Intrusion Tolerance for Networked Systems Through Two-Level Feedback Control

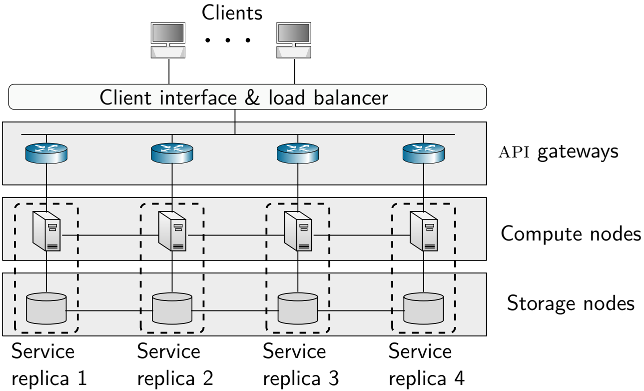
NSE Seminar

Kim Hammar

kimham@kth.se
Division of Network and Systems Engineering
KTH Royal Institute of Technology

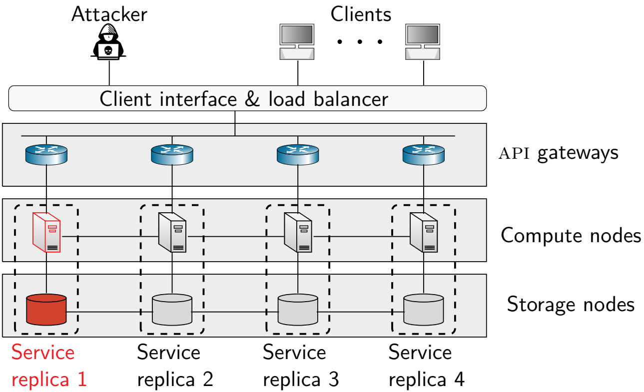
Nov 10, 2023

Use Case: Intrusion Tolerance



- A **replicated system** offers services to a client population.
- The system must be highly available and provide correct service without disruption.

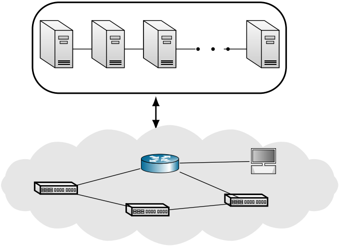
Use Case: Intrusion Tolerance



- An attacker seeks to intrude on the system and disrupt service
- The system should **tolerate intrusions**
  - it should provide correct service
    even if a fraction of replicas are compromised

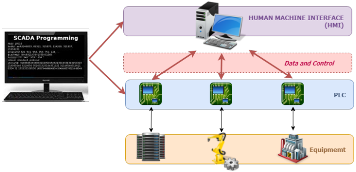
Examples of Systems that Need to Tolerate Intrusions

Control planes

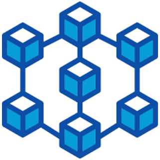


Embedded systems

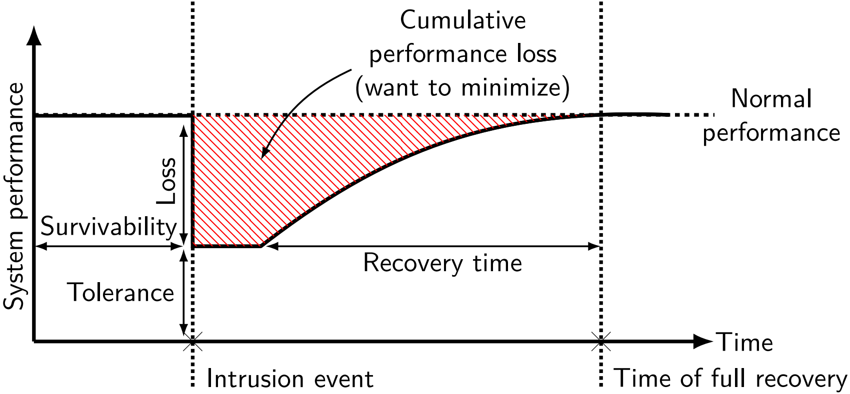
SCADA systems



Payment systems



Intrusion Tolerance (Simplified)

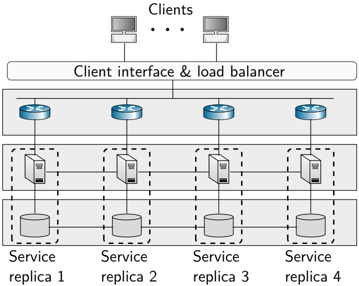


Intrusion-Tolerant Systems - State of The Art

- State-of-the-art intrusion-tolerant systems involve 3 **building blocks**:
  1. a protocol for service replication
  2. a scaling strategy
  3. a recovery strategy

- Given N replicas, the system provides correct service with up to $f = \frac{N-1}{3}$ compromised replicas.
  - Theoretical upper bound
  - f is the tolerance threshold.

- **Simple control strategies**:
  - Fixed number of replicas (no scaling)
  - No recovery or periodic recovery



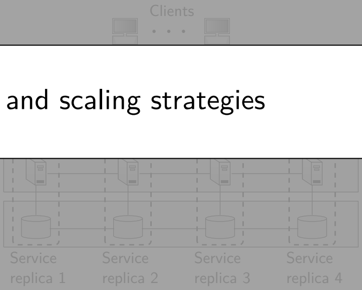
Intrusion-Tolerant Systems - State of The Art

- State-of-the-art intrusion-tolerant systems involve 3 **building blocks**:
  1. a protocol for service replication
  2. a scaling strategy
  3. a recovery strategy

This work: Optimal intrusion recovery and scaling strategies

  - Theoretical upper bound
  - f is the tolerance threshold.

- **Simple control strategies**:
  - Fixed number of replicas (no scaling)
  - No recovery or periodic recovery



Can we use decision theory and learning-based methods to automatically find effective security strategies?

Intrusion prevention
Simulation.
Small-scale. (2020)[1].

Intrusion prevention
Optimal stopping.
Emulation, small-scale
Static attacker. (2021)[2].

Intrusion response
Optimal **multiple** stopping.
Emulation, small-scale.
Static attacker. (2022)[3]

Intrusion response
Dynkin game.
Emulation, small-scale.
Dynamic attacker. (2022)[4]

Intrusion response
Decomposition.
Emulation, **large-scale**
Dynamic attacker. (2023)[5]

Intrusion tolerance
Two-level control.
Integration with BFT.
Static attacker.
(<u>This work</u>)

---

[1] Kim Hammar and Rolf Stadler. "Finding Effective Security Strategies through Reinforcement Learning and Self-Play". In: *International Conference on Network and Service Management (CNSM 2020)*. Izmir, Turkey, 2020.

