MITOCW | 23. A brief history of the Internet

PROFESSOR: So this is the last but one lecture of the term. So what I'm going to do today is in about 45 minutes, give you a quick history of the internet from-- we'll start in the late 1950s and then get to today. And then next time on Monday, I'll conclude this class by first doing a wrap-up of 6.02, and then also telling you a little bit about where I think the future of communications systems might be going I'll probably be wrong about it, but I'll be confident about it.

And today, the idea is to try to connect some of the topics we've studied in the class so far to this history. Of course, we're not going to be able to do all of it. So the story so far in terms of the history, you have to assume-- so we're going to start in 1959, or 1957. And by this time, the history of communication systems has had a lot of successes.

And a common theme is that there's the technology that comes around, it succeeds, it tries to take over the world. And right at the time and it looks like there's nothing else that's going to happen, some other new technology comes around that over time kills it. So the first successful network technology was the electric telegraph.

And the electric telegraph was done by a number of different people-- Wheatstone and Cooke built an electric telegraph in England, Morse and Vail built one in the US. The Morse code was developed as part of the electric telegraph. And that did a great job in the 1830s, 1840s, 1850s, and so forth. And then other technologies came around.

Now, this sort of story keeps repeating. And so by the 1950s, the dominant player in the communication area is the telephone network. So we have the Bell Telephone Company, and in the US, it dominates.

There's equivalent telephone companies in other countries. And you have this massive, amazing telephone network. And increasingly, many people have telephones. On the wireless side, there isn't a wireless telephone system in the 1950s. But what we have are radio-- broadcast radio and broadcast television, and some very powerful companies that own wireless spectrum to offer television and radio.

So the story starts in the late 1950s. And in 1957, a big thing happened-- Sputnik launched. And that caused the US to assume that they were falling behind in science and technology. And that led to the creation of ARPA, the Advanced Research Projects Agency that still exists. It's known as DARPA today. And it is probably the biggest federal funder of fundamental research and a lot of applied research and development.

Paul Baran in many ways is one of the fathers, or the father of packet switch technologies. He was working at a think tank called the Rand Corporation, which was an organization that really was a think tank. It allowed people to think about long-term fundamental directions in terms of technology and where technology was heading. And he was looking at the problem of trying to build what he called a survivable communication network.

And the story is that he's trying to build a network that can continue to work in the face of a nuclear war. But that's really not what he was trying to go after. What he was trying to build was just understanding how do you design communication networks that allow you to handle failures? And a lot of the then telephone network was a very centralized kind of structure where you would have a network that had very little redundancy built into it.

And you'd have lots of these central star-like topologies. And the problem, of course, when you connect these stars together, and these star-like pieces connect to other star-like pieces, is that you had better make these points here, these nodes or switches here, extremely, extremely reliable, and those links that are connecting them to these other central structures. Because otherwise, the failure of those things would kill the system. And the Bell Telephone Company actually understood how to build these very expensive, very, very reliable switches, but they were very, very, extremely expensive.

The other problem is, and we'll get back to this later today, is that a telephone network is a great network. But it supports exactly one application. The application is you pick up the phone and you talk.

It's very hard, on the face of it, to imagine how a telephone network would do a great job at supporting the web, for example. No one even thought of the web. But even something basic like being able to watch a video stream whose quality might vary with time-- these are things that the internet did differently and did better than the telephone network. But the telephone network fundamentally had, in those days, a fault tolerance issue. And the way they dealt with it was to build extremely reliable components.

Paul Baran's idea, which other people had been thinking about and toying with, was he observed that-- and in those days, the telephone network was largely analog. The digital telephony wasn't really there. And digital computers were just starting to come in.

And Paul Baran was probably one of the first few people who realized that computing and digital technologies can change life in terms of how you build these systems. Because the digital abstraction allows you to know for sure whether a component is working or not. Because if it works, it gives you an answer that then you can verify. And if it doesn't work, it just stops working.

It isn't like an analog system where it may or may not be working, and as the noise increases or some fault occurs, you're starting to see a lot of noise. And it's garbled, you're not quite sure. With a digital system, it either works or it doesn't work. You can build systems like that.

And he noticed that you could now start to think of building reliable systems out of lots of unreliable individual components. And this is the fundamental guiding theme for large-scale computing systems from the late 1950s. I mean, this goes all the way to how Google, or Amazon, or Facebook, or any of these big data centers work.

Any single one of those computers there is highly unreliable relative to what you could do if you put in a lot of money to build a very reliable computer. But the ensemble is highly reliable. And to do that requires a lot of cleverness and care. And the first real example of this is a digital communication network built out of this idea of packets. In a couple of papers that he wrote in the late 1950s and early 1960s, he said that with the digital computer, you could now start to build highly reliable communication networks out of unreliable components.

