00:00 We'll get the message. Okay. We are, we are live on the cloud. So yeah I'm still letting people in, but this is Zha County.
00:11 We're doing our distributed systems reading group. This is number two for 2023 on this paper called Distributed Reset by Anish Aurora and Mohamed gda.
00:23 I actually got this paper or I came across this paper. Does anybody here read Miro Murro Dumont Duma Probably gonna say his same wrong Mi Murra's post.
00:34 He's at aws. Paul, I know Paul, obviously he does Miro works, right? Miro works at aws. Yeah, so he, he, but he had a couple papers in I would say the 20 19, 20 20 about like audit auditability and high performance systems and around reset and coordination.
00:53 So this paper is like, was 1990 obviously influenced things, but I've seen it a lot in papers like as a reference for papers coming out in 20 19, 20 20, and, and a lot of distributed algorithm and, and, and system literature.
01:10 So I got it from his bog post, which if I can make sure I have it here. Yeah, I have it here.
01:16 He has a really good foundational distributed system, systems, papers post doesn't really go into a lot, all the detail and stuff like that in terms of like why every one of them but on this post, but he does talk, you know, obviously there's a lot of good stuff here.
01:34 We've read some of these as part of the distributed systems reading group previously last year. You know, there's some, there's some classics on here, so if you get the, a chance to look at that.
01:45 I mean you know, I was around at, at bachelor. I mean, there was the, or soon after where there was, there is no, there is no, now Justin Sheehy won, which is really good.
01:54 I mean, there's a lot of good stuff in here. You can spend a whole, you could have a whole year of reading stuff just from that post.
02:00 But this came from that because actually I had not, I had not seen this one, and I thought it would be a really cool a really cool paper for us to read and kind of go into and think about especially how to, how to think about distributed reset for arbitrary systems.
02:13 I think maybe some people here will bring about languages that might have these concepts, but, and, and and run times, but maybe not, you know, in this kind of, you know, attempt at a generic way.
02:24 So I leave it open to somebody in the audience I've, than letting more people in to give a to give a little introduction about this paper.
02:36 Any volunteers? Come on, <laugh>. I'll, I'll just, we go, I can't really summarize it, but is it kind of like a layer on top of the existing distributor system with what waves and trying to maintain kinda attractive state and taking out cycles?
03:16 Let's see. So I'm getting stuck now, so, No, no, I, I mean, it is you know, I think a big key in this paper is yeah, it's a layer to work a subsystem, you could say, to work with you know, arbitrary distributed systems.

03:31 That's a good, that's a good thing to find. I think people maybe what I was kind of referencing earlier, maybe people who had worked in languages with run times and stuff like er Lang and stuff might be used to some of these concepts.
03:41 Obviously it's message passing, and they talk about that briefly toward the end of this paper where you have some of these paradigms kind of baked in obviously certain distributed systems that are out there have some of these paradigms baked in, but it's not, you know, kind of all, you know, it's kind, it's, it's essentially kind of a hard in the way they kind of reference it.
03:59 So that's a good, that's a good point. There's some interesting stuff there about how they deal with the spanning tree, I think as, as Brendan's getting at spanning Tree is all things, right?
04:07 Spanning Tree is the like definitive <laugh> piece of everything kind of running, running, all networking. Anyone else wanna kind of give a a a little bit of intro?
04:17 That's pretty good. That's a good setup. I think it is a, a huge part of this paper compared to other systems, I can give it a shot.
04:27 So the paper starts with the idea that you've got some sort of distributed system, but it makes very little assumptions about it and says, if, let's say that the problem you want to solve is that at a given moment, you want to bring all of the notes of that distributed system to a predetermined known state.
04:51 So that state has to be known when you set it up or when you first program it. And so you wanna bring all of the notes back to that state.
04:59 So this is not consensus because you're not proposing a value during runtime. Instead what you're, it, it's like consensus in that everybody passes through a state.
05:14 But unlike consensus, you can't propose a value. You have to already know what the value is, so you bring everybody to that value.
05:22 And then this is an algorithm for solving that problem. And the algorithm is to first construct a spanning tree with a single route so that you can sh distribute the, it's time to reset message, tell the nodes.
05:41 So I think that's interesting because you don't, if it's a distributed system, there's probably some way for the notes to communicate with each other already, but this paper doesn't want to assume that that exists or that it's working at any particular moment.
05:55 So instead it creates its own protocol. And then on top, once that spanning tree is established you kind of bubble the, it's time to reset message up to the root, the roots sends, okay, let's reset as a message down through the tree to all of the leaves, and that all of the leaves felt acknowledgement back up to the root.
06:22 And once that's done then you know that this, that every note has passed through the reset state. There's no one moment where everybody's guaranteed to actually be in the resetted state, but you know that since everybody passed through that state that any subsequent states of the system are as if everybody had once at one moment been resetted.

