Introduction to Networks

Introduction to the Course

Why We Are Here

The university is a place where a student “apprehends the great outlines of knowledge, the principles on which it rests, the scale of its parts, its lights and its shades, its great points, and its little, as he otherwise cannot apprehend them. Hence, it is that his education is called ‘Liberal.’ A **habit of mind is formed which lasts through life**, of which the attributes are, freedom, equitableness, calmness, moderation, and wisdom.” —John Henry Newman, The Idea of the University, 1856

I believe that this is what a university education is all about. Some may disagree. This is why I came to a university over 50 years ago. The real crux of the matter is that we are here to create a habit of mind that lasts through life.

This course is about learning how to think critically to see through the hype. Networking is going to be the case study. This is not about memorizing what is in the book and repeating it back on exams. This is about understanding. This is the really important because there are not always courses around for the knowledge you need to acquire or the problems you need to solve. You must have a critical mind about what you learn.

In your career, you are going to see a lot of papers by researchers, white papers from companies, and other material. Some are correct; many are exaggerated or even flawed. You need to be able to see through them. My contribution is meager. I’ve never figured out how to teach this in any way, other than by example, and that’s why we have this course.

Networking is unlike other topics:

Security is a specialization a lot of you are specializing in;

Analytics is a specialization;

Databases is a specialization;

Operating Systems is a specialization;

Software Engineering is a specialization.

But networking is a generalization.

Networking requires all of those.

To understand security and secure a network or its equipment, you must understand how the network influences security and how security influences networking. There is strong evidence that strong design is as important to security as security. Networks are very dynamic systems. We must be able to analyze the network, hence data analytics is important. Databases are used extensively for network management. Being very dynamic, concurrency is a major issue in networking, which requires a clear understanding of the discipline of software engineering. Networking is all about dynamic resource allocation and very closely involved with operating systems. All of these things are important, as well as what is unique to networking.

Our Methods

Over the years, I have come to use several of methods in my own thinking of how to go about these things. I have to say, one of the advantages of being an old guy is my education in computer science is very different than yours. I always say that we came through computer science at the tail end of when you could know everything that was going on in the field. Because there wasn't that much going on. Not only did we have much broader exposure to the field, but we were also exposed to much more outside the field. Over time, I have assembled various methods that have proved useful. They will be our guides through networking:

“A well-defined problem is a problem half solved.”—Charles Kettering. You probably don’t recognize Charles Kettering. You’ve no doubt heard of the Sloan-Kettering Institutes, the Sloan School at MIT, and other institutions. Sloan and Kettering were the two who saved General Motors in the early part of the 20th century. Sloan was the business guy; Kettering was the engineer. Kettering was very well-known public figure in his time. This is one of the things for which he advocated.

A lot of people don’t want to take the time to define a problem precisely. They don't want to sit down and really think hard about what the problem is talking about. What is it saying? What are its constraints? What is it you know? I have found that if you are very careful about defining the problem, you’ll see that he’s right. Once you get it well defined, often, the problem has solved itself.

The problem with reductio ad absurdum is knowing when to stop. That’s a bit of a contortion of what reductio ad absurdum means, but the idea is there. Any models or principles are only effective within a certain domain; outside that domain they become more of a hinderance than a help. One of the dangers is not recognizing when a principle has been taken too far, to the point where it is no longer effective or is making the problem more complex.

Look at the problem from the point of view of the organism, not the observer. This is especially important in networking. We were very fortunate to study with and be good friends with two of the four founders of the field of cybernetics. Heinz von Forester always stressed this, drummed this into our heads. He had a very interesting way of making the point. Von Forester was Viennese and very charismatic, but he also didn’t tolerate sloppy thinking. To illustrate the importance of the organism, rather than the observer, Heinz would tell a story:

> Heinz was not a big fan of B. F. Skinner. He would illustrate this by saying, “We are going to teach an urn. We're going to put an equal number of red balls and white balls in an urn and we're going to teach the urn to give us red balls. The way we do this is: we reach in the urn and pull out a ball. If it's a red ball, we **reward** the urn and we put the ball back. If it is a white ball, we **punish** the urn and we throw the ball away. By this means, the urn learns to only give us only red balls!” Clearly, this is absurd. But the point is that the reward and punishment is in the eye of the observer, not the organism, the urn. Often, you will see this in networking. What you might know or think you know (as an observer) looking at the network is not what the elements of the network know.

Model building—look for the invariances. This is something I always described with various adages: the separation of mechanism and policy, which is discussed in operating systems, and is especially useful here; the old math joke that I suffer from the topologist’s vision defect: I can’t tell a coffee cup from a donut! (They are both what in topology are called *genus 1 surfaces*. They both have one hole, the hole in the donut and the handle of the coffee cup.)

So what you’re saying when you say, “we need an abstraction of the problem,” is that you’re factoring out the important invariances. One of the things we have learned is that when the invariances are pulled out and the solution built without breaking those invariances, there are no “devils in the details.“ In fact, there are angels. This is a bit of an elaboration of our quote from Kettering.

Throw away the ladder. I told the von Forester story about the urn. von Forester was supposedly the nephew of a very famous philosopher, Ludwig Wittgenstein, who wrote the *Tractatus Logico-Philosophicus*. Bertrand Russell wrote the introduction and claimed it destroyed 2000 years of philosophy. If you’ve ever said, “we have to define our terms,” it’s because of the *Tractatus*. It is one of the densest books you will ever encounter and very important not only for philosophy, but logic and the fundamentals of computing, and it gave rise to the Vienna Circle. (For fun, find a graphic novel, called *Logicomix*, about the early 20th century mathematicians Cantor, Frege, Gödel, Poincaré, Russell, etc. who did this groundbreaking work. The book tries to make the case that doing logic will drive you crazy.)

It is written as all single statements in outline form, all the way through with statements, sub-statements, etc. Statement 1 starts:

> 1 The world is all that is the case.
>
> 1.1 The world is totality of facts, not things.

It goes through six major statements like that, deriving truth tables, etc., that math is a tautology, that says nothing, and ends with

> 7 that of which we cannot speak, we must pass over in silence.

In other words, if you can't define clearly what you're talking about, SHUT UP! (I am told that the original German is closer to my version.)

But the important one for me was the statement right before that, which is this:

> 6.54 My propositions serve as elucidations in the following way: anyone who understands me eventually recognizes them as nonsensical, when he has used them —as steps—to climb up beyond them. (He must, so to speak, throw away the ladder after he has climbed up it.)
>
> He must transcend these propositions , and then he will see the world aright.

We will see this happen with networking where we will develop an idea, and once we have developed it, we can better see what it's saying and see a better model as a result. We can then adopt the new insight and throw away the ladder and move on in a much better way, with a much clearer idea of the solution.

All scientific principles only work within a given set of bounds. I often cite here a wonderful play by Tom Stoppard called *Rosencrantz and Guildenstern Are Dead*, which illustrates that principle quite well. There is a running gag in the play that scientific principles only work within certain bounds. (All of Stoppard's plays have a scientific connection. Look on YouTube for *Rosencranz and Guildenstern are Dead: Gravity Question* to see semantic tennis and Newtonian physics.)

At one point, there was a fad that everything had to be RPC—Remote Procedure Call. I was shocked to learn that Digital Equipment Corporation (DEC) in their VAX operating system was using RPC for the low-level character handling software, the terminal device drivers. I've written that code, and that is the last thing you want to do at that level. It is entirely too much overhead. In fact, it is such a bad idea that it wouldn't even have occurred to me to consider it and discard it as impractical, let alone user it! But RPC was supposed to be the solution for everything. Except nothing ever is. They tried to take it too far. It becomes overhead that you don't want, but you need to shift the model. Another example: One of the things we'll see in networks later is that at the top of the application layer, one should shift from an InterProcess Communication model of networking to a programming language model.

There are places, where that happens. You always have to be sensitive to the signs that the model being used isn't fitting nicely. It usually means that the invariances are wrong and need to be re-considered.

And all this can be summed up as "listen to the problem." This is one of the things that I emphasize: Do what the problems says. It understands things better than we do. If you don't, the problem will extract a cost. The longer one waits to do what the problem says, the greater the cost to fix it. However, learning to listen to the

problem is difficult. There is a lot of trial and error but eventually you get there. (A common mistake is to start at too low a level of abstraction.)

Those are the sorts of guidelines, and not so much rules, that I've used to recognize what I call “first-order effectors.” These there are things that we will see, along with principles, as we go through this. Principles are usually the big general statements we will uncover. First-order effectors are more subtle. Often, they are little things that, when they are done right, allow everything to go smooth as glass. But get them wrong, and the whole system design is broken no matter what else you do. Those are the ones you want to stay away from, but you see them quite frequently. A quick example that we will see much later: The number of active elements in a Bluetooth network is 7. My reaction when I saw that was, “Good grief! They didn't put a three-bit field in the protocol? It is going to be very difficult to increase it.” But, yes, they did. And we will learn that it wasn't even necessary!

These are the little things that we can pick up. They will likely drive your colleagues crazy because without doing a thorough analysis you will be able to pick out the problems. Then later, the thorough analysis can be done to confirm your suspicions.

Introduction to Networking

Finally, we move on to Tanenbaum and the details of networking. We will start with Tannenbaum, then we’ll shift into a more historical view and a little later on.

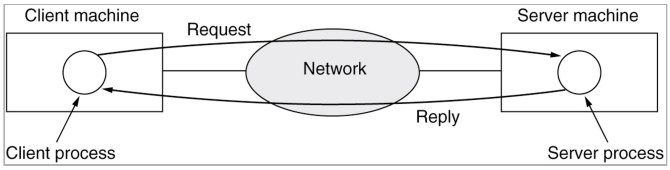
Introduction

Read Tannenbaum Chapter 1, Section 1.1, pp. 1–7.

Uses of a computer network: You all use networks all the time. You know you’re using them daily. You know all about most of this, from a user’s point of view at least, even if you aren’t familiar with how it works internally.

Access to Information

Access to Information

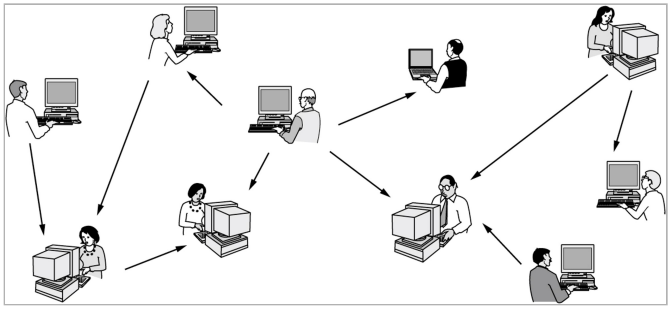


Communication takes the form of the client process sending a message over the network to the server process. The client process then waits for a reply message.

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

There is a strong emphasis on client/server. This is basically the mode that harkens back to the IBM SNA of the mid-1970s. It is very simple. Most of its uses are synchronous, i.e., when a request is sent, the process is blocked until there is a response. It is much more interesting when it is asynchronous and there can be multiple requests outstanding. As we will learn, the Internet today is, for the most part, limited to this mode of interaction. This has pushed us toward a more centralized use of the network: the idea that generally there's little functionality at the client side and more functionality at the server side. We will see examples where the opposite is the case for the same reasons.

Access to Information



In a peer-to-peer [sic] system, there are no fixed clients and servers. (There never were. The Internet has been a peer network since it was turned on.)

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

Peer-to-peer [sic] has generated a lot of excitement. Supposedly, what is so "innovative" is that a system can be both a client and server at the same time. However, this has been true since the internet was "turned on" in 1970. They have done some interesting forms of name resolution, e.g., name look-up, that can grow from amorphous caching to a hierarchical directory.

We could spend a whole course discussing the advantages and disadvantages of social media, messaging, etc., but that would be more of a sociology course than a networking course. It appears that we are learning that there can be too much of a good thing.

Electronic Commerce

Electronic commerce has become very important, so much so that it has caused many brick and mortar businesses to fail. This is not all for the good, especially where bookstores are concerned.

Tag	Full Name	Example
B2C	Business-to-consumer	Ordering books online
B2B	Business-to-business	Car manufacturer ordering tires from a supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products online
P2P	Peer-to-peer	Music or file sharing, Skype
Source: Tanenbaum, A., Wetherall, D., & Feamster, N. (2020). *Computer Networks* (6th ed.).		

This table is typical of what we will see. It represents marketing more than any actual technical differences. It is doubtful one could find significant differences among any of these. To make distinctions where none are necessary and fewer would be much more beneficial. It is important to see through false distinctions like this and see them for what they are: marketing hype. While it is true that these specific kinds of business interaction exist, they have little or no relevance to networking. People became attached to the abbreviations of B2B, P2P, etc.

Internet of Things

Internet of Things (IoT) has been the latest “hot” buzzword. Everyone thinks it is quite new, when, in fact, we have been doing it for over 50 years. At the very least, network management is a form of IoT. There has been a proliferation of products that are serving more to create market fragmentation than real value. The most important aspect of IoT is consistency and commonality across object models in related domains of use. People love to create new terms for the same things to make what they are doing sound different.

New buzzwords come and go regularly in this field. It is done to generate excitement around new products that have been around for a long time but with new players (at new price points) entering the business. While these products are touted as new areas of innovation, often one finds that they’re not all that new and that the aspects that could be innovative are ignored. In this case, we have been doing IoT in one form or another for 30+ years. Probably the biggest use of IoT has been network management. Basically, network management is IoT controlling network devices over the network. What is often ignored by the new hype (which doesn’t see that network management is IoT) is that if there is an Internet of Things network, then that network must be managed as well. Why have two different architectures, protocols, etc. for basically the same task with different object models? People have been proposing and building SCADA networks to control electric grids for least 30–40 years. IoT has been around a long time. It’s just that now the cost has come down to the point where it can be a lot more ubiquitous than it was before. (Thirty years ago, a major U.S. utility asked me what the differences were between managing an electric grid and gas distribution. Thinking about it for a moment, my reply was, “in the grid, events happen at the speed of light; in the gas network, they happen at the speed of sound. Otherwise, they are pretty much the same.”

