PERFORMANCE MEASUREMENT FOR LOCAL AREA NETWORKS

Alfred C. Weaver, Ph.D.
Catherine F. Summers, M.S.

Computer Science Report No. RM-85-05
November 8, 1985

PERFORMANCE MEASUREMENT FOR LOCAL AREA NETWORKS

Alfred C. Weaver
Catherine F. Summers

Department of Computer Science
University of Virginia

October 7, 1985

(slide title)

Section 1

INTRODUCTION

LOCAL AREA NETWORKS

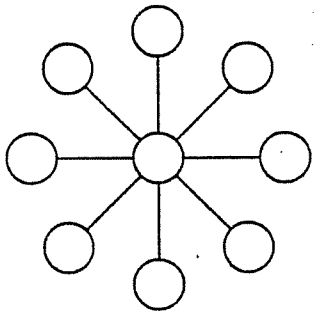
Three predominant topologies

- Star
- Ring
- Bus

Three predominant protocols

- Fully scheduled
- Contention
- Token passing

STAR



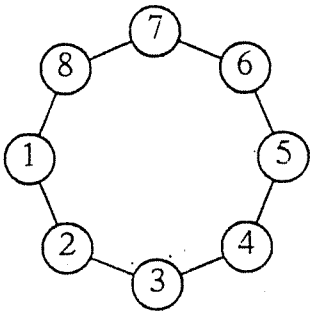
Advantages

- Simple access protocol
- Known transmission distance
- Two hop routing

Disadvantages

- Single point of failure
- All wiring to the "center"
- Sharing typically accomplished via time-slice

RING



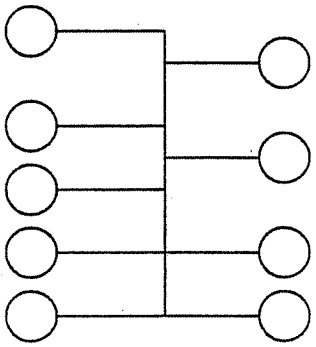
Advantages

- Simplicity
- All wiring is point-to-point; permits mixed media
- Message regenerated at each node

Disadvantages

- Failure of one link may break the ring
- Must be wired to a physical neighbor
- More complex management since message goes completely around the ring

BUS



Advantages

- Simplicity of installation
- Potential for maximum channel utilization
- Stations sense common network state
- Twisted-pair and coax media well understood

Disadvantages

- Needs complex access control
- Limitations on bus length

PROTOCOLS

Fully scheduled

- Time division multiple access
- Polling

Contention

- Radio channels (Aloha)
- CSMA/CD (Ethernet)

Token passing

- Token passing bus
- Token passing ring

CONTENTION

"Schedules" on a packet-by-packet basis

Each node arbitrates for access independently

No guarantee of fairness

No concept of priority

Restricted to baseband implementation

Performance limited by bus length

Message delay near zero at low offered load

Message delay is exponential with offered load

Message delay highly variable

Only 20-40% of bus capacity effectively utilized

TOKEN PASSING

Token confers the right to transmit

Token holder uses bus for a (bounded) period of time, rather than for a single packet

Fairness can be assured or ignored

Priorities easily accommodated

Performance affected by bus length

Message delay is exponential with offered load

Message delay is non-zero even at low offered loads

Message delay increases less rapidly than CSMA/CD

Perhaps 60-80% of bus capacity utilized

LAN SUITABILITY

Star

- Centralized control
- Deterministic traffic patterns
- Circuit-oriented communication
- Continuous (or at least frequent) transmission
- Popular with military

Ring

- Need to know your physical neighbor
- Wiring distance should be short
- Popular with vendors of engineering workstations

LAN SUITABILITY

Contention Bus

- Low average offered loads
- Traffic pattern unpredictable and bursty
- No need for fairness or priority
- Intended for office automation

Token Bus

- Handles deterministic and non-deterministic loads
- Can accommodate real-time demands
- Implements priority
- Intended for general purpose applications

IEEE COMMITTEE 802

Charged to develop a local area network standard

Committee membership predominately industrial

Recognized that application areas had different needs

- office automation
- factory automation
- real-time control systems

Impossible to agree on a single standard

- application areas are truly diverse
- mixed corporate membership inhibited agreement

802 SUBCOMMITTEES

802.1 -- Relationship to ISO OSI model

802.2 -- Logical Link Control of Data Link Layer

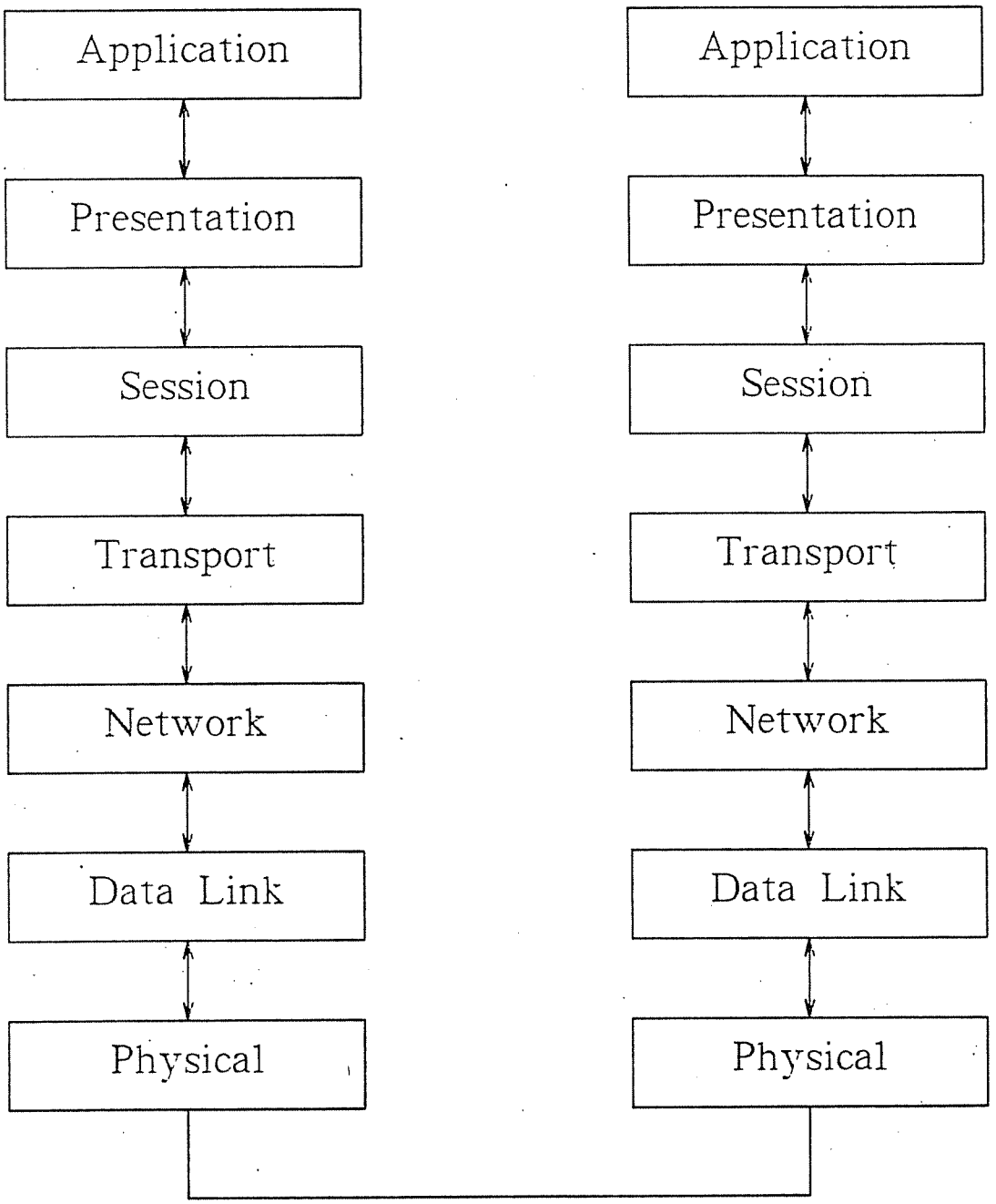
802.3 -- Contention Bus

802.4 -- Token Bus

802.5 -- Token Ring

802.6 -- Metropolitan Networks

ISO OSI MODEL



LAYERS 1, 2, and 3

Layer 3 -- Network

- Performs routing in an arbitrary network
- Usually null in a broadcast protocol

