Towards Ultra-Reliable CPS: Reliability Analysis of Distributed Real-Time Systems

Arpan Gujarati

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

Cyber-Physical Systems (CPS)

*Integration of **computation**, **networking**,*
*and **physical processes****

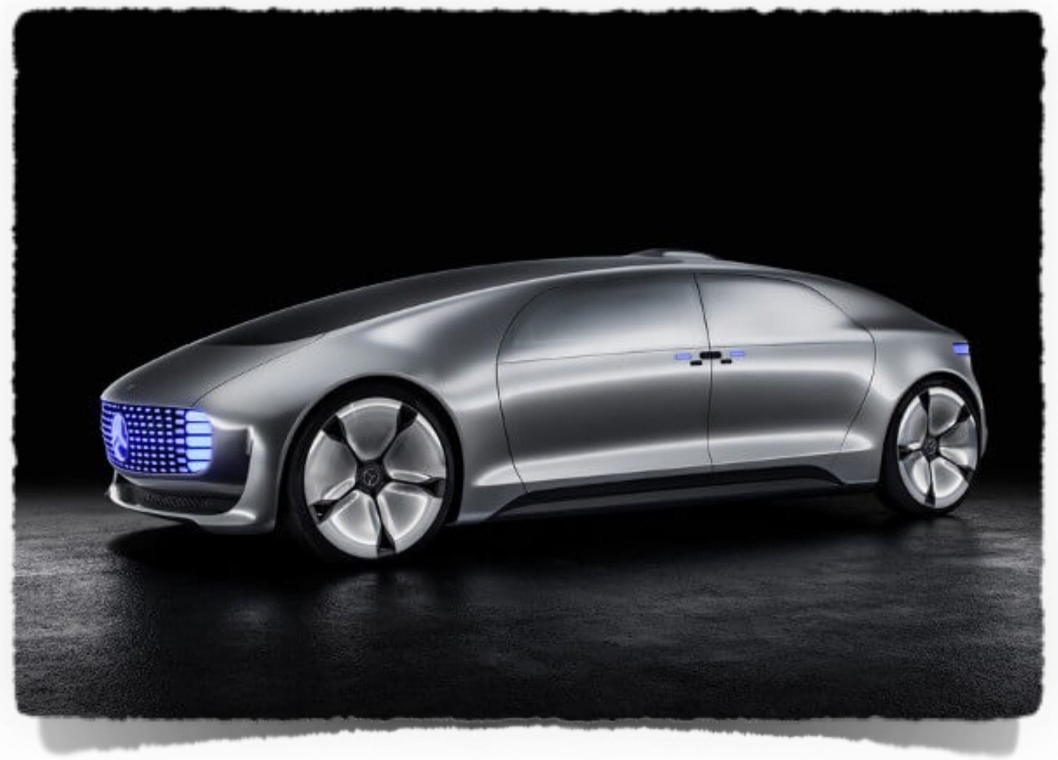
- *Computers control physical processes with **feedback loops***
- *Physical processes affect computations, and vice versa*

* Lee. “The past, present, and future of cyber-physical systems: A focus on models.” Sensors 15.3 (2015)

Cyber-Physical Systems (CPS)

*Integration of **computation**, **networking**, and **physical processes****

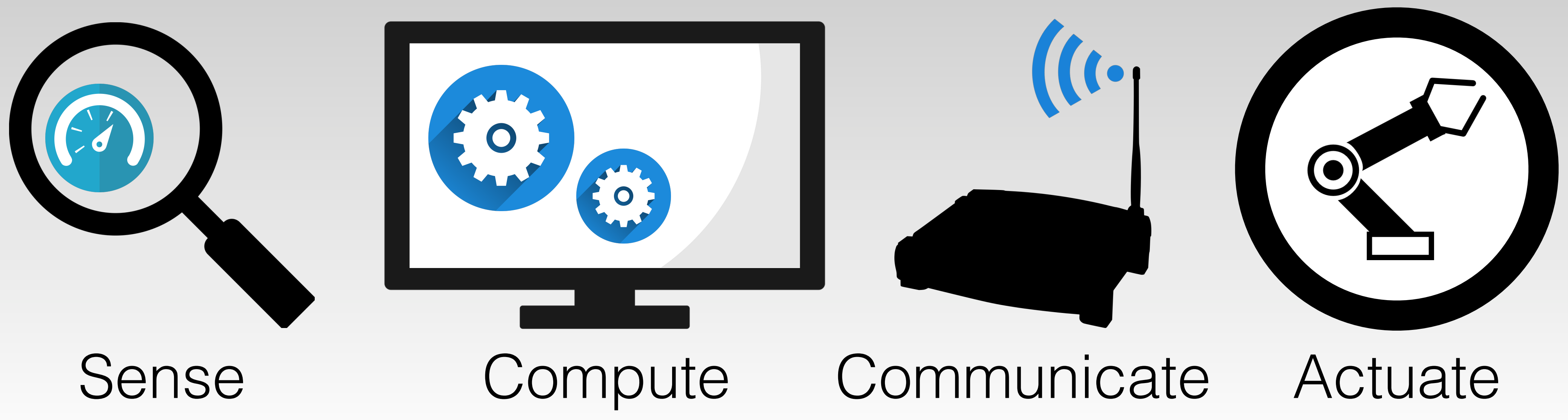
- *Computers control physical processes with **feedback loops***
- *Physical processes affect computations, and vice versa*



* Lee. "The past, present, and future of cyber-physical systems: A focus on models." Sensors 15.3 (2015)

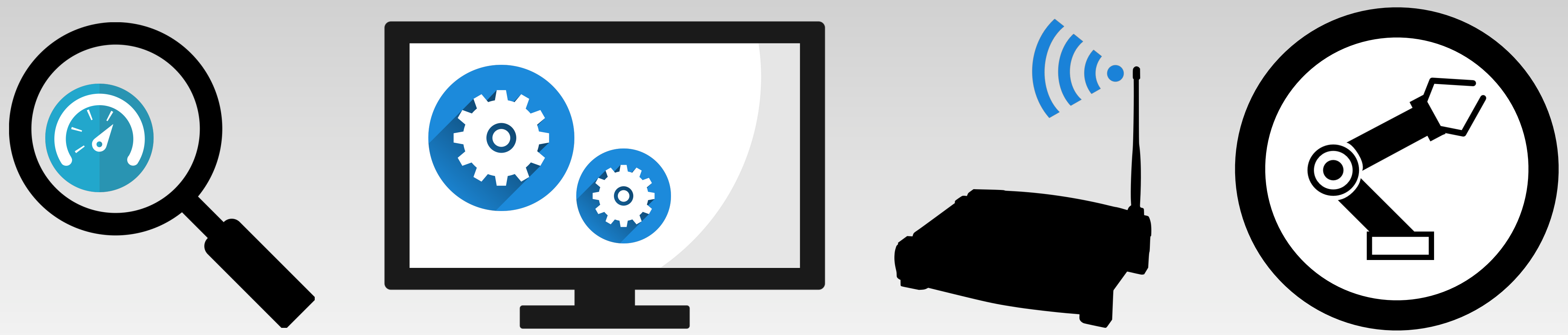
Developing **safety-critical** CPS

Feedback control theory



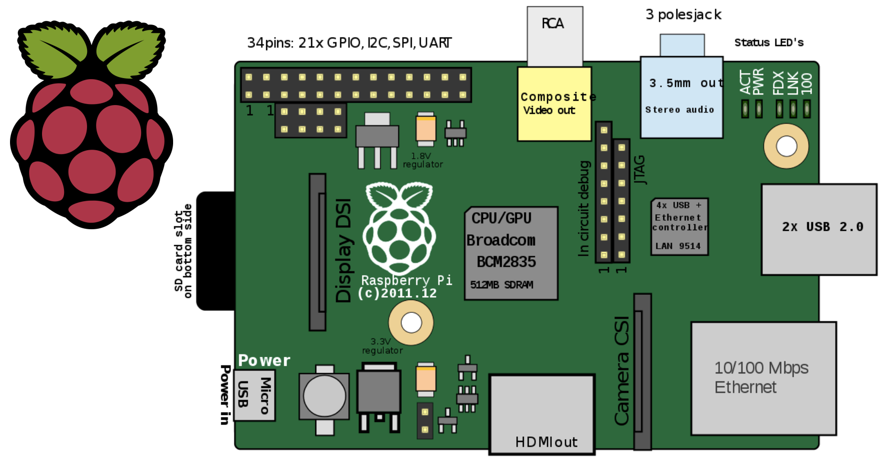
Developing **safety-critical** CPS

Feedback control theory



Sense Compute Communicate Actuate

Embedded System Engineering

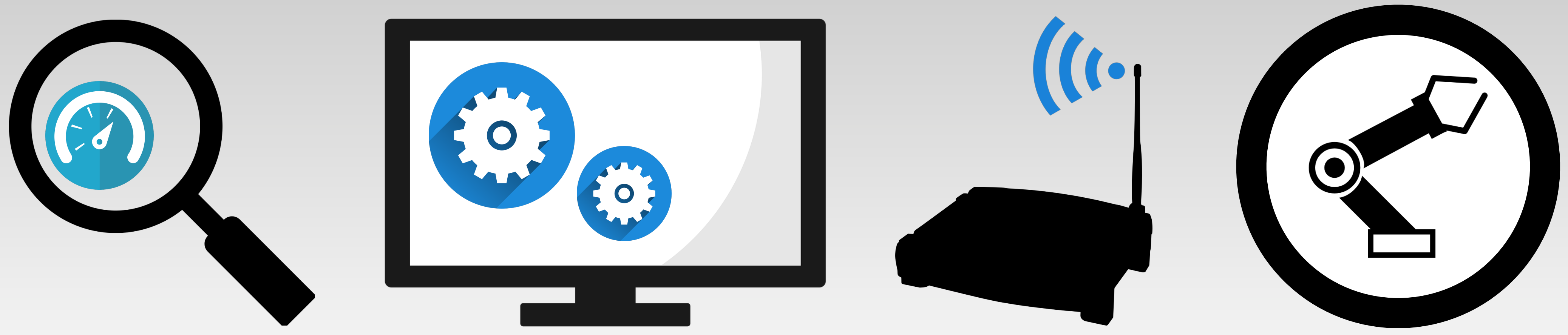


SWaP-C

Size, Weight, and Power ...
plus Cost

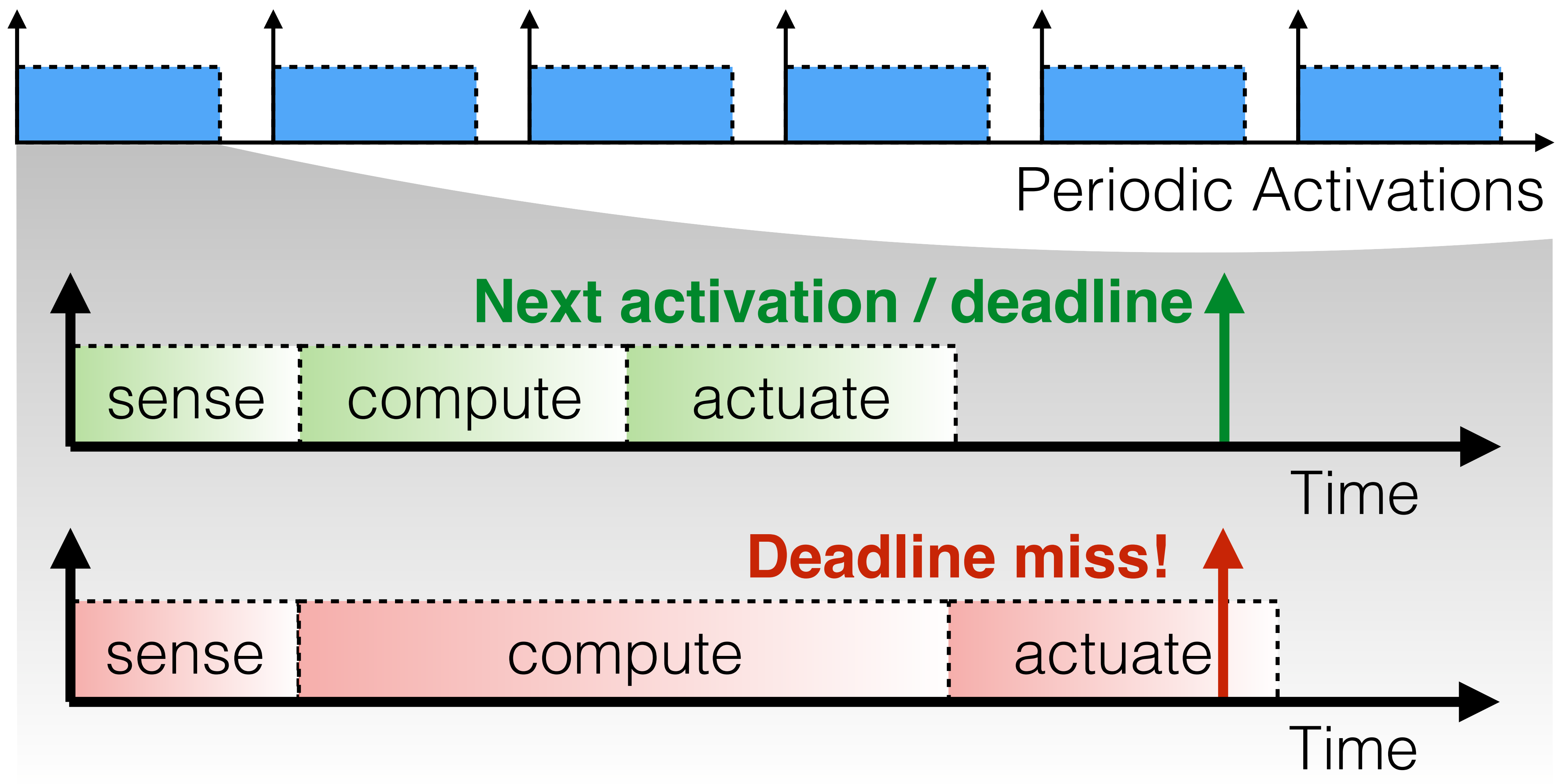
Developing **safety-critical** CPS

Feedback control theory

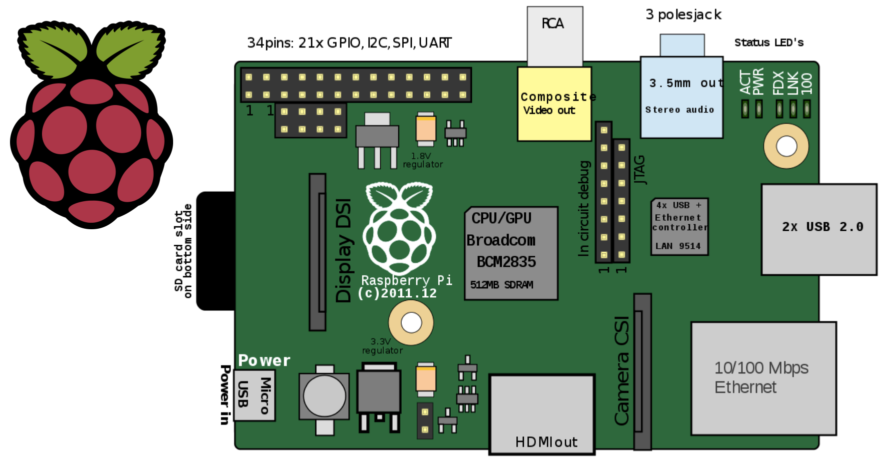


Sense Compute Communicate Actuate

Real-time Requirements



Embedded System Engineering

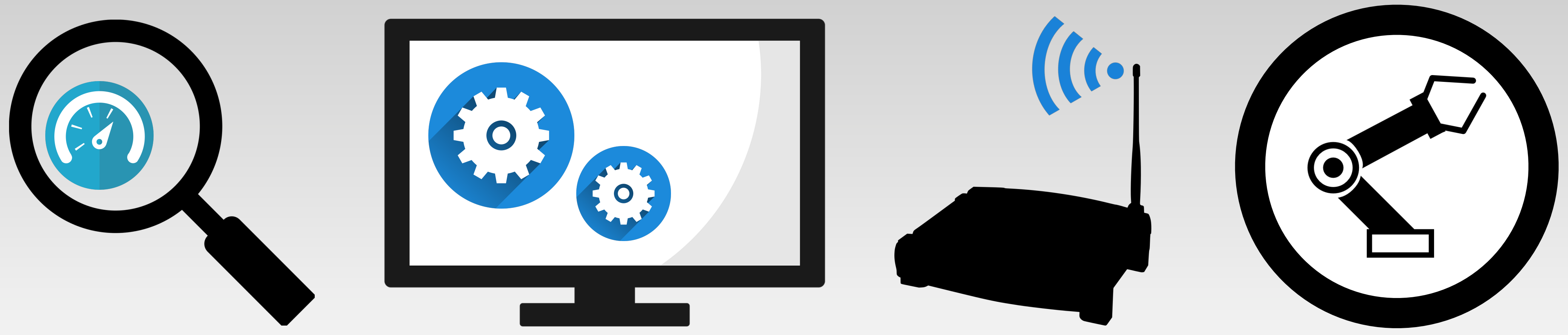


SWaP-C

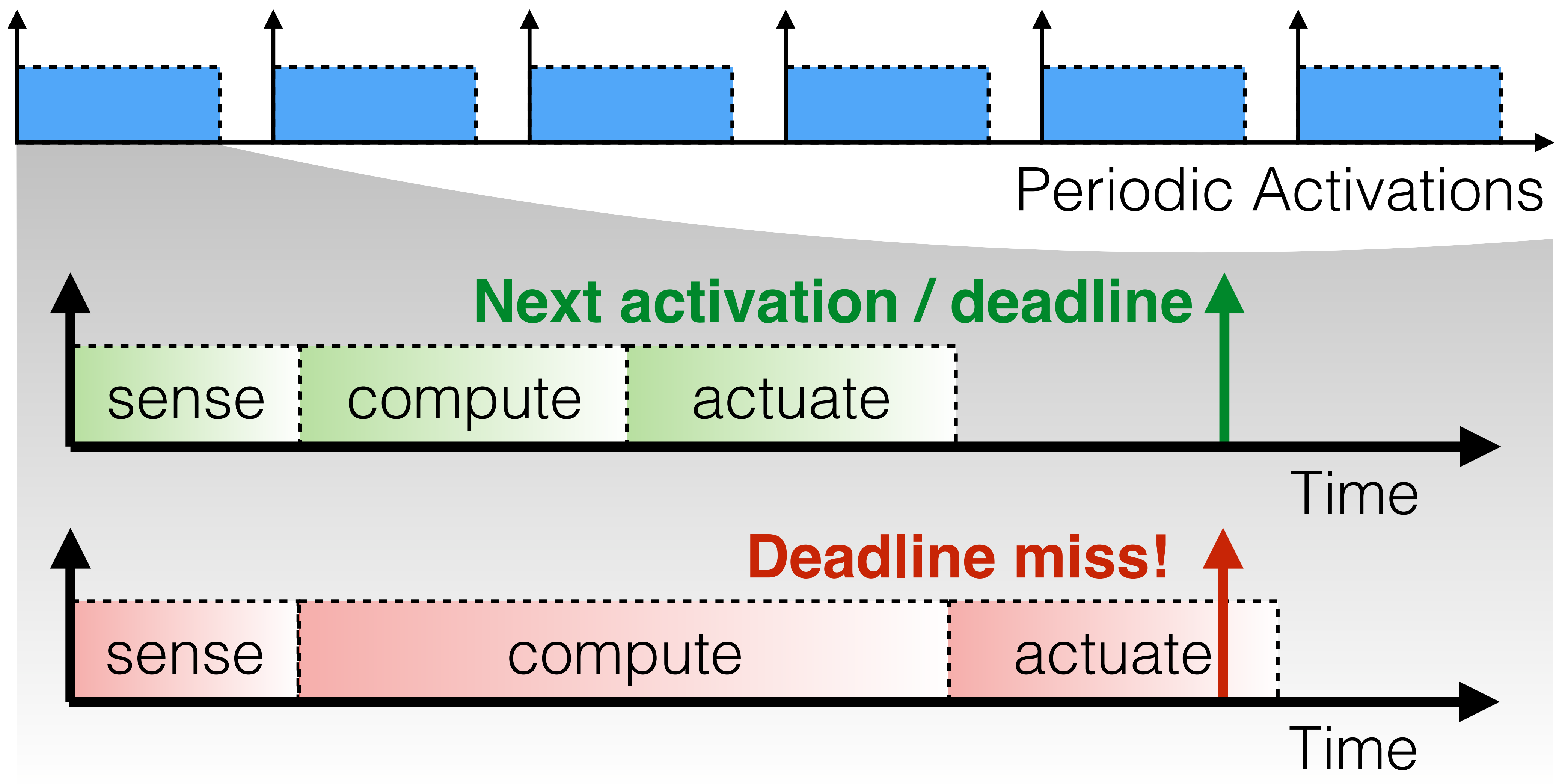
Size, Weight, and Power ...
plus Cost

Developing **safety-critical** CPS

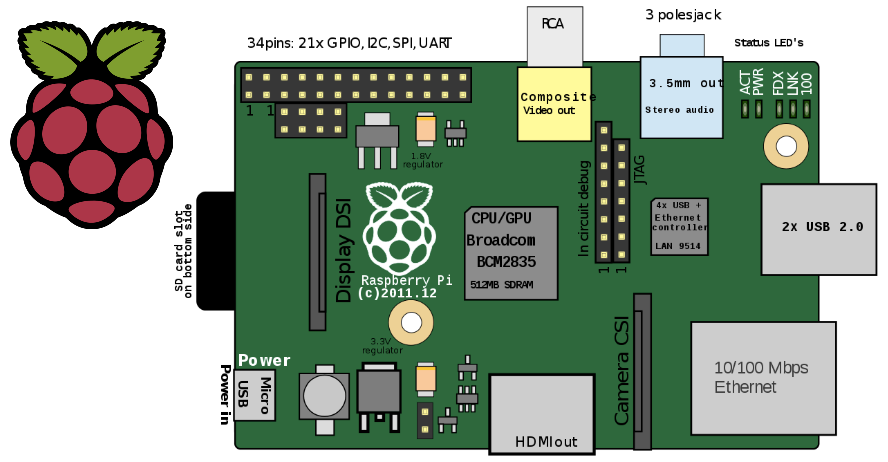
Feedback control theory



Real-time Requirements



Embedded System Engineering

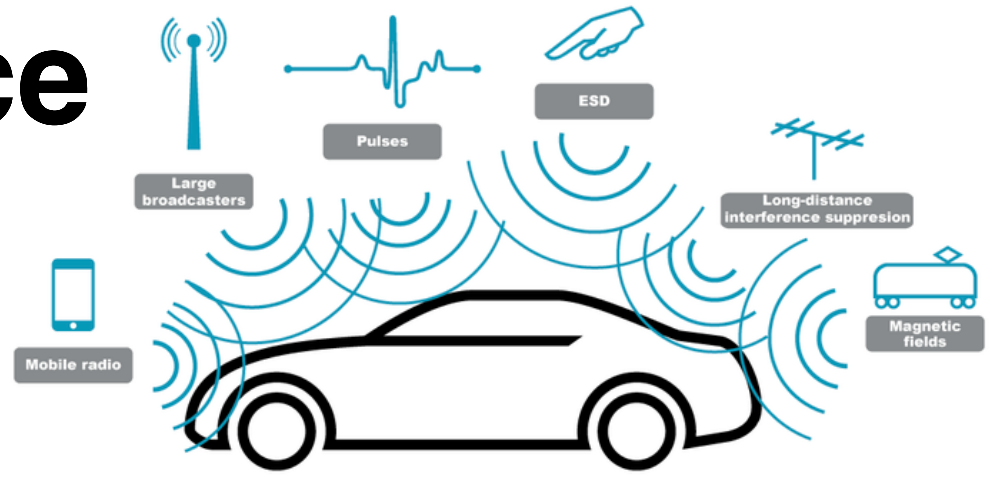


SWaP-C

Size, Weight, and Power ...
plus Cost

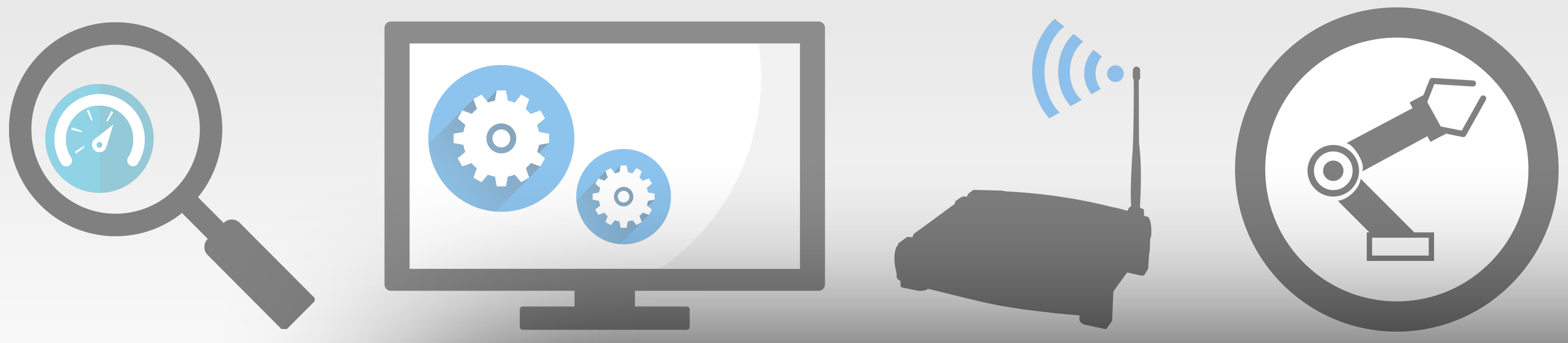
Fault Tolerance

Harsh environment can result in hardware faults

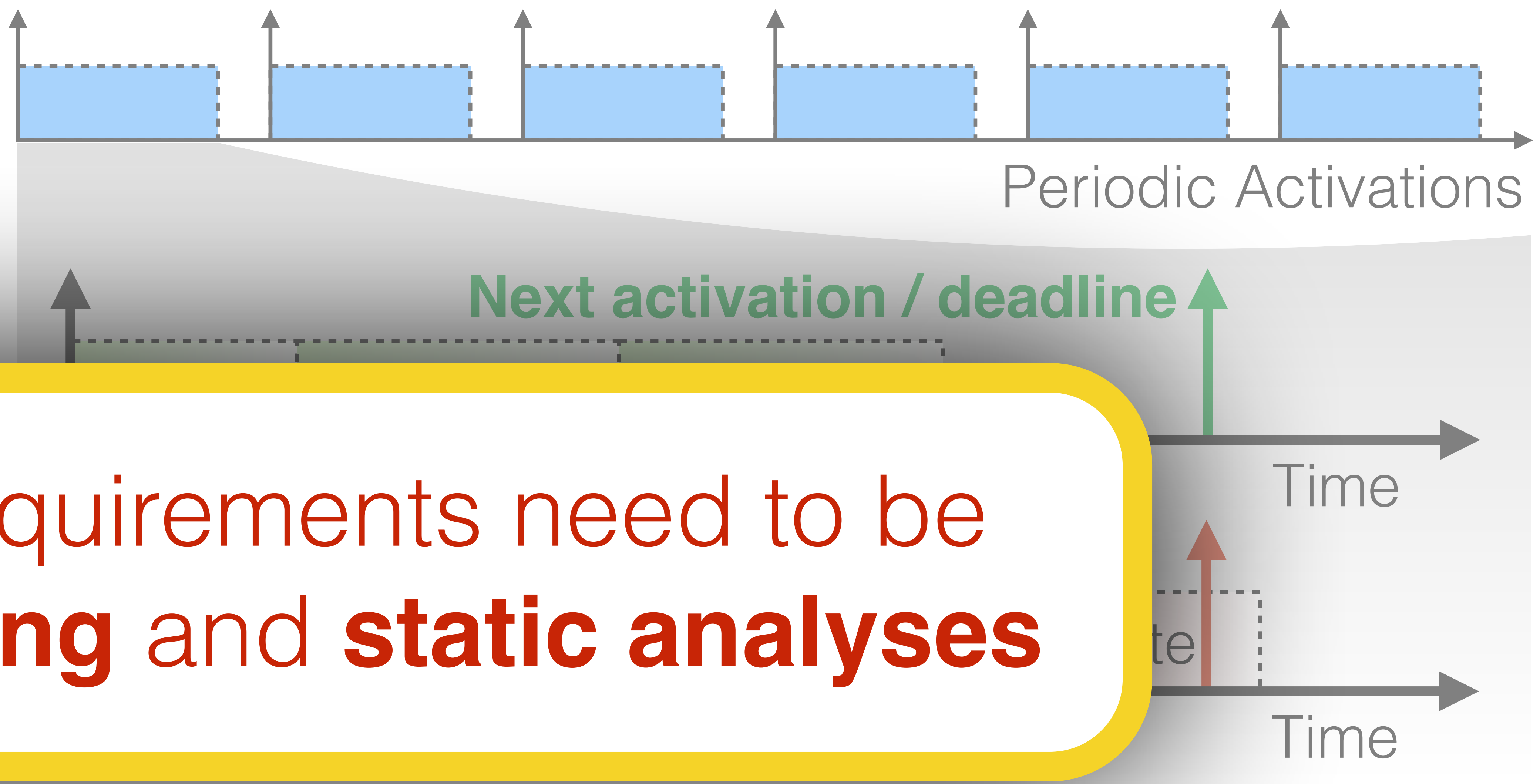


Developing **safety-critical** CPS

Feedback control theory

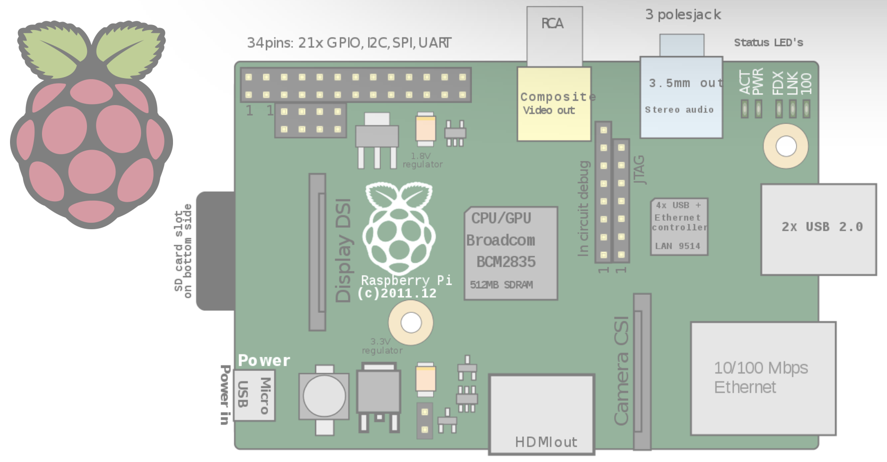


Real-time Requirements



For safety certification, these requirements need to be validated **in advance** using **testing** and **static analyses**

