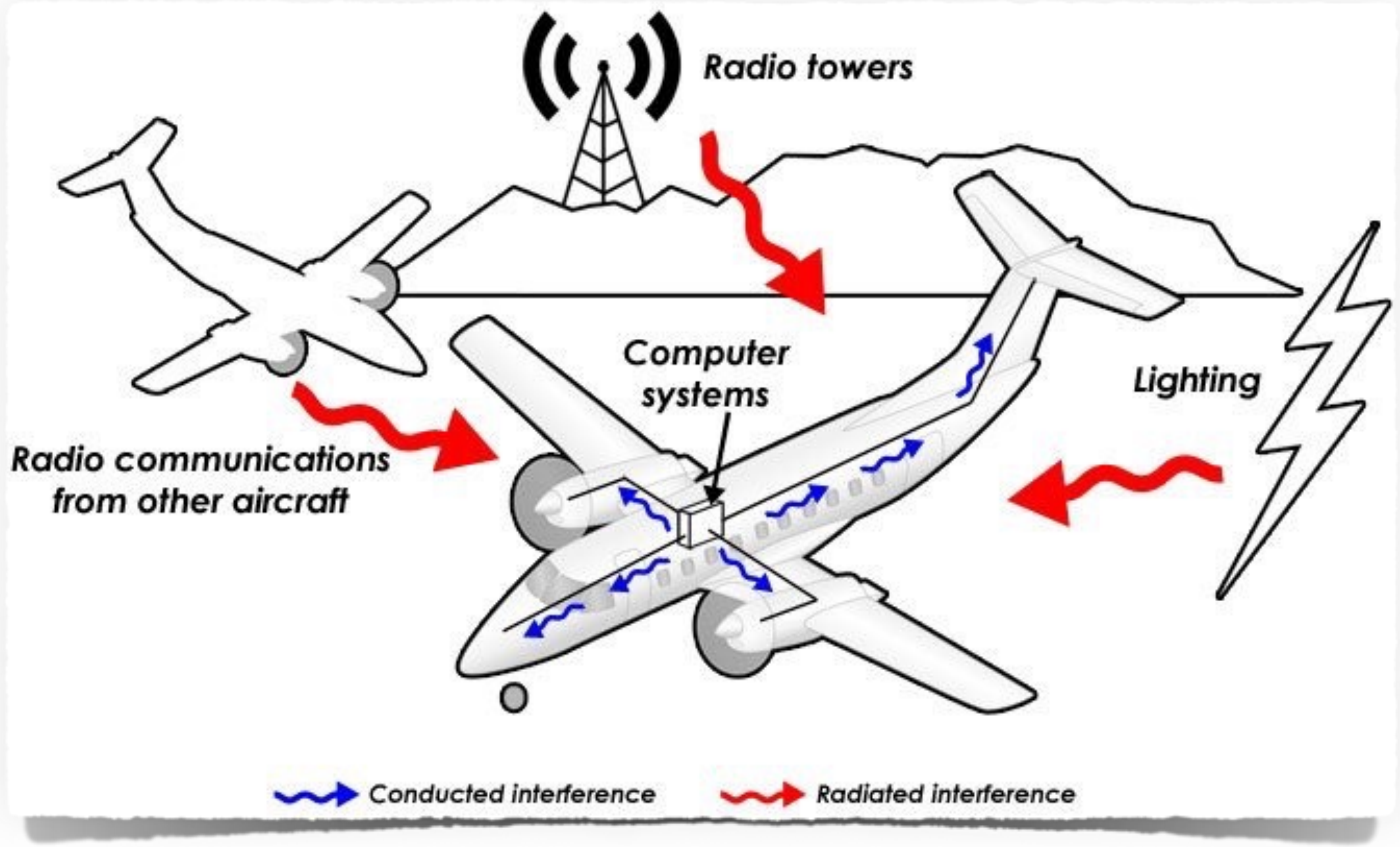
Real-Time Replica Consistency over Ethernet with Reliability Bounds

Arpan Gujarati, Sergey Bozhko,
and Björn B. Brandenburg

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

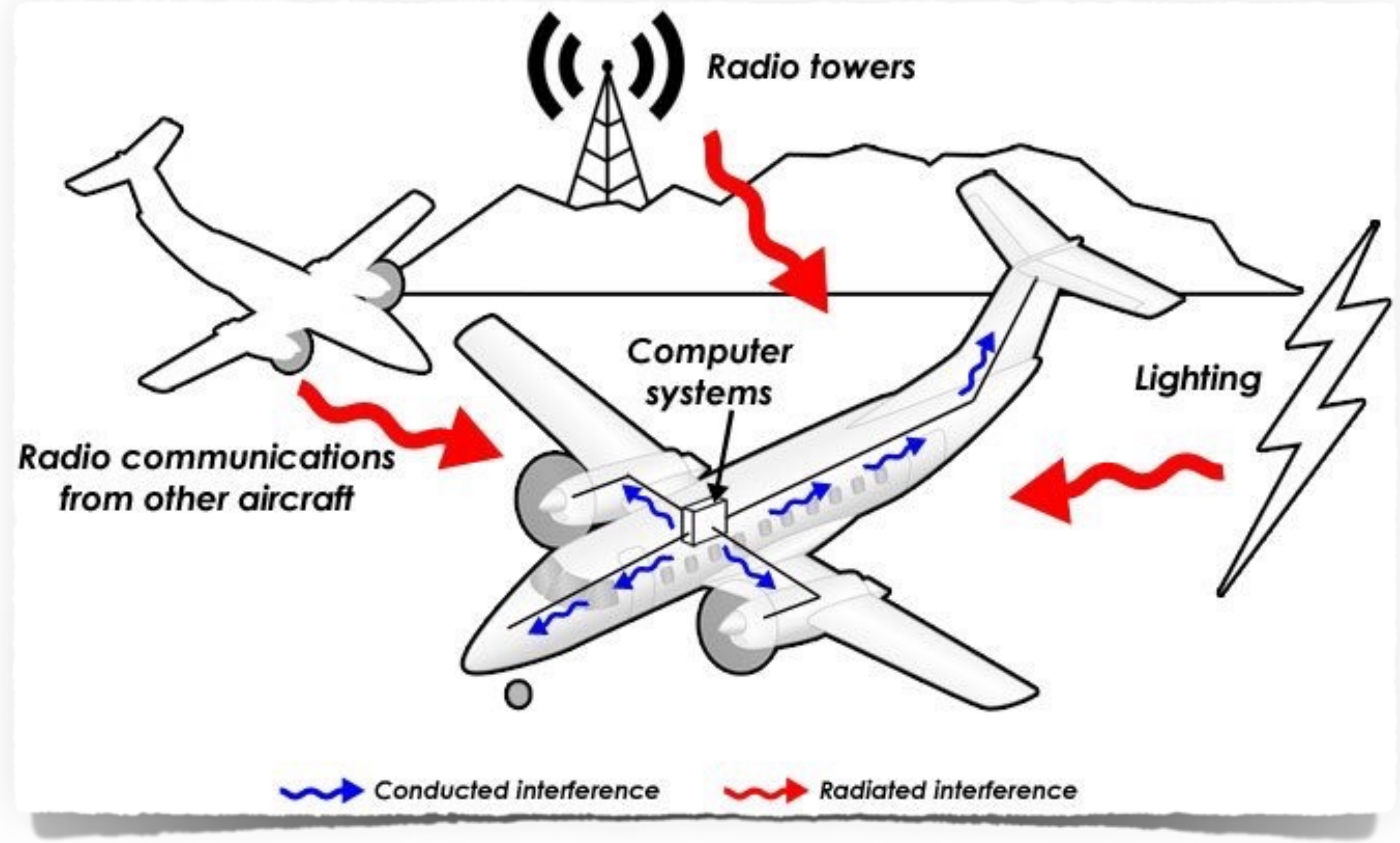
Environmentally-induced **transient faults**

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



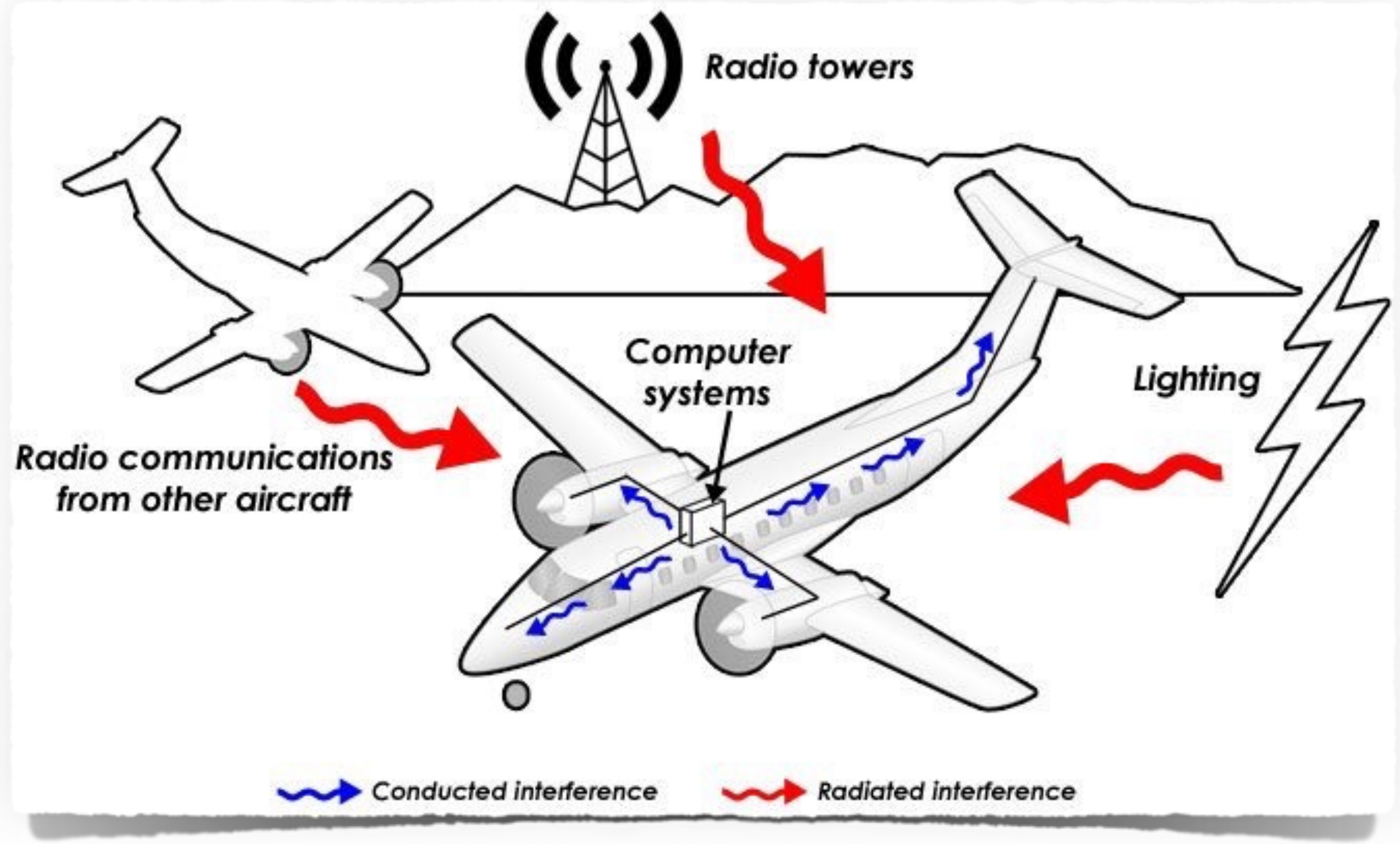
Environmentally-induced **transient faults**

- 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 systems connected over point-to-point networks like Ethernet

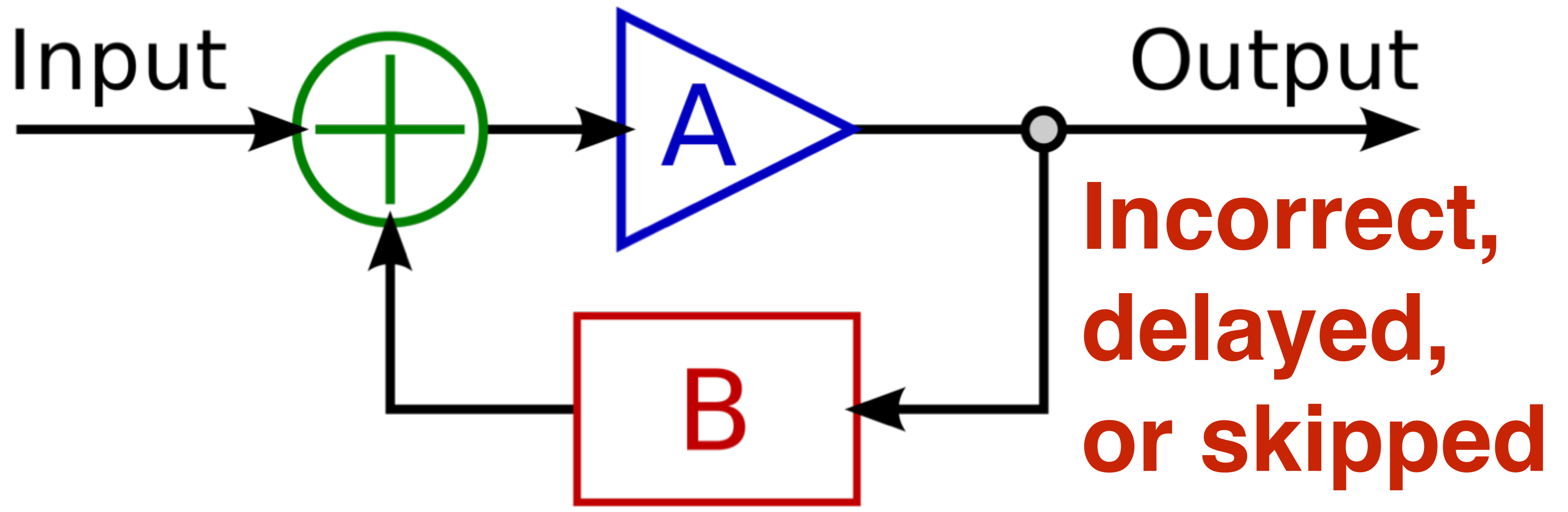
Errors and failures due to transient faults

- 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 systems connected over point-to-point networks like Ethernet

Failures in

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

E.g., safety-critical control system



Errors and failures due to transient faults

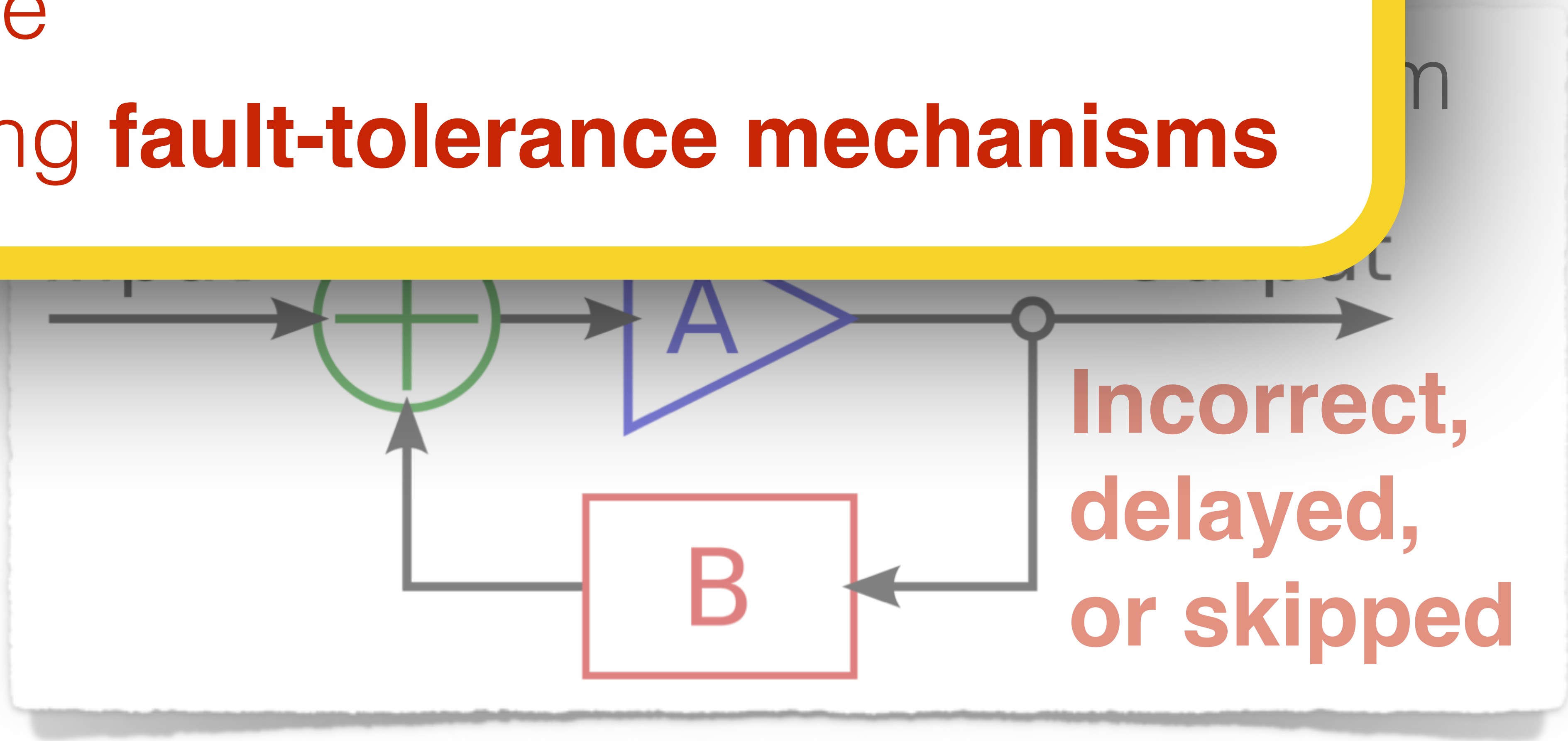
- Transmission errors
  - ➡ Faults on the network
- Omission
  - ➡ Fault-in
- Incorrec
  - ➡ Faults in
- Inconsistent broadcast errors
  - ➡ Faults in systems connected over point-to-point networks like Ethernet

Failures in

- ➡ **value domain (incorrect output)**
- ➡ **put)**

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 systems connected over point-to-point networks like Ethernet

Byzantine Fault Tolerance (BFT)

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 systems connected over point-to-point networks like Ethernet

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
  - ➡ Faults on the netw
- Omission errors
  - ➡ Fault-induced kernel panics, hangs
- Incorrect computation errors
  - ➡ Faults in memory buffers
- Inconsistent broadcast errors
  - ➡ Faults in systems connected over point-to-point networks like Ethernet

Industry:  **RULE**

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
  - ➡ Faults on the netw
- Omission errors
  - ➡ Fault-induced kernel p
- Incorrect computatio
  - ➡ Faults in memory buffers
- Inconsistent broadcast errors
  - ➡ Faults in systems connected over point-to-point networks like Ethernet

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 p
- Incorrect computati
  - Faults in memory buffers
- Inconsistent broadcast errors
  - Faults in systems connected over point-to-point networks like Ethernet

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?