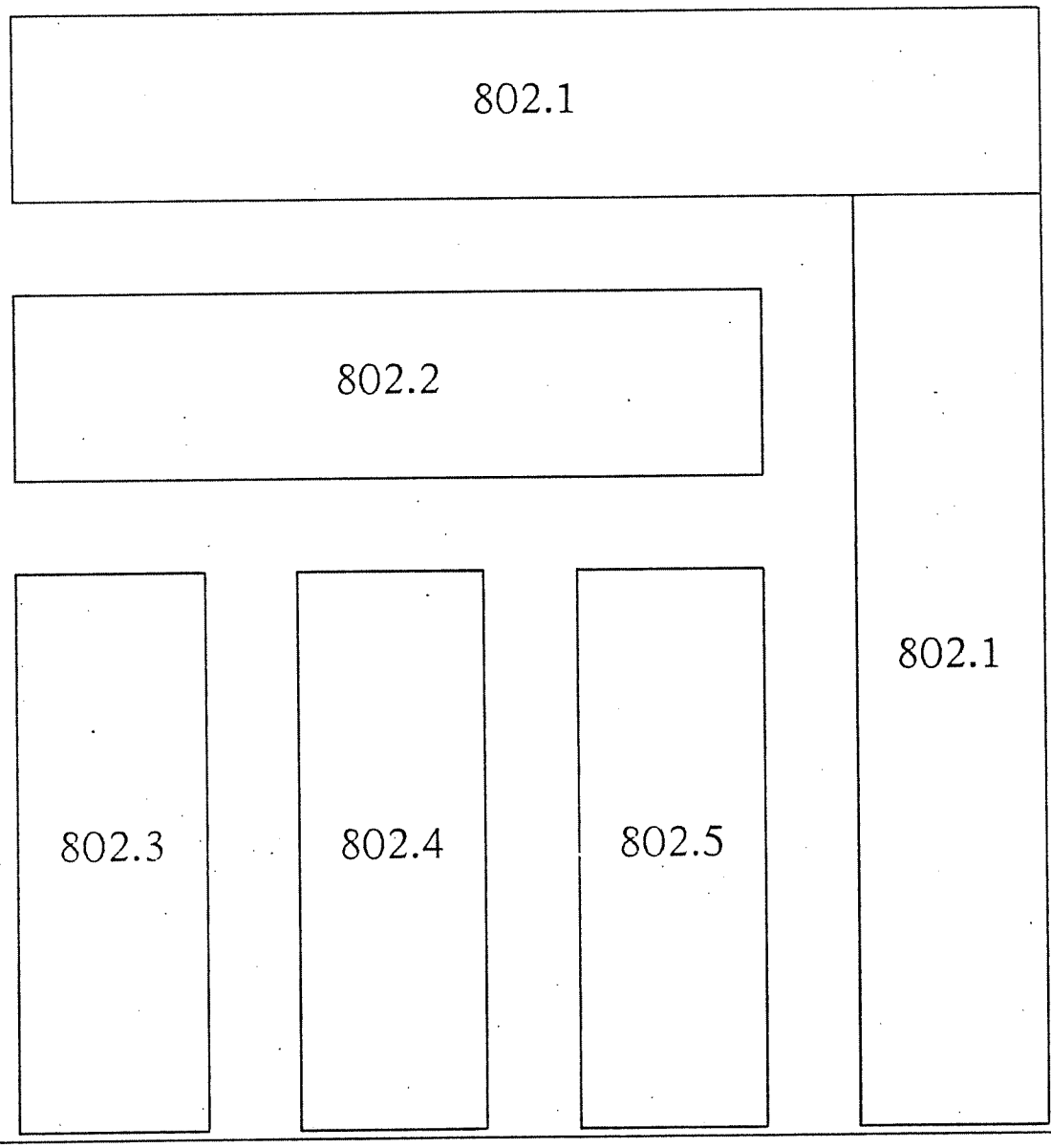
Layer 2 -- Data Link

- Logical Link Control -- framing
- Medium Access Control -- controls transmission

Layer 1 -- Physical

- Wiring considerations
- Modulation and propagation

802.x RELATIONSHIPS



Section 2

PERFORMANCE ANALYSIS

PERFORMANCE ANALYSIS

Techniques to estimate and evaluate performance

- Analytic models
  - Simple arithmetic models typically yield bounds
  - Simple queueing theory models typically yield averages
  - Complex queueing theory models can yield both
  - Underlying assumptions may be unrealistic
- Simulation
  - More effort than analytic modeling
  - Yields detailed performance results
  - Can be tailored to yield specific analyses
  - Difficult to verify correctness

PERFORMANCE ANALYSIS

- Network Measurement
  - — Requires that an implementation exist
  - — Most reliable technique
  - — Some details are hidden and cannot be measured
  - — Results come very late in the design cycle

In UVA Computer Networks Laboratory we use all three

PERFORMANCE ANALYSIS TERMS

Local Area Network -- A communications mechanism used to interconnect geographically adjacent computers, peripherals, and control devices.

Performance Analysis -- Determination of the operating characteristic of a LAN through analytical modeling, simulation, or measurement.

Performance Indices --

- *LAN Capacity* -- the maximum rate at which the physical medium carries information
- *Throughput* -- the fraction of LAN capacity actually used to transmit data
- *Utilization* -- the fraction of LAN capacity utilized by the protocol
- *Buffers* -- the queue of messages at each station

QUEUEING THEORY

Developed in response to the need to model telephone systems

Example -- A telephone switch has a single queue for incoming calls. Two processors handle all calls. As soon as a processor is free it serves the call at the head of the queue.

- What is the average time which a call spends in the system (queue delay plus service delay)?
- What is the average number of calls in the system?
- What is the average queue size?
- What is the probabilty that there are exactly k calls in the system?

CLASSES OF QUEUEING SYSTEMS

Queueing systems classes are "x/y/n"

- "x" is the type of arrival process
  - M -- Markov
  - D -- Deterministic
  - G -- General
- "y" is the type of service process
  - M, D, or G
- "n" is the number of servers
  - n = 1, 2, 3, 4, ...

QUEUEING THEORY CLASSES

Complexity varies with the class

For M/M/1 -- everything is known

For G/G/n -- virtually nothing is known

For M/D/n or M/G/n -- some things are known

We will assume the M/M/1 class for purposes of illustration

M/M/1 ASSUMPTIONS

Arrival process

- arrivals are a Poisson process
- arrival rate has an exponentially distributed probability density function
- mean arrival rate is λ (things)/(unit time)
- infinite number of potential customers to assure independence and no blocking

Service process

- exponentially distributed probability density function
- mean service rate of μ (units of time)/(thing)

System reaches a state of equilibrium

VALIDITY OF ASSUMPTIONS

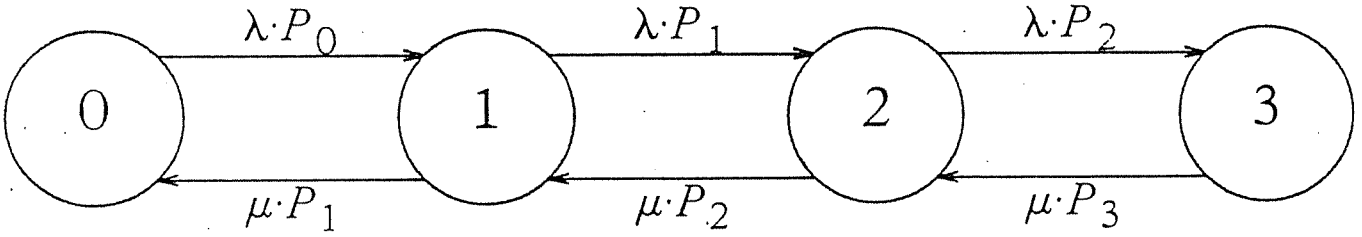
Assuming that arrivals are Poisson has been shown to be a reasonable assumption for communications systems

Exponentially distributed arrival rate is generally realistic

Exponentially distributed service rate is harder to justify, but can fit some situations

Care is required when interpreting results !

STATE SPACE



State k represents having k customers in the system (queue plus server)

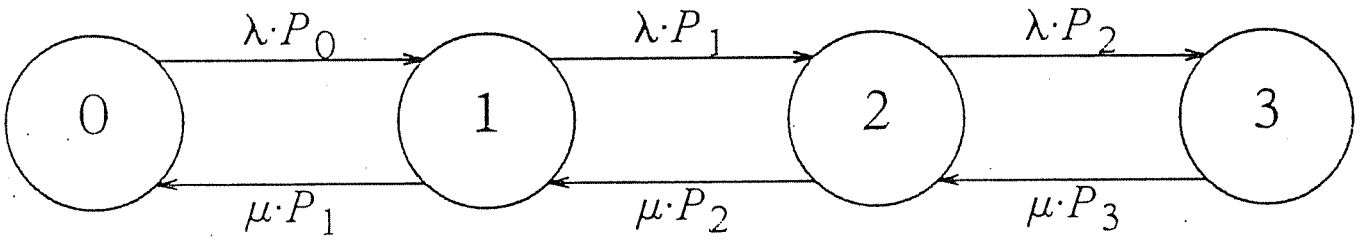
P_k is the probability of the system being in state k

System state increases at rate $\lambda \cdot P_k$

System state decreases at rate $\mu \cdot P_{k+1}$

For equilibrium, rates into and out of state k must be equal

LOCAL BALANCE EQUATIONS



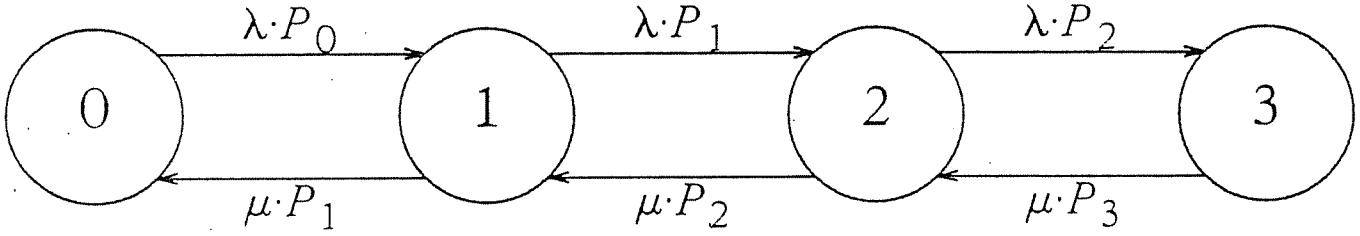
$$\lambda \cdot P_0 = \mu \cdot P_1$$

$$\lambda \cdot P_1 = \mu \cdot P_2$$

$$\lambda \cdot P_2 = \mu \cdot P_3$$

$$\lambda \cdot P_k = \mu \cdot P_{k+1}$$

LOCAL BALANCE EQUATIONS



$$P_1 = \frac{\lambda}{\mu} \cdot P_0$$

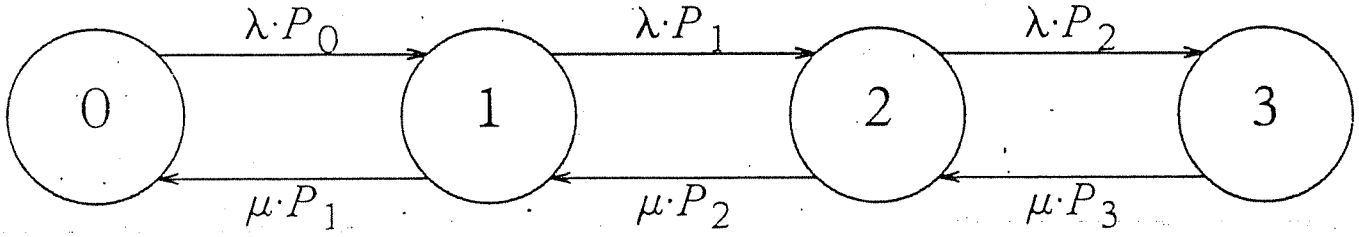
$$P_2 = \frac{\lambda}{\mu} \cdot \frac{\lambda}{\mu} \cdot P_0$$