And so he articulated this idea that you can connect these switches or nodes together in highly redundant structures. And if you have a stream of data to send, you don't have to really pick a particular path through the structure. You could take that message and break it up into different pieces and ship them in different directions. And even if you chose to ship them all in one direction, if a failure occurred, these switches could themselves start moving the data in different directions, which meant that you no longer think of a communication as a big stream that you have to send in one way, but you can start thinking about splitting it up into these different pieces.

Now, like with many of these ideas, it's rare to find-- sometimes it happens, but it's rare to find exactly one person in the world thinking about it. No matter how groundbreaking the idea, there were other people working on it. And Donald Davies in the UK in the early '60s was looking at similar ideas. And he actually coined the term packet. We use packets now to mean these little messages that you ship through the network that are atomic units of delivery. This term was coined by Donald Davies in the 1960s.

Now, all of this was wonderful and sort of theoretical abstractions. But how do you start to come up with some design principles for building communication networks? In particular, how do you deal with the problem of having these links come together at a switch, and try to share the links going out of a switch in a way that allows traffic from different conversations to multiplex on the same link?

The idea of having a queue in a switch in retrospect seems completely obvious. But if you're the first time you're seeing something like this, and the telephone network had really no queues, the idea that you would build a queue and now start to analyze is a pretty groundbreaking result. Again, there were many people involved. But probably the leading contributions came from a person called Len Kleinrock, who was a PhD student at MIT. And in his 1961 PhD thesis, "Information Flow in Large Communication Nets," wrote about how you could use queuing theory to analyze and to model communication networks.

Now, at around the same time, again at MIT, Licklider and Clark wrote a really interesting paper. It's actually worth reading now. I mean, it's 50 years ago, but it's interesting.

You have to go back and think, this was at a time when people didn't really have this idea that people could sit in front of computers. Computers were used to maybe count votes. I suspect they get it wrong today more than they did in those days. But computers were used to count votes, they were used to help with the US Census.

But nobody thought about people sitting in front of computers. And they wrote this wonderful paper called "On-line Man Computer Communication." I guess in those days, you know, man meant people. So anyway, they wrote this paper.

And in fact, Licklider had this vision of what he called a galactic network that would span the globe and beyond, which was-- for the early '60s, it was a pretty remarkable vision. Now, using these ideas, and particularly Len Kleinrock's ideas, and this idea of man-computer interactions-- which of course, the idea of everybody having their own computer wasn't this paper's vision. This paper's vision was there's a lot of computers out there. And people just had remote terminals, and you would log in and have these big computers that you could use.

But then you would have nice interactions on your own terminal. That was what that paper was about. Larry Roberts was first at MIT, and then moved to ARPA to run this program, created something called the ARPANET, and wrote a paper and wrote a program that is a call for proposals for the ARPANET, which was a plan for timesharing remote computers.

So the internet-- the ARPANET was the precursor to the internet. And it started not because we wanted to build a communication network to prevent-- for it to work when there was nuclear war or any of these major disasters. It actually had a very concrete goal-- just allow people-- computers were really, really expensive-- just allow people, no matter where they were, to be able to harness the power of expensive computing far away, and make it look to the extent possible as if the computers were with you. That was the vision-- pretty compelling, but simple.

And they decided, for very good reason, to pick packet switching. The reasons primarily had to do with economics. This was a network that was being proposed for an application whose utility was questionable. And the idea of investing huge amounts of money was not quite palatable.

And Larry Roberts and others were very taken by this vision of packet switching. So they said you know what, the ARPANET is going to be a packet switch network. I'll come back to this later, but of course, the telephone companies like AT&T just thought this was a terrible idea, and were using every opportunity to ridicule the idea.

The ARPANET was created, and a few teams bid on the contract for it. And BBN-- Bolt, Beranek, and Newman, that's near Alewife in Cambridge. They're still there, they're part of Raytheon now. And they still continue to do pretty interesting research. They won the contract to build this network, build the technology for this network.

And because the processing involved in the network, the protocols that were involved were considered complicated, and considered to be computationally intensive, they had to build a separate piece of hardware that they named the IMP, or the Interface Message Processor. And BBN won the contract to do that.

The idea of an interface message processor is that every computer, as well as every switch, that the switch would have some hardware to forward stuff. But you needed something to do the computation of both the routing tables as well as actually every packet. Every packet would show up, and you would have to compute some sort of a checksum on it. And you'd have to do this computational task of figuring out how to forward that packet.

