Layers With Relaying (The Traditional Network Layer)

Introduction

We have now come to the stage in the IPC Model where we consider how to increase the scope of a layer by relaying. We have previously seen how relaying arises in the IPC Model, where it is initially used to overcome the combinatorial increase in cost when there is a line from every host to every other host. In addition, there are limitations on physical media as to how far a signal can travel without regeneration. Here we will consider the issues incurred by relaying. As we have already noted in the IPC Model, relaying can introduce memory errors during the store, and forward operation and congestion can occur as well. We will consider the issues of error and flow control in the presence of relaying later.

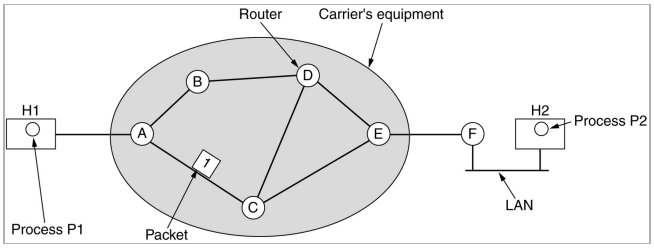
Traditionally, relaying has been considered a separate layer. It really isn't but let us treat it that way to better follow the textbook (Chapter 5) and to understand current practice. To consider the Network Layer design issues surrounding store and forward packet switching, the following topics need to be discussed:

- services provided to the Transport Layer
- implementation of a connectionless ~~service~~ or
- Implementation of connection-oriented ~~service~~.
    - As we will see, these are not services but functions of the layer, i.e., the difference is not visible at the layer boundary.
- Comparison of the two: virtual circuit and datagram subnets.

Read

Read Tanenbaum Chapter 5, Section 5.1, pp. 360–366.

Store-and-Forward Packet Switching



The environment of the network layer protocols

This is Tanenbaum's and the Internet's characterization of the Network Layer. It correctly distinguishes what may belong to a carrier or ISP and what belongs to the customer. Of course, today there could well be more relaying equipment that belongs to the customer. In fact, in some cases the entire network may have a single owner. As you know, today it would be common for the "customer" to have their own extensive network, even when the customer is a homeowner.

There are two more subtle points in this figure.

- The first is that each host has a single connection to the network. This seems obvious enough, but it will turn out to have major implications. Hosts may have more than one connection to different routers.
- The second is that the hosts are "outside" the network. This view is central to the ITU and Internet models. However, as we have seen, the host also has a Network Layer that communicates with a Network Layer in routers and can participate in the network. In addition, as we will see later, one can have networks consisting only of hosts relaying among themselves and no routers. A model/design that recognizes such things as degenerate cases is far preferred to one that considers such things as special cases.

The last and most important characteristic is that these are packet-switching networks. We touched on this and its significance in Module 1. Below, we will consider this in greater detail.

Service Provided by the Network Layer

Tanenbaum discusses the services offered to the Transport Layer. Some are correct, some are not. First and foremost, the service provided by the Network Layer should be independent of the network technology. The layer above should not be aware of the number, type, or graph of the routers. (Tanenbaum says "topology," but this is not a topology. In this course, we will distinguish the "graph" of the network (the graph formed by the nodes and lines) and rarely have cause to talk about the "topology" of the network (a mapping between sets that maintains an invariant). Although we may speak of the topology of sets of networks.) As we have seen, the Network Layer must meet the minimum requirements of the Quality of Service (QoS) expected by the layer above.

Tanenbaum further says that the Network Addresses should be a uniform numbering plan (a good idea) and available to the Transport Layer. As we have seen from the IPC Model, addresses are not available above the layer boundary for lots of good reasons, not the least of which is security. However, the IPC Model, in effect, treats the Network and Transport Layers as intimately involved with each other. So, this is acceptable.

He then adds, "even across LANs and WANs." This is a pretty odd comment. LANs are in the Data Link Layer and only visible to the Network Layer as a service. Not distinguishing the nature of the network technology is the primary purpose of the

Network Layer, so why this comment? This questions his understanding of layers.

Virtual Circuits vs. Datagrams

The textbook then launches into a discussion of the great connection/connectionless debate and makes the common mistake of characterizing it as a debate about a minimal unreliable service and a reliable service connection-oriented service. Let us first describe both of these technically and then look at what is really going on here.

Why not? Consider the following:

The user of the layer requests a flow with a given QoS. The layer must determine how to provide that QoS. If the layer has a level of service with the destination from the layers below that meets the requested QoS, it might use connectionless. (Why do anything more than necessary?) If not, it would use more robust mechanisms to meet the QoS requested. What are the mechanisms that make it more robust? As we have seen, they are data corruption detection, and the feedback mechanisms, retransmission and flow control.

As we found earlier, there has been (and to some degree continues) to be a huge and intense war between the virtual circuit and connectionless advocates. The debate has centered around two major issues: reliability as noted by Tanenbaum and the amount of state required in the network. As it turns out, neither of these is really the core of the difference.

How so? To some degree, because we came in in the middle and didn't understand why packet switching was such a major improvement. It obviously was an improvement that was a *fait accompli*, whether virtual circuit or connectionless. That it was we understood just not why and how much. (Yes, in some sense we weren't as smart as we thought we were.)

What was the big innovation about packet switching? Earlier it was stated that for some it wasn't. That it depended on how old you were at the time. For those older with telecom backgrounds, the idea of breaking up messages into small packets was a radical idea. On the surface, it seemed to require more overhead and thus be less efficient, but that was only part of it. For those a little younger whose background was computing, packet switching was obvious. (Given the problem of communicating between two computers, it was clear data would have to be read into a buffer and then sent. The buffer was the packet. What could be more obvious!) As it will turn out, quite by accident those without a telecom background had the background for the new paradigm without the baggage of telecom. But why was it such an innovation to those with a telecom background? But at the same time, we had a missed something important by coming in in the middle.

To really understand why packet switching was such an innovation, we need to consider what preceded

it. Before packet switching, there was *message-switching*. In message switching, each message was transferred in its entirety (regardless of its length) to the next node in the network, stored there before being passed on to the next node and so on, eventually reaching the destination. This was analogous to what in operating systems would be first-come-first-served batch scheduling, executing one program at a time in the order of arrival. If a small message arrived after a long one, the long one had to complete before the short one could be sent. Delivery time for a message (in terms of our OS analogy, the job completion time) could be quite long for a message queued behind long messages. This also ignored the fact that if the message was long, it was probably being read in shorter increments (into buffers), sent (but again, given the slow speeds, buffers could be sent essentially continuously) and reassembled into the original message, e.g., a file.

The Vagaries of Message Switching

Once when asking why the military thought they needed “flash override” in the ARPANET, I was told of an incident in the DoD’s AutoDin I message switching system, where a radioman had started retrieving a long memo at 70 baud (taking a very long time), when the Admiral needed to get a critical message to the ship. This is what could happen with message switching. So, I asked, “does it make a difference if the admiral’s message takes 500ms or 250ms?” No. “That is what we are talking about.” Okay.

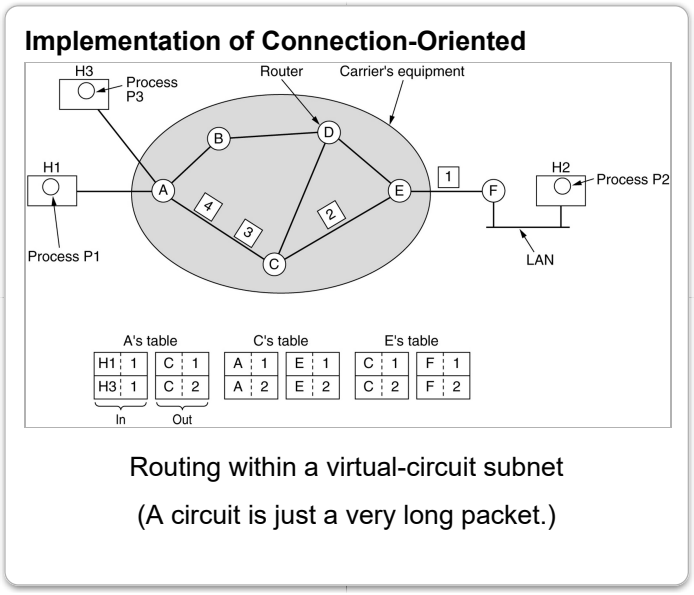
Paul Baran was one of the earliest to have the insight that data traffic was naturally bursty and didn’t need to look like continuous voice traffic. It would be more effective (and less delay) for shorter messages. This would allow interleaving packets from different messages. The shorter messages would finish sooner than the longer ones. While short messages would still be delayed, they wouldn’t be delayed as long, analogous to scheduling in a multiprogramming operating system. They would have shorter “completion times.”

As we go forward and describe what will turn into a major conflict, it is important to keep in mind that there were two very different views of applying packet switching from the outset: 1) the phone company view intent on replicating how the phone system worked, i.e., linking segments of circuits from source to destination; and 2) a computing view that saw parallels to operating systems and were intrigued by ideas of distributed computing and non-deterministic approaches that leveraged the statistical nature of traffic. In the ARPANET, there were immediate comparisons to InterProcess Communication in 1972 and before.

It is also worth noting that when ARPA proposed building a computer network, *all of the experts* unanimously agreed that packet switching would definitely never work. It was only after the ARPANET became operational, worked and worked well, that suddenly everyone was trying to build a packet switching data network.

With this in mind, let us consider technically both virtual circuit and datagram and then look at what is really going on here. What exactly is a virtual circuit technology?

Virtual Circuits



Routing within a virtual-circuit subnet

(A circuit is just a very long packet.)

In the 1970s, the telephone companies did what they knew by simply replicating physical circuit switching in software using packet switching. The classic example of a virtual circuit solution was ITU-T Recommendation X.25 (see sidebar). When a connection is requested, the management software in the switches allocates a path through the network for that request as a sequence of channels, hop-by-hop. Each end of a channel is distinguished by a local channel-id. The PDUs carry these smaller channel-ids, rather than much longer addresses. All packets sent on the connection follow the same path. Because the channel-ids were only locally unique, they are swapped as each node relays a PDU to the next hop. The VC is a sequence of these channels. (See the table in the figure above.) Given the low-speeds of lines and the small packet sizes, e.g., 512 bytes or less was typical, using the much smaller channel-ids was more efficient and a major advantage over using full addresses.

What is This X.25?

X.25 was the PTTs' first attempt at a modern data network protocol. Strictly speaking, it defined the interface between a host and a network switch (DTE to DCE, see Module 1). It specified three layers: the physical layer, which was X.21; a Data Link layer protocol called *LAPB*, a variant of HDLC; and a Network Layer called the *Packet Layer Protocol* (PLP) that basically defined a connection-oriented protocol like the figure above. LAPB used a CRC to detect corrupt PDUs and fixed window flow control with retransmissions. PLP did lost and duplicate detection with retransmissions as well as fixed window flow control hop-by-hop. Supposedly it only defined the interface to the network, not how the network worked internally, but the networks of many PTTs did work that way internally.

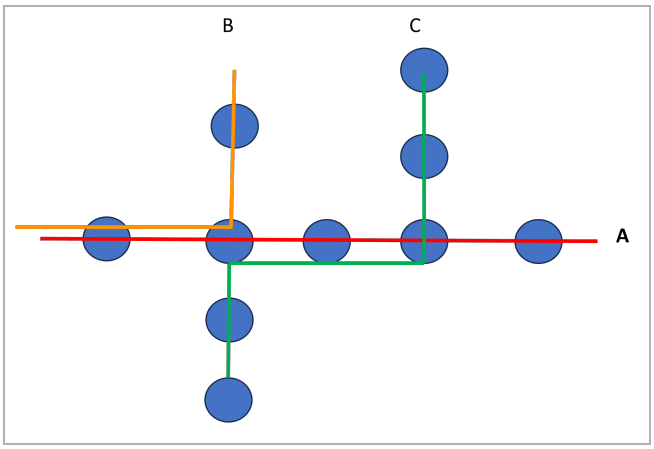
Packet switching with virtual circuits solved the problem with message switching. On any link, there could be multiple virtual circuits sharing the line from one node to the next. Of course, they may not share the entire path, but they could. With packet switching, the node has the ability to interleave PDUs from different channels on the line. This meant that short messages would not be blocked as long behind long messages. To use the OS analogy, just as an operating system gives the illusion of running multiple processes at the same time (even though only one was running at any instance); packet switching gives the illusion of multiple virtual circuits sharing the capacity of the layer at the same time (even though only one packet is being sent at any instance in time). When a virtual circuit is allocated, it requests its data rate (capacity). The layer translates the request into the priority for scheduling sending packets for this virtual circuit that will allocate the requested data rate to the channel. This is analogous to a round-robin with priority scheduling policy. The shorter messages would be delayed less but would have a shorter "completion time."[1]

> In Module 1, when we "derived" networking from IPC and at the step with multiple applications between two computers, we had to introduce a multiplexing task to "manage multiple users of a single resource, the wire." This is that, occurring in each relay. At that time, I remarked that this was the definition of what an operating system does. In this case, scheduling. Here we see it.

Consequently, virtual circuits are analogous to contiguous memory allocation. In operating systems, we know contiguous memory allocation can create memory fragmentation, i.e., a situation where there is sufficient free memory to execute another program but no blocks big enough to use.

Can something analogous happen in networks? Yes. For example, even though, allocating a VC allows other VCs to "share" links in the network, a high-capacity VC would limit the number of virtual circuits that could share some or all of the path. There could be a situation where flows that shared some links, prevented sharing unused capacity in other links blocking the creation of a VC. (see the figure below). Even if overbooking is allowed, sooner or later something like this can happen. It isn't congestion, more like getting a busy signal, because the virtual circuit couldn't be allocated.

Okay, but what about reliability?



Above there is part of a network with virtual circuits, A, B, and C. A (red) is a high-capacity virtual circuit, with B (orange) and C (green) being lower-capacity but still significant. We would like to allocate another virtual circuit, D, like C and parallel to it. There is plenty of capacity on either side, but requirements imposed by static allocation does not leave enough capacity on the links shared by the virtual circuits A, B, and C to allocate D. This is the analog to memory fragmentation with contiguous memory allocation.

Early VC technology, such as X.25, closely followed physical circuit switching, it statically allocated resources to each VC and limited the data rate to the capacity of the VC, thus avoiding congestion while the hop-by-hop error control of X.25 gave the illusion of reliability. However, this meant that X.25 couldn't leverage the statistical nature of the traffic across VCs. PDUs may be lost during relaying because of either rare memory errors. The PTTs prided themselves that their networks didn't lose data, but of course they did. Static allocation required considerably more resources, as we saw earlier with the Denning result.

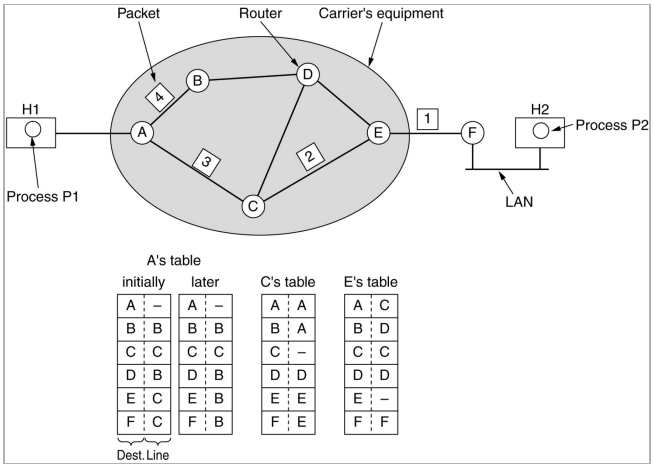
Virtual circuits are quite brittle. If any link or switch fails, all data in transit on the VCs passing is lost. This could be a substantial amount of data, not just a PDU or two. The virtual circuit would have to be re-created from scratch, which could take considerable time and whatever the application was doing would have to be recovered and restarted. These technologies required an additional Application Layer protocol to operate end-to-end between the applications to do checkpoint recovery of long (or not so long) transfers. This 'checkpoint' protocol had to do double duty and recover from transient data loss (like a Transport Protocol) and much larger losses when a virtual circuit failed. Hence one got the worst of both worlds. But by putting the recovery in the Application Layer, the PTTs preserved their business model.[2]

There is no requirement that a virtual circuit technology will be more reliable. Later ones, such as Frame Relay, ATM or MPLS, only provide error detection code but no retransmission or flow control.[3]

Jumping ahead a bit, does connectionless provide these functions? Connectionless technologies usually provide data corruption detection, but as we have seen rely on the Link Layer to provide a minimum level of reliability and a higher layer, traditionally the Transport Layer, to provide recovery from errors incurred during relaying end-to-end.

However, the origins of datagrams or connectionless was entirely different. In fact, it wasn't even supposed to be a solution but a tool to find a solution.

Datagram or Connectionless



Routing within a datagram subnet.

In the 1960s, Donald Davies at National Physical Laboratory (NPL) in the UK had independently discovered packet switching. Quite by accident at a conference, a British officer pointed him at Baran's work. The difference between Baran's and Davies' work is striking. While Baran was focused on survivability for military applications, Davies was not all concerned with military applications but was interested in the potential for packet switching to be more efficient than message switching, Davies had been exploring the problem of packet switching, proposing various ideas to explore, such as isorhythmic routing, or what would constitute a minimum packet with the least assumptions, etc. However, Davies didn't have the chance to explore these ideas. Two things got in his way: By this time, he was promoted to be the director of NPL which, of course, entailed considerable administrative duties that didn't leave much uninterrupted time to for exploring new ideas, and the British Post Office (PTT) was pressuring NPL to just pursue a "practical" virtual circuit solution. Research could wait.