There appears to be a lot of diversity in the IoT field, but from our work, it is clear the diversity is not necessary. It really is primarily being done to create barriers to entry, i.e., barriers against competitors. It has also become clear that there are much greater advantages to leveraging commonality than differences.

Policy and Social Issues

In the last few years, this has gone from a minor problem to a major problem, with everything from online speech to the proliferation of misinformation and fake news. It is just incredible how outlandish the information can be, and that some people will believe it. It boggles the mind.

For several years, *net neutrality* is a big issue because it became embroiled in combining commercial and engineering issues that should not have been confused. When we get further into how things work, we should come back and talk about net neutrality a bit more. A big part of the controversy happened because certain technical issues weren't addressed, so the technical aspects (merely trying to manage the network effectively) were indistinguishable from the commercial aspects (discriminating against your competitors). These two things should not have been combined. Now it has pretty much gotten to where the two things have been separated again. So the question becomes, how do we distinguishing between the discrimination aspects and the traffic engineering aspects of net neutrality? (We will look at this in more detail later.)

Security, of course, is a major problem. There is a general belief that the bad guys are ahead of the good guys, and there is not much chance that the good guys are ever going to catch up. The general excuse is that the network was not designed with security in mind. It is believed that security can be retrofitted. It never works. It has been known since the mid-70s that a system must be designed for security from the beginning, but management always wants to do it later. But we will consider this in greater depth as we get further into the course.

Privacy, as you are probably aware, is a big issue. People are tracking what we look at, what we buy and sell, and what our habits are, and they are learning things about us that are often incorrect or that we don't even know ourselves.

There are a lot of really touchy issues here that need to be dealt with and it's not at all clear how we're going to do that. But this is more an application issue rather than a network issue. Moore's Law and the Internet have been enablers more than causes.

What Really Characterizes Networks

The characteristics of the media create bounds.

- Is it point-to-point or multi-access?
- Are there physical limits on distance?

- What are the error characteristics of the media?
- What is its bandwidth? What is the data rate?
- What is its capacity?
- If media is wireless,
    - What are its propagation characteristics?
    - What will the radio waves pass through?
- And, of course always an issue is cost, cost, cost.

We talk about bandwidth as if it were speed. After all, it is units of bits per second. At low rates, it does act like speed. But what it is, really, is capacity. For example, if we send a large file at 56kbps, it takes a significant amount of time to receive the file. Let's say 5 minutes. If I send it at 100kbps, we receive the file in half the time. We get it measurably sooner. If I try to send two files at the same time, interleaving them, it could take twice as long for either one to be delivered.

Jumping ahead a bit, what is going on here? First, we know that the pulses that represent the data are traveling at the speed of light. They have to. It is physics. So what is "bits per second"? How much time do the bits take? A pulse for a bit at 56 Kbps is longer than a pulse for a bit at 100 Kbps. We can actually compute the physical length of a bit (or a message) in meters. At low bandwidths, we might still be sending the last bits of a message while the first bits are already arriving at the other end. If the length of the wire is long enough and the bandwidth is high enough, the sender may finish sending long before the first bits arrive at the other end. Then the sender could start sending the bits of another message that might be or might not be part of the same file. The capacity of the line has increased from one message at a time to more than one.

If I send the file at 1 Gbps, I might receive it in a half second (500ms). If the data rate is doubled, I receive the file in 250ms. For a human, that is not much difference. But in terms of the network, I could also be sending many more files and not significantly increase how soon any one of them is delivered. The bandwidth is acting more like the capacity of the media to carry bits than the speed with which it carries them.

On the other hand, if the file was for a process control application in a chemical plant the difference between 250 ms and 500ms could mean the difference between destroying and not destroying the plant. But there is still the capacity of the media, so the sender might want to send the important messages first. Sometimes it is speed, and sometimes it is capacity.

The Importance of Market Overlap

An important aspect of product design in networking is that network technology should not be built to support specific applications because there is a good chance that markets will be missed. The nature of the trade-offs in building product means that network technologies must be built to support a specific operating region. There's a certain range of the various properties of error rate, capacity, bandwidth, etc. that that equipment is optimized for.

Different market segments require different operating ranges.

Some market segments may require the same range. The challenge of designing a network or the equipment is that you need to match the operating ranges of the equipment to the operating ranges of a collection of applications, i.e., the market segment.

We have seen it happen where a company decides they're going to go after the banking market. This market requires equipment that does certain things. At the same time, the company is completely missing that there's another industry they are not targeting that has almost the same requirements as banking, and they could sell the same equipment to that market as well. They end up making a product that is just narrow enough that they completely miss the other market, leaving it to a smarter competitor.

It is important to have a clear and distinct understanding of the operating regions required by the markets the company is targeting, and separately an understanding of the operating regions that engineering trade-offs can effectively cover. Engineering should focus on making the operating regions as general and broad as is practical, while marketing should focus on the exact opposite: defining the operating regions to be as specific as possible. Marketing should be continuous evaluating potential markets. When a particular market is quite successful, engineering should optimize a product to lower cost. It is then a question of matching market operating regions to engineering operating regions. In this way, a rational decision can be made about which markets are compatible with where the engineering effort is being expended.

Word of Warning

In the early 80s, engineering in several companies saw a market for a router. Marketing (and VCs) insisted if there was a market, it wasn't large enough to be worth pursuing and squashed the effort. We all know where that went. (Cisco avoided the problem because they had 100 customers before they needed investment.) Marketing tends to have 20/20 hindsight and quite myopic foresight.

Management-Proof Development

As long as we are on this topic: suppose management tells you that the product has to support some limit, such as an upper bound like small 3-node networks or 100 simultaneous flows, etc., where it would be advantageous to use a less-than-general implementation. Don't believe them! It is virtually guaranteed that before you can get the equipment built, they'll come back and raise the limit. What you should do is adopt the general solution that works best. Put an upper bound of what they told you as a compile time variable.

When they tell you the limit has to be raised, complain like the Dickens! You know, "What you are telling me means a complete redesign! You promised me I only had to do that upper bound! That is a huge change! Good grief![1] What am I supposed to do!? Etc. Okay, if we are going to make the deadline, I have to be left alone to redesign and rework the implementation. Give me three uninterrupted weeks. No meetings. I will turn off my phone." etc. etc. Go to Aruba for two weeks, then come back, change the upper bound to the new number, re-compile, and you're all done. You beat the deadline and you are a hero and can ask for a raise.

1. More emphatic expletives are encouraged if appropriate. There is only so much I can do here.

The Environment of Networking

The environment of networking is a little strange, because it is more normal than most. All endeavors are affected by economics and politics, but not quite so much as networking.

In the early days before networking, selling computers competed on price and capability, but almost not at all on moving code between different computers. Each computer installation was a world unto itself. Within that world, you could do pretty much what you wanted. There were standards: character sets, programming languages, etc. But if they differed some because the systems were different, it was not that important. There were many more problems with moving code from one system to another. Networks are different. Different equipment has to talk to each other. They must use the exact same messages, or they won't understand each other. Consequently, getting agreement, down to the details, becomes very important and destroys traditional barriers to entry from competition.

We must understand where networking has come from for two reasons: first, to make sense of the less-than-perfect world where we find ourselves, and second, only by understanding the processes can we affect them.

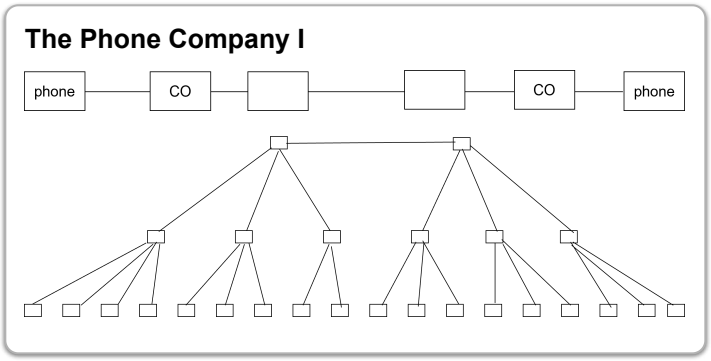
I will be telling you various stories about what I've seen happen. I'm not telling you the stories because I like the stories or because they are entertaining. I'm telling you the stories because you *will* see the same things play out either in companies or in groups where you end up working. Be aware of the games people play, because they are rampant in this field. You may find that you need to use them or know how to counter them. As the old saying goes, those who don't know history are doomed to repeat it, and so, far we've been doing a pretty good job of that. Along the way we will pick up a few basic networking concepts.

The Early Telephone Network

Computer networks are not the first networks. In France in the early 19th century, there was a sophisticated semaphore (flag signaling) network that had a lot in common with the signaling of early data networks. But for our purposes, the early telephone network is a good place to start because we are still dealing with some of its legacy.

Believe it or not, the early telephone network has a rather odd history. In the beginning, after Alexander Graham Bell invented the telephone and turned it into a business, one would go to the telephone store and buy two phones and a lot of wire. You would give one phone to the person you wanted to call, string the wire over to your place, and hook up the second phone. Lots of people started doing this. Wires were being strung on anything people could find—along fences, over tree branches, etc. This was especially true in the cities of the

East Coast. It got to be a real mess and people complained about it. Finally, the phone company decided to (and/or city councils made them) do something about it.



They created central offices and ran all of the lines (on telephone poles) from their subscribers to the central office (one wire per subscriber). At the central office, there was a big plug board, as you have perhaps seen in old movies. When a subscriber picked up the receiver, it would go off-hook (completing a circuit that made a buzz at the operator). An operator would say, “Number, please?” You would tell her the number and the operator would connect you to it on a plug board which would make a *physical electrical connection* between the two phones.

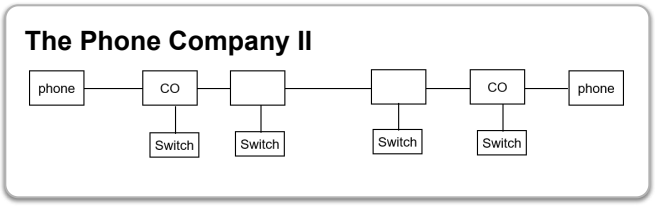
> I remember this. We had two-digit phone numbers. (Well, in a town of 900, how many phones are going to be needed?) Our number was 61 and my grandparents’ number was 55. Even when we had to move to seven-digit phone numbers, we only had to dial the last five digits.
>
> Telephony has always been based on a deterministic circuit model, which I call “beads on a string.” A physical circuit had to be completed from source to destination. Consequently, the network was fundamentally a tree.

Somewhere in the 1920s, AT&T did a study that showed at the projected rate of growth, by approximately 1960, they were going to have to employ every woman between the ages of 18 and 45 to be telephone operators! That wasn’t going to scale. They had to do something, and so they invented the rotary dial and automatic switching. Well, actually, they didn’t invent it. The story goes that it was invented by an undertaker in Kansas City. There were two undertakers in Kansas City at the time. One of them was friends with the telephone operator. When people would call up and say that someone had died and they needed an undertaker, she would send them to her friend. Of course this was hurting the other guy’s business, so he decided to take her out of the loop by inventing the rotary dial. Creating central offices had started the convention of networks that were a tree with central offices, and then regional office connected those central offices.

This replaced the operators in the local central offices with mechanical relay switches but operators were still needed for long distance. You may even have seen in old movies that when someone needed to make a long-

distance call. They would call the operator and say, “Please get me this number in San Francisco” and hang up. The operator would go off and set up the circuit and then call them back and say, “We have your party.”

The system had some advanced capabilities, such as call forwarding, which disappeared with mechanical switches. When my father got called back into the Navy for the Korean War and was stationed in the Aleutians, we lived with my grandparents. Dad would call home and Mom would be off visiting someone in town. My grandmother would say that my mother was at so-and-so’s house, and the operator would re-route the call to her there. That capability didn’t return for 20 years (probably longer in major cities) and, even then, it had to be set up in advance and set to forward all calls to another number, not just selected ones. One can imagine that an operator with local knowledge could provide other “advanced” capabilities, not to mention knowing who was calling whom or even listening in from time to time. Town rumors often originated in the telephone office.



Recognizing that using operators wasn't going to scale, the phone company started developing and deploying mechanical switches. These were introduced alongside the network. After nearly 75 years, the switches became electronic, and when the switches needed to communicate, it was alongside the network.

There were essentially two networks: one for the phone conversation and a second one that was doing the switching. It is important to understand that there were lots of little phone companies throughout the last part of the 19th and early 20th century. Bell Telephone (AT&T), through a lot of patent litigation, essentially bought or put out of business all but a few.

In the rest of the world, there was a very different situation. The phone system in other countries was often created by the post office and became known as the PTT, the Post, Telegraphy, and Telephony. The PTTs were very powerful. They rolled into one organization, what in the United States is the FCC and AT&T. In most of the rest of the world, the PTT is a ministry of the government. It makes the rules for the phone system to follow and provided the phones, switches and the network to follow their own rules. The customers had little or no voice in this, not a good situation.

Deregulation didn’t begin to change that until the mid 80s in the United States and into the 90s and beyond in Europe and the rest of the world. There are still countries where the government provides the phone service or owns a significant part of the phone company. Most all of divested themselves of manufacturing network equipment.

Change Was Coming

With the advent of computers after World War II also came the transistor. As computers were being made smaller and cheaper, it also meant they could be used in more places.

The phone companies basically began to deploy computer-controlled electronic switches. At first, for the computer industry, communications was just a means to an end. The two industries quite carefully stayed completely out of each other's way. This was the era of dumb terminals talking to a mainframe computer. And on the computing side, everything was the mainframe. Most terminals were in the same building (or nearby) and the organization ran wires to connect them. Once in a while, an organization might lease a line from AT&T to connect a distant terminal, but it was rare and expensive. Data rates were low: perhaps 300 bits/sec, or, very rarely, 9600 bits/sec.

There was minimal functionality in the terminals. Everything but the wires belonged to IBM, who was leasing lines from the phone company. The mainframe IT companies and phone companies coexisted at this point. IBM had 85% of the computer industry.

