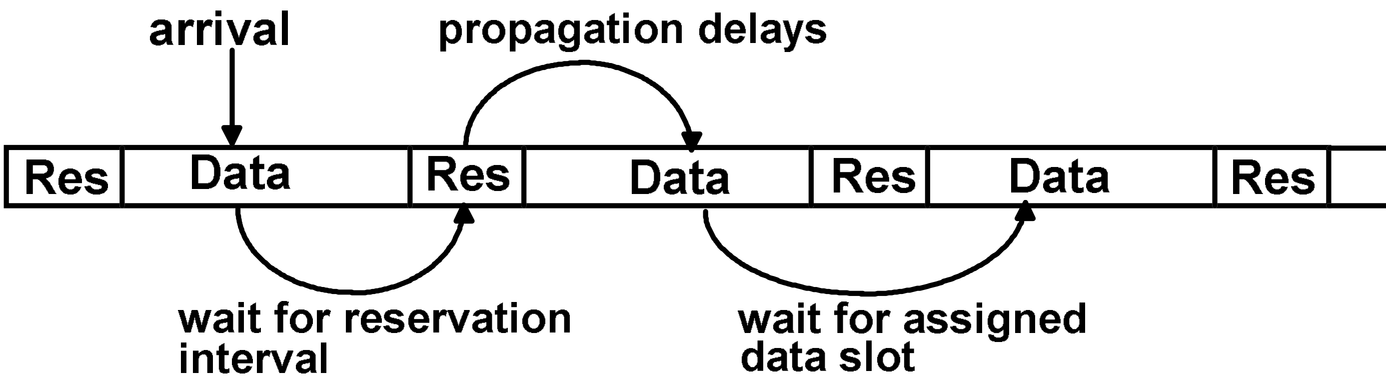
Reservations can be established by TDM, FDM or contention resolution, but once reservations are made, the channel can carry one packet per unit time.



Each one of the *m* users has his own reservation interval

Each node may reserve many data slots.

Reservation with propagation delays



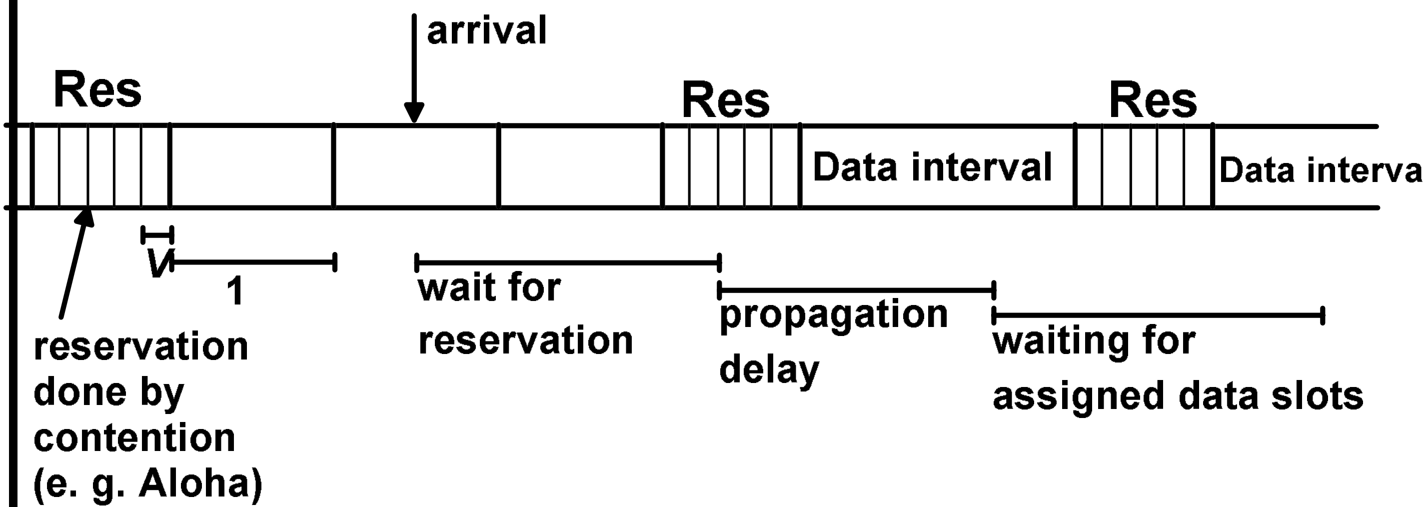
All node use same algorithm to assign slots.