06:45 And then I think the final key detail I would add is that you can have, I think, up to two resets going on simultaneously, but no more and the way that's, yeah, That's kinda an interesting question to this paper actually.
07:03 How many could you have simultaneously even with their session index? Yeah. Yeah. I'm not clear on that, so I'd like to hear more.
07:13 But whatever the guarantee is, it's enforced by saying that notes can't talk unless they have the same session number and a reset wave causes the session number to increment.
07:29 And the session number late in the paper, they say the session number could just be zero or one, which makes me think that there can only be two going on simultaneously, but I'm right, right.
07:39 I'm foggy If you want, like, I, I think it's in particular for like a bounded and it's like, in a sense you're probably you're bounding it purposely for that, or essentially like in a way to like just, you know declare that in the system.
07:54 It's a good question though, if that actually works for kind of real systems or practical systems, you know, that we, that we actually think about.
08:01 So I think it's, that's a great question to that. Yeah, I, so that's a great run through on this. We talked about, you know, there's trees, there's waves, these things kind of go back and forth on how they're, how they're se how messages are sent, how things are done.
08:14 And then you have this idea of sessions as I, I think I have in this part right here. Or you have sessions in this piece to actually kind of distinguish these waves that are coming through.
08:22 And I think that's a good question. If there's, how many can you actually have simultaneously? So that's really good. I think the other like, little thing I'll bring up too is like how, you know, how to interface in the application with this.
08:34 You know, there are a couple modifications that the application has to make, but essentially the be able, the ability to be able to say like, I need to do a reset is one and the ability to acknowledge that, that a reset is occurring.
08:46 And so have that and have that say. So there's only, there's like those small pieces, which I think they talk about importantly, like, you don't have to do very much to your application compared to, you know, possibly other things where you have to like, involve a whole you know, some sort of shim or, you know, another module that kind a whole interfacing that you have to do, which I think is pretty interesting.
09:07 So that's a great, it's a great roundup. I think the one part I'll highlight to you before kind of open it up too is, you know, this idea of distributed reset they talk a little bit at the end of the paper about how this could also kind of inform or be attached to air error, state error handling and itself being a former distributed reset, but also these things right here to like reconfiguration mode change, coordination, loss, and periodic maintenance.
09:31 Of course upgrade seems like an another case where these kind of things are so, you know, upgrading versions is always the the hell of a conservative system.

09:40 And kind of part of these kind of things like what does it mean to actually also be reset? Brendan, please.
09:46 Yeah, just wanted to add totally and add to what you're saying that in relation to the how many resets can be happening at once also not, doesn't take on the net split problem.
09:56 Alright. And it's, I think it's worth noting that net splits are not not covered in this. I just think, think it's worth bringing up as like a important characteristic that the the paper just Yeah.
10:07 Pushes to the side at the, at the beginning says if you have a net split, we're not resets gonna be Yeah.
10:14 Good luck, <laugh>. Yeah, I thought a lot too about things around like you know, obviously very early in the paper they cover the, you know, what is it, you know, to have their agency you know, graph of nos that are up, right?
10:27 And obviously that the whole premise for the rest of the paper is the definition of understanding what is the agency knows and like which ones are up, right?
10:33 But up is wonky, right? You know, so I think a lot about flas in certain systems. So if you have a constantly flapping system, you know, what kind of happens here over time in this kind of model that's kind of trying to constantly reset, send you know, essentially send waves down and then back up.
10:50 And once you're getting all this flabbiness, what happens? Like if you got constant churn this way you know I, I used to work with a coworker, Scott Ritchie, who had a paper one of the airline workshops about, you know, this thing about humming consensus and how do you, how do you, how do you come to, how do you kind of understand things at a higher level around the flas?
11:07 So that's a, that's another one that comes into play and that's splits another one too. Anyone else kind of thoughts?
11:14 Some of the motivations on this paper you know, maybe even some linking to what we think about what do we think about distributed reset today in, in relation to this at a high level?
11:35 Well, I would love to talk about what we're doing. We just we're building a system you know, that really does solve this problem in a very different way.
11:44 But we also use distributed trees. And we do have a i e e paper published about four years ago. But most of the new work is with related to a D project that we're working on, sorry, a d i e project we're working on which is about resilience and robust computing.
12:03 But that's probably not a good thing to talk about today because it will distract us from what I wanna listen to about this <laugh>.
12:09 Yeah. <laugh>. No, I mean I think you know I think a big thing obviously to this point in this paper, I mean, really the kind of crux, and for me this motivation is like, how do you, you know and using systems today reset is a big problem.
12:30 So actually maybe Jesse's actually asking Paul, maybe you could tell us or a link to the work. Yeah. So you can put that in the chat, actually.
12:37 That would be really good. Yeah, certainly. Yeah, we, the, we don't get our

