Distributed Agreement and its Relation with Error-Correcting Codes

Michel RAYNAL[⋆]

joint work with:

R. FRIEDMAN[‡] A. MOSTEFAOUI[⋆] S. RAJSBAUM[†]

[‡]Technion, Haifa, Israel

[⋆]IRISA, Campus Universitaire de Beaulieu, Rennes, France

[†]Instituto de Matematicas, UNAM, Mexico City, Mexico

Summary

- Computation Model
- Interactive Consistency
- IC and Failure Detectors
- The Condition-Based Approach
- Characterization of the Conditions
- A Protocol
- From Error-Correcting Codes to IC
- From IC to Error-Correcting Codes
- The Consensus Problem
- Conclusion

Main references

- A. Mostefaoui, S. Rajsbaum, M. Raynal. Conditions on Input Patterns for Consensus Solvability in Asynchronous Distributed Systems. *33rd ACM Symposium on Theory of Computing (STOC'01)*, pp. 153-162, July 2001, Crête (Greece). Extended version to appear in *Journal of the ACM*.
- A. Mostefaoui, S. Rajsbaum, M. Raynal, M. Roy. Condition-Based Consensus Solvability: a Hierarchy of Conditions and Efficient Protocols. *20th ACM Int. Symposium on Principles of Distributed Computing (PODC'01)*, pp. 151-160, 2001. Extended version to appear in *Distributed Computing*.
- R. Friedman, A. Mostefaoui, S. Rajsbaum, M. Raynal. Asynchronous Distributed Agreement and its Relation with Error Correcting Codes. *Proc. 16th Int. Symp. on Distributed Computing (DISC'02)*, Springer Verlag LNCS 2508, pp. 63-87, 2002.

A Mathematician View

"*Beauty is the first test: there is no permanent place in the world for ugly ideas*".

G. H. Hardy (1969)

Computation Model

- A set of n processes $p_1, \ldots, p_n$
- A reliable communication system (SM/MP)
- No bound on communication, no bound on the time required by a process to execute a step
- Failure: At most f_c processes may crash
  Correct process = a process that dont crash
  Faulty process = a process that crashes

IC and ECC

Interactive Consistency and

Error Correcting Codes

The Interactive Consistency Problem (1)

- A fundamental problem (Paradigm)
- Defined by Pease, Shostack and Lamport in 1980 in the context of synchronous systems with processes prone to Byzantine failures
- Informally:
  - ⋆ Each process p_i proposes a value v_i
  - ⋆ Each p_i has to decide the vector of proposed values
- Here we address it in asynchronous systems with process crash failures

The Interactive Consistency Problem (2)

- Agreement:
  No two different vectors are decided
- Termination:
  A process that does not crash decides
- Validity: Any decided vector D is such that
  - ⋆ $D[i] \in \{v_i, \perp\}$, and
  - ⋆ $D[i] = v_i$, if p_i does not crash

IC in Asynchronous Systems (1)

- As hard as the consensus problem
- FLP 85 impossibility result

IC in Asynchronous Systems (2)

- Perfect failure detector (Chandra-Toueg JACM 96): each process p_i is provided with a set $suspected_i$ such that:
  - ⋆ Strong Completeness: Eventually each process that crashes appears in $suspected_i$
  - ⋆ Perpetual Strong Accuracy: No process appears in $suspected_i$ before crashing
- Problem reduction: Build a perfect FD $\simeq$ IC

  (Helary-Hurfin-Mostefaoui-Raynal-Tronel "Computing Global Functions ..." IEEE TPDS 11(9):897-909, 2000)

IC in Asynchronous Systems (3)

