MITOCW | 18. MAC protocols

PROFESSOR: So the goal for today is to talk about a particular kind of communication network called a shared medium network. And there are many, many examples of shared media communication networks that exist today. Now, these are, generally speaking, networks that don't span the globe or even span the country, with one exception, and that exception is the satellite network shown here. The model here is that you have a set of ground stations.

This is a picture from an actual internet service provider in the middle of the Pacific. I mean, I'd love to go there on assignment. And the way this network works is in order to communicate between these islands, the way they do this is they have satellite ground stations that can receive and transmit data. And there's a satellite up in the sky. So you communicate between two of these islands by transmitting data from one of the ground stations up to the satellite and down there.

And the way this system works is not to divide frequencies amongst the different satellites. And the reason they don't do that is that they don't know how often each of these satellites-- these ground stations is going to communicate. So you don't divide up the frequencies upfront between the ground stations. So all of the ground stations sending to the satellite do so on the same frequency.

And the satellite downlink is on a different frequency. So what could happen if two satellites communicate up at the same time is that the satellite up-- if two ground stations communicate to the satellite at the same time, the satellite may not be able to pull together the two different transmissions apart because they're on the same frequency. Now, they could use frequency division multiplexing. That's a perfectly reasonable solution. But they don't do it because if one satellite-- one ground station has data to send and the other doesn't, you end up wasting frequencies and not getting as high a throughput. So that's one example of a shared medium network.

This here is a picture of something called ethernet, which was one of the most successful networking technologies developed. It was invented in the early 1970s. And the idea here is you have a shared bus, a shared wire, and many stations connect to it. And if two stations transmit at the same time, the two transmissions could collide and you don't actually receive the data. And what you would like to do is to figure out a way by which these different stations or nodes on the ethernet can somehow manage to communicate by collaboratively figuring out how to transmit on the medium. And the idea is to make it so that only one node transmits at any point in time.

Other examples of shared media are wireless networks. 802.11 is an example of a shared communication medium. If a bunch of us communicate and share that access point, we're all running on the same channel. And in 802.11, or Wi-Fi, there are a bunch of different channels available. But any given axis point tends to have a cell, which is some area around it that, in order to communicate with that access point, you use the same channel.

So we need a way by which we have to take the shared network and allocate access to this medium amongst these different nodes. And if you look at an entire building, there's a big plan in place for how the different access points communicate potentially on different channels or the same channel. And there's an entire process by which this building's network is laid out and the campus network is laid out and so forth.

And cellular networks-- you know, Verizon, AT&T, Sprint and T-Mobile and all these guys have the same problem that they have to solve. So these are very interesting questions which boil down to how you take a shared medium-- whether it be a wire like an ethernet net or radio, which is over the air, or it could be acoustic-- how do you take all of that and have these different nodes that are communicating with each other on that medium somehow make it so that they can all share the medium without colliding with each other?

And that's the problem that's solved by these media access or MAC protocols. Now, all of the stuff that I'm going to tell you is based on chapter 15, which is on MAC protocols. And there's a little more in it than we'll be able to cover in recitation, in lecture and recitation. But it's hard for me to keep straight what we cover and what we don't. So you're sort of responsible for everything in that chapter. We'll cover most of the issues.

There's a bunch of details in the chapter that are worth understanding in order for you to really get your understanding to be clear. So I want to caveat that upfront. We lost a lecture because of the hurricane. And I just can't keep straight what I tell you and what I don't. So everything in that chapter is fair game. But we'll cover the most important issues.

OK. So here's the abstract model. And you know, there's a shared medium. It could be a wire. It could be wireless. And you have these nodes. And the nodes have two things there that you have to worry about. The first thing here is just a detail. It's the channel interface. It's a way by which the data on that node gains access to the medium.

And then each node has a set of queues-- or one queue or two queues, a transmit queue and a receive queue, which has packets waiting to be sent on that medium. And the basic idea is that you have all these nodes on the medium. And it may be that these nodes are all trying to communicate with a single router or a switch or a satellite. Or it may be that the nodes are directly communicating with each other, like your laptop is communicating with mine. Your phone is communicating with your laptop. And somehow they're all sharing this medium.

And we're going to come up with the world's simplest shared medium network because it's simple and because it's a reasonable model of reality, which is that at any point in time if more than one transmission is on that shared medium, then you end up having what's called a collision. And you cannot decode the packet.

So let me repeat. The model here is if there's exactly one transmission on the medium, we'll assume that it's delivered correctly. If there is no transmission on the medium at some point in time, nothing happens. The channel is sort of wasted. It's not used. If there's more than one transmission on the medium overlapping in time, neither transmission successfully gets decoded. So there are two or three or four people hitting the channel at the same time overlapping in time. Nobody succeeds.

