MITOCW | 24. History of the Internet cont'd, course summary

PROFESSOR: So my goal for today is twofold. The first is, we've looked at the history of the internet in the '60s and the '70s. And today, my goal is to tell you about what happened in the '80s and the '90s, as well as the last decade. And we'll do that maybe 30, 35 minutes.

But I want to tell you about two interesting problems that got solved, both related to topics we've studied. The first is on how the internet dealt with a serious problem called congestion collapse, which happened in the mid-1980s, where TCP, the transmission protocol at the time, was still the dominant protocol in the world, didn't have many of the new ideas that it now has. In fact, the implementation of TCP in the 1980s was almost exactly the implementation of your sliding window protocol from the second lab, from the second task of the last lab.

And in fact, as some of you are-- I think Tyler posted it on Piazza, there was a particular way in which he had dealt with the windowing scheme that happened to do better than the lab solution. And I remarked there that the reason the lab solutions are the way they are are a little bit more general than the particular topology. But the topology that I picked was specifically picked in the lab so you didn't have to worry about congestion happening.

But today, I'll tell you what happens when congestion happens and the solutions that were adopted. And it'll be a little bit of a preview to 6.033. You'll study this topic at some length in 6.033.

The other problem I want to tell you about today through these remarks is about how easy it is to hijack routes on the internet. And I'll go through some examples of this happening in reality. I mean, it is literally something a bunch of-- you know, you can do it in your dorm. You just have to convince some sort of-- pretend you're an ISP, or set yourself up as a small internet service provider. And you can actually wreak a fair bit of damage onto the rest of the world if you're so inclined, and then pretend that it was just a, you know, error.

So those are the two technical ideas. I mean, none of this stuff is really on the quiz, though it's probably helpful to think about these things about route hijacking because those apply to some of the concepts we've studied. But a lot of this is just for-- out of your own interest. And I'm hoping to pique your curiosity so later courses in the department would be interesting.

OK, so in the 1980s, the people designing the protocols of the internet started to get organized. And as you recall, Vint Cerf and Bob Kahn were the leaders of the effort. There was a big community of people contributing to what became the internet.

And back in those days, it was still the ARPANET. ARPA had funded this entire project. And it was starting to be successful.

In the 1980s, rapid growth started. People said how the internet is booming, and how it's exploding at 80% or 90% a year. It's been growing at about 80% to 90% a year since about 1983 or 1984. This explosive growth has been happening for several decades now.

Dave Clark, who was a senior research scientist at MIT and on the faculty in our department, was designated as the internet's chief architect. And one of the things he did was to get the community organized and formalize the creation of internet standards. You know, you have all these different companies and organizations and universities coming together. How do you standardize protocols?

And the argument-- the approach they came up with was called the Internet Engineering Task Force, or IETF. And they would write these proposals. There had been this trend of writing these proposals for people to comment on, called RFCs, or Request For Comments. And now, there are several thousand requests for comment.

Nowadays, requests for comment are after the comments have been made. Usually, you go through an internet draft stage. And by the time it's a request for comment, it pretty much means that the comments have already been done. But historically, they were requests for comment, so they called it request for comment.

So if you ever go to a company where you're asked to-- lots of things that are on internet request for comments. And often, people are asked to implement pieces of various standards. And you typically look at this document, and you try to see if there's someone written code around it. And then you adapt it or write it from scratch to the specification. So it specifies the protocols.

And the interesting part of the standard was very much in keeping with the general approach of the internet ethos, which was to try to make everything be open. There are many standards bodies in the world. The IEEE has various committees. The International Telecommunications Union does the various telephone standards. There's many, many standards.

And often, they're based on voting. And voting means that people horse trade. You know, you have a favorite feature, I have a favorite feature. They're both pretty crappy features, but you know, I'll vote for your feature if you vote for mine. And people end up horse trading.

And in the IETF, in the good old days, this kind of stuff didn't happen. Now, it's changed. But in the good old days, there was this quote, it says, we reject kings, presidents, and voting. We believe in rough consensus and running code.

The idea was that you show me what it does. Allow people to experiment with it. And only after we see some prototypes and some actual experiments in the lab will we even consider making it into a standard. And this was the good old days of the internet in which things were on rough consensus and running code. So it wasn't around, you know what, let's just vote, and we'll get all our lousy features in just because we care enough about it personally.

A big break happened in 1982 when the US Department of Defense decided that proprietary technology was not the way to go with networking, and standardized on TCP/IP as the standard. And back in those days, the Defense Department was a huge consumer of IT. It still is, but back then, it was just hugely dominant. And they could really influence how the world went.

In 1983, MIT created a Athena, the first really big, large-scale computing project. We still have Athena machines. And it showed how to build campus-area networks and campus-area technologies. They built distributed file systems, they built systems like Kerberos. And in fact, they were one of the first groups, the first networks to experience network congestion problems because everybody started using the network to do their work rather than have their own computers at their desk or use the terminal to log in to some remote mainframe, which was the way in which things used to be done.

