Twin-bus-controller protocol for fibre optic networks

Ron Yu, John J Metzner* and Asok Ray[†] propose a Data Link Layer protocol for fibre optic networks with unidirectional bus topologies

A new Data Link Layer protocol, named the Twin-Bus-Controller (TBC) protocol, is proposed for a fibre optic network with unidirectional bus topology. The TBC protocol operates on a contention-based, time-division multiplexing scheme, and is managed by two centralized bus controllers. These controllers, which also function as network managers cooperate with each other to control and coordinate the activities on the twin bus. The TBC protocol has the capability to perform at a very high network utilization, and uses simple hardware at all stations except the two bus controllers. This arrangement provides a relatively inexpensive means to accommodate a large number of stations. Heterogeneous data consisting of real-time sensor and control signals, voice and video data, and non-real-time data such as those due to accounting and administration, can be simultaneously handled by the TBC protocol. The TBC protocol maintains global queues for all different types of data, and each class of data has a bounded delay. Furthermore, any new type of data can be added easily to the network without shutting it down or affecting those stations that are unrelated to the new data. A finite-state-machine model has been used to describe the TBC protocol. Performance of the TBC protocol has been evaluated by statistical analysis as well as via simulation for multiple classes of data traffic. Performance of the TBC protocol has been compared with that of Buzznet and Fasnet. The TBC protocol can be directly applied to diverse computer communication systems, e.g. office, manufacturing, and banking environments.

Keywords: protocols, fibre optic network, real-time, performance analysis

Sprint International Communications Corporation, Mail Stop 0P124D, 12490 Sunrise Valley Drive, Reston, VA 22096, USA
*Electrical Engineering Department, †Mechanical Engineering Department, The Pennsylvania State University, University Park, PA 16802, USA
Paper received: 10 October 1990; revised paper received: 30 March 1991

A fibre optic-based network should be able to accommodate a large number of stations having heterogeneous traffic consisting of both real-time and non-real-time data. Real-time data includes voice, video and sensor/control signals. Control and sensor data transmission under scheduled deadlines are extremely important in factory automation for computer-integrated manufacturing (CIM)[1, 2], but is rarely mentioned in most of the reported fibre optic network protocols[3]. Normally, controller-sensor data require much stricter delay bounds than those for voice and video communications, and violation of these bounds may have serious impact on the controlled process resulting in potential instability. Voice communications can tolerate typically up to 300 ms delay; whereas, the control-sensor data for a manufacturing robot may have to be periodically updated with a sampling time in the order of tens of milliseconds. It is an important and critical issue for the communication network to meet the deadlines for processes that are scheduled in real-time. In addition to serving subscribers with heterogeneous data, an integrated network must also accommodate a large number of stations. Fibre optics provides high capacity, high data transmission rate, immunity to noise corruption, and other attractive features[4]. Therefore, a fibre optic network can serve a local area, and is capable of being expanded to be a metropolitan area network.

The proposed protocols for fibre optic networks[5-14] have their own merit and demerit. For example, Expressnet[5, 6] suffers from the limitation to integrate a large number of stations because of its folded topology, and the existence of only one controller limits the failure detection and recovery capabilities. Fasnet[7, 8] uses a single locomotive generator to control the 'fixed timed slot'; problems arise due to dependence on a single controller and the use of a fixed data slot. Furthermore, the data travels around the network in Fasnet and can be removed by the locomotive station which potentially reduces the network utilization. Distributed Queue Dual Bus (DQDB)[9] uses a fixed size slot in data transmission,

0140-3664/91/010598-10 © 1991 Butterworth-Heinemann Ltd

and the data is divided into short segments and desegmented at the destination station. In other words, data is transmitted in a frame relay manner. Although this approach is convenient for providing reserved connection-oriented slots, it has disadvantages for bursty traffic with duration of multiple slots. To provide fairness to all subscribers under distributed control, it is necessary to have rather complex request-counting procedures which may require extra hardware and/or software at each station. Furthermore, a station may have to issue many requests and have its message segmented into small packets that would arrive at the destination at irregular intervals. Consequently, the percent overhead becomes higher than necessary, and the efficiency and reliability of acknowledgement schemes are degraded. Another popular protocol, Fibre Distributed Data Interface (FDDI)[10, 11] is basically a token ring in which the token rotation time monotonically increases with the number of integrated stations, and thus becomes a source of unavoidable delay for real-time scheduling. Furthermore, hardware complexity (e.g. construction of token rotation times) is another problem for FDDI to handle heterogeneous traffic in a single communication channel. Buzznet[12] is a hybrid token/random access scheme that performs a random access mode during a light load status and switches to controlled access mode (virtual token mode) whenever collision occurs. If the traffic is bursty, the stations with higher priority jam both buses whenever they have data to transmit, and the jamming will force both buses to switch to the controlled access mode. Once the switching between the random access and controlled access modes become frequent, the performance of the network significantly degrades. Moreover, network utilization is inherently low in the Buzznet protocol because each data is transmitted simultaneously in both buses (this problem has been addressed by Ayyagari and Ray[1], but no specific solution is provided). If there are multiple classes of data, then the jamming sequences or patterns become a complex issue for Buzznet. In general, complexity of the protocol operations may restrict Buzznet's productivity and capability.

To integrate a large number of stations and heterogeneous traffic, the protocol should be easily implemented. Furthermore, the network management must be as simple as practical, and the induced delay should be bounded such that real-time data can be accommodated. Once these requirements are satisfied, a fibre optic network protocol can be applied to any environment that includes real-time communications. Based on the above discussion, the proposed Twin-Bus-Controller (TBC) protocol is designed and evaluated. Performance of the TBC protocol has been compared with that of Fasnet and Buzznet by statistical analysis and discrete-event simulation.

