ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Lecture 7: Dolev-Strong Byzantine Broadcast

CS 539 / ECE 526

Distributed Algorithms

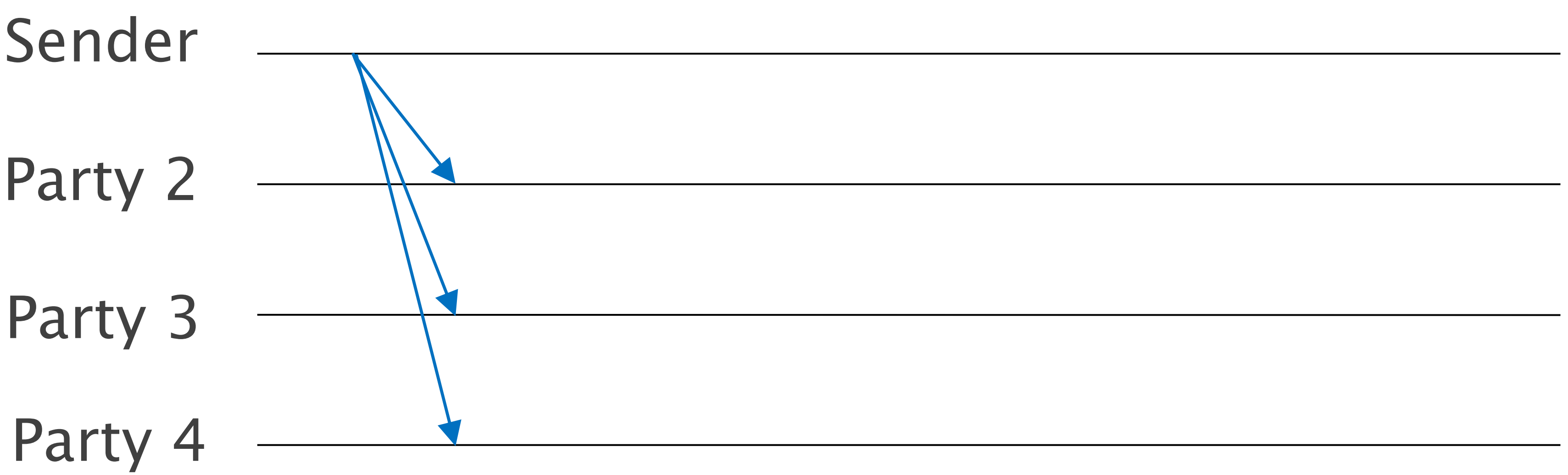
Tolerating Faults is Hard!

- In general, when there are faults, we almost always study the consensus problem. Why?
- Partly because it is the easiest problem!
- But still quite hard! (and deceptively simple)

- Let us start from the simplest model
  - f crash faults out of n parties in total
  - Pair-wise reliable links, lockstep synchrony
  - Binary input: x is 0 or 1
- Try to come up with an algorithm!

