Epidemic Algorithms

Slides by Amir H. Payberah (amir@sics.se), Jim Dowling
Introduction

• Motivations
  - Existing information dissemination protocols have scalability problems.
  - Randomized protocols may have a smaller overhead.
  - Trade-off between reliability and scalability.

• Can be applied
  - To large-scale distributed systems (millions of nodes).
  - When real-time information dissemination is not required.
Epidemic Protocols

- **Epidemics** study the **spread of a disease or infection** in terms of populations of infected/uninfected individuals and their rates of change.

- **How does it work?**
  - Initially, **a single individual** is **infective**.
  - **Individuals get in touch with each other**, spreading the update.

- **Our goal** is to spread the infection (update) **as fast and completely as possible!**
Two Styles of Epidemic Protocols

- Anti-entropy
- Rumor mongering
Each peer $p$ periodically contacts a random partner $q$ selected from the current population.

Then, $p$ and $q$ engage in an information exchange protocol, where updates known to $p$ but not to $q$ are transferred from $p$ to $q$ (push), or vice-versa (pull), or in both direction (push-pull).
Rumor Mongering

- Peers are initially ignorant.

- When an update is learned by a peer, it becomes a hot rumor.

- While a peer holds a hot rumor, it periodically chooses a random peer from the current population and sends (pushes) the rumor to it.

- Eventually, a node will lose interest in spreading the rumor.
Rumor Mongering: Loss of Interest

- **Counter vs. Coin**
  - **Counter**: lose interest after $k$ contacts.
  - **Coin (random)**: lose interest with probability $1/k$.

- **Feedback vs. Blind**
  - **Feedback**: lose interest only if the recipient knows the rumor.
  - **Blind**: lose interest regardless of the recipient.
Epidemic Protocols Scale Very Nicely

- Participants’ load is independent of size
- Information spreads in log(system size) time.

![Graph showing the spread of infection over time](image)
Use of Epidemic Protocols

- Aggregation protocols
- Membership management (Cyclon)
- Topology management (T-man)
- Etc.
Aggregation Protocols
Aggregation is a common name for a set of functions that provide an estimate of some global system property.

Aggregation functions enable local access to global information, in order to simplify the task of controlling, monitoring, and optimizing distributed applications.

Some examples of aggregation functions:
- The average load of nodes in a distributed storage system.
- The sum of free space in a distributed storage system.
- The total number of nodes in a P2P system.
// timed event
timer(T time units)
q = SelectPeer()
send S to q

// handle event
recv S_p from p
send S to p
S = Update(S,S_p)

// handle event
recv S_q from q
S = Update(S,S_q)
Some Comments

- **Local state** maintained by nodes:
  - a real number representing the value to be averaged.

- **selectPeer()**
  - performs a *random* selection among the set of current nodes.

- **update(sp, sq)**
  - **Avg**: return \( \frac{sp+sq}{2} \)
  - **Max**: return \( \max(sp,sq) \)
Average Aggregation (1/5)
Average Aggregation (2/5)
Average Aggregation (3/5)

(10 + 2) / 2 = 6
Average Aggregation (4/5)
Average Aggregation (5/5)

(16 + 4) / 2 = 10
Some Comments

- If the graph is connected, each node converges to the average of the original values.

- After each exchange the variance is reduced.
Illustration of Averaging
Network Size Estimation

- Any ideas?
Network Size Estimation

• Any ideas?

• All nodes set their states to 0.

• The initiator sets its state to 1 and starts gossiping for the average.

• Eventually (after predefined $k$ rounds) all nodes converge to the $\text{avg}=1/N$. 
Membership Management
Membership Management

- In a gossip-based protocol, each node in the system periodically exchanges information with a subset of peers.

- The choice of this subset is crucial.

- Ideally, the peers should be selected following a uniform random sample of all nodes currently in the system.
Achieving a Uniform Random Sample

- Each node may be assumed to know every other node in the system.

- However, providing each node with a complete membership table from which a random sample can be drawn, is unrealistic in a large-scale dynamic system.
An Alternative Solution

- Peer sampling

- Every node maintains a relatively small local membership table that provides a *partial view* on the complete set of nodes.