- Transmission errors

Real-time requirements

Industry: RULE

Safety certification

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

SWaP-C

Size, Weight, and Power ... plus Cost

- Inconsistent broadcast errors
  - ➡ Faults in systems connected over point-to-point networks like Ethernet

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?

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)

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

This work!

Checksums and retransmissions

Dual Modular Redundancy (DMR)

ECC Memory +
Triple Modular Redundancy (TMR)

Byzantine Fault Tolerance (BFT)

Focus

Design and reliability analysis of a **BFT protocol**
for **Ethernet-based** distributed real-time systems

System design

Network control system

Physical plant reliable

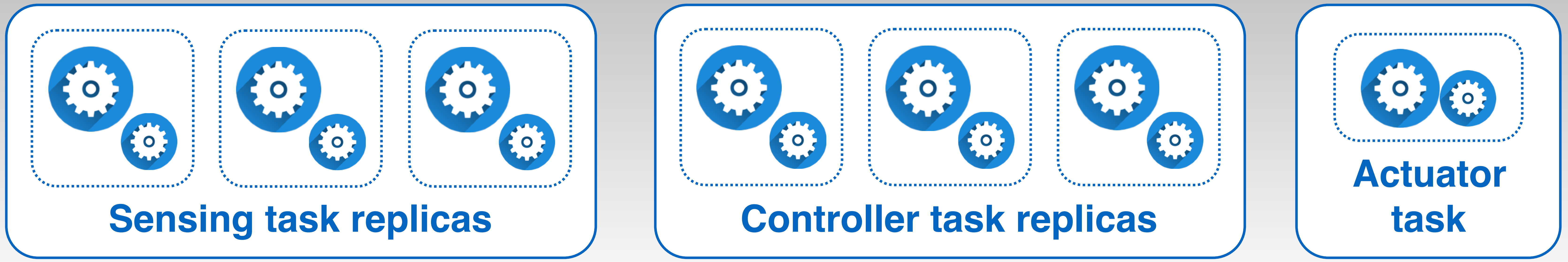
System design

Network control system

Physical plant reliable

Active Replication

DMR / TMR / Hybrid



System design

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

Ethernet Time-Sensitive Networking (TSN)

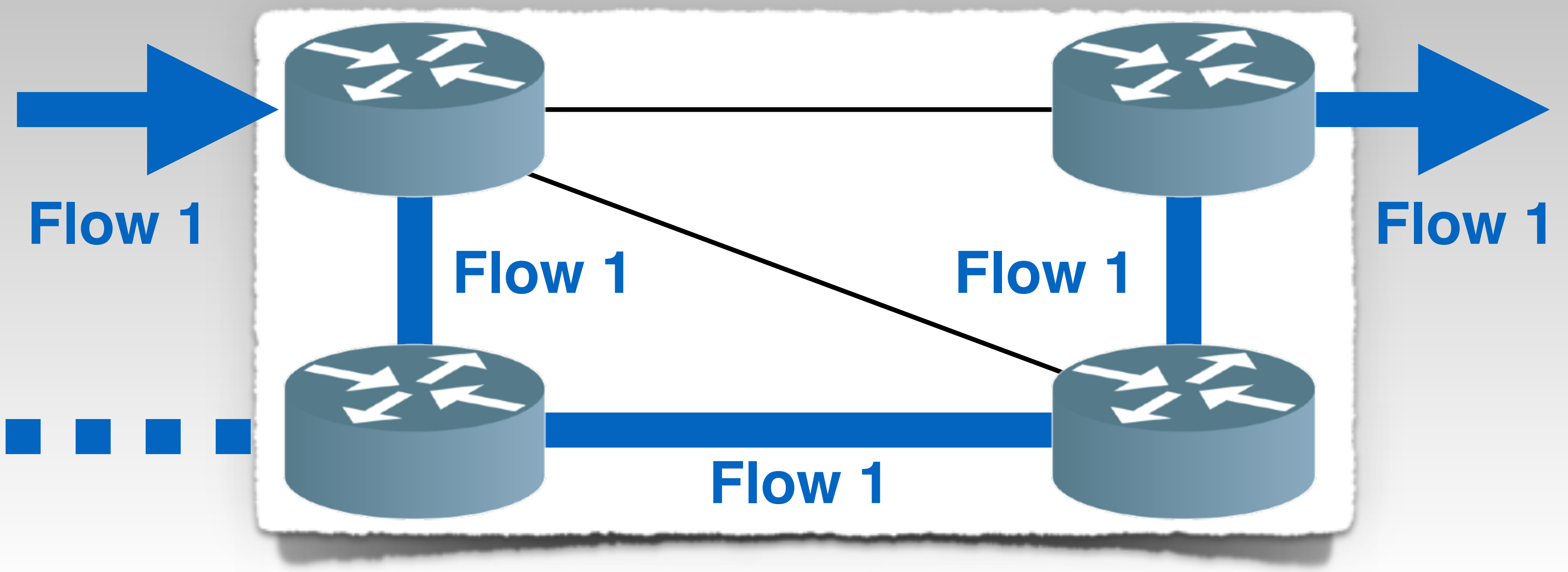
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



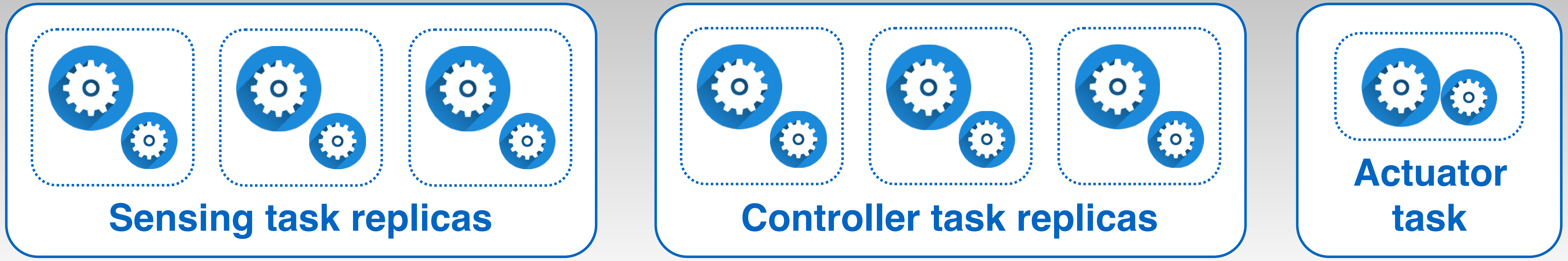
System design

Network control system

Physical plant reliable

Active Replication

DMR / TMR / Hybrid

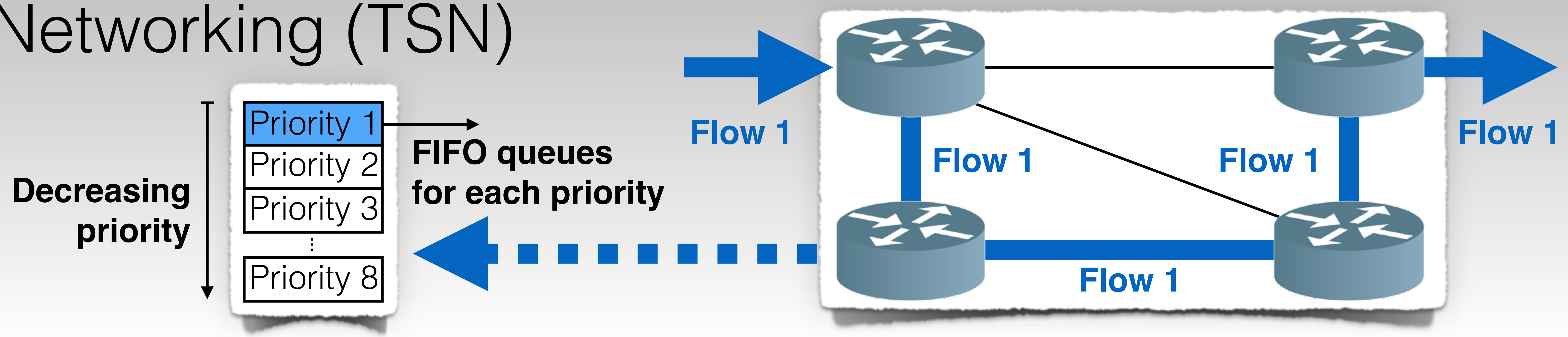


Problem: Replicas can diverge due to Byzantine errors

Ethernet Time-Sensitive Networking (TSN)

Statically reserved routes

Priority classes



System design

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

Problem: Replicas can diverge due to Byzantine errors

Key idea: Byzantine fault tolerant (BFT) atomic broadcast layer

Ethernet Time-Sensitive Networking (TSN)

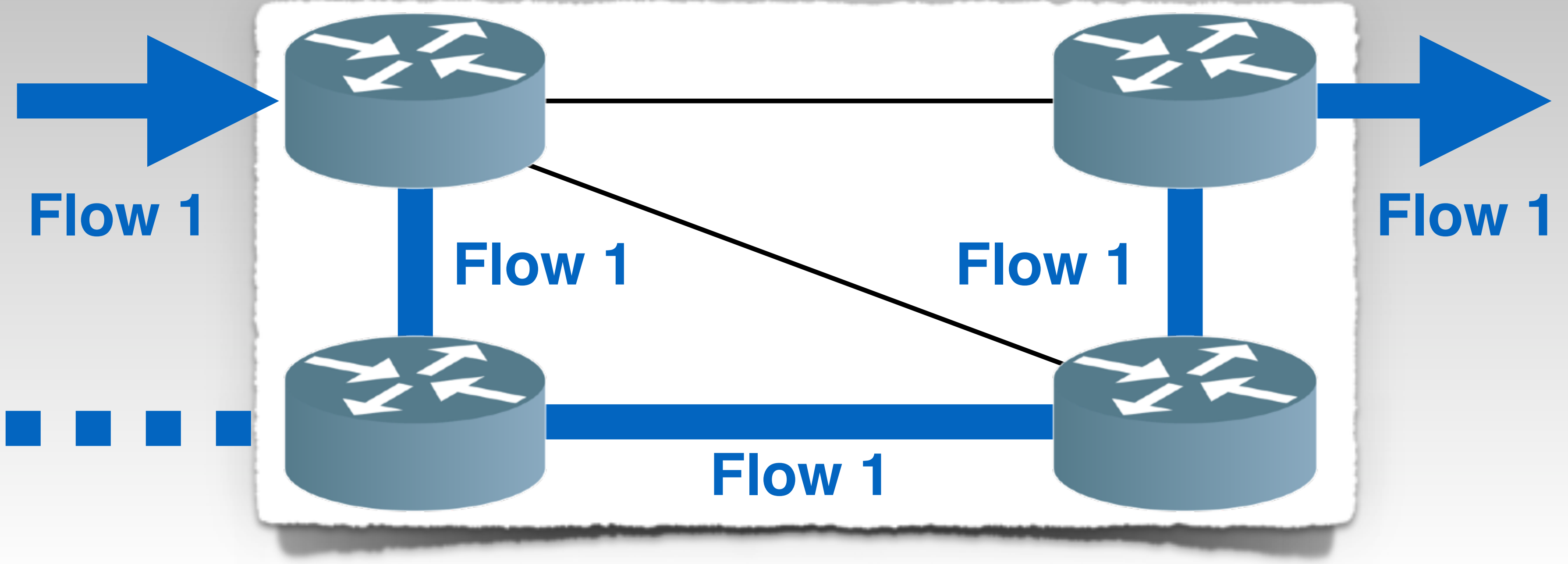
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



System design

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

Problem: Replicas can diverge due to Byzantine errors

Key idea: Byzantine fault tolerant (BFT) atomic broadcast layer

Challenge: Prior work does not consider hard real-time predictability

Ethernet Time-Sensitive Networking (TSN)

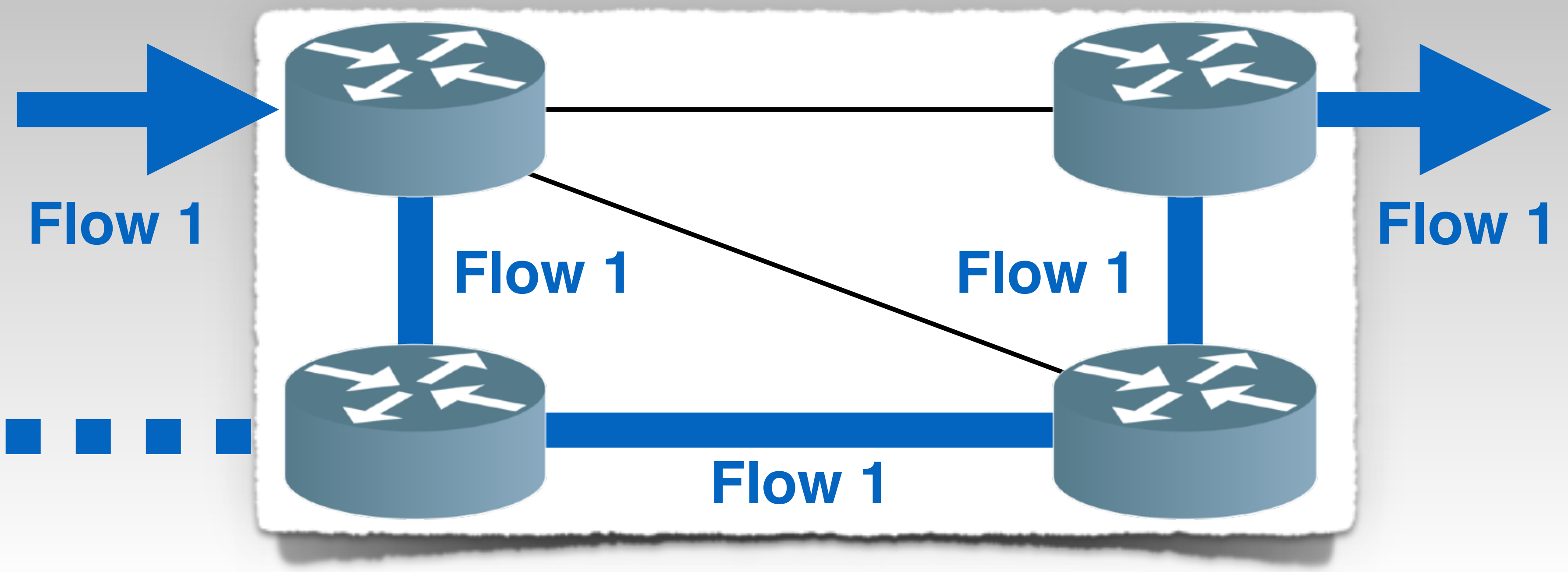
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



System design

Network control system

Physical plant reliable

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

Active Replication

DMR / TMR / Hybrid

Sensing task replicas

Controller task replicas

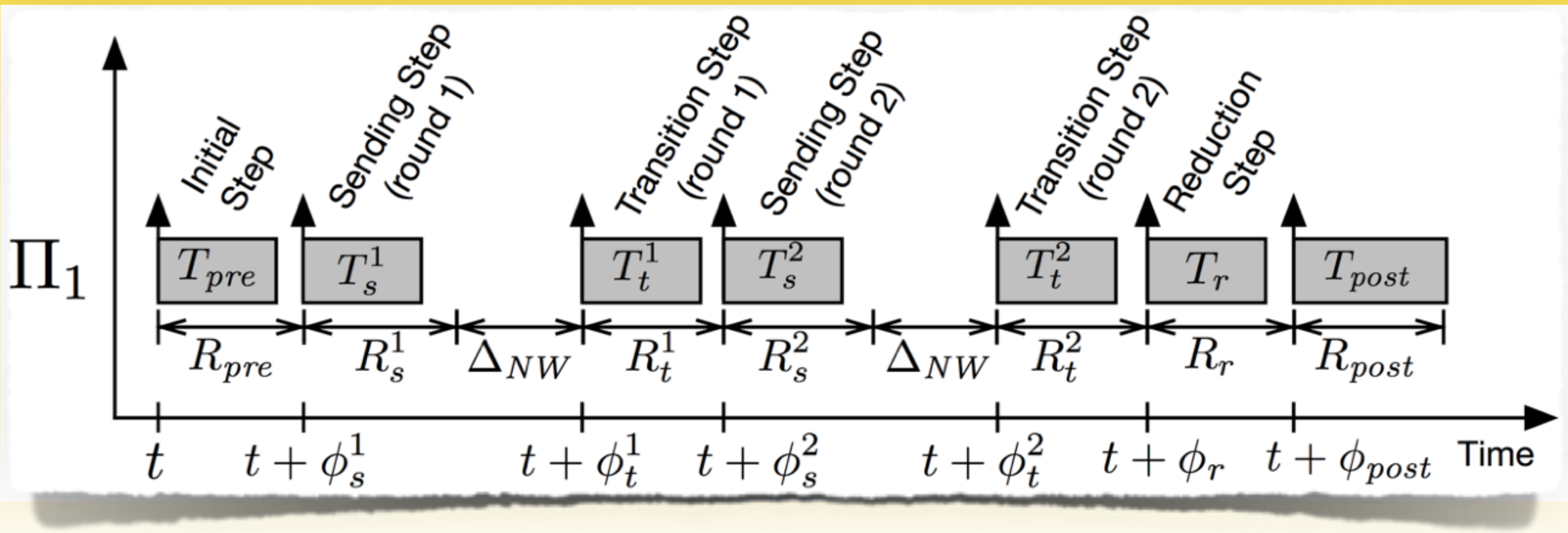
Actuator task

BFT 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)

