Smoothing Algorithms for DTM Networks

Csaba Antal (1), Sándor Molnár (2)
(1) Traffic Analysis and Network Performance Laboratory Ericsson Telecommunications Ltd.
e-mail:csaba.antal@eth.ericsson.se
(2) High Speed Networks Laboratory, Dept. of Telecommunications and Telematics, Budapest University of Technology and Economics
e-mail:molnar@ttt-atm.ttt.bme.hu

Abstract: Previous studies of Dynamic Synchronous Transfer Mode (DTM), which is based on multi-access dual fiber bus, showed that it could be a viable technology in the future. This paper points out that the characteristics of active and passive nodes are not balanced and the performance of the network can be increased by decreasing the need for slot allocation during call set-up. We propose a new type of smoothing algorithms, which is a new channel allocation algorithm that improves the fairness of DTM network with proper priority settings and the performance of channel allocation in ordinary operating conditions. Results are based on simulation, which was carried out by a self-developed simulator.

Key words: distributed channel allocation, fast circuit switching

1. INTRODUCTION

Dynamic Synchronous Transfer Mode (DTM) [5,10] is an attempt to build the next generation of local and metropolitan area networking technologies on fast circuit switching basis. It is a new broadband network architecture developed at the Royal Institute of Technology in Stockholm (KTH). The technology is in the focus of several Swedish companies, because it is the competitor to technologies like Asynchronous Transfer Mode (ATM) in special application areas.

ATM transfer capabilities define service classes, which are appropriate for several application types. However, ATM is not the most natural solution for applications with low delay and delay variance requirements due to its inherent cell (packet) switching characteristics. Applications with high bandwidth and low delay variance needs (e.g.: video on demand, video telephony) are more suited to the philosophy of circuit switched networks.

On the other hand ATM is a connection-oriented protocol. Therefore, interoperation with IP networks needs the emulation of the operation of connectionless networks. Interoperation with IP also limits the usage of the quality of service features of ATM.

DTM inherently supports real-time guarantees, which are crucial for services like voice or streaming video, and it was designed with IP technology in mind. Therefore it enables the efficient integration of voice and data traffic in the same network. DTM is therefore a candidate technology serving as a fast backbone for IP networks. DTM may also utilise wavelength division multiplexing (WDM). Therefore, it is one of the candidate technologies that may allow IP over WDM to evolve [10].

As DTM is a fast circuit switching technology, DTM connections are used only for the duration of data burst. If there is an idle period, resources are released. When data transmission starts again a new DTM connection is established. That is, user connections are split to several successive DTM connections.

The most important performance characteristics of the network *are connection set-up time* and *blocking probability*. If different bursts have considerably different set-up times and bursts are blocked within the call then there is less QoS guarantee for the whole connection. That is, the main benefits of circuit switching (like low and deterministic delay during the connection) are lost. Consequently, optimizing the mentioned characteristics is advantageous for burst-switching as well. This work is aimed at improving

the performance and fairness of DTM networks via improving set-up time and blocking probability.

DTM is not a well-known technology, therefore before we present our results we shortly describe its main characteristics and the most important results of its performance evaluation.

DTM is designed for unidirectional medium with multiple access. The total medium capacity is shared by all connected nodes using time division multiplexing. Previous proposals and implementations [5,6] are based on dual-bus topology.

The total capacity of the bus is divided into cycles of 125 microseconds, which are further divided into 64 bit long slots. The sequence of slots at the same position in successive cycles is called DTM channel or simply slot.

There are two types of slots (and so DTM channels): data and static ones. Data slots are used for data transfer. For each DTM channel there is a token, which is assigned to one of the nodes. Both free and used data channels are assigned to nodes. Each channel has exactly one owner at a time. If a node has the right to use a channel (i.e. it has the token), then it has full control on it: it can setup a connection on it, send data within the connection, release a connection using the channel, or give the channel ownership to another node.

At system start-up, data channels (tokens) are allocated by the nodes but they can be transferred dynamically during operation. *Nodes can ask others for free data channels if they do not have enough to serve a new request.* To facilitate channel reallocation, nodes maintain status tables that contain the number of free channels at other nodes. This procedure is called *set-up time channel allocation or set-up time slot allocation.*

The other type of slot, called static slot, is used for broadcast control channels between the nodes. Nodes send control information in their static slots and listen to all the other static channels to receive control information.