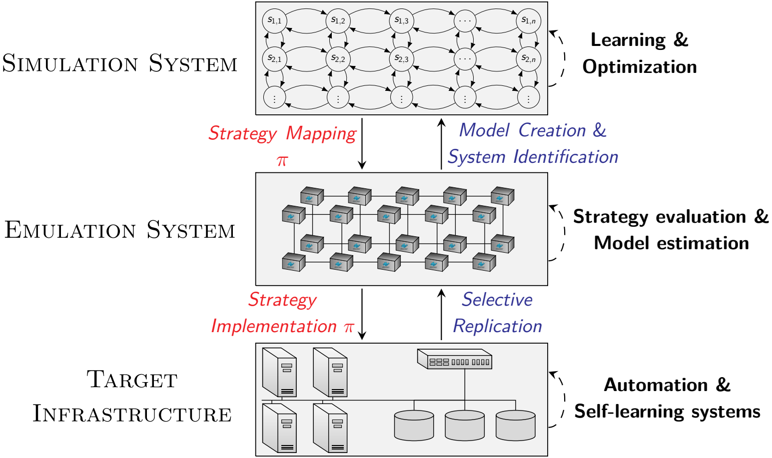
[2] Kim Hammar and Rolf Stadler. "Learning Intrusion Prevention Policies through Optimal Stopping". In: *International Conference on Network and Service Management (CNSM 2021)*. Izmir, Turkey, 2021.

[3] Kim Hammar and Rolf Stadler. "Intrusion Prevention Through Optimal Stopping". In: *IEEE Transactions on Network and Service Management* 19.3 (2022), pp. 2333–2348. DOI: 10.1109/TNSM.2022.3176781.

[4] Kim Hammar and Rolf Stadler. "Learning Near-Optimal Intrusion Responses Against Dynamic Attackers". In: *IEEE Transactions on Network and Service Management* (2023), pp. 1–1. DOI: 10.1109/TNSM.2023.3293413.

[5] Kim Hammar and Rolf Stadler. "Scalable Learning of Intrusion Response through Recursive Decomposition". In: *14th International Conference on Decision and Game Theory for Security*. Avignon, France, 2023.

Our Framework for Automated Security



- Source code: https://github.com/Limmen/csle
- Documentation: http://limmen.dev/csle/
- Demo: https://www.youtube.com/watch?v=iE2KPmtIs2A&

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Outline

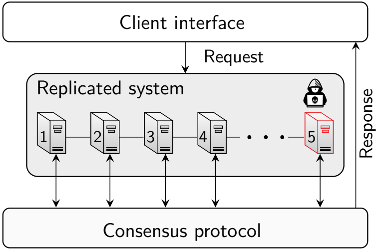
- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Background: Fault Tolerance and Intrusion Tolerance

- Seminal work made by von Neumann and Shannon in 1956.
  - Initially focused on building fault-tolerant circuits.
  - Fault tolerance includes tolerance against: software bugs, malicious attacks, operator mistakes, etc.
- Key strategy for fault tolerance: redundancy
- Redundancy is achieved through service replication. Replicas are coordinated through a **consensus protocol**.



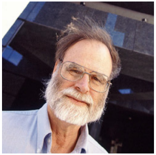
Background: Consensus

- Consensus is the problem of *reaching agreement on a single value among a set of distributed nodes subject to failures*.

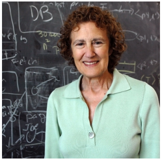
- Fascinating problem for many reasons:
  - Key problem to build **practical systems**
  - The problem comes in many flavors
  - Paradoxical cases
  - Rich theory

- Turing awardees working on consensus:

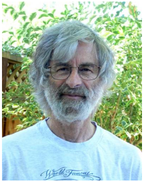
Dijkstra, 1972



Gray, 1998



Liskov, 2008



Lamport, 2013

Background: Consensus

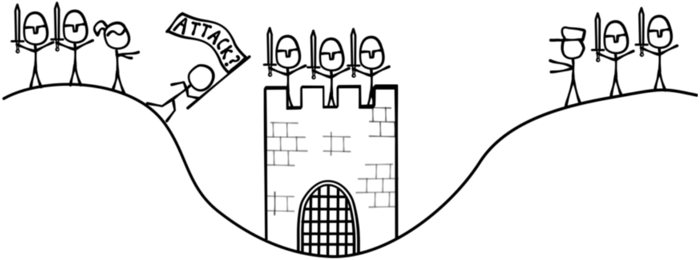
Definition (Consensus)

We have N nodes indexed by $1, \ldots, N$. The nodes are connected by a complete graph and communicate via message passing. Each node starts with an input value $v \in \mathcal{V}$. **The goal is to agree on a single value in $\mathcal{V}$**.

An algorithm A solves consensus if the following hold:

1. **Agreement**: No two correct nodes decide on different values
2. **Termination**: All nodes eventually decide.
3. **Validity**: If all nodes start with input v then they decide on v

Consensus Example: The Two Generals Problem (Gray '78)

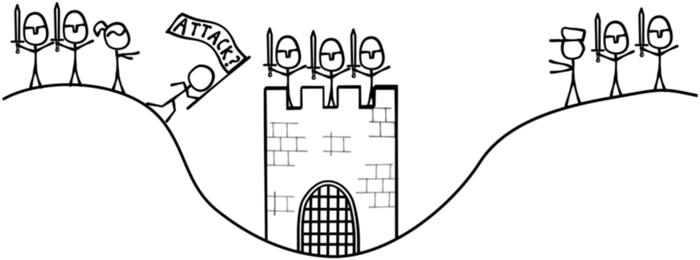


- Two generals are planning a coordinated attack from different directions.

- One of the simplest consensus problems:
  - Only two nodes: 1 and 2
  - No process failures but link failures may occur

- **Is the problem solvable?**

Consensus Example: The Two Generals Problem (Gray '78)



- Two generals are planning a coordinated attack from different directions.

- One of the simplest consensus problems:
  - Only two nodes: 1 and 2
  - No process failures but link failures may occur

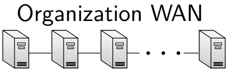
- **Is the problem solvable?** No! Can be proven by contradiction.

When Is Consensus Solvable?

- Solvability depends on synchrony and failure assumptions.
- Three synchronicity models:
  - **The asynchronous model**: no bounds on either delays or clock drifts.
  - **The partially synchronous model**: an upper bound exists but the system may have periods of instability where the upper bound does not hold.
  - **The synchronous model**: there is an upper bound on the communication delay and clock drift between any two nodes.

Synchronous system

Asynchronous system



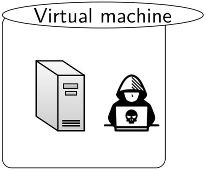
Partially synchronous system

When Is Consensus Solvable?

- Three main failure models:
  - **Crash-stop**: nodes fail by crashing.
  - **Byzantine**: failed nodes may behave arbitrarily (e.g., be controlled by an attacker)
  - **Hybrid**: Byzantine failures but each node is equipped with a trusted component that only fails by crashing.

Crash-stop failure

Byzantine failure



Hybrid failure

When Is Consensus Solvable?

