Communication between Nodes
Epidemic Diffusion and Gossiping

Smruti R. Sarangi
Department of Computer Science
Indian Institute of Technology
New Delhi, India
Outline

1. Communication between Nodes

2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution

3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
Overlay Network

It is an application level network that is independent of the underlying network topology.

- Most common overlay networks are a ring and star.
A **star** is a centralized configuration.
- The central node is typically a server.
- The rest of the nodes are clients.

A ring is the basis of a structure called a **DHT** (distributed hash table)
- We will study about ring topologies in third generation peer to peer networks.

Let us first focus on unstructured overlay networks.
- There is no fixed global topology.
- A node typically only knows a subset of other nodes.
The Problem
Multicast a message to a group of nodes.

- Send the message to all the neighbors.
- Ask the neighbors to further forward the message to their neighbors.
- Need a method to solve the exponential flooding of messages.
- Need mathematical techniques for analysis.
Epidemic Algorithms

Paper

Epidemic Algorithms for Replicated Database Maintenance by Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry, PODC 1987

- **Problem**: Propagate updates to a large set of databases in Xerox’s corporate intranet.
- **Updates**: Updates are injected at one site and propagated to the rest of the sites.
General Mechanisms

Direct Mail
Each update is sent from one site to all sites.

Anti-Entropy
Anti-entropy chooses a site at random and synchronizes the content of the database by exchanging contents.

Rumor Mongering
A site distributes updates to other sites. When a site sees that most of its neighbors have the update, the rumor ceases to be hot. Gradually it dies away.
Tradeoffs

- **infective** Already received the update, and willing to propagate.
- **susceptible** Has not received the update.
- **removed** Not participating in propagating updates.

- Anti-Entropy
  - Takes longer to propagate updates as compared to direct mail.
  - Does not have a built in termination mechanism.
  - Simple epidemic

- Rumor mongering
  - There is a chance that updates might not reach a node.
  - Has a built in termination mechanism.
  - Complex Epidemic
Outline

1 Communication between Nodes

2 Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution

3 Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
A network contains $S$ sites.

Database copy at $s \in S$ is

$$s.valueOf : K \rightarrow (v : V \times t : T)$$

- $v$ is the value
- $t$ is the timestamp
Algorithm 1: Anti-entropy algorithm

1. ResolveDifference–push(s,s') {
   if s.valueOf.t > s'.valueOf.t then
   s'.valueOf ← s.valueOf
   end
}

5. ResolveDifference–pull(s,s') {
   if s.valueOf.t < s'.valueOf.t then
   s.valueOf ← s'.valueOf
   end
}

9. ResolveDifference–push,pull(s,s') {
   ResolveDifference–pull(s,s')
   ResolveDifference–push(s,s')
}
Analysis

- Anti-entropy distributes updates in $O(\log n)$ time (see results from epidemic theory).
- Pull-based algorithms
  - Let $p_i$ be the probability of a site remaining susceptible after the $i^{th}$ cycle.
  - $p_{i+1} = p_i^2$

- Push-based algorithms
  - Expected number of infective nodes: $n(1 - p_i)$
  - Probability of not contacting node $X$: $1 - 1/n$
  - $p_{i+1} = p_i (1 - 1/n)^{n(1-p_i)}$
  - Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \to \infty$
  - Thus, for large $n$, and small $p_i$:
    - $p_{i+1} = p_i e^{-1}$
Anti-entropy distributes updates in $O(\log n)$ time (see results from epidemic theory).

**Pull-based algorithms**
- Let $p_i$ be the probability of a site remaining susceptible after the $i^{th}$ cycle.
  \[ p_{i+1} = p_i^2 \]

**Push-based algorithms**
- Expected number of infective nodes: $n(1 - p_i)$
- Probability of not contacting node $X$: $1 - 1/n$
  \[ p_{i+1} = p_i (1 - 1/n)^{n(1-p_i)} \]
- Now, $(1 - 1/x)^x$ tends to $1/e$ as $x \to \infty$
- Thus, for large $n$, and small $p_i$:
  \[ p_{i+1} = p_i e^{-1} \]

Pull based, and push-pull based methods are better at the end.
Discussion

- Push based methods are better at the beginning
- Towards the end pull based methods are better
- Instead of comparing entire database contents, we can do better:
  - First compare recent entries (Less than $\tau$ seconds old
  - If they match, then nothing needs to be done.
  - If they do not match, then update recent entries, and compare check sums of the rest of the database.
  - If the checksums do not match, then synch. databases.
Outline

1. Communication between Nodes

2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution

3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
Terminology

- $s \rightarrow$ fraction of nodes that are susceptible
- $i \rightarrow$ fraction of nodes that are infective
- $r \rightarrow$ fraction of nodes that are removed

Governing Equations

- Sum of nodes is 1

\[ s + i + r = 1 \]
Equations

- Rate of decrease of susceptible nodes
  \[
  \frac{ds}{dt} = -si
  \]

- Nodes lose interest in propagating rumors by a probabilistic factor of 1/k.
  \[
  \frac{di}{dt} = si - \frac{1}{k} (1 - s)i
  \]

Solution

\[
i(s) = \frac{k + 1}{k} (1 - s) + \frac{1}{k} \log(s)
\]
Implication

There are still some susceptible nodes. They decrease exponentially with $k$. 
Implications of the Solution

Let us now see what percentage of nodes are still susceptible, when no other node is infective \( i(s) = 0 \). 

\[
s = e^{-\frac{k+1}{1-s}}
\]

- Exponentially decreases with \( k \).
- Some nodes still remain susceptible.
- This value is called the residue.
**residue** Sites that are still susceptible after the end of the epidemic.