In the basic set-up time channel allocation algorithm of DTM (KTH algorithm) [5,6], the request or of nodes during channel allocation is based on the closest first algorithm. We proposed two other request orders in [1,3,4] and carried out a performance evaluation study focusing on the fairness of the three variants, which are:

– Closest First : KTH-CF algorithm [6]
  It operates according to the closest first rule. That is, the node that received a connection request when having too few free slots sends the first slot request to the closest node out of nodes with free slots according to the status table. If the some of free slots at these two nodes is not enough for the pending request, the second closest node with free slots is also asked for slots. If further slots are needed the closest non-requested nodes are contacted.

– Logical Ring : KTH-LR algorithm
  Nodes are ordered into a logical ring [2]. The order of channel requests is based on the location in the logical ring instead of the bus. That is, closest node is the first ring neighbour, second closest is the second ring neighbour, etc.
– Random : KTH-RA algorithm [3]
  Nodes choose the next node randomly to ask for channels out of nodes with free slots according to the local status table.

In all of the three above algorithms the number slot requests that a node can send out during a call set-up (retry limit) can be limited. The request order and the retry limit will be denoted in the name of the algorithms in the followings. For example, KTH-RA algorithm with retry limit of 10 is denoted by KTH-RA-10.

We have analysed the algorithms in two extreme conditions for short bus-length in [3]:

– Model 1 – status tables are *up-to-date* and nodes only try to get free channels from nodes with free channels
– Model 2 – status tables are *useless* and nodes try to get free channels from others without any apriori knowledge

Algorithms with up-to-date status tables (i.e. operating according to model 1) are denoted as KTH-S while the ones with useless status tables (i.e. operating according to model 2) are simply denoted by KTH (e.g. model 1: KTH-S-CF, model 2: KTH-CF).

Based on the study of the performance of set-up time slot allocation algorithms we showed in [2] that the addition of a new smoothing algorithm, which is called Background Channel Allocation algorithm (BCA), improved the performance of algorithms without status tables. In this paper we add another point of view for the motivation of smoothing algorithms, which is the lack of the balance between the performance of very active and relatively passive nodes. We also propose a new algorithm, which is called Set-up time Smoothing Algorithm (SSA), to improve the BCA algorithm. We show by simulation that the SSA algorithm is able to improve the fairness of the DTM network and it is also able to improve the performance of the DTM protocol. The analysis of the new SSA algorithm was carried out by self-developed simulation software.

The structure of the paper is as follows.

First, we present the motivation of smoothing algorithms. Then in Section 2, we describe the BCA algorithm and we also propose the new SSA algorithm. Simulation results can be found in Section 4. Finally in Section 5, we conclude the paper.

2. MOTIVATION

2.1 Asymmetry

In set-up time slot allocation algorithms, nodes initiating calls more often than the others have better performance characteristics. This effect is due to the cache-like behaviour of the KTH channel allocation protocol of DTM. That is, active nodes can reuse the free channels that are produced by local terminated connections. The effect can be observed at both algorithm types (i.e. with and without status table). In network where all - so-called client - nodes communicate to a dedicated - so-called server - node in the middle of the bus, the server node has lower blocking probability and shorter set-up time than clients have. KTH protocol is not efficient due to the asymmetry of the directions in a bi-directional client-server connection. The same effect can be interpreted as unfairness.

Table 1 illustrates the above property of KTH algorithm. Instead of displaying blocking probability or set-up time, the average number of slot request retries - needed for a successful connection establishment - are shown in the table. *Table 1* shows the average number of retries for KTH-RA algorithm (without status table) separately for the client-to-server, server-to-client directions and the average for all calls. The averages are calculated for different network loads. *Table 2* shows the same values for KTH-S-RA algorithm (with status table).

Table 1. Asymmetry of server-to-client and client-to-server directions, KTH-RA-10 without status table

Avr. # of slot req. retries vs. offered load	50%	70%	90%	110%	130%
Client-to-server	0.416	1.145	2.474	3.541	4.072
Server-to-client	0.009	0.033	0.121	0.26	0.389
All unidirectional	0.213	0.588	1.255	1.696	1.842

Table 2. Asymmetry of server-to client and client-to-server directions, KTH-RA-10 with status table

# of slot req. retries vs. offered load	50%	70%	90%	110%	130%
Client-to-server	0.193	0.438	0.73	0.905	0.969
Server-to-client	0.004	0.013	0.06	0.187	0.29
All unidirectional	0.099	0.226	0.395	0.547	0.629

It can be seen that *due to status table* the average number of retries is less than 1 also at 130% offered load. *Without status table* an average client node needs to request slots from 4 other nodes to collect the slots for a uni-directional connection to the server in average.

