Healing and Dealing

Self-* properties in Networked Systems

Johan Lukkien, Ehsan Warriach, Sila Ozen, Tanir Ozcelebi

Department of Mathematics and Computer Science

System Architecture and Networking Group

ProHeal

Technische Universiteit Eindhoven University of Technology

Where innovation starts

Healing or Dealing?

Self-* properties in Networked Systems

Johan Lukkien, Ehsan Warriach, Sila Ozen, Tanir Ozcelebi

Department of Mathematics and Computer Science

System Architecture and Networking Group

ProHeal

Technische Universiteit Eindhoven University of Technology

Where innovation starts

ProHeal partners

Project partners with funding

TU/e Technische Universiteit Eindhoven University of Technology
Where innovation starts

DEVLAB DEVELOPMENT LABORATORIES

Ware Valley
EXCELLENCE IN IT SOLUTIONS

ITEA 3

Project partners without funding

SIVECO
Project: Progress™

ALTFACTOR
EDUCATIONALLY YOURS

SAN

TU/e

Pro-heal

Automated Protection and Healing Software Solutions

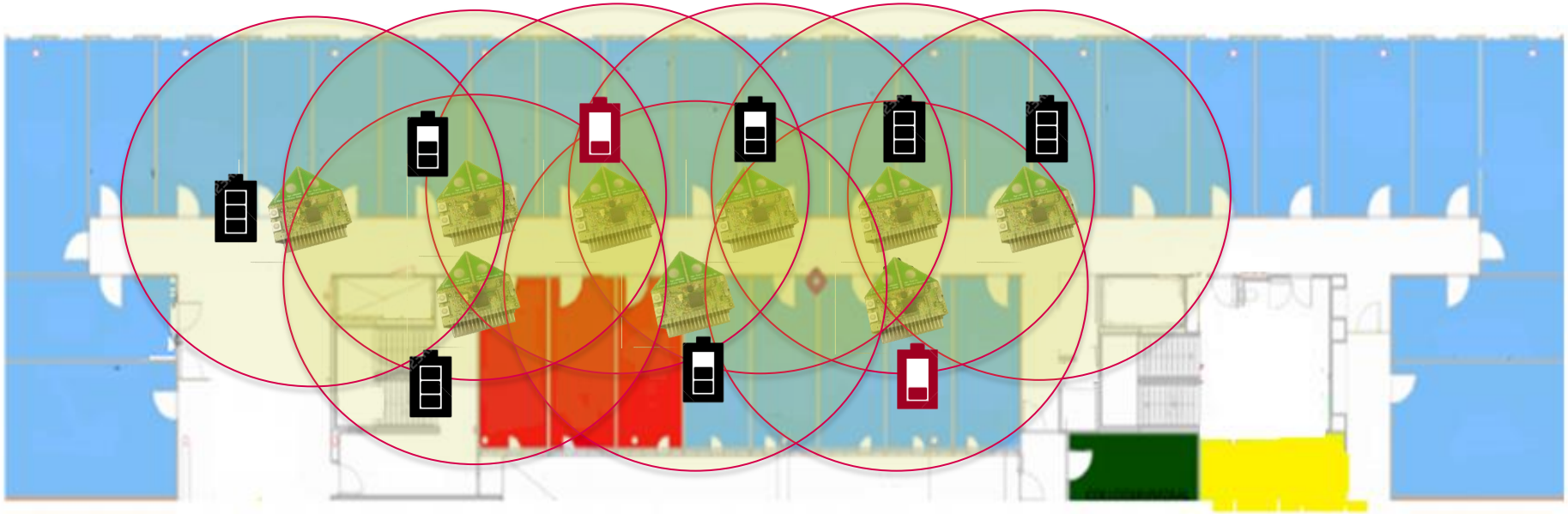
- The Pro-heal European ITEA project aims to improve operation of smart environments by reflective (self-*) technology
  - Self-organization
  - Self-configuration
  - Self-aware
  - Self-healing
  - Self-protection

