Consensus in Mobile Dynamic Distributed Systems

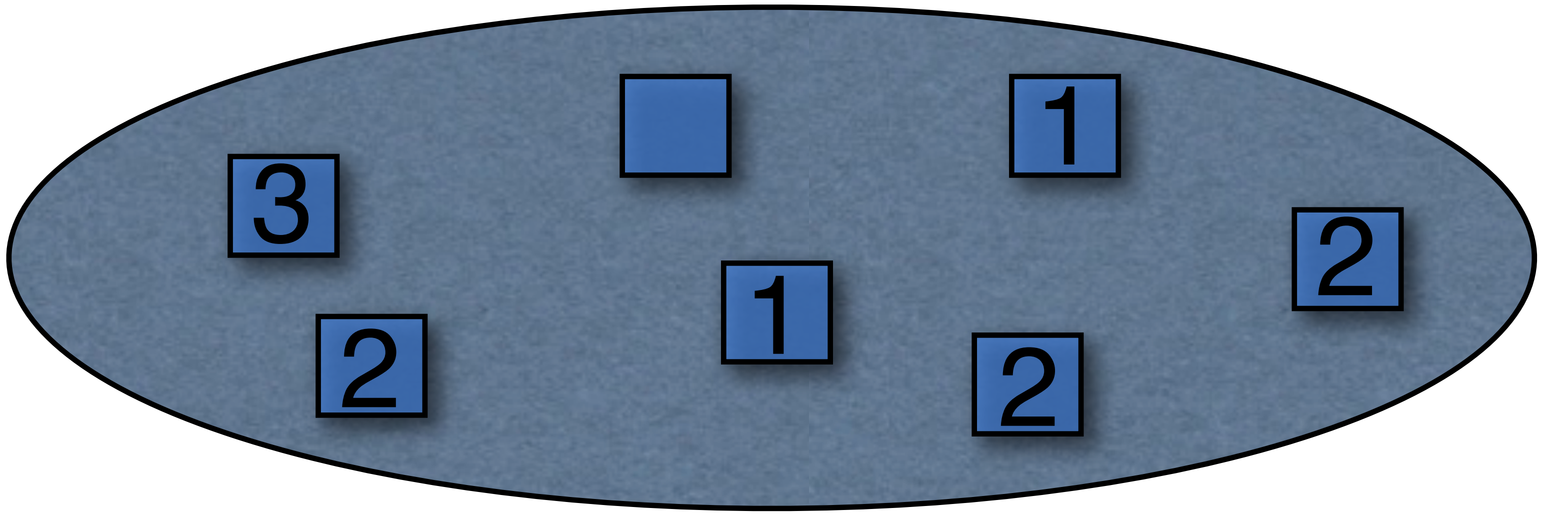
Hyunyoung Lee
Texas A&M University

The Problem

Consider a mobile dynamic distributed system, where processes come and go.

Each process proposes an initial value and the processes will agree on one of those values suggested by correct processes remaining in the system.

This is the strong dynamic consensus problem.



Static vs. Dynamic Models

We model dynamicity by a stochastic model of the joining and leaving processes.

One cannot model the problem by strong consensus with crash failures from classical static models, since the value proposed by leaving processes must be discarded.

Review

Strong Consensus in

Static Distributed Systems

(Static) Strong Distributed Consensus

Each process begins with an input value.

The solution algorithm satisfies:

1. Termination: each process chooses a single output (decision) value and halts
2. Agreement: the decision values of the correct processes are identical
3. Strong Validity: the output value of each correct process is the *input value of some correct* process.

Static Model

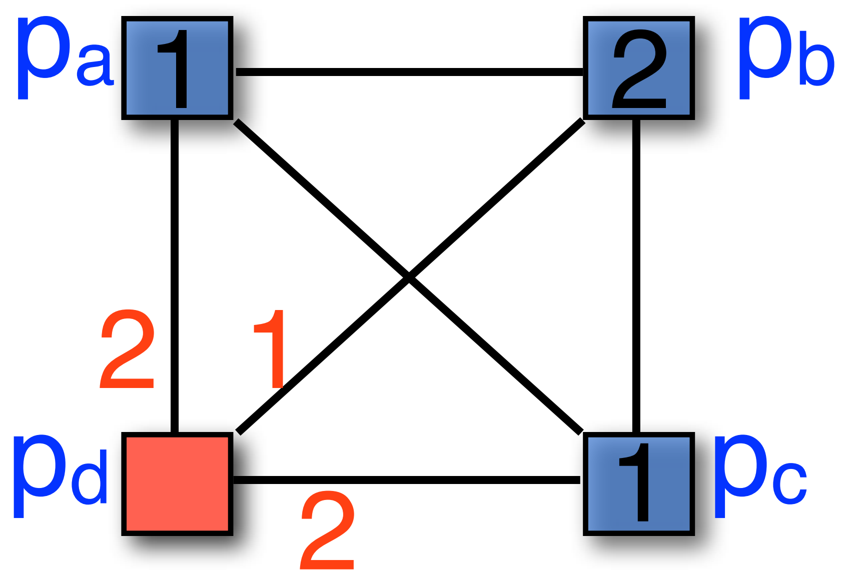
- Synchronous message passing
- Synchronous round: processes first send, then receive, and execute local computation
- The input domain D contains more than 2 values
- A process is either correct or Byzantine faulty. There are *at most f* faulty processes.

