On Achieving Fairness and Efficiency in High-Speed Shared Medium Access

R. Srinivasan, Member, IEEE
Arun K, Somani, Fellow, IEEE

IEEE/ACM Trans. On Networking
Feb. 2003

Introduction

- Collision-free (e.g. TDMA)
  - Good at high load
  - Good for uniform traffic
- Collision-based (e.g. CSMA/CD)
  - Good at low load
  - Variations of Reservation ALOHA (R-ALOHA)
    - Poor utilization due to random back-off under low load
    - Unfairness due to multiple slots allowed for one station

Introduction

- To achieve well at both low and high load, collision resolution is used to dynamically allocate channel bandwidth to contending stations.
- Tree splitting
  - Upon collision, colliding stations are split into s groups, stations in group 0 are allowed to transmit, followed by stations in group 1, 2, ….
  - Upon collision in groups, second-level groups are created.
- For s=2, it is called *binary tree-splitting*.
- Probabilistic/deterministic

Evolutions

- The upper bound of using data packets for collision resolution has been shown as 0.568.
- To go beyond 0.568, control mini-slots (CMS) are proposed. (DQRAP)
  - mini-slots are small thus minimizing losses upon collisions.
  - Three-CMSs-per-slot is stable for all offered loads up to 1.
- Cable-TV protocol, IEEE 802.14, uses a ternary tree-splitting technique with p-persistence.

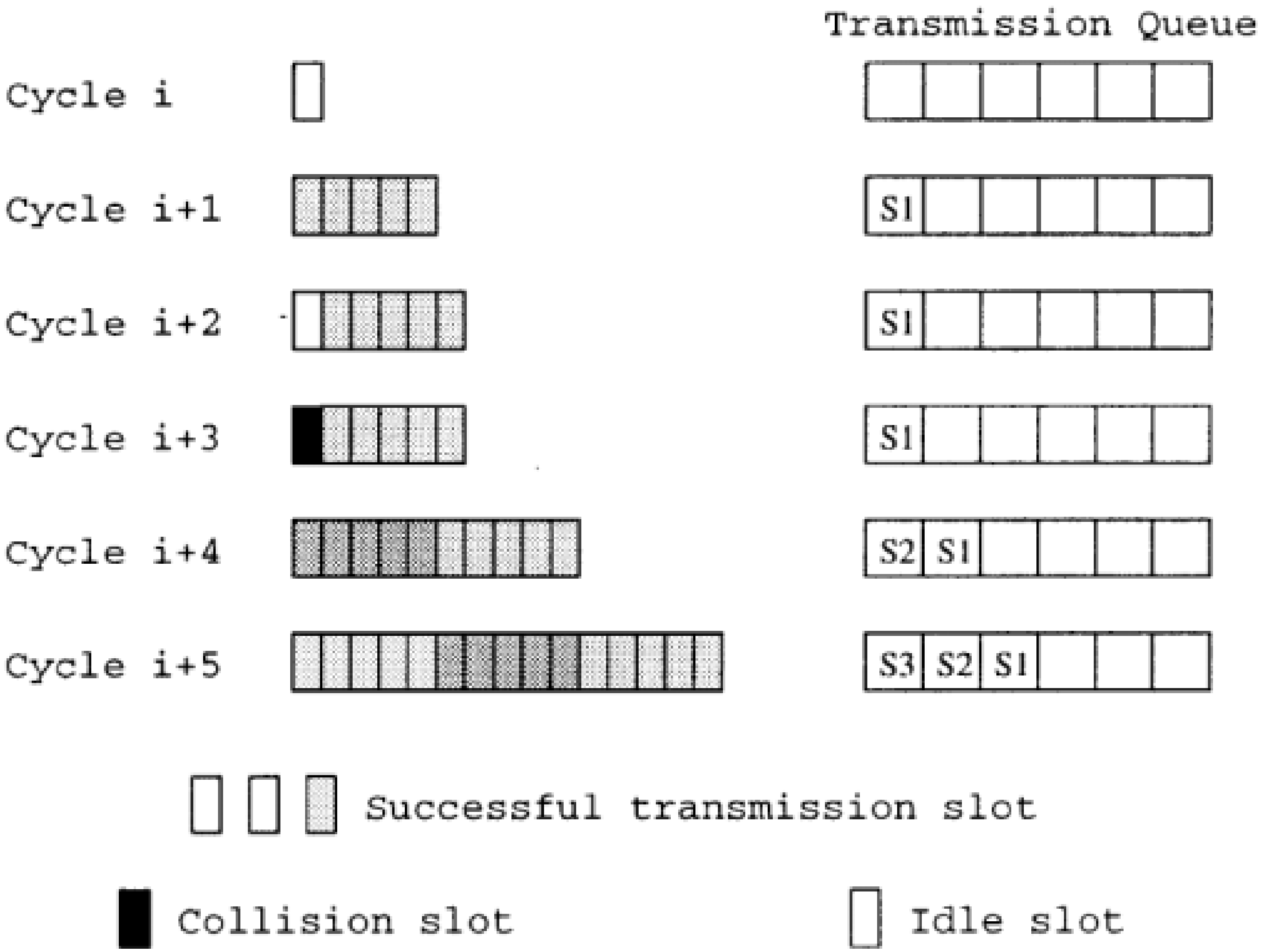
Probabilistic v.s. Deterministic

- Limitations of probabilistic approaches:
  - Poor throughput at high load
  - Unbounded packet delay
- Deterministic approaches:
  - Incremental collision resolution multiple access (ICRMA)
  - Collision resolution by small control packets
  - Increased delay for shorter messages (unfairness)

