Distributed Consensus with Link Failures

Paulo Sérgio Almeida

Distributed Systems Group
Departamento de Informática
Universidade do Minho

The problem

- Each process starts with some value of a type;
- Even though inputs can be arbitrary ...
- ... all processes must output the same value;
- Processes must *agree* on possible outputs for each pattern of inputs through a *validity condition*;
- Easy to solve in a synchronous network with no failures;
- We consider here link failures;

Motivation

- Consensus problems arises in many applications;
- Many examples:
  - agreement on commit or abort a distributed transaction;
  - agreement on estimate using readings from multiple sensors;
  - agreement on considering some process faulty;

Coordinated attack problem

- Several generals plan a coordinated attack;
- Each one may or not be ready to attack;
- Success depends on all attacking together;
- If only some attack, they will be destroyed;
- In that case none should attack;
- They should attack if possible;
- Coordination involves sending messengers;
- Messengers can be lost or captured, and message lost;
- There is an upper bound on time taken by successful messenger to deliver message;
- Communication paths are bidirectional;
- Everyone knows communication paths available;
- How can they coordinate and agree on whether to attack?

Solution in synchronous network with no failures

- If there are no process or link failures can we solve the problem?
- If so, what else do we need to assume?
  - topology?
  - directed / undirected graph?
  - unique identifiers?
  - size of network?
  - ...
- How?

Solution in synchronous network with no failures

- Assumptions:
  - synchronous network with no link or process failures;
  - connected graph of diameter (at most) d;
- Solution:
  - Processes maintain set of choices, starting from $\{yes\}$ or $\{no\}$;
  - In each round:
    - processes send set to all neighbors;
    - processes merge sets in reveiced messages to set maintained;
  - After d rounds:
    - if set equals $\{yes\}$, decide attack;
    - otherwise, decide no attack;

What if messages may be lost?

- Can the algorithm solve the problem if messages may be lost?
- Can we find some other algorithm to solve it in such scenario?
- And in similar scenarios, may be with more knowledge?
- Before trying to prove an impossibility, better to remove ambiguities, state assumptions and formalize problem;
- When proving an impossibility result, may be useful to:
  - use stronger assumptions;
  - use weaker requirements;
- This way, result will be more general;

Coordinated attack problem – deterministic version

- Consider n processes, $1, \ldots, n$ in arbitrary undirected graph;
- Each process knows entire graph, including indices;
- Input state variable in $\{0, 1\}$;
- An arbitrary number of messages may be lost in each round;
- Processes make deterministic choices;
- Goal: all processes set *decision* output variable to either 0 or 1, subject to:
  - agreement: no two processes decide different values;
  - validity:
    1. if all start with 0, decision must be 0;
    2. if all start with 1 and all messages delivered, decision must be 1;
  - termination: all processes eventually decide;

Impossibility of deterministic coordinated attack with link failures

- Let's prove result for two nodes;
- Generalizable to arbitrary network of two or more nodes;

Theorem

Let G be the graph with nodes 1 and 2 connected by an edge. There is no algorithm that solves the coordinated attack problem on G.

Impossibility of deterministic coordinated attack with link failures

Proof.

- By contradiction. Suppose a solution exists; WLOG, assume a single start state for each process, containing input value;
- For each assignment of inputs and message failure pattern, the system has exactly one execution;
- WLOG, assume both processes send messages every round;
- Let e_0 be execution obtained when both processes start with 1 and all messages delivered;
- In e_0 they will eventually decide (termination) both 1 (validity);
- Let's say they decide within r rounds, for some r;

(continues) □

Impossibility of deterministic coordinated attack with link failures

Proof.

(continued)

- Let e_1 be the same as e_0 except messages after round r are lost;
- In e_1 both decide 1 within r rounds as before;
- Let e_2 be the same as e_1 except last message from 1 to 2 is lost;
- From process 1, $e_1 \stackrel{1}{\sim} e_2$; therefore it decides 1 in e_2;
- By termination and agreement, process 2 also decides 1 in e_2;
- Let e_3 be the same as e_2 except last message from 2 to 1 is lost;
- From process 2, $e_2 \stackrel{2}{\sim} e_3$; therefore it decides 1 in e_3;
- By termination and agreement, process 1 also decides 1 in e_3;

(continues) □

Impossibility of deterministic coordinated attack with link failures

Proof.

