Radoslaw Krzeski
Summary of Topic XII – Advanced Blockchain Storage

IPFS – Content Addresses, Versioned, P2P File System (Draft 3)
The project with a very ambitious name – The InterPlanetary File System is a peer-to-peer distributed system which wants to connect all devices to the same system of files. Described as something between the Web and a BitTorrent swarm, a block-storage models for files and hyperlinks. The system uses a Merkle DAG (Directed Acyclic Graph), a version of Merkle - trees for storage, combining it with a distributed hashtable, incentivized block exchange and a self-certifying namespace. The summary also mentions that the system has no single point of failure and no nodes need to trust the other.

Many similar attempts on a globally distributed files systems have been tried before, such as the Academic AFS(Andrew File System), along with p2p systems such as Napster, KaZaa and BitTorrent. It is said however that "no-general fil-system has emerged that offers global, low-latency and a decentralized distribution"- 'seeing the HTTP protocol as getting the closest to this goal.
Introducing new file storage protocol, backwards compatibility and risk of degrading user experience are amongst the issue that surface when altering the current generation.
New challenges have emerged; transfers of petabyte datasets, large-data computing, high-volume streams, versioning and linking of datasets and other security measures. Git has already implemented some changes on protocol-level, and uses the Merkle DAG system that IPFS aims to use.

A main aspect of the systems is mentioned by successful examples:
- Distrubted Hash Tables (Kademlia DHT, Coral DST(with cluster lookup), S/Kademlia DHT(increased security)
- Block Exchanges (BitTorrent – tit-for-tat ability, priority pieces, swarm performance in PropShare)
- Version Control Systems (Git)
- Self-certified filesystems (distributed trust chain & egalitarian shared global namespaces)
    - Entry point scheme: /sfs/<Location>:<HostID>

IPFS takes these parts and provide and aim to provide a greater sum than the individual parts. No nodes are privileged in the system, and nodes store IPFS object in local storage. A few sub-protocols are mentioned that leverage blended properties:
- **Identities** – node identity and verification (uses NodeId as PK, achieves node benefits with age)
- **Network** – connection management protocols for Transport, Reliability, Connectivity, Integrity, Authenticity (does not assume IP!)
- **Routing** – information to locate peers and serve objects (<1KB)
- **Exchange** – novel block exchange protocol (BitSwap client, takes blocks from any file system, distributed mutual block agreements, Bitswap Credit for pure leechers, debt ratios and ignore_cooldown ability)
- **Objects** - immutable objects with links (file hierarchies and communication systems along with object "being cryptographic hashes of the targets embedded in the sources".
  Has two requirements: content addressed, and encoded in Merkle DAG).
- **Files** – versioned file system, Git-inspired (block/blob, list, tree, commit) (flattened tree, tree caching (indefinitely)
- **Naming** - self-certifying mutable system.

Summarized the system is a mounted global filesystem, that can sync and publish encrypted files or systems. It is a version manager, potentially being used a virtual machine, database, communication platform, Permanent WEB (since objects are permanent and as an extension library. Information will be shared globally, decentralized and with those actors interested in the content.

As an extension of this article there is a discussion going at this site that considers some of the storage capacity of the IPFS system: https://github.com/ipfs/faq/issues/47

It is mentioned that while content is permanent to a degree, it can be removed by agreement form multiple parties; "users and groups can express what content should or should not be stored and/or distributed".
Data related to these individual groups is only stored amongst the nodes interested in the content. Content policies will also regulate how it is distributed and broadcasted in the future amongst other nodes.
FileCoin is also mentioned as a tool to pay networks or nodes to keep content live for you. Encryption will also be added on the fly to add an additional security layer to the system.

A Secure Sharding Protocol For Open Blockchains
The article begins by mentioning the different blockchain currencies, and how they have challenges with security and large-scale consensus models. It then mentions what it calls ELASTICO. It scales transactions proportionally with computational power (higher number of transaction blocks selected per unit of time.). Its BFT tolerance is up to a fourth of the computational power available in the network.

It partitions files into smaller chunks, uniformly, and paralyses(simultaneously) the network into committees, which each creating a “disjoint set of transactions or “shards””. It argues that it is the first sharding protocol resistant to byzantine adversaries and being scalable. Each committee should have a reasonable small number (few hundreds) of members, and these committees scale with network power.

ELASTICO “decouples the consensus from block-data broadcast”, efficiently stabilizing bandwidth spend at each node. “The efficiency property represents the sharding advantage, where the cost is localized within a committee. Once the network agrees on the set X , it can create a cryptographic digest of X and form a hash-chain with previous agreed sets in the previous runs of Π. This serves as a distributed ledger of facts or transactions.” (Page 2 in the paper)

Challenges in the system are securing identities before sharing information in-between nodes. The algorithm in the system proceeds in epochs. Key idea is to automatically parallelize the available computational power, dividing in the several small committees, each processing a disjointed set of transactions or shards. All committees run a byzantine consensus protocol internally to agree on one value. A final committee connects the shards from other committees and broadcasts a cryptographic digest. The last epoch stage the final committee generate a set of shared public random bit string (used as a source of randomness). Used to reduce insight for adversaries and gains in computational power over others.

Processors execute 5 steps:

1- Identity Establishment and Committee Formation
2- Overlay Setup for Committees
3- Intra-Committee Consensus
4- Final Consensus Broadcast
5- Epoch Randomness Generation

(Skipped section 4 as indicated about Security.)

In terms of network implementation, it us using a peer-to peer basis, rather than point to point. This is because a node requires much more resources to open socket connections to tenfolds of nodes. It allows scaling up the underlying blockchain protocol without degrading any security property.

The paper is difficult to understand, and I`m afraid the complexity of this sharding protocol is too complex to attract investors and software integrations early on.

Keywords: blockchain agreement problem. Sharding already uses in cloud software, and secure internal networks. Expected 10.000`s of numbers of values before each epoch.

References:

IPFS white paper
https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

Selected info and responses to questions in
https://github.com/ipfs/faq/issues/47

A Secure Sharding Protocol For Open Blockchains
Luu, Loi and Narayanan, Viswesh and Zheng, Chaodong and Baweja, Kunal and Gilbert, Seth and Saxena, Prateek
https://dl.acm.org/citation.cfm?id=2978389