PERFORMANCE STUDY OF DISTRIBUTED CHANNEL ALLOCATION TECHNIQUES FOR A FAST CIRCUIT SWITCHED NETWORK

Csaba Antal[1], József Molnár[2], Sándor Molnár[3], Gábor Szabó[4]
High Speed Networks Laboratory
Department of Telecommunications and Telematics
Technical University of Budapest
Sztoczek u. 2., Budapest, Hungary, H-1111
E-mail: {[1]antal; [2]molnarj; [3]molnar; [4]szabog}@ttt-atm.ttt.bme.hu

Abstract

Dynamic Synchronous Transfer Mode (DTM) is a next generation high speed networking technology. It is based on fast circuit switching and uses distributed channel allocation on shared media.
The paper proposes a channel allocation algorithm for DTM, which improves the average call set-up time and call blocking probability characteristics. Another modification is also recommended to the operation, which provides fairness in the case of the examined network model.
Many variants of the proposed and known techniques are compared in various network conditions. The performance evaluation of the algorithms is carried out by simulation and practical conclusions are derived from the results.

Keywords: circuit switching, channel allocation methods, dual-bus, performance

1. Introduction

The most dominant broadband networking technology in research and development has been Asynchronous Transfer Mode for several years [13,15]. It is planned to support applications with requirements spanning from videoconference (real-time, high bandwidth) to telephony and general computer data transmission. ATM transfer capabilities define service classes, which are appropriate for several application types. ATM is also the transfer mode of broadband ISDN. However, it 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. Dynamic Synchronous Transfer Mode (DTM) is an attempt to build the next generation of 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. Performance analysis of DTM in the case of bus and fully connected mesh topology was also investigated in [4]. Extending DTM to provide various ATM-like service classes was proposed in [2].
Performance studies were based on a single distributed channel allocation algorithm, which will be referred to as KTH algorithm. This paper proposes a new algorithm [1], which can be used as an extension to the earlier method.

Other areas of telecommunications like mobile communications use several types of algorithms for distributed channel allocation [10, 12] but it is difficult to adapt them to the specific area of DTM networks.
DTM uses dual bus architecture so as with DQDB networks. However, the main disadvantage of DQDB, which prevented its wide acceptance, was unfair operation in the case of high network loads [11]. Because of this similarity fairness is an interesting issue in the case of DTM networks as well. Though we recommend a modification to the KTH channel allocation procedure [1], which makes it fair in the examined circumstances, an extensive fairness study is out of the scope of this paper.
The main emphasis is on the performance characteristics of the network like average set-up time and blocking probability. Both of these parameters are especially important if burst-switching [3] is applied. Computer traffic has a bursty behaviour that is the basis of the train model [9] and the basis of the traffic model of our simulation [8]. Burst switching can result in a better utilisation. However, 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 optimising the mentioned characteristics is advantageous for burst-switching as well.

The paper is structured as follows: in Section 2 we introduce the general DTM concept. The next section describes network elements of the simulation model. In Section 4 we compare set-up time and blocking parameters of various algorithms. The algorithms are variants of two basic methods. We recommend a modification to one of the basic methods, which is fair in the case of homogeneous network load in contrast to the know version. The other algorithm is developed by the authors, and is aimed to improve the performance parameters of the network due to background channel allocation. We examine the effect of status tables, limited channel allocation retries and limited control capacity. Finally in Section 5 we conclude the paper.

2. Dynamic Synchronous Transfer Mode

The operation of DTM is based on multirate and either unicast or broadcast channels. It is designed for unidirectional medium with multiple access. The total medium capacity is shared by all connected nodes. Previous proposals and implementations are based on dual-bus topology. The architecture can be extended to include a large number of connected buses using switching nodes.
The most important elements of a DTM network are the nodes and the hosts. *Nodes* are networking devices connected to the dual bus. *Hosts* are end-devices with a simple interface that connects them to a node. Host-host communication is based on the assistance of nodes. Nodes are responsible for resource allocation, connection establishment and release along the bus. Figure 1 shows the set-up of a single-bus network.

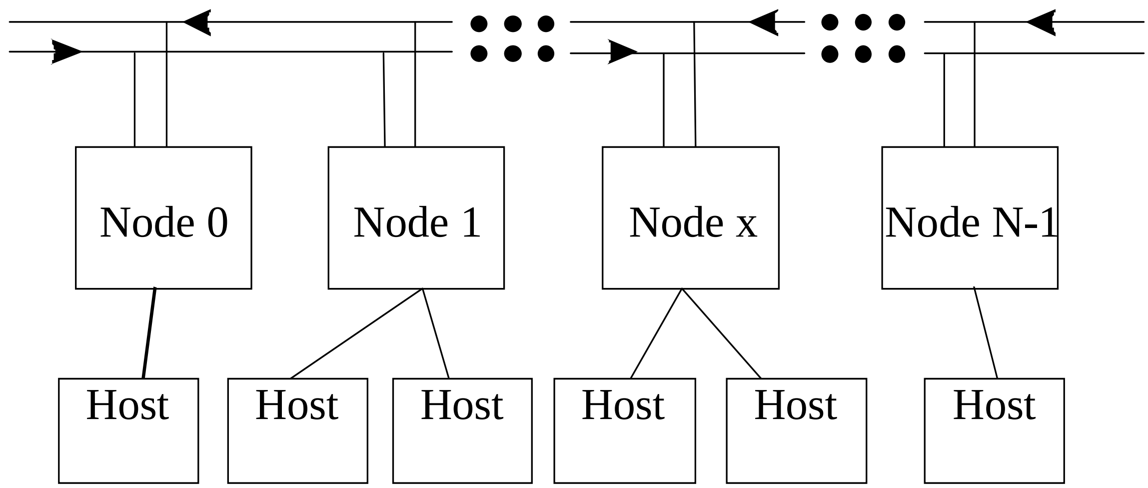

Figure 1: Structure of a DTM bus

Buses can operate at different bit-rates so as with the existing computer and telecommunication networks, a hierarchical structure spanning from local area networks to wide area networks can be constructed.

The total communication channel on the physically shared medium is realised by a time-division multiplexing scheme. The total capacity of the bus is divided into cycles of 125 microseconds, which are further divided into slots. A slot consists of a 64-bit data-word and some additional management bits. The sequence of slots at the same position in successive cycles is called DTM channel.

There are two types of slots (and so DTM channels): data and static slots.

Data slots are used for data transfer. The number of data channels specifies the bit-rate of a DTM connection in a cycle. There is a token for each DTM channel, 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 owns the token for a channel, then it has full control on its use: it can set up a connection on it, release a connection using the channel, or give the channel ownership to another node.

At system start-up, data channels (tokens) are allocated to the nodes but they are transferred between nodes dynamically during the operation. Nodes can ask others for free channel if they have not got enough free data channels to serve a new request. This procedure is called channel reallocation.

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

3. Simulated Network Structure

The analysis of channel allocation algorithms can be done based on different assumptions regarding the network structure. This section explains the main assumptions and decisions we made when choosing the simulated network structure. It has three parts as shown in Figure 1: network model, node model and host model.