However, in 1971, there was a young researcher at IRIA[4] in France looking for a new project who had no administrative duties to distract him.[5] After a visit to the US to tour the ARPANET, Louis Pouzin[6] initiated the CYCLADES project to design and develop a network to do research on networks. To do that, CYCLADES (like Davies) proceeded to first understand what constituted the minimal assumptions for a network, i.e., the

normal research method,[7] and then planned to carefully introduce additional constructs as needed. This would let them understand why those additions were necessary, how well they fit as well as any implications for their initial assumptions. Pouzin remembered hearing about Davies' idea of a minimal PDU, where each PDU was routed independently. CYCLADES adopted it as a fundamental building block and dubbed it a datagram. With datagrams, there was no concept of flows. PDUs found their way through the network. (Following their method, flows were not minimal, but a construct to be built from the minimal elements of the network.)[8] This opened up new possibilities for concepts of flows/connections that weren't as deterministic, as rigid as virtual circuits that could better leverage the statistical burstiness of the traffic and have emergent properties. Taking their approach eld out the possibility for a flow construct that would be simpler, more elegant, i.e., not be special cases. This emphasized not only dynamic allocation of the lines, avoided the problems with "contiguous" allocation, but also left open dynamic (pooled) allocation of buffers in the routers.[9]

Alone datagrams were not sufficient, as we have seen, a means was needed to provide reliability. The CYCLADES team reasoned that no matter how reliable the network was, the hosts would always check to ensure they got their data. The team then concluded that in that case, the network didn't have to be perfect. It only had to make a "best effort," which meant the switches and routers could be less complex, more flexible, and less expensive. Reliability would be ensured by the Transport Layer. The datagram Network Layer could focus on exploring concepts of flows that could leverage the statistical nature of the traffic.[10] That completed the minimal network that they would use for their research program. (And as we saw previously, because CYCLADES assumed hosts were not collocated with the network switches and could be connected to more than one switch, it also solved the internetworking problem by the simple expedient of the Transport Layer being an overlay over multiple networks, as the Network Layer was an overlay over multiple Link Layers.[11]) By the Spring of 1972 all of the pieces had come together and they could then consider what else would be needed.

But instead, they got one of those unexpected outrageous surprises. They realized that for the data applications everyone was developing, *nothing else was needed!*

At least not yet.

They had stumbled onto a far simpler, more elegant network architecture than anyone else was even coming close to (a true complexity collapse) and solved problems the field was only beginning to be aware of! This was incredibly exciting! In fact, this was a whole new way of conceiving networks, a new paradigm (as Kuhn intended the word). (It is worth mentioning that one thing they did recognize immediately that was needed (as Davies had when he proposed the idea) was solving congestion. By happenstance, CYCLADES immediately handed off the congestion problem to a brilliant young PhD student at the University of Waterloo.[12])

Researchers in the United States became aware of CYCLADES breakthrough at a conference in late 1972 but mainly during 1973 as the excitement spread. DARPA's focus had always been on production networks,

not research on network issues. Consequently, they saw the CYCLADES breakthrough as the answer to their issues, but their focus was almost exclusively on the CYCLADES Transport protocol. The DARPA issues of interworking the ARPANET, packet radio and SATNET (satellite) were not as different as first expected and required nothing beyond the CYCLADES approach. But datagram's role as a building block was quickly lost in the shuffle.

Why? The basic idea was exciting enough! It worked for everything everyone was doing. The idea that more might be needed was soon forgotten. As we just saw, CYCLADES was aware of at least the congestion issue and was acting to find a solution. While congestion came up from once in a while on the Internet, it was set aside to such a degree that when congestion collapse occurred in 1986 many were totally surprised.[13] (This seems to be rooted in the misconception still prevalent in the Internet, that congestion is caused by too much load and more capacity would solve the problem.) This is distinctly peculiar. Soon afterward, the ARPANET added a connectionless mode for experimental use.[14] Datagrams quickly became a cause célèbre, and connectionless was adopted by Ethernet and the Internet. CYCLADES that supported internetworking became operational in mid-1973 which put it at the forefront as a contender for the future. The datagram which had been intended as a tool, became a significant alternative to the virtual circuit.

But at this critical moment, CYCLADES' influence was lost. There was much more to the CYCLADES breakthrough than just its Transport Protocol. CYCLADES despite being an unexpected and unqualified success and placing France at the forefront networking. In the late 1970s, the project was shut down, primarily because it was an embarrassment to the French PTT, whose packet networks were far less successful. Consequently, many of the CYCLADES insights never reached fruition.

What about the OS analogy? What do datagrams say about that? Unfortunately, that was one of the casualties of shutting down CYCLADES. Investigation of new concepts of flows never got done. Even though the US was an advocate of connectionless saw datagrams as an end and never pursued datagrams as a building block. To some degree, this was undoubtedly influenced by the intense debate over virtual circuits and datagrams and the bunker mentality that set in.

But now we come back to the virtual circuit vs datagram issue. This set up the high stakes war between the telecom old guard devoted to determinism and virtual circuits against this radical new approach. This led to very heated debates and more than one less than ethical game being played over a period of 15 years.

As pointed out in the Introduction, this was not just a technical disagreement. This incredible breakthrough invalidated the business models of the PTTs and IBM. It was a war between researchers with a few computer companies and the two largest monopolies or near-monopolies on the planet. The war with IBM was a bit different. The new networking was a peer network, not a hierarchical terminal-host network. This was serious. What was becoming clear was that connectionless could leverage Moore's Law to be successful primarily by how the Internet with the deep pockets of the US DoD behind it spread the technology. What was not apparent to either side at the time was that connectionless required 90% fewer resources. This didn't matter to the PTTs. This new model deprived them of their exclusive claim to the high

margin, “value-added services in the network.” In the 1980s, IBM failing to adapt first to advent of minicomputers (and their growth into mainframes) and then to the PC revolution went through a catharsis laying off 10s of thousands and entirely changed its direction. Meanwhile the PTTs were deregulated, and forced to adapt to the Internet approach, but were aided by the fact that the Internet had been in the ITU model all along rather than the new model. To make matters worse, the advocates of the new model of networking had failed to re-educate the networking professors trained in the ITU model. It would be roughly 20 years before requirements arose that would require using datagrams as a building block. By that time, datagram as a building block had been forgotten and become an end carved in stone tablets.

So, what have we learned from this:

1. Packet switching took data communications from being analogous to first-come-first-serve batch processing to round-robin multiprocessing.
2. Virtual circuits are analogous to contiguous memory allocation and can cause the equivalent of external fragmentation. The loss of a line or router can cause considerable loss of data and a long delay in re-creating the virtual circuit if it can be.
3. Dynamic allocation of both lines and router memory enables leveraging the statistical properties of the traffic and is orders of magnitude more efficient than static allocation.
4. Datagrams provide a minimal building block for dynamic allocation.
5. Static allocation is useful as an implicit means of limiting a flow, i.e., when it is desirable to run out of buffers.
6. Datagrams are not an end, but a beginning.
7. Datagrams open the door to flow constructs that are less extreme than virtual circuit.
8. Tools are still needed for modeling bursty traffic.
9. The concept of connection in error and flow control protocols is not related to the datagram vs. virtual circuit debate. That relation is a consequence of feedback mechanisms.
10. Major breakthroughs never come from incremental change. Just as with code, line-by-line optimization may achieve a 25-30% improvement in performance, to achieve a 70 or 80% improvement requires rethinking the problem from a very different perspective. This is what CYCLADES discovered by doing what the problem said and not jumping to conclusions about what networking was.

But how could this happen? How could the benefits have been so completely overlooked? We have touched on some of it. We have seen that probably the majority of the weight wasn’t technical at all but political and economic. We will look at that aspect next.

The Great Connection/Connectionless War

Tannenbaum is correct when he alludes to the debate over connection and connectionless network services. We have already touched on this. The phone companies detested connectionless for several reasons: they

saw these newcomers treading on their turf, they were uncomfortable with non-deterministic solutions, how could they charge for it if they didn't know how much data was delivered?! (At the time, networks charged for both connect time and bytes delivered.) But most of all, they saw it as a threat to their business model.

Because of how connectionless was embedded into the model with an end-to-end transport protocol, it relegated the phone companies to just carrying bits, a low-margin commodity business, which they have been trying to get out from under ever since. 5G is their latest attempt. The phone companies in Europe were very powerful and led the charge against connectionless. Oddly enough, in the 1980s, AT&T supported connectionless, but mainly because their participants in the standards were from Bell Labs and had a broader background than just telecom. (Later, that changed significantly and even they succumbed to the ITU model.)

This battle has raged from about the mid-70s to the present day, with the most intense conflicts in the late '70s and into the '90s. This was first fought in the INWG meetings of 1973–1978, and then they shifted into high gear in OSI in 1978-1990. (The ARPANET and the beginning Internet being entirely in DARPA were for the most part buffered from the debates, and virtually unknown to the companies in the commercial world with the exception of DEC, Data General, and many of the LAN companies. In fact, at the time there was strong support for connection-oriented X.25 within the more operational US Defense Communications Agency.) In OSI, the battle was between the PTTs and the computer companies, which were not exactly united in their support of connectionless. Both sides were convinced that their approach was right. The debates were quite heated with raised voices, pounding tables, even fear that people would come to blows, etc. (This is not exaggeration. They were not pleasant meetings.) This quickly generated a bunker mentality. The connectionless advocates definitely had the weaker position in the sense of market power. In the early 80s, datagrams were mainly still an idea. CYCLADES was a research network, the ARPA switch to IP was still coming, LANs were just beginning to reach the market. The PTTs, on the other hand, had all of the force of governments behind them as well as nearly 100 years existence with a huge installed base. Their position was rock solid: They had already committed to development, of product, they were unwilling to change to anything they were doing. The PTT counter-arguments could not be refuted without an existing network.

By 1978, France, Germany, and the United Kingdom had all produced high-level government reports focused on networking as a major future technology, and considerable funding followed soon after. However, all attempts to build a "European ARPANET," although technically successful, were never able to survive PTT pressures. While European countries championed high-tech and innovation, their actions had precisely the opposite effect. (The US wasn't much better but had some unique conditions that allowed a couple of breakthroughs to occur, but it was more accidental than on purpose.) Europe more or less deferred to the traditional power structure. Interestingly, 30+ years later when the Future Internet effort was in full swing, Europe took the same path they had in the '70s.

In the end, the difference was not technological, but that DARPA was willing to provide far greater subsidies for the Internet (and that AT&T and IBM never saw it as competition). Business was concentrating on the current market: corporate networks. There was no business case yet for an Internet. It was coming but was

several years away. The advent of the Web with a browser (from outside DARPA) accelerated that. The greater subsidies by the DoD and NSF allowed the Internet to achieve critical mass. However, these political and economic subsidies were not accompanied by a commensurate transition from research demo to product, leaving the result lacking critical structures.

But the Europeans insisted on X.25 as the connection-oriented network protocol.[15] After a bitter battle, the United States was able to insert connectionless in the architecture but had to accept constraints that made any interworking of connection mode and connectionless mode impossible. Connectionless could operate over connection mode or vice versa, but there was no means to interwork them as peers. As it turned out, this was less a problem. However, it was enough for the United States to move ahead to develop connectionless routing and data transfer protocols.

The Battle Over Connectionless

As with any consensus organization (standards or legislative), a topic is never really resolved. Issues with enough support can always be raised again. Connectionless is a case in point. The United States voted “No” on the first ballot to move the OSI Reference Model (RM) to a standard in Berlin in 1980. The No vote was conditional, the vote would change to “Yes”, if connectionless issue was resolved to US satisfaction. There was much angst among clueless American commentators who didn’t understand that under ISO rules, a “Yes” vote with comment meant that none of the comments had to be accepted. (You had voted “Yes,” after all.) All leverage was lost. By voting “No” conditionally, leverage was preserved. It was the only way to ensure the comments were addressed. (National bodies with active participation in a standard always voted “No” on condition that major comments were accepted for just this reason.)

That “No” vote was resolved with an agreement to develop a connectionless addendum to the OSI Reference Model. There then ensued three years of meetings, two or three times a year for a week at a time to hammer out a document that added connectionless to the Reference Model. The Europeans contested and argued over every word and phrase, inspected to ensure that it did not give an advantage to one side or the other. Both sides felt that any compromise would lead to a slippery slope and the loss of the argument. This did not facilitate finding a useful middle ground.[16] Finally, in Ottawa in 1983, the document was ready to be progressed to a standard. The chair of the Architecture Working Group found me at lunch in the food court of the Rideau Center and said he wasn’t sure the votes were there for it to pass. As Head of Delegation for the WG, this was my problem. We had an agreement, but that was three years ago. The Chair was sympathetic to the connectionless position (after all, he was one of its creators), but there was only so much he could do as Chair. We had negotiated everything there was to negotiate.

Finally, it came down to no connectionless addendum, no further U.S. participation. It was passed and became an approved standard two years later. However, the

issue was never really resolved and continued to be contested in every meeting.

As the 1980s wore on and the effects of Moore's Law began to be felt, X.25 was becoming less important (supplanted successively by Frame Relay and the promised hype of ATM). It was being relegated to a role as a subnetwork access protocol similar to its role in the ARPANET and Internet in the 1970s,[17] with connectionless operating over it as a subnetwork-independent protocol. The European faith in the ability of centralist initiatives to override consumer demand lost them the war (sound familiar?). The connection/connectionless debate between the computer companies, mainly in the US, and the PTTs, mainly in Europe, combined with the factions within the United States jockeying to dominate the direction of OSI, as well as the effect of egos both national and individual, created so much internal strife that it essentially self-destructed. OSI lost any cohesive market focus. As foreseen in the early 80s, the collaboration with the ITU-T introduced such huge divisions the effort was doomed. This left the Internet as a production network without having ever been updated and productized or had its deep flaws rectified. Flaws!? It worked! What flaws? It was selling well. Then it must be right!

The Internet community (the United States and the international academic community) maintained a fairly pure, parochial connectionless stance outside the battle, while the ITU, European computer companies, and PTTs similarly maintained a fairly pure parochial virtual-circuit stance. European standards stance had a majority connection-oriented bias rooted in the influence of the PTTs and a pragmatic view that the PTTs were part of reality and had to be accepted. (And with no hint of telecom deregulation on the horizon, what else could they think? European computer companies saw IBM, not as a competitor, but as "part of the environment.") The U.S. standards group was primarily connectionless with a connection-oriented tendency by the IBM participants.[18] (It must also be recognized that in the 1970s and 1980s, the split between connection and connectionless was to a large extent, but not entirely, generational. This is no longer the case.[19]) Thus, the protectionist attitudes toward the status quo in Europe ultimately undermined Europe's strategy for economic parity. The connectionless forces were woefully outnumbered. The "solution" was less than satisfactory and, in fact, never really resolved. The Europeans attempted to legislate connectionless either out of existence or constrain it to such a degree as to make it useless. (There are arbitrary statements in the OSI RM limiting the use of connectionless.) The solution in OSI can only be likened to a cease-fire rather than a real solution.

By the same token, this was the forum where the concepts were given the greatest scrutiny. But the scrutiny only led to heated debate and an uneasy truce, not to any deeper understanding. There is only so far one can go in standards. Then it is necessary to step back and let the market decide. In a real sense, connectionless was kept alive from an entirely different quarter, the success of the LAN industry.

The phone companies wanted to eradicate connectionless and not have it at all. They came very close to succeeding. When they were unable to, the fallback position was to place outrageous strictures on connectionless in the OSI Reference Model as to make connectionless OSI useless.[20] There had to be a

resolution. One had to admit that there were use-cases where connections appeared to make sense just as there were use-cases where connectionless made sense. Perhaps it would be useful to look for a synthesis.

Finding a Synthesis

For years we would characterize the difference between connection and connectionless as two extremes of the amount of shared state: connection had a lot of state; connectionless had very little. The trouble was that there didn't seem to be anything in the middle. It was either one extreme or the other. There didn't appear to be anything in between that made any sense. This seemed to indicate that if there was a unifying principle, it must be something else.[21] However, at the time, the intensity of the debate made exploring this direction impossible. Any hint of something like a flow and the PTTs argued it was virtual circuit. Effectively backing the connectionless advocates into the corner with a pure datagram service.

As the text says, many drew the distinction that connection-oriented was reliable, connectionless wasn't. The first years of the debate had concentrated on this point. This argument has already been dispelled above. It was clear that X.25 was not reliable and involved considerable overhead. So, the important distinction wasn't reliable vs. unreliable. Later, it became clear that if one considered the cases where connections made sense and where connectionless made sense, that the distinction had more to do with how heavy the traffic was. This led to the conjecture that:

- As traffic density increases and it becomes more deterministic, connections should be preferred.
- As traffic density decreases and it becomes less deterministic, connectionless would be preferred.

Hence, as we go down in the layers and in toward the backbone, traffic is being combined (multiplexed), traffic gets more deterministic and less stochastic. Connection-oriented should be preferred.

As one moves up and out, traffic spreads out and becomes less dense, so connectionless should be preferred. (Notice how the synthesis is moving toward our characterization above.)

Many have argued that this debate had become a matter of "religion." (And given the intensity with which the views are defended, one might get that impression.) But it wasn't. It was much more visceral than that, it was about money! Traditional phone company people were devout virtual circuit advocates, while Internet people were devout connectionless advocates. And it's also not a question of reliability, which a lot of people will also say. Neither one is what this is all about. Technically, it is the trade-off between *static and dynamic resource allocation*, not reliability or error control.

The important thing to realize about this is it is not necessarily a condition that exists end-to-end. There are intermediate flows in the network. For example, there may not be heavy constant traffic between Lexington, Massachusetts, and Lake Forest, Illinois, but there is constant traffic between the Boston and Chicago metro areas. One would expect that backbone trunks Boston-to-Chicago to be connection-oriented, but the traffic in the "distribution" networks in the Boston and Chicago areas to be connectionless.