Modified ICRMA

- Channel access is divided into cycles, which have a contention slot followed by transmission slots.
- Stations having message to send transmit a (control) packet during the contention slot.
  - No collision: entering the transmission queue
  - Collision: deterministic binary tree-splitting

MICRMA


Transmission Queue
Cycle i
Cycle i+1
S1
Cycle i+2
S1
Cycle i+3
S1
Cycle i+4
S2
S1
Cycle i+5
S3
S2
S1
Successful transmission slot
Collision slot
Idle slot

Considerations

- Consider a network with RTT of 54 us, the slot duration is set 1.5RTT, 81 us:
  - If avg. message length is ten packets, with a packet size of 53 bytes, then at 1Mb/s a packet extends over approximately five slots.

    $(53x8/1M)/81us \approx 5$
  - If applied at 8Mb/s, a packet can be sent in a single slot.

Effect of Contention Slot at Low Loads

- Consider only one station has a message of length 10 packets to transmit:
  - It takes ten cycles to transmit the message and nine slots are wasted as contention slots.
  - 20% extra delay at 1Mb/s example
  - 100% extra delay if at 8Mb/s
- It is not desirable to have a contention slot for every cycle

Effect of Contention Slot at High Loads

- Under high load, a station encounters large delay contending to enter the transmission queue.
- Stations in the transmission queue can transmit packets with smaller delay, leading to a bias in favor of longer messages.
- It is necessary to ensure that no station transmit more than one packet unless all the stations that have a packet to transmit get their chance to transmit.

Effect of Transmission Queue Size

- If stations can remain in the queue as long as they have a packet to send, then the throughput is determined by the maximum queue size.
- If stations leave the queue after every message transmission, then the maximum throughput is affected by the average number of packets in a message.

Deterministic BTS Algorithm

- Physical identifier, 0<=pid<=N
- N=2^k, number of leaves of a k-level binary tree
- Virtual identifier, 0<=vid<=2^k
- Each station maintains a stack that holds identifier intervals, denoted by (vid_l, vid_h).
- The top entry is the *allowable interval* and stations lie within the interval are allowed to contend in next cycle.

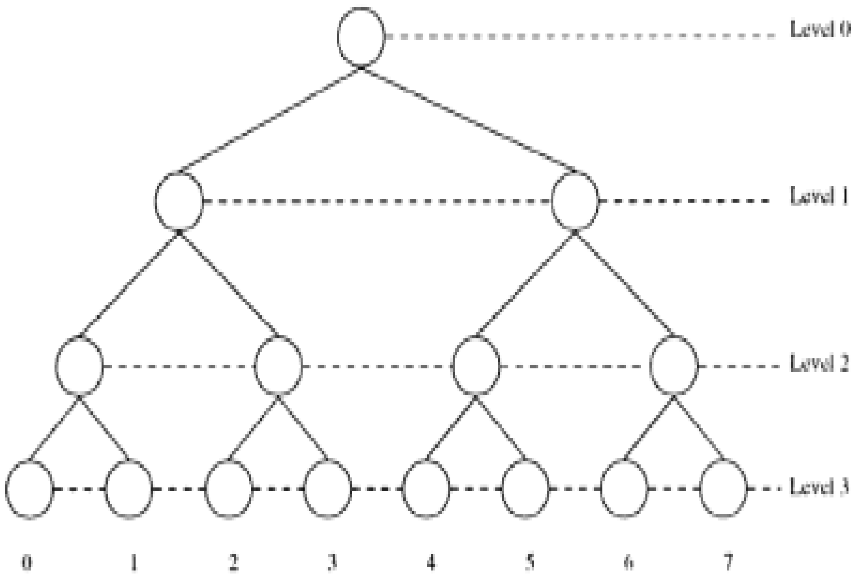
Deterministic BTS Algorithm

- Upon a collision, the allowable interval is split into two subintervals, and they replace the top entry in the stack with the higher interval on top.
- when a successful transmission or an idle slot is observed, the allowable interval is popped.
- When the stack becomes empty, the interval (0, N-1) is pushed back onto the stack (BTS cycle).

Example

Suppose that stations 1, 6, and 7 have messages to transmit. The stack is initialized with interval (0, 7). The tree-splitting algorithm proceeds as follows.

1) Stations 1, 6, and 7 transmit, resulting in a collision. The allowable interval is split into two subintervals (0, 3) and (4, 7). The allowable interval for the next contention slot is (4, 7).
2) Stations 6 and 7 transmit and collide again. The allowable interval is further split into two subintervals (4, 5) and (6, 7). The interval (6, 7) is the allowable interval for the next contention slot.
3) Stations 6 and 7 transmit and collide again, resulting in further split of the allowable interval. The stack after the end of this step has the subintervals (0, 3), (4, 5), (6, 6), and (7, 7).
4) Station 7 transmits successfully. The interval (7, 7) is popped out.
5) Station 6 transmits successfully. The interval (6, 6) is popped out.
6) The slot for the interval (4, 5) goes idle. The interval (4, 5) is popped out.
7) Station 1 transmits successfully and the interval (0, 3) is popped out. This marks the end of BTS-cycle and stack is reinitialized with interval (0, 7).



