Outline

❖ Timeline
❖ CAP Theorem
❖ Epidemic algorithms for replicated database maintenance
❖ Managing update conflicts in Bayou, a weakly connected replicated storage system
❖ Conclusion
Timeline

1978
Lamport
Time, Clocks, and the Ordering of Events in a Distributed System

1982
Lamport
The Byzantine Generals Problem

1985
FLP
Impossibility of Distributed Consensus with One Faulty Process

1987
Demers
Epidemic algorithms for replicated database maintenance

1990
Schneider
Implementing fault-tolerant services using the state machine approach: A tutorial

1995
Terry
Managing update conflicts in Bayou, a weakly connected replicated storage system

1998
Lamport
The part-time parliament
CAP

● Consistency -- all nodes contain the same state
● Availability -- requests are responded to promptly
● Partition
  ○ part of a system completely independent from the rest of the system
  ○ ideally should maintain itself autonomously
● Partition tolerance -- system can stay online and functional even when message passing fails
CAP Theorem

Paxos & Gossip

- Paxos: prioritize consistency given a network partition
- Gossip: prioritize availability given a network partition
Gossip
Gossip Overview

- Authors
- Motivations
- Epidemic Models
  - Direct Mail
  - Anti-Entropy
  - Rumor mongering
- Evaluation
- DC’s
- Spatial Distribution
Alan Demers
Cornell University

Dan Greene
PARC Research

Scott Shenker
EECS Berkeley

Doug Terry
Amazon Web Services

Carl Hauser
PhD Cornell
Washington State University
Motivations

- Unreliable network
- Unreliable nodes
- CAP: *AP*
  - always be able to respond to a (read/write) request
  - eventual consistency
Epidemic Models
Proposers and Acceptors

- **Proposer**
  - In Paxos: clients **propose** an update to the database
  - Epidemic model: a node **infects** its neighbors

- **Acceptor**
  - In Paxos: acceptor **accepts** an update based on one or more proposals
  - Epidemic model: a node is **infected** by a neighbor
Types of Epidemics

- Direct Mail
- Anti-Entropy
- Rumor Mongering
Advantages

- Simple algorithms
- High Availability
- Fault Tolerant
- Tunable
- Scalable
- Works in Partition
● Notify all neighbors of an update
● Timely and reasonably efficient
● \( n \) messages per update
Direct Mail
Direct Mail
Direct Mail

Messages sent: $O(n)$ where $n$ is number of neighbors

Not fault tolerant -- doesn’t guarantee eventual consistency

High volume of traffic with site at the epicenter
Anti-Entropy

- Site chooses random partner to share data
- Number of rounds til consistency: $O(\log n)$
- Sites use custom protocols to resolve conflicts
- Fault tolerant
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy
Anti-Entropy

What happens next?
Mechanism: Push & Pull
Push vs. Pull

**Push**

- \{A, B\}
- \{A, C\}

- \(H(A), H(B)\)
- \(H(B)\)
- \(B\)

**Pull**

- \{A, B\}
- \{A, C\}

- \(H(A), H(B)\)
- \(C\)
What is Push-Pull?
Propagation times of Push vs. Pull

**Push:** \( P_{i+1} = P_i e^{-1} \)

**Pull:** \( P_{i+1} = P_i^2 \)

Pull is faster!!

\( P \) = Probability node hasn’t received update after the \( i^{th} \) round
Rumor Mongering

1. Sites choose a random neighbor to share information with

2. Transmission rate is tuneable

3. How long new updates are interesting is also tuneable

4. Can use push or pull mechanisms
Rumor Mongering Complexity

- $O(\ln n)$ rounds leads to consistency with high probability
- Push requires $O(n \ln n)$ transmissions until consistency
- Further proved lower bound for all push-pull transmissions: $O(n \ln \ln n)$

Analogy to epidemiology

- **Susceptible**: site does not know an update yet
- **Infective**: actively sharing an update
- **Removed**: updated and no longer sharing

Rumor mongering: nodes go from *susceptible* to *infective* and eventually (probabilistically) to *removed*
Rumor mongering
Rumor mongering
Rumor mongering
Rumor mongering
Rumor mongering
Rumor mongering
Rumor mongering
Rumor mongering

Pros:

● Fast
● Low call on resources
● Fault-Tolerant
● Less traffic

Cons:

● A site can potentially miss an update
Backups

- Anti-entropy can be used to “update” the network regularly after direct mail or rumor mongering
- If inconsistency found in anti-entropy, run the original algorithm again
Death Certificates

❖ How are items deleted using epidemic models?
I like Bread

I DON'T like Bread!

I like orange juice
Death Certificates

- How to remove items from epidemic model?

- Drawbacks
  - Space
  - Increases traffic
  - DC Can be lost

- Dormant death certificates & retention
Evaluating Epidemic Models

➢ **Residue:** remaining susceptibles when epidemic finishes

➢ **Traffic:** $\frac{\text{updatetraffic}}{\text{number of sites}}$

➢ **Delay:**
  ○ $T_{\text{avg}}$: Average time between start of outbreak and arrival of update @ given site
  ○ $T_{\text{last}}$: Delay until last update
Spatial Distribution

Helping
Or
Hurting
Convergence Times and Traffic

- Linear network: anti entropy
  - Nearest-neighbors
    - $O(n)$ convergence
    - $O(1)$ traffic
  - Random connections
    - $O(\log(n))$ convergence
    - $O(n)$ traffic
Optimizations for realistic network distributions

- Select connections from list of neighbors sorted by distance
- Treat network as linear
- Compute probabilities based on position in list
Rumor Mongering Non-Standard Distribution

- Increase $k$ -- number of rounds a rumor is “interesting”
- Use push-pull
Takeaways

- Availability $>>$ consistency
- Updates can be expensive
- Distribution protocols should be robust
- Network design can hurt overall performance
- Byzantine Behavior not addressed

Questions?
Additional Reading

Managing update conflicts in Bayou, a weakly connected replicated storage system

1995
- Weak consistency makes unstable network applications possible
- Developing good interfaces allows for complex functions like merging to be interchangeable via the application
Timeline

1978
Lamport
Time, Clocks, and the Ordering of Events in a Distributed System

1982
Lamport
The Byzantine Generals Problem

1985
FLP
Impossibility of Distributed Consensus with One Faulty Process

1987
Demers
Epidemic algorithms for replicated database maintenance

1990
Schneider
Implementing fault-tolerant services using the state machine approach: A tutorial

1995
Terry
Managing update conflicts in Bayou, a weakly connected replicated storage system

1998
Lamport
The part-time parliament
What is Bayou?

- Storage system designed for mobile computing
  - Network is not stable
  - Parts of the network may not be connected all the time
  - Goal: high availability
  - Guarantees weak consistency
Bayou System Diagram
Consistent Replicas

- Writes are first tentative
- Eventually they are committed, ordered by time
- Clients can tell whether writes are stable (committed)
- Primary servers deal with committing updates
Detecting and Resolving Conflicts

- Dependency checks
- Merge procedures
- Described by the clients, application-dependent
Conclusions

- Distributed systems need a form of consensus
- Effectively choosing the correct consensus model for a system has to be weighed carefully with the attributes of the system
Acknowledgements

Content Inspired by:

Ki Suh Lee: “Epidemic Techniques”[2009]

Eugene Bagdasaryan: “P2P Gossip Protocols” [2016]

Photos

www.pixabay.com

www.unsplash.com

www.1001freedownloads.com/free-cliparts