3.1 Network Model

In this paper we examine a network consisting of a dual-bus and 100 nodes. The total bus length is 10 km. One host is assigned to each node. Hosts, which act like traffic generators in the simulator, generate the same amount of traffic with the same distribution (described in Section 3.3). The parameters of the traffic generators are varied to set the proper offered load for the network.

The destination of all point-to-point bi-directional connections is the host connected to the node at the end of the bus (server host), independently of the generator host. The host at the end of the bus is referred to as server and the others as clients, because it can be a model of a client-server network. There is no data transfer between any two client hosts.

DTM is based on a ”sender based channel reservation” algorithm. That is, bi-directional point-to-point connections correspond to two unidirectional point-to-point connections at the DTM access control level. In the direction towards the server, client nodes are responsible for channel reservation. In the backward direction, the server reserves the channels.

This network model results homogeneous offered load along the bus. In the direction towards the server, client nodes generate the same load and compete for tokens while the server is idling. In the other direction all nodes are idling, only the server builds up connections and transmits data, therefore there is no competition for tokens.

This homogeneous load will help us to find the reasons why KTHCF channel allocation is unfair (see in Section 4.1.1). This model allows us the simplified discussion of the BCA algorithm (see in Section 4.1.2) because the right choice of priorities is a difficult task in a general situation, but straightforward in this case.

Though this model is simple, it is appropriate for the performance analysis of different variants of channel allocation methods. A sophisticated fairness analysis requires more difficult network model, where the load of the bus is irregular. A thorough fairness study is the subject of our future work.

3.2 Node Model

Building up a proper node-model is essential if we want to analyse the operation of the network in the case of different overload situations. In the case of overloaded processing capacity input control buffers are used for storing messages that are waiting for the node processor. If control capacity is too low, output control buffers are needed to delay control messages until free control slots are available. The node model can be seen in Figure 2.

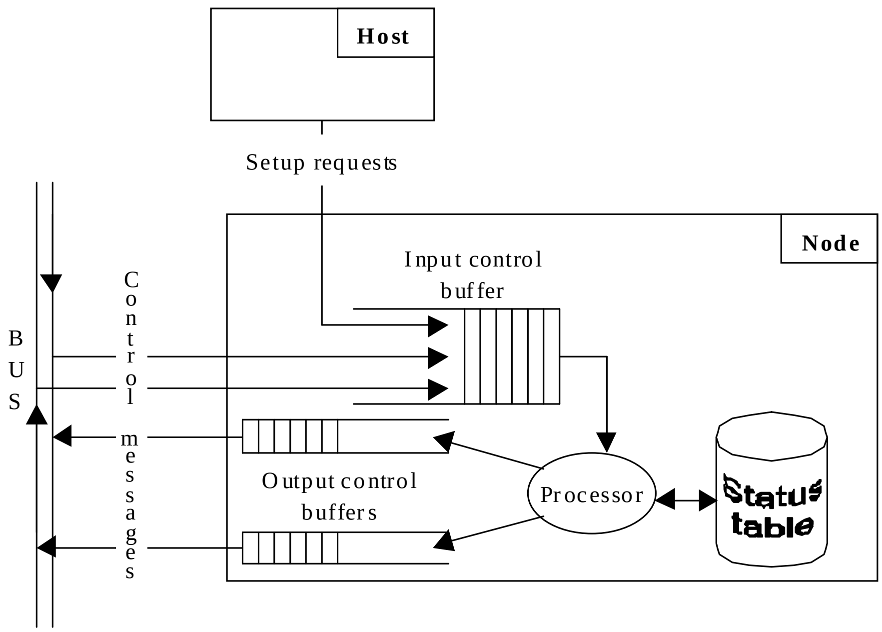

Figure 2: DTM node model

Input control buffer is used to store control messages received from other nodes or local hosts until the processor can serve them. The buffer-size should be large enough to store control messages of a few cycles. We assumed that the processing time for all control messages is the same ($5\mu s$).

Nodes have output control buffers for buses in both directions. The first control message in an output buffer is transmitted to the proper bus if the next control slot assigned to the node arrives. For simplicity, we assume that each control message can be transmitted in a single time-slot (64 bits). We assign one control channel to each node.

In order to keep even a congested node in operation, message dropping and call blocking mechanisms have to be applied at the node. We assume the following rules independently from the used channel allocation algorithms.

- Control messages from other nodes that require a reply (connection set-up request, channel request, connection release request, BCA request), or necessary for the node to continue its operation (connection set-up reply, channel request reply, BCA reply) are never dropped even if the input buffer overflows. If these types of messages were discardable, only time-outs would solve the problem of closing broken channels, which should be avoided in a high-speed network.
- Control messages from other nodes that don't require a reply (e.g.: status table updates) are dropped if the input buffer exceeds a given value. Though it causes small inconsistencies (e.g.: in status tables), it does not set back the operation of the nodes while the number of messages waiting for the node processor is decreased.
- Auxiliary messages sent to all other nodes (e.g.: status table updates and balancing messages) are dropped if the output buffers exceeds a given value. This reduces the congestion in the control capacity, while it causes only a small inconsistency in the operation.

- Set-up requests from a local host are blocked immediately, and they are not passed to other nodes, if the output buffer exceeds a given value. This rule moderates the congestion in the signalling capacity as well.
- If the output buffer of an initiator node overflows, the calls being set up are blocked, if the node tries to send a connection set-up request for this call.

3.3 Host Model

Hosts are traffic generators in our model. Our traffic model is based on World Wide Web traffic, because according to the present network traffic predictions a dominant part of future data services will generate WWW traffic. The analysis of Web traffic showed that the user-initiated TCP session arrival process could be well modelled by Poisson processes like in classical telephony [14]. However, the Poisson process cannot be used for modelling the arrival of WWW requests because it contains several non user-initiated requests. When a user requests a page, the browser program generates a series of additional requests to download the images of the requested page. Several studies suggest the use of long-tailed distributions such as Weibull or Pareto distributions for modelling the arrival process of WWW and for estimating the size of requested documents [8, 7].
Our traffic model defines the distribution of three parameters based on these studies:

- interarrival time, which is the time between the connection set-up requests
- holding time, which is the duration of a connection
- bandwidth, which is the bandwidth reserved for the connection

The inter-arrival time *X* is modelled by a Weibull distribution given by the probability density function

$$f(x) = \lambda^{\beta} \beta x^{\beta-1} e^{-(\lambda x)^{\beta}} \tag{1}$$

where the parameter β and the parameter λ depend on the generated traffic profile. Analytical studies of arrival process of WWW requests suggested the use of parameter $\beta = \frac{1}{3}$ [8]. With this value the mean of the inter-arrival time is

$$E(x) = \frac{\lambda}{6} \tag{2}$$

The holding time *T* of a request is modelled by the Pareto distribution given by the probability density function

$$f(t) = \frac{\alpha k^{\alpha}}{t^{\alpha+1}} \tag{3}$$

where the parameter is chosen to be $\alpha = 1.9$. The parameter *k* depends on the assumed mean size of the files to be transmitted.
The mean holding time *T* of a requested connection is

$$E(t) = \frac{\alpha}{\alpha - 1} k \tag{4}$$

The parameters were selected based on the analysis of measured WWW traffic [8].
In our model, all hosts initiate bi-directional point-to-point connections and they require the bandwidth of one channel in both directions. That is, the requested bandwidth is deterministic and its value is 512 kbps.