And that was just considered too much work to have on an actual little computer. So they actually had to build a separate piece of hardware that you would attach to your computer, and it would probably be about as big. And you can see the picture here. These IMPs were attached to bigger computers or computers of the same size. And these did the networking.

I mean, today, all of that stuff, a million times more is going on this device. But back in the day, that's how it was. So they won this interface message processor contract called the IMP.

And when you win a big federal contract, oftentimes, your Congressman or Senator writes to you. In fact, it's funny because a bunch of us got-- for a period of time before the Senate election, a bunch of us were getting emails, letters from Scott Brown congratulating us on winning some dinky little NSF proposal. I don't know if other people here got it, other faculty here got it. But it's sort of like in those days, if you won a big contract, you'd get money-- you'd get a letter from your congressman.

So in fact, Ted Kennedy, who was the senator at the time, and was for many years, congratulated the team for winning this. Except he got it wrong-- he congratulated them on winning the contract to build the interfaith message processor. I assume that if they actually had built that, it might have been a more useful contribution to world peace. But all they managed to get was the contract to build the interface message processor-- just details.

Anyway, so this team was a pretty remarkable team. They built the first-- they didn't build the first email program. That was done over at MIT in the '60s.

But what they did do was the first email program that crossed different organizations. And in fact, the @ symbol in your email addresses, which of course, is sort of the right symbol to use, if you use the @ symbol. But there was a person at BBN, Ray Tomlinson, who said, I'm going to put the @ symbol in email addresses. And a lot of early stuff happened that continues to this day.

So they built this network. And Kleinrock over at UCLA was the principal investigator on this-- now building systems out of this piece of hardware that was built. And this was the picture of the ARPANET in 1969.

This became the internet. There's a continuous evolution from this four-node picture to the internet today. And in 1969, they finally connected initially two, and then four nodes. And they had to do the first demonstration.

And to listen to Len Kleinrock tell the story, this was his story. He says that his group at UCLA tried to log into a computer at SRI, which is in Palo Alto. And he said, we set up a telephone connection between us and the guys at SRI.

We typed the L, and we asked on the phone-- because they had to check whether it was working, so they had the phone to check. We asked on the phone, do you see the L? Yes, we see the L, came the response.

We typed the O, and we asked, do you see the O? Yes, we see the O. And we typed the G, and the system crashed.

But you know, they got something working. And of course, there's a nice statement here. You know, a lot of people worry about performance optimizations. But the most important optimization in a system is going from not working to working. And the fact that something worked is extremely important.

Very soon after, they connected the East Coast-- a bunch of computers and organizations on the East Coast to the West Coast, MIT among them. So there was a team over at BBN, and a lot of them were from MIT. So MIT, BBN, Harvard, and Lincoln Labs on this side, and MITRE got connected over at Carnegie-- today, it's Carnegie Mellon University, I think it was called Carnegie Tech at the time-- University of Illinois, and then long lines across the country. There was a group of Utah and in California.

Now, what were these links? Anyone want to guess-- these links across the country, or between Harvard and MIT, or across over there, do you think they actually went and put in new cables and laid these wires? What do you think they were?

They were phone lines. And so this idea shows up over and over again. The ARPANET was essentially an overlay built on top of the telephone network.

And in fact, it was a hostile overlay. Because the telephone network didn't really like-- I mean, at the time, they thought this was just an academic joke. But over time, it became clear that this underlying network was being used on top to do something different.

And so there is an overlay network that's built-- and overlays show up again and again and again. It's just that they're not as hostile these days. Another example of an overlay is BitTorrent, or any peer-to-peer applications-- Skype, all of these things are overlays that are built on top of the internet.

And in fact, a lot of the reason for their existence is because the internet doesn't quite do the right thing in terms of the right behavior for certain applications. So people say, let me go build an overlay on top of it, wherein you take a path involving multiple links on the underlying network and make it look like one link in the higher level network. And when you do that, you get an overlay network.

So this single link on the ARPANET is actually many, many links with many switches, and who knows how expensive it is underlying in the telephone network? But all you have to do is to pay the telephone network some amount of money and make a call or whatever, and you get to view it as a single link. And you can do the same thing on the internet.

Now, this protocol the routing protocol they used was a distance-vector routing protocol. It wasn't actually even as sophisticated as the one we studied. But it was a distance-vector protocol.

And distance-vector was the first routing protocol ever used in a packet switched network. And it continued on the ARPANET for many years. They continued running this protocol.

OK, moving on, we move from basic packet networks to this problem of internetworking. And that went through a series of demos. So one of them was they had a big conference in 1972.