answer back from the DOE for another 10 to 12 days.
12:45 So I'll, I'll put the information up on our website after that. Before that. Something that's really quite important here is someone who is an expert on this particular problem, and I believe also on this paper is Edward Lee professor of computer science at, at uc, Berkeley.
13:05 And Edley has done some really wonderful work on actors versus reactors and about this problems of ordering and about these relaxation events this thing that you just called, I think humming consensus and so on.
13:18 Edward's a pretty good expert on all of these things, and we have him on the clubhouse clubroom it's about time on Saturday at 9:00 AM Pacific time.
13:26 Oh, cool. Very cool. Yeah, the, the, the, the like pod actually, it's like, like it's a podcast essentially, right? Yeah.
13:34 Very cool. Yeah, to reach a link to the Everly Everly's work is really good. I, yeah, I remember, I've actually read, what is it Computing needs time, right?
13:41 That's yeah, that's like a dissertation reports, yeah. Back in 2009. Very cool. Yeah, I mean you know I think when I, I was kind of coming back to this paper, I would look, get people's thoughts, even with practical System today, obviously having used a bunch of systems this idea of, again, yeah, of reset you know you know how, for me, I actually like designing these kinds of things.
14:14 I've always felt like I've had design this, even if I'm working with an existing system, I have to design this concept myself or something on top of this, right?
14:24 That it's not typically as, as baked in as I would walk as I was like, right. Obviously there's some stuff you can do and if you're using orchestration, you know, people might think in Kubernetes land and stuff like that, you might get some of this with, and there's things like affinities and all this stuff that you can leverage.
14:39 But you know, I think there's kind of an understanding of the orchestration of a system like this that's kind of baked in.
14:46 Does anybody have kind of any thoughts about that? I mean, obviously we can talk about too, if you felt like a paper kind of is too generalized or, or has you know, has maybe some impractical ideas I would like to hear people have thoughts on, on that.
15:02 You know or, or any good stories really about self stabilizing systems that, you know, you've been involved with. I mean, what do people think when they think about self stabilization today?
15:12 That would be actually really interesting. Brenda, please. Yeah. I mean, I just want to state my own bias and then reaction.
15:19 Like my bias is towards peer-to-peer systems, not necessarily distributed systems, right? And like, I find this work like highly in inapplicable <laugh> to the, the peer-to-peer context because I mean the definition, the, the leader election can is going to create a natural hotspot.
15:35 The sort of just natural state of the whole system and the notion that we would reset to some state is just really, really challenging in that context, right?
15:44 And so it's, And imagine like thousands of pe you know, you start saying,

yeah, what is the tree of a thousand?
15:48 Like, Creating a root spinning tree is, yeah. And then like any kind of faithfulness where we're having, like, we're passing through what is effectively like three network-wide types of tex locks is like just a very it's expensive <laugh>, it's a really expensive process.
16:02 And so I generally sort of, what I find interesting to recognize here is, is the, and the other thing that that sort of jumps out is paper seems to like, in the construction of the rooted spinning tree, have to, at bare minimum estimate the si number of nodes in the network, right?
16:16 Like you, you had had this moment where you have like a k value that needs to exceeded to detect cycles in the That's right in the spinning tree.
16:22 Yeah. A cycle detection. Yeah, that's right. Yeah. And so, like, you know, as soon as you have that, it's like, okay, cool.
16:27 Like you have a system that has likely a monotonic counter built into the application layer, you'll likely have a, you need, you have a need for some sort of heartbeat.
16:35 Okay, cool. That's, that's that. But I find in uncontroversial and then, but then just this addition of like a sort of three phased wave and a rooted spanning tree construction places a lot of that's a lot of labor in terms of anology.
16:49 Yeah, that's right. That, that's that really I'd rather not pay that overhead ever <laugh> and, and try to design the whole system to not need to arrive at any type of global state whatsoever favoring the local law, local knowledge approach.
17:02 Yes. Yes. And so like, I, I think it's really fun. This is like a fun chance to like go and look at a different neck of the woods when it comes to distributed systems where you have smaller numbers and you can do more interesting stuff because you generally, it seems like you would have full control over the system, right?
17:20 Like where you, you have a, this is operating in a space where the, the peers are known entities and, and yeah, That, yeah.
17:27 That's actually a great point. Or even in the trip, this space, like could this work in a large kind of multi-region, multi-tenant way, or is it more like each region operates distributed reset within itself this way, but can, you know, if you wanna have more like a mesh topology for this stuff, which we've seen larger distributed compute work with, obviously, again, thinking about locale and where things are, I mean, that puts a lot into play here if, like, if, you know, what does it mean for like a wave to happen <laugh>?
17:55 Like how many processes, right? You know, I think, yeah I think that that brings up a lot of complication that, you know, the paper that doesn't really dive into as much, right.
18:07 For sure. That I agree with completely. Anyone else? Jesse, Yeah, this is just a details question that might then be relevant to that larger question of when is this applicable?
18:24 But the Hispanic tree at any given moment could have disconnected

subnets or cycles, and if somebody starts a reset, well, the tree is poorly constructed that way.
18:43 How does the system eventually get fixed? Right? <laugh>, How, how does that like reset message not end up caught in a loop or only distributed to a subnet?
18:57 How do, how do we manage the concurrency of constructing the tree and diffusing messages through the tree? Right? I mean, so they talk I mean, I think this is a, it's a really good question.
19:10 They do talk about I forget what page is on, actually. They do have a part in the paper where they talk, try to talk about, like, I think, what is it here?
19:19 I think it's in, it's in the like, later part of the paper, right? I guess city stuff, I forget. It's like somewhere where they try to talk, talk about the possible like stages that you might be in that they have to, they have to try to fix things.
19:37 But that's a good question. In a sense. I don't think there's like a really a firm kind of grasp on like what that, like what would happen in each of these kind of, there's no like case study scenarios, right?
19:46 Which is almost kind of what you wanna have in this papers for them to take the, the motivations from earlier and then apply here's how, here's how this would happen in each of these different results that might occur.
19:59 So that's a good question cuz somebody else will ask, right? What is an appropriate initial state? And I think there is this kind of thing of like, when do we get, when we are in a good state to then have this process?
20:07 Now the other thing they kind of talk about, which is interesting, part of this paper, which I highlighted, which is the goal in terms of also for their, for their proofs, right?
20:14 Is that they don't want to do any kind of global freezing. So you might not be in an initial state, you might be at some arbitrary state, right?
20:21 Which we talk about, which is actually the you know, I would say is showcases like a, a a view towards something very complex to, to work for any arbitrary state, but what, you know, what happens if things are not in a good state?
20:33 When do you get back? Right? That's a, that's a great question. I don't, you know, in my mind that's <laugh>, you know I don't know.
20:41 I would get people's thoughts on that. I have a couple ideas there, but any thoughts that that that point that Jesse brought up?
20:58 So there's a, there's a, a major potential strategy here to solving these problems. Whenever you run into these kinds of of errors, an interesting question is, can you use the notion of reversibility to undo what went wrong?
21:15 In other words, to make time go backwards. Now I'm talking about making time go backwards logically here. But our sense would be an underused strategy in this, in computer science, and this is exactly how we solved this particular problem which we thought about five years ago maybe or longer.
21:32 And on spanning trees, by the way. And the idea is that when you run into this notion of two things have been symmetric, competing with each other, one