Now, one interesting thing about this is that while IBM had 85% of the industry, there were still 6 to 10 other computer companies dividing up the remaining 15% and being profitable, companies like Burroughs, NCR, Univac, Honeywell, CDC, General Electric, RCA, DEC, Xerox, etc. Not at all like today or even in the beginning of the networking era when Microsoft and Intel dominated as IBM did, and there were only two or three others dividing up what was left. This is a significant change.

Big Changes Were A Foot

In the early 1960s, Paul Baran at the Rand Corporation wrote a series of reports investigating the network, who inquire moments of the Department of Defense. He found that the traffic characteristics for data are very different than those for voice:

> Data is bursty; voice is continuous.
> Data connections are short; voice connections are long.

He proposed that there would be a distinct advantage for a network built specifically for data. It would be different from the voice network and provide greater efficiency and survivability. Data would be sent in individual packets, rather than as a continuous stream as voice was. At roughly the same time, Donald Davies at the National Physical Laboratory in the UK was proposing the same thing.

Packet switching was born and, by the late 1960s, the Advanced Research Projects Agency (ARPA) decided to build a resource sharing network to reduce the cost of research using this packet switching idea. They didn't know it yet, but the real advantage of the network would be to facilitate collaboration.

Keep in mind that computing was a very small field at this point. It was not what all of you are used to seeing. In the 1950s when IBM started to build computers, it was proposed that there was a market for maybe six

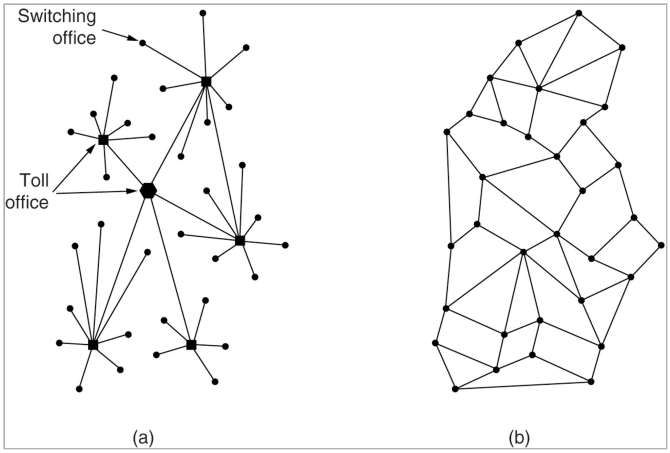
computers in the world. Even by 1970, there was still only one computer conference a year, which was also a trade show, as opposed to two or three a week today. The only computer journal was *Communications of the ACM*.

In 1977, the CEO of DEC, the people who built the first minicomputers and largest employer in the state of Massachusetts, said, “Why would anyone want a computer at their home?” What an incredible lack of vision! (From the first time we saw a time-sharing system, its purpose was to provide the illusion of having our own computer while we waited for the hardware to catch up to make it a reality.)

A Fundamental Departure: The ARPANET

ARPA decided they were going to build this resource sharing network. It was going to be a fundamental departure from the telephone network. Rather than being a hierarchical tree of central nodes that all go to the same place, it would be a mesh network with lots of nodes and lots of different paths through the network.

A Fundamental Departure



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

ARPA put out an RFP, and a company out on Fresh Pond in Cambridge came back with the winning bid. The company, Bolt, Beranek, and Newman (BBN), proposed to build this network using minicomputers as switches, they called *IMPs or Interface Message Processors*.

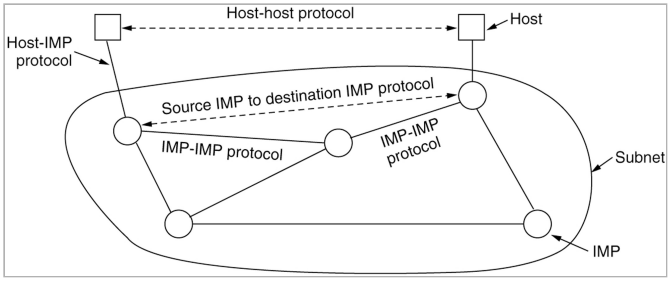
Note: You can forget about the full name of Interface Message Processor. Nobody ever used the full name. They were always just “IMPs.” But if I didn’t mention it, someone would ask what it stood for. This happens often. Much of the time, what an acronym stands for is unimportant. What you want to remember is what the acronym refers to. In this case, an IMP is an ARPANET switch or, in today’s terminology, a router.

The design was split into two parts: the subnetwork, which BBN built, and the various universities and research facilities ARPA was contracting with. ARPA strongly suggested they were going to be on the network. Now remember, they were building this to be a resource sharing network. It was up to each of those facilities to figure out how to get themselves on the ARPANET. It required a combination of a hardware interface and software in each host.

ARPA's idea was that they wouldn't have to buy very expensive computers for every site. The sites could share resources over the ARPANET. There would be sites with big computers—supercomputers—that everybody could use, with modest-sized computers elsewhere to support local work. But the emphasis was on a resource sharing network.

In the early releases, the host had to be within about 10 meters of their switch. Hence, it was assumed that the IMP was co-located with the host. (Later this restriction was relaxed, but rarely used.) The IMPs were a minicomputer, Honeywell 316, which was basically one rack in a hardened military cabinet. They were using very high-speed lines! 50 kilobits per second! That may not sound like much to you, but at the time, dial up was 300 bits per second and, generally, what was available in an office or lab was a blistering 9.6 kbps. So 50 kbps was really, really fast. In fact, very few of the hosts could sustain that rate over a long period.

The ARPANET



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

There was the IMP-to-Host protocol and then the IMP-to-IMP protocol. The IMPs provided a reliable data stream through the network. This freed the Host-to-Host protocol to just provide a basic Interprocess communication (IPC) facility.

Getting to Layers: ARPANET

Operating systems were a major influence in all of this. In 1968, Dijkstra published a very famous paper on the operating system, where he proposed that the appropriate structure was a layered model, where a Layer is a stack of black boxes, each using the layer below to provide greater service to a layer above and hide its internal workings. Furthermore, Dijkstra conjectured that once a function was done in a layer, there was no need to repeat it in higher layers. To attach a host to the ARPANET required writing a new device driver in a low layer of the OS to provide an abstraction of the flow of information from the network. The device driver would have to be

written by the operating system team at each site. An OS perspective was going to be very natural. In addition, networking is a resource allocation problem, just as an OS is.

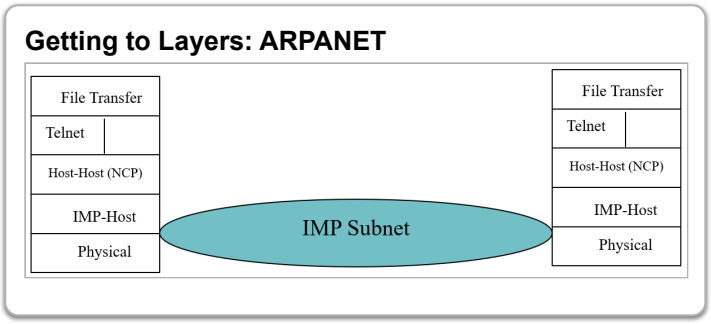
In 1970, Dave Walden, who was in charge of developing the IMP software, wrote an RFC called “IPC for a resource sharing network.” Most everyone saw network communication essentially as interprocess communication. In fact, in 1972, Bob Metcalfe—who invented Ethernet—wrote about how to write an ARPANET NCP, the Network Control Program said, for no particular reason, it wasn’t major point in the paper, “networking is interprocess communication.” IPC was an organizing factor for the ARPANET. In RFC 871, Mike Padlipsky called IPC “axiomatic” to networking. Layers were a natural occurrence in networks.

I’ve always believed that the ARPANET was successful because it was done by OS guys not telecom guys. They looked at this as an operating system. Because it was supposed to be a resource sharing network, we were going to have to move programs around and move files around. That was all part and parcel of the whole thing.

No one had ever built one of these. This was to be a production network. So BBN was working to a deadline to build the network. Once they had enough of it specified, enough of it figured out, and implemented, they started to involve the other organizations that were going to have to put their computers on the internet. To do that, they had to write device drivers to interface to it. I have checked with Dave Walden who was in charge of the IMP development. There are no diagrams of layers in the subnetwork design. It is easy to see from their point of view how they built that first network, when there was no reason to think in terms of layers.

Then, after BBN brought in the teams from the sites who needed to attach their hosts, the Host teams found that the problem looked very layered. (The initial IMP subnet was not layered. There are no layer diagrams of it.) The BBN guys specified the IMP-to-Host protocol that defined how to talk to the network, how does a host talk to its IMP. So the Host teams implemented it as a device driver that provides a service. The IMP-Host protocol created flows with the IMPs. The next task that was needed was IPC between the processes on the hosts. The Host Teams then defined a Host-to-Host protocol. (For some odd reason, the implementation of the Host-to-Host protocol was always called the Network Control Program, or the NCP. It was never called HHP. In fact, even Host-to-Host protocol was seldom used. If you read about this, you will see it is always referred to as the NCP.)

So now that we have this interprocess communication facility, what are we going to do with it? All the things we would do on an operating system: terminal handling, using files, submitting jobs to run, etc. Those functions were layered on top, so that very quickly, the ARPANET had the layers shown below:

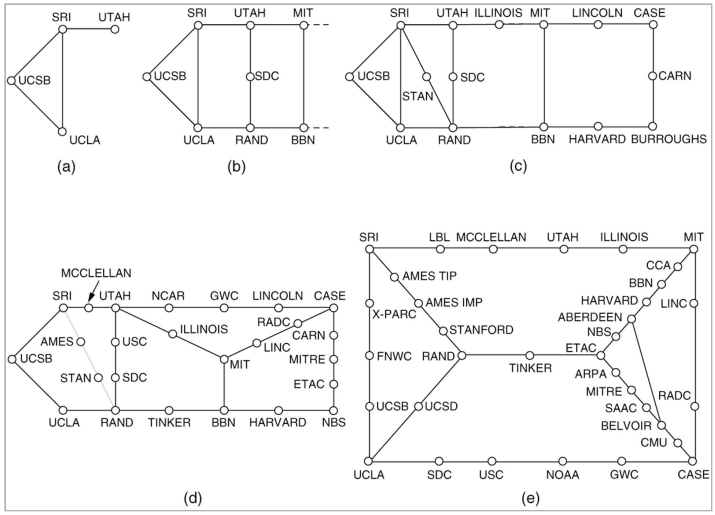


This is when layers first appeared. To this day in the ‘Net, layers remain all about modularity. As we will learn later there was much more to layers.

Growth of the ARPANET

The first three nodes installed were UCLA, UCSB, and Utah.

Growth of the ARPANET



(a) December 1969. (b) July 1970. (c) March 1971. (d) April 1972. (e) September 1972.

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

You can see all the people who were on the machine or on the Net by the end of 1970 and it was growing very, very fast. At the beginning, there could only be one host connected to an IMP, later when they were able to make it four hosts. Your host address was your IMP port number or your IMP number, and so we were host 12. We were the 12th IMP to be installed. They were just numbered in the order they were installed, which was common practice at the time.

While our group at the University of Illinois was the 12th IMP installed in 1970, we were the 6th to come up on the ARPANET, the precursor to Internet. We were designing and writing operating systems to put us on the ‘Net. We were building the first supercomputer, a 64-processor machine. Today that is not a big deal; there are more than 64 processors on a chip.

The ARPANET a huge success. By 1973, within the small group of people that were working on it, there was essentially a mini-boom. The first three applications were created—TelnetFTP and RJE—which we will talk about those next time.

Almost immediately, we were relying on the Internet. All sorts of great things were going on. People were trying everything, such as packet radio, packet voice, and distributed databases. NLS was on the ‘Net. In 1970, Doug demonstrated this at the only computer conference of the year, the National Computer Conference. To understand how impressive this was, consider that in 1970, 98% of the people at this conference still used punch cards and did not even have terminals.

Doug Engelbart at Stanford Research Institute (SRI) developed NLS, the “oN-Line System.” It was a full hypertext system. Engelbart invented the mouse. They couldn't do figures, but everything in NLS was in outline form. The way it worked was, you sat in front of a screen with a five-finger keyset and a three-button mouse. You could point and select with the mouse and chord commands with the keyset. You didn’t really need the keyboard unless you were going to do a lot of text entry. You could do things like pick up a section and all of its sub sections and move it elsewhere or rearrange a document. There was something called viewspecs that allowed you to say, “show me the first two lines of the first three levels of this document,” and so you got an overview of the document.

But the one I always like was the following: NLS was written in a language called L10. A procedure call in L10 looked like a link to NLS. Suppose you were looking at a piece of code, and you wondered what does that procedure do. With your left hand on the keyset, you chorded the command “Split Screen;” with your right hand on the mouse, you pointed and selected the procedure, and chorded the command “Jump Link.” The result would be that you would have the body of the code in the top half of the screen and the body of the procedure in the bottom half. It was very impressive. If you have time soon, on YouTube look up, “the mother of all demos.”

We also put the first UNIX system on the ‘Net in the summer of ‘75. Then, for a land-use management system for the six counties around Chicago that used databases on both coasts, we stripped down UNIX to run on a one-board PDP-11 (the LSI-11). We put that in a pedestal cabinet with a floppy drive with a plasma screen and touch to create an intelligent terminal. You could do everything you wanted to do: display stuff, draw graphs, and plot things, all by touching the screen. The system could do maps down to the square mile. We couldn’t do color (color plasma screens didn’t exist yet), but we could do different shading, using different patterns for various things. People were doing some pretty sophisticated stuff. We were also doing what you would call instant messaging in 1971. By 1972, I had a terminal at home where I would spend the evening instant messaging about designing software with people in DC, Boston, California, and Illinois.

By mid 1973, the Users Interest Group (USING) was formed to make the Internet a true resource sharing network with new protocols and advanced capabilities and to be an advocate for the users. By mid-1974, ARPA shut down the USING group for fear they were losing control of the network. In reality, they already had; they just didn’t know it.