The asymmetry is also reflected in these numbers. E.g. at 50% offered load, 46 times more slot requests are needed to establish a client-to-server connection than for a server-to-client connection at both algorithms. Though this ratio decreases at higher offered load, it can not be neglected.

As this phenomenon may be the cause of unfairness, an algorithm is needed that makes balance between the characteristics of very active and less active nodes.

2.2 Too many slot requests

In addition to asymmetry a possible *improvement opportunity of algorithms without status table* can be seen in *Table 1* and *Table 2*. The number of slot requests needed to set-up a client-to-server connection is very high even at moderate loads. It is 1.145 at 70% load, which means that an average node asks slots for a successful connection more than once, which is a high value. Note that connections can be set up without slot request if the source node has free slots. The probability mass function of the same random variable (number of retries needed for a successful connection) can be seen for client-to-server connections in *Figure 1* and *Figure 2*.

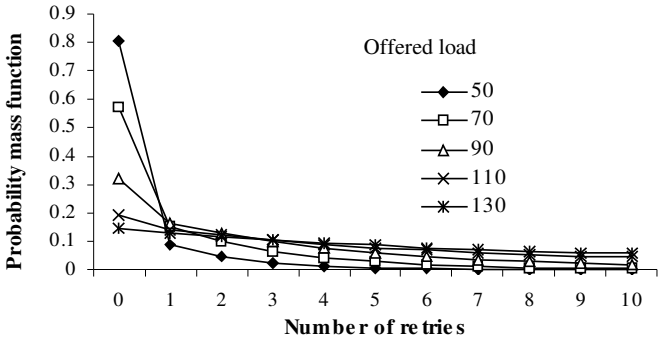

Figure 1. Probability mass function of number of slot request retries during a successful connection set-up, KTH-RA (without status table)

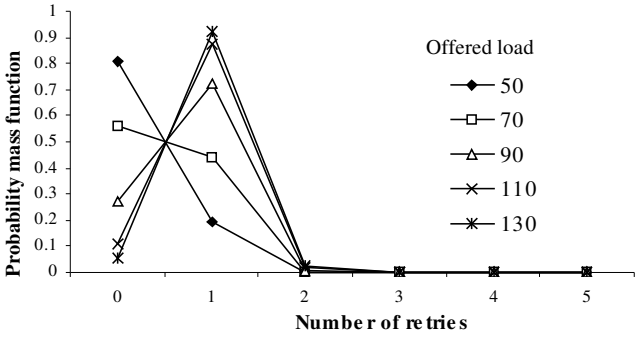

Figure 2. Probability mass function of number of slot request retries during a successful connection set-up, KTH-S-RA (with status table)

It can be seen that KTH-S-RA algorithm manages to collect the slot without slot request or with one slot request, as the probability of 2 or higher retries is very low. At the algorithm without status table, if the offered load is as low as 70% the probability that slots are needed from other node is 100-55=45%.

3. DESCRIPTION OF SMOOTHING ALGORITHMS

The goal of smoothing algorithms is to decrease (or eliminate) the need for slot allocation during call set-up and to balance the characteristics of active and passive nodes. To reach the goal smoothing algorithms are trying to distribute free slots amongst nodes according to their loads. We proposed BCA algorithm in [2] to decrease the need for slot allocation, i.e. to improve performance. Here, we propose a new and improved version of this algorithm: the Set-up-time Smoothing Algorithm. This section describes both algorithms

3.1 Background Channel Allocation Algorithm

Background Channel Allocation algorithm (**BCA algorithm**) [2], transfers slots between nodes in the background, independently of set-up requests coming from hosts. It can work parallel with any set-up-time algorithms. It is based on logical ring architecture.

In the algorithm nodes regularly exchange free channels with their direct neighbours along the logical ring.