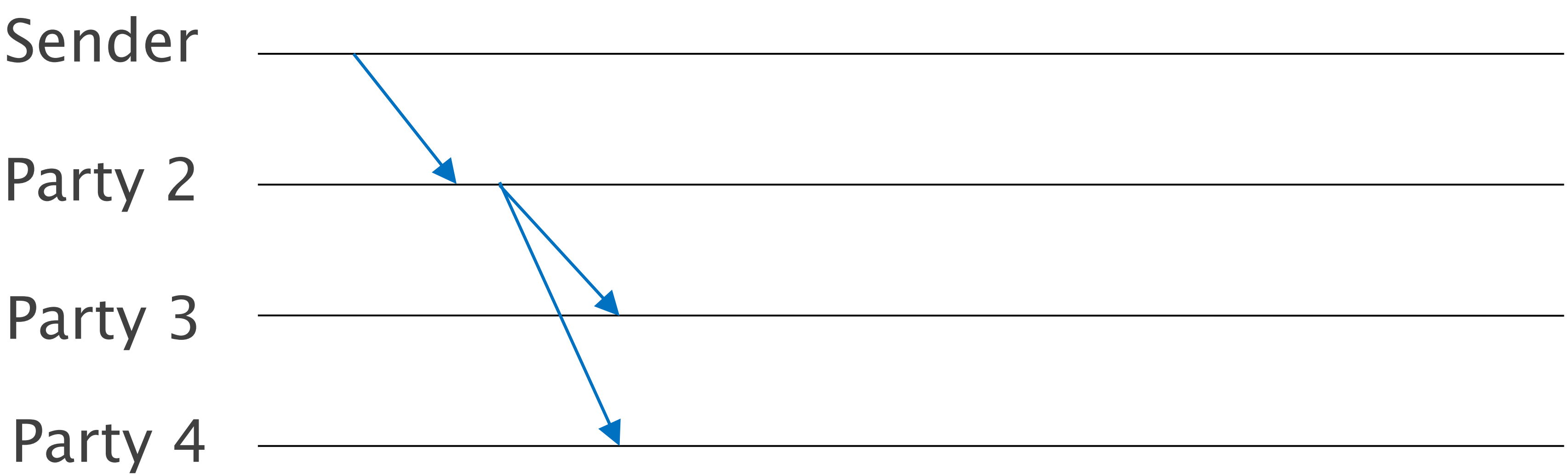
First Try

- Output what sender says?
- Lose safety if sender crashes half-way
  - Some output sender input; others output Ø / ⊥



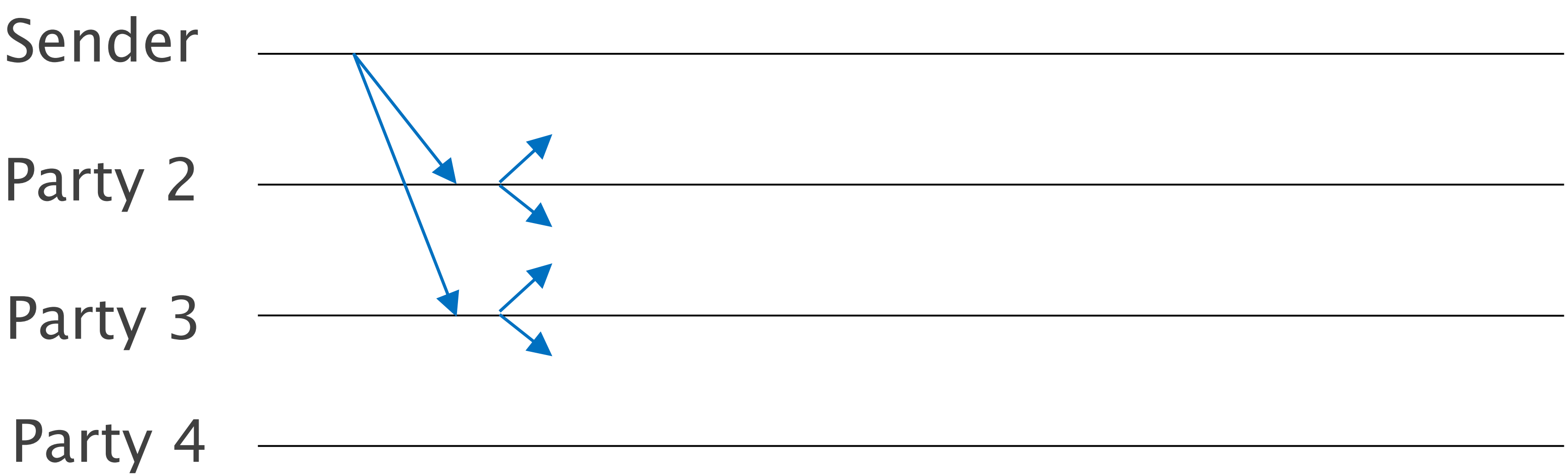
Second Try

- Output if anyone echoes sender's message?
- Lose safety if the "echoer" crashes half-way



Third Try

- Output if majority echoes sender's message?
- Lose safety if some receive majority echoes but others do not

