



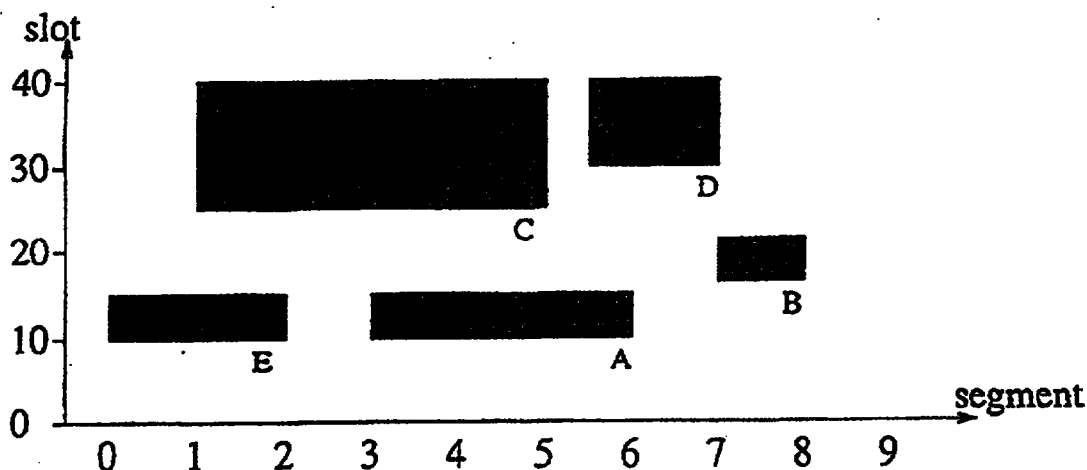
PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : H04L 12/64, 12/40	A1	(11) International Publication Number: WO 97/24844 (43) International Publication Date: 10 July 1997 (10.07.97)
(21) International Application Number: PCT/SE96/01748 (22) International Filing Date: 23 December 1996 (23.12.96) (30) Priority Data: 9504679-3 28 December 1995 (28.12.95) SE (71) Applicant: DYNARC AB [SE/SE]; Hagavägen 10, S-171 43 Solna (SE). (72) Inventor: RAMFELT, Lars; Svanckegränd 1, 2tr., S-164 40 Kista (SE). (74) Agents: L.A. GROTH & CO. KB et al.; P.O. Box 6107, S-102 32 Stockholm (SE).	(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG , SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). Published With international search report. With amended claims.	

(54) Title: METHOD AND ARRANGEMENT FOR SLOT REUSE IN A DYNAMIC SYNCHRONOUS TRANSFER MODE NETWORK ✓



(57) Abstract

The present invention relates to a slot reuse method and arrangement for increase of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network. The slot reuse according to the invention comprises extending the DTM block token format to include parameters describing the segment(s) between source and destination node, reserving block token capacity only on the segment(s) between source and destination node, and enabling simultaneous transmissions in the same slot over disjointed segments of the network. The slot reuse can also comprise merging two or more segment-consecutive tokens, representing the same slot range, into a single token when existing in the free pool of a node.

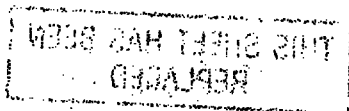
**METHOD AND ARRANGEMENT FOR SLOT REUSE IN A DYNAMIC SYNCHRONOUS TRANSFER
MODE NETWORK****TECHNICAL FIELD OF THE INVENTION**

The present invention relates to a slot reuse method and arrangement for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network.

5 DESCRIPTION OF RELATED ART

New high-capacity communication networks and protocols are constantly being developed by the communications industry and academia. This development change frequently and new results are important to application developers who integrate real-time
10 audio, video, and asynchronous communication services into applications. The applications can be found on a wide range of network access terminals. Terminals act as network hosts and may be almost any electronic device, including small pocket telephones or television sets, multi-media workstations, and
15 million-dollar supercomputers. Hosts differ by orders of magnitude in their needs for processing power, and in their communication services requirements. These disparate needs are currently reflected in a set of independent network classes. Each network class is optimized for its particular traffic and
20 applications: Cable Television networks use unidirectional broadcast networks where the capacity is divided into fixed-sized subchannels carrying video. A telephone network uses only 64 kbit/s duplex circuits with guaranteed throughput and tightly controlled delay variations. Computer networks, such as
25 the Internet, allow a large number of parallel network sessions by use of connectionless packet switching. They also use statistical multiplexing to efficiently utilize links. A backbone network for mobile systems needs extra control (or signaling) capacity to dynamically track all active terminals.





THIS SHEET HAS BEEN
REPLACED

WO 97/24844

2

PCT/SE96/01748

With this wide spectrum of applications existing today and to be augmented in the future, it is infeasible to continually invent new, sometimes global, networks and a new terminal interface for each new type of service. Instead, an integrated
5 services network supporting existing and new services needs to be deployed. The overall goals for such a network are scalability to global size and maximum sharing of expensive network components to minimize cost. Optical transmission
10 technology has been shown to provide the necessary link capacity at a low enough price to make integrated services networks a realistic solution.

A new integrated optical network with much higher capacity will, however, introduce new problems not seen in today's more specialized and lower performance networks. First, when network
15 capacity increases and the information flight latency remains limited by the speed of light, the increasing bandwidth delay product will put higher demand on mechanisms that isolate a user's traffic from third-party network traffic. A telephone session, for example, should not be affected by another user
20 opening a high-capacity video channel. Second, applications and protocols will need to operate reliably with an increasing amount of information in transit to benefit from the increased network capacity. This will lead to larger burst and transaction sizes in the network.

25 Current networks using stateless packet switching protocols such as the IP (Internet Protocol) have turned out to be extremely scalable. They have evolved from small networks connecting just a few DARPA (Defense Advanced Research Projects Agency) research computers in the mid seventies to the current
30 global and ubiquitous Internet. Shared medium LAN's (Local Area

This Sheet Has Been Replaced

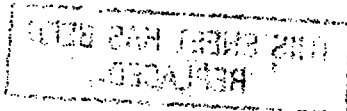


Networks) such as CSMA/CD (Carrier Sense Multiple Access/
Collision Detection, token ring and FDDI (Fiber Distributed
Data Interface) are used in the Internet as simple building
blocks connected by routers or bridges. The combination of easy
5 expansion, low incremental node cost and tolerance to faulty
nodes has resulted in simple, flexible, and robust networks.
Also, the shared medium allows efficient application of new
multicast protocols such as IP multicast.

A drawback of the shared medium is that it typically permits
10 only a single terminal to transmit at any time not utilizing
all network segments efficiently. A scheme allowing the
capacity of the medium to be reused may be designed, but this
is often at the cost of complexity in the high-speed access
control hardware. Access control mechanisms for a shared medium
15 also depend strongly on the size of the network and are usually
efficient only for local area environments.

The two main types of network are connection oriented circuit-
switched networks used for telephony, and connectionless
packet-switched networks exemplified by the Internet. When a
20 circuit-switched network is used for data communication,
circuits need to remain open between bursts of information,
wasting network capacity. This problem arises because circuit
management operations are slow compared to dynamic variations
in user demand. Another source of overhead in circuit-switched
25 networks is the limitation of requiring symmetrical duplex
channels, which introduce 100% overhead when information flow
is unidirectional. This constraint also makes multicast
circuits difficult to implement and inefficient. A
connectionless packet-switched network, on the other hand,
30 lacks resource reservation and must add header information to





each message before transmission. Furthermore, latency in a connectionless packet-switched network cannot be accurately predicted and packets may even be lost due to buffer overflow or corrupted headers. The latter two factors make real-time service difficult to support. Congestion avoidance mechanisms can isolate traffic streams of different users. These schemes are, however, limited to operating on a time scale comparable to the round-trip packet delay.

To address the problems mentioned above, the communications industry is focusing on the development of ATM (Asynchronous Transfer Mode). ATM has been proposed for LAN's and many future public networks. CCITT (International Telegraph and Telephone Consultative Committee) also adopted it to be used as the transfer standard in B-ISDN (Broadband- Integrated Services Digital Network). ATM networks are connection-oriented and establish a channel just as circuit-switched networks, but use small fixed-sized packets called cells for information transfer. The packet-switched nature of ATM means that the network needs to have new mechanisms such as buffer resource managers and link schedulers to establish real-time guarantees for a connection i.e. virtual channel/path.

Another solution for providing real-time guarantees centers on a circuit switched network and must therefore address circuit-switching concerns such as described above. A new shared-medium control protocol is also employed, and so have common shared-medium problems to be considered. This design, called DTM (Dynamic Synchronous Transfer Mode), (see e.g. Christer Bohm, Perl Lindgren, Lars Ramfelt, and Peter Sjödin, The DTM Gigabit Network, *Journal of High Speed Networks*, 3(2):109-126, 1994 and Lars Gauffin, Lars Håkansson, and Björn Pehrson, Multi-gigabit



networking based on DTM, *Computer Networks and ISDN Systems*,
24(2):119-139, April 1992) uses channels as the communication
abstraction. This channels differ from telephony circuits in
various ways. First, establishment delay is short so that
5 resources can be allocated/deallocated dynamically as fast as
user requirements change. Second, they are simplex and so
minimize overhead when the communication is unidirectional.
Third, they offer multiple bit-rates to support large
variations in user capacity requirements. Finally, they are
10 multicast, allowing any number of destinations.

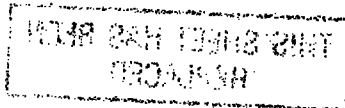
DTM channels share many beneficial properties with circuits.
There is no transfer of control information after channel
establishment, resulting in very high utilization of network
resources for large data transfers. Support of real-time
15 traffic is natural; there is no need for policing, congestion
control or flow-control within the network. The control
information is separated from data, which makes multicast less
complex. The transmission delay is negligible (i.e. less than
125 μ s) and there is no potential for data loss because of
20 buffer overflow as in ATM. Bit-error rates depend on the
underlying link technologies, and switches are simple and fast
due to strict reservation of resources at channel setup.

DTM can show good performance in areas where traditional
circuit-switched networks fall short: dynamic bandwidth
25 allocation, channel establishment delay, and as shared media
networks.

SUMMARY OF THE INVENTION

One of the problems using DTM is that a data slot never can be
used by two nodes simultaneously in the network.





WO 97/24844

6

PCT/SE96/01748

The object of the invention is to solve this problem. This is accomplished with a slot reuse method and arrangement for better utilization of shared links and consequently an increase of the throughput in the DTM network.

- 5 The slot reuse according to the invention comprises extending the DTM block token format to include parameters describing the segment(s) between source and destination node, reserving block token capacity only on the segment(s) between source and destination node, and enabling simultaneous transmissions in the
10 same slot over disjointed segments of the network.

The slot reuse can also comprise merging two or more segment-consecutive tokens, representing the same slot range, into a single token when existing in the free pool of a node.

- The slot reuse method and arrangement according to the invention
15 avoids the problem mentioned above. The method let channels only reserve capacity on the segments between source and destination node. A single slot may then be used multiple times simultaneously in the network.

- A very important advantage with the invention is that no
20 hardware changes to the original prototype implementation is needed.

Another advantage with the invention is that it can be realized in both a central and a distributed token management scheme and in a combination of these schemes.

- 25 Yet another advantage is that the performance gain is clear: on a dual-bus where source and destination node pairs are uniformly distributed it has been shown that throughput may be increased by a factor of two.

This Sheet Has Been Replaced



Yet another advantage is that the performance gain can be even higher in other network designs; e.g. in a double ring structure with source and destination nodes uniformly distributed throughput may be increased by a factor of four. Having
 5 nonuniformly distributed source and destination nodes the gain may be much higher.

Yet another advantage is an improved performance when combined with a defragmentation method returning tokens to their home nodes as a way to increase the probability that two consecutive
 10 tokens can be merged in the free pool of the nodes, which decreases fragmentation.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described in greater detail below with reference to enclosed drawing, in which

- 15 fig. 1 illustrates a dual-bus DTM network,
- fig. 2 illustrates a DTM 125 μ s cycle with a home node defined for each data slot according to the invention,
- fig. 3 illustrates a token map showing slot number and segment,
- fig. 4 illustrates a slot-segment map showing reuse of slots
 20 according to the invention,
- fig. 5 illustrates the performance (throughput and access delay, respectively, versus offered load) for small user requests (16 kbyte) and for different minimum acceptable capacities as simulated using a distributed token server according to
 25 the invention,
- fig. 6 illustrates the performance for small user requests (16 kbyte) and for different number of allowed retries before request blocking as simulated using a distributed token server according to the invention,



- fig. 7 illustrates access delay as a function of simulated time using A) the defragmentation scheme according to the invention and no fragmentation at simulation start, B) the defragmentation scheme according to the invention and maximum fragmentation at simulation start, and C) no defragmentation scheme and maximum fragmentation at simulation start,
- fig. 8 illustrates theoretical dual-bus and DTM throughput versus offered load for a dual-bus DTM network with and without slot reuse,
- fig. 9 illustrates the performance for different packet sizes as simulated using a central token manager in a 10km long dual-bus DTM network without slot reuse,
- fig. 10 illustrates the performance for different packet sizes as simulated using a central token manager in a 10km long dual-bus DTM network with slot reuse according to the invention,
- fig. 11 illustrates the performance for different packet sizes as simulated using a central token manager in a 1000km long dual-bus DTM network with slot reuse according to the invention,
- fig. 12 illustrates the performance for different packet sizes as simulated using a distributed token manager in a dual-bus DTM network with defragmentation and slot reuse according to the invention,
- fig. 13 illustrates the performance for different bus lengths as simulated using a distributed token manager with defragmentation and slot reuse according to the invention, and
- fig. 14 illustrates the performance for different traffic conditions as simulated using a distributed token manager



in a dual-bus DTM network with defragmentation and slot
 reuse according to the invention.

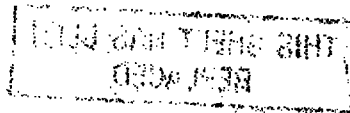
DETAILED DESCRIPTION OF EMBODIMENTS

First, the DTM MAC (Medium Access Control) protocol will be
 5 described. The basic topology of a DTM network is a bus with
 two unidirectional optical fibers connecting all nodes as is
 showed in fig. 1. Several busses with different speeds may be
 connected together to form an arbitrary multistage network. In
 the current prototype implementation, buses can be combined
 10 into a two-dimensional mesh. A node at the junction of two
 buses can synchronously switch data slots between the two
 busses. This allows for efficient switching with constant delay
 through the node. The primary communication abstraction in DTM
 is a multi-rate, multicast channel.

15 The DTM medium access protocol is a time-division multiplexing
 scheme. The bandwidth of the bus is divided into 125 μ s cycles,
 which in turn are divided into 64-bit time slots (or slots for
 short) as illustrated in fig. 2. The number of slots in a cycle
 thus depends on the network's bit-rate; for instance, on a 6.4
 20 Gbit/s network there are approximately 12500 slots per cycle.

The slots are divided into two groups, control slots and data
 slots. Control slots are used to carry messages for the
 network's internal operation, such as messages for channel
 establishment and bandwidth reallocation. The data slots are
 25 used to transfer user data and are not interpreted by
 intermediate network nodes. Intermediate nodes are nodes
 between the source and destination.





THIS SHEET HAS BEEN
REPLACED

WO 97/24844

10

PCT/SE96/01748

In each network node there is a node controller NC, which controls the access to data slots and performs network management operations, such as network start-up and error recovery. The main tasks of the node controller are to create
5 and terminate channels on demand from users, and to manage network resources in response to user requests and in the background.

Control slots are used exclusively for messages between node controllers. Each node controller has write permission to at
10 least one control slot in each cycle that it uses to broadcast control messages down-streams to other nodes. Since write access to control slots is exclusive, the node controller always has access to its control slots, regardless of other nodes and network load. The number of control slots a node use
15 may vary during network operation.

The network is not restricted to a dual-bus, but can be realized by other kind of structures, e.g. a ring structure with an arbitrary number of nodes. The transmission media can
20 besides to optical fibers be coaxial cables or any other high bandwidth transmission media. Hereafter the transmission media will be referred to as optical fibers. The bandwidth of the DTM dual bus in the preferred embodiment is divided into 125 μ s cycles, which in turn are divided into 64-bit time slots. The invention is not restricted to DTM networks with this values,
25 but can be used in networks with cycles and slots of arbitrary sizes.

Principles for resource management (called token management) will be described below. The majority of the slots in a cycle are data slots. Access to data slots changes over time,
30 according to traffic demands. Write access to slots is

This Sheet Has Been Replaced



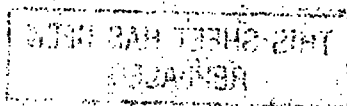
controlled by slot tokens (or tokens for short). A node controller may write data into a slot only if it owns the corresponding token. The token protocol guarantees the slot access to be conflict free, which means that several nodes do not write data into the same slot.

Control messages for channel establishment and bandwidth reallocation carry set of slots or tokens as parameters. However, a control message is 64 bits and can therefore have only a small number of parameters. This means that if a user requests a large bandwidth transfer, it may be necessary to send several control messages to create the channel. This introduces extra access delay and consumes signaling capacity.

We are considering several mechanisms to decrease the amount of information that needs to be sent during channel creation and token reallocation. The first optimization in token management is to introduce block tokens. A block token is transferred in a single control message and represents a group of tokens, but can only be used for particular combinations of tokens. For instance, in the simulator a block token is denoted by a slot number and an offset giving the number of contiguous slots in the group. The block token optimization assume that the token pool is not fragmented into small pieces. This will be referred to as fragmentation of the token pool.

The token protocol guarantees that a data slot can never be used by two nodes simultaneously on the bus. Sometimes this protocol is too conservative. Fig. 3 shows an example of how three tokens (A, B, and C) are reserved for three channels. Nodes are connected by bus segments and channels typically use a subset of the segments on the bus (gray color) and the rest are reserved (white color) but left unused and thus wasting





THIS SHEET HAS BEEN
REPLACED

WO 97/24844

12

PCT/SE96/01748

shared resources. A better alternative is to let channels only reserve capacity on the segments between the sender and the receiver as the example in fig. 4. A single slot may in this case be used multiple times on the bus. Channel D and E are
5 using the same slots as channel A and C but on different segments. This is referred to as slot reuse. Slot reuse enables simultaneous transmissions in the same slot over disjointed segments of the bus.

Slot reuse is a general method to better utilize shared links
10 in ring and bus networks. The slot reuse algorithms in DQDB (Distributed Queue Dual Bus), Simple and CRMA (Cyclic Reservation Multiple Access) depend on control information in the slots. Buffer insertion networks when combined with destination release as in METARING may reuse capacity of
15 individual links and resolve eventual contention by delaying the incoming packet stream through an elastic buffer.

With slot reuse, the access scheme complexity is increased, regardless of whether it is done in hardware as in DQDB, Simple, and CRMA; or in software, as in DTM. When implemented
20 in systems other than DTM, slot reuse also add complex hardware to the critical high-speed path through a node and therefore increases the node delay.

To allow slot reuse in DTM the block token format is according to the invention extended to include parameters describing the
25 segment(s) it is representing. The token management protocol is also according to the invention modified to avoid conflicts in the slot number dimension as well as the segment dimension. The most important assumption is that no hardware changes to the original prototype implementation where allowed or needed. The
30 performance gain is also quite clear: on a dual-bus where

This Sheet Has Been Replaced



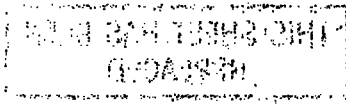
source and destination pairs are uniformly distributed it has been shown that throughput may be increased by a factor of two. The performance gain can be even higher in other network designs; e.g. in a double ring structure with source and destination nodes uniformly distributed throughput may be increased by a factor of four. Having nonuniformly distributed source and destination nodes the gain may be much higher.