I mentioned last time that in 1984, we created the domain name system. This is the system that goes between domain names like mit.edu to an IP address. And before that, it was an FTP-- how many people have heard of FTP? A few of you, OK-- File Transfer Protocol. Nobody really uses that these days.

But you used to download this-- every night, computers would download it from one machine located, I believe, in California. And it didn't really scale, you know? And people decided that you needed to get organized and build a domain name system.

In 1985, ARPA was-- the Defense Department was starting to get out a little bit of maintaining the entire communication network. And the National Science Foundation started taking over the running of the backbone, connecting all the non-military networks in the United States. And this led to the NSFNET. I'll have more to say about NSFNET in a little bit.

There were two big growth problems that were experienced. The first was congestion, and the second had to do with how addresses started running out, and we had to deal with problems in the routing system. So I want to talk about congestion.

In 1986, the internet experienced the first of a series of congestion collapses. A congestion collapse is a situation where you end up with a picture that looks like this, where if you were to draw the offered load on a network, or any system for that matter-- which is just how many people are clamoring to get access to the network and send their data on the network.

If you were to plot the throughput that you get, or the utilization that you get from the network, you typically would see a curve that looks like this. As the offered load grows, you might see a somewhat linear increase in the throughput with slope 1 because the network isn't congested, no packets are being dropped. You push some data in, you get the data out. The throughput tracks the offered load.

And at some point, you reach the capacity of the link, or of the network in general. And you might end up with a flat curve like that. And that's fine.

What you would really like to do is, as the overload load keeps increasing and the throughput saturates at the capacity-- of course, that means that-- what does it mean? Either the delays are growing to infinity or packets are being dropped, right? And what you would really like intuitively is for the sources to realize that packets are being dropped or they're not being delivered in time, so maybe slow down. Something has to happen.

But congestion collapse is a worse phenomenon. What happens is, beyond a point, your throughput [INAUDIBLE] drops, sometimes precipitously, and it might go down to zero. So people were running network applications-- you know, FTP and remote logins and so forth, email and so forth. And what they were finding was that you could be running-- the ratio between what the capacity of the network was and what you were getting when you were running was a factor of 100 to 1,000 worse.

So this is a real collapse. I mean, you wouldn't be able to get anything through. So people were talking about going from, in those days, tens of kilobits per second to bits per second.

In fact, there was a joke that there was a path in Berkeley between the University of California and Lawrence Berkeley Labs, which is probably 400 meters from each other. And the network rate was such that you could actually run up the hill with a tape drive and you'd get 100 times higher throughput than what you were getting through the network. And you know, this was not even running up very fast. So this was a serious problem.

This problem was dealt with by multiple people who were working on it. And it led to a set of algorithms where the idea was that you would like-- because all these switches and routers were already deployed, people were interested in end-to-end solutions to the problem. By end-to-end, I mean solutions you could deploy at the centers and the receivers in the network without worrying about what the network was doing in between.

The actual algorithm that we all run today had its roots in an algorithm developed by Van Jacobson at Lawrence Berkeley Lab. And there's a lot of work that has been done since then by many other people in the community as well. And in parallel, there were people inside of Digital Equipment Corporation over here in Massachusetts working on similar problems. And they came up with ideas.

And both these ideas were basically the same idea, more or less, except in the detail. And they also resembled what we've seen with MAC protocols. The idea is that if the network is working reasonably well, we're going to try to be greedy and start to send faster and faster. At some point, we're going to find that the network doesn't work so well. And we can determine that in one of a few different ways.

One way to determine that is that packets start getting lost. And we will assume that packets are lost because queues overflow and congestion happens. We might also alternatively assume that as queues in the network grow, packets get delayed more and more when the network starts to get congested. And if we find that the roundtrip times are starting to increase, maybe we determined that congestion has happened.

Now, there's been 30 years of literature. The problem has not yet been completely solved. And in fact, we have now an active research project going on in my group about how to deal with these problems in the context of video and video conferencing in cellular, in wireless networks that you run with your phone, on your phone. So it's still an problem, lots of interesting research.

But the basic idea is that you have to adapt to what the network is doing. And the way you adapt is by watching things in the network. You watch where the packets are being lost. You watch whether delays are growing. You might watch what the receiver's getting-- you know, how fast is the receiver getting data from the center and the network.

And the intuition is the following-- let's suppose that we were to pick the correct window size. This goes to the sliding window and the window size that you use. We said that the right value of the window size is roughly the bandwidth-delay product, where the bandwidth-delay product is the product of the bottleneck link rate multiplied by the minimum roundtrip time.

But the problem is the bottleneck-link rate is not fixed. It is in the lab we studied, but in reality, you have many connections sharing the network, many applications sharing the network. And people come and go, so the rate keeps changing.