OK. That's the abstraction. And what we want is a communication protocol or rules of engagement between the nodes that allow us to get reasonably good performance in sharing the medium. Now, depending on the details of the network, the nodes that are on this shared medium may be able to hear each other perfectly. Or maybe they can't hear each other at all. Or maybe they can partially hear each other. And we're going to deal with all of these cases.

But for now, just assume that you have these nodes on the medium. And for simplicity and completeness, let's take the example of the satellite network. You have a satellite up here. And you have a bunch of ground stations that are all trying to communicate up to the satellite whenever they have data to send. And just assume for now that the nodes cannot hear each other.

And the satellite doesn't really know whether a node has packets waiting to be sent or not. And somehow, we're going to invent a protocol or a rules of engagement that allow these nodes to, in a distributed way, figure out when it's OK for them to transmit data and when they should keep their mouth shut. And each of these nodes has a queue of packets waiting to be sent. Sometimes the queue may be empty. Sometimes the queue may have one or more packets. That depends on what the application is doing and how quickly that node has been able to transmit data on the channel.

If the node has packets waiting to be sent, so a queue can either be empty or it has one or more packets waiting to be sent. In which case, we're going to use the word backlogged. So at any point in time, some subset of the nodes may have backlogged packets and that may have backlog queues. And our goal is to come up with a way by which these nodes can transmit data on the channel.

Now, there are two or three different ways of solving this problem. The first approach to solving this problem is to do frequency division multiplexing. You could just allocate different frequencies and you've solved the problem. But as I mentioned before, we don't want to do that because if you've allocated and predefined a frequency to a node and the node is empty, it doesn't have packets, then you've essentially wasted bandwidth because you've allocated a portion of the frequency to a node that isn't actually going to send any data. Somebody else who had data to send could have more profitably used it and helped you win, helped you get better performance.

The second thing you can do is to somehow divide up time. In other words, they all run on the same frequency. And somehow make it so that you give the illusion that each node gets access to the channel for some fraction of the time. And if you do that, it's a model of sharing called time sharing as opposed to frequency sharing. So one approach to dealing with it is frequency sharing, which is a good idea in some cases, but not in this case when the traffic is quite bursting.

The second thing you can do is time sharing. And there are two ways of doing time sharing. One of them is called time division multiplexing, or TDM. It's also called TDMA for Time Division Multiple Access. And we'll talk about this in recitation tomorrow, so I won't belabor it here. The second approach to solving this problem, which is what we'll spend the rest of today on, is the class of protocols called contention protocols.

And these protocols are really beautiful, because they're completely distributed. There's no central intelligence. It's highly distributed intelligence. Each node makes its own independent decisions as to what it should do based on very little information that it learns as it sends its packets and determines what happens to those packets.

And yet, somehow the nodes are able to come up with a way of sharing the channel by essentially cooperating but yet competing with each other. That's why these are called contention protocols. That ends up getting pretty good performance. There's a third kind of sharing, which we won't talk about at all in 602, and that's code division. The slides that are online say a little bit about it. And you can look it up on the internet, if you want. We're not going to talk about it here.

So for today, my goal is to tell you about contention protocols. And largely speaking, for MAC protocols right now in this chapter and this part of the course, we're interested in time sharing, TDMA and contention. So before I tell you about the protocols, I want to tell you a little bit about what we would like, what makes the protocol good and what doesn't, what's bad.

So if I tell you that you have a bunch of these nodes that are trying to share this medium and you would like to get good performance in this protocol to share access to the channel, by what metric or metrics would you measure this performance? How would you know it's good or bad? Yes?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Well, the model here is if two people send at the same time, you fail. We fail.

AUDIENCE: [INAUDIBLE].

PROFESSOR: OK. So good, let's keep going with that. Now let's say that someone observes the system and watches its evolution over time. They observe whether packets succeed and packets fail and so forth. And they want to count something. What would they count to determine if a protocol is good or bad?

AUDIENCE: Rate of failure?

PROFESSOR: Sorry?

AUDIENCE: Rate of failure?

PROFESSOR: Rate of failure. I mean, live your positive, people, so let's count the rate of success. OK, so there's a word for this rate of success. It's a measure of, if you succeed more, it means you're able to deliver data faster, which means you get higher throughput or a higher rate. So the first metric that you have-- so these are metrics. The first metric that you have is throughput.