$$P_3 = \frac{\lambda}{\mu} \cdot \frac{\lambda}{\mu} \cdot \frac{\lambda}{\mu} \cdot P_0$$

$$\rho = \frac{\lambda}{\mu}$$

$$P_k = \rho^k \cdot P_0, k \geqslant 1$$

LOCAL BALANCE EQUATIONS



$$P_k = \rho^k P_0$$

$$\sum_{k=0}^{\infty} P_k = 1$$

$$\sum_{k=0}^{\infty} \rho^k \cdot P_0 = 1$$

$$\sum_{k=0}^{\infty} \rho^k = \frac{1}{1-\rho}$$

$$P_k = (1-\rho)\cdot\rho^k$$

Probability that system is idle is $P_0 = (1-\rho)$

NUMBER OF CUSTOMERS IN SYSTEM

$$N = \sum_{k=0}^{\infty} k \cdot P_k$$

$$N = (1 - \rho) \sum_{k=0}^{\infty} k \cdot \rho^k$$

$$N = \frac{\rho}{1 - \rho}$$

EFFECT OF LOAD FACTOR

ρ	N
0.0	0.00
0.1	0.11
0.2	0.25
0.3	0.43
0.4	0.66
0.5	1.00
0.6	1.50
0.7	2.33
0.8	4.00
0.9	9.00
0.95	19.00
0.99	99.00

SERVICE TIME

Assume a customer consumes T seconds of (wait time + service time)

While he is being processed, $N = \lambda \cdot T$ customers arrive

The average waiting time is:

$$N = \lambda \cdot T$$

$$T = \frac{N}{\lambda} = \frac{1}{\mu - \lambda}$$

TRANSMISSION LINE DELAYS

Translating to communication terms:

- channel i has capacity C_i bits/second
- arrival rate is λ packets/second
- service rate is $\frac{1}{\mu}$ bits/packet
- so total delay, queueing plus transmission, is:

$$T_i = \frac{1}{\mu \cdot C_i - \lambda_i}$$

APPLICABILITY OF QUEUEING THEORY

Provides good results *when reality mirrors the assumptions*

Useful for a "quick and dirty" estimate of performance

Much more elaborate queueing theory models give better results for certain special cases

Simple M/M/1 model depends upon exponentially distributed service times

Slightly more complex M/D/1 model allows deterministic service times

M/M/1 and M/D/1 both require *continuous* server

Not realistic for cars at a traffic light

Unfortunately, not realistic for messages in a queue awaiting the arrival of a token

OTHER MODELS

More advanced queueing theory using concept of server "vacations"

"Vacation" models are too complex to study here

Simpler arithmetic models to predict bounds

Will use one in section 4 to predict token cycle times

Section 3

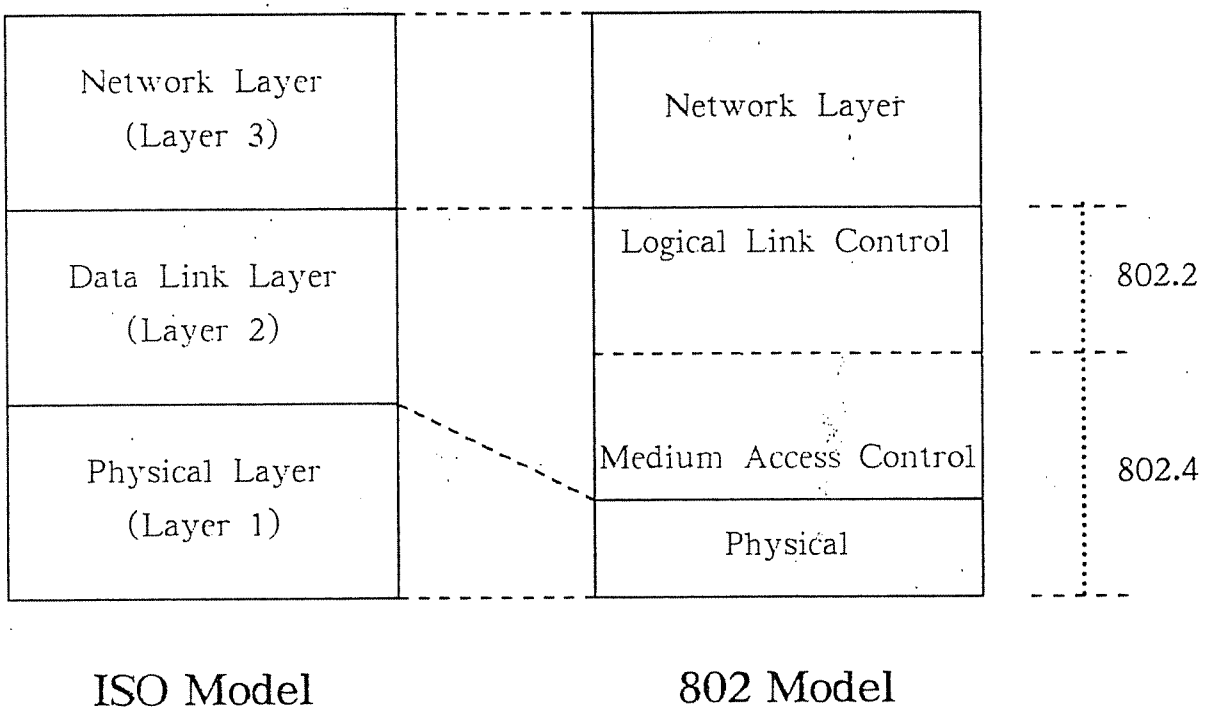
IEEE 802.4 TOKEN BUS

THE IEEE 802.4 TOKEN BUS

The 802.4 standard defines

- electrical and physical characteristics of the transmission medium
- electrical signaling method used
- frame formats transmitted
- actions of a station upon receipt of a data frame
- services provided by the Medium Access Control sublayer of the Data Link Layer

RELATIONSHIP OF 802.4 TO ISO OSI



802.4 SUMMARY

A *token* controls access to the physical medium

Token holder is momentarily the network master

Implements four priorities, or *access_classes*

Station must pass the token to a known successor within a bounded time

Orderly progression of the token from station to station forms a logical ring on a physical bus

Station's interface is a Medium Access Controller (MAC)

MAC implements the protocol, including

- token recognition, passing, and regeneration after loss
- message encapsulation and framing
- service of the four priorities
- error control and recovery

THE TOKEN

Standard allows a network-wide choice of either 16 or 48 bit addresses

Token is an explicit message of at least 96 or 160 bits depending upon address size (token frame can carry data)

Token consists of:

- preamble (1 or more octets)
- start delimiter (1 octet)
- control information (1 octet)
- destination address (2 or 6 octets)
- source address (2 or 6 octets)
- optional data (0 or more octets)
- frame check sequence (4 octets)
- end delimiter (1 octet)

MESSAGE FRAME

Message frame contains

- preamble (1 or more octets)
- start delimiter (1 octet)
- control information (1 octet)
- destination address (2 or 6 octets)
- source address (2 or 6 octets)
- data (0 or more octets up to maximum frame size of 8191 octets)
- frame check sequence (4 octets)
- end delimiter (1 octet)

TOKEN PASSING

At startup, each station is assigned a unique logical address

During startup, stations add themselves to the logical ring one at a time, thereby learning their successor

A station may transmit only while it holds the token

When all data has been transmitted or certain timers expire, token must be passed to successor

A station will periodically query the network to determine whether additional stations wish to join the ring

Special cases to recover from loss of token, failure of successor, etc.