Embedded System Engineering

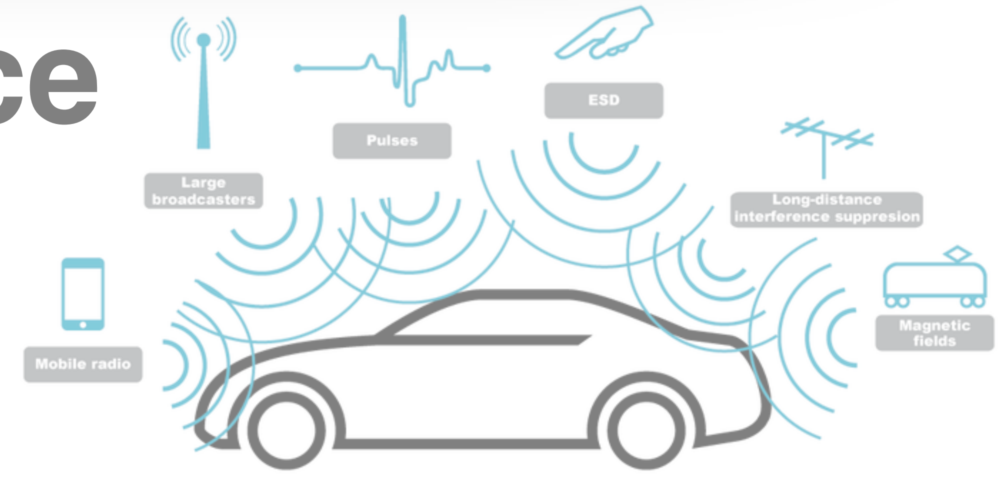


SWaP-C

Size, Weight, and Power ... plus Cost

Fault Tolerance

Harsh environment can result in hardware faults



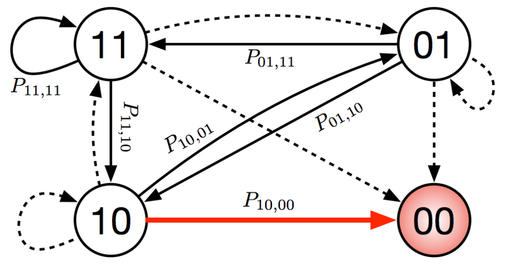
PhD Thesis

Reliability Analysis of Distributed Real-Time Systems

Best Presentation Award

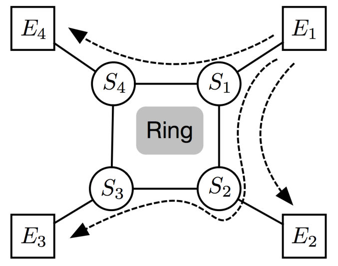
Controller Area Network [ECRTS '18]

Periodic Weakly-Hard Systems [ECRTS '19]



Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]



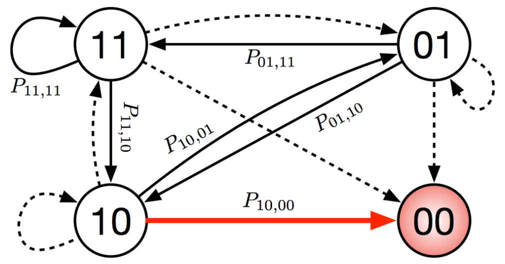
PhD Thesis

Reliability Analysis of Distributed Real-Time Systems

Best Presentation Award

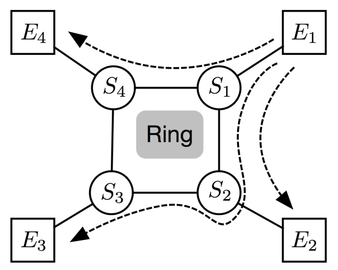
Controller Area Network [ECRTS '18]

Periodic Weakly-Hard Systems [ECRTS '19]



Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]



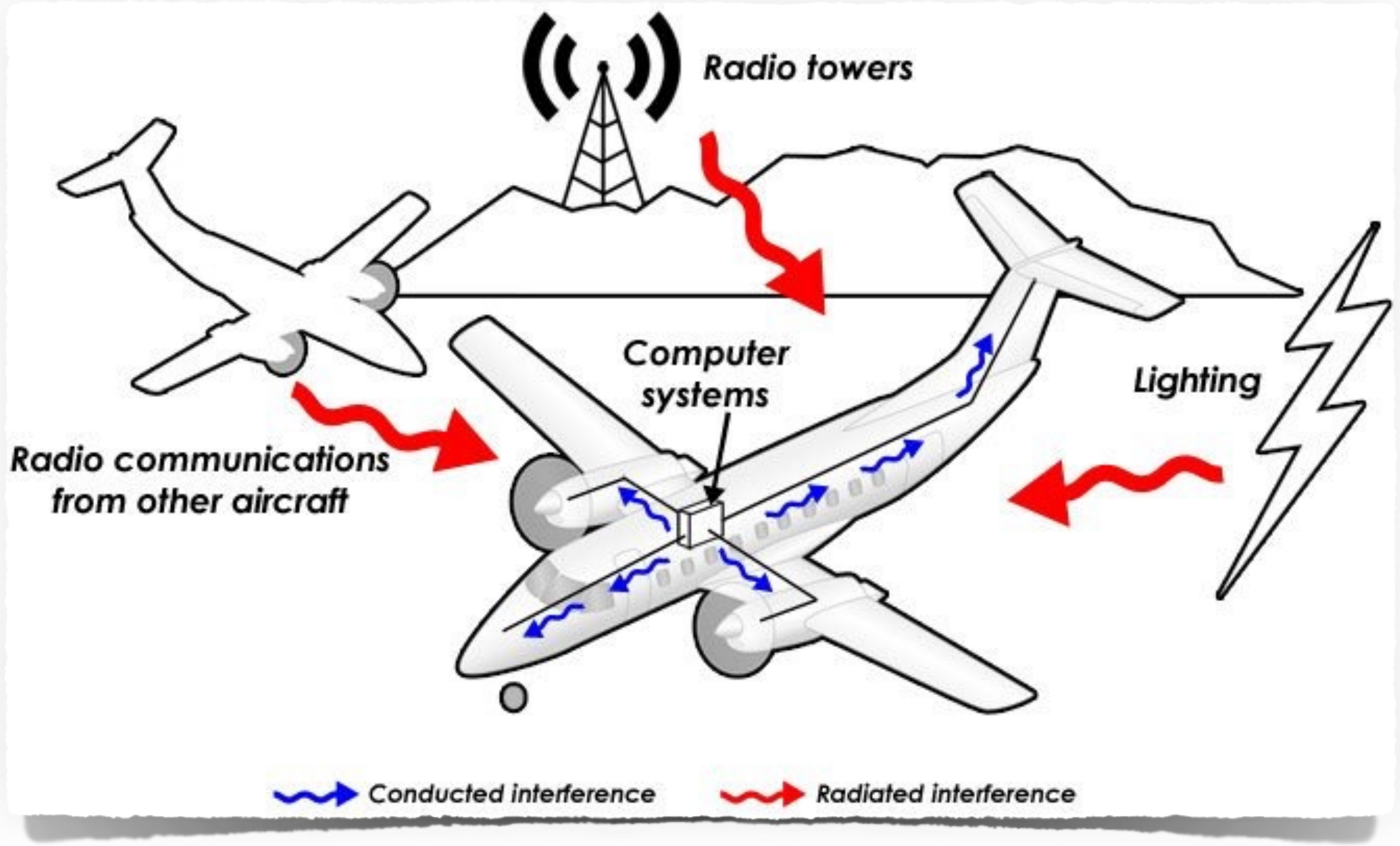
This Talk

This Talk

Why do we need **reliability analyses?**

Environmentally-induced **transient faults**

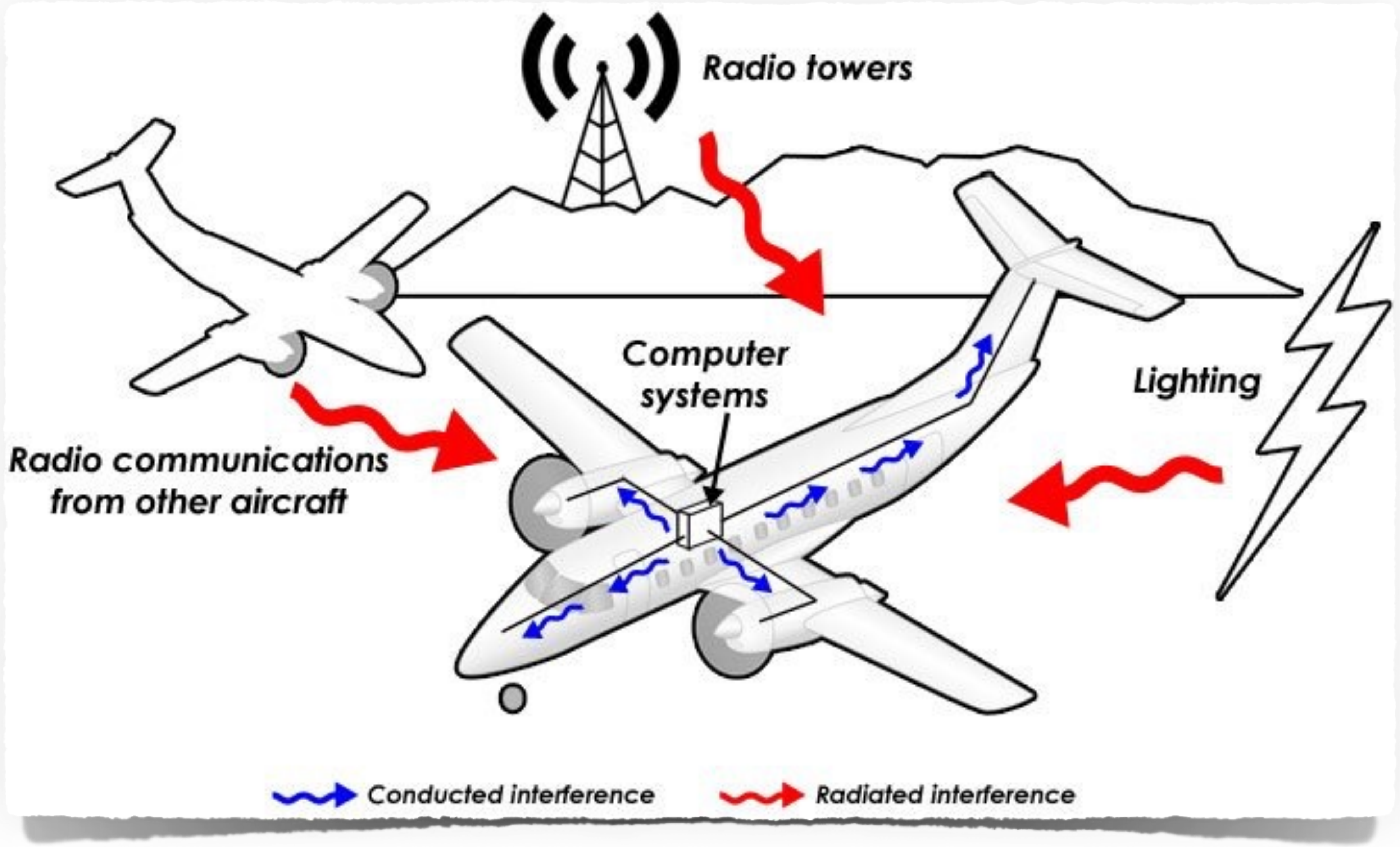
- Harsh environments
  - Robots operating under **hard radiation**
  - Industrial systems near **high power machinery**
  - **Electric motors, spark plugs** inside automobiles



Environmentally-induced **transient faults**

- Harsh environments
  - Robots operating under **hard radiation**
  - Industrial systems near **high power machinery**
  - **Electric motors, spark plugs** inside automobiles

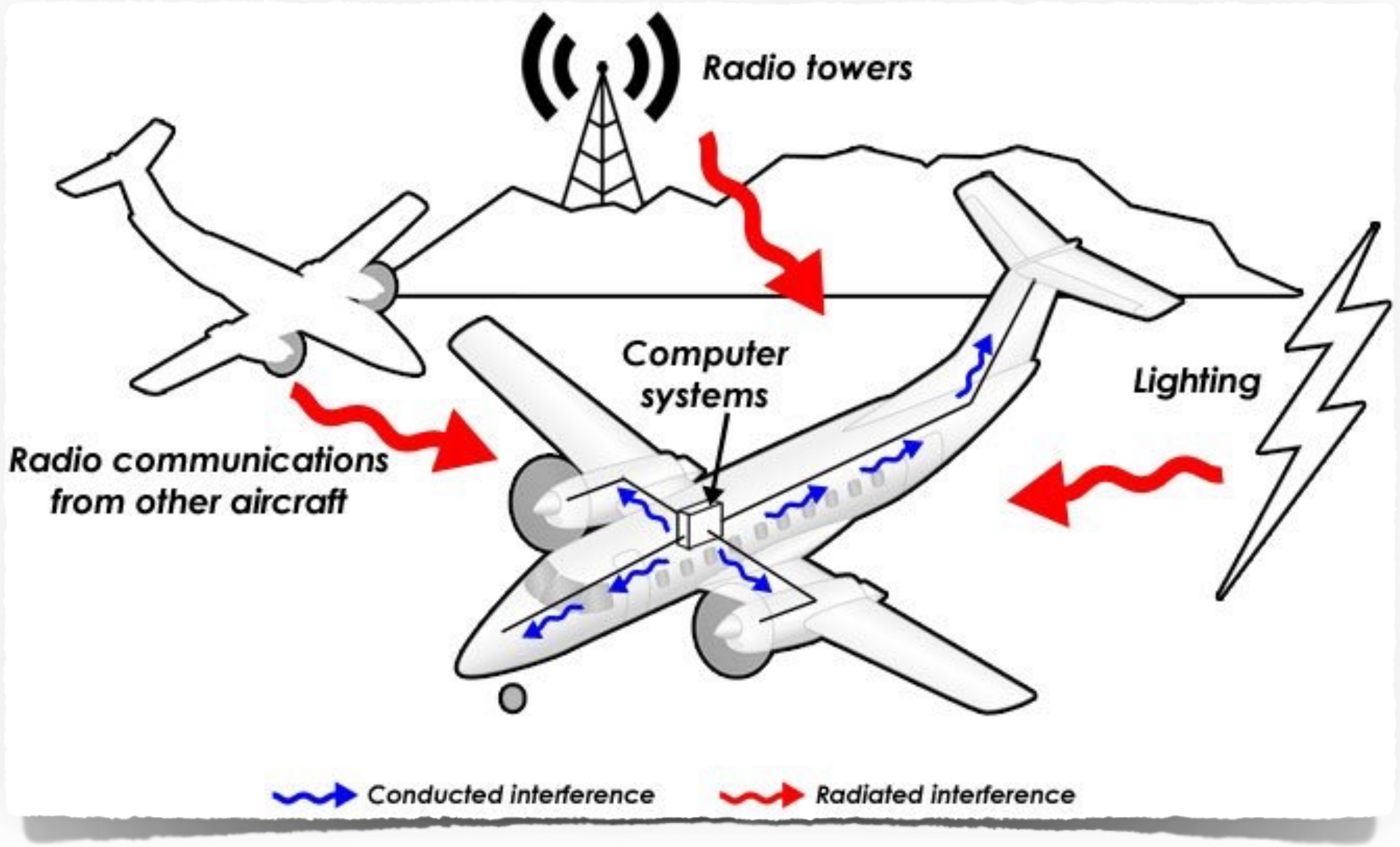
- **Bit-flips** in registers, buffers, networks



Environmentally-induced **transient faults**

- Harsh environments
  - Robots operating under **hard radiation**
  - Industrial systems near **high power machinery**
  - **Electric motors, spark plugs** inside automobiles

- **Bit-flips** in registers, buffers, networks



Example*

- One bit-flip in a 1 MB SRAM every 10^{12} hours of operation
- 0.5 billion cars with an average daily operation time of 5%
- **About 5000 cars are affected by a bit-flip every day**

---

* Mancuso. "Next-generation safety-critical systems on multi-core platforms." PhD thesis, UIUC (2017)

Errors and failures due to transient faults

Errors and failures due to transient faults

- Transmission errors
  - Faults on the network
- Omission errors
  - Fault-induced kernel panics, hangs
- Incorrect computation errors
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

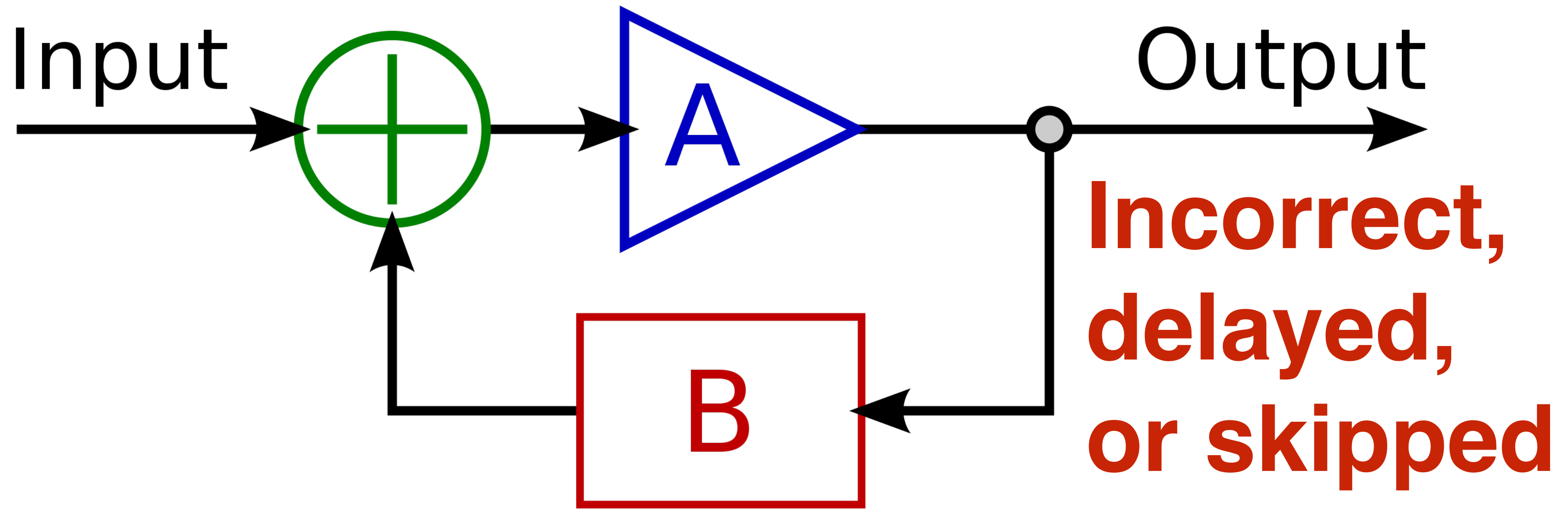
Errors and failures due to transient faults

- Transmission errors
  - Faults on the network
- Omission errors
  - Fault-induced kernel panics, hangs
- Incorrect computation errors
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

Failures in

- **value domain (incorrect output)**
- **time domain (delayed output)**

E.g., safety-critical control system



Errors and failures due to transient faults

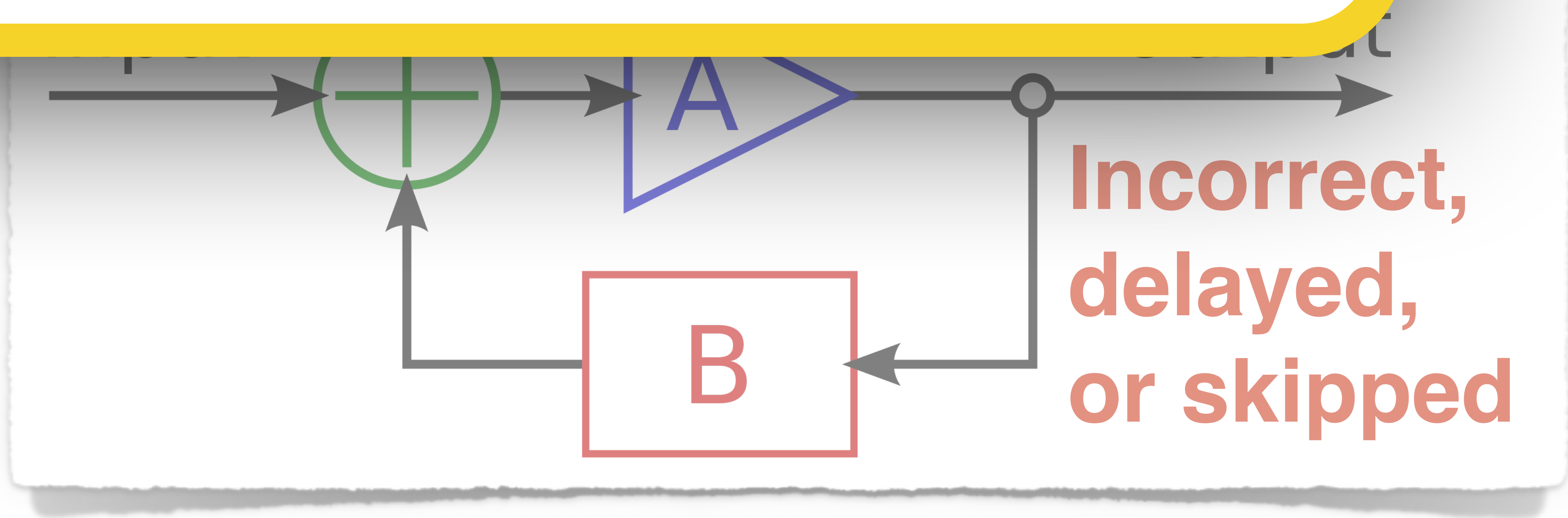
- Transmission errors
  - Faults on the network
- Omissio
  - Fault-inc
- Incorrec
  - Faults in
- Inconsistent broadcast errors
  - Faults in networked systems

Failures in

- **value domain (incorrect output)**
- **time domain (delayed output)**

Fault-induced errors are **random events**

- **Cannot be predicted** in advance
- Must be tolerated at runtime using **fault tolerance mechanisms**



Errors and failures due to transient faults

- Transmission errors
  - Faults on the network

Checksums and retransmissions

- Omission errors
  - Fault-induced kernel panics, hangs

Dual Modular Redundancy (DMR)

- Incorrect computation errors
  - Faults in memory buffers

ECC Memory +
Triple Modular Redundancy (TMR)

- Inconsistent broadcast errors
  - Faults in networked systems

Byzantine Fault Tolerance (BFT)

ansient faults

Which of these mechanisms (or a combination thereof) should be **used** in practice?

- Transmission errors
  - Faults on the network
- Omission errors
  - Fault-induced kernel panics, hangs
- Incorrect computation errors
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Which of these mechanisms (or a combination thereof) should be **used** in practice?

…ansient faults

- Transmission errors
  - Faults on the netw…
- Omission errors
  - Fault-induced kernel panics, hangs
- Incorrect computation errors
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

Industry: RULE

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Which of these mechanisms (or a combination thereof) should be **used** in practice?

ansient faults

- Transmission errors
  - Faults on the netw
- Omission errors
  - Fault-induced kernel pa
- Incorrect computatio
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

Industry: RULE

SWaP-C

Size, Weight, and Power ... plus Cost

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

ansient faults

Which of these mechanisms (or a combination thereof) should be **used** in practice?

- Transmission errors
- Fault-induced kernel pa
- Incorrect computatio
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in networked systems

Real-time requirements

Industry: RULE

SWaP-C

Size, Weight, and Power ... plus Cost

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Which of these mechanisms (or a combination thereof) should be **used** in practice?

...ansient faults

- Transmission errors
- Inconsistent broadcast errors
  - Faults in networked systems

Real-time requirements

Industry: RULE

Safety certification
- Reliability thresholds
- < 10^{-9} failures/hour

SWaP-C
Size, Weight, and Power ... plus Cost

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Which of these mechanisms (or a combination thereof) should be **used** in practice?

Real-time requirements

Industry: RULE

Safety certification

- Reliability thresholds
- < 10^{-9} failures/hour

SWaP-C

Size, Weight, and Power ... plus Cost

Reliability Analyses

Checksums and retransmissions

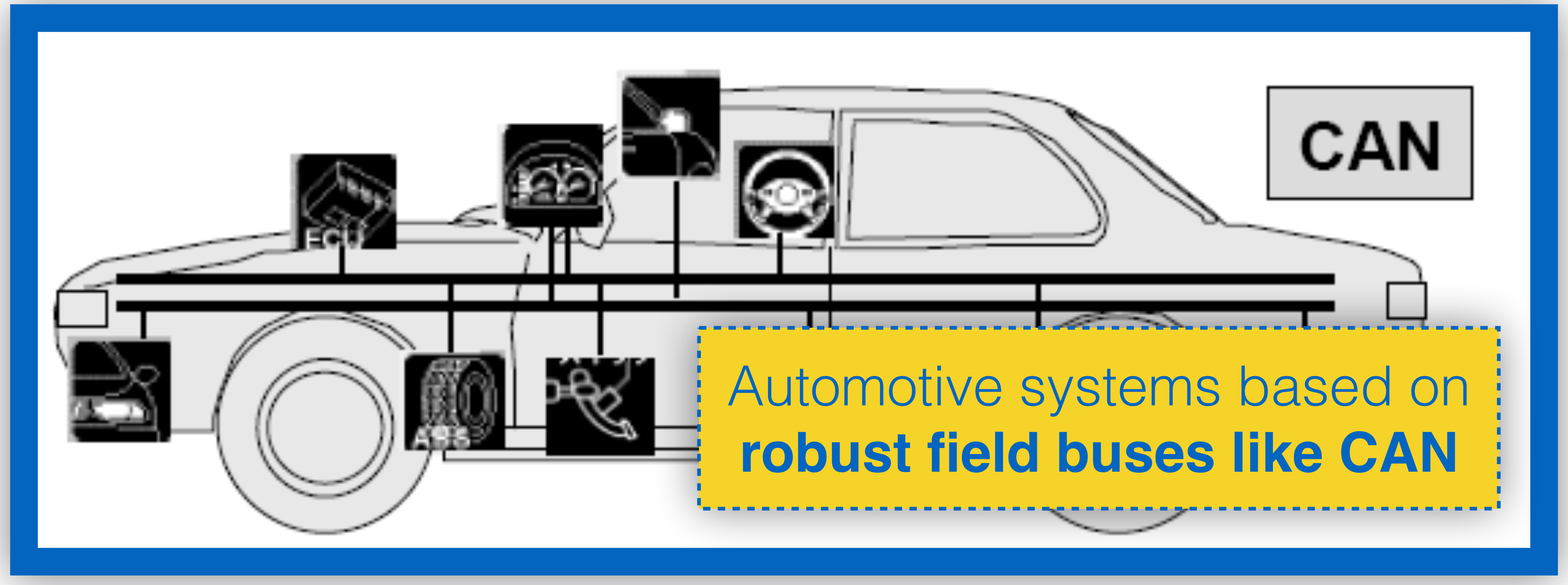
Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Autonomous CPS landscape is **changing!**
New reliability analyses are necessary.

