Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

 Lecture 3: Scaling IP for Size and Speed September 12, 2002

Overview. The goal of this lecture is to explain the techniques used in the network layer of the Internet that allow the architecture to scale to millions of networks and hundreds of millions of connected hosts.

1 Address structure

The fundamental reason for the scalability of the Internet's network layer is its use of *topological addressing.* Unlike an Ethernet address that is location-independent and always identifies a host network interface independent of where it is connected in the network, an IP address of a connected network interface depends on its location in the network topology. This allows routes to IP addresses to be *aggregated* in the forwarding tables of the routers, and allows routes to be *summarized* and exchanged by the routers participating in the Internet's routing protocols. In the absence of aggressive aggregation, there would be no hope for scaling the system to huge numbers; indeed, this is challenging enough with the techniques we use today for this. This lecture discusses the most important of these techniques and explains how they work.

Because an IP address signifies location in the network topology, the Internet's network layer can a variant of classical *area routing* to scalably implement the IP forwarding path.

1.1 Area routing hierarchy

The simplest form of classical area routing divides a network layer address into two parts: a fixed-length "area" portion (in the most significant bits, say) and an "intra-area" address. Concatenating these two parts gives the actual network address. For example, one might design a 32-bit area routing scheme with 8 bits for the area and 24 bits for the intra-area address. In this model, forwarding is simple: If a router sees a packet not in its own area, it does a lookup on the "area" portion and forwards the packet on, and conversely. The state in the forwarding tables for routes not in the same area is at most equal to the number of areas in the network, typically much smaller than the total number of possible addresses.

One can extend this idea into a deeper hierarchy by recursively allocating areas at each level, and performing the appropriate recursion on the forwarding path. With this, in general, one can define "level 0" of the area routing hierarchy to be each individual router, "level 1" to be a group of routers that share a portion of the address prefix, "level 2" to be a group of level 1 routers, and so on. These levels are defined such that for any level i, there is a path between any two routers in the level $i-1$ routers within level i that does not leave level i.

There are two reasons why this classical notion of area routing does not work in practice:

1. The natural determinant of an area is administrative, but independently administered networks vary widely in size. As a result, it's usually hard to determine the right size of the "area" field at any level of the hierarchy. A fixed length for this simply does not work.

2. Managing and maintaining a carefully engineered explicit hierarchy tends to be hard in practice, and does not scale well from an administrative standpoint.

1.2 Applying area routing: Address classes

The second point above suggests that we should avoid a deep hierarchy that requires manual assignment and management. However, we can attempt to overcome the first problem above by allocating network addresses to areas according to the expected size of an area. This was done in the Internet (in IPv4, with 32 bit addresses), and resulted in the so-called classful addressing structure. In this approach, areas were allocated in three sizes: *Class A* networks had a large number of addresses, 2^{24} each, and an address was defined as "Class A" if its first bit was "0". *Class B* networks had 2^{16} addresses each and were defined as all addresses beginning with "10". In analogous fashion, *Class C* networks had 2^8 addresses each and a Class C address was defined as one that started with "110".[1]

This approach allows areas to be of one of three sizes, depending on the expected size of the network. The forwarding path is a little more involved, but still straightforward: A router determines for an address not in its area which class it belongs to, and performs a fixed-length lookup depending on the class.

1.2.1 The problem with class-based addressing

The class-based addressing approach served the Internet well for several years, but ran into scaling problems some years ago as the number of connected networks started growing dramatically. The problem is *address depletion*—available addresses started running out. It's important to understand that the problem isn't that the entire space of 2^{32} addresses (about 4 billion) started running out, but that the class-based network address assignment started running out. This is the result of a fundamental inefficiency in the coarse-grained allocation of addresses in the Class A and (often) Class B portions of the address space.[2]

1.2.2 Solution 1: CIDR

One solution to this problem is to get rid of the class-based addressing structure and the corresponding fixed-length class-dependent lookups on the forwarding path. The IETF developed this method, called *Classless Inter-Domain Routing (CIDR)* (pronounced "Cider" with a short emphasis on the "e"), in an interesting piece of opportunistic and rapid engineering. In this approach, each network gets a portion of the address space defined by two fields, A and m. A is a 32-bit number (often written in dotted decimal notation) signifying the address space and m is a number between 1 and 32. If a network is assigned an address region denoted A/m, it means that it gets the 2^m addresses all sharing the first $32 - m$ bits of A. For example, the network "18.31/16" corresponds to the 2^{16} addresses in the range $[18.31.0.0, 18.31.255.255]$.

[1] This does not exhaust the IP address space; we'll see later that "Class D" addresses beginning with "1110" are used for IP multicast, and Class E addresses are as yet unused.

[2] For instance, MIT and Stanford each had entire Class A's to themselves. MIT still does!

1.2.3 Solution 2: IPv6

The long-term solution to the address depletion problem is a new version of IP, IPv6. We will discuss this in Section 3 (and T3 notes) after we establish how well IPv4 can be made to scale in the next few sections.

1.3 CIDR lookups: Longest prefix match

The forwarding step with CIDR can no longer be based on determining the class of an address and doing a fixed-length match. Instead, a router needs to implement a *prefix match* to check if the address being looked-up falls in the range A/m for each entry in its forwarding table.

