Distributed Consensus with Process Failures

Paulo Sérgio Almeida

Distributed Systems Group
Departamento de Informática
Universidade do Minho

2007/2008

Distributed consensus with process failures

- Here we still consider consensus in a synchronous system;
- Instead of link failures, here we consider process failures;
- Two failure models: *stopping failures* and *Byzantine failures*;
- Stopping failure model:
  - processes may stop without warning;
  - useful to model crashes;
- Byzantine failure model:
  - faulty processes may exibit completely unconstrained behavior;
  - useful to model arbitrary processor malfunction (e.g. cosmic rays that change bits of memory);
  - term introduced by Lamport in *The Byzantine Generals Problem*;

The agreement problem with process failures

- Consider n processes, $1, \ldots, n$ in arbitrary undirected graph;
- Each process knows entire graph, including indices;
- One start state for each process with input variable in a set V;
- Processes make deterministic choices;
- At most f processes may fail;
- Goal: all processes decide value in V, subject to ...

The agreement problem with process failures

- Stopping agreement:
  - agreement: no two processes decide different values;
  - validity: if all processes start with the same $v \in V$, then the decision must be v;
  - termination: all nonfaulty processes eventually decide;
- Byzantine agreement:
  - agreement: no two nonfaulty processes decide different values;
  - validity: if all nonfaulty processes start with the same $v \in V$, then the decision of a nonfaulty proces must be v;
  - termination: all nonfaulty processes eventually decide;

Relationship between stopping and Byzantine agreement

- Does an algorithm for Byzantine agreement also solves stopping agreement?
- No!
- In the stopping case, processes must decide the same value, even some faulty one that fails after deciding;
- In the Byzantine case, we allow faulty processes to decide some arbitrary value;

Alternative stronger validity condition

- An alternative validaty condition can be (for stopping failures):
  - validity: a decision must be the initial value of some process;
- This condition is stronger as it implies the previous one;
- The use of the previous one:
  - strengthens impossibility results, but
  - weakens claims about algorithms;

Algorithms for stopping failures

- We consider complete n-node graphs;
- Will present some algorithms:
  - Basic algorithm: processes repeatedly broadcast set of known values;
  - Improvements on basic algorithm;
  - Algorithms with an *exponential information gathering* strategy;
- Some conventions:
  - v_0 is some prespecified default value in V;
  - b is an upper bound on bits needed to represent a value in V;

Basic algorithm – FloodSet, informally

- Each process maintains a set $W \subseteq V$;
- Initially W contains initial value;
- In each round processes broadcast W and merges received sets to W;
- In round $f + 1$, if $W = \{v\}$, decide v, else decide v_0;

Basic algorithm – FloodSet, formally

- Process state, $state_i = (r, W, d)$ where:
  - $r \in \mathcal{N}$ – rounds, initially 0;
  - $W \subseteq V$, initially i's initial value;
  - $d \in V \cup \{unknown\}$ – decision;
- Message-generating function: $msg_i((r, W, d), j) = W$;
- Let M represent the set of messages delivered;
- State transition function: $trans_i(r, W, d), M) = (r', W', d')$ where:

$$
\begin{aligned}
r' &= r + 1 \\
W' &= W \cup \bigcup M \\
d' &= \begin{cases} v & \text{if } r' = f + 1 \wedge \exists v.\, W' = \{v\} \\ v_0 & \text{otherwise and if } r' = f + 1 \\ d & \text{otherwise} \end{cases}
\end{aligned}
$$

Some notation

- Let $W_i(r)$ be variable W of process i after r rounds;
- A process is *active after r rounds* if it has not failed until the end of round r;
- Let $A(r)$ denote the set of processes *active after r rounds* for a given failure pattern; any A satisfies:
  - $A(0) = \{1, \ldots, n\}$;
  - if $r' \geq r$, then $A(r') \subseteq A(r)$;
  - $A(r) = A(r-1)$ if no process has failed during round r;

Some lemmas

Lemma

If no process fails in some round r, $W_i(r) = W_j(r)$ for all $i, j \in A(r)$.