of the interesting challenges is can you identify one of them as Bob and one of them as Alices and do some kind of symmetry breaking protocol, which makes one of them back down.
21:52 In other words, undo what they did and makes, allows the other one to continue their wave. So that's just a, a thought, Right.
22:01 Are, are there concepts like certain waves of certain resets certain initiated resets have priority over other other ones, right. Can you determine that?
22:10 Based on anything logistically? I was actually looking at a paper in relation to this about actually resetting vector clocks, which is an interesting kind of thing as well in terms of like, when can you actually do something like that and be safe to know that in the system?
22:24 Okay. Obviously that's is a big, you know, having worked in that is a big reason to you what people have done that in the past was to get rid of obviously a lot of the space consumption that the, that that it can take for these things.
22:34 But like, when can we define a state where, yeah, okay, we can, we can kind of start over by not starting over, but knowing that we're all in a good place, right?
22:42 And but also like yeah you mentioned a great one too, like how you, when can things be definitely reversible? Right?
22:49 That's, that's a great peak Point. So the this issue of resetting vector clock, I'm very familiar with that paper. I'm not gonna tell you where I, I found this out, but it's in a large infrastructure that I was involved in.
23:03 We actually found out a case where vector clocks had actually been corrupted. We dunno whether it was a memory or on the, on the network.
23:13 But there was an actual corruption where they thought they had a large enough vector clock, 32 bits or, or, or 64 bits, I think it was, I think it was, it, it was having the problem a lot at 32 bit vector lamps, but when you used 64 bit vector lengths, everyone said, you know, well that's, that's so long that, you know, it's, we're beyond, it's 10 times or a hundred times the length of time our computers are up before we replace and, and we replace them the computers every two years.
23:41 But that's, that's not the point. What we discovered is this corruption put the the vector clock so close to the rollover point that that they, they rolled over and it locked her our entire system.
23:54 This happened twice over in two years. And for the same reason. And so obviously we went hunting for all sorts of different ways to do aero detection and neuro correction on those kind of errors.
24:05 But it just goes to show you that these are major issues and in, and in fact the, the recent word from Facebook and and Google about about cause being unreliable, giving different results, right?
24:16 Right. Cause of of semiconductor values, for example potentially, potentially might have been one of the causes. So Vector Clark's particularly long ones are actually very, very dangerous.
24:27 The longer they get, the more dangerous they are. Yeah. That's for sure as

a warning. Yeah. And so, I mean, and then, and this, yeah, and, and, and I think the tie in right is like this this determination of reset and the, and the priority of reset.
24:42 I mean, this comes in the play I think all the time. I, I also, when I, when I read this paper, it made me think a lot about I was a big fan you know, when I was working more day to day on distributed system stuff of the, the dry ad link stuff.
24:58 Also kind of like my fascination with functional programming and working with databases. But also like all the papers around, you know Microsoft's link or Dryad.
25:10 I mean, the dry ad work was really, really cool. And they had a really cool debugging, like distributed debugging system with with Dryad and you know, these kind of concepts too of like the ability to debug something.
25:24 You know, it, if you have a better understanding of debug ability into the system, could that help? And especially if you can, if you can automate that, can that help with self stabilization and understanding which waves take priority.
25:36 There's a lot in there as well that I think that can be kind of cross referenced from, from work that's happened there and kind of distributed debugging land.
25:44 That's, that's been really cool. Does yeah, so any anyone else kind of on that thread? Oh, so actually Lawrence brought in a question here.
25:56 Yeah. Implications for distributed systems, social systems, organizational resets handling hostile takeovers. Yeah, I mean, I mean yeah, I think this paper opens up a lot of obviously being, you know, again, a little bit of its time opens up a lot of these questions.
26:14 Anyone else with some thoughts? Maybe more on some of the motivation stuff before, or unless we want to get more into like, specific pieces of it or, or, you know, other, other maybe things that you don't feel like are, are, are fully in there.
26:27 I mean, they talk about implementation issues in the paper, but maybe not what people have thought had in mind when they were <laugh> they were thinking about other implementation problems.
26:44 Jessica, please. Yeah, I just, I assumed there were passing messages between notes, but the paper is so abstract <laugh>, I just wanna point that, that out that I expected at some point to say notes pass messages to each other, but what instead we got was that they variables like N I J And those variables make up the entire state, right?
27:15 Yeah, that's right. Yeah. They do talk about message passing toward the end, right? Like, I think that's what they're and channels essentially, right?
27:21 That's, I think, obviously I think in the modern era of how we think about those things, we would, we would be thinking much more in this direction today.
27:30 Right? For sure. And you're right, I mean, this is, I also, I think, I think a thing of the PA papers in this era they are kind of highly abstract, right?
27:41 You know, a systems paper in this vein, if you look at the stuff that actually