4. Channel Allocation Algorithms for DTM

In this section we examine two types of channel allocation algorithms: set-up time [4,1] and background allocation[1]. A modification to the set-up-time method and background channel allocation is proposed here. In Section 4.1 we evaluate the variants of these algorithms without using status tables at nodes. The effect of limiting the number of channel allocation cycles is also shown here.
In Section 4.2 we evaluate the same algorithms when status tables are used. The effect of limited control capacity is shown there.

4.1 Algorithms Combined with Set-up-time Channel Allocation Algorithms without Status Table

4.1.1 Set-up-time Channel Allocation Algorithms

DTM uses a distributed channel allocation algorithm [5]. Its operation is based on set-up-time channel allocation, which works as follows (without specifying all the details):
At the reception of a connection request from a host, the node first checks its local pool to see if it has enough channels to satisfy the request. If so, it immediately sends a connection establishment message to the destination node. Otherwise, the node first requires data channels from other nodes on the bus. The node that receives this request and has unused data channels, offer them to the sender node. We refer to this operation as set-up-time channel allocation because a node asks for channels after connection set-up was initiated and it noticed that the call could not be served from local free channels.

[4] proposed a procedure for channel allocation, which we call **KTH algorithm**. The algorithm works as follows. If a host requires a connection with M channels and the node has N free channels where N<M, it sends out requests requesting M-N channels. The node first sends a request to the closest node. The node that receives the request for K channels and has an amount of J free channels will always offer min(J,K) channels. If the node transferred J channels (J<K), then the requester node sends a message with channel request to the second closest node, and so on. The requester node sends out messages until the number of retries reaches a limit or the necessary number of channels is collected. If the required number of channels is together the node sends a set-up request to the destination node. After the acknowledgement arrived from the destination data transmission can start immediately.

In this algorithm, channel requests are sent out in the order of physical distance measured from the requesting node. In the case of bus topology, this channel allocation algorithm is unfair. First, nodes in the middle part of the bus need more processing power to answer channel requests. Second, they have less free slots on average. As a result of

[1] These algorithms were accepted as a patent application [1]

these two effects, outside nodes are in a more favourable situation than middle ones, this can be reflected in blocking parameters as well.
It can be seen intuitively too that the ends of the buses cause unfair operation. The difference between nodes at different locations can be seen in the following figures. Figure 3 presents the average number of free channels at different nodes. It can be seen that nodes in the middle have less free channels in average in the case of KTH algorithm. The ending part of the buses effects the free channels of just a few nodes in the outer parts of the bus (~10 nodes at each end).

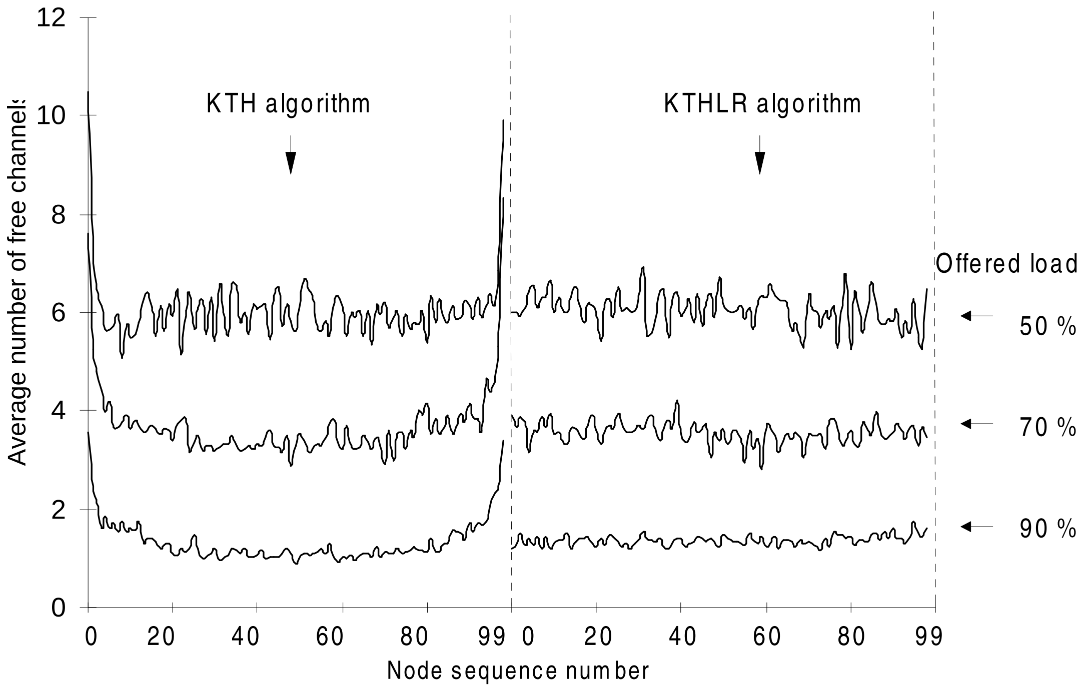

Figure 3: Average free channels of nodes at different locations (KTH and KTHLR algorithms)

Figure 4 shows the average connection set-up time at different nodes along the bus in the case of different offered loads if no retry limit is applied. In this case set-up times are shorter for nodes in the outer parts of the bus. The reason of this effect can easily be derived from Figure 3: Nodes in the outer part of the bus have more channels in average. The closer a node is to the end of the bus -> the closer is to the ”channel source” -> the fewer channel allocation retry is needed to ask from the ”channel source” -> the shorter is the set-up time.

Offered load here, and in the paper, is the ratio of the sum of the requested volumes (holding time of the call * bandwidth of the call) during T and the maximum transmittable volume during T (T * total bandwidth of the bus) where T>>0.

The blocking of algorithm with no retry limit is fair because nodes can ask all other nodes for channels. Blocking occurs if there is no free channel in the system. The location of nodes affects only the number of necessary retrials that is reflected in the set-up times. If we limit the allowed number of retrials this effect will be shown in the blocking as well.

