Cyclone Technology: An Overview[*]

Sung Lee and Ashok Agrawala
Institute of Advanced Computer Studies
Systems Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742
{sylee, agrawala}@cs.umd.edu

Abstract

The current network, which is based on managing resources on demand and accepting uncontrolled communication request, often leads to problems such as congestion and other queueing bottlenecks. The extent of congestion and queues depends on the variability in customer arrival times, services needed, and the resource allocation mechanism used by system components. The queue sizes, which results in congestion, can be reduced only by controlling the variability in customer arrival times, and this is best done by making explicit use of time. Cyclone technology uses the information based on times of events explicitly, including the design of systems. Cyclone provides the coordination of resources through dynamic, time-based resource management leading to a network that is capable of providing end-to-end low latency communications free of losses, jitter, and congestions.

---

[*] This work is funded by a gift account from Novell. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Novell.

1 Introduction

Networks have been designed to process packets at nodes taking into account the packet header information. Clearly this information is not available until packets arrive at a node. As a consequence, the processing of a packet including scheduling its transmission on an outgoing link can be carried out only after a packet has arrived. In a connection-oriented environment, a connection is usually established for a well-defined use, and the traffic and its timing information is either readily available or easily derived. However, in the current networks, no attempt is made to utilize the timing information. The key idea in a Cyclone network is to use the timing information explicitly in managing all resources and actions throughout the network.

A basic operation of computer communication is to move a packet from a location **A** to another location **B**. The packet to be moved is stored in a buffer at **A** and is to be stored in a buffer at **B**. A device or a link, once started, carries out the move operation autonomously taking a fixed amount of time **t**. Note that **t** depends on the packet size, **s**, and the link speed. When a link carries out its operation with no interference, the time **t** is fixed, once **s** is fixed. Note that the move operation can be initiated in a number of ways. Typically, a node inspects the header of a packet, which contains information such as the destination address, to determine the link by which the packet is to move. The transmission, then, is activated when the link is free. However, when establishing the connection in a connection-oriented environment, the user usually knows how the connection is to be used and when the packets are to be sent. Based on the availability of such information about the connection, it is feasible to schedule the move of packets on a link. In this case, the link is activated based on its ***schedule*** or ***calendar***, which shows the time instances that it has to move the packets. The calendar of a link shows the buffer information as well when there are multiple buffers from which the link possibly takes packets. All store and forward operations are carried out in Cyclone technology using a time-based calendar, which is created when a connection is established. As a consequence, Cyclone connections can deliver loss free, jitter free, contention free, and in order data end-to-end.

We believe that the Cyclone technology offers the capability to the applications, which is not available from the current technologies, namely providing end-to-end guarantees for the time instants when each packet of the message is delivered to the applications.

In this paper, we present an overview of Cyclone technology describing its components and its basic modes of operations. We first define terminology followed by the description of Cyclone components in Section 2. In Section 3, data move and connection establishment operations are explained. We describe implementation issues in Section 4. Section 5 and Section 6 describe clock synchronization and scheduling approaches used in Cyclone, respectively. In Section 7, we present an example of handling a connection request. Several other issues are described in section 8. Our assessment of Cyclone is found in Section 9, and the comparison to other work is described in Section 10. We conclude this paper with the current status of Cyclone technology.

2 Architecture[12,13]

A Cyclone network provides a connection-oriented environment that accepts connection requests with a *temporal profile* specifying the time instants at which the sender sends data. In a Cyclone network, all components of the network operate in a deterministic manner according to a defined schedule based on temporal profiles provided by senders, thus, assuring a temporally determinate delivery of data end-to-end. Nodes are connected via links allowing an arbitrary topology for the

Cyclone network. A unit called Temporal Regulator (TR) provides the interface to hosts and external networks, therefore, forming a boundary of Cyclone Network. While the basic concept of Cyclone technology is very general and can be used with a variety of component technologies, we present the basic idea through one such design in this paper.