And they were demonstrating the simple packet switched ARPANET. And it worked really well except when they demonstrated it to a team from AT&T, and it didn't work at all. And in fact, there were news articles that were written. And some people wrote this was a nice network, some people wrote it never worked. And AT&T just thought, ah, bunch of academics, it's never really going to work.

They wrote a modified email program. And the US was not the only place where work was going on. In France, there was a really good team building a network called CYCLADES. And Louis Pouzin was the principal investigator of that system.

I think that CYCLADES doesn't get enough credit because often, as it is with these things, the winner kind of-- ARPANET became the internet, and so sort of everybody forgot everything else. But CYCLADES actually came up with some pretty interesting, groundbreaking ideas. The idea of articulating that this network is going to be a best-effort network with these packets that they called datagrams, which is a word that continues to be used to this day, was in this French network.

They originated the sliding window protocol. It looks obvious, but it's not. You can see there's lots of subtleties in how you build such a protocol and how you argue that it's correct. The first sliding window protocol was in CYCLADES. And TCP, which today is the world standard, used a very, very similar idea.

And they also use distance-vector routing. And they also implemented, for the first time, a way to synchronize time between computers. And they had a number of interesting ideas in this network.

The work was not just being done in the wide area. In 1973, ethernet was invented at Xerox PARC by a team that included Bob Metcalfe, who was another alumnus from MIT. That was inspired by this Aloha protocol that we studied. And ethernet was essentially Aloha with carrier-sense multiple access, very similar to what we did study.

This idea of contention windows is a new idea. They actually used the probability method. Ethernet standard evolved in the late '70s and '80s to use the contention window that we now know. And that same contention window idea and carrier-sense is used in Wi-Fi. So you can draw this stream of ideas through that continue to exist to this day.

It's interesting that ethernet today doesn't use carrier-sense multiple access. Because ethernet today is no longer a slow speed network, it's a very fast network. It's not a shared bus, it's point-to-point links. But it's called ethernet, and it doesn't use the same MAC protocol other than when you have low-speed ethernet.

On the other hand, wireless uses the idea from ethernet. And in fact, a lot of people call 802.11-- they used to call it wireless ethernet. And the ideas just got moved to a different domain, but it's the same ideas.

And in fact, a lot of the early chipsets that ran the MAC protocol on 802.11 networks essentially were the same as the ethernet protocol. They use the ethernet MAC. They had that piece of hardware, they would buy it and build the box around it.

So this idea of taking older technology, and applying it to a new context, and then modifying it is something that works pretty well. Because it means that you can leverage something that already exists and start making changes to it. And over time, it looks completely different.

Now, the US government and DARPA-- ARPA was funding the ARPANET. But there were companies and other research groups in the mix here. And in those days, it was not very clear what was going to win. And everybody was doing research on coming up with different ways of connecting networks together.

And Xerox had a system called PUP. I don't know what it stands for. I think it stands for the PARC-- Xerox PARC, Palo Alto Research Center, PARC-something protocol. I don't know what the U is.

And in a way, there were many technical ideas in the Xerox system that actually were arguably better in technical terms from the ARPANET and TCP/IP. But it was proprietary, whereas TCP/IP was completely open. And open meant that you didn't have to pay anyone, you didn't have to get someone's permission to do it. The process by which things were standardized was far more open and democratic. And it won not because it was better, but because it was out there and open and free.

There's a lot to be said for that model. Because for a network to succeed, you need to lower the barrier of entry so everybody can participate and implement it. And if you make network protocols proprietary, it usually ends up not benefiting anybody.

So now, I think companies have started to realize that. So everybody understands that you want to make standards open, and then keep secret any particular implementation strategy for how you implement it. So you might gain commercial advantage from implementation, but you gain no commercial advantage from keeping a protocol closed.

There are exceptions to this rule. Like, Skype is an exception to this rule. But who knows? In 5 or 10 years, I suspect that Skype is not likely to remain dominant. There are going to be other things that will come about. And some of them might be open.

In the mid-1970s, this idea that you now really start to connect many different kinds of networks together, networks that are being run in different organizations, took root. And this was the internetworking problem. And this is the problem-- people were working on this packet switch technologies.

And there were many different kinds of packet switch networks that showed up. So there was the Aloha network over in Hawaii. There were people building packet switched networks out of ethernet. At MIT and Cambridge University, there were people who were very enamored of something called Token Ring. I don't know if Victor was at MIT at the time, or any of my colleagues were, but people were building these Token Ring-based systems that were technically pretty superior in some respects and interesting.

And so there were many different kinds of networks that people were building and connecting their own campuses internally. And you had to communicate between each other. The trouble was, there was no single protocol to do this.