The potential drawback of slot reuse in a DTM network however, is the higher algorithm complexity and eventually higher load on the node controller and signaling channels (especially if the average channel duration is short).

Two token management schemes have been evaluated. The first and simplest is to let a single node controller manage all free tokens for one fiber. This sort of centralized server mechanism has also been used in systems such as CRMA, where the head-end node distributes the fiber capacity to all other nodes. The simulator was configured so that for each fiber a node one third of the distance from the slot generator was token server (with 100 nodes on a bus node 33 and 67 will be token servers). This corresponds to the mid-point of demand for each of the two unidirectional fibers such that a token manager will have the same amount of traffic on both sides.

Each time a user request arrives at a node, the node first requests tokens from the manager and then locks them throughout the channel life time. When the user issues a request to disconnect the channel the tokens are returned to the manager immediately. All requests are delayed during the token request and accesses are serialized through the central manager.





The distributed token manager is fundamentally more complicated than the centralized. We tried to keep it as simple as possible. In our method each node regularly broadcasts status information about how many free tokens it has. The other nodes
5 store this information in their status tables. A node that wants more capacity consults its status table to decide from which node to request slots. The protocol on the initiating side works as follows. When a user request arrives to a node:

1.If the node has sufficiently many free tokens to satisfy the
10 request, it allocates the requested amount of slots to the user, and starts the channel by sending a channel establishment message to the destination node, and then transmitting data using the reserved slots.

2.Otherwise the node marks its available tokens as being
15 reserved, and then checks its status table: if the total amount of free tokens in the network is not enough to fulfill the request, then the request is rejected (blocked). Otherwise the node requests tokens from nodes with unused capacity.

20 If one of these nodes receiving a token request does not have the requested amount of free slots, it still gives away all of the slots it has. In any case, it sends a response back to the requesting node. A node fulfills incoming requests in strict FIFO order.

25 When a node receives a response to a token request, it marks the slots it receives in the response (if any) as being reserved. When the node has received responses to all requests it has sent, it either starts the channels or rejects the user request, depending on whether or not it has



acquired sufficient capacity. If the user request is rejected, the reserved slots are marked as free again.

At startup all free tokens are evenly distributed among the network nodes and each node takes at least one of its free
5 tokens, moves it (them) to the active state and declares it (them) to be a control slot. User requests may now be accepted and tokens can be moved between nodes on demand.

A weakness with this scheme is that slot reallocation is only triggered by user requests, and user requests are delayed by
10 pending token reallocation. An optimization we have implemented to remedy this is to also perform slot reallocation in the background. This results in less need to reallocate tokens for small to medium sized requests.

The pool of free tokens may be distributed in other ways than
15 evenly to increase reallocation success rate and utilization. If fewer nodes are managing the pool, channel blocking decreases as a result of a lower probability of token reallocation failure.

The complete token pool is in this case distributed
20 proportionally (nodes close to the slot generator get more tokens than nodes far away from it) among all nodes. Token transfers may occur between any pair of nodes instead of always involving the server node. When the local node contains enough tokens to fulfill an arriving user request, the request may be
25 accepted without any token reallocation. Furthermore, as long as the arriving user requests are well matched to the pool distribution no reallocation will ever be necessary.

Several questions need to be answered before deciding how to distribute the token pool. The following issues are addressed:



1. When local resources in the node are not enough to satisfy a user request, which other node should be asked for more tokens?
 2. If a node asks for tokens from several nodes, how many tokens is it going to ask for and should the node reject the channel if it receives a fraction of the requested capacity?
 3. If tokens move freely among the nodes, will the token pool be fragmented into small pieces and render the block token optimization scheme useless?
- 10 Status messages were decided to be used to distribute information about the pool of free tokens. Status message information is used to help a node choose a suitable node when asking for more resources. This method addresses the first question above.
- 15 Our scheme works as follows. Each node regularly broadcasts status information about how many free tokens it has. The other nodes store this information in their status tables. A node that wants more capacity consults its status table to decide from which node to request slots. The broadcasted state
- 20 information gives an approximate and dated view of the current state of token information, so token requests may be rejected because they were sent to nodes that no longer have tokens to give away.

Status tables are "soft" information in the sense that the

25 system will work even if they are out of date or not available. They should, however, improve the success rate of reallocation procedures.

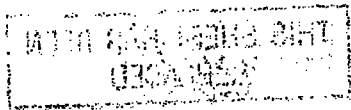


When comparing basic performance of the centralized (fig. 9) and the distributed (fig. 12) token manager we see that there is a new type of rejections that is frequent in the distributed version when resources are still unused in the system.

- 5 A node uses the status table to pick node(s) from which to request tokens. When the request reaches the target node the amount of available capacity may have changed and a smaller amount than requested may be returned to the requesting node, resulting in a user rejection.
- 10 This behavior results in even more unnecessary token movement and therefore also increases the probability of other nodes receiving fewer than requested tokens. The tokens that are moved in this case are also locked and unused during the transfer.
- 15 If the pool is distributed proportionately among a large number (100's) of nodes, the average pool size will be relatively small. When the load is high, the number of free tokens in pools decreases even further. If nodes also create and destroy channels at a very high rate, the amount of free capacity in
20 individual nodes jumps between having a small amount of capacity and having none. Now, if the average capacity requested by a user is large compared to the number of free tokens of a node, several nodes need to be asked to fulfill the request. At this point the probability that one of the
25 requested nodes do not have any free capacity increases, leading to a user rejection.

There are several ways to address this problem without going back towards a centralized model. First, we may not give away any slots at all when the complete request cannot be fulfilled.





THIS SHEET HAS BEEN
REPLACED

WO 97/24844

18

PCT/SE96/01748

This protocol applies if only a single node is asked for free tokens but if several nodes are asked it may still result in tokens being moved or locked unused. Second, if after a token request, we receive fewer tokens than we requested we may
5 simply retry the token request procedure several times. This will increase the probability of accepting a user request and that received tokens will be used. The cost of retry will be increased signaling and increased access delay, and it may worsen performance in an overloaded network. Also user retry
10 introduces longer set-up delay for requests that are retried. Third, the user may sometimes want to accept a channel with less than requested capacity instead of being rejected.

If, for example, a user receives 50% of the requested he may want to accept. In fig. 5 throughput, which is the ratio
15 between transported user data and total number of bits transmitted by the fibers, and access delay, which is the time from the arrival of a user request to the sending of the first data of the request, for small (16 kbyte) user requests with various minimum acceptable capacities [100% (40 slots), 50% (20
20 slots), and 5% (1 slot)] are plotted versus offered load. A lower average minimum acceptable bandwidth will result in higher throughput. In fig. 6 is shown the performance that results if user (requesting a 16 kbyte data transfer) retries up to 8 times before the request finally is blocked.
25 Utilization increases (and blocking decreases) at the cost of more signaling and longer delay. Retrying a large number of times is counter productive if it happens frequently.

Clearly the benefits of a flexible user request policy, is lower probability of being rejected and higher overall
30 throughput. Either of the configurations presented in fig. 5

This Sheet Has Been Replaced



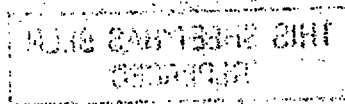
and fig. 6 may be decided at arrival time of a request. A user that has strict demands on the capacity for a channel may retry until enough capacity is allocated, but another may rather accept a channel with less than the requested amount. For the
5 rest of the simulations presented here the minimum acceptable bandwidth is defined to be 50% of the requested capacity.

In the general case, the average number of contiguous free blocks in a node is small, due to the random movement of tokens and the varying capacity of users request. This fragmentation
10 renders the block token optimization practically useless, and the access delay is relatively long (milliseconds) for high capacity channels. To make block allocation efficient it is necessary to reduce fragmentation of free tokens, otherwise
15 fragmentation will be by far the main contributor to access delay for high bandwidth channels at moderate to high load. Low capacity channels will almost always have a very short channel establishment delay independent of the current amount of
20 fragmentation. In the case of slot reuse, the fragmentation problem is even worse, as fragmentation may occur in both slot (time) and segment (space) dimensions (see fig. 4). This is in the centralized server version a particular application of the general dynamic storage-allocation problem. In the distributed token manager most of the fragmentation is a result of using
25 many free pools (one for each node). Two free adjacent tokens can only merge if they are found in the same node.

A distributed scheme that tries to avoid fragmentation if possible and increases the average block size of free tokens in the nodes has been implemented. The scheme is used both with and without slot reuse.

30 The scheme works as follows:





1. Define a home node for each token at network startup and distribute tokens in such a way so tokens sharing the same home node will always define a continuous slot range. This results in a large average token area in the token map shown in fig. 4.
2. When two slot-consecutive tokens with the same slot or segment range exist in the free pool merge them into a single token (sometimes a recursive split and merge operation is needed). When doing a merge always prioritize segment merge before a slot number merge. (The reason for this is that tokens spanning over just a few segments are less useful to other nodes than tokens spanning over many segments). Two segment-consecutive tokens, representing at least partly the same slot range and existing in the free pool of a node, are splitted to obtain segment-consecutive tokens representing the same slot range, which are merged into a single token.
3. When a node get a token request from the local user or a remote user, pick a token from the token pool using a best-fit algorithm in slot number and segment number dimension (see fig. 4). The value of a token is calculated as the area of a token in the token map and we tries to pick the token with the smallest area that fulfill the requested capacity. A cost function can also be defined as a function of e.g. number of slots, number of segments, location of slots and location of segments, which function should be minimized, however, still fulfilling the requested capacity. This mechanism can also be used by the servers when using a centralized token manager.
4. When a node needs to request tokens from other nodes it does not ask for small chunks from several nodes if it is possible



to ask for larger chunks from fewer nodes. The status tables provide this information. Transfer of tokens is therefore more efficient, and there are fewer establishment messages and less fragmentation.

- 5 5. Free tokens are sent back to home nodes when they have been idle for a significant time or after a long transfer.

This scheme returns tokens to "home" nodes as a way to increase the probability that two consecutive tokens can be merged in the free list which decreases fragmentation. If home node
10 "gravity" is too strong the scheme will result in less sharing of resources and unnecessary signaling. If it is too weak, fragmentation will still remain a problem. The "gravity" may be changed during the operation of the bus.

To evaluate the defragmentation mechanism we did another set of
15 simulations. Three different simulators [A, B, C] were configured. Simulator A was configured to have no fragmentation at simulation start time and to use the defragmentation scheme described above. B did start with maximum fragmentation of the complete resource pool. All tokens had a single slot and no
20 tokens in the "home" nodes before the defragmentation mechanism was turned on. Finally, simulator C was started without using the defragmentation mechanism and with the pool having maximum fragmentation. In all cases slot reuse were turned on and load was defined to be 80%.

25 In fig. 7 is shown the access delay as a function of simulated time for a 10 km network. Simulator C started with a long access delay and the delay increased as the signaling channels became overloaded and message queues grew. Simulator B using the defragmentation mechanism starts just as bad as C but



already after 10 milliseconds the average access delay is below 500 microseconds. Later when one second of simulated time has passed the B curve almost joins A i.e. it converges on the performance of the simulator starting without any fragmentation at all. The convergence speed depend on the amount of free capacity in the network and therefore also on load. The load during all these simulations was 80%. The defragmentation mechanism clearly improves the access delay and also makes the block token optimization meaningful in the distributed implementation.

Two kinds of performance measures are of main interest: utilization and access delay. Utilization is the portion of the nominal network capacity that is actually used for data transfer, and is a measure of the efficiency of the network. Access delay is the time from the arrival of a user request to the sending of the first data of the request, which is an important measure for how well computer communication traffic can be supported.

There are two main factors that influence utilization in DTM. First, each node is assigned signaling capacity in the form of control slots, which means that there are fewer slots available for data transfer on a bus with many nodes, given fixed link capacity. Secondly, token reallocation incurs overhead since while a slot token is being reallocated between nodes, the corresponding slot cannot be used for data transfer.

Access delay depends mainly on the load on the control slots, and on how many control messages need to be sent to establish a channel. The access delay is typically a summation of a few delays: e.g. node controller processing delay [5 μ s], delay in



finding and allocating free tokens [100 μ s], waiting for the first available control slot to pass [50 μ s], and finally waiting for the first allocated data slot to be filled with user data [62.5 μ s]. In addition, messages are delayed in
5 queues at the input to node controllers waiting to be processed. In the simulations presented below the average delay is up to a few hundred microseconds.

Results from simulations where DTM is exposed to traffic patterns that are more like the relatively short-lived
10 transfers (4-4000 kbyte) seen in computer communication will be presented. Traffics are with bursty inter-arrivals, client-server oriented as well as with exponentially distributed arrivals. In the simulation model, each transfer starts with the arrival of a new "packet" of information. The node
15 controller tries to allocate resources for the transfer, transmits the data and finally releases the channel. This is a simplification of the mechanisms in a real system, where channel set-up, data transfer and channel teardown are independent operations initiated by the user. For example, a
20 user that knows that a transfer is about to take place may "hide" the channel establishment delay by requesting a channel in advance, so that it is already established when the transfer starts. Between a setup and teardown, the capacity of the channel is completely reserved for the user. The most straight-
25 forward use of the channel is for a single transfer, such as a file transfer or a video transmission.

Depending of the characteristics of the application, it is possible to optimize the usage of the channel. For instance, a channel may be used to transfer sequences of higher level
30 messages such as ATM cells or IP packets (this is similar to



using a single VC for all IP traffic in an ATM network). If the channel is a multicast channel, messages to different destinations can be multiplexed onto the channel. This means that each message will reach every receiver on the multicast
5 channel and receivers must be able to filter messages. An alternative solution is to create and destroy a channel for each message, but reserve the tokens between messages so that the tokens are readily available for the next message in the sequence.

- 10 This type of user behavior is not incorporated in the simulations, since they are optimizations for particular applications. Instead attention is paid to how the network performs without user level optimizations.

The sender may start sending data as soon as the resources are
15 allocated, even before the receiver receives the channel establishment message. This is called *channel establishment*. The receiver will eventually respond with a control message to accept or reject the channel.

User requests have the following parameters:

- 20 • Packet size which is the amount of user data transferred between channel establishment and channel release. We simulate packet sizes from a few kbytes up to a few Mbytes.
- Requested capacity for a channel, which is the number of slots that a node tries to allocate. For all the simulations
25 in this paper the requested capacity is fixed to 40 slots or 20.48 Mbit/s.



- *Minimum acceptable capacity.* A node blocks a request if it cannot allocate this number of slots. This is normally set to 40 or 20 slots (100% or 50% of requested capacity).
- *Source address.*
- 5 • *Destination address.*

The source and destination addresses are generated randomly (all nodes with the same probability) and user inter-arrival times are exponentially distributed. The simulations investigate the effect of signaling capacity and slot
10 reallocation overhead on utilization, channel set-up delay, and blocking. We simulate varying traffic conditions, for a topology with the following characteristics:

- A dual bus network with 100 nodes. Even though it is theoretically possible to connect many more nodes to a bus,
15 we think that management of networks with more than 100 nodes on a bus may be infeasible. With 100 nodes the capacity sharing is significant enough to exercise and test the token management protocol.
- The capacity of each bus is 6.4 Gbit/s. We believe this is
20 realistic for what is realizable within a year or two; 2.4 Gbit/s optical links have been available for a few years, and 10 Gbit/s links have been announced to appear soon on the market. 6.4 Gbit/s corresponds to 100 MHz slot rate, which is the rate at which the slot processing MAC hardware would
25 operate this speed attainable with current CMOS technology.
- The total signaling capacity is the same for all nodes but the slots are partitioned between the two fiber directions



proportionally depending on where the nodes are located on the bus. The closer a node is to the slot generator the more control capacity is needed. The sum of the control capacity on both buses will however be the same for all nodes. In the
5 network with two token servers, the servers has more control capacity and higher processing capacity than the other nodes.

- The length of the bus is 10 km, which gives a large enough network that the effects of propagation delay are not negligible. Results from studies of the effects of
10 propagation delay in simulations with different bus lengths are presented in fig's 11 and 13.
- Two different token management schemes are simulated: an asymmetrical scheme where all the tokens on one fiber are managed by a single token server and a symmetrical scheme
15 where each node controller manages a small piece of the global token pool.

When analyzing the performance of the DTM dual-bus network the issue of maximum theoretical performance must be addressed and compared to the simulated performance. The maximum theoretical
20 performance is also used in this paper to compare the different schemes and implementations we are evaluating.

The maximum throughput of a dual-bus system without slot-reuse can be defined as twice the link capacity, given that both fibers receive the same traffic. In a system with slot reuse
25 the system throughput also depends on the source destination distribution. To obtain this throughput for the dual-bus we used a Monte Carlo simulation where source and destination addresses were uniformly distributed (see left graph of fig. 8) Comparing this with a DTM network (simulator configuration as



defined below using a centralized token manager) where the user requests large transfers (4 MB at 20 Mbit/s capacity) and signaling capacity is not a bottleneck (see right graph of fig. 8), utilization is close to the ideal case. Real traffic

5 behaving like this is bulk-data transfers and audio/video streams. The differences that are shown are a result of: Some capacity is in DTM used for control slots lowering the available number of slots to be used by data transfers. The random generators for the DTM system do not generate exactly

10 the same amount of traffic in upstream and downstream direction. This may result in blocking in one of the directions when capacity is available in the other. During channel establishment resources may be locked unused momentarily wasting some capacity.

15 In the case of central token manager the two managing nodes need more signaling capacity than other nodes (Eight times as many control slots to a server node than to other nodes are assigned).

Results of the first set of simulations are presented in fig.

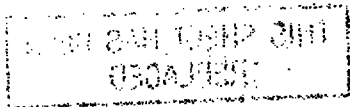
20 9. User request 20 Mbit/s channels, inter-arrival times are exponentially distributed (generated by a Poisson process) and the simulations are performed with different packet sizes. If the full capacity of the channel cannot be allocated, the request is rejected and tokens are returned to the token

25 server. Packet sizes vary from 4 Mbyte to 4 kbyte, at which point we begin to see throughput degradation.

Throughput degradation may occur if processing capacity in nodes or control channel capacity is too small. The server nodes may be especially overloaded. The result is that queues

30 containing control messages start growing very big. The control





THIS SHEET HAS BEEN
REPLACED

tokens represent unused capacity, and therefore the throughput is degraded.

In the 4 kbyte per channel simulations, the control capacity is the limiting factor and if more control slots (signaling capacity) are added, 4 kbyte and even smaller packets can be more efficiently supported.

The next set of curves in fig. 10 show how the slot reuse mechanism improves the performance of the system. Throughput increases by almost a factor of two before any significant number of channels are rejected. The uniform source and destination distributions of channels limits the amount of increased capacity gained by slot reuse. It has been shown that if the source and destinations are generated evenly, as we do, throughput may be doubled on a dual bus. In the simulations it may also be seen that beyond an offered load of 2.5 we may actually get throughput higher than 2.0. This level of throughput, however, can not be reached without some channels being rejected. The channels with the highest probability of being rejected are the ones that use many slots or segments. Therefore the system "filters" less greedy user requests and throws away the rest. This is normally not an acceptable behavior and we therefore do not investigate this any further. Throughput degradation occur at an offered load of 1 with the 4 kbyte transfers. Even if enough resources were available in the token server it is not possible to establish and destroy channels fast enough as the control channel is congested. Further is also seen a throughput degradation of the 8 kbyte simulations at an offered load of 1.8, for the same reason.