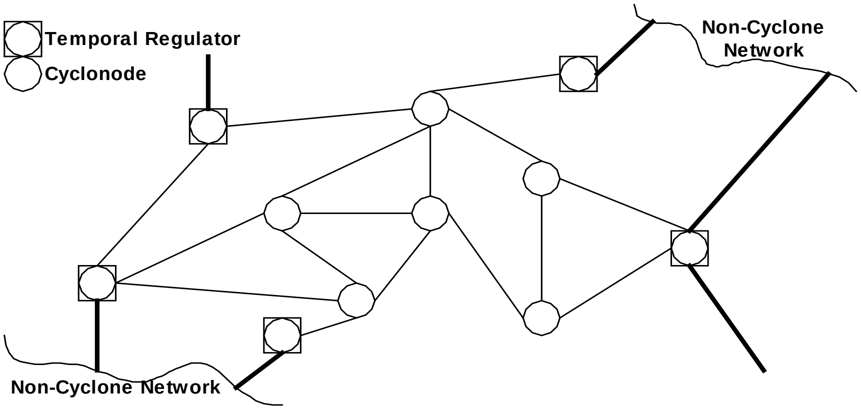


Figure 1 Cyclone Network

A message in this network is sent as a sequence of ***chunks***, each of which is transmitted in order. A chunk is a number of bytes that is fixed for the entire network. A ***control chunk*** is used to differentiate a chunk that carries a message that does not belong to any connection. A chunk that carries a message to request a connection is an example of a control chunk. The store-and-forward operation uses buffers at a node. A buffer is organized as a slotted buffer so that a chunk can be stored in a ***slot***. In a Cyclone network, a slot is a space-time description consisting of a ***chunk buffer*** and a ***time-tag***. We use a ***logical buffer*** that is of infinite size in this design description first, and then map it onto a physical buffer of finite size. Clearly, time-tags monotonically increase for the slots of a logical buffer.

Cyclone network is composed of links, ***CycloNodes***, and ***Temporal Regulators***. A link is a point-to-point unidirectional data path that connects two CycloNodes/Temporal Regulators. A link has a logical buffer at each end, a ***send-buffer*** and a ***receive-buffer***. There is a time tag for each slot, and the time tags for two adjacent slots differ by the chunk processing time of the link. For any slot in the send-buffer, there is a slot in the receive-buffer, and the time tags for the two corresponding slots differ by the link latency. Moving contents of the send-buffer to the receive-buffer is the only operation at a link. A link continuously moves one chunk at a time in a pipelined manner. Therefore, an empty slot in the send-buffer results in an empty slot in the receive-buffer.

In Cyclone, a ***CycloNode***, which has a set of incoming and outgoing links and a controller, carries out the functions of a switch/router, but in a temporally determinate manner. An outgoing link is organized in a way that it reads data from its send-buffer and stores the data in the receive-buffer, which is at the CycloNode at the other end of the link. Thus, the receive-buffers of incoming links and the send-buffers of outgoing links are visible at a node. At a CycloNode, chunks from the receive-buffers are moved to the send-buffers according to a calendar, a time-based schedule. This move operation must take place after the time-tag of the slot in the receive-buffer, but before the time-tag of the slot in the send-buffer.

Operations at a CycloNode, including switching of chunks, are carried out according to the time at its local clock. In order to maintain the required degree of temporal determinacy, it is important that the clocks in Cyclone network remain in synchrony with respect to each other. For this discussion, we assume that all clocks are synchronized and describe later how the required degree of clock synchronization can be achieved.

In addition, a CycloNode is responsible for managing connections, maintaining calendars, and routing. The controller at a CycloNode performs these operations. All communications among CycloNodes and/or Temporal Regulators are via control chunks. All control chunks are transferred to the controller, which processes them further. Note that all control chunks go across a link from one CycloNode to the next one, and not end-to-end.

