EPISODE 556

[INTRODUCTION]

[0:00:00.3] JM: Before I begin today, we have an announcement. We have launched Software Daily, which is a place to post your software projects and discuss them with other people. On Software Daily, you can find collaborators and feedback for your software project. If you have an open source application, or a side project that you've been tinkering with, or an academic computer science paper that you want to get feedback on then come to softwaredaily.com and post your project.

Software daily is about cool projects and new ideas and creativity and discussion. If your project is especially interesting, we will send you a Software Engineering Daily hoodie, or a t-shirt, or even have you on the podcast to discuss what you're building. I've been posting some of my own side projects on Software Daily and I'd love to see what you are working on. Come to softwaredaily.com and discuss your ideas with the other people in our community.

With that let's get on to today's topic, which is the interplanetary file system. The interplanetary file system is otherwise known as IPFS and it's a decentralized global peer-to-peer file system. IPFS combines ideas from BitTorrent, Git and Bitcoin creating a new way to store and access objects across the internet.

When you access an object on almost any website, you are accessing the object via a location address, a URL. The URL tells you where to find the object. If the object is a photo on Facebook that you're linking to, the URL will have an address of somewhere on Facebook that links to that photo. Other objects that we access through URLs include webpages, videos and JavaScript import packages.

URLs seem natural to us. You look up an object based on where that object is being stored. Why would you do anything differently? A downside of location addressing is that if the location that you're addressing disappears, you can no longer access that object at that location. If a government decides to censor a website that I want to visit, the government can shut down access to the server where that website originally sits and then my link will break.

This is what happened in Turkey where Wikipedia was shut down last year. Objects in IPFS are content addressed. You access an object by giving IPFS a cryptographic hash of the object and IPFS will find someone on the network who has a copy of that object and give you access to it.

To look up a web page in an IPFS browser, you put the content address in the address bar. When the HTML for that page is received, that page might have lots of other content addressed files referred to on the page. Your browser can also grab all of those content addressed files from the IPFS P2P network.

Similarly to how you would just put a URL into your internet browser in your normal internet usage today, you're just putting in a content addressed location. Your browsing experience is going to be somewhat similar if you just start with that content address instead of starting with the URL in the traditional web format. We'll explain this content addressing again, but I think it's a very important concept to understand. I think it's pretty elegant and interesting once you wrap your mind around it.

In this episode, David Dias explains how IPFS is designed David is an engineer at Protocol Labs, which is the company building out IPFS and the related suite of technologies. This episode is a great companion to our previous show with Juan Benet, the creator of IPFS.

I hope you enjoy the episode.

[SPONSOR MESSAGE]

[0:04:05.6] JM: Users have come to expect real-time. They crave alerts that their payment is received. They crave little cars zooming around on the map. They crave locking their doors at home when they’re not at home. There is no need to reinvent the wheel when it comes to making your app real-time.

PubNub makes it simple, enabling you to build immersive and interactive experiences on the web, on mobile phones, embedded into hardware and any other device connected to the internet. With powerful APIs and a robust global infrastructure, you can stream geo-location data. You can send chat messages, you can turn on sprinklers, or you can rock your baby's crib when they start crying. PubNub literally powers IoT cribs. 70 SDKs for web, mobile, IoT and more means that you can start streaming data in real-time without a ton of compatibility headaches. No need to build your own SDKs from scratch.

Lastly, PubNub includes a ton of other real-time features beyond real-time messaging, like presence for online or offline detection and access manager to thwart trolls and hackers. Go to pubnub.com/sedaily to get started. They offer a generous Sandbox to you that's free forever until your app takes off that is. Pubnub.com/sedaily, that's P-U-B-N-U-B.com/sedaily.

Thank you PubNub for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:05:48.4] JM: David Dias is an engineer with Protocol Labs. David, welcome to Software Engineering Daily.

[0:05:53.0] DD: Thank you so much for having me. I have been a fan for a very long time.

[0:05:56.8] JM: Okay, that's awesome. Well, you may have heard our first show with Juan Benet actually then about IPFS and Protocol Labs and this was around 2015 when Ethereum and IPFS were both getting off the ground. This was around the time when you started working at Protocol Labs as well. How did you get involved in IPFS?

[0:06:20.7] DD: That is correct. In 2015 was the year that we released the alpha version of the interplanetary file system. That was in March 2015. I joined the team to build the JavaScript implementation around June that year. This happened right after I finished my master program from the University of Lisbon. It was a masters in peer-to-peer and ubiquitous computing.

Essentially for the previous three years I was fascinated by mesh networking and something that is known as community clouds, where instead of you hiring resources from a cloud provider, from a company, you actually establish a network where the users, the peers can provide

voluntarily their resources and then resource can be CPU time, can be a search engine, can be storage, whatever they have available to the network so that other users benefit from it.
That interest essentially allowed me to then focus on a project that was converted into my master thesis. I call it at the time browser called JS. It was a web platform job scheduler. I was using JavaScript to write tasks, any task that was plausible and then I was using WebRTC to establish connections to other browser nodes and through an overlay network. Think like a routing network that enables you to find other nodes in the network.