The goal of the exchange *in the case of homogeneous network load* is to distribute free channels *evenly* amongst nodes. In order to achieve this goal, nodes check regularly if there is any difference between the number of local and neighbouring nodes' free channels. This process provides that neighbouring nodes have nearly the same number of free channels at any time instant, thus free channels are always distributed almost evenly amongst nodes.

This idea can be extended to a real algorithm, which considers the case of normal operation when the load is different at each node. A priority value for buses in both directions reflect the difference between nodes. That is, each node has a priority number for each bus, which depends on the traffic load sent to the given bus. Priorities can be constant or can change dynamically when adapting to the actual load of the network.

Exchange of free channels depends on the number of free channels and the value of priorities. Node *i* initiates slot allocation with its ring neighbour - node *i+1* - if expression

$$|(\text{free channels of node } i)(\text{priority of node } i+1) - (\text{free channels of node } i+1)(\text{priority of node } i)| \qquad (1)$$

can be decreased by slot allocation. The amount of slots to be transferred is determined so as to minimise (1) and considering that only free channels can be transferred. Node *i* asks slots from node *i+1* if the first term of expression (1) is below the value of the second term and it transfers slots if the first term is the higher one. That is, in the case of equal priorities, node *i* transfers one channel to node *i+1* if its number of free channels is higher by 2 than the ones of node *i+1*.

Node *i* calculates expression (1) whenever a local connection was set up or released (number of local free channels changed).

If the priority of a node is equal to zero then it is left out from the ring. The next successive node is the exchange partner instead. For example if the priority of node *i+1* is 0 for the corresponding bus then node *i+2* is the partner of node *i* for the allocation of free slots on that bus.

BCA algorithm is based on the comparison of the amount of local and neighbouring free channels. This is why it requires a very small status table where nodes keep a record of free slots of direct neighbouring nodes on the ring. Nodes send administration messages to the first upstream neighbouring node along the ring after each change in the number of local free channels in order to provide information for maintaining up-to-date tables.

Priority defined above does not effect directly the amount of bandwidth available for a node. It is rather related to the possibility of setting up a channel without slot reallocation, independently of the bandwidth used. This

definition of priority can be used for optimising the network utilisation and channel set-up times.

Priorities can be dynamic and static as well. In the case of dynamic priorities, a traffic estimation procedure modifies the priority of the node. Estimators use parameters of previous connections (e.g.: amount of required bandwidth and interarrival times) to calculate the current priority. If the characteristics of the traffic are known effective estimators can be constructed. However, estimators can be built without preliminary information about the traffic. Dynamic priorities are not used in this examination, as offered load during a simulation run was static.

The other solution is to assign static priorities to nodes where priorities are changed at management level. In this case the basis of priority assignment can be the role of the node in the network or the price paid by the customer of the node.

3.2 Set-up-time Smoothing Algorithm

Set-up-time Smoothing Algorithm (SSA) is our new and improved version of BCA. In BCA, slot exchange is performed always between the same nodes: only direct neighbours along a logical ring transfer slots between each other in the background. The advantage of this operation is that nodes only have to store status information about their ring neighbours. However, it has drawbacks too in real implementations. In certain cases, distribution of free channels may differ significantly from the priority distribution. If – for example – there are a few nodes with very low activity between nodes that have many free slots and nodes that have high priority, BCA does not transport free slots to high priority nodes.

In SSA, free slots can be exchanged between the parties of connections during connection set-up and release procedure. The rules of slot exchange are the same as it is in BCA: nodes transfer free slots if expression (1) can be decreased.

Exchange partners of SSA are always different, and we assume that SSA needs to operate without status tables. Therefore, nodes have to add additional information into DTM Control Protocol (DCP) Announce and Attach messages.

The SSA procedure during connection set-up is described below:

1. The sender node sends the number of its free slots in the DCP Announce message.
2. The receiver compares the number of its free slots and that of the sender node.
3. a, If the receiver node should send slots according to (1), it includes the number of slots to be transferred into the DCP Attach message, and

initiates a slot transfer procedure.

b, If the sender node should send slots, the receiver puts the number of slots it asks from the receiver into the DCP Attach message. When the sender received the DCP Attach message, it initiates a slot transfer procedure with the receiver.

Note that definition of sender and receiver node is based on the data transmission roles (not on the transmission of control information). As connections are uni-directional at DCP level, one of the parties of the connection is always sender, and the others are receivers.

Operation of SSA algorithm can be described in the same way for connection release procedure.