So ethernet had-- back in the day, when you bought an ethernet technology from say, Digital Equipment or Xerox or one of these companies, you wouldn't just get ethernet. You'd get the ethernet MAC protocol. Then you'd get some sort of network communication, network layer between the different ethernet devices. And you'd get something called EFTP, which was an internet file transport protocol. So you'd get applications around it.

So imagine now, you're buying a network thing. And you don't get to run your own applications, you get a stack of everything. And you get a box, and you only get to use whatever the vendor gave you. That was the state of networking at that time.

And people recognized this probably wasn't a very good thing. Because what you would like is to have a network where people can come up and invent their own applications and run their own applications on it. But you now needed a way to communicate between these different networks. So how do you do this?

So this was a huge project that a lot of different organizations were involved in. But a large part of the credit is given to two people-- Vint Cerf and Bob Kahn, who were in some sense the lead people in getting a community of other people together in building the system. And they articulated these visions and these ideas.

So Kahn's rules of interconnection are as follows. He first said that each network is independent and must not change. So the idea that you can bring networks together and communicate, if it required every network to change, that wasn't a palatable idea.

The second is that he agreed with CYCLADES and said, best-effort communication is what we need. Because we cannot assume that every network will guarantee delivery. There are some networks that may guarantee delivery in order, but you can't mandate that.

And what they said was, we will design this network with these boxes that we'll call gateways. And these gateways will translate between different network protocols. And in a pretty radical departure from the Bell Telephone network, they said that there will be no central global management control. There is no central place where the operation of this worldwide network or countrywide network is going to be managed.

So it's kind of a simple idea-- you have your own internal network. This might be an ethernet, this might be some sort of Aloha network or what have you. But you have these gateways here that sit and translate between these different protocols.

And we know the stuff as-- we now know that what they did was a pretty good decision, which is they made it so that these gateways will all agree on one protocol. And the protocol they standardized-- they got it wrong initially, but by the late '70s, they figured out that that protocol will be called IP, or the internet protocol.

So a node is on the internet if it implements the internet protocol, which means it has a agreed-upon plan for how the addressing of nodes works, and it has a plan for what happens when you forward a packet. You have a look-up and a routing table that looks up the IP address and then decides on the link. And that's all you have to agree upon.

So to be on the internet, all you have to do is-- a network has to support IP addressing, and it has to agree that it will send packets of at least 20 bytes in size, because that's the length of the IP header. There's very little else that it has to do, so much so that people have written standards on how you can send internet protocol over, you know, carrier pigeon. And you can-- and in fact, someone demonstrated something like this, where they had these things, and these pigeons were delivering these scraps of paper, and there was something looking it up and sending it on. So it doesn't take much to be on the internet.

So Cerf and Kahn started then designing the network. And they wrote in their original paper that you needed to identify the network you're in, and within the network, a host that you were in. And they said the choice of network identification allows for up to 256 distinct networks. Like, how many networks do you possibly need? How many organizations can you possibly have?

And they wrote, you know, famous last words-- this size seems sufficient for the foreseeable future. The problem is they were slightly wrong. The foreseeable future in their case was probably less than 10 years, and it may not have been more than five or six years. But you know, they made a mistake.

But what was interesting was, the next time the community got to make a change in that decision, they still made a mistake. They decided that 32-bit packet IP addresses are enough. And we've run out. We literally ran out, right now, of IP addresses. So they had these gateways that would translate, and you would run the internetworking protocol, or IP.

So in the 1970s, this idea of internetworking was all the rage. And in 1978, there was a really good decision made to split TCP from IP. And a lot of that motivation was from a group of people at MIT.

There's a paper here that you'll study at length in 6.033. It's one of these papers that you'll study two or three times, because it'll keep coming back, because these concepts are pretty important. It's called "End-to-End Arguments in System Design" by Saltzer, Reed, and Clark.

They have many examples, but the gist of the end-to-end arguments is that if you have a system, like let's say a network, and you want to be able to design a network, and you have to make a decision of what features do you put into the network? The end-to-end arguments say that you only put in features in the network that are absolutely essential for the working of the system. Anything else that's not crucial to the working of the system, you leave to the endpoints.

So if you think about reliability as a goal-- like, does the network need to put in a mechanism to guarantee the delivery of packets, the answer is no. The reason is is that that property is required, for example, if you're delivering a file, but not if you're delivering a video stream or talking. Because not every byte needs to get there, which means you don't put that functionality inside the network. You leave the function of achieving reliability to the endpoints, because not everybody needs it. And the only exception to the rule that the only function you put inside of the network is functions that are absolutely essential for the system to work is if the mechanism leads to significant improvements in performance.