I would find browsers that have idle resources; so a browser that is like physically still not doing anything, to then request those idle resources to run my tasks to run my computations. At the time, the web platform totally made sense; I had to move a lot of data and a lot of computation and a lot of computation to where the data was located. I eventually learned about IPFS and I was blown away by the mission and its ambitious goals.

It quickly dawned on me that's a lot of the issues that IPFS was solving at the time were things that I had to solve myself to build the system that I wanted to build. I started contributing to the project, and couple months after I did the responsibility of implementing the JavaScript version so it could run on the browser.

[0:08:37.1] JM: With this idea of we all have our computers with some spare compute cycles, some spare storage space, some spare memory, why don't we pool those resources and we each get rewarded for what we're contributing to the pool and people can purchase those resources off of the network and do interesting things with them? That idea has been around for a very long time. Why hasn't that idea worked out in the past?

[0:09:06.1] DD: It's true. It has came in multiple iterations, multiple waves. For example, there is a very old system mojo nation where you’d have a mojo token that would let you recruit resources from other peers in the network. Although the system at the time, well essentially the platforms that were available at the time, the availability and the interest for that system was not present and it was not as easy to do this exchange of value, this exchange of resources through the network as it is today.

Even back in 2015, the world was still learning about cryptocurrencies and the ability to have a peer-to-peer network that enables you to exchange value through the network itself without any other intermediary. Although the community clouds have been also available for a long time, people still didn't figure out a way to properly charge for the consumption of those resources. Today it's different. We keep seeing projects popping up all the time that enable this mediation, this exchange of resources through a token and those results can be – CPU time can be storage, can be multiple types of services that is used in a network.

[0:10:24.3] JM: Okay, well let's talk about IPFS in some detail. In order to get us there, I want to start with some high-level examples. Let's say I want to build a website, a website is just an HTML page with a single CAT picture on it and that CAT picture is on IPFS. IPFS is the interplanetary file system. It's a global distributed file system that everybody has access to. In a normal website, if I just had an HTML page with an image tag in the center of it and the CAT picture URL that's hosted on S3, most people know what would happen. I load that web page and the image gets fetched from Amazon S3 and it loads in my page. What about if the image address was requesting the picture from IPFS?

[0:11:16.9] DD: There is two ways for that to happen. IPFS is a peer-to-peer file system. It enables you to fetch data from a network independently of its location, but in order to provide that service you can use directly the IPFS protocol, or you can use an HTTP gateway that is available for any user to use so that they can use the familiar HTTP protocol to fetch content from IPFS. Then let other nodes do the work of finding that content and retrieving the content to you.

On that example of the web page with a cat picture, if you are using relative links, it will work in the same way essentially like the user will not even see that the cat picture as in the IPFS link, because in IPFS you can have folders just in any like other unique system. if you link to the cryptographic fingerprint what we call an hash of the file, the user will see that they have a string that instead of being just the name of the file, it is this proof that once the file is requested and fetch it, the user can then do the same verification to execute the same hash function over the file and verify that it received the right bits.

Once this request, it's an IP fast gated for example the IPFS Network will like the IPFS now real requests the network and ask any node in the network for the location of that file, connect to that node, fetch the file, verify it’s the right thing and retrieve it to the user giving the same experience of the user before, yeah.

[0:12:54.3] JM: I think you mentioned – Okay so there's two ways of accessing that image and one is if I have the direct cryptographic hash of that image and the other is if I have a relative directory structure that shows how I get to that image. Is that right? Those are the two ways?

[0:13:13.9] DD: Yeah, exactly, exactly. I was describing essentially like ways that you can link to an image from a web page and in the relative URL scheme, like the user will not see that image is located within a folder that lives on IPFS, but in a direct URL so I can extend all resource an absolute path. The user will see that that IPFS hash and will understand that is fetching it from IPFS itself.

[0:13:42.8] JM: The difference that you're talking about here is that you might – you could either link to it through a link that had two folders. It was like /images, /assets, /cryptographic hash of the cat on IPFS, or you could just have a long non-human readable cryptographic hash that is the content address of that image. We'll get into what content addressing is, but why are there these two different ways? Because if I'm a person who is authoring a webpage, why do I have these different options of how to address the content of that web page via IPFS?

[0:14:27.6] DD: It’s actually like standard to the HTTP protocol. Essentially IPFS mimics that standard or enables the – a web page to fetch resources as the users are used to, to make sure that there is an upgrade path, that there is a way for web developers, for creators of web pages to continue using the same relative links that they were using, but now instead of like serving those contents from an S3 bucket, from a static web server somewhere, they can serve it from some IPFS node. The browser will do the same – bite the same type of requests to that IPFS node and it will fetch the content directly from that IPFS node.