references this paper where you're gonna get if they're more related to like sitcom or some of the other distributed algorithm conferences, you're gonna get a full evaluation of the system, right?
27:56 You're gonna get a lot more <laugh> that kind of deep dive into like, how did this work? How did this implementation go?
28:03 Where did it, where did it falter? Here's how we're better than x, y, Z system, right? Yeah. That, that's, that's all not here, <laugh>, right?
28:13 So Another random thought I had, whether or not this is the right time to bring this up, but resetting everybody to a distinguished state is kind of an application of this protocol, but this protocol could be used for a number of other things because it seems to me that what the protocol is really doing is getting everybody to increment their session numbers.
28:35 Yeah. In the same order maybe up to two of those at a time. And then you could decide how many session numbers you want as long as there are at least two or three of them, I think.
28:48 So you could imagine you could use that to make everybody reset to a known state. But I think that they could reset to any state, which is a function of the session number.
29:00 So session number five could be different from session number four, and that could be an interesting thing. It could be like phases of a distributed workflow or something.
29:11 Yes. Your point. That's interesting. Right, right, right, right, right. Yeah, and, and you're thinking about that like as a means for how to actually like have it where if you know these known sessions, if you have these known sessions that you can go to, like not everybody has to do it at the same times, like eventually for like optimizations, right?
29:32 That that could actually take place for this, right. And piecemeal. Yeah. Yeah, that's a, that's an interesting thing too. And I, yeah, I guess, I mean, Paul, Paul's kind of brought up like obviously building, building stuff now you know, with this, with this, this, this thinking in mind.
29:51 But I kind of want every, anybody on this call, like, who has built, you know, when have you been, when have you encountered trying to do this kind of massive state, you know, people have worked with snapshotting, going back to certain snapshots of state, like I kind of wonder, you know, where people have kind of seen like kind of had that practical motivations for thinking in this way.
30:11 And to Jesse's point, I think it brings up a lot of interesting optimization points for how you might actually do this.
30:16 Distributed snapshot has been, has had a problem with time ever since the whole concept was invented. And and the thing that is most familiar here, for those of you who have some connection with this is, this is a, is what's called a superposition in quantum mechanics.
30:33 And the mathematics is absolutely identical. The concepts are identical but sometimes the computer science community doesn't pay much attention to the quantum community and the, and the, the, the, the understanding doesn't

translate very well because the, the terminology is Different.
30:48 Right. Yeah, that's a great point. Now, the, if you're interested in this, the the best work that's been done on this recently has been done by the Warfarin Physics project, where they talk about using multi-way systems, which is their mathematical equivalent of, of several things happening at the same time.
31:05 One of them has to, you know, be the, be selected that's called a multiway system and the and the wolf from rewriting system.
31:14 Yeah. Actually, this brings up, I mean, obviously re like rewriting systems term rewriting systems, obviously, like they, like, like Wolfrem is brings up some interesting things in terms of how you, like, would actually, you know, possibly implement ideas like this.
31:27 Right. And obviously that that's something they take advantage of. It was something I learned about Wilson stuff in particular, right.
31:33 Obviously getting into pl stuff like how, how that works under the hood is actually pretty cool. Yeah, I think that's a, that's a good point.
31:43 Time and snapshots have been in conflict for a long time. Anyone else but kind of, kind of avenues this way that they, that, that they, that they thought about while reading this.
32:06 Okay. <laugh> before I kind of maybe talk about a few other points and that Paul made a lot of really good points kind of toward the conclusions here I, I highlighted a few things.
32:24 You know, actually I thought there's some interesting stuff here about that they bring up other issues for like an efficient me.
32:30 So actually they, they bring up, you know, they're, they're obviously thinking about their whole thing about, you know, to, to Jesse's point about like in how they deal with indexes and processes.
32:37 They talk about, you know efficiency there. But they also bring up something too, which I think is really, really insightful.
32:43 A third issue is security problems involving an any application process to reset the system, right? Because obviously in this one here, an application says, cool, I wanna initiate a reset.
32:53 Now you've, you've created essentially a wave, right? Literally of things that might occur and, and point where things could be really complicated, maybe multiple ways.
33:01 And obviously you have to have all this optimization that we're talking about maybe in place. How do we prevent certain things or certain applications from, do applications have hierarchy here or do applications have delegation capabilities?
33:14 Like what are, what are kind of ways that we might think of reset in that with that lens? These are really important points.
33:24 The, the, the, one of the things the, that I really like about some of the distributed systems algorithms is when they pre-select their take their fill over partner.
33:33 And this is an old idea that goes back to tandem and process pers, you

