Epidemic Algorithms For Replicated Database Maintenance

Alan Demers, Mark Gealy, Dan Greene, Carl Hauser, Wes Irish, John Larson, Sue Manning, Scott Shenker, Howard Sturgis, Dan Swinehart, Doug Terry, and Don Woods

Presented by Bo Teng
April 26th 2016

Background and Motivation

- Originated from study of Clearinghouse Servers on the Xerox Corporate Internet (CIN)
- World wide CIN consists of several hundred Ethernets connected by gateways and phone lines of different capacity
- Some domains stored in all Clearinghouse servers (HIGHLY REPLICATED)
- Need to achieve and maintain mutual consistency in replicas
    - Efficient, robust and scalable algorithm

Outline

1. **Problem statement**
2. Direct Mail
3. Anti-entropy
4. Rumor mongering
5. Deletion & death certificate
6. Spacial distribution

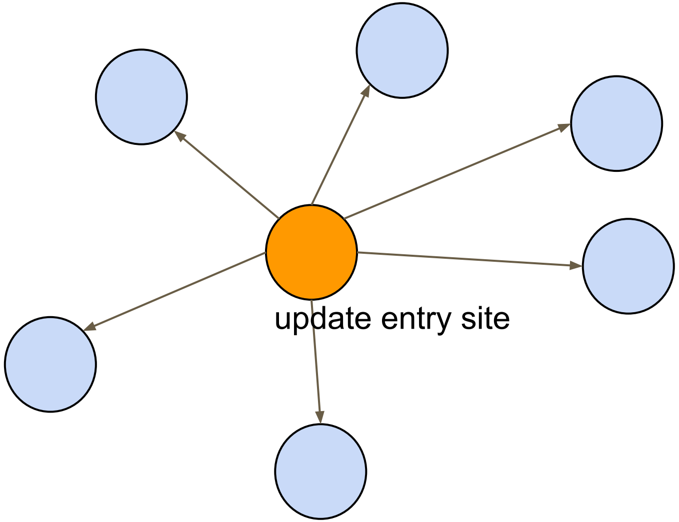
Problem statement

- Database **replicated** at many sites
- **Large**, **heterogeneous**, slightly **unreliable**, slowly **changing** Network
- Assume **synchronized** clocks in different sites
- Examine methods that achieves and maintains consistency in replicas
  - Direct mail
  - Anti-entropy
  - Rumor mongering

Outline

1. Problem statement
2. **Direct Mail**
3. Anti-entropy
4. Rumor mongering
5. Deletion & death certificate
6. Spacial distribution

