The ipfs

InterPlanetary Filesystem

Preamble

Last week, dreaming about being able to serve pars images using something like the BitTorrent protocol from the browser. I asked Jeeves: "Can browsers communicate peer-to-peer?" Turns out they can through WebRTC, released by Harald Alvestrand ("on behalf" of Google) in 2011. The WebRTC protocol is supported by some major browsers but, disappointingly, interoperability is only possible through a polyfill and neither Safari or IE have native support in any form. It would be possible to implement something like a torrent client in JavaScript that would run in the browser. As the wise programmer once said, everything you can think of has already been implemented in JavaScript, and sure enough someone has already implemented an in-browser BitTorrent client, called webtorrent.

So that could have gone somewhere, until I was on Dan O'Huiginn's blog and read his overview of the InterPlanetary Filesystem.

It kind of sounded like something I wanted.

Introduction

IPFS is being designed primarily by a Stanford CS dude (Juan Batiz-Benet), who made a company which sold to Yahoo as an "acqui-hire" last year. The protocol does not appear groundbreaking

in itself, on the face of it it's a pretty simple idea. It describes something like a single, global Git repo that supports peer-to-peer protocols as well as SSH and HTTP. You can get the data you want by cloning the relevant part of the DAG through BitTorrent (like a shallow clone in Git). The parts of the repo you have locally would be seeded to others in the network who requested them, following similar choking rules to BitTorrent.

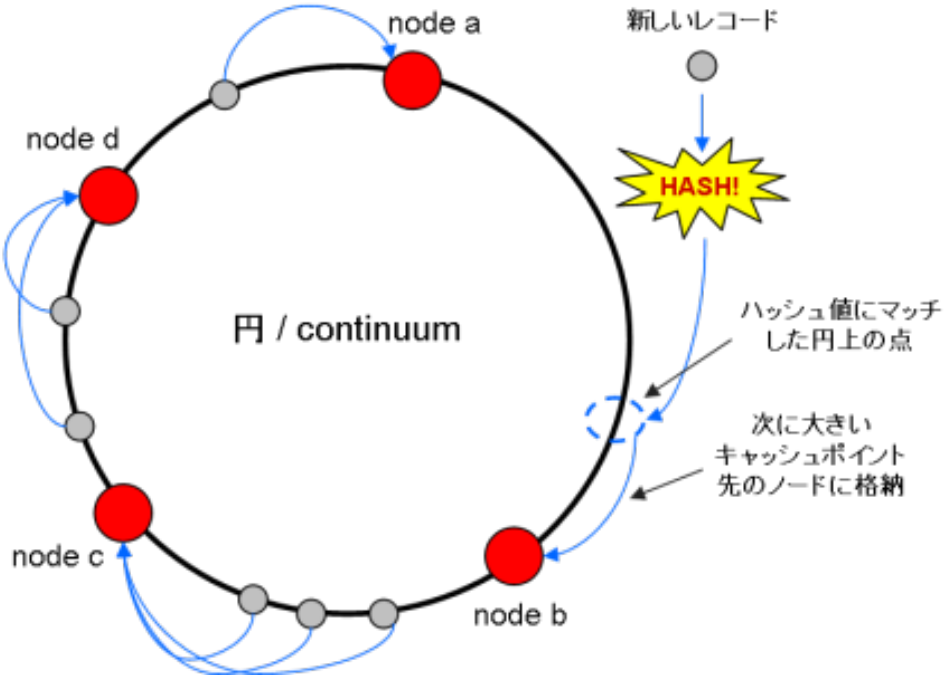
It's a great idea because it builds on things people are already familiar with that also have mature and and widely successful implementations as well as introducing some novel ideas (see Mutable namespaces) and wraps the whole lot into a pretty well cohesive whole. The clarity of vision is impressive and ambitious, yes sir.

At least one unrelated thrust (GTP) has already been made in a similar direction, but the project is smaller in scope and seems to be unmaintained.

IPFS also boasts WIP implementations in hipster[1] languages like Golang and Node.js, someone is even working[2] on a Haskell implementation. What's not to like?

However, it's not going to happen tomorrow; IPFS is still just a cool idea. There are no clients, no developer toolchain and of course no native browser support. That said, let's look at what IPFS is made of and imagine some applications of being able "mount the world at `/ipfs`".