A ***Temporal Regulator*** is a CycloNode with an additional interface, which is used to connect it to a host. All communications with any non-Cyclone network component are via this interface. A Cyclone connection is between two Temporal Regulators and operates in a temporally determinate manner. The host, however, may not supply the data with such a temporal determinacy. Therefore, a Temporal Regulator must provide mechanisms for tolerating the temporal variability observed in its host interface while maintaining temporal determinacy in its Cyclone interface. It initiates a Cyclone connection in response to the request received from the host. When it is at the receiving end of a connection, it assures that the host is ready to receive data for this connection before accepting the connection request. The source Temporal Regulator uses buffers to store data from the host until it transfers them to the Cyclone connection according to the temporal profile of the connection. The inverse operation takes place at the destination Temporal Regulator.

A Temporal Regulator provides a ***control chunk buffer***, which receives all control messages from the host such as connection request, terminate request, etc. Slots in a control chunk buffer are unscheduled; in other words, the controller examines the contents of chunks in this buffer for further action. A separate ***connection data chunk buffer*** is allocated for each connection when a connection is established. As a part of connection establishment processing, slots in a connection data chunk buffer are scheduled for outgoing transmission on a selected outgoing link or for receiving incoming data.

3 Operations

Let us consider how a chunk goes from a source Temporal Regulator to a sink Temporal Regulator at the other end of a connection. Note that when the connection is established, appropriate entries are made in the calendars at all intermediate CycloNodes for all chunks of the connection.

3.1 Data Movement

The data movement takes place across a link one chunk at a time. A link starts to move a chunk in a slot in its send-buffer with the current time-tag and places it in a slot in the receive-buffer with the corresponding time-tag. At a CycloNode, a chunk is moved from a slot in the receive-buffer to a slot in the send-buffer for an outgoing link at a time specified by the calendar. Note that this time must be after the time-tag of the slot in the receive-buffer and before the time-tag of the slot in the send-buffer. *As no header has to be processed during the move operation at the CycloNode, no headers are required for chunks*. In this way, a chunk is moved from a source Temporal Regulator to a sink Temporal Regulator for a connection.

Unlike data chunks, which go across the Cyclone network from a source Temporal Regulator to a sink Temporal Regulator, control chunks go from the controller at one CycloNode across a link to the controller at the other CycloNode. Their movement across a link is similar to that of data chunk and uses slots reserved at regular intervals or any unused slots.

3.2 Connection Establishment

A Cyclone connection is established between two Temporal Regulators with a defined temporal profile. The temporal profile for a connection request specifies all time instants when chunks would be ready for transmission. A source Temporal Regulator constructs a control chunk for this connection request and forwards it to a CycloNode selected according the routing table in its controller.

The controller at each intermediate CycloNode processes the request. First it examines the routing table to determine the outgoing link to be used. It then examines the temporal profile of the request and the current calendar of the node. If the request can be accommodated, it modifies the calendar by creating ***reserved*** entries in the calendar for this connection, and adjusts the temporal profile of the current request to conform to the reservations made. It then constructs a control chunk and forwards it to the next CycloNode through the outgoing link that is selected for this connection. In this way, the connection request traverses Cyclone network and reaches the sink Temporal Regulator.

Once the connection request reaches the sink Temporal Regulator, a confirmation message indicating the acceptance of the connection is sent back toward the source Temporal Regulator after the sink Temporal Regulator confirms that the host is ready to receive data for this connection. The confirmation message is sent along the same path that the request traversed to reach the destination. All CycloNodes on the path then change the calendar entries from ***reserved*** to ***confirmed*** state.

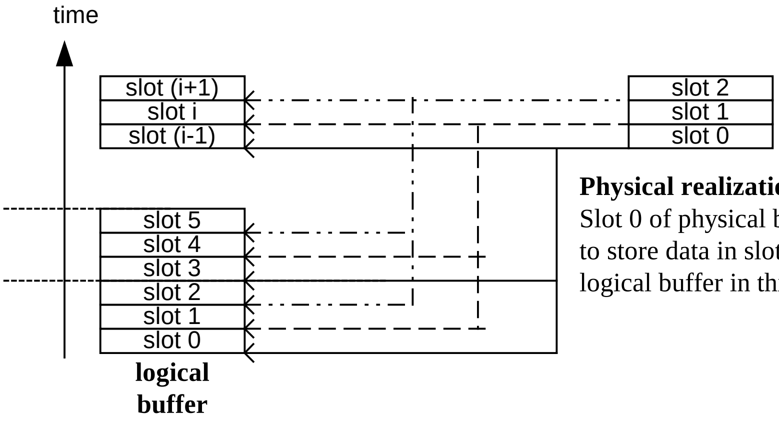
Any CycloNode can reject a connection request if it finds that the outgoing link cannot meet the request. A notification is sent on the upstream toward the source Temporal Regulator if any of intermediate CycloNodes rejects the connection request during this phase. All CycloNodes receiving this notification release resources that are reserved for this connection.

