Lecture 11: Weaker Broadcast & Agreement in Asynchrony

CS 539 / ECE 526

Distributed Algorithms
Impossibilities of Fault Tolerance in Asynchrony

• Under asynchrony, no broadcast protocol can tolerate a single crash fault (sender)

• Under asynchrony, no deterministic agreement protocol can tolerate a single crash fault

  – Fischer-Lynch-Paterson, 1985
What can we do?

- Consider easier problems
- Randomization
- Consider easier models (partial synchrony)

- Agreement, total order bcast, and replication possible in psync or async with randomization
  - Single-value broadcast still impossible
Outline

• Consider easier problems in asynchrony
  – Reliable and consistent broadcast
  – Graded agreement
Relaxing the Broadcast Problem

- \( n \) parties, including a designated sender with an input \( x \), up to \( f \) faulty

- Safety: no different outputs
- Liveness: everyone outputs
- Validity: sender honest \( \rightarrow \) everyone outputs \( x \)

- Cannot ask for both “liveness under faulty leader” and “validity under honest leader”
- Will relax liveness under faulty leader
Reliable Broadcast (RBC)

• n parties, including a designated sender with an input $x$, up to $f$ faulty

• Safety: no different outputs

• Liveness: either everyone outputs or no one outputs

• Validity: sender honest $\rightarrow$ everyone outputs $x$
A Simple Byzantine RBC

• $f < n/3$, use signatures
• Sender proposes $x$; replicas send signed votes
• Upon receiving $n-f$ votes for $x$, output $x$, and forward these votes to all other replicas

Replica 1 (Sender)
Replica 2
Replica 3
Replica 4
Safety: Quorum Intersection

- Some honest outputs $v \rightarrow 2f+1$ votes for $v \rightarrow f+1$ honest votes for $v \rightarrow$ at most $2f$ votes for $v' \rightarrow$ no honest outputs $v'$

$n = 3f + 1$
Liveness and Validity

• Validity: an honest sender proposes $v$ to all $\rightarrow$ all honest eventually vote $v$ $\rightarrow$ all output $v$

• Liveness: an honest outputs $\rightarrow$ it forwards a quorum of votes to all honest $\rightarrow$ all output
  – Hence, either all output or no one outputs
  – A quorum of votes is a *transferrable certificate*

• How does a malicious sender prevent liveness?
Byzantine RBC Efficiency

• Round complexity:
  – Under good leader: commit in 2, terminate in 3

• Communication complexity:
  – $O(n^2)$ messages
  – $O(n^3|\sigma|)$ bits
Bracha’s Byzantine RBC

- Leader proposes $x$; replicas send $\text{vote}_1$
- Upon receiving $n-f$ matching $\text{vote}_1$, send $\text{vote}_2$
- Upon receiving $f+1$ matching $\text{vote}_2$, send $\text{vote}_2$
- Upon receiving $n-f$ matching $\text{vote}_2$, output

Replica 1 (Sender)

\[ n = 3f+1 \]

Replica 2

Replica 3

Replica 4

propose \hspace{1cm} \text{vote}_1 \hspace{1cm} \text{vote}_2
Bracha RBC Correctness

• Safety: quorum intersection

• Validity: an honest sender proposes $v$ to all $\rightarrow$ all vote1 $\rightarrow$ all vote2 $\rightarrow$ all output

• Liveness: an honest outputs $\rightarrow$ $n-f$ vote2 $\rightarrow$ $n-2f = f+1$ vote2 from honest $\rightarrow$ all vote2 $\rightarrow$ all output
  – An ”amplification” of vote2
Bracha RBC Efficiency

• Round complexity:
  – 3 or 4 rounds

• Communication complexity:
  – $O(n^2)$ msgs
  – $O(n^2)$ bits
  – Signature-free
Consistent Broadcast (CBC)

- $n$ parties, including a designated sender with an input $x$, up to $f$ faulty

- **Safety**: no different outputs
- **Liveness**: none
- **Validity**: sender honest $\rightarrow$ everyone outputs $x$
A Simple Byzantine CBC

- $f < \frac{n}{3}$
- Sender **proposes** $x$; replicas send **votes**
- Upon receiving $n-f$ **votes** for $x$, output $x$

![Diagram of Byzantine CBC protocol](image-url)
Correctness and Efficiency

• Safety: quorum intersection
• Validity: an honest sender proposes $v$ to all $\rightarrow$ all vote $\rightarrow$ all output