At one moment, you might be getting 100 kilobits a second. The next moment, you might be getting a megabit per second. And on wireless networks, it's even worse.

Quite literally, if I were to start an application on my phone now, connecting to Verizon or AT&T, and if I step out of this room, it's quite likely that the actual [INAUDIBLE] experience by my phone might change by a factor of four or a factor of eight within two seconds. And that has to do, of course, with the fact that the signal-to-noise ratio-- I mean, we're surrounded by thick walls and metal. And the moment I go out, it's going to be different.

So how do you deal with this problem? There's this one basic idea that's used that's a fundamental idea, very important idea. And it's called conservation of packets. It's the same idea that you saw when you built your sliding window protocol.

It says that when you put in a packet into the network, and the packet reaches the other end, and you get an acknowledgment for the other end, it means that one packet has left this pipe. If you view this as that picture, as you see in that picture up there, packets are like water entering a pipe. And then they leave the pipe, and then another one comes back.

If you have managed to somehow, by some magic, pick the appropriate window size, then conservation of packets is a good principle for you to apply. Because what it says is that the only time you're allowed to put one more packet into the network is when you're sure that a packet has left the network. And the way you know that a packet has left the network is because you receive an acknowledgment for that packet.

Now of course, this assumes that you know the right window size. Conservation of packets has another really nice advantage, which is that-- let's say that you think you have the right window size. But in fact, more traffic comes in, and the bandwidth, the rate at which you can send data, reduces.

What's going to happen is that because other traffic came in, the roundtrip times are going to grow, which means that acknowledgments to your packets are going to come back a little slower, which means that you have a natural slowdown. Because acknowledgments come slower, you naturally slow down and send packets a little slower. And then of course, then, if the congestion persists, packets are going to get lost.

And when packets get lost, the trick is you have to reduce your window size. So let's say you're running at a window size in your lab of 50 packets when you implement this. If you find that packets are getting lost, you should drop your window size. And one way to drop the window size that TCP uses is to reduce it by one half. And then every time you find that acknowledgments are coming back, you get a little greedy and you try to increase the window size. And there are many ways to increase the window size.

Now, all of this stuff requires a way, when you start the connection, when you start an application, what do you do? And I think you pointed out an idea the last time when I first talked about this problem, which is in the beginning, you let your window size be one packet, OK? So you start your window size at one packet.

So it's like stop and wait. So at the beginning of a connection, If I were to draw time here against the window size, you start at one packet, and you just ship that packet out. It takes an entire roundtrip to get to the other side, you get an acknowledgment back.

At that point, with regular stop and wait, you keep the same window size of 1, and you send one more packet. You're not being greedy enough. So one thing you can do is, when you get one acknowledgment, you double the window size, or rather, increase the window size by 1. Every time you get an acknowledgement, you increase the window size by 1.

So the rule is, on [INAUDIBLE] you take w and you go to w plus 1. What does that do? Well, after one roundtrip, if I draw this in multiples of the roundtrip time, this is one RTT, this is two RTTs, this is three RTTs, and so forth.

So at the beginning, at the zeroth RTT, your window size is 1. You get an acknowledgment, you make your window size be 2, right, 1 plus 1. So at this point, your window size is 2. What is the window size after two RTTs? It's 4 because-- why is it 4? When you send out two packets-- yes?

AUDIENCE: You get two [INAUDIBLE].

PROFESSOR: You get two [INAUDIBLE] back. So for the first one, you went from 2 to 3. Then, you went from 3 to 4. So your window size grows to 4.

And then out here, if there's no losses, then you haven't yet reached-- the window size hasn't yet reached the bandwidth-delay product of the connection, you're increasing exponentially. So this is at 8, and so forth. You're growing fairly rapidly.

And at some point, you're going to grow too fast. Your window size is going to exceed the bandwidth-delay product plus the queue size of the bottleneck, causing a packet to be lost. And at that point, you can do a bunch of things.

But one thing you can do is to drop the window size by a factor of 2. So whenever that happens, if there's a packet loss, you drop by a factor of 2. And you could continue to try to grow exponentially, but that would be stupid. Because you would grow exponentially, and if the network conditions haven't changed, you're going to again drop.

So at this point, you could do something else. And what TCP does is to start to grow linearly, rather than exponential growth. Once you experience congestion, you start to grow linearly. And then maybe you experience congestion here, you drop by one half and grow linearly. And you have this sort of sawtooth behavior.

And for most web connections that involve downloading a small amount of data, you end over here. For video and everything else, you go all the way. And this is one strategy. There are many, many others.

And like I said, in various kinds of networks, like wireless networks, it turns out this approach is not really that good. And there are open questions around how you should actually design the system. Does everyone understand the basic idea? This is an example of adaptive congestion control. You'll look at this in 6.033 and in 6.829 in more detail if you took those classes. Any questions? OK.