know, two or four decades ago.
33:39 And and, and if you can pre-select who can take over for you if you are not if you fail is a much more powerful technique than randomly selecting someone right across a, a switch network.
33:52 Because you do, you have no idea how far away they are because all you know is their a dress, you don't know their how many hops away they are, for example.
33:58 And the more hops you have, the more likely <laugh> they are likely to fail. So you don't have the topology information, you really can't select your, your partner.
34:06 That's right. That's right. And then how much do you have it where, I mean, this is also a thing that kind of comes up.
34:12 Cause the people who program more again and ang of stuff, you can't actually kind of create your tree of supervision, but it's on the programmer to kind of think about the meeting of that.
34:22 As well, when you think about some of these things, like what how much can you kind of leverage what you have in the, in the network topology and information on the fly to kind of make, make no no certain choices.
34:31 Like this is definitely a local node that can, that can, that's local to this, you know, group or something like this.
34:38 Yeah. Any other, any other thoughts on these kind of security implications? I know in the PTP space, right, obviously Brendan's on here, like, we think about that a lot.
34:48 You know, and, and you don't want bad peers, right? And, and, and peers, you know, peers to do bad things to others.
34:56 You know, obviously a reset system like this, like, you know, by putting, by, put by, by attempting to put a lot of, a lot of this paper is about being lax about restrictions, I think, which is a really interesting concept.
35:08 They talk a lot about that. They're trying to be more lax than other you know, obviously systems that say, Hey, we're gonna stop everything and, and do that.
35:17 And I think even, even though they didn't kind of tackle security in this paper, what would be a lax way of doing this versus something, you know, very op you know, kind of oppressing in terms of like, well, you know, only these valid actors can ever reset stuff, so everybody else can't do any, you know, can't do anything and have to maybe be maybe, you know be thrown off or whatever the case might be.
35:42 Any, any thoughts of that from people here? And I guess while we're doing that, people are thinking Brandon, you had a question right?
35:53 By pre-selection, I think Paul's talking about pre-selection. Maybe maybe Paul can answer that. Yeah, go ahead. Sorry Paul, I don't think we heard you on the, I'm sorry, I interrupted the other person.
36:13 I apologize. Yeah, <laugh>, we're having a, a Canadian standoff. This is great <laugh>. My question was just like, how did Paul, just to get some intuition

of your pre-selection, is it pre-selection prior to joining the network?
36:25 Like do you know the nodes in advance or is it pre-selection in the sense that I joined and then secretly elect Okay, based on, you know, I pick the nth peer that I come in contact with and they're my failover buddy, and maybe I don't tell anybody that.
36:38 That's a really good question. So it really, the, the answer depends upon what your definition of knowledge is. And for that you have to go deeper and say, well, what is the definition of information?
36:48 And the, and the definition of information according to, to Shannon is the answer to a yes, no question. And so that <laugh> begs the question, what is knowledge and what is information?
36:58 So you have to ask a question of your neighbor and on, if you're going through a switch, you, you, you, you have complete uncertainty about how many people are beyond that switch, how big the network is, how many nos there are out there.
37:09 And so you really can't go past one, one hop, which is basically the hop between you and the switch to ask a question.
37:16 And the switch unfortunately doesn't keep this kind of information or knowledge because by definition, switches don't keep, stay on behalf of applications.
37:24 And so it's much better in a mesh network where in a mesh network you connected directly to your neighbors and you can know whether someone is one hop away or two hops away, or three hops away because it's built into your spanning tree and Invent to the overlay.
37:38 Yeah. Right, right. Exactly. Yeah. So the does the answer to your question is you, if, if you know the topology and you know someone's directly connected, then you can select them to be your failover partner, you know, and of course you'd have to assume that, that they're on a different power supply burst rail and that kind of thing.
37:56 But in general, that's not a difficult problem. And and if you just select someone who is most likely to be able to hear you when everything else is failing, then that's a good choice.
38:08 So the answer to your question is, we need overlay networks. This is brilliant. That's like the best answer I could have gotten.
38:15 I have a new justification for why overlay networks are a good idea. Yeah. <laugh>, thank you. I really appreciate your answer, Paul <laugh>.
38:20 And I think it's that point that the networking stuff is so important. When I was at Comcast, right, we were looking at this problem of pathfinding and source and source source finding.
38:31 And the way we did is we used like concepts like this thing called segment routing. Actually the information in that state would be packed into the packet compressed into essentially a, a small array of all the possible destinations for a certain service.
38:44 You would know that ahead of time, clump that into the, yeah, pop that

into the packet, and then the packet would basically go to the switches and be like, cool, I'm ready to like B processed by, you know, note A in this region.
38:56 That's, that's a disservice. And it's like, oh, A is busy right now, or A is down. I, I already know where B is.
39:01 I'm just gonna go to B, right? And then once I go to B and have a, and, and once you say, okay, now I've had, I actually have like a connection here, and now we're gonna send a bunch of packets back and forth.
39:10 You can then skip the first time you have to do the whole round trip, but the second time you do it, you don't have to do any direct re direct.
39:17 We could just do direct server return, right? So that's, that was like a thing we built at Comcast for like this kind of route finding work.
39:24 But we actually put that information in the packet cuz and then once we learned information about the network, the packet can, we can update things that switches about that information and put it back into the packet.
39:34 And we had semantics use V6 for this. This is the key. The key was V6 and the you know, which is still coming, right?
39:41 Still happening. But we could put messages at the end of the bits in the V6 address to do things to tell another node about like storing state or not for a connection.
39:54 So we actually use like message passing in the V6 addresses. The problem is to run stuff like, like certain things, when you start getting into overlay, start getting the networking, you, there's a lot of cool tricks you can do, but you get into things where like certain certain places will look at, you're trying to do, like you're trying to do spoofing and stuff of destination addresses or something like that.
40:14 So you could be a bad actor. If you have a controlled network, you can get away with this. If you have, if you're trying to run this in like a, a gen generalized way, then you'll run this problem.
40:22 So I think, yeah, I always say it like leveraging the network more and more and, and the things that it has and like trying to build on top of that instead of building something else and trying to fit them together actually.
40:33 Yeah. Yeah, it's a lot. There's a lot there. Right. I think only in the last decade have you seen more networking and distributed systems stuff kind of really start getting closer together?
40:45 I would say, I dunno if Paul agrees with that more generally, but I, yeah. People obviously network and systems were always intertwined, but I feel like it's been, it's been a little better in like, even closer in the last like decade.
41:03 No, you're you're absolutely correct and that's the right way to think about it. One of the big challenges is, is the more you started packing into a message, the more you know, the longer the packets get and the worse your latency.
41:13 The and so one of the interesting questions is if is if you were to be given a clean sheet of paper and all you had was an ethernet frame which is you know, what you have, if you were talking between peers on a, on a mesh network then you could actually do a whole mo lot more.

