1992

Multiswitch Hardware Configuration

Douglas E. Comer
Purdue University, comer@cs.purdue.edu

Victor Norman

Report Number:

92-009

MULTISWITCH HARDWARE CONFIGURATION

Douglas Comer
Victor Norman

CSD-TR-92-009
February 1992
(Revised August 1992)

Multiswitch Hardware Configuration

Douglas Comer
Victor Norman

Purdue University
West Lafayette, IN 47907

CSD-TR-92-009
August 24, 1992

The Multiswitch project is exploring network architectures that use high-speed, multiprocessor packet switches to forward traffic among connected networks. Our goal is to find an architecture that can scale to handle the traffic from thousands of communicating machines.

This paper describes a particular hardware configuration used in a prototype Multiswitch network. The configuration uses multiple INMOS transputer processors arranged in a tightly-coupled group to form a packet switch. The packet switch topology provides low internal switching delay, high path redundancy, and high reliability.

A set of one or more Multiswitch packet switches can be interconnected to form a network with arbitrary topology. We are particularly interested in a topology that uses a ring of packet switches with a special-purpose central interconnect. The topology offers high throughput and low congestion.

Our project views a Multiswitch network as a *geographically distributed router*, capable of interconnecting hundreds of networks (usually local-area networks). The technology will be flexible enough to serve as a local-area, metropolitan-area, or wide-area network.

1. Introduction

In packet switching networks, network delay (or packet delivery delay) is defined as the time required to deliver a packet from its source node to its destination node. A network's hardware configuration, software implementation, and network management each contribute to the network's packet delivery delay. Packet delivery delay affects the behavior of high-level protocols and applications [Com88]. As applications continue to show an increasing dependence on network services, the speed, consistency, and reliability of the underlying network prove to be issues of increasingly greater importance.

Network delay results from slow transmission media. Recent improvements in transmission media technology support gigabit per second transmission speeds. However, most extant technology, such as Ethernet over coaxial cable, operates at megabit per second speeds. Additionally, most computer interface hardware cannot transfer data between the memory and network as fast as the network can send it across the transmission media [GNT90].

Network delay also occurs when enqueued packets must wait to be serviced by the processor. A processor enqueues packets when congestion on the network forces the processor to delay packet processing or packet transmission, or when the transmission media delivers more data to the processor than the processor is able to handle.

The most significant network delay occurs each time a packet must traverse an intermediate gateway node. Minimizing the average number of intermediate nodes for all paths through the network significantly reduces this type of network delay.

The Multiswitch project at Purdue University is investigating a low-cost, high-speed, multiprocessor-based networking architecture. One important thrust of the project is to provide a network that offers low network delay by investigating and eliminating the sources of network delay.

Our project views the Multiswitch architecture as a geographically distributed router. As a router, the network simply forwards externally-generated packets to external destinations. The architecture provides service to multiple connections each of which may be supporting one of a variety of networking protocols.

All aspects of the Multiswitch hardware design contribute to the high-speed, low-delay transfer of packets. Low network delay requires small hop counts, high-speed processors, high-speed communication media, redundant connectivity of nodes, and the implementation of congestion control mechanisms [Yav89].

This paper focuses on the design decisions and preliminary implementation of the hardware for a Multiswitch network. Section 2 presents the design criteria that influenced the packet switch hardware design and implementation. Section 3 outlines design decisions that affected the current prototype implementation. Section 4 explains the Multiswitch hardware implementation. Section 5 outlines the hardware components in greater detail, and explains the current state of the project. Section 6 describes a recommended overall Multiswitch network configuration. Section 7 summarizes the current work and concludes the paper.

2. Packet Switch Design Criteria

The purpose of a Multiswitch packet switch is to rapidly process and forward data packets. This section outlines the design goals used to achieve that purpose.

- **Low Hop Counts:** To achieve low packet delivery delay, we seek to minimize the average path length through the network. A low path length minimizes the packet delay by reducing the number of intermediate packet switches required to deliver the packet.