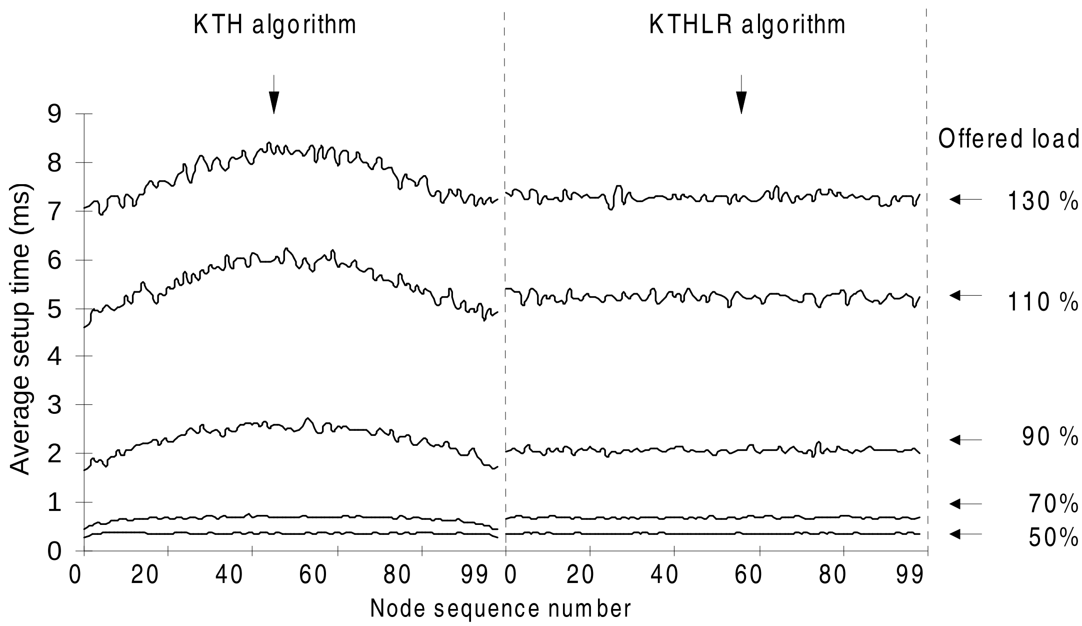

Figure 4: Average connection set-up time of nodes at different locations (KTH and KTHLR algorithm with no retry limit)

If the retry limit is 5 then the unfair distribution of free channels is reflected in the blocking probability as well. It can be seen in Figure 5 that nodes in the outer parts of the bus have lower blocking probabilities than ones in the middle in the case of KTH algorithm. The difference can be as high as 0.1. Set-up times remain unfair too.

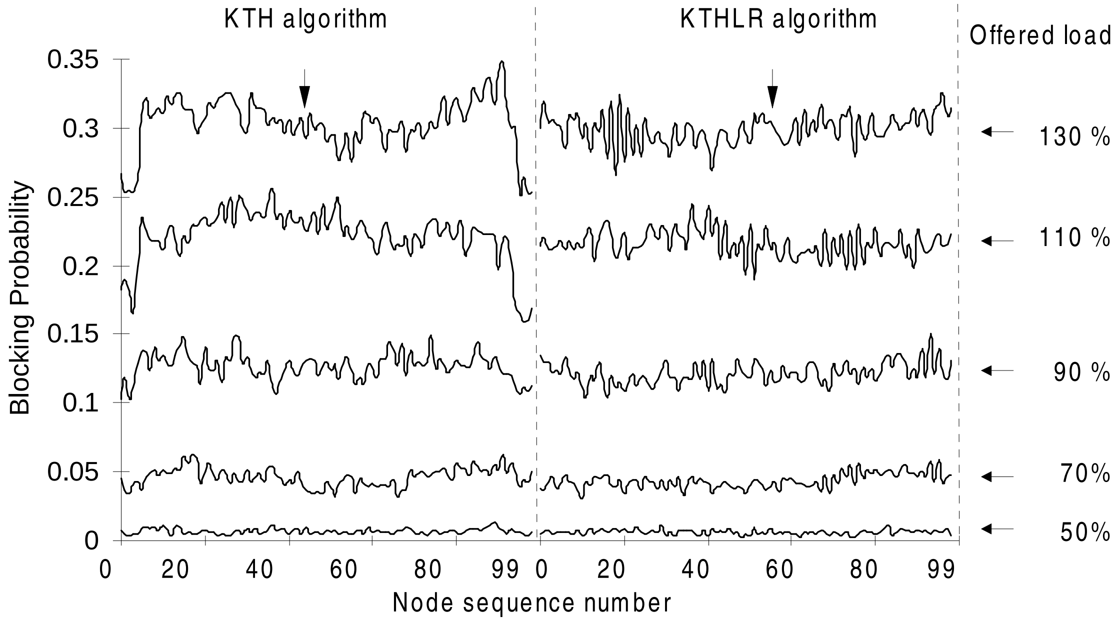

Figure 5: Blocking probability of nodes at different location
(KTH and KTHLR algorithm with retry limit of 5)

The unfair operation can be corrected in the case of homogeneous network load, if nodes of a bus are ordered into a logical ring and the order of channel requests is based on the location in the ring instead of the bus (**logical ring KTH algorithm, KTHLR**). If a node needs channels, first asks its first neighbouring node along the ring, then the second ring-neighbour and so on. If propagation time differences are negligible, then the operation is fair. Fair distribution of free channels among the nodes can be seen in Figure 3.
We used a logical ring, where the sum of the square of distances between neighbouring nodes is minimal. It can be constructed so that the second neighbouring nodes on the bus are successive nodes on the ring except the edges of the bus. The two neighbouring nodes of outer nodes on the ring are the first and second neighbours on the bus. The structure is illustrated in Figure 6.

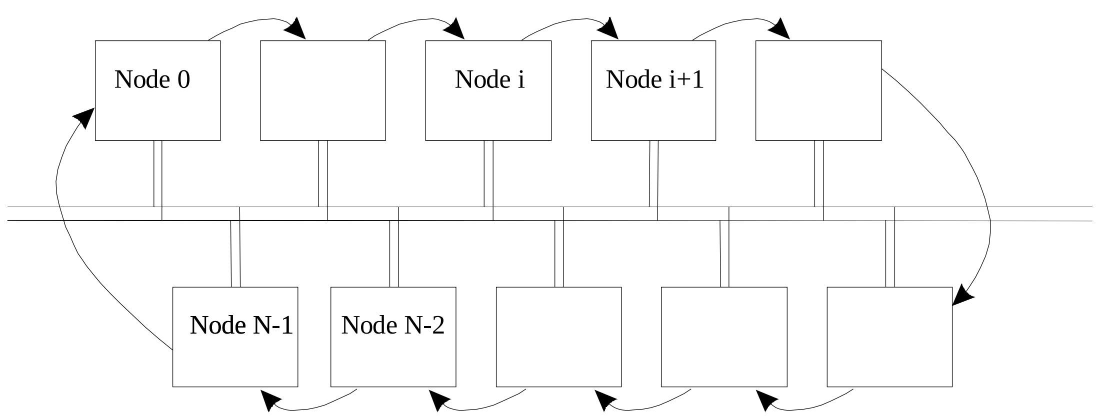

Figure 6: Logical ring structure

Set-up time with retry limit 5, and blocking with no retry limit can be seen in Figure 4 and Figure 5 in the case of logical ring assignment. We can see that the operation became fair. Though we can expect a small improvement of performance due to fair operation [6], this effect is not significant.

We have seen so far that KTH algorithm is unfair even in the case of homogeneous network load. KTHLR algorithm was proposed to provide fairness in the case of the examined model.

Both KTH and KTHLR algorithms have the drawback that even in the case of normal load conditions the number of channel requests before a connection can be established is high. Figure 7 shows the probability mass function of the number of requests sent before a connection was established (only for successful calls). We can see that even if the offered load as low as 70% the probability of channels are needed from other nodes is 100-45=55%.