The main differences between BCA and SSA are summarized in *Table 3*.

Table 3. Differences between BCA and SSA algorithms

	BCA	SSA
Who are slot request partners?	neighboring nodes along logical ring	parties of connections
When do slot requests occur?	any time	during connection set-up and release
How to send information about the number of free slots?	in separate messages	in modified connection set-up and release procedure

4. SIMULATION RESULTS

We proposed BCA and SSA algorithms to balance performance characteristics of active and passive nodes and to improve performance of the network.

The primary goal of this subsection is to examine whether the above design goals of BCA and SSA algorithms are fulfilled or not. The effectiveness dependency on burstiness of traffic and bus-length is also investigated.

We used two types of traffic generators during the simulations. The first one is based on WWW traffic sources [7,8]. The inter-arrival time of connections is based on Weibull distribution. The holding time of a request is modelled by Pareto distribution. The second traffic generator model is based on an aggregated voice model. Traffic generators are modelled with Poisson arrival processes and exponential holding times. The bandwidth of connections is deterministic in both models: 1 slot in each cycle.

We use the so-called "**client-server**" network configuration [1,2] in the paper, where each client node initiates bi-directional connections to the special "server" node, which is located in the middle of the dual-bus. We

assume that all traffic generators have the same traffic parameters, and the number of generators connected to client nodes is the same. In the simulations to be presented, there is no client-to-client traffic.

In this paper, we examine the operation at two extreme bus-lengths. The so-called short dual-bus is 1 km long with 100 nodes. That is, the inter-node distance is 10m. At the long dual-bus the inter-node distance is 10 km, so the full bus-length is 1000 km.

The basis of our simulation work is the *steady-state mean of call blocking probability and connection set-up time.*

Connection set-up time consists of two parts:

– round-trip delay of the set-up message
– delay coming from slot requests

The first part depends on the location of the other party of the connection. This factor is proportional to the distance of client nodes from the dedicated server node. As differences caused by the physical distance of the other party of the connection are usually respected by customers, the delay of set-up message and its acknowledgement is subtracted from the connection set-up time when the bus is long. The resulted new characteristic is referred to as *average slot request time.*

4.1 Short bus, WWW traffic

First, the short bus and WWW traffic case is examined. Both smoothing algorithms perform best when using together with set-up-time slot allocation algorithms. As BCA uses logical ring during background allocation, it is applied together with KTH-LR. SSA is used with KTH-RA algorithm.

4.1.1 Performance

Performance of smoothing algorithms is simulated with the following configuration:

– Retry limit = 10
– client-server load profile
– priority settings : nodes with any activity have 1 as priority, idle nodes have 0 priority (separately for both directions)

Simulation results are displayed in *Figure 3* and *Figure 4*.