- **Redundant Paths:** To achieve reliability, redundancy of paths between end points is necessary. Redundancy of paths between nodes allows the packet switch to continue to function during link or node failures. Multiple paths also provide additional network bandwidth for use by advanced routing algorithms. Additionally, alternate paths may be chosen when congestion occurs on the normal route, or alternative paths may be used in parallel with the normal route to distribute network load.

- **Multiple Processors:** Currently, processor speeds constrain the performance of packet switching technology. To handle the high volume of traffic, the Multiswitch project employs multiple processors per packet switch, allocating one processor per packet switch interface. These processors must be able to communicate well with each other and with high-speed hardware interfaces, such as fiber-optic interfaces.

- **Low Hardware Cost:** To keep hardware costs low, the Multiswitch packet switch uses off-the-shelf hardware components whenever possible. The low-cost criterion restricts us from building custom hardware that implements the lower levels of the networking software. Performance of the packet switch may be increased through a corresponding increase in cost and the installation of custom hardware. The low cost constraint distinguishes our work from that of other networking research [MDS90, Par90, GNT90].

- **Flexibility:** Our design must remain flexible to allow experimenting with alternative packet switch designs. Additionally, we wish to allow the resulting network to assume arbitrary topologies.

- **Scalability:** The Multiswitch network should scale well so that many external networks can be serviced.

3. Packet Switch Design Decisions

This section discusses design decisions that were made for the current implementation of the Multiswitch network prototype.

3.1. The INMOS Transputer

We chose the INMOS transputer [INM88c] as the processing element for our prototype. The transputer CPU operates at high speed, increasing the ability of each processor to rapidly process and forward packets. Also, the transputer supports 4 full-duplex 20 Mbps serial links. The transputer provides parallel use of these links through a microcode-supported communications protocol and through operations in its instruction set. The transputer may communicate simultaneously on all four links without significant effect on communication speeds. Thus, one transputer may provide up to 80 Mbps of network bandwidth under optimal conditions.

The trivial connection of multiple transputers into a network via the serial links provides a reliable, intra-packet switch communication medium. The communication protocol used between transputers allows simple transmission of packets over the links without the use of complex link-level headers. The limited link-level overhead helps reduce packet delay.

The transputer is a relatively inexpensive processor. Combining multiple transputers into one packet switch does not incur high hardware costs. Additionally, future versions of the transputer

