Game Theoretical Analysis of Resource Allocation in the InterPlanetary File System

David Grisham

1 Introduction

The Internet is perhaps the largest and most consistent network that has ever existed. Unfortunately, it predominantly runs on the outdated hypermedia distribution protocol HTTP. The goal of the InterPlanetary File System (IPFS) is to upgrade the Internet to a distributed peer-to-peer system, thereby making it more robust and permanent. This new Internet would be a network of peers, as opposed to clients and servers, all sharing data between one another. In order for such a system to thrive, users must be cooperative and willing to share data with their peers. The goal of this project is to analyze the resource allocation strategy space of peers interacting in an IPFS network. A combination of analytical and empirical methods will be used to glean insights into the generally intractable strategy space that users are presented with when participating in an IPFS network.

1.1 IPFS

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.

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

1.2 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 reputation 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.

1.3 Objectives

For this project, I will take the initial steps toward understanding the behavior of users in an IPFS network as predicted by game theoretical models. This will involve a combination of analytical and empirical approaches. The analytical work will focus on repeated games and, potentially, evolutionary games, while the empirical work will take a simulation-based approach. I intend to use these methods to classify various Bitswap strategy functions and determine useful strategies under certain conditions.

2 Plan

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 infeasible. 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:

- **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.
- **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 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].

[1] https://github.com/dgrisham/strategy-sim
[2] https://github.com/ipfs/iptb
[3] https://github.com/dgrisham/bitswap-tests

3 System Model

This section details the model currently used to describe Bitswap in this work. This model is the product of multiple iterations that approached a balance between the accuracy of the model to the problem and model complexity. Previous modeling approaches included tools from evolutionary game theory and statistical mechanics on the high complexity end, and repeated games on the low accuracy end. While the current model uses a repeated game model as well, the strategy space has been modified to better fit the Bitswap scenario.[4]

3.1 Network Graph

We model an IPFS swarm as a network $\mathcal{N}$ of $|\mathcal{N}|$ users. The graphical representation consists of

- *nodes* representing users, and
- *unweighted, undirected edges*, each of which represents a peering between two users.

A user's *neighborhood* is their set of peers, i.e. the set of nodes that the user is connected to by an edge. User i's neighborhood is denoted by $\mathcal{N}_i \subseteq \mathcal{N}$.

3.2 Reputation

We break Bitswap interactions into discrete rounds, with a single round denoted by a nonnegative integer t. The following two points describe the way data distribution takes place in this Bitswap model. Each of these points simplifies the problem from the real-world scenario.

1. Each user distributes exactly B bits, where $B > 0$, to each of their peers in a given round (and has sufficient resources to do so).
2. All users always have unique data that all of their peers want. So, when a user allocates b bits to a particular peer, that user has at least b bits that the peer wants.

We define b_{ij}^t as the total number of bits sent from user i to peer j from round 0 to $t-1$. Then we can define the *debt ratio* d_{ji} from user i to peer j as

$$d_{ji}^t = \frac{b_{ij}^{t-1}}{b_{ji}^{t-1} + 1}$$

d_{ji}^t can be thought of as peer i's reputation from the perspective of user j. This reputation is then considered by user j's *reputation function* $S_j(d_{ji}^t, \mathbf{d}_j^{-i,t}) \in$

[4]A description of the model from the previous iteration https://github.com/dgrisham/masters/tree/master/deprecated/analysis.

$\{0, 1\}$, where $\mathbf{d}_j^{-i,t} = (d_{jk}^t \mid \forall k \in \mathcal{N}_j, k \neq i)$ is the vector of debt ratios for all of user j's peers in round t *excluding* peer i. The reputation function considers the relative reputation of peer i to the rest of j's peers, and returns a weight for peer i. This weight is used to determine what proportion of j's resources to allocate to peer i in round t. Given all of this, we can calculate b_{ij}^{t+1} given $\mathbf{d}_j^t$:

$$b_{ji}^{t+1} = \frac{S_j(d_{ji}^t, \mathbf{d}_j^{-i,t})}{\sum_{k \in \mathcal{N}_j} S_j(d_{jk}^t, \mathbf{d}_j^{-k,t})} B$$

3.3 Game Formulation

All users in the network participate in the Bitswap game. The players in this game are the users in the IPFS network, and each player's strategy is the reciprocation function they choose to assign weights to their peers. The Bitswap game is an *infinitely repeated*, *incomplete information* game in which users exchange data. One game takes place between each pair of peers for each round t. The *players* are the IPFS users in the network $\mathcal{N}$. The utility of player i in round t is the sum of all of the data that i is sent by its peers in that round:

$$u_i^t = \sum_{j \in \mathcal{N}_i} b_{ji}^t$$

4 Preliminary Results

This section details the results obtained so far. The system model is described first (§3), followed by the simulator (§4.1), then the analytical work (§4.2), and finally the implementation progress (§4.3).