Lemma

If $W_i(r) = W_j(r)$ for all $i, j \in A(r)$ and $r' \geq r$, then $W_i(r') = W_j(r')$ for all $i, j \in A(r')$.

Some lemmas

Lemma

If $i, j \in A(f+1)$, then $W_i(f+1) = W_j(f+1)$.

Proof.

Since at most f processes are faulty, there must be some round $r \leq f+1$ at which no process fails. Combine two previous lemmas. □

FloodSet correctness

Theorem

FloodSet solves agreement for stopping failures.

Proof.

- Termination: at round $f+1$ all nonfaulty processes decide;
- Agreement: suppose any $i, j \in A(f+1)$ that decide; from previous lemma, $W_i(f+1) = W_j(f+1)$ and they must decide the same value;
- Validity: if all processes start with v, then $W_i(0) = \{v\}$, for all processes, only $\{v\}$ travels in messages, and $W_i(r) \subseteq \{v\}$ for any process i and round r; therefore $W_i(f+1) = \{v\}$ and the decision must be v;

□

FloodSet complexity analysis

- Rounds: $f+1$ until nonfaulty processes decide;
- Total number of messages: $O((f+1)n^2)$;
- Each messages contains set with at most n elements: bits per message $O(nb)$;
- Bits of communication: $O((f+1)n^3b)$;

Alternative decision rules

- The essence of FloodSet is that all nonfaulty processes have the same W after $f + 1$ rounds;
- The decision rule does not matter much as long as it is a function of W that decides on the element in case of a singleton;
- Deciding a default v_0 looks artificial;
- We can make the algorithm guarantee the stronger validity condition and decide on the initial value of some process by assuming a total order on V and deciding $\min(W)$;

OptFloodSet – an algorithm with less communication

- Improvement on FloodSet;
- Insight: a process only needs to know
  - the value of W when it has one element, or
  - that W has more than one element;
- Algorithm broadcasts at most two values:
  - at round 1 broadcasts initial value;
  - after the first round when it has received some new value, it broadcasts one of the new values received;
- Decision is either v when $W = \{v\}$ or v_0;

OptFloodSet complexity analysis

- Rounds: $f + 1$ until nonfaulty processes decide;
- Total number of messages: at most $2n^2$;
- Bits per message at most b;
- Bits of communication: at most $2n^2 b$;

OptFloodSet correctness

- Could prove from scratch as before;
- Instead, will use *simulation*: prove a formal relationship between both algorithms;
- Must obtain *simulation relation*: an invariant that relates the states of both algorithms after any number of rounds when starting with same inputs and subject to same failure pattern;
- Let's use $OW_i(r)$ for W_i after r rounds in OptFloodSet and $W_i(r)$ for FloodSet as before;
- Let's use $i \xrightarrow{r} j$ to denote process i sending a message in round r to a process j active after round r;

OptFloodSet correctness

Lemma (OFS1)

In FloodSet, if $i \xrightarrow{r+1} j$, *then* $W_i(r) \subseteq W_j(r+1)$.

Lemma (OFS2)

In OptFloodSet, if $i \xrightarrow{r+1} j$ *is possible in failure pattern, then:*

- *if* $|OW_i(r)| = 1$, *then* $OW_i(r) \subseteq OW_j(r+1)$;
- *if* $|OW_i(r)| > 1$, *then* $\left|OW_j(r+1)\right| > 1$;

Lemma (OFS3)

After any round r*:*

- $OW_i(r) \subseteq W_i(r)$;
- *if* $|W_i(r)| = 1$, *then* $OW_i(r) = W_i(r)$;

OptFloodSet correctness

Lemma (OFS4)

After any round r*, if* $|W_i(r)| > 1$*, then* $|OW_i(r)| > 1$*.*

Proof.