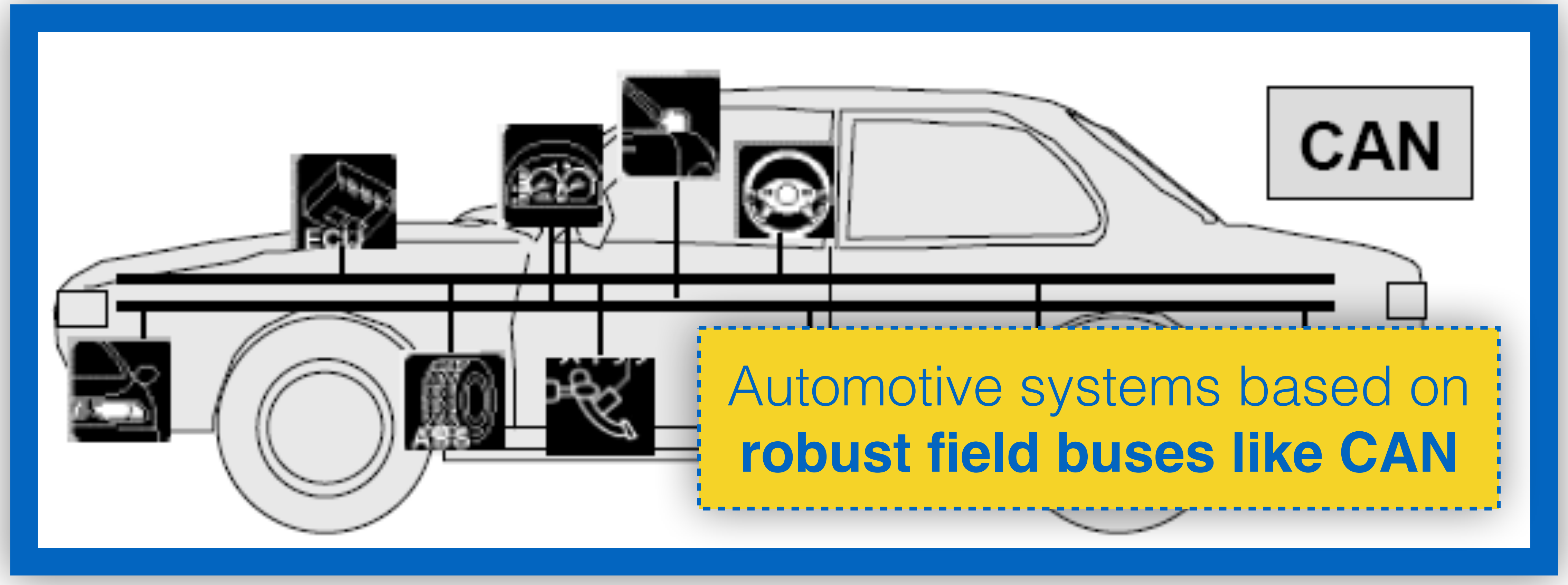
Motivating Trends



Ultra-reliability

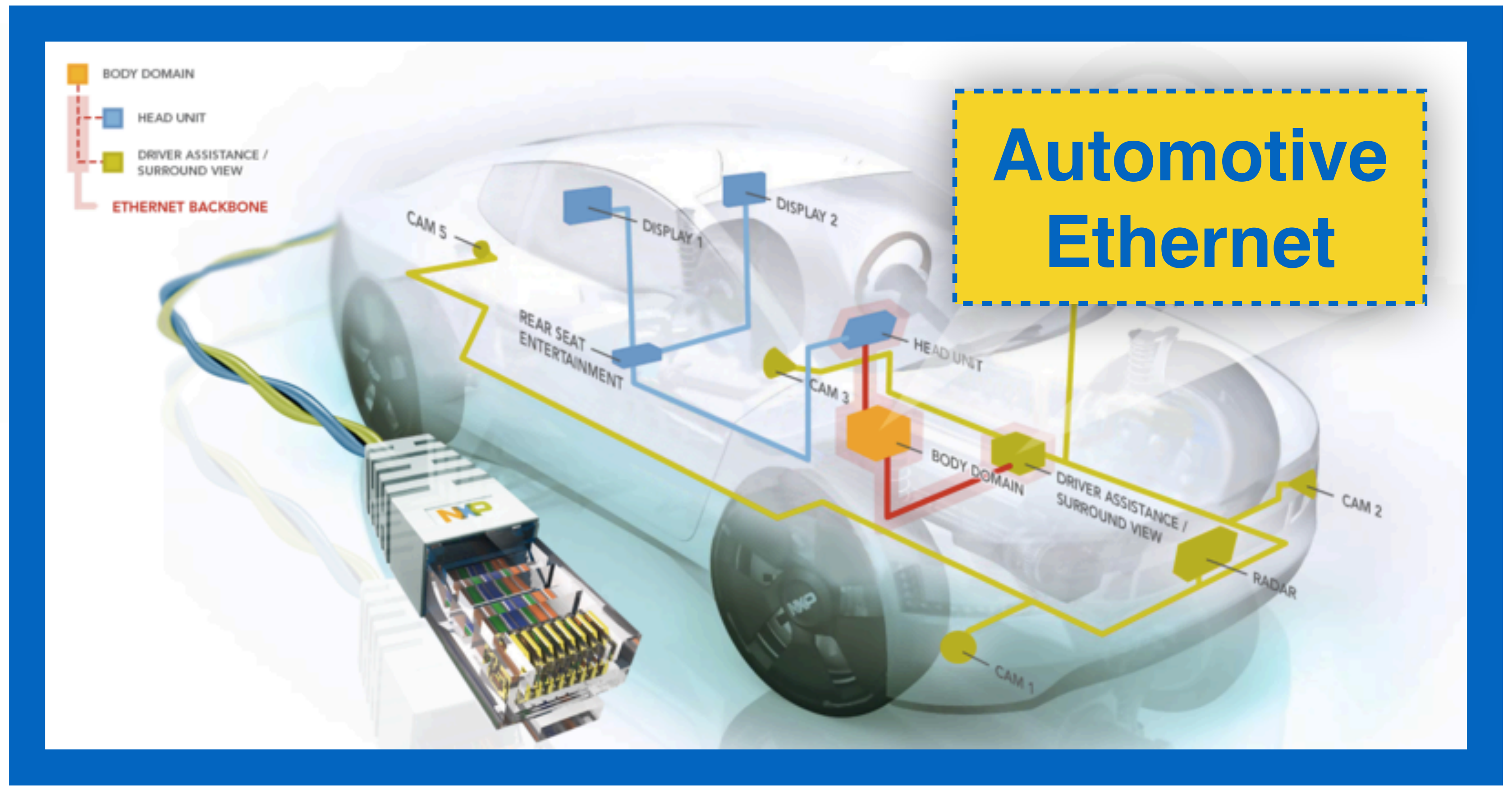
- **Quantifiably negligible** failure rates

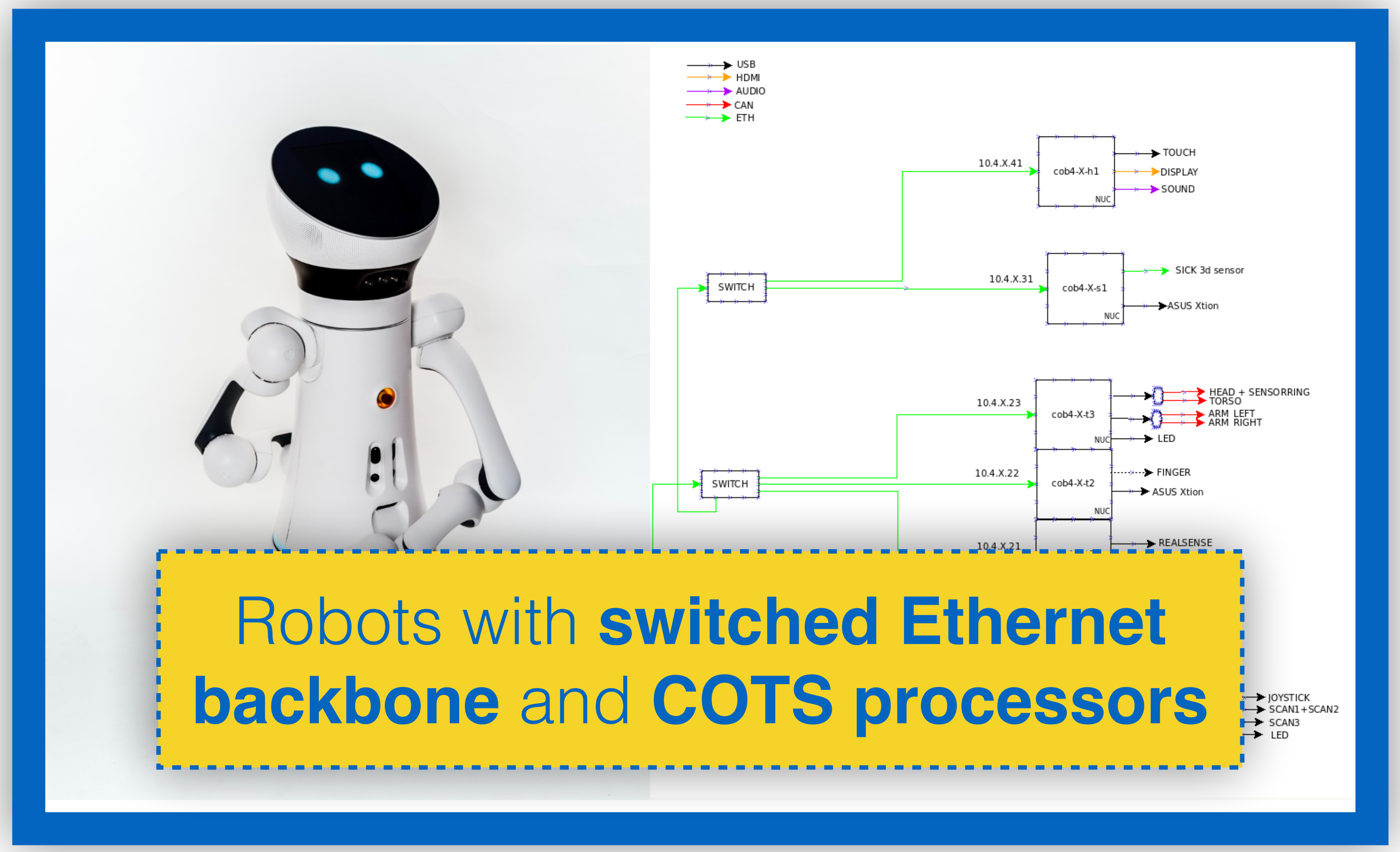
Motivating Trends



Ultra-reliability

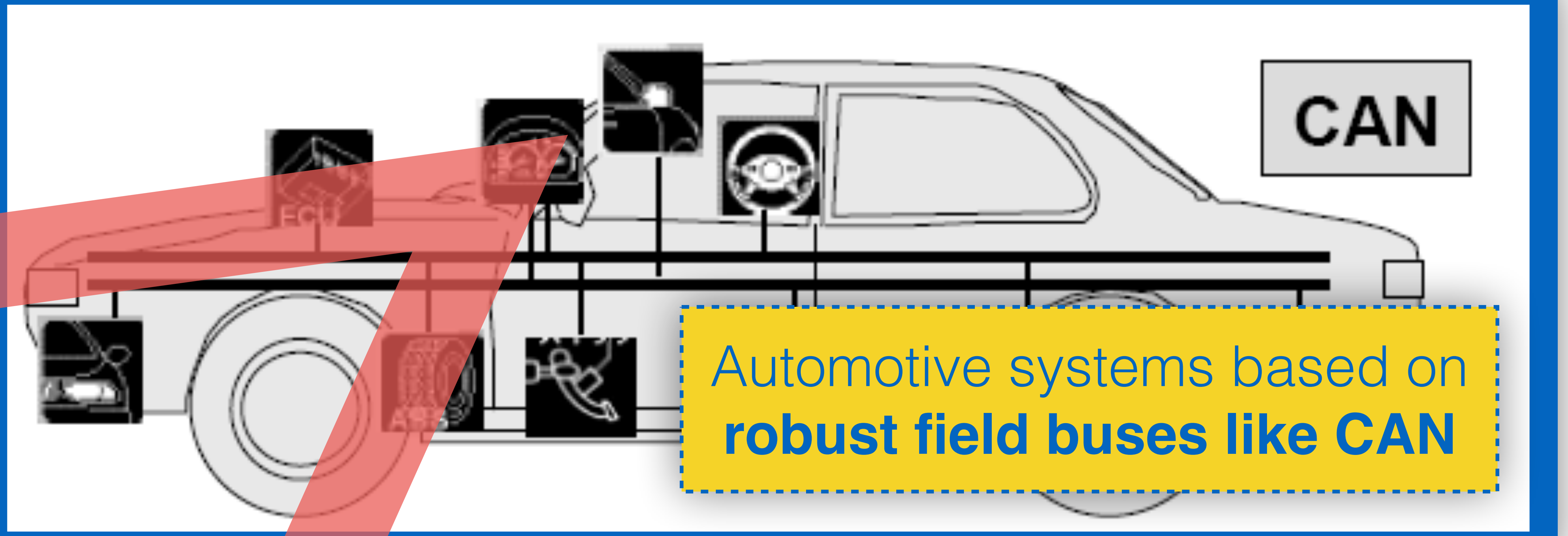
- **Quantifiably negligible** failure rates





Motivating Trends

Avionics domain systems with **custom hardware** and **very high redundancy**



Automotive systems based on **robust field buses like CAN**

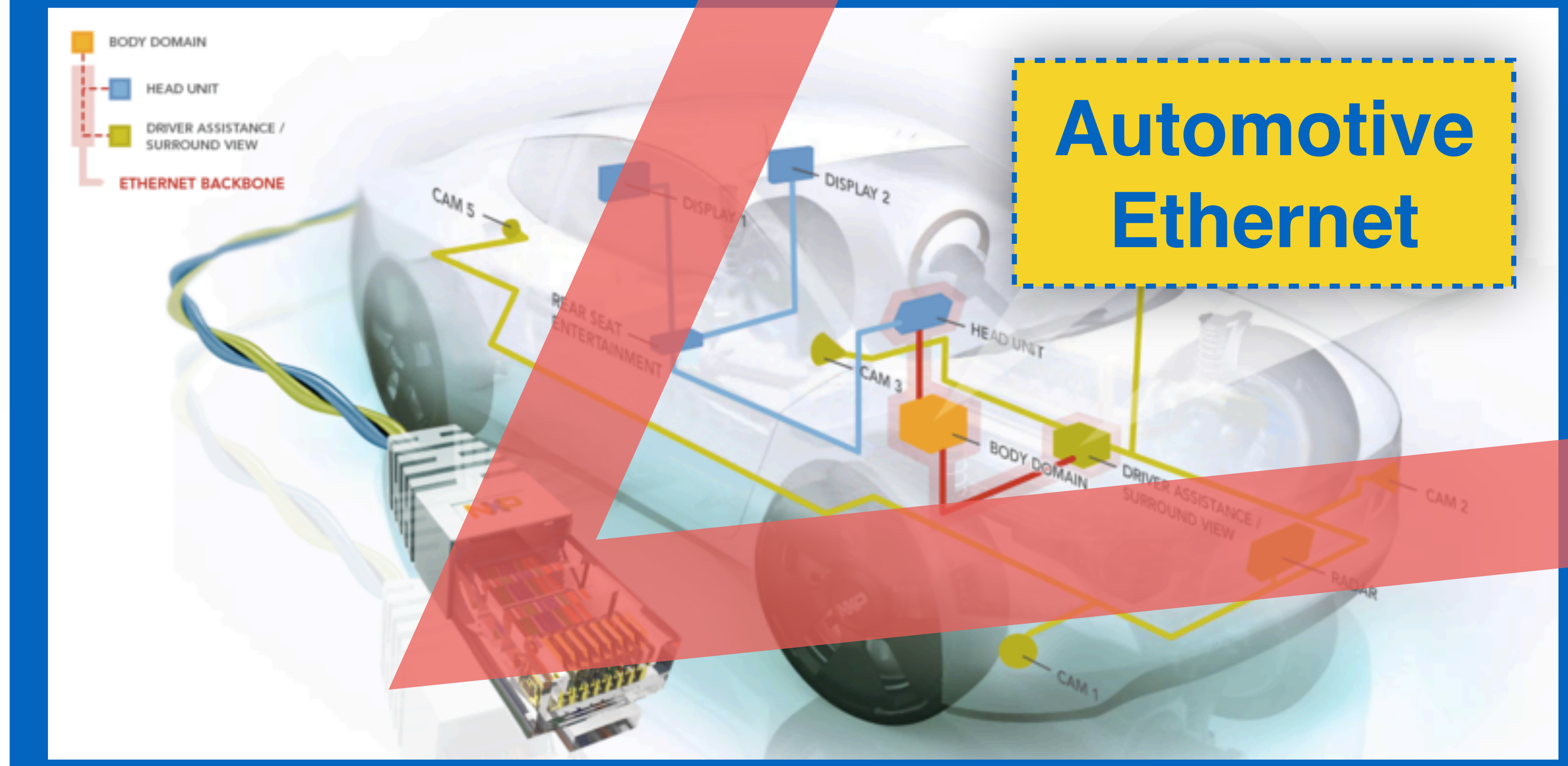
COTS hardware
(inexpensive, but unreliable)

Inadequate resources
(developer hours, computing power, component costs)

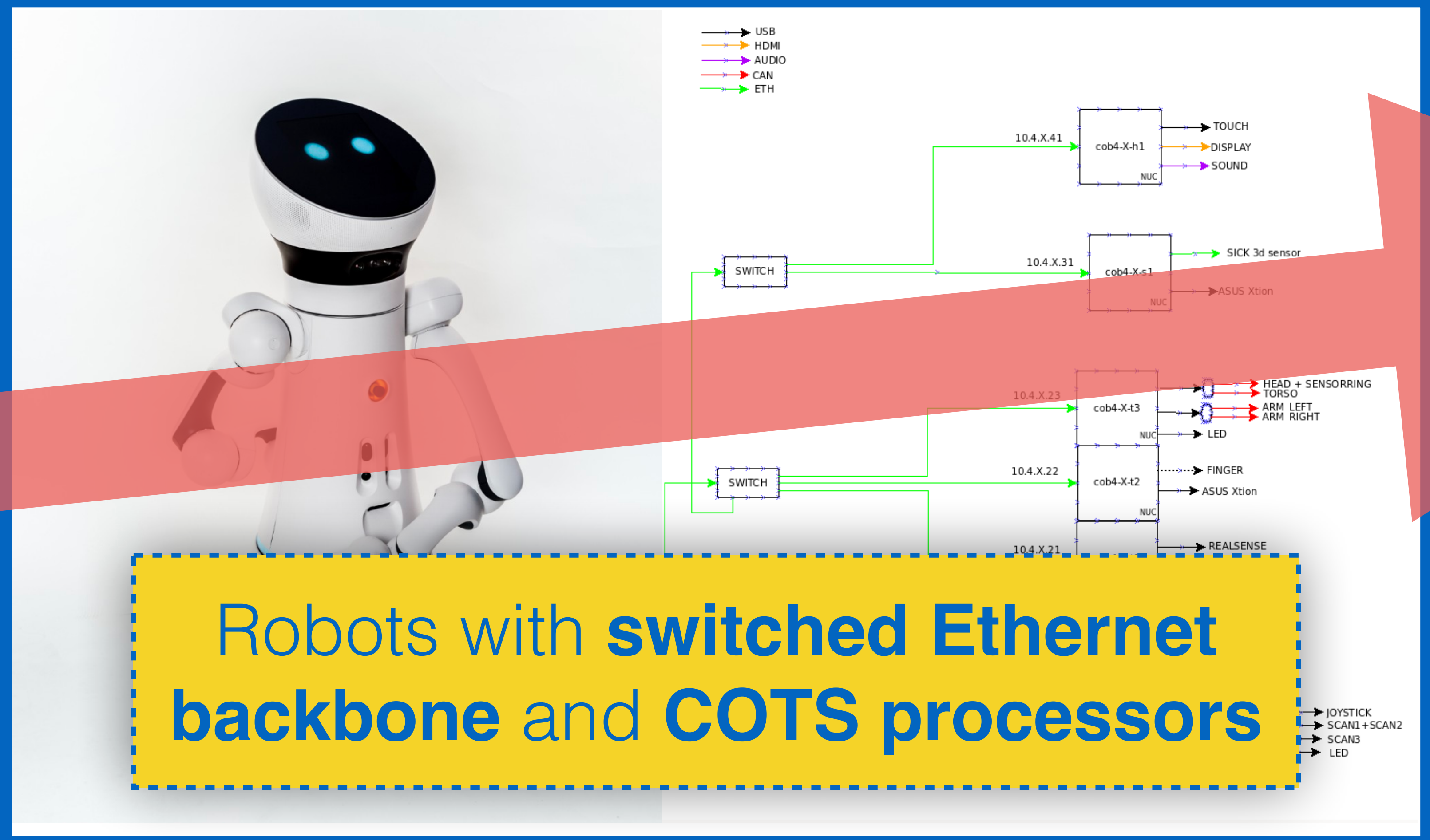
Time-to-market pressures

Ultra-reliability

- **Quantifiably negligible** failure rates



Automotive Ethernet



Robots with **switched Ethernet backbone** and **COTS processors**

Focus

Reliability analysis of **Ethernet**-based **distributed real-time** systems

PhD Thesis

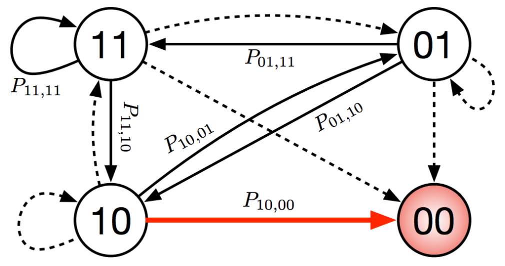
Reliability Analysis of Distributed Real-Time Systems

Best Presentation Award

Controller Area Network [ECRTS '18]

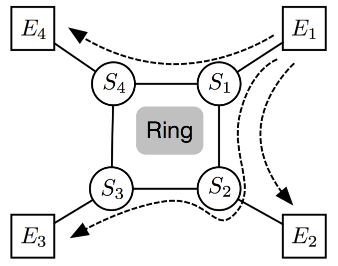
CAN

Periodic Weakly-Hard Systems [ECRTS '19]



Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]



This Talk

This Talk

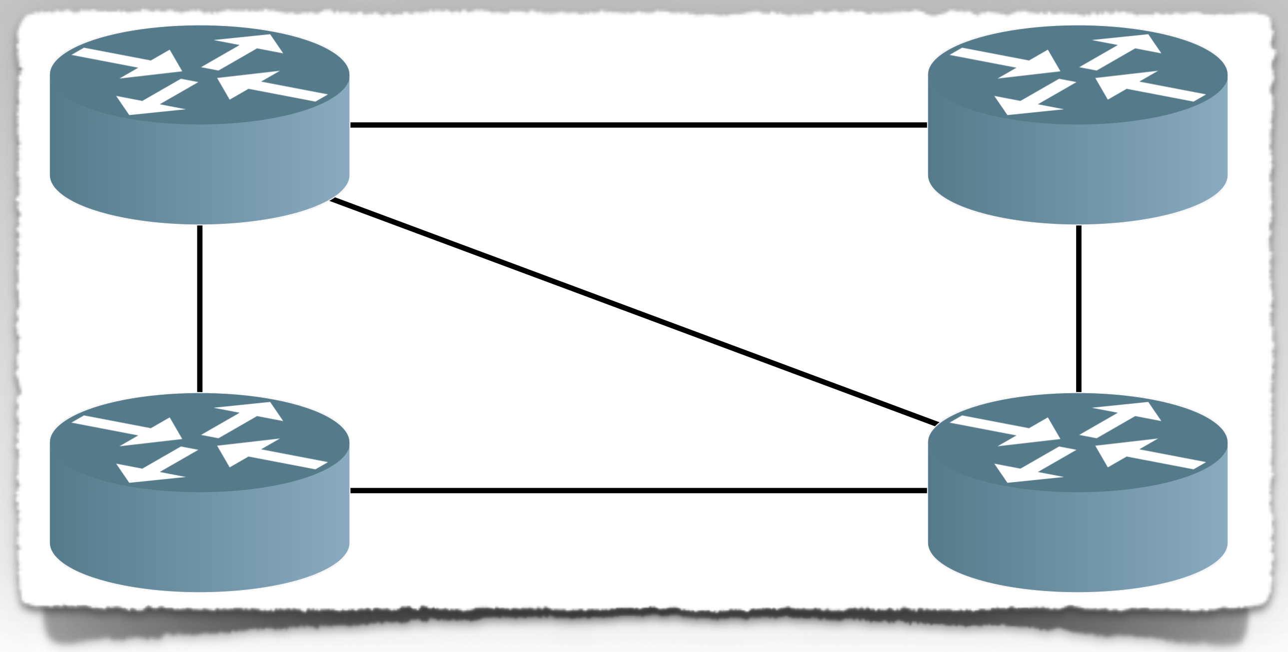
System Model

Network control system

System Model

Network control system

Ethernet Time-Sensitive Networking (TSN)