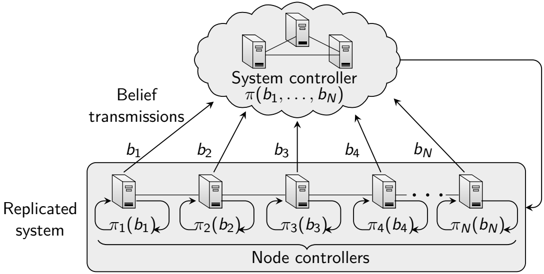
Theorem (Summary of 40 years of research)

- *Consensus is not solvable in the asynchronous model*
- *Consensus is solvable with a reliable network in the partially synchronous model with N nodes and up to*
  - $f = \frac{N-1}{2}$ *Crash-stop failures*
  - $f = \frac{N-1}{3}$ *Byzantine failures*
  - $f = \frac{N-1}{2}$ ***Hybrid failures (assuming authenticated channels)***
- *Consensus is solvable with a reliable network in the synchronous model with N nodes and up to*
  - $f = N - 1$ *Crash-stop failures*
  - $f = \frac{N-1}{2}$ *Byzantine failures (assuming authenticated channels)*
  - $f = \frac{N-1}{2}$ *Hybrid failures (assuming authenticated channels)*

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

Two-Level Feedback Control for Intrusion Tolerance

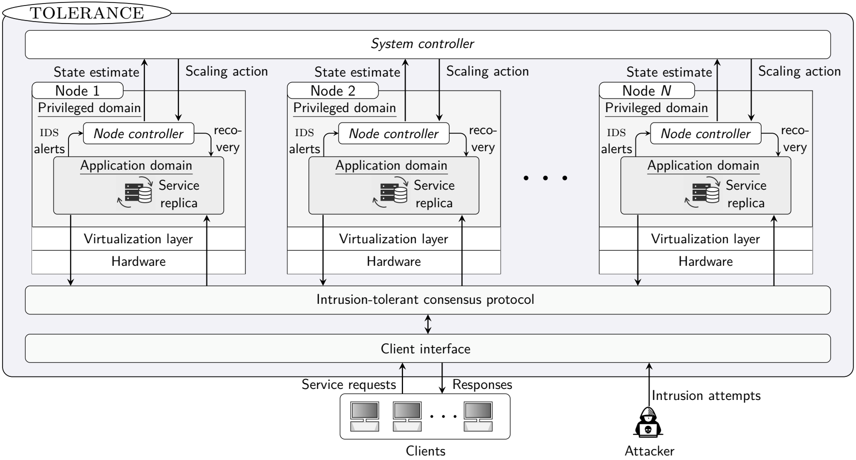


- **Node controllers** with strategies $\pi_1, \ldots, \pi_N$ compute belief states $b_1, \ldots, b_N$ and make local recovery decisions; the belief states are transmitted to a global **system controller** with strategy π, which controls the replication factor

- **Key insight**: the control problems correspond to classical problems studied in operations research, namely the machine replacement problem and the inventory replenishment problem, both of which have been studied for nearly a century.

The TOLERANCE Control Architecture

TOLERANCE: Two-level recovery and scaling control with feedback.



Correctness of TOLERANCE (1/2)

Definition (Correct service)

We say that a system provides correct service if the *healthy* replicas satisfy the following properties:

Each replica executes the same request sequence.	(Safety)
Each request is eventually executed.	(Liveness)
Each executed request was sent by a client.	(Validity)

Correctness of TOLERANCE (2/2)

Proposition

A system that implements the TOLERANCE *architecture provides correct service provided that:*

1. *The controllers can only fail by crashing.*
2. *Network links are authenticated and reliable.*
3. *An attacker can not break cryptographic codes.*
4. *The system is partially synchronous.*
5. *At most k nodes recover simultaneously and at most f nodes are compromised or crashed simultaneously.*
6. $N_t \geq 2f + 1 + k$ *at all times* t.

Remark: TOLERANCE does not ensure confidentiality as a compromised node may leak information to the attacker. By appropriate use of cryptography and firewalls, it is possible to extend TOLERANCE to provide confidentiality. Details omitted.

Correctness of TOLERANCE (2/2)

Proposition

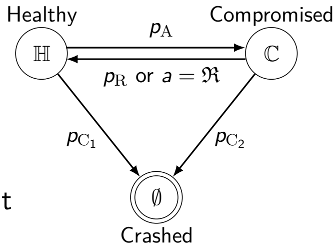
A system that implements the TOLERANCE *architecture provides correct service provided that:*

1. *The controllers can only fail by crashing.*
2. *Network links are authenticated and reliable.*
3. *An attacker can not break cryptographic codes.*
4. *The system is partially synchronous.*
5. ***At most k nodes recover simultaneously and at most f nodes are compromised or crashed simultaneously.***
6. $N_t \geq 2f + 1 + k$ ***at all times t.***

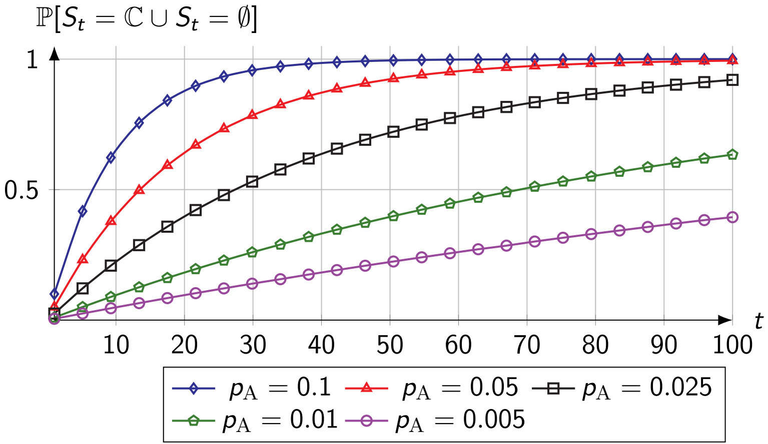
Remark: TOLERANCE does not ensure confidentiality as a compromised node may leak information to the attacker. By appropriate use of cryptography and firewalls, it is possible to extend TOLERANCE to provide confidentiality. Details omitted.

The Local Level: Intrusion Recovery Control (1/4)

- Nodes $\mathcal{N}_t \triangleq \{1, 2, \dots, N_t\}$ and controllers $\pi_{1,t}, \pi_{2,t}, \dots, \pi_{N,t}$.
- Hidden states $\mathcal{S}_{\mathrm{N}} = \{\mathbb{H}, \mathbb{C}, \emptyset\}$ (see figure).
- Actions: **($\mathfrak{W}$)ait and ($\mathfrak{R}$)ecover**
- Observation $o_{i,t} \sim Z$ represents the number of IDS alerts related to node i at time t.
- A node controller computes $b_{i,t} \triangleq \mathbb{P}[S_{i,t} = \mathbb{C} \mid o_{i,1}, \dots]$ and makes decisions $a_{i,t} = \pi_{i,t}(b_{i,t}) \in \{\mathfrak{W}, \mathfrak{R}\}$.