It may be concluded from the simulations in fig. 10 that the slot reuse mechanism almost doubles the system throughput with



only minor changes to the centralized token protocol as long as the control and server processing capacity is not a bottleneck. From the curves it can also be unexpectedly seen that access delay actually decrease when the load increase from 0.1 to 0.5.

- 5 This is a result of how slots are assigned to a channel and not from a faster token request procedure. The time for it takes to request tokens from the server is strictly increasing.

When comparing the DTM performance in fig. 10 and the theoretical values in fig. 8 it is seen that even short bursts
10 (a few milliseconds duration) can be supported efficiently.

When a single token server is used, each channel establishment first requires a token to be requested from the server before the node can establish the channel. If the length of the bus is increased the token request will take a longer time and may
15 therefore also limit the throughput and increase the access delay.

In fig. 11 is shown the result from simulations, in which the length of the bus is increased from 100 to 1000 km (node to node distance is now 50 μ s). Both access delay and throughput
20 are now limited by the round-trip latency to the token server.

Access delay in this case depends on the distance to servers, but is independent of the transfer size. Utilization depends strongly on the transfer size as the establishment phase may be amortized over a long data transfer phase. Channels
25 transferring large amounts of information such as 256 kbyte with a duration of one tenth of a second are still efficiently supported when the bus length is 1000 km.



Using a centralized token manager has several benefits. Clients may be simple as they only contain state information related to their own opened channels. Slot reuse is also simple and efficient, as the slot server has all of the free tokens to
5 choose from when trying to satisfy a user request. The server may also implement other policy related functions such as admission control and fairness. The fragmentation of the free token pool in the server is normally very modest, resulting in very few connection establishment messages per channel even for
10 high capacity user requests.

There are also drawbacks. A user that frequently establishes and destroys channels may introduce excessive signaling by always returning tokens after use, but then requesting the tokens again within a short time period. The processing
15 capacity of the server node may get overloaded if there are many nodes on the bus or if the average packet size is very small. If the media length is very large relative to the product of bit-period, bits-per-packet and media velocity the round trip time to the server may also limit performance.
20 Finally the server node contains state information that all nodes depend on to create channels. A server node failure may therefore affect all the nodes.

In next section the properties of a fully distributed token manager are simulated and investigated.

25 When evaluating the performance of the distributed token manager with slot reuse and defragmentation the same traffic and parameters are used as in the case of a central token manager, but with a policy whereby requests are accepted if 50% of the requested capacity can be allocated.



The results presented in fig. 12 are from the simulator with slot reuse, a fully distributed token manager, status messages describing how much capacity a node owns, and the defragmentation scheme. All nodes have the same processing capacity and the processing load is much lower than what the servers in fig. 10 receive. The dependencies between nodes are also much smaller resulting in a higher degree of reliability. The system performs better than any system not providing slot reuse but not as good as the centralized system described earlier.

When comparing the performance of this configuration and the centralized token manager in fig. 10, one sees that blocking is higher and some requests may be blocked at much lower load in the distributed implementation.

One result not expected was that performance actually decreased when the packet size was increased! After checking the results again it was founded that a large average transfer size results in less movement of tokens and that the status information actually gives an even worse view of free resources in the network than for short transfers. In this case, a request is rejected if we do not think we will find any resources. This mechanism was introduced to avoid wasting control capacity when resources was exhausted.

The reason for this is that the status messages only describe "global" tokens covering all segments on the bus. A global token may be used by any node and is the only type of token in a DTM system without slot reuse. At a load higher than 1.00 a large number of tokens are segmented and the reuse scheme needs them on them being used by new requests. Therefore, the status message mechanism which is used, that was developed for a



system without reuse, is limited in its ability to help new requests find free capacity, and can in the worst case result in higher blocking.

- 5 The throughput and access delay of the distributed token manager with bus lengths from 1 km to 1000 km is presented in fig. 13. A small 16 kbyte packet is sent between establishment and tear-down of the channel, for large packets (100's of kbytes) the throughput performance of the system is much less affected of the bus length. The 1 km and 100 km busses give
10 about the same throughput and access delay results as the 10 km bus because the latency introduced by using a 125 μ s cycle dominates over the time of flight latency in the system. For the 1000 km long bus one see that the access delay is found to be much shorter than in the 1000 km system using a centralized
15 token server, especially at low load are tokens found very close to where the user request arrived and delay is about the same for all systems. Even at very high load the access delay is about 1 millisecond shorter than in the system with the centralized server.
- 20 The centralized token manager system will have the same performance almost independently of the traffic as long as processing and signaling are sufficient. To evaluate the distributed system two other traffic generators were used. First, we used a generator that simulate user requests arriving
25 in bursts. When a request arrives we defined a new request to arrive with 90 % probability after 200 μ s. The result is that a burst of requests arrives to a node and forces high temporal locality of the source addresses. Second, to generate traffic more similar to client-server behavior the amount of traffic

arriving were increased to 5 server nodes 0, 25, 50, 75 and 99. The probability of a server destination is also higher.

In fig. 14 is presented throughput and access delay performance of the distributed token server system.

- 5 It is clear that the distributed implementation have several benefits but also drawbacks compared to the centralized token server. The nodes may share the processing load, there is less need for high-performance token servers, redundancy may be higher and access delay may be shorter for low capacity
- 10 request. The drawbacks is higher blocking for an inflexible user which may therefore need to retry several times to gain access to necessary resources. It is also clear that the status message and status table mechanism must be changed to avoid unnecessary blocking when slot reuse is allowed.
- 15 When the load increases above a certain point, it gets difficult to find free tokens, blocking becomes significant and utilization only increases marginally with load. The degree of blocking at a given load depends mainly on the minimum capacity requirement of user requests. The lower the minimum capacity,
- 20 the higher the probability that the node will be able to satisfy the request. Furthermore, in these simulations only one round of token requests is sent out for each user request. Utilization can be increased by making several rounds (i.e., using retry), at the expense of longer access delay.
- 25 Assuming that the above waiting times are independent, the average medium access delay, if having a single control slot and if the network is lightly loaded, meaning that borrow is

infrequent, will result in an average access delay around 125 μ s (a single cycle).

In addition, due to fragmentation it may take many control messages to establish a channel, which adds even further to the load of the control channel, and thereby the access delay. In fact, without an efficient defragmentation scheme fragmentation is the dominating contributor to access delay in the simulations.

A consequence of DTM's slot reallocation mechanism is that it takes longer time to establish channels that need high bandwidth. This trade-off might be justified: the type of traffic that needs short medium access delay is usually less sensitive to the amount of bandwidth allocated for the transfer, and such traffic therefore can be accepted without much involvement of the reallocation protocol. For transfers that require high bandwidth, the access delay will be much longer and will almost always involve the reallocation protocol. However, in general high bandwidth transfers will probably be less sensitive to access delay.

These simulations have shown that the DTM fast circuit switching protocol performs well in a dual-bus shared media environment. Two slot (token) management schemes are analyzed and both perform well and can benefit from slot reuse. The centralized scheme performs closest to the ideal protocol and also result in a simple implementation. The distributed system is more sensitive to user behavior and must therefor rely on status information that is broadcast frequently and on the defragmentation method according to the invention to decrease the number of control messages needed for channel establishment

and slot reallocation. Using the defragmentation method for long busses the distributed model shows better performance than the centralized model (comparing fig's 11 and 13). A resource management system that combines the centralized and the distributed models using a small set of token server nodes is also possible.

Furthermore, connection establishment overhead can be designed to be very small, resulting in high utilization even for small (a few kbytes) transfers. Access delay is found to be a few hundred microseconds even at high load. The slot reuse method and arrangement can increase performance by a factor of two without introducing any extra hardware in the nodes. Using the slot reuse scheme it is even more important to use the defragmentation method according to the invention as fragmentation may occur in both slot and segment dimensions.



CLAIMS

1. A slot reuse method for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network characterized in
 - 5 - that the DTM block token format is extended to include parameters describing the segment(s) between source and destination node,
 - that the block token capacity is only reserved on the segment(s) between source and destination node, and
 - 10 - that simultaneous transmissions in the same slot over disjointed segments of the network are enabled.
2. A slot reuse method according to claim 1, characterized in that two or more segment-consecutive tokens representing the same slot range are merged into a single
15 token when existing in the free pool of a node.
3. A slot reuse method according to claim 1 or 2, characterized in that it is implemented in a central token management scheme.
4. A slot reuse method according to claim 1 or 2,
20 characterized in that it is implemented in a distributed token management scheme.
5. A slot reuse method according to claim 1 or 2, characterized in that it is implemented in a resource management system that combines the centralized and the
25 distributed models, hence using a small set of token server nodes.



6. A slot reuse method according to one of the claims 1-5, characterized in that it is combined with a defragmentation method in which

- a home node is defined for each token at network startup or during network operation or during network operation in such way that the tokens sharing the same home node will always define at least partly a continuous slot range,
- free tokens are sent back to their respective home nodes when a significant time has passed and,
- two or more slot-consecutive tokens with the same slot range are merged into a single token when existing in the free pool of a node.

7. A slot reuse method according to claim 6, characterized in that two or more tokens with the same slot or segment range are merged into a single token when existing in the free pool of a node.

8. A slot reuse method according to claim 7, characterized in that when merging slot- and/or segment-consecutive tokens which have the same slot or segment range always a segment merge is prioritized before a slot number merge even if the number of tokens will augment.

9. A slot reuse method according to one of the claims 1-8, characterized in that when a node get a token request from a local user or a remote user a token is picked from the token pool using a best-fit algorithm in slot number and segment number dimension, in which algorithm the token with smallest area in the token map that fulfill the requested capacity is chosen.



10. A slot reuse method according to one of the claims 1-8, characterized in that when a node needs to request a token from other nodes and no token that fulfill the requested capacity exists the number of tokens that are prerequisites is
5 minimized.

11. A slot reuse arrangement for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network characterized in that it comprises

- an extended DTM block token format including parameters
10 describing the segment(s) between source and destination node,
- nodes arranged to reserve block token capacity only on the segment(s) between source and destination node, and
- enabling of simultaneous transmissions in the same slot over disjointed segments of the network.



AMENDED CLAIMS

[received by the International Bureau on 28 May 1997 (28.05.97);
original claims 1-11 replaced by new claims 1-10 (2 pages)]

1. A slot reuse method for increasement of the throughput in a DTM
(Dynamic Synchronous Transfer Mode) network characterized
in
5 - that the DTM block token format is extended to include parameters
describing the segment(s) between source and destination node,
- that the block token capacity is only reserved on the segment(s)
between source and destination node,
- that two or more segment-consecutive tokens representing the same
10 slot range are merged into a single token when existing in the free
pool of a node, and
- that simultaneous transmissions in the same slot over disjointed
segments of the network are enabled.
2. A slot reuse method according to claim 1, characterized
15 in that it is implemented in a central token management scheme.
3. A slot reuse method according to claim 1, characterized
in that it is implemented in a distributed token management scheme.
4. A slot reuse method according to claim 1, characterized
in that it is implemented in a resource management system that
20 combines the centralized and the distributed models, hence using a
small set of token server nodes.
5. A slot reuse method according to one of the claims 1-4,
characterized in that it is combined with a defragmentation
method in which
25 - a home node is defined for each token at network startup or during
network operation or during network operation in such way that the
tokens sharing the same home node will always define at least partly a
continuous slot range,
- free tokens are sent back to their respective home nodes when a
30 significant time has passed and,
- two or more slot-consecutive tokens with the same slot range are
merged into a single token when existing in the free pool of a node.



6. A slot reuse method according to claim 5, characterized in that two or more tokens with the same slot or segment range are merged into a single token when existing in the free pool of a node.
7. A slot reuse method according to claim 6, characterized in that when merging slot- and/or segment-consecutive tokens which have the same slot or segment range always a segment merge is prioritized before a slot number merge even if the number of tokens will augment.
8. A slot reuse method according to one of the claims 1-7, characterized in that when a node get a token request from a local user or a remote user a token is picked from the token pool using a best-fit algorithm in slot number and segment number dimension, in which algorithm the token with smallest area in the token map that fulfill the requested capacity is chosen.
9. A slot reuse method according to one of the claims 1-7, characterized in that when a node needs to request a token from other nodes and no token that fulfill the requested capacity exists the number of tokens that are prerequisites is minimized.
10. A slot reuse arrangement for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network characterized in that it comprises
- an extended DTM block token format including parameters describing the segment(s) between source and destination node,
 - nodes arranged to reserve block token capacity only on the segment(s) between source and destination node,
 - two or more segment-consecutive tokens representing the same slot range are merged into a single token when existing in the free pool of a node, and
 - enabling of simultaneous transmissions in the same slot over disjointed segments of the network.

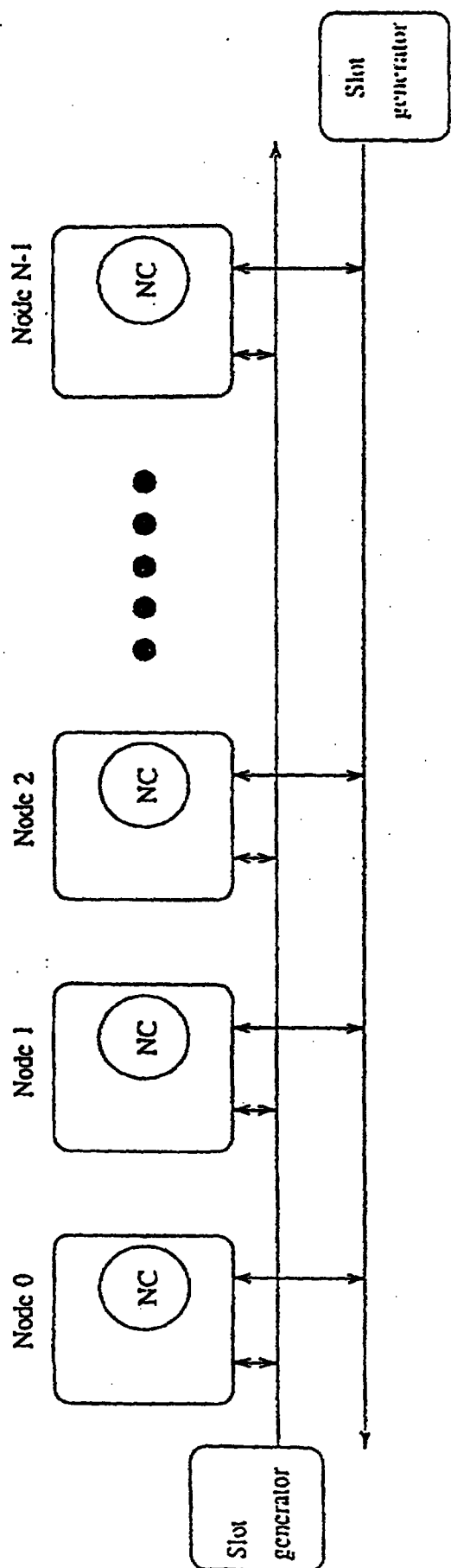


Fig 1

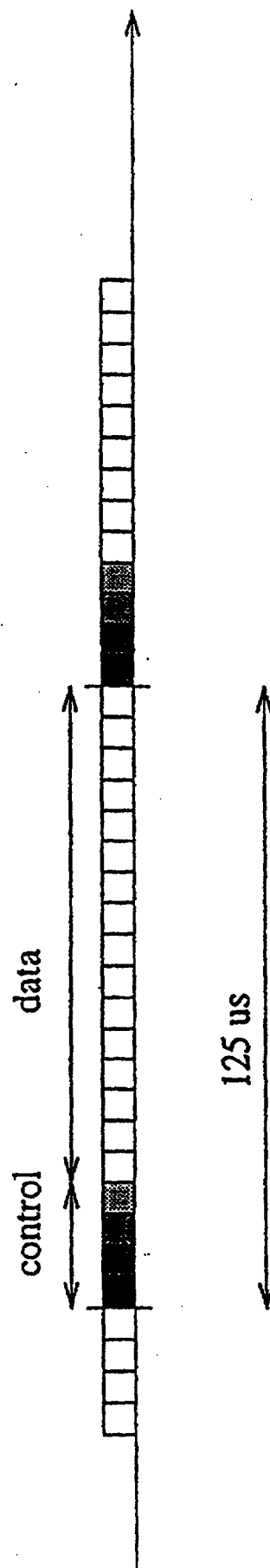
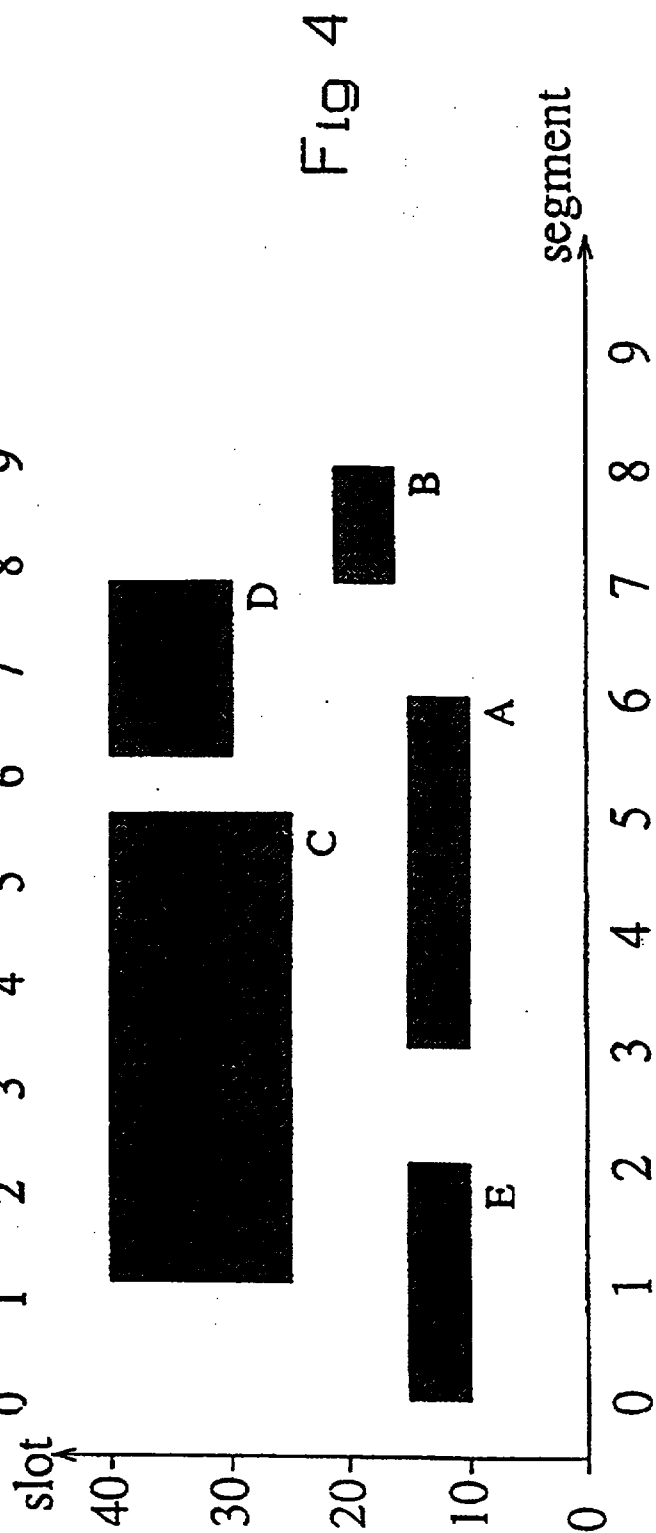
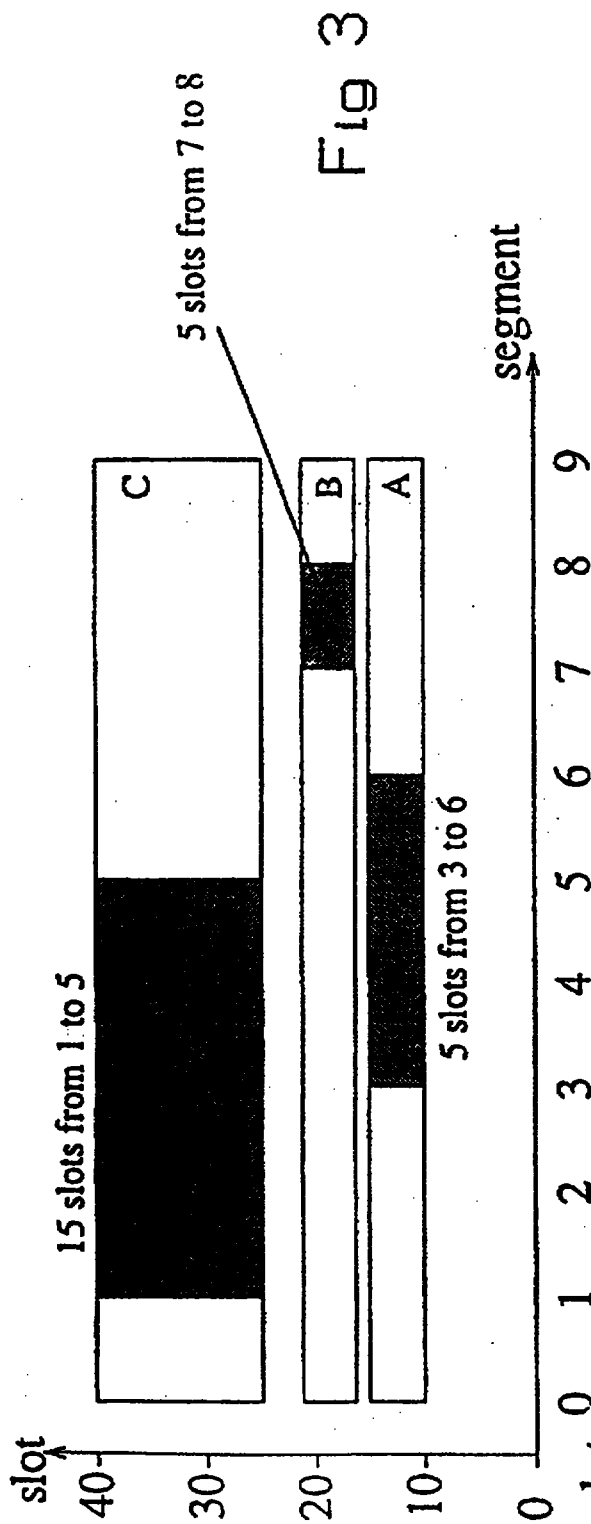


