1986

The Cypress Coaxial Backbone Packet Switch

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

Thomas Narten

Rajendra Yuavatkar

Report Number:
86-648

THE CYPRESS COAXIAL BACKBONE PACKET SWITCH

Douglas Comer
Thomas Narten
Rajendra Yavatkar

CSD-TR-648
December 1986

The Cypress Coaxial Backbone Packet Switch

Douglas Comer
Thomas Narten
Rajendra Yavatkar

Computer Science Department
Purdue University
West Lafayette, IN 47907

CSD-TR-648

1. Introduction

The Cypress project [1] is investigating a low-cost, low-speed IP-based network that uses best-effort delivery to route IP datagrams among subscribers' sites and the DARPA Internet. Cypress is a leased-line based packet-switched network that forms part of the DARPA Internet. The basic technology around which Cypress is built consists of small packet switches called *implets* [3]. Point-to-point leased lines interconnect implets, and provide the fundamental communication media. Hosts using Cypress communicate using the DARPA protocol suite, popularly known as TCP/IP, which stands for Transmission Control Protocol and Internet Protocol.

An implet is placed at each subscriber's site, where it connects to the rest of the network over serial lines and to the subscriber's machines over a Local Area Network (LAN). At present, all implets connect to an Ethernet (we will often refer to the local network connection as an Ethernet throughout the rest of this paper.)

Cypress implets consist of small minicomputers that run a modified version of a conventional operating system. At the lowest level, the implet functions like a store-and-forward packet switch, receiving packets over leased lines, queuing them temporarily, and forwarding them on toward their final destination. At the second level, each implet functions like an Internet gateway [2], accepting packets from the local area network and routing them onto the Cypress network or vice versa. Indeed, an implet performs all the gateway functions. At the third level, the implet functions as a host computer capable of executing processes. It executes processes that monitor network traffic, log errors, and maintain routing tables.

Originally, the Cypress network was envisaged as a long vine of Cypress sites with each new site connecting to its nearest neighbor and a "root" site connecting the entire vine to the DARPA Internet. The initial Cypress prototype attached to the Internet at Purdue and included

This work was supported in part by grants from the National Science Foundation's Office of Advanced Scientific Computing (NSF-851-8369), with additional support from Digital Equipment Corporation, CSNET, ATT Foundation and Purdue University

sites at BBN (CSnet CIC), Boston University, U. of Arizona, and Digital Equipment Western Research Lab. But the vine topology is not suitable for a large number of nodes. As the length of the vine increases, the round trip times between pairs of hosts increase as the number of hops (packet switches traversed) between them increase resulting in longer delays in communication.

Also, with the expansion of the ARPANET and the advent of NSFnet and its associated regional networks, it has become clear that Cypress technology will be useful to many groups as an access technology.

Therefore, instead of expanding on the initial prototype that consisted of a single, connected network with only one "root", we are shifting toward a hub-site approach in which multiple, independent Cypress networks will provide access to the Internet. Conceptually, each hub site has a single implet that connects to ARPANET or NSFnet locally, and to a set of n implets at n other sites. Thus, these connections form the spokes of a wheel as shown in Figure 1. The roots of these multiple networks connect to a higher speed backbone (such as an Ethernet) and also have access to the Internet through a gateway connected to the same backbone. The network can now be incrementally expanded by adding an implet (a root) at the hub whenever necessary.

An implet is typically a minicomputer (such as a DEC Microvax II). The serial line hardware used with an implet is such that the implementation of the Cypress protocol for data transfer over the serial lines uses an interrupt driven I/O and involves servicing an interrupt for every character transferred. At the data rates of 9600 baud or above, an implet can only service a few lines before it becomes saturated. Thus, the expansion of the network is costly becuase a new minicomputer must be added to the hub site for every 3-4 lines. We have been working on ways to provide a high degree of fan-out at low cost.

To build an inexpensive, expandable packet switch at a hub site, we are exploring a new packet switch architecture in which the function of routing is separated from the function of data transfer to and from the network interface. The new architecture uses small, inexpensive machines (called *X-boxes*) that connect individual Cypress lines to the *Backbone Ether* (we will refer to the high-speed backbone network used at a hub as the Backbone ether) and handle the data transfer functions of the packet switch.