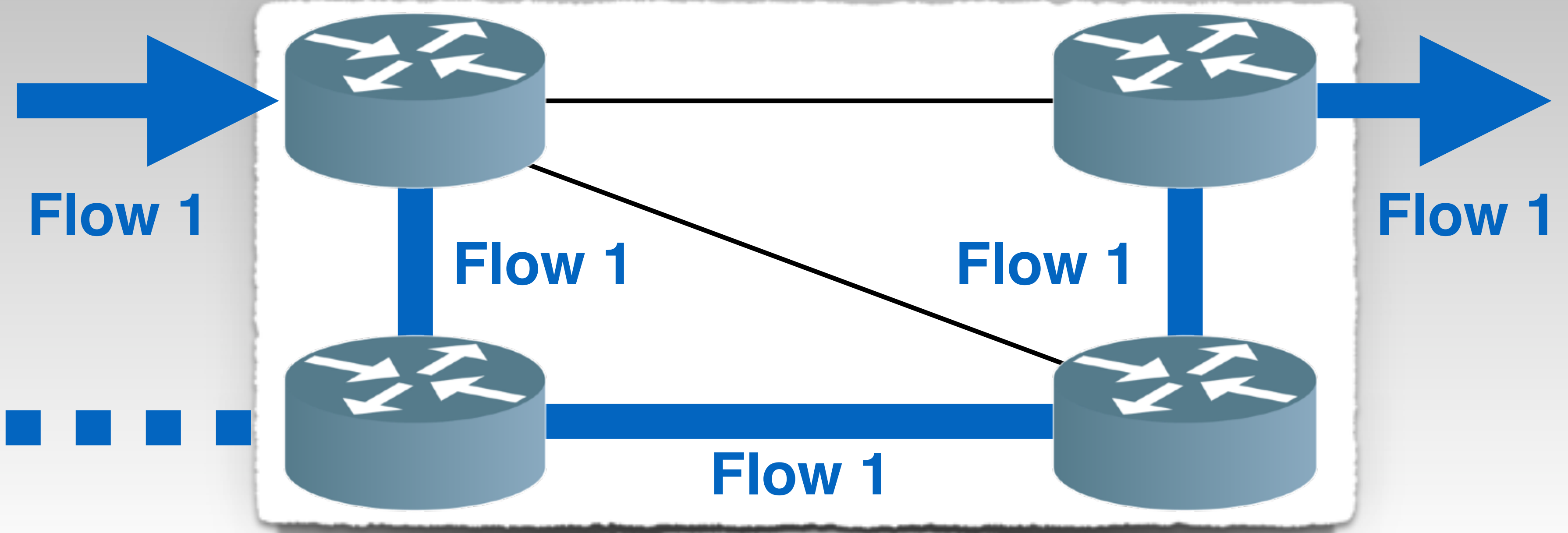
Statically reserved routes

Priority classes

Decreasing priority

Priority 1
Priority 2
Priority 3
⋮
Priority 8

FIFO queues for each priority



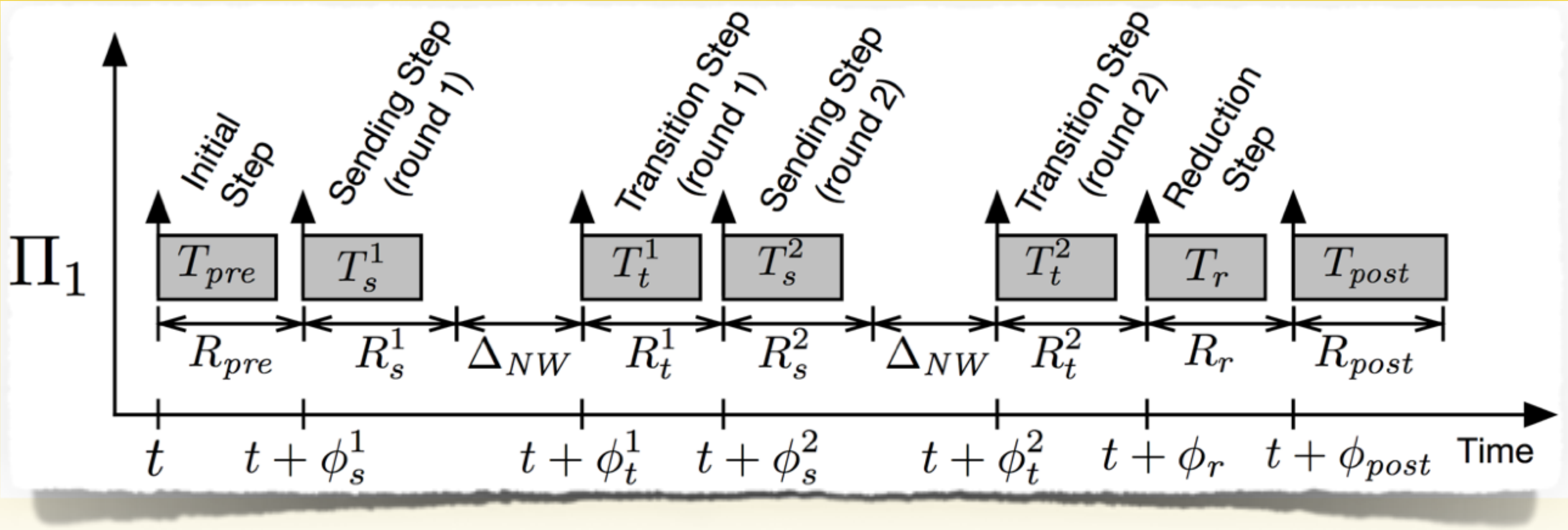
System design

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

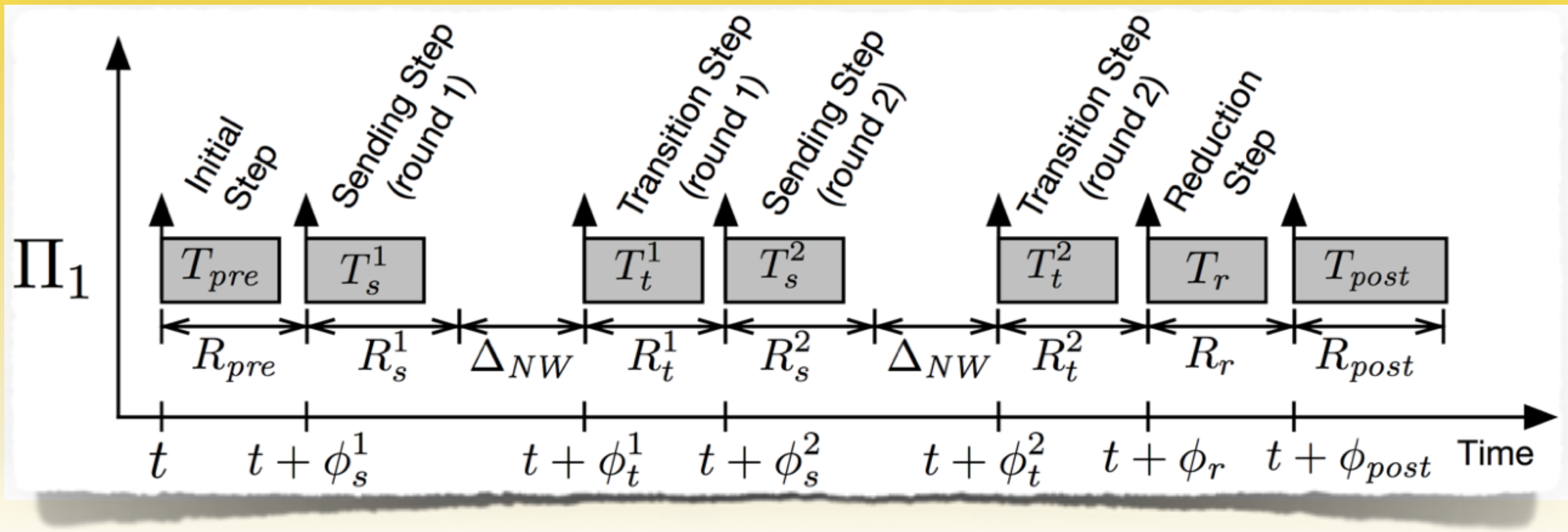
System design

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

This talk: Reliability Analysis

System design

What is the probability of an **atomic broadcast failure?**

Transient fault-induced errors

Ethernet frame corruptions / omissions

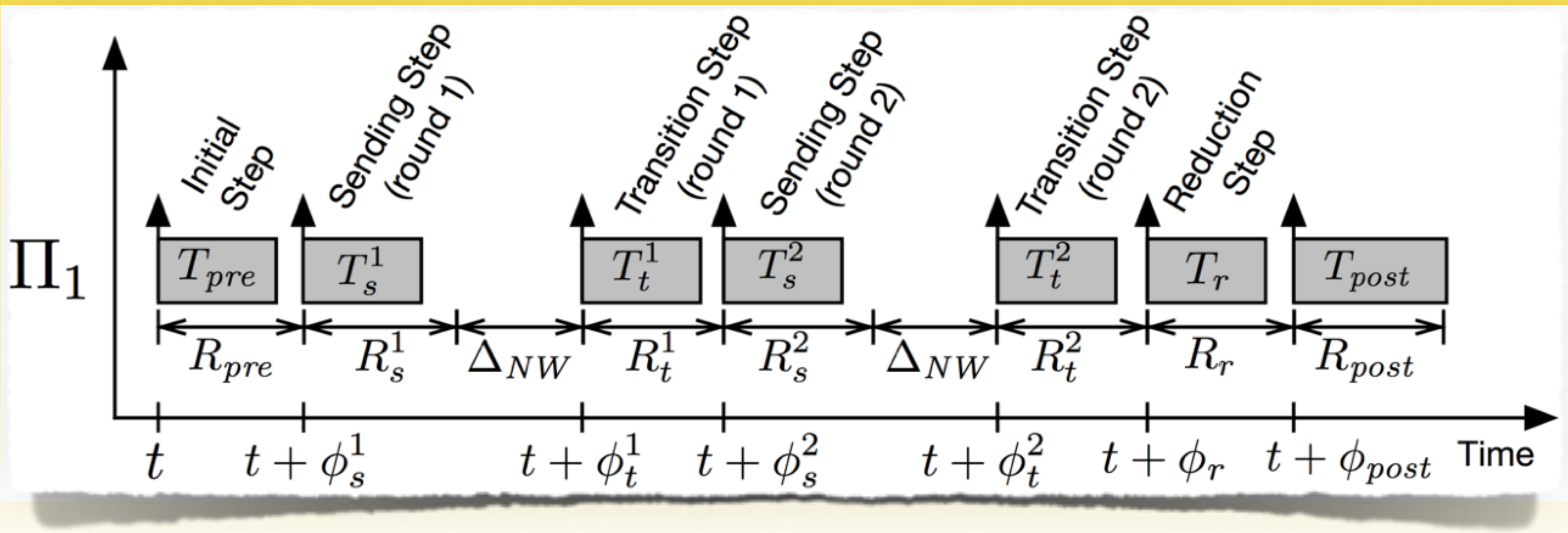
Omission / incorrect computation at nodes and switches

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

This talk: Reliability Analysis

System design

What is the probability of an **atomic broadcast failure?**

Transient fault-induced errors

Ethernet frame corruptions / omissions

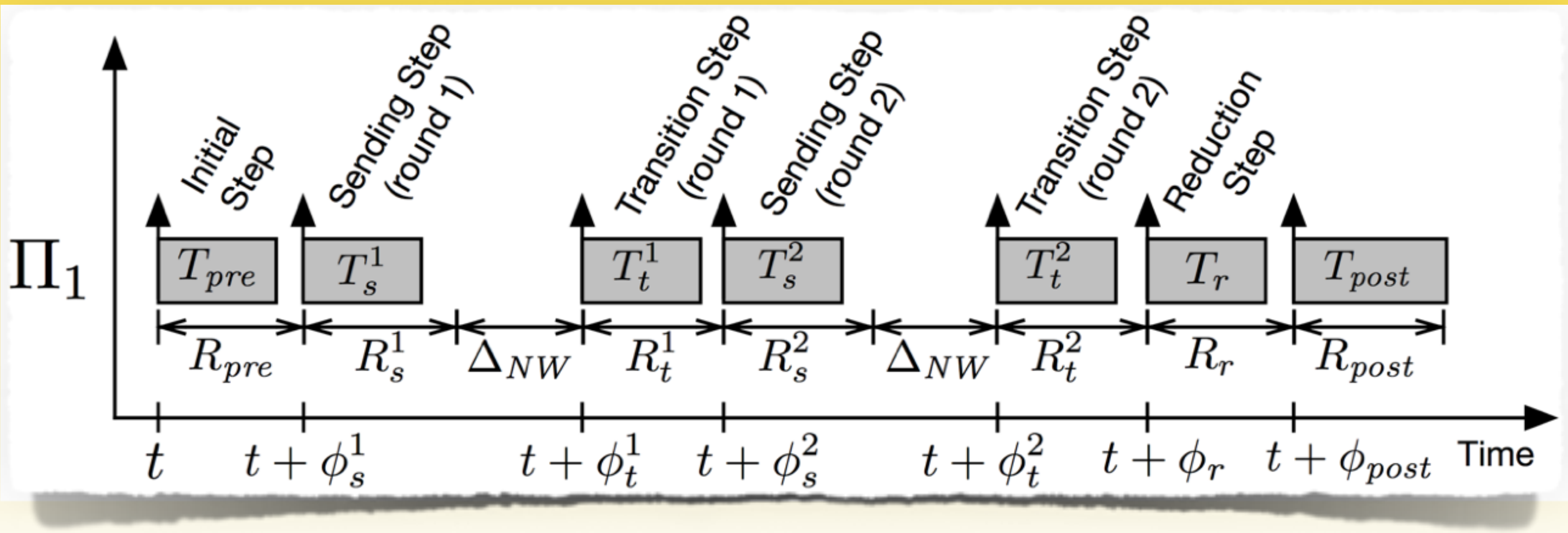
Omission / incorrect computation at nodes and switches

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

This talk: Reliability Analysis

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

System design

Ethernet frame corruptions / omissions

Transient fault-induced errors

Omission / incorrect computation at nodes and switches

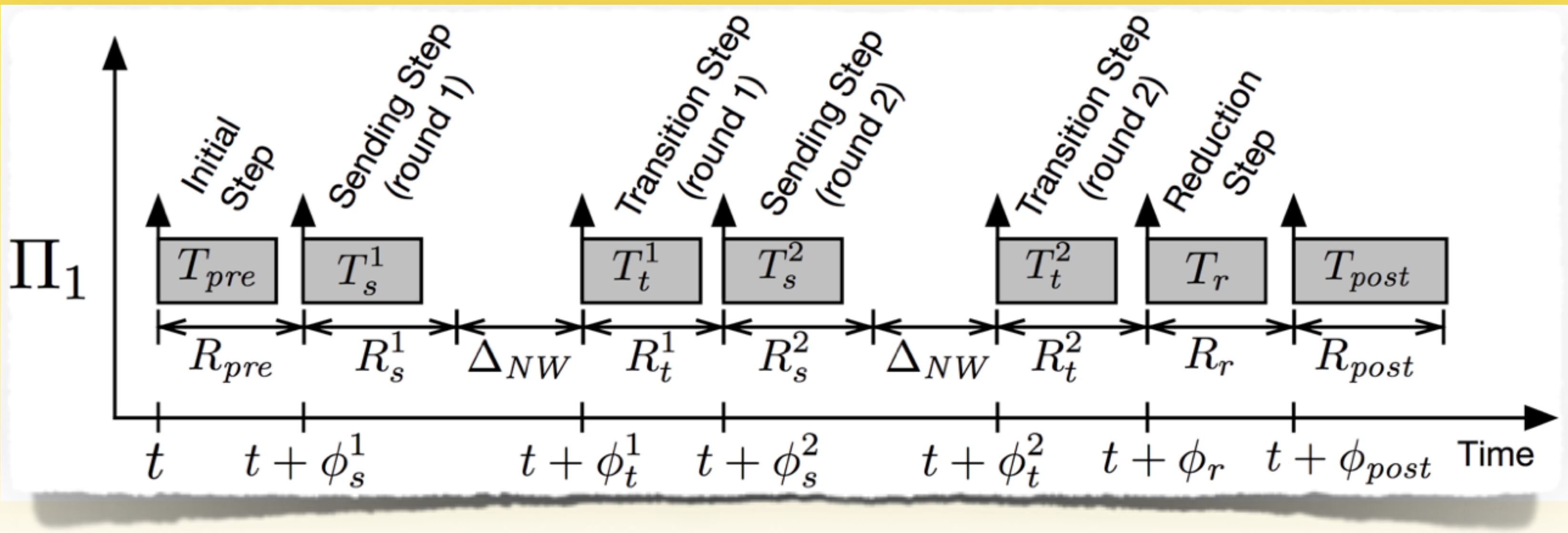
What is the probability of an **atomic broadcast failure?**