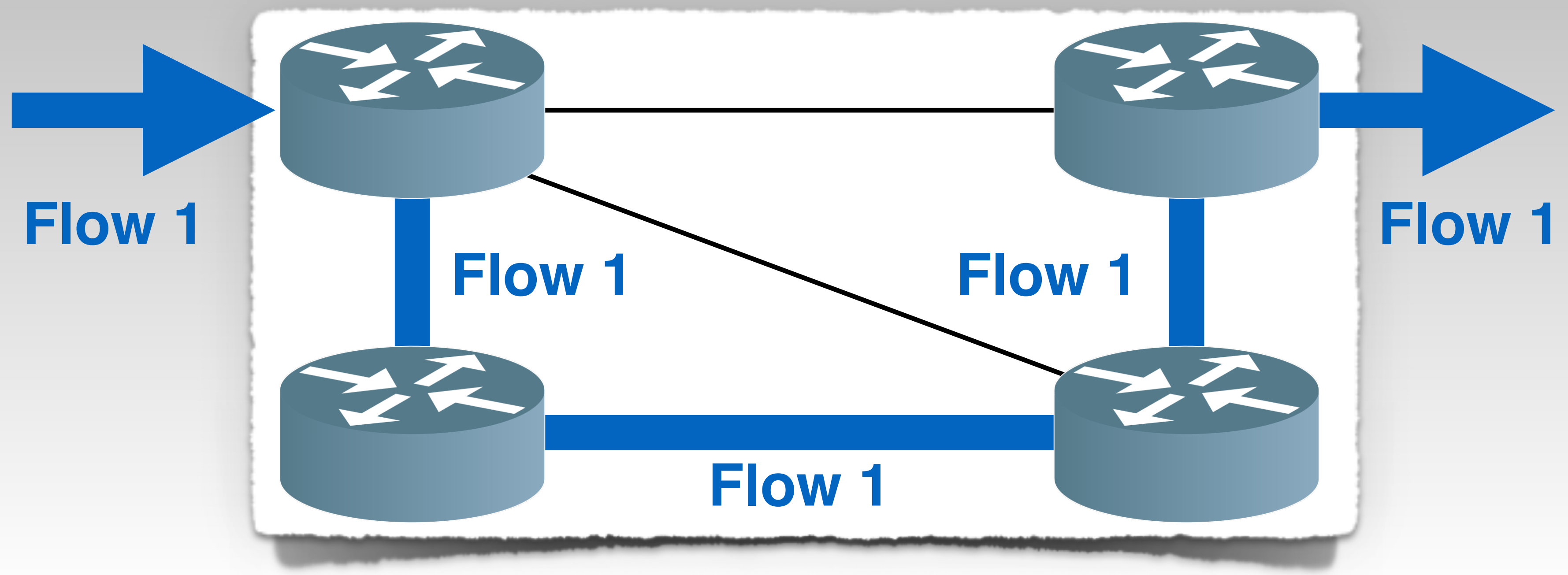


System Model

Network control system

Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes



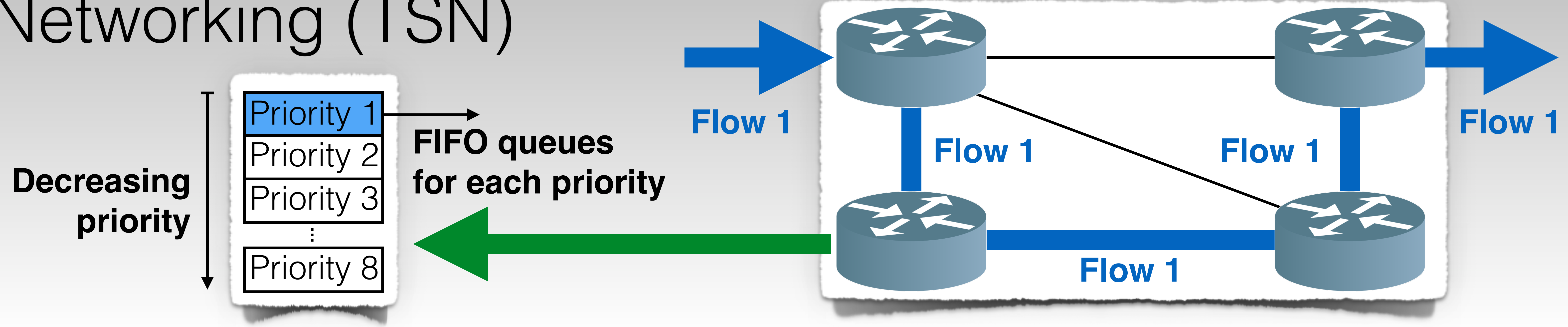
System Model

Network control system

Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes



System Model

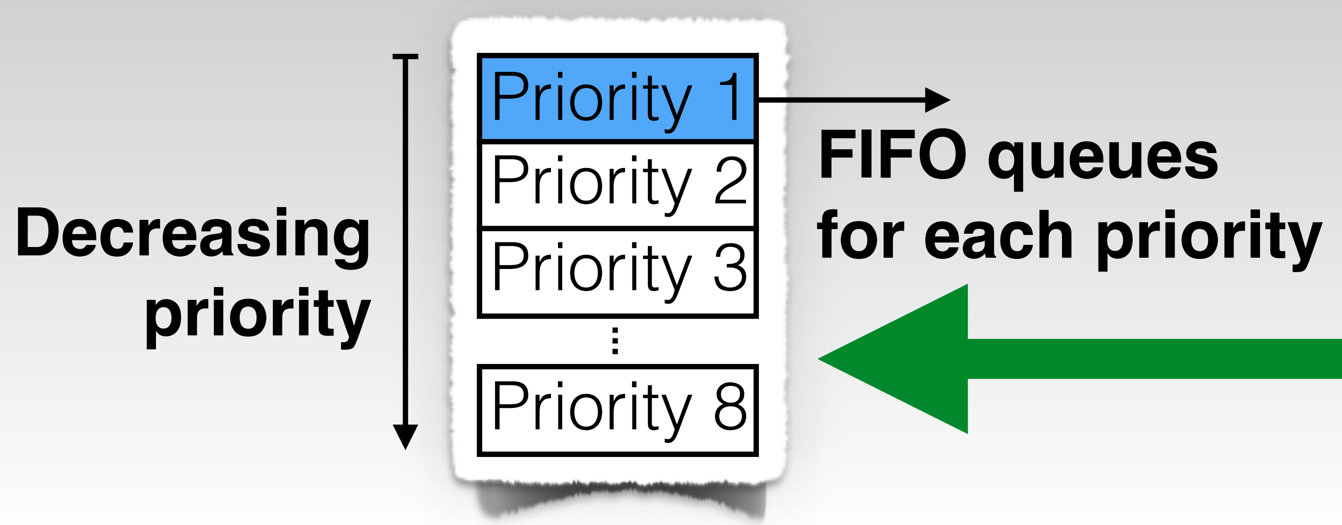
Network control system

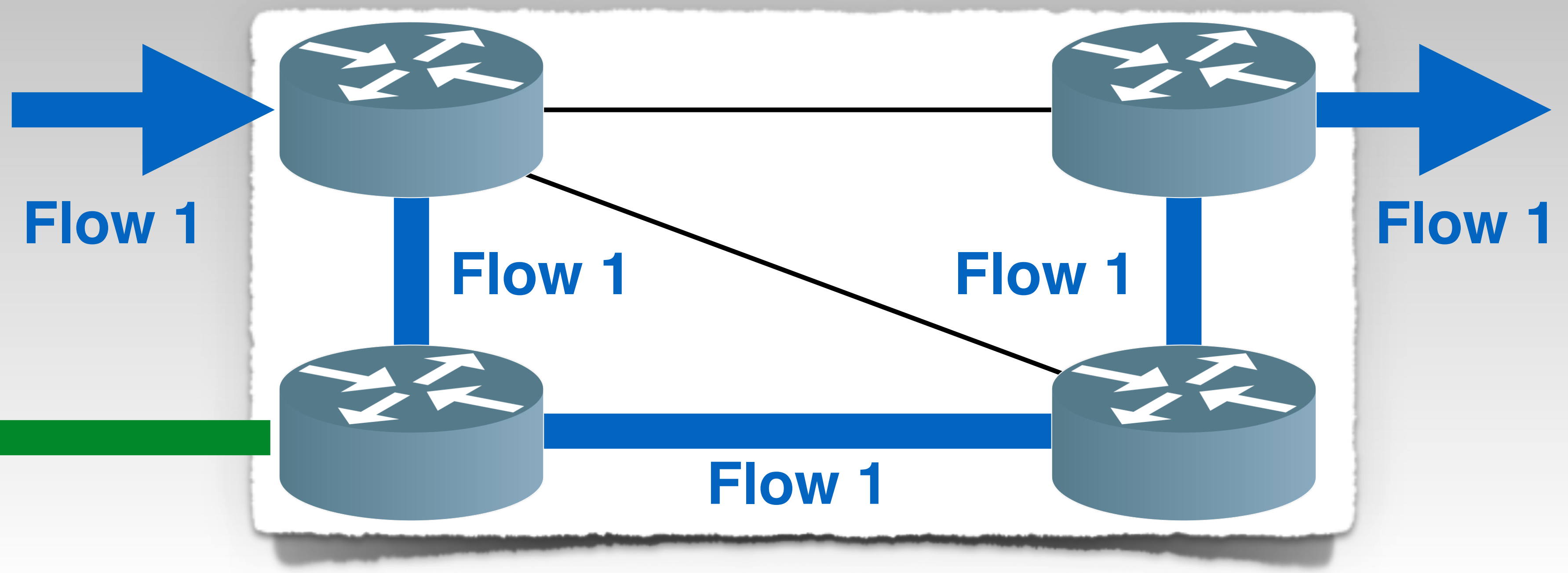
Atomic Broadcast

Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes





System Model

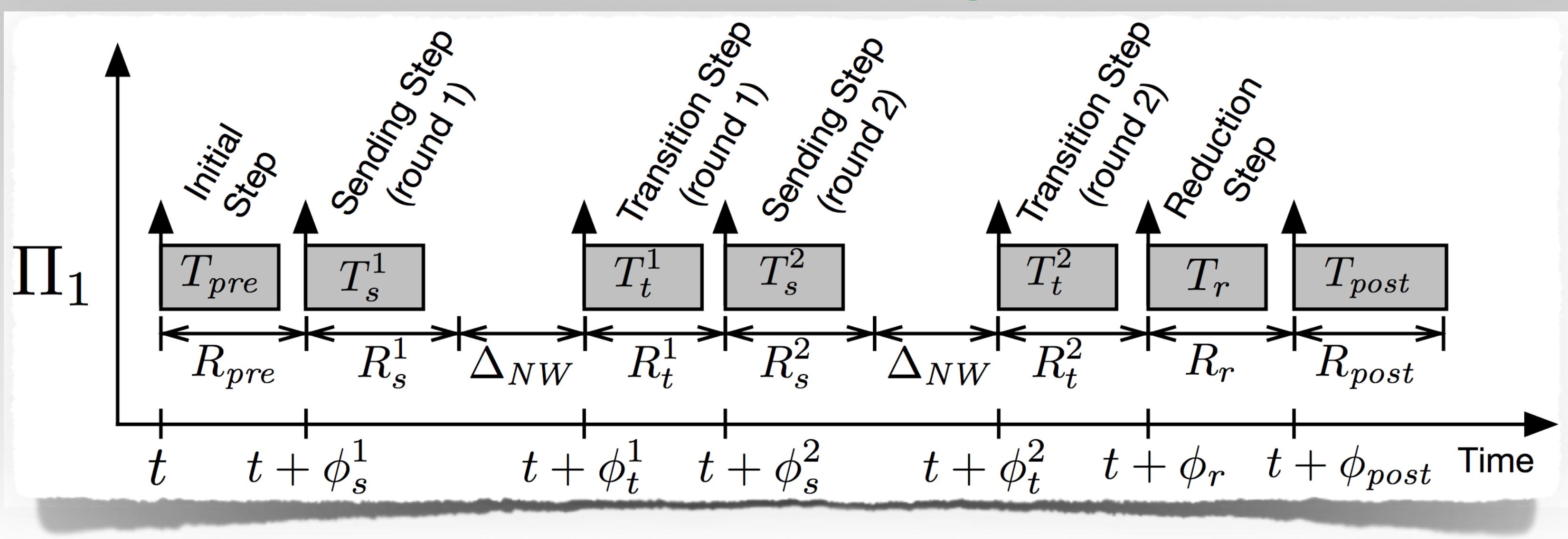
Network control system

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

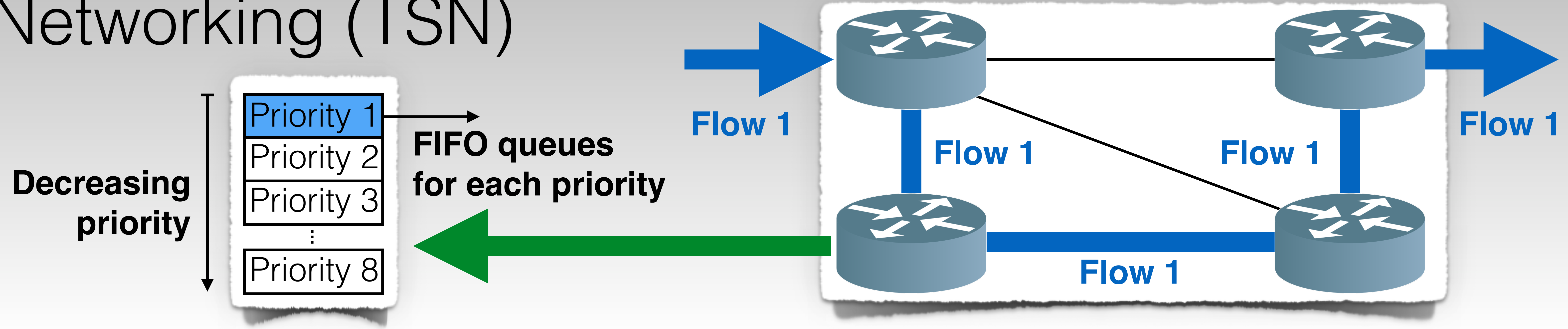
Periodic tasks and messages



Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes



System Model

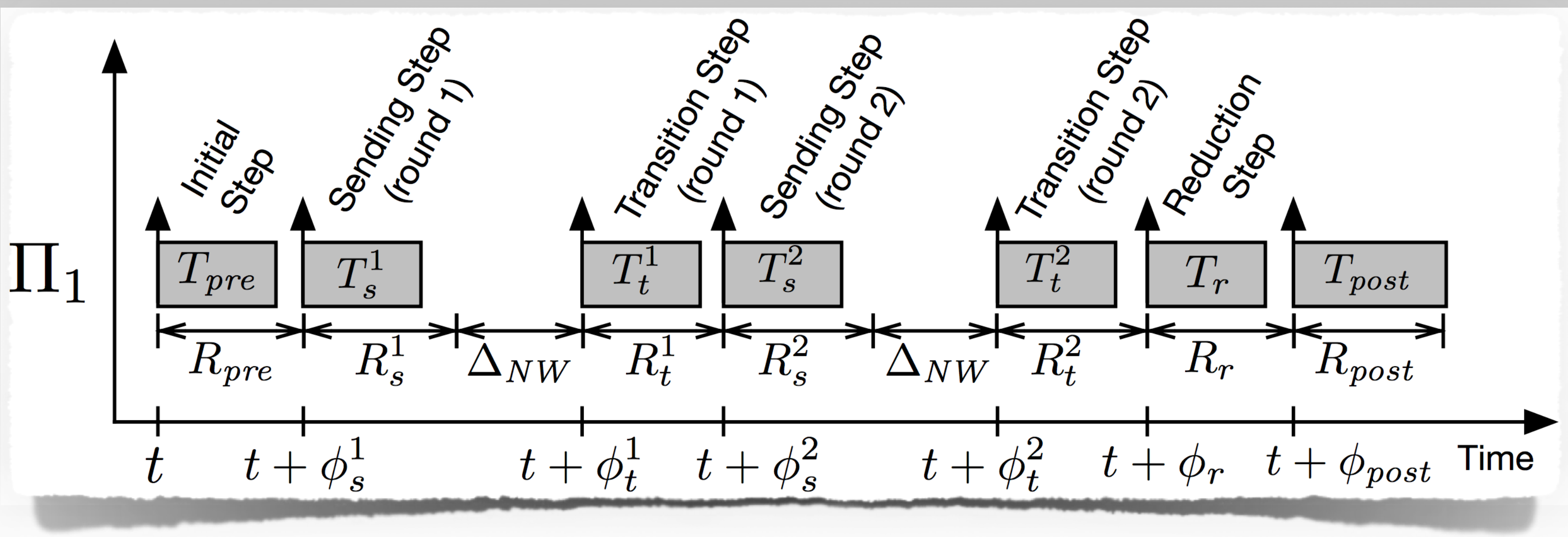
Network control system

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages

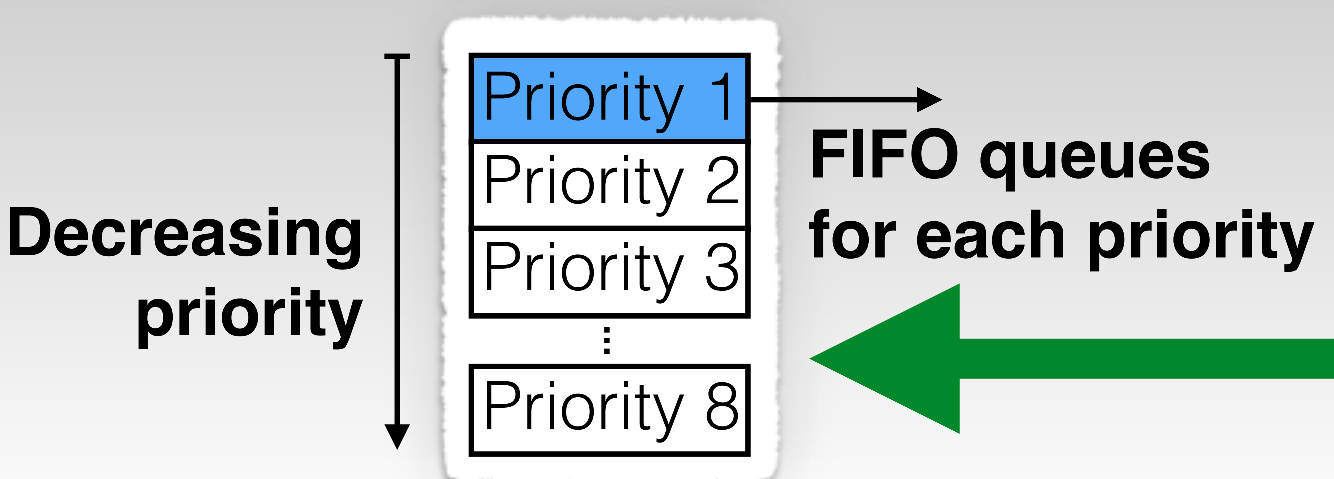


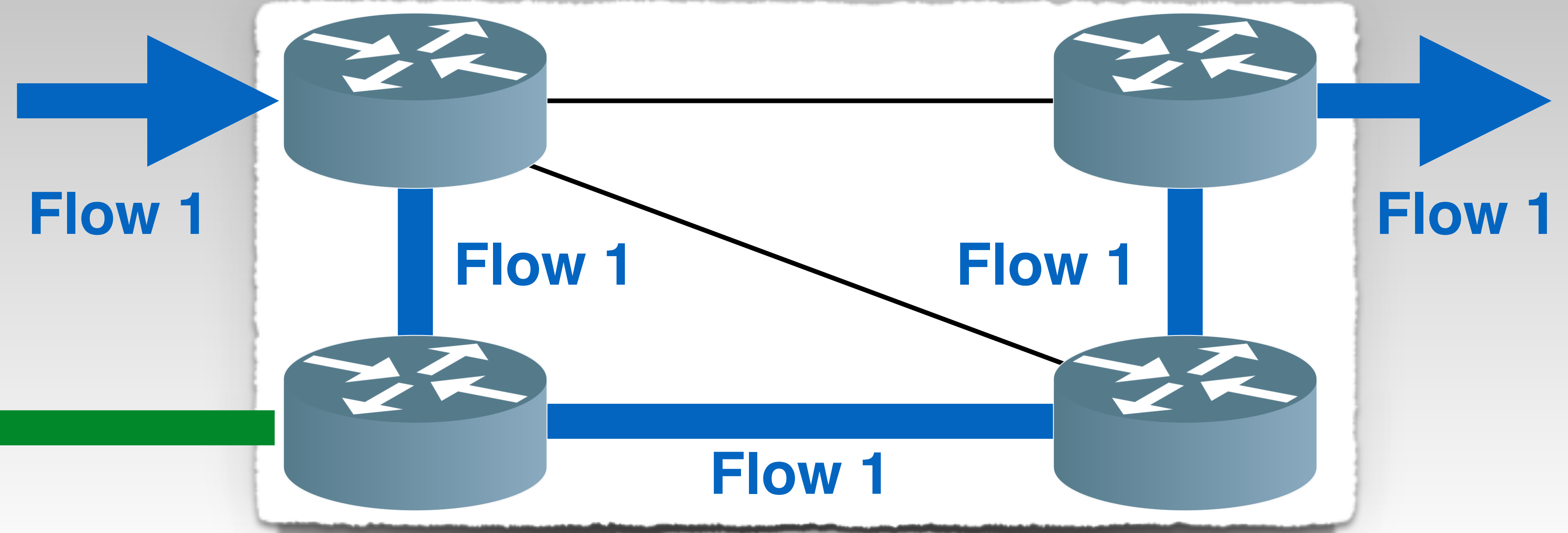
Clock synchronization

Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes



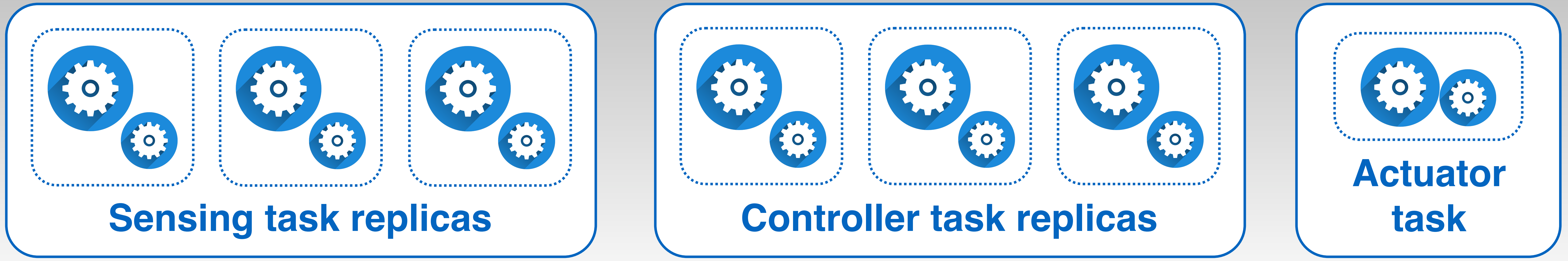


System Model

Network control system

Active Replication

DMR / TMR / Hybrid

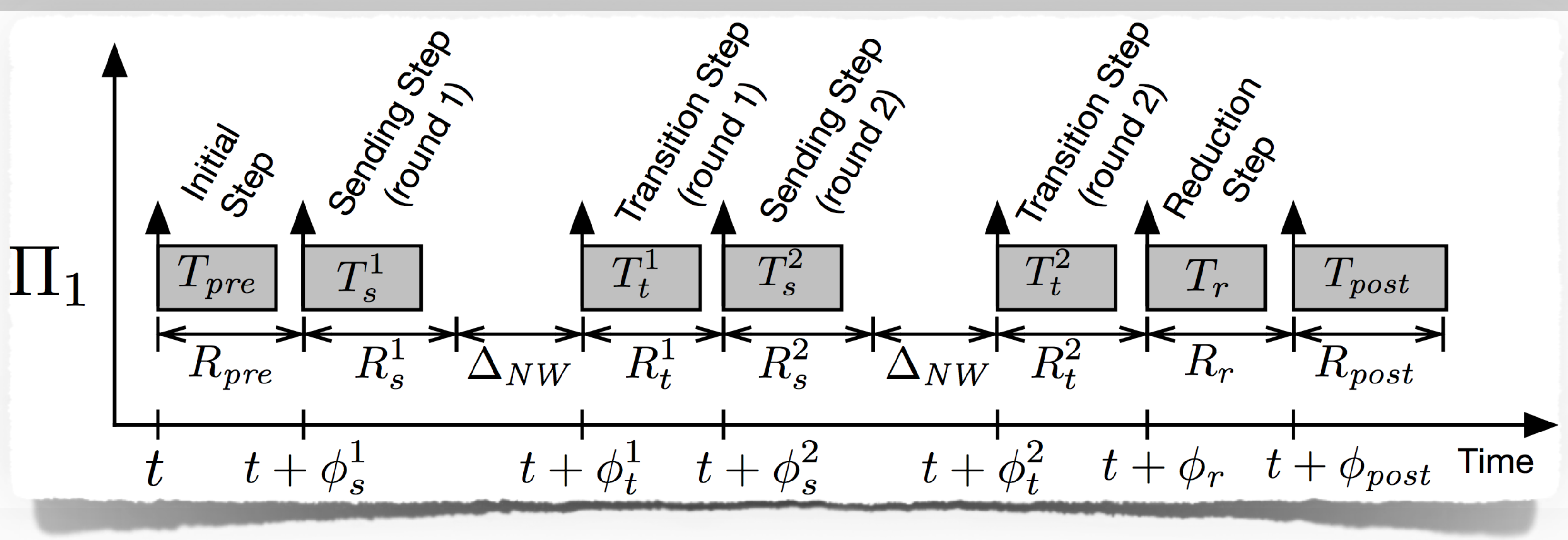


Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

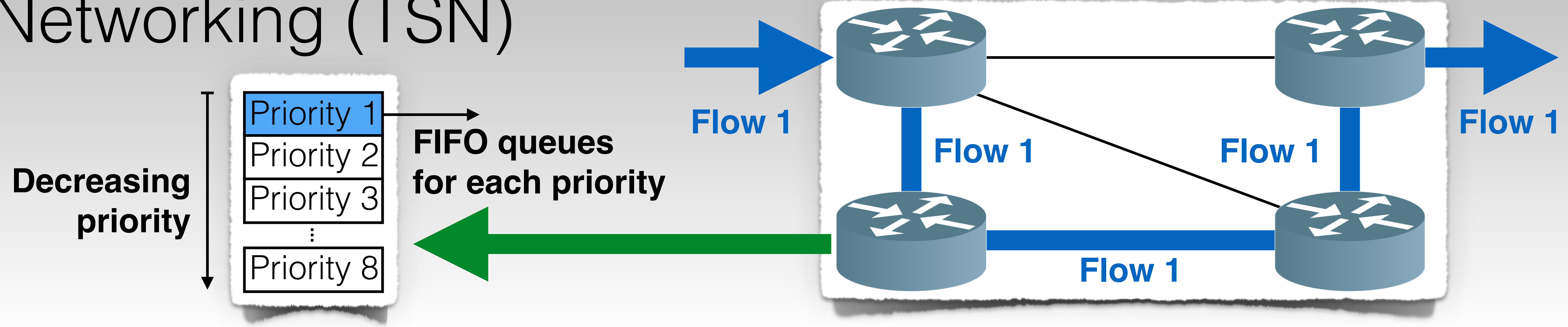
Periodic tasks and messages



Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes



System Model

Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Active Replication

DMR / TMR / Hybrid

Sensing task replicas

Controller task replicas

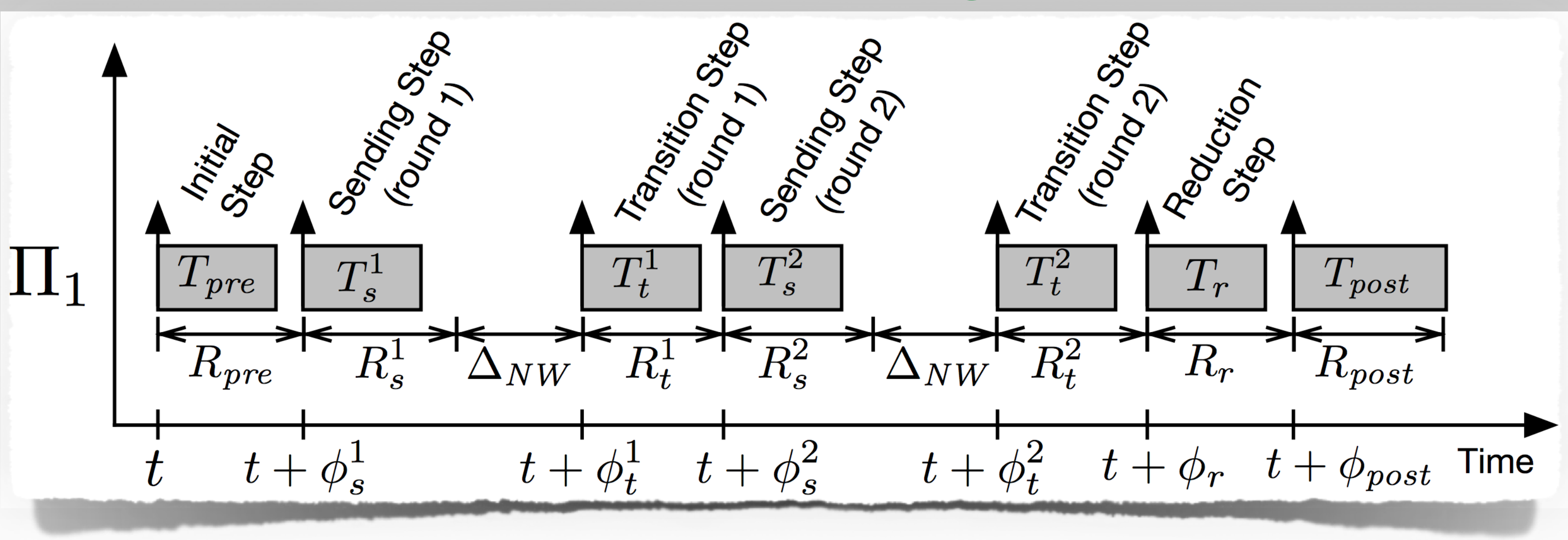
Actuator task

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Clock synchronization

Ethernet Time-Sensitive Networking (TSN)

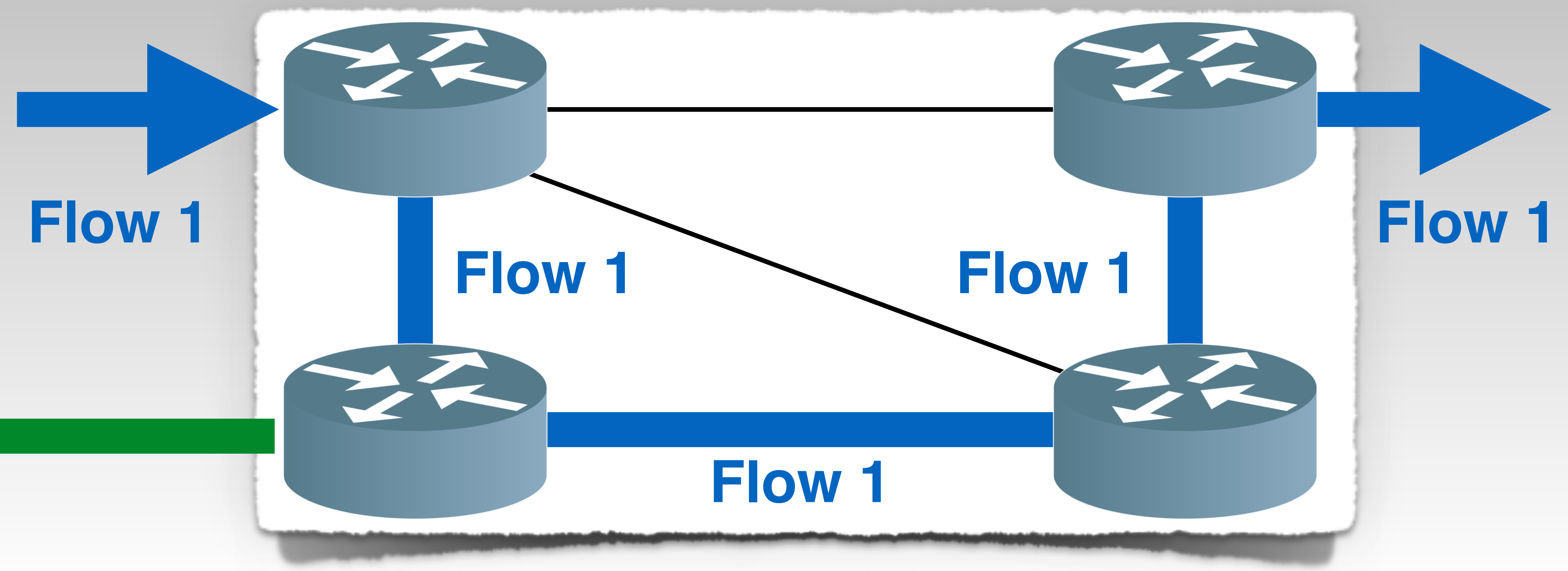
Statically reserved routes

Priority classes

Decreasing priority

- Priority 1
- Priority 2
- Priority 3
- ⋮
- Priority 8

FIFO queues for each priority



System Model

Network control system

Physical sensor — Controlled plant — Physical actuator

Physical plant reliable

Active Replication

DMR / TMR / Hybrid

Omission / incorrect computation

Sensing task replicas

Controller task replicas

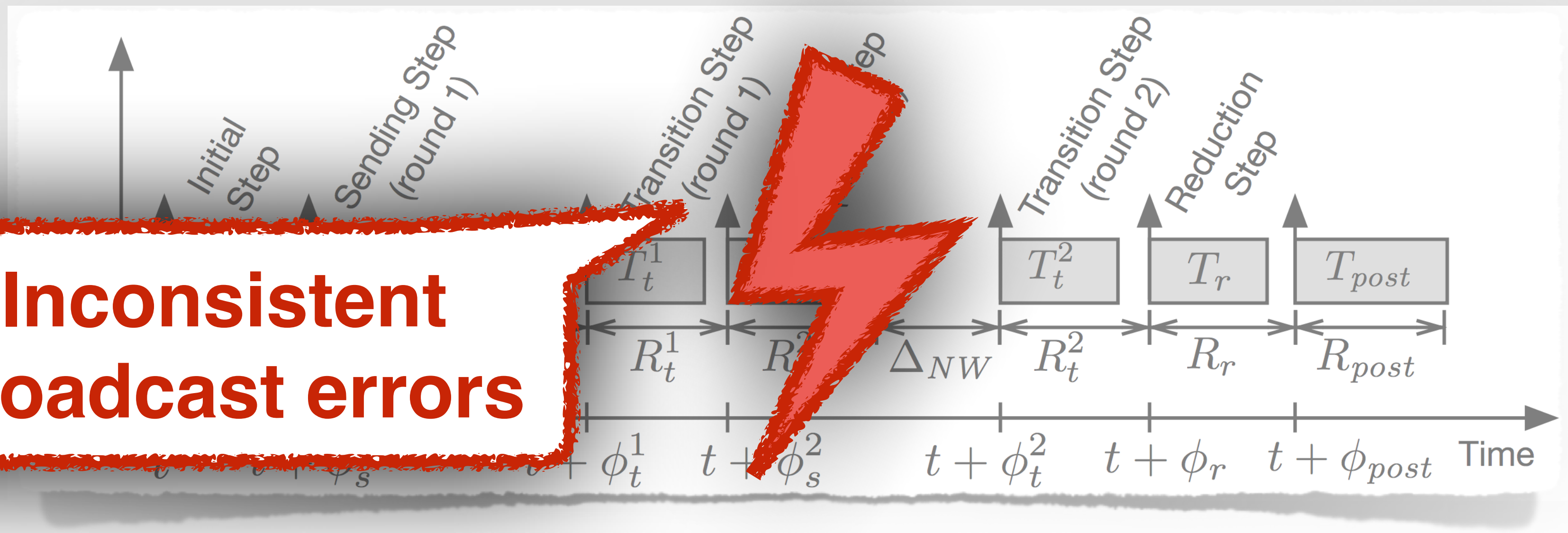
Actuator task

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 19…]

Periodic tasks and messages



Inconsistent broadcast errors

Clock synchronization

Ethernet Time Sensitive Networking (TSN)

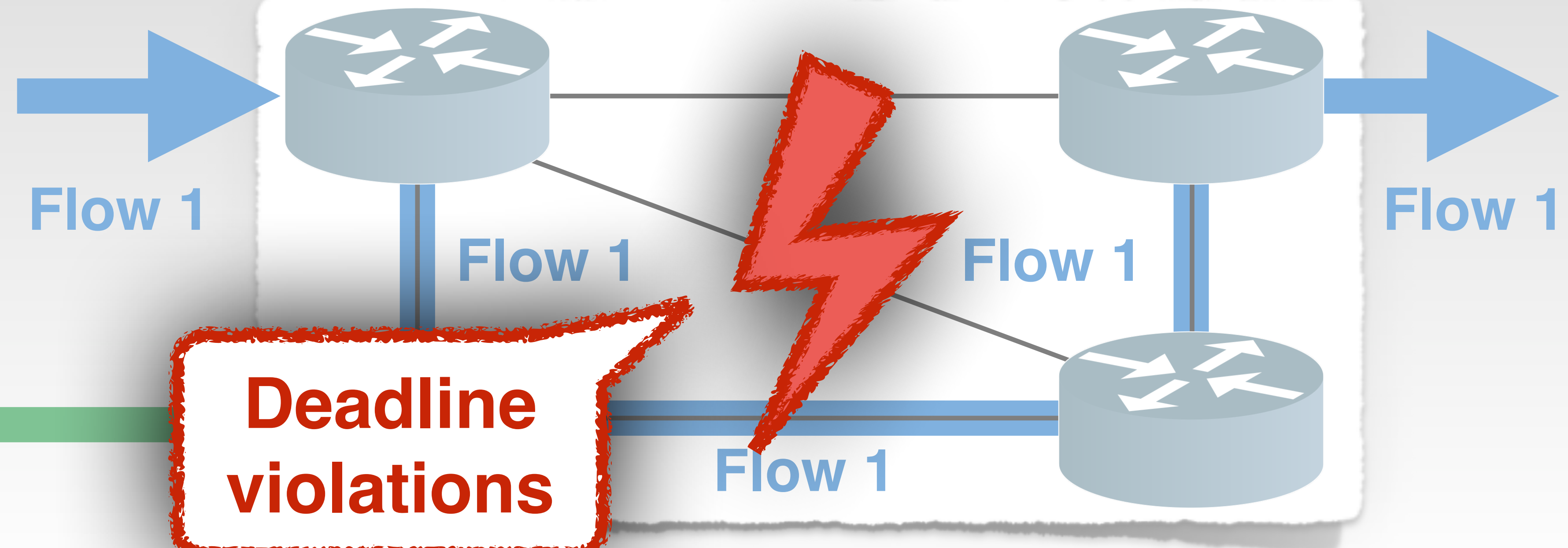
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



Deadline violations

System Model

Network control system

Physical plant reliable

Physical sensor — Controlled plant — Physical actuator

Active Replication

DMR / TMR / Hybrid

Omission / incorrect computation

...sing task replicas

Co...

