

Department of Computer Science

PURDUE
UNIVERSITY

CS505: Distributed Systems

Lecture 16: Probabilistic Consensus

Outline

- ▶ **Randomized Binary Consensus**
- ▶ **From Binary Consensus to Multivalued Consensus**
- ▶ **Hybrid Multivalued Consensus**

(Uniform) Binary Consensus

To start consensus a process proposes either `false` (0) or `true` (1)

I. Validity

Any value decided is a value proposed

II. Agreement

No two processes decide differently

III. Integrity

No process decides twice

IV. Termination

Every correct process eventually decides

Binary 1-Consensus

I. Validity

Any value decided is a value proposed

II. Agreement

No two processes decide differently

III. Integrity

No process decides twice

IV. Termination

With probability 1 every correct process eventually decides

For Process p_i

[Ben-Or'83] tolerates crashes of a minority f of n processes

upon propose(v):

$r \leftarrow 0, \text{ est} \leftarrow v$ // *current round and estimate*

 while true do

$r \leftarrow r + 1$

 send(phase1, $r, \text{ est}$) to all processes // *phase 1*

 wait for (receive(phase1, r, u) from a majority)

 if all received with same u then

 send(phase2, r, u) to all processes // *phase 2*

 else

 send(phase2, r, \perp) to all processes // *phase 2*

 wait for (receive(phase2, r, u) from a majority)

 if some $u \neq \perp$ then

$\text{ est} \leftarrow u$

 if all received with same $u \neq \perp$ then

 decide(u)

 else then

$\text{ est} \leftarrow \text{ coinFlip}$

Analysis

► Agreement

- If some process decides v , then no other decision can be taken anymore, even if the process crashes
- A majority of confirmations of v are gathered in phase 2, which means that every other process has observed some message with v
 - It retains the value in its `est`
- Since v results from a majority, there can not be another value

► Termination

- How can we *not* get a majority of same values?
 - If a minority of processes fails, and the live majority does not all “happen to” propose the same value
 - Flip coins (note that both values are valid)
 - Thus probabilistic guarantee
- With $n^{1/2}$ failures we get constant average time

From Binary to Multivalued Consensus

▶ Binary Consensus provides only two possible inputs/outputs

- Most decisions require a choice from a larger set
 - E.g., which message to deliver first?
 - In general case, the set is not bound a priori

▶ Can we construct a Multivalued Consensus

- Assuming we have a Binary Consensus algorithm?
- Lets start from the deterministic case

Proposition [MRT'00]

```
init
val  $\leftarrow (\perp, \dots, \perp)$ 
decided  $\leftarrow$  false
k  $\leftarrow$  0

upon deliver(v) from  $p_j$ :
    val[j]  $\leftarrow$  v

upon propose(v):
    broadcast(v)
    while not decided do
        k  $\leftarrow$  k + 1
        prop  $\leftarrow$  (val[k mod n]  $\neq \perp$ )
        proposeB(k, prop)
        r s.t. decideB(k, r)
        if r then wait until (val[k mod n]  $\neq \perp$ )
            decided  $\leftarrow$  true
            decide(val[k mod n])
```

Assessment

▶ Idea

- Every process broadcasts proposition to everybody
- Binary consensus is used to decide who's proposition to adopt
 - Sequence of consensus instances started, first one (k -th consensus) to decide `true` yields the process (k -th process)

▶ What guarantees do we get?

- *Uniform Agreement?*
- Termination?

▶ Broadcast?

- Implementable?

Multivalued Randomized Consensus

- ▶ Can we use the previous scheme to implement a *randomized* multivalued approach?
- ▶ With Binary 1-Consensus?
- ▶ Succinct proposition [EMR'01]

For Process p_i

```
init
  val  $\leftarrow (\perp, \dots, \perp)$ ; decided  $\leftarrow$  false

upon deliverU(val, v) from  $p_j$ :
  val[j]  $\leftarrow$  v

upon deliverU (dec, v):
  decide(v)

upon propose(v): // invocation of decide stops all tasks/input
  broadcastU (val, v); r  $\leftarrow$  0, est  $\leftarrow$  v // current round and estimate
  while true do
    r  $\leftarrow$  r + 1
    send(phase1, r, est) to all processes // phase 1
    wait for (receive(phase1, r, u) from majority of processes)
    if all u are same then est  $\leftarrow$  u
    else then est  $\leftarrow$   $\perp$ 
    send(phase2, r, est) to all processes // phase 2
    wait for (receive(phase2, r, u) from majority of processes)
    if all u  $\neq$   $\perp$  are same then
      broadcastU (dec, u); stop
    else if at least one u  $\neq$   $\perp$  then est  $\leftarrow$  u
    else then est  $\leftarrow$  val[random(1,n)]
```

Assessment

▶ Guarantees?

▶ Why Uniform Reliable Broadcast?

▶ Optimization

- Pick value $\neq \perp$ from process with smallest index *starting* at random index (cycle when reaching n)
- Can accelerate establishment of majority value

References

- ▶ ***Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols.*** M. Ben-Or, ACM PODC'83, 27-30, 1983.
- ▶ ***From Binary Consensus to Multivalued Consensus in Asynchronous Message-Passing Systems.*** A. Mostefaoui, M. Raynal, F. Tronel, IPL 73(5-6): 207-212, 2000.
- ▶ ***Randomized Multi-Valued Consensus.*** P. Ezilchelvan, A. Mostefaoui, M. Raynal, IEEE ISORC'01, 195-200, 2001.