And throughput is generally measured in bits per second or packets per second. So let's just imagine that it's packets per second for simplicity to assume that all the packets are the same size. So you can translate into bits per second. Now, throughput by itself is a good metric. But really, we would like protocols-- we would like to evaluate protocols without worrying about whether the underlying channel can send data at 1 megabit per second or 10 megabits per second or a gigabit per second or 100 gigabits per second.

I mean, we really don't want to care about what the actual throughput or the rate supportable by the channel is. So we're going to translate this throughput into a different word which means-- which is a proxy for throughput called the utilization. And we'll represent that by U.

And the utilization of a MAC protocol-- or in fact, of any protocol over a network-- is defined as the throughput that the protocol achieves divided by the maximum rate at which it's possible to send data over that channel or over that network. So if you have a sudden maximum rate at which if everything were in your favor you could send data at a certain maximum rate, stick that in the denominator. Look at what throughput you actually get. And take the ratio.

The higher the utilization, the higher the throughput. We've just normalized out by the maximum rate. And of course, by definition, we know that the utilization must be between 0 and 1 because you can't exceed the maximum. And this is an aggregate measure. So if you have, let's say, four nodes and the max-- just for example, let's say the max rate is 10 megabits per second and you have four nodes.

And the throughput that the four nodes get are, let's say, 1 megabits per second, 2 megabits per second, 4 megabits per second, and 1 megabit per second. What's the utilization? The total throughput is 8 megabits per second. The maximum is 10. So the utilization in this example is 0.8.

In fact, if you had-- for the same four nodes, if the throughput you got was 0, 1, 7, and 1, the utilization would also be 0.8. Can't add. Can't multiply. But can convolve. All right. Now, which of these two is better? That depends on if you're a Democrat or a Republican. But which of those two is better?

If you were designing a network, which of these would you want?

AUDIENCE: [INAUDIBLE]

PROFESSOR: What?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, a Democrat. This is a tough question. You might say that everybody should-- they should be as equal as possible. You might say that they should get proportion to what they pay for. There's various ways of thinking about this. We're not going to worry about the-- I mean, it's actually a pretty deep set of topics that connect with economics and social justice and so on.

And a lot of people do work on what it means to have fairness and what it means-- different kinds of fairness. And for those who are interested, I can point you to lots of literature. And it's still somewhat open in terms of, what is the right way to think about it?

We're simple people. We'll just assume that absent any other information, we would like to get as equal as possible. And to do that, there are many ways to define fairness. And there's a particular one that I like because it's simple and somewhat intuitive. And it's used a lot in networking. It's called the Fairness Index.

This isn't the only way to measure fairness. But this is one that we'll use. And the definition of it is that I will look at either the utilization or the throughput. It doesn't matter. Let me call that XI. In other words, XI is the throughput or the utilization-- let me call it throughput-- achieved by node I.

And if I look at this number here, XI squared divided by n summation XI squared, that's my definition of fairness. Now, this looks a little daunting. But it's actually very simple. What it's saying is that I take each of the throughputs that I get and I add them all up and I square them.

So if I were to divide both sides by n squared-- essentially, this is capturing for me something that's related to the ratio of the square of the mean to the variance. So in other words, what ends up happening with a term like this-- this is a second moment kind of a definition of fairness-- this thing here looks like the square of the mean. This thing here is related to the variance. And this is capturing the ratio of those two terms.

If you have a situation where you have n nodes and you end up with everybody getting equal, then the fairness index is 1, because if everybody has an equal value and you just run the calculation through, you'll find that the answer is equal to 1. So if this is F, F is between 0 and 1.

What's the minimum value of the Fairness Index from this formula? Well, that happens when you have n guys and the throughput looks something like this. One of them gets all the throughput and the others get 0. And if you plug that in here, what you'll find is that only one term survives, the 1 over n term, and everything else becomes 0. And so the minimum value of the fairness is 1 over F.

And this is intuitive in that it says that if you have two people and one guy hogs everything, that's a little bit less unfair than if you have five people and one guy hogs everything, because in a sense, one guy hogging everything out of five is worse than one guy hogging everything out of two.

The only real reason we care about this fairness is we're going to compare different variants of a protocol along this index. And I'd like you to get a feel for what is a little fairer and what's less fair. There's nothing particularly sacred about this particular definition of fairness. And indeed, people will argue also that this is a terrible definition of fairness because it doesn't reflect how much people have paid for the resource.

But those are details because you could have weighted versions where people are weighted by how much they pay, and so forth. So you can handle some of that stuff. Is this kind of clear, these two basic definitions? So we're going to worry about throughput and fairness in protocols. Any questions so far? Yes?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Depends on whether you care about it or not. I mean, it depends on the application. So typically speaking, when I look at, for example, measuring the performance of my-- let's say I'm downloading web pages. When I measure the performance of web downloads, all header information is just pure overhead. So I only look at how long it's taken to download my content.