4 Implementation Issues

We have described the concept of a Cyclone network using time, which increases continuously, and logical buffers of infinite size. In a practical implementation, we define a period **P** that is common for the whole network. Let us consider a link **i**, which operates at a link speed, $\mathbf{L_i}$. To move a chunk of size, **C** bytes, this link takes $\mathbf{D_i}$ seconds. As a consequence, the time tags of consecutive slots associated with this link differ by $\mathbf{D_i}$. In the period **P**, the number of time slots, $\mathbf{N_i}$ is equal to $\mathbf{P/D_i}$. A physical buffer associated with this link has $\mathbf{N_i}$ slots, and we emulate the concept of an infinite buffer by circularly assigning the time tag to the slot of a physical buffer (Figure 2). One of these slots is reserved for sending control chunks, which can also be sent in any unused slot.

In the discussion above, we described the operations using two buffers for each link, which requires a buffer copy operation for each chunk. We can eliminate the copy operation by not having a send-buffer. Instead we have a sequence of pointers which point to receive-buffer slots and time tags. At the time specified by the time tag, the link starts the transfer of the chunk pointed to by the pointer reading it directly from the receive-buffer. By using dual-ported memory for the receive-buffers and having appropriate data paths between the outgoing links and the receive-buffers, we can carry out these operations with no conflicts.

In this mode of operation, we note that we need to construct a calendar for each outgoing link independently. This makes the Cyclone design highly scalable, as the complexity of constructing the calendar does not depend on the number of links connected to a node.



Physical realization of logical buffer: Slot 0 of physical buffer will be used to store data in slot 0, slot 3, etc. of logical buffer in this figure.

Figure 2: Buffer Model

5 Clock Synchronization

For any node, its own clock is the only mechanism for obtaining the knowledge of time and carrying out operations according to its calendar. These operations, however, have to maintain a fixed timing relationship with operations carried out at other nodes. Therefore, it is essential that the clocks in a Cyclone network be synchronized for their rates as well as phases. While the clocks do not have to maintain exactly the same time, they can not drift apart. As all practical clocks do show some jitter, and have varying rates, a specialized clock synchronization approach is used in Cyclone that synchronizes the clock rates and minimizes the jitter. The knowledge about the maximum permissible jitter is taken into account in scheduling.

We note that the timing on a link is controlled by the sending CycloNode and is visible to the receiving CycloNode in terms of the times when a period starts and when a slot starts. Based on this information, we have developed a synchronization algorithm [12], which maintains a high degree of synchronization among clocks, and gives a tight bound on the clock jitter. While the details of the clock synchronization approach are not included here, the results indicate that an accuracy of the order of a few parts in a million can be easily maintained when each node is using quartz clocks with known variability of a few hundred parts in a million.

Our design takes the phase variability into account by maintaining a small delay between the finish of a period and the start of the following period, which is adjusted to maintain high degree of synchrony. Even with a conservative estimate on the bound of phase variability this delay causes very little (less than .1%) loss of time or bandwidth.

6 Scheduling

As described in Section 2, data must be moved to a send-buffer only after it is placed in a receive-buffer for correct operations. Limited buffer space in the physical implementation imposes additional requirements. Recall that a link moves data continuously from one end to the other. When there is infinite buffer space, data reaching the end of a link is always written into a new slot.

