Reducing Round Complexity of Byzantine Broadcast

Linda Chen
Mentor: Jun Wan
Byzantine Agreement and Broadcast

- n users, up to f are corrupted
- Honest users must agree

Byzantine Agreement

Byzantine Broadcast
Properties of Byzantine Broadcast

At the end of the protocol, each user \( i \) outputs \( b_i \)

**Consistency**: all honest users agree
Properties of Byzantine Broadcast

Validity: if the leader is honest, all honest users output the leader’s bit

Liveness: all honest users will eventually terminate
Assumptions

**Synchronous**: messages sent in round $r$ are received before round $r+1$

**Digital signatures**: each message is accompanied by a user’s signature
Honest or Dishonest Majority

Honest Majority ($f < n/2$)  

Dishonest Majority ($f > n/2$)
Static or Adaptive Adversary

**Static adversary:** corrupts up to $f$ users at the beginning of the protocol

**Adaptive adversary:** corrupts users in the middle of the protocol

- If a user is corrupted in round $r$, the adversary can inject, modify, or remove messages sent in round $r$
- Users that are corrupted stay corrupted
## Expected Round Complexity Results

<table>
<thead>
<tr>
<th></th>
<th>Best Previous Result</th>
<th>Our Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honest Majority Static Adversary</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Honest Majority Adaptive Adversary</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Dishonest Majority</td>
<td>3d per epoch</td>
<td>3d-2 per epoch</td>
</tr>
</tbody>
</table>

- Communication complexity is $\tilde{O}(n^4)$; previous honest majority result is $O(n^2)$
Attacks by Corrupt Users

1. \( w \) sends equivocating messages \( \rightarrow u \) and \( v \) detect equivocation from \( w \)

2. \( w \) does not send message to \( u \) \( \rightarrow v \) knows at least one of \( u \) or \( w \) is corrupt
Previous work: Trust Graph

- $n$ nodes, edge between nodes = trust
- Maximum diameter of $d = \lceil n/h \rceil + \lfloor n/h \rfloor - 1$

$w$ sends equivocating messages:

$w$ does not send to $u$:
Previous work: TrustCast Protocol

- s wants to send a message to all users
- For every round $1 \leq r \leq d$:
  - If a user does not receive s’ message, remove edges with all neighbors that are distance less than r from s

If u does not receive s’ message in round 3, remove edge with v
Byzantine Broadcast Protocol

For each epoch:

- **Propose**: the leader TrustCasts its input bit to other users
- **Vote**: users TrustCast the leader’s proposal to other users
- **Commit**: if a user receives votes on the leader’s proposal from everyone in their trust graph, output the proposed bit and TrustCast a commit message to other users

**Terminate**: if a user receives commit messages from everyone in their trust graph, terminate
Reducing Round Complexity of TrustCast Protocol

- **d rounds of TrustCast**: every user $u$ either (1) received $s'$ message or (2) $s$ is removed from $u$’s trust graph
- **d-1 rounds of TrustCast**: either (1), (2), or $s$ is distance $d$ from $u$ in $u$’s trust graph

![Diagram showing trust relationships and message transmission](attachment:trust_cast_diagram.png)

- $u$ does not receive $s'$ message
Reduced Round Complexity: Propose

- For Propose and Vote phases: use modified TrustCast protocol
- **Propose**: at least one honest user $u$ receives proposal

![Diagram showing L connected to u with a dashed line labeled <d]
Reduced Round Complexity: Vote

Vote: every honest user $u$ receives a vote on the leader’s proposal from at least one other honest user $v$

- If all honest users are distance $d$ from $u$ or distance $d$ from $L$, then there needs to be more than $n$ users.
Dishonest Majority Round Complexity

- **Propose:** $d-1$ rounds
- **Vote:** $d-1$ rounds
- **Commit:** $d$ rounds

$3d-2$ rounds per epoch
Honest Majority: Trust Array

- **Trust array**: $u.A[v,w] = 1$ (trust) or $0$ (not trust)

1. $w$ sends equivocating messages
   
   all users $u$ set $u.A[v,w] = 0$ for all $v$

2. $w$ does not send message to $v$
   
   all users $u$ set $u.A[v,w] = 0$
Honest Majority Protocol

- \( d = \lceil n/h \rceil + \lfloor n/h \rfloor - 1 = 2 \)
- Propose, Vote: \( d-1 = 1 \) round
- Commit: \( d = 2 \) rounds
- **To send messages**: broadcast to all users
- **To commit**: \( u \) receives votes from all users \( v \) such that \( u.A[u,v] \times u.A[v,L] = 1 \)
- **To terminate**: \( u \) receives \( f+1 \) commit messages
Honest Majority Protocol

Propose: a → a
Vote: a → a
Commit (1): a → a
Commit (2): a → a

b → b
b → b

C → C
C → C

terminate
terminate
Honest Majority, Adaptive Adversary

Adaptive adversary repeatedly corrupts the leader

- Delay leader election

Adaptive adversary forges equivocating proposals after leader election

- **Propose round 1**: every user broadcasts a proposal
- **Propose round 2**: relay all proposals
Honest Majority Round Complexity

- If leader is honest, all users terminate in that epoch
- Expected 2 epochs

<table>
<thead>
<tr>
<th>Adversary Type</th>
<th>Rounds per Epoch</th>
<th>Total Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Adversary</td>
<td>4 rounds</td>
<td>Expected 8 rounds total</td>
</tr>
<tr>
<td>Adaptive Adversary</td>
<td>5 rounds</td>
<td>Expected 10 rounds total</td>
</tr>
</tbody>
</table>
Acknowledgments

- Mentor: Jun Wan
- MIT PRIMES program

Thank you!