The basic idea behind the solution to triangle routing is to let the sending node know the care-of address of the mobile node. The sending node can then create its own tunnel to the foreign agent. This is treated as an optimization of the process just described. If the sender has been equipped with the necessary software to learn the care-of address and create its own tunnel, then the route can be optimized; if not, packets just follow the suboptimal route.

When a home agent sees a packet destined for one of the mobile nodes that it supports, it can deduce that the sender is not using the optimal route. Therefore, it sends a "binding update" message back to the source, in addition to forwarding the data packet to the foreign agent. The source, if capable, uses this binding update to create an entry in a "binding cache," which consists of a list of mappings from mobile node addresses to care-of addresses. The next time this source has a data packet to send to that mobile node, it will find the binding in the cache and can tunnel the packet directly to the foreign agent.

There is an obvious problem with this scheme, which is that the binding cache may become out-of-date if the mobile host moves to a new network. If an out-of-date cache entry is used, the foreign agent will receive tunneled packets for a mobile node that is no longer registered on its network. In this case, it sends a "binding warning" message back to the sender to tell it to stop using this cache entry. This scheme works only in the case where the foreign agent is not the mobile node itself, however. For this reason, cache entries need to be deleted after some period of time; the exact amount is specified in the binding update message.

Mobile routing provides some interesting security challenges. For example, an attacker wishing to intercept the packets destined to some other node in an internetwork could contact the home agent for that node and announce itself as the new foreign agent for the node. Thus it is clear that some authentication mechanisms are required. We discuss such mechanisms in Chapter 8.

Finally, we note that there are many open issues in mobile networking. For example, the security and performance aspects of mobile networks might require routing algorithms to take account of several factors when finding a route to a mobile host; for example, it might be desirable to find a route that doesn't pass through some untrusted network. There is also the problem of "ad hoc" mobile networks—enabling a group of mobile nodes to form a network in the absence of any fixed nodes. These continue to be areas of active research.

4.2.6 Router Implementation

In Section 3.4 we saw a variety of ways to build a switch, ranging from a general purpose workstation with a suitable number of network interfaces, to some sophisticated hardware designs. In general, the same range of options are available for building routers, many of which look something like Figure 4.23. The control processor is responsible for running the routing protocols discussed above, among other things, and generally acts as the central point of control of the router. The switching fabric transfers packets from one port to another, just as in a switch; and the ports provide a range of functionality to allow the router to interface to links of various types (e.g. Ethernet, SONET, etc.).

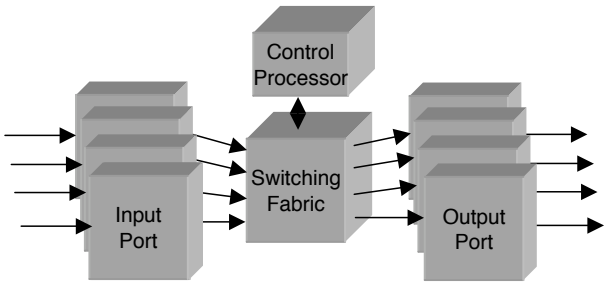

Figure 4.23: Block diagram of a router.

A few points are worth noting about router design and how it differs from switch design. First, routers must be designed to handle variable-length packets, a constraint that does not apply to ATM switches but is certainly applicable to Ethernet or Frame Relay switches. It turns out that many high performance routers are designed using a switching fabric that is cell-based. In such cases the ports must be able to convert variable length packets into cells and back again. This is very much like the standard ATM segmentation and re-assembly (SAR) problem described in Section 3.3.2.

Another consequence of the variable length of IP datagrams is that it can be harder to characterize the performance of a router than a switch that forwards only cells. Routers can usually forward a certain number of packets per second, and this implies that the total throughput in *bits* per second depends on packet size. Router designers generally have to make a choice as to what packet length they will support at *line rate*. That is, if pps (packets per second) is the rate at which packets arriving on a particular port can be forwarded, and linerate is the physical speed of the port in bits per second, then there will be some packetsize in bits such that:

$$\mathsf{packetsize} \times \mathsf{pps} = \mathsf{linerate}$$

This is the packet size at which the router can forward at line rate; it is likely to be able to sustain line rate for longer packets but not for shorter packets. Sometimes a designer might decide that the right packet size to support is 40 bytes, since that is the minimum size of an IP packet that has a TCP header attached. Another choice might be the expected *average* packet size, which can be determined by studying traces of network traffic. For example, measurements of the Internet backbone suggest that the average IP packet is around 300 bytes long. However, such a router would fall behind and perhaps start dropping packets when faced with a long sequence of short packets, which is statistically likely from time to time and also very possible if the router is

subject to an active attack (see Chapter 8). Design decisions of this type depend heavily on cost considerations and the intended application of the router.