But now I can go in and look at how well is my TCP, which is the transport protocol, or IP, or whatever, some lower level protocol working, in which case, everything else is overhead. But I will include the particular headers related to TCP in my measurement. So the answer to whether something is overhead or not is it sort of depends on what it is you care about.

So if I'm delivering video, you know, all this other stuff is overhead. And the only thing I care about is my video frame. So typically, the word throughput is by itself not meaningful. You have to say throughput of what. And here I'm talking about throughput of a protocol, which is what you always have to say. And this is the throughput of the MAC protocol. And it does include the MAC header overhead if you have any headers.

Any other question? OK. There's a third metric that's important, which is that we would like to have reasonable delays. So in general, delay is a good metric, or bounded delay is a good metric. And this matters because I can get extremely high utilization and extremely high fairness by doing something utterly dumb and naive, which is that if I have all these nodes with a lot of packets, if they're all backlogged, then what I will say is, today this node gets access to the network. Tomorrow, he gets access to the network. Day after tomorrow, he gets access to the network.

If they're always backlogged, then clearly the throughput is-- the utilization is very high, because the network is always being used profitably and there are no collisions and no idle slots, no idle time. The network is fair, because if I measure this over a month or a year or something, everybody gets equal throughput. But I've clobbered the delay because this guy got lucky. But everybody else is waiting and waiting.

And in fact, even he has to wait. Once today is over, he's got to wait many days before he gets a turn. So you actually would like to have bounded delay, or at least low delay. And this is something we're going to measure. We're not going to try to optimize in the work we're going to talk about. OK. So that's the statement of the problem and the statement of the metrics that we wish to optimize. You have to ask me questions because if the problem setup wasn't clear, the solution is kind of going to be completely meaningless. So does anyone have any questions about the problem statement?

It's one of these things where stating the problem is actually a little harder than the actual solution. So I'm going to tell you now one method to solve this problem. I'll get to you. And the method we're going to use to solve this problem was invented in the context of satellite networks and then it got put into ethernet and then it's now part of Wi-Fi. So everybody uses a variant of the method that we are going to study. Yes?

AUDIENCE: [INAUDIBLE] how do you measure delay?

PROFESSOR: The delay is measured per packet. It's measured between when the packets showed up at the queue to when it actually successfully got received at the receiver. And then typically we'll take an average across all of the packets and report an average delay, and perhaps the standard deviation. You had a comment?

All right. So the solution we're going to study-- the basic solution is something called ALOHA. ALOHA was the protocol developed by a group led by Norm Abramson, who was a professor at the time he did this at the University of Hawaii. I believe he moved from Stanford to Hawaii because he was an avid surfer. And he decided that-- this was in the late '60s-- that he wanted a scheme to connect the different islands together. There were seven of these islands-- seven of these stations that he wanted to connect together.

And he came up with a scheme that on the face of it should really not work. And only think good about it is how utterly simple it is. And the fact that it works is actually very fortunate and very useful. And the reason it works is because, in a way, nodes doing things that look completely random-- as long as the probability with which they do these things is controlled, it turns out they work pretty well.

So let me first show you a picture that will define a few terms. And I'm going to come up with a version of this ALOHA protocol that ends up being a pretty popular version. And it's called slotted ALOHA. And this model that I had from before where the nodes have queues, and when two packets run at the same time they end up colliding, all of that remains. I'm going to add one or two more restrictions to the kinds of-- to the model.

The first-- an important thing that defines slotted ALOHA is that-- and in fact, it defines real implementations of this kind of protocol-- is that time is slotted. What that means is that you cannot send a packet at an arbitrary point in time onto the network. Instead, what ends up happening is that if time-- you view time as a continuous-- as a continuous variable, you divide up time into time slots.

I mean, these slots could be any length. It doesn't matter. And the assumption we're going to make is that packets can only get transmitted at the beginning of a time slot. So these are legal. Let me-- this is a legal packet transmission. And this is a legal packet transmission. But this is not a legal packet transmission. Not allowed.

And the second assumption is that every packet is an integer number of time slots. So in other words, this is legal. But this is not legal. You cannot have a packet that starts here and ends in the middle of there. All packets are an integer number of time slots. OK.

So if I have both of those assumptions, that tells me ALOHA. By slotted ALOHA, we're also going to make the additional assumption that each packet is exactly one time slot long. Later on, we'll relax this assumption. But I want to come up with the world's simplest working protocol. In other words, the only legal packets in slotted ALOHA are like that. OK.