Fig 2



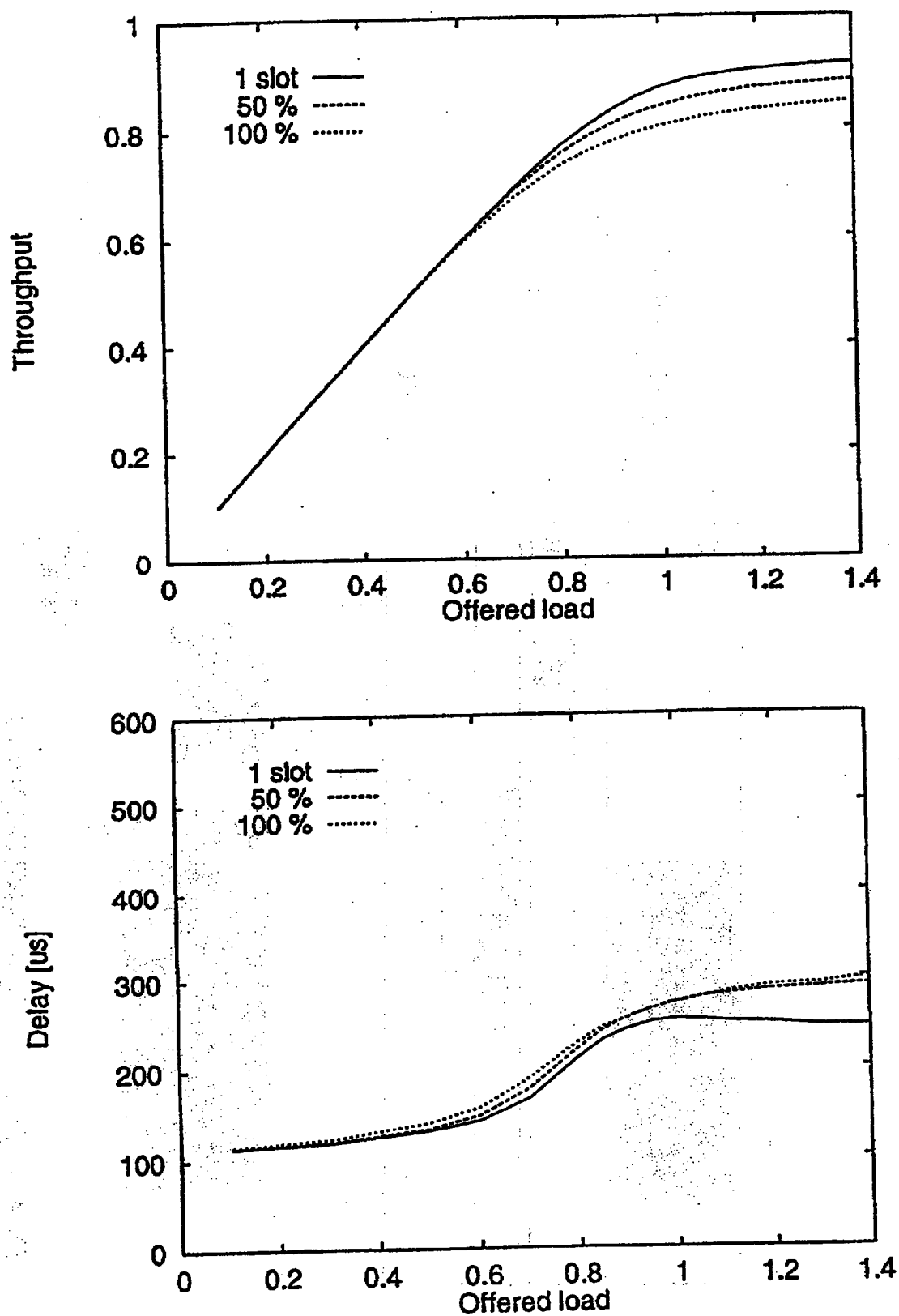


Fig 5

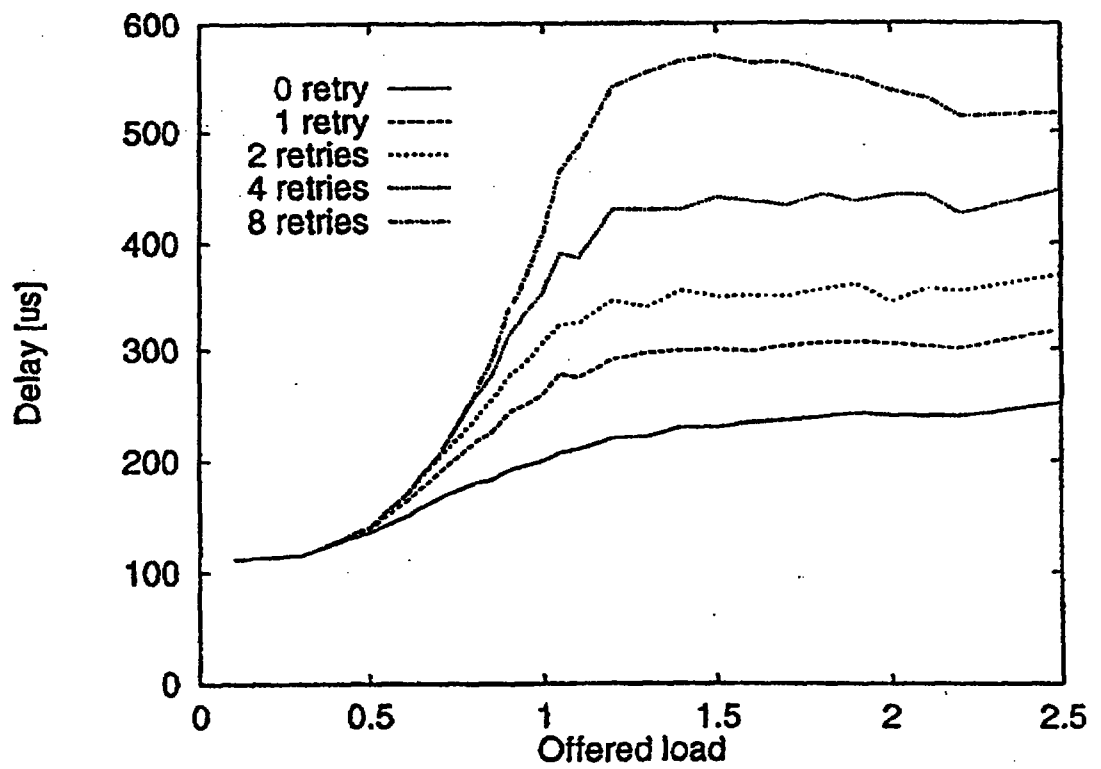
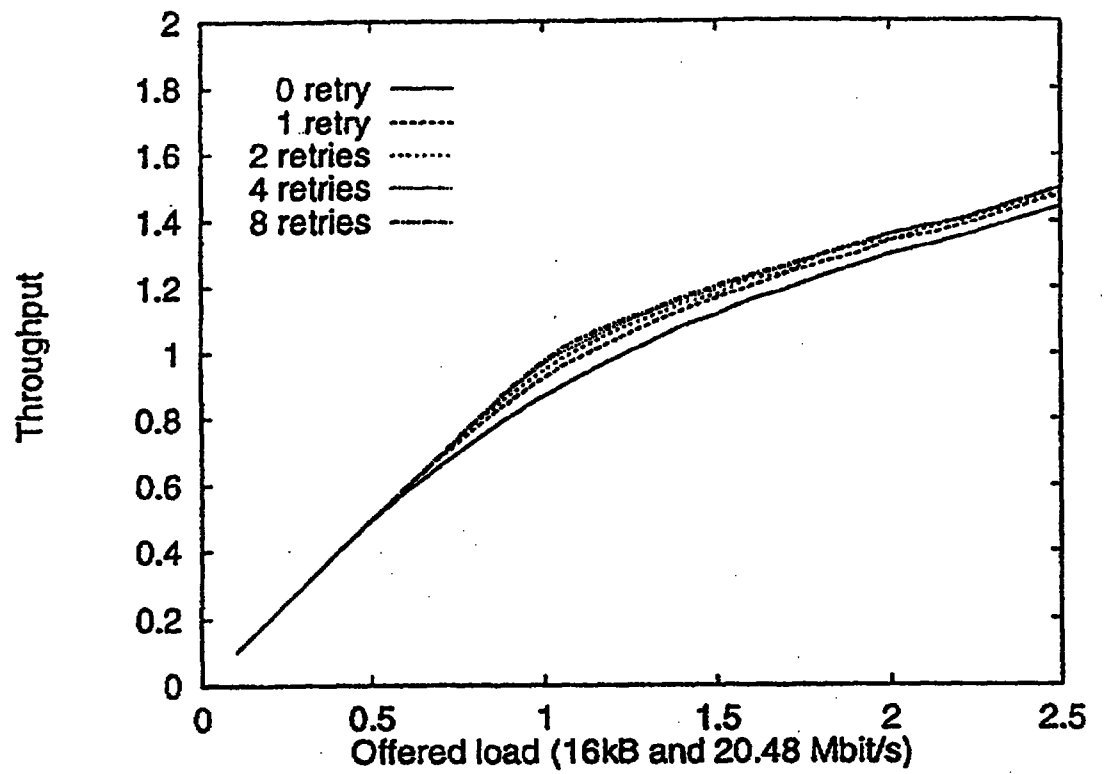


Fig 6

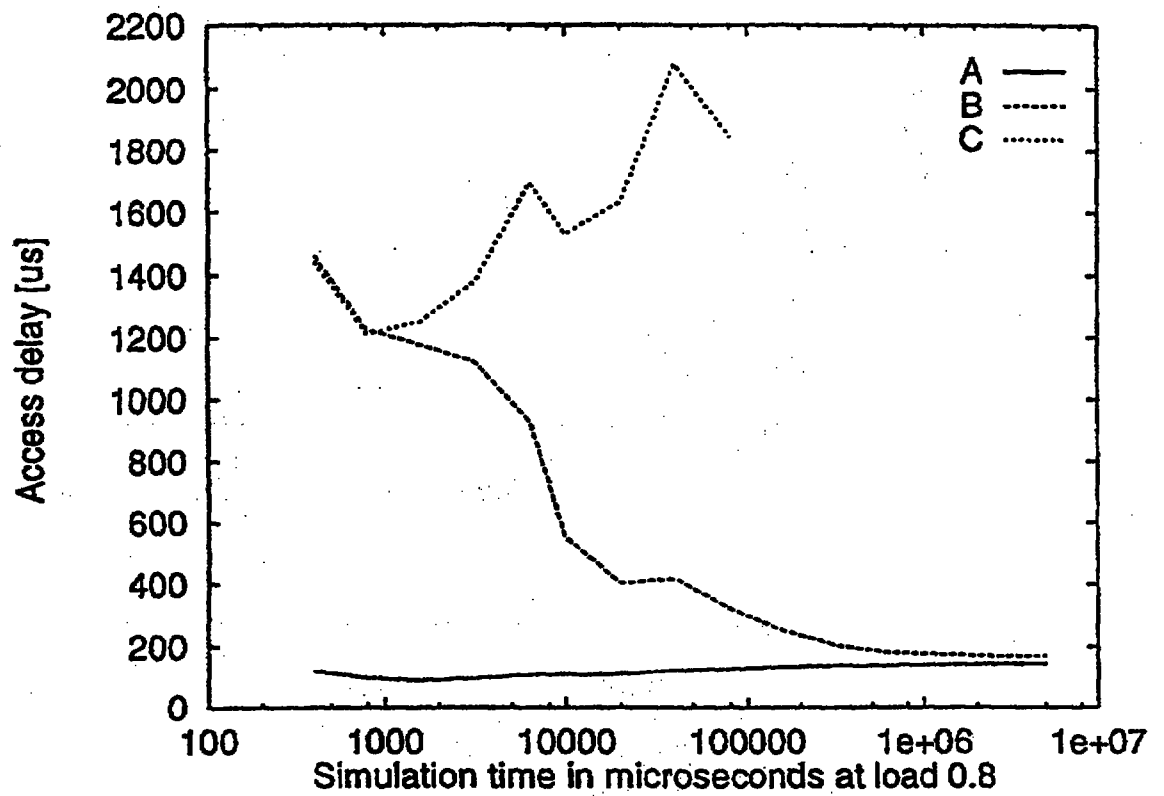


Fig 7

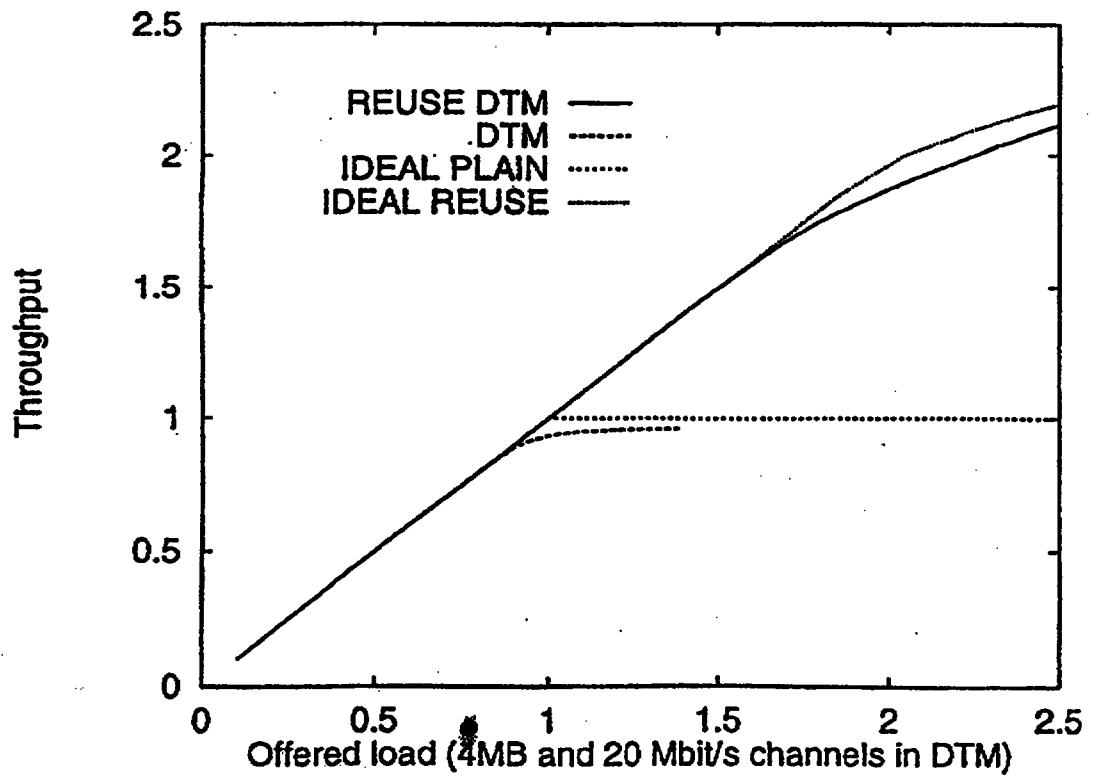
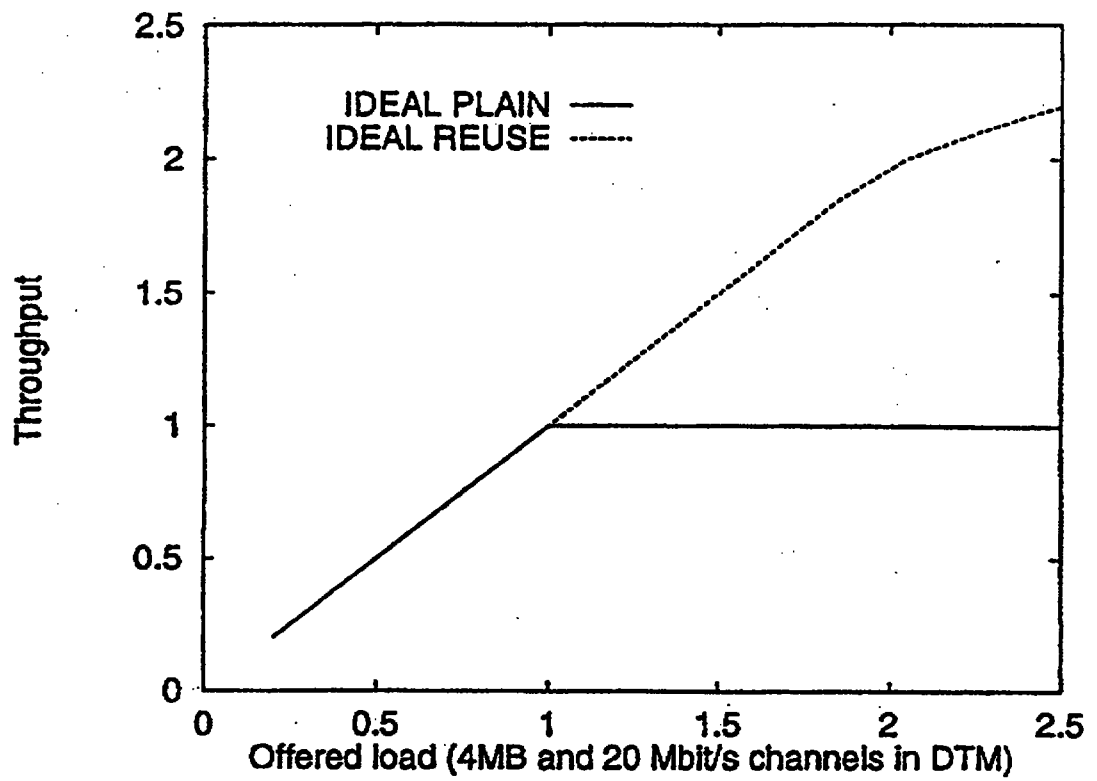


