Gossipping in Bologna

Ozalp Babaoglu
Background

- 2003: Márk Jelasity brings the gossipping gospel to Bologna from Amsterdam
- 2003-2006: We get good mileage from gossipping in the context of Project BISON
- 2005-present: Continue to get mileage in the context of Project DELIS
What have we done?

- We have used gossipping to obtain fast, robust, decentralized solutions for
  - Aggregation
  - Overlay topology management
  - Heartbeat synchronization
  - Cooperation in selfish environments
Collaborators

- Márk Jelasity
- Alberto Montresor
- Gianpaolo Jesi
- Toni Binci
- David Hales
- Stefano Arteconi
// active thread
do forever
    wait(T time units)
    q = SelectPeer()
    push S to q
    pull S_q from q
    S = Update(S, S_q)

// passive thread
do forever
    (p, S_p) = pull * from *
    push S to p
    S = Update(S, S_p)
Proactive gossip framework

• To instantiate the framework, need to define
  • Local state $$S$$
  • Method $${\textbf{SelectPeer}}()$$
  • Style of interaction
    ▲ push-pull
    ▲ push
    ▲ pull
  • Method $${\textbf{Update}}()$$
#1
Aggregation
Gossip framework instantiation

- Style of interaction: push-pull
- Local state $S$: Current estimate of global aggregate
- Method `SelectPeer()`: Single random neighbor
- Method `Update()`: Numerical function defined according to desired global aggregate (arithmetic/geometric mean, min, max, etc.)
Exponential convergence of averaging
Properties of gossip-based aggregation

• In gossip-based averaging, if the selected peer is a globally random sample, then the variance of the set of estimates decreases exponentially

• Convergence factor:

\[ \rho = \frac{E(\sigma_{i+1}^2)}{E(\sigma_i^2)} \approx \frac{1}{2\sqrt{e}} \approx 0.303 \]
Robustness of network size estimation

1000 nodes crash at the beginning of each cycle
Robustness of network size estimation

20% of messages are lost
#2

Topology Management
Gossip framework instantiation

- Style of interaction: push-pull
- Local state $S$: Current neighbor set
- Method `SelectPeer()`: Single random neighbor
- Method `Update()`: Ranking function defined according to desired topology (ring, mesh, torus, DHT, etc.)
Mesh Example

Mesh.mov
Sorting example

Line.mov
Exponential convergence - time

(e) $N=2^{17}$

The graph shows the number of missing target links against cycles for different network structures and connectivity parameters. The y-axis represents the number of missing target links on a logarithmic scale, ranging from $10^1$ to $10^5$. The x-axis represents the cycles, ranging from 15 to 100. Different line styles and labels indicate various network configurations, such as binary tree with different values of $c$, ring with different values of $c$, and torus with different values of $c$. The graph illustrates the exponential convergence of the number of missing target links over time.
Exponential convergence - network size

(a) ring

convergence factor

cycles

c=20, N=2^{14}
c=20, N=2^{17}
c=20, N=2^{20}
c=40, N=2^{14}
c=40, N=2^{17}
c=40, N=2^{20}
c=80, N=2^{14}
c=80, N=2^{17}
c=80, N=2^{20}
#3

Heartbeat Synchronization
Synchrony in nature

- Nature displays astonishing cases of synchrony among independent actors
  - Heart pacemaker cells
  - Chirping crickets
  - Menstrual cycle of women living together
  - Flashing of fireflies
- Actors may belong to the same organism or they may be parts of different organisms
• The “Coupled oscillator” model can be used to explain the phenomenon of “self-synchronization”
• Each actor is an independent “oscillator”, like a pendulum
• Oscillators coupled through their environment
  • Mechanical vibrations
  • Air pressure
  • Visual clues
  • Olfactory signals
• They influence each other, causing minor local adjustments that result in global synchrony
Fireflies