And as shown on this picture, this is a picture of how slotted ALOHA works. You have time going on that axis. Time's divided into slots. And I have these three different nodes-- blue, red, and green. When no node sends packets in a time slot, that time slot is said to be idle and the channel is said to be idle in that time slot.

If you have more than one node sent in a time slot, we have a collision. And in our model, none of the packets that sent in that time slot gets decoded. All of them are wasted. And everything else is a success. If you have a time slot in which exactly one packet is sent, we'll assume in this model that the packet is successfully decoded and we get to count that as a successful packet reception.

So if you count and look in this picture, the utilization here is 65% because we have 20 time slots here. And in 13 of those time slots, we were able to successfully transmit exactly one packet each. And that gives us the utilization of 65%. And the advantage of picking many of these things is you can't really check in real time if I got the numbers right or wrong. You can check that later on.

I'm pretty sure-- I'm not sure of anything. It's probably correct. You should just count the number of slots in which exactly one guy is sent. OK. So that's the picture here. So what I want to do now is to come up with an algorithm, with a protocol, that each of the nodes can implement that allows us to get reasonable utilization, reasonable fairness, and reasonable delay.

And an example of this protocol, and one way of solving this problem, is to solve the problem in this context under these assumptions. And then we're going to calculate the utilization of that protocol. So let me start by telling you what the protocol is.

The protocol-- each node independently runs a version of this protocol. And in the protocol, each node maintains-- in the simplest version of the protocol, each node maintains one variable. And the variable it maintains is a probability. So each node maintains a variable. I'm going to call it p. And p is the probability with which the node will transmit a packet if it has a packet to transmit.

In other words, each node has its own variant, its own version of p. And the semantics of this is that if backlogged, we won't just greedily go ahead and transmit. But instead, if we're backlogged, we will transmit our packet on the channel with probability p.

How do you generate-- how do you actually do something with probability p? Like, what would you do to do something-- if I were asking you, write a program to transmit a packet with probability p, how will you actually do that? Yes?

AUDIENCE: Call a human to roll a dice for you.

PROFESSOR: Call a human to roll a dice. All right. Let's try to make it a little more practical. Actually, a dice has only got six sides. How will I get p out of a dice?

AUDIENCE: A lot of dice.

PROFESSOR: A lot of dice. OK. What if I want p with 10 to the minus 17?

AUDIENCE: A lot of [INAUDIBLE]?

PROFESSOR: Yeah, that's-- all right. Does someone have a slightly more practical solution? Yes?

AUDIENCE: Could you have a random number between 0 and 1 [INAUDIBLE]?

PROFESSOR: That sounds good. So you pick a random number between 0 and 1. I mean, how do you get that? Well, that's a deep question. But I would just call random.random in Python, or whatever the thing is. And you know, how Python does it is very-- there's ways to botch it. But for our purposes, we'll assume it's correct. And if the number you get is less than or equal to p, that tells you an event with probability p.

That's great. OK. So suppose we did this. I'm not telling you how to pick p yet. That's magic that's going to come up a little bit later. But suppose every node had a value of p, that someone came and told it, you know what? There are n nodes in the system. Let's assume there are n backlog nodes in the system. And I told you that somebody came and told each node that it can transmit with p. What is the utilization of this protocol?

So if I have n backlog nodes, each transmitting with this probability p, what is the utilization of the protocol? I want to know what is the utilization, which is, of course, a function of n and p? And the way you have to answer this question is, of course, you draw this picture in time and you divide time into slots. You've got some of these things where you have one packet going through, some of these things where more than one goes through, in which case, this is a collision.

And I ask you, what is the utilization? How would you calculate it? Suppose you observe the running of the protocol and you find that in some time slots there's a collision and in some time slots there's nothing. And in some time slots, there's exactly one transmission.

If you look at this, how would you calculate the utilization of the protocol? The utilization is the throughput over the maximum rate. What's the maximum rate of this channel? Well, the rate is measured in-- the rate here is measured in packets per time slot. And I've said that each packet occupies one time slot. So the maximum rate is every time slot is occupied with a packet. So the maximum rate is one packet per time slot. You cannot send faster than that.

So the denominator is just one. Therefore, the utilization in this model is simply equal to the throughput, which is simply equal to the number of time slots in which I have exactly one packet. Or put another way, if I look for a long enough amount of time, it's the number of time slots with exactly one packet that tells me the throughput, and therefore, tells me the utilization.

So if I look for a very, very long time and I count the number of successful packets, that's going to tell me the utilization if I take the number and divide by the number of time slots. And therefore, the utilization is equal to simply the fraction of time slots in which I have one packet and exactly one packet sent, which is equivalent to asking, what's the probability that in any given time slot, I have exactly one successful-- one exactly one packet sent?