[0:15:10.7] JM: Okay, understood. Actually, it would not be /images, /assets, /cryptographic hash. It would be /images, /assets, /cat picture?

[0:15:20.7] DD: Exactly, exactly.

[0:15:22.4] JM: For the relative link?

[0:15:23.5] DD: Exactly.

[0:15:25.0] JM: Right. If I'm a normal Internet user and I want to upload an image to IPFS and I want to be able to access that image, what am I going to do?

[0:15:36.7] DD: Well, if you are user that is comfortable with a terminal, or at least is curious, you can always download the IPFS daemon which executes the whole protocol. When you download IPFS daemon you get this really nice CLI, the command-line interface of IPFS and there is one command, one single comment that you can call called IPFS add. You do IPFS add of that file and IPFS will convert that file into a graph.

It will hash the graph, give you this cryptographic fingerprints; the content address of the file and from that point on the file is available to be accessed through the IPFS network. Of course, you might be a user that is not interested on installing a daemon. No, we don't want to force everyone to use IPFS through a terminal.

What we are seeing and what we have been working on is multiple graphic interfaces that users can use and this graphical interface interfaces come in multiple forms, from a desktop application where you get that icon at the top of your screen, that you can drag and drop the file and it will do the work of again checking that file to [inaudible 0:16:48.5] to a graph and then making it available to the network.

This is what you can install a web extension which supercharges your – give superpowers to your browsers to access the IPFS network. From that web extension, you can then again do the same thing, drag and drop the file and from that point on you have this link, this content address to share with anyone that you want to give access that file, or even given that we now – like we have a full implementation of the protocol in JavaScript, there are web pages out there. For example, one that is very interesting is File Nation, that when you load that web page you are actually installing the protocol itself.

We did the webpage using JavaScript only. That IPFS then starts running on that tab of your browser, can accept files and add them to the network or can access files that existed on a network already.

[0:17:44.2] JM: If I upload that image to IPFS, what is going on across the network? How is that image getting propagated throughout the network of people running IPFS nodes?
[0:17:55.9] DD: Got it. There is two things there. When you add one thing to IPFS – IPFS should not be seen as this global drive that is replicated in every single machine in the world. It is a protocol for resource discovery. It's a protocol for finding the content where it lives and then giving the option to the user to fetch it if they want.

When you add this image to IPFS, what is happening is you are getting that cryptographic fingerprint and then you are telling other nodes in the network through a DHT protocol that if someone is looking for that specific file, that unique identifier, tell them to come to you. If I want to exchange a file with you right now, I would add it on my machine and then I would give you that ID, that unique ID.

When you ask the network, you might not even be connected to me directly, but when you ask the network where the file lives, they will give you my address and then you are able to call to me, to dial to my machine and ask the file directly. From that moment on forward you are also a provider of that file. You can also serve requests to that file.

[SPONSOR MESSAGE]

[0:19:19.7] JM: We are running an experiment to find out if Software Engineering Daily listeners are above average engineers. At triplebyte.com/sedaily you can take a quiz to help us gather data. I took the quiz and it covered a wide range of topics, general programming ability, a little security, a little system design. It was a nice short test to measure how my practical engineering skills have changed since I started this podcast.

I will admit, although I've gotten better at talking about software engineering, I have definitely gotten worse at actually writing code and doing software engineering myself. If you want to check out that quiz yourself and help us gather data, you can take that quiz at triplybyte.com/sedaily. In a few weeks we're going to take a look at the results and we're going to find out if SE Daily listeners are above average.

If you're looking for a job, Triplebyte is a great place to start your search, fast-tracking you at hundreds of top tech companies. Triplebyte takes engineers seriously and does not waste their time. I recommend checking it out at triplebyte.com/sedaily. That's tripleB-Y-T-E.com/sedaily.

Thank you Triplebyte for being a sponsor.

[INTERVIEW CONTINUED]

[0:20:50.8] JM: Okay, so if I host my – if I have my own IP FS node and I stand up a website and I make a link to a cat picture that's sitting on my IPFS node and you David load the web page that I have created that has this link to the cat picture, you are actually loading the cat picture directly from my file system onto your file system. If you're running an IPFS node, you're also going to store the cat picture on your computer?

[0:21:26.2] DD: Yeah exactly. This is all tunable. There are default modes and configurable options. By default, when you fetch something to IPFS you automatically become a provider. However, there are use cases that you might not want to do that. For example, if you are on a mobile device or just like a low power device, you might not want to start becoming the node that every single other machine asks for the content.

You can adjust this. You can say, "Okay, I want to fetch content X, but I don't necessarily have other resources on this specific machine to provide X to the rest of the network. Avoid telling the rest of the world that X lives here.