$suspected_i \leftarrow \emptyset$; $x \leftarrow 0$;
cobegin
task $T1$: **while** $true$ **do**
    $x \leftarrow x + 1$; % IC instance nb %
    $D_i \leftarrow$ IC_Protocol(x, v_i); % $v_i \neq \perp$ %
    $suspected_i \leftarrow \{j \mid D_i[j] = \perp\}$
  enddo

task $T2$: **when** p_i **issues** *QUERY*: *return*$(suspected_i)$
coend

Weakening the Computation Model

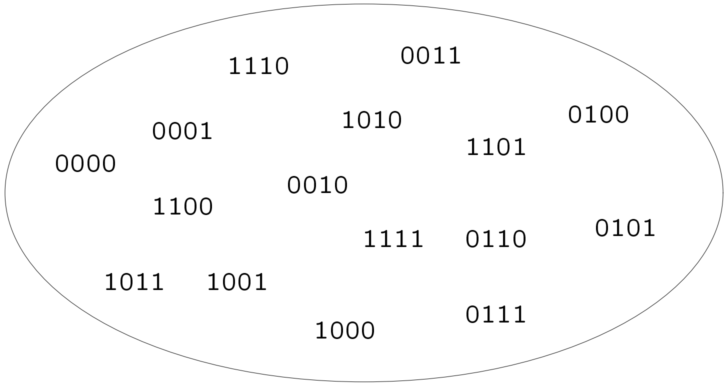
- "Value domain" faults: a process proposes value a while it was supposed to propose value b
- At most f_e processes can be value-faulty
- Assumption: $f_c + f_e < n$
- The notion of "correct/faulty" with respect to crash is related to an execution, as it is not known in advance if a process will crash
- Similarly, the notion of "correct/faulty" with respect to a proposed value is also related to an execution. If $D[i] = v_i$ (where D is the decided vector and v_i is the value proposed by p_i) then p_i is value-correct, otherwise it is value-faulty

The Condition-Based Approach (1)

The processes are assumed to propose some vector I, and have to decide on it

Idea: Identify sets of vectors (called Conditions) that allow processes to recover the actual vector despite up to f_c process crashes and f_e erroneous proposals

The Condition-Based Approach (2)

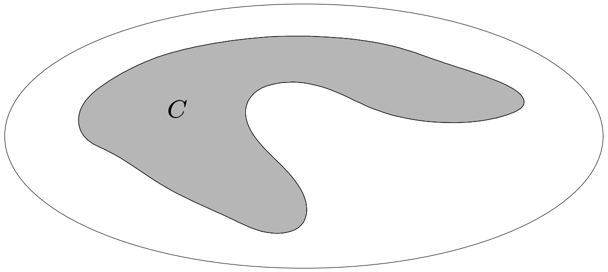


$\mathcal{V} = \{0, 1\}$ and $n = 4$

$\mathcal{V}$ in the set of values that can be proposed

$\mathcal{V}^n$ in the set of all possible input vectors

The Condition-Based Approach (3)



The Condition-Based Approach

We are interested in:

- Design conditions C
- Design protocols that:
  - ⋆ Solve IC when $I \in C$ and at most f_c processes crash and f_e erroneous proposals
  - ⋆ Are always safe (agreement and validity)
  - ⋆ Make best effort to terminate (e.g., when no p_i fails despite the fact $I \notin C$)

Notations (1)

- $\mathcal{V}$= set of values that can be proposed
- $J[1..n]$: vector of values actually proposed
- $\mathcal{V}^n$= set of all possible input vectors
- I: a vector of $\mathcal{V}^n$
- $\mathcal{V}^n_{fc}$= set of vectors with at most f_c entries equal to $\perp$
- $J1, J2 \in \mathcal{V}^n_{fc}$, $J1 \leq J2$ if $\forall k: \; J1[k] \neq \perp \Rightarrow J1[k] = J2[k]$
- $\#_x(J)$ = nb of entries of J whose value is x ($x \in \mathcal{V} \cup \{\perp\}$)
- $d_{\not\perp}(I, J)$ = nb of corresp. non-$\perp$ entries that differ
- $d(I, J)$ = Hamming distance = $\#_x(J) + d_{\not\perp}(I, J)$

Notations (2)