So I'm going to repeat this. The utilization of this protocol is exactly equal to the probability that in any given time slot, I have exactly one transmission. If you disagree with that statement, you have to raise your hand now, or if you don't understand it, so I can explain it again, because we're going to use this idea repeatedly in pretty much-- there's guaranteed to be some question on the quiz related to this idea. And then you have to work some probability out.

So does everyone understand why the utilization of the protocol is equal to the probability that in any time slot there's exactly one transmission of a packet? The reason why that's true is it follows from this definition because the maximum rate here is one packet per time slot. And so I want to know what the throughput is.

The throughput is simply, I look over a long period of time, many time slots, and I count what's the number of packets I sent. So if I take the number of successful packets I sent and divide by the number of time slots, well, that's actually the definition of a probability that in any given time slot I have exactly one successful transmission. And therefore, the utilization is equal to the probability that I have exactly one transmission.

So I have n backlog nodes. Each guy sends with probability p in a time slot. What's the probability that I have exactly one transmission?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Well, there's certainly a p, which is success, times 1 minus p to the n minus 1, which is all the other guys keep quiet. Is this right? It's almost right.

AUDIENCE: And then times n because there's n ways to [INAUDIBLE].

PROFESSOR: Times n. There's an n choose one, which is there are n ways to pick the winner, and therefore that's the answer. All right. Let's hold n times-- so let me write this as u equals n times p times 1 minus p to the n minus 1. Suppose you knew n. Suppose n were some value-- 10, 15, whatever. Therefore, I would view this-- assuming n is constant, I could view this as a function of p.

If I want to maximize this utilization, I want to-- obviously I want to maximize it, right? If I want to maximize the utilization, what should be p be? Or I'll call it p star. What's the value of p equal to p star that maximizes this utilization? Yes?

AUDIENCE: 1 over n.

PROFESSOR: 1 over n. Yeah, you could do this the hard way or the easy way.

AUDIENCE: Differentiating [INAUDIBLE].

PROFESSOR: Great. So if you did that, the answer works out to be p star-- if you do u prime of p equals 0 and you solve that, you'll find that p star is 1 over n. The long way to do it is the way you describe. But the answer is intuitive because what it really says is that, if I did have every node transmit with probability 1 over n in a time slot, the expected number of transmissions within any given time slot is n times 1 over n, which is 1, which is kind of what you would expect.

You would expect that to maximize the utilization, the expected number of nodes that transmit at any given time slot is 1. And that's fortunately what you do get if you solve the equation. So p star is 1 over n. So if somebody told you the number of backlogged nodes in the network and they had the ability to program each node to set the appropriate probability, the probability you would use is 1 over n.

So let's assume still this world where we know the number of backlog nodes. And somebody came and told us the probability we should use. Do I need anything here? Let me erase this. If somebody came and told you the probability you should use, you would pick 1 over n. That's great.

So now therefore I can now view u star of n, which is the maximum probability now becomes a function of n, because I can go back in here and I can set-- assuming you picked this value of p, you can stick p equal to 1 over n in this formula and then you get a utilization that's purely a function of n because I want to draw this picture of what it looks like with n.

So you start off n is equal to n times 1 over n, which is the value of p that's the best value of p, times 1 minus 1 over n to the n minus 1. Does that makes sense? Haven't done anything other than substitute a value of p star equal to 1 over n. And that's equal to 1 minus 1 over n to the power n minus 1.

Now, one of the questions we want to ask is, is this how-- this is the best you can do in this protocol. How good or how bad is it? So let's draw a picture of what this looks like as n becomes large. So I want to draw a picture of n on this axis and the best value of the utilization, which is u star of n on this axis.

Well, when n is 1, get to n is 1-- actually, intuitively, when n is 1, what's the utilization? When you have one backlog node and the protocol runs with the value of p equal to 1 over n, what's the utilization? The utilization is 1. This formula, you got to take the limit, and so on. But the answer is 1. So let's assume this is 1 and 2 and 3 and 4, and so on.

So it's 1. What is it when n equals 2? What's the utilization when n equals 2? Well, it's 1/2 to the power 1, which is 50%. So I get a value which is 1/2. When n is 3, what happens? Well, 1 minus 1/3 squared, which is 4 over 9. As n goes bigger and bigger and bigger, what does this value become? Yeah?

AUDIENCE: [INAUDIBLE].