[0:22:12.5] JM: We're talking about accessing these images for example, like a cat picture image, or a text file, or any file. Again this is content addressing, so location dressing in traditional web development you type in a URL to your web browser and you are taken to that URL. You are taken to the location that that URL maps to and the web page at that location loads, its location addressing. Content addressing is different. Describe the difference between location addressing and content addressing.

[0:22:50.5] DD: Okay, so that's essentially one of the core primitives that IPFS brings into the web. To explain a little bit better, let's talk about the web day. We are so used to rely, like we learn to love the web and like we learn to rely on the web, but the reality it is fragile, so all of these applications that we use to do e-mail, to talk with our loved ones, to run our businesses, to acquire new knowledge are flaky.

As soon as something happens to the connectivity, if you don't have that access to the backbone, if you don't have access to DNS, those applications will just break. The reason why they break is because there is an assumption that you can always identify the location of the data and that there is a trust model where if I'm for example accessing a cat picture on a website called catpictures.com, there's address model that if I want a cat picture from that website, I need to go to where that website lives and fetch that cat picture.

This creates a problem, because if I give you a link to catpictures.com/cate or Ozzie – for example catpicture.com/ozzie and I give you that link, someone can after go to that location, change the cat picture to a dog picture. If you didn't have any knowledge about what cats are, now you rely on the trust that you have on me for giving you that link, and you will now think that the cat is actually a dog. You will download that file and you'll believe that file is correct and you have no way to verify that the thing that I linked for you is the thing that I wanted you to see.

With content addressing these changes, right? Content addressing is saying, "Okay, let's forget about location. Let's forget about the central point of authority and then let's give the user the pure way to prove that the data that was fetched is exactly what they intended to fetch."

This is where cryptographic hash functions enter into play. When you use a cryptographic hash function over some piece of data, you essentially get a unique identifier. It is pretty much impossible to find two pictures that match to two different, or find one unique identifier that matches two pieces of data.

When you are searching a network for a file, a cat picture, whatever, and once you download it you can apply the same hash function that I applied to that file and verify bit by bit that that ID is exactly the same. You can do a manual proof locally.

If someone were just using this, the previous example, if someone really changed that cat picture to a dog, the moment that you fetch that image you would hash it and see that the hash, the cryptographic fingerprint is different and you would discard it, because now you know that it is not the file that I was linking to you in the first place.

[0:25:57.6] JM: If I am loading, or let's say I've got this webpage. To continue our example, I've got this webpage that I've created and I'm hosting it somewhere on the internet, but I've got a URL to a content addressed cryptographic hash named cat picture that's sitting on my IPFS node on my computer and you load that webpage and your computer tries to load the link of the cat picture, the cryptographically hashed cat content or dressed cat picture. It's going to request it from the IPFS network and assuming it hits my node, it's going to essentially, if I have it right, it's going to do a hash table lookup, or it's going to map the cryptographic hash to the content that's associated with that hash on my node and it's going to deliver you the cat picture.

Then on your local node, you can do this, run the same hash function over that cat picture, compare it to the cryptographic hash that you used to request the cat picture from the IPFS Network and you will be able to detect whether it matches or not.

[0:27:17.6] DD: Exactly, exactly. That's a 100% correct. The content address – okay, content address of an object is calculated through cryptographic hash. What are the other advantages of this content addressing model?

[0:27:31.6] DD: With this property with this primitive, you no longer require – you are not required to go again to that central point of authority, to that server somewhere to fetch the content. Imagine that like we are sitting in the same room and I want to share a document with you, or a picture.

Today, what would probably happen is that I would upload that picture to some external service, to some cloud service, or to some social network and then you would grab your device and you download that picture from that cloud service. If we happen to be disconnected again from TNS or from – or if that service was down, then that service, that that application of sharing the picture would just break down. It will not work.

With content addressing, what happens is that given that we are in the same network, that we have access, that we can see each other, you can fetch, you can ask the local area network for that exact same file. My machine will catch that request and will serve the file to you directly.

This means like in areas of the world where well, the internet connectivity is not just as good as we are used to in developed countries, or simply because networks are like sensor in some way, that users cannot access certain types of content or just cannot access the content at all, because the content is far enough that things like a TLS handshake would timeout. Things like a cryptographic handshake and then server will timeout, you can still serve that content to those nodes through IPFS, because as long as the content is close enough, the content will travel from the closest point to the machine that is requesting that content without needing to rely again on some central point of authority to tell which content is which and which is the correct one.

You can think of like that. How many times have you checked a webpage? Then you are boarding your flight. You are about to take off. You were just checking the webpage with some news article and by mistake like you click the refresh button, or you happen just to click the refresh button, but then the connectivity was no longer there because you took off and now there is no Wi-Fi, or there is no 4G connectivity. Suddenly your browser says, “Oh, I cannot access this page anymore.”

Yet, you just accessed that page, right? You just had it in your machine. The same thing goes for all the other users that are on that flight. You add that file on your machine and someone else in the plane might want to see that news article as well, but they have no way in the current web to find your machine inside that flight and to request that file directly to you, because there is no way to validate that they would get the correct file in the end.