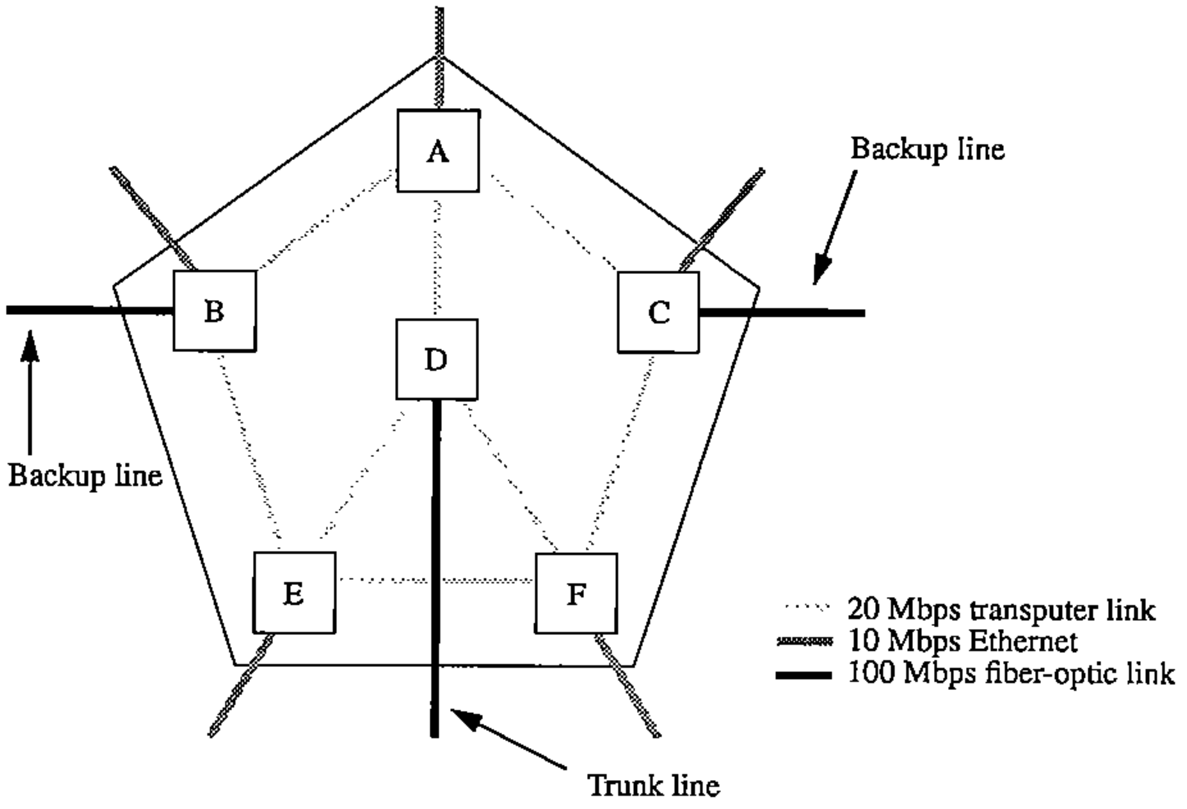

Figure 1: Packet switch architecture

promise to provide more on-chip RAM, higher clock speeds, and link speeds that are 5 times faster than the current version.

3.2. The 50% Rule of Network Bandwidth

To limit the congestion on packet switch links, we have defined and adopted the *50% rule of network bandwidth.* This rule states that offering load over 50% of the stated capacity of a link significantly increases the possibility of occurrence of congestion, high packet loss, and data corruption in the network. Thus, we have restricted the amount of possible offered bandwidth to 50% of any of the links within the packet switch.

4. Packet Switch Implementation

Figure 1 shows a diagram of the Multiswitch packet switch. Each of the five peripheral boxes represents one processor, and is referred to as a *ring* or *peripheral* node. The central processor D, or *hub node*, connects to three ring nodes. Together, the 6 nodes comprise a *wheel.* The peripheral ring nodes forward traffic within the packet switch and serve as gateways to externally connected networks. The hub node primarily forwards packets into and out of the packet switch. We refer to each individual processing element as a *node* and each collection of 6 nodes as a *packet switch.*

Each of the processors services 4 connections, many of which are 20 Mbps transputer serial links. Processors B, C, and D service 100 Mbps fiber-optic interfaces. Each peripheral node supports an Ethernet interface. These Ethernet interfaces connect to local area networks that are the source and destination for data traffic flowing through the Multiswitch network. Up to 5 Ethernets connect to each packet switch, so that each packet switch may offer up to 50 Mbps of traffic at maximum load.

The wheel configuration allows traffic from any node to reach any other node in 2 or fewer network hops, with the average path length being 1.47 hops. The low hop count helps reduce network delay. Each transputer supports 4 links. Reserving one link per node for an external (Ethernet or fiber-optic) connection reduces the degree of each node to 3. The wheel topology, with 5 peripheral nodes and 1 hub node, uses all 4 connections of every node in the packet switch.

To comply with the 50% rule of network bandwidth, five nodes comprise the ring. Because each node may offer 10 Mbps of data from the Ethernet connection, a total of 50 Mbps may be offered to the hub node to deliver over its 100 Mbps fiber-optic trunk line. Thus, a maximum of 50% of the stated link capacity may be offered.

In the unlikely occurrence of node or link failure, the wheel topology offers many alternate paths. Given any single link failure, there exists at least 3 alternate paths between the two directly connected nodes. Additionally, the wheel topology does not present any inherent bottlenecks because of the high number of alternate routes.

5. Hardware Implementation Details

In this section we present details on the current working configuration of the Multiswitch hardware. The current prototype includes 3 nodes, each fabricated on its own separate board, and communicating with each other via the transputer high-speed links. The Multiswitch software, built on top of the Xinu operating system, controls the hardware. The Xinu operating system was ported and adapted to the transputer to make use of its fast context switching times, multiple process priorities, and concurrent processing capabilities. The following sections introduce details on each of the components of the current system.