Used in some satellite nets (roundtrip propagation delay $\approx \frac{1}{4}$sec)

Data interval > propagation delay

Data intervals do not have to have fixed length (although usually they do)

Reservation by Contention (example)



If a reservation packet collides, it is retried in next reservation interval.

$1 =$ **transmission time of data of a user**

$V =$ **reservation minislot**

$P_{success} =$ **success rate of contention scheme**
($= \frac{1}{e}$ for slotted Aloha)

Max. Throughput $$\frac{1}{1+\frac{V}{P_{success}}}$$

Need Data Interval > Propagation delay.

Reservation system contd.

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	
15	idle	3	20	collision	2	Frame 1
15	7	3	idle	9	2	Frame 2
idle	7	3	collision	9	idle	Frame 3
18	7	3	collision	9	6	Frame 4
18	7	3	15	9	6	Frame 5

Reservation strategy where first packet of node captures a slot in frame, and keeps the slot until finished.
Here each node sends many short packets.

First packet serves as reservation

After slot is captured it is kept until entire message transmitted.

After a slot stays idle once, other nodes may try to capture it.

Other variations:
1) Use a bit in the header of the packet to indicate whether the node has finished.
2) Each source has its own slot in the frame; when not used in one frame, other nodes may try to transmit in that.
If collision occurs, other nodes are forbidden from transmitting in slot in the next frame.

Carrier Sensing Multiple Access with Collision Detection (CSMA/CD)

- **On a bus, it is possible for nodes to listen at the same time as sending. Thus collision detection is possible after a propagation delay (which for is small local area networks.**
- **Nodes transmit when channel is detected idle.**
- **When two nodes transmit almost simultaneously, they shortly detect a collision and stop.**

The Ethernet, a popular protocol for LAN's, uses unsoltted persistent CSMA/CD with exponential backoff.

Conceptually, view the system as having minislots of duration β and full slots of duration 1.

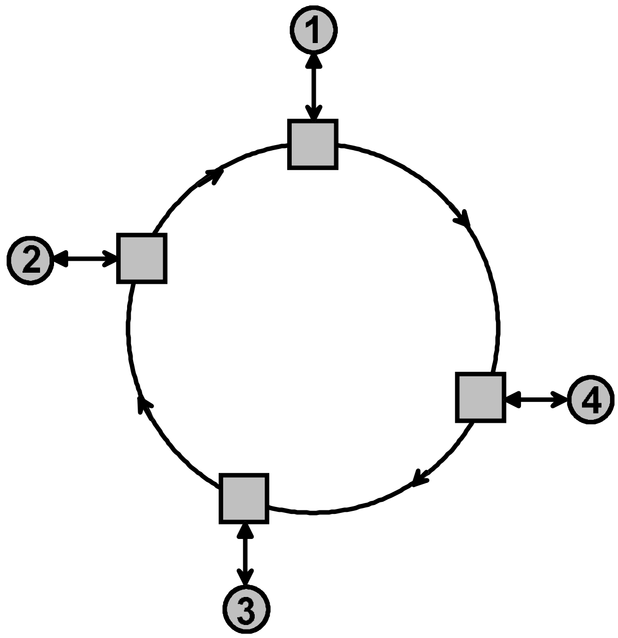
The maximum throughput is approximated

$$\boxed{\frac{1}{1+3.31\beta}}$$ **for slotted CSMA/CD**

and is upper-bounded by

$$\boxed{\frac{1}{1+6.2\beta}}$$ **for unslotted CSMA/CD**

Token Rings



Packets flow around the ring in same direction.

One node transmits and the interface units for the other nodes relay the incoming data.