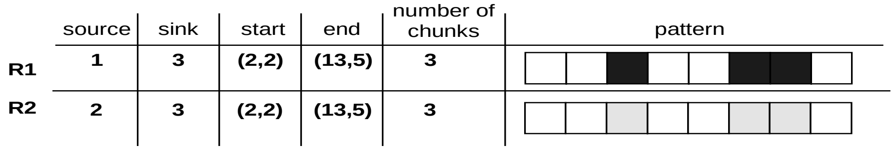
However, in a physical implementation where infinite buffer space is emulated through cyclic reuse of slots, data must be moved out before the slot is used again. This timing relationship is the basis of Cyclone scheduling.

When the controller of a CycloNode receives the temporal profile with a connection request, it starts the scheduling process by examining the calendar for the outgoing link that the connection would use. Beginning with the slot time for each incoming chunk, as defined by the temporal profile, the calendar of the outgoing link is examined to find and reserve the first available slot. The next incoming chunk is assigned to the next available slot. This process continues until all entries of a temporal profile have been assigned a slot in the calendar for the outgoing link. The collection of these assigned slots forms the temporal profile, which is forwarded to the next CycloNode with the connection request.

Our approach to scheduling in Cyclone only requires that the calendar of one outgoing link be examined for a connection. As a consequence, the complexity of scheduling does not depend on the number of links connected to a node. While the initial scheduling approach requires a search, the finite buffer size of the receive-buffer limits the search space significantly making the search based scheduling very efficient. This approach to scheduling has been incorporated in our current design to show the feasibility [12]. Clearly, more complex scheduling techniques can be used to reflect realistic constraints imposed by applications.

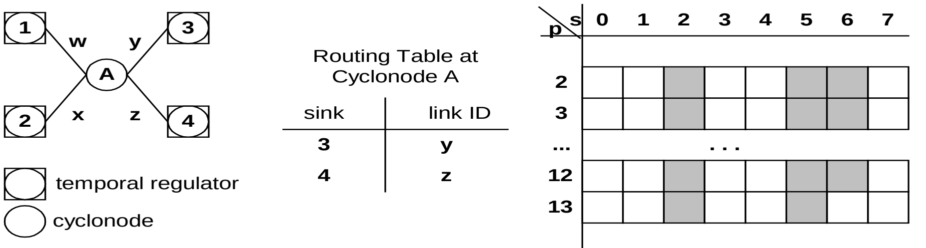
7 Example

We present an example of how a connection request is handled in a Cyclone network. Although a more complex network topology represents more realistic Cyclone network, we illustrate our current approach to scheduling in a simple network. The network, as shown in Figure 3 (b), is made up of four Temporal Regulators and a CycloNode. We concentrate on the processing at the CycloNode **A**. To simplify the example we use all links of the same speed and express time simply as (period number, slot number)



	source	sink	start	end	number of chunks	pattern
R1	**1**	**3**	**(2,2)**	**(13,5)**	**3**	☐☐■☐☐■■☐
R2	**2**	**3**	**(2,2)**	**(13,5)**	**3**	☐☐▨☐☐▨▨☐

(a) Requests



(b) Network Topology (c) Routing Table at **A** (d) Temporal profiles for R1 and R2

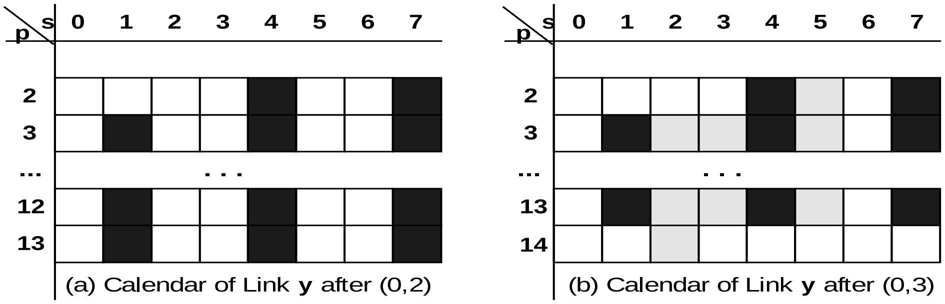
Figure 3 Network Topology and Requests

In this example, we use eight slots in a period and reserve the first slot in each period to transfer control messages. *We assume that it takes one slot time (**D**) to transfer a chunk across a link.* A source Temporal Regulator generates a connection request with the following information;

source, sink, start time, end time, the number of chunks in a period, and a pattern for a period.