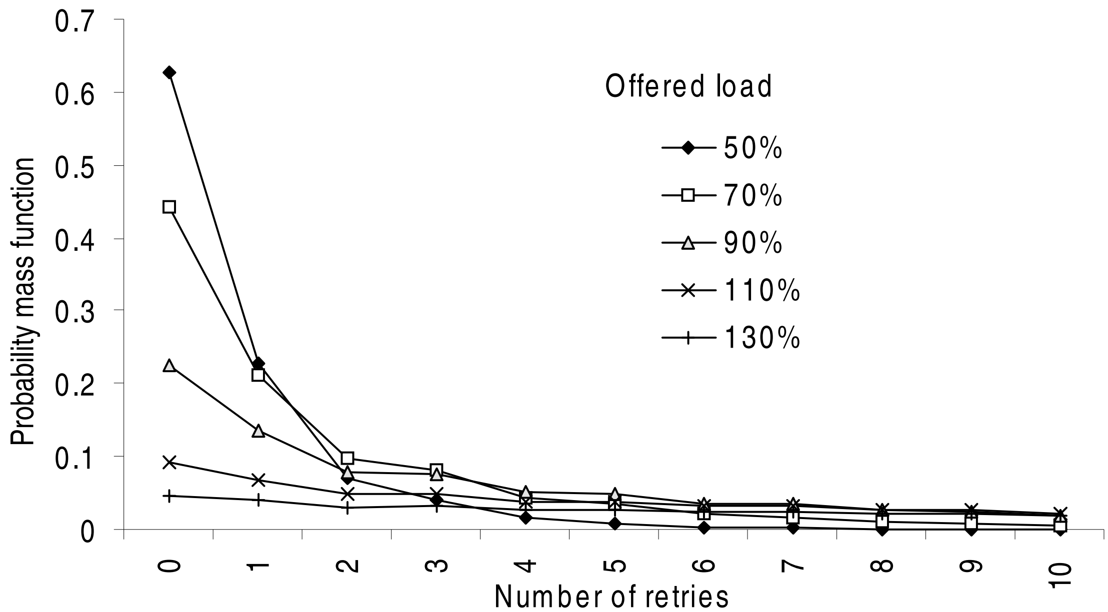

Figure 7: Probability mass function of number of channel allocation retries per connection
(KTHLR algorithm with no retry limit)

We propose a background channel allocation algorithm, which is aimed to decrease the number of retries. The next subsection presents this algorithm.

4.1.2 Background Channel Allocation Algorithms

In the Background Channel Allocation algorithm (**BCA algorithm**) [1] channel allocation is performed in the background, independently of set-up requests coming from

hosts. Its goal is to decrease (or eliminate) the need for slot allocation during call set-up. It can work parallel with any set-up-time algorithms.
In the algorithm nodes regularly exchange free channels with their direct neighbours. The goal of the exchange *in the case of homogeneous network load* is to distribute free channels *evenly* amongst the 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 the nodes. Simulation results show that the possibility of having no free channels, the blocking probability and the set-up time became lower when the algorithm was added to the set-up-time algorithms.
This idea can be extended to a real algorithm, which considers the case of normal operation when the load is different at each node. The difference between nodes is reflected in a priority value for buses in both directions, that is each node has a priority number for each bus, which depends on the traffic load to the given bus. Priorities can be constant or can change dynamically when adapting to the actual load of the network.
The exchange of free channels depends on the value of free channels and priorities. Based on the notions of Figure 6, node i initiates channel allocation to node $i+1$ if the expression

$$|(\text{free channels of node } i)*(\text{priority of node } i+1)-(\text{free channels of node } i+1)*(\text{priority of node } i)| \qquad (5)$$

can be decreased by channel allocation. The amount of channels to be transferred is determined so as to minimise (5) and considering that only free channels can be transferred. Node i asks channels from node $i+1$ if the first term of expression (5) is below the value of the second term and transfers channels 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 (5) 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 of it. For example if the priority of node $i+1$ is 0 for one of the buses then node $i+2$ is the partner of node i for the allocation of free channels on that bus.
The 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 channels of the downstream neighbouring node 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 as well. This type of priorities is not used in this examination.
The other solution is to assign static priorities to the nodes where the priorities are changed at the management level. In this case the basis of the priority assignment can be the role of the node in the network or the price paid by the customer of the node.
If the priority is based on the role of the node, we can assign higher priority to nodes connected to servers or to switching nodes, and lower priority to nodes connected to clients.
If the priorities are proportional to the charges paid by the customers then it is a better solution to rewrite expression (5) so as those priorities are compared to the number of *all the channels* owned by nodes.
In this case priority is related to the bandwidth that can be used by the connections of the node without reallocation during set-up. If the priority is high, many channels can be used without the additional delay of channel reallocation. If the slots of the node are used by connections, then channel allocation is required at every new connection set-up. This kind of priority use is appropriate for charged systems, because the customer who pays more can build up more connections without the delay of set-up-time channel allocation. There are significantly fewer channel allocations in this system compared to the one using the number of free channels for calculating the function.

From the above variants of the **BCA algorithm** we simulated the one where the comparison is *based on free channels* because our goal was to optimise the operation of the network. We used *static priorities* for the algorithm because there is no need for adaptation to the varying traffic load as we assumed static network model in Section 3.1. The priorities of the nodes were chosen to be fair. Precisely, the priority of a node to one direction of the bus was chosen to be proportional to the load sent to that bus. Priority of client nodes for the bus to the server equals to 1, the priority of the server on this bus is 0. On the other bus clients have 0 priority and the server's priority value is1.
BCA algorithm was applied in line with KTHLR algorithms in the case of various values of the channel allocation retry limit.
We proposed the BCA algorithm because we expected that the number of channel allocation retrials necessary to establish a connection decreased. As a consequence of this effect, we expected that the set-up times decreased in average. As another result, if the number of channel allocations were limited the blocking probability at nodes would also decrease.

Avr. Setup time (ms)	Channel Allocation Algorithms									
		Retry limit=2			Retry limit=5			Retry limit=99		
Offered load(%)	BCA	KTH	KTHLR	KTHLR +BCA	KTH	KTHLR	KTHLR +BCA	KTH	KTHLR	KTHLR +BCA
50	0.191	0.255	0.256	0.206	0.295	0.294	0.226	0.311	0.309	0.234
70	0.191	0.277	0.278	0.226	0.375	0.376	0.303	0.552	0.529	0.435
90	0.191	0.294	0.294	0.244	0.445	0.446	0.388	1.953	1.645	1.66
110	0.191	0.304	0.304	0.261	0.495	0.498	0.452	4.516	4.195	4.199

Table 1: Comparison of average set-up times in the case of different algorithms and retry limits

Blocking probability	Channel Allocation Algorithms									
		Retry limit=2			Retry limit=5			Retry limit=99		
Offered load (%)	BCA	KTH	KTHLR	KTHLR +BCA	KTH	KTHLR	KTHLR +BCA	KTH	KTHLR	KTHLR +BCA
50	0.068	0.051	0.051	0.023	0.006	0.005	0.003	0	0	0
70	0.159	0.129	0.126	0.089	0.045	0.045	0.034	0	0	0
90	0.249	0.219	0.219	0.174	0.127	0.121	0.113	0.010	0.009	0.010
110	0.333	0.301	0.298	0.267	0.219	0.216	0.203	0.103	0.106	0.106

Table 2: Comparison of blocking probabilities in the case of different algorithms and retry limits

In Table 1 and Table 2 we can see that the BCA algorithm improved both the set-up time and blocking probability parameters. The numbers in parenthesis after the names of algorithms show the allowed channel allocation retrials.