But, of course, connectionless does have better resiliency and survivability properties and can make more efficient use of resources, so let's find a synthesis here. It is also interesting to note that these trunk/backbone connections are much longer lived, lasting hours, if not days or weeks.

If traffic is managed so that all of the traffic for the Boston or Chicago metros are on the same lines, why look at every header? The router knows where it is going. Just move it! However, near the periphery where traffic is going to all sorts of different places, looking at each PDU not only makes sense, but it is necessary!

Even the experts on the connectionless side of the argument had not yet seen what was described at the beginning of this section about static versus dynamic resource allocation.

Now the discerning reader will say, “but there was nothing stopping them from using pooled buffers with virtual circuits!” Correct, there wasn’t, other than their pre-conceived intuitions. In fact, even with datagrams, they didn’t see this. Router developers simply moved to allocating buffers to destinations rather than virtual circuits. It took until 2004, almost 40 years after the first router, before someone published results showing that pooled buffers were much more efficient (and 90% of the memory in high-end routers was unnecessary!) Intuitions die hard!

Over the centuries, Science has had a nasty habit of showing us that our intuitions are often wrong! In their defense, Denning’s result is very counter-intuitive and even more counter-intuitive that the difference would be so great! All the more reason that this a lesson that it is wise to keep in mind.[22]

As a corollary, it is also important when reading papers to recognize when results are much more general than just the paper, they are in. Few papers will point it out (and some will for results that are not general (TCP doesn’t support wireless) or just plain wrong (network traffic is self-similar). Developing the sense to know which is which is what we need to cultivate.

Really Datagrams vs. Virtual-Circuits, Not Connection vs Connectionless

In this discussion, we contrasted connection and connectionless. Early on when the emphasis was on reliability, the PTTs claimed the connectionless advocates were being inconsistent: The Transport Protocols created connections. (This argument is still heard today.) This argument assumed that the pairwise synchronization required for the feedback mechanisms was an example of connection-oriented. This is an apples-and-oranges comparison. The intensity of the debate at the time made the subtle distinction between a static virtual circuit and feedback over a datagram service was impossible. An EFCP can operate over either a virtual circuit or datagram network. They aren’t related. What makes virtual circuit connection-like and what makes an EFCP appear connection-like are qualitatively different properties. The distinction is between “virtual circuit and datagrams” and is not related to protocols with feedback.

So, what is our solution to this? As we saw in the early ARPANET, there was a talent for taking what appeared to be oil-and-water problems—total opposites—and finding an elegant synthesis that unified the two efficiently. We saw that in the last module.

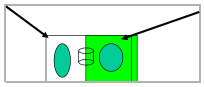
Where did we go wrong? For decades we (me included) had said connectionless is the simple minimal shared state, while connection oriented is the maximum scale shared state, but by this logic it's exactly the opposite!! We were focused too much on the data transfer part and ignoring the rest of what was going on the layer.

Finding a Synthesis Was Hard

Let's see if we can do that for connection/connectionless. Consider the problem *very carefully*. The thought process went like this: Reading an important paper on the topic for the umpteenth time, I began to think along these lines.

> What makes connection-oriented so brittle to failure?
>
> - When a failure occurs, no one knows what to do. You have to go back to the management of the network to find out how to recover and allocate a new path through the network.
>
> Then, what makes connectionless so resilient to failure?
>
> - Everyone knows how to route everything!
>
> Wait just a minute! That means, connectionless isn't minimal state but maximal state! The dumb network ain't so dumb!

Where did we go wrong? For decades we (me, included) had said connectionless is the simple minimal shared state, while connection oriented is the maximum-scale shared state, but by this logic, it's exactly the opposite. We were focused too much on the *data transfer part* and ignoring *the rest of what was going in the layer*.



We needed to look at the whole picture. The total amount of state is about the same although where it is, is different. The amount of replication is different. We have been distributing connectivity information to every node in a layer, but we've insisted on distributing resource allocation information only on a "need to know basis", i.e., connections. Now we need to work out how to do resource allocation more like we do routing. But we will leave that for an advanced course or PhD research.

To recap,

Connectionless is characterized but maximal dissemination of state information and dynamic resource allocation.

Connection oriented mechanisms attempt to limit the dissemination of state information and tends towards static resource allocation.

Applications request the allocation of communication resources; the layer determines what mechanisms and policies to use and tends towards connection-oriented when traffic density is high and deterministic and connectionless when traffic density is low and stochastic. Similarly, PDU size should increase as we move to the backbone. One wants to be forwarding more stuff less often, rather than less stuff more often.

As we saw at the beginning, CYCLADES adopted datagrams as part of their effort to build a network to do research on networks not as an alternative to virtual circuits. They needed to define a minimal packet a network required.

The next step was to see what else was needed to provide whatever the network needed to support. To their surprise, even shock, nothing else was needed. That state of affairs existed for 25 years with help from Moore's Law. When the DARPA community adopted the datagram concept and ran with it, they assumed it was fully formed. There was only the minimal pure datagram (and for some purposes that is still true). The proper use was in the pure form. But that's no longer the case. DARPA took the CYCLADES insight as an end which, as we have seen, it was just the beginning. (How do I know this? Because Pouzin told me this one afternoon in Paris over beers!)

So, what is next? As pointed out earlier, shutting down CYCLADES meant this never got explored. The Internet's focus at first held tight to the pure connectionless solution. So much so, that they advocated (or try to) apply it to everything, even though that was clearly never the intent. Ultimately, they flipped to the other extreme, completely over the ITU virtual circuit solution. As we said before, the work remains to be done. It is beyond the scope of this course, but all indications are showing that there are new exciting results to be learned and operating systems should be our guide.

We have spent considerable time on this debate and many students may think it was a waste of time. We hope not. If you read between the lines, you will notice that this has been greatly abbreviated. Not only is this debate still raging, but you will encounter it. And it is still misunderstood. But even more importantly, when one is out in the real world, these kinds of considerations are as much a part of the job as developing code or managing a network. The topics may not be the same as the one here, but the "games" will be similar. (We will see examples of these "games" happening more than once later in the course.) Here it was an international debate. The same things happen both nationally and within corporations, especially when a company needs to "move on" and adapt to change, change direction. Similar situations will arise during your career, probably more than once.

It is beyond the scope of this course, but we can say that as exploration and testing of the concepts has progressed, it has become more and more clear that virtual circuit becomes a degenerate case of connectionless. Or, one connection advocate who was quite relieved when for a project we had finished working out connectionless routing, remarked, “Good! NOW we can get to the real problem of routing for virtual circuits!” To which the rejoinder was, “Oh we have done that too. A virtual circuit is just a very long packet!”

1. It is hard to believe in retrospect that as late as 1990, Europeans were still expecting X.25 to be the future of networking. How X.25 could have supported the data rates and applications is hard to imagine.
2. Why did IBM support developing connectionless? Simple. IBM supported anything that slowed down the process and gave them more time to adapt.

---

1. It is intriguing how this analogy of message switching to batch OSs (1950s) and packet switching to multiprogramming OSs (1960s) parallels the timeline in operating systems with the network technologies lagging perhaps a few years.
2. Remember it was critical to the PTTs business plan that there is no Transport Protocol, so that they could locate services (applications) “in the network.”
3. In fact, ATM, using 53-byte cells, was shown to drop cells. Making the industry favorite of IP over ATM to fragmenting IP packet over 10s of ATM cells so that a dropped cell was both hard to detect and unrecoverable.
4. Now INRIA, Institut National de Recherche en Informatique et Automatique.
5. There is a lesson here.
6. Earlier Pouzin had worked on Project Mac at MIT and was familiar with the US research community.
7. The “derviation” of networking from IPC in Module 1 is basically following the same approach.
8. It should be noted that these sorts of non-deterministic ideas where the consequences of the structure implicitly yielded the desired result were very much in the air at the time, even the choice of the name, CYCLADES, was a reference to the decentralized nature CYCLADES federation in ancient Greece.
9. Oddly enough, the traditional telecom view of networks was so strong, it took until 2004 for the dynamic allocation of buffers to be recognized, 35 years after Denning.
10. This is admittedly a rather radical view of the datagram/virtual-circuit debate. How do I know it is true? Because the guy who built the CYCLADES CIGALE network confirmed it.
11. Meanwhile, the Internet work was more focused on the differences in technologies, struggling with the problem of translation at the gateways. Indicating they were still thinking in terms of traditional datacom networks, similar to the ITU-T model.
12. There is an odd aberration here. Both Hafner’s book and Abbate’s book note in detail that Robert Kahn was very interested in the problem of congestion in the ARPANET and did a considerable amount of work to demonstrate it. However, there seems to be no indication that Kahn recognized the problem in the datagram model. Perhaps Kahn mistook a potential deadlock situation (which did exist) for congestion but didn’t really see the possibility of congestion in datagram networks, even though it was obvious.

13. A few years ago on the Internet History list, one former MIT expert said of the congestion collapse, “No one saw *that* coming!” (emphasis probably added). He was quickly disavowed of that idea.
14. This just further complicates the confusion over congestion. BBN limited the use of Type 3 (datagram) messages because of concerns about congestion. However, in the run up to the 1983 conversion to IP, nothing was done about congestion in the Internet, and BBN was less than 5 miles from MIT!
15. It is hard to believe in retrospect that as late as 1990, Europeans were still expecting X.25 to be the future of networking. How X.25 could have supported the bandwidths and applications is hard to imagine.
16.
17. Surprisingly, this turned out to be non-controversial (much to the PTTs’ chagrin) because the title of the X.25 Recommendation said it was a subnetwork access protocol, i.e., a DTE to DCE interface.
18. Why did IBM support connectionless? Simple. IBM supported anything that slowed down the process and gave them more time to adapt.
19. A lot was generational during that period with the boomers becoming an influence and initially I thought this was too, but since it has become apparent that some people are just not comfortable with non-deterministic solutions regardless of age.
20. Similarly, when the PTTs couldn’t stop standardizing a Transport Protocol, the backup was to add 4 optional protocols. These are common tactics.
21. This gives the reader an idea of how trying to analyze concepts can get off on the wrong foot. We had come into this in the middle of the argument without considering how packet switching came about (see above). Why? Because most of us were too young to have had seen what went before.
22. At the same time, during this period memory was expensive and very limited as much for the PTTs as everyone else. Yet even given the huge advantages of dynamic allocation, the PTTs refused to adopt it. There is a lesson here as well.

Internetworking

Read

Read Tanenbaum Chapter 5, Section 5.5, pp. 423–431.

Overview

It should be pretty obvious that once the concept of a packet switching network had been shown to work, many of these networks would be built with different approaches and there would be a need to hook them together. Of course, this could be made difficult, because these networks can differ in all sorts of ways.

How Networks Differ

Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

Tanenbaum has a list of some of them in the textbook. It is interesting that with the prevalence of feedback mechanisms and distributed algorithms, the one thing that the textbook does not discuss is the interactions of feedback loops at the boundaries of these networks and the implications this has for stability. Notice that routing is not in Tanenbaum's list. (To some degree this is not surprising. The topic has been missing from the general network research topics as well and is beyond the scope of this course.)

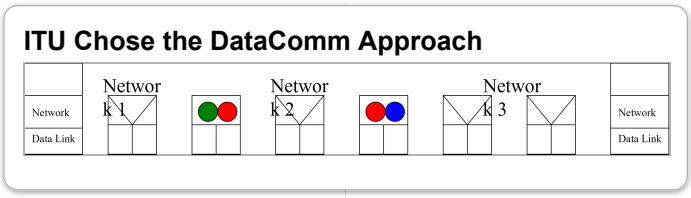
But just as important are the similarities across networks: They all do IPC over a Given Range of Capacity, QoS, and scope.

It was recognized early on that there were two ways to do Internetworking:

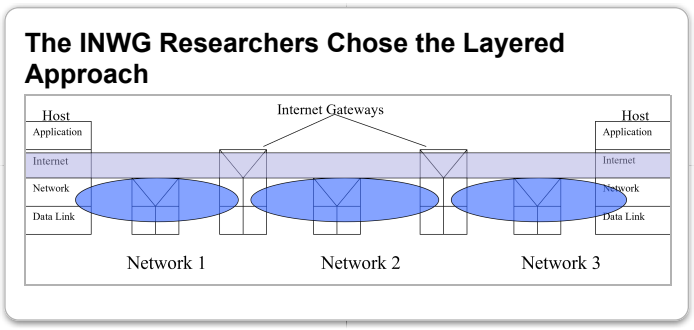
1. **The datacom/ITU approach**. This is the approach used in the phone system – do protocol conversion at the boundaries. This is a bit of a problem if the networks are very different. There is a potential for n^2 translations. As anyone who has ever written emulators for device drivers or similar problems, translation can get very ugly very quickly and never be quite perfect. There are always corner cases that don't quite work right, OR
2. **The layered approach**. This was chosen by the researchers mainly because they were expecting a wide variety of new kinds of networks. With this approach, a common layer is added over the top. This approach changes the n^2 problem into an $nx1$ problem. Each different network would only have to support the common layer, a much simpler problem, which avoids the sticky corner cases. This implies that there's a minimal level of service required by the common layer. This is essentially the same solution we saw in the ARPANET upper layers. One could characterize the common overlay

layer as the “Network Virtual Layer.”

(Tanenbaum credits this to Cerf and Kahn’s 1974 paper, but it was actually first developed by CYCLADES in 1972 and discussed at the second INWG meeting in June 1973. However, by 1980 the Internet had instead moved to the ITU approach and lost the Internet Layer to become a large network with a common address space and protocol conversion at the gateways.)



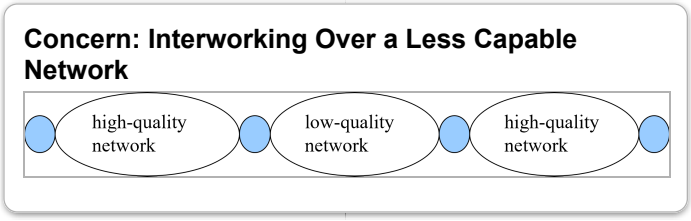
As you would expect, the phone companies chose the datacom approach. Just hook them together and convert one into the other. Basically, this is the same as how the Public Switched Telephone Network (PSTN) works. This was quite reasonable for them. They chose this approach because at the time it retained their familiar beads-on-a-string model and it was also where they wanted to go with future services they planned to offer “in the network.” (Today, we would say “in the cloud.”) Also, they were interconnecting relatively similar protocols, so the n2 problem wasn’t that big an issue. Furthermore, it didn’t require a transport protocol, which would prevent them from offering value-added services “in the network”, i.e., under their monopoly and worse, relegate them to a commodity business.



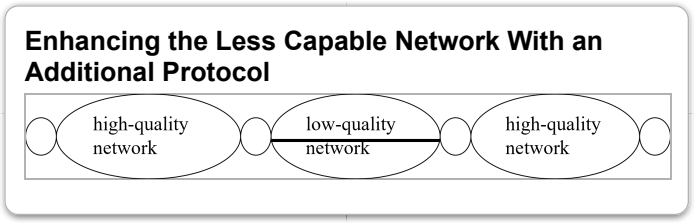
On the other hand, the International Network Working Group (INWG) researchers adopted the overlay model and built a common layer over the top of different networks. The overlay layer requires at least minimal service from the layers below. If the layer below can’t meet that level of service, the lower layer is required to enhance its service to the Network Layer. INWG assumed there would be a wide variety of potentially very different networks, raising the specter of messy, complex translations. Here layers are a locus of distributed shared state of different scope cooperating to provide resource allocation. This is what an Internet should look like. There are gateways between networks, so the Internet Layer is operating between hosts and gateways, while the networks are operating between hosts, routers, and gateways. This is a distributed computing model. Then this knowledge was lost.

A decade later, the OSI Network Layer group was working through the virtual-circuit/connectionless war and had to analyze the Network Layer to figure out how it went together. They knew they would have to

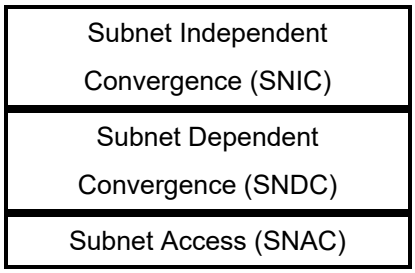
accommodate different networks of different quality and different technologies. One of the big concerns is shown in the figure. (In fact there's a picture like this in the OSI Reference Model.)



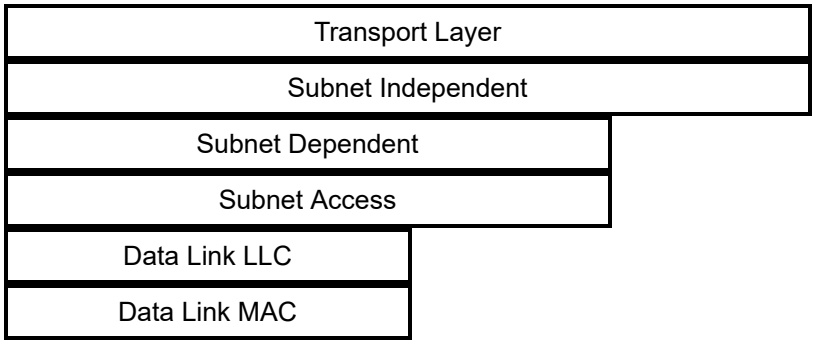
If you had to interconnect two high-quality networks over a low-quality network, you would have to do something over the top of the low-quality network to bring it up to an acceptable level.



To model this, they subdivided the network layer into essentially three different sublayers or roles.



