Some Multi-Agent Systems Must not Fail …

… not even with such drivers …
Outline

- Basics of Combinatorial Topology in Distributed Computing
- [Topological Modeling of Message-passing systems]
- Topological Modeling of Byzantine Systems
- [Point-Set Topology in Distributed Computing]
Decision Tasks

• **Important class of distributed computing problems („one-shot“)**
  – Example 1: **k-set agreement**
    • n+1 processes with local input values \( x_0, \ldots, x_n \), taken from some finite set \( V \) [with \( |V| \geq k+1 \) to circumvent a trivial solution]
    • Every [non-crashing] process \( p \) must irrevocably compute a local output value \( y_p \in \{x_0, \ldots, x_n\} \)
    • System-wide at most \( k \) different \( y_p \)
  – Example 2: Muddy Children Puzzle
    • Recast as Cheating Husbands Problem (CHP) in [MDH86]

• **Processes execute a distributed algorithm** (protocol) for solving the task. It depends heavily on the underlying system model:
  – Message-passing or shared memory communication
  – Synchronous or asynchronous
  – Number and severeness of faults
Key for Solvability: (In)Distinguishability

- Kripke model for Muddy Children Puzzle (3 children):

- Categorically equivalent: Simplicial complex
  - makes local (= per agent) indistinguishabilities explicit
Epistemic Logic and Topology

Use simplicial complexes/combinatorial topology for epistemic reasoning:

- Interpreting existing epistemic results topologically may result in new insights regarding computability
- The powerful theorems of combinatorial topology may lead to new methods for epistemic reasoning
Basics Simplicial Complexes [HKR14]

- **Geometric simplex** $S^n$, of dimension $n$
  - Used to model global state of $n+1$ processes, represented by *the polytope spanned by* $n+1$ affinely independent vertices $\{v_0, v_1, \ldots, v_n\}$
    - colored by the process ids
    - labeled by the local states of the processes
  - **Face** = simplex defined by a subset of the vertices

- **Simplicial complex** $C$
  - set of simplices
  - closed w.r.t. faces
  - **Facets** = maximal faces of $C$
  - $C$ is pure if all facets have same dimension (usually $n$)
Topological Modeling of Decision Tasks (1)

- **Input complex $I$**
  - Encodes all allowed input value assignments
  - Often a pseudosphere (union of simplices labeled independently with a value from some finite set $V$)

- **Output complex $O$**
  - Encodes all allowed output value assignments

- **Protocol complex $P$**
  - Evolution of input complex during protocol execution
  - Depends on synchrony, communication and fault model
Example Protocol Complex (3 Unfaithful CHP)

Before queen’s statement:
Input complex $I$

After first night

After second night

After queen’s statement
More on Protocol Complexes: Layering

Processes are usually assumed to execute a full-history protocol:

- Executions evolve in layers 1, 2, …
  - At the beginning of layer k, processes broadcast their complete layer-(k-1) state [called view; the layer-0 state contains the input value]
  - Processes build their layer-k views as the union of the views received in layer-k
  - Form a simplex $\alpha$ by joining the vertices (process id, layer-k view) originating in a single layer-k communication pattern

- The layer-k protocol complex $\mathcal{P}_k$ consists of all such simplices $\alpha$
Topological Modeling of Decision Tasks (2)

- Decision task \( \mathcal{T} \) is specified by \( \mathcal{T} = (I, \Delta, O) \)
  - Task specification carrier map \( \Delta : I \rightarrow 2^O \) assigns to each possible input simplex \( \sigma \in I \) a set of allowed output simplices \( \Delta(\sigma) \subseteq O \)
  - \( \Delta \) is usually rigid, i.e., preserves the dimension of \( \sigma \)

- Solution algorithm is given by \( P + \) decision map \( \mu : P \rightarrow O \):
  - Vertex map \( \mu \) maps simplex \( \tau \in P \) in the carrier \( \sigma = \sigma(\tau) \in I \) to simplex \( \mu(\tau) \in \Delta(\sigma) \)
Protocol Complex for IIS SHM Model (I)