Redundant, but correlated & erroneous, inputs

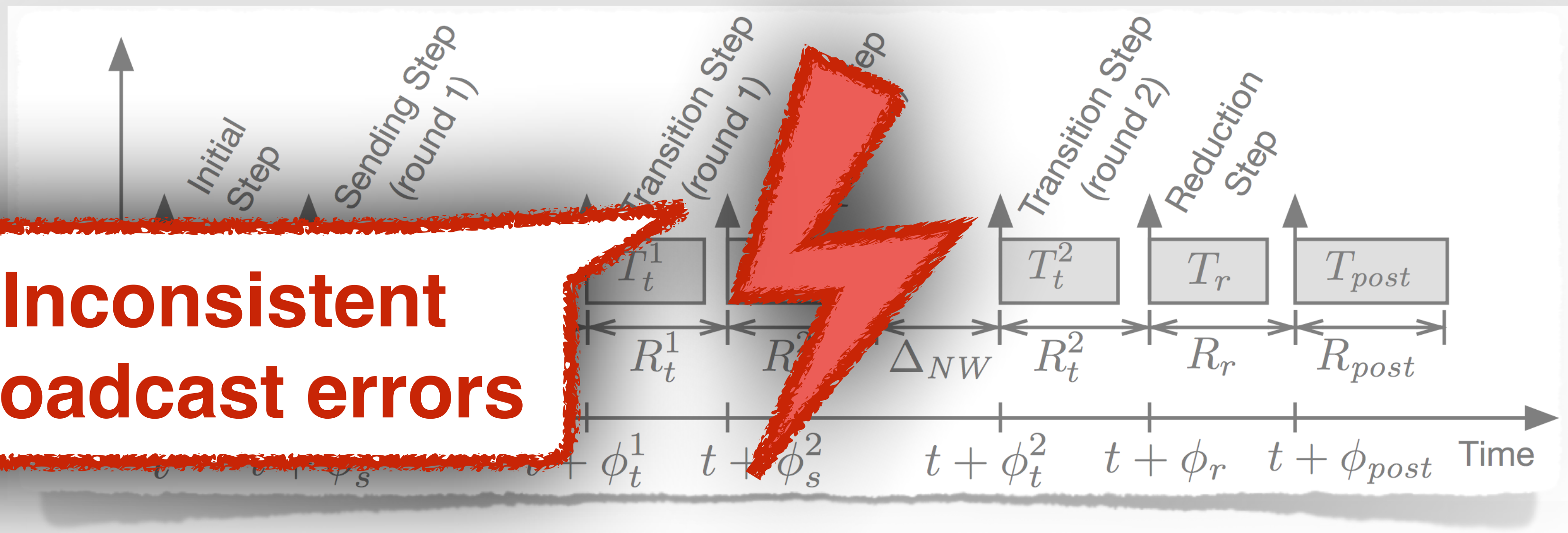
Actuator task

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT proto... [Pease et al., 19...

Periodic tasks and messages



Inconsistent broadcast errors

Clock synchronization

Ethernet Time Sensitive Networking (TSN)

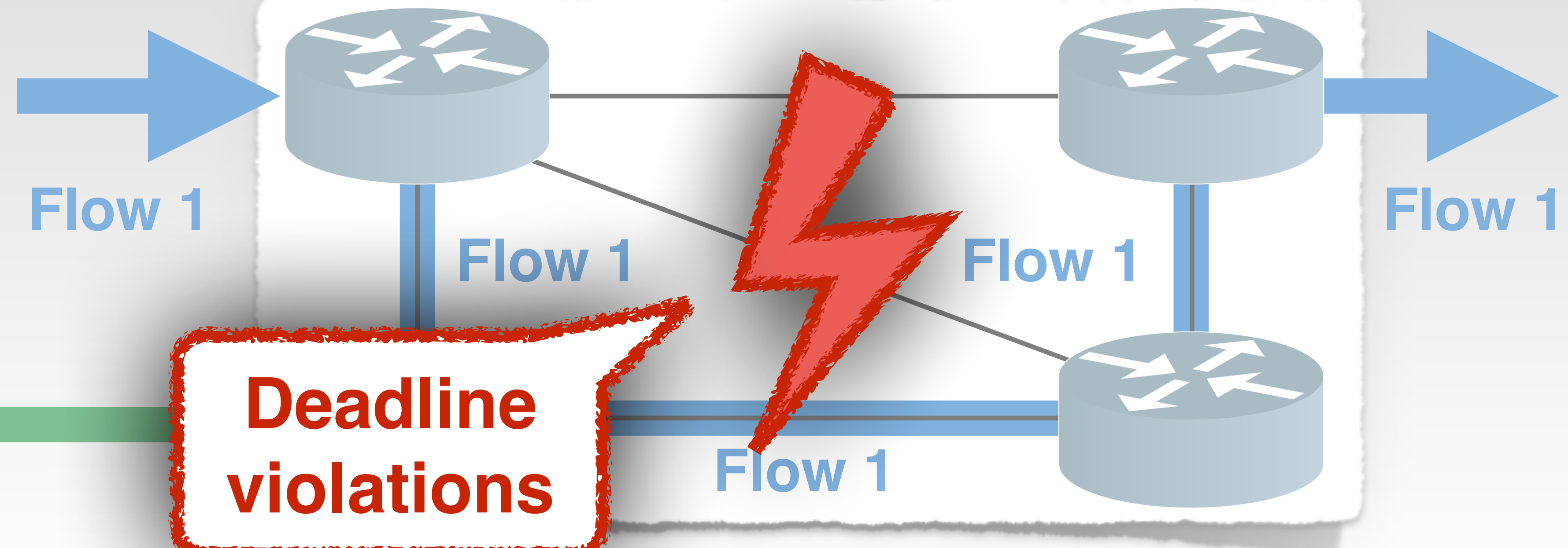
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



Deadline violations

System Model

Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Active Replication

DMR / TMR / Hybrid

Sensing task replicas

Controller task replicas

Actuator task

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages

Clock synchronization

Ethernet Time (TSN)

Statically reserv

Priority classes

y 1
y 2
y 3
Priority 8

FIFO queues for each priority

Transient fault-induced errors

System Model

Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Active Replication

DMR / TMR / Hybrid

Sensing task replicas

Controller task replicas

Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 198…]

Periodic tasks and messages

Ethernet Time… (TSN…

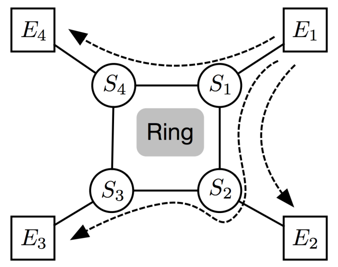
Statically reserv…

Priority…

Transient fault-induced errors

Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



System Model

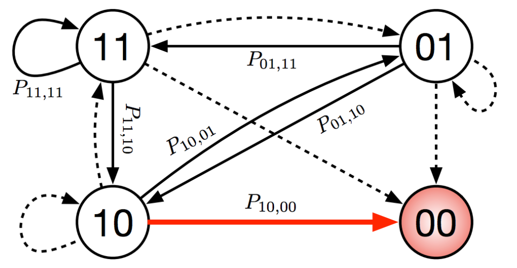
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

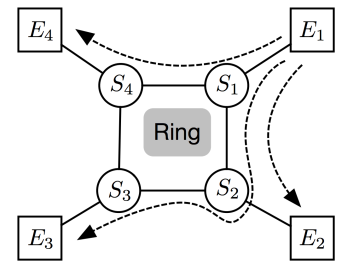
Q2. What is the likelihood of a **control failure?**

Periodic Weakly-Hard Systems [ECRTS '19]



Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



Transient fault-induced errors

System Model

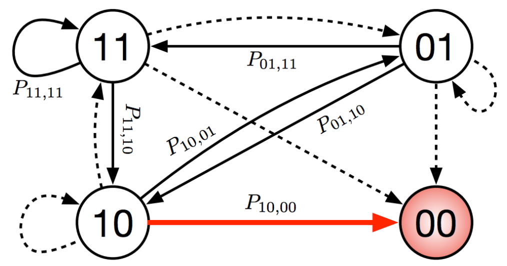
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

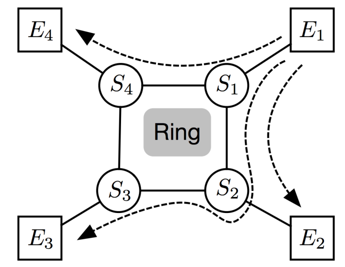
Q2. What is the likelihood of a **control failure?**

Periodic Weakly-Hard Systems [ECRTS '19]



Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



Transient fault-induced errors

Stochastically modeled basic errors

Basic errors due to transient faults are random, independent events

- E.g., node crashes, link corruption

Stochastically modeled basic errors

Basic errors due to transient faults are random, independent events

- E.g., node crashes, link corruption

Poisson distribution using **peak rates** from maximum interference periods

Stochastically modeled basic errors

Basic errors due to transient faults are random, independent events

- E.g., node crashes, link corruption

Poisson distribution using **peak rates** from maximum interference periods

For processors and switches	**Poisson(n, δ, λ_{crash})** = Pr(n crashes in an interval of length δ I crash rate λ_{crash})
For processors, switches, and network links	**Poisson(n, δ, $\lambda_{corruption}$)** = Pr(n corruptions in an interval of length δ I corruption rate $\lambda_{corruption}$)

System Model

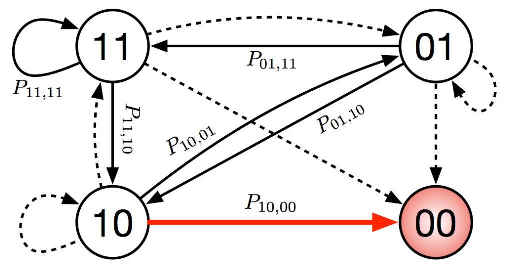
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Q2. What is the likelihood of a **control failure?**

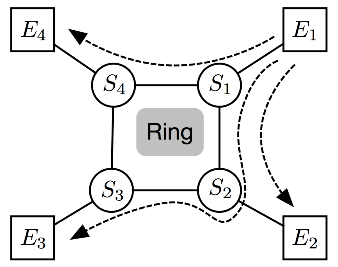
Periodic Weakly-Hard Systems [ECRTS '19]



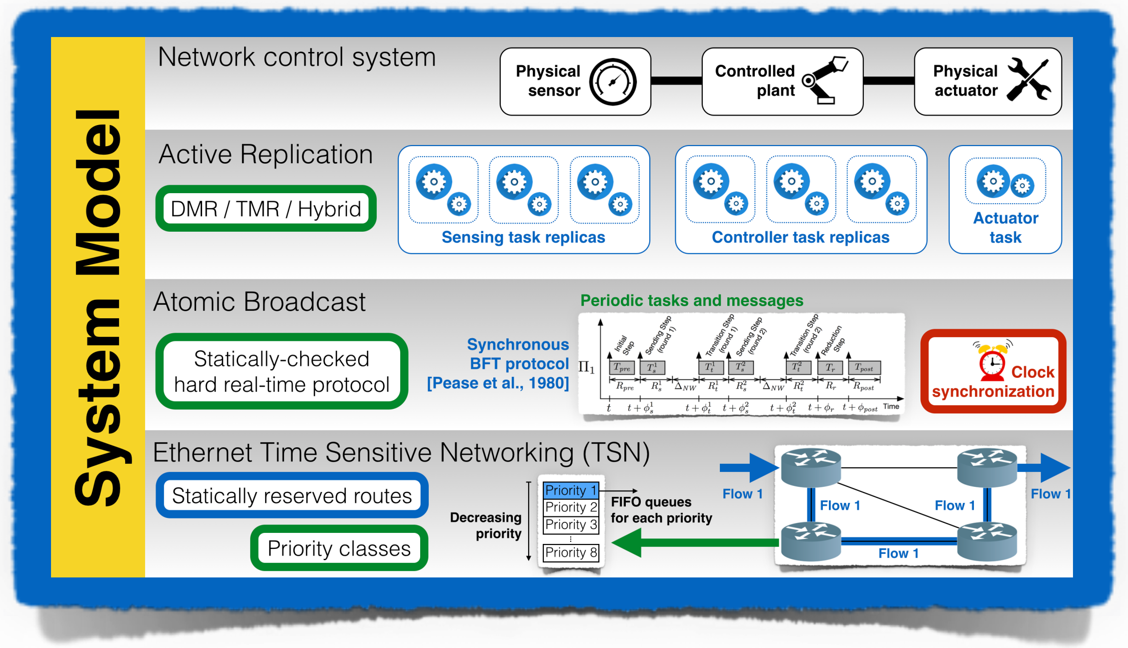
Transient fault-induced errors

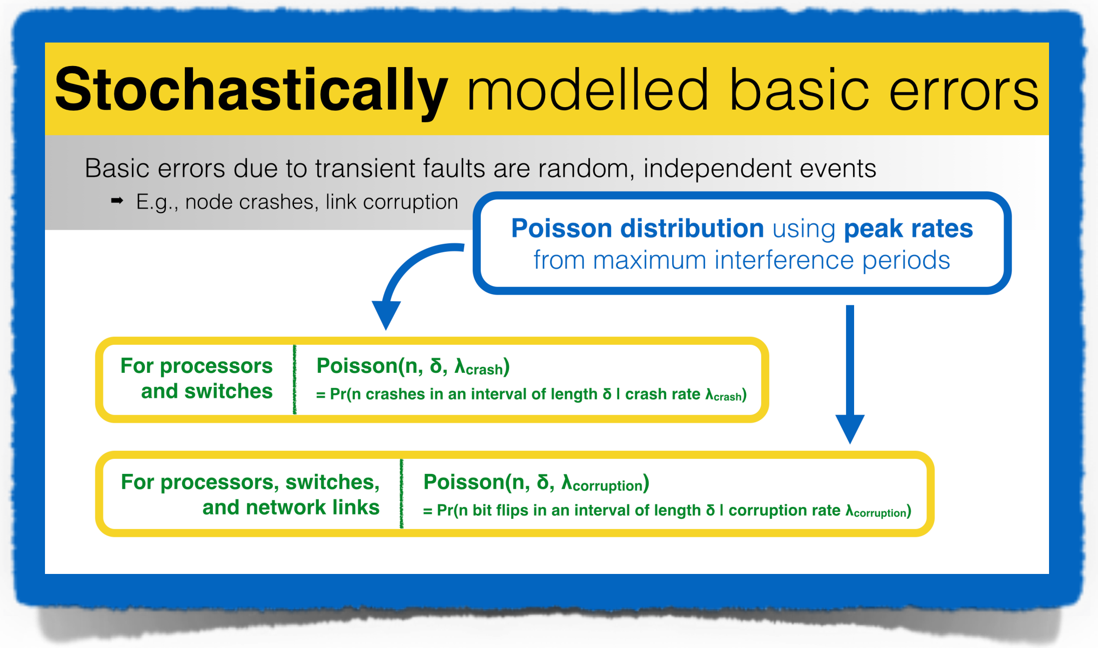
Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



Straw-man Solutions

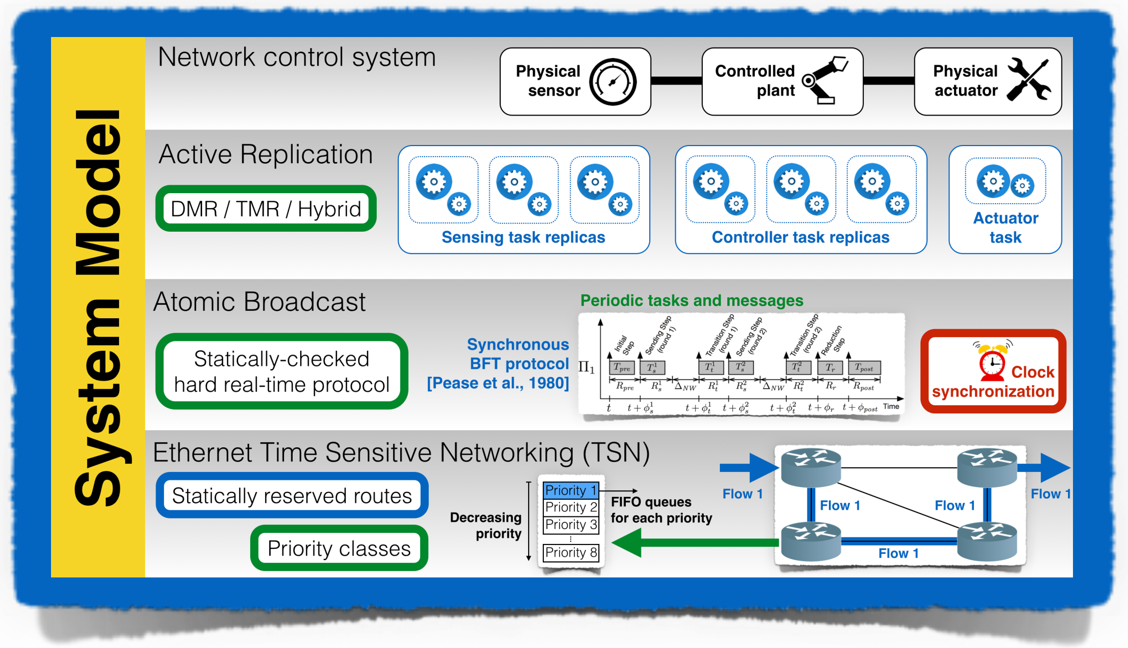


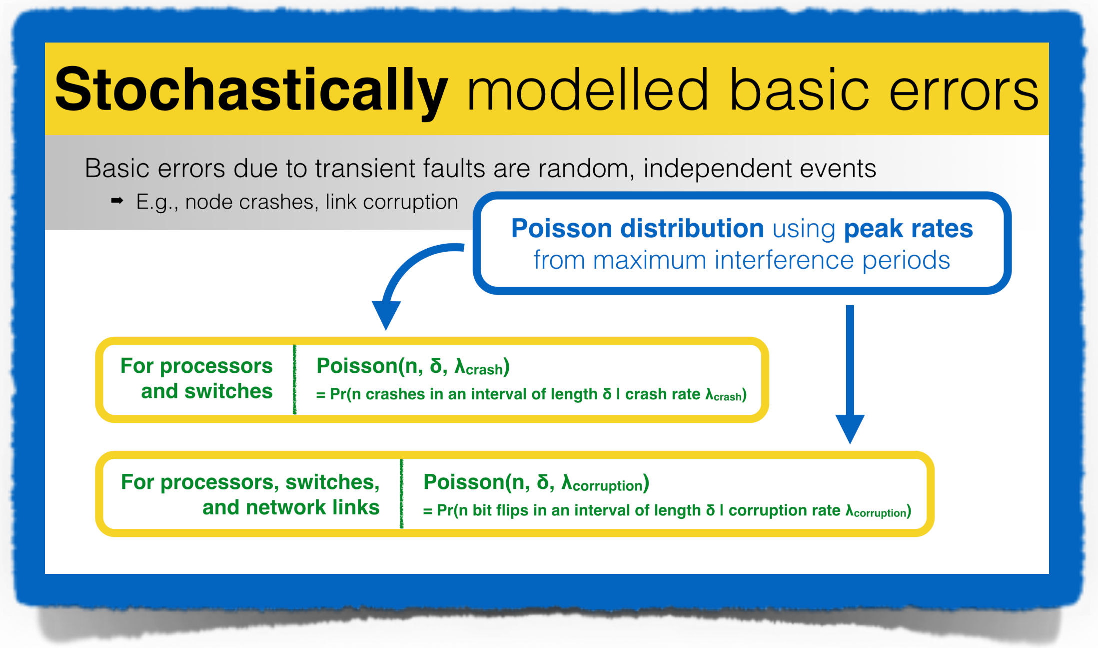


Model checking or simulation

Pr (iteration failure)

Straw-man Solutions





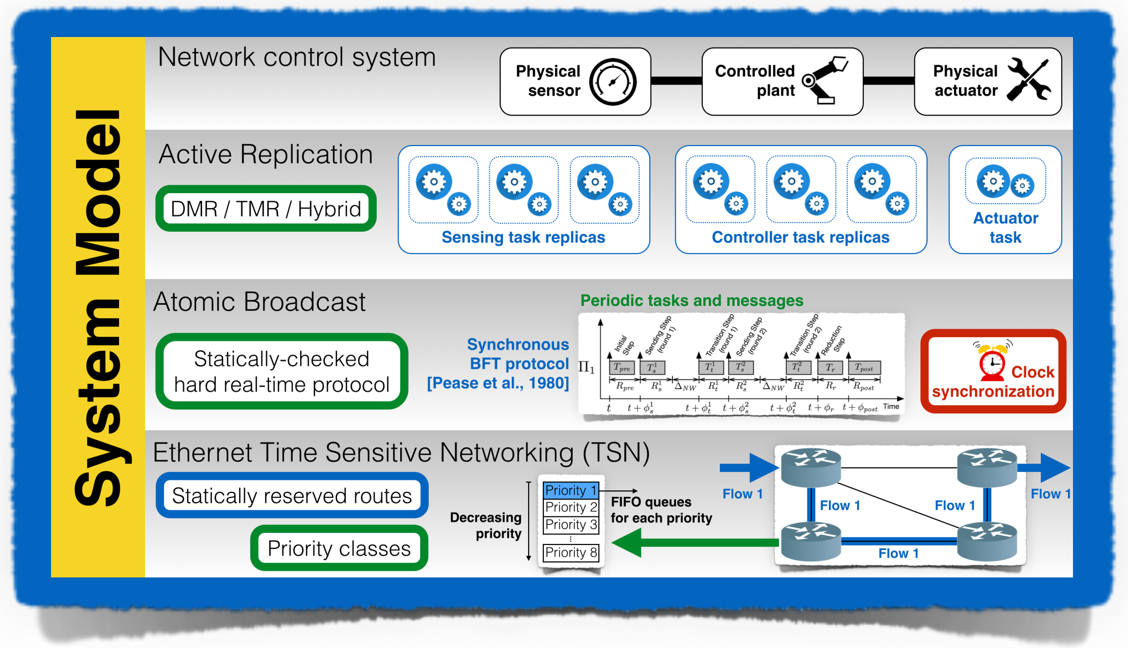
Model checking or simulation

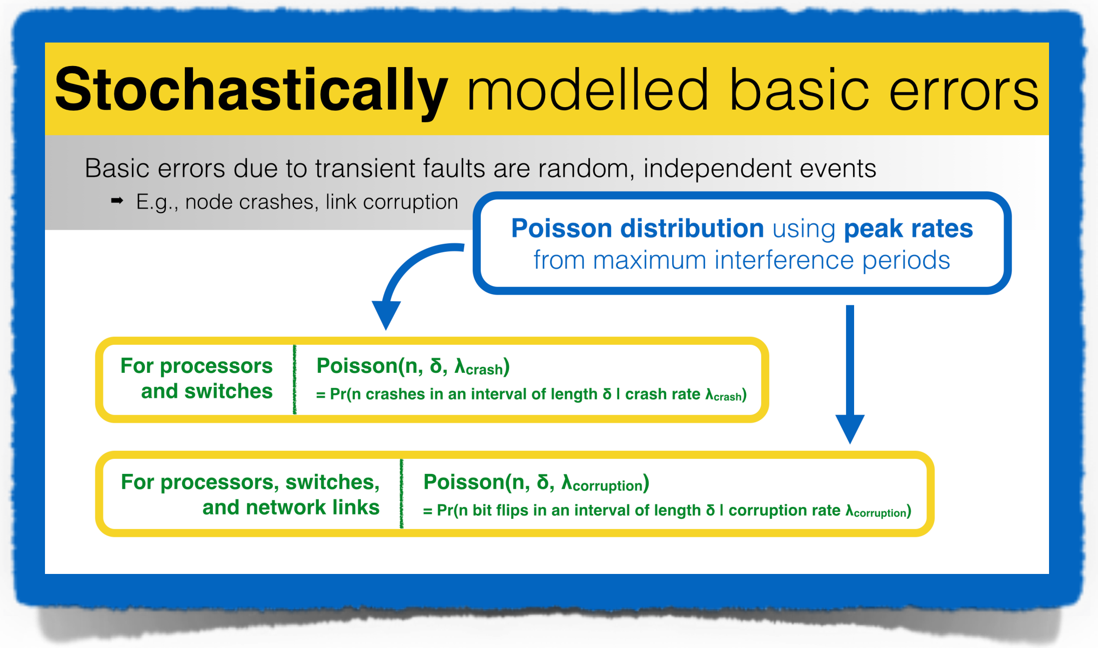
Pr (iteration failure)

Scalability challenges

- Empirical techniques scale poorly when evaluating low-probability events
- Formal methods often do not scale beyond small distributed models

Straw-man Solutions





Model checking or simulation

Pr (iteration failure)

Scalability challenges

- Empirical techniques scale poorly when evaluating low-probability events
- Formal methods often do not scale beyond small distributed models

Reliability anomalies

- In practice, the iteration failure probability may **significantly exceed** the estimated Pr (iteration failure)

Key idea 1: Scalability through **abstraction** and **pruning**

Goal: P_{UB} > Pr (iteration failure)

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1
- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2
- **Corrupted ...**
- **Omitted ...**

Msg. M_3
- **Corrupted ...**
- **Omitted ...**

Goal: P_{UB} > Pr (iteration failure)

Application-specific message errors

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2

- **Corrupted ...**
- **Omitted ...**

Msg. M_3

- **Corrupted ...**
- **Omitted ...**

Abstraction

Group identical messages into a single error event

Goal: P_{UB} > Pr (iteration failure)

↓

Application-specific message errors

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2

- **Corrupted …**
- **Omitted …**

Msg. M_3

- **Corrupted …**
- **Omitted …**

Abstraction

Group identical messages into a single error event

Error event E_1

Round 1 messages sent by Π_1 omitted at source

Goal: P_{UB} > Pr (iteration failure)

↓

Application-specific message errors

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Abstraction

Group identical messages into a single error event

Msg. M_2

- **Corrupted …**
- **Omitted …**

Msg. M_3

- **Corrupted …**
- **Omitted …**

Goal: P_{UB} > Pr (iteration failure)

Application-specific message errors

Error event E_1
Round 1 messages sent by Π_1 omitted at source

Error event E_2
Round 1 messages sent by Π_1 corrupted at source

Network error event E_3
Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

…

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. **M_1**

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. **M_2**

- **Corrupted …**
- **Omitted …**

Msg. **M_3**

- **Corrupted …**
- **Omitted …**

Abstraction

Group identical messages into a single error event

Error event E_1
Round 1 messages sent by Π_1 omitted at source

Error event E_2
Round 1 messages sent by Π_1 corrupted at source

Network error event E_3
Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

…

Goal: P_{UB} > Pr (iteration failure)

↓

Application-specific message errors

↓

For each E_i, derive **upper bound $P_i \geq Pr(E_i)$**

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1
- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2
- **Corrupted ...**
- **Omitted ...**

Msg. M_3
- **Corrupted ...**
- **Omitted ...**

Abstraction

Group identical messages into a single error event

Error event E_1
Round 1 messages sent by Π_1 omitted at source

Error event E_2
Round 1 messages sent by Π_1 corrupted at source

Network error event E_3
Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

...

Goal: P_{UB} > Pr (iteration failure)

↓

Application-specific message errors

↓

For each E_i, derive **upper bound $P_i \geq Pr(E_i)$**

↓

Basic Errors + Poisson Model + Timing Analysis

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2

- **Corrupted ...**
- **Omitted ...**

Msg. M_3

- **Corrupted ...**
- **Omitted ...**

Abstraction

Group identical messages into a single error event

Error event E_1
Round 1 messages sent by Π_1 omitted at source

Error event E_2
Round 1 messages sent by Π_1 corrupted at source

Network error event E_3
Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

...

Example!

