With Finite Memory Consensus is Easier Than Reliable Broadcast

Carole Delporte-Gallet[1] Stéphane Devismes[2]
Hugues Fauconnier[1] Franck Petit[3] Sam Toueg[4]

[1] LIAFA, Paris VII
[2] VERIMAG, Grenoble I
[3] INRIA/LIP, University of Lyon/ENS Lyon
[4] Dept. of Computer Science, University of Toronto

OPODIS 2008, Luxor - Egypt

15 december 2008

Roadmap

Roadmap

Topic

Impact of the *finite process memory* on the solutions of three classical problems in *asynchronous systems with failures*:

- *Consensus*,
- *Repeated Consensus*, and
- *Reliable Broadcast*.

A failure detector approach

[FLP 85]

The consensus cannot be solved in asynchronous systems with failures.

Additional mechanism: *Failure Detectors* **[Chandra-Toueg, 96]**

A failure detector is a distributed oracle that gives (possibly incorrect) hints about the process crashes.

Results

In *asynchronous systems with failures and finite process memories*:

	Necessary	Sufficient
Reliable Broadcast	$\mathcal{P}^-$	$\mathcal{P}^-$
Consensus		$\mathcal{S}$
Repeated Consensus	$\mathcal{P}^-$	$\mathcal{P}^-$

Results

In *asynchronous systems with failures and finite process memories*:

	Necessary	Sufficient
Reliable Broadcast	$\mathcal{P}^-$	$\mathcal{P}^-$
Consensus		$\mathcal{S}$
Repeated Consensus	$\mathcal{P}^-$	$\mathcal{P}^-$

Remark I

In *asynchronous systems with failures and finite process memories*:

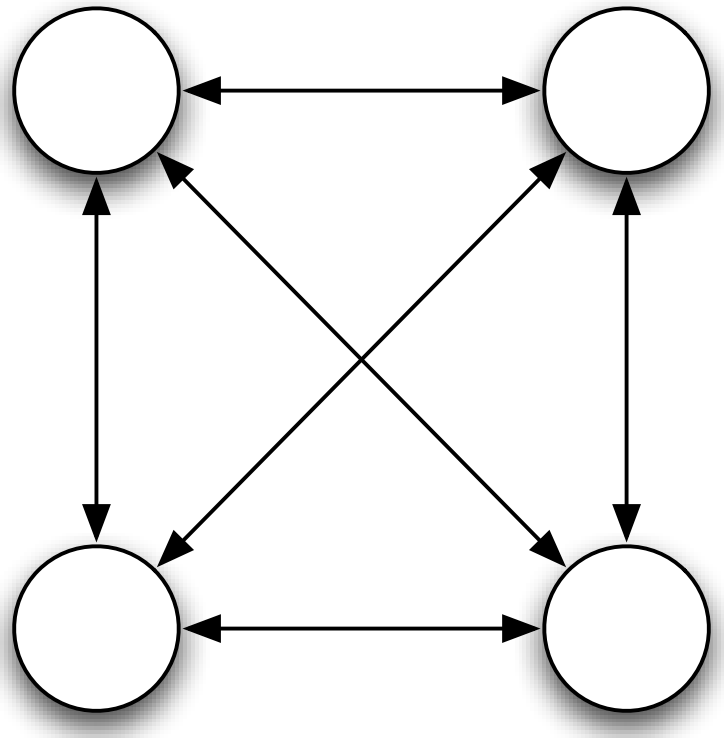
- *consensus* is easier to solve than *reliable broadcast*, and
- *reliable broadcast* is as difficult to solve as *repeated consensus*.

Remark II

When processes have infinite memory, *reliable broadcast* is easier to solve than *consensus*.

Roadmap

Distributed Systems: $\Phi^{\mathcal{F}}$



- **Topology**: Fully Connected
- **Communication**: By messages
- **Process**: Asynchronous, can crash, and with finite memory
- **Link**: Fair lossy and FIFO

Failure Detectors

A *failure detector* is a local module that outputs a set of processes that are currently suspected of having crashed.

A failure detector can be defined in terms of two *abstract properties*: *Completeness* and *Accuracy*.

Failure Detector Classes

Perfect ($\mathcal{P}$). A failure detector is said to be *perfect* if it satisfies:

- **[Strong Completeness]** Eventually every process that crashes is permanently suspected by *every correct process*.
- **[Strong Accuracy]** No process is suspected before it crashes.

Almost Perfect ($\mathcal{P}^-$). A failure detector is said to be *almost perfect* if it satisfies:

- **[Strong Completeness]** Eventually every process that crashes is permanently suspected by *every correct process*.
- **[Almost Strong Accuracy]** No correct process is suspected.

Strong ($\mathcal{S}$). A failure detector is said to be *strong* if it satisfies:

- **[Strong Completeness]** Eventually every process that crashes is permanently suspected by *every correct process*.
- **[Weak Accuracy]** There is a correct process that is never suspected.