TBC PROTOCOL

The TBC protocol uses a dual-unidirectional bus topology, as illustrated in Figure 1. Two active end stations serve as

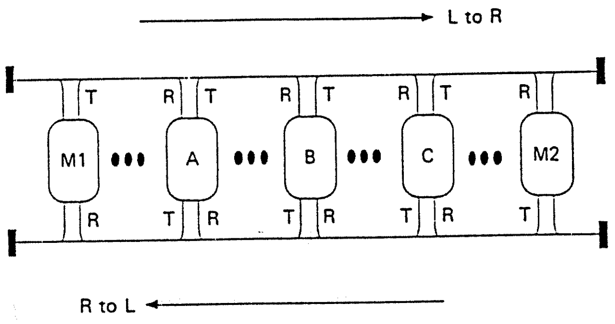

Figure 1. TBC protocol topology. M1: bus controller (left); M2: bus controller (right); A, B, C: remote terminals served by the network

bus controllers, and also function as network managers. Each bus controller controls the activity on one bus and monitors the activity on the other bus.

Functionally, any station with waiting data sends a request to the appropriate bus controller for permission to transmit. For example, station A in Figure 1 wishes to send a message to another station B, which is on its right. The request occurs in two steps: a short signal, known as the preemptive signal, is first sent by station A to the right bus controller M2 which, in turn, initiates a request frame; when this request frame is received by station A, it puts a request to the left bus controller M1. Upon acceptance of this request, M1 schedules station A to transmit the waiting message. The rationale for the request frame in the first step is to allow centralized management of bus operations by the two controllers M1 and M2. This is especially important for efficiency and fairness to all network subscribers. Since there is no reservation for sending the request and the message frame, the complexity of the bus controller operations is reduced, and any possible wastage of bandwidth is avoided because there is no reservation for the data that has not yet arrived.

In the TBC protocol, each station may interface with the network via a passive coupler such that the network operates in the bus topology, i.e. the transmitted message will not be interrupted and dropped off at the end station which is a bus controller. However, a repeater, or a regenerator, will eventually be required for strengthening the optical signal as the length or size of the network increases.

Two different frames are designed for the TBC protocol: request frame, and message frame.

Request frame and format

The request frame (as shown in Figure 2) is designed for a station to send a request to the bus controller. It is a relatively small message which allows the bus controllers to schedule message transmissions by the individual stations. The request is sent opposite to the direction of the later data transmission. The request frame is of constant length and has the following four different fields.

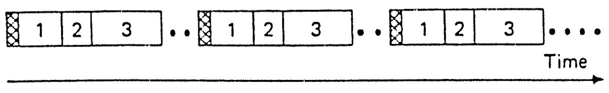

Figure 2. Request frame format. ⊠: Preemption (to bus controller); 1: broadcast field (from bus controller); 2: contention field; 3: request field, a fixed length

- preemptive field;
- broadcast field;
- contention field;
- request field.

The request field, in turn, consists of the following four subfields, as shown in Figure 3:

- source address field;
- destination address field;
- data request field;
- status.

A request frame starts with the preemptive field for station(s) to send a preemptive signal. Figure 4 illustrates the details of the request frame timing. To explain how the request frame operates, the preemptive field is ignored for the time being. The bus controller sends a request frame identifier pattern in the broadcast field. Let t_1 be the time to detect that this is a request frame, and t_2 be the allocated time for contention, i.e. the time to detect the presence of any signal on the bus. A station wishing to transmit any information sends a short initial burst during the interval t_2. It is likely that more than one station will start sending the burst. However, each station, while transmitting its own message, can listen to the incoming signal from other station(s). Due to the uni-directional nature of the transmission only the most upstream of all stations that are transmitting at that time will not hear any reception. In the example shown in Figure 4, station B will not hear the transmission of other stations whereas any other stations (such as C), located downstream of B will abort their transmissions as soon as they detect an upstream transmission. Thus there will be no conflict in the request information field, which will be captured by the most upstream requestor. It is important to note that t_1 and t_2 are not related to the propagation time or the number of stations; t_1 depends only on how long it takes to recognize the request identifier, and t_2 is essentially based on the time it takes to recognize the presence of an initial signal burst, probably on the order of only a few bit lengths.

The request field in this frame is of fixed length, and contains several identical subfields which are used to specify the message priority and length. In other words,

S.A.	D.A.	#1	#3	#4	....	Status

Figure 3. Request field structure. S.A.: source address; D.A.: destination address (= bus controller); #1, #3, #4: number of packets for priorities 1, 3, 4.; Status: for special purpose

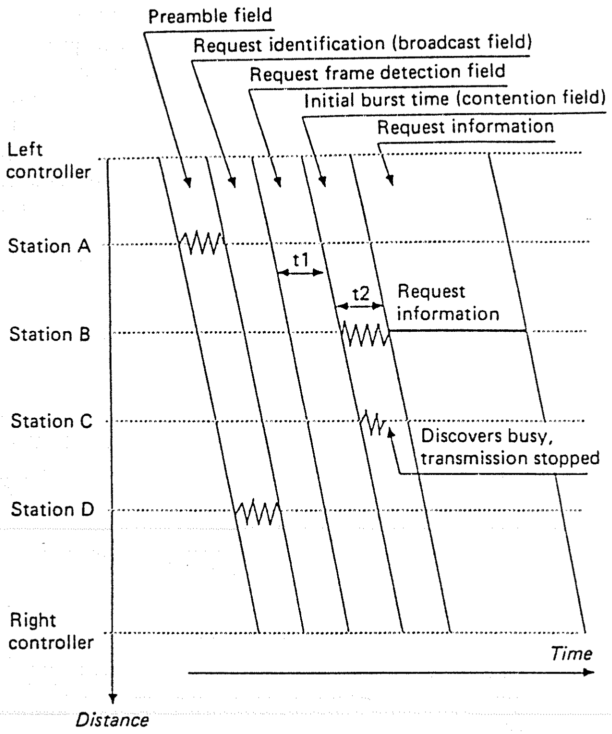
Preamble field
Request identification (broadcast field)
Request frame detection field
Initial burst time (contention field)
Request information
Left controller
Station A
t1
t2
Request information
Station B
Station C
Discovers busy, transmission stopped
Station D
Right controller
Time
Distance

Figure 4. Request frame timing details

requests for multiple classes of data can be grouped together in a single request frame. The preemption is included in both request and data frames. It is a short notice (e.g. about the same length as t_2) that any arbitrary (i.e. unidentified) station needs to transmit in that direction, and therefore must send a request frame to the bus controller in the other direction. It does not matter in this case whether there is a conflict. The receiving bus controller simply interprets it as a signal to issue one or more request frames. In the example of Figure 4, station A wishes to transmit in the L to R direction, and therefore it requires a request frame from the right bus controller in the R to L direction.