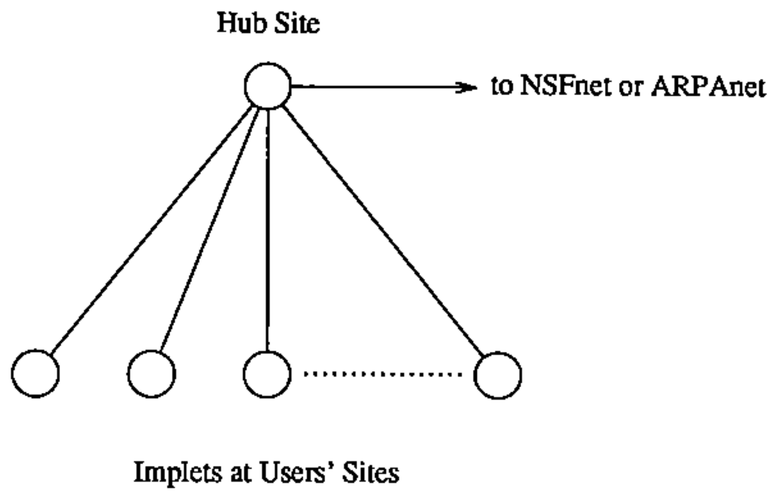

Figure 1. Connections at a Hub Site.

The remainder of this paper describes X-box in more details. Section 2 describes the architecture and the design of X-box. Section 3 describes the current prototype and gives the preliminary observations on performance.

2. X-box Architecture

A Cypress X-box provides a method of incremental hub site expansion. An X-box is a small, inexpensive machine which connects a Cypress line to the Backbone Ether at a hub site, as shown in Figure 2.

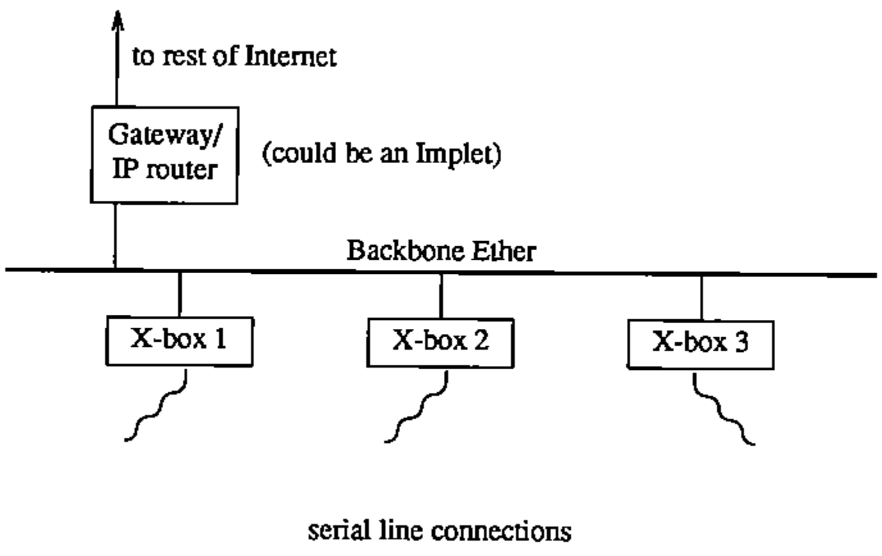

Figure 2. The connection of X-boxes at a hub site.

As described earlier, a Cypress implet performs the functions of routing (as in a gateway) and also handles data transfer to and from the network interface. The transfer of characters from and to the serial line involves servicing an interrupt per character which limits the number of lines an implet can handle without performance degradation. Our idea is to design a packet switch in which the function of data transfer is separated from that of routing.

Conceptually, the X-boxes handle the data transfer functions of the packet switch while a central router handles the IP-level routing functions. Each X-box accepts packets from the serial line and forwards them to the router for routing.

To make our novel architecture efficient, X-boxes cache routing information so that, in the steady state, each X-box routes IP packets (received over the serial line) using the Backbone Ether to their proper destination without using the central router. An Xbox acquires the IP routes through the ICMP redirects it receives from the central router. Thus, because the central router is only needed to reestablish caches after a power failure, many X-boxes can be connected to a given hub.

Constraints and Desires

The design of the X-box was dictated by the following constraints and desirable characteristics.

1. The system should use standard IP encapsulation on the Ethernet.
2. It should be possible to use a "standard" IP routing gateway for the Internet Router/Gateway in Figure 2. Specifically, the router should not be required to understand the Cypress protocol.
3. X-boxes will not contain fixed routing tables (i.e., they will not be real gateways). However, IP routes will be cached. The X-box will handle the ICMP redirects from the router.
4. The hub site should be incrementally expandable.
5. The most important traffic consists of IP datagrams. This is a crucial assumption.
6. It should be possible to test liveness of the X-box from both the Ethernet and Cypress sides.
7. It is desirable to have X-boxes fully implement the Cypress protocols. The Cypress Link level protocol is described in [4]. It follows from point 6 that X-boxes should respond to Cypress echo requests.