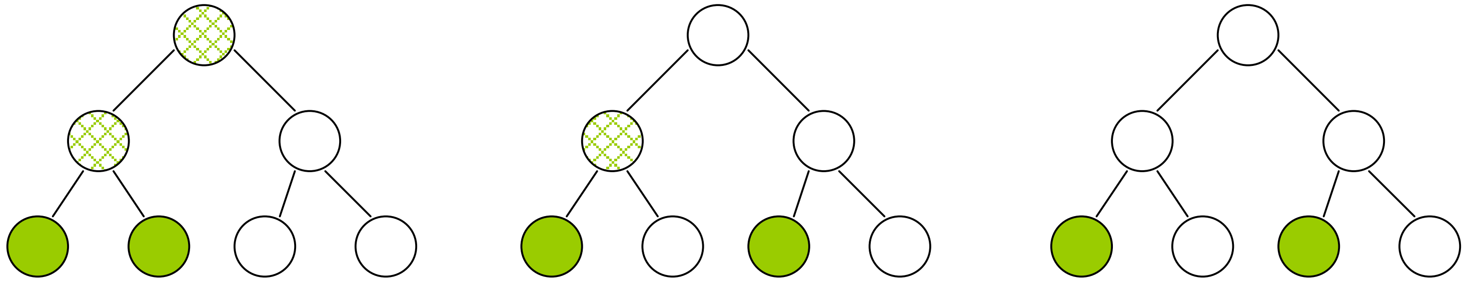
3 collision slots
3 successful transmissions
1 idle slots

Theorems

- *Theorem 1*: The minimum number of collision steps required to resolve a contention with the deterministic BTS technique involving m stations mapped uniquely to the leaf nodes of a k-level binary tree is lower bounded by (m-1).
- *Theorem 2*: Given m contending stations , there exists an arrangement on a k-level binary tree, where m<=2^k, such that the contention is resolved with exactly m-1 collisions.

Observations

- W(k, m) = (m-1) + Xe(k, m) + I(k, m)
- Possible improvements:
  - Reducing excess collision slots
  - Reducing idle contention slots
    - Arranging the stations optimally on the binary tree
  - Reducing necessary collisions
    - Starting the BTS-cycle at intermediate level