All right, I'm now moving over to the 1990s. Nothing else interesting happened in the '80s. But in the 1990s, more things happened. ARPANET essentially ended as far as universities and everybody else was concerned. And in fact, it transitioned into-- you know, there were separate military networks.

And in 1991, Tim Berners-Lee, who is also now here at MIT, a professor, invented a little thing called the WorldWideWeb, called with one world. WorldWideWeb was the name of the program. And I found this thing which was very interesting.

So he wrote a proposal in 1989, I think, to CERN, where he was working, to his boss at CERN. And it was called, "Information Management-- A Proposal." And there were all these things, you know, about how what became the web should work, with links and so forth.

And his boss at the time, on top wrote, "vague but interesting" as his feedback on the proposal. Presumably, the interesting part trumped the vague part, and allowed him to actually proceed on this project, which became the World Wide Web. Now, obviously, it's been tremendously successful.

Now, by the mid-1990s, the NSFNET, which was the backbone connecting the US, various US organizations, the government decided to essentially get out of the internet service provider business. Or rather, the government decided not to fund that activity anymore. And many of you have probably heard about this joke about Al Gore inventing the internet and not inventing the internet, and people saying he didn't invent the internet.

Well, there's sort of a little bit of truth in this. Al Gore was very instrumental in the government kind of getting out of the internet business, and was involved in committees that set up regulations that led to internet service providers, commercial internet service providers actually forming, and commercial ISPs starting to take off, which was a really, really big change for the internet. Because no longer was it the case that there's this one organization and one backbone network that connects MIT and Harvard and everybody else together, and all the companies together. You had many, many people who could offer internet service, and in fact compete with each other.

The idea was the internet service providers or different network operators have to cooperate with each other because we are interested in connecting everybody on the internet together. But they also compete with each other. And their reason to compete is they compete for customers. If I'm a customer of Verizon, I'm not a customer of Comcast, for example. And yet, Verizon and Comcast and other ISPs have to actually cooperate to get packets through. So how do you do this?

And it turns out that this is a tougher problem than you might think. And the world-- people invented this protocol called BGP, or the Border Gateway protocol, which uses an idea that we've seen. It uses Path Vector to solve this problem. And I'll talk a little bit about that as well.

The other thing that happened in the internet was that IP addresses started to get depleted. And we saw why the last time-- everybody wanted those Class B addresses. And now, in fact quite literally, there are no more IP version 4 IP addresses. And so there was a lot of work done on moving to other versions of IP.

But the part that is interesting is this idea of classless addressing. So the idea was rather than have organizations that either have to have 2 to the 24 addresses, or 2 to the 16 addresses, or 2 to the 8 addresses, let's allow organizations to have any number of addresses. So I want to tell you a little bit about what an IP address means, because everyone has seen an IP address. I want to explain what it means, and how it really actually works.

So if you look at an IP address, 18.31.0.82, which is one of my machines, that dotted decimal notation with human-readable numbers used to make sense in the old days. It used to be that this is a Class A address from MIT. It's 18 dot whatever, and MIT owned all of that stuff. Now, that also happens to be true, but as far as the network infrastructure and the switches are concerned, this thing is nothing more than a 32-bit number that looks like that, OK?

Now, when a packet with that number shows up at the switch with a destination address with that number, really what happens is that the switch, the router does not have an entry for every one of those destinations in the world. If it did, it just would be too much information. So what it has is information corresponding to a certain prefix.

Now, that prefix could be of arbitrary length. It could have an entry in it with just the first 8 bits, which would signify that all packets that show up with that first prefix of 8 bits would have to be forwarded according to a rule-- according to the link that was set for those first 8 bits. Or it could have an entry in the routing table for 16 bits. Or it could have an entry in the routing table for 19 bits, or whatever. And that depends on how the routing system-- how we advertise the routes, and what it contains.

So there's an important lesson here. When a switch advertises a route for a destination, on the internet, the destination is not the destination of-- is not the IP address of an endpoint. But what that destination is is a prefix that represents a range of IP addresses, all of which are forwarded the same way by the switch.

So one way of writing this in notation that we can understand as human beings more conveniently is this idea of writing it as 18 slash 8. What that means is-- this notation stands for all IP addresses which have the first eight bits in common, which will be 0001011010, which stands for the human readable number 18. And it contains all 2 to the 24 addresses corresponding to that prefix.

So as another example, if I have that in my routing table with a slash 17, it stands for 2 to the 15 consecutive IP addresses, all of which share the first 17 bits in common, OK? Does this makes sense? So this is what an IP address means.

And as far as a switch is concerned, a routing table entry is not the IP-- it's not a destination IP address. But it's something in this form. It contains a prefix, and it contains a [INAUDIBLE]. So in human notation, 18.31 slash 17 would be 17-- 2 to the 15 bits, which share the first 17 bits in common.