SAN

TU/e

Two examples

Consider a battery-powered network of sensor nodes

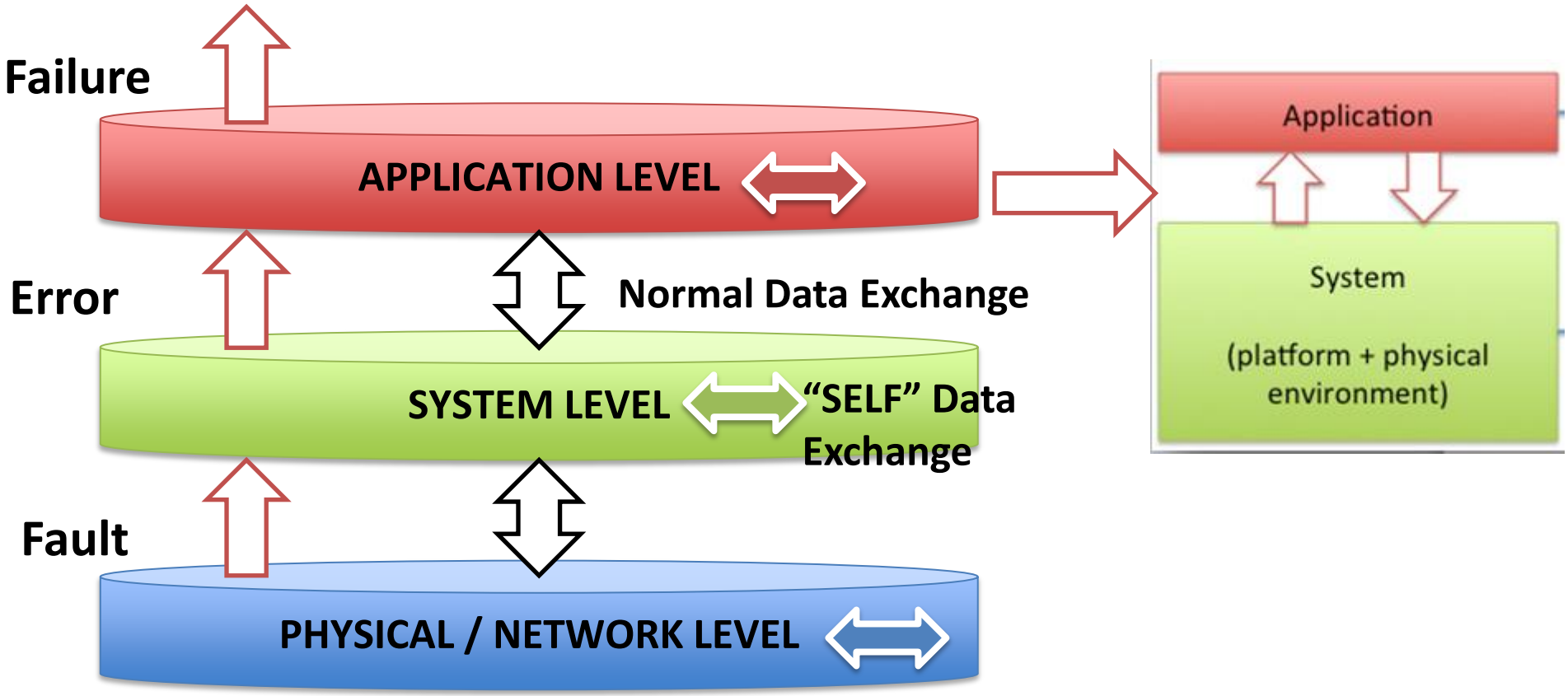


1. battery lifetime is limited → predict lifetime and adjust system activity
2. at times, communication is hindered → recognize failing states and adjust accordingly

SAN

TU/e

Goal: avoid application failure



Self-healing

- *Self-healing*: ability of a system to perceive that it is not operating correctly and to restore itself to (normal) operation without human intervention