So for example, if you run on a network with a 20% packet loss rate, it makes sense to have some degree of reliability and retransmission built on a network hub. Like, that's what Wi-Fi would do. Because if you didn't do it, you'd have sometimes a 20% or 30% packet loss rate. And that would make everything not work.

But we don't try to design our network so that between the Wi-Fi access point and your computer, we produce perfectly reliable transmission. If you did that, it would then mean that you would have really long delays. And you would be providing that function for applications that don't need it.

If I want an application that I would like to just send the bytes through, if it gets through, great, if it doesn't, then I'll do something else, that's a bad network design. Unfortunately, there are real networks today that don't obey this principle. cellular networks are sometimes problematic, like Verizon or AT&T or something.

You find there in real data that there are long delays in these networks. Because between the cellular base station and your phone, they have decided to provide something that looks like highly reliable TCP. It's kind of a bad network design, but that's how they do it, some of them. And so this is an old principle, but it's sometimes not followed, and that's not so good.

Now, the reason why packet switching and this TCP/IP split won in the internet compared to various other proposals that were floating around at the time is that this architecture, the internet architecture, is good enough for everything, but optimal for nothing. There is really no application for which the design of the internet network infrastructure is optimal. If you wanted to build a network to support voice, you'd go build a telephone network. You wouldn't build a network that looks like this.

If you wanted to build a network to distribute television data to television streams to a bunch of people, you wouldn't build the internet. If you wanted to build a network that wanted to support Facebook and nothing else, you probably wouldn't build the internet.

But if you want a network that's going to support all those applications reasonably well, including applications that you cannot imagine today, this design is a very good idea because it's very minimalist. There's almost nothing that the network does. It leaves everything to the endpoint. So I would say in fact that the most useful lesson, which you will apply over and over again-- I mean, let's say you go work at a company, or work on some sort of research project.

And at various stages of the project, there are endless discussions on what you need to do-- whether it's worth doing something or not doing. And the most important lesson that you can take away in system design is that when faced with a choice, try to make a choice that's the simplest possible choice that gets the job done. Because most likely, if you get the application wrong of whatever it is you're building, if it's simple and minimalist, you could probably pivot around and use the same thing for that other application.

So there's a famous set of quotations here. One should always architect systems for flexibility because you'll naturally never-- almost never know when your design-- everybody has these use cases in mind. But let's face it, when you're at the early stages of a project, nobody actually knows. And you'll almost always get it wrong. So it's important to architect them for flexibility, not for performance, not for lots of functionality, just for being flexible, and the bare minimum to get the job done based on what you think is necessary.

And it usually means doing that even if it means sacrificing performance. Like I said, the most important improvement in the system's performance is getting it to work. Everything else is secondary.

So there's a nice quote here, I don't know if-- any French speakers or readers? Yes?

AUDIENCE: [SPEAKING FRENCH]

PROFESSOR: Yeah, I know. Just tell me in English.

[LAUGHTER]

Good enough, all right.

AUDIENCE: Seems that perfection isn't attained when there's nothing to add, but there's nothing to--

PROFESSOR: Yeah, that's great. Yeah, perfection is achieved not when there's nothing more to add, but there's nothing left to take away. And this is a really, really good lesson.

I mean, you guys, every one of you is going to go into the real world, either at a startup company or a big company where you're defining a new product, or a research project, you go to graduate school-- at the beginning, you won't know the right answers. You have some vague ideas of what it's useful for when you design anything. And it's really important to understand that you should do the bare minimum to get it work.

And it's a really, really good idea-- less is more. I have a very simple way to think of it. I tell my students this repeatedly-- when in doubt, just leave it out. If you're not sure if you need it or not, don't do it. There's enough stuff to do.

And that's probably the most important lesson from many of the classes, at least on the system side that you'll be learning. Of course, it takes a lot of good taste and insight and intuition to figure out what's really important. I can't help you there.

OK, so by the 1980s, the internet started to grow up. And the way in which you wanted to handle growth was this simple but brilliant idea called topological addressing. So I'm going to explain what that means.

In the very early days of the internet, and including the simple small networks that we studied, every network node had a network identifier-- an IP address or some sort of a name for that node. So in the way in which we looked at it, nodes would have names like A, B, C, D, and E. But in reality, A, B, C, of course, there are some set of bits that communicated.

And in the old days of the internet, you would have a two-phase identifier. You'd have a sort of a network identifier or an organization identifier. So MIT would have a set of 8 bits, and then you would have a set of other bits here that communicated within MIT what that number meant.