LOGICAL RING MEMBERSHIP

At startup, each station is assigned a unique logical address

The station's *inter_solicit_count* is initialized to zero

The *Max_Inter_Solicit_Count* is randomized in its low order two bits every 50 milliseconds or after each use

The *inter_solicit_count* is decremented once per token receipt, just prior to passing the token

When the *inter_solicit_count* reaches zero, and if the *ring_maintenance_timer* has not expired, a *response_window* is opened

A *solicit_successor* message is sent

The response, if any, is resolved

If no response is received, *inter_solicit_count* is reloaded with *Max_Inter_Solicit_Count*

RESOLVING SOLICIT SUCCESSOR

Only three possible responses: none, one, or many

- No response
  - — pass token
- One response
  - — responding station sends *Set_Successor* frame to token holder
  - — token holder changes his "next station" variable to be address of new station
  - — pass token

RESOLVING SOLICIT SUCCESSOR

- Many responses (collision)
  - open four demand windows and send *Solicit_Successor*
  - other stations respond in window 0,1,2,3 depending upon value of the one's complement of the first two bits of station address
  - contending stations who hear a valid frame drop out of contention
  - process continues, using successively lower ordered pairs of address bits, as long as there are multiple responses
  - if no resolution on lowest pair of address bits, there is a duplicate address error
  - try once more; competing stations generate random address
  - stations which lose report duplicate address to network manager

(slide many responses)

LEAVING THE LOGICAL RING

Station has two variables, *in_ring_desired* and *any_send_pending*

Variable *any_send_pending* is true whenever there are any messages in any *access_class*

When ready to leave, sets *in_ring_desired* to false

When both flags are false, waits for token, then sends this station's predecessor a *Set_Successor* message with this station's successor as its value

OTHER ERRORS

Standard specifies recovery actions for

- lost or multiple tokens
- token pass failure
- deaf stations
- duplicate address
- stuck transmitter

ACCESS CLASSES

Four priorities or *access_classes*

- Synchronous
- Urgent Asynchronous
- Normal Asynchronous
- Time Available

Only *Synchronous* is guaranteed a level of service

Other classes receive "best effort"

An 802.4 station must implement either the *Synchronous* class alone or else all four classes simultaneously

Note that priority applies to a message, not a station

NETWORK PARAMETERS

Let the *Synchronous*, *Urgent Asynchronous*, *Normal Asynchronous*, and *Time Available access_classes* be abbreviated by *S*, *UA*, *NA*, and *TA*, respectively

High Priority Token Hold Time (HPTHT) -- the maximum amount of time a station may serve its *Synchronous* class

Urgent Asynchronous Target Rotation Time (TRT_{UA}) -- goal token rotation time for class UA

Normal Asynchronous Target Rotation Time (TRT_{NA}) -- goal token rotation time for class NA

Time Available Target Rotation Time (TRT_{TA}) -- goal token rotation time for class TA

STATION TIMERS

token_hold_timer (tht) -- when it expires, station may complete sending the packet in progress, but must then sequence to the next lower priority *access_class* or, if serving class TA, must pass the token to the station's successor

$token\_rotation\_timer_{UA}\,(trt_{UA})$

$token\_rotation\_timer_{NA}\,(trt_{NA})$

$token\_rotation\_timer_{TA}\,(trt_{TA})$

- used to monitor the token cycle time at *access_classes* UA, NA, and TA

$token\_rotation\_timer_{RM}\,(trt_{RM})$

- ring maintenance timer

SERVICE DISCIPLINE

At station startup, *trt*'s are initialized to zero

Token arrives at a station; its *tht* is set to the *HPTHT* and begins timing

Station serves its *Synchronous* class until its queue is empty or the *tht* expires

If station is not implementing priority, it passes the token

Station copies the residue of its trt_{ua} into the *tht*

Station reloads the trt_{ua} with TRT_{UA}

Station serves its *Urgent Asynchronous* class until its queue is empty or the *tht* expires

SERVICE DISCIPLINE

Station copies the residue of its trt_{na} into the *tht*

Station reloads the trt_{na} with TRT_{NA}

Station serves its *Normal Asynchronous* class until its queue is empty or the *tht* expires

Station copies the residue of its trt_{ta} into the *tht*

Station reloads the trt_{ta} with TRT_{TA}

Station serves its *Time Available* class until its queue is empty or the *tht* expires

Station passes the token to its successor

PERFORMANCE ANALYSIS

802.4 TOKEN BUS

ESTABLISH BASE CONFIGURATION

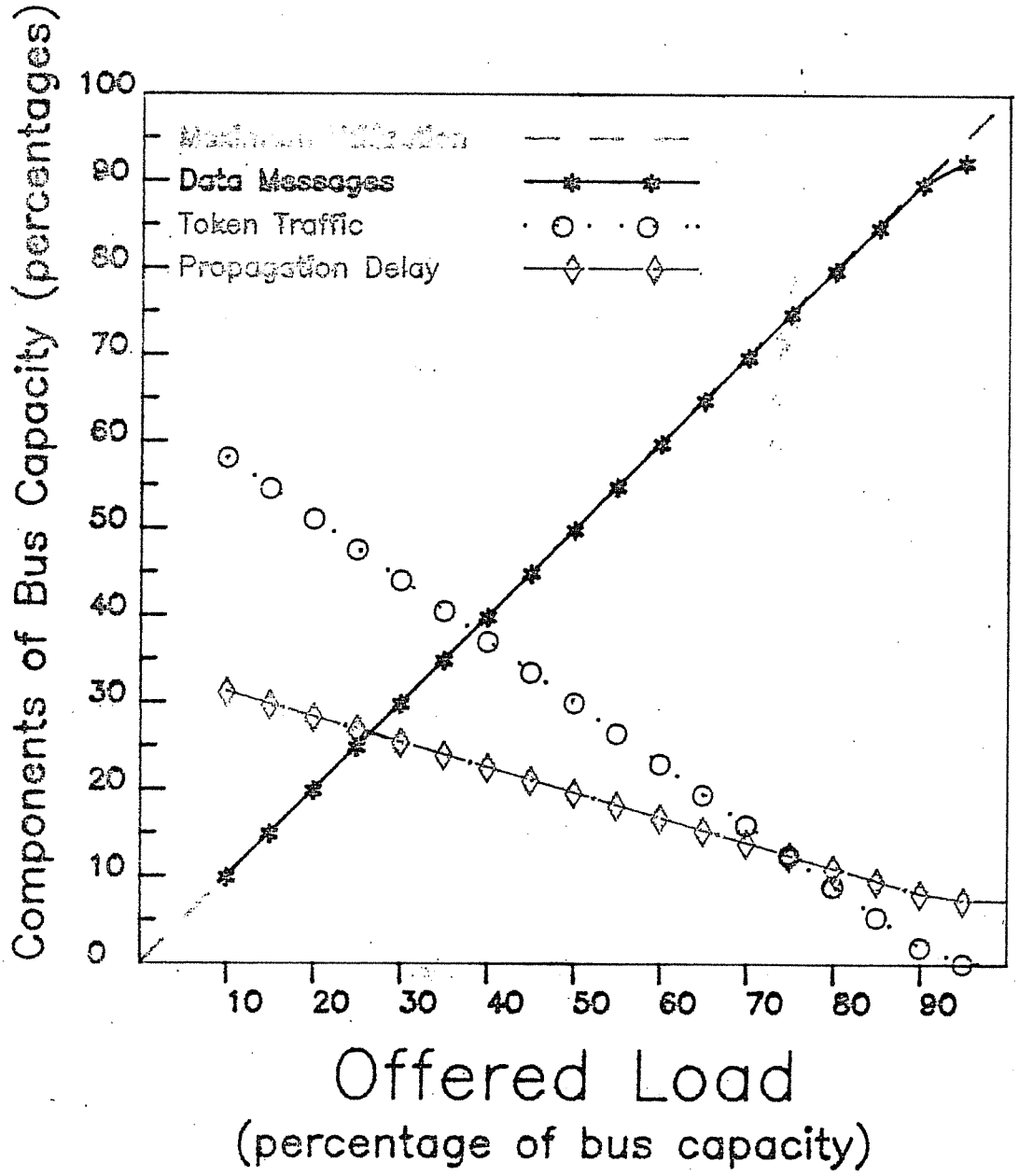
- 64 stations, always members
- Synchronous traffic only
- Infinite *High_Priority_Token_Hold_Time*
- Constant length 160 bit messages (256 bit frames)
- Max_Inter_Solicit_Count = 255
- Bus capacity = 10,000,000 bits per second
- Error free
- 16 bit addresses
- 96 bit tokens

COMPONENTS OF BUS CAPACITY

At any offered load, 100% of bus capacity is divided among

- data messages
- token traffic
- propagation delays

Graph shows distribution for base configuration


Components of Bus Capacity (percentages)
100
90
80
70
60
50
40
30
20
10
0
Data Messages
Token Traffic
Propagation Delay
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