Goal: P_{UB} > Pr (iteration failure)

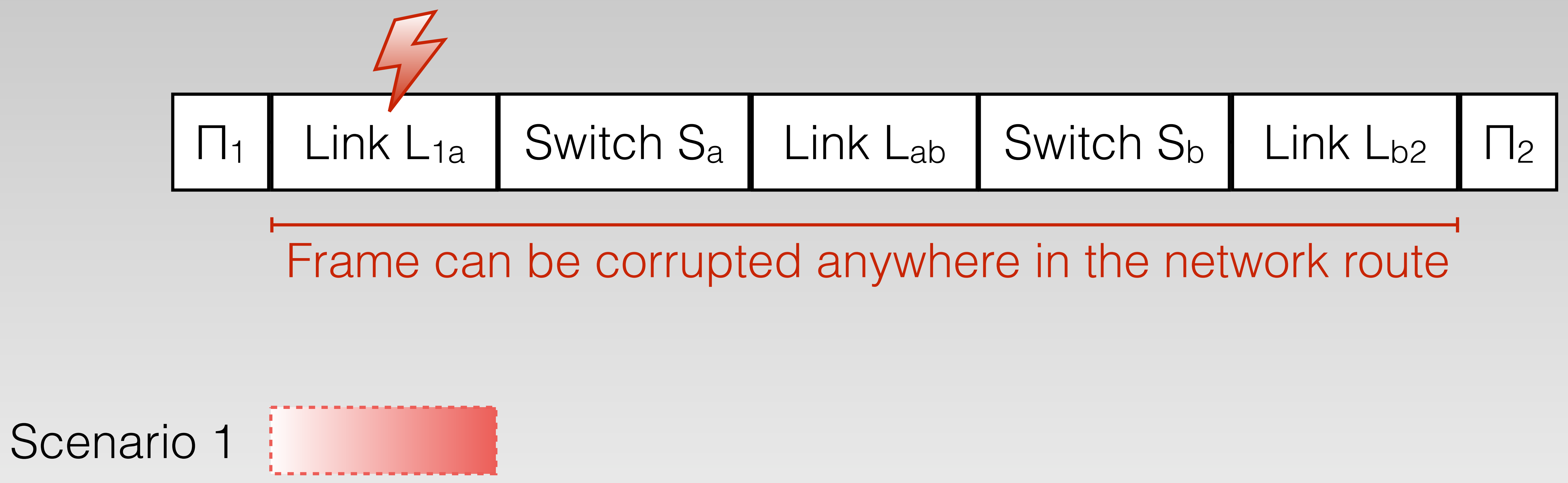
Application-specific message errors

For each E_i, derive **upper bound $P_i \geq Pr(E_i)$**

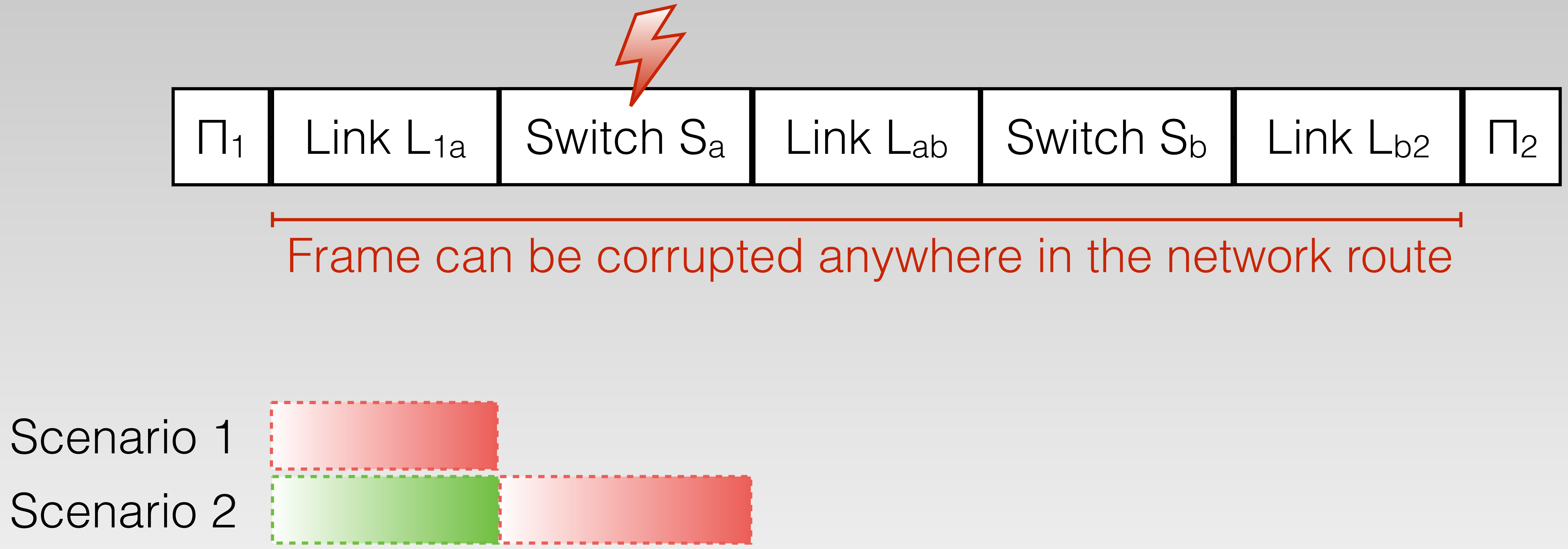
Basic Errors + Poisson Model + Timing Analysis

Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network

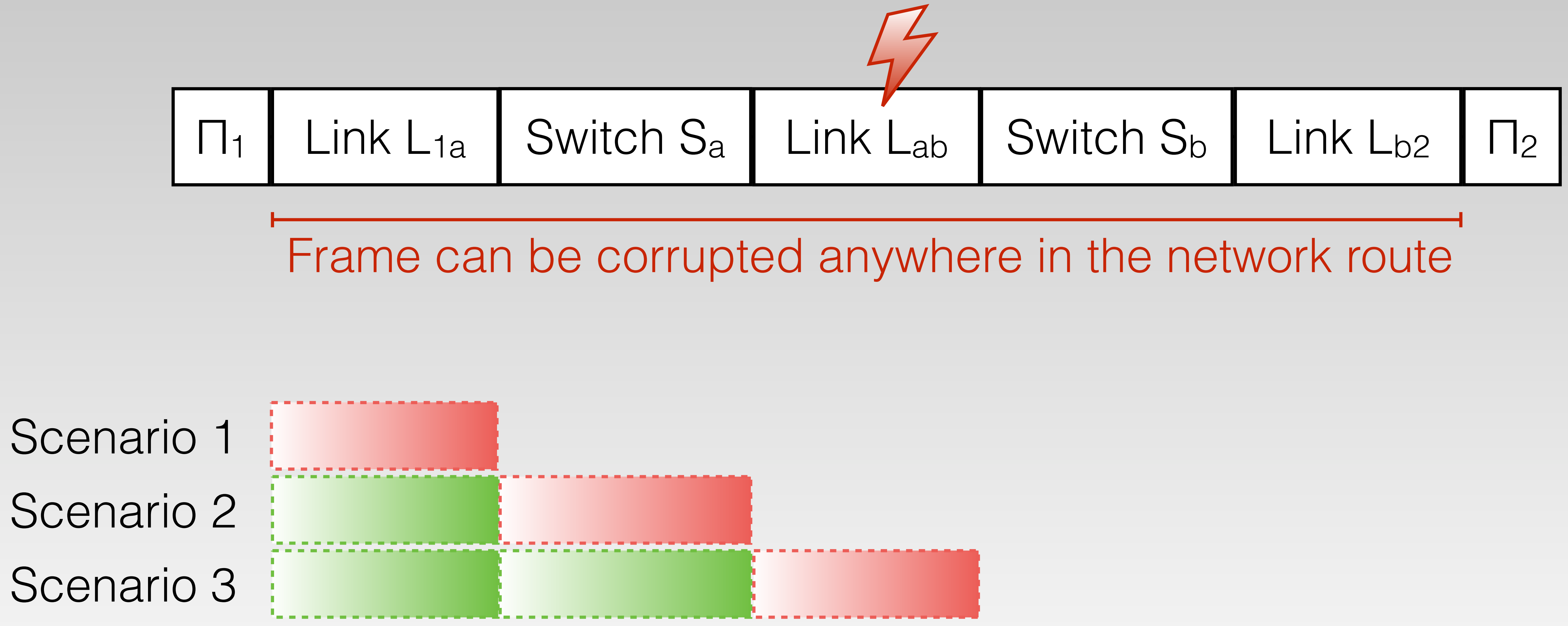
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



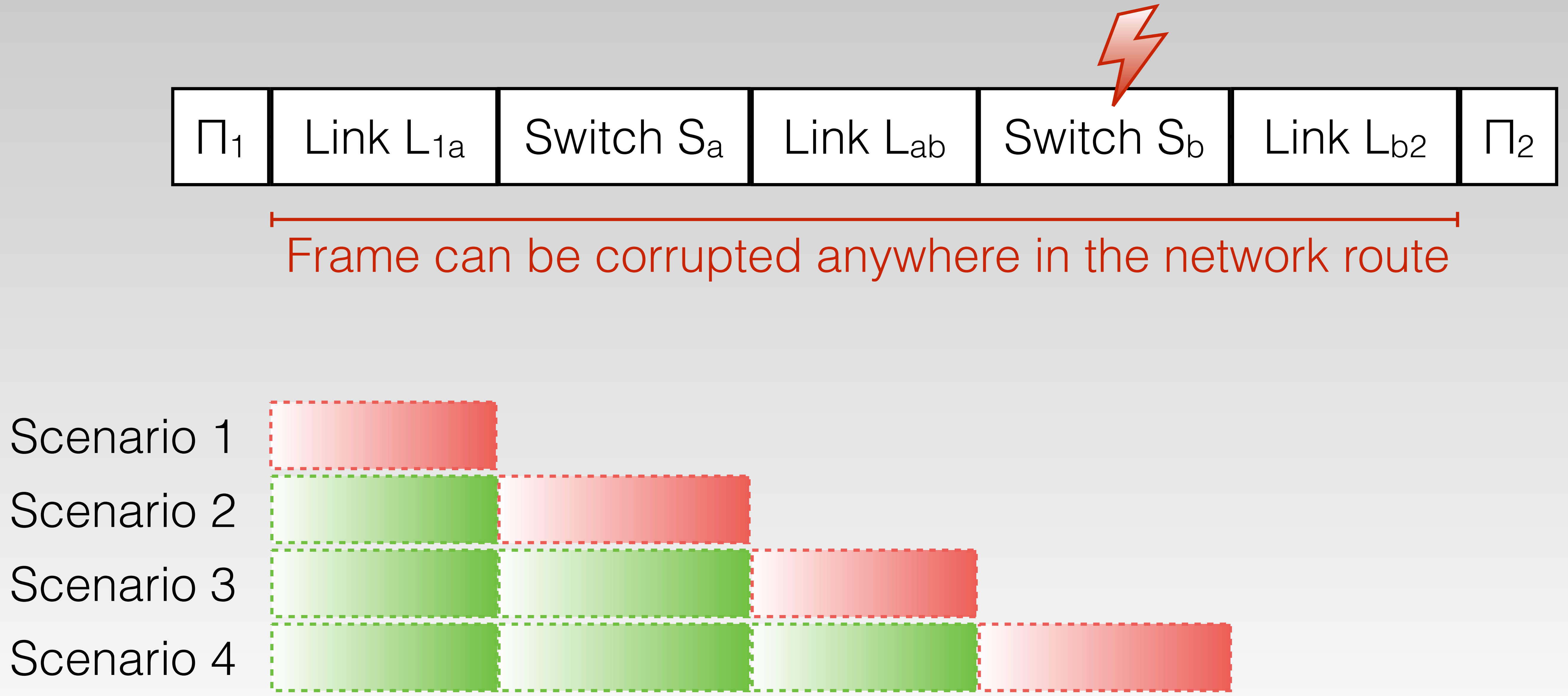
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



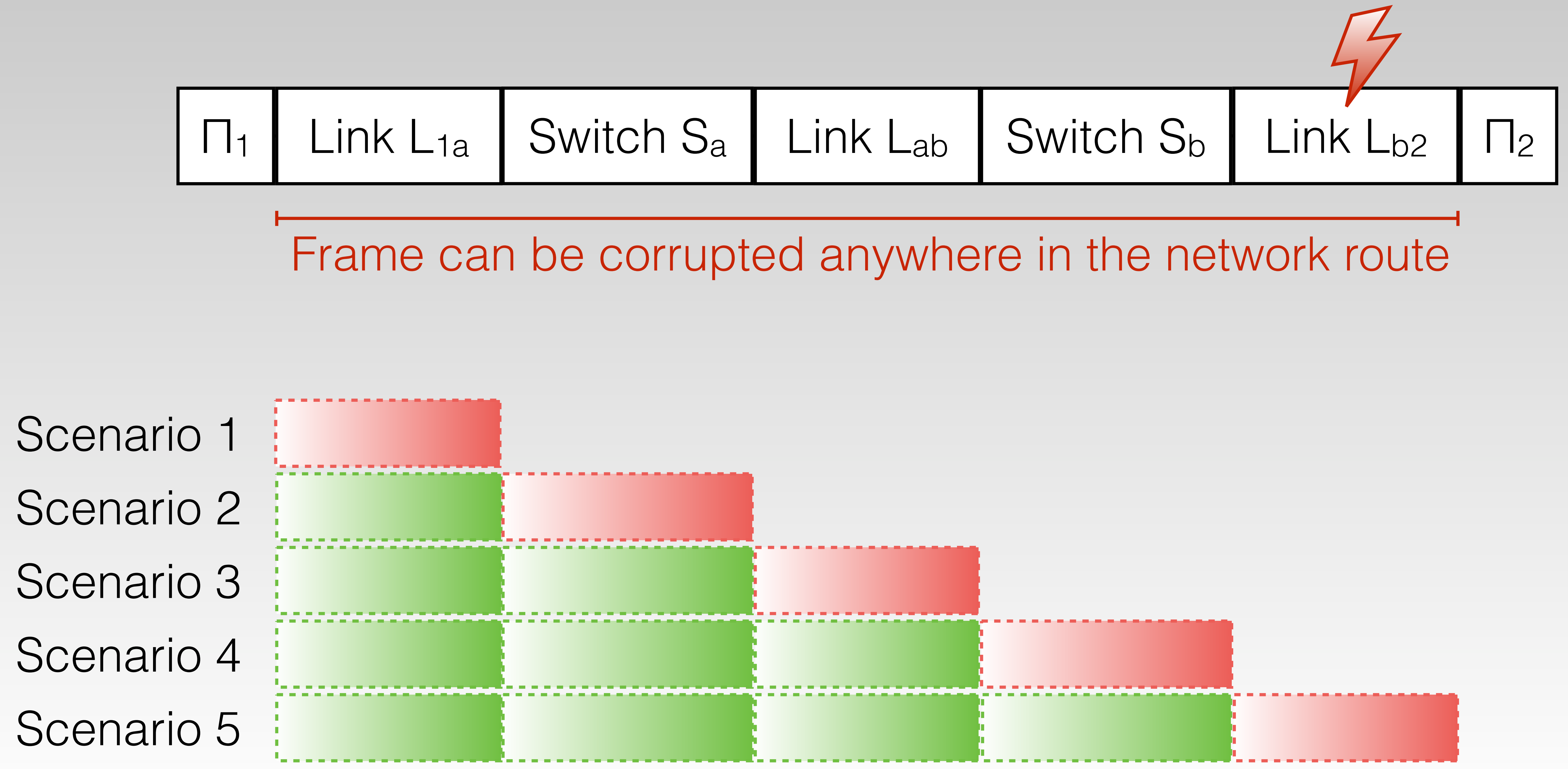
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



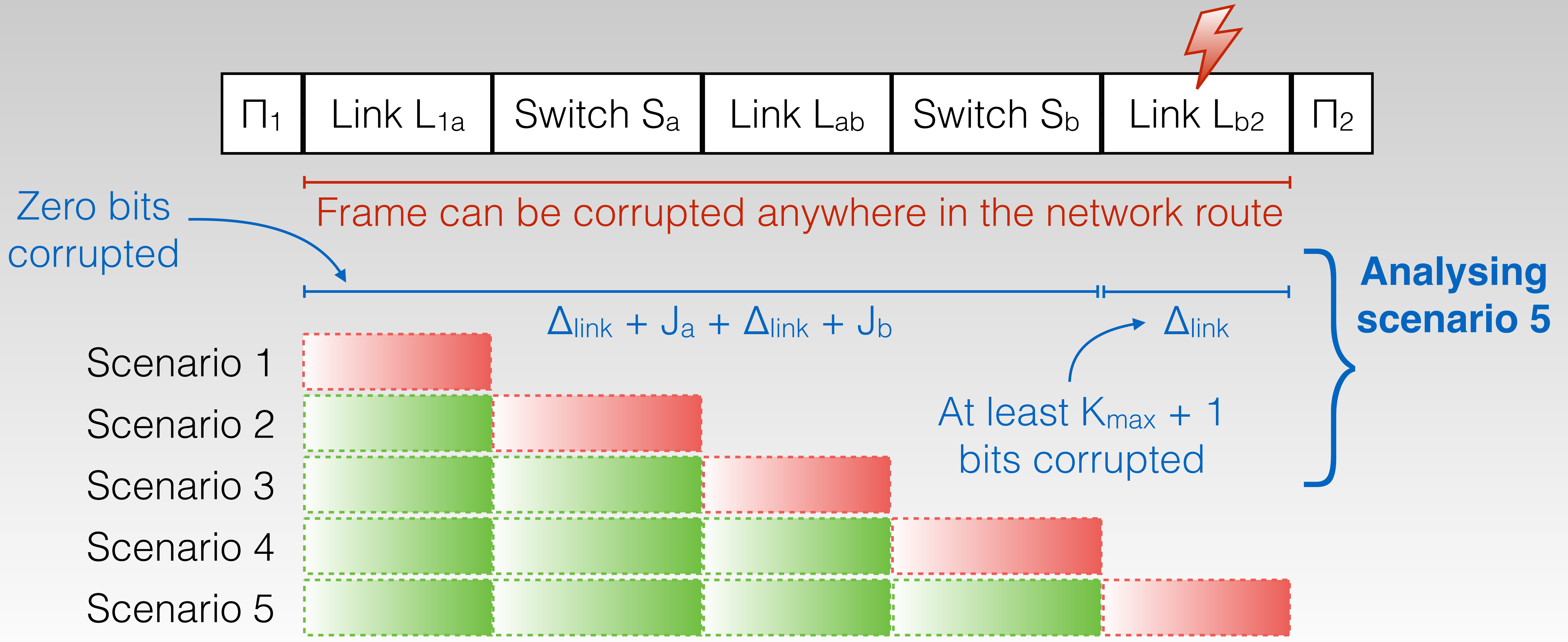
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



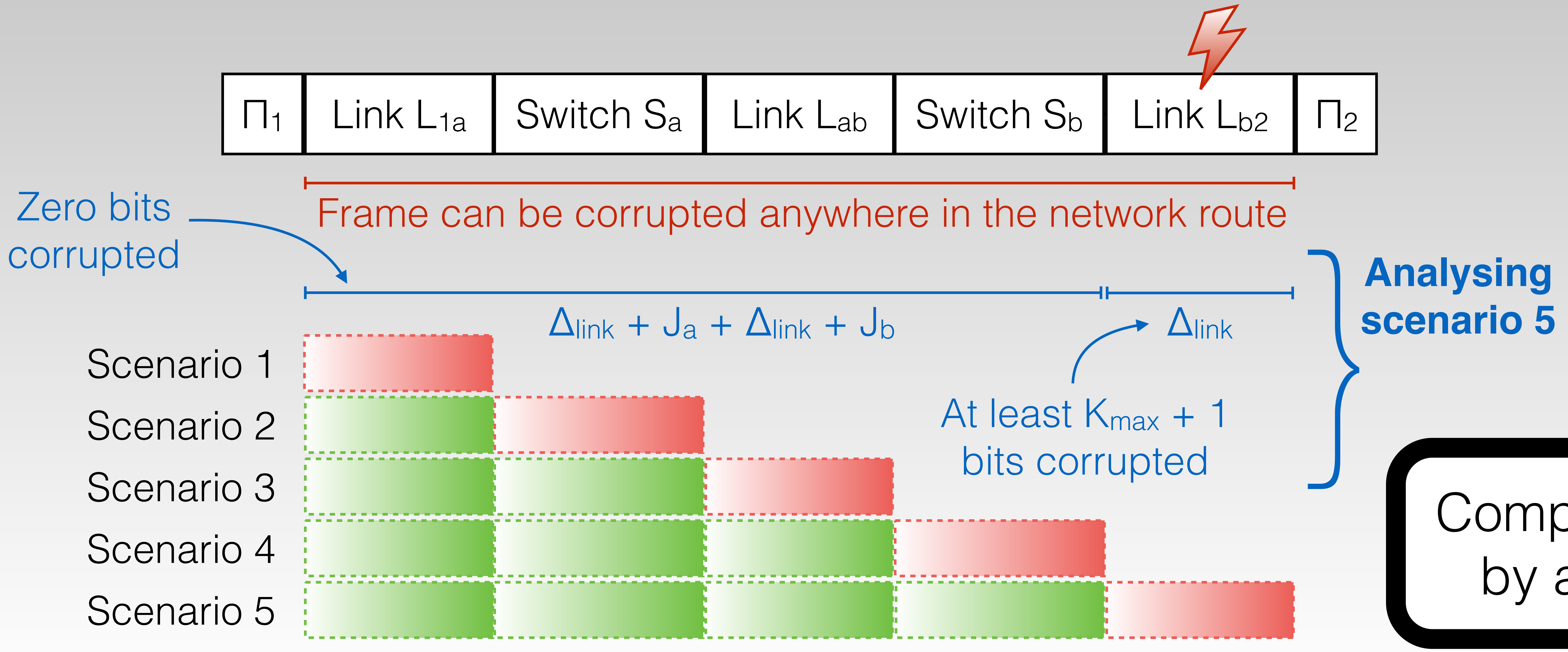
Notation

Δ_{link} = Link transfer time

J_a = Maximum scheduling jitter at Switch S_a

K_{max} = Maximum bit flips detected by the CRC

Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



Notation

Δ_{link} = Link transfer time

J_a = Maximum scheduling jitter at Switch S_a

K_{max} = Maximum bit flips detected by the CRC

Compute upper bound $\mathbf{P_3 \geq Pr(E_3)}$ by accounting for all scenarios

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. M_1

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Msg. M_2

- **Corrupted ...**
- **Omitted ...**

Msg. M_3

- **Corrupted ...**
- **Omitted ...**

Abstraction

Group identical messages into a single error event

Error event E_1

Round 1 messages sent by Π_1 omitted at source

Error event E_2

Round 1 messages sent by Π_1 corrupted at source

Network error event E_3

Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

...

Goal: P_{UB} > Pr (iteration failure)

Application-specific message errors

For each E_i, derive **upper bound $P_i \geq Pr(E_i)$**

Combinatorial analysis

$$\left(\begin{array}{l} P_1 \cdot C_1 \\ +\overline{P_1} \cdot P_2 \cdot C_2 \\ +\overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5 \cdot C_5 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot P_6 \cdot C_6 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot C_7 \end{array} \right)$$

Basic Errors + Poisson Model + Timing Analysis

Key idea 1: Scalability through **abstraction** and **pruning**

Msg. **M_1**

- **Corrupted** at source, network, or destination
- **Omitted** at source or network

Abstraction

Group identical messages into a single error event

Msg. **M_2**

- **Corrupted ...**
- **Omitted ...**

Msg. **M_3**

- **Corrupted ...**
- **Omitted ...**

Error event E_1
Round 1 messages sent by Π_1 omitted at source

Error event E_2
Round 1 messages sent by Π_1 corrupted at source

Network error event E_3
Frame carrying round 1 messages from Π_1 to Π_2 corrupted by the network

...

Goal: P_{UB} > Pr (iteration failure)

Application-specific message errors

For each E_i, derive **upper bound $P_i \geq Pr(E_i)$**

Protocol-specific **pruning**

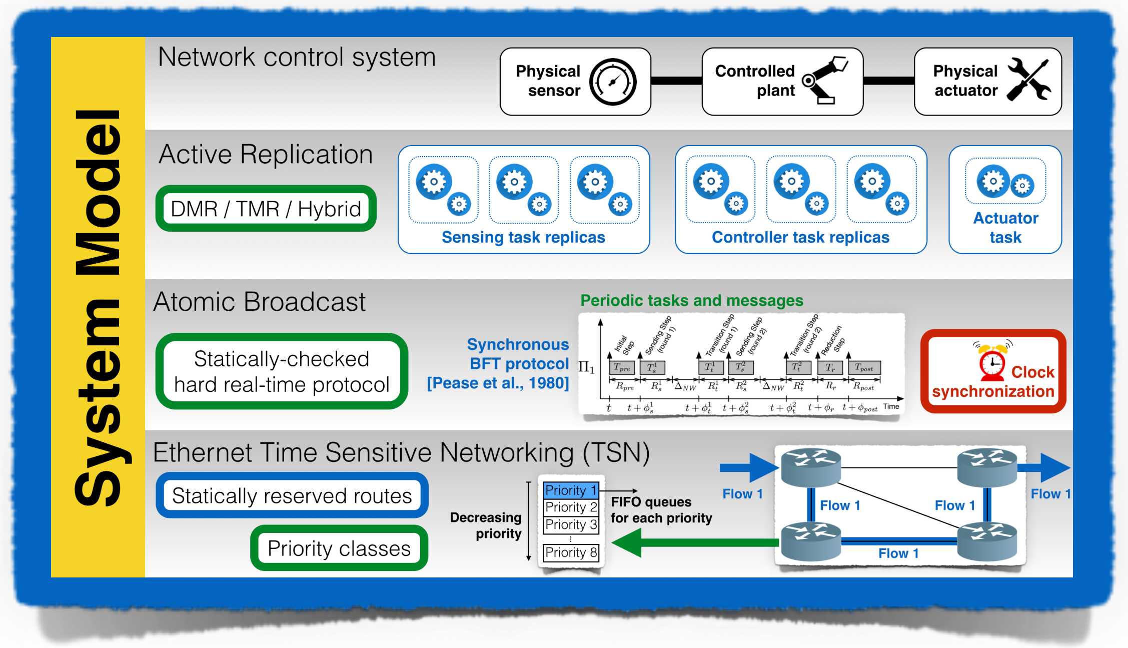
- Cannot omit a message twice
- Do not care if an omitted message is corrupted

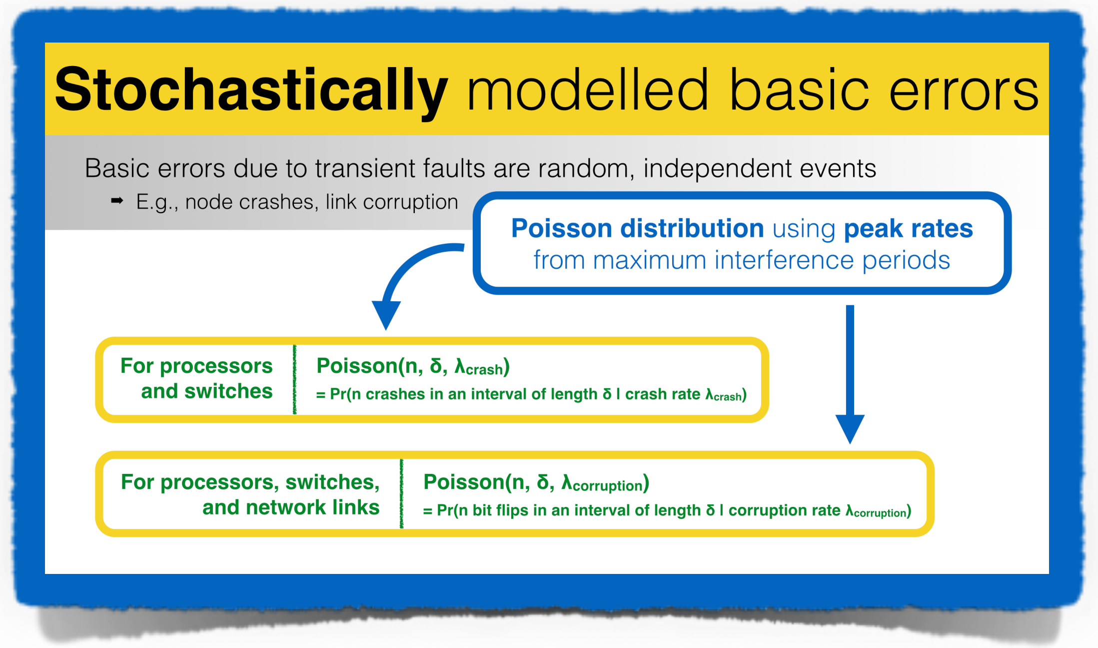
$$\left(\begin{array}{l} P_1 \cdot C_1 \\ +\overline{P_1} \cdot P_2 \cdot C_2 \\ +\overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5 \cdot C_5 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot P_6 \cdot C_6 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot C_7 \end{array}\right)$$

Combinatorial analysis

Basic Errors + Poisson Model + Timing Analysis

Key idea 1: Scalability through **abstraction** and **pruning**





Model checking or simulation

Pr (iteration failure)

