Notes form Moxie Marlinspike's presentation titled The ecosystem is moving: challenges for distributed and decentralized technologies from the perspective of Signal development

at the 36th Chaos Communication Congress in Leipzig, Germany.

I spent time on this to better understand whySignal chose a centralized servers approach rather than afederated approach. This presentation answers that question.

tl;dr

- Software evolves - evolution is incompatible with the static nature of decentralized systems. Swift and timely upgrades are not easy.
- Centralization is not as bad as it seems, you can iterate quickly - centralizing protocols has been a recipe for success. Its what slack did with IRC, its what Facebook did with email, It's what WhatsApp did with XMPP.
- By decentralization we really want; privacy, censorship resistance, availability and control - you can get all of these properties without decentralization
- Open-source software (OSS) plays a larger role in availability and control than decentralization.
- Anyone who is working on decentralized systems today needs to do so without forgetting that the ecosystem is moving

The ecosystem is moving

Signal is a private messaging app but it is not decentralized. There is no

federated, mesh , p2p, blockchain something something. Now and then people say there should be a federated, mesh, p2p, blockchain something something. So let's talk a bit about decentralized systems from the perspective of the time I've spent at signal and the work done there and what we are trying to accomplish.

At a high level, while I work in Software, I greatly envy musicians, writers, filmmakers painters - these are people who create things and can be finished forever. You can record an album today and twenty years later you can listen to that album and appreciate the same. But software is never finished. You cannot write a piece of software today - like write an app and then twenty years later enjoy that app in the same way. Because software is part of an ecosystem and the ecosystem is moving

. The platform changes out from under it. Networks evolve, security threats are in constant flux, the UX language that we've all learned to speak and learn together rarely sits still. As more money time and focus has gone into this entire ecosystem the faster the whole thing has begun to travel.

The world is now filled with rooms that look like these in buildings that look like these that are packed to the rafters with people who sit in front of a computer for eight hours per day every single day. All the energy and momentum behind that means that user expectations are evolving rapidly. Evolving rapidly is in contradiction with decentralization. How is that possible, what does that mean?

Afterall the internet is decentralized, that seems like a dynamically rapidly evolving place. When you look at the fundamentals of the internet, that's not always the case. For instance, if you look at IP one of the fundamental protocols of the internet. How have we done with that? We've gotten the first production of IP, but we have not gotten to the second production version. HTTP we got to version 1.1 in 1997 and we've been basically stuck there until now. SMTP, IRC, XMPP, DNS, its all the same, they are all frozen in time circa the 1990s.

Once we decentralize a protocol it becomes extremely difficult to change. You put it out there in the world, there are many clients, different implementations, different deployments, so making any changes becomes extremely difficult. Meanwhile, centralizing protocols has been a recipe for success

. Its what slack did with IRC, its what Facebook did with email, It's what WhatsApp did with SMPP. In each case, the decentralized protocol is stuck in time, but the people who have centralized them have been able to iterate super quickly and develop these products that are extremely compelling to people.

The fact that email is decentralized also means that email is not encrypted, and it never will be. Why? Because its too difficult to change at this point. Its out there in the world and making that change is probably not going to happen. WhatsApp is centralized so I don't run my own WhatsApp server or have my own data store, but its end-to-end encrypted for billions of people by default. And WhatsApp was able to just role that out with a software update. Why do we want decentralization anyway? People discuss decentralization a lot, but what is it that we are really after?

When you break it down the participants of decentralization are advocating for increased privacy, censorship resistance, availability, and control

. These the things I thing people who are into decentralization are really kind of looking for and hope to get out of that world. Let's look at these things in turn.

Privacy

We've already seen that decentralized systems are not inherently encrypted. In fact, most decentralized systems in the world are not end-to-end encrypted by default. There's nothing about decentralization that makes the things encrypted. I think advocates of decentralization have a different take on Privacy , which is one of data ownership, the idea that you can run a service yourself and can maintain ownership over that data and that includes metadata, not just the contents of things like messages but also the metadata about them in a sense that is better than just some encryption solution. But in numerous ways, this is an antiquated notion that is left over from a time when computers were for computer people. In the 1990s the general thesis was let's develop these powerful tools and teach everyone to be like us - that's not really how the world developed. Not only would everyone be a consumer and producer of content but also of infrastructure. Neither of those things bore out. In reality, things seem to naturally rollup and converge into these super nodes that people are making use of, people aren't all producers and consumers of content or infrastructure.

Given that world, while I host my own email, since the world looks like this I don't actually have any meaningful data ownership even though I run my own mail server. I don't actually have any kind of metadata protection because every email that I send or receive has Gmail on the other end of it. Given the world has developed in this direction, real data protection comes from things like end-to-end encryption than it is from data ownership. Things like metadata protection will require new techniques and that those new techniques are more likely to evolve in centralized rather than decentralized environments because centralized environments are where things tend to change.

For example, at Signal, this is an area we've been working on a lot. At signal, we have technologies like private groups, contact discovery, sealed sender. These are things that mean that the Signal service has no visibility into group communication state, or membership information, no visibility into your contacts or social graph, and no visibility into who is messaging who.