Design Decisions and Resultant Properties of X-boxes

In the following, we describe the design of X-box based on the constraints mentioned above.

Dual Nature of X-box

An Xbox plays a dual role with respect to the sites in the Internet world and the Cypress world. On the Internet side, an Xbox is viewed as an extension of a Cypress implet and an Internet router forwards the IP packets destined for Cypress hosts to the Xbox. On the other hand, the Cypress hosts view the Xbox as an extension of an Internet router and as a means of getting to other Internet hosts. In the Cypress world, an Xbox is also treated as another Cypress node. The latter view is very useful for testing and maintenance purposes.

Internet View

Each X-box has an Ethernet IP address and uses the standard communication methods (RARP/ARP/IP encapsulation) when communicating with the router-gateway. The Xbox handles ICMP requests addressed to its Ethernet address from either side. It honors the ICMP redirects from the Ethernet interface and builds a cache of IP routes based on them. It does not act as an IP gateway and hence does not necessarily perform the gateway functions such as fragmentation.

Cypress View

From the point of view of the Cypress world, an Xbox is a Cypress node with its own Cypress implet id and a Cypress IP address. The former is necessary for Cypress-level routing and addressing; the latter is necessary for IP-level addressing and permits ICMP echo requests from machines on the Cypress side.

Routing at the IP and Cypress Level

The X-box knows (or learns at boot time) its implet id (and hence, its Cypress IP address), its Ethernet IP address, and the IP address of a default router/gateway. All X-boxes on a physical net need not use the same router and an X-box need not know *a priori* what other X-boxes are on the same net at a hub site.

The X-box uses its route cache when routing IP packets from the serial line to the Backbone Ether, but ignores the routing cache when routing from the Backbone Ether to the serial line. It also uses the cache when routing ICMP echo replies back to the Backbone Ether. Routes in the cache are flushed periodically; if no clock is available, they are flushed after every *k* uses or by a global sweep that is activated by traffic.

In addition to the ARP cache, the X-box handles ICMP redirects from the router and maintains a table of ICMP redirects for routing purpose.

Apart from IP, the Xbox recognizes and fully implements Cypress neighbor handling and Cypress link-level protocols [3]. The X-box does an Ethernet broadcast of Cypress *RPF* and *Flood* packets using a packet type field that indicates "Cypress" in the type field of the Ethernet packet. It forwards the *direct* packets coming over the Ethernet over the serial line if they are not broadcast (using the Ethernet broadcast facility), or if they are not destined for itself. It uses neighbor handling for such packets to avoid fragmenting very large packets before forwarding. The Xbox design correctly handles cypress packets that are RPF, flood, or directly routed IP requests. It does not handle other directly routed Cypress control messages. We could extend the X-box easily to handle such packets by making the router a true implet and having the X-boxes send Cypress control packets to the implet without removing the header.

3. X-box Implementation

Basically, an Xbox consists of a CPU, memory, boot program (perhaps in ROM), Ethernet interface, and serial line. The current prototype uses a Digital Equipmment Corp. LSI-11/23 processor with 64Kb memory.

Current X-box Algorithm

The following describes the algorithm currently used in X-boxes.

Algorithm for packet handling in Xbox

I. Key data used by the algorithm

cip_addr X-box's IP address on Cypress
hard-wired or obtained at boot time with cypress protocol

cip_net Network portion of X-box's IP address on Cypress
obtained by masking off host portion of cip_addr

eip_addr X-box's IP address on Ethernet
hard-wired or obtained at boot time using RARP

eip_net Network portion of X-box's IP address on Ethernet
obtained by masking off host portion of eip_addr

rip_addr Router's IP address
hard-wired (zero for leaf node site where there is no router)

dip_addr destination IP address
obtained from each IP packet received

dip_net Network portion of destination IP address
obtained by masking host portion of dip_addr