41:33 You could actually, you know, basically get rid of source destination addresses, which just get gigantically larger when you Yeah, I pb six.
41:41 And because not only do you have the MAC addresses, you also have the IP addresses, and now they're a lot, a lot larger.
41:46 These are great resources to use in interesting ways, but if you use them in a, in a legacy, everything has to be compatibles.
41:54 The, the first day of, of, of ethernet, you know then you're just gonna make the messages. Yeah. If you have to make it work for everybody, it becomes a bit too hard.
42:03 <laugh>, if you have to make it work for a certain subset of things Yeah. Then it works better, right? Yeah.
42:08 So This, this idea of doing it the end of an, of a V6 address we've thought about using some of the V6 address base to actually do some, I wouldn't call it message passing cause we really don't like to think the applications get to touch that.
42:23 We call it distributed metadata, in which case our specs inspired in dis epic restricted metadata is is actually packed and, and, and exchanged in the I P V six frames.
42:36 And then we realized that we can make the frames a lot smaller if we if we just go straight to the ethernet and get rid of the source destination addresses and, and use a, just a single tree ID, for example with that address space.
42:50 Yeah, that's actually a great idea. And, and there there was some tricks you know N S C I I think some years ago that Fastly was doing with some of their routing techniques to, to work directly on the frame information for their routers.
43:06 So yeah, you know, I mean, to that whole point, I think a lot of these kinds of ideas of thinking about optimization and how to like, deal with state and information, can I actually cut be leveraged more and more by what we have in the networking already in the networking space.
43:21 And that's where you, you said you can, you, you have a lot more information than trying to kind of build something.
43:27 So someone had a question here, I think goes, Lawrence, could you hash the topological space and use that as the routing method?
43:35 That way you can direct a packet based on the topological shape of the roots. You maybe Lawrence, you wanna talk a little bit more about that question?
43:49 I get a higher sense of what you're asking about. Sorry, it's maybe a, maybe an naive question. I was just really curious.
43:58 So I, sorry, you, you're blowing my mind in terms of like things to think about, but the idea would be like so if you've, so the, I I really like the idea of embedding, of looking at information as an embedding problem and looking at things around you as ways of like hallmark basically things, characteristics about the environment around you.
44:18 They can then encode into a, a small packet of information and you can use that, that as a way of transferring that knowledge around.
44:25 So it's kinda like landmarks and going through places to place, like look for

the, the red stop sign, look for the, the green tree, look for the, the blue house.
44:35 In a topological, in a map, in a network scenario the, the, the characteristics that you have per node is a topological map of your, of your interconnects for everybody in that node.
44:47 So one way to, to route, so you can make the assumption that everybody on a particular node can, can look at least one hop out for what the topological map of each of the different routes are.
44:58 And if they're unique enough, you could use that as a way to to map a, assuming that the routes are, are consistent enough that you can map the, the pathway between one place to another based on the shape of the topological map.
45:15 So you, so essentially the packet wouldn't have to keep a track of each individual node, it just has to know roughly the, the flow of, of information from, from space to space.
45:29 And, and the, the packet router would essentially look at each would have a, basically a kind of a map of where, where you want to go versus where you want to be, and the map not according to like a, a pre-described route, but the shape of the but the shape of the, of the, of the node that you're going towards.
45:49 So I, what I'm imagining is that you'd have, like, let's say you've got a a, a, you've got a, a heavily connected node and at least lightly connected node.
45:58 You could, I think it'd be easy. I think, I think where the idea falls down is that if you were trying to track between a a highly connected space and a lightly connected space, you'd have to have uni enough uniqueness between both spaces to be able to do it.
46:13 So you might have to encode more information about each individual node and what the properties are, which then gets you back to the, why don't you just apply a a Mac address to it.
46:25 So I think I might might advanced my own Question. Yeah, yeah. I mean No, that's good. I think that you, yeah, right.
46:31 I mean the other thing is like stuff we did or, you know if you, if you're building down to the switch level which is what we did like a Comcast where we, we built a software switch we could essentially now then participate in B G P.
46:47 And so we would know when roots were coming in and out at the, at that level. Cause we would also get all the messaging that's coming from the network, right?
46:53 And so obviously, you know, I think to, we talk about networking distributed systems. I think the other kind of cool thing happening is there's actually a closer mix now between network networking and programming languages and software engineering.
47:08 So you and you seem like languages like you know, P four has shown this with its work, but a lot of people who work on P four now, Nate Foster and others I'm forgetting the one big paper, the Python, the Python networking people language, David Walker, it says like David Walker, Nate Foster, and all these cats and others have done a lot and, and P four has an interesting language that