Fig 8

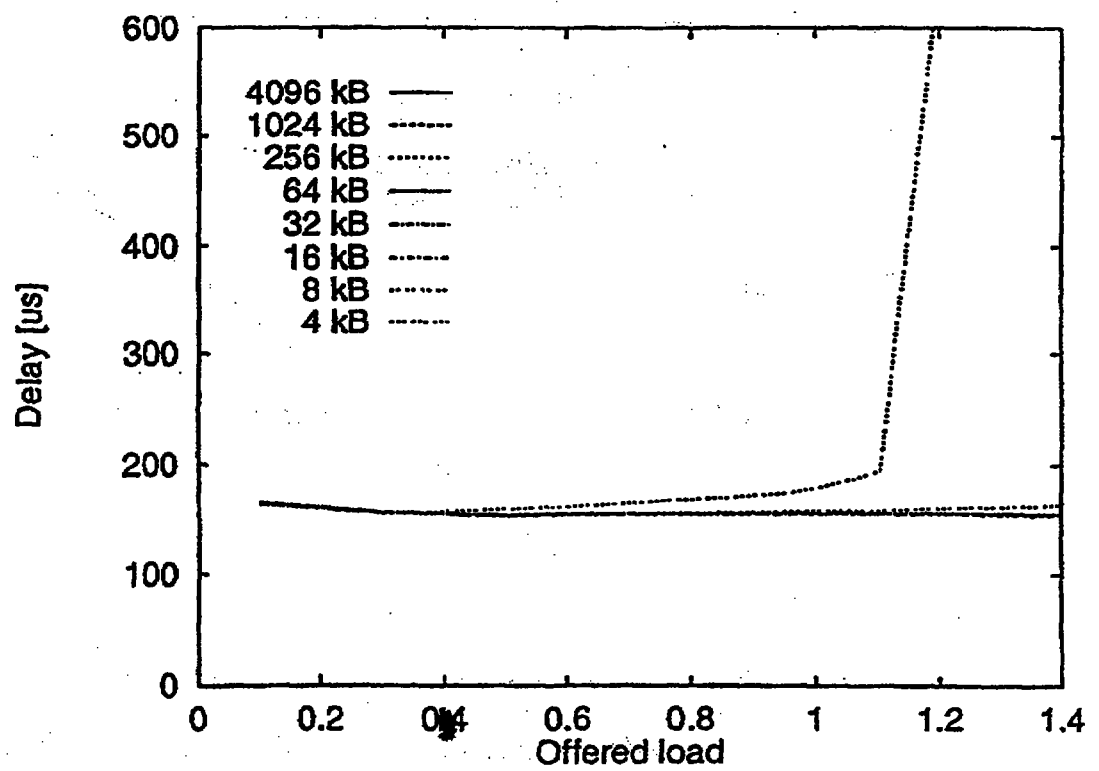
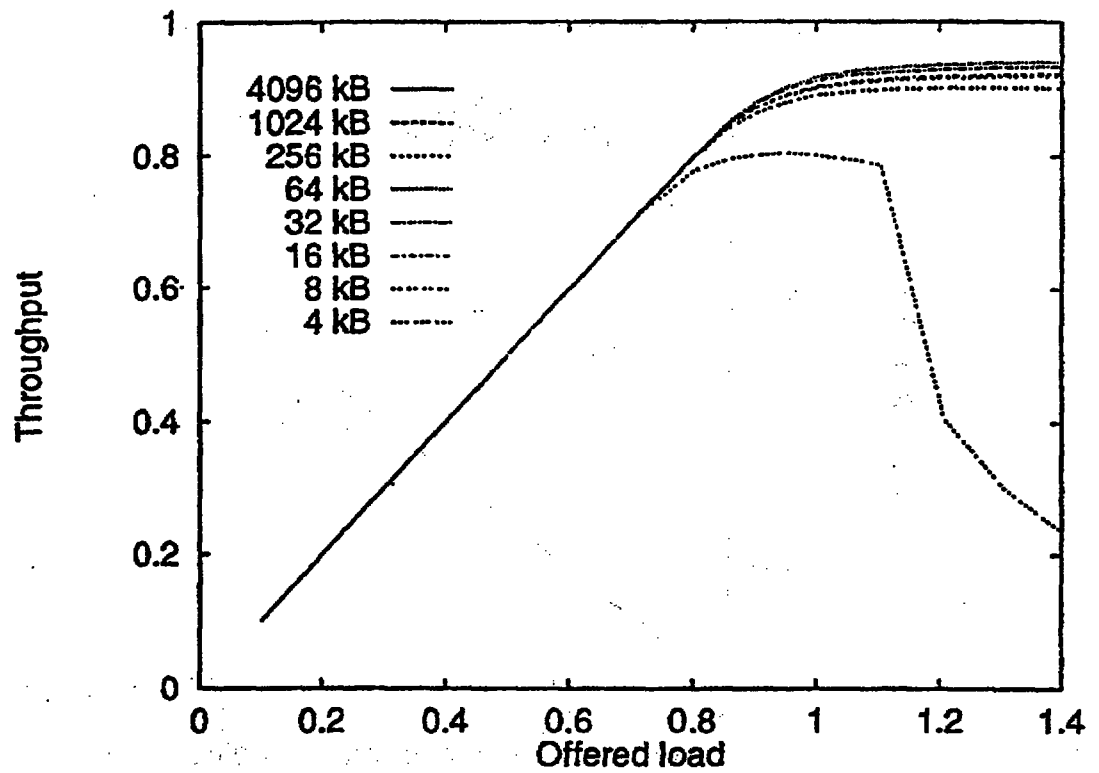


Fig 9

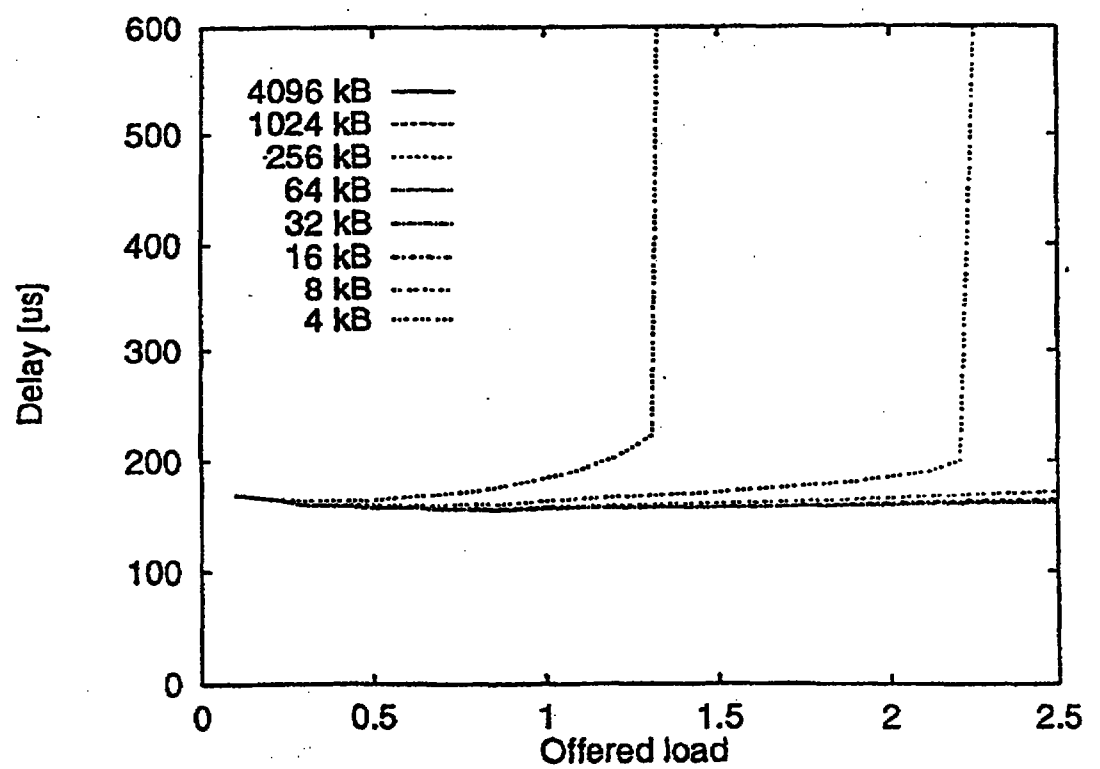
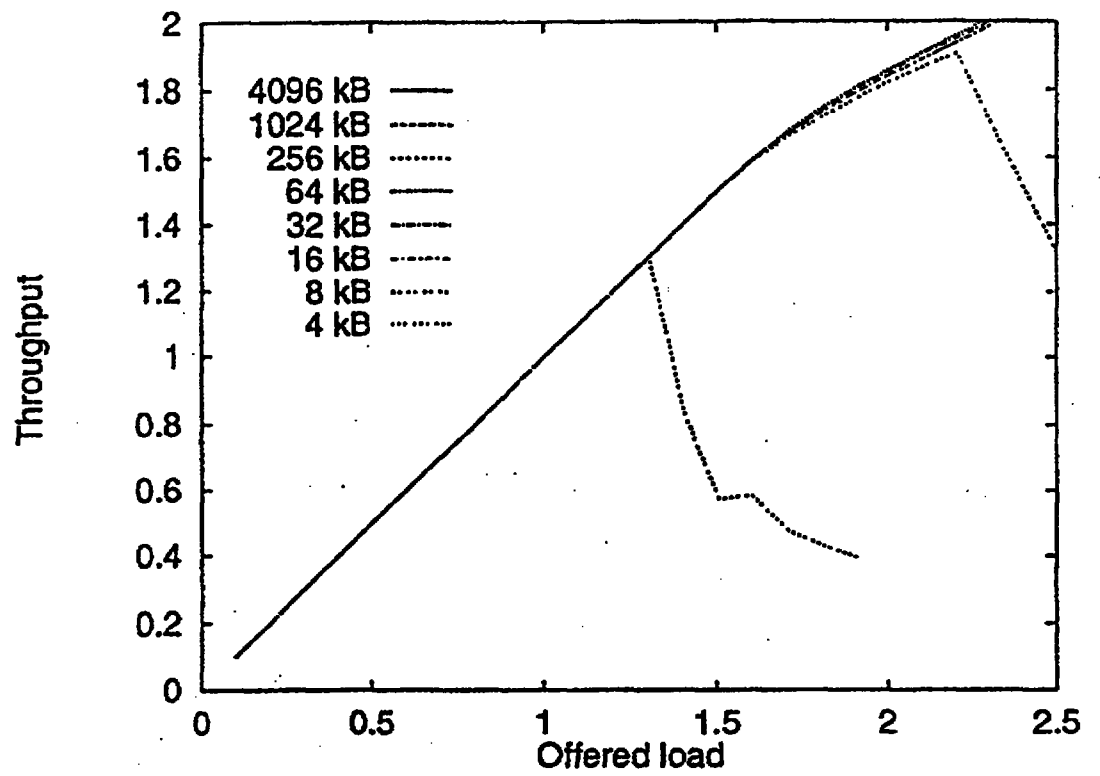


Fig 10

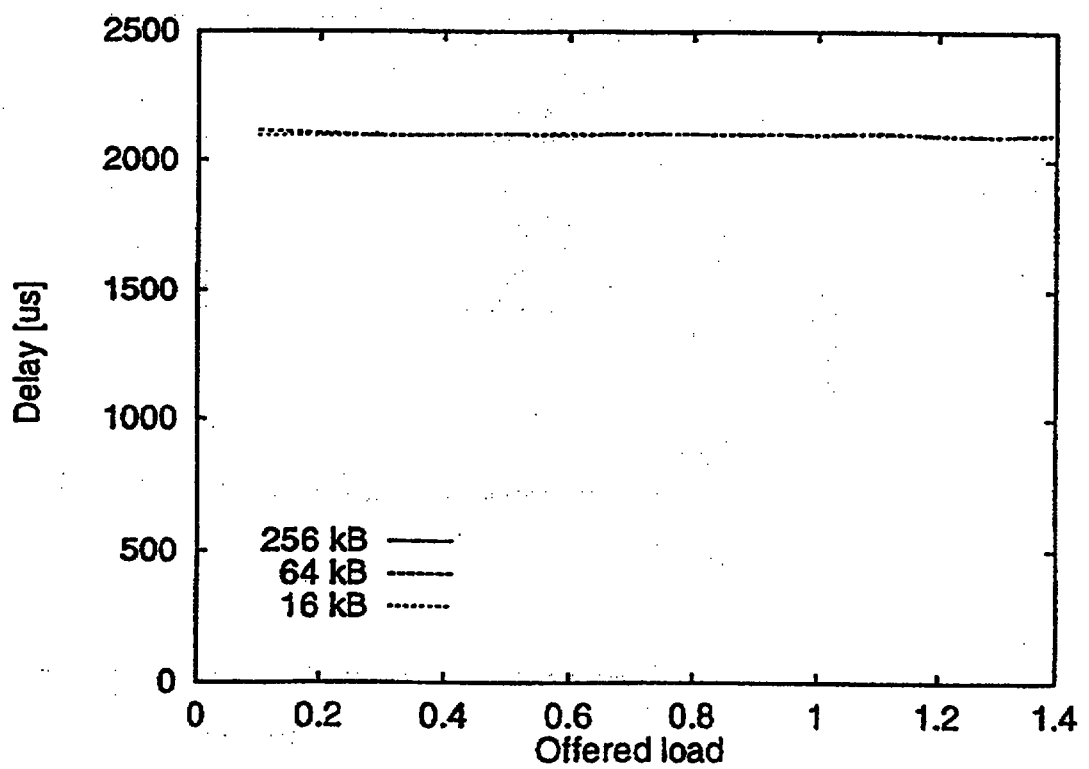
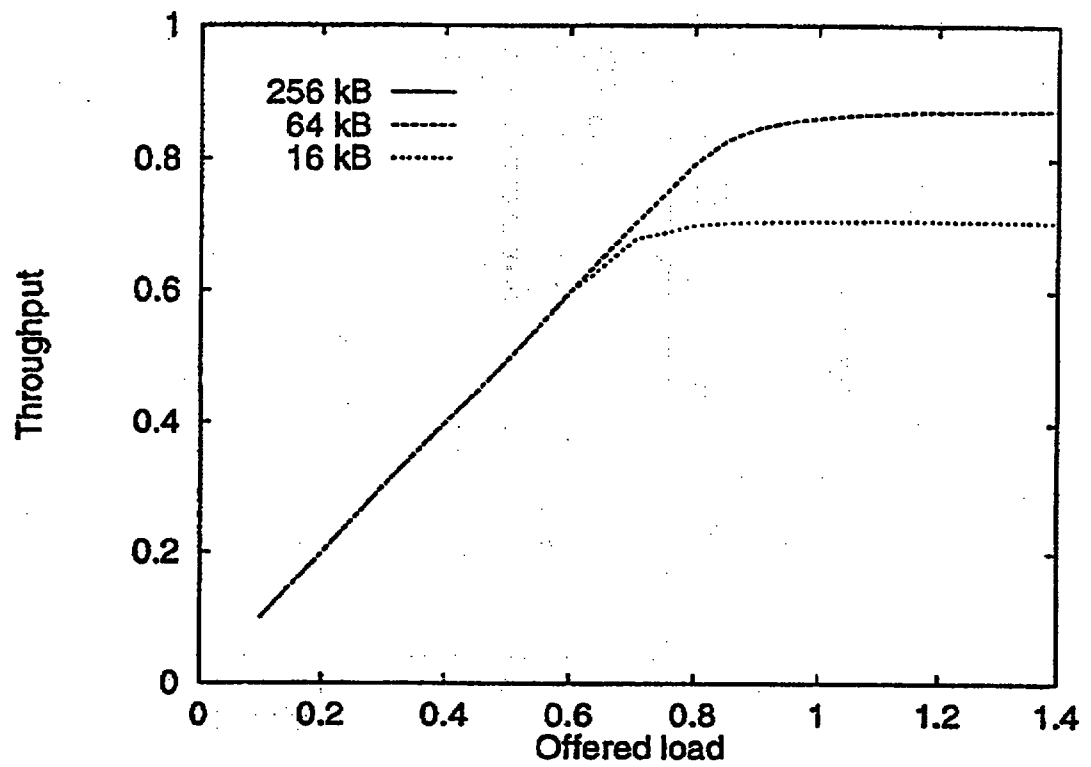