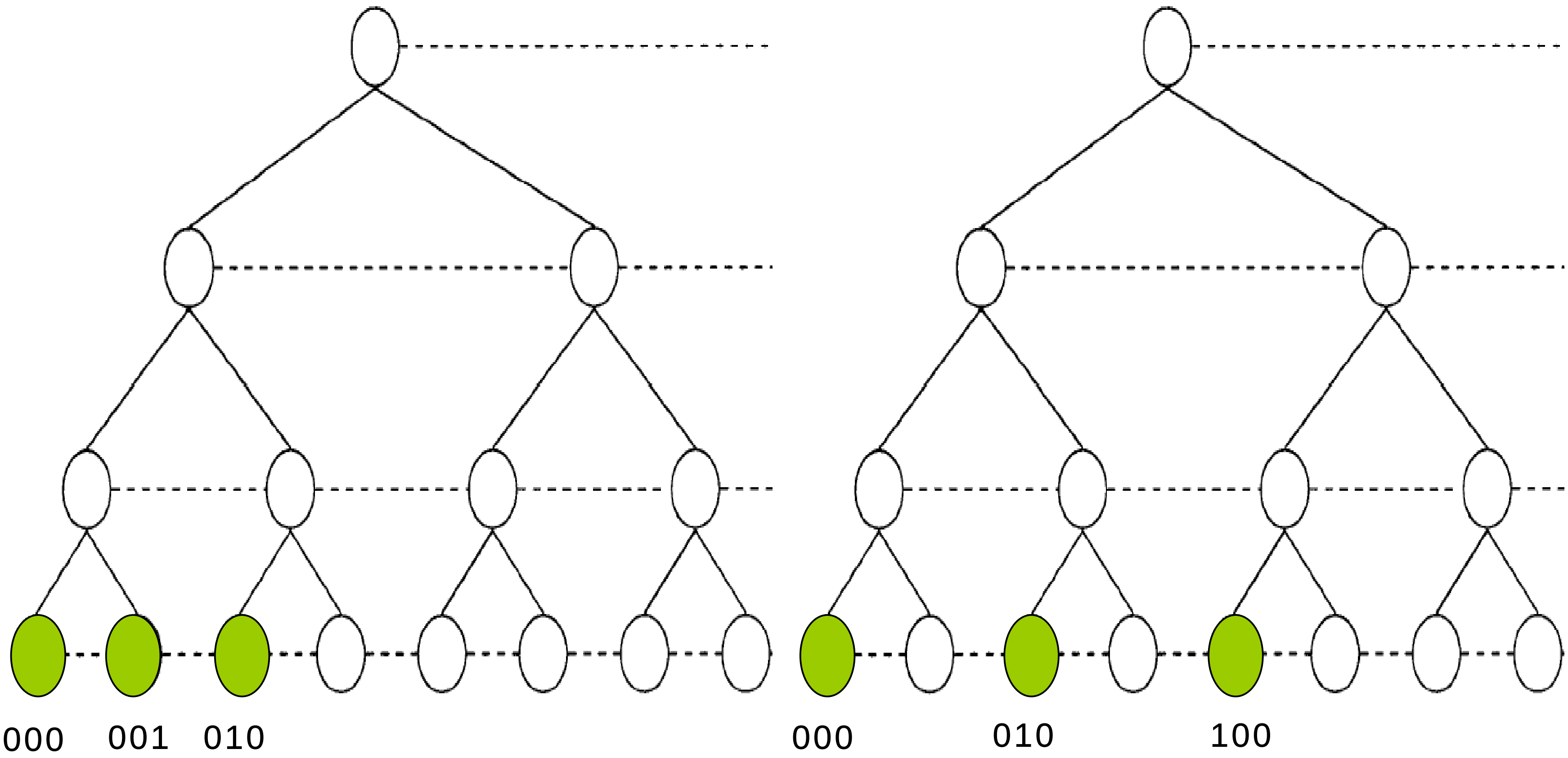
Proposed Protocol: AMES-BM

- **A**ccess **M**echanism for **E**fficient **S**haring in **B**roascast **M**edium Netowrks
- Continuing station: a station that transmits a packet successfully in the current cycle and also has a packet to transmit in the following cycle.
  - *Pkt.cont* flag / *ncont*
- Dynamic node mapping
- Dynamic grouping
- Dynamic collision resolution

Dynamic Node Mapping

- The procedure is to arrange m stations on a k-level binary tree, such that a contention involving these m stations can be resolved in exactly (m-1) steps.
- A special case of the optimal arrangement is to arrange stations such that the number of stations in the left and right sub-tree rooted by any intermediate node does not differ by more than one.
- 000->000, 001->100, 010->010

Dynamic Node Mapping

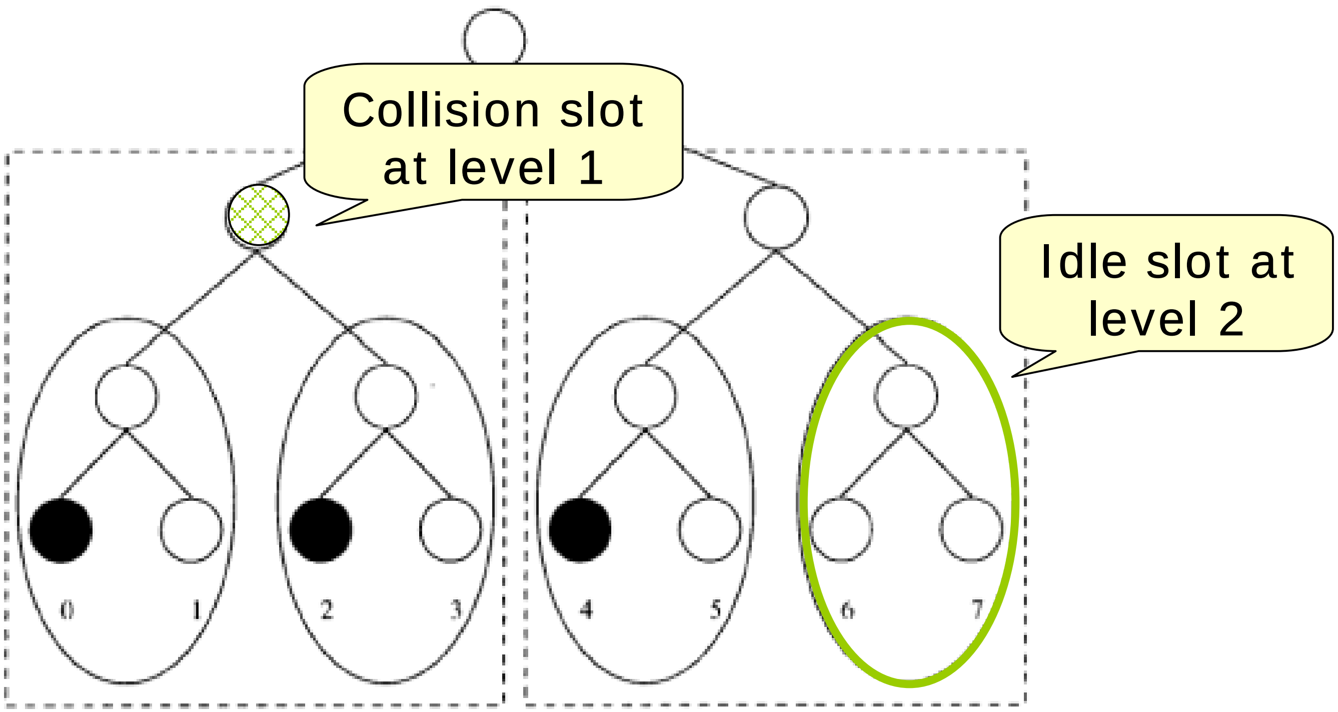


Dynamic Node Mapping

- Stations maintain *lrt* that indicates how recently a station transmitted in the current cycle. It is carried in the packet denoted as *pkt.lrt*.
  - If *pkt.cont* is set, the sending station set its *lrt* to 0, while others increase their *lrt* by 1.
  - If *pkt.cont* is not set, the *lrt* values of all stations are not updated.