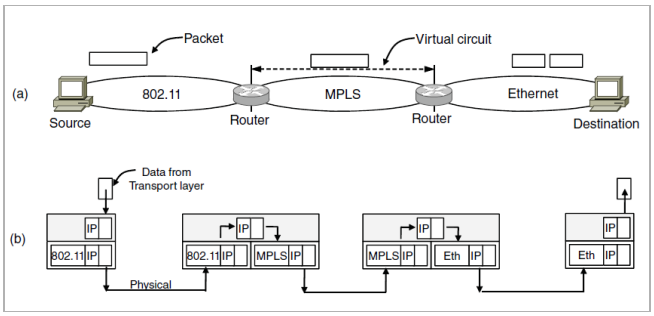
There would be a network layer (Subnetwork Access, SNAC). If a layer needed enhancing, it would require an EFCP-type protocol to enhance it (Subnet Dependent Convergence, SNDC), and then there was the Internet Layer (Subnet Independent Convergence, SNIC). They rediscovered the INWG model 10 years later.



For completely different political reasons, OSI made a similar distinction as TCP/IP between the internet relaying function and the end-to-end Transport function. But two different groups with different participants

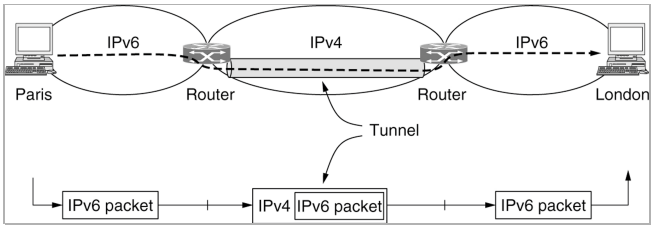
10 years apart came to the same conclusion, a pretty good indication that this is at least on the right track. (I watched both of them come up with the same model.)

How Networks Can Be Connected

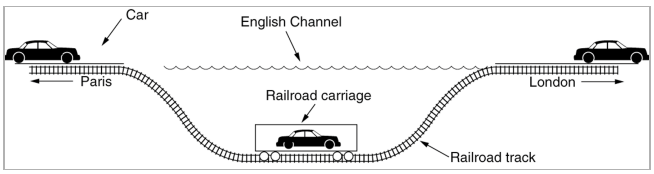


In this figure, Tanenbaum tries to illustrate how the IP forms the common layer and can operate over different networks. However, his examples are flawed. The figures show IP operating over two Data Link Layer protocols (802.11 and Ethernet) and MPLS as a SNAC protocol in the Network Layer. Unfortunately, there isn't room to show relaying and routing within the MPLS network.

Connecting Endpoints Across Heterogeneous Networks



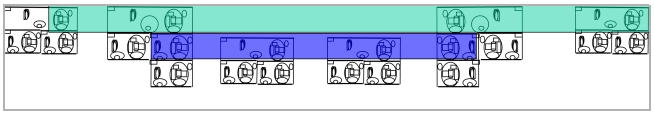
No, actually this is tunneling. Tunneling a packet from Paris to London.



Tunneling a car from France to England. But the Internet doesn't "tunnel" the whole car.

Tanenbaum takes this opportunity to illustrate tunneling of IPv6 over IPv4 and he can't resist an analogy to the Chunnel under the English Channel. This is what the Internet does, and it does a similar thing with IP over MPLS. However, this adds to the complexity of the network and misses an opportunity for simplification and introducing better scaling. If the reader reviews how the overlay layer works above, it will be clear that the problem Tanenbaum is having is that the Internet lost the Internet Layer and is actually a large network.

Connecting Endpoints Across Heterogeneous Networks



Rather than merely encapsulating one protocol in another, there are distinct advantages to having an entirely separate layer of less scope as illustrated here. Notice that in the green layer, the middle two relays are one hop apart in that routing domain. The routing between them is done in a lower layer in a separate routing domain. We will see later how this can be used to advantage.

Fragmentation Across Networks

Read

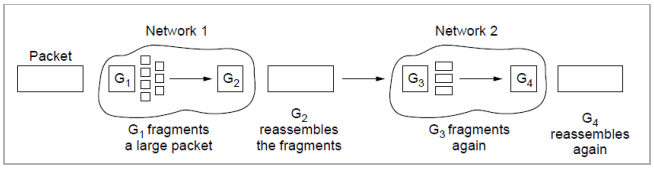
Read Tanenbaum Chapter 5, Section 5.5.6, pp. 431–435.

One minor aspect of internetworking that has caused more consideration than it should have is that different networks have different maximum transmission units (MTU), i.e., different maximum-size packets that they can handle. There are a variety of ways that this constraint arises: the hardware, the operating system, the standards, in reducing re-transmissions, in ensuring fairness, etc. We saw that with a hostile environment such as low speed wireless, shorter PDUs are less likely to have errors than longer PDUs. (At higher speeds, such as 802.11, it is much more likely that electrical noise will obliterate entire PDUs rather than just damage them.)

Transparent and Non-Transparent Fragmentation

There are two approaches to fragmentation: transparent and non-transparent.

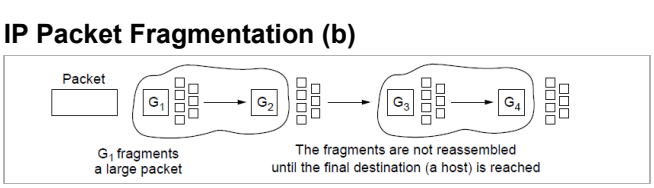
IP Packet Fragmentation (a)



(a) Transparent fragmentation.

With transparent fragmentation, when PDUs reach a network with a smaller MTU, they are fragmented at

the gateway, sent over that network, and the PDUs are reassembled when they exit the network at the next gateway. The drawback of this approach is that all of the fragments have to exit at the same gateway so that they can be reassembled. This may also mean that the PDUs are fragmented and reassembled more than once. It also has the further complication that if the gateway goes down, all of the fragments must be re-routed to the same alternate gateway, which might be hard to do and certainly violates the datagram concept of independent routing of PDUs.



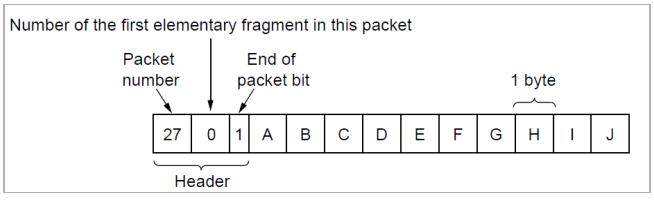
(b) Nontransparent fragmentation.

With non-transparent fragmentation, when PDUs reach a network with a smaller MTU, they are fragmented at the gateway sent over that network, and the PDUs are not reassembled until they reach the destination. This avoids repeated fragmentation/reassembly unless a network is encountered with an even smaller MTU, and it also means that fragments can exit the network at any gateway. IPv4 uses non-transparent fragmentation.

An Example of Non-Transparent Fragmentation

Let's look at an example. We already discussed the common means for doing fragmentation in protocols in the earlier module. This example works pretty much the way IPv4 does, with one exception. For some reason, Tannenbaum has an End-of-Packet bit, where IP has a More Fragments bit. The difference is that the End of Packet bit is 1 if there is no fragmentation, or 0 for all fragments and 1 for the last one. Where for IP, the More Fragment bit is 0 if there is no fragmentation or 1 for all fragments, and 0 for the last one. While this is how IP does it, the example has this slight difference. Why Tannenbaum would introduce this difference on such a trivial point is beyond me.

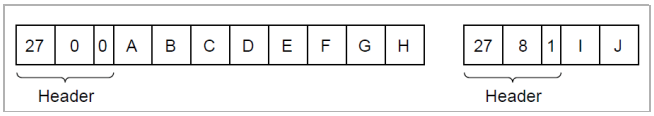
For the example, we start with a PDU that has a Packet-id of 27; an offset of 0; and the End bit=1. It is not fragmented; hence it is both the first and last PDU, with 10 bytes of data.



Fragmentation when the elementary data size is 1 byte.

(a) Original packet, containing 10 data bytes.

This PDU is sent, and at a gateway, it is fragmented into 2 PDUs.

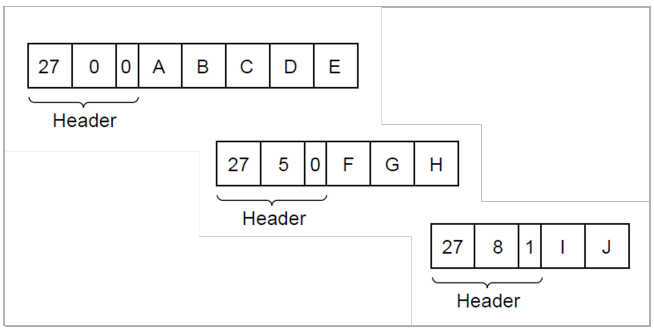


Fragmentation when the elementary data size is 1 byte.

(b) Fragments after passing through a network with a maximum packet size of 8 payload bytes plus header.

- In the first PDU, the End bit has changed to 0, because it is no longer the last PDU. Everything else is the same.
- The second PDU or fragment has the same Packet-id=27 (it goes with the other one), the offset is now 8 (the number of data bytes in the first fragment), and the End bit is 1 because this is the last one.

These two PDUs are sent, and at another gateway, the MTU is even smaller. The first fragment has to be fragmented again.



Fragmentation when the elementary data size is 1 byte.

(c) Fragments after passing through a size 5 gateway.

- The first PDU or fragment is unchanged. It is still the first.

- The second PDU or fragment has the same Packet-Id=27; the offset is 5 (the number of user-data bytes left in the first PDU), and the End bit=0. This isn't the end.
- The third PDU or fragment, which was the second fragment before, is unchanged, because it is still 8 bytes from the beginning, and it is still the last one.

This basically how IPv4 and IPv6 work.

The Dirty Little Secret About IP Fragmentation

It doesn't work. It never has. It has not worked since 1978, when IP was separated from TCP. Before that it worked.

What am I talking about? We just went through an example, and it worked fine. Consider the following sequence of events:

Because relaying can cause PDUs (and fragments) to arrive out of order, each IP fragment has a Packet-Id to know where it belongs relative to other IP fragments, just as we saw in the example. So, let's see what happens in the real world:

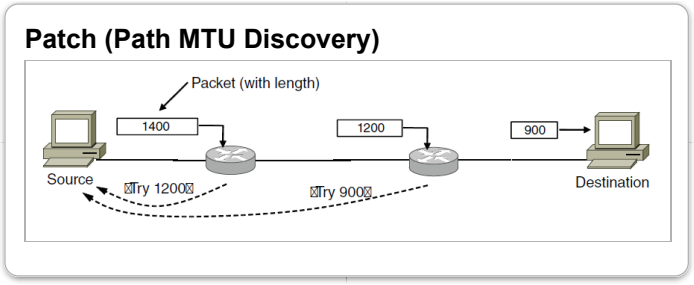
1. TCP creates a PDU and hands it to IP.
2. IP encapsulates the TCP PDU, now an IP SDU, to become an IP PDU, which is sent through the network.
3. At some point, the PDU encounters a gateway to another network with a smaller MTU. The PDU is broken into fragments and forwarded.
4. Now there is a probability, p, that at least one fragment is lost. The fragments that do arrive must be held for at least one MPL or five seconds.

   How do we recover the lost fragment? Retransmission! But IP doesn't do retransmission. IP just has to wait for the lost IP fragment to arrive. Transport Layer does retransmission, as we'll see when we get to the Transport Layer. The IP PDU can't be delivered to the destination TCP, so Ack is never sent.
5. After a while, the sending TCP's retransmission timer to retransmit the PDU will expire after one round trip time plus some epsilon. When the retransmission timer expires, it retransmits all unacknowledged PDUs.
6. The PDU is delivered to IP. IP will dutifully assign a new Packet-id to it and send it.
7. Go to step two.

The result is obvious. The same thing can happen again. The destination host can end up with several

copies of the same packet and not know that they are copies or that some copies could complete other copies. All of this is consuming needed buffers and holding up delivery to the application. Nothing will be delivered to the application until the IP fragments for that PDU all arrive intact. Obviously, this is going to happen, and it does.

But There's a Patch!



There is a patch to fix this. It is called *Path MTU Discovery*.

To do that, the host sends an IP PDU of some large size with the Don't Fragment bit set. If it encounters a smaller MTU network, an ICMP error message is returned. Then the host tries sending a smaller PDU with the Don't Fragment bit and so on until it finds a size that gets all the way to the destination. Based on that, we know the MTU of the path to the destination and can send only PDUs of that size. (Yes, a routing update can change the path of the MTU.) This is how we get around the fact that IP fragmentation doesn't work.

This is equivalent to the old joke:

"Doc, it hurts when I do this."
The doctor says, "Then don't do that!"

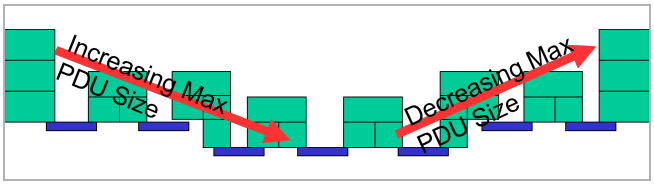
There's just one problem with this, too. It doesn't work either. Most sites block ICMP messages because they are a source of DoS attacks.

Conclusions on Fragmentation

There are several things that can be said about this, and none of them are good:

1. This is the overlay layer where the differences in networks are supposed to be invisible. Different MTU sizes are different. This is protocol translation. This is the approach the phone companies use, not networks. MTU should be a property of the layer.
2. As we saw earlier, PDU size should increase as one goes down in the layers and in toward the backbone. There should not be any fragmentation. There might be concatenation, which fragments back into the original PDUs as the PDU moves from the backbone to the periphery.

Increasing Max PDU Size and Decreasing Max PDU Size



3. A rule of thumb for servers: if a client using the service is sending PDUs of size X, respond with PDUs of size X.
4. As we will see when we get to TCP, this worked with no problem until they separated IP from TCP.
5. For IPv6, it is worse. Fragmentation is an option and can only be done by the sending host. In which case, why bother?
6. We will see a cleaner solution when we get to the Transport Layer.

Why does Tanenbaum spend so much space on such a simple concept, which is also flawed, and yet he doesn't mention that it is flawed?

The Network Layer of the Internet

(What about the Internet Layer of the Internet?)

Read

Read Tanenbaum Chapter 5, Section 5.7.1, pp. 441–444.

Overview

Tanenbaum starts this section with a discourse on the 10 principles of the Internet. (There is a similar list in a section in the OSI Reference Model.) A common mistake made by most readers is that they assume these principles were laid out at the beginning and then followed as a guide to create what the readers see. In virtually all cases, including these two, these lists were assembled after the fact to justify what had been done rather than guide what is to be done. Once the list is assembled, there is often an appeal to the list to argue for or against new proposals.

Let's review them briefly:

- *Make sure it works* – Isn't it interesting that immediately following the explanation of fragmentation

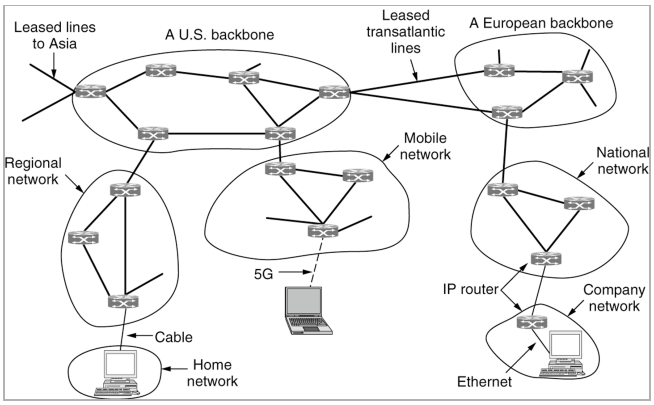
that doesn't work, Tanenbaum leads with this principle? It has been a hallmark of the Internet that two or three independent implementations that could interwork were a necessary requirement for an Internet standard. But this requirement has for the most part fallen aside and is seldom practiced. It is generally a good idea to ensure that the ideas work and to improve the specifications. Using two independent Unix implementations is okay, but using different implementations on different systems is far better.

- *Keep it simple* – This, of course, is a motherhood issue. However, the simplest solution (that ensures future simplicity) is not necessarily the least change to the existing Internet. One of the realities is that often "good enough" is not good enough in the long term.
- *Make clear choices* – This one is good and makes a strike against standards with many options. Options usually arise in standards because one or more factions can't agree on the solution, and it is too early to see the market decision. Options may also be used when a faction was unable to defeat a proposal, so the fallback position is multiple options to make the solution sufficiently unattractive that it won't be adopted. The other source is incremental functions that are not always required, such as with IPv4 and v6 options. The latter have proliferated so much that there may be little room for data. There is more about this and unforeseen implications below.
- *Exploit modularity* – This is always a good recommendation for good software (and hardware) engineering. The primary rule, of course, is creating boundaries between modules so that the modularity will not break the internal functionality of the entity before being split into modules. As we have already seen, IP violates this in a major way.
- *Expect heterogeneity* – This is also important to confirm the generality of the solution. This was an axiom of the ARPANET and much easier to satisfy then when there were 10–15 different machine architectures on the 'Net. Today, this is more difficult but still important to uncover unstated assumptions and discover broken invariances.
- *Avoid static options and parameters* – Or re-stated: avoid sending the same parameters many times. This is especially a problem where IP options carry the same information in packet after packet.
- *Look for a good design*; *it need not be perfect* – Perhaps not, but a good design should be pretty close to perfect, or one will find it is inadequate for future problems and a barrier to adopting a simple solution. To reiterate, "good enough" often proves to be not good enough in the long term. Be careful with kludges; you may have to live with them a long time! (That's 50 years with well-known ports!)
- *Be strict when sending and tolerant when receiving* – Most definitely!
- *Think about scalability* – This is very important but requires that design considerations be imposed early. One is incredibly lucky if scalability can be retrofitted.
- *Consider performance and cost* – It is interesting that, if the above are followed, these things will often take care of themselves.