4.1 Strategy Simulator

The strategy simulator is a Python tool to empirically test whether a Bitswap reciprocation function is a Nash equilibrium under a particular set of conditions. As of now, the simulation is restricted to a 3 node network. All users are connected to all other users (i.e. it is a fully connected graph). The simulation parameters are:

- A Bitswap reciprocation function to test, `rf`. Currently, the following reciprocation functions are supported (descriptions are provided as Python anonymous functions – the `exp` and `tanh` functions are from the `numpy` library):
  - Linear: `lambda x: x`

    - Sigmoid: `lambda x: 1 / (1 + exp(1-x))`
    - Tanh: `lambda x: tanh(x)`
- An initial set of ledgers, `initial_ledgers`.
- A set of resources, `resources`, where `resources[i]` gives the total amount of data `i` will send its peers in each round.
- The resolution of the deviation, `deviation`, whose purpose is described below.

4.1.1 Simulation Description

We enumerate the users 0, 1, and 2. The simulation takes place over a single 'non-deviating' run, followed by one or more 'deviating' runs. The non-deviating run is where peer 0 follows the allocation determined by `rf`. In the deviating runs, peer 0 tries other allocations to see if it can achieve a better payoff than the non-deviating case. These runs are described next.

Non-deviating run: Each user allocates data to their peers for round `t=0` based on `rf` and `initial_ledgers`. Then, we calculate the amount of data peer 0 receives in the following round, `t=1`. This amount of data is 0's *payoff*.

Deviating runs: Users `1` and `2` allocate data to their peers for round `t=0` based on `rf` and `initial_ledgers`, while user 0 tries every possible allocation, to a resolution of `deviation` (one allocation per run). So, if `deviation` is 1 and `resources[0] = 10`, user 0 will try the allocations `(0, 10)`, `(1, 9)`, `(2, 8)`, ..., `(10, 0)`, where the first element of each tuple represents the amount of data 0 sends to 1 and the second element is the amount 0 sends to 2. For each of these allocations, we calculate the payoff (amount of data) 0 gets in round `t=1` as a result of the allocation.

TODO: example plot

TODO: summarized results

4.2 Analytical

The analytical work done verifies a subset of the empirical results from the strategy simulator.

TODO: explain

4.3 Go-IPFS and IPTB

I have added the ability to specify a Bitswap strategy in `go-ipfs`[5]. `go-ipfs` is the primary, reference implementation of IPFS (written in the Go programming

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

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 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`[6]. `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.

5 Timeline

5.1 April

5.1.1 Week 3 (4/16 - 4/20)

- Analytical results (via Mathematica notebook)
  - Characterize results
- Start thesis layout

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

- IPTB simulations
  - Preparation, figure out important test cases

5.1.2 Week 4 (4/23 - 4/27)

- Thesis layout
- Analytical work
  - Figure out where to go based on results so far
- IPTB simulations
  - Preparation/running
  - Start working on visualizations for results

5.2 May

5.2.1 Week 1 (4/30 - 5/4)

- IPTB simulations
  - Re-evaluate progress and important test cases
  - Visualizations
- Analytical work
  - Continue trajectory from previous week
- Start writing thesis
  - Intro/background – IPFS, Bitswap, motivation

5.2.2 Week 2 (5/7 - 5/11)

- IPTB simulations
- Analytical work
  - Figure out where to go based on results so far
- Thesis writing
  - Intro/background – approach, why Bitswap analysis with game theory is hard

5.2.3 Week 3 (5/14 - 5/18)

- IPTB simulations
  - Re-evaluate progress and important test cases
- Analytical work
  - Continue trajectory from previous week
- Thesis writing
  - Explain analytical results, use graphs from `strategy-sim` if helpful

5.2.4 Week 4 (5/23 - 5/25)

- IPTB simulations
- Analytical work
  - Continue trajectory from previous week
- Thesis writing
  - Explain analytical results, use graphs from `strategy-sim` if helpful
  - Start describing IPTB simulation testbed

5.2.5 Week 5 (5/28 - 6/1)

- IPTB simulations
  - Re-evaluate progress and important test cases
- Analytical work
- Thesis writing
  - Explain analytical results, use graphs from `strategy-sim` if helpful
  - Start describing IPTB simulation testbed

5.3 June

- Continue IPTB simulations
  - Re-evaluate current state periodically
- Analytical work
  - Loop on re-orienting trajectory + moving forward
- Thesis writing
  - Characterize results as they come
  - Plots and visualizations
  - Go to writing center to review formatting

5.4 July

- Analytical work should be done by this point
- Finish up simulations
- Primary focus: thesis
  - Visualizations, formatting, explaining results, etc.
- Tuesday, July 24$^{\text{th}}$: Thesis submission due