Private Groups

Looking at something like Private Groups. The way the group state is usually maintained is on a server you have a database and you have a record for that group. The group needs to contain information like what's the group title, avatar, what is the group membership, who is in this group, what are their roles, are they an administrator, a creator, is this read only member? Maybe some group attributes like a pinned message or something like that. Clients can query this database to render group information for the user. Given that it seems unlikely we move to a world where everyone is running their own servers in addition to their own clients, merely just putting this plain text database on everyone's servers is unlikely.

One thing you might think about doing is just encrypting it. You have a server-side database and in the database all the entries are encrypted with a key shared by group members that the server has no visibility into. The problem is - you also need a server to be able to enforce access controls and basic rules. The server should be able to look at the members of your group and determine whether a group member is authorized to make changes, change the title, add or remove group members, but if the data is encrypted, how is the server going to do that?

At signal, we developed an anonymous credential scheme that allows the server to store encrypted membership lists. The server has a record for some random group of its members, but each of its members are encrypted, so the server doesn't know who those members of the group are. Let's say Alice wants to add someone to the group. She can construct a zero knowledge proof and authenticate to the server, proving that she has a signed credential that matches the encrypted contents of one of the group members, without ever revealing what the contents of that record are or who she is.

Once authenticated, Alice can add another member to the group, like Frank, who can come along and do the same as Alice - he constructs a ZKP and can prove in zero knowledge who he is without revealing or the content of this record to the server that he is a group member. If he requests a group membership list, the server transmits the encrypted values to him which he can decrypt locally with the key that's shared among group members, and determine who was in the group and display that to the user.

New Techniques

This is an example of some new cryptography we developed to solve this problem. It's a new technique to offer some privacy preserving technology in a space that is more likely to happen in places where we can make these changes and roll them out super easily. All of this adds up to a world where I can publish the server-side state for my signal account. There is nothing in it really. Even the profile data is encrypted, the only real unencrypted values are the last seen time and when the account was created. There is no group information about what groups I'm in, the titles of those groups, who the other group members are, my contacts are not stored there, my social graph is not visible to the service, and my profile data is not visible to the service. The server doesn't have visibility into when I message people, and when they message me.

Meanwhile, my email still isn't end-to-end encrypted, and it never will be. Even if we did live in this world where the internet looks differently and everyone is a consumer and producer of content and infrastructure, this p2p world is not necessarily privacy preserving in of itself. For example, when we first rolled out voice and video calling in Signal we designed it so it did establish p2p connections between both parties of the call. If I call somebody, I establish a direct connection to that device. However, when we deployed that people were like wait a second, does that mean someone can just call me and learn my IP address? What about the metadata here, my ISP or someone on WiFi on the same network can see who is calling, and who is calling me. Is there anything we can do about this? Yes, route it through a server instead, that is what we do in numerous instances.

In thinking about privacy, decentralized systems aren't inherently going to give us the privacy properties that we desire. Its more likely we can develop technology to offer what it is people want in centralized environments.

Censorship Resistance

This is another area where I feel like the idea for censorship resistance in decentralized environments is that, many things are more difficult to block than just one thing. If you have many servers, it's more difficult for a censor to block access to those than block access to one server. I feel like this is an antiquated notion left over from a different time. In today's world, if something is user discoverable, that is also going to be censor discoverable in many ways. But the basic idea if you are such and such at something.com and something.com gets blocked you can just move to somethingelse@something else.com and you can sort of keep moving around like that. The problem is when you do that, you blow up your entire social graph. If you imagine a scenario where there are a bunch of different users who are affiliated with a bunch of different servers that if one server gets blocked by a censor, users who can no longer access that server switch to different servers.

The problem is that as soon as they do that, they have to be rediscovered by everyone else on the network because they now have a new address. And its more likely that if one server is blocked at any given moment that all servers will be blocked at that specific moment and everyone has to switch to a whole other thing. At that point, you have blown up your social network - everyone has to rediscover everyone from the beginning. You are basically playing a game of whack-a-mole that is asymmetric. Everyday that censors take an action to block known servers is basically the first day of your social network - everyone has to start from scratch and rediscover each other all over again.

To the extent that your strategy is just bouncing around, it's actually more effective to have one centralized service with multiple Ingress points. If you have a service and there is a bunch of users using that service, if access to that service gets blocked, to just spin up another ingress point, a proxy or VPN that everyone can switch to. At the moment people switch it's the same switching strategy, but you are not blowing up your social network - everyone has the same address and can be identified, if that gets blocked you switch to another one. You are playing a game of whack-a-mole, but it's not asymmetric because the switching cost is very low. This is the kind of strategy that WhatsApp and Signal have used to resist censorship attempts most times that they have been attempted.

Domain Fronting

Domain fronting

is a technique for Internet censorship circumvention that uses different domain names in different communication layers of an HTTPS connection to discreetly connect to a different target domain than is discernable to third parties monitoring the requests and connections.