So just abstractly, these numbers meant nothing in the global internet. This could just be 110111-something. And then you would have another sequence of something else. That was the basic idea.

Now, in the networks we studied so far, we wouldn't even have this. Every network node would just have some name. So what that meant is you could have a network address that was some set of bits. I could have a network address that was some other set of bits.

And the switches in the network, in order to forward packets to you or to me, would have to have entries in the routing tables that were one-to-one with all of the different nodes that they wanted to communicate with. So you would have a routing table that would be essentially one entry for every host in the network, which doesn't scale. It's just too much information.

So topological addressing is the idea that per-node routing entries don't scale very well. So what you would like to do is organize the network hierarchically. And it's sort of similar to the way in which the postal system works.

So in the 1980s, they came up with a way of doing it using three kinds of addresses. I'll call it class A, B, and C address. We don't use those anymore, but let me describe what this kind of area-based addressing means.

So here's a very simple, abstract view of this. The internet used to adopt this in some approximate way. But this is the conceptual idea.

You design the network into areas. So MIT might be an area, Stanford might be an area, Berkeley might be an area-- you know, BBN, all these different people are their own areas, organizations.

Areas have numbers that everybody knows. So that's the first part, which might be an area identifier. And then this is a-- within the area, you might have a host identifier, or more generally, an interface identifier.

What I mean by interface is that really on the internet, my computer doesn't have an IP address. If I'm connected to the internet by the ethernet, the ethernet has an IP address. It gets an IP address by virtue of connecting to a switch upstream of it. M Wi-Fi network has an IP address. If I use the Bluetooth, my Bluetooth has an IP address.

In fact, in general, sometimes, my computer might have four IP addresses-- one if I'm connected on ethernet, one on Bluetooth, one on Wi-Fi. And if I have one of those cellular modems, if I tether through my phone, every time I do one of those things, I get an IP address, OK? So IP addresses on the internet name the network interface.

So the way this area routing idea works is that within these areas, there's routing as usual and forwarding as usual. So all these nodes have-- you could recursively build sub-areas. But if you didn't, each of these guys would have an entry for all of the other nodes here.

And within these areas, you would have border routers. And these border routers would only have entries for the other areas. So if you wanted to send a packet from area 1 to area 4, what you would do is you would send a packet to one of your border routers. And that border router would have an entry in its routing table to get to area 4. It wouldn't know anything about the details inside area 4.

And so you have a nice hierarchy where inside the network, you only know how to get inside your network and to the border. The borders know how to get inside, and the borders know how to get to other borders. But the borde of one area doesn't know how to get inside any other network. So you can see now, you can recursively apply this idea and start to scale the routing system.

Now, on the internet, what ended up happening was, well, they had to apply this area hierarchy. And very soon, organizations started saying, well, I have a big area, and I have a small area. So how big do you make this thing?

In the very old internet, these were 8 bits long, and these were the rest of the address. If you have 8 bits, you can only have 256 organizations. And although Kahn and Cerf thought that was plenty enough, that clearly wasn't the case.

So by this time, people were starting to build equipment with these 32-bit addresses. And all this hardware was out there, so what do you do? So what they said was, all right, let's have three classes of areas.

For the really big guys, we'll have class A addresses. Then for the medium guys, we'll have class B. And for the little guys, we'll have class C.

What that meant is that we're going to have class A allows an organization to have up to 2 to the 24 addresses. Because class A is identified by 8 bits. So you get 24, which is 32 minus 8.

So MIT was pretty smart. They decided that they would go and-- you know, they were up there, they were doing a lot of networking research. So they said, we're going to go get ourselves one of these class A addresses because we're a big university, and we've got lots of computers.

And it was probably the case that at the time, MIT probably had more computers than most other places. So even to this day, they maintain this address, which was 18-dot-star, where the star refers to-- well, technically, star dot star dot star. So all 2 to the 24 addresses that start with the number 18, or in binary terms, whatever the 18 is-- 000-- I'm going to get this wrong, but there's some 8-bit number for 18. So anyway, they went and got this done.

Now, nobody wanted the class Cs because the classes were-- you could get 2 to the 8 addresses because the class C was defined as a 24 base. So you have 24 bits to define the organization. So you could have 2 to the 24, or some large number of organizations-- not quite 2 to the 24. But you'd have some large number of organizations, but then you'd only get 256.

Now, the organization doling out these numbers-- there's a particular organization. Actually, it was not even an organization at the time. It was like this-- Jon Postel at UCLA, one guy was doling this stuff out, and then it became an organization. Jon Postel is great. I mean, he was really-- the social aspects of how he managed this was remarkable.