Tensions Build

But big things were happening in the outside world. Tensions were building. This was the beginning of a paradigm shift. These new packets switched networks required fairly complex software. The most complex software of the day was operating systems. Not only that, but this was fundamentally an operating system problem. Telecom was unfamiliar with and suspicious about this new software stuff. These new networks were developed by people with operating systems backgrounds. Not everybody was making the shift as quickly.

Was Packet Switching a Major Breakthrough?

Packet switching had been the first part of the idea, but it was not all that radical. Radical change was still to come. Was packet switching really a major breakthrough?

Strange as it may seem, it depends on how old you were: If your formative years had occurred prior to the mid-1960s, you are a pre-Boomer and your model of communication was defined by telephony. Packet switching was revolutionary! What an idea! Networks had been deployed that moved whole files at very low bandwidth, called *message switching*, but this idea of breaking data up into little packets was very different.

However, if you're just a little younger, and your model of the world is more determined by computers, then packet switching was not a big deal. If asked to send data between two computers, how would you do that? Well, the data is in buffers, you would pick up a buffer and send it. Packet switching is obvious. Anything else would be inefficient.

But even more radical changes were coming. For some, these ideas were very comfortable and not at all revolutionary. During the 1970s, these camps were often defined by age, partly due to the huge influx of boomers. In 1970, you could tell hardware people from software people by looking at them. In 1982, when I took the chair of the new OSI architecture group in the United States, I was more than 10 years younger than anyone else on the committee, and I was about the only person with first-hand experience with large networks. A packet-switch network required dedicated minicomputers to do the switching instead of hardware switching that was then the norm in telecom. Hence, the early networks were built by operating systems experts, with little input from telecom. The ARPANET's success also spawned efforts elsewhere and independently. Donald Davies at the National Physical Laboratory (NPL) in the UK had independently discovered packet switching and in France, it was adopted at IRIA under Louis Pouzin.

New Ideas from Europe

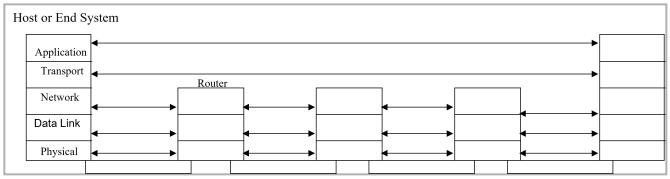
Starting soon after the ARPANET came online, Louis Pouzin visited the US to see the ARPANET. He was intrigued. Pouzin had been to the US earlier as a post-doc on Project MAC at MIT. That was the group that built Multics, where Pouzin had invented the shell. Louis looked at the ARPANET and resolved to go back to France to build a network. While ARPA built a production network to lower the cost of research at different sites, Pouzin was determined to build a network to do research on networks.

To do that, the CYCLADES group proceeded scientifically and very methodically. They needed to go back to the basics. Pouzin and his team essentially took a clean-slate approach to what they had learned from the ARPANET. Dave Walden, the leader of the ARPANET IMP development at BBN, made several trips to France to work with Pouzin and his team so that, as he told me, “they wouldn’t make the same mistakes we did.”

The CYCLADES team started out to understand what were the minimal assumptions needed for a network, e.g., what was the least information that had to be in a packet, how minimal could the switches be, etc. Once they had that then determine what was else needed to be useful. Davies at NPL had proposed a minimal packet, but didn’t follow up on it. Pouzin’s team adopted it and called it a datagram. They assumed that all packets are routed independently of each other. Why assume all packets had to follow the same fixed path? (Today, this is also referred to as connectionless.) Their network used a protocol not unlike HDLC at the Link Layer that was reliable. That meant that the only loss in the Network would be due to congestion and rare memory errors during relaying. In trying to determine how much reliability had to be in the network, they had a major insight: They realized that no matter how reliable the network was, the hosts would never trust the network to be reliable. Then why go to the extra effort to make the network perfect? The hosts would always check to ensure that the data arrived. As long as the network made a “best effort” to keep losses low, the hosts checking for losses would be sufficient and more efficient. For that an end-to-end error and flow control protocol would be needed for what they called the Transport Layer. There was no need for the network to be perfect; it could be cheaper, more resilient, and more flexible. Through a process like this, the team developed the minimal assumptions they would start with. Once this was complete, they turned their attention to what else would be needed to meet the requirements of the current applications. The very unexpected realization was: NOTHING. Their minimal assumptions met the requirements of all applications being considered in 1972! And the result was far simpler than anything else that was being done. An incredibly exciting result!

Pouzin and his team (one of the best teams I have ever worked with) built the network CYCLADES with the subnet of switches called *CIGALE*. CYCLADES is named after a group of Greek islands that in ancient times were loosely federated without central control and would come together for their own defense. CIGALE, of course, means grasshopper in French. (What was hopping? The packets, of course, node to node.) This was a truly radical idea, a non-deterministic decentralized reliable transmission from unreliable components. CYCLADES is where the fundamental concepts of modern networking originate. These kinds of probabilistic, non-deterministic ideas were really in the air at the time, with quantum devices, e.g., transistors, and their implications were still new.

The CYCLADES Architecture: CIGALE Subnet



And that wasn't all—there was much more. The CYCLADES layered architecture should be more familiar to what some have seen but probably have not really understood the systems implications of what CYCLADES had worked out. The layers as developed by CYCLADES were as follows:

First, there were the **Physical Layers**, which were the media, e.g., the wires.

Then there were the **Data Link Layers** to detect corrupted packets, and perhaps do flow control. Close to the media there needed to be enough error control to make sure packets were readable. For networks of this period, they were using leased lines at 9.6 kbps or in the ARPANET rates of 50 kbps, which were not terribly error free. The source of the errors was electrical noise from external equipment or the weather. Some error checking at that low level was necessary. The protocol (an early version of HDLC) used by both the ARPANET and CYCLADES was reliable point-to-point.

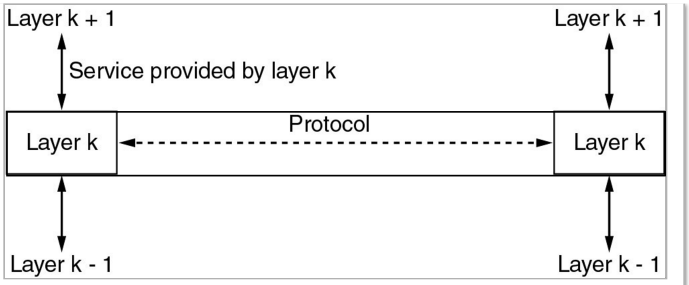
Then, there was a **Network Layer**, which did relaying and multiplexing of datagrams. Here the primary loss was due to congestion and rare memory errors incurred during relaying.

Then, there was the **Transport Layer** in the hosts, which did end-to-end reliability to recover from loss due to congestion. This formed the basis for the OSI Reference Model and the Internet as well.

> Notice that error recovery is done at two layers, but the layers have different scope and the errors had different sources. (Bursts of electrical noise at the Data Link layers; lost packets and single bit errors at the Network Layer.) It is more expensive to recover from errors incurred in a smaller scope than recovery in a greater scope. (The Transport Layer operates over several link layers of smaller scope.) One does not want to propagate errors to a wider scope. But there are diminishing returns if one attempts to make it perfect. At some point, it becomes expensive to get the last rare errors. The normal solution is commonly referred to as the 80/20 rule: It is relatively easy to get the 80% but may take 80% more effort to get the last 20%. (The numbers aren't as important as the idea.) Therefore, the purpose of the Data Link Layer is to provide sufficient error control, that error control at the transport layer is cost-effective. That means that while the Data Link Layer doesn't have to be perfect, the error-rate for any one Data Link Layer must be well below the error rate due to congestion. (With multiple Data Link Layers, the errors will be additive.) This is the kind of system dynamics that comes into understanding the rationale behind the design.

The Nature of Layers, Services, and Protocols

Relationship of Services to Protocols



In networks, layers are not just functions within a single box. A layer is a distributed black box of shared state that provides a service to the layer above, while hiding the internal working of the layer. There is a difference in mindset between the data comm world and the computing world. For example, in data comm, an interface is between boxes; in networking, an interface is between layers or modules. This, combined with connectionless, can lead to some radical new paradigms for communications.

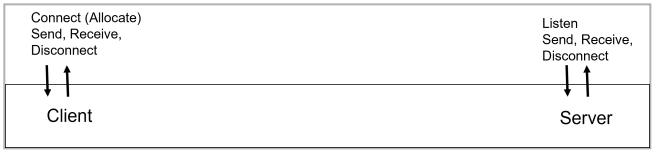
Services

Why the distinction?

Standards turf. When OSI group began to define an API for protocol standards, they were told that APIs were language-dependent and therefore belong to the various language standards committees. But, the definition of the inputs and outputs of the protocols were necessary, so an abstraction was created to get around the objection.

Service primitives or the API define what is visible from the outside layer, as opposed to what goes on inside the black box. It provides a service to the layer above. Most of the time, service primitives and API can be used interchangeably. But there is a difference. Service primitives are abstract, language-independent operations that are performed at the layer boundary. They are the inputs necessary to drive the internal workings of the layer. An API is a service definition at the implementation level, and it is the language-dependent library or set of system calls that are operations on the layer. An API may include calls that only have local significance, e.g., requesting the status of a connection, whereas these are not included in a service definition. Today, when there are fewer kinds of systems in use, APIs are more common than service definitions.

Service (Interface) Primitives



A Layer is a (distributed black box) that provide a service. It hides the complexity of the functions within the layer.

A process uses the service primitives to access the service to perform some action or report or an action taken by a peer entity. There are five core primitives that are commonly seen for the Internet:

Listen, wait for a connection

Connect or Allocate and accept an incoming connection

Receive, receive data

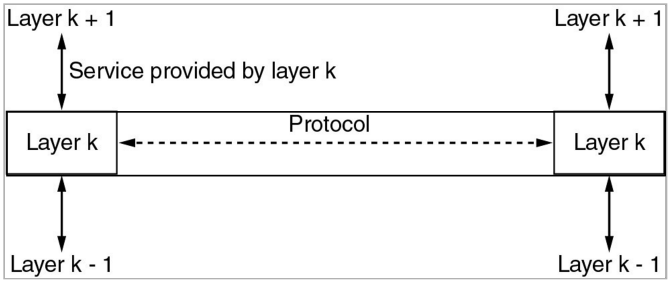
Send data

Close, Disconnect, or Deallocate, etc.

Listen is unique to the Internet, in the sense that, they had the idea that, an application could only be connected to if it were already executing. Therefore, for any process to be available to accept connections, it would have to be listening for a connection attempt. The requested application would send back a connection response and then data could be sent and received, and when all was done, disconnect. (It would also be straightforward for the connect request to fork the requested application and hand it to the connection-endpoint. Listen is not required.)

Protocols

The Important Thing About Protocols

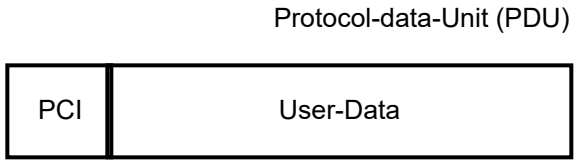


Protocols create and maintain the shared state of the layer. To do this, protocols exchange messages called *PDUs* (protocol data units), carry state information. When an API operation is performed at the layer boundary, the action must be communicated with its apposite. Because the protocol machines in the layer have no direct connection, the service of the layer below must be used to deliver PDUs to its peer. Asynchrony is inherent. The state of the protocol state machines in members of the layer may not be in the same state at any point in time. However, if left alone, the members of the layer will converge to a consistent state.

One aspect that is often ignored when talking about the definition of layers is that the emphasis is on the service the layer provides to the user above. It is equally important to understand that any given layer assumes that the service provided by the layer below must meet a minimal level of service.

As noted above, we will call the packets protocols generate: protocol data units (or PDUs). As you read more about networking, you will find that these messages are called just about every name you can think of: packets,

segments, frames, cells, even messages! You name it, it's been called that. It isn't rigid, but some terms are used with specific protocols or layers, e.g., frame is almost always used for data link layer protocols. They are all the same thing. It is much simpler when things are always the same to have one term for it. (We can claim it is a consequence of not being able to tell coffee cups from donuts. But really, it is just common sense.



What is the important thing about PDUs? They're divided into two parts: protocol-control-information (PCI) and user-data.

If data is sent from one system to another, we have to be able to tell the other protocol machine what to do with the PDU. This is the **protocol-control-information** or **PCI**. It is often called the *header.* The PCI contains the information that tells the receiving protocol machine what to do with this PDU, e.g., how to update the state of the protocol.

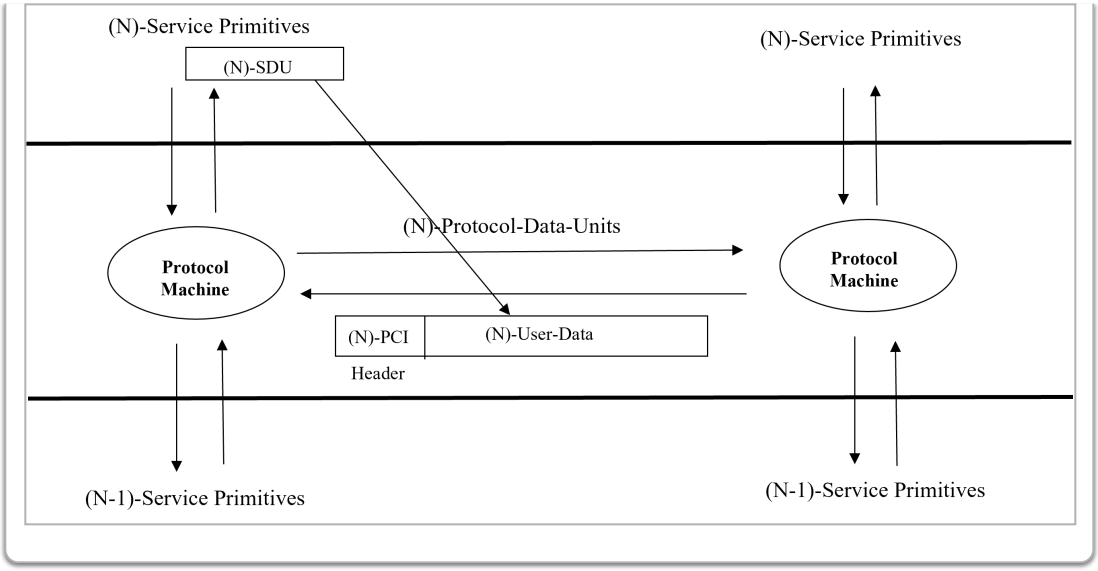
Information is what the protocol understands. *Data* is what it doesn't understand.