Domain fronting, basically, is a technique where a client connects to a CDN that's operated by a large CDN provider and does a DNS and SNI TLS connection with one host like some large service like google maps, but then the https host header specifies a different address like a proxy. In order to block this the censor now needs to block access to some larger set of services rather than one specific service.

Proxy Sharding

You set up multiple different ingress points and you shard access to them to different users. Only some users can discover some access points, which means a censor can't discover all the access points quickly. As things get blocked, you keep shuffling around. These are things that require that you move quickly. As people block access to a service, you are moving quickly to respond. This is not something that comes easy in decentralized environments. When it comes to censorship resistance you are more likely to see effective censorship resistance in centralized environments rather than decentralized environments, and often this is what we have seen.

Availability

Every time there is an outage people say we should decentralize because we wouldn't have as many outages. The reality is you would just have more outages. If you have a centralized service, and you want to move it into two different data centers, the way did that is by splitting the data up between these two data centers. You have just halved your availability because the mean time between failure now goes up. Since you have two data centers which means that it's more likely, there will be an outage in one of those data centers at any moment, since you've split your data between the two, then you halved the availability of that data. I don't think availability is necessarily something you are likely to see better in decentralized than centralized environments.

Control

This is a fascinating moment. The current sort of sentiment in the world today has changed a lot. Now people feel the Internet is this terrible place in ways that people didn't used to feel - the era of utopianism and this vision of technology providing a better and brighter future is coming to an end. A lot of that comes down to a feeling that we have a lack of control of it. Technology is not serving our needs in the way that we want it to and we don't have any control or agency over how that is manifest. I think the strategies that partisans of decentralized environments have for manifesting that control is this switching idea primarily. If you have a federated environment, different services can behave differently. If you were a

subscriber of one service and your provider started to behave in a way you felt was inappropriate, you could just switch providers but not lose access to the entire network.

This does have a certain appeal. If this is true and a strategy worth pursuing, we need to ask ourselves why do people still use Yahoo Mail. It hasn't been updated in 10 years, they had a massive series of security incidents, it's not clear who owns it anymore, but many people are still using Yahoo Mail. Why? Because changing e-mail is hard. Switching from Yahoo to Gmail is actually harder than swatching from WhatsApp to Telegram to Signal. Every time you switch email providers, you basically blow up your social network. Everyone has to rediscover their new federated identifier. If you use a non federated identifier like a phone number as the basis for your social network, switching between different services that actually aren't connected to each other is easier than switching between federated services. The notifications on your device or desktop becomes the federating bridge between those networks, in a way that is in some sense more effective than the federated models ever were.

The other strategy for regaining control from a decentralized environment is extensibility. This is the idea that what we can do is develop a protocol that is designed to be extended. So that different people can modify this technology in ways that feel like it meets their needs. The most well-known example of this is a protocol known as XMPP, which was a chat protocol designed to be extensible. In the end, what we winded up with was this morass of XEPs which were the extensions and there wasn't ever a feeling of strong consistency, which generated a lot of uncertainty within the user experience. Even today, if you want to send a video over XMPP, there is a XEP for that, does the recipient support it? Can you send a GIF? It's a little dicey. None of the extensibility that was built into the protocol could adapt to major changes like adapting to mobile environments. The whole extensibility thing didn't really provide the control people wanted because those XEPS were of little value until they were adopted everywhere, which is difficult to do.

In pseudo-distributed models like Bitcoin, the control people are seeking is in the form of forks. So when there is a disagreement, people just start a new network like Bitcoin Cash or various alt coins that people take the existing code base and just start another service. This has led to a lot of confusion for users who are engaging with these networks. To the extent that people are manifesting the control they would like to see - to me it seems like it doesn't have to do with the decentralized nature of these protocols, it has more to do with the open-source nature of these projects. That because these projects are open source, it's easy for people to take what is there, change it, and redeploy it as something else. In a sense, open source is the best tool that we have in terms of manifesting control.

But even that is a difficult ask because if what we want is for technology to better serve us then if what technology demands is rooms that look like this, in buildings that look like this, full of people who sit in front of a computer for eight hours a day everyday forever than its unlikely we're going to see technology meeting our needs in the way we want it to.

Conclusion

All the time, people have these ideas, like what if there was Uber, but it was decentralized, so the money goes to drivers, wouldn't that be cool. If what it takes to build that requires the prerequisites, guess where the money is going to go? It's going to go to those places. If we are serious about changing technology so that it better serves us, the best thing we can do is to make the deployment and development of technology easier. Decentralized systems are not the first thing that pop into my mind when I think of easy. In many ways, I feel like decentralized systems make things harder in a world where we should be trying to make things easier for people to deploy.

On a whole, these are the challenges of decentralized systems. In many ways, we need to reimagine how it is we think about technology given the direction the world has gone. We can find the solutions to these problems and the things we are looking for in things that are more effective than building decentralized systems. I am not entirely optimistic about the future of decentralized systems but would also like to be proven wrong. Anyone who is working on decentralized systems today needs to do so without forgetting that the ecosystem is moving. In the words of Marx, "we can create our own history, but only under circumstances that are directly transmitted from the past."