Kademlia DHT



Yes, it's that exciting (source).

One routing mechanism IPFS proposes to use is the "distributed sloppy hash table" employed by BitTorrent. The spec also states that the routing layer should be "swappable", meaning more traditional (or more exotic) routing could be used in place of a DHT. The DHT named in the spec is Kademlia[3], intended as a successor to CHORD[4]. It has nice properties for high-churn applications[5]; we're all guilty of shutting down µTorrent as soon as that latest Linux distro has finished downloading. Kademlia's design is, in part, informed by analysis of data collected from the Gnutella network[6]. Remember LimeWire and BearShare? They ran on Gnutella.

Mutable namespaces

Aside from borrowing ideas from successful applications of DAGs and DHTs, the spec has a novel take on the URL. Novel, but apparently just an idea borrowed from SFS, designed for his doctoral thesis in 2000 by David Mazières.

In IPFS, files are addressed by the cryptographic hash of their content and meta data, like objects in Git, rather than a file path or web address decided by a human, the content-hash becomes a file's "name". This is convenient for programmatically addressing

files, but supremely un-human-readable.

On the internet, we rely heavily on the same address refering to different things at different times. For example, consider the domain `news.com`. When we request that content at that address, we would probably expect to find the lastest news. This would not be possible if we were using a content-addressed system because the content of `news.com` (and therefore its address) would change every time an event was reported.

The IPFS would interface with DNS to offer domain names and web addresses, or in the context of a content-addressed system; mutable namespaces. These would operate something like a signed ref (tag or branch) in Git, addressed on a DHT[7] via your public-key. Basically, everyone would get a namespace rooted in their key pair, which could be mapped (somehow) to a "proper" domain name in a DNS record.

In the analogy of the "single global Git repo", this would solve the problems of someone pushing with `--force` on to `master`, everyone wanting a branch called `dev` as well as making it possible to offer new news on `news.com`.

Trust here would be provided by PGP, which I guess is pretty good :wink:

Security'n'Privacy

The security of a system such as IPFS presents different problems to traditional web security. The normal scenario would be that the trusted DNS server gives me the IP for the domain I request, I setup a connection to that trusted IP address over HTTP/TLS. As long as the box answering to that IP is secure (lol, Sony) and I trust the owner of that box means me no harm then I can safely transfer files in good faith that the content will be what I expect.

This is not the case with peer-to-peer networks. It is not feasible to categorically say that we trust all the nodes in the swarm. There are things that help (see the Merkle tree section in a related post), but it boils down to PGP-signing.

On the other hand, a peer-to-peer system is much less vunerable to many types of attack than a centralised system. Kademlia has some built in stuff for dealing with attackers flooding the cluster with new nodes.

Conclusion/conjecture

Expressed in the most general terms, IPFS proposes an other web. It's a very different place to the web we know today. Instead of all web traffic being mediated by trusted central servers and services and these entities being the things that people really look at when they use the web, users of the internet would have the opportunity to communicate more directly with each other, not relying on a third party to securely transmit their messages, documents, programmes and images.

The theme of decentralised systems and decentralisation has caught the attention of the zeitgeist and if the impact of widespread adoption (after the development works are underway) of a public, global, "interplanetary" filesystem can be communicated to computer users as a whole, the IPFS could become a reality and some pretty exciting stuff would be able to follow.

[1] Read "young".

[2] Well, there's a GitHub issue that says someone expressed an interest at https://github.com/jbenet/ipfs/issues/4

[3] "Kademlia: A Peer-to-peer Information System Based on the XOR Metric" Petar Maymounkov, David Mazières (2002) http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

[4] "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications" Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan (2001) http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

[5] http://michaelnielsen.org/blog/consistent-hashing/

[6] "A Measurement Study of Peer-to-Peer File Sharing Systems" Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble (2001) http://research.microsoft.com/en-us/um/people/ssaroiu/publications/tr/uw/2001/uw-cse-01-06-02.pdf

[7] http://en.wikipedia.org/wiki/Hash_function#Uniformity

[8] Probably a dedicated "namespaces" DHT that would store named pointers to objects in the "content" DHT.