The protocol understands what is in the PCI but has no idea what this data stuff is. The protocol machine strips off the PCI and hands the data to the layer above, where this whole process is then repeated because the PCI for the layer above was in the data. (We will adopt a notation where this layer is the (N)-layer, the layer above is the (N+1)-layer, and the layer below is the (N-1)-layer.) This PCI is for the (N)-Layer. There may be an (N+1)-PCI for a higher layer in the user data, but we don't know that. All we know is this is data. At each layer, the protocol only processes its PCI.

Notice, this is a place where organism vs. observer is useful. As a network engineer, the observer may know (or worse, think they know) there is PCI in the user data. But it is important to look at the problem from the point of view of the organism (the protocol machine) to reason about what it knows. The engineer is analyzing a problem, trying to figure out what's going on. It is going to be very important to remember to have blinders on because all the protocol machine knows about is the message it has received with the information in the PCI. That is all it knows and has to act on. It doesn't know why the PCI was generated. It doesn't know where the PDU was generated. The protocol machine knows nothing other than it is in a certain state and it got this PDU. It can't draw any other conclusions.

How Layers Come About

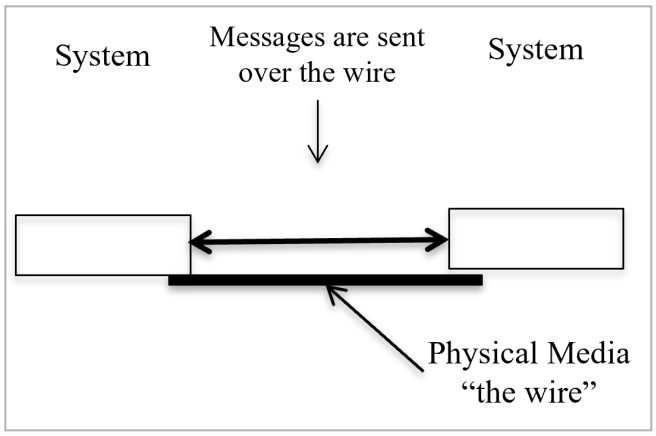
Protocol Operation



Thus, we see that the layer above passes a service data unit (or SDU) across the layer boundary. (We use this term to emphasize that the layer doesn't know if it is an (N+1)-PDU or just data.) The SDU is processed, and the protocol creates the necessary PCI and puts the SDU in the user-data field of the PDU. There is a de-coupling between the service and protocol. The layer above delivers SDUs, which are made into PDUs consisting of PCI and user data.

There is not necessarily a 1:1 relation between service primitives and the PDUs being generated.

How Do Layers Come About?



How do layers come about? Consider we have two systems connected by a wire. We want to send a PDU across the wire. But for the other side to know what to do with the PDU, we need to have a protocol to tell it what to do, provide the means to tell if it got trashed in transit, and keep the sender from sending it too fast.

How Layers Come About

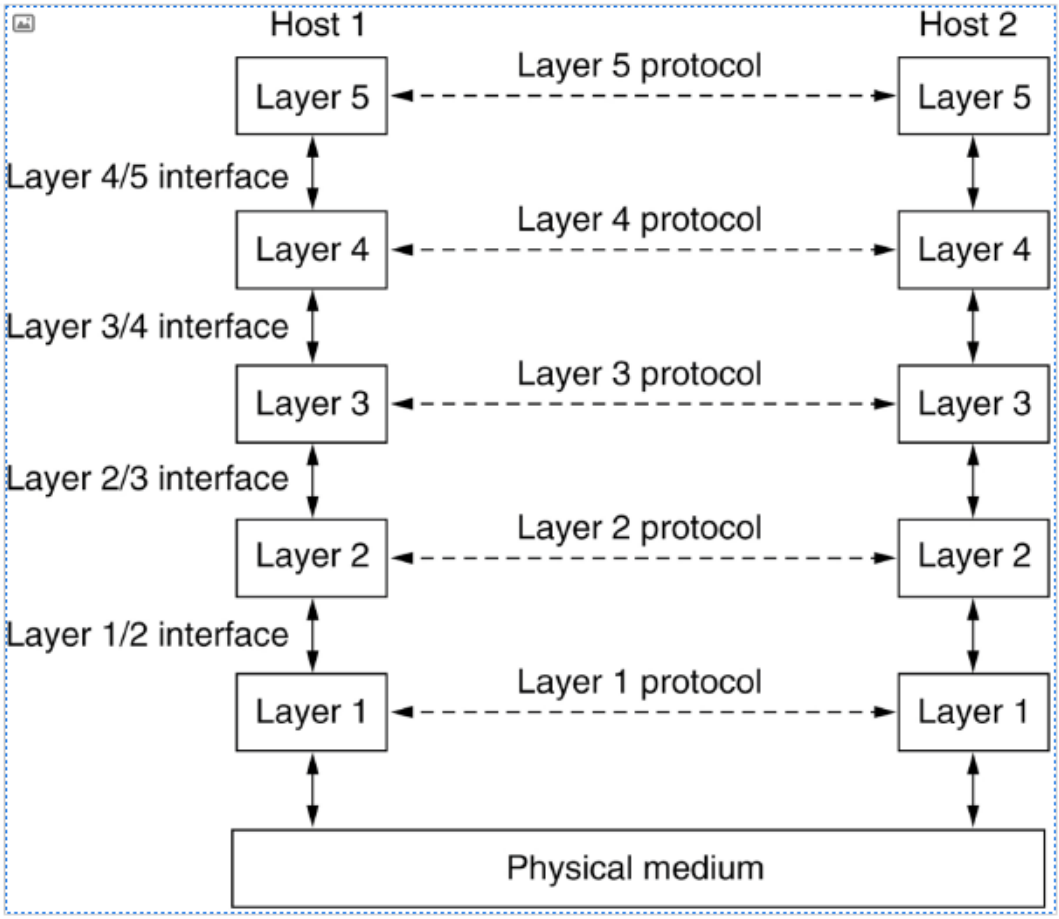
met_cs535_mod1_lec1_animation video cannot be displayed at the printable page. Go to the module page to watch.

There will be limitations on what we can do with that physical media. The wire has certain physical constraints, which were mentioned earlier, such as distance, data rate, PDU size, cost, and error rate. Any network will be made up of many of these point-to-point connections perhaps using different kinds of media. This would yield something like, the figure above.

How do we get between them? There are two ways: we can either do translation or relay. Translations can get messy and, worse, it is an N^2 problem. If there are N different kinds of media, and if any of the two media occur next to each other, there are N^2 translations to do. In this case, it would be better to do a relay.

A relay may have multiple wires, so it will be necessary to indicate where the PDU is going. This is how we end up with multiple layers because another layer will be needed to do something like this.

Layers



A five-layer network where the entities comprising the corresponding layers are on different machines are called *peers*. (Space does not permit important concept.)

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

One of the things to keep in mind is that while we have just been talking about how the protocol machine on one side exchanges PDUs with its peer on the other side, that is a lie. It doesn't do that at all. It can't, right? Here we see it. (This figure should really have a relay in between to be close to accurate and to illustrate layers of different scope. However, we will use this as it is to make other points.)

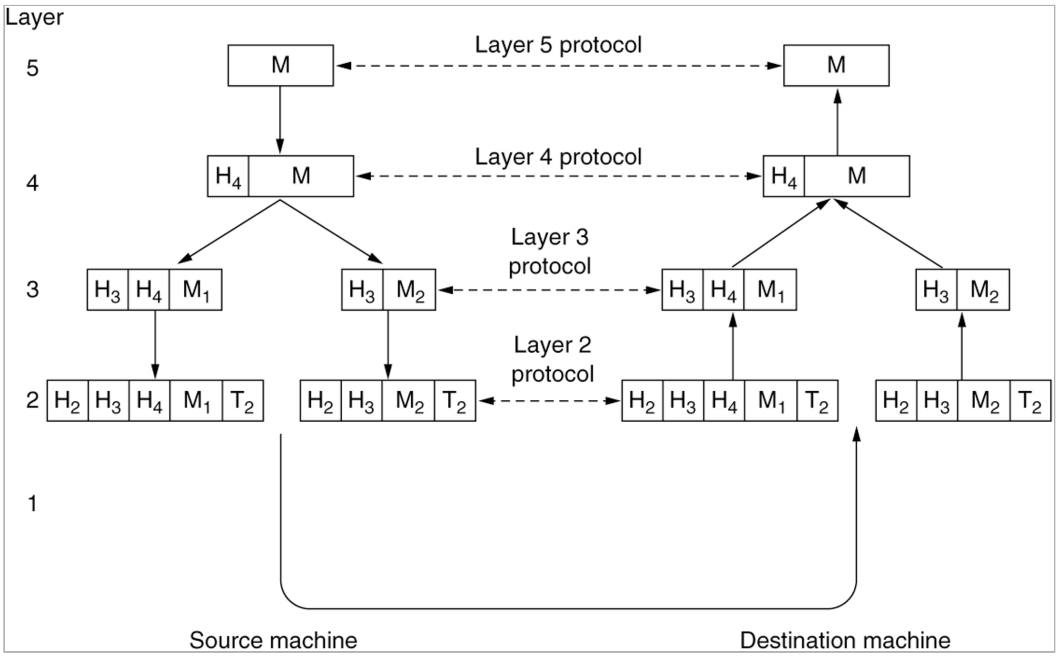
This layer 5 application is going to exchange messages with its apposite on the other machine, and layer 4 with its apposite layer 4, and the layer 3 with its opposite layer 3, and so on. But we know that's not possible! There's no wire between those protocol machines.

What really happens is: The application is going to formulate a message that is bound for its apposite. It can't send it there directly, but it believes that the layer below provides the service to deliver SDUs to layer 5. So, it passes the message down to the layer 4 protocol machine (PM), who puts (4)-PCI on it for layer 4. Layer 4 wants to send a PDU over to the other layer 4 to do its thing to it. It can't do it directly, but it believes that layer 3 will deliver SDUs to layer 4. So it has to encapsulate it and send it down to layer 3, which would encapsulate it in its (3)-PCI. And so it goes down to layer 2. Now layer 2 can finally hand the PDU to layer 1, which sends it across the wire.

As the PDU comes up to layer 1 on the other side, layer 1 strips off its 1-PCI, using the information from the PCI to update its state, and hands use-data up to layer 2. Layer 2 does the same thing, stripping off the 2-PCI to update its state and hands the user-data up to layer 3 and 4, and so on. Finally, the layer 5 PDU is delivered.

We will often talk about protocols exchanging PDUs back and forth, and every time we do, we are suppressing the lie. We all know that what's really happening is the PDU is working its way down the layers across the network and then back up. This makes use of the properties of the black box concept. Each layer is operating under the illusion that the layer below can do more than it can—that it can deliver its PDU to its peer.

Protocol Layering



An example of information flow supporting virtual communication in layer 5.

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

To show that a little better: our layer 5 protocol creates a message for its correspondent. To send the message, it hands it down to layer 4. Layer 4 puts its (4)-PCI around it and hands it down to layer 3. Layer 3 decides that this PDU is too long for the upper bound on its layer, so it fragments the PDU into two parts. Remember, all of this is now user data to layer 3. Layer 3 will pre-pend its (3)-PCI on both fragments. Note that these fragments are user data to Layer 3, but we know that in one PDU is the layer 4 PCI and part of the layer 4 user data, and the second PDU is another layer 3 (3)-PCI, and the rest of the (4)-PDU (user-data). The layer 3 PCI will contain information in both PDUs that will let the apposite layer 3 put these back together. But remember, layer 3 has no idea that part of this user data is a layer 4 PCI. Layer 3 then hands it down to layer 2, that adds its (2)-PCI to it. And layer 2 gets them to send it across the wire, and it comes up in the apposite system.

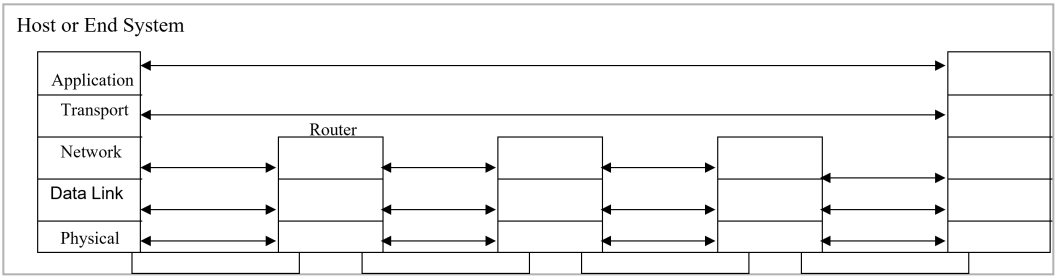
Next, layer 2 is going to update its state, do whatever this information in its PCI tells it to do, strip off the header for both of these PDUs, and hand them up layer 3. Layer 3 is going to look at this and process what's in the PCI for both. Remember, layer 3 can’t see anything in the user data at all, but it knows from the PCI that these two

PDUs have to be concatenated. So it strips off the layer 3-PCI, puts them back together, and hands one PDU up to layer 4, as if nothing happened. Layer 4 reads its PCI, does whatever updating is required, strips off its PCI, and hands it up to layer 5, which does its thing.

That's how all this works. Now, there are a lot of other things that go on in the PCI, and that's what we're going to talk about in detail in subsequent lectures.