So what this means is that with this notation, inside the forwarding table, you can have an entry for one IP address, or two, or four, or eight, or 16, all the way up to whatever the maximum is, right? So it allows us to build networks of different sizes, and let that network's identifier be known to the rest of the internet.

Now, in principle, you could put every host in the network. And that would mean that you have an entry that-- each of which looks like a slash 32. If I did a slash 32, it meant that it's an individual IP address. But that wouldn't scale. And so we want to allow people the flexibility of having very different ranges sitting inside the routing system.

And there's one more rule. And this rule is important because I want to tell you this rule-- because I'm going to tell you how YouTube was hijacked by an ISP in Pakistan. And it relies on your understanding this particular forwarding rule. And then I'll tell you about how an ISP in China hijacked 15% of the internet for a couple of hours, for a few hours. And that didn't require this rule, but it's two examples I want to tell you about.

But let me explain the rule-- the forwarding at a switch uses an idea called the longest prefix match. So what that means is that if you have entries in your forwarding table-- let's take those two examples. And let's say I have an entry in the forwarding table. This is a particular switch, I have a forwarding table or a routing table.

And the first entry says 18 slash 8. So what this means is it's 2 to the 24 addresses that share the first eight bits, which is whatever corresponds to 18.0.0-- whatever it is, right? And then let's say I have another entry, which is some 18.31 slash 17. And what that would be, of course, is 2 to the 15 addresses. And the prefix would be shown in that picture there, whatever the first 17 bits-- so some 17 bits.

Now, if the switch received a packet with an IP address, and that IP address matched multiple entries-- you know, you might have other entries sitting here. Let's say that I have 128.32 slash something. And I might have various other entries sitting in my forwarding table.

When a packet arrives, it may, in general, match multiple entries here, right? Because the packet has a certain bit string in its destination address. And that destination address now matches multiple of these. It could match one or more of these entries here.

What an IP router does when it gets such a packet is to find the entry which matches in the longest prefix. In other words, the routing table entry that corresponds to the longest prefix match between the destination address of the packet and between the entry in the forwarding table is what you use to send the packet on. So if you got a packet that was 18.31.6.5 and it happened to match this entry, it would go on one link. Let's say this was link 1.

And if you have got this other thing that didn't match this, but matched that, you might have a different link. Let's call it link 0. So the output link that you use depends on the longest prefix match.

And MIT and many organizations use this extremely well. So I was remarking to some of you at the end of last time that-- You know, it's funny, MIT has multiple internet service providers.

And it turns out that if I use this network here and I download-- you know, I go to linux.org, which is a place you could download Linux code. If I go from here, it so happens that MIT users level three, which is I think the world's biggest, or US's biggest internet service provider to get those packets. The same thing-- if I just go up to [? Stata, ?] and I connect to the [? Stata ?] wireless, and go to linux.org, packets from linux.org come back to me through a different ISP, Cogent.

So MIT has decided that it wants to load balance its traffic. So it advertises the prefix corresponding to the network in this room, whatever the Wi-Fi network in this room, through one of the ISPs. And it advertises the other prefix in Stat through the other ISP. And it does it presumably to load balance traffic.

And it also has this idea that if one of those links were to fail, it would switch the traffic through the other link. Organizations do this because they would like to provide good service to the people inside their organization. So the longest prefix match is very crucial to how this stuff really kind of works. So keep that in mind. I'm going to come back to this. On to the next slide.

In the rest of the 1990s, a few interesting things happened. One of them was that work started on this new proposal for IP called IPv6, which said, let's not use 32-bit addresses. Let's go to 128-bit bigger addresses and try to solve the address depletion problem. IPv6 has taken a really, really long time to get deployed for reasons I won't go into here. But it seems to be happening now.

But people keep saying that it seems to be happening now. I said that three years ago when I did the wrap-up in this class. So some time-- at some point in the future, that statement will actually be true.

Now, you know, everybody knows about Google reinventing how search is done, and it starts to dominate. Another thing that happened in 1998 was content distribution networks started getting created. And these are networks that you deploy as overlay networks atop the internet to serve content better, in a more reliable way.

Now, in the 2000s, the internet matured. And I have the top five things that I think happened in the networking industry and the networking world in 2000. So this dot-com bust happened, and then 9/11 happened.

The first thing that happened was the rise of peer-to-peer networks. I'm sure many of you have used this-- Gnutella and Freenet, BitTorrent is the latest one. There was a lot of research done, including here at MIT, on how you build these peer-to-peer networks, the idea being, you don't have a central point of failure. And you can use this to distribute files extremely efficiently. I mean, BitTorrent is still highly dominant.