The Local Level: Probability of Failure (2/4)



Probability of node compromise ($S_t = \mathbb{C}$) or crash ($S_t = \emptyset$) in function of time t, assuming no recoveries.

The Local Level: Intrusion Recovery Control (3/4)

Goals: minimize the average time-to-recovery $T_i^{(\mathrm{R})}$ and minimize the frequency of recoveries $F_i^{(R)}$:

$$\begin{aligned} \text{minimize } J_i &\triangleq \lim_{T\to\infty} \left[\eta T_{i,T}^{(\mathrm{R})} + F_{i,T}^{(\mathrm{R})}\right] \\ &= \lim_{T\to\infty} \left[\frac{1}{T}\sum_{t=1}^{T} \underbrace{\eta s_{i,t} - a_{i,t}\eta s_{i,t} + a_{i,t}}_{\triangleq c_{\mathrm{N}}(s_{i,t},a_{i,t})}\right] \end{aligned} \tag{1}$$

We define **intrusion recovery** to be the problem of minimizing the above objective subject to a bounded-time-to-recovery (BTR) safety constraint which ensures that the time between two recoveries of a node is bounded to by Δ_{R}, which can be configured by the system administrator.

The Local Level: Intrusion Recovery Control (4/4)

Problem (Optimal Intrusion Recovery Control)

$$
\begin{aligned}
\underset{\pi_{i,t} \in \Pi_{\mathrm{N}}}{\text{minimize}} \quad & \mathbb{E}_{\pi_{i,t}}\left[J_i \mid B_{i,1} = p_{\mathrm{A}}\right] && \forall i \in \mathcal{N} && \text{(2a)}\\
\text{subject to} \quad & a_{i,k\Delta_{\mathrm{R}}} = \mathfrak{R} && \forall i, k && \text{(2b)}\\
& s_{i,t+1} \sim f_{\mathrm{N}}(\cdot \mid s_{i,t}, a_{i,t}) && \forall i, t && \text{(2c)}\\
& o_{i,t+1} \sim Z(\cdot \mid s_{i,t}) && \forall i, t && \text{(2d)}\\
& a_{i,t+1} \sim \pi_{i,t}(b_{i,t}) && \forall i, t && \text{(2e)}\\
& a_{i,t} \in \mathcal{A}_{\mathrm{N}}, s_{i,t} \in \mathcal{S}_{\mathrm{N}}, o_{it,} \in \mathcal{O} && \forall i, t && \text{(2f)}
\end{aligned}
$$

Numerical Results for the Intrusion Recovery Problem

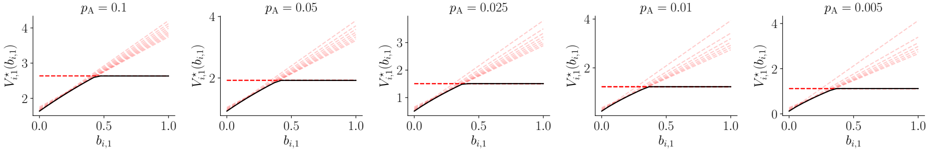


Illustration of the optimal value function $V^\star_{i,t}(b_{i,t})$ for the local control problem. $V^\star_{i,t}$ was computed using the incremental pruning algorithm; **black lines indicate the value function**; red lines indicate the alpha-vectors.

Numerical Results for the Intrusion Recovery Problem

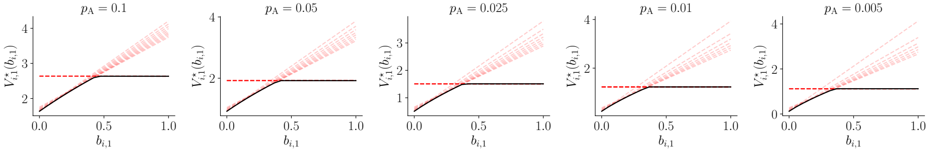
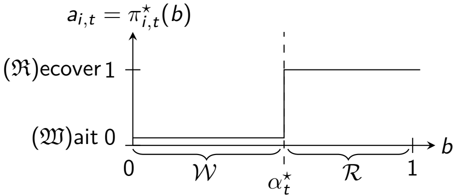


Illustration of the optimal value function $V^{\star}_{i,t}(b_{i,t})$ for the local control problem. $V^{\star}_{i,t}$ was computed using the incremental pruning algorithm; **black lines indicate the value function**; red lines indicate the alpha-vectors.

Based on the above results we hypothesize that there exists an optimal recovery strategy of the form:



Structure of an Optimal Intrusion Recovery Strategy (1/2)

Theorem (Optimal Threshold Recovery Strategies)

If the following holds

$$p_{\mathrm{A}}, p_{\mathrm{U}}, p_{\mathrm{C}_1}, p_{\mathrm{C}_2} \in (0,1) \tag{A}$$

$$p_{\mathrm{A}} + p_{\mathrm{U}} \leq 1 \tag{B}$$

$$\frac{p_{\mathrm{C}_1}(p_{\mathrm{U}} - 1)}{p_{\mathrm{A}}(p_{\mathrm{C}_1} - 1) + p_{\mathrm{C}_1}(p_{\mathrm{U}} - 1)} \leq p_{\mathrm{C}_2} \tag{C}$$

$$Z(o_{i,t} \mid s_{i,t}) > 0 \qquad \forall o_{i,t}, s_{i,t} \tag{D}$$

$$Z \text{ is } \mathrm{TP}\text{-}2 \tag{E}$$

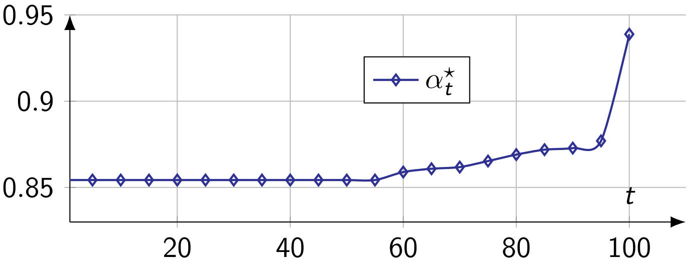
then there exists an optimal recovery strategy $\pi^{\star}_{i,t}$ *for each node* $i \in \mathcal{N}$ *that satisfies*

$$\pi^{\star}_{i,t}(b_{i,t}) = \mathfrak{R} \iff b_{i,t} \geq \alpha^{\star}_t \qquad \forall t, \alpha^{\star}_t \in [0,1] \tag{3}$$

Structure of an Optimal Intrusion Recovery Strategy (2/2)