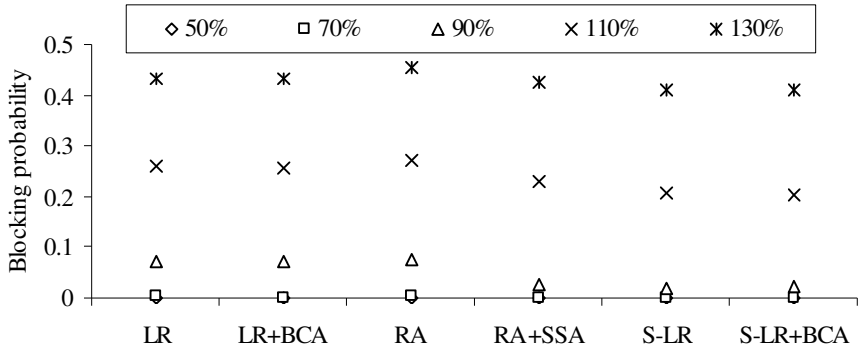

Figure 3. Blocking probability of smoothing algorithms

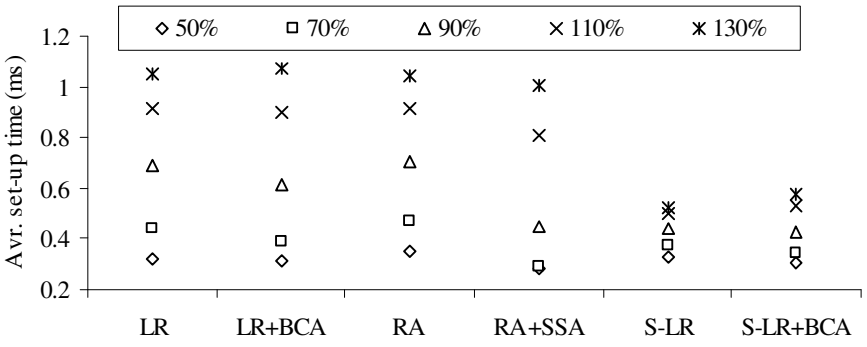

Figure 4. Average set-up time of smoothing algorithms

Simulation results can be used to define the application area of smoothing algorithms, to select the most appropriate offered load range and to study the effect of status tables.

Figures show that smoothing does not improve the performance of algorithms with status table. Though a very small improvement can be noticed in set-up time at 50%, 70% and 90% load, at higher loads the performance of algorithms with status table degraded due to smoothing algorithms. The main performance gain of smoothing algorithms is that they could decrease the number of slot allocation retries needed for a successful connection. As *Figure 2* shows, the number of slot allocation retries is low when status tables are present, so there is no room for this kind of optimisation.

Improvement of set-up time of algorithms without status table is, however, significant if the network is not overloaded. Among KTH-LR, KTH-LR+BCA, KTH-RA and KTH-RA + SSA algorithms the last one is the best. SSA algorithm is most effective in 50%-100% offered load range. The highest improvement on set-up time is 0.25 ms at 90% load and KTH-RA + SSA algorithms. That is, SSA decreased average set-up time by 35 %. At 130% offered load, the gain of SSA algorithm is only 0.03 ms. Less effective ranges of smoothing algorithms can be explained with the followings:

Above 100% load, "there is nothing to smooth", i.e. there are very few free slots in the system.

Below 50% offered load, "there is nothing to optimise". i.e. there are many free slots in the system.

Blocking probability of the system was decreased due to SSA algorithm at each offered load, but its effect is small. The highest improvement is at 90% offered load, where blocking is lower with 0.05 (from 0.07 to 0.02).

The range between 50% and 100% offered load is the most important one for well-designed networks. If offered load is higher for longer time, the network is mis-dimensioned, and it is not able to provide efficient services. Lower offered load - for a long time - means that the network is over-dimensioned and the operator paid for unused bandwidth.

It is also interesting that at 50%, 70% and 90% offered loads KTH-RA with SSA have nearly the same performance as KTH-RA with status table.

4.1.2 Priority settings

In the previous section performance was examined with fixed priority settings. Now, we show how to find an optimal priority settings at client-server configuration taking into account the viewpoint of *symmetry and performance*.

Priorities of client nodes in this section are the same as they were before. Priority of server node, however, is varied between 50 and 0.05. The effect of priority settings on client--to-server connections, server-to-client connections and bi-directional connections can be seen in *Figure 5* and *Figure 6*. The first item is KTH-RA algorithm without smoothing at each offered load. At KTH-RA + SSA algorithm, the ratio of server priority and client priority is displayed after the name of SSA algorithm.