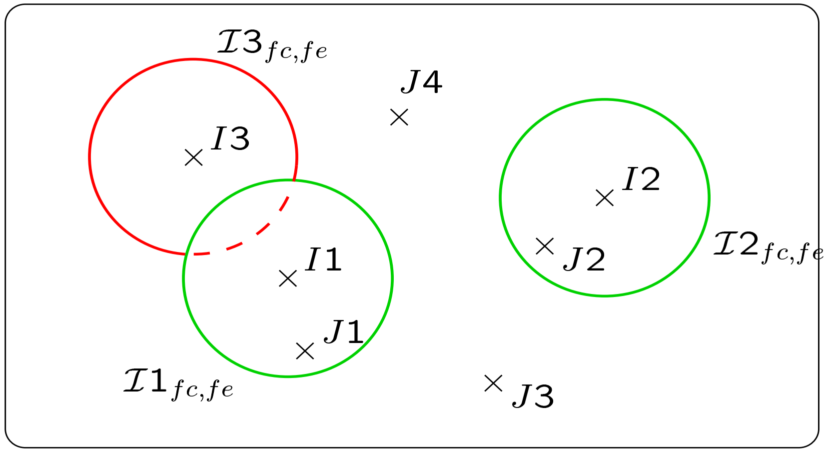
Given a vector $I \in \mathcal{V}^n$:

$$\mathcal{I}_{f_c,f_e} \; = \; \{ J \mid \#_\perp(J) \leq f_c \; \wedge \; d_{\not\perp}(I, J) \leq f_e \; \}$$

and for a subset C of $\mathcal{V}^n$,

$$\mathcal{C}_{f_c,f_e} \; = \; \bigcup_{I \in C} \mathcal{I}_{f_c,f_e}$$

Example of Spheres



Condition-Based IC Problem

We say that *an (f_c, f_e)-fault tolerant protocol solves the CB_IC problem for a condition C*, if in every execution whose proposed vector J belongs to $\mathcal{V}^n_{fc}$, the protocol satisfies the following properties:

- CB_IC-Agreement: No two different vectors are decided
- CB_IC-Termination: If
  - ⋆ (1) $J \in \mathcal{C}_{f_c,f_e}$ and no more than f_c processes crash, or
  - ⋆ (2.a) no process crashes, or (2.b) a process decides

  then every crash-correct process decides
- CB_IC-Validity: If $J \in \mathcal{C}_{f_c,f_e}$, then the decided vector D is such that $J \in \mathcal{D}_{f_c,f_e}$ with $D \in C$

A Set of IC Conditions

- There are different ways to define conditions
- Acceptability: useful to design IC protocols

  From an operational point of view we define a condition with a predicate P and a function S
- Legality: useful to prove impossibility results

Acceptability of a Condition

A condition C is (f_c, f_e)-*acceptable* if there exist a predicate P and a function S satisfying the following three properties:

- Property $\mathsf{T}_{C\to P}$:

$$I \in C \Rightarrow \ \forall J \in \mathcal{I}_{f_c,f_e} : \ P(J),$$

- Property $\mathsf{A}_{P\to S}$: $\forall J1, J2 \in \mathcal{V}^n_{f_c}$:

$$(J1 \le J2) \wedge P(J1) \wedge P(J2) \Rightarrow S(J1) = S(J2)$$

- Property $\mathsf{V}_{P\to S}$: $\forall J \in \mathcal{V}^n_{f_c}$:

$$P(J) \ \Rightarrow \ S(J) = I \text{ such that } I \in C \wedge J \in \mathcal{I}_{f_c,f_e}$$

Main Results

- Definition: condition C is (f_c, f_e)-*legal* if for all distinct $I1, I2 \in C$, $d(I1, I2) \ge 2f_e + f_c + 1$.
- Theorem 1: If there is an (f_c, f_e)-fault tolerant protocol that solves CB_IC for C, then C must be (f_c, f_e)-legal
- Theorem 2: (f_c, f_e)-legality $\Leftrightarrow$ (f_c, f_e)-acceptability
- Theorem 3:

> The CB_IC problem for a condition C is (f_c, f_e)-fault tolerant solvable iff the condition C is (f_c, f_e)-legal

Shared Memory Condition-Based Protocol (1)