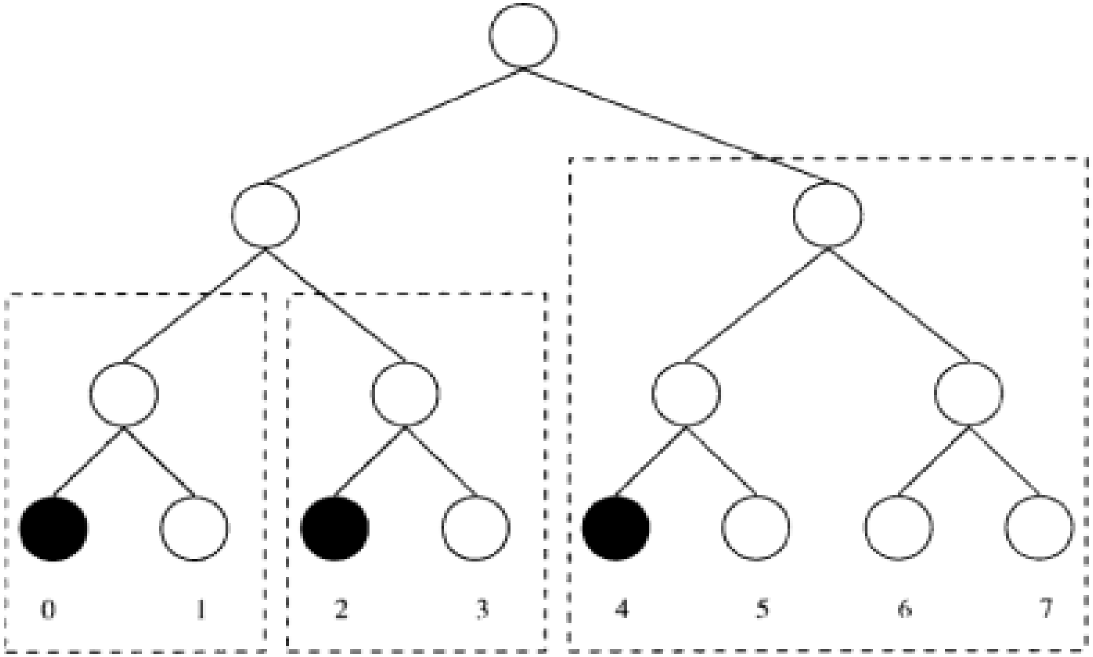
Dynamic Grouping

- To avoid collisions caused by the conventional BTS technique, the tree is split into smaller sub-trees before starting the next cycle.



Dynamic Grouping

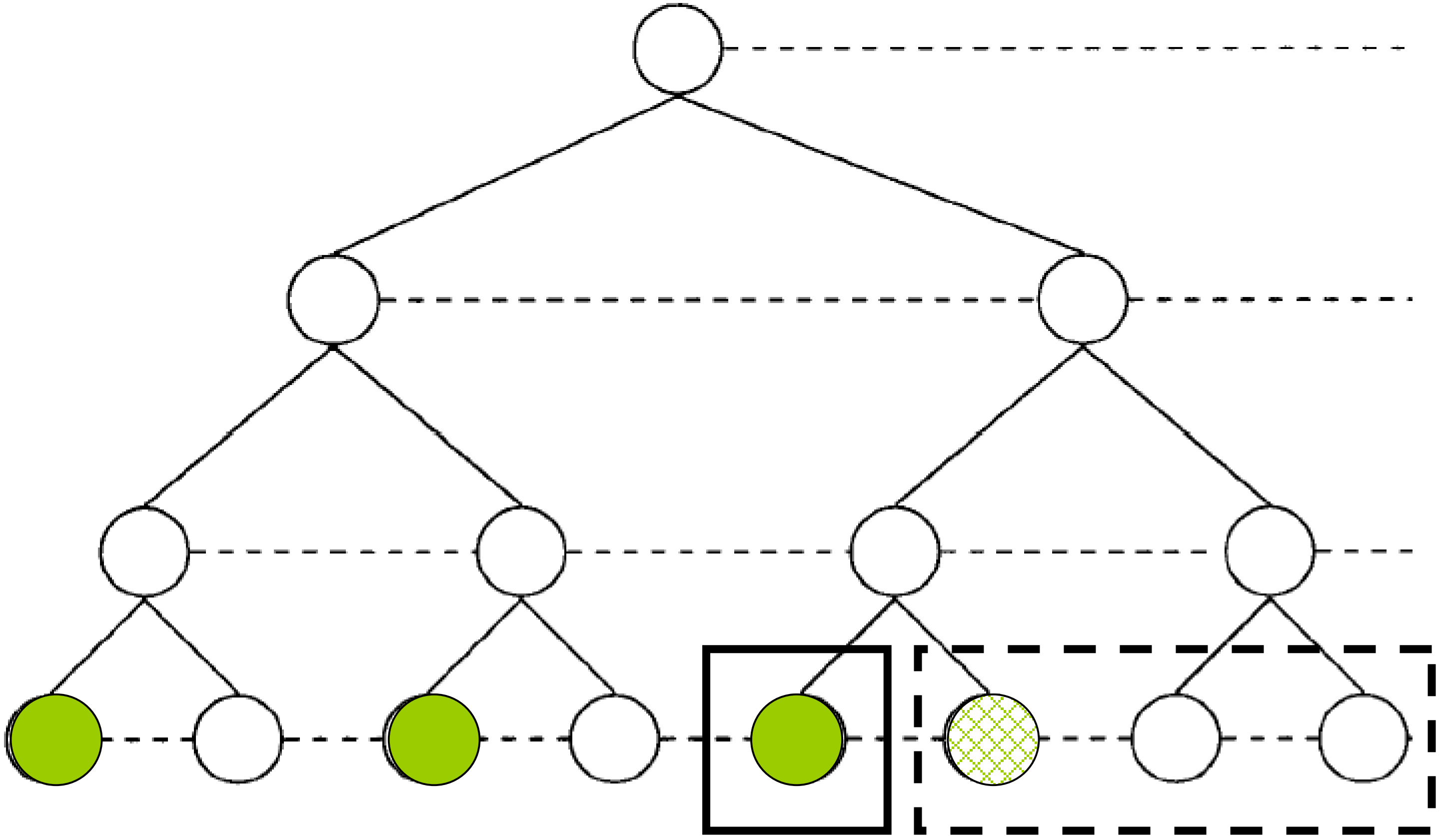
- To avoid collision and idle slots, the tree can be split into exactly m binary sub-trees with exactly one continuing station in each group.