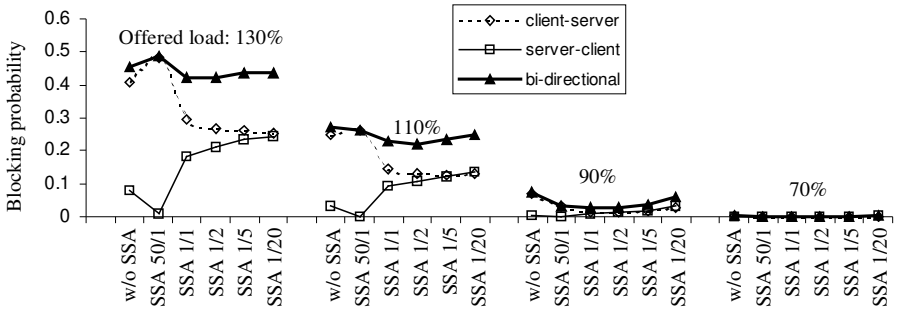

Figure 5. Effect of priority settings on blocking probability

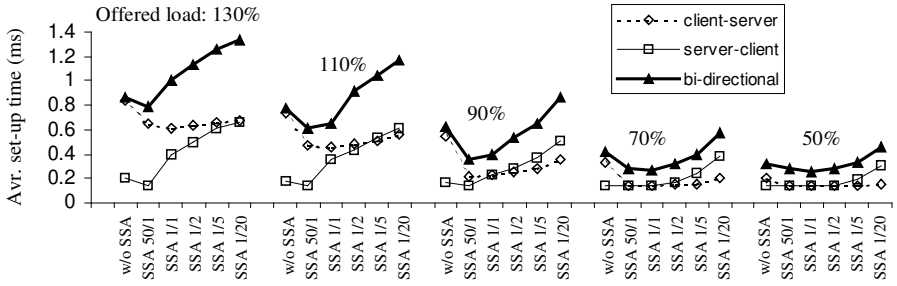

Figure 6. Effect of priority settings on average set-up time

First, let us take the *viewpoint of symmetry vs. priority ratios*. *Figure 5* shows that the lower the priority of server is the closer the blocking probabilities of client-server (uplink) and server-client (downlink) connections are. Blocking of uplink connections decreases and blocking of downlink connections increases when the priority of server node decreases. Blocking probability of uplink connections, however, is always above that of downlink connections.

Average set-up time of uplink connections can exceed that of downlink connections at certain priority ratios. The cross-point of uplink and downlink curves depend on the offered load of the network. At 130% load downlink/uplink ratio is 1/20, at 90% load it is 1/1. Priority setting of server node mainly influences the set-up time of downlink connections. Set-up time of uplink connections does not decrease in case the server has lower priority.

Both figures show that priority is an effective tool to balance the characteristics of nodes with different load of connections.

The next question to answer is *how performance of bi-directional connections depends on priority ratio*. Based on *Figure 6*, optimal priority is the one, which strengthen cache effect, i.e. the higher the priority of server the lower average set-up time of bi-directional connections is. Priority of server does not effect so significantly the blocking probability curve, except that blocking increases if the server has too high priority.

4.2 Effect of long bus and less bursty Poisson sources

The previous subsection analysed SSA algorithm in detail. This section examines the effect of longer bus-length and less bursty traffic sources. Main configuration settings are the same as in the previous section:

- Retry limit = 10
- client-server load profile
- priority settings : nodes with any activity have 1 as priority, idle nodes have 0 priority (separately for both directions)

4.2.1 Long bus

Table 4 shows the performance of smoothing algorithms. The same conclusions can be drawn from this table as we obtained for short bus. Namely, smoothing does not improve significantly (rather degrade) the performance of slot allocation algorithms with status table. Set-up time of algorithms without status table, however, is decreased significantly in 50%-100% offered load range. SSA improved slightly blocking at almost every offered load. The largest improvement of set-up time is at 90% offered load: it is equal to 0.81 ms, which is 40% of the set-up time of KTH-RA algorithm. The largest improvement of blocking is 0.05 at 70% and 90% load.

Table 4. Performance of smoothing algorithms, long bus, bursty traffic

Load(%)	50	70	90	110	130	50	70	90	110	130
Algorithm	Blocking probability					Avr. slot request time				
KTH-LR	0	0.004	0.07	0.262	0.438	0.832	0.977	1.28	1.572	1.753
KTH-LR+BCA	0	0.007	0.075	0.252	0.441	0.822	0.885	1.194	1.54	1.764
KTH-RA	0	0.005	0.076	0.273	0.466	0.994	1.358	2.04	2.596	2.933
KTH-RA+SSA	0	0	0.025	0.227	0.427	0.786	0.808	1.23	2.325	2.89
KTH-S-LR	0	0	0.018	0.205	0.42	0.835	0.891	1.004	1.163	1.251
KTH-S-LR+BCA	0	0	0.019	0.203	0.423	0.81	0.842	0.99	1.196	1.315

4.2.2 Poisson traffic

Our last topic is the effect of burstiness on the effectiveness of smoothing algorithms. Poisson traffic is usually less demanding for the network, so the performance of set-up time algorithms improved compared to WWW traffic.