- Periodically refreshes the table using a gossiping procedure.
Peer Sampling Generic Framework (1/3)

// timed event every \( T \) time units
handle
  q = view.SelectPeer()
  buf = ((myAddress, 0))
  view.permute()
  move oldest \( H \) items to the end of view
  buf.append(view.head(c/2-1))
  send buf to q
  recv buf\(_q\) from q
  view.select(c, H, S, buf\(_q\))
  view.increaseAge()
// receiver handler
handle
    recv buf_p from p
    buf = ((myAddress, 0))
    view.permute()
    move oldest H items to the end of view
    buf.append(view.head(c/2-1))
    send buf to p
    view.select(c, H, S, buf_p)
    view.increaseAge()
// view select method
method view.select(c, H, S, buf_p)
    view.append(buf_p)
    view.removeDuplicates()
    view.removeOldItems(min(H, view.size-c))
    view.removeHead(min(S, view.size-c))
    view.removeAtRandom(view.size-c)
Design Space

- **Peer Selection**
  - **Rand**: uniform random
  - **Tail**: highest age

- **View Propagation**
  - **Push**
  - **Push-Pull**

- **View Selection**
  - **Blind**: $H = 0, S = 0$
  - **Healer**: $H = c / 2$
  - **Swapper**: $H = 0, S = c / 2$
Gossip-based Peer Sampling Protocol (1/7)
Gossip-based Peer Sampling Protocol (4/7)
Gossip-based Peer Sampling Protocol (5/7)
Gossip-based Peer Sampling Protocol (6/7)
Gossip-based Peer Sampling Protocol (7/7)
Newscast as a Peer Sampling Example

- **Peer Selection**
  - Rand: uniform random
  - Tail: highest age

- **View Propagation**
  - Push
  - Push-Pull

- **View Selection**
  - Blind: \( H = 0, \ S = 0 \)
  - Healer: \( H = \frac{c}{2} \)
  - Swapper: \( H = 0, \ S = \frac{c}{2} \)
Newscast (1/7)
• Pick a random peer from my view
Newscast (3/7)

- Pick a random peer from my view
- Send each other view + own fresh link
- Pick a random peer from my view
- Send each other view + own fresh link
Newscast (5/7)

- Pick a random peer from my view
- Send each other view + own fresh link
- Keep c freshest links (remove own info and duplicates)
Newscast (6/7)

- Pick a random peer from my view
- Send each other view + own fresh link
- Keep c freshest links (remove own info and duplicates)
Newscast (7/7)

- Pick a random peer from my view
- Send each other view + own fresh link
- Keep c freshest links (remove own info and duplicates)
Cyclon as a Peer Sampling Example

• Peer Selection
  ▪ **Rand:** uniform random
  ▪ **Tail:** highest age

• View Propagation
  ▪ **Push**
  ▪ **Push-Pull**

• View Selection
  ▪ **Blind:** $H = 0, S = 0$
  ▪ **Healer:** $H = c / 2$
  ▪ **Swapper:** $H = 0, S = c / 2$
Cyclon (1/5)
• Pick the oldest peer from my view and remove it from the view.
- Pick the oldest peer from my view and remove it from the view.
- Exchange some of the peers in neighbours (swap policy)
- The active peer sends its fresh address
• Pick the oldest peer from my view and remove it from the view.
• Exchange some of the peers in neighbours (swap policy).
• The active peer sends its fresh address
Cyclon (5/5)

- Pick the oldest peer from my view and remove it from the view.
- Exchange some of the peers in neighbours (swap policy).
- The active peer sends its fresh address.
Cyclon Properties: Connectivity

- In a fail-free environment, no peer becomes disconnected in the undirected graph.
- Pointers move, so peers change from being neighbor of one peer to being the neighbor of another peer
Cyclon Properties: Convergence

- Starting from a state, where peers are connected in a chain.
- Convergence is defined by having the same average path length as a random graph.
Cyclon Properties: Clustering Coefficient

- **Clustering Coefficient** (of a node): the ratio of existing links among the node’s neighbors over the total number of possible links among them.
- Shows what percentage the neighbors of a node are also neighbors among themselves.