• 2 rounds
• $O(n^2)$ messages (all-to-all voting)
Outline

• Consider easier problems in asynchrony
  – Reliable and consistent broadcast
  – Graded agreement
Graded Agreement (GA)

• n parties, each with an input, up to f faulty
• Each party outputs value y and “grade” bit g
  – g is roughly “confidence”

• Liveness: everyone outputs
• Validity: every non-faulty inputs x → every non-faulty outputs (x, 1)
• Safety: no distinct confident outputs: no two non-faulty output (y, 1) and (y’, 1) with y ≠ y’
  – Other variants exist
Async GA for $f < n/2$ Crash

• Party $j$ has input $x_j$:
  – **Round 1**: party $j$ sends $(\text{vote}, x_j)$
    • Wait for $n-f = f+1$ vote msgs ($n=2f+1$)
  
  – If all $f+1$ votes are for the same $x$, then output $(x, 1)$;
    Else, output $(x', 0)$ for any $x'$ with one vote
    • Will just output own input
GA Correctness

- **Liveness**: waits for \( n-f \) msgs, will get that many

- **Validity**: same input \( x \) $\rightarrow$ matching votes $\rightarrow$ everyone outputs \((x, 1)\)

- **Safety**: quorum intersection
Quorum Intersection (Crash)

- Impossible to have two non-faulty party output $(x,1)$ and $(x',1)$ for $x' \neq x$

$$n = 2f + 1$$
Graded Agreement (GA)

- n parties, each with an input, up to f faulty
- Each party outputs value y and “grade” bit g
  - g is roughly “confidence”

- Liveness and validity as before
- Many variants of safety:
  - S1: No (y, 1) and (y’, 1) for y ≠ y
  - S2: One outputs (y, 1), all output (y, *)
  - S3: No (y, *) and (y’, *) for y ≠ y’, y ≠ ⊥, y’ ≠ ⊥
GA Safety Variant Relations

– S1: No \((y, 1)\) and \((y', 1)\) for \(y \neq y'\)
– S2: One outputs \((y, 1)\), all output \((y, \ast)\)
– S3: No \((y, \ast)\) and \((y', \ast)\) for \(y \neq y', y \neq \bot, y' \neq \bot\)

– S2 strictly stronger than S1
– S3 strictly stronger than S1
  • With a reasonable assumption that \(\bot\) cannot be output with confidence
– S3 does not imply S2: \((y, 1)\) and \((\bot, 1)\)
– S2 does not imply S3: \((y, 0)\) and \((y', 0)\)
Async GA for $f < \frac{n}{2}$ Crash

- Party $j$ has input $x_j$:
  - Round 1: party $j$ sends $(\text{vote1, } x_j)$
    - Wait for $n-f = f+1$ vote1 msgs ($n=2f+1$)
  - Round 2: if all $f+1$ vote1 are for the same $x$, party $j$
sends $(\text{vote2, } x)$; else, sends $(\text{vote2, } \perp)$
    - Wait for $n-f = f+1$ vote2 msgs ($n=2f+1$)
  - If all $f+1$ vote2 are for the same $x$, then output $(x, 1)$;
    Else if there is one vote2 for $x$, then output $(x, 0)$;
    Else, output $(\perp, 0)$. 
GA Correctness

• Liveness: waits for n-f msgs, will get that many

• Validity: same input x \rightarrow matching\ vote_1 \rightarrow \\
  matching\ vote_2 \rightarrow \rightarrow everyone\ outputs\ (x, 1)

• Safety: quorum\ intersection \rightarrow at\ most\ one\ \\
  non-\perp\ value\ in\ vote_2 \rightarrow both\ S2\ and\ S3
Summary

• Broadcast (the strongest formulation) is impossible with a single crash under psync.

• Weaker primitives are possible in async:
  – Reliable or consistent broadcast
  – Graded agreement
  – May even be useful in sync

• Quorum intersection & certificates are common tools in psync / async.
Graded Broadcast (Gradecast)

• n parties, including a designated sender with an input x, up to f faulty

• Each party outputs value y and “grade” bit g
  – g is roughly “confidence”

• Liveness: everyone outputs

• Validity: every non-faulty inputs x → every non-faulty outputs (x, 1)

• Safety: many variants similar to GA

• Impossible in psync/async but useful in sync