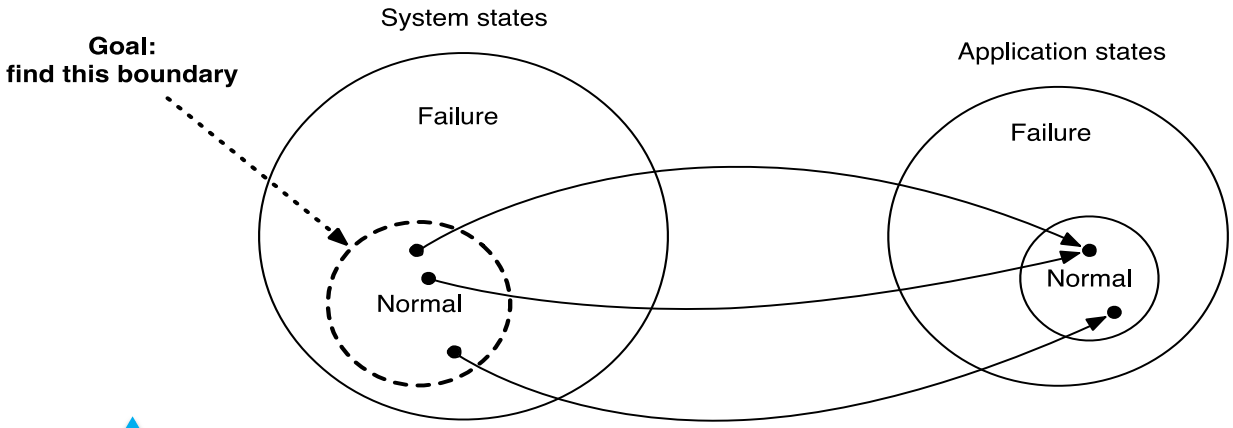
- We interpret this with respect to the correct operation of the application
  - the system must support an application's requirements
    - services, resources
  - 'system not operating correctly': danger of application failure
    - (= system error state)

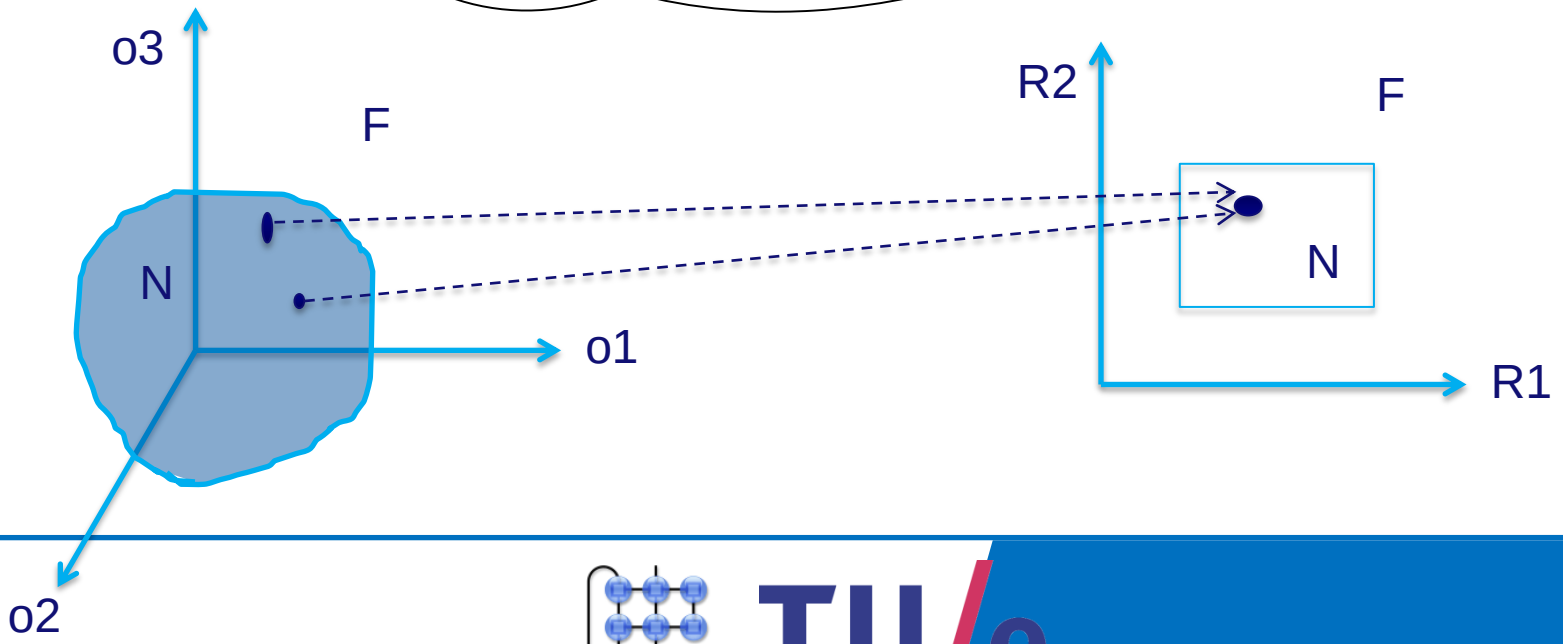
- Example application: smart lighting:
  - sustain correct behavior of the system in response to users (including timing)