II. For packets coming over the serial line

```
for each packet {
    if (handling is Flood or RPF) {

            decrement hp count;
            broadcast complete Cypress packet on Ethernet
                   (including header) with Ethernet packet
                   type set to Cypress;

    } else if (handling is neighbor) {

            if (packet type is not Cypress) {
                   discard the packet;
            } else {
                   respond to the packet over the serial line;
            }

    } else if (packet type is not "IP")
            discard the packet;

    /*  Packet contains a directly routed IP datagram */

    if (dip_addr == cip_add || dip_addr == eip_addr) {    /* for me */
            if (ICMP echo request)
                   reply over serial line;
            else
            if (ICMP redirect)
                   update the ICMP redirect table;
            else
                   discard the packet;

    } else if (dip_net == eip_net ) {    /* for someone on Ethernet */
            Map dip_addr to a physical Ethernet address, P, using ARP;
            Send packet over Ethernet to P;

    } else if (rip_addr == 0) {                /* No router available */
            discard the packet;

    } else {
            Map dip_addr to a new destination address, D, using the
              IP routing cache (D may be rip_addr);
            Map D to a physical Ethernet address, P, using ARP;
            Send packet over Ethernet to P;
    }
}
```

III. For packets coming in over the ethernet

```
for each packet {

    case (packet_type) {

     cypress:
         if (handling is Flood or RPF)
             send the packet over the serial line;

         if (handling is neighbor)
             discard the packet;

         else if (handling is direct and packet is not for me)
             send the packet over the serial line;

         else
             process packet locally;
             /* note: can only send responses to requests*/
             /* when there is a cypress-level packet router        */
             /* on the Ethernet.  Otherwise, discard.       */
     ARP:
         if (I have sender's IP address in ARP routing cache)
             update cache entry;
         if (packet is a request for eip_addr)
             send a response over the Ethernet;
         else if (packet is a response for which I am waiting)
             send packet that was waiting;
         else
             discard the packet;
     RARP:
         if (packet is a response for which I am waiting)
             extract eip_addr from packet;
         else
             discard the packet;
     IP:
         if (packet's physical ethernet address is broadcast)
             discard the packet;
         else if (dip_addr == eip_addr || dip_addr == cip_addr)
             if (ICMP echo request)
                 respond using routing cache;
             else
                 discard the packet;
         else
             send packet over the serial line with
             handling set to neighbor.
    }
}
```

Performance

One of the most important considerations in the design of the Xbox was not to degrade the performance of the Cypress system by introducing a bottleneck at the Xbox. We carried out some performance experiments by using the Xbox between two Cypress sites. The experiments involved ICMP echo requests and file transfers (using FTP) between the two sites. Preliminary experiments were restricted to at most one simultaneous file transfer in both the directions. The performance measured does not represent *maximum* since all the other sources of network traffic (such as broadcast packets generated by RWHO and ROUTED) were not eliminated.

All of the experiments involved data transfer between the two Cypress hosts (referred to here as *Cypress1* and *Cypress2*) with an Xbox acting as a packet switch. In particular, Cypress1 forwarded all the IP packets destined for cypress2 to the Xbox over the Backbone Ether. Similarly, Cypress2 routed all the IP packets destined for Cypress1 over a Cypress serial line to the Xbox.

In experiments involving ICMP echo requests, Cypress1 and Cypress2 sent ICMP echo requests (with packet size varying from 56 to 1000 bytes) to each other at the same time. In the experiments involving file transfers from Cypress1 to Cypress2 and vice versa, at most one file transfer was made simultaneously in both the directions. The size of the file being transferred varied between 140,000 to 220,000 bytes.

ICMP echo protocol and File Transfer Protocol (FTP) both provide the round trip time spent in the data transfer. A number of experiments (as described above) were carried out and we observed an average data transfer of 640 bytes/sec over a set of experiments. Additional experiments will be performed by varying the number of simultaneous file transfers and sending ICMP echo requests to the Xbox while the transfers are taking place.

4. Conclusion

We have designed a novel packet switch architecture which relieves a Cypress implet of the function of data transfer to and from network interface. Each Xbox can handle a Cypress line and can route packets over the Ethernet in the presence of a central router. It also uses a routing cache to route packets over the Ethernet to their proper destinations without using the central router. This reduces the load on the router and allows us to connect many Xboxes to a given hub.

Xbox is implemented using inexpensive hardware. The performance measurements on a prototype have yielded satisfactory results. Thus, this architecture provides a means of building an expandable packet switch at a hub site at a low cost.

References

[1] D. Comer, Thomas Narten and R. Yavatkar "The Cypress Network" CSD-TR 653, Purdue University, January 1987.

[2] Hinden, R. and Sheltzer, A., "The DARPA Internet Gateway", RFC 823, DARPA Network Information Center, September 1982.

[3] D. Comer and T. Narten,, "The Cypress Multifunction Packet Switch", CSD-TR 575, Purdue University, March 1986.

[4] D.E. Comer, "Cypress Link Level Protocol" CSD-TR 652, Purdue University, September, 1986.