PROFESSOR: Yeah, as n goes bigger and bigger and bigger, you do the limit when n goes to infinity of this thing here. n minus 1 and n are more or less the same thing, so you can get rid of that. This should be a well-known limit. If you don't know it, you take the log. You take the log and you find the limit as it goes to infinity. You can expand that into a power series. And you'll find that the answer-- the limit of the log is minus 1 or this value, the limit goes to 1 over e.

So in fact, it goes to a value which is 1 over e when n is large, or about 37%. This is actually not bad. It's actually very good. For a protocol that did nothing sophisticated, all it did was pick a value of this probability. The fact that it's able to get not a zero utilization but a reasonably good utilization is an extremely strong-- is a pretty strong result.

And that's the basic ALOHA protocol. The basic ALOHA protocol, or a fixed probability ALOHA protocol, is somebody telling you the number of backlogged nodes and you using that information to make sure that every node sends with some probability. And they just-- and the probability you would pick is 1 over n.

Now, this is not actually a very practical protocol, because how do you know which nodes have backlogged packets and which nodes don't? What we would like us to come up with a way by which we automatically have a protocol that somehow gets us to the correct utilization. And we're going to do that by adding a method to this ALOHA protocol that will make it completely practical. And that method is called stabilization.

And the purpose of stabilization is to determine at each node what the actual value of p it should use is. And that value of p is going to change with time as other nodes have traffic to send and if nodes go away. So the magic protocol is going to be somehow that we're able to change the value of p at every node. Every node runs an algorithm which adapts its value of p over time. And if there's a lot of competition, the value of p will reduce. If there's very little competition, the value of p increases. And if we do that, we're going to be able to get pretty good utilization. And this process is called stabilization.

AUDIENCE: Are we still assuming that all of the nodes are transmitting with the same probability?

PROFESSOR: Nope. The nodes will end up not transmitting with the same probability because they're each going to be making independent decisions. So the first thing we're going to do is each node is still going to have a p. But in fact, node i is going to have its own variable. So we're going to say that node i has its own variable that only it knows. And its probability is p i.

So the way we're going to do the stabilization is very, very simple. We're going to say that at node i, it's going to do-- well, the only information it gets is it sends a packet. And if the packet succeeds, it knows something. If the packet doesn't succeed, it knows something.

So let's say that a node sends a packet and the packet fails. And the way you know that a packet fails-- and I haven't talked about this. But the way this protocol-- all these protocols-- end up working is they send a packet and then they watch to see if the packet succeeds or not. They can get that information by an acknowledgment coming from the receiver.

Or in the case of certain networks, like ethernet, when you send a packet, if you aren't able to receive your own packet on that bus, then you know that it's failed. So that's just a detail. But the assumption here is there's some feedback that tells the node whether a packet transmission succeeded or not. In general, it's with an acknowledgment that comes from the receiver. If you get an ACK, it means it succeeds.

So we're going to have two rules. If you don't succeed-- in other words, there's a collision-- then you do something. And in contrast, if you succeed, you do something. So what we're going to do on a collision is, let's say that you send a packet and it didn't succeed. It collided. Yes?

AUDIENCE: So with the acknowledgement, what if the acknowledgement fails?

PROFESSOR: Yeah. You're out of luck, because in reality, in Wi-Fi, when an acknowledgment fails, what ends up happening is you assume the packet collided. So how people deal with this problem is typically to-- essentially to do very strong error protection on the acknowledgment. You send it at a very low bitrate. And what that really means is you're adding a lot of redundancy to that acknowledgment. So imagine you're coding the heck out of it using channel coding. That's what happens.

OK. Let's say you send a packet and it collides. What should you do to the node's transmission probability? Increase it or reduce it?

AUDIENCE: Reduce it.

PROFESSOR: Sorry?

AUDIENCE: Reduce it.

PROFESSOR: Reduce it, because the assumption is it collided because presumably there's a lot of competition and you're a nice guy. And your assumption is that all the other nodes are nice people, too, which is kind of changing nowadays with all these software defined radios and the hacking that you can do on Wi-Fi cards. It's possible for your node to not actually back off.

But if you're a nice node, what you're going to do is you're going to reduce the probability. And one way of doing that, a good way of doing that, is called multiplicative reduction, or multiplicative decrease. You reduce it by a factor of 2. You just halve it. And on a success, you could do a bunch of different things. But one thing you can do is you can be a little bit greedy and say, all right. I succeeded, which means there's not that much competition in the network. Maybe there's nobody else in the network, in which case, I want to keep trying to increase my transmission probability. And maybe you double it.

And my notes talk about whether this is right or something else is right. But it really doesn't matter. It turns out the important rule is this rule. The protocol turns out to be not very sensitive to how you increase. You do have to increase. But it doesn't matter how you do it.