Fig 11

10/12

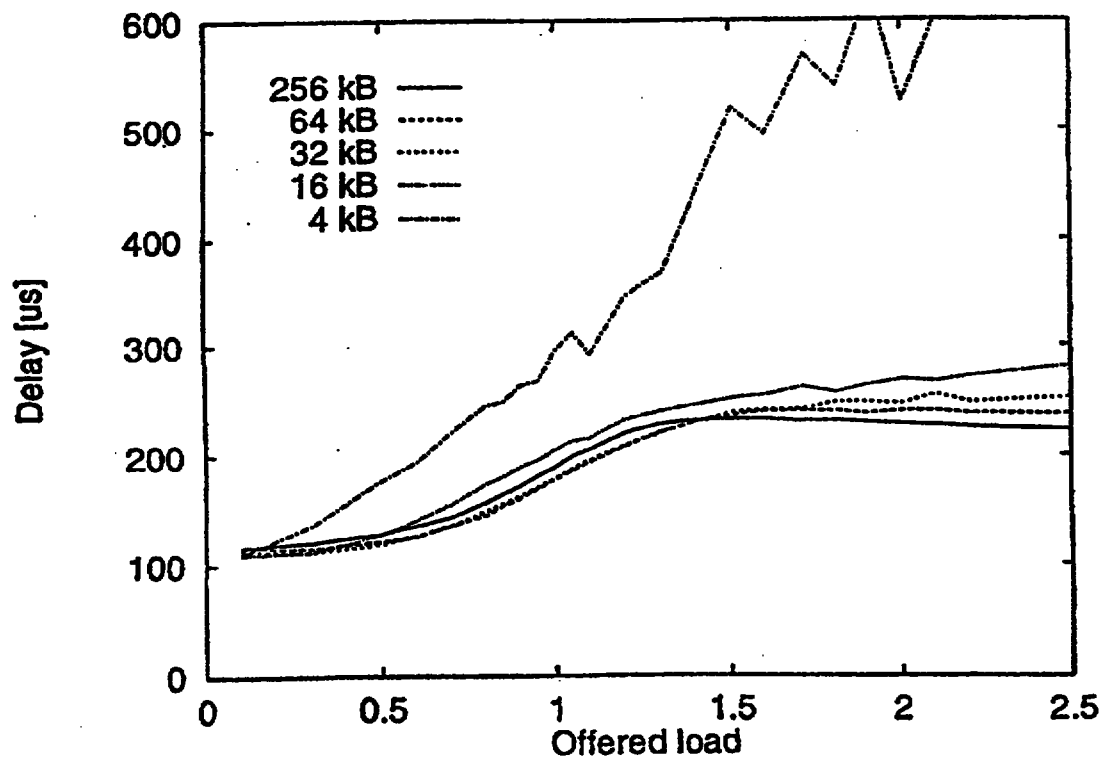
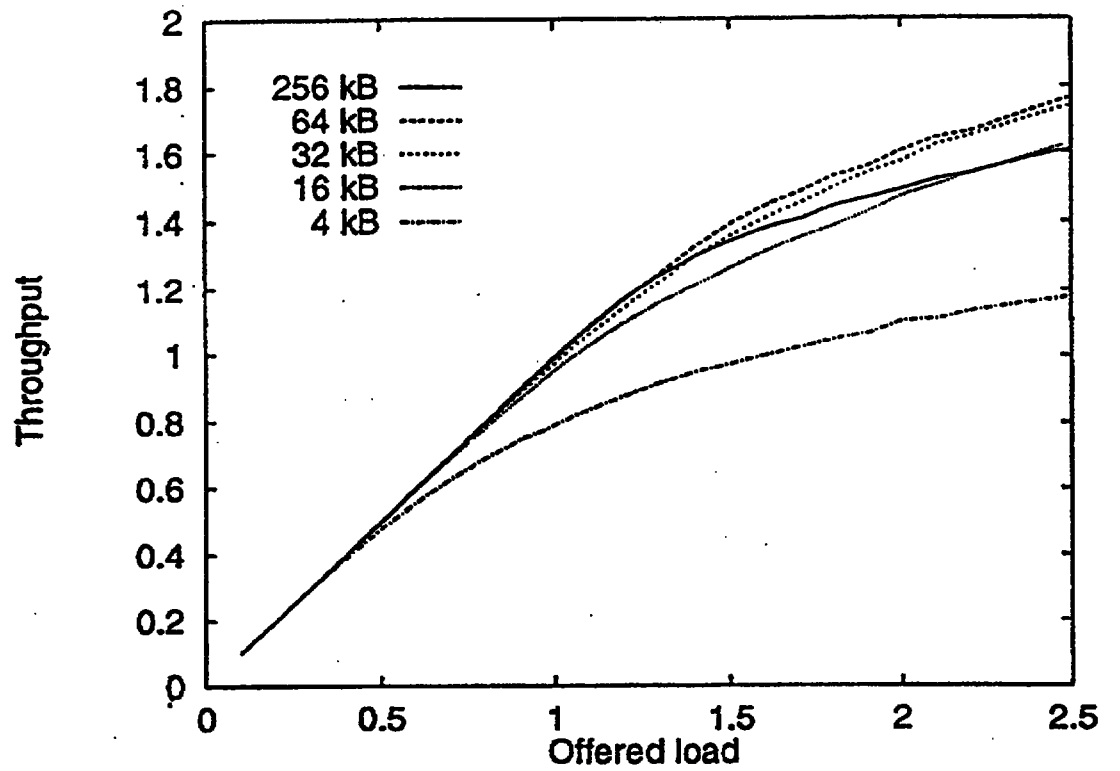


Fig 12

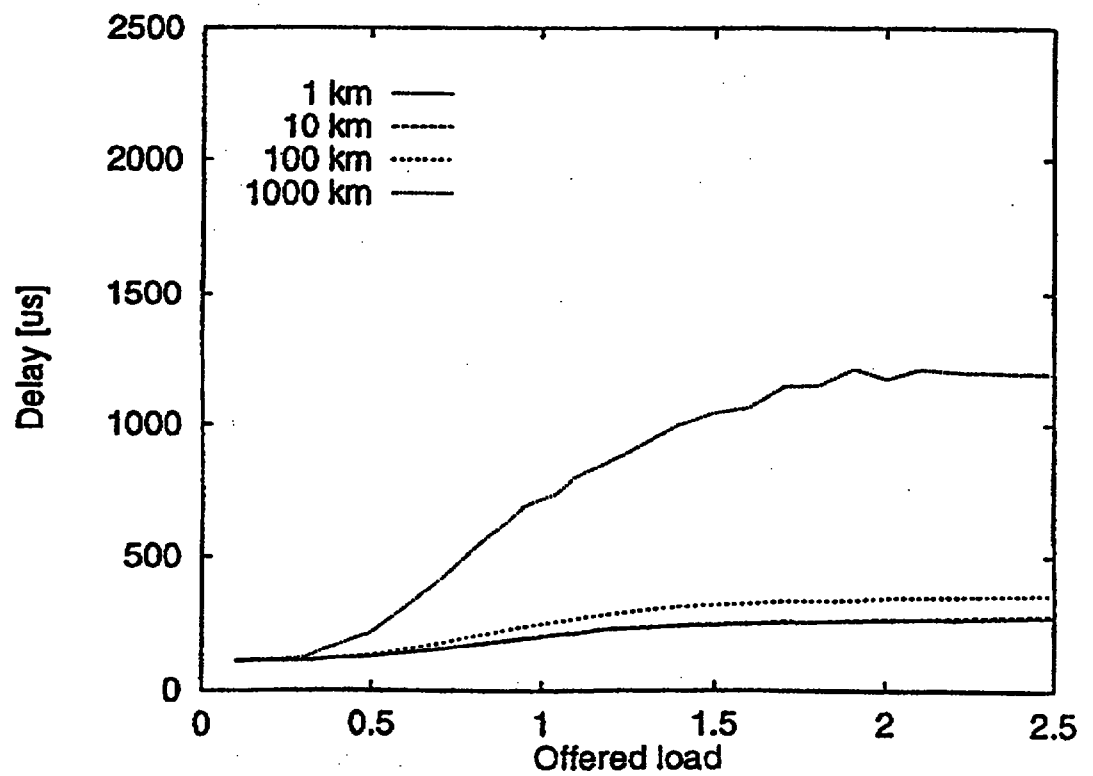
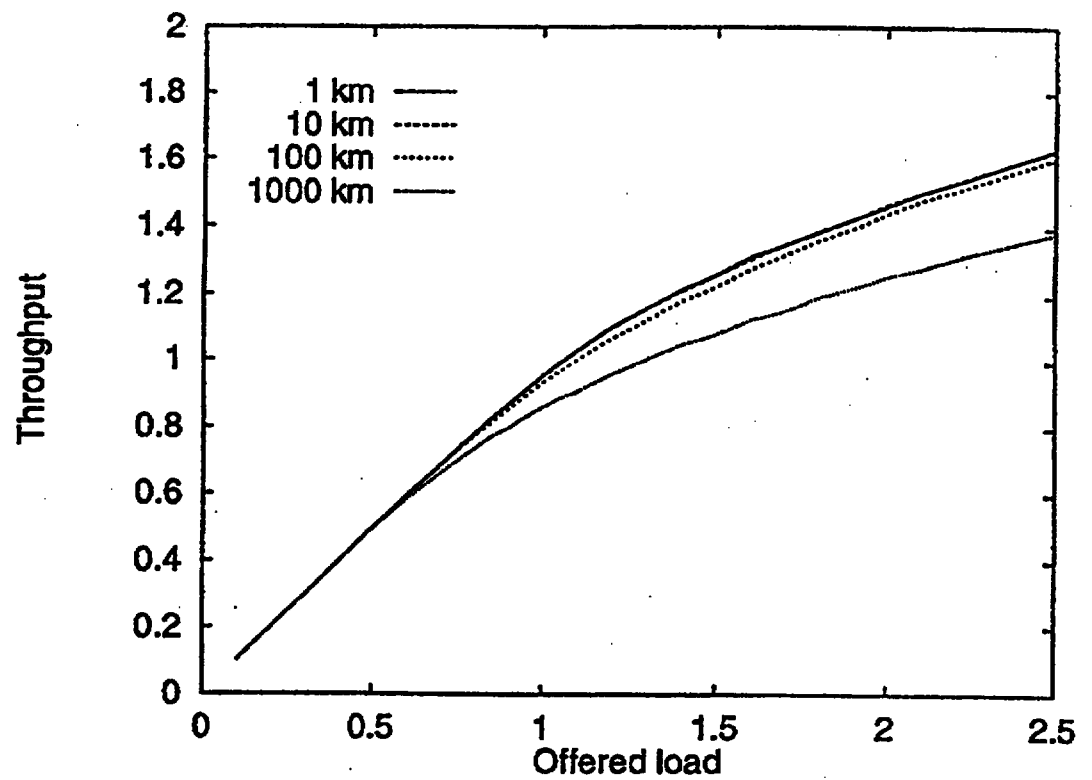


Fig 13

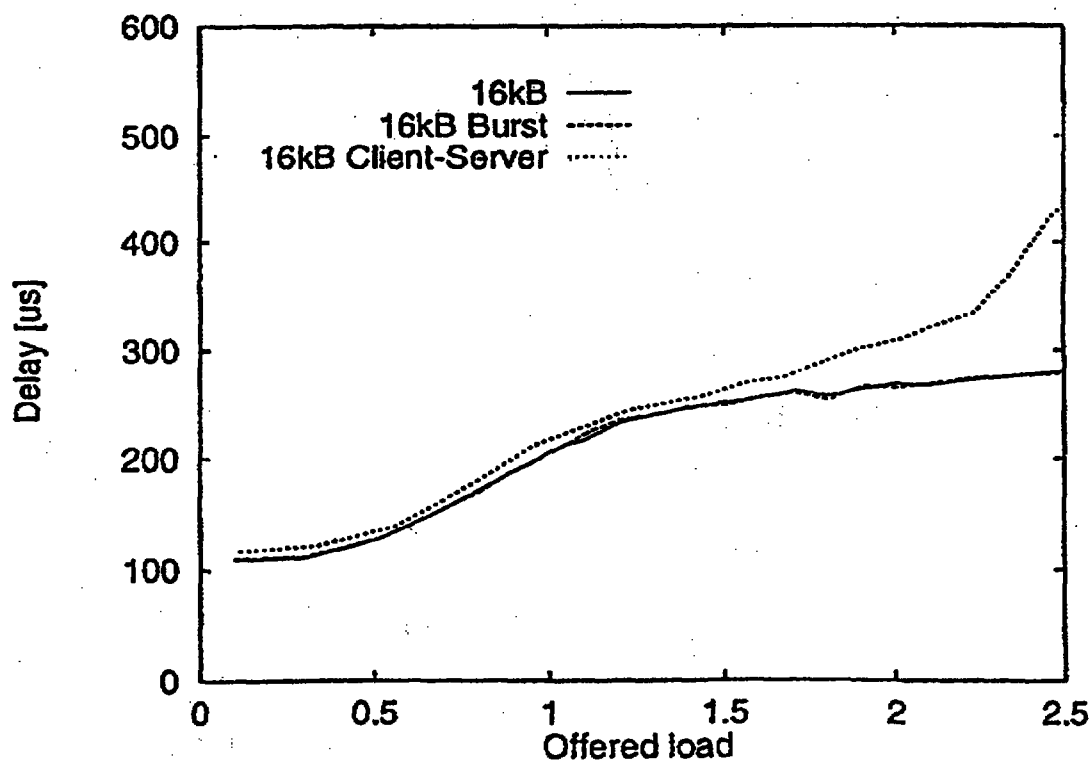
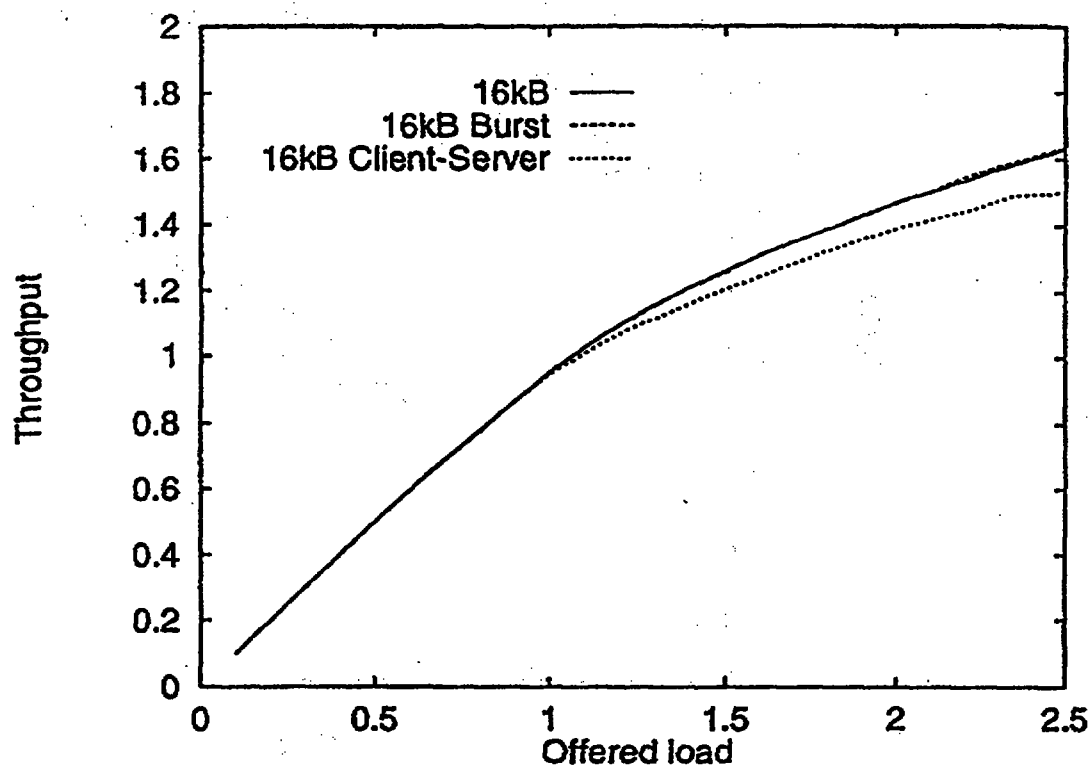


Fig 14

METHOD FOR SLOT REUSE IN A DYNAMIC SYNCHRONOUS TRANSFER MODE NETWORK

TECHNICAL FIELD OF THE INVENTION

- 5 The present invention relates to a slot reuse method for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network.

DESCRIPTION OF RELATED ART

- 10 New high-capacity communication networks and protocols are constantly being developed by the communications industry and academia. This development change frequently and new results are important to application developers who integrate real-time audio, video, and asynchronous communication services into
- 15 applications. The applications can be found on a wide range of network access terminals. Terminals act as network hosts and may be almost any electronic device, including small pocket telephones or television sets, multi-media workstations, and million-dollar supercomputers. Hosts differ by orders of
- 20 magnitude in their needs for processing power, and in their communication services requirements. These disparate needs are currently reflected in a set of independent network classes. Each network class is optimized for its particular traffic and applications: Cable Television networks use unidirectional
- 25 broadcast networks where the capacity is divided into fixed-sized subchannels carrying video. A telephone network uses only 64 kbit/s duplex circuits with guaranteed throughput and tightly controlled delay variations. Computer networks, such as the Internet, allow a large number of parallel network sessions
- 30 by use of connectionless packet switching. They also use statistical multiplexing to efficiently utilize links. A backbone network for mobile systems needs extra control (or signaling) capacity to dynamically track all active terminals.
- With this wide spectrum of applications existing today and to
- 35 be augmented in the future, it is infeasible to continually invent new, sometimes global, networks and a new terminal

AMENDED SHEET

interface for each new type of service. Instead, an integrated services network supporting existing and new services needs to be deployed. The overall goals for such a network are scalability to global size and maximum sharing of expensive network components to minimize cost. Optical transmission technology has been shown to provide the necessary link capacity at a low enough price to make integrated services networks a realistic solution.

A new integrated optical network with much higher capacity will, however, introduce new problems not seen in today's more specialized and lower performance networks. First, when network capacity increases and the information flight latency remains limited by the speed of light, the increasing bandwidth delay product will put higher demand on mechanisms that isolate a user's traffic from third-party network traffic. A telephone session, for example, should not be affected by another user opening a high-capacity video channel. Second, applications and protocols will need to operate reliably with an increasing amount of information in transit to benefit from the increased network capacity. This will lead to larger burst and transaction sizes in the network.

Current networks using stateless packet switching protocols such as the IP (Internet Protocol) have turned out to be extremely scalable. They have evolved from small networks connecting just a few DARPA (Defense Advanced Research Projects Agency) research computers in the mid seventies to the current global and ubiquitous Internet. Shared medium LAN's (Local Area Networks) such as CSMA/CD (Carrier Sense Multiple Access/Collision Detection), token ring and FDDI (Fiber Distributed Data Interface) are used in the Internet as simple building blocks connected by routers or bridges. The combination of easy expansion, low incremental node cost and tolerance to faulty nodes has resulted in simple, flexible, and robust networks. Also, the shared medium allows efficient application of new multicast protocols such as IP multicast.

A drawback of the shared medium is that it typically permits only a single terminal to transmit at any time not utilizing

all network segments efficiently. A scheme allowing the capacity of the medium to be reused may be designed, but this is often at the cost of complexity in the high-speed access control hardware. Access control mechanisms for a shared medium
5 also depend strongly on the size of the network and are usually efficient only for local area environments.

The two main types of network are connection oriented circuit-switched networks used for telephony, and connectionless packet-switched networks, exemplified by the Internet. When a
10 circuit-switched network is used for data communication, circuits need to remain open between bursts of information, wasting network capacity. This problem arises because circuit management operations are slow compared to dynamic variations in user demand. Another source of overhead in circuit-switched
15 networks is the limitation of requiring symmetrical duplex channels, which introduce 100% overhead when information flow is unidirectional. This constraint also makes multicast circuits difficult to implement and inefficient. A connectionless packet-switched network, on the other hand,
20 lacks resource reservation and must add header information to each message before transmission. Furthermore, latency in a connectionless packet-switched network cannot be accurately predicted and packets may even be lost due to buffer overflow or corrupted headers. The latter two factors make real-time
25 service difficult to support. Congestion avoidance mechanisms can isolate traffic streams of different users. These schemes are, however, limited to operating on a time scale comparable to the round-trip packet delay.

To address the problems mentioned above, the communications
30 industry is focusing on the development of ATM (Asynchronous Transfer Mode). ATM has been proposed for LAN's and many future public networks. CCITT (International Telegraph and Telephone Consultative Committee) also adopted it to be used as the transfer standard in B-ISDN (Broadband- Integrated Services
35 Digital Network). ATM networks are connection-oriented and establish a channel just as circuit-switched networks, but use small fixed-sized packets called cells for information

transfer. The packet-switched nature of ATM means that the network needs to have new mechanisms such as buffer resource managers and link schedulers to establish real-time guarantees for a connection i.e. virtual channel/path.