![Graph showing clustering coefficient over time with different values of c and random graphs](image-url)
Cyclon Properties: Indegree Distribution
Topology Management
T-Man

- **T-man** is a protocol that can construct and maintain any **topology** with the help of a **ranking function**.

- The **ranking function** orders any set of nodes according to their **desirability** to be neighbors of a given node.
// timed event every T time units
handle
    q = view.selectPeer()
    myDescriptor = (myAddress, myProfile)
    buf = merge(view, myDescriptor)
    buf = merge(buf, rnd.view)
    send buf to q
    recv buf_q from q
    buf = merge(buf_q, view)
    view = selectView(buf)
A Generic T-Man Framework (2/2)

// receiver handler
Handle
    recv buf_p from p
    myDescriptor = (myAddress, myProfile)
    buf = merge(view, myDescriptor)
    buf = merge(buf, rnd.view)
    send buf to p
    buf = merge(buf_p, view)
    view = selectView(buf)
Some Comments

- **SelectPeer**
  - Sort all nodes in the view based on ranking.
  - Pick randomly one node from the first half.

- **rnd.view**
  - provides a random sample of the nodes from the entire network, e.g., using cyclon

- **SelectView**
  - Sort all nodes in buffer (about double size of the view)
  - Pick out c highest ranked nodes.
Ranking Function

• Sample ranking functions:
  - **Line**: \(d(a, b) = |a - b|\)
  - **Ring**: \(d(a, b) = \min(N - |a - b|, |a - b|)\)
Illustration of T-Man

after 3 cycles

after 5 cycles

after 8 cycles

after 15 cycles
Connectivity Problems on the Open Internet
NAT Environments (1/4)

Private node

Public node

shuffle request
NAT Environments (1/4)
Solutions for Communicating with Private Nodes (1/2)

- Relay communications to the private node using a public relay node.
Solutions for Communicating with Private Nodes (2/2)

- Use a NAT **hole-punching** algorithm to establish a direct connection to the private node using a **public rendezvous node**.
Relaying or Hole Punching?

- **Relaying?**
  - **Lower latency** message exchange.
    - Enables lower gossip cycle periods.
    - Necessary in dynamic networks

- **Hole punching?**
  - **Decreases load** on public nodes.
    - But not if shuffle messages are small.
Gozar as a NAT-aware Peer Sampling Example

- In Gozar, each private node connects to one or more public nodes, called partners that act as a relay or rendezvous server on behalf of the private node.

- A node's descriptor consists of both its own address, its NAT type, and its partners' addresses at the time of descriptor creation.

- When a node wants to gossip with a private node, it uses the partner addresses in its descriptor to communicate with the private node.
Partnering (1/10)
Partnering (2/10)

Bootstrap server

n2

n1

n3

n4

n5
Partnering (3/10)
Partnering (4/10)

Bootstrap server

request

n1, public, null
n4, public, null

request

n1
n2
n4
n5

n3

...
Partnering (5/10)

Bootstrap server

n1, public, null
n4, public, null

ACK

NACK
Partnering (6/10)

Bootstrap server

n1

n2

n1, public, null
n4, public, null
...

n4

n5

n3

...
Partnering (7/10)

Bootstrap server

n1

n2, private, n1

n1, public, null
n4, public, null

Shuffle exchange

n2

n4

n5

n3
Partnering (8/10)
Partnering (9/10)

Bootstrap server

n1, public, null
n4, public, null
...

n2, private, n1
n2, private, n1
...

Shuffle request

n5
n3
n1

...
Partnering (10/10)

Bootstrap server

- n1, public, null
- n4, public, null

Shuffle response

- n2, private, n1
- n2, private, n1

- n5
Summary
Summary

- Epidemics algorithms are important technique to solve problems in dynamic large scale systems
  - Scalable
  - Simple
  - Robust to node failures, message loss and transient network disruptions (network partitions ...)

- Applications:
  - Aggregation
  - Membership management
  - Topology management
Question

Acknowledgement
Some slides were derived from the slides of Alberto Montresor and Seif Haridi