Now, of course, the probabilities can't exceed 1. So we're going to actually pick the minimum of 2 times p and 1. This is our basic stabilization protocol. Now, every node has its own version of p. And so you may run with a p of 0.8 and I might be at a p of 0.1. Presumably, if I succeed, I'm going to increase. If you fail, you're going to decrease. And something happens. The question is, what happens?

And that's what I want to show you. I'm going to skip all the stuff we went through. So if I run this protocol exactly as I've described on this board here-- and this is an experiment with-- you'll be doing all this stuff in the lab yourself. This is with 10 nodes.

What you see is that you have a utilization of 0.33, which is not too far from the 1 over e utilization, which is remarkable that this protocol where everybody just jumped around multiplicatively changing p i like this worked out to be a pretty good utilization.

But there's a big problem in the protocol. And the problem in the protocol is that the fairness is pretty terrible. As it happens, it's 0.47. I was reminded that 47% of the nodes here got pretty healthy throughput. It's probably looking for handouts or something. But anyway, it's pretty bad.

And what's going on here is that when a node is running at a high value of its probability and some other node is at a low value of the probability, if they collide, the guy with the higher value of its probability reduces some by a factor of 2, but it's still a pretty high value, whereas if a node has a probability of transmission, somehow it's got screwed and it's now running at 1 over 32, it becomes 1 over 64, then 1 over 128. It's practically 0. It doesn't ever get out of that real morass that it's in and ever start being able to successfully transmit again.

And that's what's happening here. And the way you solve this problem-- there's a very simple solution to this problem. The way you solve this problem is you decide that nobody should get really, really poor, that you decide that you're going to have a value of the probability p min that will never actually get-- you'll never go below that. So you modify the protocol to do that.

And if you do that, you end up with much better performance. You end up with performance that looks like this. And you'll see this in the lab. But what you find here is something very puzzling. What you find here is that the fairness is amazing. It's 99.99, which is really, really good fairness. But what you find is that the utilization is 0.71. That's actually too good to be true, because the best utilization you could expect is probably around 37%.

And the fact is, we're getting something astonishingly good. What's happening here actually is something that's really important. It took people a few years to figure it out. It's called the capture effect. What's happening here is that some node captures the channel for quite a substantial period of time, shutting everybody else out. And then some other node captures the channel for some period of time, shutting everybody else out.

So you get significant short term unfairness. Or equivalently, you get a long delay. So some nodes may end up waiting for a long time. And then they get access. And then they keep access for quite a while. And then they lose access. And then some other nodes get it.

So what's really going on here is, even though you have many, many nodes competing, at any point in time, effectively the competition is only between one or two nodes. And this is a problem called the capture effect. And the way you solve this problem is symmetrically to change this value of the probability, there's a maximum value, which is p max. So you have to pick a different value that's less than 1 that's the maximum probability.

In other words, once a node has a transmission probability of, let's say, 0.25 or 0.3, you don't want to have it keep increasing, because what ends up happening is then it captures the channel for quite a while. And it's only upon successive collisions that it comes down to the point where other nodes can gain access to the channel. So if you put all of that together and run the experiment, we're just putting this protocol as described exactly on this board.

And you can play around with this thing. There's a lot of leeway in picking these parameters. If you run this protocol, you get a utilization, in this case, of 0.41, which for n equal to 10 is pretty much what you would get according to sticking into that formula a fairness that's extremely high. And it's super cool because this is exactly what you would want to get. If somebody magically told you the number of backlogged nodes and you theoretically calculated the optimal value of the probability, you couldn't do better than this protocol.

And we managed to do it by simply these nodes that are just independently making these decisions, figuring out what they should do. And if you were to plot the actual evolution of the probabilities, you find that at no point in time does any node have a value of 1 over n. Even though in the experiment there's some value of the number of backlogged nodes, the nodes kind of dance around it. But they never actually stick to it. But they conspire to dance around it in a way that gives you exactly the same result as you would get if you, in fact, had a pretty good thing.

I'm going to close with one last comment. This is from s student from about, I think, two terms ago, a guy called Chase Lambert, who took this class. And I was very gratified to hear this, because it turned out he interned or he is working or worked or working at a company called Quizlet, which is a startup company. And one of the things he had to do was a load generator. And he ended up using exactly these ideas to come up with a load generating scheme that was stable, because you want to be able to generate load that allows you to measure the throughput of a system. And he used this random back off idea.

So it's not that you'll find these ideas useful only if you were doing this kind of networking. You actually find these ideas useful in other contexts. So I'm going to stop here. We'll pick up on some of these topics in recitation, and then back here on Monday.