So he didn't dole these out randomly. And nobody wanted this, everybody wanted this. I mean, everybody wanted that, but they got this. And this was 16 and 16. So you'd get 2 to the 16 addresses, and you get your 16-bit identifiers for these areas.

And over time, as the internet grew, the obvious thing happened. Because the thing is this is like the Goldilocks story, right? This is like, it's too big, and this is too small. This is just right.

And pretty soon, there were 2 to the 16 organizations on the internet, and we ran out. Literally, they ran out of class B addresses. And by this time, by the early '90s, they realized that this rigid decomposition into addresses in this form was just not quite right.

Because what you really wanted was a system where you allowed-- I mean, this idea of an organization ID didn't make sense. Like, what if I need 2 to the 12 addresses? Today, you would have to give me 2 to the 16, which is ridiculous.

So if I needed 2 to the 12, well, how do you actually do that? Well, I could get four 2 to the 8's, but if those 2 to the 8's were not contiguous, then those routers in the middle of the internet would not be able to treat them as one and have the same prefix to define the entire network. They would have to have four different entries, defeating the purpose of, in fact, using this kind of area-based routing. So the whole thing was kind of messed up.

So they actually got wise to this problem. And they came up with a more sensible, sane way to deal with this problem. I'll talk about that probably on Monday.

Now, in the meantime, in the 1980s, this growth was happening pretty rapidly. And they started getting organized. Vint Cerf, who was by then at ARPA, appointed Dave Clark, who was a senior research scientist and professor at MIT to a position of internet's chief architect.

And he was instrumental in writing and bringing together a lot of people in organizing how people do this kind of standardization. So there was an organization called the Internet Engineering Task Force, or IETF. That's the organization that determines and sets the standards for the protocols that run on the internet.

In 1982, a really important thing happened. This idea of this community with the internet architecture community got a real boost when the US Department of Defense looked at various ways and competing ways of designing networks, and kind of remarkably for a Department of Defense decided to pick the open standard rather than some proprietary standard, rather than some closed standard that is potentially more secure, though it really isn't. Remarkably, they said, we're going to standardize our entire systems on TCP/IP.

And the Defense Department-- I don't know if it's still the case, it probably is. But in those days, it was a huge consumer of information technology. It still is, it's just that other people consume it, too. But in those days, it was probably the dominant consumer of information technology.

And they standardized on it, and they awarded a contract to the Berkeley computer systems group to build TCP/IP, which by then had become standard. And there were many implementations, but they said, take Unix and go build the TCP/IP stack. And Berkeley did a lot of interesting things with it including creating the Sockets Layer. They came out with what today is known as open source implementations of the TCP stack.

In 1983, MIT created Project Athena, which was the world's first campus-wide-- campus-area network system. And they did a lot of work on things like filesystems, distributed file systems, and the Kerberos authentication scheme, and a lot of important ideas from this network. And they also ran the TCP/IP stack. They didn't run anything proprietary.

In 1984, the domain name system was introduced. For those who don't know it, when you go type in www.mit.edu, something converts it to an IP address. And then your network stack communicates and sends packets over TCP or UDP or these protocols to that address. How do you convert this?

Well, again, originally, this was maintained in a file. This file was called host.txt. And believe it or not, the way this file would work was that every night, I think in the middle of the night, every computer on the internet would go and download this file from one computer that was located somewhere on the west coast. Like, literally, you would get a host.txt file.

And of course, you could hack it. You could do whatever you want. And the assumption, of course, was that no one was-- I was told that in the early days, every computer had a root password which was empty, or these computers at MIT had a root password that was empty. Everybody could log in to any computer. And everybody was completely trusted, which I think is sort of not true anymore.

But you would download this host.txt file every day. And it started-- as the internet grew really fast-- You know, the internet has been growing by 80% to 90% a year, not just in the past few years, not just in the '90s. You know, it's been growing at 80% to 90% a year since 1980. So it's just on this amazing tear.

And so anyway, this idea of downloading a file every night is just not a good idea. So they created the domain name system. They had to create-- the NSF, which was the National Science Foundation, got into the act. And they became the first internet backbone.

And the backbone-- the idea is that this is a backbone that connects all of these different networks together-- in particular, all of the universities together. So they also picked TCP/IP as a standard. And again, the important lesson here is they picked it as a standard because it was open, because It was very clear that these implementations were available, they were free, everybody could contribute to it, and everybody could beat on it and improve it, and there was no proprietary technology that was held.

So what I will do here is I'm going to stop. I'll pick it up at this point on Monday, talk about congestion control, how to hijack routes, and how to send spam without being detected, and then talk about the future of networking and communications.