Requests in Figure 3 (a) show how two or more connections share resources in a Cyclone network. The Temporal Regulators **1** and **2** generate requests **R1** and **R2**, respectively. According to the requests, the actual data transmission starts at the same time, (2,2). However, the Temporal Regulator **1** generates **R1** at time (0,1), and the Temporal Regulator **2** generates **R2** at time (0,2). The pattern and the number of chunks in a request are for a full period. The transmission pattern and the total number of chunks transmitted for the entire connection can be computed from a request. Figure 3 (d) shows the computed transmission pattern, and that a total of 35 chunks will be transmitted.

Since we assume that the transmission delay on a link is one slot time, request R1 arrives at CycloNode **A** at time (0,2). CycloNode **A** looks up its routing table, shown in Figure 3 (c) and finds that link **y** will be used for this connection. We assume that there are no other outstanding connections on link **y**; thus, the calendar for link **y** is empty. Since the calendar is empty prior to this request, chunks from this request can be scheduled to go on the outgoing link **y** as they arrive except the chunk arriving at the 8th slot. Since the first slot is reserved for control chunks, it is scheduled to go on the outgoing link **y** in the second slot. Figure 4 (a) shows the calendar of link **y** after reservations are made for request **R1**, and Figure 5 (a) shows the updated request **R1** to reflect the transmission pattern at CycloNode **A**.



(a) Calendar of Link **y** after (0,2)

(b) Calendar of Link **y** after (0,3)

Figure 4 Calendars at Link y

At (0,2), the request **R2** is sent from Temporal Regulator **2** to CycloNode **A**. This request arrives at **A** at time (0,3). Link **y** is selected according to the routing table at CycloNode **A**. Figure 4 (a) shows the calendar of link **y** at time (0,3). According to the approach presented in Section 6, the calendar for link **y** is examined to determine if there are enough slots to support this request. Figure 4 (b) shows the calendar of outgoing link **y** after reserving resources for the request **R2**, and Figure 5 (b) shows the updated request **R2** to reflect the transmission pattern at CycloNode **A**. Note that these entries are reserved. This node must receive a confirmation message, which must arrive before the starting time for these connections to be valid.

source	sink	start	end	number of chunks	pattern
1	**3**	**(2,4)**	**(13.7)**	**3**	□■□□■□□■

(a) Modified Request R1

source	sink	start	end	number of chunks	pattern
2	**3**	**(2,5)**	**(14.2)**	**3**	□□▨▨□▨□□

(b) Modified Request R2

Figure 5 Modified Requests

As the connection data transfer is to start in period 2, there is enough time for the confirmation message to be received. The connections can start transferring data chunks in period 2 and have them delivered to TR 3 according to the schedule.

Clearly this is a highly simplified example of setting up Cyclone connections. We have tested the approach with realistic parameters through simulation.

8 Other Issues

8.1 Routing

The controller at a CycloNode maintains a routing table that is used to determine an outgoing link that a connection is to use. By design, a Cyclone network does not have any congestion. Therefore, routing tables do not have to be modified except when the topology changes due to failures or recoveries. Any standard routing technique can be exploited for this purpose. Our current design is based on static routing tables that are defined at the design time.

8.2 Failure Handling

While there is no chunk loss in a Cyclone network, two types of failures are possible. Noise on links may cause some bit errors. As the contents of chunks are not examined by Cyclone network components, detection or recovery from bit errors is carried out at a higher level. As bit errors on links are becoming rare, we do not expect this error handling to play a significant role.

When a link or a node fails, its failure becomes visible to the nodes connected to it within one slot time. After this detection, several options are available for recovery [12]:

- Terminate connections and leave re-connections up to the applications,
- Terminate and re-connect for lost portion: after re-connecting all chunks that should have been received in the absence of failure are transmitted on the new route automatically, and
- Terminate and re-connect for application specified portion: after re-connecting, transmission starts from the application specified portion. This option may be useful for movies on demand, for example.