• Certain species of (male) fireflies (e.g., *luciola pupilla*) are known to synchronize their flashes despite:
  • Small connectivity (each firefly has a small number of “neighbors”)
  • Communication not instantaneous
  • Independent local “clocks” with random initial periods
• Style of interaction: push
• Local state $S$: Current phase of local oscillator
• Method SelectPeer(): (small) set of random neighbors
• Method Update(): Function to reset the local oscillator based on the phase of arriving flash
Experimental results

fireflies.mov
Exponential convergence

Period adjustment of ADAPTIVE model

![Graph showing exponential convergence](image-url)
#4

Cooperation in Selfish Environments
• P2P networks are usually open systems
  • Possibility to free-ride
  • High levels of free-riding can seriously degrade global performance
• A gossip-based algorithm can be used to sustain high levels of cooperation despite selfish nodes
• Based on simple “copy” and “rewire” operations
Gossip framework instantiation

- Style of interaction: pull
- Local state $S$: Current utility, strategy and neighborhood within an interaction network
- Method **SelectPeer()**: Single random sample
- Method **Update()**: Copy strategy and neighborhood if the peer is achieving better utility
SLAC Algorithm: “Copy and Rewire”

“Copy” strategy
“Rewire”

Compare utilities
“Mutate” strategy
Drop current links
Link to random node
Prisoner’s Dilemma in SLAC

- Nodes play PD with neighbors chosen randomly in the interaction network
- Only pure strategies (always $C$ or always $D$)
- Strategy mutation: flip current strategy
- Utility: average payoff achieved
Cycle 180: Small defective clusters
Cycle 220: Cooperation emerges
Cycle 230: Cooperating cluster starts to break apart
Cycle 300: Defective nodes isolated, small cooperative clusters formed
Phase transition of cooperation
• How to communicate a piece of information from a single node to all other nodes

• While:
  • Minimizing the number of messages sent \((MC)\)
  • Maximizing the percentage of nodes that receive the message \((NR)\)
  • Minimizing the elapsed time \((TR)\)
Broadcast Application

- Given a network with $N$ nodes and $L$ links
  - A spanning tree has $MC = N$
  - A flood-fill algorithm has $MC = L$
- For fixed networks containing reliable nodes, it is possible to use an initial flood-fill to build a spanning tree from any node
- Practical if broadcasting initiated by a few nodes only
- In P2P applications this is not practical due to network dynamicity and the fact that all nodes may need to broadcast
The broadcast game

- Node initiates a broadcast by sending a message to each neighbor
- Two different node behaviors determine what happens when they receive a message for the first time:
  - Pass: Forward the message to all neighbors
  - Drop: Do nothing
- Utilities are updated as follows:
  - Nodes that receive the message gain a benefit $\beta$
  - Nodes that pass the message incur a cost $\gamma$
  - Assume $\beta > \gamma > 0$, indicating nodes have an incentive to receive messages but also an incentive to not forward them
1000-node static random network
1000-node high churn network
Fixed random network

Average over 500 broadcasts x 10 runs

PASS - NR

PASS
NR
MC

network size

0
5
10
15
20

0
0.2
0.4
0.6
0.8
1
20000
50000
High churn network

Average over 500 broadcasts x 10 runs
Some food for thought

• What is it that makes a protocol “gossip based”?
  • Cyclic execution structure (whether proactive or reactive)
  • Bounded information exchange per peer, per cycle
  • Bounded number of peers per cycle
  • Random selection of peer(s)
Some food for thought

- Bounded information exchange per peer, per round implies
  - Information condensation — aggregation
- Is aggregation the mother of all gossip protocols?
Some food for thought

- Is exponential convergence a universal characterization of all gossip protocols?
- No, depends on the properties of the peer selection step
- What are the minimum properties for peer selection that are necessary to guarantee exponential convergence?
Gossip versus evolutionary computing

- What is the relationship between gossip and evolutionary computing?
- Is one more powerful than the other? Are they equal?