And distributed hash tables like Chord, which was developed here, and other schemes that are used now by systems like Skype, they are used inside data centers like Amazon. If you go to Amazon and buy stuff, it uses a system called Dynamo, which is a key value store that basically builds a distributed hash table. So it's had a lot of impact, both in data centers and in systems like Skype.

The second thing that happened was that in the early to mid-2000s, security became a huge deal. People started attacking the internet, which came as somewhat of a surprise to people who grew up in the good old days of the internet where, as I mentioned last time, computers with root passwords that were empty because everybody could be trusted. And then they found that as people started making money on the internet, people started trying to attack the internet as well.

Denial of service attacks started, where people would launch attacks on websites. And they would often use it to extort money. This would be like, if you don't pay me, I'm going to continue to pummel your website so that you can't sell flowers, or whatever it is you were doing on the web.

People found vulnerabilities in software, and there were many worms that spread, often pretty quickly. SQL slammer is a particularly interesting one of these. We studied this stuff in 829 and in 6.033 in some detail.

But this was remarkable because in 30 minutes, it clogged the world's networks. I mean, here's a picture of a screenshot. This was at 5:30 in the morning, I guess UTC, Greenwich Time. And you know, nothing's going on.

And then half an hour later, the blue splotches show the networks that were clogged. And almost every computer that was vulnerable to this-- there weren't that many computers, relative to the world's computers, that were vulnerable to this. But all these networks got hammered, and in fact, traffic came to a halt.

And this worm showed that the power of spreading-- if machines trust each other, either implicitly or explicitly, it's very easy to actually find a vulnerability in one and then spread very, very quickly. So a lot of work was done on how to handle worm attacks. A lot of this has to do with putting things inside of networks, which is running at high speeds, to identify patterns that-- in the payload of-- in the data that's being sent in packets to quickly identify that this corresponds to a worm, and then throw those packets away before they hit the actual endpoints.

Right now, we don't see too many worms spreading. The ones that spread now are slow-spreading worms that are often spread by human contact. It's like people clicking on links they shouldn't click on. It runs the program, finds a vulnerability on their machine, and then it resides on their machine. And often, these are then used to create these big botnets that are used to launch denial-of-service attacks, or are often used to send spam and do other things like that. So they're still going on, but you don't hear about them in the newspaper.

Spam became a huge problem. And it continues to be somewhat of a problem, though these days, the distinction between spam and internet marketing is kind of coming down-- the gap's closing. But by and large, spam now is, while I wouldn't say it's a solved problem, it's generally combated by big organizations that have enough data, enough email coming in that they can identify spam and then filter those away.

Route hijacking is the other problem. That remains a huge vulnerability. So I want to tell you about two examples of route hijacking. And I don't think there's easy-- the technical side of this problem, we understand how to solve. But how to deploy good solutions is still unclear.

So the first example is from-- this problem has been going on. Every three years, you see a big route hijacking problem. The first one was from 2008-- I think it was 2008, where YouTube was unavailable to people for a few hours everywhere in the world.

Now, you could argue whether watching cats dance or whatever is not important. But the fact is that YouTube has a lot of money. Google has a lot of money, and even they were vulnerable to this trouble.

The second example was a little different. China Telecom managed to get about 15%, roughly speaking, of the internet traffic to go through them. Now, the interesting thing about that attack is that it wasn't clear it was an attack. I should just say that error, or failure, was that people didn't even notice.

Because unlike YouTube, where you go, you couldn't get your data, and then people noticed it, and then they were scrambling to solve the problem, with the Chinese attack, or the Chinese vulnerability, what happened was that China Telecom managed to divert the traffic that wasn't supposed to go through them-- to them. And then they forwarded the traffic on to the rest of their destinations, the actual destinations. So you would find things like, instead of my latency being 100 milliseconds, it might be 500 milliseconds, which is-- you may not even notice. Or sometimes, you notice it, and you go, ah, yeah, that's just the internet being the internet, you know? Sometimes, that happens.

But the fact is that they were able to-- an ISP was able to essentially divert a large fraction of the world's traffic. And both these attacks fundamentally stem from the following problem, which is that at the end of the day, despite all of this investment into the internet, and the importance of the internet, and the amount of money in it, internet routing works because of essentially an honor code. It's like, ISPs at some level, internet service providers and organizations trust each other.

And there's this transitive trust, which is I might trust you, and you might trust her. And implicitly, the way routing works is that ends up in me trusting her. Because I trust everything you tell me, and you happen to trust everything she tells you, which means that if she were to make a mistake, and you were to believe it, then in essence, everybody else in the world is vulnerable to this problem.

So let me explain what happened in the case of YouTube, because it's reflective of how things really work. So here's this little ISP called Pakistan Telecom. Tiny, tiny ISP, you know? Hardly anyone uses it outside-- I mean, everyone in Pakistan probably uses it. But in the grand scale of things, it's completely tiny.