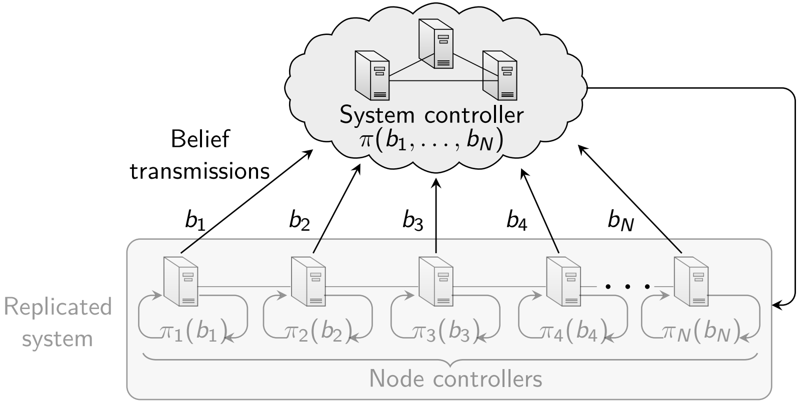
Corollary (Stationary Optimal Strategy as $\Delta_{\mathrm{R}} \to \infty$)

The recovery thresholds satisfy $\alpha^{\star}_{t+1} \geq \alpha^{\star}_{t}$ for all $t \in [k\Delta_R, (k+1)\Delta_R]$ and as $\Delta_{\mathrm{R}} \to \infty$, the thresholds converge to a time-independent threshold $\alpha^{\star}$.

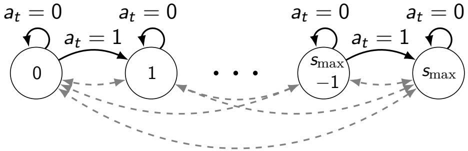


The thresholds were computed using the Incremental Pruning algorithm.

The Global Level: Controlling the Replication Factor (1/6)

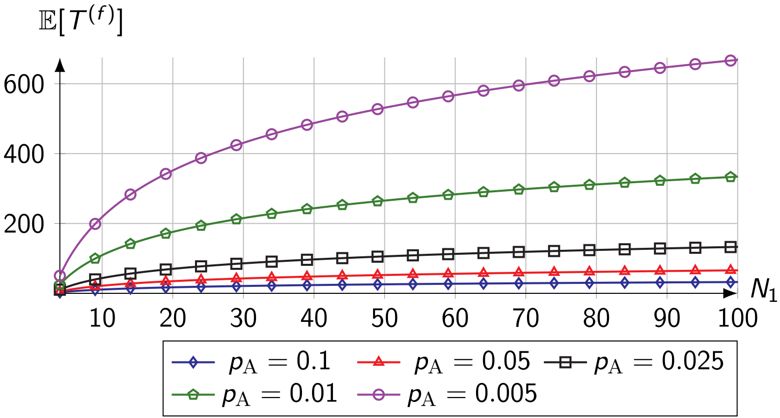


The Global Level: Controlling the Replication Factor (2/6)



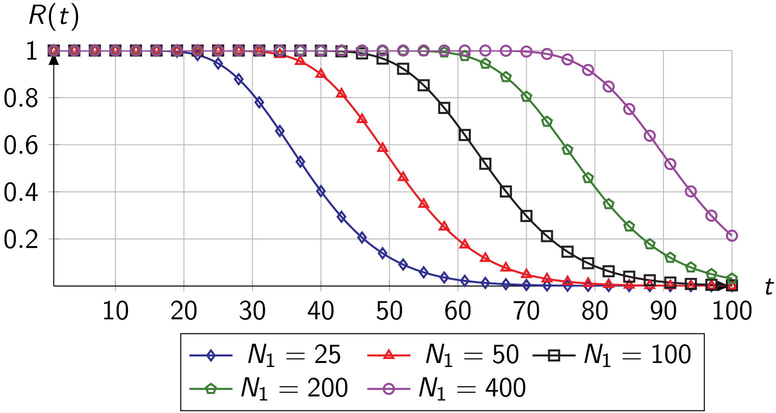
- At each time t, the system controller **receives the belief states** $b_{1,t}, \ldots, b_{N,t}$ from the node controllers and decides if the replication factor N should be increased.
- State s_t: estimated number of healthy nodes based on $b_{1,t}, \ldots, b_{N,t}$
- Actions: $a_t \in \{0, 1\} \triangleq \mathcal{A}_{\mathrm{S}}$, where $a_t = 1$ means that a new node is added to the system and $a_t = 0$ is a passive action.
- $s_{\max}$ is the maximum number of nodes (needed to define the theoretical model). In practice $s_{\max}$ may be very large.

The Global Level (3/6): Mean Time to Failure



Mean time to failure (MTTF) in function of the initial number of nodes N_1; $T^{(f)}$ is a random variable representing the time when $N_t < f + 1$ with $f = 3$ and $k = 1$; the curves relate to different intrusion probabilities p_{A}.

The Global Level (4/6): Reliability Curve



Reliability curves for varying number of nodes N; The reliability function is defined as $R(t) \triangleq \mathbb{P}[T^{(f)} > t]$ where $T^{(f)}$ is a random variable representing the time when $N_t < f + 1$ with $f = 3$.

The Global Level (5/6): Controlling the Replication Factor

Goal: **maximize the average service availability** $T^{(\mathrm{A})}$ and minimize the number of nodes. We model these two goals with the following constrained objective

$$\begin{aligned}
\text{minimize} \quad & J \triangleq \lim_{T\to\infty}\left[\frac{1}{T}\sum_{t=1}^{T} s_t\right] \qquad (4)\\
\text{subject to} \quad & T^{(\mathrm{A})} \geq \epsilon_{\mathrm{A}} \\
& \implies \lim_{T\to\infty}\left[\frac{1}{T}\sum_{t=1}^{T}[\![N_t \geq 2f+1]\!]\right] \geq \epsilon_{\mathrm{A}} \\
& \implies \lim_{T\to\infty}\left[\frac{1}{T}\sum_{t=1}^{T}[\![s_t \geq f+1]\!]\right] \geq \epsilon_{\mathrm{A}}
\end{aligned}$$

where ϵ_{A} is the minimum allowed average service availability with respect to the tolerance threshold f.