[0:30:28.5] JM: Are there places in the world where the internet connectivity is so partitioned that that would be a useful application? Or do you have a sense for how well internet connectivity has permeated the world?

[0:30:46.3] DD: The reality, it's improving, but not as fast as we would like. .The reality is it's really hard to get the same level of internet connectivity that we have in a developed country in a developing country, or even just remote areas. If you go to a mountain, like the internet connectivity situation there is going to be completely different.

What is happening is that the amount of data that we generate today with video, with photos, with like all sorts of media is growing up so fast, that compared to our bandwidth is evolving that is not growing as fast. What is happening like the Internet is starting to feel slower, because each time want to pipe video that is even more high resolution like 4k, 8k and so on, and because the pipes aren't the way as we would like them to be, the amount of time that is required to pipe all the data through the same cables is going to be longer. It's going to take more time.

It will reach a point where there are some areas where the time that it would cost to transfer all the data to that location will just not justify, like fetching the data from some specific center points at all.

[0:32:02.7] JM: I think we've done a good job of communicating how IPFS works at a basic level. It's a peer-to-peer file system. We need to have ways that nodes can join the network and identify themselves and they're going to do that by creating a private and public key pair and then joining the network. When I join a network with my IPFS node, how do I find other peers?

[0:32:28.3] DD: When you turn on your IPFS node, multiple things happen. We actually promoted the networking stack of IPFS to what we say a first-class project and we gave it a name called we peer-to-peer. IPFS, the file system uses the networking stack we peer-to-peer to do the discovery and the routing amongst nodes.

What happens when you connect is first your node is going to check if there are other nodes in your local area network to connect to using multicast DNS, so one discovery service. Then it is going to connect to some bootstrapping nodes, some nodes that we provide with a default configuration of IPFS. You can also add your own if you want, but these nodes are the railing points into the peer-to-peer network.

Once you connect to those nodes, then you can start doing routing, or you can start asking those nodes for the whereabouts of other nodes. When I need to find a specific node in a network, I can do a recursive, actually an iterative query of asking the whereabouts of a specific peer ID, a specific public key. Even if a node doesn't know a direct – a like doesn't know where the peer is, it can tell me more nodes that I should contact to ask for the whereabouts of that node. After some time, eventually find that node that I want to dial to.

[0:33:53.1] JM: If I want to connect to nodes who have a particular object, like if I'm requesting a cat picture from the IPFS network, I know I need to find that cat picture, how does the network route me to the correct nodes into the correct object?

[0:34:14.0] DD: The way that it works right now is using a DHT, a distributed hash table. This is a term that gets thrown around a lot nowadays in peer-to-peer LAN. DHT essentially is two things. First, it's a routing algorithm. Think off in the same way that IP packets get routed on the IP network. You can route requests on an overlay network that uses a specific algorithm to organize in a logical way, so you don't have any [inaudible 0:34:44.4] per se, like you have a logical way to organize the nodes.

The second part of a DHT is an agreement between all the nodes in the network that if they are storing a file, they will create that record saying that they are storing that file and they will put that record on the network itself. When I'm looking for a specific file on the network, I first go to the network and ask, “Does anyone know if someone has stored this record on the network containing a location of this file?” Once I find that record on a network, then I know the exact peer, or peers that are storing that file. Once I have that information, I can do a direct connection to that peer and request that file directly.

[0:35:32.2] JM: In most internet systems, DNS plays some key role. You have a domain name system that lets you find IP addresses that can serve content associated with a website. Is there any role for DNS in the IPFS system, or is this all about just having the right content addresses?

[0:35:55.0] DD: No, absolutely. DNS also plays an important role on IPFS. What we did with IPFS and again is to grab all of these other systems, all of these other great ideas that existed before and put them on boxes with very well-defined interfaces, so that we can use them depending on the scenario, the network scenario.

In IPFS you have multiple transports, like if you are running, you're nailed in a desktop, you can use TCP and WebSockets and UDP. If you are running on the browser, now you have to use WebRTC or WebSockets. For DNS, like the specific service that DNS provides is content routing. This is the agreement I was talking about when I was describing up the DHT. With DNS we can again, in the same way store keys that basically contain hashes, like we can store for example a text record on a DNS name that as an IPFS hash.

Then when a node, an IPFS node tries to resolve that DNS name, it will check its text records and it will find an IPFS hash and then it will resolve that IPFS hash and serve that content that is behind IPFS hash. A practical example is the ipfs.io web page today. When you go to ipfs.io what is exactly happening is that that request is being directed to an IPFS node, the IPFS node understands that the browser that is requesting is coming from a referrer ipfs.io. It will resolve that domain name, it will get the text record, it will see that there is an IPFS hash there, it will resolve that hash on the IPFS network, get the website and then serve that website to you as a regular web server. From the user perspective, you don't even see that IPFS is there, but it's IPFS doing the work.