So they end up connecting to a bunch of people outside in different parts of the world. And one of the people they ended up connecting to was another ISP out in Hong Kong called PCCW. And these guys connect to the rest of the internet. And presumably, Pakistan Telecom connects to other people-- I don't know.

Now, here's what happened. Now, where does YouTube fit into all this? You know, YouTube is sitting somewhere over here. And presumably, it's not directly-- I mean, these guys have nothing to do with each other.

These guys are connected to some other big ISPs and small ISPs. And eventually, there a set of internet service providers that somehow constitute the internet. And each of these is independent. Each of these is known as an anonymous system, or AS.

And each of these autonomous systems has a number, a 16-bit number. MIT is an anonymous system. And because MIT was very early in the internet, MIT'S autonomous system number is 3.

But right now, there are many, many ISPs. You know, right now, you can have-- there are tens of thousands of autonomous systems-- I don't know, 35,000, 40,000, 45,000, something like that. We're number 3! So anyway--

AUDIENCE: Who's number one?

PROFESSOR: Who's number one? BBN. Yeah, BBN. I don't know what AS2 is. BBN, of course, is number one. But actually, that number is owned by somebody who acquired BBN and who acquired-- somebody acquires it.

Now, here's an interesting thing that's happening-- these autonomous system numbers-- I remember I was a very young graduate student when people were talking about these autonomous numbers. And I remember these mailing list discussions. I wasn't working on this problem then, I worked on it much later.

But people were saying, yeah, 16 bits is plenty enough for an autonomous system identifier. Because remember, NSFNET was one. There was one internet service provider. And in the early '90s, mid '90s people were talking about ISPs, and they said, oh, 16 bits is plenty.

And I remember there were some people, actually graduate students who were saying, maybe we should make it 32 bits? Because people remembered that-- you know, the internet started-- you remember those old things on the internet where people said, 8 bits for a network identifier are plenty enough. And of course, they got screwed.

So anyway, what's happening now, of course, is the older guys were saying 16 bits is plenty enough because we don't have too much overhead on packets. And they put in 16 bits. And now, we're at 45,000 or 50,000.

And guess what, there's a proposal now on how do you-- how the heck do you extend this to more than 2 to the 16 autonomous system, because now, the internet is growing. So you know, if ever you're given an opportunity to design the number of bits for something-- and you will always have to do something-- just pick something much, much bigger than you imagine, and then double it. Because it's always-- you'll never get it right.

So anyway, each of these autonomous systems has an identifier in it, OK? And each of these guys, when they make an announcement in the routing system, the way it works is that you create your identifier, and then you tell people all of the IP prefixes that you own, OK? So when I create my distance vector, or in this case a path vector advertisement, if I am autonomous system 3, I have a set of IP addresses.

So MIT might have 18 dot whatever slash 8. MIT has 128 dot something slash-- let's say 19. MIT has a whole slew of these IP addresses that they've acquired. And what they're going to say is, I'm autonomous system 3, and I know how to get to these guys because I own these guys. This is the origin announcement.

And then other people-- you know, MIT might-- this is AS3. MIT sends it to its ISPs, and they send it to their ISPs. And every time an autonomous system receives multiple advertisements along different paths for the same prefix, they pick one. They select among them.

They have some rules to select among them. And these rules have to do with the length of the path between autonomous systems. They have to do with how much you're paying.

So for example, MIT might be getting a better deal from Cogent than from Level 3. And so it would want more of its traffic to come from Cogent. And therefore, it would decide that it would only advertise certain of its addresses on certain paths.

So there's lots of policy. It's very, very complicated. But yet, you know, some miracle, the whole thing works.

So anyway, these things go through from autonomous system to autonomous system. So what's this path vector, right? Remember, I told you about the path vector.

The path vector is a sequence of paths. So it could be 3, 17, 26, and so forth. So I have an animation of this thing. So I want to show that to you because it's totally interesting.

So there's this website called [? BGPlay, ?] if I can find it. So there's a way to get MIT'S route advertisements over the past month. And you can kind of see how these stats change.

So anyway, what happened to YouTube? What happened to YouTube was the government of Pakistan decided that what they would tell Pakistan Telecom was to not allow their users to go to YouTube. So what they did was, rather than simply drop those requests, they wanted to get Pakistan Telecom, when somebody clicked on a YouTube link, to go to a website that Pakistan Telecom would run that basically said, you're not allowed to use YouTube, but would you like to see something else?

OK, so the way they did that was, they decided-- YouTube has some address. Let me call YouTube's address Y something, something. Let me just call it Y, all right? Y is a set of IP addresses that correspond to YouTube. It's not one, they have many machines.

So what these guys did was, they did something they thought was very clever. They have a whole network of users there. They decided they would advertise a route for destination Y inside their network. But rather than use the actual route advertisement for Y, they would actually send it to their own machine.