In Table 1 the absolute minimum of the set-up time is shown at the BCA algorithm. This algorithm does not use set-up-time channel allocation, this is why the time shown here is the connection set-up time without any channel allocation delay. The addition of BCA algorithm to KTHLR decreased the set-up time with about 10^{-4} seconds, which is in the order of one cycle time.

In Table 2 we can see the effect of BCA algorithm on blocking probability. We can conclude from the values that the proposed algorithm is the most effective one if channel allocation retry limit is low. In the case of 50% offered load and retry limit 2 the algorithm decreased the blocking probability from 5% to 2.3%, which means 2.7% improvement in the throughput. In the case of other offered loads with retry limit 2 the throughput increased with 2-4%. If the retry limit is increased, then the gain of BCA algorithm is decreased. In the case of retry limit 5 the gain is between 0.5% and 1.5%. If the retry limit is equal to the number of nodes (with previous terminology: there is no retry limit) the difference between KTHLR and KTHLR+BCA algorithms is not significant.

We have seen that though the BCA algorithm improved the performance of the system, the effect of limiting the number of channel allocation retrials has more significant impact on its behaviour. In contrast with BCA algorithm, which improved both set-up

time and blocking parameters, applying a retry limit has the opposite effect on these two characteristics. If lower retry limit is applied the blocking probability increases and the set-up time decreases. In order to find the compromise between the two most important performance characteristics of the system, we examined them by evaluating the KTHLR+BCA algorithm as the function of channel allocation retry limit.
Figure 8 shows how blocking probability is effected by the limit. Figure 9 displays how the set-up time depends on the limit.

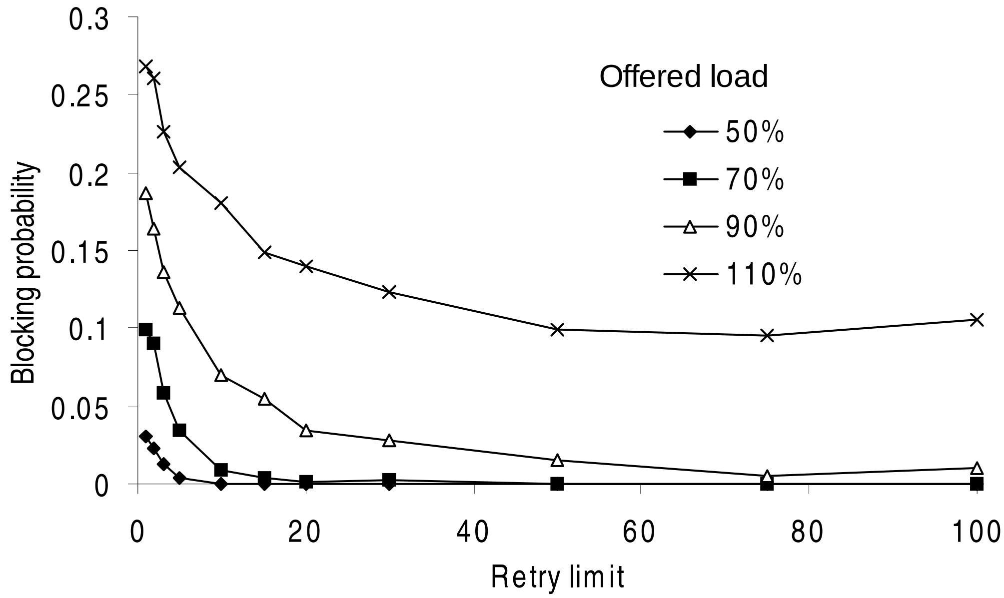

Figure 8: Effect of retry limit on blocking probability (KTHLR+BCA algorithm)

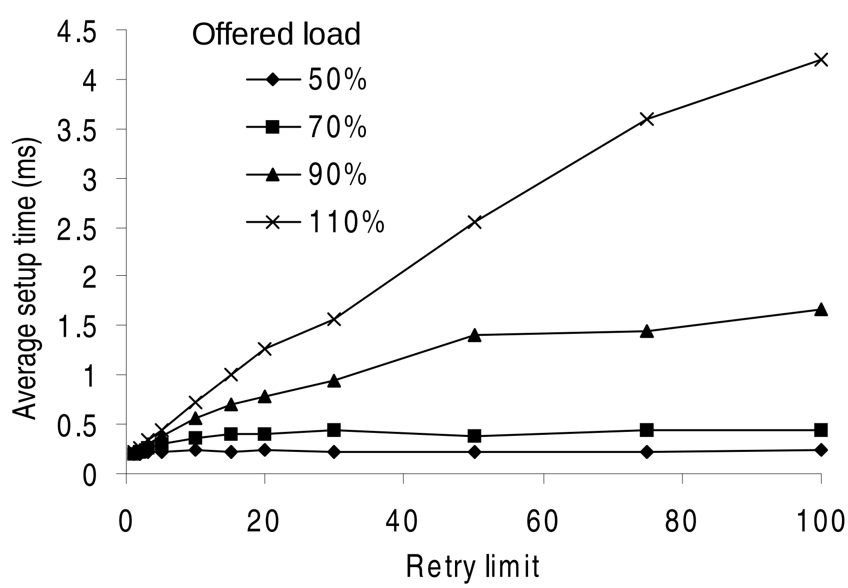

Figure 9: Effect of retry limit on average set-up time (KTHLR+BCA algorithm)

Blocking probability decreases almost exponentially if we increase the number of allowed channel allocation retrials. In the case of lower offered loads, the gradient of the blocking curves is bigger, in other words it increases faster. There is no blocking at the offered load level of 50% and 70% if the retry limit is higher than 10 and 30, respectively.

The shape of the set-up time vs. retry limit curve depends on the load of the system. At low offered load (50-70%) the limit has a minor effect on set-up time. At higher offered loads (110%) it is closely linear.

The optimal operation of the system depends on the specific requirements. If set-up time is more important than throughput, a lower retry limit can be chosen. If keeping blocking on a low level is the highest priority, then a higher retry limit can be applied. As blocking decreases more dramatically in a general case it is better to determine the retry limit based on the blocking function. The optimal limit is different for every load condition.