The Global Level (6/6): Controlling the Replication Factor

Problem (Optimal Control of the Replication Factor)

$$
\begin{aligned}
\underset{\pi \in \Pi_{\mathrm{S}}}{\text{minimize}} \quad & \mathbb{E}_{\pi}\left[J \mid S_1 = N\right] && && \text{(5a)} \\
\text{subject to} \quad & \mathbb{E}_{\pi}\left[T^{(\mathrm{A})}\right] \geq \epsilon_{\mathrm{A}} && \forall t && \text{(5b)} \\
& s_{t+1} = f_{\mathrm{S}}(s_t, a_t, \delta_t) \in \mathcal{S}_{\mathrm{S}} && \forall t && \text{(5c)} \\
& \delta_t \sim p_{\Delta}(s_t) && \forall t && \text{(5d)} \\
& a_{t+1} \sim \pi_t(s_t) \in \mathcal{A}_{\mathrm{S}} && \forall t && \text{(5e)}
\end{aligned}
$$

Structure of an Optimal Scaling Strategy

Theorem

If the following holds

$$\exists \pi \in \Pi_{\mathrm{S}} \text{ such that } \mathbb{E}_{\pi}\left[T^{(\mathrm{A})}\right] \geq \epsilon_{\mathrm{A}} \tag{A}$$

$$f_{\mathrm{S}}(s' \mid s, a) > 0 \qquad \forall s', s, a \tag{B}$$

$$\sum_{s'=s}^{s_{\max}} f_{\mathrm{S}}(s' \mid \hat{s}+1, a) \geq \sum_{s'=s}^{s_{\max}} f_{\mathrm{S}}(s' \mid \hat{s}, a) \qquad \forall s, \hat{s}, a \tag{C}$$

then there exists two strategies π_{λ_1} *and* π_{λ_1} *that satisfy*

$$\pi_{\lambda_1}(s_t) = 1 \iff s_t \leq \beta_1 \qquad \pi_{\lambda_2}(s_t) = 1 \iff s_t \leq \beta_2 \quad \forall t \tag{6}$$

and an optimal randomized threshold strategy $\pi^{\star}$ *that satisfies*

$$\pi^{\star}(s_t) = \kappa \pi_{\lambda_1}(s_t) + (1-\kappa)\pi_{\lambda_2}(s_t) \qquad \forall t \tag{7}$$

for some probability $\kappa \in [0, 1]$.

Efficient Algorithms for Computing the Optimal Control Strategies

- The problem of computing an optimal scaling strategy has polynomial time-complexity. This follows because the problem can be formulated as a linear program of polynomial size.

- The problem of computing the optimal intrusion recovery strategies is in the complexity class PSPACE-HARD, and thus no efficient (polynomial-time) algorithm for solving this problem is known. (Note that $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$.)

- **To manage the high computational complexity of computing the optimal recovery strategies we leverage Theorem 1 and Corollary 1.**

Algorithm for The Local Control Problem

Algorithm 1: Recovery Threshold Optimization (RTO)

Input: $\eta, p_{\mathrm{A}}, p_{\mathrm{C}_1}, p_{\mathrm{C}_2}, p_{\mathrm{U}}, Z, \Delta_{\mathrm{R}}$
Parametric optimization algorithm: PO

Output: A near-optimal local control strategy $\hat{\pi}_{\theta,t}$

Algorithm

$d \leftarrow 1 - \Delta_{\mathrm{R}}$ if $\Delta_{\mathrm{R}} < \infty$ else $d \leftarrow 1$

$\Theta \leftarrow [0,1]^d$

For each $\theta \in \Theta$, define $\pi_{i,\theta}(b_t)$ as

$$\pi_{i,\theta}(b_t) \triangleq \begin{cases} \mathfrak{R} & \text{if } b_t \geq \theta_i \text{ where } i = \max[t, d] \\ \mathfrak{W} & \text{otherwise} \end{cases}$$

$J_\theta \leftarrow \mathbb{E}_{\pi_{i,\theta}}[J_i]$

$\hat{\pi}_{\theta,t} \leftarrow \mathrm{PO}(\Theta, J_\theta)$

return $\hat{\pi}_{\theta,t}$

Algorithm for The Global Control Problem

Algorithm 2: Linear Program for Scaling (LP-R)

Input: $s_{\max}, \epsilon_{\mathrm{A}}, N, f, p_{\Delta}$

Linear programming solver: $\mathrm{LPSolver}$

Output: An optimal global control strategy $\pi^{\star}$

Algorithm

Solve the following linear program with $\mathrm{LPSolver}$

$$\underset{\rho}{\text{minimize}} \sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} s\rho(s,a) \tag{8a}$$

$$\text{subject to} \tag{8b}$$

$$\rho(s,a) \geq 0 \qquad \forall s \in \mathcal{S}_{\mathrm{S}}, a \in \mathcal{A}_{\mathrm{S}} \tag{8c}$$

$$\sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) = 1 \tag{8d}$$

$$\sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) = \sum_{s' \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s',a) f_{\mathrm{S}}(s'|s,a) \; \forall s \in \mathcal{S}_{\mathrm{S}} \tag{8e}$$

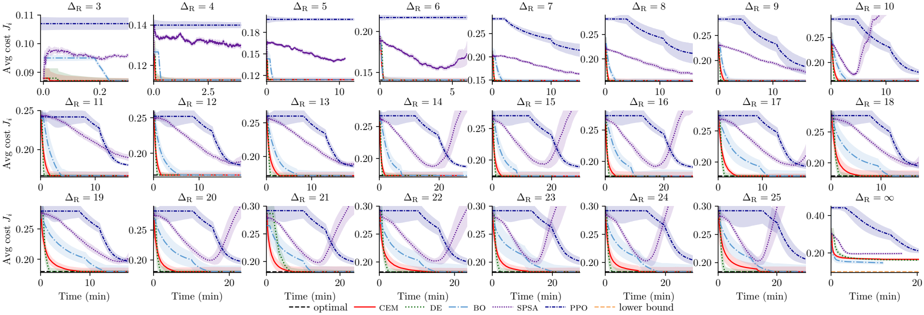
$$\sum_{s \in \mathcal{S}_{\mathrm{S}}} \sum_{a \in \mathcal{A}_{\mathrm{S}}} \rho(s,a) [\![s_t \geq f+1]\!] \geq \epsilon_{\mathrm{A}} \tag{8f}$$

Let $\rho^{\star}$ denote the solution to the above program and define $\pi^{\star}$ as

$$\pi^{\star}(a|s) \triangleq \frac{\rho^{\star}(s,a)}{\sum_{s \in \mathcal{S}_{\mathrm{S}}} \rho^{\star}(s,a)} \qquad \forall s \in \mathcal{S}_{\mathrm{S}}, a \in \mathcal{A}_{\mathrm{S}}$$

return $\pi^{\star}$

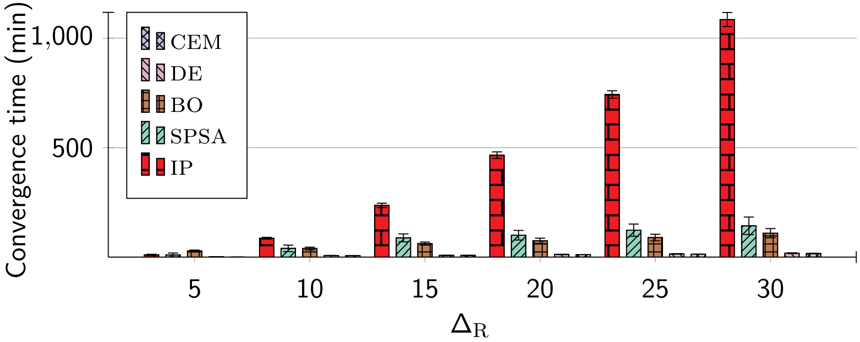
Evaluation of the RTO Algorithm (1/2)



mean values from evaluations with 20 different random seeds; $\pm$ indicate the 95% confidence interval based on the Student's t-distribution.