Dynamic Collision Resolution

- This procedure considers new station joining a BTS-cycle.
- The representation of an interval that is stored in the stack is modified to a 3-tuple, denoted by (vid_l, vid_h, *mode*), here *mode* can be either TX or RS.
- Each station also maintains a status variable, *cmode*, with the value being TX, RS, or IDLE.
- (vid_l, vid_h, TX)->(vid_l, vid_l, TX),(vid_l, vid_h, RS)

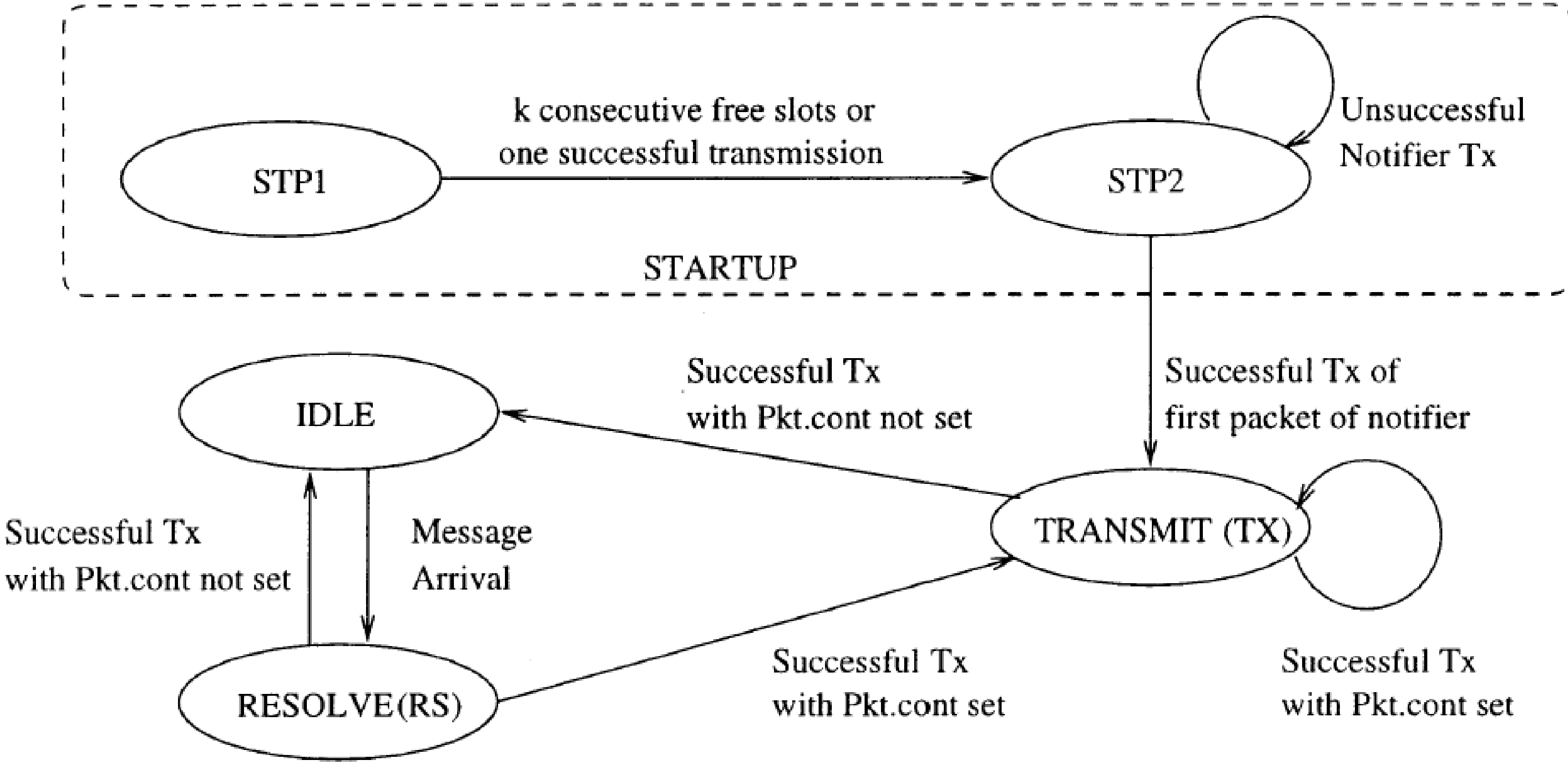
Dynamic Collision Resolution



Addition/Deletion of a Station

- Addition
  - Consistent stack [m, (vidl, vidh, mode), ncont]
  - K consecutive idle slots -> N-1
  - STP1 -> STP2 -> TX
- Deletion
  - *exit-notifier* packet with the *lrt* value
  - Stations with larger *lrt* decrement their *lrt* by 1.