It is interesting that security was not a design principle of the Internet, nor is, "Do what the problem says." One could also add, "Don't fall prey to 'Not Invented Here'." If someone else has done it and it is right, use it. Don't reject it just because someone else did it first.

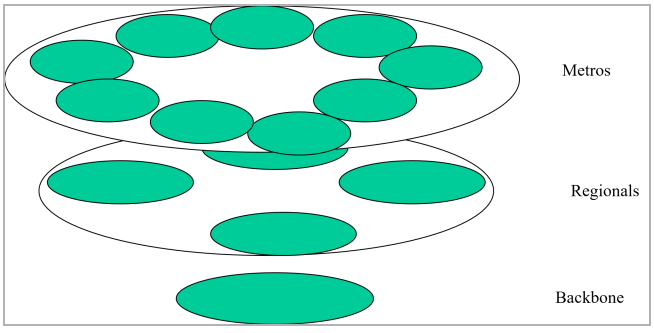
One that could be added based on observation of the IETF would be, “Don’t make a change until it is absolutely necessary. Don’t solve problems before they arise. When they do arise, make the smallest change possible to get by.” We have seen this time and again with the host file, congestion, router table size, details of IPv6, etc. There seems to be a belief that no decision now will not preclude any other action in the future. It isn’t true. And I thought we were suppose to be teaching engineers to see problems in advance and to solve them before they occur.

The Network Layer in the Internet

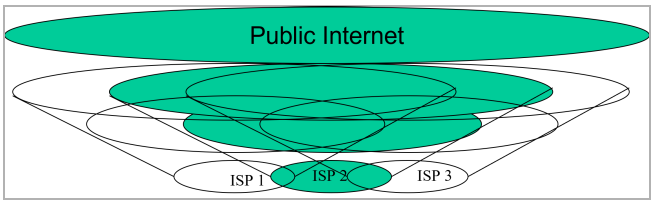


Tanenbaum next discusses the hierarchical structure of the Internet, with local or corporate networks at the edge, feeding metro area subnets, feeding regional subnets, and lastly a Tier 1 provider backbone. Not all of these levels may exist depending on the density of users. At the same time, more levels may exist in the corporate, metro, and regional subnets depending on the number of users and traffic density. If one thinks of this hierarchy as an inverted cone, the component networks may have their own inverted cones. The IPC Model implies that leveraging the concept that layers repeat yields a cleaner and more powerful way to look at the problem is like this:

Implications of the Model & Names (Routing Table Size)



In this that case the Internet is a layer that floats on top of the provider networks. This approach has many advantages, including being more secure and making the provider networks totally independent of the public service and totally under their owners control.



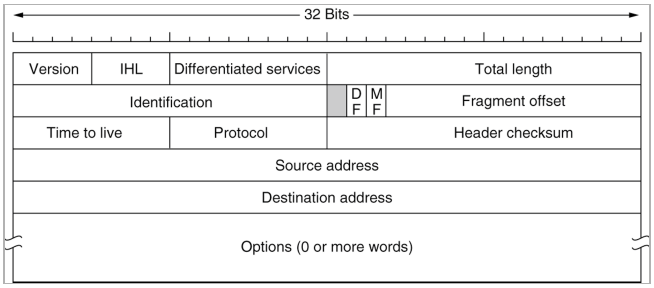
- The Internet floats on top of ISPs, e.g., an “e-mall.”
    - One in the seedy part of town, but “e-mall.”
    - Not the only e-mall and not one you always have to be connected to.

Considering IPv4

Read

Read Tanenbaum Chapter 5, Section 5.7.1, pp. 444–448.

The IP Version 4 Protocol



Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

We now come to what we have been waiting for: a close look at IPv4. The best way to do this is just to go over what the PCI or header looks like. Referring to the figure above:

- We have a *Version field* (4 bits). This will be 4 or 6 in most cases.
- The *IHL or Internet Header Length* (4 bits) is the header length in 32-bit words, including options. The minimum value with no options is 5; the maximum value is 60. This makes some options, such as Trace Route or Source Routing, essentially useless.
- *Differentiated Services* (8 bits), where 2 bits are reserved to identify classes of service (and originally called *Type of Service*, ToS) where there are 3 bits for priority, 3 bits to indicate importance of delay, throughput, or reliability, but it was unused for decades. It has been renamed differentiated services,

which we will discuss later. The field has been reused with 6 bits to designate service class and 2 bits to support explicit congestion notification.

- *Total Length* (16 bits) is the length of the PDU in bytes.

The next 32 bits are all about fragmentation:

- The Identification field is a 16-bit, Packet-Id field used for fragmentation, which is used as we just described above.
- Then 1 reserved bit, followed by the *DF* or *Don't Fragment* bit that indicates that this packet should not be fragmented, then the *MF* or *More Fragments* bit used to distinguish the intervening fragments (MF=1) from the last fragment (MF=0).
- Then the *Offset field* 13 bits, which is the number of bytes in multiples of 8 bytes. This is sufficient for the total length.
- Next, the *Time to Live (TTL)* field (8 bits). The purpose of this field is to delete PDUs that seem to be looping for long periods. It is difficult to actually do an upper bound in time, so this is a hop count. An initial value indicates roughly the number of routers this PDU should take to get to the destination, or perhaps more precisely, the number that if it hasn't reached the destination by now, it isn't going to. When the router receives the PDU, it decrements TTL by 1. If the result is zero, discards the PDU. Values can, of course, range from 0 to 255. Something around 32 is common, as is 255. Note that this enforces an upper bound on MPL as required by Watson's result.

> **Nota Bene**
>
> This merely discards looping PDUs. It does not eliminate loops. There is no guarantee that the next routing update will eliminate the loops. Loops have been known to last for hours or even days. There are some improvements to IS-IS that make it loop-free.

- Next is the *Protocol-id* field (8 bits) we have briefly discussed this earlier. Most people will tell you this identifies the encapsulated protocol. Tanenbaum makes it worse and says it indicates the *transport process* to hand the PDU to.

  Neither of these is correct. The field encodes the kind of protocol encapsulated (UDP, TCP, SCTP, etc.). The reason this is wrong, and Tanenbaum's is very wrong, is that it cannot be used to distinguish two (or more) instances of the same protocol, e.g., two TCPs or three UDPs, etc. What this does is indicate the syntax of the encapsulated protocol, so that the process the user data is handed to knows how to interpret the encapsulated bits. This is also an indication of an ill-formed protocol. There is no allocation phase and no connection-endpoint identifiers. If there were, this field would be unnecessary. (Just because IP is a datagram does not preclude it from having an allocation phase. The datagram property refers to each PDU being routed independently, which had never really been considered in the Internet until recently.)

- Next, *Header Checksum* is the 16-bit 1's complement sum of the header. This is done to detect single bit errors during relaying. To minimize relaying overhead, this only applies to the header. Checking the data is done end-to-end by the layer above. While this has to be recomputed at each hop (because of TTL), there are methods to avoid complete recalculation.
- Finally, there are 32-bit *destination* and *source IP addresses*, followed by any *Options*. We will have much more to say about IP addresses. It should also be noted that Options are avoided wherever possible because an IP packet with options takes the slow path through the router.

A Bit More on Options

Options allow IP to be extended for experimentation or to perform functions for debugging. Options are variable length in increments of 4 bytes. The first byte is an option code. If the option is longer than 4 bytes, the 2nd byte is a Length field. The important property of all options both in IP and TCP is that an implementation can skip to the next option, if it doesn't implement an option. The first five options defined for IP were:

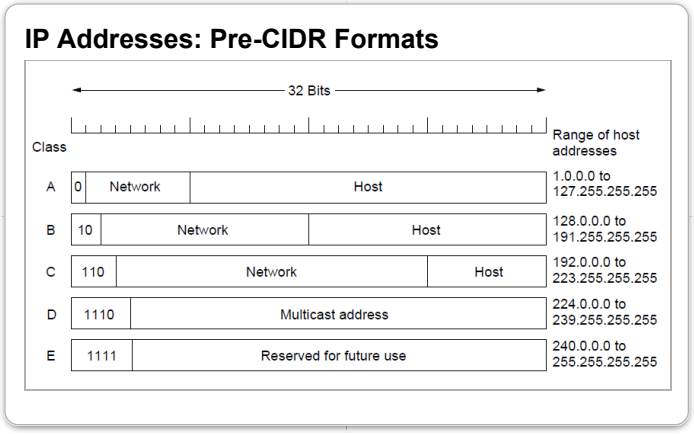
1. **Security**, which merely provided a label for how sensitive the information was. Supposedly, it could be used to avoid routing packets through unfriendly territory. Today it is ignored by all routers.
2. **Strict Source Routing**, which contains a list of IP addresses that defines the exact path a PDU is to follow.
3. **Loose Source Routing**, which contains a list of IP addresses that the PDU must go through in that order, but there may be other routers between them. The PDU can be forced to go from A to B, but might go through C and D in the process of going from A to B.
4. **Record Route**, which records the IP addresses the PDU goes through between the source and destination. This is useful for debugging purposes. This was defined when the Internet was quite small and recording nine addresses was sufficient. The diameter of the Internet is now around 15-20, so this option is not that useful.
5. There are many other options, such as **timestamping**, etc.

IP Address

We will have a lot more to say about addressing in the next lecture but here we will cover how to use IP addresses.

You will hear people refer to an IP address as a host address. It isn't. The most important thing to keep in mind about IP addresses is that they name the interface that connects to the switch. This is a mistake. As we will see next, this is a holdover from the ARPANET. The Internet is the only network design that makes this mistake.

In the beginning, there was classful addressing.

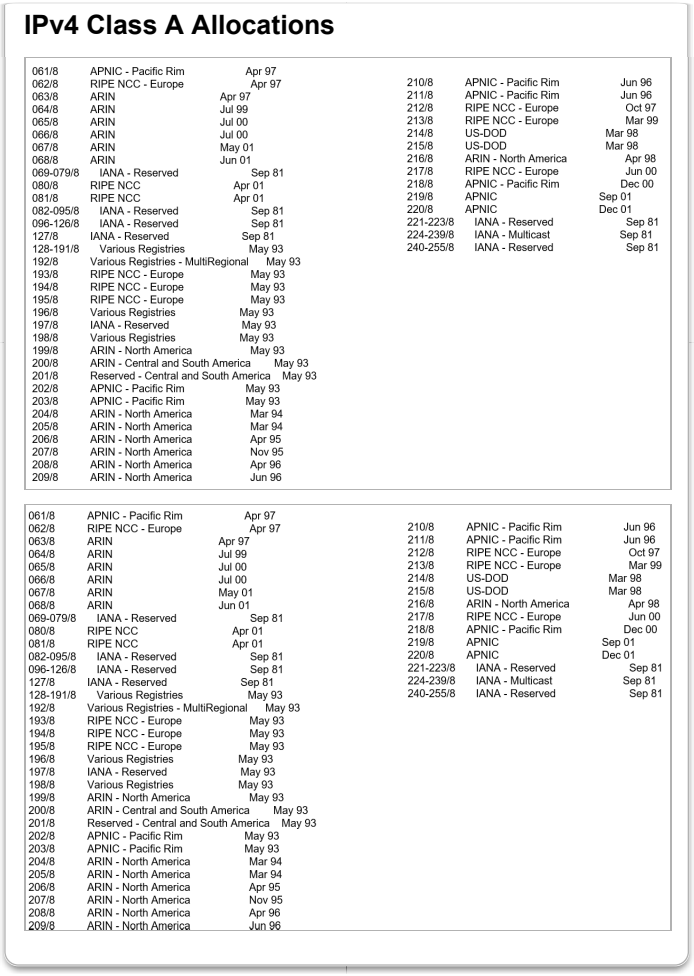


There were three kinds of addresses, A, B, and C, plus a very large block of multicast addresses and a very large block of reserved addresses. The three classes of addresses attempted to divide up the 32-bit address into a network part and a host part.

1. Class A: For very large networks (~16 million hosts), where the first byte indicated the network, and then the rest of the 24 bits would be for hosts and routers on that network.
2. Class B: For medium size networks (~65,000 hosts), where the first 14 bits indicate the network, and 16 bits are for the hosts.
3. Class C: For smaller networks (255 hosts), where 21 bits specify the network and the remaining 8 bits are for the hosts.
4. Class D for multicast addresses.
5. Class E were reserved.

This division of A, B, and C is fairly coarse. It is easy for a small network with a Class C address to exceed to 255 hosts, but be very wasteful of 65,535 hosts having a Class B address and the same between Class B and Class A. But in the early 70s, 32 bits of address space seemed like more than would ever be needed! So IANA was fairly free in handing out blocks of addresses.

The Internet Assigned Numbers Authority (IANA) managed assigning IP addresses and other identifiers. Organizations would apply to IANA for addresses. In the early days they were fairly liberal in handing out Class A addresses (see the table below).



IPv4 Class A Allocations

061/8	APNIC - Pacific Rim	Apr 97
062/8	RIPE NCC - Europe	Apr 97
063/8	ARIN	Apr 97
064/8	ARIN	Jul 99
065/8	ARIN	Jul 00
066/8	ARIN	Jul 00
067/8	ARIN	May 01
068/8	ARIN	Jun 01
069-079/8	IANA - Reserved	Sep 81
080/8	RIPE NCC	Apr 01
081/8	RIPE NCC	Apr 01
082-095/8	IANA - Reserved	Sep 81
096-126/8	IANA - Reserved	Sep 81
127/8	IANA - Reserved	Sep 81
128-191/8	Various Registries	May 93
192/8	Various Registries - MultiRegional	May 93
193/8	RIPE NCC - Europe	May 93
194/8	RIPE NCC - Europe	May 93
195/8	RIPE NCC - Europe	May 93
196/8	Various Registries	May 93
197/8	IANA - Reserved	May 93
198/8	Various Registries	May 93
199/8	ARIN - North America	May 93
200/8	ARIN - Central and South America	May 93
201/8	Reserved - Central and South America	May 93
202/8	APNIC - Pacific Rim	May 93
203/8	APNIC - Pacific Rim	May 93
204/8	ARIN - North America	Mar 94
205/8	ARIN - North America	Mar 94
206/8	ARIN - North America	Apr 95
207/8	ARIN - North America	Nov 95
208/8	ARIN - North America	Apr 96
209/8	ARIN - North America	Jun 96
210/8	APNIC - Pacific Rim	Jun 96
211/8	APNIC - Pacific Rim	Jun 96
212/8	RIPE NCC - Europe	Oct 97
213/8	RIPE NCC - Europe	Mar 99
214/8	US-DOD	Mar 98
215/8	US-DOD	Mar 98
216/8	ARIN - North America	Apr 98
217/8	RIPE NCC - Europe	Jun 00
218/8	APNIC - Pacific Rim	Dec 00
219/8	APNIC	Sep 01
220/8	APNIC	Dec 01
221-223/8	IANA - Reserved	Sep 81
224-239/8	IANA - Multicast	Sep 81
240-255/8	IANA - Reserved	Sep 81

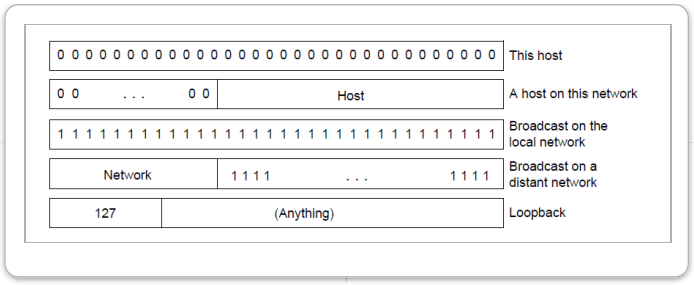
061/8	APNIC - Pacific Rim	Apr 97
062/8	RIPE NCC - Europe	Apr 97
063/8	ARIN	Apr 97
064/8	ARIN	Jul 99
065/8	ARIN	Jul 00
066/8	ARIN	Jul 00
067/8	ARIN	May 01
068/8	ARIN	Jun 01
069-079/8	IANA - Reserved	Sep 81
080/8	RIPE NCC	Apr 01
081/8	RIPE NCC	Apr 01
082-095/8	IANA - Reserved	Sep 81
096-126/8	IANA - Reserved	Sep 81
127/8	IANA - Reserved	Sep 81
128-191/8	Various Registries	May 93
192/8	Various Registries - MultiRegional	May 93
193/8	RIPE NCC - Europe	May 93
194/8	RIPE NCC - Europe	May 93
195/8	RIPE NCC - Europe	May 93
196/8	Various Registries	May 93
197/8	IANA - Reserved	May 93
198/8	Various Registries	May 93
199/8	ARIN - North America	May 93
200/8	ARIN - Central and South America	May 93
201/8	Reserved - Central and South America	May 93
202/8	APNIC - Pacific Rim	May 93
203/8	APNIC - Pacific Rim	May 93
204/8	ARIN - North America	Mar 94
205/8	ARIN - North America	Mar 94
206/8	ARIN - North America	Apr 95
207/8	ARIN - North America	Nov 95
208/8	ARIN - North America	Apr 96
209/8	ARIN - North America	Jun 96
210/8	APNIC - Pacific Rim	Jun 96
211/8	APNIC - Pacific Rim	Jun 96
212/8	RIPE NCC - Europe	Oct 97
213/8	RIPE NCC - Europe	Mar 99
214/8	US-DOD	Mar 98
215/8	US-DOD	Mar 98
216/8	ARIN - North America	Apr 98
217/8	RIPE NCC - Europe	Jun 00
218/8	APNIC - Pacific Rim	Dec 00
219/8	APNIC	Sep 01
220/8	APNIC	Dec 01
221-223/8	IANA - Reserved	Sep 81
224-239/8	IANA - Multicast	Sep 81
240-255/8	IANA - Reserved	Sep 81

Browse the list of Class A addresses as of several years ago. At one point, BBN (Bolt, Beranek, and Newman), the company that built the ARPANET with never more than 5000 employees, had five Class A address blocks! They have since returned them. Scan the list and you will see companies that no longer exist and companies that are large but certainly don't need 16 million addresses: HP, Xerox, Bell Labs, Merck, UK Social Security, etc.