Let's go back to CYCLADES. This is how the layers look in CYCLADES. It really mirrors what we just went through.

Back to CYCLADES: CIGALE Subnet



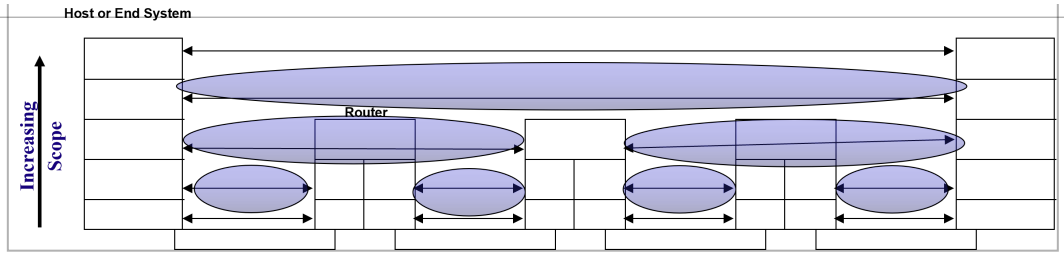
The layers as developed by CYCLADES are as follows:

- The Physical Layer is the wire.
- The Data Link Layer detects corrupted packets, which are discarded and may do flow control. It must keep the loss rate much less than the loss due to congestion by the network layer.
- The Network Layer does relaying and multiplexing of datagrams with the primary loss due to congestion and rare memory errors during relaying. With datagrams, every message is independent of every other. The network only made a "best effort" to deliver messages.
- The Transport Layer provided end-to-end reliability, primarily recovering from loss due to congestion.

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

There's a very important property of layers that is often overlooked, and that is the **scope of a layer**. This is probably the most important concept to understand. All of these layers don't have the same scope. In fact, this is something the Internet missed entirely. Notice that, in the figure, the Link Layer in the host on the left only goes a short distance. There are actually four layer-2s here, which could be all different technologies. There are two layer-3s and one layer-4.

The Scope of a Layer



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

People talk about protocol stacks or layers, because they're focused on the host, but that's very deceptive. If you look at a specific system, there may be a stack of protocols in the host. But the other end of those layers is not in the same system. There are peer-reviewed papers in the literature by major professors proposing that applications specify the Quality of Service for their connection by telling the Link Layer what to do. How is that one Link Layer supposed to affect how packets are handled in the other Link Layers? It can't. Thinking about layers in terms of modules, the system, and the stacks is very misleading. The scope of a layer is critically important.

What do we mean by *shared state*? For these protocols to talk to each other, they must have some amount of shared state. Depending on the protocol, the amount of shared state could be minimal. At the very least they must be able to understand the same PDUs. For others, there are feedback mechanisms that must coordinate with each other constantly. If they get out of sync, bad things can happen. The receiver is sending acknowledgments to coordinate with the sender. If one gets lost, then their shared state is no longer consistent, and deadlock could result. Keeping track of acknowledgements received would be an example of the shared state. The receiver knows what it has sent, and the sender knows what it has received. They may not be the same (organism vs. observer again). A lost acknowledgement could cause a temporary loss of synchronization.

Why History Is Important

The model described above was the view that network researchers had in the mid-70s. However, in the late 1970s, ARPA split off and, because it had much more funding than everyone else combined, eventually became dominant with different protocols. The Internet used the CYCLADES ideas of datagrams and end-to-end transport protocol, but they built it on top of the ARPANET. There was little or no significant contact between the core Internet group and CYCLADES after 1976. The Internet saw it with the traditional view of telecom as just a connectionless network with a reliable transport protocol. They missed that the scope of layers was important and that this was a whole new way of conceiving networks.

This was a major paradigm shift. It had four major characteristics:

1. It was a *peer network* of communicating equals, not a hierarchical network connecting a mainframe master with terminal slaves.

2. The approach required coordinating *distributed shared state at different scopes*, which were treated as black boxes. This led to the concept of layers being adopted from operating systems.
3. There was a shift from largely deterministic to *non-deterministic* approaches. This was not just with datagrams and networks, but it was also when we made the shift from polled operating systems to interrupt-driven systems. It was also in physical media with developments like Ethernet, which we'll talk about a little bit later and which is also a very non-deterministic approach to communications.
4. This was a computing model, a *distributed computing model*, not a telecom or data comm model. The network was the infrastructure of a computing facility.

Throughout this course, we will see the tension between the traditional beads-on-a-string model and the CYCLADES model.

The Role of Standards

For networking, standards are a must. If one buys equipment by multiple manufacturers with the intent that they must talk to each other, there must be some agreement on how they do that. There had been computing standards before for programming languages, magnetic tape formats, character sets, etc., but computers were still entities unto themselves. If manufacturers varied from the standards a little bit, it was not a big problem. The difficulty those differences made were minor compared to bigger problems that made portability or communication difficult. The fact that one machine rounded and another truncated; or that one had 24-bit mantissa and another 28 bits; or that instructions given to one machine had slightly different results than similar instructions on another machine were much greater barriers to porting software or to communicating. From the manufacturers point of view, this wasn't all bad. The difficulty also helped to keep customers locked in.

There was very little done in computing about communication standards. Standardizing the data link protocol, HDLC (based on IBM's SDLC), didn't start until the mid-70s and wasn't completed until the late 70s. Most everything was related to the phone companies. However, everyone could see that computing and telecommunications were going to merge. Furthermore, for networks, the standards were going to have to be very, very specific; otherwise, it wasn't going to work.

There were two major standards bodies, each representing the vested interest of a particular group.

The International Telecommunications Union (ITU)

First, there were the telephony standards. In most of the world, the telephone company was a ministry of the government, the Postal, Telegraphy, and Telephone (PTT), essentially combining the roles of manufacturer, provider and regulator. They built the equipment, installed it, ran the networks, and made the rules. Because the PTTs were ministries of the government, agreements among them were at the level of treaties. Therefore telephony standards, and by extension data network standards, were done in the International Telecommunications Union (ITU), a treaty organization of the United Nations. ITU is divided into three groups:

ITU-T, the telecommunications group (formerly called CCITT or Consultative Committee for International Telegraphy and Telephony) and the radio sector, ITU-R, (formerly CCIR = Consultative Committee for International Radio.); and ITU-D, the telecommunications development. The United States is unlike the rest of the world. In the United States, telephone companies were private companies regulated by the Federal Communications Commission. A special category was created so that American phone companies could attend ITU meetings, but for others to attend required credentials from the US Department of State.

The International Organization for Standardization (ISO)

The other major standards organization is ISO, the International Organization for Standardization. ISO has no connection to the United Nations. It is entirely voluntary whether an organization follows the standards. While ITU's scope is limited to telecommunications, ISO's is not. ISO standardizes almost everything one can imagine from screw threads to paper size, European paper sizes are all ISO standard; the symbols on the labels in your clothes; highway signs; nuclear reactors and much more are all ISO standards. One of the areas that the technical committees within ISO standardize is computing.

What is amusing about all of this is the terminology used for their output. ITU does not produce standards. If ITU did, being a treaty organization, it would mean that telephone companies would have to implement precisely what was in the standard. Therefore, ITU produces Recommendations. In contrast, ISO is not a treaty organization. Conforming to ISO standards is voluntary. In reality there is little or no difference between an ITU Recommendation and an ISO Standard. In fact, almost all OSI standards were published with a different cover page as ITU Recommendations.

IEEE

IEEE creates standards mainly on topics closely related to electrical engineering. The area we will be mostly concerned with will be IEEE 802 for local area network standards. IEEE is a professional organization. For a long time, IEEE was viewed as a U.S. professional organization, and in the 1960s until the early 80s, it probably was. IEEE would approve their standards, then submit them to the U.S. ISO committee, which would rubber stamp them and submit them into the appropriate ISO Study Committee (SC) that passed them as international standards. Several years ago, I pointed out that everyone knew that IEEE participation was not just American, but international. They should be recognized as an international standards organization as well. For some topics, IEEE no longer goes through ISO. Because some countries take the position that if there is no ITU or ISO standard for a topic, then they are free to create their own national standards. To prevent and ensure a global market for products that some IEEE standards are also approved through ISO. This occurs primarily for wireless and local area network standards. In IEEE, 802 began in 1980 with 802.1, 2, 3, 4, and 5 to develop the basic Ethernet standards. As one can see from the table, it has gotten a little out of hand since then.

Who Is Who in the International Standards

Number	Topic
802.1	Overview and architecture of LANs
802.2	Logical link control
802.3 *	Ethernet
802.4 †	Token bus (was briefly used in manufacturing plants)
802.5 †	Token ring (IBM's entry into the LAN world)
802.6 †	Dual queue dual bus (early metropolitan area network)
802.7 †	Technical advisory group on broadband technologies

▶ Check more international standards.

The Role of National Standards Organizations

Most countries have their own standards organization, which are the ISO member. In France it is AFNOR or the Association Française de Normalization; in England it is the British Standards Institute (BSI), and so on. It doesn't work that way in the United States. First of all, what used to be called the National Bureau of Standards (NBS), now called the National Institute for Standards and Technology (NIST), develops US government standards. U.S. standards are developed under the auspices of the U.S. member of ISO, the American National Standards Institute (ANSI). ANSI doesn't make standards either. Like ISO it oversees US standards participation on a wide variety of topics by accrediting industry groups to manage standards efforts for specific topics. The American Banking Association, for example, is accredited by ANSI to set standards for banking. INCITS (InterNational Committee for Information Technology Standards) is accredited to set standards for computing and communications, and so on.

Everybody has their own standards groups. Industry has come to understand, that standards are bottom-up and a standard only has force by adoption. There has been a proliferation of standards organizations around specific efforts. There now must be tens, if not hundreds, of them.

The Battle Is Shaping Up, But…

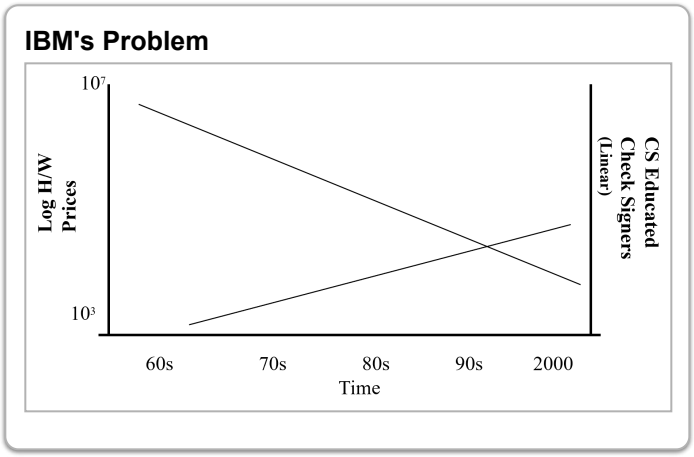
Some of the main players had major vulnerabilities that undermined their business strategy. As networking was increasing in importance, there were two big players: IBM and the phone companies, mainly AT&T in the US. The PTTs of the world were developing a beads-on-a-string, connection-oriented, deterministic standard called X.25, which looked more like the early ARPANET and not at all like the idea that the new networking efforts were pursuing. Researchers had discovered the limitations of the ITU approach. Similarly, the PTTs didn't like

the new approach and, most of all, they didn't want a Transport Layer. Researchers realized that they needed to influence the process, so they created the International Network Working Group (INWG), which was accredited under IFIP as Working Group 6.1. The researchers attempted to get datagrams in X.25. They succeeded in name, but the PTTs chose to ignore it. Given the organization of ITU, only phone companies had a voice and a vote. The PTTs had this locked up.

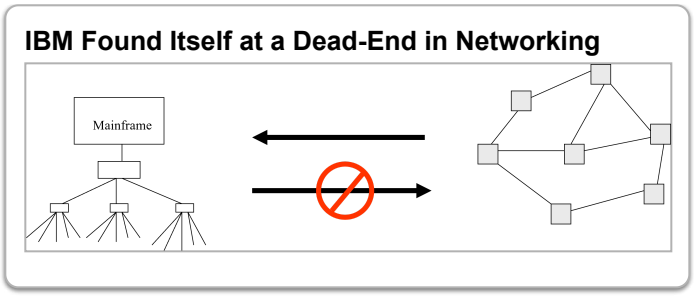
IBM

There were two other things going on here with these major players. At this time, IBM had 85% of the computer market. But they never had the best systems. In fact, their systems were incredibly painful to use. IBM hardware and software were never top of the line. They were never the best. But they always made it work, often at a much greater cost than originally promised. But once an organization committed, it was even more expensive to change. As the saying went at the time, “No one was ever fired for choosing IBM.”

A marketer once explained to me that IBM got where they were because *they always sold to the person who signed the check*. Very smart, good advice—always remember that. In the 1960s and 1970s, computers cost a lot of money, often millions of dollars. Obviously, the person who was high enough in a corporation to sign the check for that kind of money had no background in computer science. They had to trust the salesman and IBM played to that.



When I learned that, I knew that computing and memory prices were dropping fast and computing power was increasing fast. This made it obvious that as the field matured and people with computing experience would move up the corporate ladder, it would be harder and harder for IBM to make a sale. I didn’t propose they would go out of business, but they would fall on hard times. When I said that everyone said “Day! You are crazy! IBM has 85% of the market. Nobody is ever going to unseat them.” But the falling prices of computers accelerated my prediction and, by the 1980s, that’s exactly what happened. Not only were people moving up the ladder, but the price of computing was dropping like a rock. People could buy what they wanted, and by the mid to late 1980s, IBM was laying off tens of thousands of people. IBM does not have 85% of the computer market anymore.