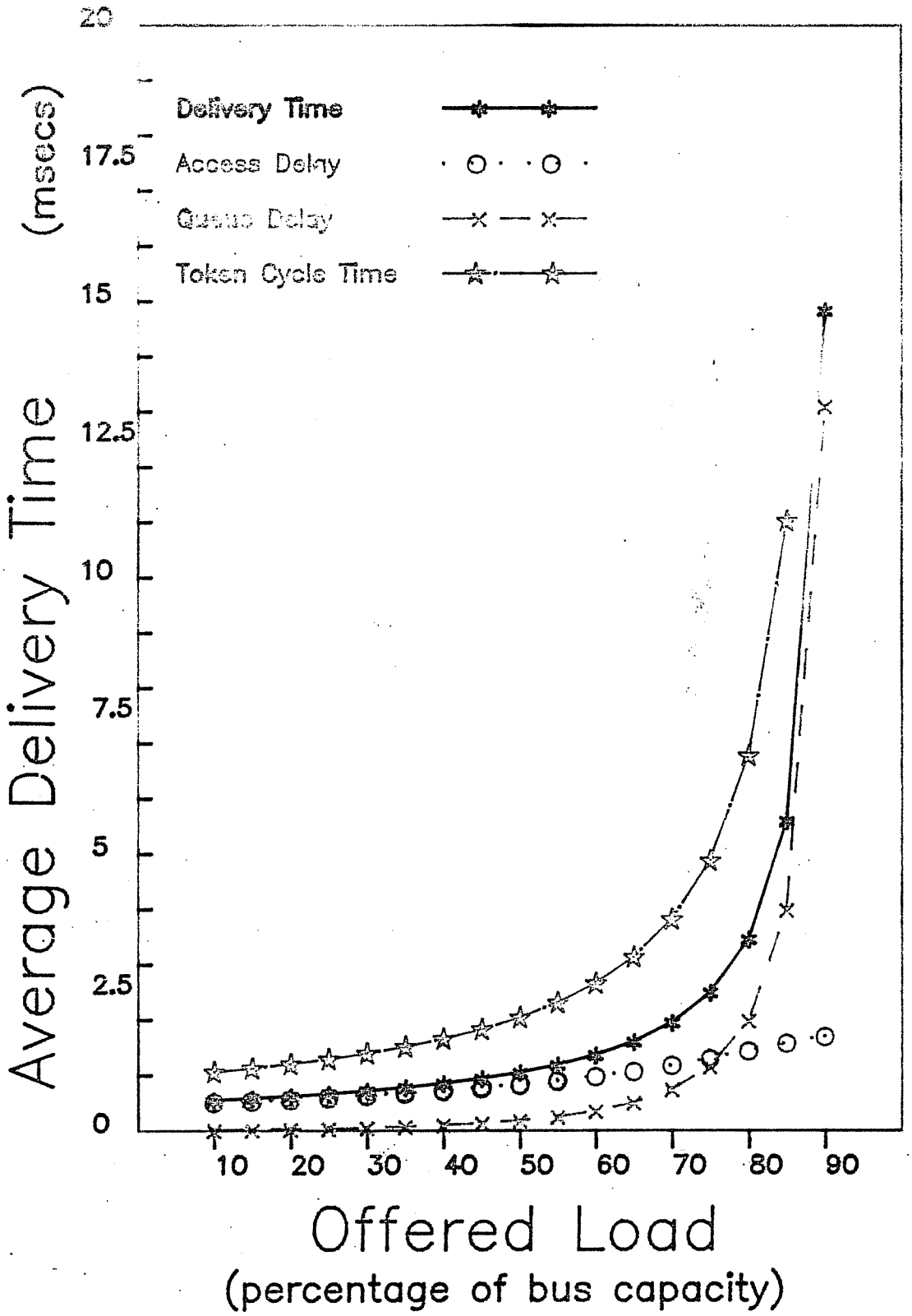
AVERAGE MESSAGE DELIVERY TIME

Average message delivery time consists of

- queueing delay
- network access delay
- propagation delay (constant here)

Graph shows components of total observed delay

Note that average message delay is about one-half the token cycle time


Delivery Time
Access Delay
Queue Delay
Token Cycle Time
Average Delivery Time (msecs)
20
17.5
15
12.5
10
7.5
5
2.5
0
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

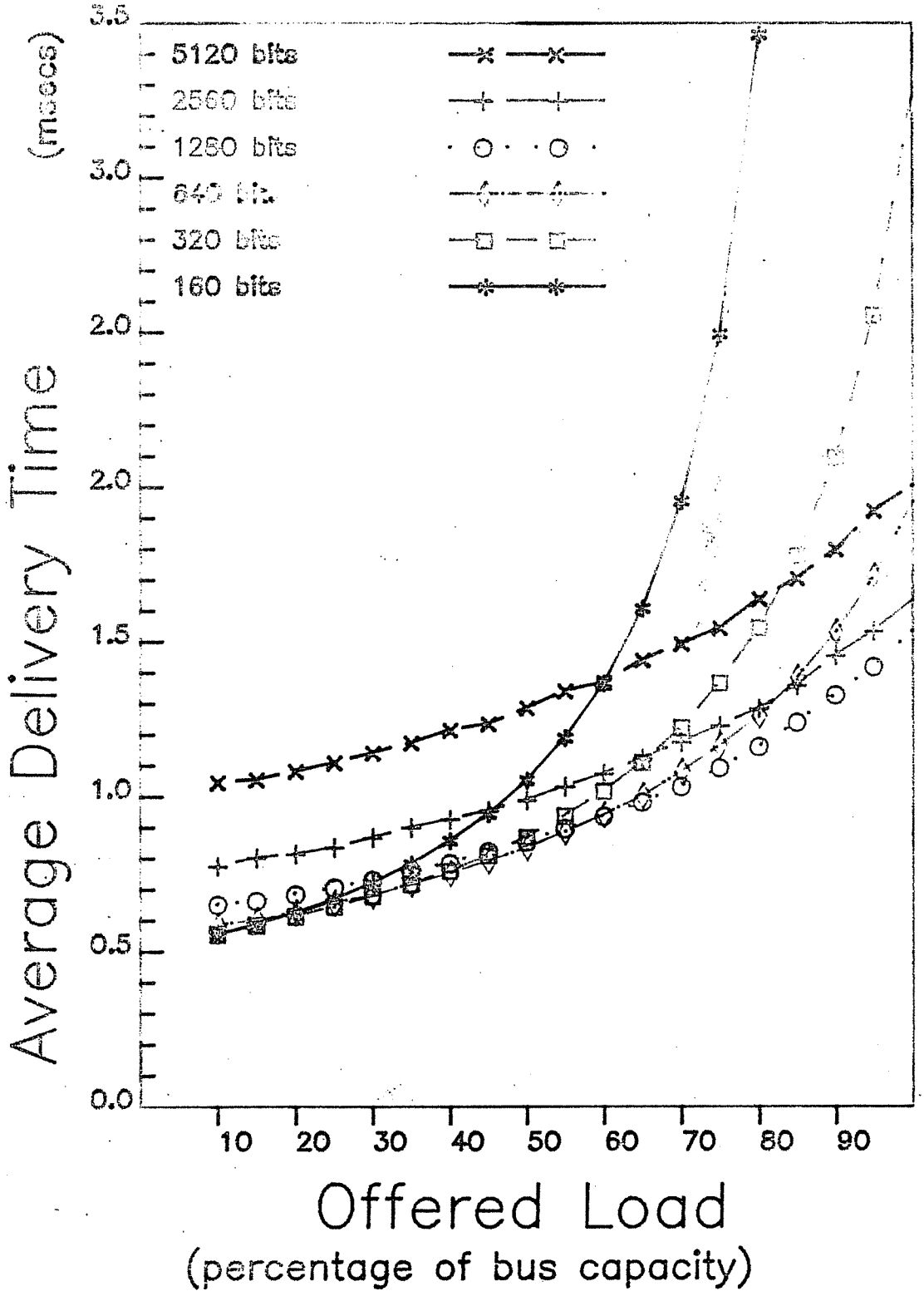
IMPACT OF PACKET SIZE

When a station sends multiple packets, they are separated by one inter-frame gap (2 microseconds)

When a station with small packets generates as much offered load as a station with large packets, then at large offered loads the station with small packets suffers longer total delays from its higher frame overhead and delays

Average delivery time is plotted against offered load for data lengths of 160..5120 bits

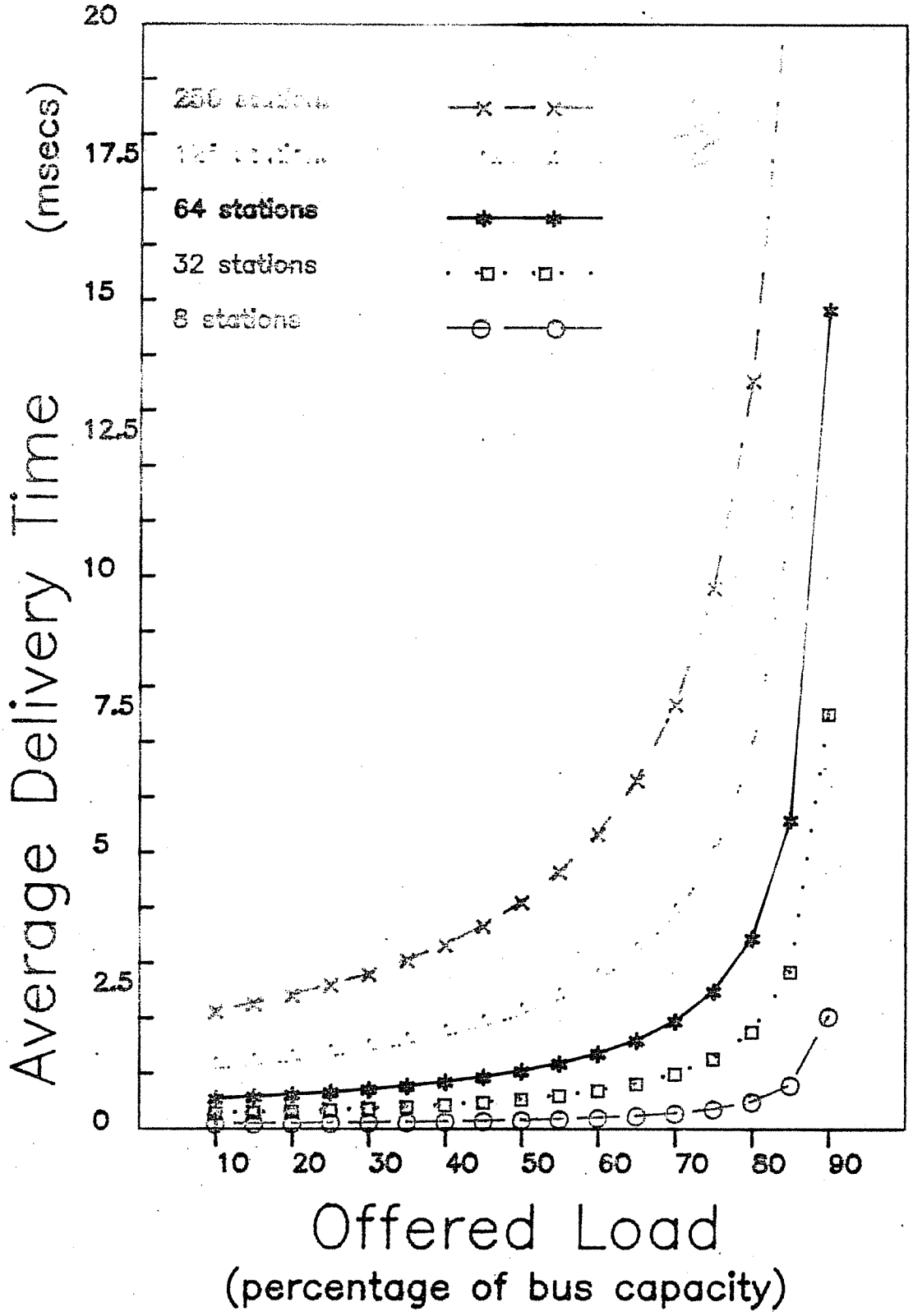
Frame size is 96 bits longer than data length