Putting It All Together



Putting It All Together

Variable	Description	Range of values
pid	Physical identifier of the station	0 to $N-1$
vid	Virtual identifier of the station	0 to $N-1$
cmode	State of the station at a given time	$STP1$, $STP2$, $IDLE$, RS, TX
lrt	Denotes how recently the station transmitted in the current cycle	0 to $N-1$
ncont	Number of continuing stations	0 to $N-1$
m	Number of participating stations in the current BTS-cycle (*ncont* of previous BTS-cycle)	0 to $N-1$
vidl	Lowest identifier in the allowable interval	0 to $N-1$
vidh	Highest identifier in the allowable interval	0 to $N-1$
mode	Mode of the allowable interval	RS or TX
Pkt	A packet that is transmitted successfully	-
Pkt.cont	Indicates whether the source of the successful transmission is a continuing station	0 or 1
Pkt.lrt	Indicates the current *lrt* value of the source of transmitted packet Pkt	0 to $N-1$

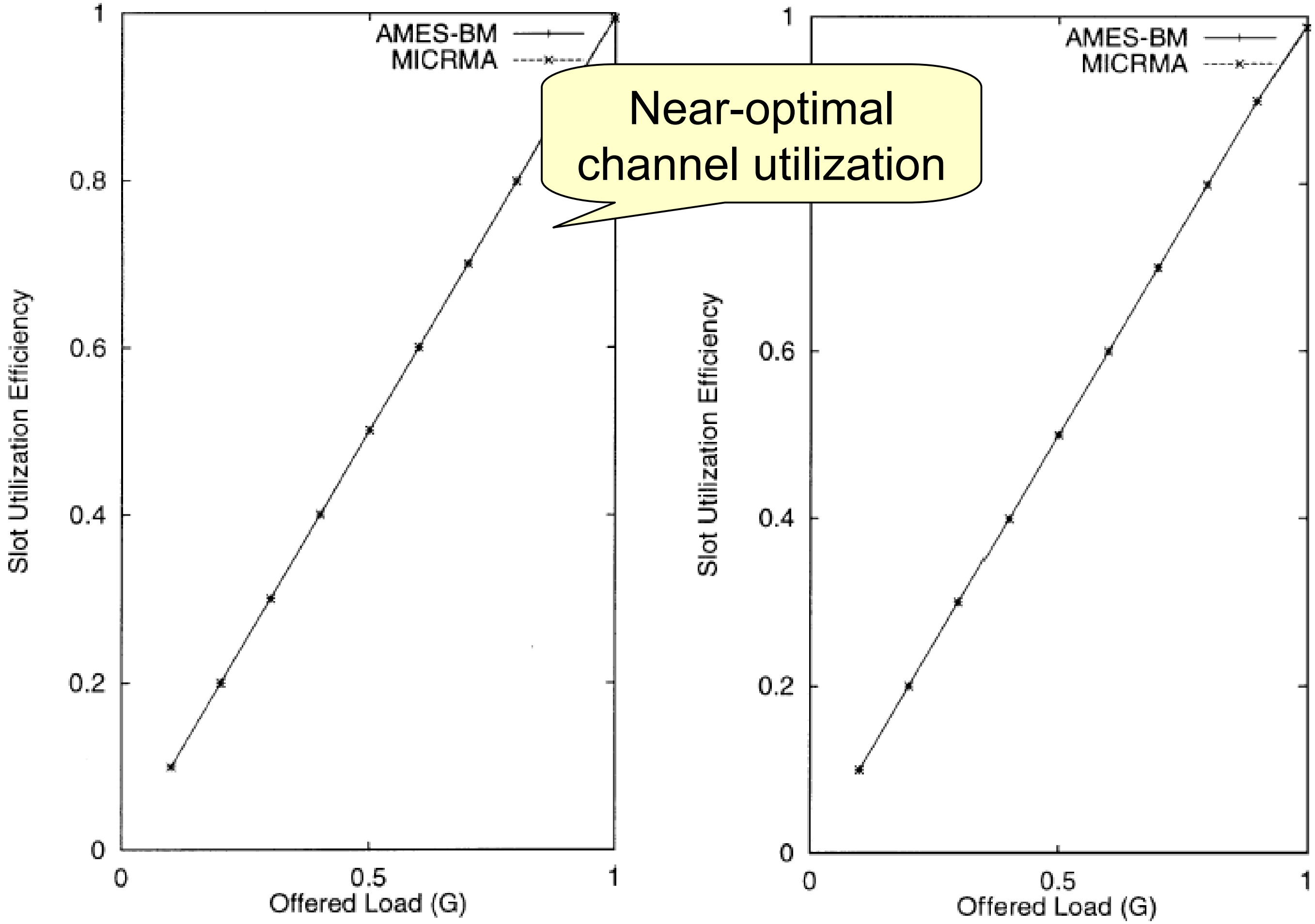
Simulation Settings

- AMES-BM v.s. MICRMA
- N=128, k=7
- Message: Poisson (λ/slot)
- Msg. Length: geometrical (mean *P*)
- Offered load $G=N\lambda P L_p$
- Maximum queue size=32 (MICRMA)?
  - If 128 is used, only 8% decrease in avg. delay