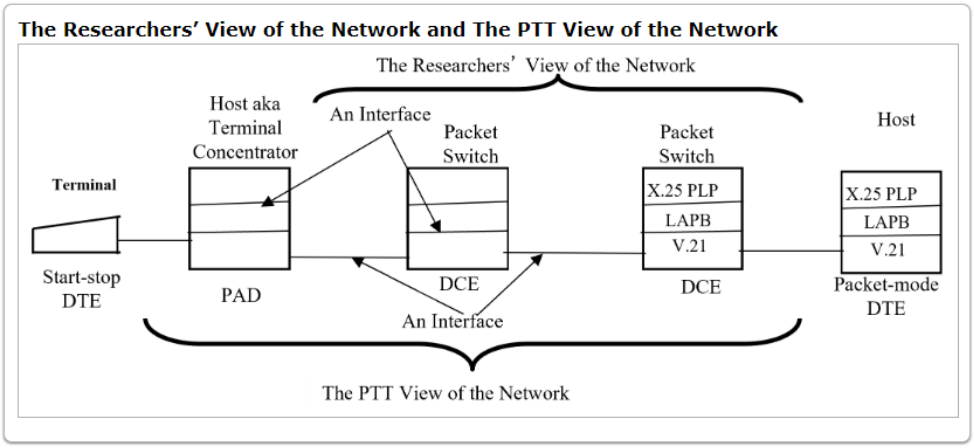


The other problem that IBM had was that its network architecture—Systems Network Architecture (SNA). SNA was centered on the relative minimal terminals talking to a mainframe. It was a hierarchical, deterministic datacom model. The first rule of architecture is, a peer architecture can always be subset to be hierarchical, but you can't go the other way. And IBM knew that. So here we were proposing a peer architecture for the future, and IBM was stuck with a large installed base and a product and architecture that can't go there. But they couldn't outright oppose it. It would be too obviously obstructionist and could run afoul of anti-trust laws. They were going to have to stonewall.

The PTTs

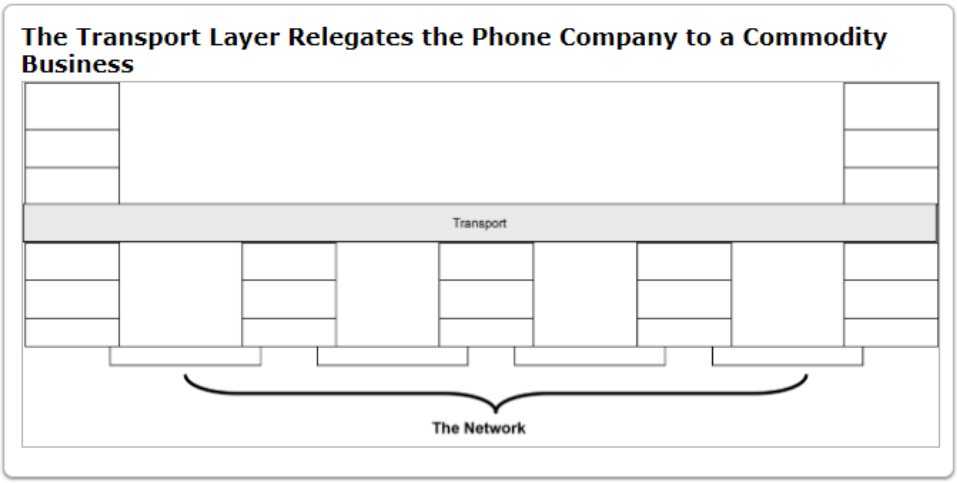
Meanwhile, the phone companies had their own problems. For data networks, they were trying to emulate the phone system, to do what they'd always done. Their attitude was, "Hey, we've been building networks for a hundred years. We know this better than you guys." But their approach was always asymmetrical, connection oriented, and deterministic. The other problem they had was they had such long deployment times. Because of their way of looking at things, you could say that the phone companies were targeting a window of opportunity that was 15 minutes in the future. Because the PTTs were a monopoly, much of the purpose of their architecture was to define who got to own what boxes. It was not only a technical blueprint but part of the business strategy. That translated to there being as little owned by the customer as possible. Keep in mind that, prior to 1970, "networks" within a company were limited to wiring terminals (usually in special terminal rooms) to computers, even if this was more than one building. There wasn't much else for the customer to own—perhaps a terminal concentrator or multiplexor, but that was about it. That was about to change.

In those early days, one of the things that was important was providing what was essentially a terminal concentrator to provide users access to the network. In fact, we built a small host like that for the ARPANET.[2] It would interface several terminals and other peripherals to the Internet but otherwise had limited functionality. In the computing model, we saw this as small hosts. (See the upper labels of the figure below).



In the PTT world, they said these were not small hosts. A host was a Packet-Mode Data Terminating Equipment (DTE). This small host was an interface between Start-Stop-Mode DTEs (terminals) and Packet-Mode DTEs. That wasn't a small host at all. It was a Packet Assembler Disassembler (PAD). (Complicated, eh?) Why did they do this? Because they wanted to draw the boundary between the PTT and the customer, so that the PAD was part of their network. Notice how differently the bottom part of the figure is labeled. I like this picture, even though it is a bit dated now, because here is the same picture with two different views of the same thing, but they have vastly different outcomes in terms of who gets to build what and how things are looked at. And most important of all, they are different in how viable they are for doing the next step. This was at the heart of the conflict between the computer and communications businesses. How so?

This was just the first step. The PTTs' ultimate goal was putting "value-added services," such as reservations, ecommerce, etc. in the network. If they could put them in the network, then they could claim that they were part of their monopoly, and no one else could offer competing services. If you have ever come across Minitel in France or Videotex, this was a prime example of what they were trying to do. They wanted to be able to own these database services and airline reservation services and what you now see on the Web. "In the network" was code for what we would call "in the cloud" today. Pouzin, the leader of CYCLADES, gave some memorable talks warning of this potential grab by the PTTs. With no Carterfone decision in Europe, there was even the specter of the PTTs claiming the only computers that could be connected to their network would be their computers. (Before Carterfone, every phone in every house or business in the Unites States was owned by the phone company. The Carterfone decision allowed customers to buy their own phones. The first step in deregulation.) However, being in a position to offer such services and actually doing it are very different things. The PTTs were never able to do it, but they are continuing to try with 5G. Pouzin's predictions turned out to be true. But it was just different companies that became dominant: Google, Amazon, Microsoft, Apple, Netflix, etc.

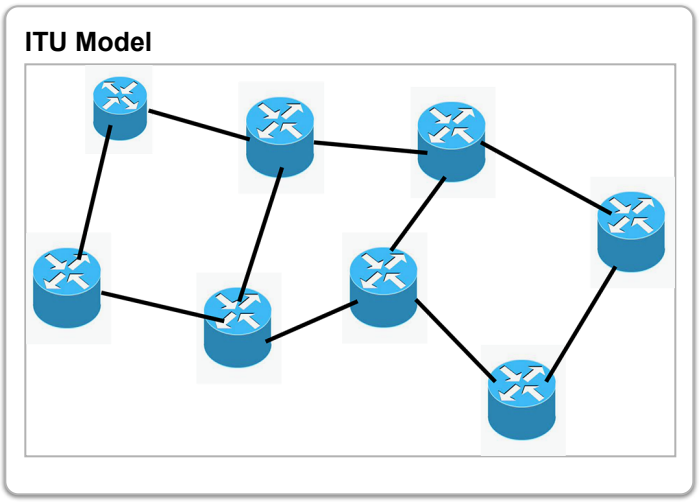


The new model of networking was a real problem for the PTTs. The proposed transport layer sealed off the network to just moving bits, a commodity business. No one wants to be in a commodity business! The margins (and profits) are low and it is hard work to get them. The PTTs didn't want to be there. They didn't like this at all. This means all services were at the edge. They were always in the hosts. But this meant that, if the PTTs were the host, then the phone companies couldn't claim that services were theirs exclusively. The services weren't part of their monopoly. The PTTs could still offer services, but they would have competition. They didn't want that at all. There was a huge war over this.

The Battle Breaks Out

There were two paradigms at war here: the ITU or beads-on-a-string model and the layered or CYCLADES model. The difference can be summed up as follows:

- In the ITU model, boxes and wires are dominant; layers are merely modules in boxes; and end devices are not part of the network. The ITU model is characterized by figures like this:

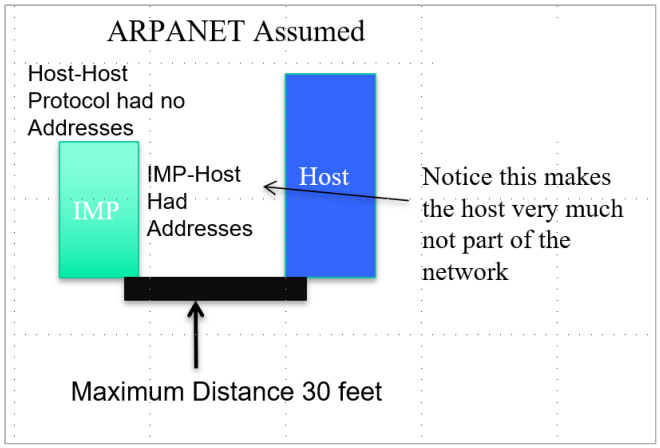


- In the CYCLADES model, layers and processes are dominant; boxes are merely containers; and end devices are part of the network.

Small Differences Can Make Major Differences

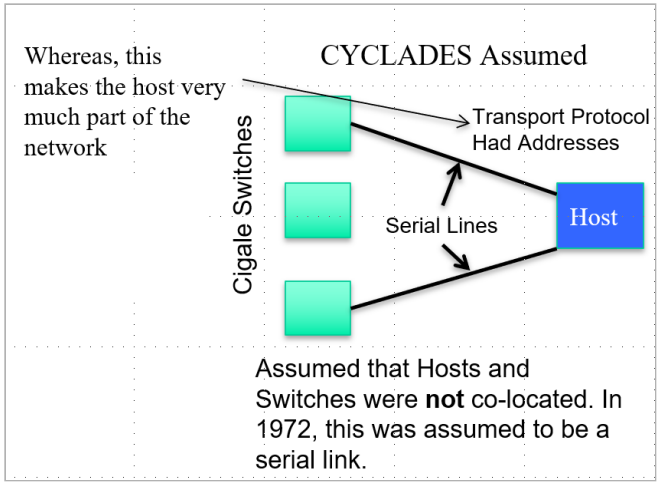
Previously we noted that in the ARPANET, it was assumed that the Host had to be within 10 meters of the IMP (the router), a limitation of the hardware interface. (Later enhanced hardware relinquished that restriction but these were the exception rather than the rule.) To open a connection to another site, a host sent commands to the IMP which created the connection. The Host-to-Host protocol (the NCP) operated on top of this. It carried no addresses. In this configuration, the host was not part of the network. The Host talks to the network.

ARPANET Assumed



CYCLADES made a very different assumption. It assumed that Hosts were not co-located with the CIGALE switches and would be some distance away and could be connected to more than one switch. (At the time, this would have been the norm. Network equipment would on the premise of the network provider, and hosts would be on the customer's premises.) At the time, this connection was assumed to be a serial line. The Transport Layer carried the address of the destination host. In CYCLADES, the Host is an integral part of the network.

CYCLADES Assumed



With CYCLADES, the “serial line” connecting a Host to a switch could be replaced by a network with no other changes. CYCLADES was an internet in 1972 (and operational in 1973), while the ARPANET was a network.

Who Cares About This?

The other computer companies care. They see an opportunity. Even though IBM has 85% of the market, there are close to a dozen other computer makers. They see that IBM and the PTTs are tied down with a huge installed base and the technology trends are against them. With falling computer prices, everyone sees computing and communications merging. And this trend plays to their strengths. They want a piece of the action. In 1978, they create the OSI (Open Systems Interconnection) effort and begin work on the seven-layer OSI model, followed by a protocols built to the model. While the initial model used the connection-oriented approach, the initial goal was to move it to a connectionless model. But this connectionless layered model had other implications that were not greeted with favor. There's an all-out war. The layered model had invalidated the business models of the two largest monopolies on the planet. This is going to get nasty.

IBM and the phone companies were still in the data comm model. The other computer companies were mostly in the distributed computing model. The IBM strategy was to delay as much as possible. They couldn’t really come out against it because of the potential for breaking anti-trust laws.

There is a common misconception on the Internet and among many academics that OSI was a PTT or telephone company effort. This is not true. OSI was initiated by the U.S. computer to counter the direction of the PTTs.

By 1982, the Europeans decided that inviting the PTTs to collaborate was better than having a competing effort. Without deregulation in Europe anywhere on the horizon, the Europeans really had no choice. This dooms OSI.[3]

This was oil and water: what the phone companies wanted and how they viewed the problem versus what the computer companies wanted and their view were totally different. There was no middle ground. This is what makes the OSI effort such a hard-fought political battle. I have been in meetings where people were yelling and screaming, pounding on the table, nearly coming to blows. I have seen people use every dirty trick in the book to try to get their view passed. This was really tense.