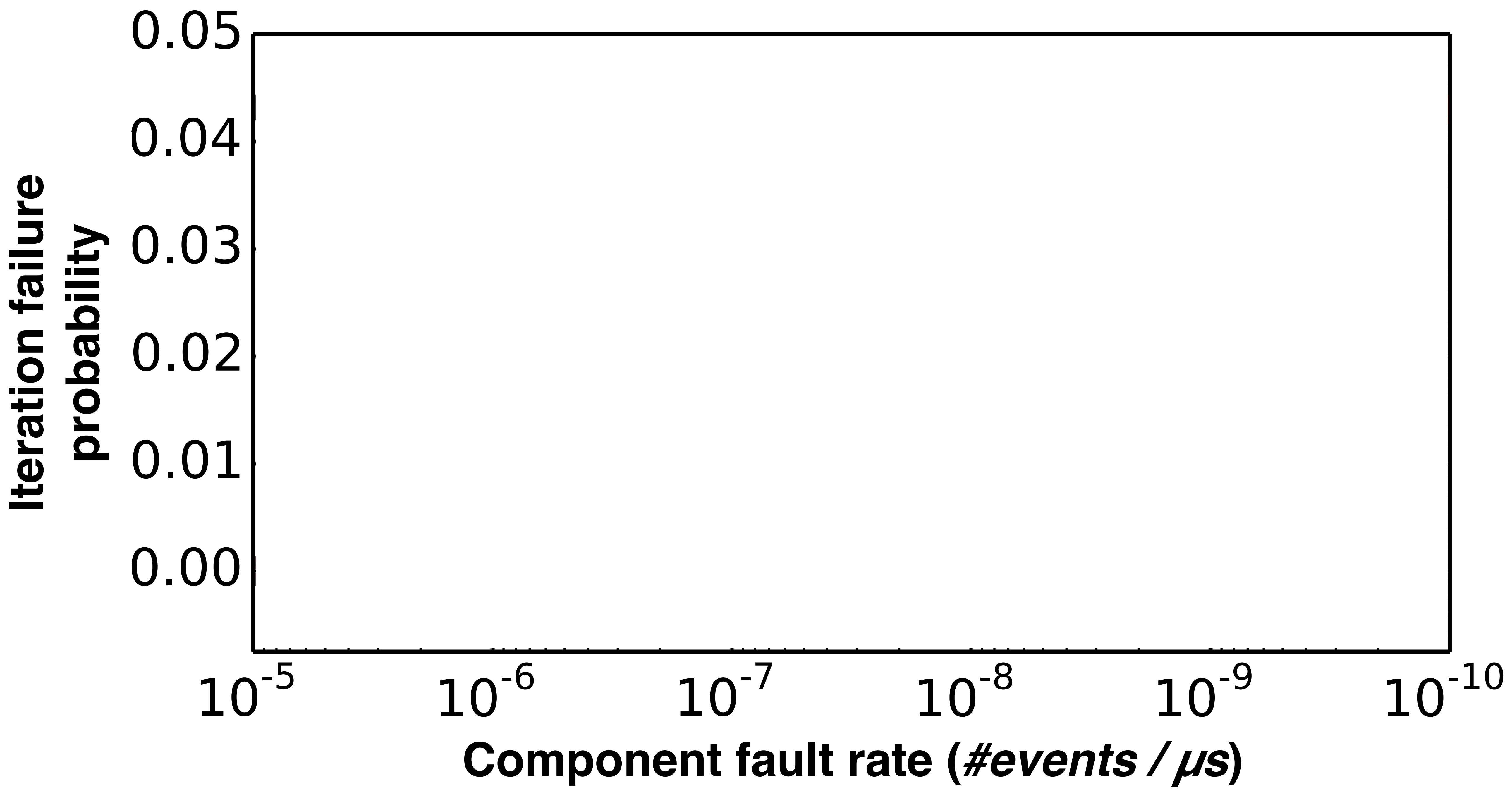
Scalability challenges

Key idea 1: Tackle scalability through *abstraction and pruning*

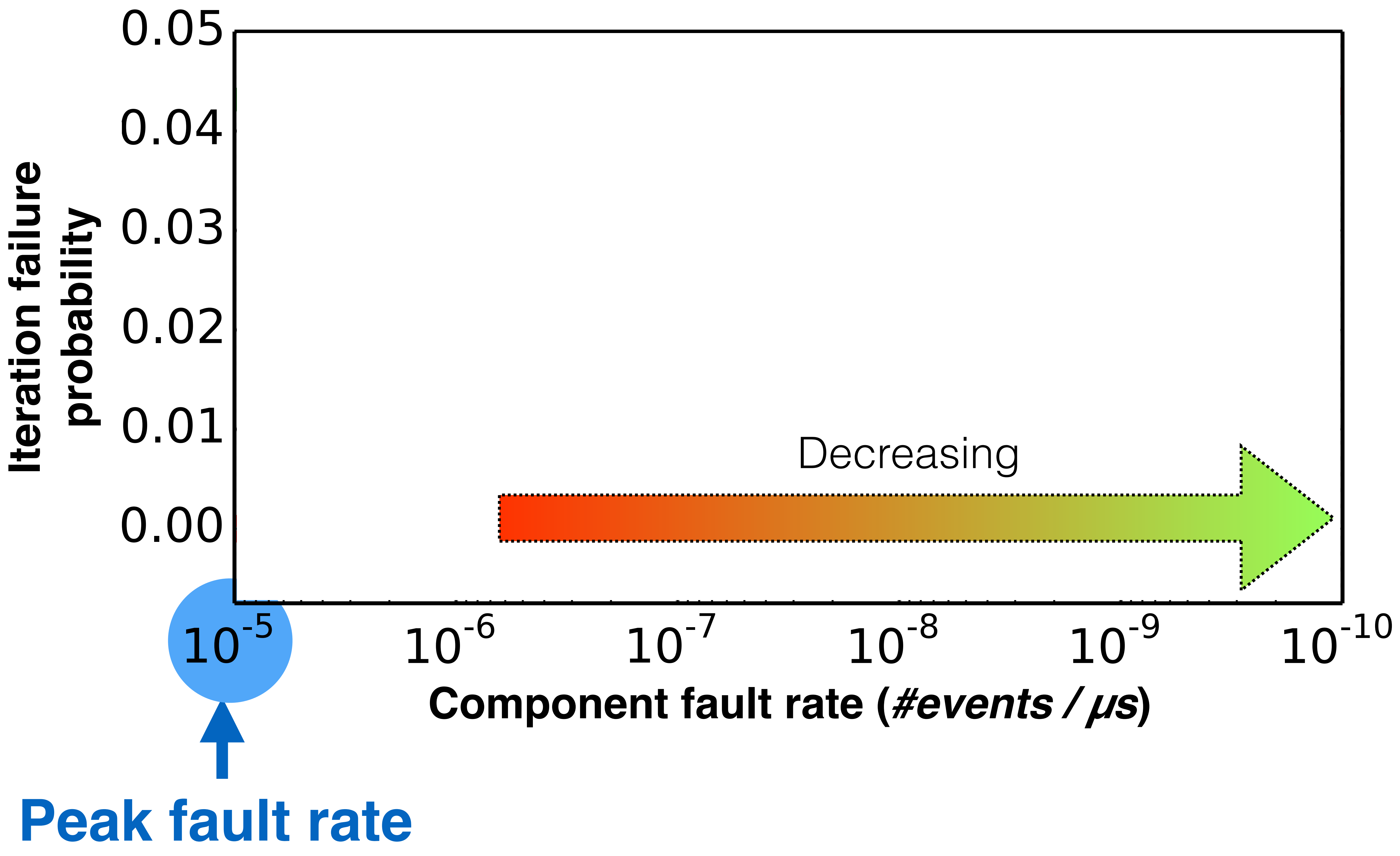
Reliability anomalies

- In practice, the iteration failure probability may **significantly exceed** the estimated Pr (iteration failure)

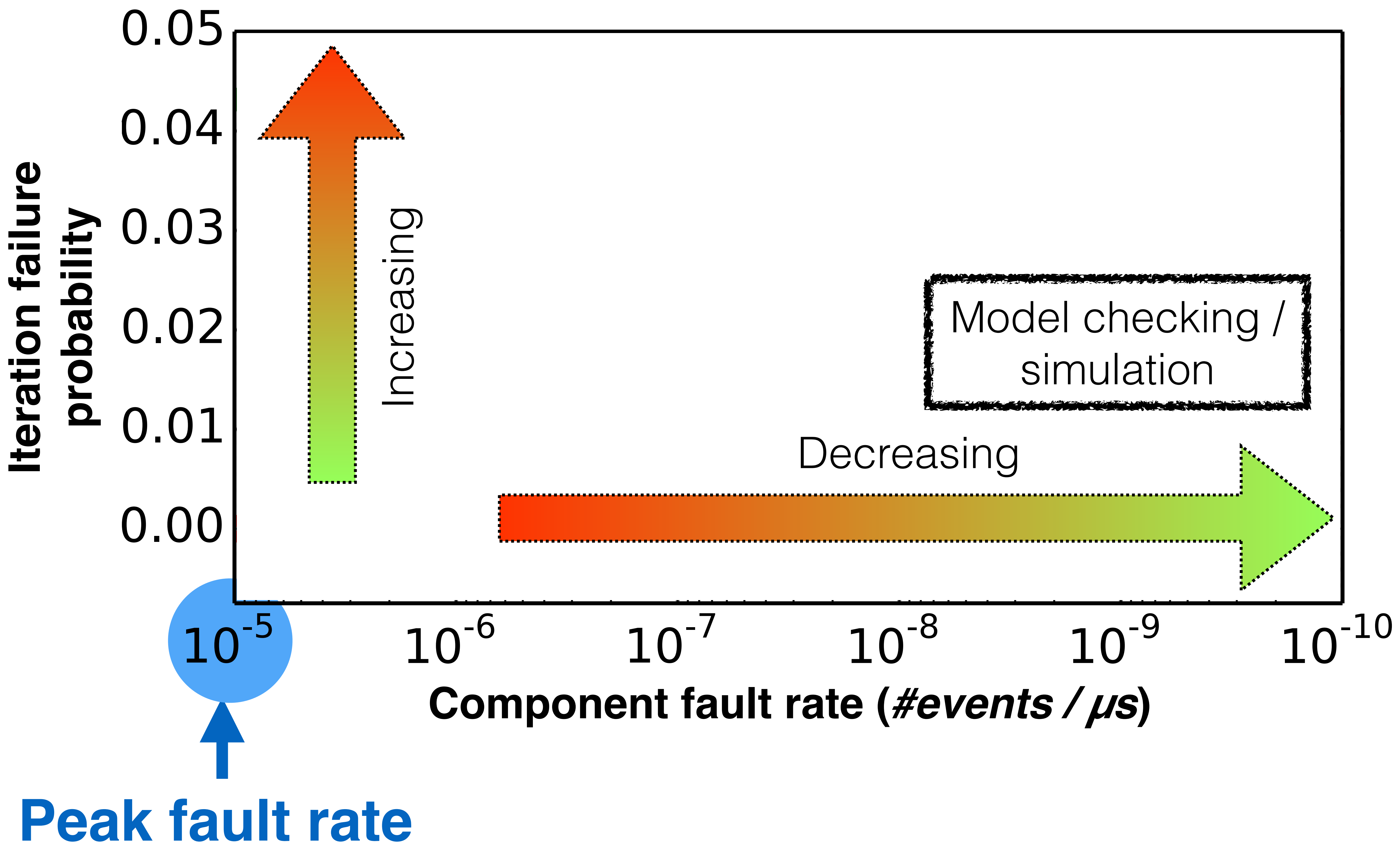
The problem of **reliability anomalies**



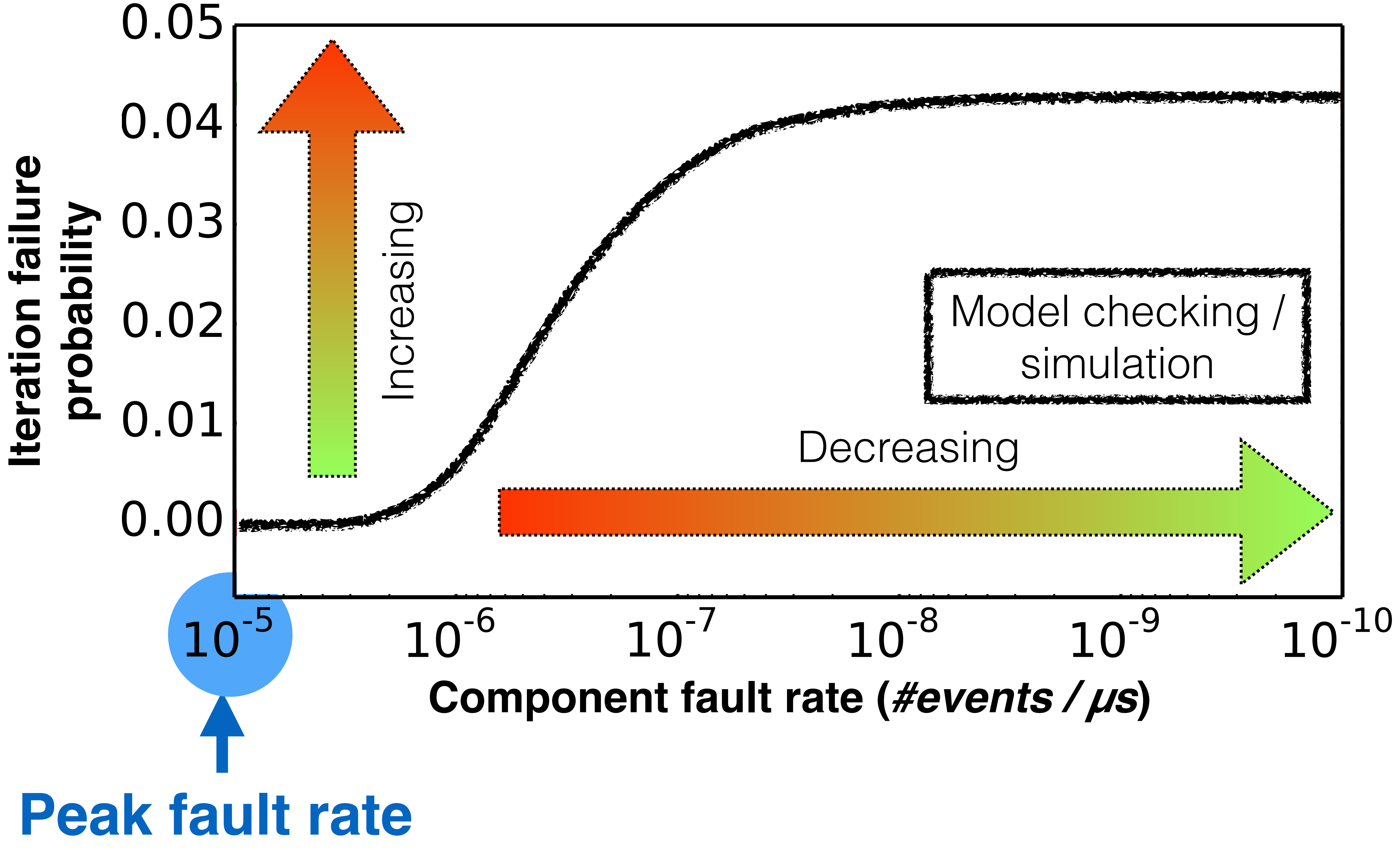
The problem of **reliability anomalies**



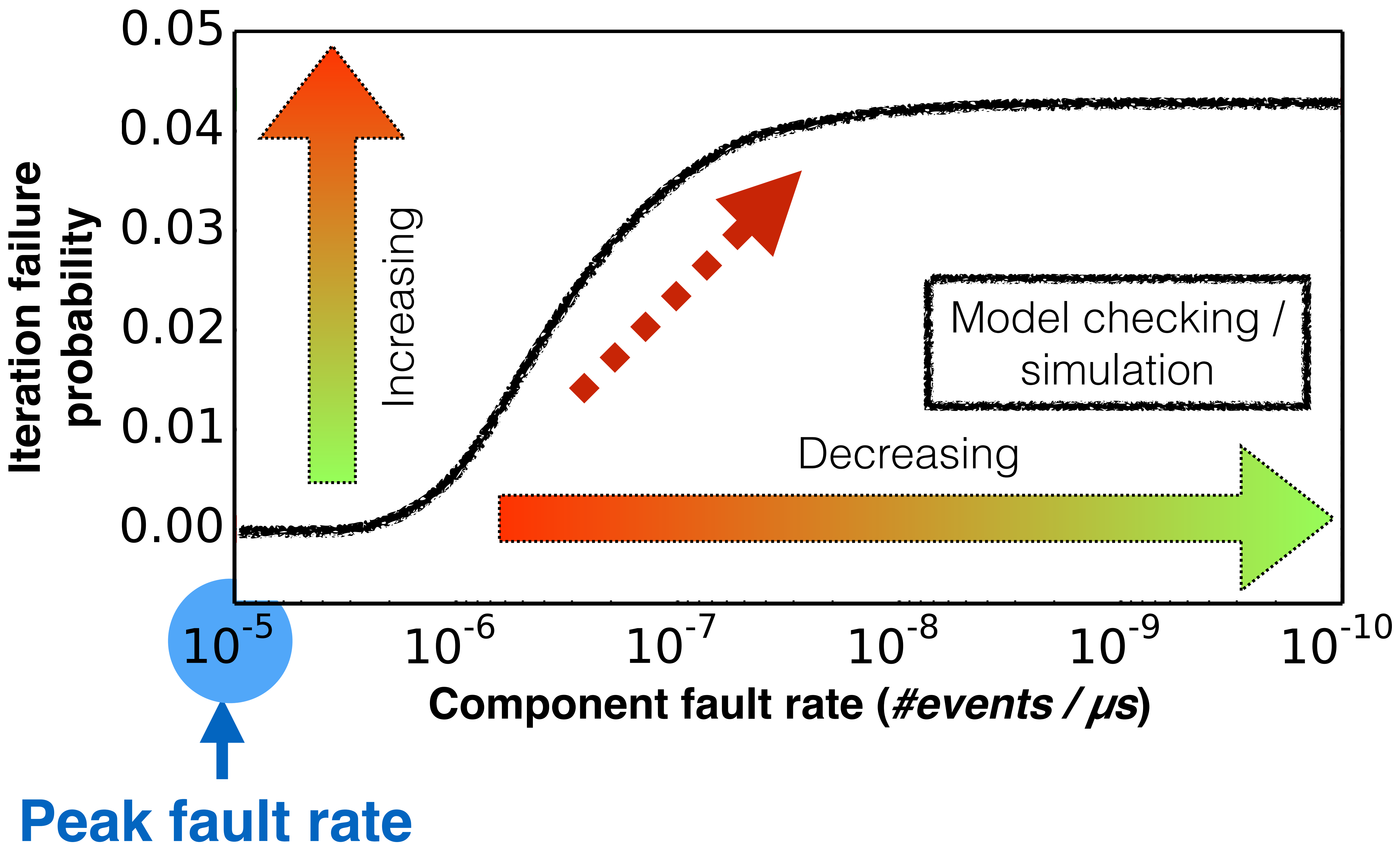
The problem of **reliability anomalies**



The problem of **reliability anomalies**



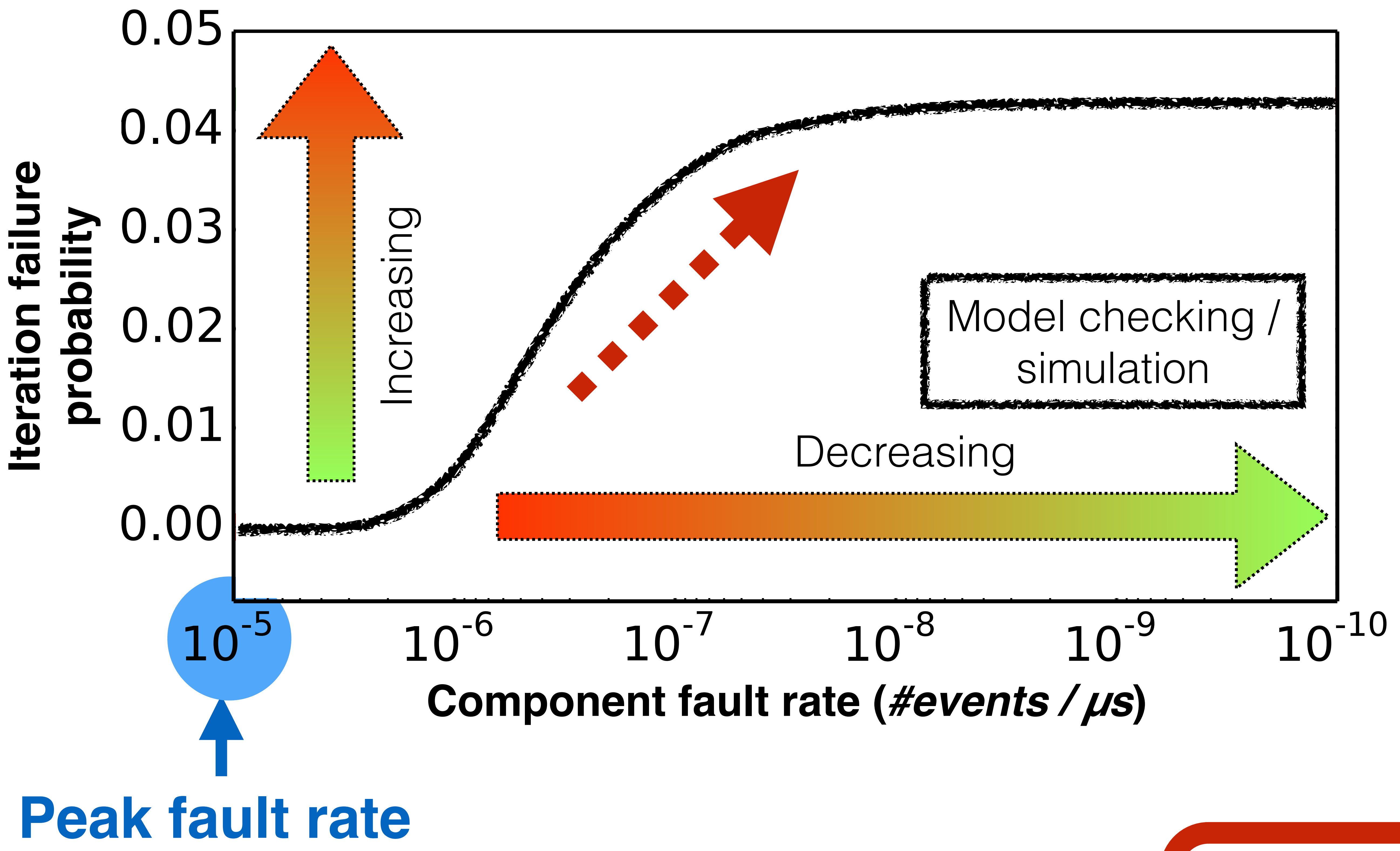
The problem of **reliability anomalies**



Pr (iteration failure) increases despite decreasing component fault rate

Intuition: *Sometimes, a node crash is good for the overall system, because it may reduce the probability of confusing a majority voting protocol in another part of the system!*

The problem of **reliability anomalies**



Pr (iteration failure) increases despite decreasing component fault rate

Intuition: *Sometimes, a node crash is good for the overall system, because it may reduce the probability of confusing a majority voting protocol in another part of the system!*

For soundness, need to estimate failure probabilities for the **entire search space [0, 10^{-5}]**

Key idea 2: Ensure **monotonicity** to eliminate anomalies

Combinatorial analysis

$$P_{\mathrm{UB}} = \left(\begin{array}{l} P_1 \cdot C_1 \\ + \overline{P_1} \cdot P_2 \cdot C_2 \\ + \overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ + \overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ + \overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5 \cdot C_5 \\ + \overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot P_6 \cdot C_6 \\ + \overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot C_7 \end{array} \right)$$

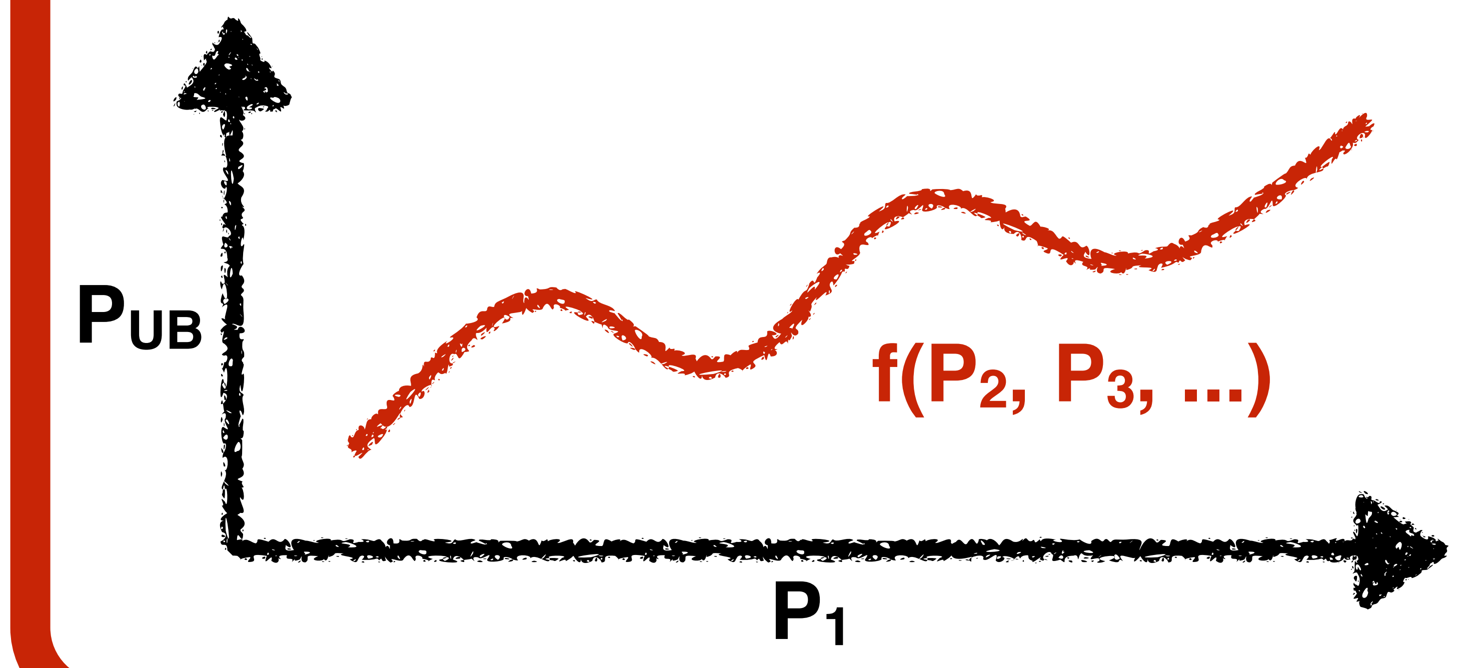
Key idea 2: Ensure **monotonicity** to eliminate anomalies

Combinatorial analysis

$$P_{\text{UB}} = \begin{pmatrix} P_1 \cdot C_1 \\ +\overline{P_1} \cdot P_2 \cdot C_2 \\ +\overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5 \cdot C_5 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot P_6 \cdot C_6 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot C_7 \end{pmatrix}$$

Root cause of reliability anomalies

P_{UB} may not monotonically increase with P_1, P_2, P_3, ...



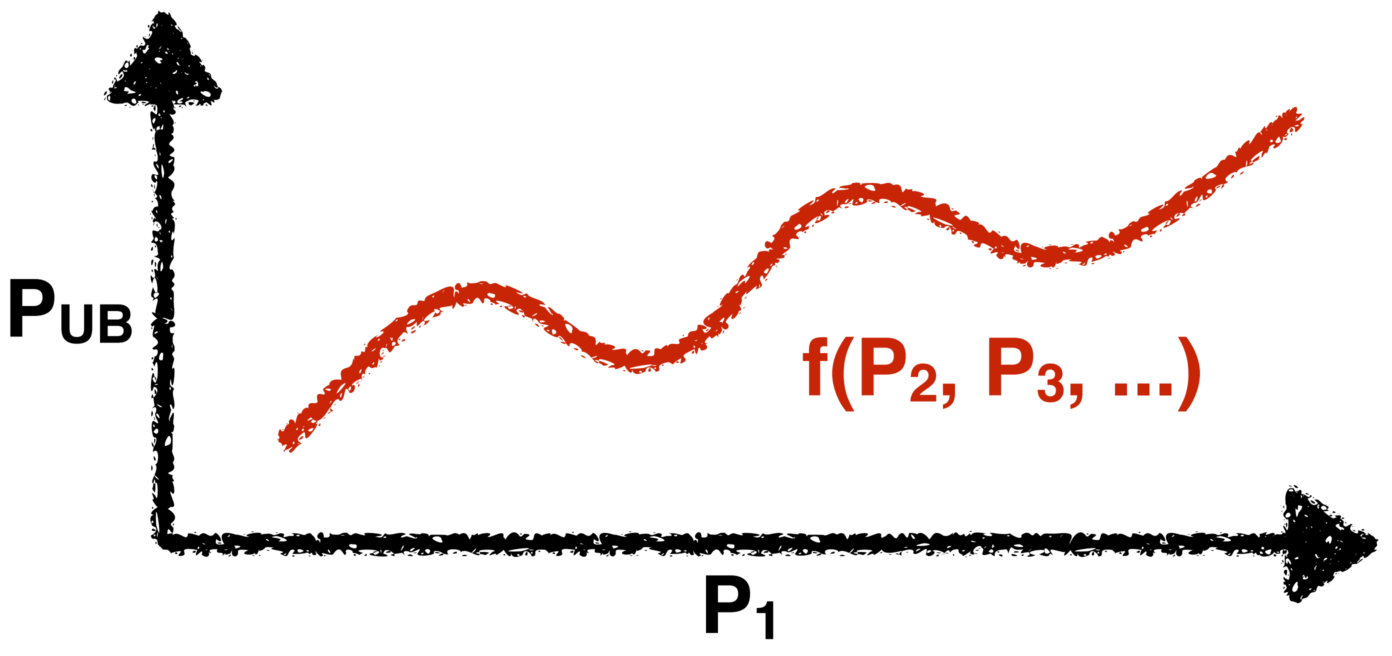
Key idea 2: Ensure **monotonicity** to eliminate anomalies

Combinatorial analysis

$$P_{UB} = \begin{pmatrix} P_1 \cdot C_1 \\ +\overline{P_1} \cdot P_2 \cdot C_2 \\ +\overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5 \cdot C_5 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot P_6 \cdot C_6 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot \overline{P_5} \cdot \overline{P_6} \cdot C_7 \end{pmatrix}$$

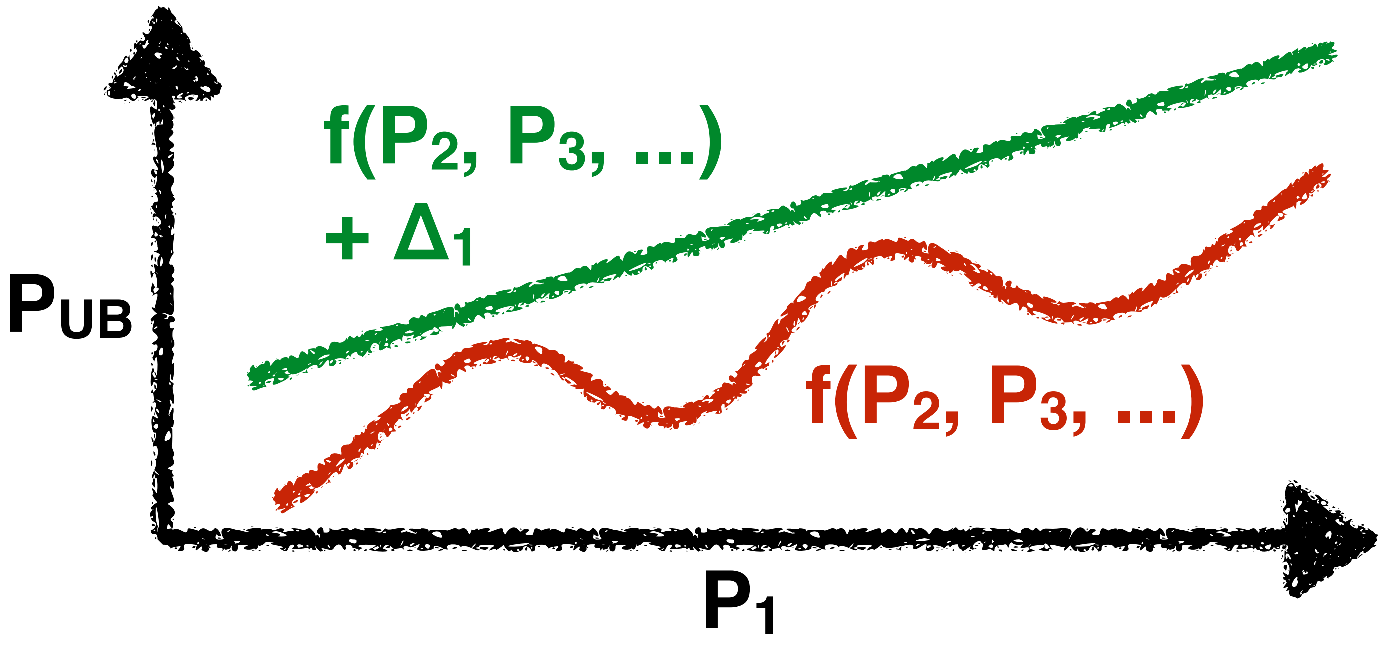
Root cause of reliability anomalies

P_{UB} may not monotonically increase with P_1, P_2, P_3, ...