Message frame and format

The message frame, as shown in Figure 5, is used to transmit data, and is of variable length. The frame length is assigned by the bus controller according to the information specified in the corresponding request frame. Each message frame can only convey one class of data, and the transmission of each message frame is coordinated and controlled by one of the two bus controllers. There are four different fields in a message frame:

- preemptive field;
- broadcast field;
- synchronous field;
- data field.

The preemptive field and broadcast field are similar to those defined in the request frame, except the content of the broadcast is a message frame not a request frame. Furthermore, the broadcast will specify the sender's identity, the type and length of the transmitted data, as illustrated in Figure 5. Right after the detection of a broadcast signal, the bus controller issues a special message as a synchronous signal to activate the message transmission. Upon receipt of the synchronous signal, the sender starts transmitting the message.

Bus controller functions and network management

As illustrated in Figure 1, there are two bus controllers (BC) located at two ends of this dual-bus system. Each BC has all functional features of a remote terminal (RT), which is essentially any station that does not serve as a BC. The additional features of a BC are to control and monitor the activity of the buses and perform network management functions. On the receiving side, a BC has to be aware of the messages (i.e. preemptive signal), the request frame and message frame, and it coordinates the issue of request frames and message frames on the transmitting side. Actually, a single computer can serve as a bus controller if the network is configured as a bidirectional loop.

Now we consider the control and coordination of the request frames and message frames. Upon receipt of a preemptive signal, a BC (say the left one) schedules the next outgoing frame to be a request frame. Since the exact number of requests is not known to the BC, one procedure would be to issue a sequence of request frames. The other BC (the right BC in this case) recognizes an unused request frame it receives. Since the pair of bus controllers has complete control over the events occurring on the bus, the right BC can immediately tell the left BC to stop transmitting request frames. This requires sending one additional request frame. An alternative method would be to send only one request frame, and rely on receiving additional preemptive signals if there are more requests. This prevents wastage of bus bandwidth due to unnecessary transmission of request frames at the expense of increased delay whenever there is more than one simultaneous request. The bus controller's strategy of request frame transmission could also be made traffic-dependent. For example, if the left-to-right line is lightly loaded, the left BC could send a sequence of request frames; on the other hand, if this line is heavily loaded, then it could send just one request frame.

The BC has complete control over the instant each terminal may transmit and the duration of its transmission. It can maintain a memory of outstanding requests of all active terminals, the time of arrival of the requests and their priority and delay constraints. Therefore, an environment can be created to exercise intelligent network management for performance optimization. This is especially important when the network serves a large number of stations.

The BC can also handle initial entry of a station based on the current network load and the requestor's signal class based on the priorities of the messages. For example, a new request involving transmission of real-time messages (e.g. video, voice or control-sensor data) could be refused if this additional traffic is expected to exceed the constraints of average channel capacity. This protective measure would ensure maintenance of high quality of service. However, non-real-time data with much more relaxed delay constraints could be admitted more freely, subject primarily to overall queuing constraints.

Adding and deleting terminals does not interfere with the system operations. An entering terminal could send its address initially in both directions to announce its presence to all other terminals and the BCs. This could reveal its direction to all the other terminals, although not all the terminals may have the capability to record this information. The BCs could record the presence of the new station to maintain a list of all connected stations, but its location cannot be identified from this information. Another consideration when a terminal needs to start transmitting it that it may not know the direction of the destination terminal. This could be determined by sending an initial message in both directions. Once a response is received, both terminals know their relative position on the bus which suffice to determine the direction of transmission.

Fault management is a critical function for network operations. Each of the two BCs should be made fault-tolerant to ensure reliable and uninterrupted operations of the network. This pair is in a good position to monitor and manage failures of the remaining stations in the network. Moreover, each BC can also detect the failure of the other. In fault management, any missing message or error in the message transmission can be detected by the BC on the receiving end of that line, as it will see the allocated address without any use of the allocated time slot. The BC can inform the terminal via the other line that the terminal's transmitter is not responding, and this information is also received by the other BC. A station could similarly inform the BCs of its disconnection; if it does not, the fact could be discovered by failure to respond to its next reception, possibly followed by test signals from the BC to confirm the disconnection.

Finite state machine model of TBC protocol

A finite-state-machine model has been generated to describe the functions of the TBC protocol, as shown in Figure 6. The model is formulated on the basis of the following two assumptions for each RT:

- each RT has separate buffers to store different classes of data. Also, an RT has the capability to know the individual number of messages in each of the buffers;
- each RT has the capability to check the destination address for the incoming messages and place them in the appropriate buffer either on the upper bus or on the lower bus (or on both if the message is to be broadcast).