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages

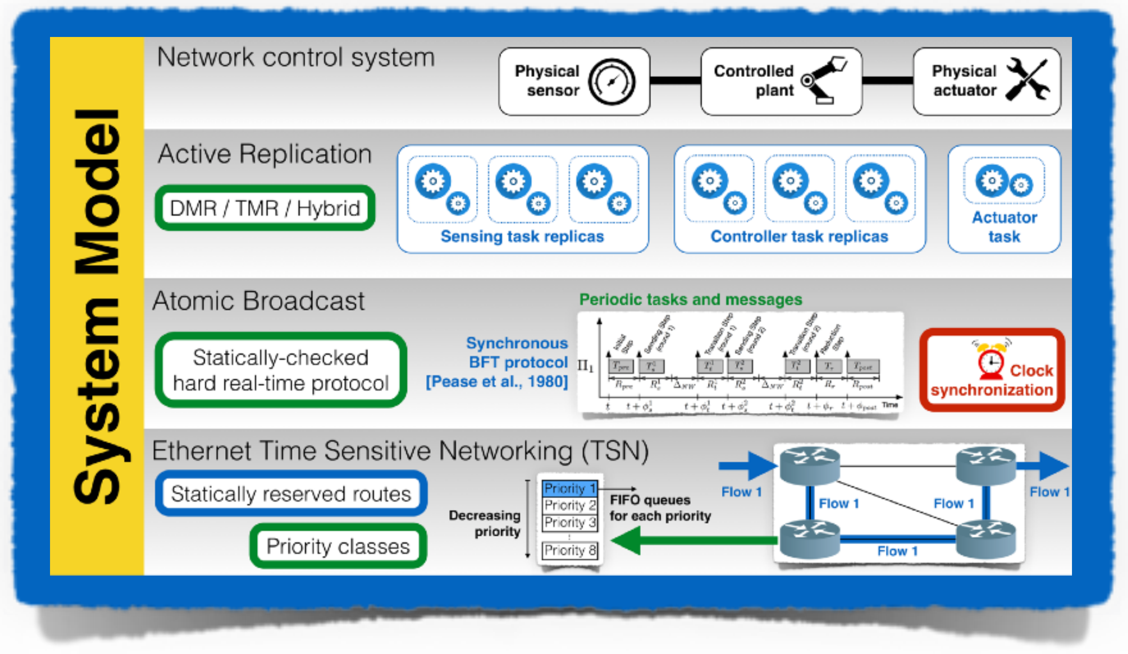


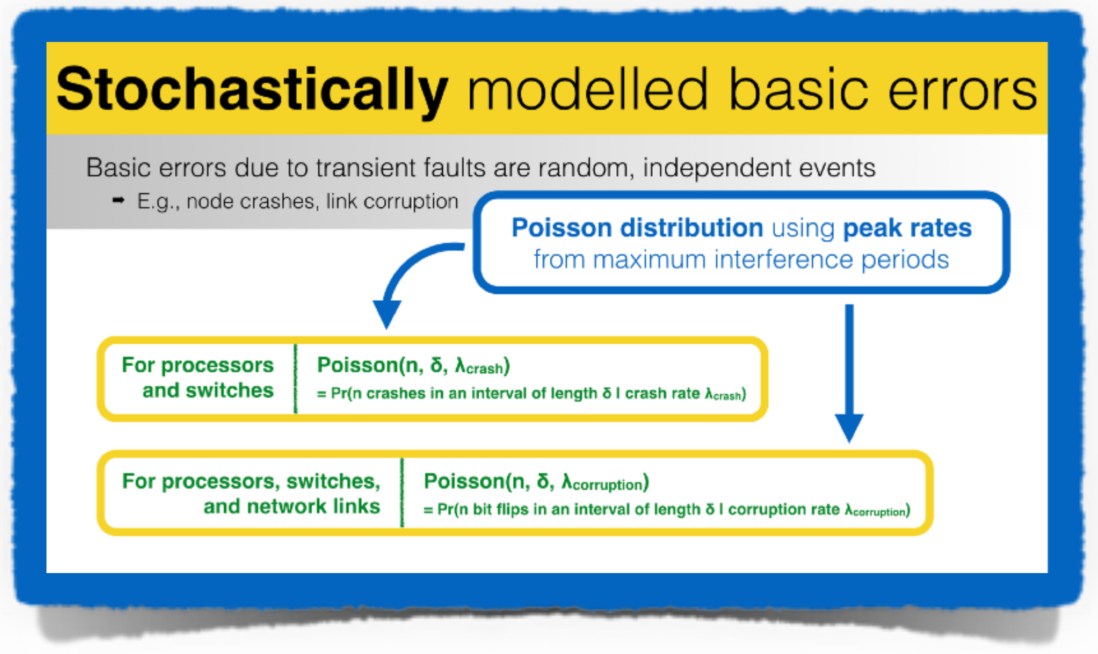
Details in the paper!

Clock synchronization

This talk: Reliability Analysis

Straw-man solutions

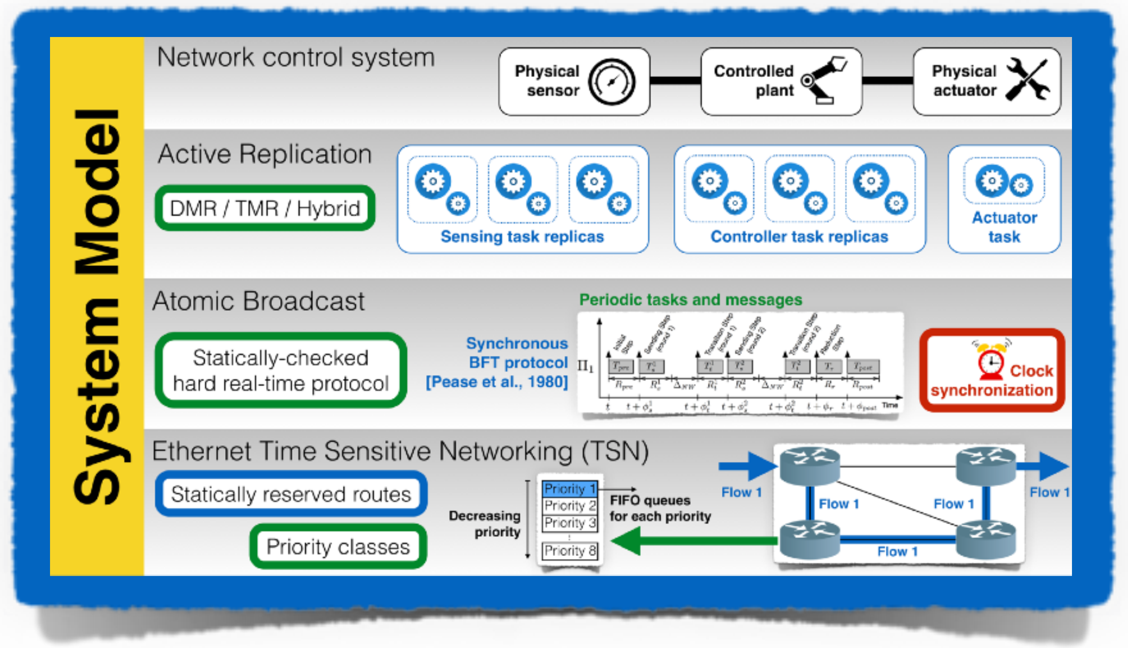


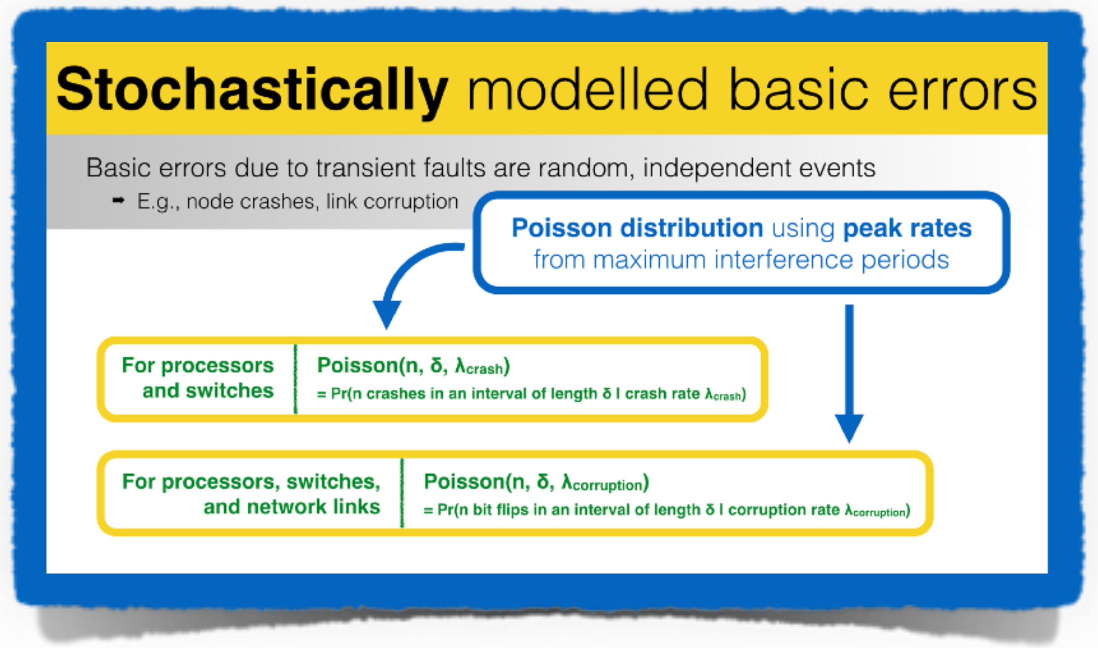


Model checking or simulation

Pr (atomic broadcast failure)

Straw-man solutions





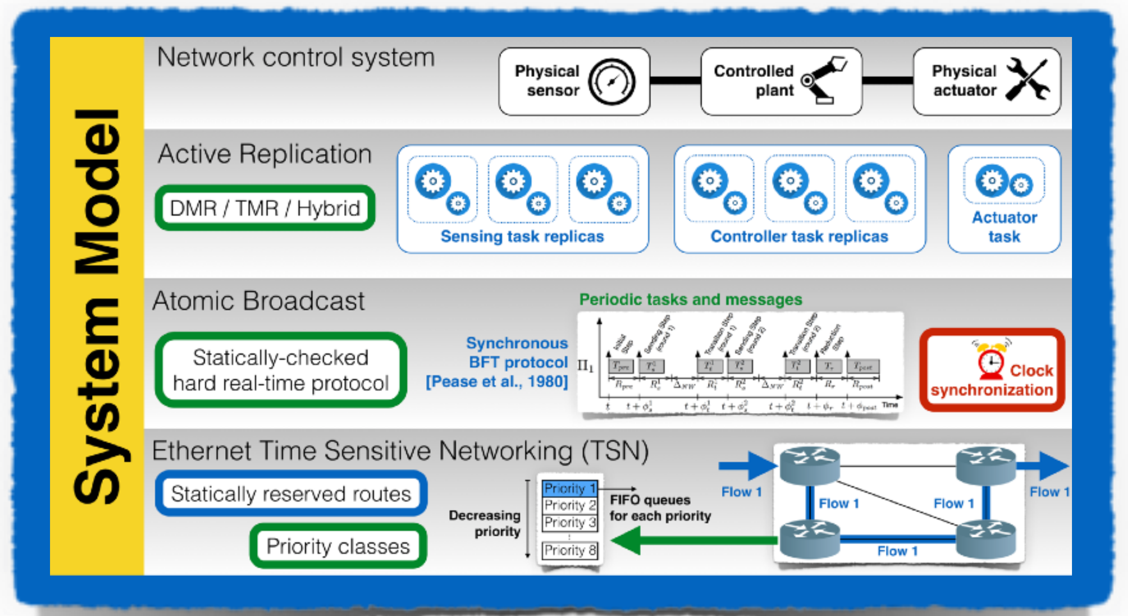
Model checking or simulation

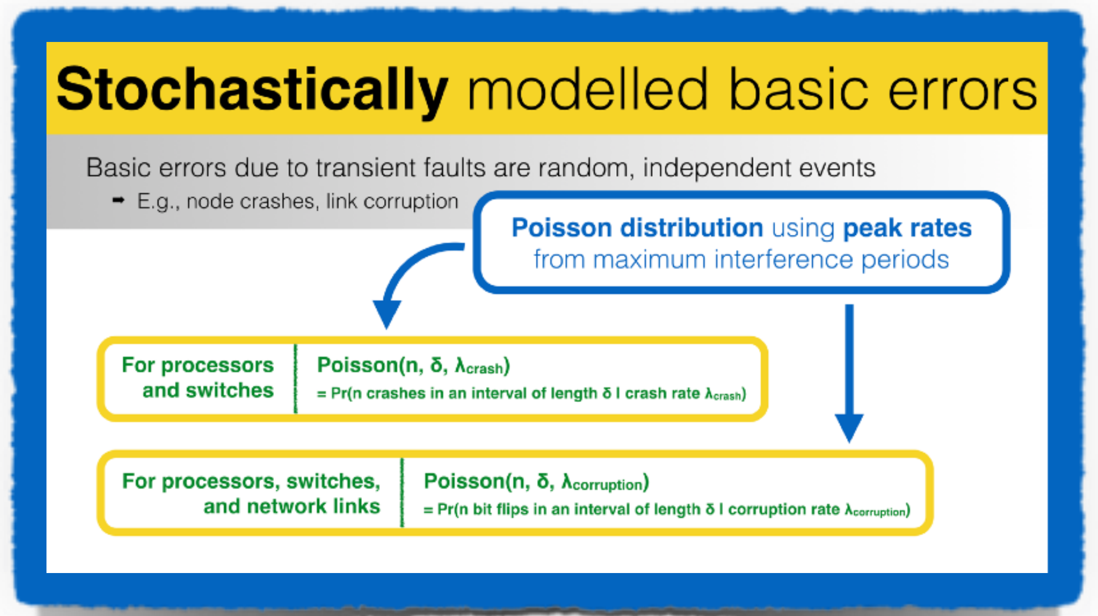
Pr (atomic broadcast failure)

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 (atomic broadcast 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 failure probability may **significantly exceed** the estimated Pr (atomic broadcast failure)

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

Goal: P_{UB} > Pr (atomic broadcast 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 (atomic broadcast 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 (atomic broadcast 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 ...**

Error event E_1

Round 1 messages sent by Π_1 omitted at source

Goal: P_{UB} > Pr (atomic broadcast 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 (atomic broadcast 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 (atomic broadcast 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

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 (atomic broadcast 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 (atomic broadcast failure)

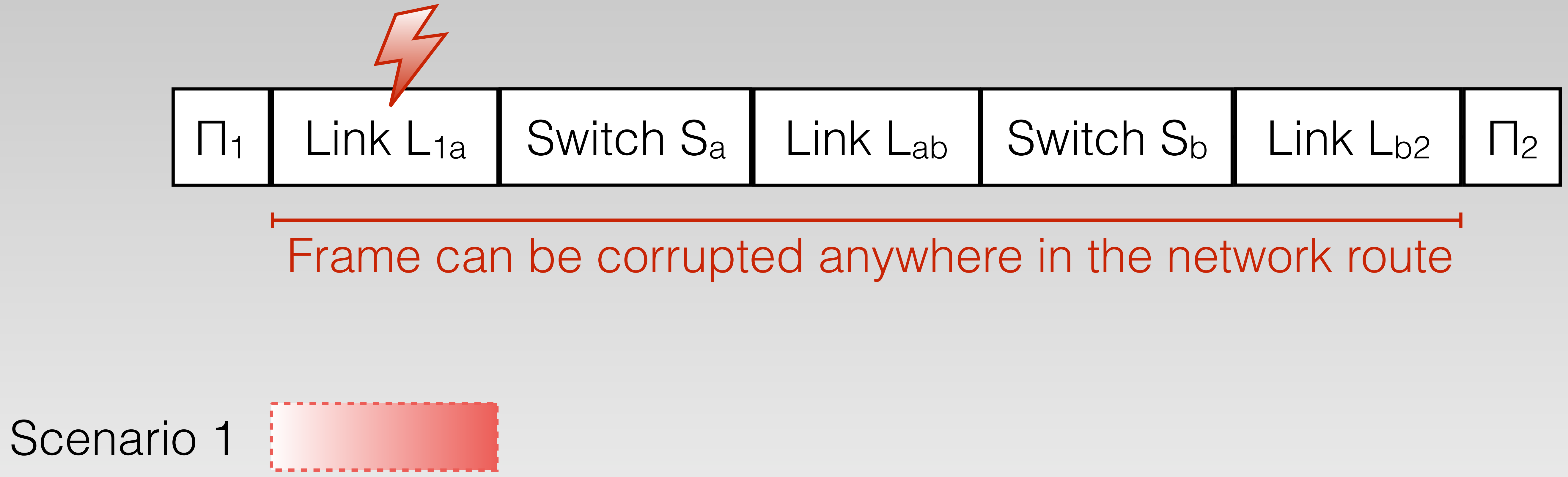
Application-specific message errors

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

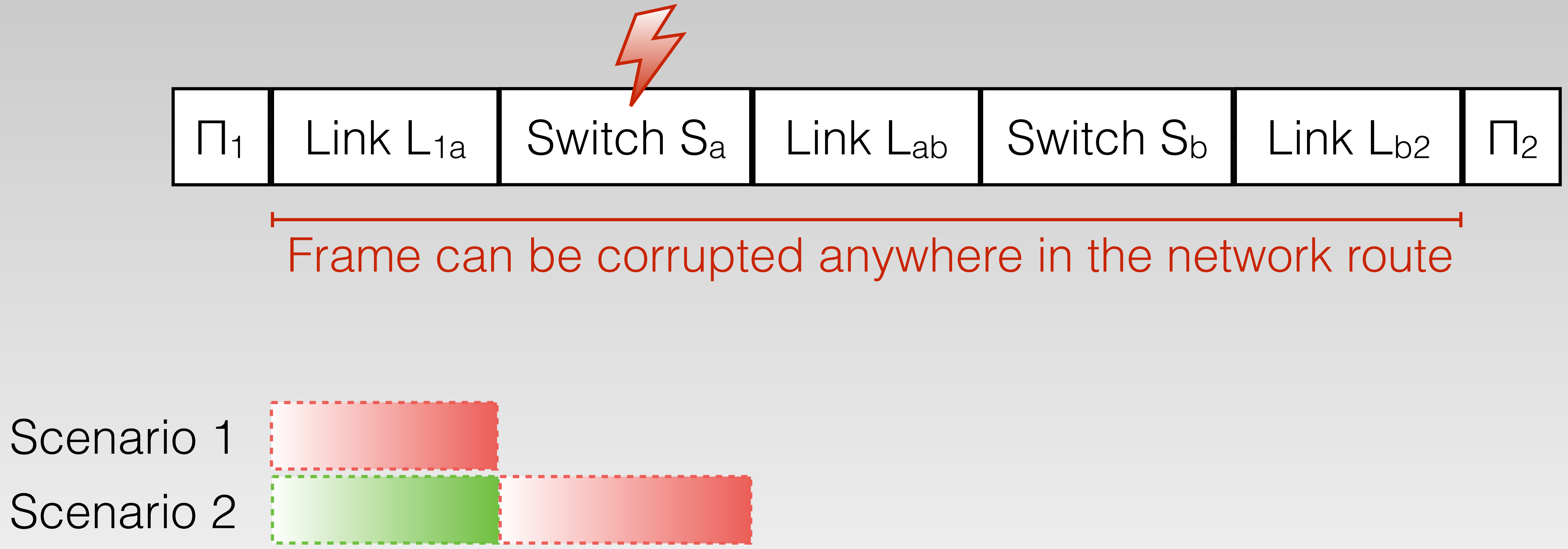
Basic Errors + Poisson Model + Timing Analysis