- IIS model (≡ Asynchronous wait-free R/W shared memory) [HS99]:
  - 1-layer protocol complex is **chromatic subdivision** $\chi(I)$ of input complex $I$
  - $k$-layer protocol complex is $\chi^k(I)$
  - $\chi^k(I)$ is carrier-preserving: Every simplex $\tau \in \chi^k(I)$ lies in some input simplex $\sigma \in I$ (the carrier of $\tau$)
Protocol Complex for IIS SHM Model (II)

Subdivided neighbor-simplices are **boundary-consistent**: 

![Diagram of Protocol Complex for IIS SHM Model (II)]
Protocol Complex for IIS SHM Model (II)

\[ \chi^2(I) : \text{two rounds of IIS} \]

U. Schmid
Example Decision Task: k-Set Agreement (1)

• Recall problem description:
  – \( n+1 \) processes with local input values \( x_0, \ldots, x_n \), taken from some finite set \( V \), with \( |V| \geq k+1 \)
  – Every [non-crashing] process \( p \) must irrevocably set output value \( y_p \in \{x_0, \ldots, x_n\} \)
  – System-wide at most \( k \) different \( y_p \)

• Task specification:
  – \( I \) is a pseudosphere with labels from \( V \)
  – \( O \) is the union of all pseudospheres with labels from any \( k \)-subset of \( V \)
  – \( \tau \in \Delta(\sigma) \) if \( \text{labels}(\tau) \subseteq \text{labels}(\sigma) \)

• Wait-free impossible for \( k \leq n \)
Example Decision Task: \(k\)-Set Agreement (2)

Proof of wait-free impossibility for \(k = n\):

- Decision map \(\mu\) assigns a label
  \([= \text{the decision value}]\) to every vertex in \(P\)

- Sperner’s lemma:
  - For a disjointly labeled facet \(\sigma \in I\)
    (where all \(n+1\) vertices are labeled differently)
  - for an arbitrary (but carrier-consistent) labeling of the vertices of \(\chi^k(\sigma)\), any \(k\)
    - there is a disjointly labeled \(\tau \in \chi^k(\sigma)\)

- Contradicts specification for \(n\)-set agreement among \(n+1\) processes
Topological Modeling of Byzantine Systems
Byzantine Agents Cause Troubles!

• What if process P (😊) in \( P^1 \) for IIS is byzantine?
  – (P,p) in Q’s view (Q, {(P,p), (Q,q)}) can be anything (only P and Q know what it actually is!)
  – (P,p) in R’s view (R, {(P,p), (Q,q), (R,r)}) can also be anything, and can even be different from (P,p) in Q’s view if P commits equivocation!

➢ P may cause (possibly even infinitely!) many instances of the original facet \( \alpha \)!

Protocol complex blows up tremendously …
What can we do?

1. Assign some local state also to a byzantine process:
   - As we have advocated in [KPSF19], even a byzantine agent P can be attributed a well-defined state $S_P$, as defined by its current history
   - P‘s (faulty) behavior can be viewed as following a hidden protocol

2. Employ a generalized protocol complex construction:
   - Incorporate a state obfuscation function $M_k$ that, given P‘s state $S_P$ in layer k-1 and some process Q, defines the view $(P,p) = M_k(S_P,Q)$ that Q will receive from P for layer k
   - Note: If P commits equivocation, then $M_k(S_P,Q) \neq M_k(S_P,R)$
   - Limitation: All vertices for receiver Q will get the same view $M_k(S_P,Q)$ [which is reasonable, as P would need clairvoyancy w.r.t. the global system state otherwise]

3. Do not restrict the behavior of byzantine agents:
   - The task specification $(I, \Delta, O)$ may only restrict correct processes
   - Exempt byzantine processes from applying the decision function $\mu$
Consequences

• This generalizes much of the topological modeling of crash-prone processes to byzantine agents:
  – A faulty process is assigned some specific state
  – The layer-k view construction for non-byzantine processes is the same as in crash-prone systems, except that we need to apply the state obfuscation function M

