BAR Gossip

Lorenzo Alvisi
UT Austin
MAD Services

- Nodes collaborate to provide service that benefits each node
- Service spans multiple administrative domains (MADs)
- Examples:
  - Overlay routing, wireless mesh routing, content distribution, archival storage, ...
How MAD Services Fail

 Nodes can break

 - Fail-stop e.g., disk crash
 - Byzantine – arbitrary deviation

 Misconfigured, compromised by virus, operator error (“rm -rf *”), malicious user, ...
How MAD Services Fail

- Nodes can break
  - Fail-stop e.g., disk crash
  - Byzantine – arbitrary deviation
    Misconfigured, compromised by virus, operator error (“rm -rf *”), malicious user, ...

- Nodes can be selfish
  - Minimize work and maximize gain
    e.g., in a cooperative backup service, store less than fair share of data
Byzantine Model
[Lamport 1982,...]

- Tolerates arbitrary deviations from specification
- Can be practical

Byzantine Model
[Lamport 1982,...]

- Tolerates arbitrary deviations from specification
- Can be practical
- Limits number $f$ of faulty nodes
  - e.g. Agreement requires $f < n/3$
- Assumes all other nodes are correct
  - Inappropriate when all nodes may deviate when in their interest
Rational Model
[Nash 1950,...]

All nodes are rational, and rational nodes can deviate selfishly from their specification

Rational Model
[Nash 1950,...]

- All nodes are rational, and rational nodes can deviate selfishly from their specification

- Does not tolerate Byzantine behavior
  - Broken nodes may violate assumptions
  - Malicious nodes may cause unbounded damage

Inappropriate when some node may deviate against its interest
Three Challenges

1. To develop a model in which it is possible to prove properties about MAD services
2. To understand how to simplify the development of MAD services in the new model
3. To demonstrate that MAD services developed under the new model can be practical
Who's to blame

Jeff Napper
Allen Clement
Edmund Wong
Lorenzo Alvisi
Mike Dahlin
Indrajit Roy
Harry Li
Jean-Philippe Martin
Amit Aiyyer
A First Foray

BAR (Byzantine, Altruistic, Rational) Tolerance
- no bound on rational nodes
- utility functions add expectation of Byzantine behavior

BAR-B, a BAR tolerant cooperative backup service (SOSP 05)
- uses BAR-tolerant RSM to implement abstraction of Altruistic node on top of Rational and Byzantine ones

FlightPath, a BAR tolerant data streaming application (OSDI 06)
- uses BAR-tolerant gossip protocol to disseminate updates
Live Streaming

Examples: Internet radio, NCAA tournament, web concerts, Internet TV

Practical challenges:
- Reduce broadcaster’s used bandwidth
- Minimize latency
- Increase reliability
- Tolerate link and node failures
Live Streaming Setup

Broadcasters

Clients
Live Streaming Setup

Broadcaster

Clients
Live Streaming Setup

Broadcaster

Clients
Live Streaming Setup

Broadcaster

Clients
Live Streaming Setup

Broadcaster

Clients
Rational Peers Don’t Share!

Broadcaster

Clients
Rational Peers Don’t Share!

Broadcaster

Clients
Reliability Degrades...

![Graph showing the probability of receiving an update against the proportion of rational nodes, with a note for Traditional gossip.]
...and Altruistic nodes suffer
BAR Gossip

Probability of receiving an update

Proportion of rational nodes

Traditional gossip
The Setup

Application
- Altruistic broadcaster
- BAR clients
- Static membership
- Full membership list
- Updates useful for finite time
The Setup

Application
- Altruistic broadcaster
- BAR clients
- Static membership
- Full membership list
- Updates useful for finite time

Crypto
- Public/Private key pairs
- Notation: $\langle M \rangle_A$
The Setup

**Application**
- Altruistic broadcaster
- BAR clients
- Static membership
- Full membership list
- Updates useful for finite time

**Crypto**
- Public/Private key pairs
- Notation: $\langle M \rangle_A$

**Incentive Structure**
- Benefit: playing updates
- Cost: bandwidth
- No long-term reputations
BAR Gossip Overview

Balanced Exchange

Optimistic Push
BAR Gossip Overview

Balanced Exchange

- Select partner
- Exchange histories
- Trade equal number of updates

Optimistic Push
BAR Gossip Overview

Balanced Exchange

In each round:
- Select partner
- Exchange histories
- Trade equal number of updates