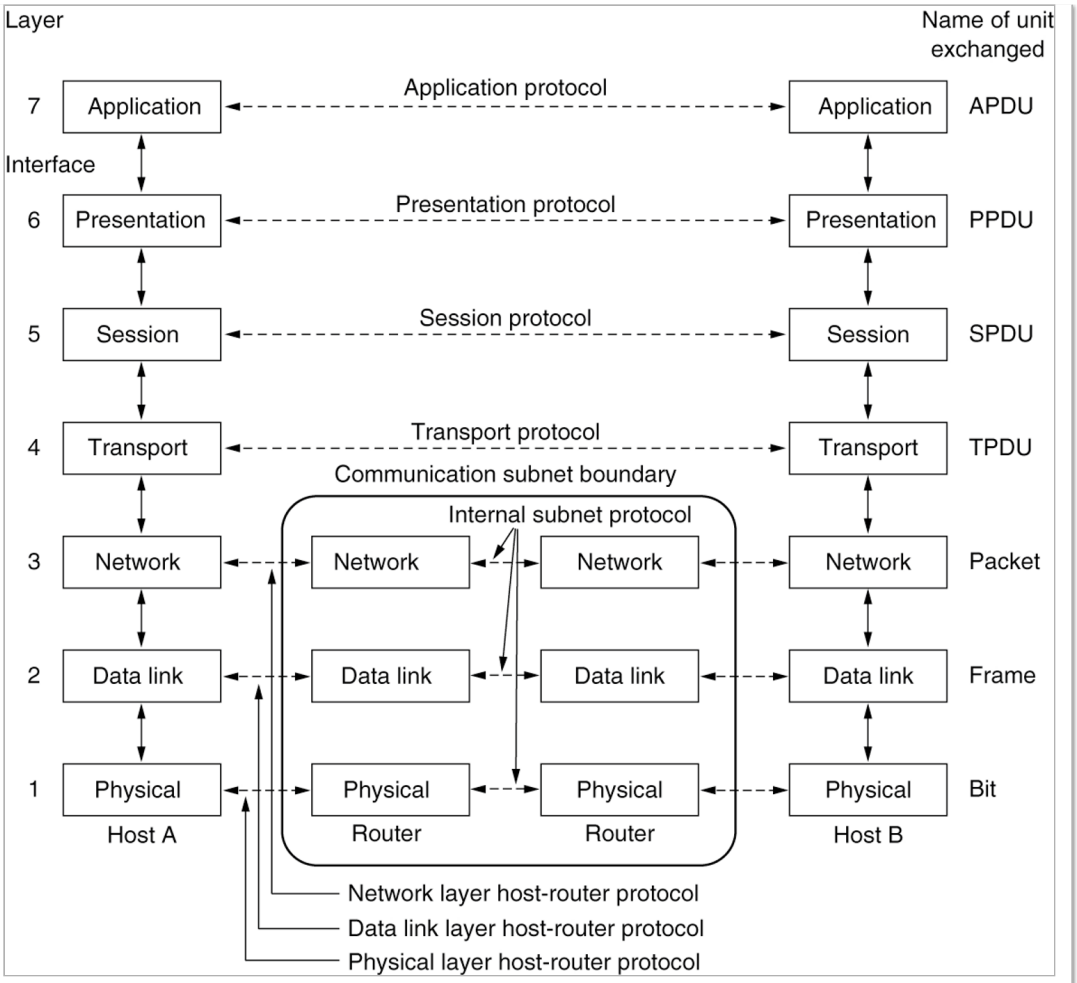
This begins a 30-year war that is playing out to this day. There has been no reconciliation of the connection versus connectionless problem, either technically or in terms of a business model. We are still seeing it. For a while, the phone companies lost out to what the Amazons and Googles were doing in the network, although they call it *in the cloud*, the PTTs dream of the 1970s. (Note too that this is nothing but the original IBM SNA model of simple terminals talking to a central host.) Currently, the carriers are still trying to use 5G to re-coup what they failed to achieve before, now disguised as edge computing. As we have just seen, there's no such thing. All computing is at the edge. When confronted with that, they respond that they mean “low latency,” “low response time,” or “short response time.” But that's an engineering problem, not an architecture problem. It is all about getting the two things close enough together, i.e., speed-of-light issues.

But even now the phone companies and a lot of other companies are going back to exactly this model of trying to put applications or services i*n the network or in the cloud*.

The battlefield for this was standards. It was a complicated war, of the US versus Europe versus Japan versus the phone companies and everyone against IBM. In the early 1980s, with no phone company deregulation in the US, let alone Europe, the European computer companies had to coexist with the PTTs. They assumed they would have to compromise with their position. The US was perceived as having a dominant market position primarily owing to IBM. However, IBM's network architecture was a dead-end. (The developments in the ARPANET and Internet were virtually unknown in the world of commercial networking in both the US and the rest of the world. This new peer layered model is a perfect foil to IBM, providing an opportunity to the other computer companies to outflank IBM. The deterministic mindset permeating both the computer and PTT organizations was a drag on change. It had religious overtones, e.g., "I believe this because I always have." The non-deterministic view was definitely from the younger generation. (In those early days, it appeared this was an age characteristic. But with more data, it is clear that there are many people who are simply uncomfortable with non-determinism.) Another significant factor was the lack of senior management. Software development was so new that the leaders of these groups were in their 20s and early 30s. Few if any senior management had any software experience.

The OSI Model

At the first meeting of OSI in March 1978, they adopted the seven-layer model. This turns out to have been a swift piece of political manipulation. The model was proposed by Charles Bachman of Honeywell Bull. Charlie had invented the entity-relational database, which you may have encountered and been realized the Turing Award, the highest honor in computing. I worked for Charlie for a while and collaborated with him for decades. We were good friends. We only lost him recently.



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

The core of the OSI model was the CYCLADES model. One of the people working with CYCLADES was from Honeywell Bull in France (Honeywell had merged with the French computer company, Bull). Unlike the earlier pictures, which just showed the layers in the host, this had the communications subnet in the middle of it, so that it unknowingly) called out the differences in scope. The CYCLADES model became the core of the Honeywell Bull Distributed Systems Architecture, to which Charlie added the upper three layers modeled after IBM's SNA. In the first meeting, it was more likely they would choose a U.S. proposal, especially with Charlie as the chairman of the committee and Zimmermann from CYCLADES as the chair of the architecture working group. It was a very deft way that the French got their approach accepted and then were put in a position to guide it. This was far more subtle than the attempts by the PTTs to get their positions adopted, which were more, "we are the PTTs, do what we say."

ITU developed one connection-oriented technology after another: ISDN, Frame Relay, ATM, etc. The Internet in general, maintained their worldview of being a unique embattled elite. However, they had only grafted datagrams and end-to-end transport onto the traditional ITU model. Meanwhile, the influx of people from outside the initial DARPA contractors was pushing the Internet deeper and deeper into the ITU model. At the same time, the computer companies were competing with each other to set the direction of the OSI standards and market. IEEE 802 was formed in 1980 by DEC, Intel, and Xerox, which was a good counterbalance to the phone

companies and IBM. (We will have more to say about this when we cover LANs later in the course.) Interestingly enough, AT&T saw this as an opportunity.

AT&T (or more precisely Bell Labs) AT&T wanted to get into computers. Up to this time, because of AT&T's monopoly situation in the phone industry, previous anti-trust agreements prevented them from being in the computer business. AT&T was sued again for anti-trust and this time saw it as an opportunity. They allowed themselves to be broken up. (This is the start of deregulation in the United States.) The Regional Bell Operating Companies (RBOCs), were divided up into seven companies called the "Baby Bells." AT&T retained the Long Lines division that connected the Baby Bells. But AT&T was now freed up to buy a computer company, which they quickly did by buying National Cash Register (NCR), an odd fit, and proceeded to make a mess of it. Running a computer company is not like running a monopoly phone company.

There was also a delusion or two built into all of this. The first was the idea that each regional BOC was a viable business unto itself. Within a few years, through acquisitions and mergers, the seven Baby Bells were down to two or three, but not before millions were spent getting there. Competitive phone companies became possible. They could own or lease their own long lines and then put equipment in the Baby Bell central offices to handle the "last mile." The Baby Bells dragged their feet in allowing the competitive companies to put equipment in their central offices. They were able to wait them out. Furthermore, the margins were smaller than expected. It was no longer a viable business. (That should have been obvious from the beginning.)

---

2. BBN produced a version of the IMP, called a *TIP* (Terminal Interface Processor), that was even more minimal, but the TIP software appeared to the rest of the IMP as a host. IOW, even though it was one box, looked like two to the network and was consistent with the figure.
3. Contrary to everything else you may have read about the battle between the PTTs and computer companies, the fact that DARPA was spending orders of magnitude more money on the Internet than all other companies combined (and even then, it was a round-off error in the DoD budget), and that computer companies were targeting the current market of corporate networks, The market for an Internet did not exist yet. It was coming but was still years away, when the advent of the Web made that much more immediate. The reason OSI failed was more political than technical, in fact, it had nothing to do with technical solutions.

There Were Issues

By the late 1970s, it was clear that the new direction had some issues as well:

> First of all, the Host-to-Host protocol for the ARPANET wasn't going to scale. It needed to be replaced. The datagram model indicated a new direction, but the Internet had its own proposal.
>
> Had we gotten the basic structure correct? This was all new. We hadn't seen that much of the problem space. This was still in the late 1970s when networks were operating at

fairly low data rates. We knew we had had to make a few short-term kludges that needed to be rectified. Were these the right layers?

What was the upper-layer architecture? We had a few simple applications, but what was above transport? Was there a general or common structure there?

Connectionless looked very promising, but we had only used it in the small. Was this going to scale?

We needed application names and a directory. But what did that look like? We needed a solution to numerous addressing problems that had turned up. But with an operating system background, the solutions were clear. It was just a question of doing it. We needed to take a hard look.

We needed better security. It was essentially non-existent.

That is quite a list.

Given this list and removing itself from the political fray, the Internet was in a position to make good progress on the open questions and produce a model network that didn't have to compromise with the warring political factions that OSI had to contend with. But the Internet had lost the vision of where to go and was more content in showing that researchers can do operations. The noise from the battles that OSI was in carried over to the Internet. They began to see themselves as an isolated embattled elite, who already had the network everyone was working toward and had had it since 1970. They were the experts. (It is not yet clear that they had lost the Internet layer, and that at six or eight major decision points, with the right answer and wrong answer known, they had consistently made the wrong decision.) But it worked! Moore's Law is masking a lot.

What Happened?

Users Interest Group (USING) started in mid 1973 with many good ideas, but as noted earlier, it was shut down very quickly in less than a year. For the next 20 years, there were no new applications on the Internet. This would prove to be a critical juncture. The emphasis on resource sharing was over. So was the driver to finish the architecture.

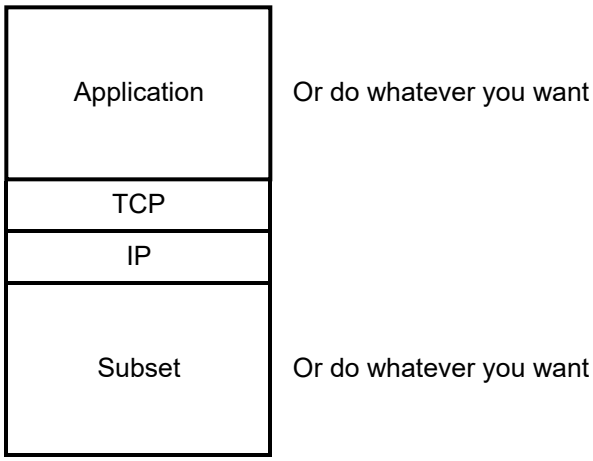
Newcomers, especially those not from the US, saw the Internet as patterned after the phone system, when, in fact, the whole DARPA effort saw the phone company as an anathema.

In 1974, INWG developed an international transport protocol. In 1976, the outcome was a synthesis of TCP and the CYCLADES TS proposals, known as INWG 96 (its document number). With the National Physical Laboratory (UK), CYCLADES, and the European Informatics Network (EIN) immediately announcing that they would adopt INWG96, DARPA was unwilling to compromise with the majority saying they were too far into developing TCP to change and went off on its own. (Even though TCP went through five more revisions before

it was deployed in 1983, even years later.) INWG96, improved by new results, becomes OSI Transport Class 4. This would prove to be another critical juncture.

With the loss of institutional memory, the Internet made a string of errors. But Moore's Law allowed the problems to be papered over. With major efforts going on elsewhere, they turned inward, protecting what they had.

We will come back to this later, but nothing new was going to happen in the Internet. There was no agreement about or interest in what was below the network layer or above the transport layer. Consequently, there was a tendency to try to see everything as an issue in one of these two layers.



The Internet ended up with a standards organization consisting of the Internet Activities Board, renamed the Internet Architecture Board; the Internet Research Task Force (IRTF) for long-term research problems; and the Internet Engineering Task Force (IETF), which actually sets the standards within the Internet. The Internet Society (ISOC) essentially creates an umbrella for all of those organizations. The World Wide Web Consortium (W3C) is a separate entity unto itself. Standards define what needs to be interoperable with other organizations to promote specific standards, such as WIFI Alliance, the Open Networking Foundation, etc.

So let's pause here.

A series of problems arose over the next decade or so.

Congestion collapse, exponential growth of router tables, quality of service, the Web, etc. The Internet boom was on, confirming that everything that had been done was right! But does economic success imply solid engineering?

Things were looking good if you didn’t look too closely. Moore’s Law (throw hardware at the problem) was keeping the bad problems at bay. But many knew that all was not well. It had been a very fast 20 years. Something was wrong and all the signs were there, but it was working. So why worry about it?

What went wrong?

ARPA killing USING removed the only driver for change. With Moore’s Law, growth alone was not enough to drive change. Research was pushed onto the world stage too soon. (At the same time during the 1970s and

1980s, the cost of building meaningful networks for experimentation was prohibitive without something like DARPA funding.[4])

With the battle over connectionless, a bunker mentality formed, where neither side could risk compromise, neither side could see where the compromises were going, and both sides were afraid that the other side was going to use any compromise to eradicate their position. Everyone dug in their heels and wouldn't budge.

By the late '70s, a second-generation effect was in place. Newcomers saw the Internet as if it already existed and everything was done right. (I have had people tell me what a brilliant idea well-known ports were! We knew they were a kludge when we did them!)

Breaking with INWG the Internet was continually being populated by newcomers in the ITU model. The problems the ITU model would expose would not become critical until much later. Groupthink set in, which was very unfortunate.

Textbooks and courses are essentially vocational, merely reporting what is out there, not why. This essentially condemns new engineers to repeating the mistakes of the past.

The success of venture capitalists led to emulation by research funding. Consequently, they look for short-term turnarounds when they should be looking at long-term, high-risk research proposals.

And that gives a good introduction to see where this goes.

---

4. In 1969–70, 50 kbps lines were outrageously expensive and very fast for the computers of the day. Computer communications by private corporations over long distances were commonly 9.6 kbps well into the 1980s. I have conjectured that, had ARPA built the ARPANET with 9.6 kbps, it would have worked, but it would have been viewed more as a curiosity and not really practical. Those “fast” lines opened people’s eyes to what was possible. (It turns out that ARPA was planning to use 9.6, but Roger Scantlebury of NPL recommended they go for higher speed because NPL had found 9.6 too slow. Of course, the NPL network was entirely within the NPL campus, and not covering thousands of miles across the United States. Luckily, the DoD could afford it.)