- 5 Another solution for providing real-time guarantees centers on a circuit switched network and must therefore address circuit-switching concerns such as described above. A new shared-medium control protocol is also employed, and so have common shared-medium problems to be considered. This design, called DTM
10 (Dynamic Synchronous Transfer Mode), (see e.g. Christer Bohm, Per Lindgren, Lars Ramfelt, and Peter Sjödin, The DTM Gigabit Network, *Journal of High Speed Networks*, 3(2):109-126, 1994 and Lars Gauffin, Lars Håkansson, and Björn Pehrson, Multi-gigabit networking based on DTM, *Computer Networks and ISDN Systems*,
15 24(2):119-139, April 1992) uses channels as the communication abstraction. This channels differ from telephony circuits in various ways. First, establishment delay is short so that resources can be allocated/deallocated dynamically as fast as user requirements change. Second, they are simplex and so
20 minimize overhead when the communication is unidirectional. Third, they offer multiple bit-rates to support large variations in user capacity requirements. Finally, they are multicast, allowing any number of destinations.

- DTM channels share many beneficial properties with circuits.
- 25 There is no transfer of control information after channel establishment, resulting in very high utilization of network resources for large data transfers. Support of real-time traffic is natural; there is no need for policing, congestion control or flow-control within the network. The control
30 information is separated from data, which makes multicast less complex. The transmission delay is negligible (i.e. less than 125 μ s) and there is no potential for data loss because of buffer overflow as in ATM. Bit-error rates depend on the underlying link technologies, and switches are simple and fast
35 due to strict reservation of resources at channel setup.

DTM can show good performance in areas where traditional circuit-switched networks fall short: dynamic bandwidth

allocation, channel establishment delay, and as shared media networks.

The article "Multi-gigabit networking based on DTM, A DTM medium access technique with dynamic bandwidth-allocation", by
 5 L. Gauffin et al., Computer Network and ISDN Systems, Vol. 24, No. 2, April 1992, pp. 119-130, discloses a distributed management protocol according to which the time slots are distributed evenly among the nodes. This implies a completely symmetrical system.

10 SUMMARY OF THE INVENTION

One of the problems using DTM is that a data slot never can be used by two nodes simultaneously in the network.

The object of the invention is to solve this problem. This is accomplished with a slot reuse method and arrangement for
 15 better utilization of shared links and consequently an increase of the throughput in the DTM network.

The slot reuse method for increase of the throughput in a DTM network comprises the following steps:

- that the DTM block token format is extended to include
 20 parameters describing the segments(s) between source and destination node,
- that the block token capacity is only reserved on the segment(s) between source and destination node,
- that two or more segment-consecutive tokens representing the
 25 same slot range are merged into a single token when existing in the free pool of a node, and
- that simultaneous transmissions in the same slot over disjointed segments of the network are enabled.

The slot reuse can also comprise merging two or more segment-
 30 consecutive tokens, representing the same slot range, into a single token when existing in the free pool of a node.

The slot reuse method and arrangement according to the invention avoids the problem mentioned above. The method let

channels only reserve capacity on the segments between source and destination node. A single slot may then be used multiple times simultaneously in the network.

A very important advantage with the invention is that no hardware changes to the original prototype implementation is needed.

Another advantage with the invention is that it can be realized in both a central and a distributed token management scheme and in a combination of these schemes.

Yet another advantage is that the performance gain is clear: on a dual-bus where source and destination node pairs are uniformly distributed it has been shown that throughput may be increased by a factor of two.

Yet another advantage is that the performance gain can be even higher in other network designs; e.g. in a double ring structure with source and destination nodes uniformly distributed throughput may be increased by a factor of four. Having nonuniformly distributed source and destination nodes the gain may be much higher.

Yet another advantage is an improved performance when combined with a defragmentation method returning tokens to their home nodes as a way to increase the probability that two consecutive tokens can be merged in the free pool of the nodes, which decreases fragmentation.

25 BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described in greater detail below with reference to enclosed drawing, in which

fig. 1 illustrates a dual-bus DTM network,

fig. 2 illustrates a DTM 125 μ s cycle with a home node defined for each data slot according to the invention,

fig. 3 illustrates a token map showing slot number and segment,

fig. 4 illustrates a slot-segment map showing reuse of slots according to the invention,

- fig. 5 illustrates the performance (throughput and access delay, respectively, versus offered load) for small user requests (16 kbyte) and for different minimum acceptable capacities as simulated using a distributed token server according to the invention,
- fig. 6 illustrates the performance for small user requests (16 kbyte) and for different number of allowed retries before request blocking as simulated using a distributed token server according to the invention,
- fig. 7 illustrates access delay as a function of simulated time using A) the defragmentation scheme according to the invention and no fragmentation at simulation start, B) the defragmentation scheme according to the invention and maximum fragmentation at simulation start, and C) no defragmentation scheme and maximum fragmentation at simulation start,
- fig. 8 illustrates theoretical dual-bus and DTM throughput versus offered load for a dual-bus DTM network with and without slot reuse,
- fig. 9 illustrates the performance for different packet sizes as simulated using a central token manager in a 10km long dual-bus DTM network without slot reuse,
- fig. 10 illustrates the performance for different packet sizes as simulated using a central token manager in a 10km long dual-bus DTM network with slot reuse according to the invention,
- fig. 11 illustrates the performance for different packet sizes as simulated using a central token manager in a 1000km long dual-bus DTM network with slot reuse according to the invention,
- fig. 12 illustrates the performance for different packet sizes as simulated using a distributed token manager in a dual-bus DTM network with defragmentation and slot reuse according to the invention,

fig. 13 illustrates the performance for different bus lengths as simulated using a distributed token manager with defragmentation and slot reuse according to the invention, and

- 5 fig. 14 illustrates the performance for different traffic conditions as simulated using a distributed token manager in a dual-bus DTM network with defragmentation and slot reuse according to the invention.

DETAILED DESCRIPTION OF EMBODIMENTS

10 First, the DTM MAC (Medium Access Control) protocol will be described. The basic topology of a DTM network is a bus with two unidirectional optical fibers connecting all nodes as is showed in fig. 1. Several busses with different speeds may be connected together to form an arbitrary multistage network. In
15 the current prototype implementation, buses can be combined into a two-dimensional mesh. A node at the junction of two buses can synchronously switch data slots between the two busses. This allows for efficient switching with constant delay through the node. The primary communication abstraction in DTM
20 is a multi-rate, multicast channel.

The DTM medium access protocol is a time-division multiplexing scheme. The bandwidth of the bus is divided into 125 μ s cycles, which in turn are divided into 64-bit time slots (or slots for short) as illustrated in fig. 2. The number of slots in a cycle
25 thus depends on the network's bit-rate; for instance, on a 6.4 Gbit/s network there are approximately 12500 slots per cycle.

The slots are divided into two groups, control slots and data slots. Control slots are used to carry messages for the network's internal operation, such as messages for channel
30 establishment and bandwidth reallocation. The data slots are used to transfer user data and are not interpreted by intermediate network nodes. Intermediate nodes are nodes between the source and destination.

In each network node there is a node controller NC, which
35 controls the access to data slots and performs network

management operations, such as network start-up and error recovery. The main tasks of the *node controller* are to create and terminate channels on demand from users, and to manage network resources in response to user requests and in the background.

Control slots are used exclusively for messages between node controllers. Each node controller has write permission to at least one control slot in each cycle that it uses to broadcast control messages down-streams to other nodes. Since write access to control slots is exclusive, the node controller always has access to its control slots, regardless of other nodes and network load. The number of control slots a node use may vary during network operation.

The network is not restricted to a dual-bus, but can be realized by other kind of structures, e.g. a ring structure with an arbitrary number of nodes. The transmission media can besides to optical fibers be coaxial cables or any other high bandwidth transmission media. Hereafter the transmission media will be referred to as optical fibers. The bandwidth of the DTM dual bus in the preferred embodiment is divided into 125 ms cycles, which in turn are divided into 64-bit time slots. The invention is not restricted to DTM networks with this values, but can be used in networks with cycles and slots of arbitrary sizes.

Principles for resource management (called token management) will be described below. The majority of the slots in a cycle are data slots. Access to data slots changes over time, according to traffic demands. Write access to slots is controlled by *slot tokens* (or *tokens* for short). A node controller may write data into a slot only if it owns the corresponding token. The token protocol guarantees the slot access to be *conflict free*, which means that several nodes do not write data into the same slot.

Control messages for channel establishment and bandwidth reallocation carry set of slots or tokens as parameters. However, a control message is 64 bits and can therefore have

only a small number of parameters. This means that if a user requests a large bandwidth transfer, it may be necessary to send several control messages to create the channel. This introduces extra access delay and consumes signaling capacity.

- 5 We are considering several mechanisms to decrease the amount of information that needs to be sent during channel creation and token reallocation. The first optimization in token management is to introduce block tokens. A block token is transferred in a single control message and represents a group of tokens, but
 10 can only be used for particular combinations of tokens. For instance, in the simulator a block token is denoted by a slot number and an offset giving the number of contiguous slots in the group. The block token optimization assume that the token pool is not fragmented into small pieces. This will be referred
 15 to as *fragmentation* of the token pool.

- The token protocol guarantees that a data slot can never be used by two nodes simultaneously on the bus. Sometimes this protocol is too conservative. Fig. 3 shows an example of how three tokens (A, B, and C) are reserved for three channels.
 20 Nodes are connected by bus segments and channels typically use a subset of the segments on the bus (gray color) and the rest are reserved (white color) but left unused and thus wasting shared resources. A better alternative is to let channels only reserve capacity on the segments between the sender and the
 25 receiver as the example in fig. 4. A single slot may in this case be used multiple times on the bus. Channel D and E are using the same slots as channel A and C but on different segments. This is referred to as *slot reuse*. Slot reuse enables simultaneous transmissions in the same slot over disjointed
 30 segments of the bus.

- Slot reuse is a general method to better utilize shared links in ring and bus networks. The slot reuse algorithms in DQDB (Distributed Queue Dual Bus), Simple and CRMA (Cyclic Reservation Multiple Access) depend on control information in
 35 the slots. Buffer insertion networks when combined with destination release as in METARING may reuse capacity of

individual links and resolve eventual contention by delaying the incoming packet stream through an elastic buffer.

With slot reuse, the access scheme complexity is increased, regardless of whether it is done in hardware as in DQDB, Simple, and CRMA; or in software, as in DTM. When implemented in systems other than DTM, slot reuse also add complex hardware to the critical high-speed path through a node and therefore increases the node delay.

To allow slot reuse in DTM the block token format is according to the invention extended to include parameters describing the segment(s) it is representing. The token management protocol is also according to the invention modified to avoid conflicts in the slot number dimension as well as the segment dimension. The most important assumption is that no hardware changes to the original prototype implementation where allowed or needed. The performance gain is also quite clear: on a dual-bus where source and destination pairs are uniformly distributed it has been shown that throughput may be increased by a factor of two. The performance gain can be even higher in other network designs; e.g. in a double ring structure with source and destination nodes uniformly distributed throughput may be increased by a factor of four. Having nonuniformly distributed source and destination nodes the gain may be much higher.

The potential drawback of slot reuse in a DTM network however, is the higher algorithm complexity and eventually higher load on the node controller and signaling channels (especially if the average channel duration is short).

Two token management schemes have been evaluated. The first and simplest is to let a single node controller manage all free tokens for one fiber. This sort of centralized server mechanism has also been used in systems such as CRMA, where the head-end node distributes the fiber capacity to all other nodes. The simulator was configured so that for each fiber a node one third of the distance from the slot generator was token server (with 100 nodes on a bus node 33 and 67 will be token servers). This corresponds to the mid-point of demand for each of the two

unidirectional fibers such that a token manager will have the same amount of traffic on both sides.

Each time a user request arrives at a node, the node first requests tokens from the manager and then locks them throughout the channel life time. When the user issues a request to disconnect the channel the tokens are returned to the manager immediately. All requests are delayed during the token request and accesses are serialized through the central manager.

The *distributed token manager* is fundamentally more complicated than the centralized. We tried to keep it as simple as possible. In our method each node regularly broadcasts status information about how many free tokens it has. The other nodes store this information in their *status tables*. A node that wants more capacity consults its status table to decide from which node to request slots. The protocol on the initiating side works as follows. When a user request arrives to a node:

1. If the node has sufficiently many free tokens to satisfy the request, it allocates the requested amount of slots to the user, and starts the channel by sending a channel establishment message to the destination node, and then transmitting data using the reserved slots.
2. Otherwise the node marks its available tokens as being reserved, and then checks its status table: if the total amount of free tokens in the network is not enough to fulfill the request, then the request is rejected (blocked). Otherwise the node requests tokens from nodes with unused capacity.

If one of these nodes receiving a token request does not have the requested amount of free slots, it still gives away all of the slots it has. In any case, it sends a response back to the requesting node. A node fulfills incoming requests in strict FIFO order.

When a node receives a response to a token request, it marks the slots it receives in the response (if any) as being reserved. When the node has received responses to all requests it has sent, it either starts the channels or

rejects the user request, depending on whether or not it has acquired sufficient capacity. If the user request is rejected, the reserved slots are marked as free again.

At startup all free tokens are evenly distributed among the
 5 network nodes and each node takes at least one of its free tokens, moves it (them) to the active state and declares it (them) to be a control slot. User requests may now be accepted and tokens can be moved between nodes on demand.

A weakness with this scheme is that slot reallocation is only
 10 triggered by user requests, and user requests are delayed by pending token reallocation. An optimization we have implemented to remedy this is to also perform slot reallocation in the background. This results in less need to reallocate tokens for small to medium sized requests.

15 The pool of free tokens may be distributed in other ways than evenly to increase reallocation success rate and utilization. If fewer nodes are managing the pool, channel blocking decreases as a result of a lower probability of token reallocation failure.

20 The complete token pool is in this case distributed proportionally (nodes close to the slot generator get more tokens than nodes far away from it) among all nodes. Token transfers may occur between any pair of nodes instead of always involving the server node. When the local node contains enough
 25 tokens to fulfill an arriving user request, the request may be accepted without any token reallocation. Furthermore, as long as the arriving user requests are well matched to the pool distribution no reallocation will ever be necessary.

Several questions need to be answered before deciding how to
 30 distribute the token pool. The following issues are addressed:

1. When local resources in the node are not enough to satisfy a user request, which other node should be asked for more tokens?

2.If a node asks for tokens from several nodes, how many tokens is it going to ask for and should the node reject the channel if it receives a fraction of the requested capacity?

3.If tokens move freely among the nodes, will the token pool be fragmented into small pieces and render the block token optimization scheme useless?

Status messages were decided to be used to distribute information about the pool of free tokens. Status message information is used to help a node choose a suitable node when asking for more resources. This method addresses the first question above.

Our scheme works as follows. Each node regularly broadcasts status information about how many free tokens it has. The other nodes store this information in their *status tables*. A node that wants more capacity consults its status table to decide from which node to request slots. The broadcasted state information gives an approximate and dated view of the current state of token information, so token requests may be rejected because they were sent to nodes that no longer have tokens to give away.

Status tables are "soft" information in the sense that the system will work even if they are out of date or not available. They should, however, improve the success rate of reallocation procedures.

When comparing basic performance of the centralized (fig. 9) and the distributed (fig. 12) token manager we see that there is a new type of rejections that is frequent in the distributed version when resources are still unused in the system.

A node uses the status table to pick node(s) from which to request tokens. When the request reaches the target node the amount of available capacity may have changed and a smaller amount than requested may be returned to the requesting node, resulting in a user rejection.

This behavior results in even more unnecessary token movement and therefore also increases the probability of other nodes

receiving fewer than requested tokens. The tokens that are moved in this case are also locked and unused during the transfer.

If the pool is distributed proportionately among a large number
5 (100's) of nodes, the average pool size will be relatively small. When the load is high, the number of free tokens in pools decreases even further. If nodes also create and destroy channels at a very high rate, the amount of free capacity in individual nodes jumps between having a small amount of
10 capacity and having none. Now, if the average capacity requested by a user is large compared to the number of free tokens of a node, several nodes need to be asked to fulfill the request. At this point the probability that one of the requested nodes do not have any free capacity increases,
15 leading to a user rejection.

There are several ways to address this problem without going back towards a centralized model. First, we may not give away any slots at all when the complete request cannot be fulfilled. This protocol applies if only a single node is asked for free
20 tokens but if several nodes are asked it may still result in tokens being moved or locked unused. Second, if after a token request, we receive fewer tokens than we requested we may simply retry the token request procedure several times. This will increase the probability of accepting a user request and
25 that received tokens will be used. The cost of retry will be increased signaling and increased access delay, and it may worsen performance in an overloaded network. Also user retry introduces longer set-up delay for requests that are retried. Third, the user may sometimes want to accept a channel with
30 less than requested capacity instead of being rejected.

If, for example, a user receives 50% of the requested he may want to accept. In fig. 5 throughput, which is the ratio between transported user data and total number of bits transmitted by the fibers, and access delay, which is the time
35 from the arrival of a user request to the sending of the first data of the request, for small (16 kbyte) user requests with various minimum acceptable capacities [100% (40 slots), 50% (20

slots), and 5% (1 slot)] are plotted versus offered load. A lower average *minimum acceptable bandwidth* will result in higher throughput. In fig. 6 is shown the performance that results if user (requesting a 16 kbyte data transfer) retries
 5 up to 8 times before the request finally is blocked. Utilization increases (and blocking decreases) at the cost of more signaling and longer delay. Retrying a large number of times is counter productive if it happens frequently.

Clearly the benefits of a flexible user request policy, is
 10 lower probability of being rejected and higher overall throughput. Either of the configurations presented in fig. 5 and fig. 6 may be decided at arrival time of a request. A user that has strict demands on the capacity for a channel may retry until enough capacity is allocated, but another may rather
 15 accept a channel with less than the requested amount. For the rest of the simulations presented here the minimum acceptable bandwidth is defined to be 50% of the requested capacity.

In the general case, the average number of contiguous free blocks in a node is small, due to the random movement of tokens
 20 and the varying capacity of users request. This fragmentation renders the block token optimization practically useless, and the access delay is relatively long (milliseconds) for high capacity channels. To make block allocation efficient it is necessary to reduce fragmentation of free tokens, otherwise
 25 fragmentation will be by far the main contributor to access delay for high bandwidth channels at moderate to high load. Low capacity channels will almost always have a very short channel establishment delay independent of the current amount of fragmentation. In the case of slot reuse, the fragmentation
 30 problem is even worse, as fragmentation may occur in both slot (time) and segment (space) dimensions (see fig. 4). This is in the centralized server version a particular application of the general *dynamic storage-allocation* problem. In the distributed token manager most of the fragmentation is a result of using
 35 many free pools (one for each node). Two free adjacent tokens can only merge if they are found in the same node.

A distributed scheme that tries to avoid fragmentation if possible and increases the average block size of free tokens in the nodes has been implemented. The scheme is used both with and without slot reuse.

5 The scheme works as follows:

1. Define a home node for each token at network startup and distribute tokens in such a way so tokens sharing the same home node will always define a continuous slot range. This results in a large average token area in the token map shown
10 in fig. 4.
2. When two slot-consecutive tokens with the same slot or segment range exist in the free pool merge them into a single token (sometimes a recursive split and merge operation is needed). When doing a merge always prioritize segment merge
15 before a slot number merge. (The reason for this is that tokens spanning over just a few segments are less useful to other nodes than tokens spanning over many segments). Two segment-consecutive tokens, representing at least partly the same slot range and existing in the free pool of a node, are
20 splitted to obtain segment-consecutive tokens representing the same slot range, which are merged into a single token.
3. When a node get a token request from the local user or a remote user, pick a token from the token pool using a best-fit algorithm in slot number and segment number dimension
25 (see fig. 4). The value of a token is calculated as the area of a token in the token map and we tries to pick the token with the smallest area that fulfill the requested capacity. A cost function can also be defined as a function of e.g. number of slots, number of segments, location of slots and
30 location of segments, which function should be minimized, however, still fulfilling the requested capacity. This mechanism can also be used by the servers when using a centralized token manager.
4. When a node needs to request tokens from other nodes it does
35 not ask for small chunks from several nodes if it is possible to ask for larger chunks from fewer nodes. The status tables

provide this information. Transfer of tokens is therefore more efficient, and there are fewer establishment messages and less fragmentation.