Little help to peers that fall behind

Optimistic Push
BAR Gossip Overview

Balanced Exchange

In each round:
- Select partner
- Exchange histories
- Trade equal number of updates

Little help to peers that fall behind

Optimistic Push

In each round:
- Select partners
- Exchange histories
BAR Gossip Overview

Balanced Exchange

In each round:
- Select partner
- Exchange histories
- Trade equal number of updates

Little help to peers that fall behind

Optimistic Push

In each round:
- Select partner
- Exchange histories
- Trade possibly unequal numbers of updates

Safety net for lagging peers
Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates
Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates
Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates

- fair exchange
Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates

- fair exchange is impossible without a trusted third party

Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates

- fair exchange is impossible without a trusted third party
- so we settle for fair enough!
Balanced Exchange

In each round

- Select a partner
- Exchange histories
- Trade equal number of updates
  - Exchange briefcases
  - Exchange keys

fair enough exchange
Design principles
Design principles

- Restrict choice
Design principles

- Restrict choice
  - Eliminate non-determinism
Design principles

- Restrict choice
  - Eliminate non-determinism
  - Evict provably deviant peers
Design principles

- Restrict choice
  - Eliminate non-determinism
  - Evict provably deviant peers
- Delay gratification
Design principles

- **Restrict choice**
  - Eliminate non-determinism
  - Evict provably deviant peers

- **Delay gratification**
  - Postpone payoff to keep rational peers engaged
The Intuition

Select C
Select B
Select D

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification
The Intuition

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Select B
The Intuition

Select C

Select D

Select C

Select B

Send history

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification
The Intuition

Select C
Select D
Select C

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Claim less
Send history

Send history
The Intuition

Select C

Select D

Select B

Send history

Select C

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Claim more

Claim less
The Intuition

Select C
Select B
Select D

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Claim less
Claim more
Send history
Send briefcase
The Intuition

Select C

Select B

Select D

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Claim less
Claim more

Send briefcase
Don't send briefcase

Send history
The Intuition

Select B

Select D

Select C

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Claim less
Claim more

Send history

Don’t send briefcase
Send bad briefcase

Send briefcase
The Intuition

Select D

- Claim less

Select B

- Send history

Select C

- Claim more

- Send bad briefcase

- Don’t send briefcase

- Send briefcase

- Send key

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification
The Intuition

Select C

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification

Select D

Select B

Claim less
Claim more
Send history

Don’t send briefcase
Send bad briefcase

Send briefcase

Don’t send key

Send key
The Intuition

- Select D
- Select B
- Select C

- Claim less
- Claim more

- Send history

- Don’t send briefcase
- Send bad briefcase

- Send briefcase

- Don’t send key
- Send wrong key

- Send key

Restrict choice
Eliminate non-determinism
Evict provably deviant peers
Delay gratification
Balanced Exchange is a Nash Equilibrium

Theorem: A balanced exchange is incentive compatible for strategies that maximize the number of useful updates received in that exchange.
Balanced Exchange is a Nash Equilibrium

Theorem: A balanced exchange is incentive compatible for strategies that maximize the number of useful updates received in that exchange

- Partner selection
- History exchange
- Briefcase exchange
- Key exchange
Balanced Exchange is a Nash Equilibrium

Theorem: A balanced exchange is incentive compatible for strategies that maximize the number of useful updates received in that exchange.

- Partner selection
- History exchange
- Briefcase exchange
- Key exchange

』

Incentive compatible
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?
Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection

A's PRNG seed in round $r : \langle r \rangle_A$
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection

As PRNG seed in round $r : \langle r \rangle_A$
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection

As PRNG seed in round $r : \langle r \rangle_A$
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection

- A's PRNG seed in round $r : \langle r \rangle_A$
- Eliminates non-determinism

(\text{check current round check selection})
Partner Selection

Q: How do we limit a peer to one uniformly selected partner per round?

A: Verifiable pseudo-random partner selection

- A’s PRNG seed in round \( r : \langle r \rangle_A \)
  - Eliminates non-determinism
  - Retains strength of randomness:
    - uniform selection of partners
    - unpredictability
Q: How do we handle a client lying about its history?
Q: How do we handle a client lying about its history?