Scan the list and you will see companies that no longer exist, companies that are large but certainly don't need 16 million addresses: HP, Xerox, Bell Labs, Merck, UK Social Security, etc.

There are also a number of address blocks assigned to the U.S. Department of Defense that certainly should not be on the public Internet, such as the Defense Information Systems Agency! IANA clearly didn't see the distinction between addresses in the public Internet and addresses in other internets.

We will return to this shortly. In addition to these five classes of IP address, several special forms of IPv4 addresses have been defined.



Referring to the figure in order:

An address of all zeros means this host or this network and is used primarily when a system boots.

An address with zeros as the network number (8, 16, 24) and a host number means it is on this network.

An address of all 1s means a broadcast on this local network.

An address with a network number and all 1s as the host number is a broadcast on that network.

An address with 127 as the network number (0 and 7 1s) and anything in the rest of the address is called the *Loopback Address* for testing. A packet with this address does not leave the system but is processed as if it were an incoming packet.

The First Great Internet Addressing Crisis

We will consider this much more in the next lecture, but for now we will consider how the use of IP addresses has changed in response to the Great Internet Addressing Crisis.

By the late 1980s, several problems with IP addressing were getting to the point that they could not be ignored. As already mentioned, there was the rather liberal handing out of Class A addresses. It was easy to see with the growth of the Internet that there would be a shortage of IP addresses.

But the real problem was the exponential growth of router tables. For some reason, which I still don't understand, IANA had been handing out blocks of IP addresses *in order*! IP addresses were not acting like *addresses*. They were not location-dependent and route-independent.

This meant that if a company in New York City requested a block, they got X. If the next request was for a company in Sri Lanka, they got block X+1. This meant that every assignment required another entry to the router tables and sometimes more than one. By 1990, router table size had passed 200K! and was growing exponentially with no end in sight.

Being Class A Doesn't Mean Qualifying for Class A Addresses!

There was once an elite financial house that thought they deserved a Class A address block because they were a Class A company! But because they were elite, they didn't have anywhere near enough sites and systems to need a Class A address block. They just couldn't understand why not. They even took it to the President , yes, the one in the White House, where it was turned back to IANA, who gave them a few more Class Bs than they could use.

And there was the multihoming problem that we had known about since 1972, which also needed to be solved.

The Internet Architecture Board (IAB) convened the ROAD process to make recommendations. They concluded that the following actions should be taken:

- First, they *created private address* space, which we'll talk more about below. It was recognized that not everyone needed to be able to accept incoming connections, e.g., they did not provide a service. They didn't need globally visible IP addresses. Also, by the late 1980s, it took about 20 seconds after turning on a new PC for it to be attacked. So, it was proposed to declare certain blocks of addresses to be private addresses that would not be used on the public Internet and would be mapped to a public IP address by Network Address Translation (NAT). This would greatly reduce the demand for public IP addresses. Most people in an office had no need of a public address, nor the technical background to safely use it.
- *Move to Classless Inter-Domain Routing (CIDR)*. We have already seen that the Class A, B, and C addresses create blocks that are too big or too small. It is a big jump from 24 million to 65,000 and from 65,000 to 256. This allows allocations of finer granularity, the boundary to be anywhere. This is the first step toward making IP addresses, addresses.
- *Give Tier 1 providers large blocks of addresses*. This is the next step to making IP addresses, addresses. Today if you look at the IANA list Class A addresses, you will find that they have all been returned to the address authorities with a few exceptions. These are then in turn given to Tier 1 providers.
- *Tighten the rules on handing out large blocks of addresses and push people to get addresses from their Tier 1 provider*. This makes the addresses provider dependent. All addresses assigned to the same provider have the same prefix. From one Tier 1 provider, all addresses for another Tier 1 provider can be mapped to a single router table entry, i.e., routed to the other provider as early as possible and let them figure out where it is. We will look at this in more detail in a moment.
- And last but far from least, *recommend a replacement for IPv4*.

This was a smart move. By giving large blocks of addresses to Tier 1 providers and the Tier 1 providers assigning them to their customers, all addresses with the same network number or prefix would be routed the same way and be one entry in the router table. This is often referred to as "route aggregation," which it is, but what it really is, is making the addresses location-dependent. Finally, IP addresses have the same

property as postal addresses. My post office in Foxboro doesn't have to know the street address and house number where my friend in Seattle lives. They see "Washington" and put it in a bag going west. Now IP address have a similar property; the router sees the prefix and knows what provider to route it to without having to know the whole route to the destination host.

Wouldn't it have been better if it were geographic for routers too? No, this is a common misconception. Once one says "location-dependent," some people immediately think "geographic location." In networking, "location" is always location relative to the graph of the layer. Since providers often cover the whole country, the nearest peering point between two Tier 1 providers might be in the opposite direction. Provider-based addressing is required because IP addresses name the interface, which is route-dependent.

Of course, once the packet gets to its Tier 1 provider's network, all of the addresses have the same prefix. In some sense, we are back where we started. Unless, of course, the provider has been smart and done something similar with the way it has assigned addresses within the blocks of IP addresses assigned to it. They treat the more of the address (but not all of it) as hierarchical to reflect the natural subnets within the provider's network.

This is precisely what subnetting and CIDR are all about.

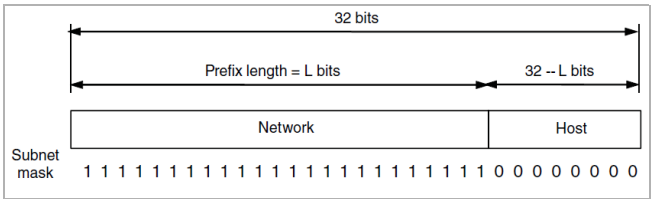
Prefixes, Subnetting, and CIDR

Read

Read Tanenbaum Chapter 5, Section 5.7.2, pp. 448–456.

Early in the development of the Internet, it was decided to express IPv4 addresses in what is called ***dotted-decimal*** format. Each byte is written as a decimal number from 0 to 255. For what we are about to do, it would have made much more sense if they had written each byte as 2 hexadecimal numbers. (I have always thought that this was the usual hubris of the expert who worked with IP addresses so much that they had memorized all of the bit patterns from 0 to 255. However, it appears to be more a lack of foresight.) For the rest of us that only do it occasionally, it is a pain! To aid this, several conversion tables or "cheat sheets" are posted on Blackboard.)

IPv4 Addresses: CIDR- Classless Interdomain Routing



An IP prefix and a subnet mask.

As we saw, having Class A, B, and C address blocks was too coarse. The idea of creating Classless Inter-Domain Routing (CIDR) was to be able to put the boundary between the network part and the host part anywhere. To do that, we need to know where the boundary is between the network part and the host part. The solution is to have a mask that when AND'ed with the address yields the network part. With classful addresses, this was obvious. The mask for Class A would be 255.0.0.0; for Class B, 255.255.0.0; and for Class C, 255.255.255.0.

The further convention for specific address blocks is the "slash followed by a number," where the number is the length of the network part. So, a Class B is /16 and a specific Class B might be 128.208.0.0/16, while Class C in the same block might be 128.208.2.0/24. The mask for this latter address block would be 255.255.255.0. One can't tell by inspection what the mask should be so they are circulated as part of the routing protocol updates.

Clearly, we are creating a hierarchical addressing scheme and we want hosts in the same part of the network to be in the same part of the hierarchy. As we keep saying, we want them to be location-dependent without being route-dependent. Why is this last property so important? Because in a network there can be more than one way to get to it. For example, isn't there more than one way to get to your house? Of course… Well, if you live in Boston, maybe not! Redundant paths are good.

In describing this, Tanenbaum points out two disadvantages of hierarchical addressing. (See the first paragraph on page 449.) The first is that the "IP address of a host depends on where it is."

Of course it does! That is why they are *addresses*!

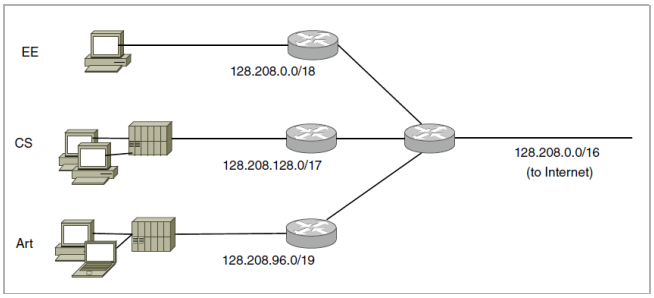
We have known since the mid 1970s (and we will talk about this more next time) that application names indicate *what* (location-independent), network addresses indicate *where*, and routes are *how* to get there. Tanenbaum is an OS expert. Someone should ask him if virtual memory addresses should be location independent! He points to Ethernet addresses as a good example of an address space. He clearly doesn't understand addresses. The creators of MAC addresses knew they were device-identifiers being used as an "address." For a multi-access media, every destination will see every PDU. Hence, a flat address space is acceptable, because the only property the "address." must have is the destination recognizes its address when it sees it. First of all, as we saw Ethernet addresses were made a flat 48 bits address space to make

enrollment simple. (That has become a security problem because they are a device-id and enable tracking of users.) An address never needs to be seen outside its layer, its scope. For a traditional Ethernet, 12 bits would have been more than enough. (Show me a 1 km Ethernet with 1024 devices on it, let alone 4096, and I will show you an Ethernet that would be all contention and no work!) Tanenbaum brings up mobility as a counterexample, when it is exactly the opposite. We will show how this is a non-issue later. This paragraph is just totally wrong!

His second objection is more reasonable. That if the sizes of the blocks indicated by the hierarchy are not well chosen, it can be wasteful of addresses. This is true, but here too there are other considerations. First, those environments where address length is an issue are small networks. While addresses take up the largest part of the PCI, they aren't that expensive relative to MTU. In fact, if longer addresses can be effectively used for finer granularity in routing (and therefore greater efficiency), it is generally preferred.

The unfortunate thing is that Tanenbaum is not the only textbook author that doesn't understand addressing.

But this means we should pay attention to how we design an address space to take advantage naturally occurring subnets. The book has a nice, simple example of subnetting.



Splitting an IP prefix into separate networks with subnetting.

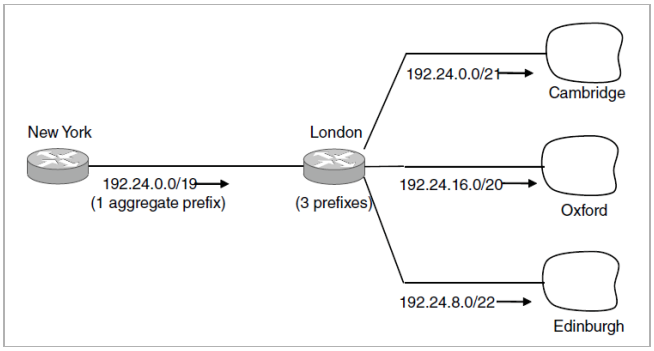
To parallel the book, suppose the CS department gets a Class B 1ta28.208.0.0/16 when the university first gets on the Internet. Soon the EE department wants to get on the Internet. The CS department may have a lot of computers but nowhere near 65,000, so rather than get another block, they subdivide the /16 in half into a /17 for them as 128.208.128.0/17 and a /18 for EE at 128.208.0.0/18. Then the Art department also wants to get on the Internet, so they get a /19 at 128.208.96.0/19. As other departments join, the /16 can be further subdivided. All traffic to the university is going to go through the router at the top of the tree. Now to the outside world, the address of the university is the /16. One entry in their router table for the potentially thousands of systems in the university. For the university, its network uses a longer mask, so it can easily route to the various departments.

A similar thing can happen at the next level up.

A Set of IP Address Assignments

University	First Address	Last Address	How Many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

IP Addresses



Aggregation of IP prefixes.

Here we have address blocks for three universities in the UK: Edinburgh, Oxford, and Cambridge. A router in New York has one entry in its routing table, 192.24.0.0/19, that gets the packets across the pond to London, which has three entries in its router table, one for each university's address block.

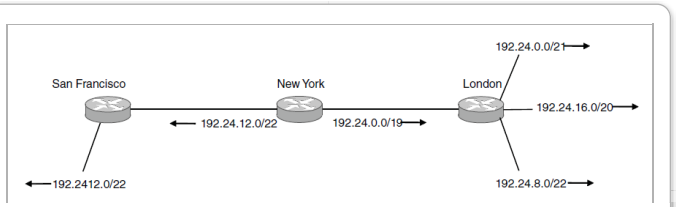
Now each router in the university must have a router table entry for each of its subnets. A large university might have multiple levels within the university as well as entries for destinations outside the university based on who else it is connected to.

Perhaps it will help to understand what is going on here by taking a break from all of the technical gobbledygook. It is really like this:

This is basically what CIDR does: It allows the routers to have detailed knowledge locally and less knowledge for networks far away.

CIDR has done a lot to arrest the exponential growth of router tables. However, other factors are causing them to increase again. It seems that router table size is now approaching a million routes in the core. At one point, to expand the number of routes for IPv4 addresses that could be supported, network administrators had to reduce the number of IPv6 routes they could support. (That certainly encourages the transition to IPv6!)

Aggregation is basically preprocessing the network graph relative to this router to create a new graph with fewer nodes. Sets of addresses are being aggregated into a single node of the new graph. Then the routing algorithm is run on the new graph. This works because all of the aggregate addresses would have been forwarded to the same next hop anyway. The graph that each router uses the routing algorithm on is potentially a different graph. Notice that not once was the term “topology” used. That is because these aren’t topologies. They are graphs.



Longest matching prefix routing at the New York router.

However, as allocations get tight, it can be difficult to keep the “nearness” as logical as one might like. Consider the address assignments in figure above. There is an available block at 194.24.12/22. Suppose this block is allocated to a company in San Francisco. What does the New York router do? It could add 4 entries to its router table: the 3 UK universities and this new /22 in San Francisco. There is another rule to simplify this: the longest prefix match rule. The rule is to choose the longest match. If the New York router adds an entry for /22 in San Francisco, a packet for San Francisco will match both the /22 and the /20. The /22 is longe, so the packet is routed to San Francisco. A packet for one of the UK universities will only match the /20.

NAT or Network Address Translation

Read

Read Tanenbaum Chapter 5, pp. 456–461.

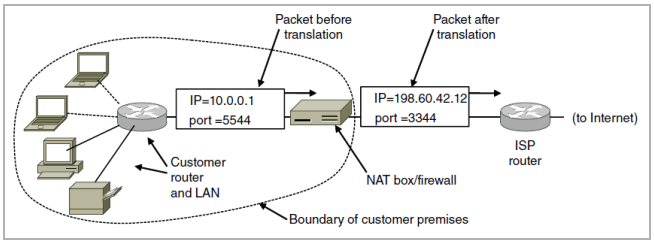
One of the other measures the IAB took to alleviate address consumption and increasing router table size was to create private addresses with Network Address Translation (NAT). With IPv4, there could only be approximately 4.2 billion devices of the Internet, which was going to be far too few. People realized that very few, if any, of these machines were going to be used as servers. They didn’t need to accept incoming connection requests. They were only going to connect to servers. They didn’t really need a global IP address. In addition, by the by the late 1980s, it had gotten to the point that when a brand-new laptop was turned on, it took about 20 seconds for it to be attacked. As a first line of defense, it would be useful to give devices private addresses behind a NAT box. You can’t attack what you can’t address.

The idea was that the NAT box (firewall) would be assigned one IP address. All of the devices behind it would have private addresses. Three private address spaces were carved out of the IP address space:

10.0.0.0 – 10.255.255.255/8 (the old ARPANET net number)
172.16.0.0 – 172.31.255.255/12
192.168.0.0 – 192.168.255.255 /16

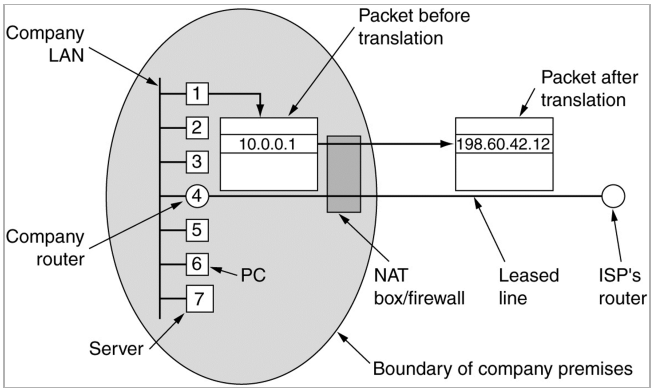
These should never occur in the public Internet. Devices on a private network are assigned private addresses. When they connect out, the connection request goes through a NAT box. The NAT maps the source address and ports of the connection using private addresses to a new connection using a global IP address and source port. When a device behind the NAT opens a connection to an address outside this network, the PDUs go to the NAT. The NAT records the private address and the source port of TCP or UDP

in a table. (From the IPC model, we know that the source and destination ports form a connection-id.) Then using the global IP address, the NAT initiates a connection with a different source port and records the source port in the table.



Placement and operation of a NAT box.

As in the example, the laptop with address 10.0.0.1 opens a TCP connection with source port = 5544. At the NAT box, a packet's address is translated to 198.60.42.12 with port = 3344. The NAT records that 5544 maps to 3344. When a response comes back on 198.60.42.12 port 3344 (the outgoing source port is now the incoming destination port), it is translated to 10.0.0.1 port 5544 and forwarded on the private network, and everything works fine.