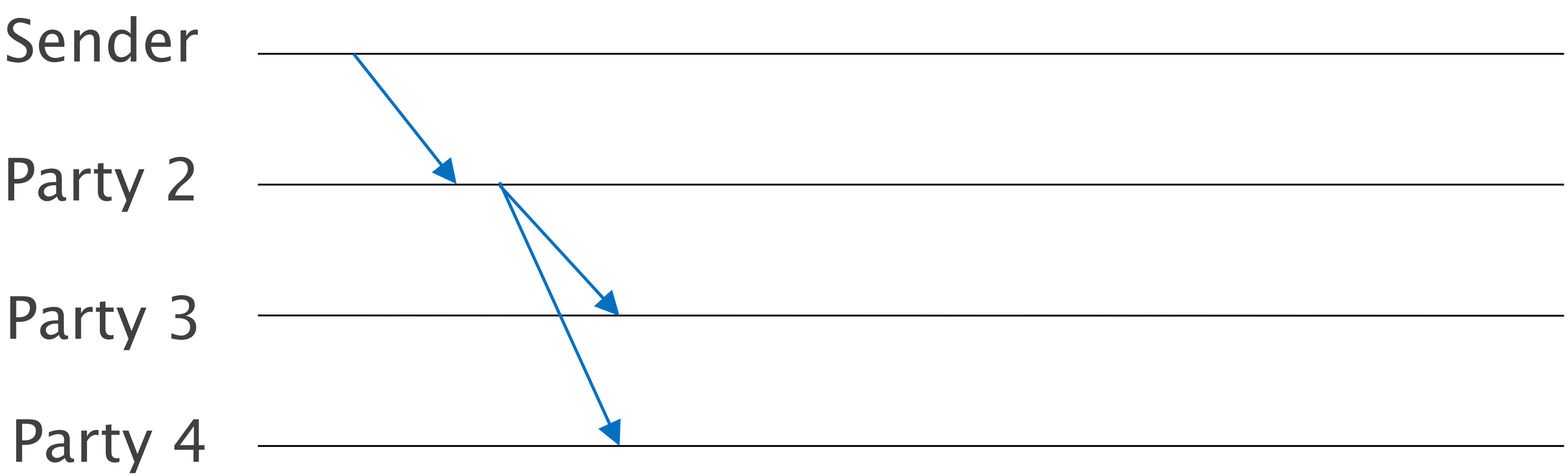


Outline

- Flooding broadcast with crash faults
- Dolev-Strong Byzantine broadcast
- Fault tolerant clock synchronization

Flooding Broadcast

- Sender sends its input to all
- In each round, echo to all if you receive a value
- After f+1 rounds, output v (if seen v) or Ø / ⊥



Flooding Broadcast

- Liveness and validity obvious
- Safety: if any non-faulty party outputs $v \neq \perp$, then all non-faulty parties output v
  - Proof sketch: When does this non-faulty receive v?
  - If not last round, this party echoes v to everyone
  - If last round, exists propagation chain of length f+1; last party is non-faulty and echoes v to everyone

Complexity of Flooding Broadcast

- Round complexity: f+1 rounds

- Communication complexity
  - $O(n^2)$ msgs each of input size
  - Not $O(n^2 f)$!

Challenges for Byzantine

- What goes wrong in flooding broadcast if there are Byzantine faults?
  - Sender sends multiple values
  - Byzantine parties “make up” values
  - Byzantine parties delay forwarding

Outline

- Flooding broadcast with crash faults
- Dolev-Strong Byzantine broadcast
- Fault tolerant clock synchronization

Dolev-Strong

- Solves broadcast with $f < n$ Byzantine faults
  - Resembles flooding broadcast with a clever twist
- Proposed in 1983, still the best in its setting
- Lock-step synchrony, pairwise reliable links
- Handles multi-value inputs (not just binary)
- **Use digital signatures**

