Master's Thesis Proposal

David Grisham

Introduction

IPFS

The InterPlanetary File System (IPFS) is a peer-to-peer hypermedia distribution protocol developed by Protocol Labs. It is a content-addressed, versioned filesystem. While a variety of use cases exist for such a protocol, the most ambitious goal of the project is to replace HTTP as the primary file exchange protocol used in the Internet. This could ultimately result in the decentralization of the Internet.

IPFS synthesizes various technologies developed since the Internet's inception. These technologies include Git, BitTorrent, distributed hash tables (e.g. Kademlia), and self-certified filesystems. The IPFS stack is shown in Figure 1. One way to conceptualize an IPFS network is as a Git repository shared within a torrent-esque swarm.

Bitswap

Bitswap is the block exchange protocol of IPFS. The most direct inspiration of this submodule is the BitTorrent peer-to-peer file distribution protocol. Bitswap is the layer of IPFS that incentivizes users to share data. A Bitswap *strategy function* determines which peers to send data to, and in what relative quantities. The input to the strategy function is a set of metrics that may be used to weight peers – e.g. peer bandwidth, reputation, and/or location. The output is a set of weights, one for each peer, that assign the relative resource allocations for the peers. These weights are periodically recalculated to reflect changes in both the network and peer behavior.

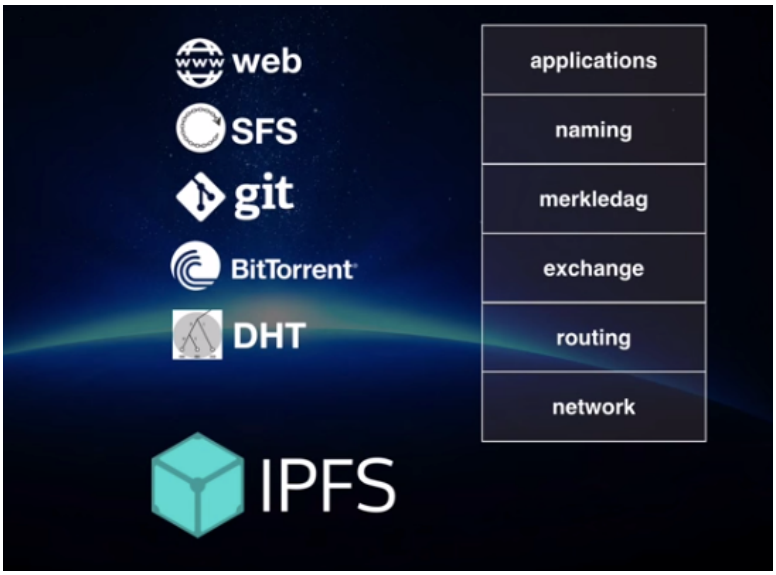

Figure 1: **The IPFS Stack** – Bitswap is at the exchange layer.

Objectives

When considering peer reputations, an ideal Bitswap strategy function would reward long-lasting, healthy relationships and punish defectors. In order to find functions with this quality, we turn to game-theoretical modeling and simulations. Game theory may be used to analyze Bitswap strategies for a very small number of peers. However, as the size of the network increases, the complexity explodes and an analytical approach quickly becomes infesaible. Simulations will be leveraged to help glean empirical insights into the larger-scale system dynamics, which will be compared with the small-scale theoretical analysis. Further, as the simulations take network effects such as bandwidth, jitter, and packet loss into account, they reveal important practical details that would otherwise be missed by a purely theoretical analysis.

The objectives of this project are:

- Extend game-theoretical analysis in the following ways:
  1. **Repeated game analysis**: Extend preliminary analysis by increasing the number of players in the analysis. Characterize how results with small number of nodes generalizes to larger populations.
  2. **Evolutionary game theory**: If time allows, model the Bitswap game using evolutionary game theory to give a more sophisticated analysis of the large-scale dynamics.
- Create an run simulation testbed for Bitswap strategy analysis, with the following goals:

1. Simulate cases that reflect the theoretical analysis and compare results under various network conditions.
2. Implement developer-friendly API for configuring Bitswap strategies for future testing.

Progress

Theory

On the theory side, a specific formulation of the two player Bitswap game has been analyzed against the well-studied tit-for-tat (TFT) and grim trigger (GT) strategies[1]. In summary, the results showed that neither TFT nor GT were subgame perfect Nash equilibria for this game. This analysis will be expanded in the following ways:

1. **Larger network sizes** – This is crucial as the peers are weighted based on their relative behaviors, so the results should be very different from the two player game.
2. **Persistent reputations** – The initial analysis only considered a peers' behavior in the immediately preceding round, as opposed to taking the entire history of the peer-wise interaction into account.
3. **Varying the strategy function** – Strategy functions beyond the simple linear function used in the initial analysis should be considered. This becomes particularly important when the previous two points are applied.

Implementation

On the implementation side, I have added the ability to specify a Bitswap strategy in `go-ipfs`[2]. `go-ipfs` is the primary, reference implementation of IPFS (written in the Go programming language) and the one that I will be using in my research. The Bitswap strategy is simply a function that takes a peer's reputation as input and produces a weight for that peer. These relative weights are then used in a single round of a weighted round robin queue, which distributes data to each of the peers in these relative quantities.

The next step is to set up a simulation testbed based in the InterPlanetary TestBed[3] (IPTB). IPTB allows the intialization and control of IPFS nodes within docker containers. Network parameters such as bandwidth and jitter may be configured between the containers and, since the containers are hosted on the same machine, this provides a means of simulating IPFS nodes in a more stable environment than actual networks provide.

[1] This preliminary analysis is included as a separate document.
[2] https://github.com/ipfs/go-ipfs
[3] https://github.com/whyrusleeping/iptb

IPTB will serve two purposes:

1. Give a means of comparing my round-robin implementation of the Bitswap peer queue with the original peer queue. This will help verify that the new implementation performs at least as well as the old so that we can confidently integrate it into `go-ipfs`.
2. Provide a highly configurable and isolated testbed for initial simulation purposes. The game-theoretical results will be compared to the interactions between nodes using IPTB.

Finally, while the IPTB makes simulations simpler to run and understand, we ultimately want to implement this research in real networks. The simulations will be expanded to test IPFS nodes running on separate hosts where the unpredicability of network effects will become relevant. Currently, the best option for this is to use `kubernetes-ipfs`[4]. `kubernetes-ipfs` is a tool which allows the user to set up and control IPFS nodes running on separate devices via a DSL. While it would be preferable to use IPTB for this step rather than switching tools, the purposes of the tools are different enough that `kubernetes-ipfs` may be the better choice for the sake of time.

Timeline

[4] https://github.com/ipfs/kubernetes-ipfs/