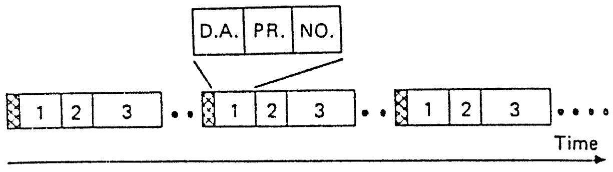

Figure 5. Message frame format. ▨: preemption (to bus controller); 1: broadcast field (from bus controller) — D.A.: destination address (requestor); PR.: message priority (one type of data only); NO.: number of packets; 2: reset (syn.) field (from bus controller); 3: data field, a variable length

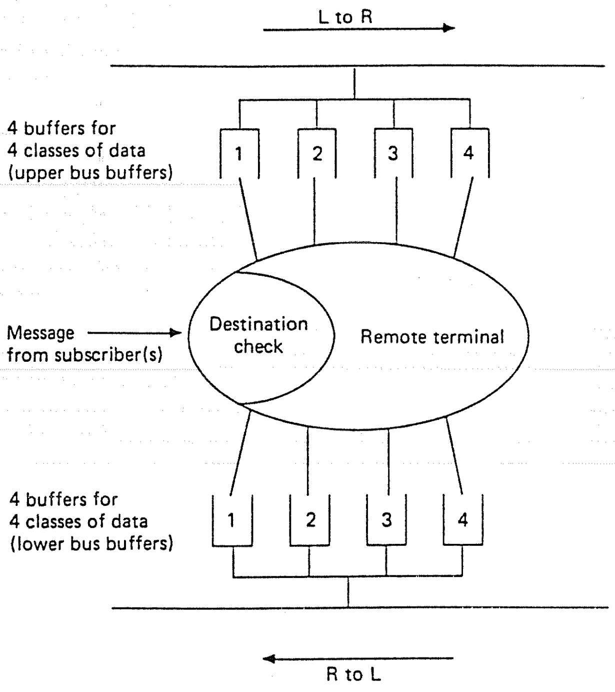

Figure 6. Remote terminal internal structure

It is assumed that the information regarding which bus (e.g. lower or upper) will be used by an RT for a particular message is already known, or can be obtained from the network manager, i.e. the bus controller. To find out the right location of the destination address for each message, each RT can either acquire the information from a BC or wait for the acknowledgement from the destination station. Only BCs which also serve as network managers need to know the locations of all RTs. The finite-state-machine model presented in Figure 6 is defined for a RT with a specific class of data, and its buffer size is limited to one on each side of the bus. Six states, as shown in Figure 7, exist in an RT:

0. Idle: the state that the buffer is empty and is capable of receiving any incoming message;
1. Backlogged: the state that the message arrives at the buffer;
2. Preemption: the state that the RT sends a bit-string as the preemptive signal;

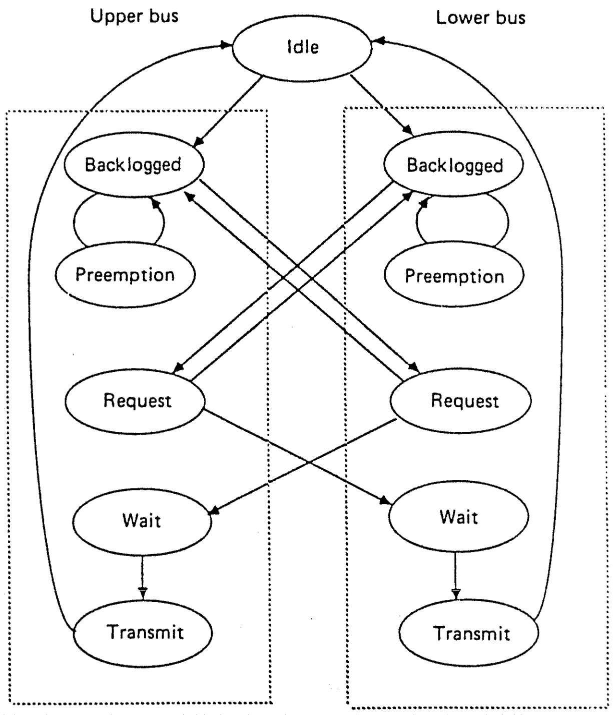

Figure 7. Finite state machine model of a remote terminal for a specific class of data

3. Request: the state that the RT attempts to capture a free request frame;
4. Wait: the state that the RT waits for the permission to transmit its data;
5. Transmit: the state that the RT is transmitting its data.

It is to be noted that a preemption is always successfully transmitted while a request may be delayed, depending on when the RT wins the contention. If the RT is competing for a free request frame, the preemptive signal may not be transmitted.

PERFORMANCE ANALYSIS

The delay induced by the TBC protocol can be attributed to four sources:

1. The delay δ_1 in having the opportunity to insert a preemptive signal to inform the bus controller.
2. Delay δ_2 to put in a request frame.
3. The Waiting time δ_3 for the permission of the bus controller to transmit.
4. The delay δ_4 due to transmit the message.

Both δ_1 and δ_4 range between 0 and l_{max}, where l_{max} is the length (in bit time) of the largest message. δ_2 ranges between 0 and $(l_{max} + (N - 1)l_{req})$, where N is the total number of stations to send a request and l_{req} is the length (in bit time) of a request frame. δ_3 is a random variable whose statistics depend on several factors of network traffic, such as average arrival rates of messages at different priority levels, and number of stations.

Assuming a data packet arrives at a particular station at time τ, then the actual arrival of the request to the bus controller is $(\tau + \delta_1 + \delta_2 + l_{req})$. Since $(\delta_1 + \delta_2 + l_{req})$ may have general probability distributions, the actual model for the network is essentially a G/G/1 with a priority queue. This model is very difficult to solve analytically. Assuming the Poisson distribution of the arriving message with an interarrival rate λ at every station, if the delay $(\delta_1 + \delta_2 + l_{req})$ is sufficiently small compared to the interarrival time $(1/\lambda)$, then request arrivals at the bus controller can be assumed to have the same distribution as the message arrivals at the station in which M/G/1 model can be applied. Obviously, certain errors will be introduced in the analysis by this approximation of an M/G/1 model.

M/G/1 head-of-the-line model

In the M/G/1 model, all message arrivals destined for one of the two directions is lumped together as if they were generated at the same source. Although an upper stream station on the bus carrying requests has an earlier opportunity to send its requests, this is not relevant to computing average delay over all packets of a given class, which is independent of the location of the originating station. In the M/G/1 model, the general equation for the expected waiting time of the data with priority level j for a total k different priorities is[15, 16]:

$$E\{W_j\} = \frac{\sum_{i=1}^{k} \lambda_i E\{S_i^2\}}{2 \times \left[1 - \sum_{i=1}^{j-1} \rho_i\right] \times \left[1 - \sum_{i=1}^{j} \rho_i\right]}$$

where λ is the average message arrival rate, S is the service time, ρ is the intensity (i.e. the product of the average arrival rate and average service time), W is the waiting time, and the subscript i corresponds to the i^{th} priority level.