- 5 Free tokens are sent back to home nodes when they have been idle for a significant time or after a long transfer.

This scheme returns tokens to "home" nodes as a way to increase the probability that two consecutive tokens can be merged in the free list which decreases fragmentation. If home node "gravity" is too strong the scheme will result in less sharing
10 of resources and unnecessary signaling. If it is too weak, fragmentation will still remain a problem. The "gravity" may be changed during the operation of the bus.

To evaluate the defragmentation mechanism we did another set of simulations. Three different simulators [A, B, C] were
15 configured. Simulator A was configured to have no fragmentation at simulation start time and to use the defragmentation scheme described above. B did start with maximum fragmentation of the complete resource pool. All tokens had a single slot and no tokens in the "home" nodes before the defragmentation mechanism
20 was turned on. Finally, simulator C was started without using the defragmentation mechanism and with the pool having maximum fragmentation. In all cases slot reuse were turned on and load was defined to be 80%.

In fig. 7 is shown the access delay as a function of simulated
25 time for a 10 km network. Simulator C started with a long access delay and the delay increased as the signaling channels became overloaded and message queues grew. Simulator B using the defragmentation mechanism starts just as bad as C but already after 10 milliseconds the average access delay is below
30 500 microseconds. Later when one second of simulated time has passed the B curve almost joins A i.e. it converges on the performance of the simulator starting without any fragmentation at all. The convergence speed depend on the amount of free capacity in the network and therefore also on load. The load
35 during all these simulations was 80%. The defragmentation mechanism clearly improves the access delay and also makes the

block token optimization meaningful in the distributed implementation.

Two kinds of performance measures are of main interest: utilization and access delay. Utilization is the portion of the nominal network capacity that is actually used for data transfer, and is a measure of the efficiency of the network. Access delay is the time from the arrival of a user request to the sending of the first data of the request, which is an important measure for how well computer communication traffic can be supported.

There are two main factors that influence utilization in DTM. First, each node is assigned signaling capacity in the form of control slots, which means that there are fewer slots available for data transfer on a bus with many nodes, given fixed link capacity. Secondly, token reallocation incurs overhead since while a slot token is being reallocated between nodes, the corresponding slot cannot be used for data transfer.

Access delay depends mainly on the load on the control slots, and on how many control messages need to be sent to establish a channel. The access delay is typically a summation of a few delays: e.g. node controller processing delay [5 μ s], delay in finding and allocating free tokens [100 μ s], waiting for the first available control slot to pass [50 μ s], and finally waiting for the first allocated data slot to be filled with user data [62.5 μ s]. In addition, messages are delayed in queues at the input to node controllers waiting to be processed. In the simulations presented below the average delay is up to a few hundred microseconds.

Results from simulations where DTM is exposed to traffic patterns that are more like the relatively short-lived transfers (4-4000 kbyte) seen in computer communication will be presented. Traffics are with bursty inter-arrivals, client-server oriented as well as with exponentially distributed arrivals. In the simulation model, each transfer starts with the arrival of a new "packet" of information. The node controller tries to allocate resources for the transfer,

transmits the data and finally releases the channel. This is a simplification of the mechanisms in a real system, where channel set-up, data transfer and channel teardown are independent operations initiated by the user. For example, a user that knows that a transfer is about to take place may "hide" the channel establishment delay by requesting a channel in advance, so that it is already established when the transfer starts. Between a setup and teardown, the capacity of the channel is completely reserved for the user. The most straightforward use of the channel is for a single transfer, such as a file transfer or a video transmission.

Depending of the characteristics of the application, it is possible to optimize the usage of the channel. For instance, a channel may be used to transfer sequences of higher level messages such as ATM cells or IP packets (this is similar to using a single VC for all IP traffic in an ATM network). If the channel is a multicast channel, messages to different destinations can be multiplexed onto the channel. This means that each message will reach every receiver on the multicast channel and receivers must be able to filter messages. An alternative solution is to create and destroy a channel for each message, but reserve the tokens between messages so that the tokens are readily available for the next message in the sequence.

This type of user behavior is not incorporated in the simulations, since they are optimizations for particular applications. Instead attention is paid to how the network performs without user level optimizations.

The sender may start sending data as soon as the resources are allocated, even before the receiver receives the channel establishment message. This is called *channel establishment*. The receiver will eventually respond with a control message to accept or reject the channel.

User requests have the following parameters:

- *Packet size* which is the amount of user data transferred between channel establishment and channel release. We simulate packet sizes from a few kbytes up to a few Mbytes.
- 5 - *Requested capacity* for a channel, which is the number of slots that a node tries to allocate. For all the simulations in this paper the requested capacity is fixed to 40 slots or 20.48 Mbit/s.
- *Minimum acceptable capacity*. A node blocks a request if it cannot allocate this number of slots. This is normally set to
10 40 or 20 slots (100% or 50% of requested capacity).
- *Source address*.
- *Destination address*.

The source and destination addresses are generated randomly (all nodes with the same probability) and user inter-arrival
15 times are exponentially distributed. The simulations investigate the effect of signaling capacity and slot reallocation overhead on utilization, channel set-up delay, and blocking. We simulate varying traffic conditions, for a topology with the following characteristics:

- 20 - A dual bus network with 100 nodes. Even though it is theoretically possible to connect many more nodes to a bus, we think that management of networks with more than 100 nodes on a bus may be infeasible. With 100 nodes the capacity sharing is significant enough to exercise and test the token
25 management protocol.
- The capacity of each bus is 6.4 Gbit/s. We believe this is realistic for what is realizable within a year or two; 2.4 Gbit/s optical links have been available for a few years, and 10 Gbit/s links have been announced to appear soon on the
30 market. 6.4 Gbit/s corresponds to 100 MHz slot rate, which is the rate at which the slot processing MAC hardware would operate this speed attainable with current CMOS technology.
- The total signaling capacity is the same for all nodes but the slots are partitioned between the two fiber directions
35 proportionally depending on where the nodes are located on

the bus. The closer a node is to the slot generator the more control capacity is needed. The sum of the control capacity on both buses will however be the same for all nodes. In the network with two token servers, the servers has more control capacity and higher processing capacity than the other nodes.

- The length of the bus is 10 km, which gives a large enough network that the effects of propagation delay are not negligible. Results from studies of the effects of propagation delay in simulations with different bus lengths are presented in fig's 11 and 13.

- Two different token management schemes are simulated: an asymmetrical scheme where all the tokens on one fiber are managed by a single token server and a symmetrical scheme where each node controller manages a small piece of the global token pool.

When analyzing the performance of the DTM dual-bus network the issue of maximum theoretical performance must be addressed and compared to the simulated performance. The maximum theoretical performance is also used in this paper to compare the different schemes and implementations we are evaluating.

The maximum throughput of a dual-bus system without slot-reuse can be defined as twice the link capacity, given that both fibers receive the same traffic. In a system with slot reuse the system throughput also depends on the source destination distribution. To obtain this throughput for the dual-bus we used a Monte Carlo simulation where source and destination addresses were uniformly distributed (see left graph of fig. 8) Comparing this with a DTM network (simulator configuration as defined below using a centralized token manager) where the user requests large transfers (4 MB at 20 Mbit/s capacity) and signaling capacity is not a bottleneck (see right graph of fig. 8), utilization is close to the ideal case. Real traffic behaving like this is bulk-data transfers and audio/video streams. The differences that are shown are a result of: Some capacity is in DTM used for control slots lowering the available number of slots to be used by data transfers. The

random generators for the DTM system do not generate exactly the same amount of traffic in upstream and downstream direction. This may result in blocking in one of the directions when capacity is available in the other. During channel
 5 establishment resources may be locked unused momentarily wasting some capacity.

In the case of central token manager the two managing nodes need more signaling capacity than other nodes (Eight times as many control slots to a server node than to other nodes are
 10 assigned).

Results of the first set of simulations are presented in fig. 9. User request 20 Mbit/s channels, inter-arrival times are exponentially distributed (generated by a Poisson process) and the simulations are performed with different packet sizes. If
 15 the full capacity of the channel cannot be allocated, the request is rejected and tokens are returned to the token server. Packet sizes vary from 4 Mbyte to 4 kbyte, at which point we begin to see throughput degradation.

Throughput degradation may occur if processing capacity in
 20 nodes or control channel capacity is too small. The server nodes may be especially overloaded. The result is that queues containing control messages start growing very big. The control tokens represent unused capacity, and therefore the throughput is degraded.

25 In the 4 kbyte per channel simulations, the control capacity is the limiting factor and if more control slots (signaling capacity) are added, 4 kbyte and even smaller packets can be more efficiently supported.

The next set of curves in fig. 10 show how the slot reuse
 30 mechanism improves the performance of the system. Throughput increases by almost a factor of two before any significant number of channels are rejected. The uniform source and destination distributions of channels limits the amount of increased capacity gained by slot reuse. It has been shown that
 35 if the source and destinations are generated evenly, as we do, throughput may be doubled on a dual bus. In the simulations it

may also be seen that beyond an offered load of 2.5 we may actually get throughput higher than 2.0. This level of throughput, however, can not be reached without some channels being rejected. The channels with the highest probability of
 5 being rejected are the ones that use many slots or segments. Therefore the system "filters" less greedy user requests and throws away the rest. This is normally not an acceptable behavior and we therefore do not investigate this any further.

Throughput degradation occur at an offered load of 1 with the 4
 10 kbyte transfers. Even if enough resources were available in the token server it is not possible to establish and destroy channels fast enough as the control channel is congested. Further is also seen a throughput degradation of the 8 kbyte simulations at an offered load of 1.8, for the same reason.

15 It may be concluded from the simulations in fig. 10 that the slot reuse mechanism almost doubles the system throughput with only minor changes to the centralized token protocol as long as the control and server processing capacity is not a bottleneck. From the curves it can also be unexpectedly seen that access
 20 delay actually decrease when the load increase from 0.1 to 0.5. This is a result of how slots are assigned to a channel and not from a faster token request procedure. The time for it takes to request tokens from the server is strictly increasing.

When comparing the DTM performance in fig. 10 and the
 25 theoretical values in fig. 8 it is seen that even short bursts (a few milliseconds duration) can be supported efficiently.

When a single token server is used, each channel establishment first requires a token to be requested from the server before the node can establish the channel. If the length of the bus is
 30 increased the token request will take a longer time and may therefore also limit the throughput and increase the access delay.

In fig. 11 is shown the result from simulations, in which the length of the bus is increased from 100 to 1000 km (node to
 35 node distance is now 50 μ s). Both access delay and throughput are now limited by the round-trip latency to the token server.

Access delay in this case depends on the distance to servers, but is independent of the transfer size. Utilization depends strongly on the transfer size as the establishment phase may be amortized over a long data transfer phase. Channels
5 transferring large amounts of information such as 256 kbyte with a duration of one-tenth of a second are still efficiently supported when the bus length is 1000 km.

Using a centralized token manager has several benefits. Clients may be simple as they only contain state information related to
10 their own opened channels. Slot reuse is also simple and efficient, as the slot server has all of the free tokens to choose from when trying to satisfy a user request. The server may also implement other policy related functions such as admission control and fairness. The fragmentation of the free
15 token pool in the server is normally very modest, resulting in very few connection establishment messages per channel even for high capacity user requests.

There are also drawbacks. A user that frequently establishes and destroys channels may introduce excessive signaling by
20 always returning tokens after use, but then requesting the tokens again within a short time period. The processing capacity of the server node may get overloaded if there are many nodes on the bus or if the average packet size is very small. If the media length is very large relative to the
25 product of bit-period, bits-per-packet and media velocity the round trip time to the server may also limit performance. Finally the server node contains state information that all nodes depend on to create channels. A server node failure may therefore affect all the nodes.

30 In next section the properties of a fully distributed token manager are simulated and investigated.

When evaluating the performance of the distributed token manager with slot reuse and defragmentation the same traffic and parameters are used as in the case of a central token
35 manager, but with a policy whereby requests are accepted if 50% of the requested capacity can be allocated.

The results presented in fig. 12 are from the simulator with slot reuse, a fully distributed token manager, status messages describing how much capacity a node owns, and the defragmentation scheme. All nodes have the same processing capacity and the processing load is much lower than what the servers in fig. 10 receive. The dependencies between nodes are also much smaller resulting in a higher degree of reliability. The system performs better than any system not providing slot reuse but not as good as the centralized system described earlier.

When comparing the performance of this configuration and the centralized token manager in fig. 10, one sees that blocking is higher and some requests may be blocked at much lower load in the distributed implementation.

One result not expected was that performance actually decreased when the packet size was increased! After checking the results again it was founded that a large average transfer size results in less movement of tokens and that the status information actually gives an even worse view of free resources in the network than for short transfers. In this case, a request is rejected if we do not think we will find any resources. This mechanism was introduced to avoid wasting control capacity when resources was exhausted.

The reason for this is that the status messages only describe "global" tokens covering all segments on the bus. A global token may be used by any node and is the only type of token in a DTM system without slot reuse. At a load higher than 1.00 a large number of tokens are segmented and the reuse scheme needs them on them being used by new requests. Therefore, the status message mechanism which is used, that was developed for a system without reuse, is limited in its ability to help new requests find free capacity, and can in the worst case result in higher blocking.

The throughput and access delay of the distributed token manager with bus lengths from 1 km to 1000 km is presented in fig. 13. A small 16 kbyte packet is sent between establishment

and tear-down of the channel, for large packets (100's of kbytes) the throughput performance of the system is much less affected of the bus length. The 1 km and 100 km busses give about the same throughput and access delay results as the 10 km bus because the latency introduced by using a 125 ms cycle dominates over the time of flight latency in the system. For the 1000 km long bus one see that the access delay is found to be much shorter than in the 1000 km system using a centralized token server, especially at low load are tokens found very close to where the user request arrived and delay is about the same for all systems. Even at very high load the access delay is about 1 millisecond shorter than in the system with the centralized server.

The centralized token manager system will have the same performance almost independently of the traffic as long as processing and signaling are sufficient. To evaluate the distributed system two other traffic generators were used. First, we used a generator that simulate user requests arriving in bursts. When a request arrives we defined a new request to arrive with 90 % probability after 200 ms. The result is that a burst of requests arrives to a node and forces high temporal locality of the source addresses. Second, to generate traffic more similar to client-server behavior the amount of traffic arriving were increased to 5 server nodes 0, 25, 50, 75 and 99. The probability of a server destination is also higher.

In fig. 14 is presented throughput and access delay performance of the distributed token server system.

It is clear that the distributed implementation have several benefits but also drawbacks compared to the centralized token server. The nodes may share the processing load, there is less need for high-performance token servers, redundancy may be higher and access delay may be shorter for low capacity request. The drawbacks is higher blocking for an inflexible user which may therefore need to retry several times to gain access to necessary resources. It is also clear that the status message and status table mechanism must be changed to avoid unnecessary blocking when slot reuse is allowed.

When the load increases above a certain point, it gets difficult to find free tokens, blocking becomes significant and utilization only increases marginally with load. The degree of blocking at a given load depends mainly on the minimum capacity requirement of user requests. The lower the minimum capacity, the higher the probability that the node will be able to satisfy the request. Furthermore, in these simulations only one round of token requests is sent out for each user request. Utilization can be increased by making several rounds (i.e., using retry), at the expense of longer access delay.

Assuming that the above waiting times are independent, the average medium access delay, if having a single control slot and if the network is lightly loaded, meaning that borrow is infrequent, will result in an average access delay around 125 ms (a single cycle).

In addition, due to fragmentation it may take many control messages to establish a channel, which adds even further to the load of the control channel, and thereby the access delay. In fact, without an efficient defragmentation scheme fragmentation is the dominating contributor to access delay in the simulations.

A consequence of DTM's slot reallocation mechanism is that it takes longer time to establish channels that need high bandwidth. This trade-off might be justified: the type of traffic that needs short medium access delay is usually less sensitive to the amount of bandwidth allocated for the transfer, and such traffic therefore can be accepted without much involvement of the reallocation protocol. For transfers that require high bandwidth, the access delay will be much longer and will almost always involve the reallocation protocol. However, in general high bandwidth transfers will probably be less sensitive to access delay.

These simulations have shown that the DTM fast circuit switching protocol performs well in a dual-bus shared media environment. Two slot (token) management schemes are analyzed and both perform well and can benefit from slot reuse. The

centralized scheme performs closest to the ideal protocol and also result in a simple implementation. The distributed system is more sensitive to user behavior and must therefor rely on status information that is broadcast frequently and on the defragmentation method according to the invention to decrease the number of control messages needed for channel establishment and slot reallocation. Using the defragmentation method for long busses the distributed model shows better performance than the centralized model (comparing fig's 11 and 13). A resource management system that combines the centralized and the distributed models using a small set of token server nodes is also possible.

Furthermore, connection establishment overhead can be designed to be very small, resulting in high utilization even for small (a few kbytes) transfers. Access delay is found to be a few hundred microseconds even at high load. The slot reuse method and arrangement can increase performance by a factor of two without introducing any extra hardware in the nodes. Using the slot reuse scheme it is even more important to use the defragmentation method according to the invention as fragmentation may occur in both slot and segment dimensions.



CLAIMS

1. A slot reuse method for increasement of the throughput in a DTM (Dynamic Synchronous Transfer Mode) network characterized in
 - 5 - that the DTM block token format is extended to include parameters describing the segment(s) between source and destination node,
 - that two or more segment-consecutive tokens representing the same slot range are merged into a single token when existing in
10 the free pool of a node, and
 - that simultaneous transmissions in the same slot over disjointed segments of the network are enabled.
2. A slot reuse method according to claim 1, characterized in that it is implemented in a central token
15 management scheme.
3. A slot reuse method according to claim 1, characterized in that it is implemented in a distributed token management scheme.
4. A slot reuse method according to claim 1, characterized
20 ized in that it is implemented in a resource management system that combines the centralized and the distributed models, hence using a small set of token server nodes.
5. A slot reuse method according to one of the claims 1-4, characterized in that it is combined with a defragment-
25 ation method in which
 - a home node is defined for each token at network startup or during network operation in such way that the tokens sharing the same home node will always define at least partly a continuous slot range,
 - 30 - free tokens are sent back to their respective home nodes when a significant time has passed and,
 - two or more slot-consecutive tokens with the same slot range are merged into a single token when existing in the free pool of a node.
- 35 6. A slot reuse method according to claim 5, characterized in that two or more tokens with the same slot or



segment range are merged into a single token when existing in the free pool of a node.

7. A slot reuse method according to claim 6, characterized in that when merging slot- and/or segment-consecutive
5 tokens which have the same slot or segment range always a segment merge is prioritized before a slot number merge even if the number of tokens will augment.

8. A slot reuse method according to one of the claims 1-7, characterized in that when a node get a token request
10 from a local user or a remote user a token is picked from the token pool using a best-fit algorithm in slot number and segment number dimension, in which algorithm the token with smallest area in the token map that fulfill the requested capacity is chosen.

15 9. A slot reuse method according to one of the claims 1-7, characterized in that when a node needs to request a token from other nodes and no token that fulfill the requested capacity exists the number of tokens that are prerequisites is minimized.