Approaches: self-healing

- (Fault tolerance techniques)
- Reactive mechanisms: detect anomaly → recover
- Proactive mechanisms: predict anomaly → adapt
  - self-healing: predict fault → prevent failure

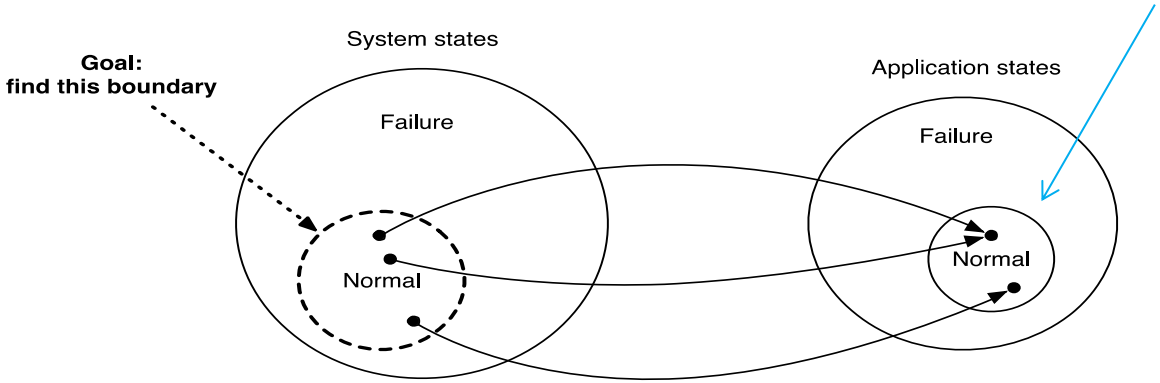
Relate system states to application states





Challenges for self-healing

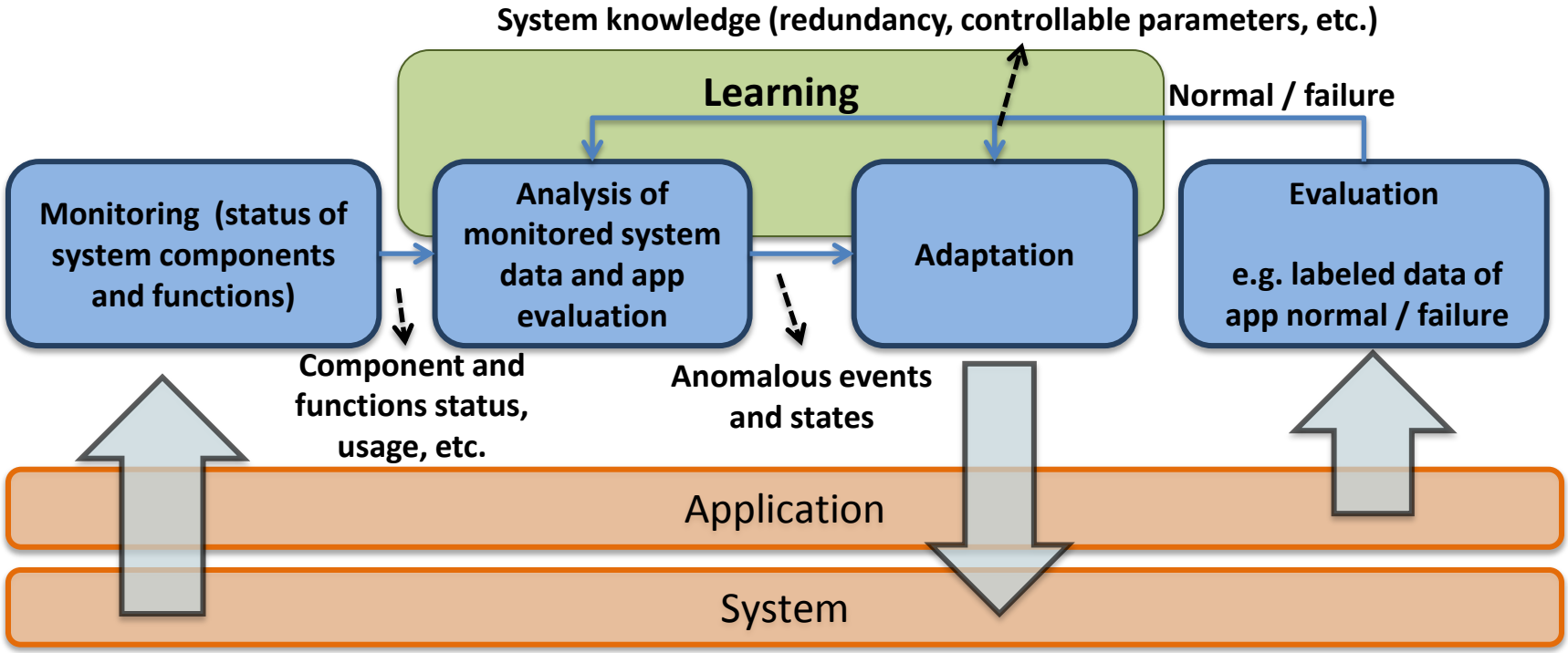
- The system needs to learn application failure boundaries.