works on works, really works on like, is a pretty robust language to write actual, like programs in that run.
47:39 And if you have, you know, very if you have switch hardware switches that can actually run P four, you can get a lot of a lot of performance.
47:48 But the, you know, there's a lot of interesting things about how to work at that level and program with that.
47:53 Like, almost like we're programming today in the web world, right? It's actually gotten so much better than it used to be where it was like kind of just, you know only people who knew networks could do networking things and it's all configuration based and there's no language and no DSLs and, and that kind of thing.
48:09 Paul was kinda, Paul were you, you were, I think you were saying it, that's also, it's something you're seeing as well.
48:15 I've seen this for two generations of war network engineers. Yes. Yeah. And that's what we're addressing too. Now. The, besides sending, you know, things like, you know, E B P F packets or fragments, you know to do P four down into the NACO or into the switch another alternative to say, well, why don't you just send fragments of very log down into the, to the F PGAs and It's huge Fragments and, and, and bypass an entire tool chain.
48:46 Yeah. And, and the f PGA thing is huge, right? I mean that's you know even obvious was saying, I know larger companies I was at talk Microsoft, you know, just there's boards and hoards of huge F P G A you know setups.
49:02 And I think, yeah, I mean, and the program ability, there's a lot of work coming outta at Cornell. And there was just recently the, the as plus conference, I think just happened and there's a lot of interesting work with how to program f PGAs and, you know, do verification and, and much more.
49:19 Cause actually the hard part, and a lot of this is writing verlo for a lot of people and like, how do you make that easier?
49:24 And there's been some really good stories there. I think Jane Street actually has hired the person who wrote like the OCaml you know, OCaml tooling kind of compiler for, for Verlo.
49:39 And they're doing that for obviously high frequency trading work and they're running fpg. So you're, you're, you know, you're obviously seeing where the, where the, these kind of cool these kind of combinations are happening, which I think is really awesome.
49:54 Cool. We're almost about time. Did anyone have anything else before I kind of wrap up here on the call? Cool.
50:04 We got a lot of links in there. The Smart Nick Summit's a great one. Really cool col run through there.
50:10 Lawrence too at the end. And yeah, Brendan put his, like, his link to his videos. They're amazing. I've been saying this to everybody I work with at Fission.
50:17 I actually think more people should just be like, I read a, actually something I should, we should do in like my PhD reading group is like, you know, read a

page, like actually do a video of it.
50:25 And <laugh>, I think it actually, it makes you learn a lot too, right? Get, you know presenting on papers and, And, and please, if, if anyone has a paper they think should be videoed I'd, I'd be delighted to for any leads.
50:40 They take a long time to make <laugh> understand the paper thoroughly enough. It's the whole phenomenon of staying one lesson ahead of the student.
50:47 But <laugh> Yeah, for sure. B five. I could, I'd love to, to help on some stuff we're doing. Can I dunno your email address?
50:56 So who are you, what's your name? Oh my, on the internet, I'm B five. I'm Brendan O'Brien and MeetSpace I'm this, oh yeah, we're just talking so I'm assuming some robot will transcribe this, but I'm B five N zero.computer is my email.
51:11 If you wanna email me, be delighted. A chat though. N computer Yeah. Dot com or Computer.computer. Yeah. Nice. Great. <laugh>, they opened up.computer as a tld When we feel this as an underutilized resource <laugh>, Well, your Earlier question, your earlier question was really interesting about being interested in overlays.
51:36 Whether you call them overlays or underlays or whatever else is, you know, a product issue. But the, there was some really great work that's done by the University of Waterloo.
51:44 If you're a Canadian by Samir El Al Kwani and his group. And, and that's that was called partial network, partitioning.
51:51 I think I mentioned this in our last session a month ago. Mm-hmm. <affirmative>. Mm-hmm. <affirmative>. Ok. That's awesome. Yeah. I'd have love to have a look.
51:58 Let's thank you and thank you Sean. Hey, we're here to, I think as Lauren, I think Lauren said it like we're just here to the, the whole group is about to blow people's minds and find, you know, also that, but papers and things, what you end up finding is just more and more references.
52:12 Like, it, it, you know, we are the spanning tree of <laugh> of more, more things to read. Right? Never ending worlds of practice and, and reading.
52:23 No. Cool. Thank you to everybody. Before I go the one thing and alright, on video FIMAN is running, we're running like a future computing conference along with ard and Protocol Labs.
52:38 A sponsor it's called causal Islands. We'll be in Toronto in April. We just put, got our first like eight speakers up there.
52:51 And there's gonna, we have like four or five more getting their bios and stuff like that. So it'll be really cool.
52:58 So if anyone can make to Toronto there's tickets and, and it'll be a lot of cool speakers. Maggie Appleton Jen Schiffer we'll have Evan Brady, Ramey Nash for a lot of cool people from all, all different backgrounds and topics kind of in the space.
53:17 Cool. That's all I got. I will stop recording now. So thank you for those on the recording, watching this.