5.1. The Transputer

The transputer serves as the processing element for the Multiswitch packet switch. We use the T800 transputer [INM88c, INM88b], which runs at speeds up to 30 MHz, executing up to 15 MIPS. The T800 also provides 4Kbytes of high-speed, on-chip RAM, a configurable external memory interface, and 4 high-speed communication links. The communication links operate bi-directionally and in parallel at either 5, 10, or 20 Mbps.

Although INMOS designed the transputer and its corresponding instruction set to directly implement the OCCAM language [INM88a], we program the transputer using the C language. The transputer instruction set contains approximately 100 operations, divided into 2 sets. The most commonly used functions, called *direct* functions, contain both operation and operand in 1 byte. The rest of the functions, known as *indirect* functions, require multiple bytes to specify operation and operand. The instruction set contains instructions useful for communication, image manipulation, floating point processing, and process manipulation.

The transputer microcode contains the concepts of processes, process priorities, ready queues, timer queues, and I/O queues. The transputer specifies a process as an `(instruction pointer, stack pointer)` pair. The rudimentary, microcode-encoded operating system timeslices processes according to their priority. These context switches complete in 1 microsecond. For more information on the transputer operating system, see [INM88b].

The instruction set includes operations to directly support communication over the high-speed links. These instructions perform communications synchronously between sender and receiver and provide the same reliability as that of writing or reading to memory cells.

The transputer does not provide support for virtual memory and thus does not allow protection of data through separate user and kernel modes. Additionally, the transputer does not support interrupts or interrupt processing.

5.2. The Xinu Operating System

The Xinu Operating System [Com84], running individually on each Multiswitch node, controls the Multiswitch hardware, and provides a user interface for monitoring and manipulating the operation of the nodes. Adapted for the transputer, Xinu makes use of the transputer's fast context switching times, two-level process priority scheme, timer queues, and communications instructions.

The Xinu operating system writes console output over a serial line to a program running on a UNIX machine. This UNIX program monitors both the serial line and standard input to the program. The program transfers the characters received from the transputer to standard output, while also transferring the characters received from standard input to the serial link device.

5.3. The Prototype Hardware

Table 1 shows the individual components that make up the current hardware. These compo-

Device	Quantity	Purpose
INMOS T800 transputer	1	Processing element: controls all other components
NSC Network Interface Controller	1	Interface to Ethernet LAN
Signetics DUART	1	Controls the serial line that is used as a console line.
RAM	512 KB	General purpose memory
EPROM	128 KB	Contains boot code and debugging mechanisms
EEPROM	128 bytes	Non-volatile storage of network id, node id, etc.
Glue logic	lots	Glue

Table 1: List of components included on one board

nents were fabricated on wire-wrap boards. The boards are installed in a single rack and receive power and ground via the common backplane. The high-speed links of the transputers also connect to the backplane so that the links can be connected easily into arbitrary topologies.

Currently we are testing the Multiswitch hardware using 3 boards. Two of the boards, B and C, contain an Ethernet interface, a serial line interface, and the 4 transputer links, while the third, A, simply communicates using a serial line interface and the 4 transputer links. The nodes currently represent a simple linked list with A serving as the middle node (see figure 2).

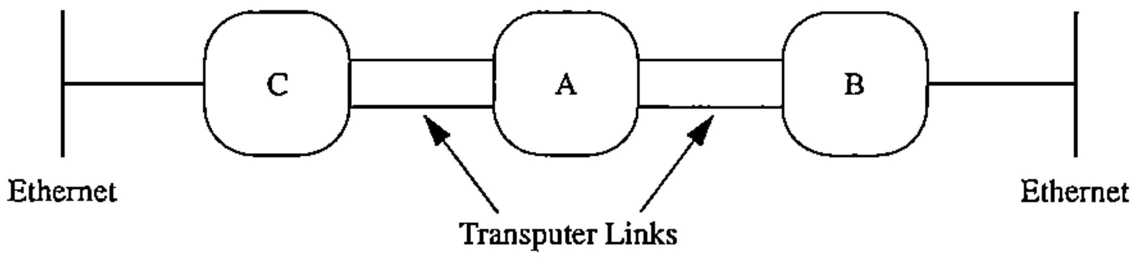

Figure 2: Currently, 3 nodes serve as an IP gateway between two IP networks. Peripheral nodes B and C each serve 2 transputer links and 1 Ethernet interface. Interior node A serves simply to transfer encapsulated IP packets through the Multiswitch network to the peripheral Multiswitch nodes and on to the destination IP hosts.