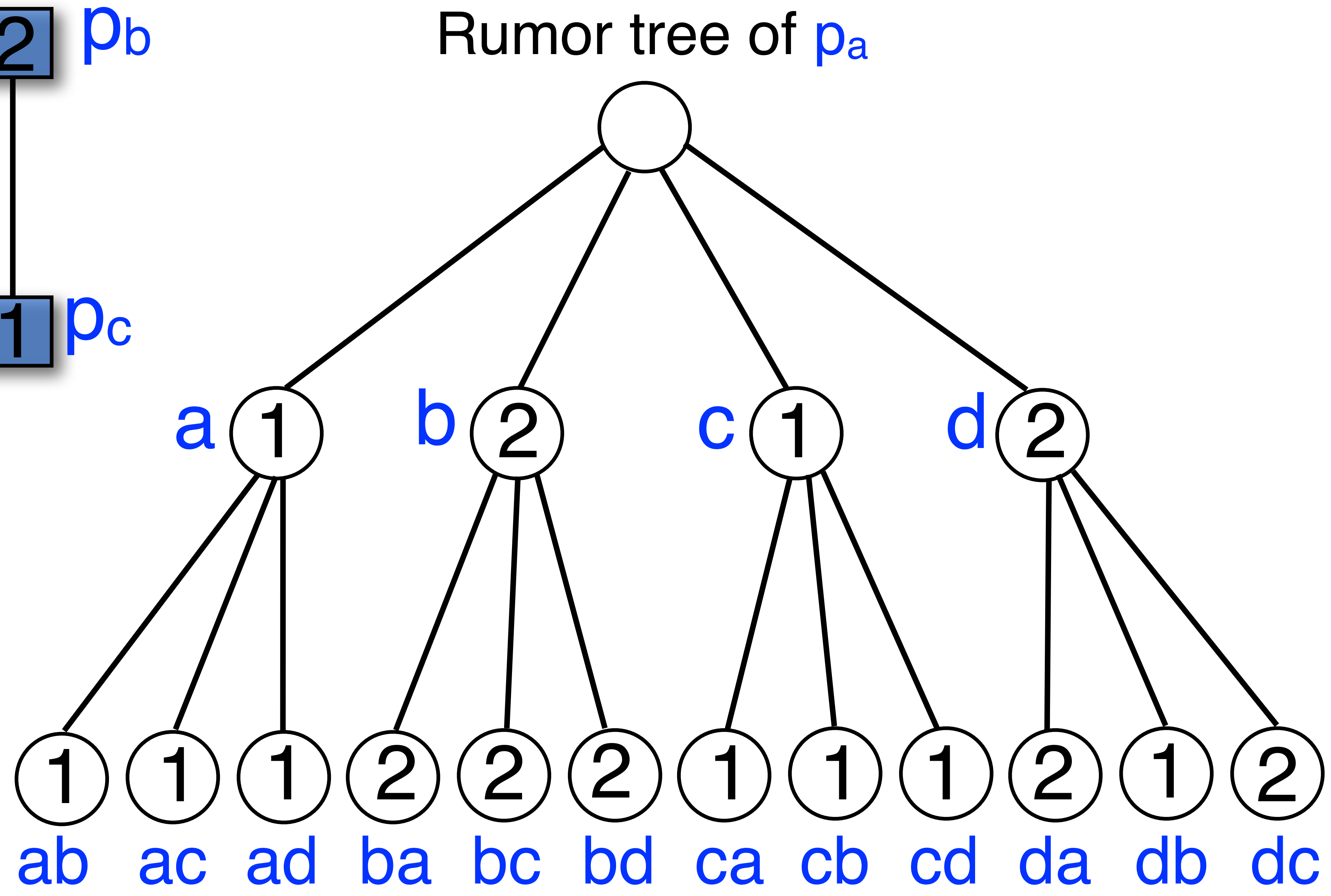
The total number of processes n need be > 3f to have a solution to the problem [Lamport et al.]

Strong Consensus Algorithm

1. Information gathering phase (build rumor tree)

Example: f=1, n=4, p_d is faulty, D={0,1,2}, 0 is default value





Strong Consensus Algorithm

2. Resolve phase

After f+1 rounds, each process applies a recursive function **Resolve** to the root of its tree and decides upon the value returned by the function.

Function **Resolve**:

If applied to a leaf node, returns the value stored at that node.