By induction; base case vacuous; assume lemma holds for r; assume $|W_i(r+1)| > 1$; we have two cases:

- $|W_i(r)| > 1$: by I.H. $|OW_i(r)| > 1$, which implies $|OW_i(r+1)| > 1$;
- $|W_i(r)| = 1$: by lemma OFS3, $OW_i(r) = W_i(r)$; two cases:
  - $\forall j \mid j \xrightarrow{r+1} i \text{ in FloodSet. } |W_j(r)| = 1$: for all such j, lemma OFS3 implies $OW_j(r) = W_j(r)$, lemma OFS2 implies $OW_j(r) \subseteq OW_i(r+1)$; therefore $OW_i(r+1) = W_i(r+1)$;
  - $\exists j \mid j \xrightarrow{r+1} i \text{ in FloodSet. } |W_j(r)| > 1$: by I.H. $|OW_j(r)| > 1$ and lemma OFS2 implies $|OW_i(r+1)| > 1$;

□

OptFloodSet correctness

Lemma

After any round k, state variables r and d have the same values in FloodSet and OptFloodSet.

Proof.

Trivial for r. Variable d only changes at round $f + 1$; it follows from applying lemmas OFS3 and OFS4 at round $f + 1$. □

Theorem

OptFloodSet solves agreement for stopping failures.

Proof.

By previous lemma and correctness of FloodSet. □

Sketch of another algorithm

- Based on alternative version of FloodSet with stronger validity;
- Assumes total order on V, decides on minimum of W;
- Algorithm stores and relays just the minimum known so far;
- Uses $O((f+1)n^2b)$ bits of communication;
- Can be proven correct by a simulation relating it to the FloodSet version with the alternative decision.

Exponential information gathering algorithms

- Send and relay intitial values in several rounds;
- Record values received along various communication paths in a EIG tree;
- Use a decision rule based on values in their trees;
- Are overly costly for stopping failures;
- EIG trees useful for solving Byzantine agreement;
- Presented for stopping failures to introduce EIG trees;
- Algorithms can be adapted to *authenticated Byzantine failure model*.

EIG trees

- An EIG tree $T_{n,f}$ has $f+2$ levels from 0 to $f+1$;
- Nodes at level $0 \leq k \leq f$ have $n-k$ children;
- Nodes at level k are labelled by a string of k distinct indices;
- The root is labelled by the null string ϵ;
- Children of node $i_1 \dots i_k$ have label $i_1 \dots i_k j$ with $j \in \{1, \dots, n\} \setminus \{i_1, \dots, i_k\}$;
- We can represent EIG trees by mappings from labels to values;
- It is convenient to store only mappings to non-null values; a label not in the mapping means the corresponding node contains *null*;
- For an EIG tree T, let $T^{|k}$ denote T restricted to level k:

$$T^{|k} = \{(l, v) \in T \mid |l| = k\}$$

- Labels are partially ordered using prefix order:

$$r \sqsubseteq s \iff r \text{ is a prefix of } s$$

Algorithm EIGStop – sketch

- Each process maintains own EIG tree;
- Root is decorated with input value;
- At round k, processes:
  - broadcast values at level $k-1$ to all, including itself;
  - decorate level k according to messages received;
- Paths from the root represent chains of distinct processes along which values are propagated;
- $T_i(i_1 \ldots i_k) = v \in V$ means that i knows input value of i_1 to be v due to chain of communication $i_1 \xrightarrow{1} i_2 \xrightarrow{2} \ldots \xrightarrow{k-1} i_k \xrightarrow{k} i$;
- Otherwise, the chain of communication $i_1 \xrightarrow{1} i_2 \xrightarrow{2} \ldots \xrightarrow{k-1} i_k \xrightarrow{k} i$ was broken by a failure;
- At round $f+1$, processes decide as a function of the tree;