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

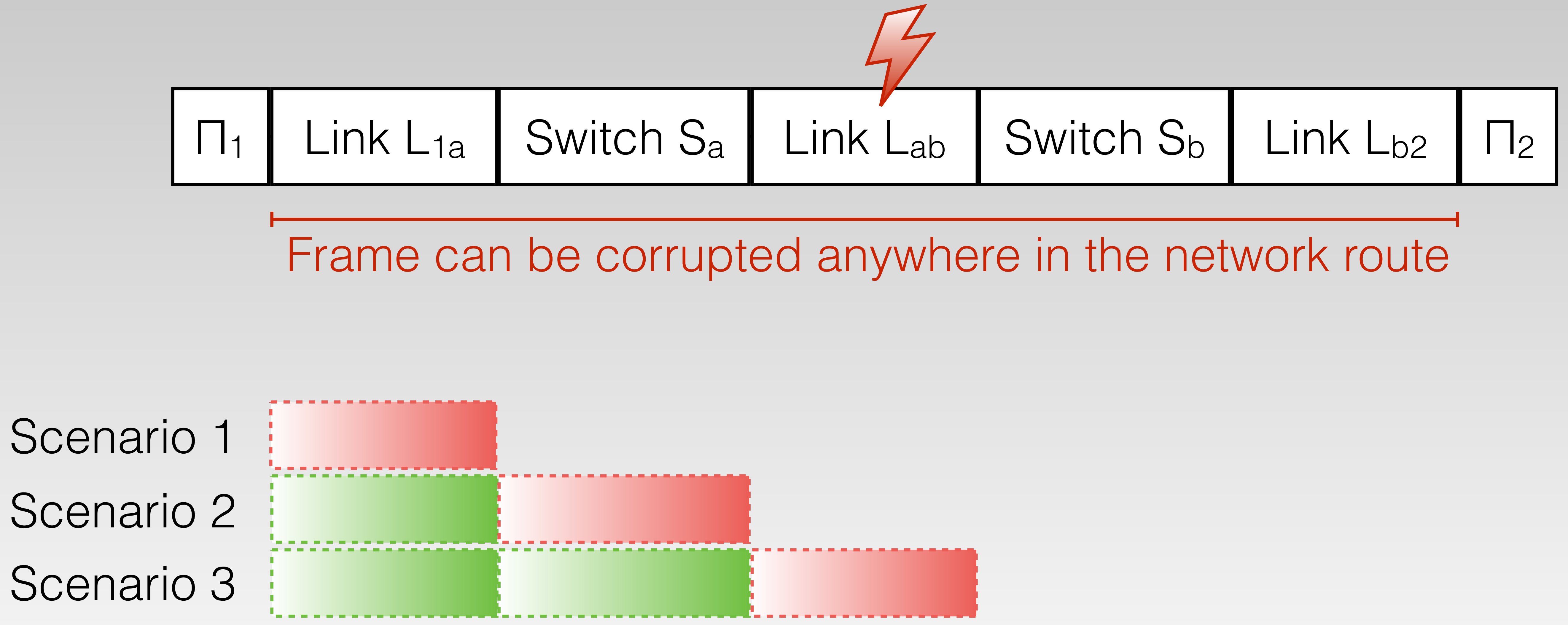
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



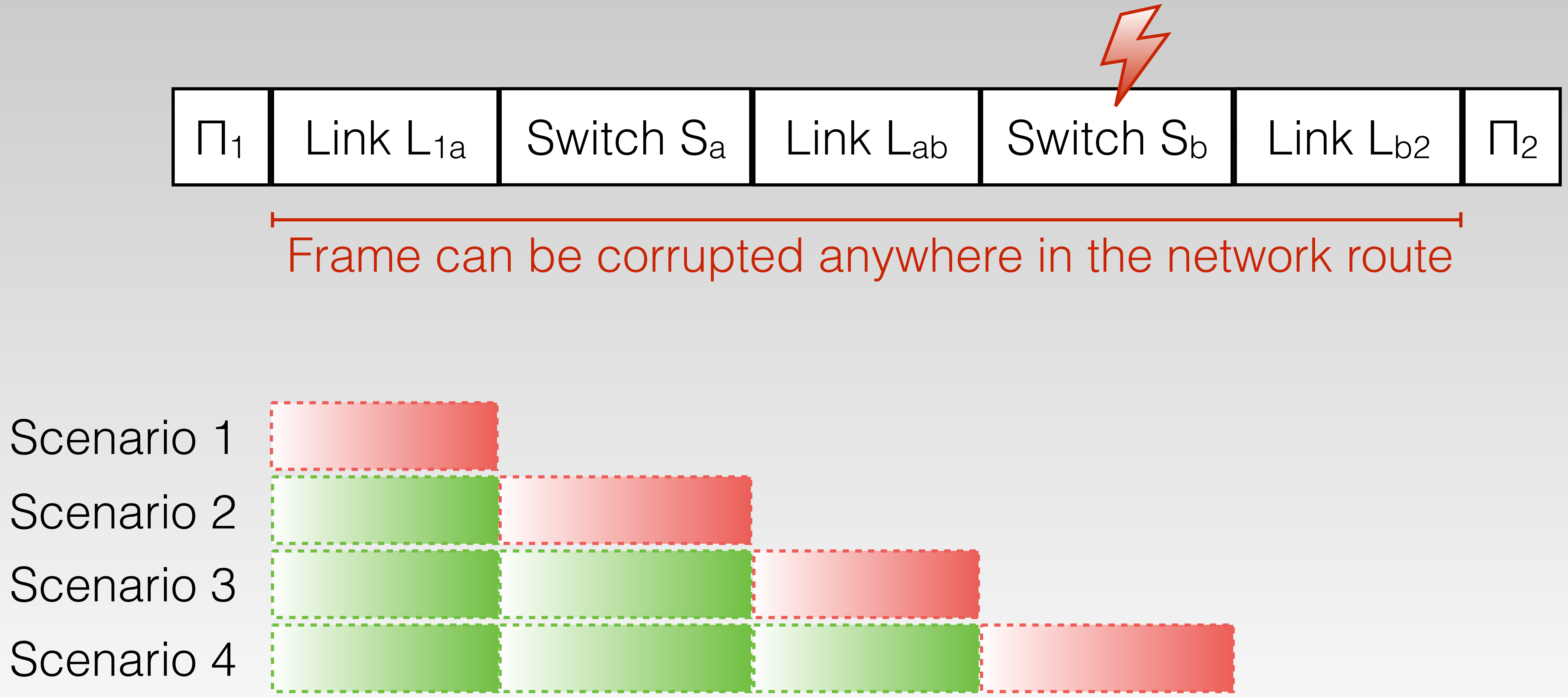
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



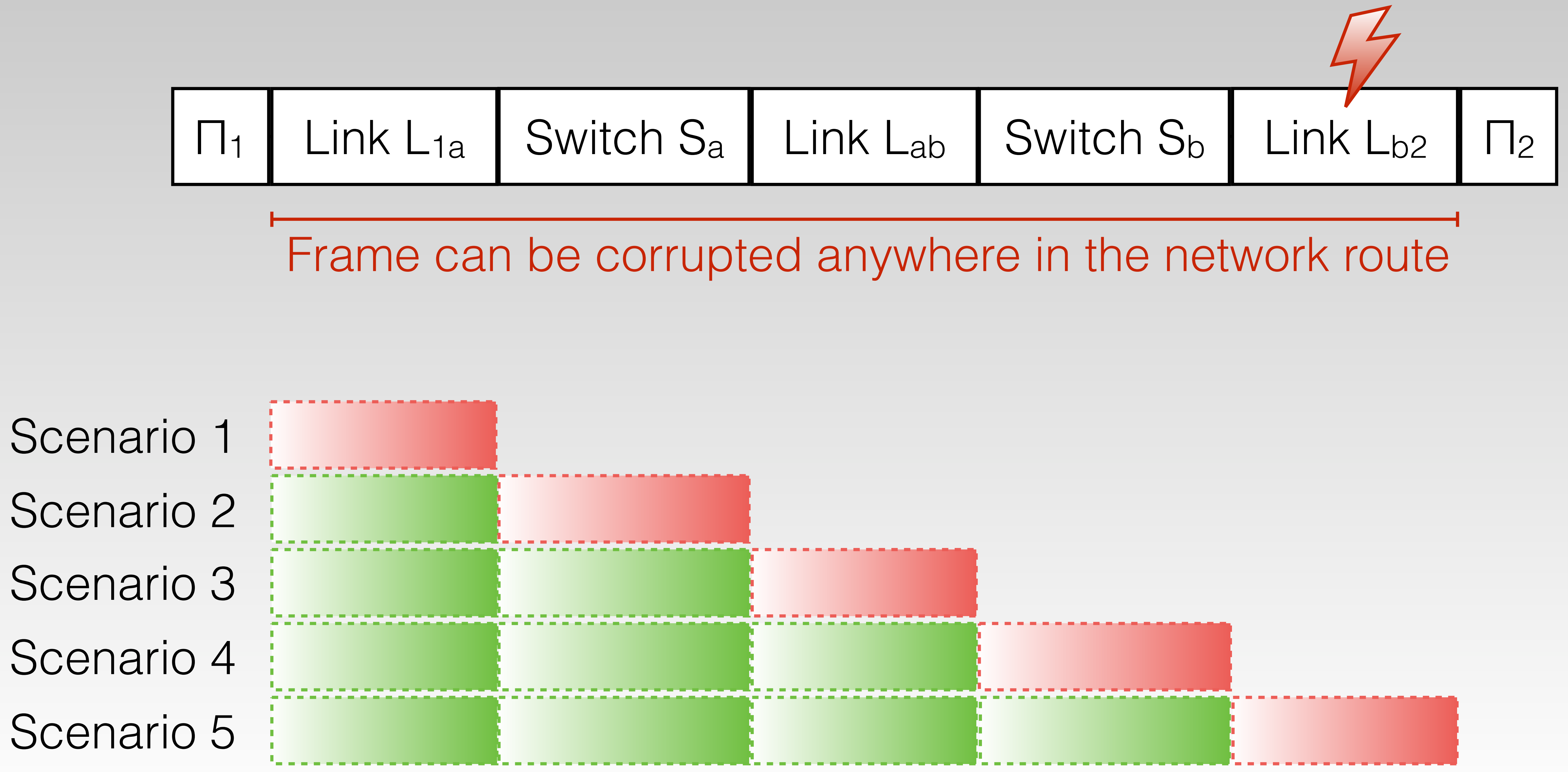
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



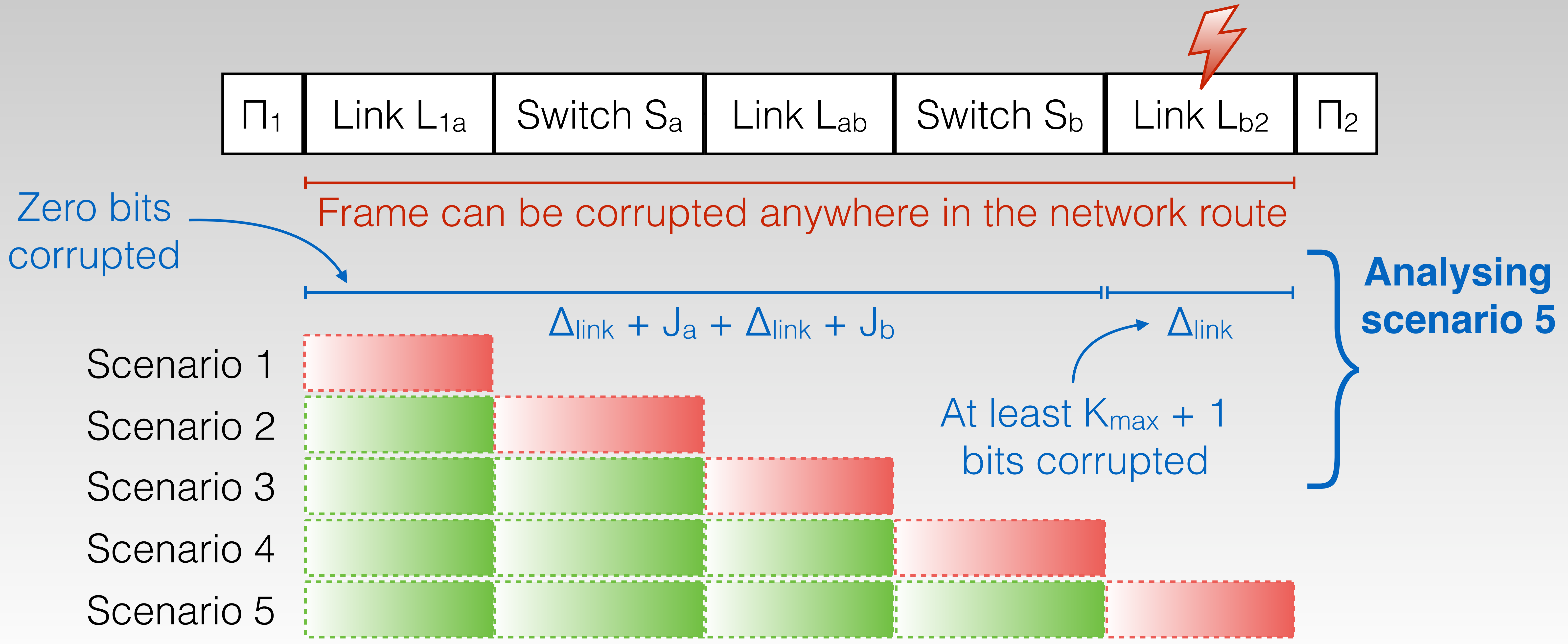
Example: E_3 = Frame carrying Round 1 messages from Π_1 to Π_2 corrupted by the network



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



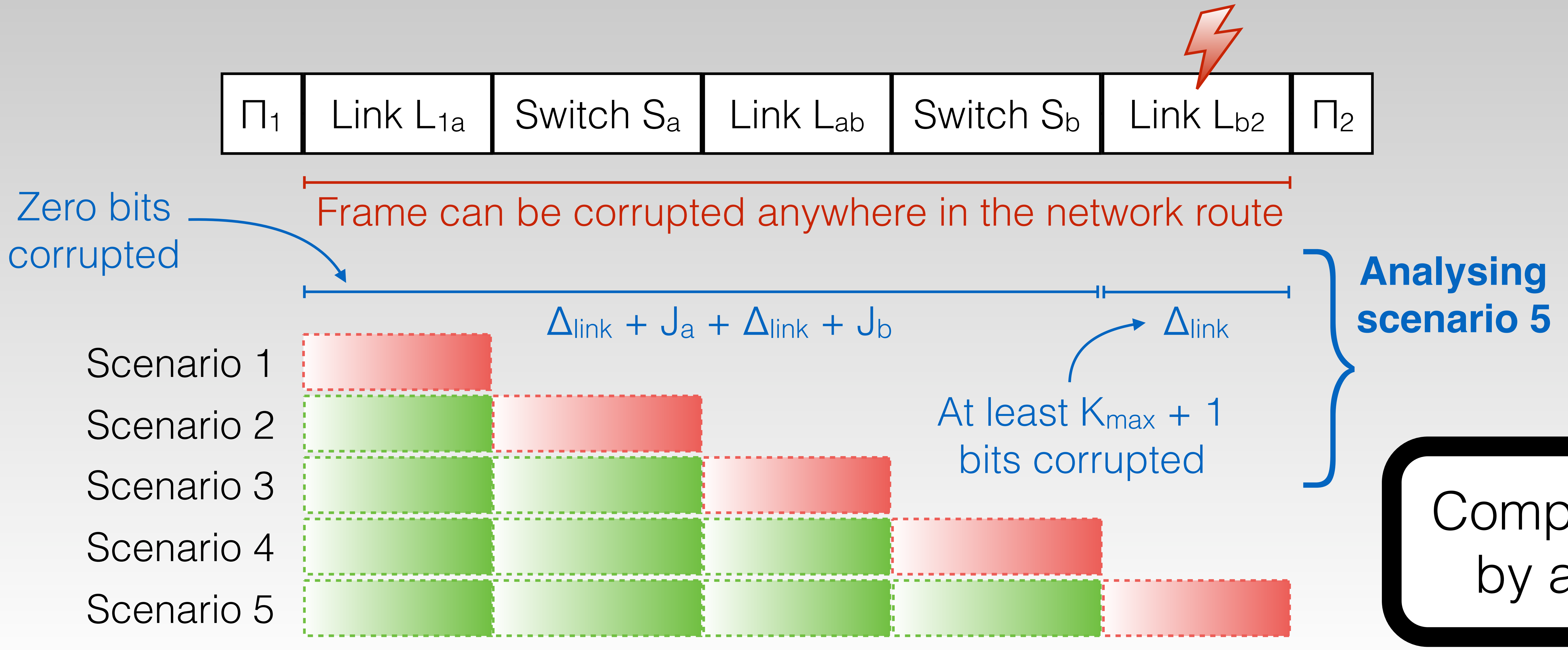
Notation

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 (atomic broadcast failure)

Application-specific message errors

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

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

Combinatorial analysis

Basic Errors + Poisson Model + Timing Analysis

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

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 (atomic broadcast failure)

Application-specific message errors

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

Combinatorial analysis

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

Protocol-specific **pruning**

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

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

...