Average Delivery Time
(msecs)
3.5
3.0
2.0
2.0
1.5
1.0
0.5
0.0
5120 bits
2560 bits
1280 bits
640 bits
320 bits
160 bits
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

NUMBER OF ACTIVE STATIONS

Average message delivery time increases linearly with the number of active stations

Graphs shows 8..256 stations


Average Delivery Time (msecs)
20
17.5
15
12.5
10
7.5
5
2.5
0
64 stations
32 stations
8 stations
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

OBSERVATIONS

For *Synchronous* traffic with *HPTHT* set to infinity:

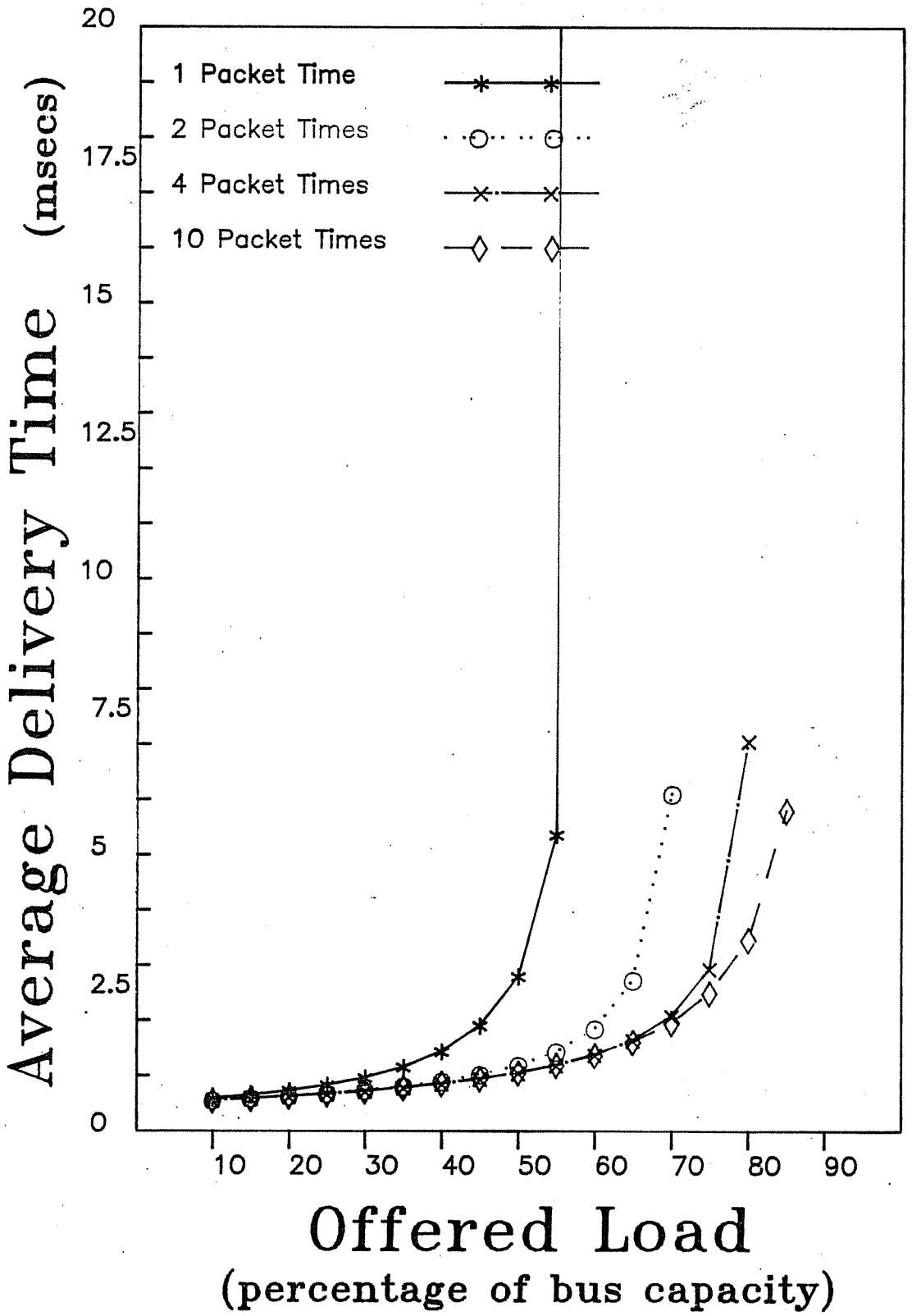
- Fraction of bus capacity used for data traffic (throughput) increases with offered load because protocol traffic decreases
- Average message delivery time is approximately one-half the token cycle time
- At high offered loads, long messages experience shorter average delivery times than short messages
- Network performance improves as the number of active stations decreases

HIGH PRIORITY TOKEN HOLD TIME

Setting the *HPTHT* has two side-effects:

- As desired, it guarantees a minimum frequency of service to the *Synchronous* class
- The penalty is that it also bounds the maximum data utilization of the bus

As *HPTHT* increases (thus draining the *Synchronous* queues more often and minimizing token traffic), bus utilization for data throughput increases


1 Packet Time
2 Packet Times
4 Packet Times
10 Packet Times
Average Delivery Time (msecs)
20
17.5
15
12.5
10
7.5
5
2.5
0
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

HIGH PRIORITY TOKEN HOLD TIME

Setting the *HPTHT* has two side-effects:

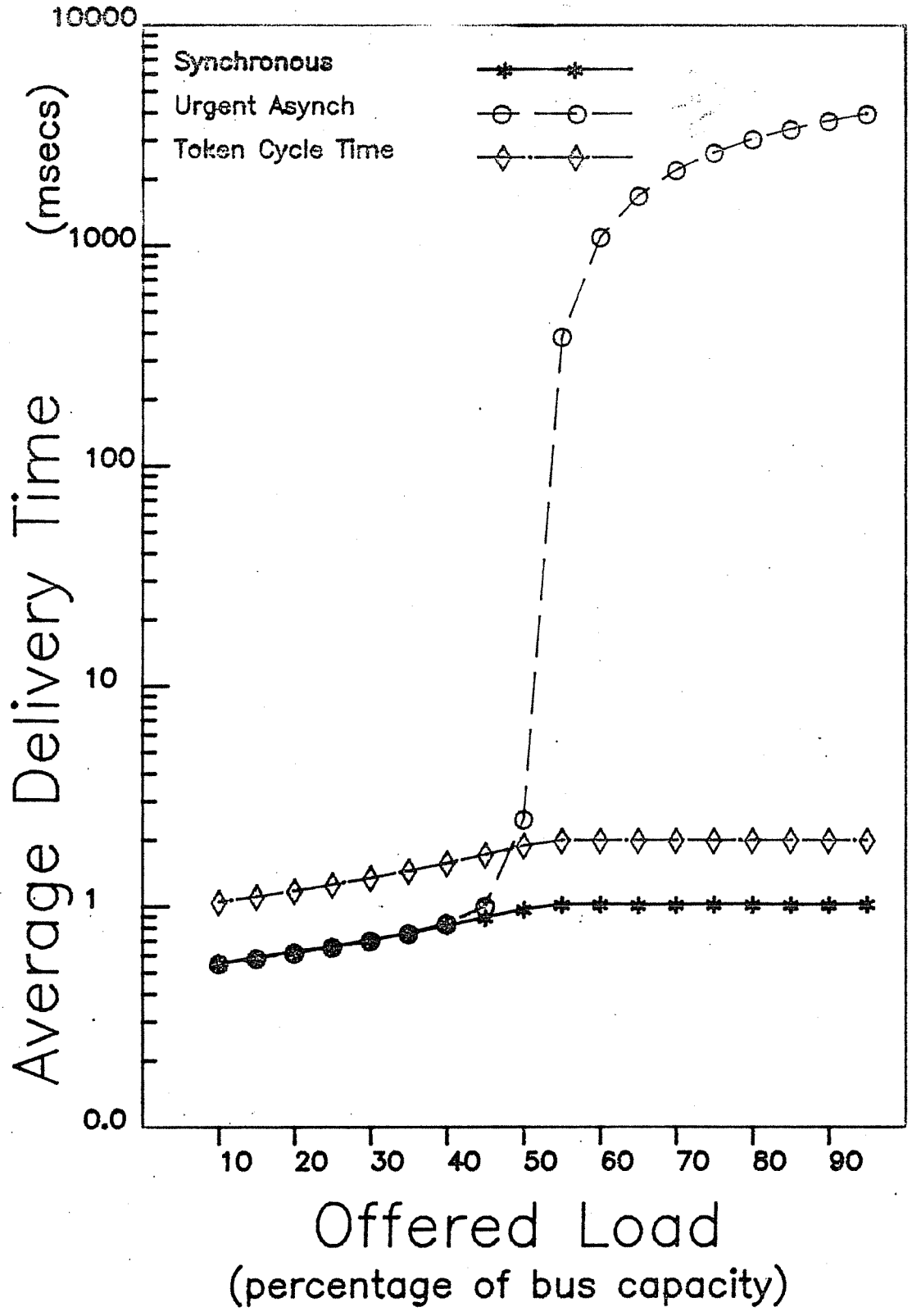
- As desired, it guarantees a minimum frequency of service to the *Synchronous* class
- The penalty is that it also bounds the maximum data utilization of the bus