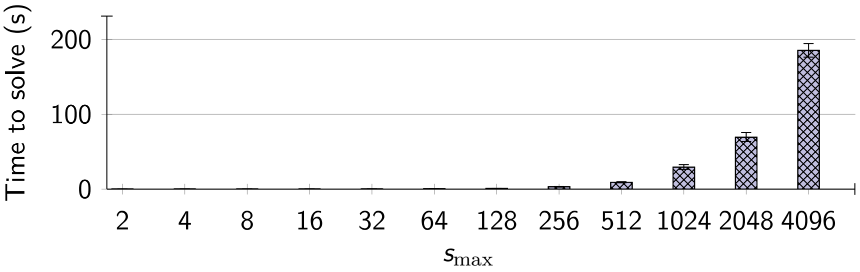
Method	$\Delta_R = 5$		$\Delta_R = 15$		$\Delta_R = 25$		$\Delta_R = \infty$	
	Time (min)	J_i	Time (min)	J_i	Time (min)	J_i	Time (min)	J_i
CEM	**1.04**	$\mathbf{0.12 \pm 0.01}$	**8.84**	$\mathbf{0.17 \pm 0.06}$	**14.48**	0.19 ± 0.08	11.81	0.16 ± 0.01
DE	2.35	$\mathbf{0.12 \pm 0.03}$	8.98	$\mathbf{0.17 \pm 0.01}$	15.45	$\mathbf{0.18 \pm 0.02}$	22.68	0.16 ± 0.01
SPSA	10.78	0.18 ± 0.01	88.35	0.58 ± 0.40	123.85	0.77 ± 0.48	**4.20**	0.20 ± 0.02
BO	29.18	$\mathbf{0.12 \pm 0.02}$	62.57	$\mathbf{0.17 \pm 0.05}$	90.26	$\mathbf{0.18 \pm 0.12}$	9.07	$\mathbf{0.15 \pm 0.06}$
PPO	28.20	0.18 ± 0.01	30.01	0.19 ± 0.02	30.33	0.21 ± 0.07	28.95	$0.21 + \pm 0.09$
IP	11.11	**0.12**	237.06	**0.17**	743.73	**0.18**	> 10000	not converged

Evaluation of the RTO Algorithm (2/2)



Time required to compute optimal intrusion recovery strategies; the x-axis indicate different values of Δ_{R}; the error bars indicate the 95% confidence interval based on the Student's t-distribution with 20 measurements.

Evaluation of the LP-R Algorithm

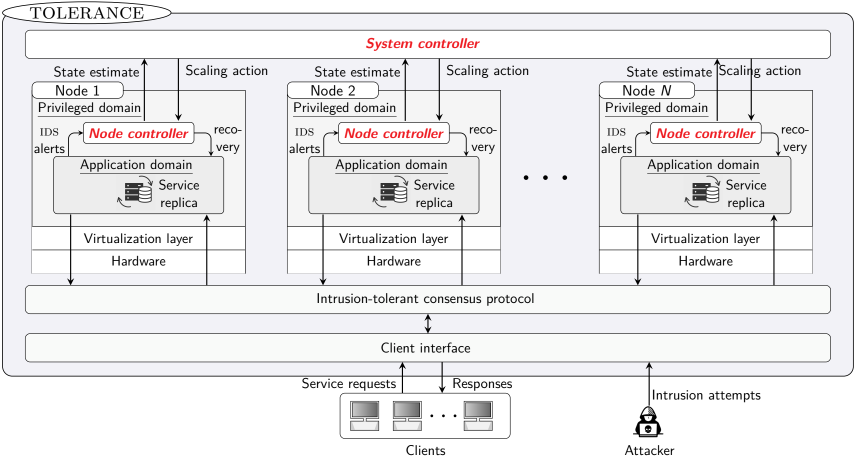


Time required to compute optimal scaling strategies; the x-axis indicate different values of $s_{\max}$; the error bars indicate the 95% confidence interval based on the Student's t-distribution with 20 measurements.

Outline

- **Use Case & Research Problem**
  - Use case: intrusion tolerance
  - Goal: optimal control strategies for intrusion-tolerant systems
- **Background**
  - Fault tolerance and intrusion tolerance
  - State machine replication
- **Our Contributions**
  - The TOLERANCE control architecture
  - Constrained two-level control problem
  - Theoretical results
  - Computational algorithms
- **Comparison with State-of-the-art**
  - Implementation and evaluation
- **Conclusions**

We Instantiate The TOLERANCE Control Architecture with The Computed Control Strategies



Experiment Setup - Physical Servers

Server	*Processors*	RAM (GB)
1, R715 2U	2 12-core AMD OPTERON	64
2, R715 2U	2 12-core AMD OPTERON	64
3, R715 2U	2 12-core AMD OPTERON	64
4, R715 2U	2 12-core AMD OPTERON	64
5, R715 2U	2 12-core AMD OPTERON	64
6, R715 2U	2 12-core AMD OPTERON	64
7, R715 2U	2 12-core AMD OPTERON	64
8, R715 2U	2 12-core AMD OPTERON	64
9, R715 2U	2 12-core AMD OPTERON	64
10, R630 2U	2 12-core INTEL XEON E5-2680	256
11, R740 2U	1 20-core INTEL XEON GOLD5218R	32
12, SUPERMICRO 7049	2 TESLA P100, 1 16-core INTEL XEON	126
13, SUPERMICRO 7049	4 RTX 8000, 1 24-core INTEL XEON	768

Table 1: Specifications of the physical servers.

Experiment Setup - Replica Configurations

Replica ID	*Operating system*	*Vulnerabilities*
1	UBUNTU 14	FTP weak password
2	UBUNTU 20	SSH weak password
3	UBUNTU 20	TELNET weak password
4	DEBIAN 10.2	CVE-2017-7494
5	UBUNTU 20	CVE-2014-6271
6	DEBIAN 10.2	CVE-89 on CVE
7	DEBIAN 10.2	CVE-2015-3306
8	DEBIAN 10.2	CVE-2016-10033
9	DEBIAN 10.2	CVE-2010-0426, SSH weak password
10	DEBIAN 10.2	CVE-2015-5602, SSH weak password

Table 2: Replica configurations.