Goal: P_{UB} > Pr (atomic broadcast failure)

Application-specific message errors

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

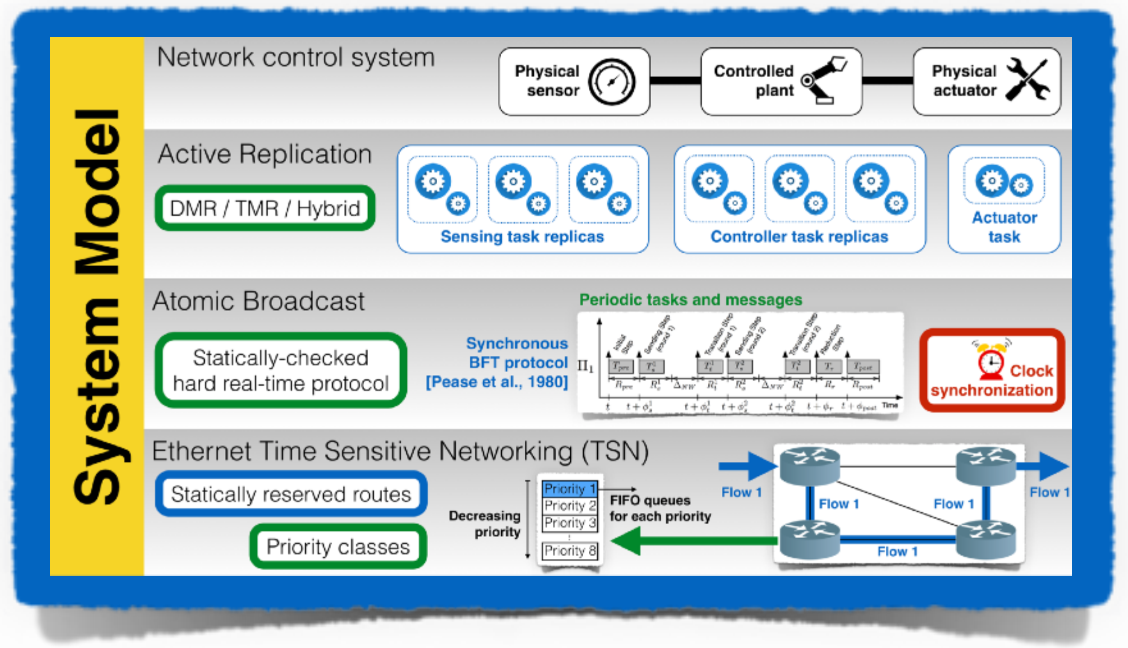
Protocol-specific **pruning**
- Cannot omit a message twice
- Do not care if an omitted message is corrupted

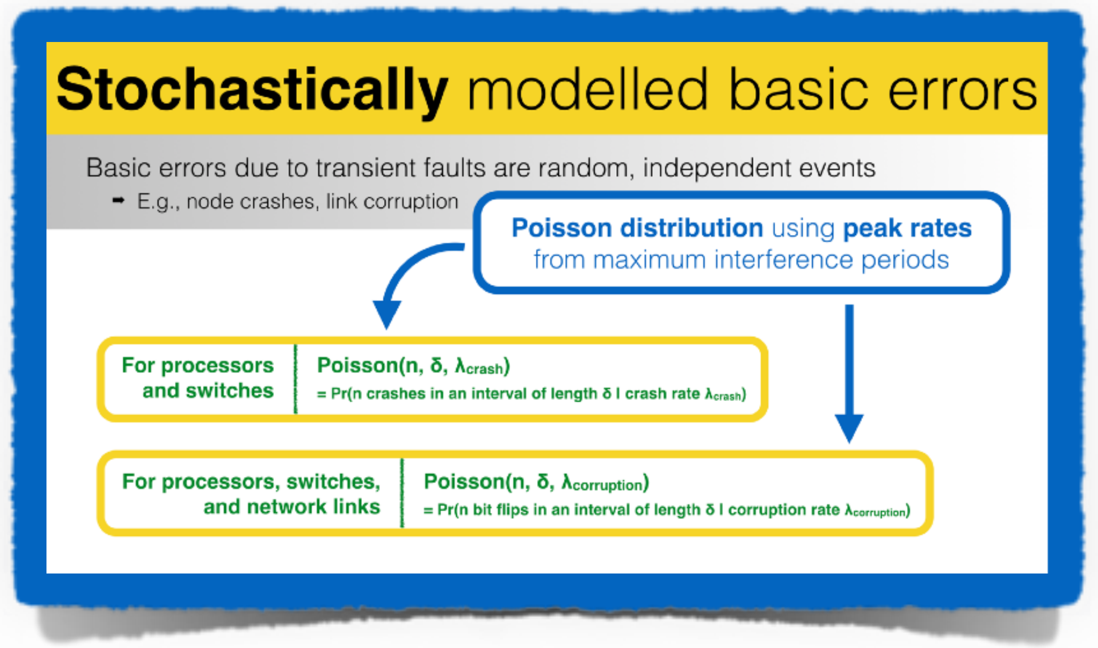
$$\begin{pmatrix} P_1 \cdot C_1 \\ +\overline{P_1} \cdot P_2 \cdot C_2 \\ +\overline{P_1} \cdot \overline{P_2} \cdot P_3 \cdot C_3 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot P_4 \cdot C_4 \\ +\overline{P_1} \cdot \overline{P_2} \cdot \overline{P_3} \cdot \overline{P_4} \cdot P_5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

Combinatorial analysis

Basic Errors + Poisson Model + Timing Analysis

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





Model checking or simulation

Pr (atomic broadcast failure)

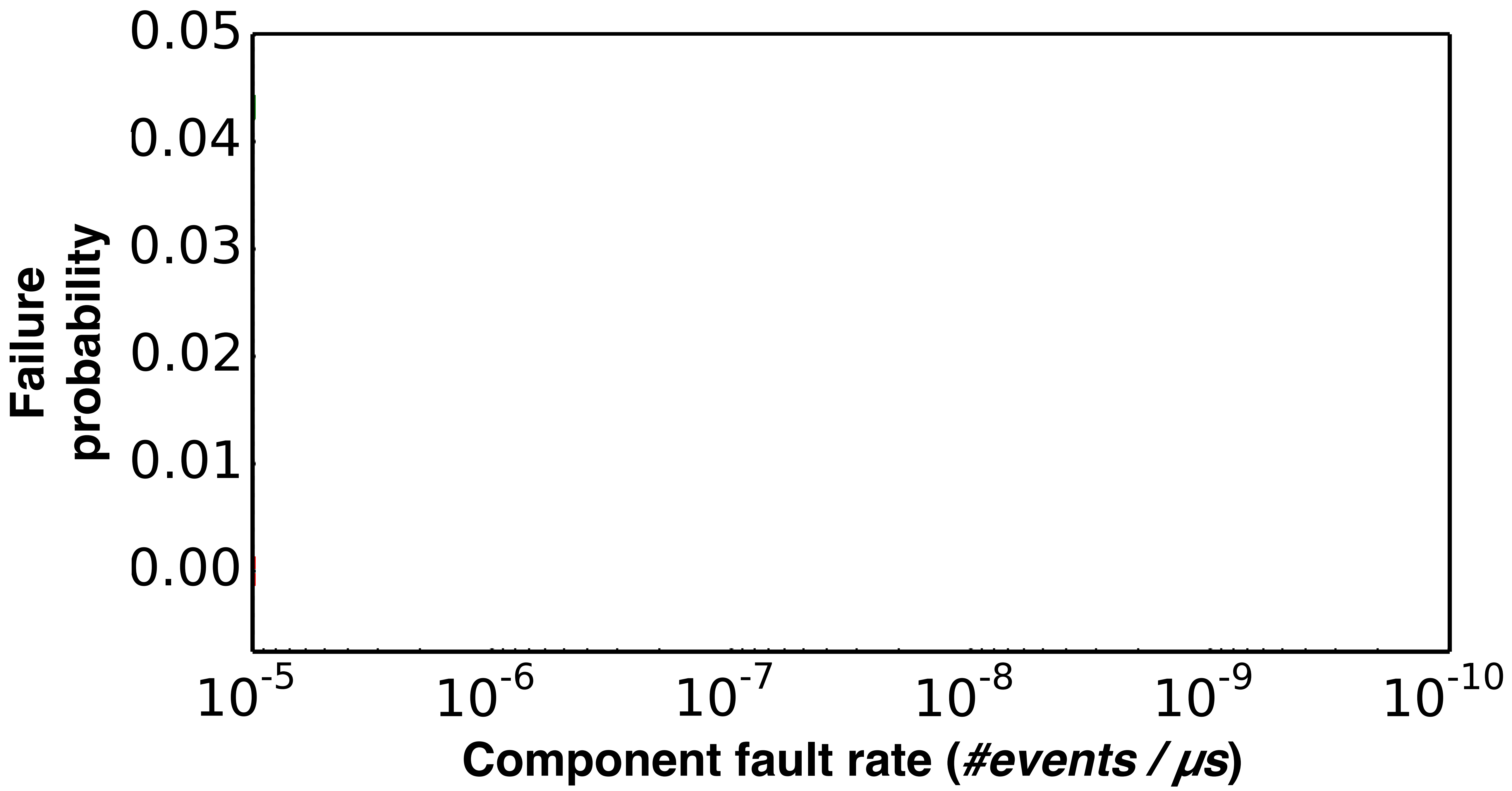
Scalability challenges

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

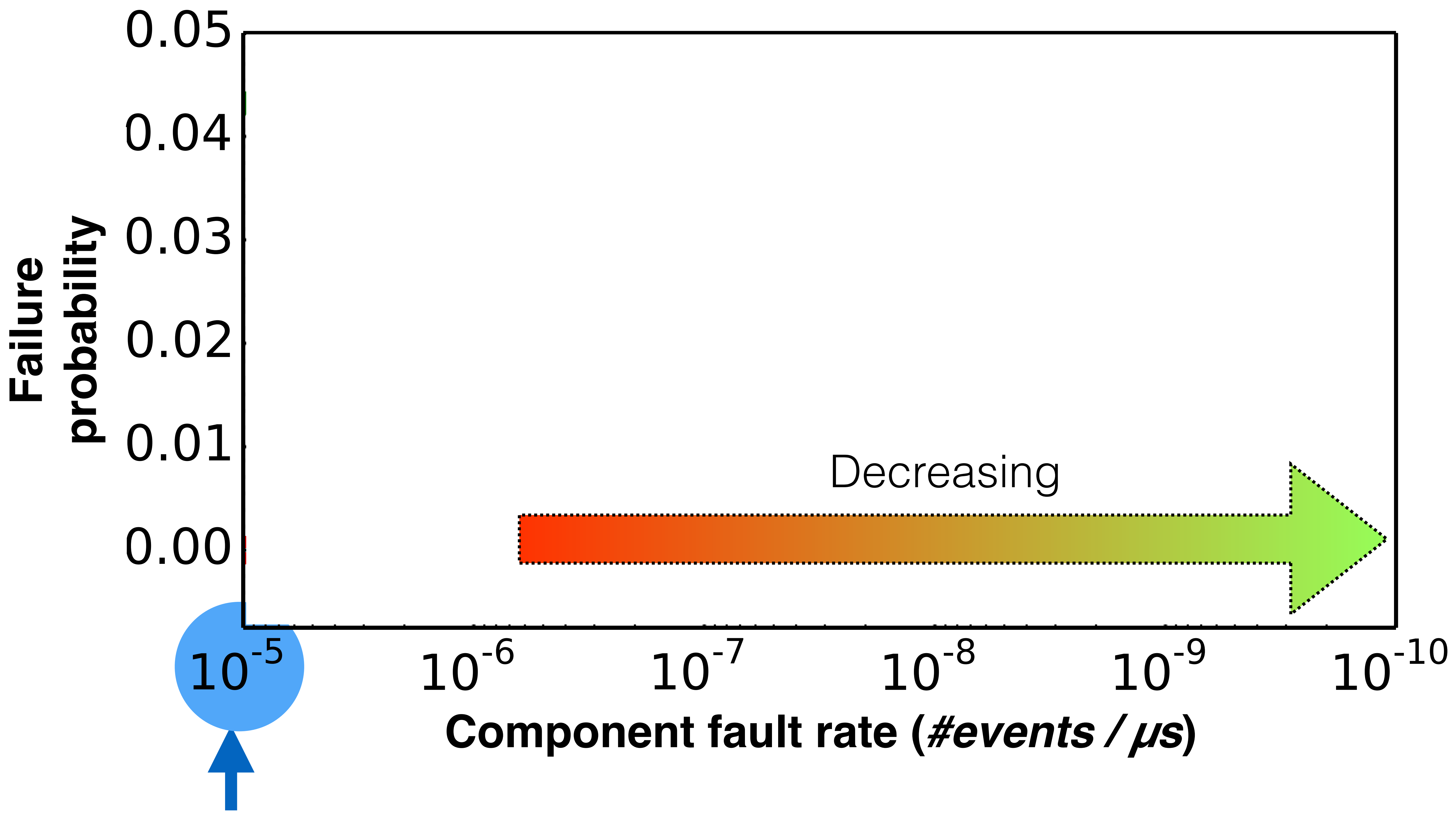
Reliability anomalies

➡ In practice, the failure probability may **significantly exceed** the estimated Pr (atomic broadcast failure)

The problem of **reliability anomalies**



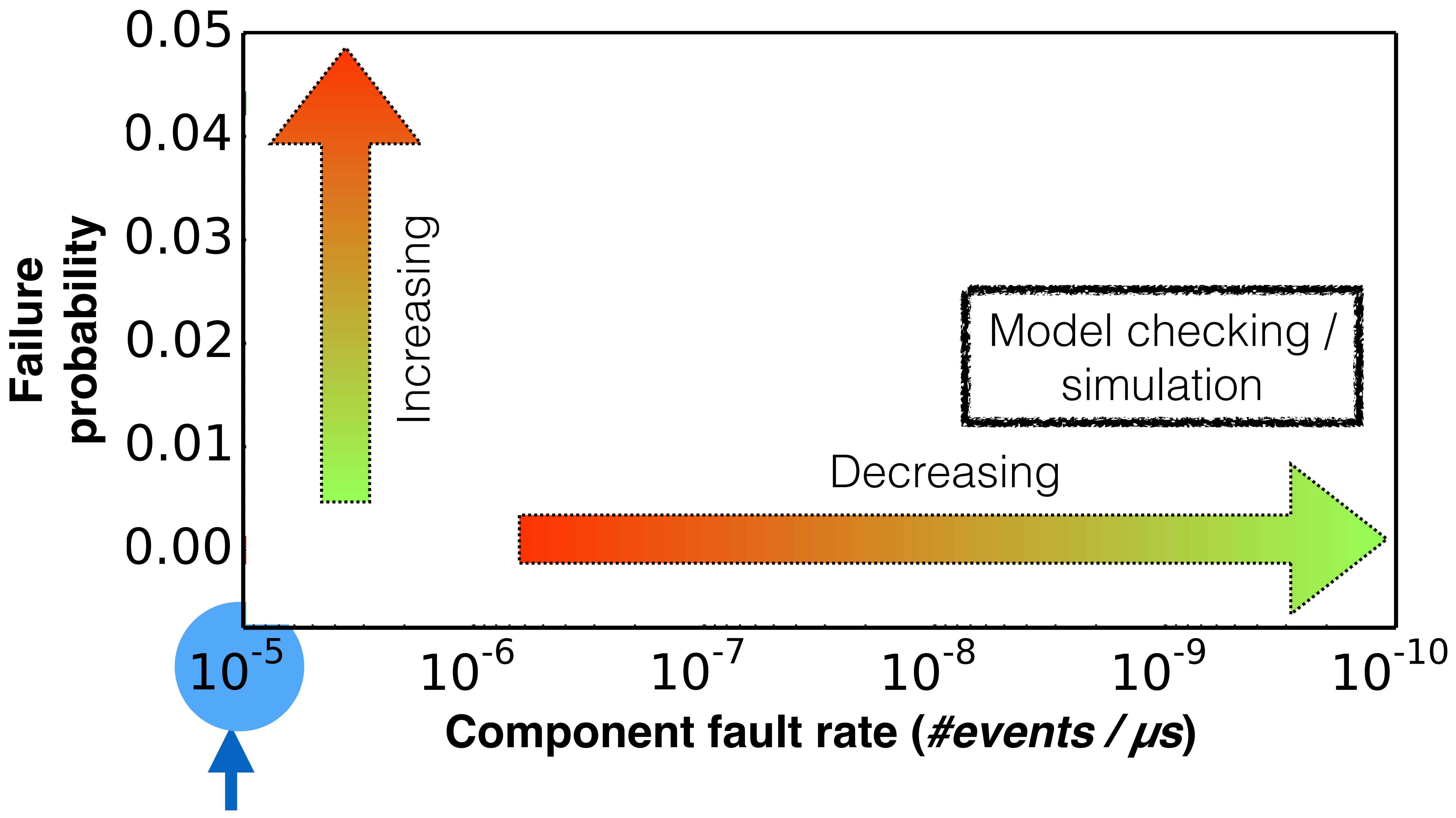
The problem of **reliability anomalies**



Peak fault rate

- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

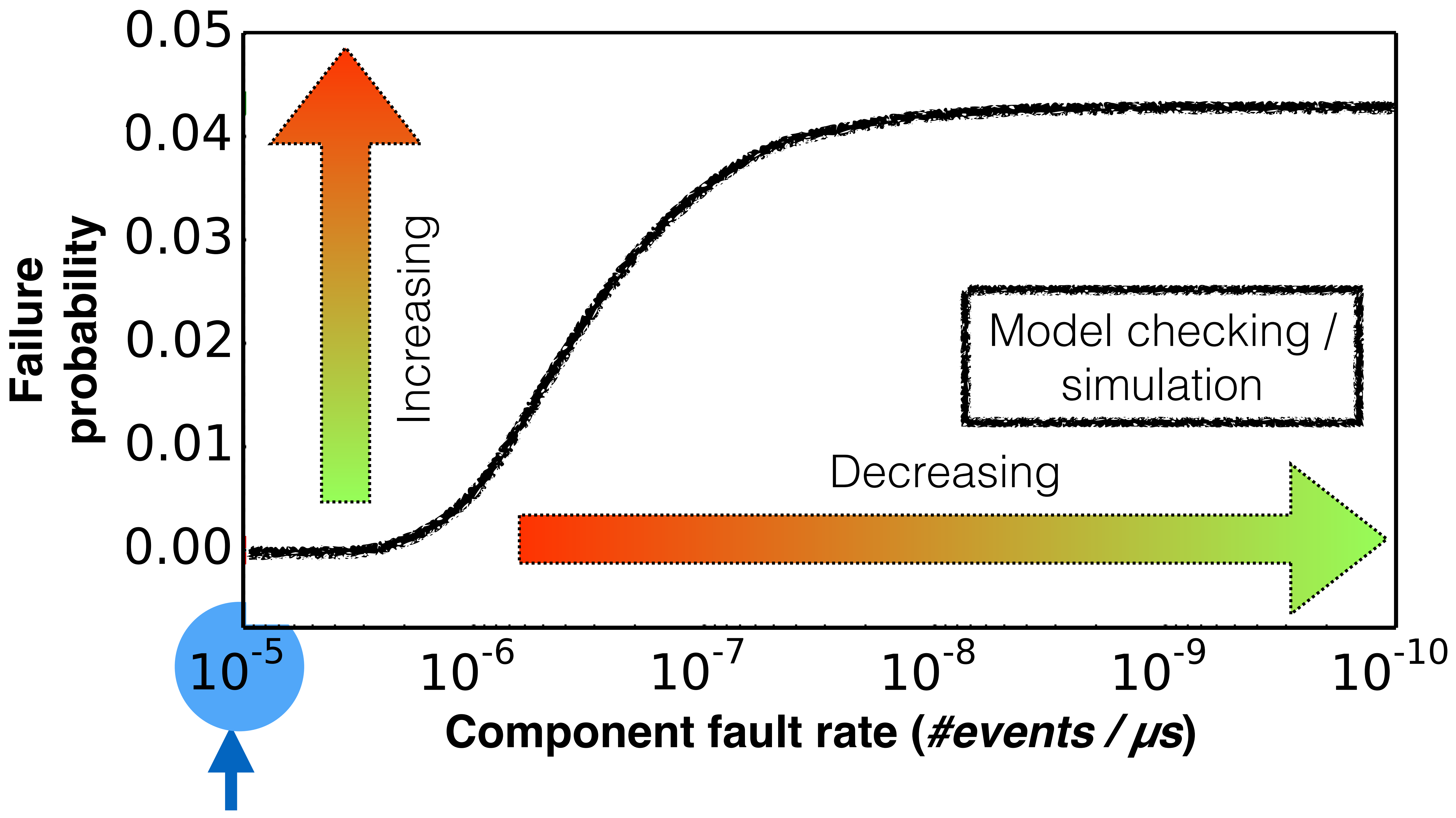
The problem of **reliability anomalies**



