CS 591, Lecture 16

Jared Saia
University of New Mexico

Class Outline

- Presentations
- The Free Rider Problem
- Informers
- Using Informers to Restrain Free Riders

Class Presentations

- Each student will give a 30 minute presentation
- By October 29th (today), you must speak to me (in office hours or by appointment) about your proposed presentation.
- Aaron - Max Cut in Random Graphs, Vishal - Tutte's Thm?, Jun - Online Machine Learning
- By November 5th, you must turn in a 1 page detailed summary of what you will present. Give a description of the abstract problem, the algorithm(s), and all major theorems (you should try to prove no more than 1 or 2 theorems).
- Class Presentations will be given on November 19, 21 and 26th - choose a date now.

Class Presentations (II)

- The presentation should be on an algorithmic approach to a "real-world" problem or on a mathematical tool that is useful for real-world algorithms
- There should be some "mathematical meat" to your presentation. In particular, there should be some use of mathematical tools like the ones we learned in class (e.g. graph theory, probability theory, online analysis, etc.)
- It's ok to present in groups of up to 3 people
- Find something that interests you!!!

Presentation Signup

- 11/19 Vinay and Jijun
- 11/21 Ravi and Vishal
- 11/26 Jun and Aaron

“Free Rider” Problem

The “Three” ruled with the help of an elaborate network of agents and informers. The city was divided up, with a set number of informers for each ward and parish - Donald Phau in “Venice: The Methodology of Evil”

- The problem: how can we enforce good social behavior for peers?
- Question: What is “good social behavior”?
- More generally: how can we maintain “law and order” on the p2p frontier?

What is “good social behavior”?

- “Good social behavior” varies by context
- In a large p2p system for sharing music, may only need small number of peers to contribute content and resources
- In a more specialized p2p system (e.g. sharing genetic sequencing data among researchers), may want all of the peers to share some of their resources

What can we hope for?

- Recall that the peer-to-peer graph is sparsely connected - typically no more than $O(\log n)$ neighbors per peer
- Hence any single peer, can collude with all its neighbors and get away with bad social behavior
- We can really only hope to enforce the law for “most” of the peers

Informers

- A key idea: Let the peers "inform" on each other
- Each peer, in addition to playing its normal roles in the network, also plays the role of informing on a fixed number of other peers
- All communication that any peer has with the p2p network first passes through the informers for that peer.

Informers

- Some of the informers will be untrustworthy
- They may try to collude to let bad peers get away with things
- They may try to collude to unfairly punish good peers

Handling "Bad" Informers

We use the following tools to handle bad informers:

- Require a large number of informers - massive redundancy
- Informers are assigned "at random" to inform on someone

Random Assignment of Informers

- Since there is no centralized authority, getting a random assignment is tricky
- Improper that peers perform the random selection themselves since
- 1) “bad” informers could all assign themselves to some “bad” peer and thus protect that peer
- 2) “bad” informers could all assign themselves to some “good” peer and thus punish that peer unfairly

Unique ID

- In order to have any hope of success, each peer must have a unique, verifiable id determined by an some outside authority
- This id could be the ip-address of the peer
- Without such id's, it's possible for one bad peer to pretend that it is a very large number of separate peers and thus take over the network
- For a peer p, let i_p be the unique id for p

Random Assignment

- Let n be the total number of peers in the network
- We make use of a "random oracle" function f which maps Z to $\{0, \ldots, n\}^{\text{polylog (n)}}$
- This function essentially maps integers to random tuples of length polylog (n) over the set $\{1, \ldots, n\}$
- For a peer p with id, i_p, $f(i_p)$ will give the ids of the informers of p

Join Protocol for joining peer p

- p broadcasts message that i_p is joining to all peers in network
- Let S be the set of peers p receives replies from that claim to be p's informers. Let S' be the set of verified informer in S (i.e. $S' = \{s \in S | i_s \in f(i_p)\}$
- p sets it's informer set $I(p)$ to be S'
- p establishes connections with all peers in $I(p)$

Join Protocol for non-joining peer o

- o receives join message from p and obtains p's id, i_p
- If $i_o \in f(i_p)$, o broadcasts a claim that it is an informer for p to all other peers. It also adds p to its informant set, $I^{-1}(o)$
- o receives messages from all peers which claim to be informers for p. Let $I(p)$ be the verified set of all such peers
- Let $I^{-1}(o)$ be the set of peers that o informs on. For each $x \in I^{-1}(o)$ such that (x, p) is an edge in the underlying network, o establishes an informer-to-informer link to x

Fail-stop faults

- First consider the situation where bad peers suffer "fail-stop" faults
- Recall "fail-stop" means that bad informers simply stop communicating
- We give a very light-weight scheme for informing and communicating through the network in this case

Communication: Peer to Informer

- Peer protocol: When a peer p requests a data item or serves a data item, it sends out a message to a random informer in $I(p)$
- Informer protocol: When an informer receives a message from its peer, it does the following
  - If the message is a data item requested by a previously logged request, it logs this response and broadcasts information on the response to all other informers in $I(p)$
  - Otherwise, with probability p_i, it logs information on the message sent out and broadcasts this information to all other peers in $I(p)$
  - Finally, it sends the message to the next appropriate informer cloud using the protocol described under *Informer to Neighboring Informer Cloud*

Informer to Peer

- When an informer receives a message to its peer, it does the following:
- If the message is a request for data, with probability p_i, it logs information on the request and broadcasts this information to all other informers in the informer cloud.
- Next, it sends the message to the peer

Informer to Neighboring Informer Cloud

- When an informer has a message which is to be sent to a neighboring informer cloud, it does the following:
- Until it gets a response, it chooses a random informer in the neighboring informer cloud and pings that informer
- Once a live informer is found, the message is sent to that live informer

Analysis

- Let p_f be the ratio of the number of failed peers to the total number of peers
- Assume we want to communicate a message from some peer a to some peer b in the original network
- Let p_i be any number in the range 0 to 1
- Let m be the number of informers in each informer cloud (m is polylog in n)

Analysis

Theorem: The above protocol allows for communication along the path from a to b such that

- The message sent from a is known by all peers in $I(a)$ with probability p_i, and is known to all peers in $I(b)$ with probability p_i
- The expected time to send the message is $(1/p_f)l + 2 * p_i$
- The expected number of messages sent is $(1/p_f)l + 2 * p_i * m$
- The time to send and number of messages to send are sharply concentrated around their expected values