• However, the protocol complex still suffers from the blowup problem:
  – The layer-(k-1) view of a byzantine agent P may lead to arbitrarily many different layer-k views for P
  – State obfuscation functions that allow equivocation still cause arbitrarily many instances of any facet

• Additional measures are needed to avoid this …
Mendes, Tasson and Herlihy [MTH14]

Ground-breaking approach, developed in Hammurabi Mendes´ PhD thesis [Men16]. It reduces task solvability in

- asynchronous byzantine message passing systems
- asynchronous crash-prone message passing systems

to each other:

**Theorem:** A byzantine task \((I, \Delta, O)\) is solvable if and only if the counterpart task \((\overline{I}, \overline{\Delta}, \overline{O})\) is solvable under process crash faults

**Main ingredients:**

1. Restrict byzantine task specifications to non-faulty processes
2. Use (an extension of) the Consistent Broadcasting primitive by Srikanth and Toueg [ST87] to make equivocation impossible
3. Force state obfuscation functions to mimic crash faults
1. Restrict Byzantine Task Specification

Byzantine processes are indistinguishable from correct ones if they do not behave obviously badly, yet

- could introduce input values incompatible with the inputs of the correct processes w.r.t. $I$

Correct processes cannot reliably find out who is correct and who not, yet

- shall be restricted to choose output values that are inputs of correct processes only („strong validity“)

### Byzantine task specification $\mathcal{T}=(I, \Delta, O)$:

Any facet in the pure complexes $I$ and $O$ must also represent a non-faulty configuration for $\mathcal{T}$:

- $\sigma \in I$ if there is a facet $\sigma_1 \supseteq \sigma$ representing a non-faulty initial configuration for $\mathcal{T}$
- $\sigma \in O$ if there is a facet $\sigma_0 \supseteq \sigma$ representing a non-faulty final configuration for $\mathcal{T}$
- $\Delta(\sigma)$ is a rigid carrier map, with $\Delta(\sigma)=\emptyset$ for $\dim(\sigma) < n-f$ [remember: $n+1$ processes]

### Crash-only counterpart task $\mathcal{\overline{T}}=(\overline{I}, \Delta, \overline{O})$:

Satisfies the original $\Delta$ for any choice of up to $f$ crashing processes:

- $\sigma \in \overline{I}$ if there is a facet $\sigma_1 \supseteq \sigma$ representing a possible initial configuration for $\overline{T}$, which satisfies $\exists \sigma' \in \overline{I}$, $\sigma' \subseteq \sigma_1$, $\dim(\sigma') \geq n-f$
- $\sigma \in \overline{O}$ if there is a facet $\sigma_0 \supseteq \sigma$ representing a possible final configuration for $\overline{T}$, which satisfies $\exists \sigma' \in \overline{O}$, $\sigma' \subseteq \sigma_0$, $\dim(\sigma') \geq n-f$
- $\Delta(\sigma)$ is a rigid carrier map, with $\Delta(\sigma)=\emptyset$ for $\dim(\sigma) < n-f$, and $\Delta(\sigma) = \{ \tau \in \overline{O} | \forall \sigma' \subseteq \sigma, \sigma' \in \overline{I}, \dim(\sigma') \geq n-f \ \exists \tau' \in \Delta(\sigma') \subseteq \overline{O} \text{ and } \tau' \subseteq \tau \}$
2. Utilize Atomic Reliable Broadcasting (ARB)

ARB specification:

- (C) If non-faulty P calls ARB-send(P,k,c), then all non-faulty processes will call ARB-deliver(P,k,c) eventually.

- (U) If non-faulty P never calls ARB-send(P,k,c), then no non-faulty process will call ARB-deliver(P,k,c).

- (R) If some non-faulty process calls ARB-deliver(P,k,c), then every other non-faulty process will call ARB-deliver(P,k,c) eventually.