Placement and operation of a NAT box.

This was incredibly controversial. There were a lot of very intense debates over this, full of yelling and screaming, and flame wars were going on in the mailing lists. But it was practical and it worked, and it helped those responsible for their network's security. So, it was deployed, usually as part of the firewall. But the controversy never went away, as we will see.

- It violated the "*spirit of the Internet!"* Supposedly, the notion that everything should be under a single address space was considered one of the fundamental properties of the Internet. But most sites don't want every address globally visible. A lot of innocent people just want to use their machines safely on the Internet. They don't know about all this weird stuff that we do.

- It makes the Internet more connection-like because it has to go through a NAT box. That is an artifact of an incomplete addressing architecture.
- It violates layering. However, so does the protocol-ID field in IP, but that didn't bother them.
- It forces use of TCP or UDP (the ports). Not really. The ones claiming this didn't understand that the ports were a connection-id to begin with, and any protocol that was going to be able to have more than one flow at a time was going to have to have ports.
- Some applications pass IP addresses that won't work through a NAT. Yes, they do. They shouldn't be doing this. This is like a program in a high-level language passing around physical memory addresses, the last thing it should be doing. (I have to take some of the blame for this. One of the protocols they are referring to is FTP and I was involved in that specification. But it was 1973 and the TIPs made us do it!)

The final conclusion is that NATs only break broken architectures. A 2021 presentation by one of the IPv6 experts on the "Myths of IPv6" came to the same conclusion. Interestingly enough, our next topic is:

IPv6

Read

Read Tanenbaum Chapter 5, Section 5.7.3, pp. 461–470.

Hype and Reality

One of the recommendations of the ROAD process was to develop a replacement for IPv4. In the book, Tanenbaum has a reasonably fair analysis of what happened. Although, his description doesn't come anywhere close to the level of tension and nastiness that went on. It got very ugly. We will have more to say about this when we cover naming and addressing next.

According to the textbook, the goals of IPv6 were those listed below. However, remember what we said at the beginning of this lecture about lists like this. The same can be said about this list. When the IPv6 group first convened, the only problem they thought they needed to solve was increasing the size of the address. They were unaware of the exponential growth in router table size! The spin doctor version of the major goals for IPv6 are:

- *Support billions of hosts*. While the ROAD group was most concerned about increasing router table size, they believed (correctly) that that was harder to explain to most people and chose to emphasize the shortage of IP addresses, which was easy to understand. They were too successful.
- *Reduce router table size*. It does not appear that IPv6 has gone beyond what CIDR provides.
- *Simplify the protocol.* As we will see, the protocol itself is simpler, however, this has led to

considerable complexity and contradicting RFCs. We will touch on some of these later.

- *Better security*. This has been hyped a lot, but IPv6 security is the same as IPv4 security. In fact, one of the ground rules in developing IPv6 was that anything developed to work with IPv6 had to work equally well with IPv4, in case they had to back it out. That gives you an idea of how much confidence there was in the solution.
- *Attention to Type of Service*. This and the Flow Label are still being debated about how they should be used.
- *Aid multicasting*. There really hasn't been any. Nothing has been done.
- *Roaming Host without changing addresses*. This goal has an internal contradiction. An address that doesn't change relative to changing the position in the graph of the layer is not an address. They have improved slightly on IPv4 mobility but not enough. ETSI has determined that it can't meet the requirements of the mobile phone industry. We will look at this in detail in the next three lectures and show how there is no efficient solution with the current assumptions of the Internet.
- *Allow future protocol* evolution. This has been a humongous problem in the v6 group over the last 10 years at least. Without an architecture, this is essentially an impossible goal. It is the architecture that lays out the guidelines for the evolution.
- *Permit coexistence of old and new protocols* for years to come. To some degree this was accomplished because the only common field between IPv4 and IPv6 is the version field. The official transition plan is to operate dual stacks with NATs. (Adopting a variable length address where the shorter addresses were IPv4 addresses would have been a much better solution. But too many people didn't understand that variable length would be just as fast to process as fixed length.)

Let us review the "advantages" of IPv6 that Tanenbaum covers:

Longer addresses – This is true but only 64 bits are routable; the other 64 bits supposedly name an endpoint. (This is still being debated.)

Simplification of the header – True, but this is a very small part of the issues.

Better support for options – This has become a total mess. The header has been simplified, but there is a proliferation of options. The options have been broken into several categories such as hop-by-hop, destination, etc. There are many issues here. Options that conflict with other options and options that conflict with the addressing RFCs. Again, the lack of an architecture is the root cause. Note that the Protocol-Id field of IPv4 has become the last option.

Big advances in security – not true at all. It is the same as IPv4.

Quality of Service – This is the same as v4 and possibly worse since there is still considerable debate about the use of Traffic Class and Flow Label fields. (The lack of an architecture is confusing this problem as well.)

Increasing router table size – This is not addressed. The same as IPv4.

Transition strategy – This involves using a NAT. Once behind a NAT, why do you need IPv6?

Adoption – The real problem came up within two weeks of the approval of the RFC specifying IPv6, when somebody pointed out that this has no benefit to those who have to pay for the transition. (There was so much controversy surrounding this and so few people really understood the issues, that they were afraid to make many changes. In the end, the changes were so few that there was no benefit to those who had to pay for it.) This is why there was more IPv7 deployed in 1992 than IPv6 in 2014. Today this is used to discourage any changes to the Internet: 'Look how long it took to convert to v6; what you are proposing will take even longer.' But that is because it didn't do anything that anybody wanted. Make a significant improvement and it will change quickly.

> Can you imagine a network admin going to upper management asking for a few million to convert to IPv6. The first question would be, "How does this benefit our bottom line?" And they hear, "we will be a better network citizen." He is going to be told go find a job!

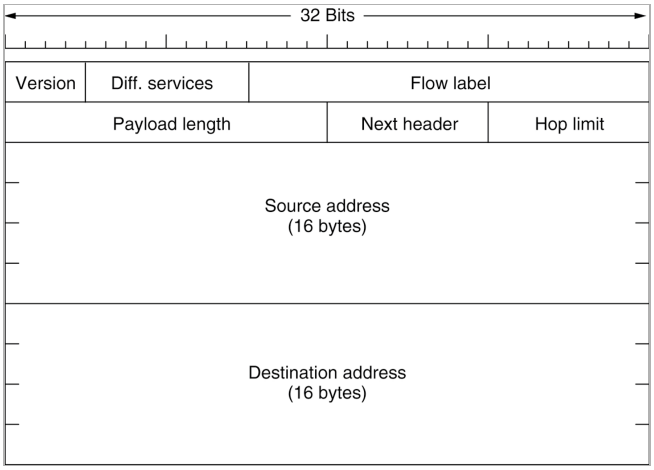
No private addresses – They have reversed themselves, two or three times on this issue, which is an indication of the level controversy associated with it. First it is in, then out, then back in, etc. The purists can't bear the idea that those responsible for networks want private addresses and those responsible for networks can't live without it. They seem to have reached a compromise by defining a private address as an address of limited scope. The problem is that the IETF has admitted (recently) that their definition of scope is broken, and they don't know how to fix it. Once again, the lack of an architecture is making the problem hard. They need to recognize that addresses belong to layers not protocols.

Potential scaling problems with the router calculation – It is not so much about how long it takes to do the routing update; it is how long it will take the network to quiesce after a routing update. If the effects of a new routing update does start while the network is still in a transient state, this is likely to lead to instabilities.

IPv6 Header Format

Let's go over the IPv6 header:

The Main IPv6 Header



The IPv6 fixed header (required).

There is both more and less of it! The header is longer because of the addresses, but the number of fields is fewer. In particular fragmentation has been moved to an option.

First, *the Version field* (4 bits) – This contains 6.

A *Differentiated Services field* (8 bits) – It is the same as in IPv4.

A *Flow Label* (20 bits) – There is still some debate about how to use this. Supposedly it is so that the source or destination can mark PDUs that are to be handled the same way. It is not clear how the definitions of these values are communicated to the routers.

A *Payload Length* (16 bits) – This contains the number of bytes in the PDU, not including the 40-byte header, which presumably means it includes the options.

A *Next Header* (8 bits) – This indicates which option extension header comes after the basic 40 byte header. If this is the last header, then the value indicates the protocol encapsulated, e.g., TCP, UDP, etc. In this case, the last header substitutes for the Protocol-ID field in IPv4. How is Next Header distinguished from the last header? Good question.

A Hop Limit (8 bits) – This replaces the Time to Live field in IPv4.

Source Address (128 bits)

Destination Address (128 bits)

While renaming TTL Hop Count, they have done this just when it has become practical to make TTL actually Time to Live, and when propagation time is often dominant over the time to relay. Remember making TTL a hop count was acceptable in the early 80s when time to relay was dominant over propagation time.

Notice there is no header checksum as in IPv4. There is no protection against memory errors during relaying. They are relying on the pseudo-header (which we will cover in the next module) to detect errors in

the IPv6 header (primarily the addresses). As we will see, the pseudo-header is wrong and is part of why TCP/IP is useless for mobility.

They have gotten sane about representing IPv6 addresses. They are written as 8 groups of 4 hexadecimal numbers separated by colons:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Leading zeros within a group can be omitted, so that 0123 can be written as 123. Any group of 16 zeros can be replaced by a pair of colons, thus rewriting the above as

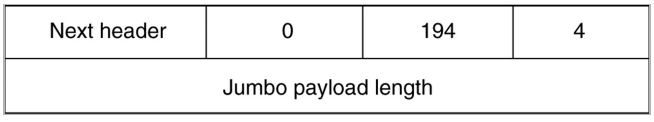
8000::0123:4567:89AB:CDEF

Tanenbaum goes off on a tangent about how 128 bits of address can name more than the number of molecules on earth. However, he seems to forget the lesson of VLAN tags: The number of molecules is irrelevant to the number of virtual machine interfaces that might need addresses.

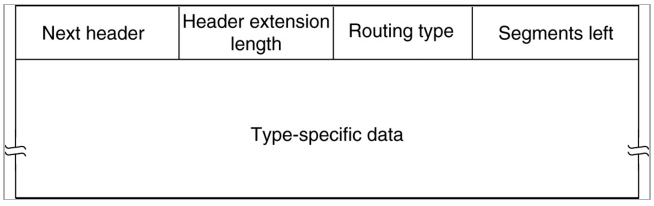
Extension Headers (Options)

Extension headers look pretty much like Options did an IPv4 and have the nice property that each one points to the next one, and an implementation doesn't have to understand an option in order to skip it. It is good to remember that IP packets with options will use the slow path through the router.

For fixed length options, there is a 1-byte *Next Header field*, followed by 1-byte *Length Field* followed by the *optional information*. Any Extension Header must be a multiple of 8 bytes. For variable length Extension headers, each defines its own format beyond the Length field.



The extension header for routing.



The hop-by-hop extension header for large datagrams (jumbograms).

The first figure is an example of a fixed-length Extension Header. The second is of a variable-length Extension header.

Summary

To summarize, the problems in IPv6 are as follows:

Fragmentation still doesn't work. It was made an Extension Header and is more limited. First, fragmentation is only done by the sending host. This means that Path MTU Discovery is required, but of course we know Path MTU Discovery doesn't work either. It is usually blocked because ICMP is used in DoS attacks. Considering that only the sending host can do fragmentation, and TCP generates a byte stream, once the MTU is known (or guessed), why is fragmentation needed at all? For protocols that do not generate a stream, why not leak the information to the protocols using IP so they don't generate PDUs larger than the MTU?

One question that has risen is what happens if there is fragmentation, and all of the options won't fit in a fragment? Nobody knows the answer.

Next header/Protocol-id conflict. These are two registries for the same field. Someone was asleep at the wheel.

MAC address in an IPv6 address. Another lunacy in IPv6 was that some developers were using MAC addresses for the lower 64 bits, the endpoint. This is not only bad addressing practice, but also a major security hole. Using MAC address in the IP addresses makes the address route dependent, which is what we are trying to avoid.

Can't route on anything larger than a /64. What is the point of a 128-bit address if it can't be used? This may be changing but can't be assumed.

Path MTU Discovery can't be relied on. Although, it was pretty useful for heartbleed.

Interactions among options and with fragmentation remain unresolved.

Will routing scale to 4.3 billion IPv4 address spaces?

Tanenbaum has an interesting section on the controversies that surrounded the debate over some options. It is an interesting read. Often it is the case that neither side was right.

The Christmas Lock-Up

There was a famous incident on the ARPANET on Christmas Eve around 1972 or so. An ARPANET message could be up to 8 packets, which were reassembled in the IMP and delivered to the host as a single message. Late in the day on Christmas Eve at UCLA, it was realized that everyone on the East Coast and Midwest had gone home. This was probably the closest they would ever see to the 'Net in a quiet

state. Someone had the idea to have all of the IMPs send their “current state” messages to the NMC, which they did. However, the message was a multi-packet message. As packets began to arrive, the NMC IMP reached a point where it had several partially reassembled messages and not enough memory to complete any of them. The NMC IMP locked up, which caused the IMPs it was connected to to lock up and in seconds it had propagated across the ‘Net and the whole ARPANET was locked up. The people on call had to come back in and run the paper tape to manually re-boot their IMPs. This deadlock condition had been forecast, but had never occurred. Once more the lesson on asynchrony and concurrency: if it can happen, it will!

Internet Control Protocols

Read

Read Tanenbaum Chapter 5, Section 5.7.4, pp. 470–472.

ICMP (Internet Control Message Protocol)

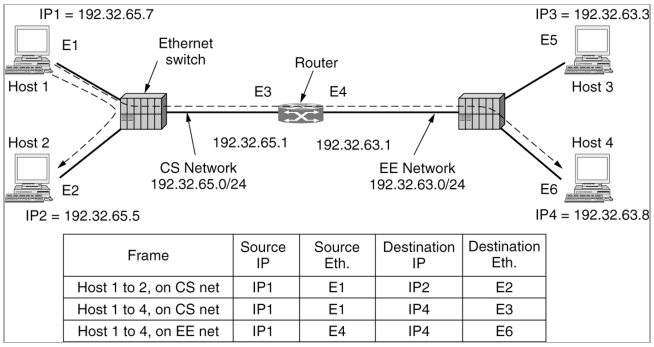
In 1981, ICMP was the first real attempt at some kind of management. The ARPANET had an excellent network management system that could deploy new software to the entire network from Cambridge, Massachusetts. When an IMP crashed, it could re-boot from its neighbors, without human intervention. The Network Measurement Center at UCLA could collect data from the IMPs. The Internet didn’t really start to consider network management until the late 1980s, which was quite late. It is a simple datagram error message protocol for IP. When a router encounters a problem, it may generate an ICMP message. Typical message types are:

- Destination unreachable – the host is unreachable, or the DF bit was set and fragmentation was required.
- Time Exceeded – TTL went to zero and to do TRACEROUTE.
- Source Quench – an early attempt at congestion control.
- Redirect – to tell sender to use a different route.
- Echo and Echo reply – what you know as ping.
- Timestamp request/reply – just what it says.
- Router advertisement/solicitation – used by a host to find a router.

ARP (Address Resolution Protocol)

ARP satisfies a unique problem created by multi-access media such as LANs—how to find what IP address maps to what MAC address or what lower layer address. If media is point-to-point, this isn't a problem: Send a request down the line and ask! But if it is a broadcast media, there is a problem. A request can't just be sent to the other end of the wire to figure out what's going on. It is necessary to figure out which MAC address on a multi-access media belongs to which IP address. ARP is used to resolve the problem.

ARP—The Address Resolution Protocol



Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6

Two switched Ethernet LANs joined by a router.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Consider the following example. Suppose Host 1 wants to send a PDU to Host 3. It has gone to DNS and gets back the IP address of Host 2 as 192.32.65.5. It can construct an IP PDU, but it needs to know what MAC address. So, it broadcasts a request on the Ethernet asking who has IP address 192.32.65.5. Host 2 will respond with its Ethernet address. Host 1 records this in a cache so it doesn't have to look it up again. Host 1 can facilitate things by including its IP to MAC address mapping in the broadcast. Also, the other hosts on the LAN can also cache it. This is what ARP does…sort of.

Now suppose that Host 1 wants to send a PDU to Host 4 at 192.32.63.8. It needs to know what MAC address to use. But Host 4 is on a different Ethernet in a different IP subnet separated by a router. Host 1 doesn't need to know Host 4's Ethernet address. It needs to know what Ethernet address is on an Ethernet with Host 4. (Although Host 1 doesn't know that.) Host 1 will again send a broadcast requesting the MAC address for Host 4. The router notices that this is on a different subnet, and it is the subnet it is connected to, so it does an ARP broadcast on its subnets. Host 4 responds with its MAC address. Now the router knows that Host 4 is on E6 that is reachable through its Ethernet address E6. It responds to Host 1 to tell it that Host 4 is available on its MAC address E3. Host 1 (and the other hosts on this Ethernet) can add this to their table. The router is acting as what is called **proxy ARP**.

Notice that the Internet (and Tanenbaum

duly reporting what they have done) has developed an entirely different terminology for all of this without ever noticing that this is precisely the same thing a Learning Bridge does except it is one layer up. In IPv6, ARP has been replaced by Neighbor Discovery Protocol, but they still don't realize it.

The idea that the hosts and network are distinct originates with the ARPANET and how it was developed. (BBN developed the IMPs. The Host systems programmers developed the host software. It was a common view at the time. This was re-enforced by BBN defining the IMP-to-Host protocol. However, that changed with IP, which both the routers and the hosts implemented.

It goes further than that. They are treating the hosts as not part of the network, as merely attached to it as opposed to participating in it. This is a defining characteristic of the ITU or beads-on-a-string model. (See sidebar.) If they had used the layered model we use, then the hosts would be an active part of the network.