A: Client commits to a history before discovering partner's history
History Exchange

Q: How do we handle a client lying about its history?

A: Client commits to a history before discovering partner’s history

- Under-reporting decreases number of useful updates exchanged
- Over-reporting risks eviction
Q: How do we encourage a rational client to send a briefcase?
**Briefcase Exchange**

**Q:** How do we encourage a rational client to send a briefcase?

**A:** Client gives key only after swapping briefcases
Q: How do we encourage a rational client to send only appropriate briefcases?
Q: How do we encourage a rational client to send only appropriate briefcases?

A: Hold client accountable for contents of briefcase
Valid Briefcase Exchange

Q: How do we encourage a rational client to send only appropriate briefcases?

A: Hold client accountable for contents of briefcase

- Briefcase contains encrypted updates and ids of updates
- Inconsistencies risk eviction
- Decryption key is reproducible by broadcaster
Q: How do we encourage a rational client to send the appropriate key?
Q: How do we encourage a rational client to send the appropriate key?
Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests

- Rational client minimizes cost by sending key
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests
   - Rational client minimizes cost by sending key
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests

- Rational client minimizes cost by sending key
- Rational client proactively sends key
Key Exchange

Q: How do we encourage a rational client to send the appropriate key?

A: Repeated Key Requests
  - Rational client minimizes cost by sending key
  - Rational client proactively sends key
  - Hold client accountable for key responses
BAR Gossip Overview

Balanced Exchange

- In each round:
  - Select partner
  - Exchange histories
  - Trade equal number of updates

Incentive compatible!

Optimistic Push

- In each round:
  - Select partners
  - Exchange histories
  - Trade possibly unequal numbers of updates

Safety net for lagging peers
BAR Gossip Overview

Balanced Exchange

- In each round:
  - Select partner
  - Exchange histories
  - Trade equal number of updates

- Incentive compatible!

Optimistic Push

- In each round:
  - Select partner
  - Exchange histories
  - Trade possibly unequal numbers of updates

- Safety net for lagging peers
BAR Gossip Overview

**Balanced Exchange**

- In each round:
  - Select partner
  - Exchange histories
  - Trade equal number of updates

**Optimistic Push**

- In each round:
  - Select partner
  - Exchange histories
  - Trade possibly unequal numbers of updates

Incentive compatible!

Safety net for lagging peers
Optimistic Push

\begin{align*}
\text{History exchange} & \quad \{u_2, u_4, u_5, u_6, u_7, u_8, u_9\} \\
\{u_1, u_3\} &
\end{align*}
Optimistic Push

\[ \{u_2, u_4, u_5, u_6, u_7, u_8, u_9\} \]

\[ \{u_1, u_3\} \]

History exchange

\[ \langle \text{ids}, \{u_7, u_8, u_9\} \rangle \]
Optimistic Push

A

\{u_2, u_4, u_5, u_6, u_7, u_8, u_9\}

History exchange

B

\{u_1, u_3\}
Optimistic Push

History exchange

\{u_2, u_4, u_5, u_6, u_7, u_8, u_9\}

\{u_1, u_3\}
Q: How do we encourage a lagging client to send as many updates as possible?
Optimistic Push

Q: How do we encourage a lagging client to send as many updates as possible?

A: Require both briefcases to have the same number of items.

If necessary, include junk.
Q: How do we encourage a lagging client to send as many updates as possible?

A: Require both briefcases to have the same number of items

If necessary, include junk
Optimistic Push

Q: How do we encourage a lagging client to send as many updates as possible?

A: Require both briefcases to have the same number of items

- If necessary, include junk
- Junk is larger than an update
Optimistic Push

Q: How do we encourage a lagging client to send as many updates as possible?

A: Require both briefcases to have the same number of items
   🔄 If necessary, include junk
   🔄 Junk is larger than an update
BAR Gossip Recap

Balanced Exchange

In each round:
- Select partner
- Exchange histories
- Trade equal number of updates

Incentive compatible!