- (NE) If two non-faulty processes call ARB-deliver(P,k,c) resp. ARB-deliver(P,k,c'), then c=c'.
A Note on „Silence“ [GM20]

• Communication by time [Lam79] for efficiently sending a binary message in a fault-free synchronous distributed system:
  – Send value 1: send a message in a round
  – Send value 0: don’t send a message in a round
  – Does not work if sender may crash!

• Goren and Moses [GM20] identified silent choirs as a pivotal communication primitive in crash-prone synchronous systems:
  – A silent choir of f+1 processes (+ conductor) is needed for communication by time in crash-prone synchronous systems
    • One correct choir member suffices to send 1
    • All f+1 choir members must be silent to send 0
    • Also works when f choir members crashed!
  – Epistemic analysis established necessary and sufficient communication patterns for efficient distributed algorithms in such systems
A Note on Consistent Broadcasting [ST87]

In byzantine systems, a stronger primitive than a silent choir is needed

Here is it:

- **Send 0**: Correct sender does not call ARB-send
  - By (U), no ARB-deliver occurs at a correct process
- **Send 1**: Sender calls ARB-send
  - By (C), all processes call ARB-deliver if the sender is correct
  - By (R), either every or no correct process calls ARB-deliver if sender is faulty

(U, C, R) is provided by Consistent Broadcasting [ST87] alone

- ARB is only needed to also rule out equivocation when sending a non-binary message (i.e., other values than 0 and 1)

CB may be considered as the **byzantine analog of a silent choir**!
A Note on „Fire!“ [FKS21]

In [FKS21], we provided a thorough epistemic analysis of the **Firing Rebels with Relay** (FRR) problem in byzantine systems.

FRR is just Consistent Broadcasting with empty messages (= events)

\[
I \models \overline{\text{fire}}_i \rightarrow B_i(\text{start} \land C^H \text{start}).
\]

- **Belief** \( B_i \varphi := K_i(\text{correct}_i \rightarrow \varphi) \)
- **Hope** \( H_i \varphi := \text{correct}_i \rightarrow B_i \varphi \)
- **Eventual common hope** \( C^H \varphi \) defined as the greatest fixpoint of the equation \( \chi \leftrightarrow E^H(\varphi \land \chi) \)
3. Force State Obfuscation to Mimic Crashes

- No equivocation (NE) + atomicity (R) $\rightarrow$ even byzantine agents can at most **consistently** lie about their state to correct agents
- Full history protocol $\rightarrow$ correct agent $Q$ can **validate** information received from agent $P$:
  - The **layer-1 view** of $P$ received by $Q$ must be consistent with some initial configuration
  - The **layer-k view** of $P$ received by $Q$ must
    1. contain at least $n+1-f$ many layer-$(k-1)$ views from different processes
    2. contain the layer-$(k-1)$ view of $P$ itself, and must match the layer-$(k-1)$ view of $P$ received by $Q$ earlier
    3. be consistent with what $Q$ **could** receive in layer $k$ in some correct run of the crash-prone solution algorithm for the counterpart task
  - An agent $P$ that ever provides non-validateable information is **persistently marked as crashed**: all its views are subsequently dropped!
Validated Views allow Simulation of Solution Algorithm for Crashes

• As soon as the validated views of Q allow it to apply the decision function of the crash-prone solution algorithm, Q can decide.

• The validated views maintained by correct processes can be proved to correspond to a single run of the crash-prone system \(\Rightarrow\) decisions of Q and R are consistent w.r.t. output complex \(O\)

Details see [MTH14] and [Men16]
Limitations of [MTH14]

The equivalence theorem is a very strong result, which might seduce one to conclude

„Don‘t worry about byzantine processes“

However, it rests on quite strong assumptions:

• The implementation of ARB requires $n+1 > 3f$ processes and a fully-connected network

• The simulation algorithm does not terminate:
  – Validated view construction needs to support late-booting processes (perpetually „announce the decision“)
  – ARB needs to support late broadcasts

• Equivalence theorem only proved for decision tasks
References


The End

Thank you for your attention!