(continued)

- Repeating, we can obtain e' in which no messages are delivered and, as before, both processes decide 1;
- Consider e'' as e' but in which process 2 starts with 0;
- From process 1, $e' \overset{1}{\sim} e''$; therefore it decides 1 in e'', and so does process 2 by termination and agreement;
- Consider e''' as e'' but in which process 1 also starts with 0;
- From process 2, $e'' \overset{2}{\sim} e'''$; therefore it decides 1 in e''';
- But this contradicts validity as in this case they would have to decide 0;

$\square$

Solving variants of the problem

- Theorem describes fundamental limitation;
- Can some version of the problem be solved?
- Necessary either to:
  - strengthen model or
  - relax requirements;
- Stronger model:
  - probabilistic assumptions about message loss;
  - allow processes to use randomization;
- Weaker requirements:
  - allow some violation of agreement and/or validity;
  - allow violation of termination;

Randomized coordinated attack

- Processes may make random choices;
- Statements are probabilistic;
- Clarify probabilistic statements;
  - under what scenario? average case? worst case?
  - how to describe scenarios?
- An algorithm may run in different scenarios regarding:
  - input values;
  - communication patterns
- A communication pattern represents the set of messages delivered in some execution;
- We may want to consider worst case scenarios;
- Useful to think of a scenario as an *adversary*;

Communication patterns and adversaries

Definition (Communication pattern)

Communication pattern C for r rounds in graph G with edges E:

$$C \subseteq \{(i, j, k) \mid (i, j) \in E \text{ and } 1 \leq k \leq r\}$$

Definition (Adversary)

An adversary is an arbitrary choice of

- an assignment of input values of processes;
- a communication pattern;

- For each adversary A, a sequence of random choices of processes determines an execution;
- For each adversary A, there is a probability distribution on the set of executions;
- Let $P^A[X]$ be the probability of X induced by adversary A;

Randomized coordinated attack formalized

- Consider n processes, $1, \ldots, n$ in arbitrary undirected graph;
- Each process knows entire graph, including indices;
- Processes have one start state, with input variable in $\{0, 1\}$;
- Processes send messages to all neighbors at every round;
- An arbitrary number of messages may be lost in each round;
- Processes may make random choices;
- Goal: all processes set *decision* output variable to either 0 or 1, subject to:
  - agreement: for every adversary A, $P^A[\text{different decisions}] \leq \epsilon$, for some $0 \leq \epsilon \leq 1$;
  - validity:
    1. if all start with 0, decision must be 0;
    2. if all start with 1 and all messages delivered, decision must be 1;
  - termination: all processes decide in a fixed round $r > 0$;

An algorithm for an *n*-node complete graph

- We will present an algorithm for the special case of an *n*-node complete graph;
- Algorithm achieves $\epsilon = 1/r$;
- Algorithm based on knowledge about other's initial values, and what they know directly or indirectly about each other;
- Need definitions to capture such knowledge, namely the *information level*;

A partial order on pairs (process, round)

- For each communication pattern C, we can define a partial order $\sqsubseteq_C$ to compare pairs (i, r) where:
  - i is a process index;
  - r is a round number;
- $\sqsubseteq_C$ is meant to compare what processes know after some round;
- Order induced by information flow resulting from messages:
  1. $(i, k) \sqsubseteq_C (i, k')$ if $k \leq k'$;
  2. $(i, k-1) \sqsubseteq_C (j, k)$ if $(i, j, k) \in C$;
  3. $(i, k) \sqsubseteq_C (i'', k'')$ if $\exists i', k'. (i, k) \sqsubseteq_C (i', k')$ and $(i', k') \sqsubseteq_C (i'', k'')$;