A simple prefix match works when the Internet topology is a tree and there's only one shortest path between any two networks in the Internet. The Internet is not a tree; many networks *multi-home* with multiple other networks for redundancy and traffic load balancing (redundancy is the most common reason today). Figure **??** shows an example of this.

The consequence of having multiple possible paths is that a router needs to decide on its forwarding path which of potentially several matching prefixes to use for an address being looked-up. Be definition, IP (CIDR) defines the correct address as the *longest prefix* that matches the sought address. As a result, each router must implement a *longest prefix match (LPM)* algorithm on its forwarding path.

2 Fast LPM

LPM is not a trivial operation to perform at high speeds of millions of packets (lookups) per second. For several years, the best implementations used an old algorithm based on Patricia trees, a trie data structure invented decades ago. This algorithm, while popular and useful for lower speeds, does not work in high-speed routers.

To understand the problem of high-speed lookups, let's study what an example high-speed Internet router has to do today. It needs to handle minimum-sized packets (e.g., 40 or 64 bytes depending on the type of link) at speeds of about 10 Gbits/s, which gives the router a fleeting 32 ns (for 40-byte packets) or 51 ns (or 64-byte packets) to make a decision o what to do with the packet!

Furthermore, a high-speed Internet router today needs to be designed to handle on the order of 250,000 forwarding table entries, maybe more (today's Internet backbones appear to have around 120,000 routes).

30–50ns per lookup is difficult to achieve without being clever. One can't really store the forwarding tables in DRAM since DRAM latencies are on the order of 50ns, and unless one has a huge amount of memory, doing an LPM in one lookup is impossible.

We are left with SRAM, which has latencies (on the order of 5 ns) that are workable for our needs. Furthermore, over the past couple of years, SRAM densities have approached DRAM densities, allowing router designers to use SRAM for largish forwarding tables.

We can formulate the forwarding table problem solved in Internet routers as follows. Based on the link speed and minimum packet size, we can determine the number of lookups per second. We can then use the latency of the forwarding table memory to determine M, the maximum number of memory accesses allowed per lookup, such that the egress link will remain fully utilized. At the same time, we want all the routes to fit into whatever amount of memory (typically SRAM), S, we can afford for the router. The problem therefore is to maximize the number of routes that fit in S bytes, such that no lookup exceeds M memory accesses.

This problem has been studied by many researchers in the past few years. The paper by Degermark et al. describes one such scheme (the "Lulea" scheme), which uses aggressive compression to fit as many routes as possible into S bytes. Although their paper was originally motivated by software routers, these are the same optimizations that make sense in fast hardware implementations of the IP forwarding path.

The Lulea scheme observes that a forwarding table viewed as a *prefix tree* that is complete (each node with either 0 or 2 children) can be compressed well. The scheme works in three stages; first, it matches on 16 bits in the IP address, and recursively applies the same idea to the next 8 and the last 8 bits.

The paper describes the details of the scheme; we also went through a detailed explanation in class.

3 IPv6

See also Jacob Strauss' notes from T3 on IPv6.

IPv6 is the next version of IP. Its development is motivated by several shortcomings and problems with IPv4. The main problem that IPv6 solves is the address space shortage problem.

Its design challenges, and how it achieves these goals include:

1. Incremental deployment. No more flag days are possible on an Internet with over tens of millions of nodes. Painstakingly engineered to coexist and seamlessly transition from IPv4.
2. Scale. It has been designed for a scale much larger than today's Internet. The main feature that enables this is a *huge* 128-bit address space.
3. Easy configuration. Attempts to reduce manual network configuration by an autoconfiguration mechanism. Here, end node interfaces can pick an address automatically using a combination of provider prefix and local hardware MAC address.
4. Simplification. Gets rid of little used stuff in the IP header, such as fragment offsets, etc. Even though addresses are 4X bigger, the base header size is only doubled from IPv4.
5. Improved option processing. Changes the way options are encoded to enable efficient forwarding. Options (e.g., security options) are placed in separate extension headers that routers can easily ignore.
6. Authentication & privacy. Network-level security is built-in for applications that desire it.

3.1 Addresses & Flows

IPv6 addresses are 128 bits long. The paper describes the details of how they are organized. An important point to note is that most wide-area communication is expected to use *provider-based* addresses, where contiguous address blocks are allocated to Internet service providers (ISPs) rather than directly to organizations. [What does this mean when an organization changes its ISP?]

IPv6 also has *local-use* addresses that don't have global scope, with link-local (e.g., same LAN) and site-local (within the organization) scope. These allow intra-organization communication even if disconnected from the global Internet.

To enable incremental deployment, IPv6 addresses can contain embedded IPv4 addresses. IPv6 also supports *anycast*, where routers will route a packet to the closest (defined in terms of the metric used by routers, such as hop count) of many interfaces that answers to that address. Anycast addresses aren't special in IPv6; they come from the space of unicast addresses.

Flows are an important useful addition to IPv6. You can think of a flow as a TCP connection or as a stream of UDP traffic between the same host-port pairs. IPv6 explicitly formalizes this notion and makes flow labels visible to routers and network-layer entities in the IP header. Flow labels are chosen randomly by end-hosts to enable good classification by network entities.

3.2 Fragmentation

IPv6 does away with network-layer fragmentation and reassembly. End-points are expected to perform path-MTU discovery. Locally, links are free to perform fragmentation and reassembly if they so desire. All IPv6 networks must handle an MTU of at least 1280 bytes.