Assuming the worst case when each station suffers a maximum delay (i.e. the maximum message length) to send both the preemptive signal delay δ_1 and the request delay δ_2 to the bus controller, the overall delay can be considered to be the upper bound. Similarly, the lower bound of the delay is obtained by assuming that the message suffers a minimum delay. Independent of the magnitude of the delay, the process of data arrival at the bus controllers has the same distribution as it has at the station because of the constant delay. The bounded delay for the data with priority i and length l_i can be expressed as:

- Upper bound of the average delay = $E\{W_j\} + 2 \times l_{max} + l_{req} + l_i$;
- Lower bound of the average delay = $E\{W_j\} + l_{req} + l_i$.

TBC PROTOCOL SIMULATION

The computer simulation for the TBC protocol is performed in the manner of discrete-time-event simulation. The first step is to derive a timed Petri-net model[17] for studying the concurrence of different events in the protocol operations. The discrete-time-event simulation model follows the event interactions of the Petri-net model.

The simulation model consists of two submodels: message generation, and protocol operation. The message generation submodel is responsible for generating the new messages and the associated priority levels, and the protocol operation submodel handles the activities within the network. For simplicity and space limitations, the discrete-event simulation is directly described without discussing the Petri-net model. The simulation model is presented by an event interaction flow chart, as shown in Figure 8, in which several events are defined in accordance with the protocol operation (e.g. broadcast, preemption, contention, message transmission). No gross simplifying assumption is made in the simulation model.

The simulation model starts from the message generation and overall initialization. A new arrival message interacts with the network and activates the protocol operation. To simulate a large number of stations with heterogeneous traffic, a 100 Mbit/s optical fibre communication is assumed with the following four classes of data on it. The message arrival process for each class of data is assumed to be Poisson:

- Sensor/controller data: average interarrival time = 10 ms with message length of 300 bits;
- Voice data: average interarrival time = 100 ms with message length of 6400 bits (64 kbit/s).
- Video data: average interarrival time is 33 ms with message length of 66667 bits (2 Mbit/s).
- Non-real-time data: average interarrival time is 50 ms with message length of 75000 bits.
- The number of active stations on the bus ranges from 10 to 80.
- Each station is capable of generating all four classes of data as itemized above.

The first three types of data are considered as real-time data and the number, shown above, stands for the priority level for that data. Throughout the simulation, real-time data is assumed to occupy 25% of the channel capacity, and the remaining bandwidth is used by non-real-time data. The results of simulation are presented below.

RESULTS AND DISCUSSION

The results of simulation and analysis are presented in Figures 9–12. Some differences exist between the simulation and the analytical results. With references to Figures 9–12, the reasons for these difference are explained below:

- The model in simulation is G/G/1 head-of-the-line

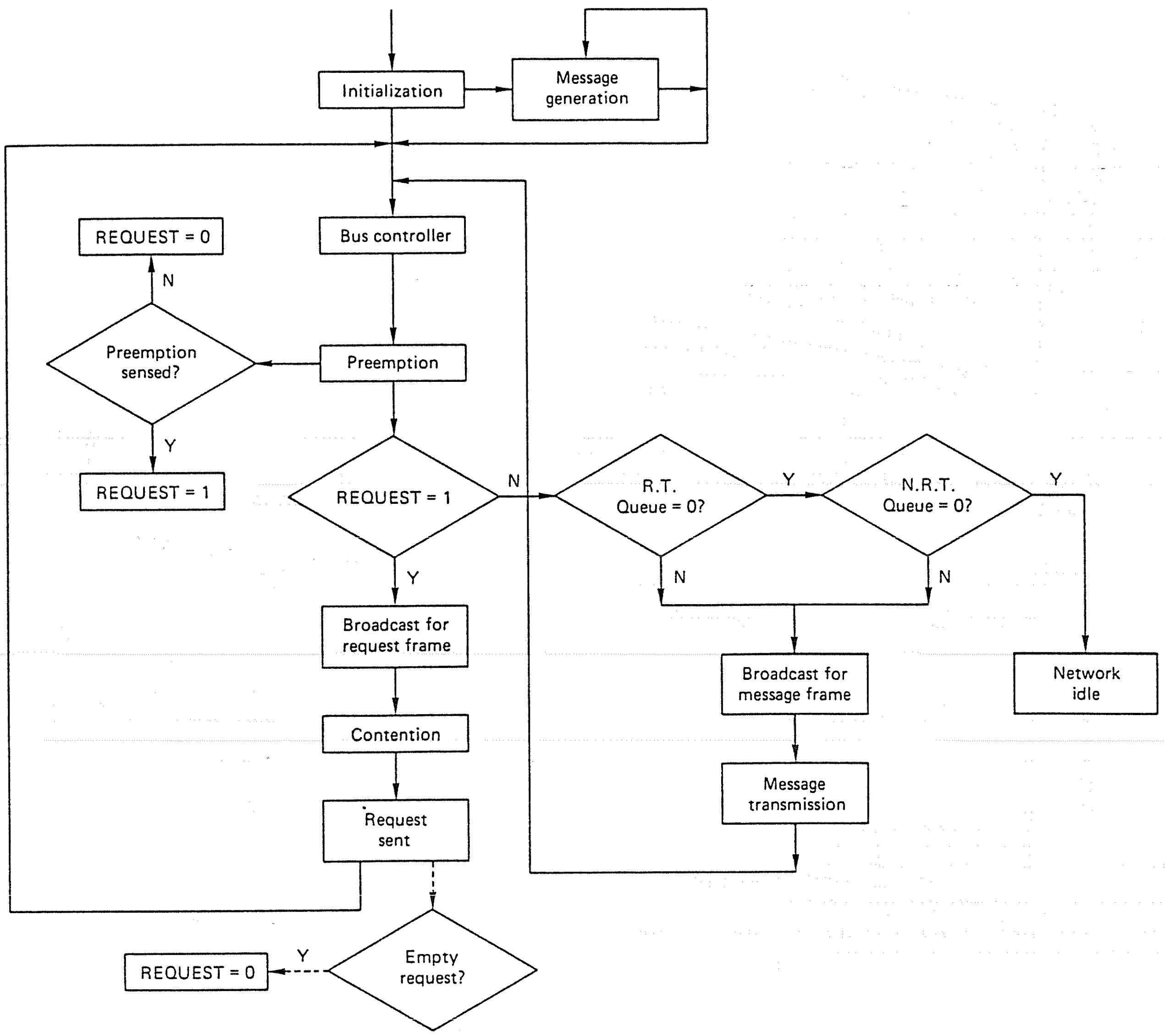

Figure 8. Protocol operations for discrete-event simulation. R.T.: real-time data; N.R.T.: non-real time data; - - -→: to and from other bus controller

with priority queue, but it is M/G/1 head-of-the-line with priority queue model in analysis.

- A multiple request scheme is applied in the simulation model, while it does not exist in the analysis.
- Stations that generate the same class of data are lumped together in the analytical model, making it different from the original FIFO scheme, while it is always FIFO in the computer simulation.
- The data collected from the analytical model is under the assumption of steady state condition, while the computer simulation model accommodates any transients that may occur before reaching the steady state.
- The request scheme used in the TBC protocol gives a preference to the upper stream station, and it is true in the computer simulation model not in the analytical model. The effect of the physical location of each RT is not considered in the analytical model.

In addition to the delay evaluation, it is interesting to know how many request frames have been used, how they relate to the channel capacity, and how many multiple requests appear in the simulation. 'Multiple request' means that more than one request has been put in a single request frame, as described previously. In Table 1, 'request throughput' is defined as the dimensionless number of *total request bits transmitted per unit time divided by the channel capacity in bits per unit time.* Similarly, 'data throughput' is the dimensionless number of *total message bits transmitted per unit time divided by*

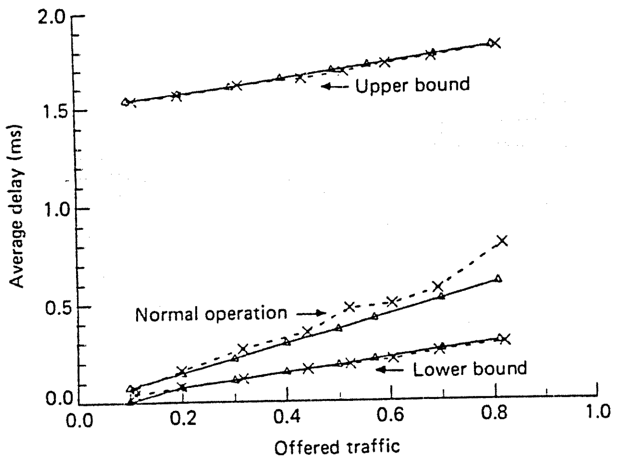

Figure 9. Delay performance of sensor-controller data (Note: offered traffic of the TBC protocol for both buses is twice that shown in the figure). —: analysis; ---: simulation

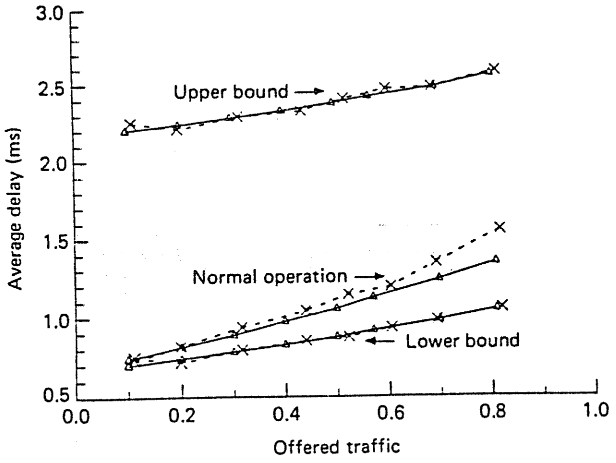

Figure 11. Delay performance of video data. (Note: offered traffic of the TBC protocol for both buses is twice of that shown in the figure). —: analysis; ---: simulation

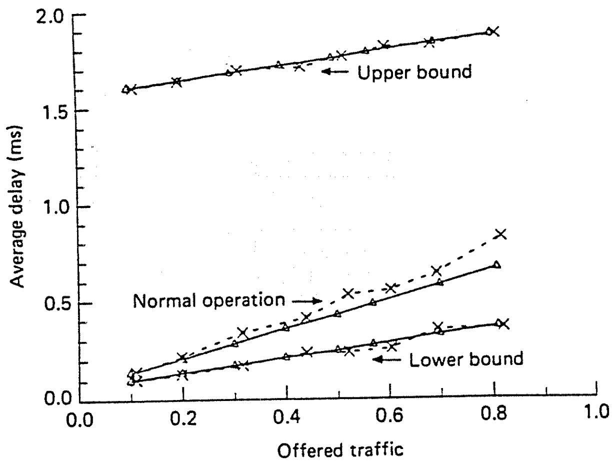

Figure 10. Delay performance of voice data. (Note: offered traffic of the TBC protocol for both buses is twice of that shown in the figure). —: analysis; ---: simulation

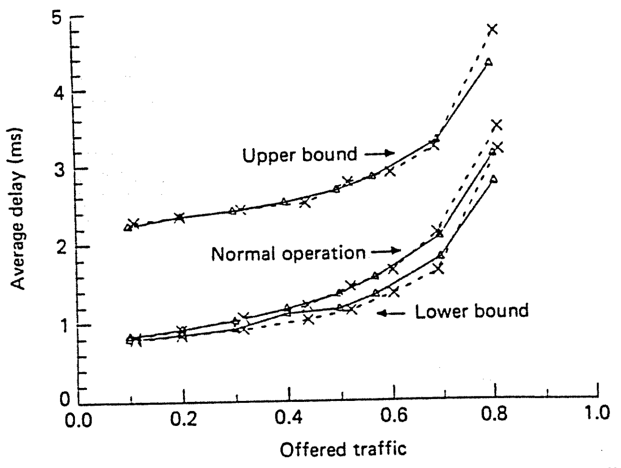

Figure 12. Delay performance of non-real-time data. (Note: offered traffic of the TBC protocol for both buses is twice of that shown in the figure). —: analysis; ---: simulation

the channel capacity in bits per unit time. Simulation results at different levels of network load show that the request throughput is insignificant compared to data throughput, and does not bear a monotonic relationship. This implies that the bandwidth consumed by request frames is negligibly small, and may be treated in the category of noise relative to data throughput.