Direct Mail

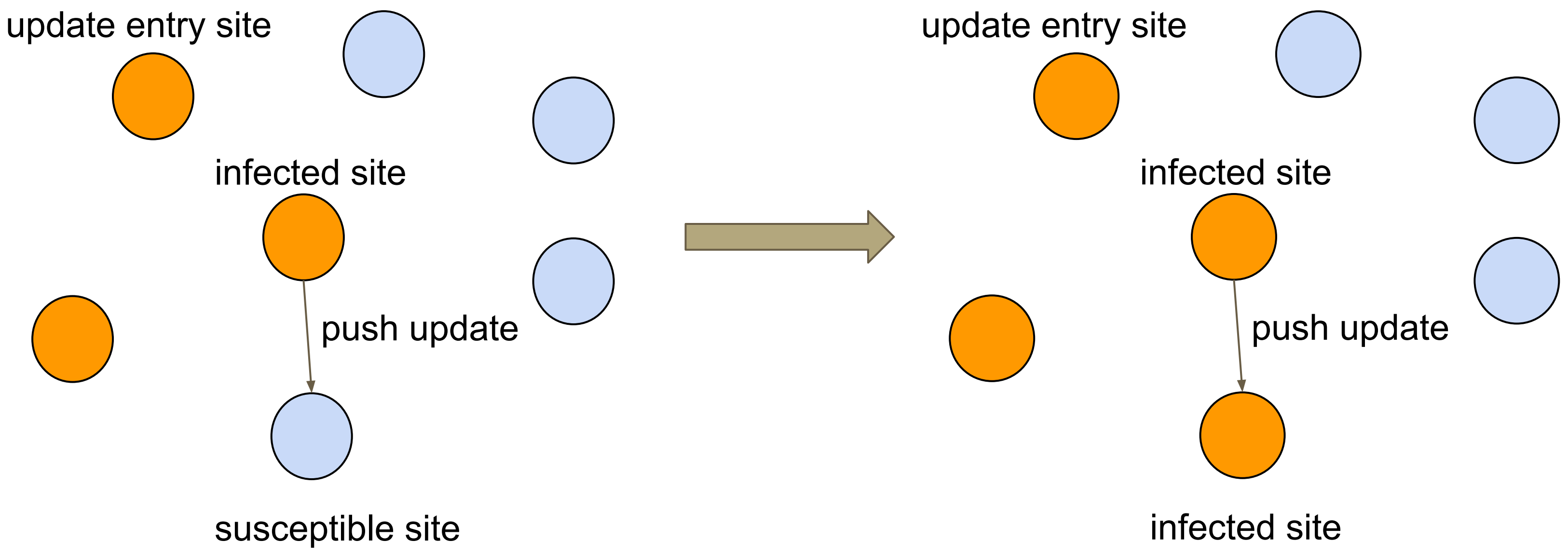


- Timely, reasonably efficient, but not reliable
- Problems?
  - Message discarded when message queue overflows
  - Message lost due to bad network
  - Inaccurate member list at sender

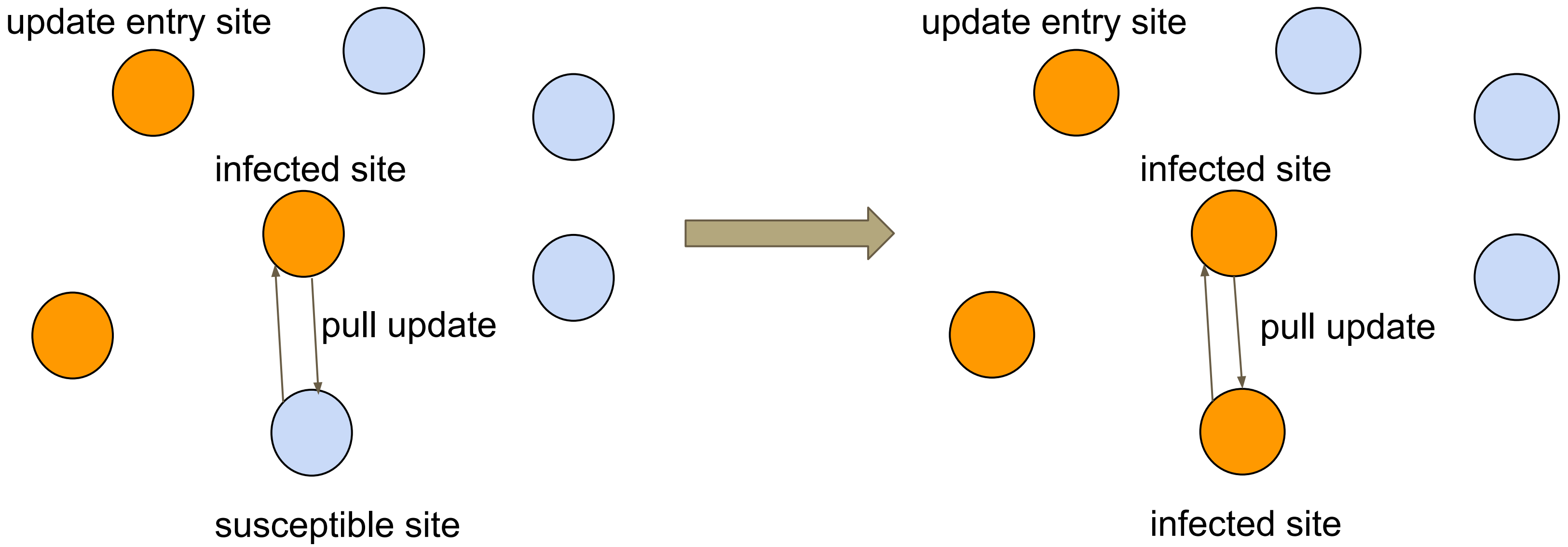
Outline

1. Problem statement
2. Direct Mail
3. **Anti-entropy**
4. Rumor mongering
5. Deletion & death certificate
6. Spacial distribution