Reduction

- A failure detector $\mathcal{D}$ is *weaker* than another failure detector $\mathcal{D}'$ if there is an algorithm that uses only $\mathcal{D}'$ to emulate the output of $\mathcal{D}$ for every failure pattern.
- If $\mathcal{D}$ is weaker than $\mathcal{D}'$ but $\mathcal{D}'$ is not weaker than $\mathcal{D}$, then $\mathcal{D}$ is *strictly weaker* than $\mathcal{D}'$.
- The *weakest* failure detector $\mathcal{D}$ to solve a given problem is a failure detector $\mathcal{D}$ that is necessary and sufficient to solve the problem.

Remark

$$\mathcal{S} < \mathcal{P}^{-} < \mathcal{P}$$

Roadmap

Definition

The reliable broadcast is defined with two primitives: BROADCAST(m) and DELIVER(m).

Any reliable broadcast algorithm satisfies the following requirements:

- **Validity:** If a correct process invokes BROADCAST(m), then it eventually executes DELIVER(m).
- **(Uniform) Agreement:** If a process executes DELIVER(m), then all other correct processes eventually execute DELIVER(m).
- **Integrity:** For every message m, every process executes DELIVER(m) at most once, and only if *sender*(m) previously invokes BROADCAST(m).

Necessary condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$ (1/2)

Necessary condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$ (1/2)

Lemma

Let $\mathcal{A}$ be an algorithm solving `reliable-broadcast` *in $\Phi^{\mathcal{F}}$ with a failure detector $\mathcal{D}$. There exists an integer k such that for every process p and every correct process q, for every run R of $\mathcal{A}$ where p* `BROADCAST`*s and* `DELIVER`*s k messages, at least one message from q has been received by some process.*

Necessary condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$ (2/2)

Let $\mathcal{A}$ be a *reliable broadcast* algorithm in $\Phi^{\mathcal{F}}$ using $\mathcal{D}$.

```
1:      /* CODE FOR PROCESS p */
2: begin
3:     Output_q ← ∅
4:     for i = 1 to k do
5:         BROADCAST(m)      /* using A with D */
6:         wait for DELIVER(m)
7:     end for
8:     Output_q ← {q}
9: end
10:     /* CODE FOR PROCESS q */
11: begin
12: end
13:     /* CODE FOR EVERY PROCESS Π − {p, q} */
14: begin
15:     execute the code of A with D for these messages
16: end
```

Figure: $\mathcal{A}_{(p,q)}$

Necessary condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$ (2/2)

Let $\mathcal{A}$ be a *reliable broadcast* algorithm in $\Phi^{\mathcal{F}}$ using $\mathcal{D}$.

```
1:      /* CODE FOR PROCESS p */
2: begin
3:     Output_q ← ∅
4:     for i = 1 to k do
5:         BROADCAST(m)      /* using A with D */
6:         wait for DELIVER(m)
7:     end for
8:     Output_q ← {q}
9: end
10:     /* CODE FOR PROCESS q */
11: begin
12: end
13:     /* CODE FOR EVERY PROCESS Π − {p, q} */
14: begin
15:     execute the code of A with D for these messages
16: end
```

Figure: $\mathcal{A}_{(p,q)}$

Theorem

$\mathcal{P}^-$ is necessary to solve Reliable Broadcast in $\Phi^{\mathcal{F}}$.

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$

We give an algorithm implementing the reliable broadcast in $\Phi^{\mathcal{F}}$ using $\mathcal{P}^-$.

To deal with crash failures:

- p executes BROADCAST(m):
  - p sends $\langle p\text{-BRD}, m\rangle$ to every non-suspected process
  - p executes DELIVER(m) only after receiving p-ACK messages from every non-suspected process
- Code for the other processes q:
  - Upon receiving $\langle p\text{-BRD}, m\rangle$ from x:
    - q broadcasts m to every non-suspected process except x
    - q sends p-ACK to x if $x \neq p$
  - q sends p-ACK to p when q has received p-ACK from every non-suspected process

To deal with message loss: Alternating-bit protocol.

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$

We give an algorithm implementing the reliable broadcast in $\Phi^{\mathcal{F}}$ using $\mathcal{P}^-$.

To deal with crash failures:

- p executes BROADCAST(m):
  - p sends $\langle p\text{-BRD}, m\rangle$ to every non-suspected process
  - p executes DELIVER(m) only after receiving p-ACK messages from every non-suspected process
- Code for the other processes q:
  - Upon receiving $\langle p\text{-BRD}, m\rangle$ from x:
    - q broadcasts m to every non-suspected process except x
    - q sends p-ACK to x if $x \neq p$
  - q sends p-ACK to p when q has received p-ACK from every non-suspected process

To deal with message loss: Alternating-bit protocol.