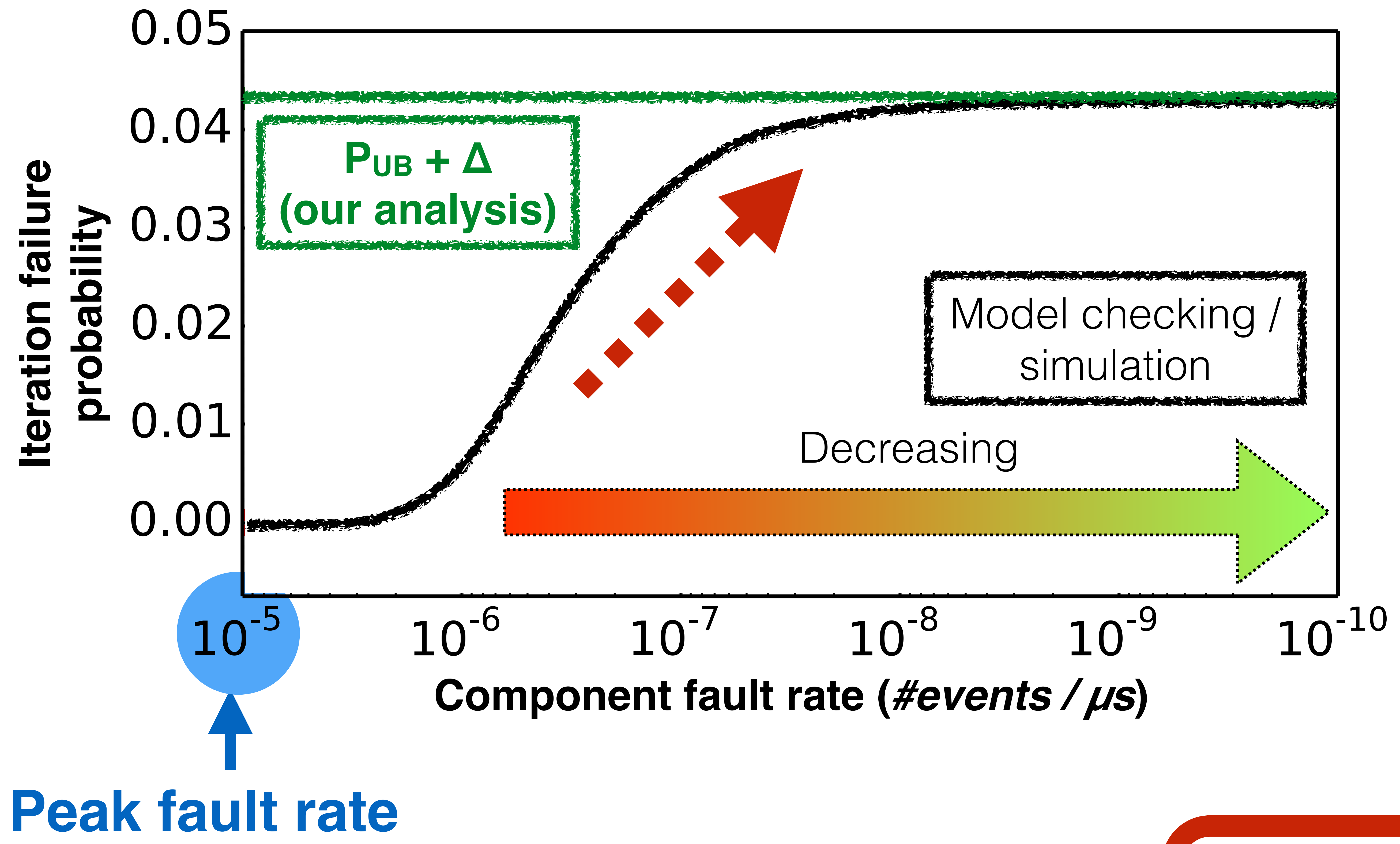


Eliminating reliability anomalies

A "fudge factor" Δ is added to P_{UB} to ensure that P_{UB} + Δ is monotonically increasing with P_1, P_2, P_3, ...



The problem of **reliability anomalies**

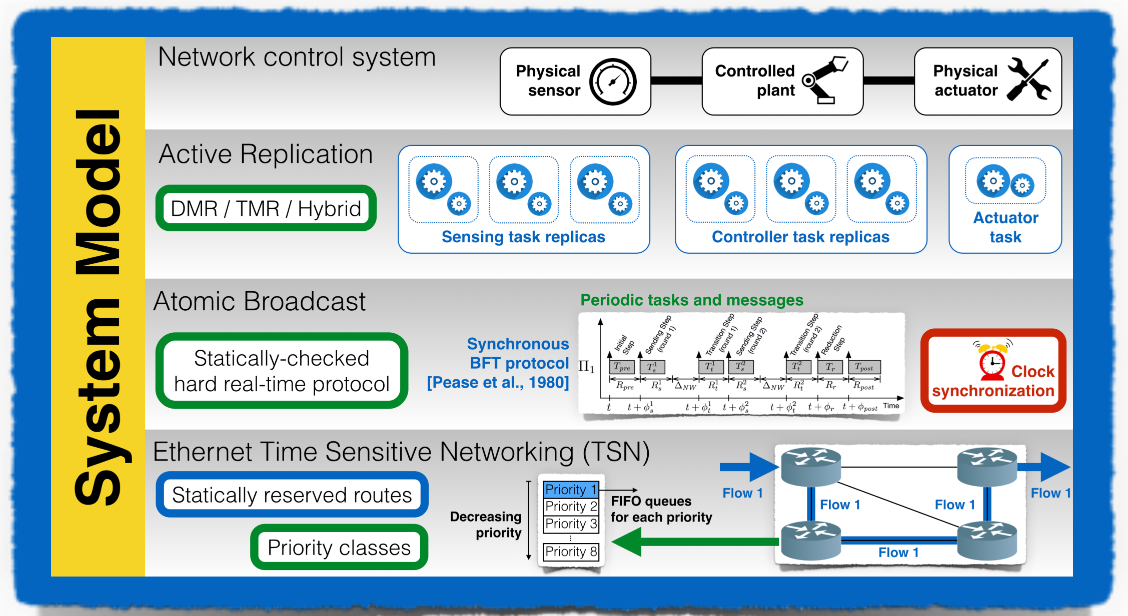


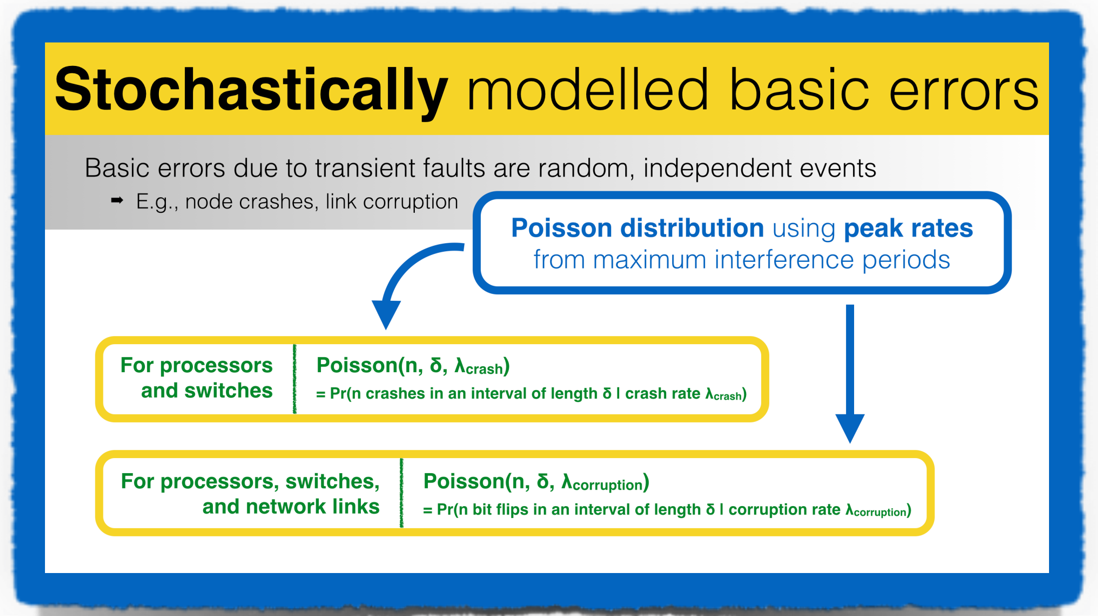
Pr (iteration failure) increases despite decreasing component fault rate

Intuition: *Sometimes, a node crash is good for the overall system, because it may reduce the probability of confusing a majority voting protocol in another part of the system!*

For soundness, need to estimate failure probabilities for the **entire search space [0, 10^{-5}]**

Key idea 2: Ensure **monotonicity** to eliminate anomalies





Model checking or simulation

Pr (iteration failure)

Scalability challenges

Key idea 1: Tackle scalability through *abstraction and pruning*

Reliability anomalies

Key idea 2: Ensure *monotonicity* to eliminate anomalies

System Model

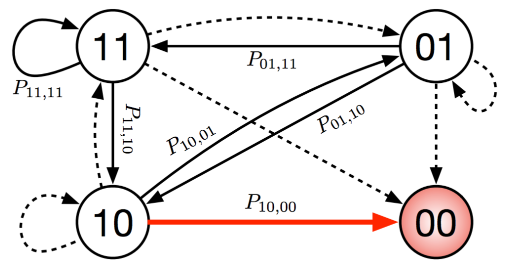
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Q2. What is the likelihood of a **control failure?**

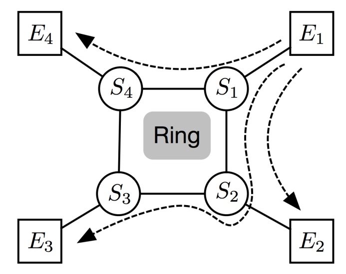
Periodic Weakly-Hard Systems [ECRTS '19]



Transient fault-induced errors

Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



System Model

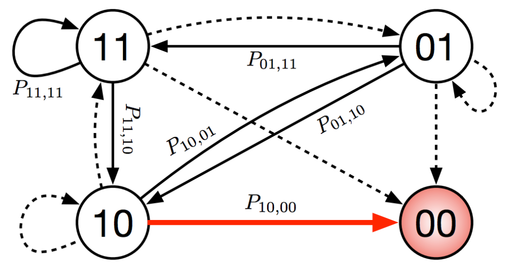
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Q2. What is the likelihood of a **control failure?**

Periodic Weakly-Hard Systems [ECRTS '19]



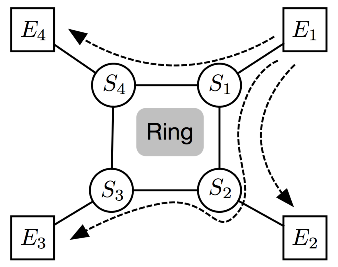
Failures-In-Time (FIT)

- Expected # failures in one billion operating hours

Transient fault-induced errors

Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure) ⟶ $FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure) - - - - - - - - - - -> **$FIT_{UB} > FIT_{control\text{-}system}$**

Assumption:

1^{st} iteration failure —> control system failure

Failures-In-Time

$$\mathbf{FIT} = \frac{\mathbf{10^9}}{\mathbf{MTTF}\ \text{(in hours)}}$$

Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure) → $FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Mean Time To Failure

$$MTTF = T \times \sum_{n=0}^{\infty} n \times f(n)$$

Time-period

Iteration #

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Simplistic approach

$P_{UB} + \Delta > Pr\text{(iteration failure)}$

$FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = \left(1 - Pr\left(\begin{smallmatrix}\text{iteration}\\\text{failure}\end{smallmatrix}\right)\right)^{n-1} \times Pr\left(\begin{smallmatrix}\text{iteration}\\\text{failure}\end{smallmatrix}\right)$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Simplistic approach

$P_{UB} + \Delta > Pr\,(\text{ iteration failure })$ ⟶ $FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = \left(1 - Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)\right)^{n-1} \times Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Example

- Networked control system operates at **100 Hz (T = 10 ms)**
- Iteration failure probability upper-bounded by $\mathbf{P_{UB} + \Delta = 10^{-10}}$

Simplistic approach

$P_{UB} + \Delta > Pr(\text{ iteration failure })$

$FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = \left(1 - Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)\right)^{n-1} \times Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF\ (\text{in hours})}$$

Example

- Networked control system operates at **100 Hz (T = 10 ms)**
- Iteration failure probability upper-bounded by $\mathbf{P_{UB} + \Delta = 10^{-10}}$

MTTF = 108 seconds

FIT = 36000

Pessimistic!

Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure)

$FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = \left(1 - Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)\right)^{n-1} \times Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)$$

Mean Time To Failure

$$MTTF = T \times \sum_{n=0}^{\infty} n \times f(n)$$

Time-period

Iteration #

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Example

- Networked control system operates at **100 Hz (T = 10 ms)**
- Iteration failure probability upper-bounded by **$P_{UB} + \Delta = 10^{-10}$**

MTTF = 108 seconds

Pessimistic!

FIT = 36000

Well-designed robust controllers

The controller **tolerates** one iteration failure every four consecutive iterations

Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure) → $FIT_{UB} > FIT_{control\text{-}system}$

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = \left(1 - Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)\right)^{n-1} \times Pr\left(\begin{smallmatrix}\text{iteration}\\ \text{failure}\end{smallmatrix}\right)$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Example

- Networked control system operates at **100 Hz (T = 10 ms)**
- Iteration failure probability upper-bounded by $\mathbf{P_{UB} + \Delta = 10^{-10}}$

→ **MTTF = 108 seconds** → **FIT = 36000**

Pessimistic!

Well-designed robust controllers → The controller **tolerates** one iteration failure every four consecutive iterations → $\mathbf{FIT = 1.08 \times 10^{-5}}$

More accurate

Simplistic approach

$P_{UB} + \Delta > Pr$ (iteration failure) → **$FIT_{UB} > FIT_{control\text{-}system}$**

Assumption:

1st iteration failure ➡ control system failure

Probab

f(n) =

Mean Time To Failure

ours)

How to compute **FIT_{UB} from $P_{UB} + \Delta$** while taking into account ***temporal robustness*** properties of **well-designed controllers?**

Example

Pessimistic!

- Networked control system operates at **100 Hz (T = 10 ms)**
- Iteration failure probability upper-bounded by **$P_{UB} + \Delta = 10^{-10}$**

→ **MTTF = 108 seconds** → **FIT = 36000**

Well-designed robust controllers → The controller **tolerates** one iteration failure every four consecutive iterations → **FIT = 1.08×10^{-5}**

More accurate

Weakly-hard constraints*

* Bernat, Burns, and Liamosi. "Weakly hard real-time systems." IEEE transactions on Computers 50.4 (2001).

Weakly-hard constraints*

Example: (m, k) constraint

- At least m out of every k consecutive iterations must be successful

* Bernat, Burns, and Liamosi. “Weakly hard real-time systems.” IEEE transactions on Computers 50.4 (2001).

Weakly-hard constraints*

Example: (m, k) constraint

- At least m out of every k consecutive iterations must be successful
- If each iteration is labeled either as a **S**uccess or a **F**ailure

Temporal robustness as per (2, 3) constraint

Robust Iterations **Robustness violation**

S S S F S S S S F S F S

---

* Bernat, Burns, and Liamosi. "Weakly hard real-time systems." IEEE transactions on Computers 50.4 (2001).

Sound Approximation (**SAp**)

$P_{UB} + \Delta > Pr\,(\text{iteration failure})$ → $FIT_{UB} > FIT_{control\text{-}system}$

~~Assumption:~~

~~1st iteration failure —> control system failure~~

Probability density function

$$f(n) = (1 - \cancel{Pr(\text{iteration failure})})^{n-1} \times \cancel{Pr(\text{iteration failure})}$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Sound Approximation (**SAp**)

$P_{UB} + \Delta > Pr\,(\text{ iteration failure })$

$FIT_{UB} > FIT_{control\text{-}system}$

~~**Assumption:**~~
~~1st iteration failure~~ —> control system failure

Probability density function

$$f(n) = (1 \cancel{- Pr(\text{iteration failure})})^{n-1} \times \cancel{Pr}(\text{iteration failure})$$

Mean Time To Failure

$$MTTF = T \times \sum_{n=0}^{\infty} n \times f(n)$$

Time-period

Iteration #

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Obtain $f_{LB}(n) \leq f(n)$

- Accounting for **temporal robustness***
- That can be quickly computed for large n

* Sfakianakis et al.. "Reliability of a consecutive k-out-of-r-from-n: F system." IEEE Transactions on Reliability 41.3 (1992): 442-447.

Sound Approximation (**SAp**)

$P_{UB} + \Delta > Pr(\text{ iteration failure })$ → $FIT_{UB} > FIT_{control\text{-}system}$

~~**Assumption:**~~
~~1st iteration failure —> control system failure~~

Probability density function

$$f(n) = \cancel{(1 - Pr(\text{iteration failure}))^{n-1} \times Pr(\text{iteration failure})}$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Obtain $\mathbf{f_{LB}(n) \leq f(n)}$
- Accounting for **temporal robustness***
- That can be quickly computed for large n

Obtain $\mathbf{MTTF_{LB}}$
- Compute $f_{LB}(n_0), f_{LB}(n_1), \ldots, f_{LB}(n_D)$
- **Numerically integrate** over subintervals $(n_0, n_1], \ldots, (n_{D-1}, n_D]$

* Sfakianakis et al.. "Reliability of a consecutive k-out-of-r-from-n: F system." IEEE Transactions on Reliability 41.3 (1992): 442-447.

Sound Approximation (**SAp**)

$P_{UB} + \Delta > Pr$ (iteration failure) - - - - - - - - - - > **$FIT_{UB} > FIT_{control\text{-}system}$**

Assumption:

1st iteration failure —> control system failure

Probability density function

$$f(n) = (1 - \cancel{Pr(\text{iteration failure})})^{n-1} \times \cancel{Pr(\text{iteration failure})}$$

Mean Time To Failure

$$MTTF = \underbrace{T}_{\text{Time-period}} \times \sum_{n=0}^{\infty} \underbrace{n}_{\text{Iteration \#}} \times f(n)$$

Failures-In-Time

$$FIT = \frac{10^9}{MTTF \text{ (in hours)}}$$

Obtain $f_{LB}(n) \leq f(n)$

- Accounting for **temporal robustness***
- That can be quickly computed for large n

Soundness

Obtain $MTTF_{LB}$

- Compute $f_{LB}(n_0), f_{LB}(n_1), \ldots, f_{LB}(n_D)$
- **Numerically integrate** over subintervals $(n_0, n_1], \ldots, (n_{D-1}, n_D]$

Scalability

Challenges

* Sfakianakis et al.. "Reliability of a consecutive k-out-of-r-from-n: F system." IEEE Transactions on Reliability 41.3 (1992): 442-447.

System Model

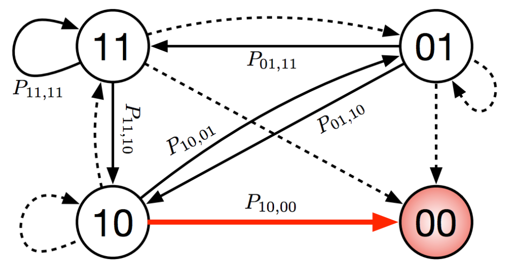
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

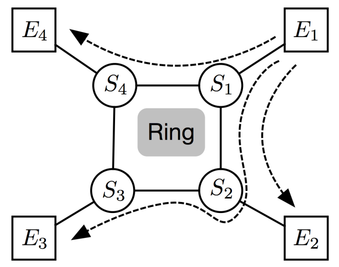
Q2. What is the likelihood of a **control failure?**

Periodic Weakly-Hard Systems [ECRTS '19]



Q1. How often does the **final actuation deviate** from an error-free scenario (iteration failure)?

Replica Consistency over Ethernet [RTAS '20]



Transient fault-induced errors

System Model

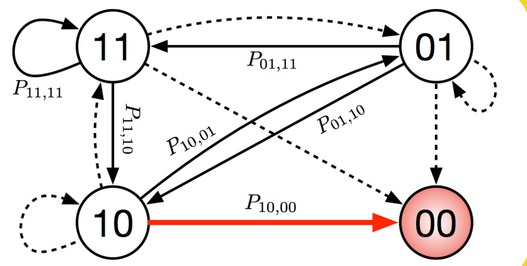
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Q2. What is the likelihood of a **control failure?**

Periodic Weakly-Hard Systems [ECRTS '19]

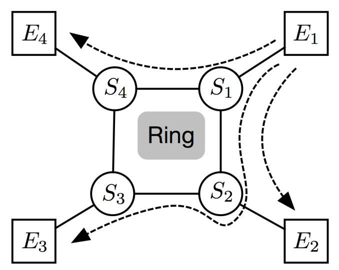


Transient fault-induced errors

Upper-bound the failure probability

- Distributed **real-time systems**
- **Reliability anomalies**
- Non-malicious **Byzantine** errors

Replica Consistency over Ethernet [RTAS '20]



System Model

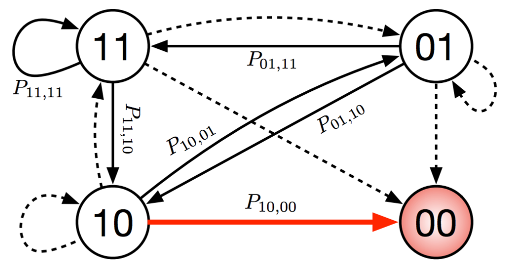
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Beyond hard constraints

Periodic Weakly-Hard Systems [ECRTS '19]

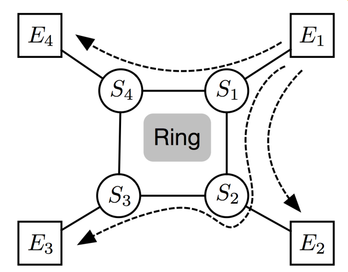


Transient fault-induced errors

Upper-bound the failure probability

- Distributed **real-time systems**
- **Reliability anomalies**
- Non-malicious **Byzantine** errors

Replica Consistency over Ethernet [RTAS '20]



System Model

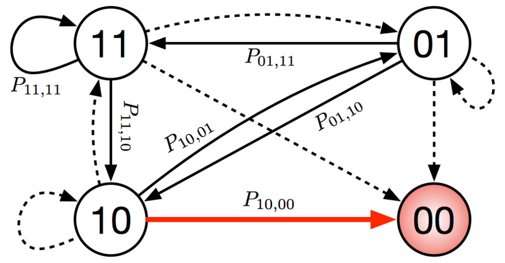
Network control system

Physical plant reliable

Physical sensor — **Controlled plant** — **Physical actuator**

Beyond hard constraints

Periodic Weakly-Hard Systems [ECRTS '19]

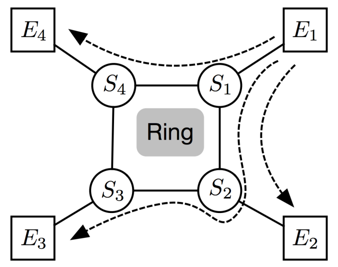


Towards **ultra-reliable CPS**

Upper-bound the failure probability

- Distributed **real-time systems**
- **Reliability anomalies**
- Non-malicious **Byzantine** errors

Replica Consistency over Ethernet [RTAS '20]



Transient fault-induced errors

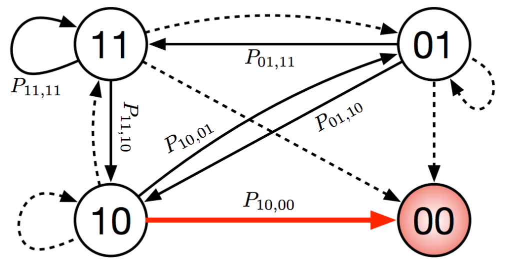
PhD Thesis

Reliability Analysis of Distributed Real-Time Systems

🏆 Best Presentation Award

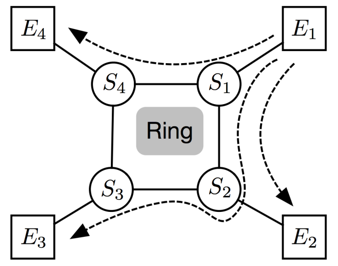
Controller Area Network [ECRTS '18]

Periodic Weakly-Hard Systems [ECRTS '19]



🏆 Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]

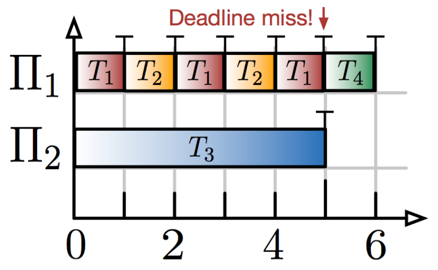


CPS

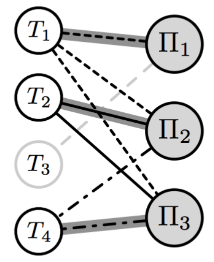
Multiprocessor Hard Real-Time Scheduling

Outstanding Paper Award

Schedulability Analysis of Linux-like Systems [ECRTS '13, RTS Journal '15]



Scheduling Policy for Improved Utilization [RTSS '14]



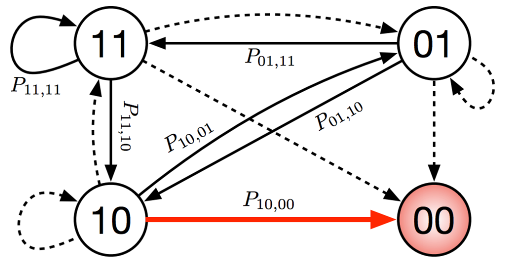
PhD Thesis

Reliability Analysis of Distributed Real-Time Systems

Best Presentation Award

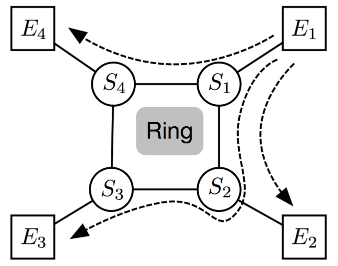
Controller Area Network [ECRTS '18]

Periodic Weakly-Hard Systems [ECRTS '19]



Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]

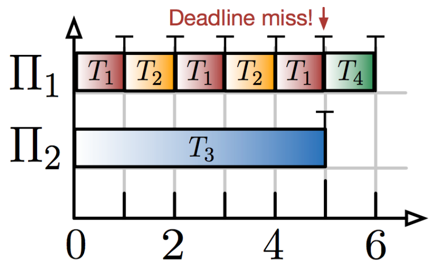


CPS

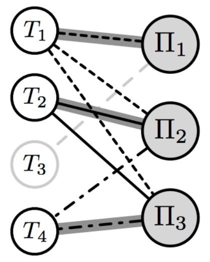
Multiprocessor Hard Real-Time Scheduling

🏆 Outstanding Paper Award

Schedulability Analysis of Linux-like Systems [ECRTS '13, RTS Journal '15]



Scheduling Policy for Improved Utilization [RTSS '14]



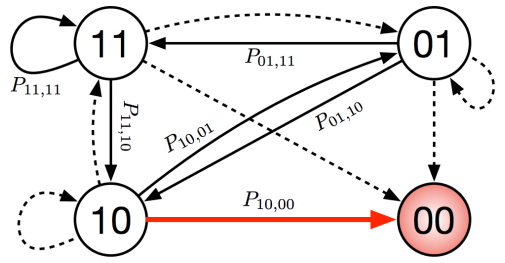
PhD Thesis

Reliability Analysis of Distributed Real-Time Systems

🏆 Best Presentation Award

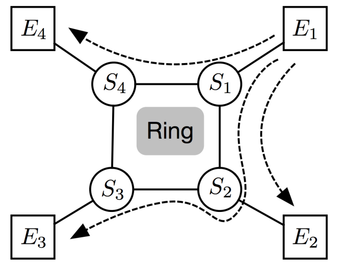
Controller Area Network [ECRTS '18]

Periodic Weakly-Hard Systems [ECRTS '19]



🏆 Distinguished Paper Award

Replica Consistency over Ethernet [RTAS '20]



Cloud

Predictable Resource Allocation

🏆 Best Student Paper Award

Swayam Autoscaler [Middleware '17]

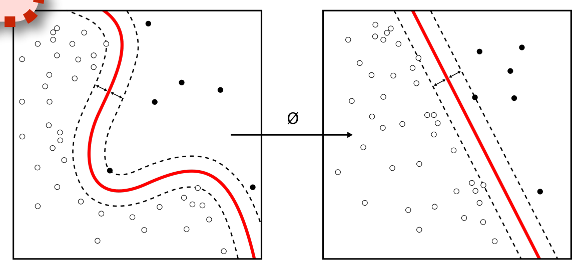
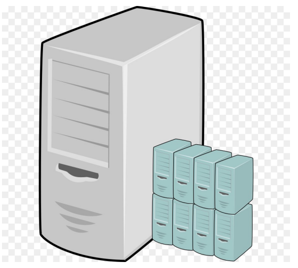


Tableau VM Scheduler [EuroSys '18]



Clockwork for DNN Serving [OSDI '20]