Anti-entropy: push



Anti-entropy: pull



Anti-entropy: Pull vs. Push

- Probability a site **s** is susceptible after (i+1)th round:
  - Pull: **s** is susceptible at round i & **s** contacts a susceptible site at round i+1
    $p_{i+1} = p_i^2$
  - Push: **s** is susceptible at round i & no infected node contacts **s** at round i+1
    $p_{i+1} = p_i(1 - 1/n)^{n(1-pi)} = p_i(1/e)$
- Convergence rate: Pull > Push
- Hybrid push-pull variation available

Anti-entropy

- Reliable, but propagates slower than direct mail and is expensive
- Could be run as back-up algorithm for direct mail
- Problem:
  - Compare two complete copies of database.
  - Most data are in complete agreement, most work is wasted
  - A copy of database is sent through network

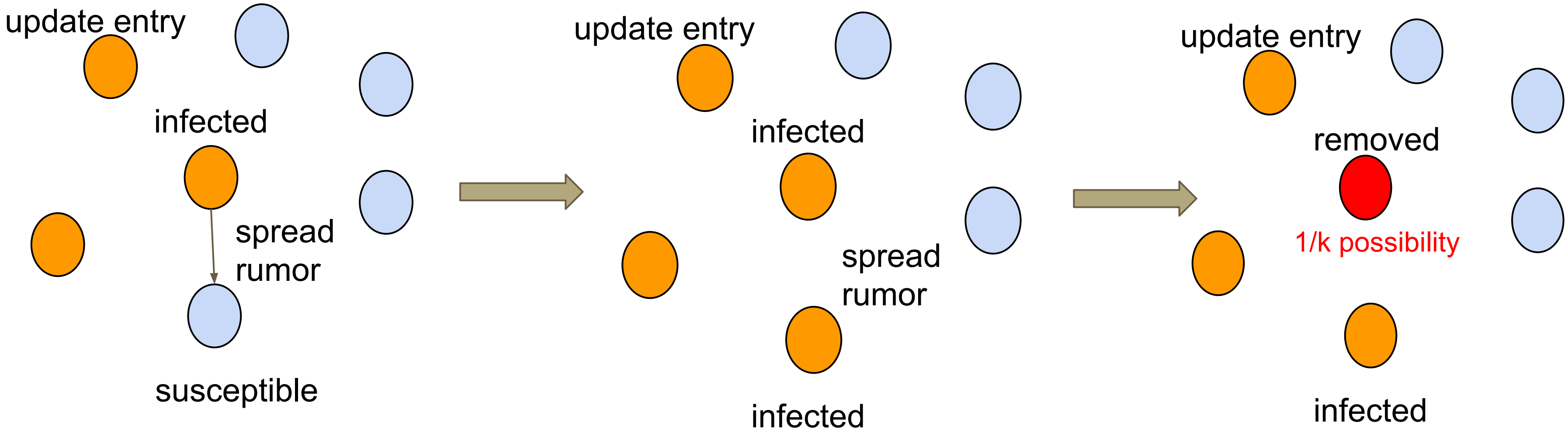
Anti-entropy: Optimization

- Maintain checksum of database
  - Compare DB only when checksum differs
  - BUT checksum likely to disagree before updates reach all replicas
- Maintain checksum and *recent update list*
  - Exchange *recent update list* with updates within time window τ first
  - Update DB and checksum using *recent update list*, then compare checksum
- Maintain *inverted index* of DB by timestamp
  - Exchange information in reverse timestamp order
  - Incrementally recompute checksums

Outline

1. Problem statement
2. Direct Mail
3. Anti-entropy
4. **Rumor mongering**
5. Deletion & death certificate
6. Spacial distribution

Rumor Mongering: Algorithm



Rumor Mongering: Probability of failure

- System becomes quiescent when all infected sites become removed
- There exists an explicit probability of failure. Fortunately, this probability could be make arbitrarily small

$$s = e^{-(k+1)(1-s)}$$

- k = 1: 20% sites miss the rumor
  k = 2:   6% sites miss the rumor
  k = 6: only 0.1% sites miss the rumor

Rumor Mongering: Variations

- Blind vs. Feedback
  - Feedback: site loses interest depending on recipient feedback
  - Blind: site loses interest regardless of the recipient
- Counter vs. Coin
  - Coin: sender loses interest with probability of 1/k
  - Counter: sender loses interest after k unnecessary contacts
- Pull vs. Push
  - $s = e^{-\lambda m}$ push: $\lambda = 1/(1-1/e)$; pull: $\lambda = -\ln\delta$

Rumor Mongering: Performance Metrics

- Residue
  - Percent susceptibles when epidemic finishes
- Traffic
  - m = (Total update traffic) / (Number of sites)
- Delay
  - Average/maximum time for receiving update

Rumor Mongering: Results

- Counter/feedback improves *Convergence time*

Table 1. Performance of an epidemic on 1000 sites using feedback and counters.

