Gossip-based self-organizing overlay networks

Márk Jelasity

University of Szeged and Hungarian Academy of Sciences
Motivation

- Massively large scale distributed systems are now common
  - clouds, (desktop) Grid, P2P
- We want them to be efficient, cheap, available, reliable, robust
  - decentralization is often preferred
- Distributed overlay structures are needed over the nodes
  - construction, maintenance, repair
What is an overlay good for?

- multicasting (replacing IP multicast)
  - based on random or tree(ish) overlays
- distributed data structures to support data storage and lookup
  - eg, distributed hash tables
- semantic clustering to enhance keyword search
- etc
Overlay Networks

- nodes are processes with access to a computer network
  - overlays are in application layer
- links are defined by “knows-about” relation
  - virtual link, can be changed dynamically and flexibly
  - nodes use the network's transport layer (TCP/UDP) to pass messages based on target address, possibly passing many routers
An overlay network

Overlay network

View of B:
- Descriptor of A
- Descriptor of C
- Descriptor of E
Outline

• gossip protocols: the basic skeleton
• “gossip” to build structures
  – random networks
  – more structured networks such as ring, mesh, tree, clustering, sorting
• concluding remarks and open questions
A Gossip Skeleton

- the push-pull model is sown
- the active thread initiates communication (push) and receives peer state (pull)
- the passive thread mirrors this behavior

\[
\text{do once in each T time units at a random time:}
\]
\[
p = \text{selectPeer()}
\]
\[
\text{send state to } p
\]
\[
\text{receive state}_p \text{ from } p
\]
\[
\text{state} = \text{update}(\text{state}_p)
\]
\[
\text{active thread}
\]

\[
\text{do forever}
\]
\[
\text{receive state}_p \text{ from } p
\]
\[
\text{send state to } p
\]
\[
\text{state} = \text{update}(\text{state}_p)
\]
\[
\text{passive thread}
\]
Rumor mongering as an instance

- **state**: set of active updates
- **selectPeer**: a random peer from the network
  - very important component, we get back to this soon
- **update**: add the received updates to the local set of updates
- propagation of one given update can be limited (max k times or with some probability, etc)
Peer Sampling and random overlays

- A key method is selectPeer in all gossip protocols (influences performance and reliability)
- In earliest works all nodes had a global view to select a random peer from
  - scalability and dynamism problems
- We use a random overlay, and we maintain it through gossip
  - random overlay has many other applications
Gossip based peer sampling

- basic idea: random peer samples are provided by a gossip algorithm: the peer sampling service
- The peer sampling service uses itself as peer sampling service
- state: a set of random overlay links to peers
- selectPeer: select a peer from the known set of random peers
- update: (simplified) for example, keep a random subset of the union of the received and the old link set
Gossip based peer sampling

- in reality a huge number of variations exist
  - timestamps on the overlay links can be taken into account: we can select peers with newer links, or in update we can prefer links that are newer
- these variations represent important differences w.r.t. fault tolerance and the quality of samples
  - the links at all nodes define a random-like overlay that can have different properties (degree distribution, clustering, diameter, etc)
  - turns out actually not really random, but still good for gossip
newscast: going for new information

- **update**: keep freshest links
- **simulations**: $N=100\,000$, view size $c=30$
  - growing: start with no nodes, add 5000 nodes in each cycle, connecting them to first node only
  - lattice: start with regular lattice
  - random: start with c-out random topology
Gossip based topology management in general

- We saw we can build random networks. Can we build any network with gossip?

- Yes, many examples
  - proximity networks
  - DHT-s (Bamboo DHT: maintains Pastry structure with gossip inspired protocols)
  - semantic proximity networks
  - etc
Gossip protocols for topology management in general
Gossip protocols for topology management in general

SelectPeer
Gossip protocols for topology management in general

Exchanges of views

A

E
Gossip protocols for topology management in general

Both sides apply update thereby redefining topology
T-Man

- T-MAN is a protocol that captures many of these in a common framework, with the help of the ranking method:
  - the ranking method orders any set of nodes according to their desirability to be a neighbor of some given node
  - for example, based on hop count in a target structure (ring, tree, etc)
  - or based on more complicated criteria not expressible by any distance measure
Gossip based topology management

- **basic idea**: random peer samples are provided by a gossip algorithm: the peer sampling service
- **state**: a set of overlay links to peers
- **selectPeer**: select the peer from the known set of peers that ranks highest according to the ranking method
- **update**: keep those links that point to nodes that rank highest
- **actual algorithm contains some more tricks...**
Sorting and clustering

Sorting

Clustering
Illustration of clustering and sorting
(c) binary tree

![Graph showing convergence factor over cycles for different binary tree configurations.](image)
T-Man: summary

- T-Man generates a wide range of topologies
- the convergence is fast
  - approx. logarithmic in the number of nodes,
  - independently of the topologies we looked at
- not only approximate, but perfect embedding can be achieved
- applications include communication topology, sorting and clustering
Some thoughts

- gossip protocols apply local operators to reduce global “energy”
  - this is true for not only structures: dissemination and many other applications too

- the operators are applied in a massively parallel way resulting in quick global convergence
Some thoughts

- strong analogies with
  - control theory,
  - heuristic optimization (simulated annealing, etc),
  - parallel matrix iterations,
  - etc

- theoretically very poorly understood!
Aggregation

• Calculate a global function over distributed data
  – eg average, but more complex examples include variance, network size, model fitting, etc

• usual structured/unstructured approaches exist
  – structured: create an overlay (eg a tree) and use that to calculate the function hierarchically
  – unstructured: design a stochastic iteration algorithm that converges to what you want (gossip)

• we look at gossip here
Implementation of aggregation

- **state**: current approximation of the average
  - initially the local value held by the node

- **selectPeer**: a random peer (based on peer sampling service)

- **updateState**($s_1, s_2$)
  - $(s_1 + s_2)/2$: result in averaging
  - $(s_1 s_2)^{1/2}$: results in geometric mean
  - $\max(s_1, s_2)$: results in maximum, etc
Illustration of averaging

12

3

6

2

7

8
Illustration of averaging

\[(12 + 6)/2 = 9\]
Illustration of averaging
Illustration of averaging

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Cycle 1</th>
<th>Cycle 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle 3</th>
<th>Cycle 4</th>
<th>Cycle 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
</tbody>
</table>