- The shared memory is made up of two arrays $V[1..n]$ and $W[1..n]$of single-writer multi-reader atomic registers (each initialized to $\perp$)
- Linearizable operations (X is V or W)
  - ⋆ write$(v, X[i])$ allows p_i to atomically write the value v into the register $X[i]$
  - ⋆ read$(X[i])$ allows p_j, $1 \le j \le n$, to atomically read the content of the register $X[i]$
  - ⋆ snapshot(X) allows p_j, $1 \le j \le n$, to atomically read the content of all the registers of X (Cost: $O(n \log n)$)

Shared Memory Condition-Based Protocol (2)

Each process p_i:

- First writes its input value in $V[i]$
- Takes snapshots until it gets a view V_i with at most $(n-f)$ inputs
- If $P(V_i)$ holds it decides $S(V_i)$ and helps the others writing the vector it decides in $W[i]$
- Otherwise, it helps the others writing $\top$ in $W[i]$, and does its best to terminate, looking at W:
  - ⋆ To benefit from another process decision
  - ⋆ To discover that all processes have proposed a value

A Shared Memory Condition-Based Protocol (3)

Function *SM_IC*(v_i)
  write$(v_i, V[i])$;
  repeat $V_i \leftarrow$ snapshot(V) **until** $(\#_\perp V_i \leq f_c)$ **endrepeat**;
  if $P(J_i)$ **then** $w_i \leftarrow S(J_i)$ **else** $w_i \leftarrow \top$ **endif**;
  write$(w_i, W[i])$;
  repeat $W_i \leftarrow$ collect(W);
    if $(\exists\, j : W_i[j] \neq \perp, \top)$ **then return**$(W_i[j])$ **endif**
  until $(\perp \notin W_i)$ **endrepeat**;
  return (collect(V))

Error-Correcting Codes vs IC (1)

- code = set of codewords
- codeword = word of length n on an alphabet $\mathcal{A}$
- Theorem:
  A code C is t *error/*e *erasure* decoding iff its minimal Hamming distance is $\geq 2t+e+1$

Error-Correcting Codes vs IC (2)

- Alphabet $\mathcal{A}$/set of values $\mathcal{V}$
- codeword/ vector of the condition
- codeword length/number of processes (n)
- code/condition
- erasure/process crash (upper bounded by f_c)
- alteration/erroneous proposal (upper bounded by f_e)
- word received/proposed input vector
- decoding/deciding

Error-Correcting Codes *vs* IC (2)

Codeword sent	channel behavior for the ith digit	Word received
Vector of the condition	behavior of process p_i	Proposed vector
$I = [v_1, \ldots, v_n]$		$J = [r_1, \ldots, r_n]$
v_1	correct transmission/p_1 is correct	$r_1 = v_1$
.	./.	.
v_x	transmission error/erroneous proposal	$r_x = v'_x$
.	./.	.
v_y	transmission erasure/process crash	$r_y = \perp$
.	./.	.
v_n	correct transmission/p_n is correct	$r_n = v_n$

From ECC to IC (1)

Let C be the set of 8 codewords (whose code dist is 3):

0 0 0 0 0 0	1 1 1 0 0 0	0 1 0 1 0 1	1 0 1 1 0 1
1 0 0 1 1 0	0 1 1 1 1 0	1 1 0 0 1 1	0 0 1 0 1 1

This code can be obtained from the check matrix A:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

From ECC to IC (2)

- Let a system with $n = 8$ processes that propose values in $\mathcal{V} = \{0, 1\}$
- As its distance is 3 the previous ECC can cope with, e.g., $f_c = 2$ crashes and $f_e = 0$ erroneous proposals
- The matrix A provides a simple way* to define C:

$$C = \{I \text{ such that } syndrome(I) = 0\}$$

- Pred P and function S associated with such a C are:

$$P(J) = \exists I: \text{ such that } J \in \mathcal{I}_{f_c,f_e} \ \wedge \ syndrome(I) = 0$$
$$S(J) = I \text{ such that } J \in \mathcal{I}_{f_c,f_e} \ \wedge \ syndrome(I) = 0$$

*$syndrome(I) = A \times I^T$.