4.2 Algorithms Combined with Set-up-time Allocation Using Status Table

In the previous subsection we examined the behaviour of set-up-time channel allocation algorithm with and without additional background channel allocation and with different retry limits. The performance of KTH and KTHLR algorithms can be enhanced without background allocation if nodes maintain a status table [4] about the amount of free channels of other nodes. If status tables are up-to-date, nodes only try to get free channels from nodes that have them, therefore they can radically decrease the amount of unsuccessful set-up-time channel requests. The operation is slightly modified if status tables are used:

A node uses its status table if it wants to ask other nodes for channels. If a host requires a connection with M channels and the node has N free channels, where $N<M$, it sends requests asking for M-N channels. The node sends requests to a node with free channels. If this node does not have enough free channels according to the status table, then it sends a query also to another node with free slots, and so on. The node sends requests in the first round until the asked nodes have at least M-N free slots according to the local status table. After this, the node waits until all the replies arrive. If the necessary channels are not collected, a new round begins. The channel allocation is finished if the required channels are collected, or all the nodes which have free channels are asked for channels. If the required number of tokens is collected, the node sends a set-up request to the destination node. After the acknowledgement arrived from the destination, data transmission can start immediately.

The maintenance of the status tables is based on administration messages received in control channels.

A node sends a broadcast administration message if the number of its free channels has changed. Administration messages have low priority at receiver nodes compared to other (e.g.: set-up, release) messages. If the processing capacity of a node is overloaded it drops incoming administration messages. If the outgoing control capacity is overloaded outgoing administration messages are dropped (see the rules in section 3.2). That is, in normal condition nodes have up-to-date status tables, but if control or processing capacity is overloaded, status tables become outdated.

First let us see the comparison of algorithms with and without status tables, when the control capacity and the processing capacity of the nodes are not overloaded. Figure 10 and Figure 11 show the comparison of seven algorithms. KTH, KTHLR and KTHLR+BCA algorithms are compared when

- no status table is used and the retry limit is 10 (notion : 10)
- status table is used and the retry limit is 100 (notion : s)

For reference purposes, we displayed the BCA algorithm without any set-up-time method as well.

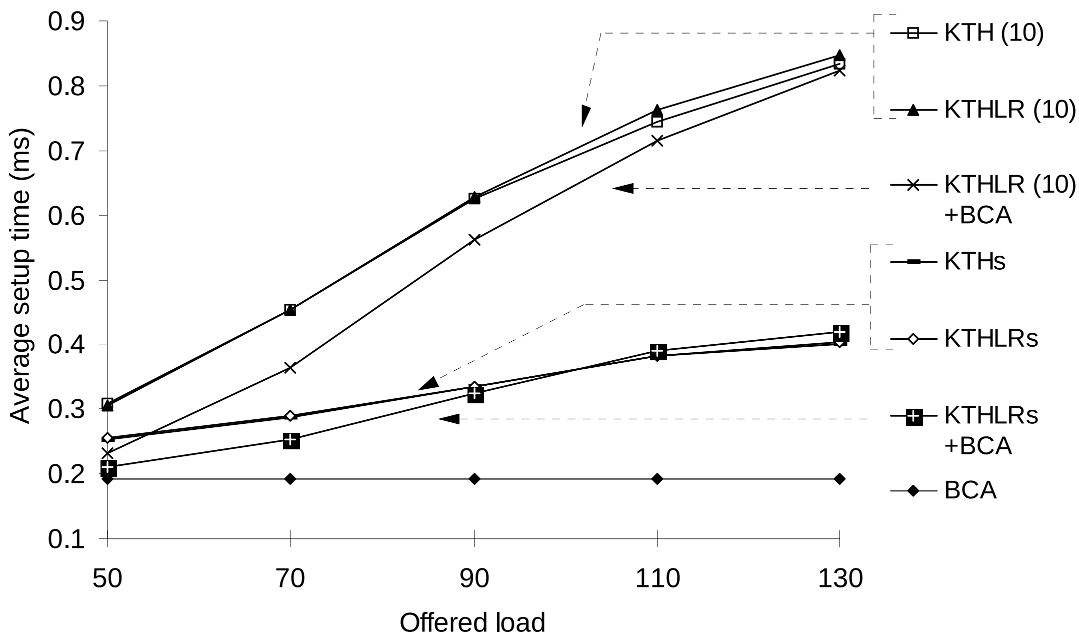

Figure 10: Comparison of average connection set-up time of algorithms with and without status table

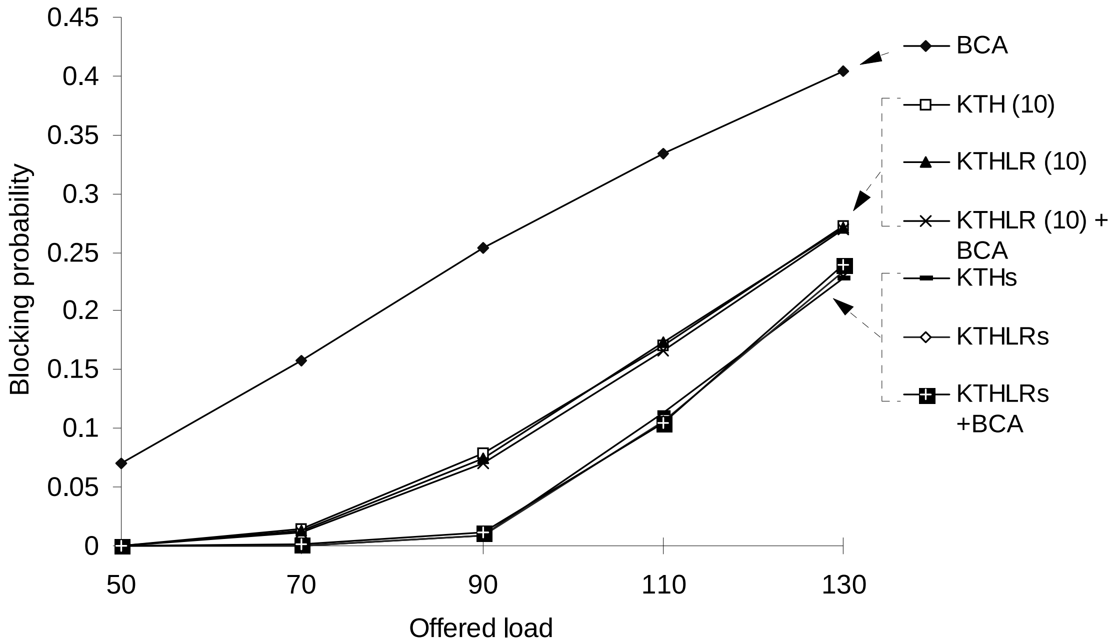

Figure 11: Comparison of blocking probability in case of algorithms with and without status table

Both figures show that algorithms with status tables perform better than their counterparts.
Figure 10 also demonstrates the power of adding BCA algorithm to set-up-time methods. In the case of 50% offered load KTHLR+BCA algorithm has even lower average set-up time than KTHs and KTHLRs algorithms. Though it is a small difference, it is due to the increased probability of having free channels, when BCA is applied.
In the case of high offered loads applying BCA is not effective, because it increases the set-up time. This is because the number of nodes is higher than the number of free channels, so the distribution of free channels does not improve the performance.
Figure 10 also shows that if the offered load increases, the difference between the set-up time of algorithm with and without status table increases.
In Figure 11 we can see that the effect of BCA algorithm to the blocking probabilities is not significant. The omission of status tables moreover increases the blocking probability. The importance of status tables is most obvious around 100% offered load. At lower and higher loads the difference is smaller. The largest difference is 6.5% in the simulated example.