[SPONSOR MESSAGE]

[0:38:04.7] JM: You're a successful developer and you couldn't have gotten to where you are without help in your education and career. Maybe you're thinking about ways to give back in the community where you live. The TEALS Program is looking for engineers from across the country to volunteer to teach computer science in high schools. Work with a computer science teacher in the classroom to bring development concepts to life through teamwork and determination.

Pay your success forward by volunteering with high school students in your area by encouraging them on the computer science path. You can make a difference. If you'd like to learn more about the Microsoft TEALS Program, or submit your volunteer application, go to tealsk12.org/sedaily. That's T-E-A-L-S-K-1-2.org/sedaily.

Thank you to the TEALS Program for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:39:10.2] JM: If there's some global internet problem and a bunch of these IPFS nodes get separated from each other, in the instance of a network partition, how do the IPFS nodes get reconnected to each other? How are they going to find each other in the instance of a network partition? Because it seems like in this peer-to-peer network, you could get instances where I try to load a Wikipedia page and it's got 200 images and if there's a network partition and I'm away – I'm partitioned away from most of the IPFS nodes in the network, maybe only half of those images load.

[0:39:52.6] DD: That's true. What you're saying is there is a physical partition, right? There is actually a disconnect at the physical layer. There is no way that machine A is going to get to machine B.

[0:40:04.8] JM: Right. We could talk about what happens there and then we could talk about what happens if that – as that partition gets resolved and nodes start to look for each other to connect to, I guess I'm just trying to understand the reconnection strategy and the data distribution strategy as there are as there's turmoil in the network.

[0:40:24.8] DD: Yeah, absolutely. We could say that there is physical partitions, where there is no way for the nodes to find each other, because there is just no medium for them to find each other and there is logical partitions when there is like a firewall, or some network policy that then limits the ability for nodes to connect to another node.

For the first case, what will happen is that as long as the data leaves on your side of the partition, you'll be able to find it. This is something that is really different from the current centralized web. Imagine like you have 10 nodes and there is a partition right in the middle. We get five to each side and you happen to have two nodes with the data that you are interested and luckily each of those nodes gets in one side of the partition. Every single other of the four nodes will be able to find that data the same way as they doing before.

Of course, if you are unlucky and if the two nodes that add the data are on the partition side that you are not present, then you will not be able to find that data. Nodes will try however again, and this is where the question about the logical partition comes into play. Nodes we will try to find other ways to round amongst themselves.

I've been this client which node has the content and then does a direct connection, but sometimes that is not possible. That actually happens a lot of times. Imagine like again, I have a desktop node that is only speaking the TCP transport. The content I'm looking for is actually on a browser node and the browser node is always speaking WebRTC. Because there is no parity between transports, no direct connection can be established.

What the IPFS protocol also provides is a relay mechanism, where both nodes can find a middle point. Some other nodes or multiple nodes that can create a bridge that they speak the two protocols and so they can create a bridge from that desktop node to the browser node. This way, it can go around this logical partitioning problem.

[0:42:30.6] JM: I want to give people another tangible example that they can keep in their heads as to the usefulness of IPFS. A year ago, the country of Turkey blocked Wikipedia and a copy of Wikipedia was put on IPFS. Then the people in Turkey were able to access Wikipedia. This is an example of censorship resistance via IPFS. Explain what happened in this scenario.

[0:42:59.7] DD: Yeah. That was a super important moment for us. Essentially, it happened very suddenly. The team, the IPFS community felt it was like a call to action, a very important one. We have been building this system. We have the tools available. We understood the problem and we wanted to make sure that every citizen, that every person that was living in Turkey at a time would still continue to access to a resource such as Wikipedia.

What we essentially did was like over a weekend, so in less than 48 hours, created a replica, like grab one of the Wikipedia snapshots and added to IPFS creating a read-only version of the Wikipedia website. Then on top of that, because well, one of the main features of Wikipedia is that you can actually search on Wikipedia, is that we created a search engine that would run locally so that a user that would load Wikipedia would be able to do searches and find files on their local copy.

Once the Wikipedia mirror was put on top of IPFS, then any node in a network could start fetching it. As any node and that we started fetching it, more and more replica started appearing and it just became available everywhere.

[0:44:19.3] JM: There were people in Turkey who were running IPFS nodes.

[0:44:23.1] DD: Yeah, yeah. There were people in Turkey and actually other people that were just incentivizing or encouraging people to learn about IPFS and see that IPFS, that through the IPFS protocol they could still access the Wikipedia page.

There is multiple people in the community, that because community some participating in the code base that was necessary to ship this feature, some participating on the infrastructure, just making sure that we had the replicas available, that if any IPFS node coming from Turkey try to access IPFS, it was available for them and it would load fast. There were people just creating the examples and explaining to the Turkish citizens that they could use IPFS to access Wikipedia.