6. Overall Network Architecture

Multiswitch packet switches may be combined to form any arbitrary network topology, with the restriction that each packet switch services at most 3 fiber-optic connections. The Multiswitch project is investigating the configuration of packet switches into a wheel topology. Combining this overall wheel configuration with each packet switch wheel configuration results in a *wheel of wheels*, as shown in figure 3. Many of the benefits of the wheel configuration for packet switches also apply to the overall configuration. The wheel of wheels configuration provides a small hop count for many network paths. Most paths traverse 2 fiber-optic links and three packet switches -- the source packet switch, the central interconnect, and the destination packet switch.

We expect most traffic to remain within each packet switch. This expectation is based on the *principle of network locality* [GR87]. This principle states that the relative frequency of occurrence of packets sent from a source node to a destination node is inversely proportional to the distance (number of hops) between the source and destination nodes. Thus, for any given link, the majority of packets communicated over the link will be between nodes that are few hops away.

Using this principle, we propose that sites that tend to exchange a large number of packets be placed on packet switches adjacent to one another on the periphery of the network, regardless of the geographic location of the sites. Then, traffic between these sites can be routed over the backup line. For example, we envision that university departments that have more than 5 Ethernets (and thus will require more than one packet switch) will arrange their multiple packet switches to be adjacent. In this case, one would expect extensive use of the backup lines.

We are currently investigating the architecture of the central interconnect. The central interconnect must be able to quickly forward traffic between many fiber-optic links each operating at 100 Mbps or more. We are considering at least 2 possible designs to handle the traffic. First, we are investigating a custom-built, hardware solution. The advantage of a custom hardware solution is that the delay for switching packets can be minimized by the cooperation of efficient hardware and software. An obvious disadvantage is the extra cost of designing, fabricating, and debugging a custom-built device.

An alternative to the custom hardware solution is the creation of a central interconnect based on the current packet switch configuration. We are investigating combining multiple packet switches that have their Ethernet interfaces replaced with links to other packet switches. By mod-