So remember, they changed the routing now. They're telling the users-- they're hijacking the route. They're telling their users that to go to YouTube, you should not use the actual link that I'm telling you to use, but instead, go to this other place inside my network, where I can show you this other website-- maybe pretend it's YouTube, or whatever.

Now, everything is so far so good, right? I mean, people do this all the time. When you go to a hotel or any internet kiosk, internet place, you take your laptop, and you go to cnn.com. And the next thing you see is, would you like to sign in?

How does that work? Well, that works because they essentially hijack the route. They make it look like you're going to CNN, but in fact, you're going somewhere else, right? After all, your computer wrote to go to CNN. And the IP address used was presumably CNN.

But then they made a mistake. Some guy here-- probably, he was too tired-- set up a configuration. So he advertised his new route to Y that he created out to PCCW.

Now, PCCW was to some degree at fault. Because PCCW should have known to some degree what actual IP addresses Pakistan Telecom owns, and only honor route requests, route advertisements coming from those things, right? PCCW needs to know how to go, how to send packets to nodes inside of Pakistan Telecom.

But clearly, this guy has nothing useful to say about how to send packets to YouTube. But yet, this guy honored this message. So there were two mistakes here. Actually, there were many mistakes.

But the two big ones were-- there was a mistake made here sending something out. There was a mistake made here honoring this request. By this time, you have transitive trust. Because PCCW would find--

And what they also did was, they made this a more specific prefix. So YouTube was advertising a slash-- I believe it was slash 21, which meant it was many, many bits in the prefix, 11 bits in common. And there were-- sorry, 21 bits in common, a slash 21.

But these guys advertised-- to guarantee that all traffic would come in, they advertised to slash 24. So it's more specific. So what happened was PCCW believed that, and they advertised that to their ISPs, and then to their ISPs, and so forth.

Now, the reason why this really got-- all of the internet's traffic were sent toward this poor guy in Pakistan was because everybody is doing this longest prefix match. And it's true that there's a legitimate route to YouTube in those routers. But they're ignoring it because they find a more specific route that somebody had advertised.

So if you want to get traffic sent to Google, figure out a way to convince some bigger ISP to take your route to a more specific prefix that you know is owned by Google, and just advertise it out, OK? You're probably going to get a lot of traffic. Now, you may not want all the traffic, but you'll get it.

So you see how this stuff spread, right? How do you solve this problem? All right, this is going on, and then people are not able to see their cats. And they're-- they're scrambling. I mean, this is literally the whole internet wasn't able to get to YouTube, which admittedly is not the biggest problem in the world. But still, if you're YouTube, this is a big problem.

So how do you solve this problem? What do you actually do? I mean, the whole world has this now, this bad entry in the routing table.

Now, there's this long-term solution, which is of course, let's figure out a way to authenticate the advertisements, and use some public key, and this and that. And you'll study this in 6.829 and other courses. But today, we don't have that. And this problem exists, so how did we ever come out of it?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah, actually-- you know, it's interesting, you and I-- you certainly thought about this in 30 seconds. But YouTube actually did-- it took them a while to figure out what was really going on. Because one of the problems is, you don't want to do something like that without knowing for sure.

But eventually, they did exactly this. They figured out the prefixes that were being advertised. And that took a while, because you don't know somebody else's-- what's in your routing table? I don't know. I have to pick up the phone, or email. And email may not work because it's coming back to YouTube.

But whatever, there's a way-- there's a way to figure this out, phone and [INAUDIBLE] Gmail or something. And then they figured out what it was. And then they inserted slash 25 that were more specific. And then once the problem kind of resolved itself, they got out of it. So let me jump on and move on. Give me two more minutes, and I'll finish up.

A similar problem happened with China Telecom. They didn't advertise a more specific route, so they only got about 15% of the internet. And they were nice enough to forward it around to the actual destinations. But this was a problem. I'm going to skip through the decade ahead because you know what, you'll find out soon enough.

All right, what I do want to do is to summarize 6.02 in one slide. This course was about how to design digital communication networks, right? I'm sure you all kind of know that.

We did this with three layers of abstraction, very simple story-- bits, signals, and packets. And I think as far as these kinds of courses go across the world, it's a pretty unique storyline. There aren't very many courses that we know of which have this vertical study across all the layers. And there are some schools that we know of that are starting to adopt this idea.

And we feel like this is a pretty unique way in which to teach this, because it demystifies all of the layers. So we didn't cover anything with-- we didn't tell you 15 ways to solve any given problem. But we told you one good way to solve each of the problems that you will see.

So we went from point-to-point links to multihop communication networks. And across these different topics-- and you can see that we studied these different topics. The two big themes that I want to leave you with are reliability and sharing. Because that's really what makes our communication systems work.

How do you make it reliable? We don't have a perfectly reliable communication medium at any layer. So how you make things reliable is important. We studied this over and over again.

And how do you share? Those are the two big topics.