When it comes to the task of forwarding IP packets, routers can be broadly characterized as having either a *centralized* or *distributed* forwarding model. In the centralized model, the IP forwarding algorithm, outlined earlier in this chapter, is done in a single processing engine that handles the traffic from all ports. In the distributed model, there are several processing engines, perhaps one per port, or more often one per line card where a line card may serve one or more physical ports. Each model has advantages and disadvantages. All things being equal, a distributed forwarding model should be able to forward more packets per second through the router as a whole, because there is more processing power in total. But a distributed model also complicates the software architecture, because each forwarding engine typically needs its own copy of the forwarding table, and thus it is necessary for the control processor to ensure that the forwarding tables are updated consistently and in a timely manner.

Another aspect of router implementation that is significantly different than that of switches is the IP forwarding algorithm itself. In bridges and most ATM switches, the forwarding algorithm simply involves looking up a fixed length identifier (MAC address or VCI) in a table, finding the correct output port in the table, and sending the packet to that port. We have already seen in Section 4.1.4 that the IP forwarding algorithm is a little more complicated than that, in part because of the need to decide whether a particular IP address is directly reachable out an interface of this router or whether the packet needs to be sent to another router. We also saw that the relevant number of bits that need to be examined when forwarding a packet is not fixed but variable, depending on whether the address is question is from a class A, B or C network. As we will see in the next section, the situation is even more complicated in today's Internet, where "classless" addressing is the norm, and the number of bits that must be examined to make the forwarding decision can be anything from 1 to 32 bits.

Because of the relatively high complexity of the IP forwarding algorithm, there have been periods of time when it seemed IP routers might be running up against fundamental upper limits of performance. However, as we discuss in the Further Reading section of this chapter, there have been many innovative approaches to IP forwarding developed over the years, and at the time of writing there are commercial routers that can forward 40 Gbps of IP traffic *per interface*. By combining many such high performance IP forwarding engines with the sort of very scalable switch fabrics discussed in Section 3.4, it has now become possible to build routers with many Terabits of total throughput. That is more than enough to see us through the next few years of growth in Internet traffic.

Another technology of interest in the field of router implementation is the *network processor*. A network processor is intended to be a device that is just about as programmable as a standard workstation or PC processor, but that is more highly optimized for networking tasks. For example, a network processor might have instructions that are particularly well suited to performing lookups on IP addresses, or calculating checksums on IP datagrams. Such devices could be used in routers and other networking devices (e.g. firewalls).

One of the interesting and ongoing debates about network processors is whether they can do a better job than the alternatives. For example, given the continuous and

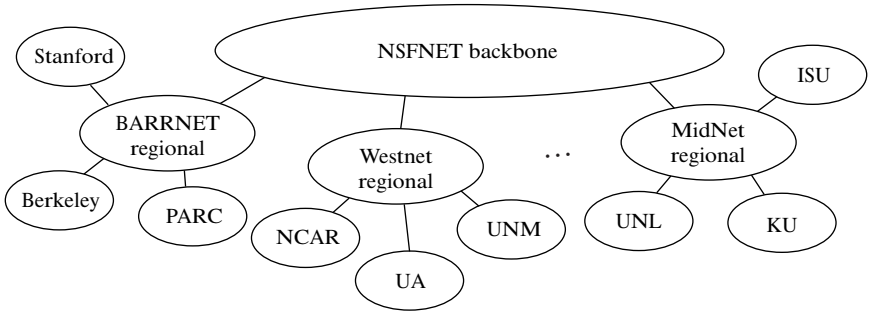

Figure 4.24: The tree structure of the Internet in 1990.

remarkable improvements in performance of conventional processors, and the huge industry that drives those improvements, can network processors keep up? And can a device that strives for generality do as good a job as a custom-designed ASIC that does nothing except, say, IP forwarding? Part of the answer to questions like these depend on what you mean by "do a better job". For example, there will always be tradeoffs to be made between cost of hardware, time to market, performance, and flexibility—the ability to change the features supported by a router after it is built. We will see in the rest of this chapter and in later chapters just how diverse the requirements for router functionality can be. It is safe to assume that a wide range of router designs will exist for the foreseeable future and that network processors will have some role to play.

4.3 Global Internet

At this point, we have seen how to connect a heterogeneous collection of networks to create an internetwork and how to use the simple hierarchy of the IP address to make routing in an internet somewhat scalable. We say "somewhat" scalable because even though each router does not need to know about all the hosts connected to the internet, it does, in the model described so far, need to know about all the networks connected to the internet. Today's Internet has tens of thousands of networks connected to it. Routing protocols such as those we have just discussed do not scale to those kinds of numbers. This section looks at a variety of techniques that greatly improve scalability and that have enabled the Internet to grow as far as it has.

Before getting to these techniques, we need to have a general picture in our heads of what the global Internet looks like. It is not just a random interconnection of Ethernets, but instead it takes on a shape that reflects the fact that it interconnects many different organizations. Figure 4.24 gives a simple depiction of the state of the Internet in 1990. Since that time, the Internet's topology has grown much more complex than this figure suggests—we present a more accurate picture of the current Internet in Section 4.3.3 and Figure 4.29—but this picture will do for now.

One of the salient features of this topology is that it consists of "end user" sites (e.g., Stanford University) that connect to "service provider" networks (e.g., BARR-