EIGStop formally

- Process state, $state_i = (r, T, d)$ where:
  - $r \in \mathcal{N}$ – rounds, initially 0;
  - T – EIG tree, initially $\{\epsilon \mapsto i\text{'s initial value}\}$;
  - $d \in V \cup \{unknown\}$ – decision;
- Message-generating function:

$$msg_i((r, T, d), j) = \{(l, v) \in T^{|r} \mid i \notin l\}$$

- Let $M = \{(j, M_j)\}$ be messages delivered, including when $j = i$;
- We wil use the range of a mapping: $\mathrm{ran}(T) = \{v \mid (l, v) \in T\}$;
- State transition function: $trans_i(r, T, d), M) = (r', T', d')$ where:

$$\begin{aligned}
r' &= r + 1 \\
T' &= T\left[[lj \mapsto v \mid (l, v) \in M_j] \mid (j, M_j) \in M\right] \\
d' &= \begin{cases} v & \text{if } r' = f + 1 \wedge \exists v.\, \mathrm{ran}(T') = \{v\} \\ v_0 & \text{otherwise and if } r' = f + 1 \\ d & \text{otherwise} \end{cases}
\end{aligned}$$

EIGStop correctness

Lemma

After $f + 1$ *rounds:*

- $T_i(\epsilon)$ *is i's input value;*
- *if* $T_i(xj) = v$, *then* $T_j(x) = v$;
- *if* $(xj, v) \notin T_i$, *then* $(x, v') \notin T_j$ *or* $j \overset{|x|+1}{\not\longrightarrow} i$;

Lemma

After $f + 1$ *rounds:*

- *if* $T_i(y) = v$ *and* $xj \sqsubseteq y$, *then* $T_j(x) = v$;
- *if* $v \in \mathrm{ran}(T_i)$, *then* $\exists j.\ T_j(\epsilon) = v$;
- *if* $v \in \mathrm{ran}(T_i)$, *then* $\exists s.\ i \notin s \wedge T_i(s) = v$;

EIGStop correctness

Lemma

If $i, j \in A(f+1)$*, then* $\mathrm{ran}(T_i) = \mathrm{ran}(T_j)$*.*

Proof.

It is enough to show that if $i \neq j$, then $\mathrm{ran}(T_i) \subseteq \mathrm{ran}(T_j)$; suppose $v \in \mathrm{ran}(T_i)$; by previous lemma $\exists s.\, i \notin s \wedge T_i(s) = v$; two cases:

- $|s| \leq f$: then $|si| \leq f + 1$; since $i \notin s$, then $i \xrightarrow{|si|} j$ containing (s, v); therefore $T_j(si) = v$;
- $|s| = f + 1$: then there must be a nonfaulty process $p \in s$; consider prefix $rp \sqsubseteq s$; by previous lemma, $T_p(r) = v$; then $p \xrightarrow{|rp|} j$ containing (r, v); therefore $T_j(rp) = v$;

□

EIGStop correctness

Theorem

EIGStop solves agreement for stopping failures.

Proof.

- termination: obvious;
- validity: if all initial values are v, then $\mathrm{ran}(T) \subseteq \{v\}$; as T contains initial value, $\mathrm{ran}(T) \supseteq \{v\}$; therefore $\mathrm{ran}(T) = \{v\}$ and decision must be v;
- agreement: from previous lemma, the decision by nonfaulty processes, at round $f + 1$, must be the same for all;

□

EIGStop complexity analysis

- Number of rounds: f+1;
- Number of messages: $O((f+1)n^2)$;
- Bits of communication exponential on failures: $O(n^{f+1}b)$;

EIGByz – an EIG Algorithm for Byzantine agreement

- Assumption: $n > 3f$;
- Similar to EIGStop, with some modifications;
- If a process receives malformed messages, it discards them;
- After $f + 1$ rounds, each process modifies tree to have v_0 in unassigned (null) nodes;
- The decision is obtained by the value at the root of a new tree constructed bottom-up;
- The leaves have the corresponding values in the original tree;
- The value at a node is:
  - the value in a strict majority of children, it such value exists;
  - v_0 otherwise;