If applied to a non-leaf node, then the function is applied recursively to the children of that node.

Return the most common value among the values returned; if there is more than one such value, then choose the first one in a known enumeration scheme.

Strong Consensus in Dynamic Distributed Systems

Strong Dynamic Consensus

From round k to round $k+1$, we have

- the set $P_k \setminus P_{k+1}$ of processes leaving the system,
- the set $P_k \cap P_{k+1}$ of processes remaining in the system, and
- the set $P_{k+1} \setminus P_k$ of processes joining the system.

There exists a (final) round d such that

- [Retention] The subset of correct processes in $P_1 \cap P_d$ is not empty,
- [Termination] Every correct process in $P_1 \cap P_d$ enters a decided state,
- [Agreement] The correct processes in $P_1 \cap P_d$ decide on the same value, and
- [Strong Validity] The decided value must belong to the set of initial values of the correct processes in $P_1 \cap P_d$.

Algorithm

Algorithm is similar to the static strong consensus algorithm.

Each process builds up a rumor tree.

If a process p is leaving, every process prunes the branch of its rumor tree that is (sub)rooted at node p.

Use **decide** function on the pruned rumor trees to reach consensus (sim. to resolve).

Safety (1)

- Lemma 1. Suppose that xp is a label in a rumor tree and p is a correct process. Then the value returned by decide (xp) is the same for all correct processes participating in the strong dynamic consensus. Furthermore, for each correct process the value of decide(xp) coincides with the value stored at the node xp in its rumor tree.
- Lemma 2. Suppose that a node x in a rumor tree has a correct frontier. Then the value of decide(x) is the same in the rumor tree of each correct process. In particular, for each correct process, decide() returns the same value when applied to the root of the rumor tree.

Safety (2)

Theorem 3. Suppose that n processes invoke our algorithm. Suppose that there are at most f faulty processes at any point in time, and precisely l processes leave until the end of round $f+1$, and $m=\max\{3, |V|\}$. If $n > l + mf$, then our algorithm achieves strong dynamic consensus.

If $n \leqq l + mf$, no algorithm can achieve strong dynamic consensus.

Liveness (1)

Dynamicity is modeled by a continuous time Markov process

- the arrival of processes: Poisson process with arrival rate λ
- the duration that the process stays active in the system is exponentially distributed with parameter μ

$$\begin{tikzcd}
0 \arrow[r, bend left, "\lambda"] & 1 \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & 2 \arrow[l, bend left, "2\mu"] \arrow[r, bend left, "\lambda"] & \cdots \arrow[l, bend left, "3\mu"] \arrow[r, bend left, "\lambda"] & n \arrow[l, bend left, "n\mu"] \arrow[r, bend left, "\lambda"] & \cdots \arrow[l, bend left, "(n+1)\mu"]
\end{tikzcd}$$

Liveness (2)

Dynamicity for our algorithm is modeled by a stochastic linear death process:

- the algorithm is invoked at time 0 when there are *n* active processes
- processes that become active after time 0 do not participate in an ongoing strong dynamic consensus

$$\begin{tikzcd} 0 & 1 \arrow[l, "\mu"] & 2 \arrow[l, "2\mu"] & \cdots \arrow[l, "3\mu"] & n \arrow[l, "n\mu"] \end{tikzcd}$$

Liveness (3)

Suppose that we start the algorithm with n voting processes and at most f fault processes among them. Let $m = \max\{3, |V|\}$ and $n \geqq mf$. If Δ is an upper bound on the time needed for any round in the consensus algorithm, then

Pr[reach consensus] $\geqq I_x(mf+1, n-mf)$

where $x = \exp(-\mu(f+1)\,\Delta)$, I_x incomplete regularized beta function (a special function that is easy to evaluate).

Conclusions

- We introduced and solved the problem of strong dynamic consensus.
- We gave an algorithm and proved its correctness.
- Under a stochastic model for churn, we were able to give probabilistic guarantees that consensus is successful.

a) One cannot model leaving nodes by faulty nodes, as in the worst case all faulty nodes remain in the system and leaving nodes are correct.

b) Arriving nodes cannot participate in an already on-going consensus, when termination needs to be guaranteed.

References

- “The Byzantine generals problem,” Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. ACM Trans. Programming Languages Systems, 4(3):382-401, 1982.
- “Distributed consensus revisited,” Gil Neiger. Information Processing Letters, 49(4):195-201, 1994.
- “Dynamic Regular Registers in Systems with Churn,” Andreas Klappenecker, Hyunyoung Lee, and Jennifer L. Welch. Theoretical Computer Science, Elsevier, 512:84-97, 2013.
- “Strong Dynamic Consensus in Byzantine Faulty Systems with Churn,” Andreas Klappenecker and Hyunyoung Lee. Proc. of the 19th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2013), pp. 323-330, 2013.

Thank you!

Questions?