Metrics

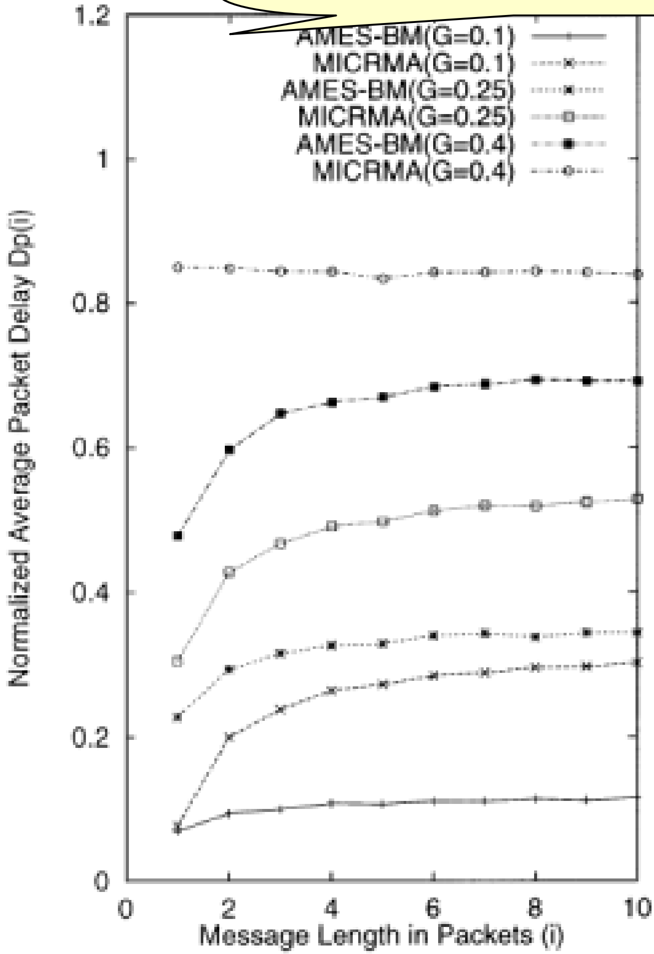
- Slot utilization efficiency
  - The ratio of the number of slots used for successful transmission to the total number of slots within a given time interval
- Average delay (fairness)
  - The ratio of the delay in transmitting a message to the number of packet in the message, normalized to the packet length

Simulation Results

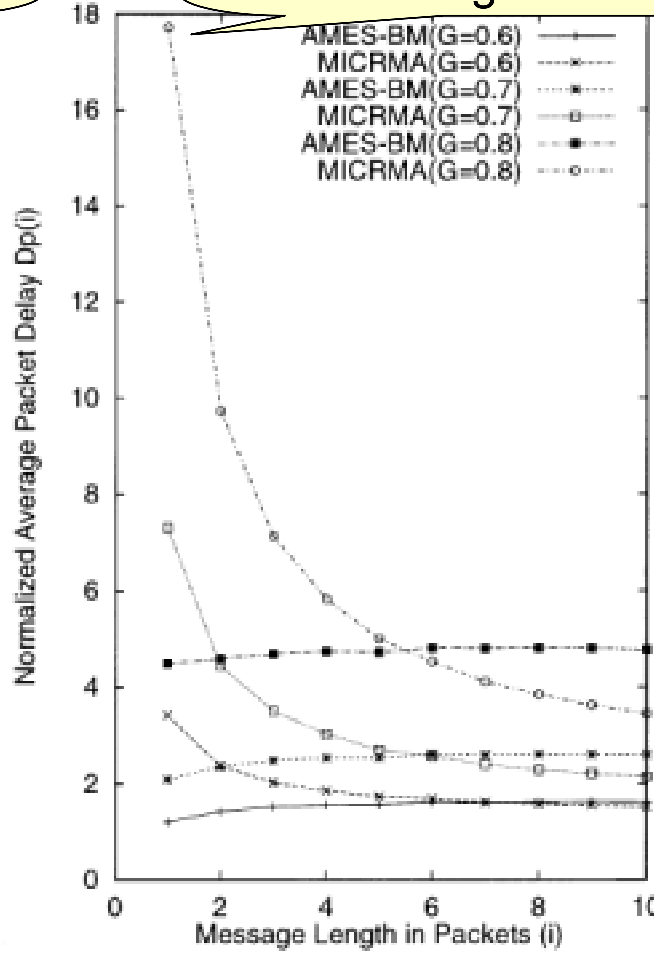


Simulation Results

The delay under lower load: AMES-BM<MICRMA



Unfairness: MICRMA favors long messages



Conclusions

- The performance of AMES-BM is similar to that of a collision-based protocol at low loads and to that of a a collision-free protocol at high loads.
- As the protocol is deterministic with respect to all the stations that are present in the system, the protocol is more attractive for real-time applications.
- The protocol can be extended to asynchronous channels, and also the collision resolution strategy can be apply in networks with a centralized channel arbitrator.