Table 5. Performance of smoothing algorithms, short bus, Poisson traffic

Load (%)	50	70	90	110	130	50	70	90	110	130
Alg.	Blocking probability					Avr. set-up time				
KTH-LR	0	0	0.005	0.202	0.419	0.266	0.274	0.388	0.813	0.999
KTH-LR+BCA	0	0	0.007	0.204	0.418	0.263	0.268	0.352	0.832	1.038
KTH-RA	0	0	0.006	0.219	0.451	0.263	0.272	0.398	0.745	0.878
KTH-RA+SSA	0	0	0.001	0.204	0.43	0.263	0.263	0.314	0.854	1.011
KTH-S-LR	0	0	0	0.186	0.413	0.265	0.271	0.319	0.438	0.463
KTH-S-LR+BCA	0	0	0	0.185	0.412	0.263	0.267	0.307	0.481	0.518

Simulation results in *Table 5* show, however, that SSA algorithm is more effective at bursty traffic than here. SSA decreased blocking of KTH-RA without status table at any load. It also decreased set-up time below 100% offered load. When, however, offered load is above 100% SSA and BCA increased set-up time, so it is worth to switch off smoothing when the system is overloaded. The performance of algorithms with status table is not improved with smoothing algorithms.

5. CONCLUSION

We have shown in this work that there is an asymmetry in set-up-time slot allocation algorithms of DTM. We have also shown that it can be corrected with smoothing algorithms using proper priority settings. As set-up-time algorithms provide better service for active nodes than for less active ones, their priority should be set to a smaller value to reach balance.

We have also shown the exact dependence of asymmetry on priority values. Simulation results show that - in the case of client-server network configuration – blocking probabilities of the down-link (server-client) and up-link (client-server) connection are equal if the priority of the server is 1 and that of the clients is 20. The priority ratio, where average set-up times of up-link and down-link connections are symmetrical, depends on the offered load in the network. E.g. at 130% offered load the symmetrical server/client priority ratio is 1/20; at 90% offered load it is 1/1.

Simulations also proved that adding smoothing algorithms to set-up time algorithms without status table improves the performance of the DTM dual-bus if offered load is between 50% and 100%.

It can be concluded that the priority that provides the smallest average set-up time for the bus is the one that strengthens cache effect. That is, the higher the priority of the server is the lower the average set-up times of bi-directional connections are. Priority of server does not effect the blocking probability curve significantly, except the case when it is too high.

It is also shown that smoothing algorithms improve the performance of the system more efficiently in the case of bursty sources than with Poisson traffic.

6. REFERENCES

1. Cs. Antal, J. Molnár, " Fairness of a DTM Dual-bus with Large Inter-node Distances", 8th International Conference on Telecommunication Systems, Modeling and Analysis; March 9-12, 2000, Nashville, Tennessee, USA, pp. 414-420
2. Cs. Antal, J. Molnár, S. Molnár, G. Szabó, "Performance Study of Distributed Channel Reallocation Techniques for a Fast Circuit Switched Network", Computer Communications Vol. 21 Num. 17, November 1998, Special Issue on the Stochastic Analysis and Optimal Management of Communication Systems, pp. 1597-1609
3. Cs. Antal, J. Molnár, S. Molnár, "Fairness Issues in a New Dual-Bus Networking Technology"; Proc. 7th IFIP/ICCC Conference on Information Networks, Aveiro, Portugal, June 1998, pp. 83-98
4. Cs. Antal, "Performance Analysis of the DTM Fast Circuit Switched Networking Technology", Ph.D. dissertation, Technical University of Budapest, Hungary, 1999
5. C. Bohm, M. Hidell, P. Landgren, L. Ramfelt, P. Sjödin, "Fast Circuit Switching for the Next Generation of High Performance Networks", IEEE J. on Selected Areas of Communications, Vol. 14., No. 2, February 1996, pp.298-305
6. C. Bohm, P. Lindgren, L. Ramfelt P. Sjödin, "Resource Reservation in DTM", 1st IEEE Symposium on Global Data Networking, Cairo, Dec. 1993, pp. 191-197
7. M. E. Crovella, A. Bestavros, "Self-Similarity in the World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, 5(6): pp.835--846, December 1997
8. S. Deng, "Empirical model of WWW document arrivals at access link", IEEE International Conference on Communications, vol.3, ICC'96, 23-27 June 1996, Dallas, TX, USA pp. 1797-802
9. A. M. Law, W. D. Kelton, *"Simulation Modeling & Analysis"*, McGraw-Hill International Editions, 1991
10. P. Lindgren, "A Multi-Channel Network Architecture Based on Fast Circuit Switching", Ph.D. thesis, ISRN KTH/IT/R-96/08-SE, Stockholm, May 1996