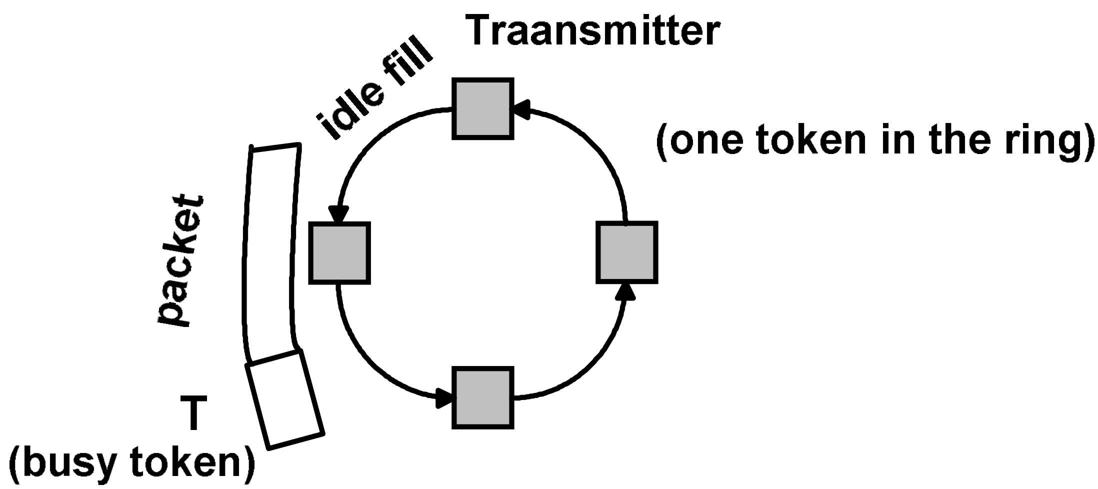
A node can transmit when it has an “idle token” (e.g. 01111110)

If a node has the idle token, it either:

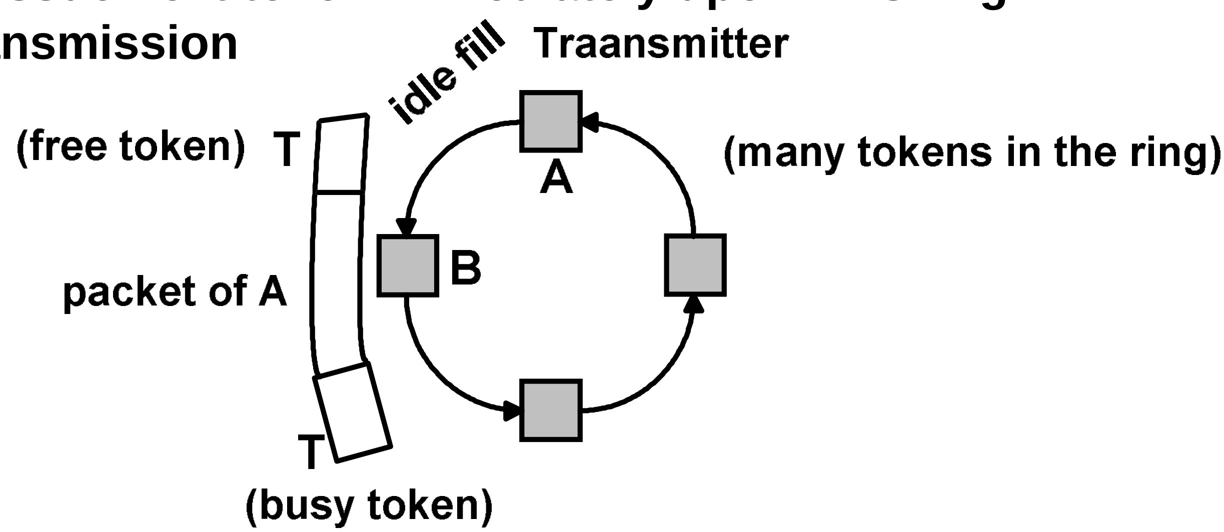
- Relay the idle token to the next node, or
- Changes the idle token to a busy token (e.g. 01111111) then sends data.

Packet is usually removed by the node that send it

Two methods:
1. A node upon finishing transmission waits till the last bit of his packet comes around and then releases token



2. Issue next token immediately upon finishing transmission



<u>Note: bit stuffing has to be used in token rings</u>

Other issues:

a) Token may be lost (or accidentally created) due to errors.

b) If node fails, he cannot forward data (bypass wire is used)

> **Max. throughput** $= \frac{1}{1+a}$

a**: % overhead for tokens and bit stuffing**

Delay Analysis: m nodes, each with rate $\frac{\lambda}{m}$**, Poisson.**

> **Exhaustive service**: $W = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \frac{(m-\rho)\overline{V}}{2(1-\rho)} + \frac{\delta_V^2}{\overline{V}}$

$V =$ **transmission time of token**

$\overline{X} =$ **average transmission time of packet token relaying delay**

$\rho = \lambda \overline{X}$