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	0.18	1.7	11.0	16.8
2	0.037	3.3	12.1	16.9
3	0.011	4.5	12.5	17.4
4	0.0036	5.6	12.7	17.5
5	0.0012	6.7	12.8	17.7

Table 2. Performance of an epidemic on 1000 sites using blind and coin.

Coin k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	0.96	0.04	19	38
2	0.20	1.6	17	33
3	0.060	2.8	15	32
4	0.021	3.9	14.1	32
5	0.008	4.9	13.8	32

Rumor Mongering: Results

- Pull improves *Residue* and *Convergence time* over Push

Table 1. Performance of an epidemic on 1000 sites using feedback and counters. push

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	0.18	1.7	11.0	16.8
2	0.037	3.3	12.1	16.9
3	0.011	4.5	12.5	17.4
4	0.0036	5.6	12.7	17.5
5	0.0012	6.7	12.8	17.7

Table 3. Performance of a pull epidemic on 1000 sites using feedback and counters†.

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	3.1×10^{-2}	2.7	9.97	17.6
2	5.8×10^{-4}	4.5	10.07	15.4
3	4.0×10^{-6}	6.1	10.08	14.0

WHAT ABOUT DELETION?

WHAT ABOUT DELETION?

Resurrection if not deleted all at once!

Outline

1. Problem statement
2. Direct Mail
3. Anti-entropy
4. Rumor mongering
5. **Deletion & death certificate**
6. Spacial distribution

Death certificate

- Replace deleted item with death certificate
- Death certificate carries timestamp and propagate as ordinary data
- Eventually all old copies are replaced with death certificates

This does not completely solve the problem...

When to delete death certificate?

- Delete *death certificate* if timestamp older by a time threshold τ
- Deciding τ: tradeoff between space usage and risk of resurrection
  - Increasing time threshold reduces resurrection risk but increases the amount of space consumed by death certificate
- Lower both resurrection risk and space usage: **Dormand Death Certificate !**
  - After time threshold, delete most death certificate
  - Retain “dormant” copies only at few sites
  - Dormant death certificates “awaken” when encounter obsolete items (unlikely)

When to delete dormant death certificate?

Dormant Death Certificate

- Two time thresholds : ʊ1and ʊ2
  - Retain death certificates till ʊ1
  - Retain dormant death certificates at several sites till ʊ1+ʊ2
- If “awakened” before before ʊ1+ʊ2 is reached?
  - Use a activation timestamp (originally the same as ordinary timestamp)
  - Upon “awakening”, set activation timestamp to current time

Outline

1. Problem statement
2. Direct Mail
3. Anti-entropy
4. Rumor mongering
5. Deletion and death certificate
6. **Spacial distribution**

Spacial Distribution

- Communication cost not uniform
- Favors closer sites in selection

  $p(d) \approx (Q(d-1)^{-a+1} - Q(d)^{-a+1}) / (Q(d) - Q(d-1))$

- Creates less network traffic compared to random uniform selection

Spatial Distribution: Results

- Traffic is greatly reduced with reasonable increase of delay

Table 5. Simulation results for anti-entropy, connection limit 1.

Spatial Distribution	t_{last}	t_{ave}	Compare Traffic		Update Traffic	
			Average	Bushey	Average	Bushey
uniform	11.0	7.0	3.7	47.5	5.8	75.2
$a = 1.2$	16.9	9.9	1.1	6.4	2.7	18.0
$a = 1.4$	17.3	10.1	1.1	4.7	2.5	13.7
$a = 1.6$	19.1	11.1	0.9	2.9	2.3	10.2
$a = 1.8$	21.5	12.4	0.8	1.7	2.1	7.0
$a = 2.0$	24.6	14.1	0.7	0.9	1.9	4.8

Thank You!