One still needs ARP, but only locally. In the layered model, when a device joins the network, it must determine who its neighbors are. (We have seen this in the routing protocols discussed.) On a broadcast network where all members are "one hop away," that translates to the address of all stations on the network. Something like ARP or just the routing protocol initialization PDUs would do this. The special case of Proxy ARP becomes a degenerate case of the normal operation of the routing protocol.

The "ARP" request is broadcast and gets back a response with the broadcast address and IP address of its neighbors. Now the routing algorithm is run by each router and host and generates the forwarding table, which maps destination addresses to the outgoing interface. Thus, Host 1 will find that to send a packet to Host 4, it should send it to E3. For a singly-homed host, this forwarding table has one entry: the router it is connected to. For a multi-homed host, the forwarding table has an entry for each interface. For a host on a broadcast network, the forwarding table contains an entry for every address on the broadcast network, with the routers having an entry for each prefix they are a route to. The problem is solved by normal operation. Admittedly, the Learning Bridge was a reaction to not repeating functions in a layer. That view precludes a much simpler solution.

MPLS

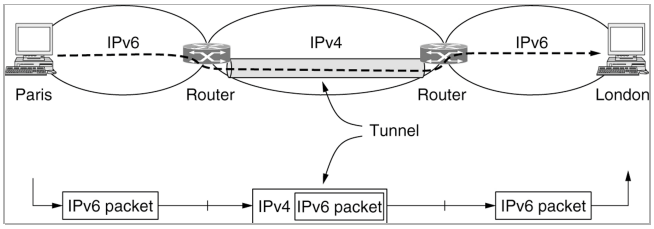
Read

Read Tanenbaum Chapter 5, Section 5.7.5, pp. 476–479.

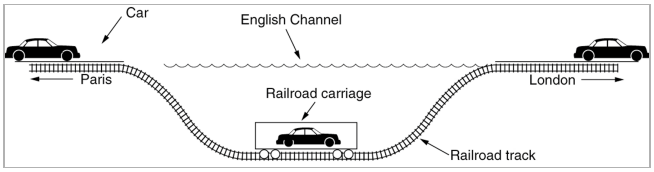
One of the techniques developed for traffic management is called MPLS or Multi-Protocol Label Switching. This is basically a strong connection-oriented technique that supports QoS by essentially hard-allocating resources. It is remarkable that the IETF, which has been so fervently connectionless, would adopt such a solution. It either shows the influence of some router vendors or the inability of the IETF to see datagrams as more than an end.

As far as approaches to QoS go, this is another form of overprovisioning, and it is expensive. MPLS is sort of a network layer protocol—the 1990s equivalent of ATM on steroids. This technology sits under IP and with IP running over a connection-oriented scheme. Like most connection-oriented proposals, instead of addresses, MPLS has “labels” or channel-ids, as we saw in circuit switching. The label is only unambiguous within the MPLS network and identifies individual flows. If there is an interruption (line goes down or MPLS router crashes), the connection has to be re-allocated as with any connection-oriented technology.

Connecting Endpoints Across Heterogeneous Networks

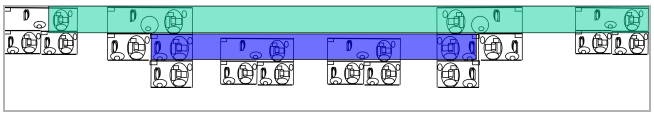


No, actually this is tunneling. Tunneling a packet from Paris to London.



Tunneling a car from France to England. But the Internet doesn’t “tunnel” the whole car.

Connecting Endpoints Across Heterogeneous Networks



These figures provide a good example of how MPLS is used in the Internet today.

The other thing that is interesting is that the MPLS PCI has three bits of quality of service and a Time to Live field, which is copied from the IP PDU that is encapsulated and copied back into the IP PDU when exiting the MPLS subnet. MPLS, it would seem, is and isn’t a separate layer.

OSPF (Open Shortest Path Forwarding)

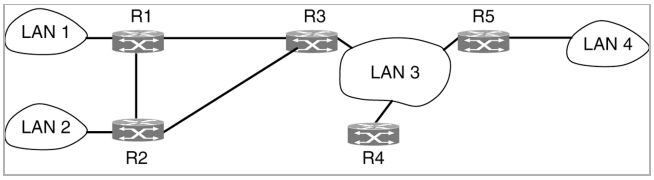
Read

Read Tanenbaum Chapter 5, Section 5.7.6, pp. 479–484.

As we saw earlier, the ARPANET used the Distance Vector Bellman-Ford algorithm. However, as the network grew, the slow convergence and the count-to-infinity problem required switching to a link-state algorithm. However, Link-state requires complete knowledge of the graph, which is itself a scaling problem and we can't have every router in the Internet exchanging router information with all the others. So, link-state is applied within AS or as an intra-domain routing algorithm, while BGP (which will be covered next) is used over ASes.

In 1990, the OSI work standardized IS-IS, Intermediate System to Intermediate System, as an intra-domain routing protocol, link-state based on work by Dave Oran at DEC. Work didn't begin on OSPF until 1988. Because of the close cooperation between the two groups (actually, a large overlap of the same people), the two protocols are more alike than different. However, IS-IS is easier to configure and more stable. IS-IS is preferred by most ISPs and is adamantly the protocol of choice in the BU network. OSPF had a long list of design goals:

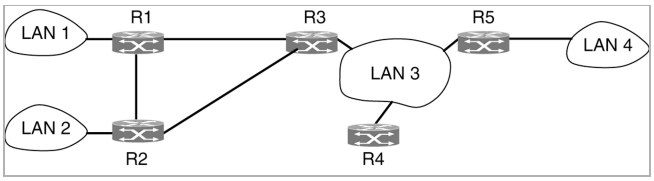
1. It had to be non-proprietary. (This is what Tanenbaum says. Actually, at the time, the squabble was that RFCs were free and ISO charged for their standards, which was their only source of income.)
2. It had to support a variety of distance metrics.
3. It had to react quickly to changes in the network graph.
4. It had to support routing on Type of Service (which is the same as 3).
5. It had to do load balancing.
6. It had to support a hierarchy of subnets.
7. It had to have some security.



(a) An autonomous system.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

The protocols start by abstracting the network into a directed graph with a pair of arcs between each router (so that different weights can be given to them).

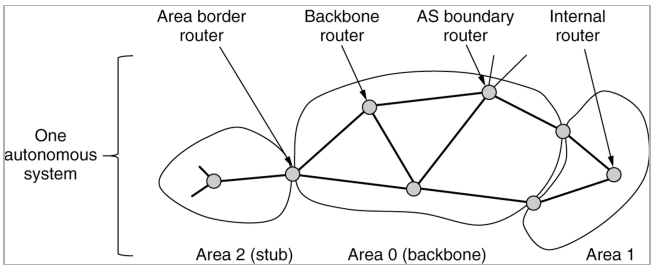


(b) A graph representation of (a).

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Broadcast networks are represented by a node with arcs running from each router-node to the network with weights, but arcs from the network to the nodes with weight 0. See LAN 3 in the figure. Networks with only hosts have a weighted arc going to them but none returning. See LAN 4 in the figure. This is because the hosts are not transit routers. (Notice there is no example here of (nor does the text cover) the case of a multihomed host, neither one that does not act as a transit nor one that does.) Based on this abstraction, each router runs the link-state algorithm to get the set of shortest paths. Some paths will turn out to have equal cost and are remembered for load leveling. This is called Equal Cost Multi-Path (ECMP). It would be better if a range of costs were considered “equal.”

OSPF—An Interior Gateway Routing Protocol



The relation between ASes, backbones, and areas in OSPF.

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

To control router table size and scaling for link-state routing, ASes are divided up into areas. Areas don’t overlap but not all routers have to be in an area. All areas connect to a *backbone area*. Routers entirely within an area are called *Interior Routers* and calculate routes for the area. All destinations are visible outside an area, but the graph of the area is not visible.

Area Border Routers (BRs) are connected to two or more areas and to the backbone (one level hierarchy). BRs summarize (who is there but not the graph) routing information for an area and exchange the information with other BRs. If there is only one BR out of an area, then it is a Stub Area. Summaries are not forwarded for Stubs because the only destination out of a Stub Area is the BR. AS Border Router has a

routing table for all other ASes it connects to. An Area BR has to have a routing table for each area it is connected to and runs the algorithm for each of them. You can begin to see why assigning addresses with common prefixes has some advantage. I have been told that link states can scale up to about 100,000 routers.

BGP (Border Gateway Protocol)

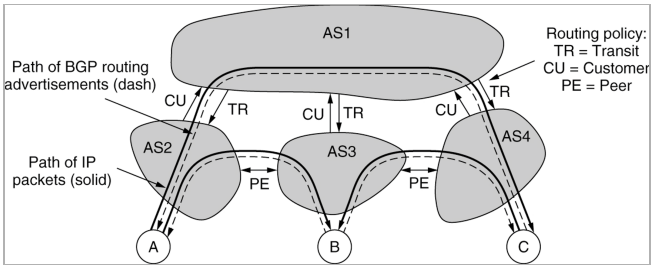
Read

Read Tanenbaum Chapter 5, Section 5.7.7, pp. 484–494.

Link-State is used within an AS, but BGP, a very different protocol, is used across ASes. Where intra-domain can focus on being efficient, BGP must take into account politics and business issues. This includes issues like: don't use our competitor as a transit network, use the cheaper provider, don't route U.S. DoD traffic through Iran, don't route Apple traffic through Google, etc. These routing policies can get very specific.

A good example of the kinds of policies one might see are in what is called settlement. On the Internet there is a more or less a "gentleman's agreement" that if the amount of traffic flowing between two ISPs is roughly the same, they don't charge each other. However, this isn't the case for many, so they have to buy transit services. The provider advertises the routes of its customers to all of the Internet, and the customer advertises all of the destinations on its network to the provider.

BGP—The Exterior Gateway Routing Protocol



Routing policies between four autonomous systems

Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).

Using Tanenbaum's example, suppose that AS2, AS3, and AS4 all buy transit service from AS1. For A to send a packet to C, it must go from AS2 to AS1 to AS4. AS4 is advertising to AS1 that C and others are on AS4.

If AS2 and AS3 exchange a lot of traffic, they establish a peering relation directly, so that traffic for A to B can go directly to AS3. Suppose that AS3 has the same relation with AS4, so that B and C can exchange

data directly. But these relations are not transitive. AS2 cannot send packets to AS4 through AS3. Once again, notice how important it would be for all destinations in each stub AS to have the same prefix. It greatly reduces the amount of router information to be exchanged. However, this doesn't necessarily prevent traffic from A to C through AS3 if there's an application layer relay in AS3.

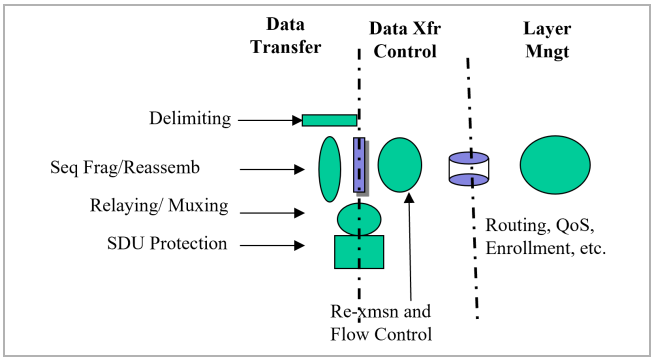
BGP is roughly based on distance vector but is used quite differently than it would be as an intra-domain routing protocol. Given the differences, some have considered it more *path vector* than distance vector. BGP uses TCP to do its routing updates. This effectively creates a separate graph of border routers over the intra-domain routing. But this is only true of the updates. It is incorrect to assume that BGP picks the best route at the AS level and the inter-domain protocol finds the best route within the AS. The two are much more tightly integrated. Although it is unclear in the text, the nature of this is that different border routers in the same AS may make different decisions. There is also the ominous statement that “Care must be exercised by the ISP to configure all of the BGP routers to make compatible choices given all of this freedom, but it can be done in practice.”

The student should read Section 5.7.7 to understand something of BGP.

BGP updates carry the prefix and the sequence of ASes and next hops. The whole description is based on picking the shortest path or least costly path. This makes detecting and breaking loops easy. If the same AS number appears twice then there's a loop. This is essentially a means to automate fixed routing. It is not clear from what is in the book how the non-metric requirements, such as “don’t route through competitors,” etc. are communicated.

Where Does Routing Go?

The Components of a Layer Entity, or IPC Process



This is precisely the system structure we want to see emerge. Where fast cycle time tasks (Data Transfer) consisting of simple functions that are decoupled through a state vector from more complex tasks (Data Transfer Control) with longer cycle times that are

decoupled through a state vector (routing database) from still more complex tasks (Layer Management) with even longer cycle times.

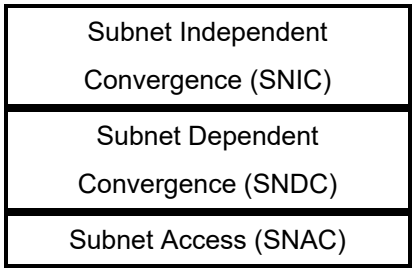
We have already seen the structure for data transfer and data transfer control. It splits nicely between feedback and non-feedback functions. But where does routing go?

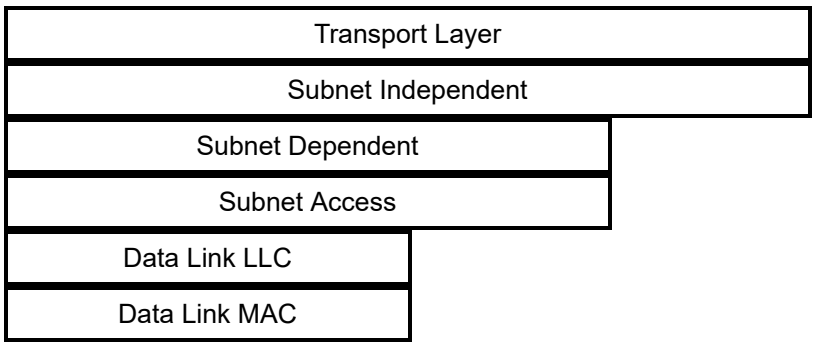
Some say that routing is an application and, strictly speaking, it is. But routing is part of the layer management or part of what some people in the industry might call *autonomic management*. We have seen that Data Transfer is the high-performance function of the layer (fastest duty cycle) decoupled from the loosely coupled mechanisms through a state vector, which are the feedback functions of Retransmission and Flow Control (with a longer duty cycle, but still reasonably fast), which are decoupled through a much larger “state vector” from routing and other much longer duty cycle functions. This “much larger state vector” is given a different name, called the *MIB* or *Management Information Base*. The MIB holds the information of routing and resource allocation.

As I have emphasized throughout this course, this is really what you want to see emerge. This are is hallmark of a good system design, and even more so in this case, because it appeared naturally. We didn’t have to force fit it.

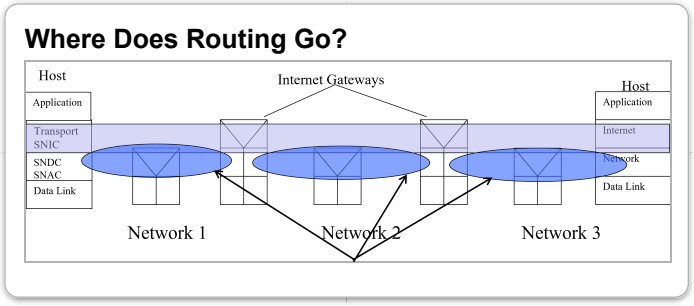
Why Is It Layer Management?

Remember OSI rediscovered the INWG model but had to partition the network layer into three sub layers like this.





The idea was to do an overlay over different networks. They ended up with a structure that looks like this, where you have data link layer and routing within networks, what we have called *intra-domain routing* or *AS routing*. This network layer would be of potentially different technologies or different networks, and then the overlay layer, an Internet layer that avoids the n x m translation problem. This leads to the structure they were trying to achieve that something that looks like this.



What does intra-domain routing use to do its routing updates? The layer below, obviously: the Data Link Layer.

Therefore, intra-domain routing must be layer management in the Network Layer.

What does inter-domain routing use to do its routing updates? The layer below obviously: the Network Layer. Actually, they use the SNDC part of the Network Layer.

Therefore, inter-domain routing is layer management in the Internet layer.

Network routing is in the Network Layer; Internet routing is in the Internet Layer.

Simple! Straightforward.

Then why is it so complicated?

Because in the Internet, we have to fit these ideas into this structure.

It is a long story, but the Internet didn't know about the INWG or OSI results. Furthermore, the attitude developed that if OSI did X, then the Internet wouldn't do X, even if it was right. This led to not doing an architecture. By 1983, the term *Internet gateway* had disappeared and was replaced by "*router*", thus losing the last vestige of an Internet layer.

From the beginning, the Internet group didn't really like layers, but they could never find an alternative that worked as well. They lost the Internet layer. Recently, a discussion broke out on the Internet history list by a group of well-known people in the Internet, including Vint Cerf, as to why BGP used TCP and OSPF used the link layer for updates. They discussed it for about two weeks, but they never came to a conclusion. I was very busy at the time, and didn't have time to jump in and explain it to them. But this is what they were trying to figure out.

So, what happened?

The guys who had rediscovered the IWNG structure did it anyway, without the layer structure. Very few people knew the details anyway and the nomenclature helped cover up what was really going on.

Well, that is just as good, right?

Unfortunately, it's not. It only solves this one problem. There was a solution that leveraged the architecture, solving this problem and several others that hadn't risen yet but were coming. That solution did it at no additional cost. The problem is that their workaround precluded the general solution.