[0:45:06.5] JM: Do you think this was – was this like a more of a proof of concept, or were their actual students, for example, sitting in a university somewhere who all of a sudden they go to Wikipedia, it's blocked because the government decided to block it, and then maybe they Google around a little bit, or they get an e-mail from one of their friends that says, “Hey, if you need to use Wikipedia today, you first have to download IPFS and then you plug into IPFS and then you've got Wikipedia.” Were there were there real-world examples of people who are doing that?

[0:45:40.8] DD: Yeah, so we definitely noticed a network growth. One important thing to note is that given that the IPFS protocol is fully distributed and there is no central point, there is no way for us to really measure the size of the network other than counting the number of peer IDs that we see passing through the network.

In events like this, what we see is like a search on peer IDs appearing and connecting to the bootstrapper nodes and data. During that time, we saw an increase of nodes appearing, new nodes appearing and we – well, given the event that was happening, we can fairly assume that lots of those nodes were coming from Turkey, and also just other people around the world that were learning about the situation and that wanted to be part of it and wanted to help. They ended up installing IPFS as well.

[0:46:32.1] JM: Okay. First person in, let's say the first person in Turkey sets up an IPFS node with the Wikipedia, the grabbable and downloadable full Wikipedia snapshot and then other people have IPFS nodes that they want to access that Wikipedia snapshot from, they need to have the content address of that Wikipedia snapshot, is that correct?

[0:47:00.9] DD: Yeah, exactly. That's correct.

[0:47:02.3] JM: Okay. If I decide I want to access that snapshot from the from the first seed, the first person on the IPFS network in Turkey that has that copy, I need to have the address just like I need to have the URL of google.com in order to go to google.com. I guess, to propagate that Wikipedia snapshot through Turkey, it's just people that are requesting the webpage, or is there some way that the network pushes the Wikipedia copies out through the network even if people are not directly requesting it?

[0:47:41.8] DD: Got it. That's a question that we get a lot is IPFS replicating the data over the network without users requests or not. Other peer-to-peer systems made a decision to treat every single node in a network as part of this global file system that everyone shares and everyone contributes to. In IPFS, we decided to leave that decision to the user itself.

A user will only replicate the data if they tell to the IPFS node to fetch the data in the first place. If a user doesn't want to store some piece of data, because well they are not interested in contributing to archive some data set or they just don't have the resources to contribute to the archiving of that data set, they don't have to. It requires again these users to go to their IPFS nodes, try to fetch the Wikipedia page. Then once they finish fetching the Wikipedia page, they become provider theirs to the network itself.

[0:48:39.2] JM: Can you tell me any more of the finer points about how this played out in Turkey? Any interesting notes that will help illustrate how IPFS works?

[0:48:48.6] DD: We've [inaudible 0:48:50.0] to IPFS. What exactly happened was like people were in about IPFS protocol, they got it installed, they started realizing that is a better alternative to HTTP and it's an uncesorable alternative to HTTP. Basically just raise a lot of awareness, raise a lot of interest and created an alternative way for people to access the Wikipedia documents.

[0:49:14.3] JM: Worth pointing out not only is it uncensorable, but if again, as we mentioned earlier if there were – if somebody spun up a malicious node on the network that had a fake version of Wikipedia, then you would enter the content address of the safe version of Wikipedia. If that malicious node served you the fake version of Wikipedia you would know, because you would hash the Wikipedia instance and it would hash to something different than the content address that you have.

[0:49:46.8] DD: Exactly, exactly. If there were any malicious nodes which was trying to attempt like say change public opinion by changing the data that is present in some of the pages of Wikipedia or even any other website that is being served by IPFS, the users would know right away that a change has happened, because again, the IPFS in every single piece of data that it would receive for the same hashes. If they didn't match, then it will alert the user saying that the data is not valid.

[0:50:18.6] JM: Okay. Let's talk about IPFS in terms of its new upsides, rather than just talking about how it prevents censorship. We've been talking about decentralized app development in some of the previous episodes and what the barriers are to widespread decentralized app development. Where does IPFS fit into the decentralized app stack?

[0:50:47.5] DD: Perfect. Yeah, like we have been seeing a surge of applications using IPFS, the distributive applications or the apps for short. IPFS makes those applications working scenarios where like even centralized apps could not work before. One example is for example peerpad.net, it's an open source application, it is an example of the full IPFS stack. What it enables you to do is to create a collaborative document.

You create this path where you can write and you can share the URL, the link with another user. When that user loads that link, both the node that is running on your web browser and the node that is running on the URL user web browser will connect. Given that there is a direct connection, they can signal changes on the document itself, and they can notify the users when there is a state change, like an edit on the document.