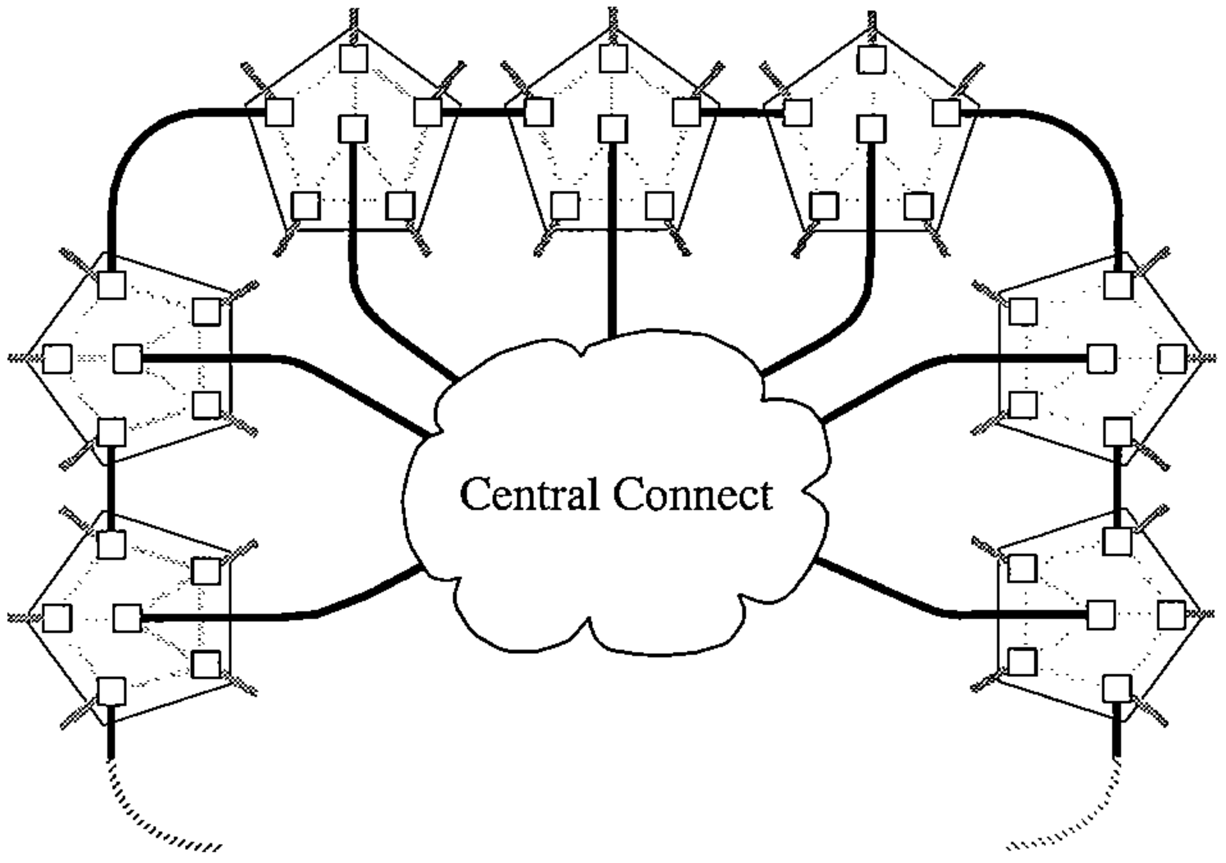

Figure 3: The wheel of wheels network configuration

ifying pre-fabricated and pre-tested hardware, we keep the cost of the central interconnect low.

7. Summary

The Multiswitch Network Architecture is a prototype configuration of multiprocessor packet switches designed to provide connected networks with high performance at low cost. The Multiswitch Network Architecture offers low packet delivery delay using a configuration that provides redundancy, small hop counts, reliability, and low error rates. The architecture is based on the INMOS transputer and uses the transputer's built-in communications links to connect multiple nodes within a packet switch. Packet switches may be connected in arbitrary topologies by fiber-optic links. The current prototype network includes 3 nodes and is being tested as an IP network gateway.

8. References

[Com84] Douglas Comer. *Operating System Design the XINU Approach.* Prentice Hall, 1984.

[Com88] Douglas Comer. *Internetworking with TCP/IP Principles, Protocols, and Architecture.* Prentice Hall, 1988.

[CN90] Douglas Comer, Victor Norman. Xinu on the Transputer. Technical report CSD-TR_1024, Department of Computer Sciences, Purdue University, September 21, 1990.

[INM88a] INMOS Ltd., *Occam 2 Reference Manual,* Prentice Hall, Inc., 1988.

[INM88b] INMOS Ltd., *Transputer Instruction Set A Compiler Writer's Guide,* Prentice Hall, Inc., 1988.

[INM88c] INMOS Ltd., *Transputer Reference Manual.* Prentice Hall, Inc., 1988.

[GR87] V.K. Garg, C.V. Ramamoorthy. Effect of Locality in Large Networks. *Proceedings 7th International Conference on Distributed Computing Systems,* Berlin, W. Germany, 1987.

[MDS90] Michael D. Schroeder, et al. Autonet: a High-speed, Self-configuring Local Area Network Using Point-to-point Links. Technical Report, Digital Systems Research Center, 1990.

[Par90] Gurudatta M. Parulkar. The Next Generation of Internetworking. *Computer Communications Review,* Vol. 20, No. 1, January 1990.

[GNT90] Gigabit Network Testbeds. *IEEE Computer,* Vol. 23, No. 9, pp. 77-80, September 1990.

[Yav89] Rajendra S. Yavatkar. An Architecture for High-speed Packet Switched Networks. Technical report, CSD-TR-898, Department of Computer Sciences, Purdue University, August 1989.