Information level

- To capture relative knowledge between processes we introduce information level:
  - processes start at level 0;
  - when a process knows level l info about all others, it advances to level $l+1$;
- For communication pattern C, we define information level $level_C(i,k)$ of process i at round k recursively:

$$\begin{aligned} level_C(i,0) &= 0 \\ level_C(i,k) &= 1 + \min\{lev(j,i,k) \mid j \neq i\} \\ &\text{where} \\ lev(j,i,k) &= \max(\{-1\} \cup \{level_C(j,k') \mid (j,k') \sqsubseteq_C (i,k)\}) \end{aligned}$$

- As an adversary A implies some communication pattern C, we can also use $level_A(i,k)$ meaning $level_C(i,k)$;

Some lemmas on levels

Lemma

For any communication pattern C, $0 \leq k \leq r$ and any processes i, j: $|level_C(i,k) - level_C(j,k)| \leq 1$.

- Levels of different processes remain within 1 of each other;

Lemma

If $(i,j,k) \in C$ for all $(i,j) \in E$ and $1 \leq k \leq r$, then $level_C(i,k) = k$.

- If no messages are lost, the level is the round number;

An algorithm: RandomAttack – sketch

- Each process keeps knowledge about other's initial values;
- Values known are sent to all in messages;
- Each process keeps level it knows about all processes;
- Information levels are sent to all in messages;
- Process 1 chooses a random key $\{1, \ldots, r\}$ in round 1;
- Random key is sent to all in messages;
- After r rounds:
  - if a process knows that all initial values are 1 and it's level is at least as large as key, it decides 1;
  - otherwise it decides 0;

RandomAttack formally

- Process state, $state_i = (k, d, key, v, l)$ where:
  - $k \in \mathcal{N}$ – rounds, initially 0;
  - $key \in \{\perp, 1, \ldots, r\}$, initiallly $\perp$;
  - $v : \{1, \ldots, n\} \to \{\perp, 0, 1\}$ with pointwise order – values, initially $\{i \mapsto \text{initial value of process } i\} \cup \{j \mapsto \perp \mid j \neq i\}$;
  - $l : \{1, \ldots, n\} \to \{-1, 0, \ldots, r\}$ – levels, initially $\{i \mapsto 0\} \cup \{j \mapsto -1 \mid j \neq i\}$;
  - $d \in \{\perp, 0, 1\}$ – decision, initially $\perp$;
- Random choice function:

$$rand_i((k, key, v, l, d)) = \begin{cases} (k, \mathrm{random}(), v, l, d) & \text{if } i = 1 \wedge k = 0 \\ (k, key, v, l, d) & \text{otherwise.} \end{cases}$$

- Message-generating function:
  $msg_i((k, key, v, l, d), j) = (key, v, l)$

RandomAttack formally – state transition function

- Let M represent the set of messages delivered;
- $trans_i((k, key, v, l, d), M) = (k', key', v', l', d')$ where:

$$
\begin{aligned}
k' &= k + 1 \\
key' &= key \sqcup \bigsqcup\{K \mid (K, V, L) \in M\} \\
v' &= v \sqcup \bigsqcup\{V \mid (K, V, L) \in M\} \\
l' &= \{i \mapsto 1 + \min\{l''(j) \mid j \neq i\}\} \cup \{j \mapsto l''(j) \mid j \neq i\} \\
&\quad \text{where } l'' = l \sqcup \bigsqcup\{L \mid (K, V, L) \in M\} \\
d' &= \begin{cases} 1 & \text{if } k' = r \wedge key' \neq \bot \wedge l'(i) \geq key' \wedge \forall j.\, v'(j) = 1 \\ 0 & \text{otherwise and if } k' = r \\ d & \text{otherwise;} \end{cases}
\end{aligned}
$$

RandomAttack: – correctness

Lemma

RandomAttack calculates information levels correctly: for any execution with communication pattern C, for any $0 \leq k \leq r$, for any process i, after k rounds $l(i)_i = level_C(i, k)$.