The network running the TBC protocol actually contains two dependent subnetworks, and the overall network utilization is twice that of the throughput value shown in Table 1. The following conclusions can be drawn from the simulation and analytical results:

- *A priori* bounds can be set for the delay of real-time data on the basis of the statistical characteristics; and the delay for non-real-time data is finite provided that the bus capacity is not exceeded.

Table 1. Request bits throughput

Data throughput	*Request throughput*	*Average value of requests per frame*
0.10	0.00021	12
0.20	0.00040	28
0.30	0.00030	76
0.40	0.00042	107
0.50	0.00042	104
0.60	0.00059	145
0.70	0.00039	222
0.80	0.00042	269

- Request frames occupy only a small portion of the channel capacity.
- The protocol is capable of integrating heterogeneous traffic, and can accommodate a large number of stations.

COMPARISON WITH OTHER PROTOCOLS

The performance of the TBC protocol is compared with that of Fasnet and Buzznet because of the similarity in topology. Generally speaking, the TBC protocol has two major advantages: structural simplicity, and accommodation of heterogeneous traffic. With the exception of the two bus controllers, the level of complexity in the remaining large number of stations (i.e. remote terminal) is very moderate in view of both hardware and software. This is possible because of the centralized bus control scheme used in the TBC protocol which can accommodate heterogeneity of network traffic. The overall throughput of the network running the TBC protocol is approximately twice that of the other two protocols, and also the number of stations being accommodated in the TBC protocol is also twice that of Fasnet and Buzznet. The comparisons apparently do not show the advantage of the TBC protocol in accommodating heterogeneous traffic, as only two types of data are considered in this simulation. If a new class of data, especially real-time data, is to be added to the network, Buzznet has to define a new jamming pattern, and Fasnet has to generate a new round-robin cycle for the new data. In other words, Fasnet and Buzznet may have to update the network interface in every station. In the TBC protocol, the priority of the new data can be defined online or offline between the specific station and the bus controller, and added to the network without affecting other stations.

In Fasnet, different types of data are transmitted by way of reservation. Network utilization will be reduced if the message does not arrive at the buffer when the slot(s) come by, e.g. silence in voice communication or data compression in video conference. Even though Fasnet allows a station to grasp any unused slot, this will give preference to a downstream station, and this is unfair to the upper stream stations. Furthermore, complexity arises for the priority cycles (for Fasnet) and jamming pattern (for Buzznet) when the level of priority increases. In addition, these two protocols mandate the same message to be transmitted in both buses, and lead to a relatively poor network utilization. The results of comparison are shown in Figure 13 and 14.

The delay performance of the TBC protocol in Figure 13 is much better than Fasnet in normal operation, i.e. throughput is less than 0.7. In contrast Fasnet, which allows each data station to transmit one slot per cycle, has better performance at high load. The TBC protocol suffers longer delays under heavy load because, in this specific scenario, at least 10% of the channel capacity is used in transmitting requests. When the network is congested or heavily loaded, round-robin disciplines used in Fasnet allow all users to transmit at approximately the same level,

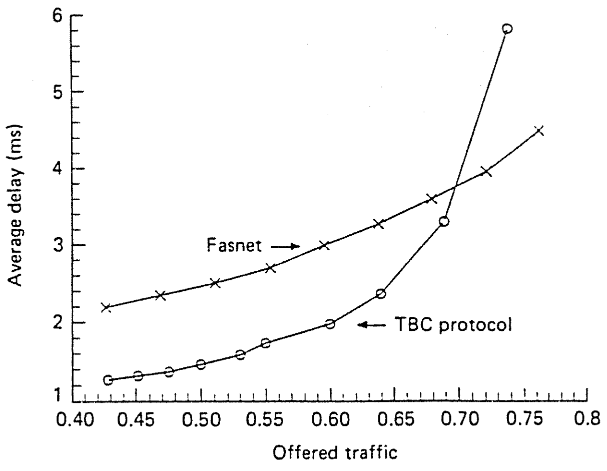

Figure 13. Performance comparison between TBC and Fasnet protocols. (Note: offered traffic of the TBC protocol for both buses is twice of that shown in the figure). X: Fasnet; O: TBC

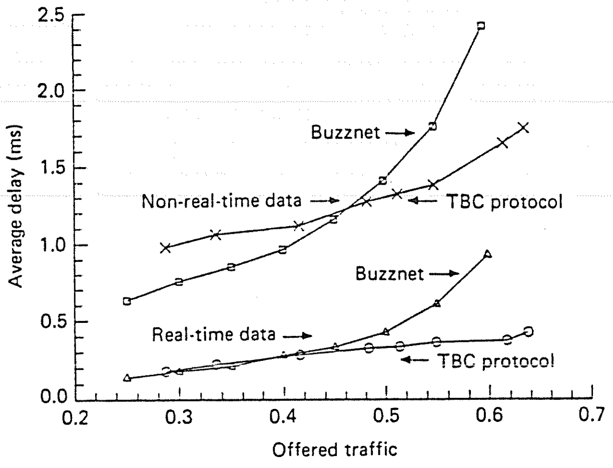

Figure 14. Performance comparison between TBC and Buzznet protocols. (Note: offered traffic of the TBC protocol for both buses is twice of that shown in the figure). □: Buzznet, non-real-time data; X: TBC, non-real-time data; △: Buzznet, real-time data; O: TBC, real-time data

which is fairer than FIFO[18]. This is the reason that Fasnet has smaller delays under high load. Although any unused voice slot in Fasnet can be grasped and used by a downstream station, the queued data in the upper stream stations would still suffer. Thus, free slots may return to the locomotive station (the first station), while the upper stream stations have a lot of messages in the queues, especially when the number of stations is large. The TBC protocol does not use the reservation scheme, and its delay performance is superior to that of Fasnet in normal traffic load.