Digital Signatures

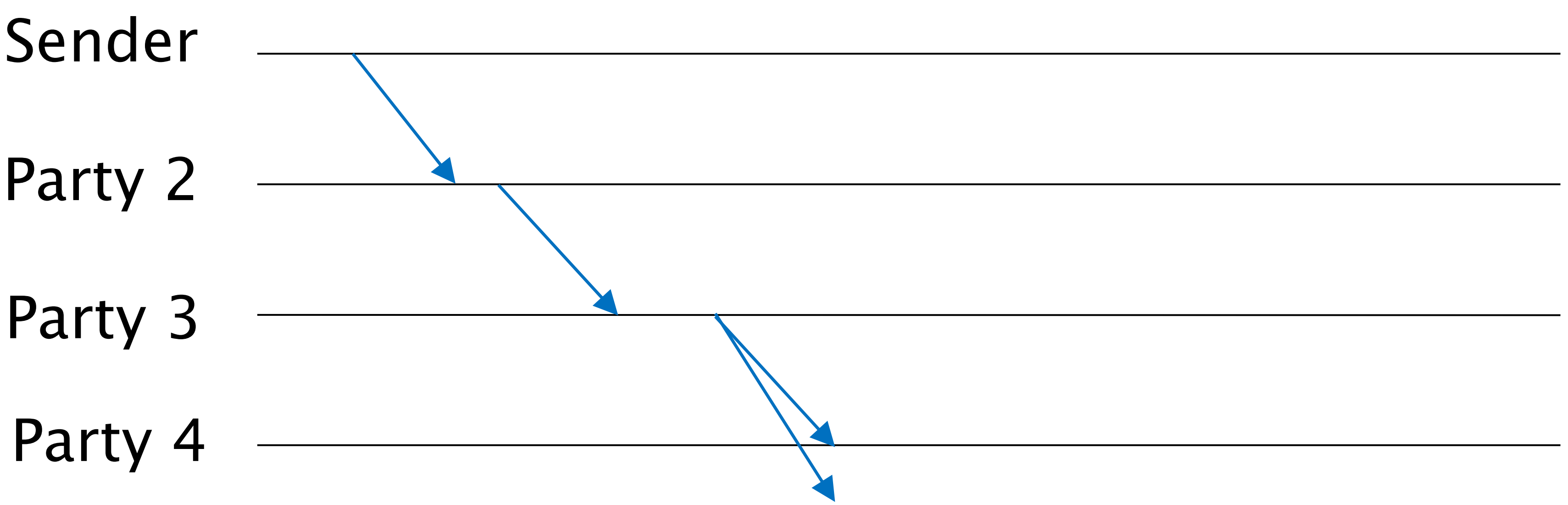
- A basic cryptographic primitive

- A signer has a secret key sk
- Everyone has a corresponding public key pk
- Signer: Sign(sk, msg) → σ (signature)
- Anyone else: Verify(pk, msg, σ) → True/False
  - Anyone can verify msg came from signer

- Why does it matter for consensus?
  - Allows verifying forwarded msgs

Dolev-Strong Intuition

- Use flooding, but msgs need to be signed, first by sender, then by each forwarding party
  - Nested signing (((x || σ_s) σ_2) σ_3) σ_4 …



Dolev-Strong Protocol

- In round r, if a party receives a chain of r signatures (innermost is the sender's) on v
  - Extract v
  - If this is the last round ($r=f+1$), terminate; Else,
    Sign and forward the chain of (now) $r+1$ signatures
    - Signature chain prevents delayed forwarding
  - Output v if extracting only v; else, output $\perp$

Dolev-Strong Safety

- Lemma: If one honest extracts v, then every honest extracts v
  - Proof: when does this honest party extracts v?
  - Before last round → this honest party echoes v
  - Last round → signature chain of length f+1 → one of them is honest and echoes v
  - An honest party always echoes with valid sig chain, so every honest party extracts v

Dolev-Strong Protocol

- In round r, if a party receives a chain of r signatures (innermost is the sender's) on v
  - Extract v
  - If this is the last round (r=f+1), terminate; Else,

    Sign and forward the chain of (now) r+1 signatures
  - **Each party forwards at most two values**
    - To avoid excessive communication
  - Output v if extracting only v; else, output ⊥

Dolev-Strong Correctness

- Liveness obvious
- Validity: if sender is honest, everyone extracts v and nothing else
  - Any value requires innermost sender signature
  - Honest sender will not double-sign

Dolev-Strong Safety

- If honest party i extracts >= 2 values, everyone extracts >= 2 values
  - Party i or last party in sig chain forwards >= 2 values
- If i extracts v and only v, so does everyone
  - If some honest j extracts v' ≠ v, i extracts v' too
- If i extracts no value, so does everyone
  - If some honest j extracts v, i extracts v too

Dolev-Strong Complexity

- $f+1$ rounds

- $2n^2$ messages
- Each message up to $(f+1)|\sigma|$
- Communication complexity in bits: $O(n^2 f|\sigma|)$

Outline

- Flooding broadcast with crash faults
- Dolev-Strong Byzantine broadcast
- Fault tolerant clock synchronization

Fault Tolerant Clock Sync

- Previously, we have seen clock sync algorithms to sync distributed clocks within U = D - d
  - Use a reference, everyone syncs within U/2 to ref
  - Periodic sync to handle drift
  - Not fault tolerant
- Today: clock synchronization tolerating crash and Byzantine faults

Crash Tolerant Clock Sync

- Synchronize every T

  Upon AC == K*T

      send "sync K" to all

  Upon receiving "sync K" for the first time

      send "sync K" to all

      set adj so that AC = K*T
- Correctness: everyone at most D apart from the first non-faulty to send "sync K"
- Efficiency: $O(n^2)$ msgs

Byzantine Tolerant Clock Sync

- Synchronize every T

  Upon AC == K*T

  sign and send "sync K" to all

  Upon receiving f+1 signed "sync K"

  send f+1 signed "sync K" to all

  set adj so that AC = K*T
- Correctness: everyone at most D apart from the first non-faulty to send f+1 "sync K"
- Efficiency: $O(n^2)$ msgs but $O(n^2 f|\sigma|)$ bits

Summary

- Dolev-Strong: classic (but still best) sync Byzantine broadcast using signatures
  - $f < n$ Byzantine faults
  - $f+1$ rounds
  - $2n^2$ msgs
  - $O(n^2 f|\sigma|)$ bits of communication

- Fault tolerant clock sync within D
  - Not as good as non-fault-tolerant ones (within U)
  - More advanced algorithms exist