Peak fault rate

- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

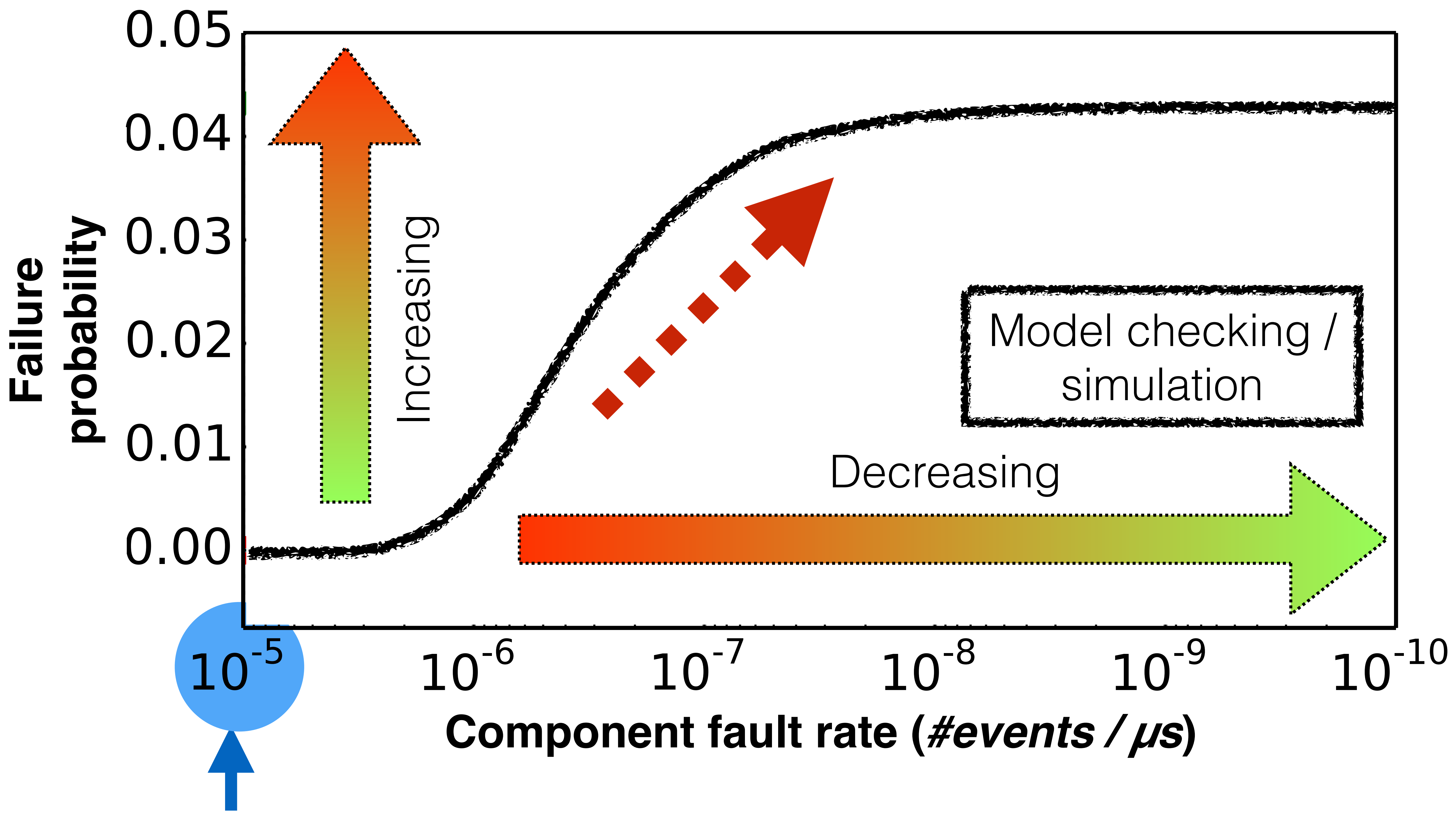
The problem of **reliability anomalies**



Peak fault rate

- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

The problem of **reliability anomalies**



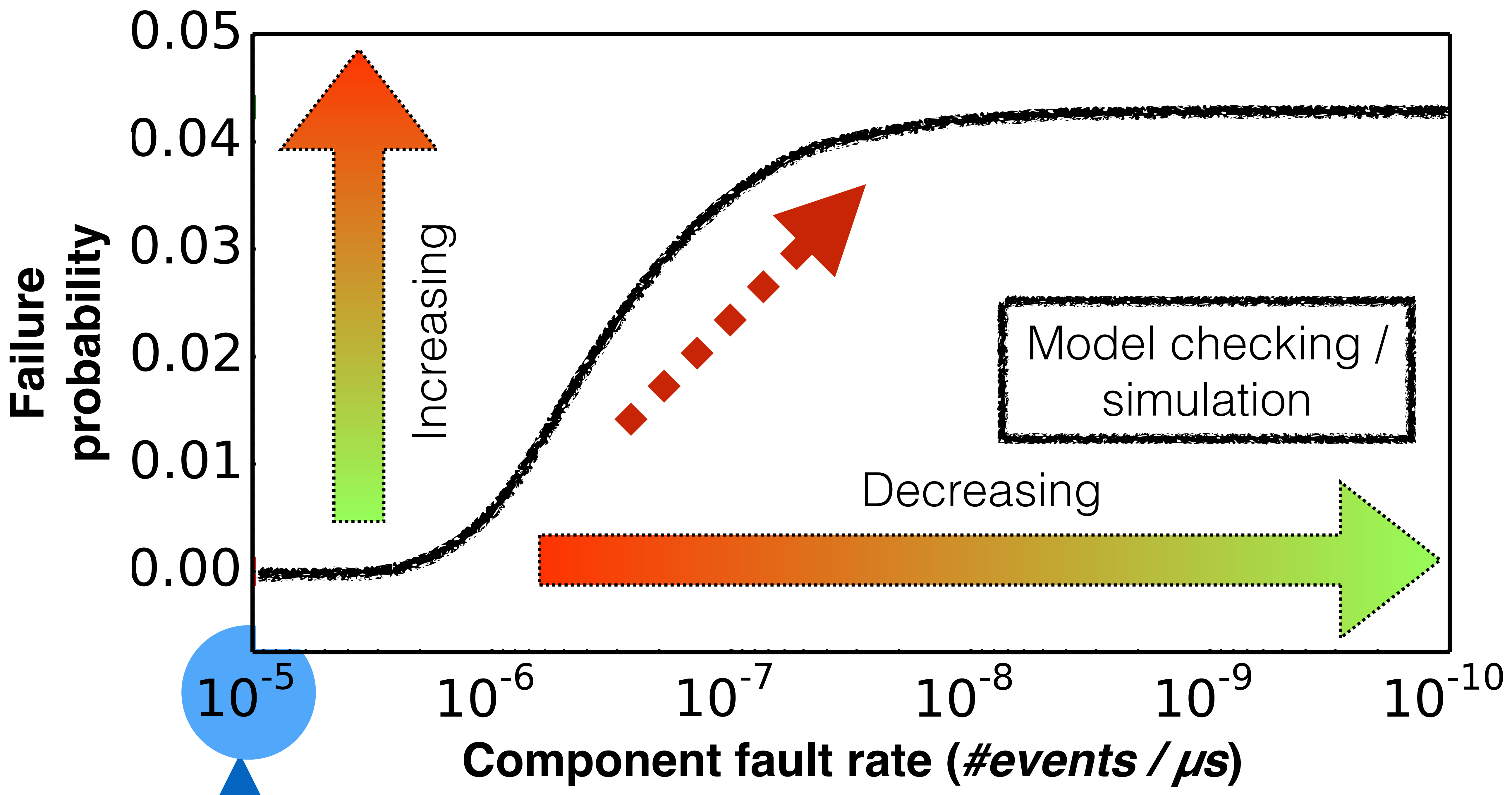
Peak fault rate

- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

Pr (atomic broadcast 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**



Peak fault rate

- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

Pr (atomic broadcast 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

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

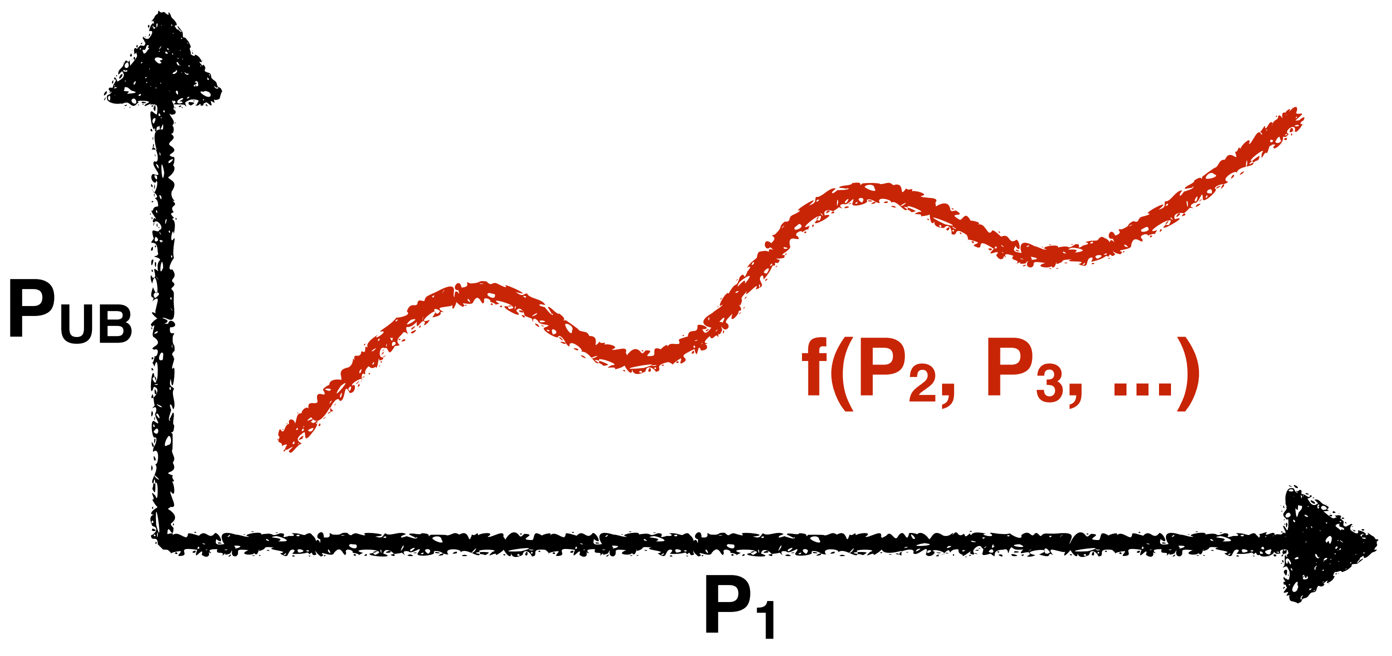
Key idea 2: Ensure **monotonicity** to eliminate anomalies

Combinatorial analysis

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

Root cause of reliability anomalies

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



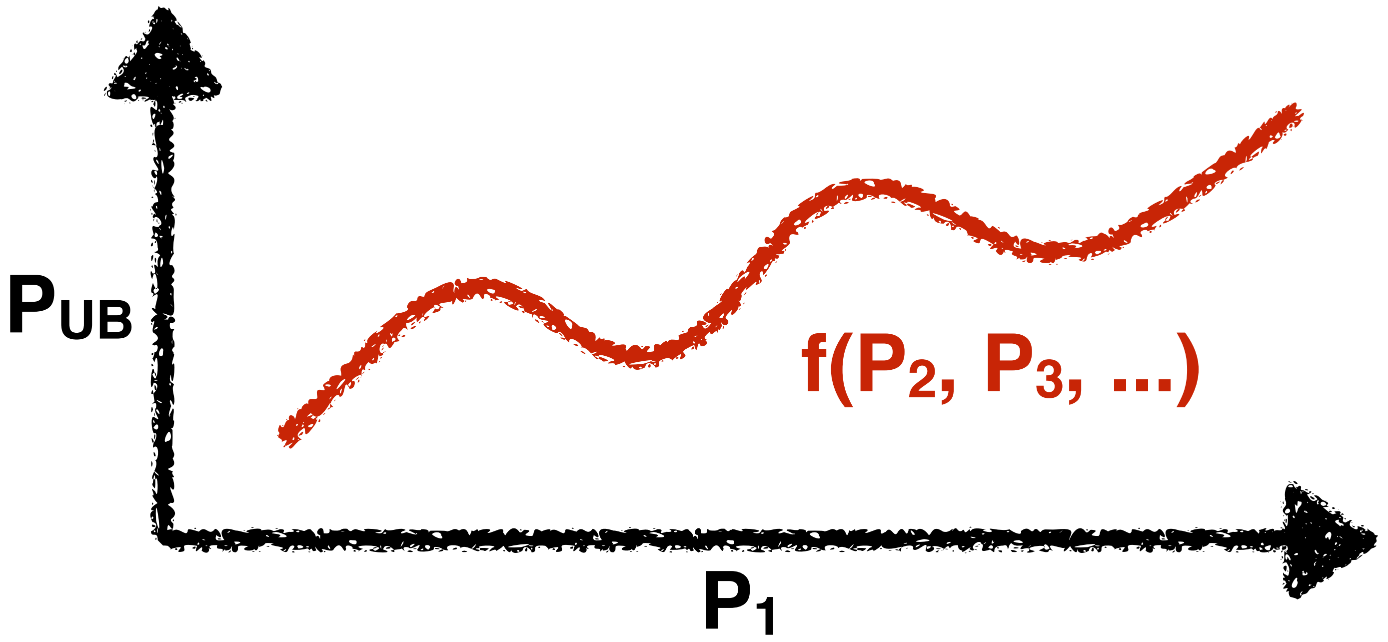
Key idea 2: Ensure **monotonicity** to eliminate anomalies