As *HPTHT* increases (thus draining the *Synchronous* queues more often and minimizing token traffic), bus utilization for data throughput increases

NON-SYNCHRONOUS CLASSES

When token cycle time increases to equal the *Target_Rotation_Time* at an access_class, service to that access_class ceases

Graph plots delay of *Synchronous* and *Urgent_Asynchronous* classes


10000
1000
100
10
1
0.0
(msecs)
Average Delivery Time
Synchronous
Urgent Asynch
Token Cycle Time
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

OBSERVATIONS

When token cycle time does not approach $N \cdot HPTHT$ or any of the *TRT*'s, the priority system does not function; all messages are transmitted

Setting the *HPTHT* implies an upper bound on the fraction of bus capacity used to carry data

Setting the *TRT*'s effectively implements the priority operation

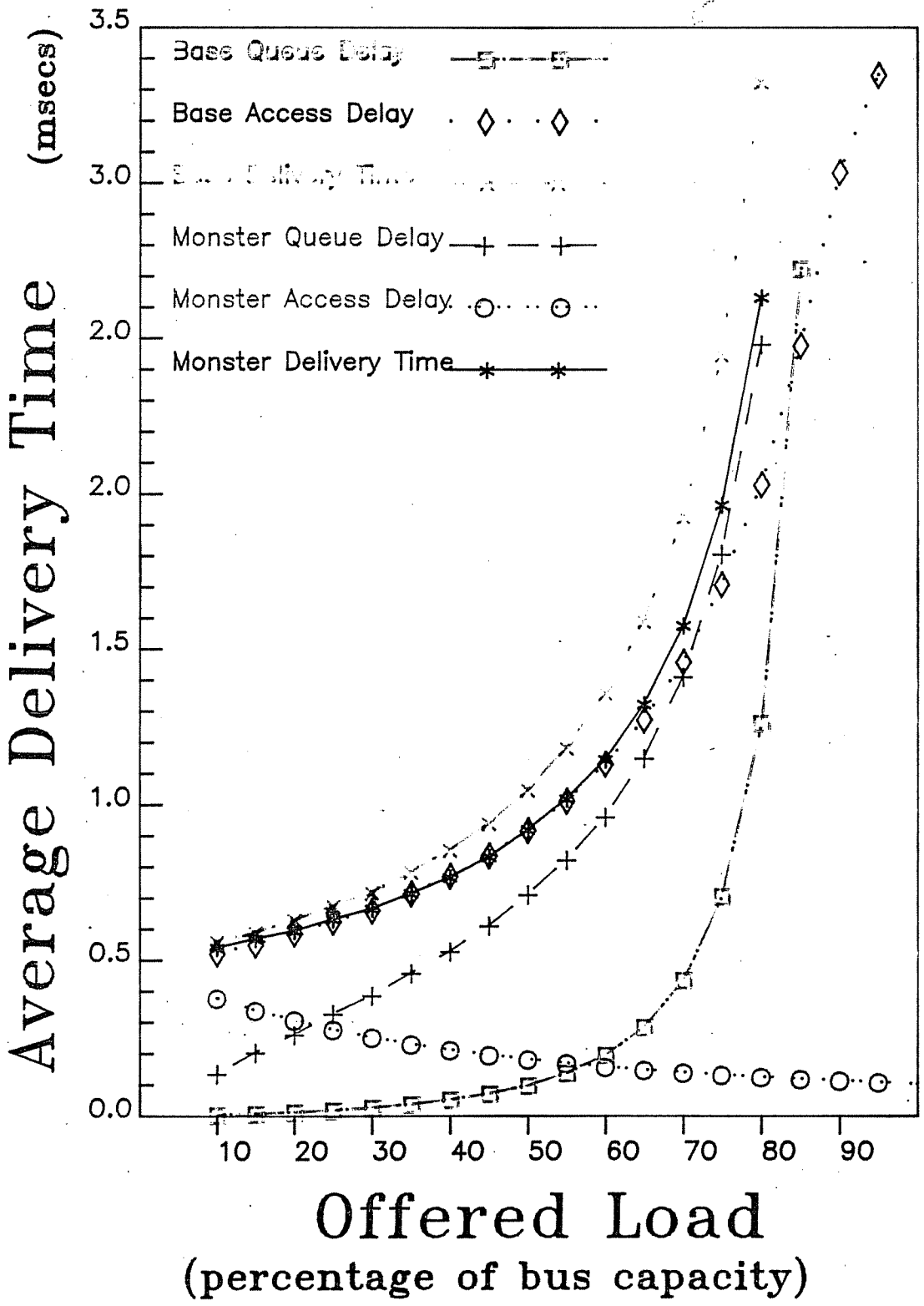
Setting the *TRT*'s intelligently is fairly difficult and requires both knowledge and insight

NON-HOMOGENEOUS LOADS

The offered load is divided:

- one station generates one-half the total offered load
- 63 stations collectively generate the other half

Graph shows the contributions of queueing delay and network access delay to total message delivery time for each class of station


Average Delivery Time
(msecs)
3.5
3.0
2.0
2.0
1.5
1.0
0.5
0.0
Base Queue Delay
Base Access Delay
Monster Queue Delay
Monster Access Delay
Monster Delivery Time
10
20
30
40
50
60
70
80
90
Offered Load
(percentage of bus capacity)

OBSERVATIONS

Highly loaded stations receive better service than other stations

The quality of service a station receives depends upon the load generated by the remainder of the network

Section 4

MATHEMATICAL MODEL OF TOKEN CYCLE TIME

TOKEN CYCLE TIME

$$TC_{r,i+1} = N \cdot X_T + V$$

$$+ \sum_{r \in S} \min(eff\,(HPTHT), f\,(Q_{r_i} + A_{r,i}))$$

$$+ \sum_{r \in UA} \min(eff\,(TRT_{UA} - TC_{r,i}), f\,(Q_{r_i} + A_{r,i}))$$

$$+ \sum_{r \in NA} \min(eff\,(TRT_{NA} - TC_{r,i}), f\,(Q_{r_i} + A_{r,i}))$$

$$+ \sum_{r \in TA} \min(eff\,(TRT_{TA} - TC_{r,i}), f\,(Q_{r_i} + A_{r,i}))$$

TOKEN CYCLE TIME

$$\overline{TC} = N \cdot X_T + V$$

$$+ N \cdot \min(\mathit{eff}\ (HPTHT), \lambda_S \cdot X_M \cdot \overline{TC})$$

$$+ \min(\mathit{eff}\ (TRT_{UA} - \overline{TC}), N \cdot \lambda_{UA} \cdot X_M \cdot \overline{TC})$$

$$+ \min(\mathit{eff}\ (TRT_{NA} - \overline{TC}), N \cdot \lambda_{NA} \cdot X_M \cdot \overline{TC})$$

$$+ \min(\mathit{eff}\ (TRT_{TA} - \overline{TC}), N \cdot \lambda_{TA} \cdot X_M \cdot \overline{TC})$$

Section 5

THE 802.4 SIMULATOR

The Simulator

- Tokbus is a 5000 line Pascal program that allows users to simulate user specifiable 802.4 networks.

- The simulator is a tool that allows
  - the network designer to test the performance of proposed configurations,
  - an advanced student of networks to study the relationships between configuration parameters and performance measures,
  - the network novice to understand the 802.4 protocol through observation.

- The simulator should
  - model the actions of 802.4 as closely as possible,
  - generate useful statistics about network performance
  - be easy to use.

Simulator Functions

- The two main functions of the simulator are to create network configurations and to simulate the actions of the network.

- To make the network configuration process easier for the user, tokbus allows the user to save, modify and reuse configuration information. There are two types of configuration information,

  — Global Information, network wide or simulator parameters, and

  — Station Classes Information

- Tokbus allows the user to control the progress of the simulation, to make limited modifications to the simulator state, and to generate statistical reports about the simulation.

Network Configuration

- To create a network configuration, the user can either enter the configuration information in response to a series of simulator prompts for specific configuration parameters, or the user can use previously entered configuration information.

- When the user is supplying information, tokbus either presents the user with a menu if the number of selections is a small finite set, or it will perform range checking to be sure that entered values meet the specifications of the 802.4 standard.

- After entering new information or after reading previously saved information the user is presented with a display of the values and a description of each parameter from which the user can select values to be modified.

- If the user enters new information or modifies saved information, tokbus allows the user the option of saving the new or modified information.

Global Information

- Global information consists of either network wide, protocol parameters or parameters specific to the simulator.

- Protocol Parameters
  - *High_Priority_Token_Hold_Time*
  - Address size (16 or 48 bits)
  - Frame size
  - Bus rate in bits per second
  - Token Pass Time
  - Bus Idle Time

- Simulator Parameters
  - Random Number Seed
  - Length of the Bus
  - Propagation Delay
  - Probability of a bit error
  - Number of Station Classes

Station Class Information

- Stations of the same class have identical loading in the *access_classes*, identical *TRT*s, and other common values.
- Class information can be saved in the Class Library after being created so that it can be used again in other simulations.
- Combining different classes of stations in a network configuration enables the user to test non-homogeneous load distributions.
- The use of saved station classes and the editing capabilities of tokbus aids parametric studies of protocol performance by reducing the number of steps required to create a series of simulations where only one parameter is varried.

Running the Simulator

- The simulator must provide the user with methods of controlling, observing, and generating reports about the simulation.

- To control the simulation the simulator provides commands that allow the user to

  — modify the state of the network or of a station

  — run the simulation for a period of time or a number of steps

- To observe the simulation the user can use a visual display on his terminal screen or traces of simulator actions that are written to files.