We are in the process of evaluating the options and incorporating the selected ones in the design.

8.3 Adaptation Layer

An adaptation layer provides the interface to non-Cyclone network components, such as hosts. Although it is anticipated that various classes of adaptation layers would be developed to support a wide range of communications to/from non-Cyclone networks, the first such layer [12] designed to date supports the functionality of TCP/IP applications. The current design provides an interface that includes all the functionality of the interface provided for TCP/IP and additional capabilities to extract information such as temporal profiles from applications.

9 Assessment

In this paper, we have presented the basic concepts of Cyclone technology. Clearly, the ideas presented here can be, and have been extended to cover a variety of issues of practical relevance. From the design, however, it is clear that a Cyclone network operates in a deterministic manner for all the connections in operation and can support the establishment of connections taking no longer than is taken for establishing TCP connections today. Our initial estimates are that the processing required for a connection request at a CycloNode will take less than a few milliseconds for a typical connection request when the controller is a current day, 200 MHz, processor, making the connection establishment times very reasonable. While a Cyclone network requires that the temporal profile for a connection is defined and used, we can always construct fixed, CBR channels that can be multiplexed and used by the hosts, with guaranteed performance.

Once a Cyclone connection has been established, the network guarantees that no chunks will be lost due to any congestion problems as all resources needed for them are reserved in a space-time domain at all CycloNodes involved in their movement. Further, the time at which they are to be delivered to the sink TR are precisely defined and adhered to by the network. As a consequence, we assure that there is no jitter in the deliver times of the chunks, as viewed by the clock of the Sink TR. If there is some jitter in the clock of the Sink TR due to clock synchronization, we expect it to be very small.

In our current design we have been working with a period of 20ms, with 1000 slots per period. This gives the slot time of 20 μs.

Cyclone technology assures a very high degree of link utilization for the transmission of data. Chunks require no header and, therefore, all their space can be used for data. Only a few bytes are required to mark the beginning and the end of a chunk. The only loss of time on a link is due to the slots reserved for the control chunks and some time required at the beginning of a period for adjusting the clocks. Our current analysis indicates that while using quartz clocks and our synchronization approach, which assures synchronization to the degree of a few parts in a million, we only need 20 μs for this adjustment. As we are working with one reserved slot for control chunks per period, we can keep a link busy with a utilization of better than 99% (up to 99 8% for the analysis above).

As the processing required for moving data through a CycloNode is very small, Cyclone technology is well suited for very high-speed networks. The simplicity of design makes hardware implementation of the CycloNodes and Temporal Regulators straightforward. Further, as a separate calendar is maintained for each outgoing link at a CycloNode, and the calendar of only one link is to be examined for connection processing which goes one CycloNode at a time, the approach is highly scalable. The only global information in the network is the routing tables that are only affected by a change in topology and not the network load.

The description of Cyclone technology has been presented for connections requiring guaranteed temporal behavior. As the control chunks go from CycloNode to CycloNode, they can be used to setup a datagram service, which will require direct participation of the controller at each node to carry out the store and forward operation. However, the Cyclone Network can support a connection-less service in this manner.

As the Cyclone network operates with reservation of resources, it may be argued that it makes inefficient use of resources when the reservations are made but the resources are not used. To our knowledge Cyclone technology is the only technology which can dynamically guarantee meeting very strict temporal requirements of an application. In order to meet such a guarantee, it is essential that no delays be caused due to the contention of resources. Further, more resources are made available for effective use by the applications with the high utilization supported by the design of the Cyclone network.

In order for Cyclone to accept a connection it is necessary that resources for the connection be available. Depending on the application temporal requirements it is possible that while the resources are available, they are not available at times to meet the temporal requirements of the application, causing Cyclone to reject the connection. Such a situation can be averted in an application by changing its acceptable temporal requirements.

10 Related Work

The basic approach used in Cyclone technology is significantly different from most other technologies. It does have similarities with a few approaches presented in the literature. Due to space limitation, we include a discussion of only a couple of them here and include several more in the references.

