IPFS Ideas for InstantOS

by Federico “EvilScript” Torrielli

Part 1: small introduction to IPFS (and problems)

IPFS is a very recent distributed (file) system that works like an always-on peer-to-peer program in the background. IPFS works with peers and gateways. Peers are, to be simple, computers. Gateways are centralized systems that bridge between the file that you want via a link and the IPFS network. They work like normal web-servers but they just convert a HTTP link into an IPFS resource. You can learn everything about them here.

The advantages of having a great IPFS network is that files never disappear. Since there is no centralized server to host the resource, there is no way to DDOS a single server to put down a resouce. The only way for a resource to disappear is to make it disappear from every computer on the distributed net (So if the file *InstantOS.iso* is hosted (or partially hosted, more on this later) on all computers, there is no way for an ISP to censor that ISO or for a server to have downtimes.

So, does it speed up downloads? You may be asking. The answer is **yes**, you sussy baka. More computers == more speed. IPFS it’s fully configurable: you can set max/min upload speed, and all the features that cool distributed network programs have.

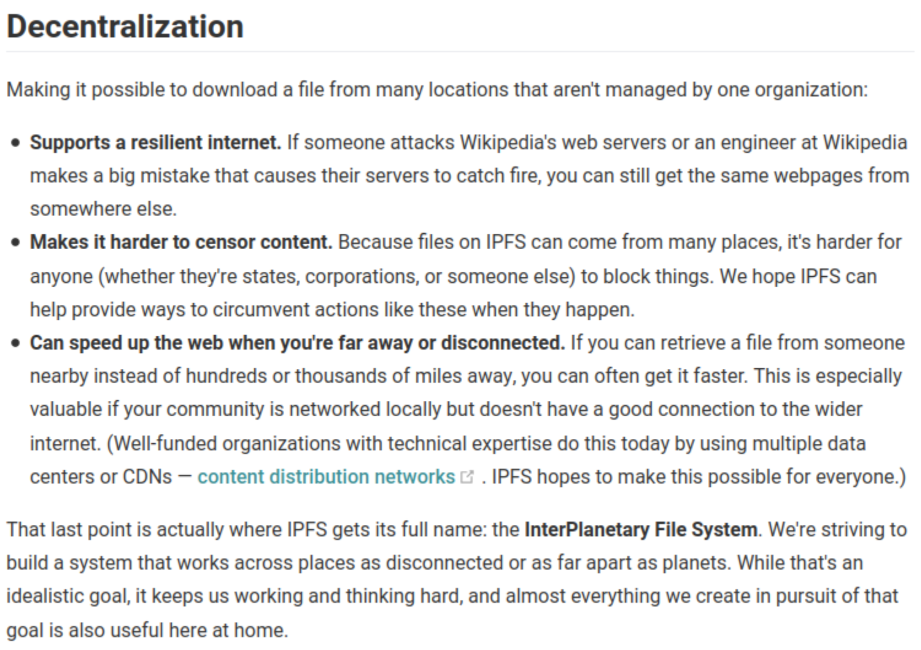
Of course, everything as a price and not everything is perfect. So to implement IPFS into InstantOS we may need some important things: 1. Participation. IPFS **must** be enabled by default 2. Content, but we have a lot to put there 3. Apps or scripts that make IPFS friendlier and more usable

Let’s talk a little bit more about IPFS and *how it works.* There are 3 fundamental principles to understanding IPFS: * Unique identification via content addressing * Content linking via directed acyclic graphs *DAGs* * Content discovery via distributed hash tables *DHTs*

Content Addressing (A.K.A. what is the content of the book, given a title name?): on the internet we usually look at links like *https://instantos.io/*, that shows us the website with the name instantos.io. That link is a resource, a file on some server. On IPFS content addressing is made with *CID* or Content Identifiers, hash(es) of files. instantos iso is *QmU4mP3VhqSB9R4WGyNAAj7Vg4RZTZpBZgnMWRwm7dC63D*, easy.

I’m not going into details about DAGs and DHTs, but bear in mind (Merkle) DAGs are used for representing IPFS directories and files while DHTs are used to tell where the content is stored in the network (trust me, it’s pretty hard to get healthy files in a giant network!). Since this is a really good system, where are the flaws I was talking about? Let’s talk about them.

1. By default, IPFS creates **a lot** of garbage. Since the primary intent is to



Decentralization

Making it possible to download a file from many locations that aren't managed by one organization:

- **Supports a resilient internet.** If someone attacks Wikipedia's web servers or an engineer at Wikipedia makes a big mistake that causes their servers to catch fire, you can still get the same webpages from somewhere else.
- **Makes it harder to censor content.** Because files on IPFS can come from many places, it's harder for anyone (whether they're states, corporations, or someone else) to block things. We hope IPFS can help provide ways to circumvent actions like these when they happen.
- **Can speed up the web when you're far away or disconnected.** If you can retrieve a file from someone nearby instead of hundreds or thousands of miles away, you can often get it faster. This is especially valuable if your community is networked locally but doesn't have a good connection to the wider internet. (Well-funded organizations with technical expertise do this today by using multiple data centers or CDNs — content distribution networks . IPFS hopes to make this possible for everyone.)

That last point is actually where IPFS gets its full name: the **InterPlanetary File System**. We're striving to build a system that works across places as disconnected or as far apart as planets. While that's an idealistic goal, it keeps us working and thinking hard, and almost everything we create in pursuit of that goal is also useful here at home.

Figure 1: A quick rundown of everything cool about IPFS

store things that are useful for the network, we usually don't want garbage (like IRL). IPFS has a really good garbage collection process, but it's not enabled by default for really good reasons (more on this later)
2. The current IPFS implementation is in go. And I don't really love it. It's not slow or resource hungry, but it can be waaay better.
3. It's pretty hard to upload and use on the net dynamic content, but *not* impossible (Gemini-like problems, you know what I mean)

Part 2: why should we use IPFS for InstantOS?

Centralized servers cost money. The more data you have, the more money you will have to pay. If you want *fast* data, you might want SSD Servers and databases, and that costs *more money* that the community usually don't want to pay.

IPFS offers free and distributed storage, really good libraries to build fast applications in a lot of programming languages and such. Aaaand.. that's it?

Of course not, but here's the interesting part. You can use IPFS for everything, with good applications that already do it! Here's a list of what can we do with it:

- Send and receive files (duh)

- Version Control (via IPVC, similar to git)
- Exchange messages (see berty)
- Host videos (DTube already uses IPFS)
- And other crazy things!

Part 3: how should we use it

1. Go-ipfs enabled by default with the –enable-gc flag to prevent users from seeing all their data not garbage-collected
2. Use IPFS for host and for the *latest* ISO. The first thing can be pinned (more on pinning here)
3. Implement an option in InstantASSISTANT to send a file via IPFS and copy the gateway link to the clipboard with one click only
4. Implement an easy *IPFS settings* under the settings tab that lets you control all the juicy things easily
5. Instant Repository on IPFS, with installed packages pinned by default and unpinned when the package is updated (unpin the old, pin the latest)
6. Use a pinning service only at first, then switch to the community IPFS when things are settled well

There is a broad *how-to* here that explains clearly how to implement lots of things I just wrote about.

Hope you have a great day.