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:

- **Simulations**
  1. **Strategy simulator**: Continue to build on the existing `strategy simulator`[1] to empirically analyze strategies – e.g. whether a particular strategy might be a Nash equilibrium in certain cases.
  2. **Bitswap tests**: Continue work on writing tests that coordinate interactions between IPFS nodes and measure the resulting dynamics. These IPFS nodes run in Docker containers, so network conditions

[1] https://github.com/dgrisham/strategy-sim

may be simulated by configuring the containers' network interfaces. *If time allows*, these tests will be extended to run on a network of separate machines so that actual network conditions (rather than simulated) may be tested. This work leverages IPTB[2] and is currently maintained in the `bitswap-tests` repository[3].

- **Analytical work**
  1. **Repeated game analysis**: Characterize smaller-scale systems analytically. This can be used to prove whether a particular strategy is a Nash equilibrium under certain conditions, encapsulate certain player dynamics as a function of network parameters, etc. The analytical results here reflect the simpler cases tested in the strategy simulator (discussed above).
  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.

Progress

Strategy Simulator

The strategy simulator currently supports various configuration options for testing whether a particular Bitswap strategy is a Nash equilibrium. Please refer to the `strategy-sim` Github repository for details on the purpose, options, and output of the simulator.

Go-IPFS and IPTB

I have added the ability to specify a Bitswap strategy in `go-ipfs`[4]. `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 (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

[2] https://github.com/ipfs/iptb
[3] https://github.com/dgrisham/bitswap-tests
[4] https://github.com/ipfs/go-ipfs

same machine, this provides a means of simulating IPFS nodes in a more stable environment than actual networks provide.

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.

The initial steps toward using IPTB for Bitswap testing are catalogued in the `bitswap-tests` repository.

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`[5]. `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

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