Proof.

- At round 0 it holds trivially;
- Propagation of l in messages leads to knowledge respecting $\sqsubseteq_C$;
- $l(j)_i$ at round k encodes $lev(j, i, k)$
- In induction step, strengthen with: $l(j)_i = lev(j, i, k)$;

□

RandomAttack: – correctness

Lemma

For each process i*, if* $l(i)_i > 0$ *then* $key_i = key_1$ *and* $\forall j.\, v(j)_i = v(j)_j$*;*

RandomAttack: – correctness

Theorem

RandomAttack solves the randomized version of coordinated attack, for $\epsilon = 1/r$.

Proof.

- Termination: trivial, at round r;
- Validity:
  - if all processes start with 0, then decision is obviously 0;
  - if all processes start with 1 and all messages are delivered, then, by second lemma of levels and as algorithm calculates levels correctly, at round r: $l(i)_i = r \geq key_1 = key_i$ and $\forall j.\ v(j)_i = v(j)_j = 1$; therefore the decision is 1.

(continues) □

RandomAttack: – correctness

Proof.

(continued)

Agreement:

- let A be any adversary; want to show that

$$P^A[\text{some process decides 0 and another decides 1}] \leq \epsilon$$

- by first lemma of levels, after round r for any processes i, j, the level $l(i)_i$ will be within 1 of $l(j)_j$;
- if $key_1 > \max_i\{l(i)_i\}$ or some process starts with 0, then all processes decide 0;
- if $key_1 \leq \min_i\{l(i)_i\}$ and all processes start with 1, then all processes decide 1;
- process can only disagree if $key_1 = \max_i\{l(i)_i\}$; this has probability $1/r$ as key is uniformly distributed in $\{1, \ldots, \mathrm{r}\}$ and $\max_i\{l(i)_i\}$ is determined by A;

$\square$

Lower bound on disagreement

- Can we obtain algorithms with smaller disagreement probability?
- It can be proven that, for n-node complete graphs:

Theorem

Any r-round algorithm for randomized coordinated attack has probability of disagreement at least $1/(r+1)$*;*

A critique of the assumptions in failure model

- Having shown that arbitrary link failures make original problem unsolvable, only probabilistic claims can be made;
- The approach taken:
  - uses randomization in algorithm;
  - assumes it will have to work for any adversary;
  - makes probabilistic claims, assuming worst case adversary;
- This results in a large error probability, as an algorithm must work in any scenario , even if not a single message is ever delivered;
- Approach not much useful for devising algorithms and making probabilistic claims for realistic scenarios:
  - e.g. assuming some probability distribution of message loss;
- This ‘worst case’ approach is not useful either for realistic situations assuming malicious adversaries:
  - e.g. man-in-the-middle that can read messages in transit and remove messages;

Other approaches to the failure model

- The approach taken:
  - used excessively worse case for assuming ‘natural’ message loss;
  - was not worse enough for malicious adversaries;
- Can different approaches be useful?
- Probabilistic model of loss:
  - probabilistic model of link loss;
  - deterministic algorithm;
  - probabilistic claims for algorithm;
- Coverage model of loss:
  - assume predicates about possible message loss, that cover some percentage of all cases;
  - e.g. not more than *f* consecutive failures in a link 99% of the cases;
  - use deterministic algorithm that works assuming predicates;
  - claims are non probabilistic, for the given coverage;

Coordinated attack under independent link losses

- Assumption:
  - message losses are independent;
  - the probability of a single message loss is p_l;
- Back to original problem: generals should try to attack if all ready, and do it at the same round;
- Goal: all processes set *decision* output variable to either 0 or 1, subject to:
  - agreement: no two processes decide different values;
  - validity:
    1. if some starts with 0, decision must be 0;
    2. if all start with 1, decision must be 1;
  - termination: all processes eventually decide at the same round, with probability $1 - \epsilon$;
- For a given probability of loss p_l, can we obtain an algorithm that satisfies some arbitrarily low ϵ? How many rounds will we need?