[0:51:52.7] JM: By the way, that's awesome. We did a whole show about CRDTs, where we were discussing this topic, the fact that when you're editing a Google Doc, the Google Doc has to basically be centralized in the Google servers. You could just have me and you sharing the Google Doc. There's not really a common place model for that document editing to take place, but you're describing one that's backed by IPFS.

[0:52:22.5] DD: Yeah, exactly. That's perfect. That's great that more people are talking about CDRTs, because that's something – a piece of technology that still doesn't get enough attention, I think. The reason why CRDTs work so well in IPFS is because first, you get the network connectivity figured out for you, like you don't have to worry about where the process where the other node leaves. IPFS and we peer-to-peer will just get you connected.
The second part is something that we haven't mentioned here yet is that IPFS is actually more than a file system. It's actually a graph database underneath. Every time that you add a file to IPFS, you are not grabbing the whole file and hashing the whole thing in one swipe, and then getting the cryptographic fingerprint. You actually transform that file into a graph, so that becomes a data structure that is very interesting for example syncing. Like if you change one byte or one line of the file, you don't have to transfer the whole file again through the IPFS network. You can just exchange the little piece that changed and the other node will be able to reconstruct a new version using pieces from the previous version.

With this graph database, with this graph structure, which we call IPLD, so interplanetary link data, you can create a CRDT, which essentially is a graph that has a merge function, that has instructions for the nodes to merge the multiple events that are generated by the multiple nodes participating in the CRDT.

These very interesting data structures where you have multiple people collaborating over a document and independently of the order of events that they see arriving. This is like one of the cool features from CRDTs is that I can be working with you on a document and then I can go offline and you can make a bunch of changes, then you work on a document with another friends and you can make a bunch of more changes, then I can meet the other friend at some later point in time and receive all the events, all of the changes to it, to apply to the document at [inaudible 0:54:26.7] point in time and without any synchronization other than just fetching the changes, I will be able to converge to the same state that you without connecting to me see you on your machine.

It's like this free synchronization property, this free synchronization primitive that you get without needing again, as you said, the central server that Google Docs has to use to merge the changes.

[0:54:53.7] JM: Okay, we're totally up against time. There's so much that we didn't cover, so I want to do more shows about IPFS may be with you or one of your colleagues, but just to wrap things and we didn't even touch on Filecoin, but just to wrap things up, when you compare using something like Amazon S3, which is really cheap to use, can you imagine a world where IPFS has some advantage over S3? Maybe it's completely free to some people, or could the cost structures get good enough, or could there be some set of features that would make it compete with something like Amazon S3 or Google Cloud Storage? Or is this is this just useful for uncensorable communication?

[0:55:41.6] DD: Not totally. When we designed Filecoin and we designed IPFS was literally thinking to create a better fabric to move data across the globe. It was not just to store files, or to do archives, or to be – or even just to be an uncensorable network, it was to have all these properties.

S3 is definitely a very cheap service and it provides a very good quality of service for its costs and its features, but it is not necessarily cheap to use S3 to serve the entire world. If you start accounting the bandwidth cost, for example and if you want to serve every single machine that exists in the world from S3, then the cost quickly become out of control.

With IPFS, one of the things that you get by default exists in a machine closer to the user, then that user can just fetch the file that is on a machine that it's closer without having to force you to pay some cloud provider for that bandwidth as well. The users will reciprocate and will help each other getting the data that they're already storing. With Filecoin, you get even one more property, which is there a huge amount of latent storage out there in the world, like data centers that are completely stopped.

Like my machine right now has a couple of hundred gigabytes that I'm not using and this machine is present and very close to other computers. Right now as a user of the web, I don't have any way to request any other machine in a network other than cloud providers to store files for me, to get my files closer to the users that I want to serve.

With Filecoin, you could literally say, "Oh, I have –" For example, you could grab all your podcasts and you could target a new region of the world to serve your podcast to and then instead of giving like a slow access to your podcasts or paying a very expensive bill to a cloud provider, you could just pay a little bit of Filecoin to the users that are close to that region that you are interested and you could store the podcast there, so that when the requests are made they get served locally, or that they get served in that region.

This is like a one of the very interesting and very compelling factors from Filecoin. It is the fact that you can put the data very close to the users through this free market and everyone can participate on this market and provide the service, provide the storage.

[0:58:11.6] JM: Sure. I mean, I'm looking forward to not paying – I think I pay $80 a month for a podcast CDN. I'm looking forward to IPFS having more wide scale distribution.

[0:58:22.9] DD: Yeah, we should write the podcast on IPFS right now and see –

[0:58:25.6] JM: Let's do. Let's do it.

[0:58:27.1] DD: Yeah, that's perfect.

[0:58:28.3] JM: All right. Okay, David thanks for coming on Software Engineering Daily. It's been awesome talking to you. I look forward to doing more coverage of IPFS.

[0:58:36.0] DD: Thank you so much. Yeah, there are many things that I would like to talk more and thank you so much.

[END OF INTERVIEW]

[0:58:46.3] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous deliver to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]