In the above comparison the control and processing capacity available for nodes were not overloaded. If the network load consists of the frequent transfer of short files, then the increased control load will effect the performance characteristics. Table 3 and Table 4 show this effect.

Blocking probability	Channel Allocation Algorithms						
	Retry limit=100			Retry limit=10			
File size	KTHs	KTHLRs	KTHLRs+BCA	KTH	KTHLR	KTHLR + BCA	BCA
2 MB	0.010	0.009	0.010	0.077	0.075	0.070	0.249
512 kB	0.008	0.009	0.012	0.078	0.075	0.070	0.254
64 kB	0.012	0.008	0.014	0.079	0.070	0.073	0.253
8 kB	0.030	0.025	0.020	0.078	0.075	0.067	0.264
1 kB	0.133	0.133	0.180	0.106	0.010	0.128	0.330

Table 3: Effect of overloaded control capacity on blocking probability

Avr. setup time (ms)	Channel Allocation Algorithms						
	Retry limit=100			Retry limit=10			
File size	KTHs	KTHLRs	KTHLRs+BCA	KTH	KTHLR	KTHLR + BCA	BCA
2 MB	0.333	0.332	0.318	0.624	0.629	0.562	0.191
512 kB	0.335	0.335	0.325	0.626	0.627	0.562	0.191
64 kB	0.364	0.358	0.380	0.628	0.623	0.574	0.192
8 kB	3.572	3.773	2.931	0.842	0.861	0.831	0.272
1 kB	2.255	2.359	3.35	1.928	1.927	2.112	0.471

Table 4: Effect of overloaded control capacity on average connection set-up time

We can see in Table 3 that the blocking of algorithms with status table is more dependent on control capacity. In the case of 1kB file sizes, blocking of these algorithms becomes higher than blocking of those without tables. This is due to various facts:
1.Calls are blocked if the control capacity become overloaded (see in the node model)
2.Administration messages are dropped if the control capacity is overloaded (see in the node model). This results in status tables with incorrect data. If a table shows that a node has no free channels (even if it has), the owner node of this status table will not ask for channels. It means that only a fraction of the nodes will be contacted for channels.
3.If messages are short the percentage of channels that are being transferred (nor free or occupied) become higher.
All the above mean that the more control messages are used by an algorithm the more the blocking probability is. This is true for BCA algorithm too. Adding BCA to other algorithms increases the blocking probability.
In Table 4 we can see the same effect, namely that in the case of short but frequent file transfers, algorithms without status tables perform better. A strange effect can be shown on this figure: the set-up time for KTHs and KTHLRs algorithms in case the average message size is 1kB is lower than in the case of 8kB. It can be explained by the huge difference between the blocking probabilities (and throughputs). In other words it justifies that we used a right node model, which can cope with increased control capacity.

5. Conclusions

Low connection set-up time (burst set-up time) and small blocking probability are important requirements if burst switching is applied in the area of fast circuit switching networks. Fairness is basic criteria for all communication networks. This article proposed two channel allocation techniques for DTM networks that fulfil these needs. When the developed algorithms were applied, the performance of the examined network was improved and fairness was provided.

The benefits of background channel allocation were most obvious when the network operated in normal conditions (control and data capacity were not overloaded). When data capacity was overloaded, the improvement of background channel allocation was lower. When control capacity was too high the BCA algorithm degraded the performance of the network. In order to avoid these overloaded situations, network dimensioning can be done based on the results presented here.

The effect of status tables, overloaded control capacity and limit on the number of set-up-time channel allocation retrials on combination of channel allocation algorithms (KTH, KTHLR and BCA) was investigated. It was shown that the performance of set-up-time algorithms was significantly increased when status tables were applied. When control channels became overloaded, algorithms with less overhead outperformed the sophisticated ones (KTHs, KTHLRs, with and without BCA). Decreasing the limit on the number of channel allocation retrials, the set-up time was decreased and the blocking probability was increased. The optimum value of the retry limit can be dimensioned based on the simulation results.

Main directions of future work are an extensive fairness study. The study should include the examination of general traffic load along the bus, the investigation of the optimal priority values for BCA algorithm, and the possibilities of using dynamic priorities for changing traffic load. To study the benefits of applying slot reuse in the DTM operation is another direction of our future research.

6. Acknowledgements

The work is a part of the "Performance Evaluation of Broadband Multimedia Networks" research project of the High Speed Network Laboratory (HSNLab), Department of Telecommunications and Telematics, Technical University of Budapest. The authors are grateful to the colleagues at HSNLab for their support.

7. References

[1] Cs. Antal, S. Molnár, L. Györfi, G. Szabó, “Resource Reallocation in a Communications Network”, patent application, May 1997, Sweden

[2] Cs. Antal, “Servicing Bursty Sources Efficiently via a Fast Circuit Switched System”, International Conference for Computer Communication '97, 19- 21 November 1997, Cannes, France

[3] S.R. Amstutz, “Burst Switching-An Update”, IEEE Communications Magazine, Vol. 27, No. 9, Sept. 1989.

[4] 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

[5] C. Bohm, P. Lindgren, L. Ramfelt P. Sjödin, “Resource Reservation in DTM”, 1st IEEE Symposium on Global Data Networking, Cairo, Dec. 1993

[6] I. Chlamtac, A. Faragó, H. Zhang, "A Fundamental Relationship Between Fairness and Optimum Throughput in TDMA Protocols", IEEE International Conf. on Universal Personal Comm., Cambridge, MA, Sep. 1996, pp. 671-675.

[7] M. E. Crovella, A. Bestavros, “Self-Similarity in the World Wide Web Traffic: Evidence and Possible Causes”, Proc. SIGMETRICS ’96, Philadelphia, May 23-26, 1996

[8] S. Deng, "Empirical model of WWW document arrivals at access link", IEEE International Conference on Communications, p. 1797-802 vol.3, ICC'96, 23-27 June 1996, Dallas, TX, USA

[9] R. Jain, S. A. Routhier, “Packet Trains-Measurement and a New Model for Computer Network Traffic”, IEEE Journal on Selected Areas in Communications, Vol. 4, No. 6, February 1986

[10] I. Katzela, M. Naghshineh, "Channel Assignment Schemes for Cellular Mobile Telecommunication Systems", IEEE Personal Communications, June 1996, p. 10-31

[11] Kumar, L.N.; Douligeris, C. "Demand and service matching at heavy loads: a dynamic bandwidth control mechanism for DQDB MANs", IEEE Transactions on Communications, Vol: 44 Iss: 11 p. 1485-95

[12] C. C. Lam, "An efficient distributed channel allocation algorithm based on dynamic channel boundaries", Proceedings. 1996 International Conference on Network Protocols, p. 236-43, Columbus, OH, USA

[13] D. E. McDysan, D. L. Spohn, “ATM, Theory and Application”, McGraw-Hill, 1994.

[14] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transaction on Networking, vol. 3, no. 3, pp. 226-244, 1995.

[15] M. de Prycker, "Asynchronous Transfer Mode, Solution for Broadband ISDN", Ellis Horwood, 1991, Chichester