To compare the TBC protocol with Buzznet, it can be seen in Figure 14 that the delay for TBC is slightly higher than Buzznet under lighter load (i.e. throughput is less than 0.5), but TBC yields smaller delays under higher load. As the traffic intensity becomes higher, the switching between random access mode and controlled access mode of Buzznet becomes more frequent, resulting in increasing delay. Conceptually, under heavier load the real-time suffers larger delays in Buzznet because it is very likely that all real-time data would encounter non-real-time data transmission. On the other hand, in TBC the opportunity of sending a message is only dependent on the activity of the other bus. Higher traffic load in one bus does not imply that the load in the other bus is also high. Furthermore, the probability of sending multiple requests in TBC is much higher in heavier traffic which, in turn, tends to reduce the overall delay.

SUMMARY AND CONCLUSIONS

The proposed TBC protocol uses a simple mechanism in all remote terminals, and makes it possible to integrate a large number of stations with heterogeneous data. It performs a partially centralized control scheme over the twin bus topology, and eases the operation of network management which may be very difficult for those protocols with completely distributed control, especially if the number of stations is large. The TBC protocol maintains global queues and enforces the priority scheme and fairness. In other words, a higher priority message is always transmitted in preference to the lower priority message. Furthermore, any addition or removal of a particular type of message can be worked out just between the bus controller and the station(s) without affecting other stations. This avoids the interruption or shutdown of the network.

The TBC protocol can accommodate more stations than Fasnet and Buzznet with fairly good performance of message delay. The TBC protocol is designed to accommodate heterogeneous traffic and generate two dependent subnetworks for the twin bus topology, resulting in a very high network utilization. Because the two dependent subnetworks cooperate with each other, failure in a station can be detected much easier by the station itself or the bus controller. For example, the receiver failure can be detected by checking the activity of the other bus, and the transmitter failure can be detected by the bus controller for the missing message after the broadcast of a message frame.

REFERENCES

1 **Ayyagari, A and Ray, A** 'A fiber-optic-based protocol for manufacturing system networks: Part I — conceptual development and architecture', *ASME Paper No. 90-WA/DSC-3*. (Also to appear in *ASME J. Dynamic Syst. Measurements & Control*)

2 **Ayyagari, A and Ray, A** 'A fiber-optic-based protocol for manufacturing system networks: Part II — statistical analysis and discrete-event simulation', *ASME Paper No. 90-WA/DSC-4*. (Also to appear in *ASME J. Dynamic Syst. Measurements & Control*)

3 **Fine, M and Tobagi, F A** 'Demand assignment multiple access scheme in broadcast bus local area networks', *IEEE Trans. Comput.*, Vol 33 No 12 (December 1984) pp 1130–1159

4 **Personick, S D** 'Review of fundamentals of optical fiber systems', *IEEE J. Selected Areas in Commun.*, Vol 1 No 3 (April 1983) pp 373–380

5 **Tobagi, F A, Borgonovo, F and Fratta, L** 'Expressnet: a high-performance integrated-services local area network', *IEEE J. Selected Areas in Commun.*, Vol 1 No 5 (November 1983) pp 898–913

6 **Borgonovo, F** 'ExpressMAN: exploiting traffic locality in Expressnet', *IEEE J. Selected Areas in Commun.*, Vol 5 No 9 (December 1987) pp 1436–1443

7 **Tobagi, F A and Fine, M** 'Performance of unidirectional broadcast local area networks: Expressnet and Fasnet', *IEEE J. Selected Areas in Commun.*, Vol 1 No 5 (November 1983) pp 913–926

8 **Limb, J O and Flamm, L E** 'A distributed local area network packet protocol for combined voice and data transmission', *IEEE J. Selected Areas in Commun.*, Vol 1 No 5 (November 1983) pp 926–934

9 **Davids, P and Martini, P** 'Performance analysis of DQDB', *Proc. 9th Ann. Int. Phoenix Conf. on Comput. & Commun.*, Phoenix, AZ, USA (March 1990) pp 548–555

10 **Kolnik, I and Joseph, M** 'First FDDI local area networks', *12th Conf. Local Comput. Networks*, Minneapolis, MN, USA (1987) pp 7–11

11 **Dykeman, D and Bux, W** 'Analysis and tuning of the FDDI media access control protocol', *IEEE J. Selected Areas in Commun.*, Vol 6 (July 1988) pp 997–1010

12 **Gerla, M, Wang, G-S and Rodrigues, P** 'Buzz-Net: a hybrid token/random access LAN', *IEEE J. Selected Areas in Commun.*, Vol 5 No 6 (July 1987) pp 977–988

13 **Tseng, C-W and Chen, B-U** 'D-Net, a new scheme for high data rate optical local area network', *IEEE J. Selected Areas in Commun.*, Vol 1 No 3 (April 1983) pp 493–499

14 **Gerla, M, Rodrigues, P and Yeh, C** 'U-Net: a unidirectional bus network', *FOC/LAN 84*, Las Vegas, NV, USA (1984) pp 295–299

15 **Miller, R G** 'Priority queues', *Annals Math. Statistics*, Vol 31 No 1 (March 1960) pp 86–103

16 **Jaiswal, N K** *Priority Queues*, Academic Press, New York (1968)

17 **Peterson, J L** *Petri Net Theory and The Modeling of Systems*, Prentice Hall, New Jersey (1981)

18 **Morgan, S P and Lo, C Y** 'Mean message delays for two packet-FIFO queueing disciplines', *IEEE Trans. Commun.*, Vol 38 No 6 (June 1990) pp 744–746