Defending Distributed Cyber-Physical Systems with Bounded Time Recovery

Brian Sandler, Neeraj Gandhi, Linh Thi Xuan Phan, Andreas Haeberlen
NSF/Intel CPS PI Meeting
July 2018
Machines in Control

Vulnerable CPS can cause disaster.
- Explosion
- Equipment damage
- Power outages
- …

We want to prevent disaster.

Bellingham, WA
*Oil pipeline explosion after the two controlling computers failed.*

Iran
*Stuxnet vulnerability destroyed centrifuges used for nuclear enrichment.*

Ivano-Frankivsk, Ukraine
*Controlling power grid systems were compromised leaving residents in the dark.*
Goal: General Defense

Crashes
Non-Crash Bugs
Hacking

Byzantine Faults

BTR - NSF/Intel PI Meeting - July 2018
Example: Industrial Automation

Let’s take a simple example system…
Example: Industrial Automation

This system will run four applications.
Example: Industrial Automation

We’ll focus on the burner control application…
Example: Impact of Failures

What can go wrong?

N₄ can drop or delay messages and ruin the chemical processing.

N₄ can send an incorrect value to A₁ and light the building on fire.
State of the Art: Byzantine Fault Tolerance

Benefits
- Adversarial Scenarios
- Strong Guarantees
- Nice Programming Model

**Problem**: Expensive
- Lots of redundancy

**Problem**: Not Sufficient
- Cannot meet deadlines
- Guarantees drop after $f$ faulty nodes
Is continuous perfection required?

How bad is it if the adversary gains control?
Many CPS have properties that resist quick changes
  • inertia
  • thermal capacity
We don’t have to always be perfect

We can leverage this!

BTR - NSF/Intel PI Meeting - July 2018
For how long is faulty behavior okay?

Different applications have different tolerances.

<table>
<thead>
<tr>
<th>Application</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC/DC converters (STM)</td>
<td>20μs</td>
</tr>
<tr>
<td>Direct torque control (ABB)</td>
<td>25μs</td>
</tr>
<tr>
<td>AC/DC converters</td>
<td>50μs</td>
</tr>
<tr>
<td>Electronic throttle control (Ford)</td>
<td>5ms</td>
</tr>
<tr>
<td>Traction control (Ford)</td>
<td>20ms</td>
</tr>
<tr>
<td>Micro-scale race cars</td>
<td>40ms</td>
</tr>
<tr>
<td>Autonomous vehicle steering</td>
<td>50ms</td>
</tr>
<tr>
<td>Energy-efficient building control</td>
<td>500ms</td>
</tr>
</tbody>
</table>

Source: M. Morari. Fast model predictive control (mpc).

A time period usually exists where faulty behavior is ok so long as the system returns to its correct behavior within that period.

BTR - NSF/Intel PI Meeting - July 2018
Approach: Bounded Time Recovery

BTR guarantees that system recovers from any fault within a short period of time, so that the end goal will be met. Weaker guarantee is often sufficient.
So, how do we make this happen?

REBOUND
. Planning

Before system is compromised, think about what it should do. System operates in different modes for any given set of faults. Can drop less critical tasks as necessary.
REBOUND

2. Detection

Nodes watch over each other to detect faults.

Evidence

N₄ is faulty.
3. Consistency

Flood evidence throughout the system.

N₄ is faulty

Evidence
SEND...
RECV...
...

N₄ is faulty.
4. Adaptation
Each node independently transitions to a new mode
Outline

Problem Introduction ✓
Bounded Time Recovery ✓
REBOUND ✓

Technical Components
  1. Planning
  2. Detection
  3. Consistency
  4. Adaptation

Results
For every* mode, we have a precomputed schedule and plan for every node.

Schedule generated offline
When tasks should run and where
Many constraints
Dependent scheduling problem
Builds a tree

Can limit the number of faults to improve computation time.
2. Detection

Omission Faults
• Declare link faulty if an expected message from a neighbor is not received
• Declaration causes other nodes to change mode.
• Leverage synchrony.

Commission Faults
• Witness/Audit Nodes and Replicas
  • If fault found, log is used as a proof of misbehavior.
  • Large improvement over PeerReview
• Adding synchrony

Challenge: Bounding Time of Detection
3. Consistency

We need a solution where…

- Any two good nodes agree on the state of the system
- The two become aware they cannot communicate

**Strawman**: flood the system periodically with signed attestations of current mode

Actual solution is more efficient
4. Adaptation

Each node individually transitions when its mode changes. When evidence is received a mode change occurs within a bounded period of time.
Challenges

Bounding every step of the algorithms

Overhead of periodic flood
  • Multisignatures → drastically reduce traffic

Handling equivocation
  • Different nodes notifying of different faults to their neighbors

Proving everything
  • Correctness
  • Completeness
  • Bounded detection
  • Bounded stabilization

Planning
  • Unique problem

Theorem 1 (Accuracy (nodes)) If nodes $N_1$ and $N_2$ are correct in round $r$, then $N_2 \notin KN_{N_1,r}$.

Theorem 2 (Accuracy (links)) If nodes $N_1$, $N_2$, and $N_3$, then $KL_r(N_1)$ will not include $(N_2,N_3)$.

Theorem 3 (Completeness) Let $(p_1, p_2, \ldots, p_L)$ be a path and let $i := p_x$ and $j := p_y$ be two nodes on the path such that $x < y$ — that is, $i$ is upstream of $j$. Suppose that...

Theorem 4 (Bounded detection time) Suppose a fault at a node $i$ becomes observable in round $r$, and let $\max E \max_{p \in \text{PATH}(E)} |p|$ be the number of hops on the longest path...

Theorem 5 (Consistency) Let $D := \max_{i,j} D_{i,j}$ be the highest max-fail distance between any pair of nodes. If $N_1$ and $N_2$ are correct and there exists a network path that...
Outline

Problem Introduction ✓
Bounded Time Recovery ✓
REBOUND ✓
Technical Components ✓
  1. Planning
  2. Detection
  3. Consistency
  4. Adaption

Results
Overhead of Schedule Tree

Time depends on:

- The number of nodes.
- Degree of network.
- Number of faulty nodes, f.

Only compute once for the lifetime of the system.

Subtrees easily parallelizable.

\[ f = \# \text{ of faulty nodes protected against} \]
Recovery

Unprotected System, $N_2$ Compromised

![Diagram of system components and actuator output over time graph.](image)
Recovery

Protected System, $N_2$ Compromised

Recovery Period

Actuator output

Time (ms)

250  500  750  1000  1250
Recovery

Protected System, N\textsubscript{1}, N\textsubscript{2}, N\textsubscript{3} Compromised
Key Idea: Period of Imperfection

Many CPS can tolerate a short period of faulty behavior.

Approach: Bounded Time Recovery

Bounded time recovery guarantees that the system quickly returns to correct behavior after a fault.

Solution: REBOUND

Algorithms and protocols to provide BTR distributed systems.

Thank you.