Optimistic Push

In each round:
- Select partner
- Exchange histories
- Trade possibly unequal numbers of updates

Explore strategy space experimentally
FlightPath Experiments

- **Setup:** 45 Emulab clients, each update multicast to random 3 clients
- **Goal:** evaluate Optimistic Push strategy space
  - Which strategies are attractive?
  - Which strategies are attractive with failures?
## Alternate Strategies in Optimistic Push

<table>
<thead>
<tr>
<th></th>
<th>Responds with updates</th>
<th>Responds with junk</th>
<th>Doesn’t respond</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initiates Pushes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Does not initiate pushes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Alternate Strategies in Optimistic Push

<table>
<thead>
<tr>
<th>Initiates Pushes</th>
<th>Responds with updates</th>
<th>Responds with junk</th>
<th>Doesn't respond</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Follow Protocol</strong></td>
<td>Follow Protocol</td>
<td>Follow Protocol</td>
<td>Follow Protocol</td>
</tr>
<tr>
<td><strong>Does not initiate pushes</strong></td>
<td>Does not initiate pushes</td>
<td>Does not initiate pushes</td>
<td>Does not initiate pushes</td>
</tr>
<tr>
<td></td>
<td>Responds with updates</td>
<td>Responds with junk</td>
<td>Doesn’t respond</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------</td>
<td>-------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Initiates Pushes</strong></td>
<td><strong>Follow Protocol</strong></td>
<td><strong>Wasteful Strategy</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Does not initiate pushes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Alternate Strategies in Optimistic Push

<table>
<thead>
<tr>
<th>Initiates Pushes</th>
<th>Responds with updates</th>
<th>Responds with junk</th>
<th>Doesn’t respond</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Follow Protocol</strong></td>
<td><strong>Wasteful Strategy</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Does not initiate pushes</strong></td>
<td></td>
<td></td>
<td><strong>Other Strategies</strong></td>
</tr>
</tbody>
</table>
Convergence Graph

Follow Protocol
Convergence Graph

![Graph showing probability of missing an update over time for 'Follow Protocol' and 'Other Strategies'. The graph plots the probability on a logarithmic scale against time in seconds.]
Reliability with Byzantine Viewable

Follow Protocol

Other Strategies

Probability of receiving an update vs. Proportion of Byzantine nodes
Conclusions

BAR Gossip:
- Balanced Exchange: provable, ~98%
- Optimistic Push: ~99.9%

Two key ideas:
- Verifiable partner selection
- Fair enough exchange

Currently working on:
- Dynamic membership
- Partial membership
- Network awareness
Backup Slides
Optimistic Push’s Effect

Probability of receiving an update vs. Proportion of rational nodes.

- Traditional gossip
- Balanced
- Balanced + Optimistic
Why Resend Key Requests?

- Cost to A is small compared to big benefit of unlocking briefcase
- Cost to B is large compared to small benefit of not sending key
TCP AND UDP

UDP necessary so that each peer *believes* its partner will send key requests

![Diagram showing TCP and UDP exchanges](Image)
Why Reject?

• Peer terminates an exchange if that peer expects nothing useful from its partner
• Peer expects something useful only if it believes in fair enough exchange
• Fair enough exchange mechanism relies on mutual fear of eviction
How Does Eviction Work?

- Broadcaster evicts clients by attaching eviction notices onto updates
- Broadcaster periodically asks clients to testify against their peers
- Clients testify because they expect nothing useful from future exchanges with those peers
# End-to-End Metric

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Jitter</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Follow Protocol</td>
<td>0.48%</td>
<td>1.16%</td>
</tr>
<tr>
<td>Wasteful Strategy</td>
<td>0.32%</td>
<td>0.78%</td>
</tr>
<tr>
<td>Initiate OP, Decline OP</td>
<td>11.59%</td>
<td>6.22%</td>
</tr>
<tr>
<td>Respond to OP with useful</td>
<td>18.10%</td>
<td>6.08%</td>
</tr>
<tr>
<td>Respond to OP with junk</td>
<td>14.76%</td>
<td>9.44%</td>
</tr>
<tr>
<td>Never run OP</td>
<td>47.94%</td>
<td>7.52%</td>
</tr>
</tbody>
</table>
• Colluding nodes use unrealistic protocol
• BAR Gossip still robust for small colluding groups
• For large groups, colluding nodes may not trust each other
**Denial-of-Service**

DoS Resistant Unforgeable Multicast (DRUM)

- Resource bounding
- Random port hopping