The Cyclone approach is similar in concept to TDMA, but with a significantly increased flexibility, along with dynamic adaptability. Synchronous Transfer Mode (STM) [1,2] is used in SONET environments as a means of managing multiple connections using time-based framing. Like Cyclone, this approach uses the time of signals explicitly and uses frame structure, which permits using time slots, usually of 125µs. Various connections are multiplexed on a single link in each frame. This approach has been designed for STS or OC links at very high speeds. Both STM and Cyclone use time explicitly. However, STM is a very rigid structure implemented using special hardware whereas Cyclone has been designed for general network applications to utilize links and processors of any speed.

Isochronets [3,4,5] is a switching architecture, which divides network bandwidth among routing trees, formed from all nodes to a given destination, in time. Traffic in Isochronets moves by following its path through an enabled tree from a leaf (source) to the root of the routing tree (destination) during its time band. Isochronets reserves resources on a per-route basis, which is not as frequent as a per-connection basis, for the cost of reservation is rather high. However, this approach tremendously cuts down on the flexibility due to the static nature of a route based reservation. Cyclone achieves the flexibility of per-connection resource reservation through efficient scheduling schemes. Synchronization in Isochronets is achieved end-to-end requiring additional delay along the links. Through this process, the latency at a node is minimized at the cost of additional delay added to the links. In contrast, through direct control of time and events in the network, Cyclone achieves a high degree of synchronization while maintaining low latency.

11 Status and Directions

In this paper, we have presented the basic concepts of the Cyclone technology, which manages all network resources in a space-time domain and dynamically adjusts the resource reservations as new connection requests are received and connections terminate. In order to present the idea clearly, we used simple examples. Our current design supports links of different speeds, receive-buffers of different sizes, etc., and has been tested through extensive simulations. Most of the protocols used in a Cyclone network have been designed and tested and an adaptation layer that provides the TCP/IP interface to applications has been completed. At present, we are critically evaluating our current design to choose among several implementation alternatives.

References

1. Black, U., “Emerging Communications Technologies,” Prentice Hall, Upper Saddle River, NJ. 1997.
2. Partridge, C., “Gigabit Networking,” Addison-Wesley, Reading, MA. 1994 pp.61-87, 253-310.
3. Florissi, D., and Yemini, Y., “The Gigabit per Second Isochronet Switch,” *Technical Report CUCS-006-95*, Distributed Computing and Communications Lab, Columbia University, 1995.
4. Florissi, D., and Yemini, Y., “An Overview of the Isochronets Architecture for High Speed Networks,” *Technical Report CUCS-007-95*, Distributed Computing and Communications Lab, Columbia University, 1995.
5. Florissi, D., and Yemini, Y., “Isochronets: a High-speed Networks Switching Architecture,” *PhD Dissertation*, Columbia University, 1995.
6. Golestani, S., “Congestion-free transmission of real-time traffic in packet networks,” *Proceedings of IEEE INFOCOM ’90*, San Francisco, CA. June, 1990.
7. Golestani, S., “A Stop-and-Go queueing framework for congestion management,” *SIGCOMM ’90 Symposium, Communications Architecture and Protocols*, Philadelphia, PA. September 1990.
8. Network Working Group, “Experimental Internet Stream Protocol, Version 2 (ST-II),” *Request for Comments (RFC): 1190*, CIP Working Group, October 1990.
9. Zhang, L., Deering, S., Estring, D., Shenker, S., and Zappala, D., “RSVP: A New Resource ReSerVation Protocol,” *IEEE Network Magazine*, September 1993.
10. Mitzel, D., Estrin, D., Shenker, S., and Zhang, L., “An Architectural Comparison of ST-II and RSVP,” *Proceedings of IEEE INFOCOM ’94*, 1994.
11. Mitzel, D., Estrin, D., Shenker, S., and Zhang, L, “A Study of reservation Dynamics in Integrated Services Packet Networks,” *Proceedings of IEEE INFOCOM ’96*, 1996.
12. Detailed Description of Cyclone Technology, Technical Report to be published.
13. Cyclone Technology, Patent Pending.