- Reports and Traces

  — The Bus Report provides the user with a summary of overall utilization of the communication medium.

  — The Class Report provides the user with a summary of utilization and delay statistics for each *access_class* in a station class.

Sample Bus Report

busrep.05.0

Elapsed Time - 16.00000000
Token Cycles - 15796.
Average Token Cycle Time - 0.00101286
Token Cycle Time Variance - 0.00000003
Minimum Token Cycle Time - 0.00045260
Maximum Token Cycle Time - 0.00135040
Total Traffic of 817809 messages for a total of 128475744 bits transmitted
Number of Ring Reconfigurations - 0

Type Of Message	Number Transmitted	Total Bits	%BandWidth
Data	312288.	79945728.	49.97
Overhead		29979648.	18.74
Actual Data		49966080.	31.23
Token	505502.	48528192.	30.33
Protocol	19.	1824.	0.00
Corrupted	0.	0.	0.00
Propagation Delay			19.70
Bus Idle			0.00

Sample Class Report

```
classrep.05.0
Class basecase
   Elapsed Simulation Time =  16.00000000
   Stations have joined the Token Passing ring     32 times.
Synchronous Service Access Class
   Total Number of Messages sent from these queues    =      312288.
   Total Number of Bits transmitted from these queues =    79945728.
   Average Queue Length  =                              0.61781464
   Queue Length Variance =                              0.23611971
   Maximum Queue Length  =                              1.00000000
   Average Queue Delay   =                              0.00000000
   Queue Delay Variance  =                              0.00000000
   Minimum Queue Delay   =                              0.00000000
   Maximum Queue Delay   =                              0.00000000
   Average Access Delay  =                              0.00066144
   Access Delay Variance =                              0.00000015
   Minimum Access Delay  =                              0.00000000
   Maximum Access Delay  =                              0.00142119
   Average Transmission Time  =                         0.00002760
   Transmission Time Variance =                         0.00000000
   Minimum Transmission Time  =                         0.00002760
   Maximum Transmission Time  =                         0.00002760
   Average Delivery Time  =                             0.00068904
   Delivery Time Variance =                             0.00000015
   Minimum Delivery Time  =                             0.00002760
   Maximum Delivery Time  =                             0.00144879
Urgent Asynchronous Service Access Class
Access Class Inactive
```

Simulator Traces

- The simulator provides the user with a selection of traces of of simulator actions or network transitions which can be recorded on a user defined output file.

- Traces consist of a description of the actions or station status, the station where the action occured, and the time at which the event occured.

- It is possible to trace

  — simulator execution,

  — the number of messages transmitted each time the station receives the token,

  — the number of messages enqueued at a station when the token arrives.

- Traces can be used for

  — validating the simulator,

  — observing the actions of the protocol.

Tokbus as a Model of 802.4

- To model the actions of 802.4
  - — minimize the number of simplifying assumptions e.g. inter station distance,
  - — implement all major actions and station states described in the 802.4 standard,
  - — translate the Ada pseudo-code that was part of the Finite State Machine description of the protocol into Pascal to insure compliance with the protocol
- Problem: Simulating concurrent actions by separate entities in a single processing environment.

Finite State Machine Representation

- The protocol specifies the set of the eleven possible states of the 802.4 *Medium_Access_Control* machine.
- From all station states there exist transitions based upon a certain input from state S to state S'.
- There is one and only one transition from state S for a given input.
- Example of the FSM

1. IDLE receive_token 5. USE_TOKEN

```
Rx_protocol_frame
AND Rx_frame.FC = token
AND Rx_frame.SA /= TS
AND Rx_frame.DA = TS
AND in_ring
     -- token receipt
     sole_active_station := false;
     PS := Rx_frame.SA;                    -- set predecessor
     -- set access_class to highest level
     access_class := max_access_class;
     token_hold_timer.start(hi_pri_token_hold_time);
     Rx_protocol_frame := false;
```

Finite State Machine Representation

- The eleven *Medium_Access_Control* machine states are

  OFFLINE
  IDLE
  DEMAND_IN
  DEMAND_DELAY
  CLAIM_TOKEN
  USE_TOKEN
  AWAIT_IFM_RESPONSE
  CHECK_ACCESS_CLASS
  PASS_TOKEN
  CHECK_TOKEN_PASS
  AWAIT_RESPONSE

- The simulator does not implement the *AWAIT_IFM_RESPONSE* state.

- The simulator implements simplified versions of the ring maintenance actions specified in the standard.

Discrete Event Simulation

- Events
  - For any action by an 802.4 station, there is an action by the simulator.
  - Each event is tagged with
    - an event type,
    - the station where the event occurs,
    - and the time of the event.
- Multiple events can occur at any given point in time.
- The events are maintained in a time ordered queue.
- Time
  - Time advances as events are dequeued.
  - Current time is the time of the most recently processed event

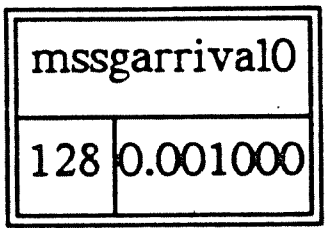
Event Queue

- The event queue is a future list not a true queue.
- List of events in increasing time order.
- Next event that will happen is at the front of the queue.
- A sample event

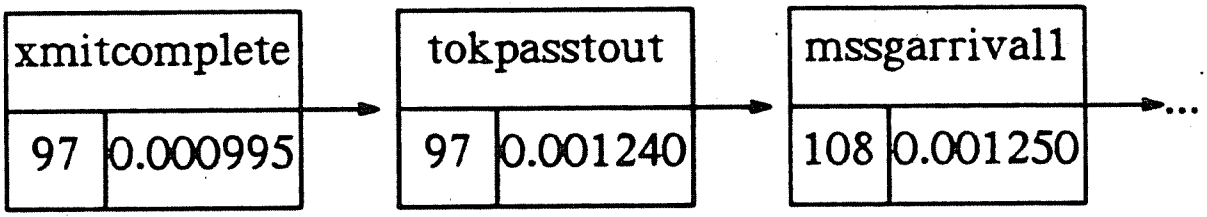
event type	
station	time

Event Queue Sample

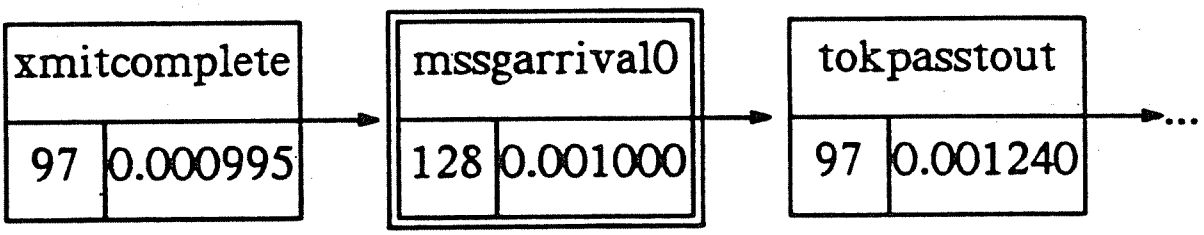
- Newly generated event



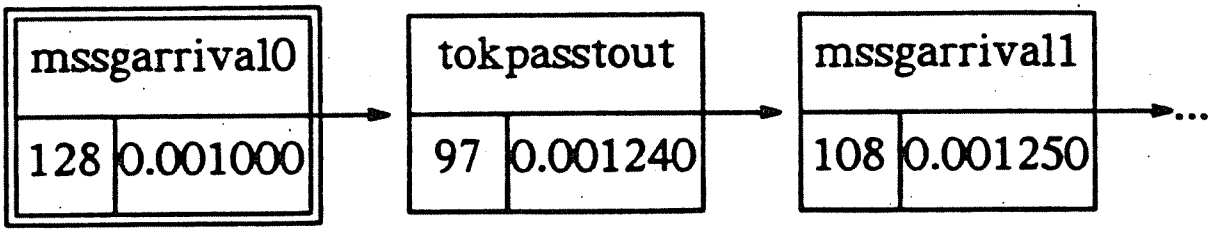
- Current state of the event queue



- Updated event queue



- Queue after next Simulator Step



Section 6

DESIGN EXAMPLES

All Synchronous Traffic

- Global Parameters

  Bus rate = 10 Mbps
  Bus length = 1000 meters
  Propagation Delay = 5 nanoseconds/meter
  HPTHT = 0.052 seconds
  token = 96 bits

- Station Class Parameters

  32 stations
  256 bit messages
  λ_S at 10% = 122 mssgs/sec

Two Active *access_classes*

- Global Parameters

  Bus rate = 10 Mbps
  Bus length = 1000 meters
  Propagation Delay = 5 nanoseconds/meter
  HPTHT = 0.052 seconds
  token = 96 bits

- Station Class Parameters

  32 stations
  256 bit messages
  λ_S at 10% = 81 mssgs/sec
  λ_{UA} at 10% = 41 mssgs/sec
  TRT_{UA} = 1.21 milliseconds

All *access_classes* active

- Global Parameters

  Bus rate = 10 Mbps
  Bus length = 1000 meters
  Propagation Delay = 5 nanoseconds/meter
  HPTHT = 0.052 seconds
  token = 96 bits

- Station Class Parameters

  32 stations
  256 bit messages
  λ_S at 10% = 81 mssgs/sec
  λ_{UA} at 10% = 11.4 mssgs/sec
  TRT_{UA} = 1.21 milliseconds
  λ_{NA} at 10% = 9.2 mssgs/sec
  TRT_{NA} = 0.85 milliseconds
  λ_{TA} at 10% = 20.4 mssgs/sec
  TRT_{TA} = 0.64 milliseconds