**traffic** Average number of messages sent per site.

- $m$ updates per site, $n$ sites, total $nm$ updates
- Chances that a site will miss all the updates:

$$s = (1 - 1/n)^{nm} = e^{-m}$$

Exponential relationship between traffic and residue
Rumor mongering can miss sites.
After a certain time, we can run an anti-entropy protocol.
If two sites discover a missing update, then they could start a hot rumor.
Xerox Clearinghouse did some redistribution through direct mail also.
Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery

Communication between Nodes
Death Certificate

We can treat the deletion of a node as an *update*, and issue a death certificate. The death certificates can be propagated through rumors or anti-entropy.

- When do we discard death certificates?
- Need to define a time threshold.
- If a death certificate is older than the time it takes to propagate the update to all sites, we can delete it.
- We can still maintain some copies at a selected number of retention sites.
Anti-entropy/Rumor with Dormant Death Certificates

- What if a dormant death certificate meets an obsolete update.
- Reactivate the dormant death certificate and distribute it.
- It is possible that a legitimate update can be cancelled.
- This can be solved by using version numbers for updates. A death certificate will be for a certain update.
- A later update will not be connected with this death certificate.
Spatial Distributions

Known Results

- If a node can contact only neighbors, it takes $O(n)$ time to spread an update using anti-entropy.
- If a node can contact any other node, it takes $O(\log(n))$ time.

General Result

Let the probability of connecting to a site at distance $d$ be $d^{-a}$.

- For $a > 2$, it takes $O(n^k)$ time for convergence.
- For $a < 2$, it takes $O(\log(n)^k)$ time for convergence.
Problem: A set of nodes fail. Design a failure detector that detects failures by gossiping.

Model of failure: Fail-Stop $\Rightarrow$ If a node does not respond to a message for $T$ seconds, then it has most likely failed.

The algorithm needs to scale in terms of the number of nodes, $n$. 
Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
Aims of a Protocol

- Probability of a false positive is independent of $n$.
- Resilient to message loss and network partitions.
- Scalability in detection time: $O(n \log(n))$
- If clock drift across nodes is negligible, then the algorithm detects all failures with a known probability of a mistake.
- Bandwidth increase is linear in terms of processes.
Basic Protocol

- Each node maintains a list. \((\text{member}_\text{id}, \text{timestamp}, \text{heartbeat counter})\)
- Every \(T_{\text{gossip}}\) seconds, each node updates its heartbeat counter, and sends a \text{gossip} message to a randomly chosen node.
- The \text{gossip} message contains the message list.
- The receiver merges the message lists, and adopts the larger heartbeat counter for a node.
- The \text{timestamp} for a member indicates the last time that the receiver thinks that a node has updated its heartbeat counter.
Failure Detection

- If the heart beat counter for a node hasn’t increased in $T_{\text{fail}}$ seconds, then a node presumes that it has failed.
- The probability of a false detection is $P_{\text{mistake}}$ or $P_{\text{fail}}$.
- However, the entry is not removed because a node can continue to get gossips about the failed node.
- It removes the node after $T_{\text{cleanup}}$ seconds.
- $P_{\text{cleanup}}$ is the probability that a gossip is gotten after $T_{\text{fail}}$ seconds.
- $(P_{\text{fail}} = P_{\text{cleanup}})$ if

$$T_{\text{cleanup}} = 2 \times T_{\text{fail}}$$
Outline

1. Communication between Nodes
2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution
3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
Assume that $f$ out of $n$ members have failed.
$k$ out of $n$ members are infective.
Only one node sends one message to another node in a round.
Probability of incrementing the number of infective nodes:

$$P_{inc}(k) = \frac{k}{n} \times \frac{n - f - k}{n - 1}$$

Probability of having $k$ infected members in round $i + 1$ is

$$P(k_{i+1}) = P_{inc}(k - 1) \times P(k_i = k - 1)$$

$$+ (1 - P_{inc}(k)) \times P(k_i = k)$$
Probability that any process does not get infected by $p$ after $r$ rounds is $P_{\text{mistake}}(p, r)$

$$P_{\text{mistake}}(p, r) = 1 - P(k_r = n - f)$$

$$P_{\text{mistake}}(r) = \bigcup P_{\text{mistake}}(p, r) \leq (n - f)(1 - P(k_r = n - f))$$
1 member has failed, 250 bytes per second (bandwidth restriction), $p = P_{\text{mistake}}$

\[ \text{time} = O(n\log(n)) \]
Performance-II

Detection time vs $P_{mistake}$
Outline

1. Communication between Nodes

2. Epidemic Protocols
   - Anti-Entropy
   - Rumor Mongering
   - Deleting Nodes and Spatial Distribution

3. Gossip Based Protocols
   - Protocol
   - Mathematical Analysis
   - Catastrophe Recovery
Catastrophe Recovery

- Gossip algorithms do not work in the case of network partitions
- Failure detector needs to broadcast messages to re-establish connections.
Broadcast Protocol

- Each second, a node probabilistically decides to send a broadcast.
- The probability depends on the last time a node received a broadcast.
- If a node received a broadcast 20 seconds ago, then it broadcasts with very high probability.
- A function of the form:

\[ p(t) = \frac{t^a}{20} \]

fits well.
A Gossip Style Failure Detection Service by Robert Renesse, Yaron Minsky, and Mark Hayden (Technical Report)

Epidemic Algorithms for Replicated Database Maintenance by Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry, PODC 1987