From IC to ECC

- Let a code C be *(f_c, f_e)-perfect* if the spheres whose centers are the vectors I of C define a partition of $\mathcal{V}^n_{f_c}$:

$$I1 \neq I2: \ \mathcal{I}1_{f_c,f_e} \cap \mathcal{I}2_{f_c,f_e} = \emptyset \text{ and } \mathcal{V}^n_{f_c} \ = \ \bigcup_{I \in C} \mathcal{I}_{f_c,f_e}$$

- Theorem:

  There is no (f_c, f_e)-*perfect* code for $f_c \geq 1$, $f_e < n$

- Proof: Follows from FLP impossibility result.

Consensus and ECC

Consensus and

Error Correcting Codes

Agreement Problem: Consensus (crash)

- C-Agreement:
  No two different values are decided
- C-Termination:
  A process that does not crash decides
- C-Validity:
  A decided value v is a proposed value

Consensus and Error Correcting Codes (1)

- An input vector can be seen as as a codeword made up of n digits (one per process) encoding the decided value (which is included in the vector)
- let us consider this weakened definition of consensus:
- CB_C-Validity:
  If the input vector belongs to the condition C, then a decided value is a proposed value
- CB_C-Agreement:
  If the input vector belongs to the condition C, then no two different values are decided
- CB_C-Guaranteed Termination:
  If at most f processes fail and the input vector belong to C, then every correct process p_i eventually decides

Consensus and Error Correcting Codes (2)

- Let a d-admissible code be a mapping $C : \mathcal{V}^n \rightarrow \mathcal{V}$ such that:
  - ⋆ The Hamming distance of every two code-words coding different values in $\mathcal{V}$ is at least d, and
  - ⋆ At least one of the digits in each codeword mapped to a value $v \in \mathcal{V}$ is v

Consensus and Error Correcting Codes (3)

Function *MP_Consensus*(v_i)

$\forall j$: send val1(v_i, i) to p_j;
wait until (at least $(n-f)$ val1 msgs have been delivered);
$\forall j$: **do if** val1(v_j, j) has been delivered **then** $V_i[j] \leftarrow v_j$
else $V_i[j] \leftarrow \perp$ **endif**
enddo;
$w_i \leftarrow$ match $(V_i$,*code*$)$;
if $w_i \neq \perp$ **then return**(w) **endif**

Consensus and Error Correcting Codes (4)

- Theorem 1:
  If a condition C allows to solve consensus in the crash failure model (with $\leq f_c$ faulty processes), then C consists of codewords of an (f_c+1)-admissible code
- Theorem 2:
  If a condition C allows to solve consensus in the Byzantine failure model (with $\leq f_b$ faulty processes), then C consists of codewords of a $(2f_b+1)$-admissible code

Consensus and Error Correcting Codes (5)

From CB_C-Agreement to C-Agreement: Enrich the protocol

$w_i \leftarrow$ match $(V_i$,*code*$)$;
$\forall j$: send val2(w_i, i) to p_j;
repeat **wait** for a new val2(w_j, j) message;
if val2$(w, -)$ with the same non-$\perp$ value w has been received from at least $(n-f)$ processes
then return(w)
endif
endrepeat

$\boxed{f_c < n/2 \text{ or } f_b < n/3}$

Consensus and Error Correcting Codes (6)

From CB_C-Validity to C-Validity: Strengthen the Condition

- Let a strongly d-admissible code be a mapping $C : \mathcal{V}^n \rightarrow \mathcal{V}$ such that:
  - ⋆ The Hamming distance of every two codewords coding different values in $\mathcal{V}$ is at least d, and
  - ⋆ At least d of the digits in each codeword mapped to a value $v \in \mathcal{V}$ are v
- Theorem 3:
  A condition C allows to solve consensus in the crash failure model (with at most f_c faulty processes) iff C consists of codewords of a strongly (f_c+1)-admissible code

Conlusion

- Condition-Based approach to the Interative Consistency pb
- Characterization of the set of necessary and sufficient conditions
- Connexion relating agreement problems with Error-Correcting codes