Experiment Setup - Emulated Intrusions

Replica ID	*Intrusion steps*
1	TCP SYN scan, FTP brute force
2	TCP SYN scan, SSH brute force
3	TCP SYN scan, TELNET brute force
4	ICMP scan, exploit of CVE-2017-7494
5	ICMP scan, exploit of CVE-2014-6271
6	ICMP scan, exploit of CVE-89 on on CVE
7	ICMP scan, exploit of CVE-2015-3306
8	ICMP scan, exploit of CVE-2016-10033
9	ICMP scan, SSH brute force, exploit of CVE-2010-0426
10	ICMP scan, SSH brute force, exploit of CVE-2015-5602

Table 3: Intrusion steps

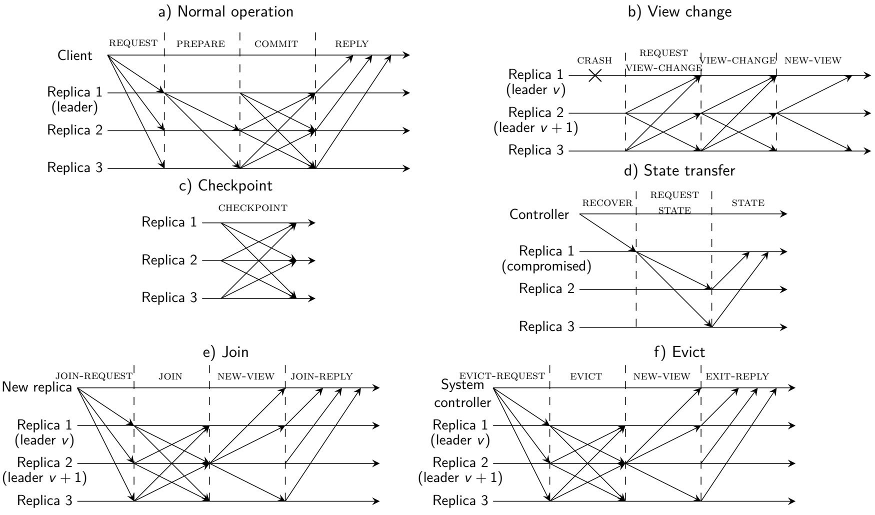
Experiment Setup - Background Traffic

Background services	*Replica ID(s)*
FTP, SSH, MONGODB, HTTP, TEAMSPEAK	1
SSH, DNS, HTTP	2
SSH, TELNET, HTTP	3
SSH, SAMBA, NTP	4
SSH	5, 7, 8, 10
CVE, IRC, SSH	6
TEAMSPEAK, HTTP, SSH	9

Table 4: Background services; each background client invokes functions on service replicas.

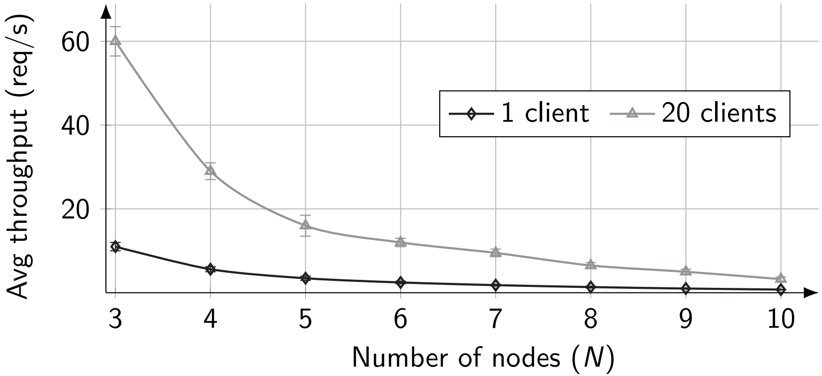
Experiment Setup - Consensus Algorithm

We implement and extend the MINBFT Byzantine fault-tolerant consensus algorithm to be reconfigurable.

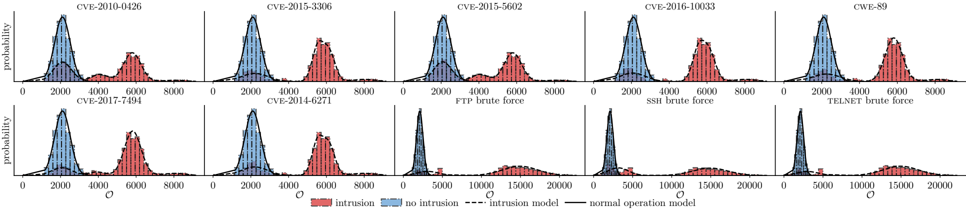


Experiment Setup - Consensus Algorithm

Throughput of our implementation of MINBFT.



System Identification



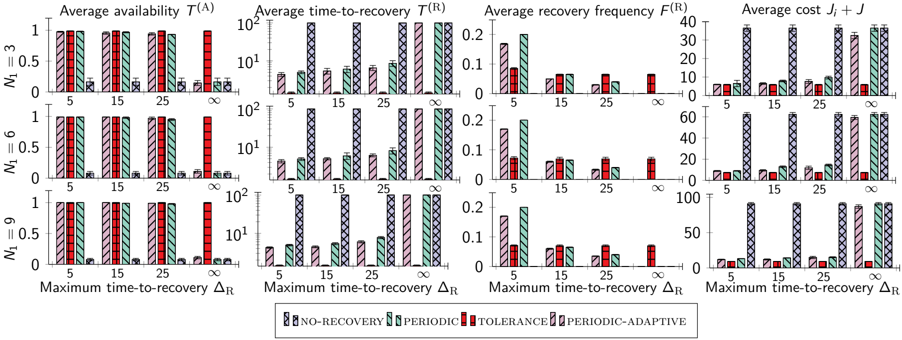
Empirical observation distributions $\widehat{Z}_1(\cdot \mid s), \ldots, \widehat{Z}_{10}(\cdot \mid s)$ as estimates of $Z_1, \ldots, Z_{10}$.

- Empirical distributions based on $M = 25,000$ samples.
- From the Glivenko-Cantelli theorem we know that $\widehat{Z} \rightarrow^{\text{a.s}} Z$ as $M \rightarrow \infty$.
- Bound:

$$\mathbb{P}\left[D_{\mathrm{KL}}(\widehat{Z}(\cdot \mid s) \parallel Z(\cdot \mid s)) \geq \epsilon\right] \leq 2^{-M\left(\epsilon - |\mathcal{O}|\frac{\ln(M+1)}{M}\right)}$$

$$= 2^{-25\cdot 10^3\left(\epsilon - 2\cdot 10^3 \frac{\ln(25\cdot 10^3+1)}{25\cdot 10^3}\right)} = 2^{-5\cdot 10^3\left(5\epsilon - 4\ln(25\cdot 10^3+1)\right)}$$

Comparison with State-of-the-art Intrusion-Tolerant Systems



Comparison between TOLERANCE and the baselines; the columns represent: average availability ($T^{(\mathrm{A})}$), average time-to-recovery ($T^{(\mathrm{R})}$); average recovery frequency ($F^{(\mathrm{R})}$); and average cost $J_i + J$; the bars indicate the mean value from evaluations with 20 different random seeds; the error bars indicate the 95% confidence interval based on the Student's t-distribution.

Conclusions

- We present TOLERANCE: a **novel control architecture** for intrusion-tolerant systems which improves state-of-the-art.
- We prove that the optimal control strategies have **threshold structures** and design **efficient algorithms** for computing them.
- We evaluate TOLERANCE in an emulation environment against 10 different types of network intrusions.