Theorem

$\mathcal{P}^-$ is sufficient to solve Reliable Broadcast in $\Phi^{\mathcal{F}}$.

Roadmap

Definition

We define the consensus problem in terms of two primitives, PROPOSE(v) and DECIDE(u).

Any consensus algorithm satisfies the following requirements:

- **(Uniform) Agreement:** No two processes *decide* differently.
- **Termination:** Every correct process eventually *decides* some value.
- **Validity:** If a process *decides* v, then v was *proposed* by some process.

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{S}$ (1/3)

Custumized version of a **[Chandra and Toueg, 96]** algorithm

Every process p stores in $V_p[q]$ the value proposed by q ($V_p[q]$ is initialized to $\perp$)

Two phases:

- A "UNION" phase
- An "INTERSECTION" phase

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{S}$ (2/3)

A "UNION" phase: $n-1$ asynchronous rounds.

- During a round, every process p sends the array V to every other process
- Upon receiving V_q from q, p updates V_p
- The round ends, when p receives V_q for every non-suspected process q
- A the end of the $n-1$ rounds, for every correct process p, V_p contains at least all non-trivial value of V_x where x is any process that is never suspected

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{S}$ (3/3)

An "INTERSECTION" phase: 1 asynchronous round

- During this round, every process p sends V to every other process
- The round ends, when p receives V_q for every non-suspected process q
- $V_p[q] \leftarrow \perp$ if there exists a non-suspected process r such that $V_r[q] = \perp$
- A the end of the round, for every two correct processes p and q, $V_p = V_q$

Decision: the first non-trivial value
(there is at least one non-trivial value because at least one process is never suspected)

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{S}$ (3/3)

An "INTERSECTION" phase: 1 asynchronous round

- During this round, every process p sends V to every other process
- The round ends, when p receives V_q for every non-suspected process q
- $V_p[q] \leftarrow \perp$ if there exists a non-suspected process r such that $V_r[q] = \perp$
- A the end of the round, for every two correct processes p and q, $V_p = V_q$

Decision: the first non-trivial value
(there is at least one non-trivial value because at least one process is never suspected)

Theorem

$\mathcal{S}$ is sufficient to solve Consensus in $\Phi^{\mathcal{F}}$.

Roadmap

Definition

We define the repeated consensus in terms of two primitives, R-PROPOSE(v) and R-DECIDE(u).

Any repeated consensus algorithm satisfies the following requirements:

- **Agreement:** If u and v are the outputs of two processes, then u is a prefix of v or v is a prefix of u.
- **Termination:** Every correct process has an infinite output.
- **Validity:** If the i^{th} value of the output of a process is v, then v is the i^{th} value of the input of some process.

Necessary condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$

Theorem

$\mathcal{P}^-$ is necessary to solve Repeated Consensus problem in $\Phi^{\mathcal{F}}$.

Proof.

Proof similar to the proof of the necessary condition for the reliable broadcast. □

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$

- Use of a simplified version of the previous consensus algorithm
- Synchronization barrier (two asynchronous rounds) between each two consensus:
  - **First Round:**
    - Each process sends *decide*
    - A process terminated its first round after receiving a *decide* or a *start* message from every non-suspected process
  - **Second Round:**
    - Each process sends *start*
    - A process terminated its first round after receiving a *start* message or a *round* message (next consensus) from every non-suspected process

Sufficient condition in $\Phi^{\mathcal{F}}$: $\mathcal{P}^-$

- Use of a simplified version of the previous consensus algorithm
- Synchronization barrier (two asynchronous rounds) between each two consensus:
  - **First Round:**
    - Each process sends *decide*
    - A process terminated its first round after receiving a *decide* or a *start* message from every non-suspected process
  - **Second Round:**
    - Each process sends *start*
    - A process terminated its first round after receiving a *start* message or a *round* message (next consensus) from every non-suspected process

Theorem

$\mathcal{P}^-$ is sufficient to solve Repeated Consensus problem in $\Phi^{\mathcal{F}}$.

Roadmap

Sum up

In $\Phi^{\mathcal{F}}$:

	Necessary	Sufficient
Reliable Broadcast	$\mathcal{P}^-$	$\mathcal{P}^-$
Consensus		$\mathcal{S}$
Repeated Consensus	$\mathcal{P}^-$	$\mathcal{P}^-$

Remark I

In $\Phi^{\mathcal{F}}$:

- *consensus* is easier to solve than *reliable broadcast*, and
- *reliable broadcast* is as difficult to solve as *repeated consensus*.

Remark II

When processes have infinite memory, *reliable broadcast* is easier to solve than *consensus*.

Perspectives

In $\Phi^{\mathcal{F}}$:

- What is the weakest failure detector for solving the consensus?
- What happens if we now consider non-uniform specifications
- What is the weakest failure detector for implementing registers?

Thank you for your attention!