Combinatorial analysis

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

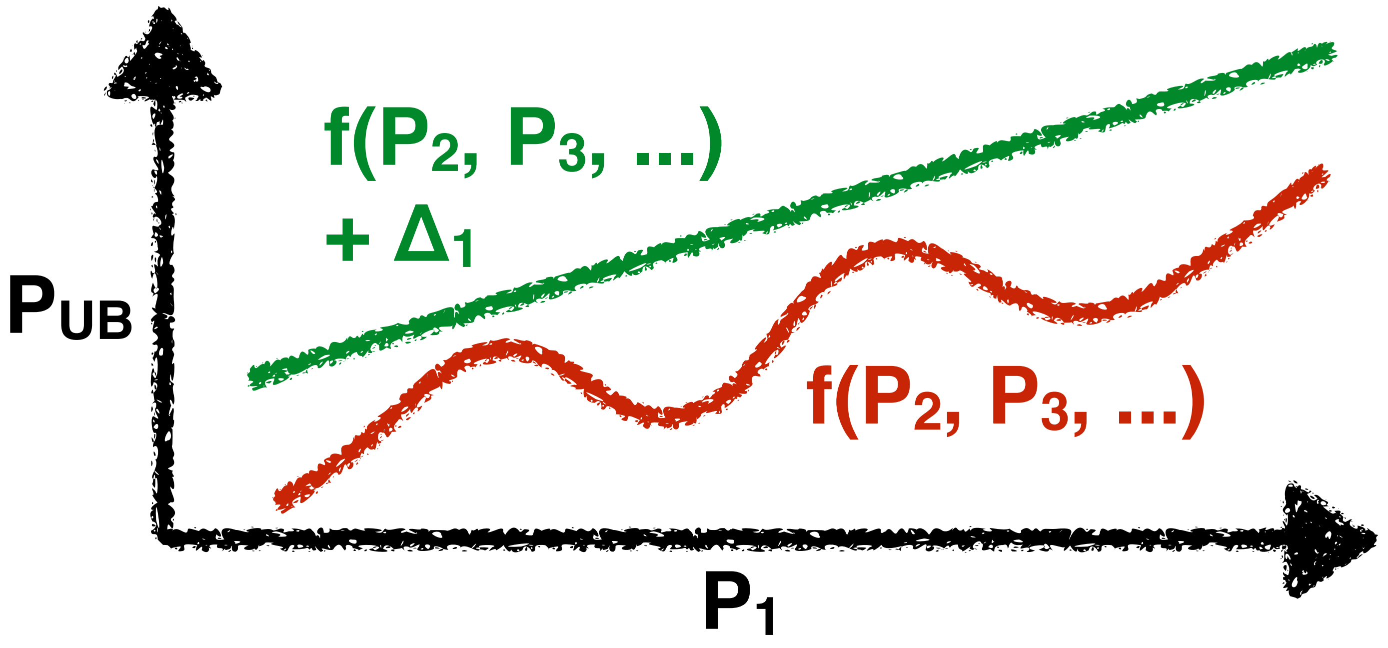
Root cause of reliability anomalies

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

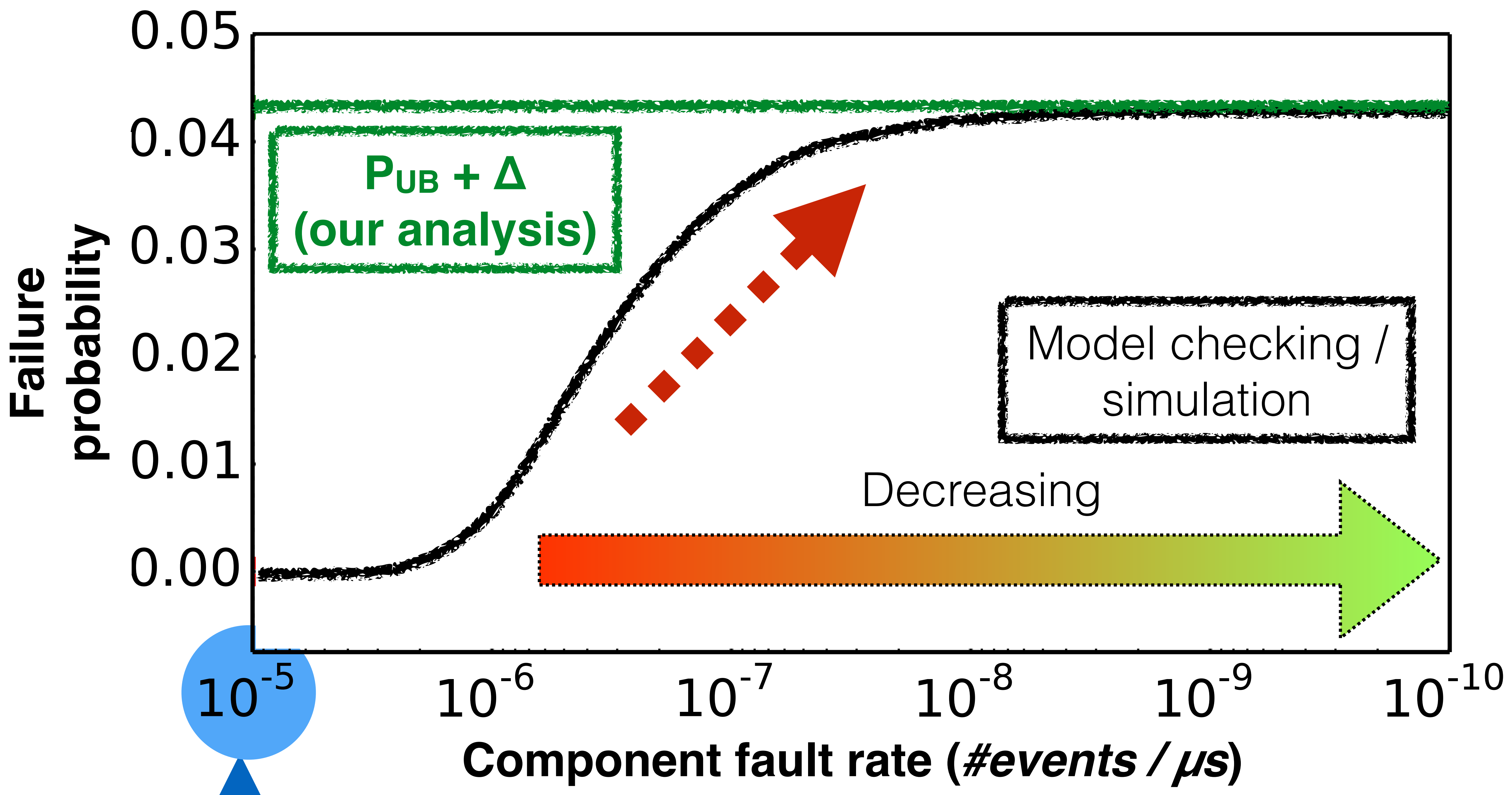


Eliminating reliability anomalies

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



The problem of **reliability anomalies**



Peak fault rate

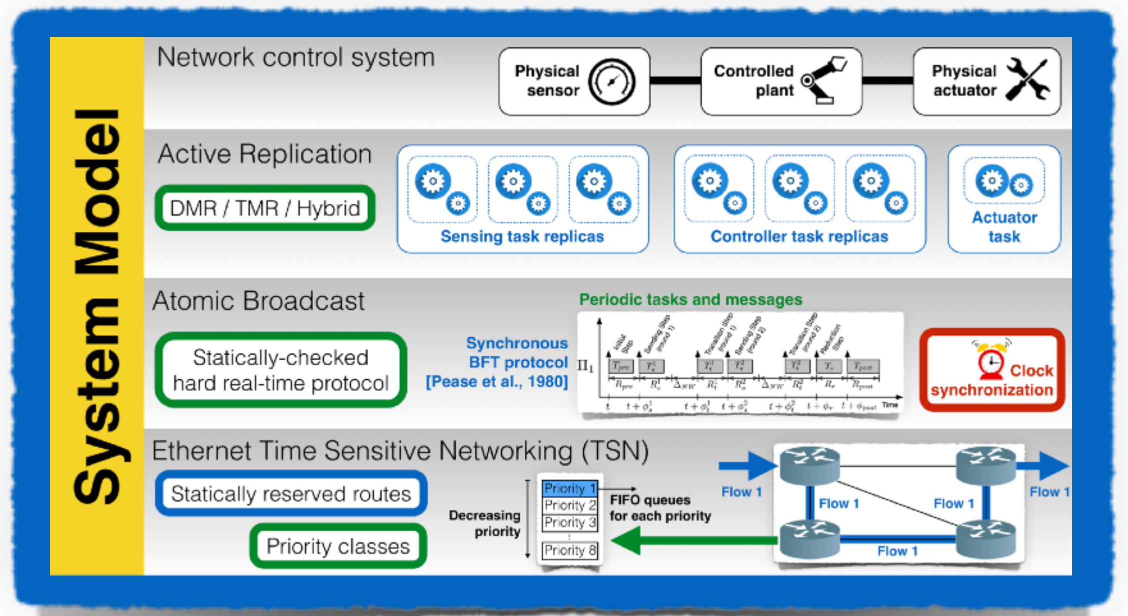
- From measurements / environmental modeling assuming worst-possible operating conditions
- Include safety margins as deemed appropriate by reliability engineers or domain experts.

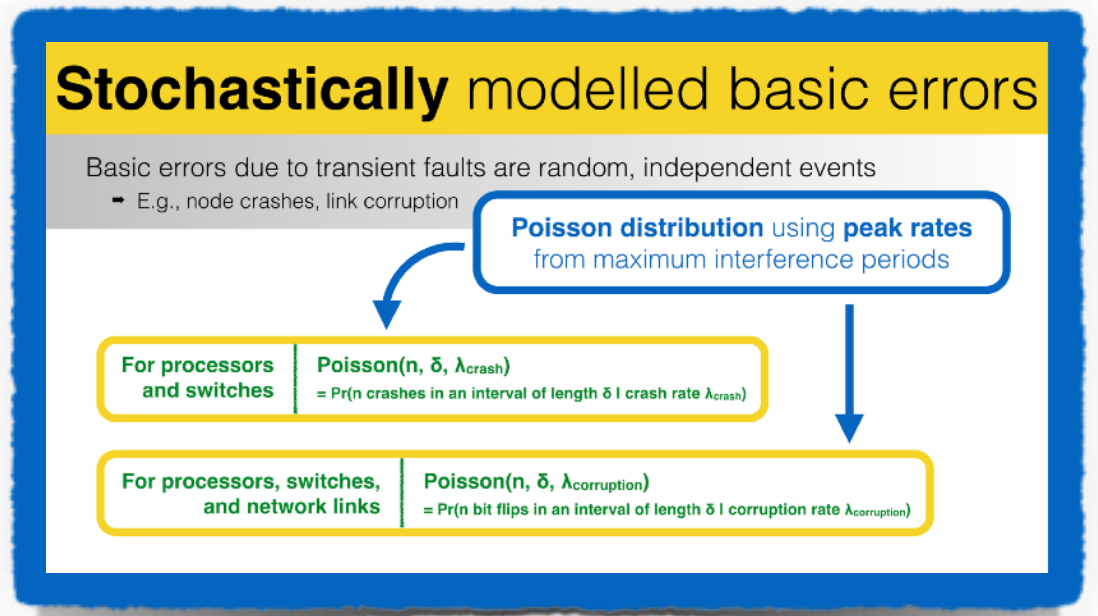
Pr (atomic broadcast failure) increases despite decreasing component fault rate

Intuition: *Sometimes, 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 (atomic broadcast failure)

Scalability challenges

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

Reliability anomalies

Key idea 2: Ensure *monotonicity* to eliminate anomalies

Summary

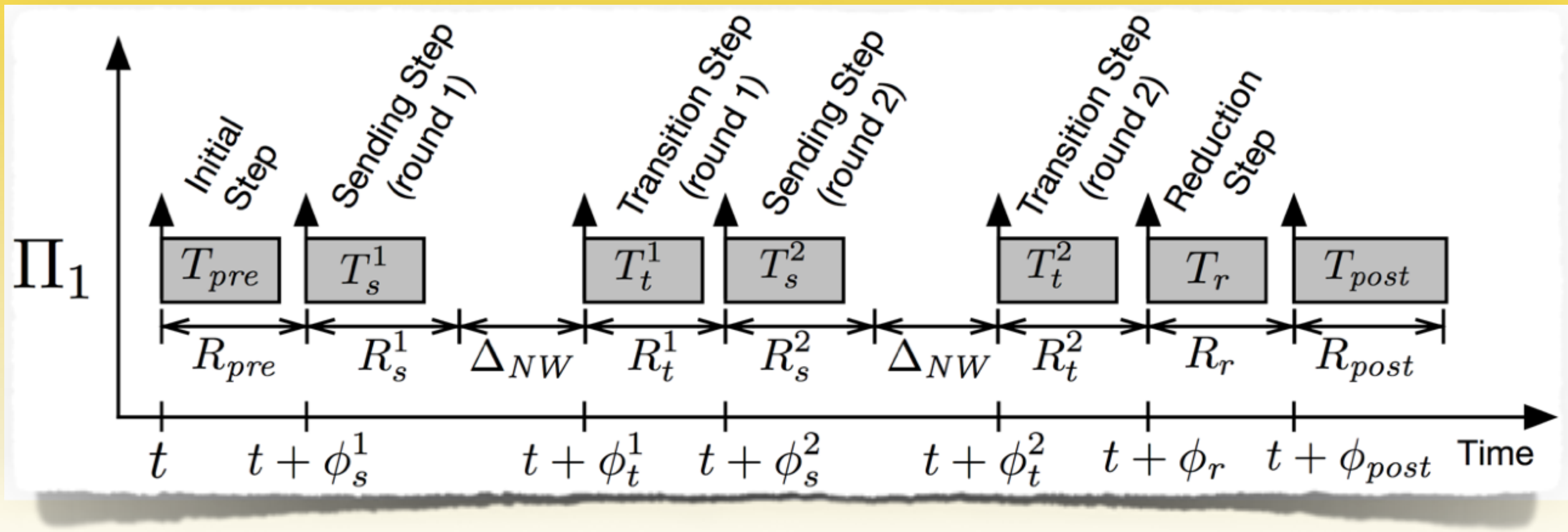
Summary

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

Summary

Ethernet frame corruptions / omissions

Transient fault-induced errors

Omission / incorrect computation at nodes and switches

What is the probability of an **atomic broadcast failure?**

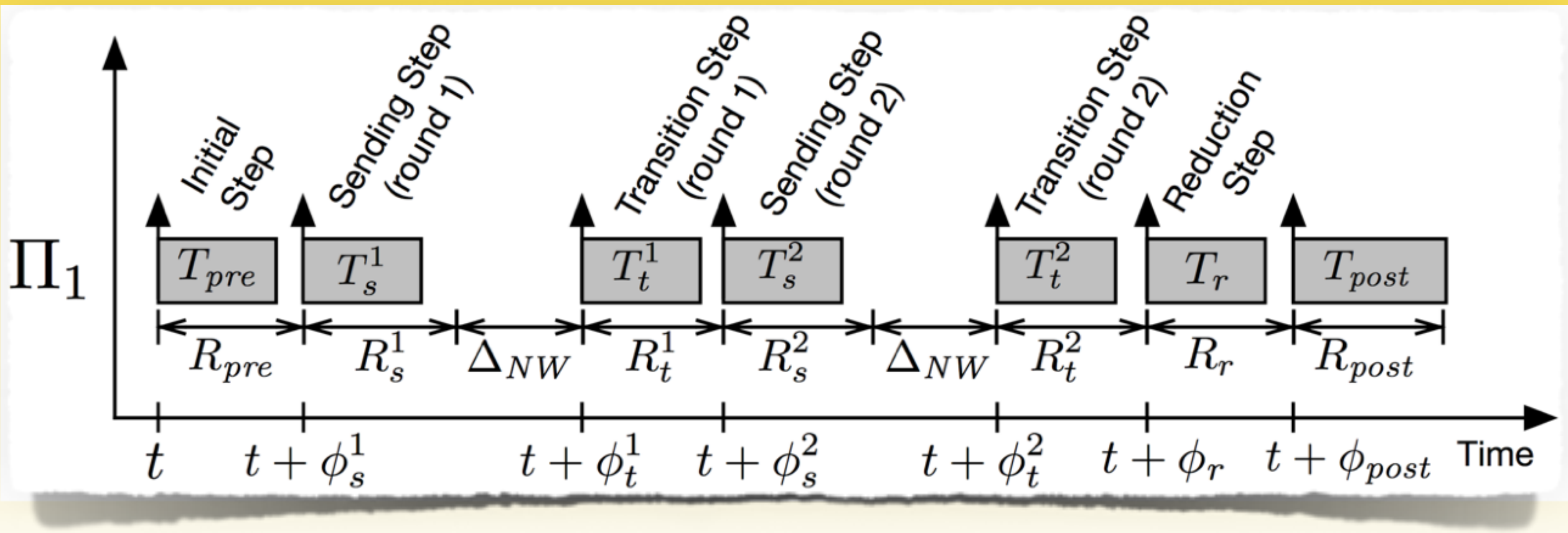
Reliability Analysis

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

Summary

Transient fault-induced errors

Ethernet frame corruptions / omissions

Omission / incorrect computation at nodes and switches

What is the probability of an **atomic broadcast failure?**

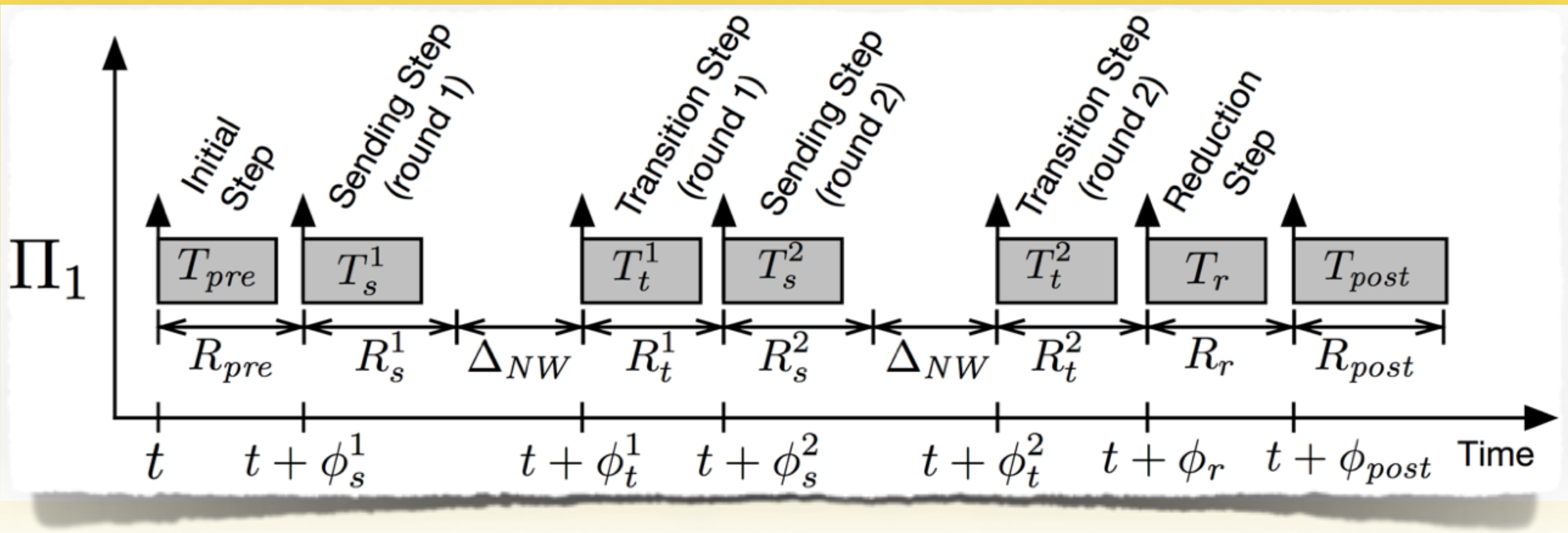
Reliability Analysis

BFT Atomic Broadcast

Statically-checked hard real-time protocol

Synchronous BFT protocol [Pease et al., 1980]

Periodic tasks and messages



Details in the paper!

Clock synchronization

Building safety-critical real-time applications

- COTS-based distributed systems with **quantifiably negligible** failure rates
- Byzantine errors with **non-uniform fault rates** resulting from transient faults
- Formalize and eliminate **reliability anomalies**

In the paper ...

Parameterized BFT interactive consistency protocol

Time-aware correctness criteria

Reliability anomalies formalization for arbitrary configurations

Analysis versus simulation experiments

Case studies with varying network topologies and protocol parameters

In the paper ...

Parameterized BFT interactive consistency protocol

Time-aware correctness criteria

Reliability anomalies formalization for arbitrary configurations

Analysis versus simulation experiments

Case studies with varying network topologies and protocol parameters

Thank you!

arpanbg@mpi-sws.org