- The system needs to learn the *boundaries of normal and failure states* of system resources and functions while running an application (corresponding to application normal states).

- The system needs to learn *effective adaptation policies* for maintaining normal system states.

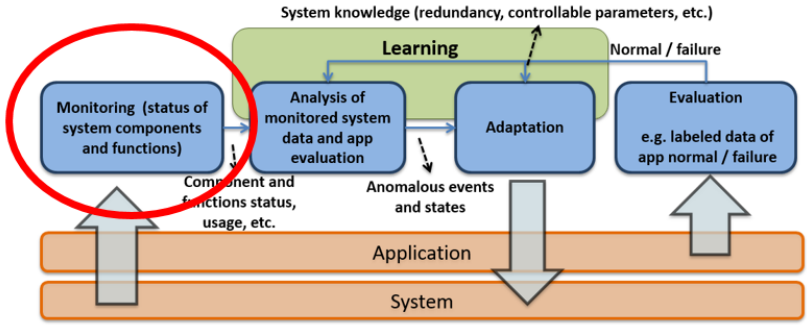
Proactive Dependability Framework

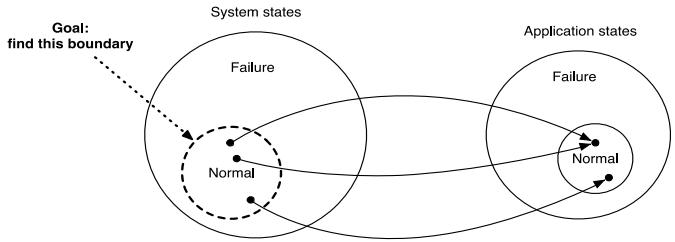


Parameters

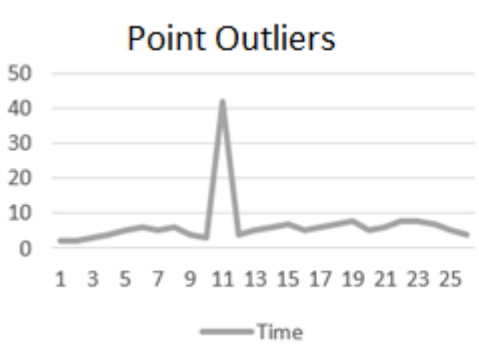
- Observable
  - in the system: internal status, low-level observations
  - in the application: metrics, outcomes
- Controllable
  - in the system: internal settings
  - in the application: settings of quality, control over running components

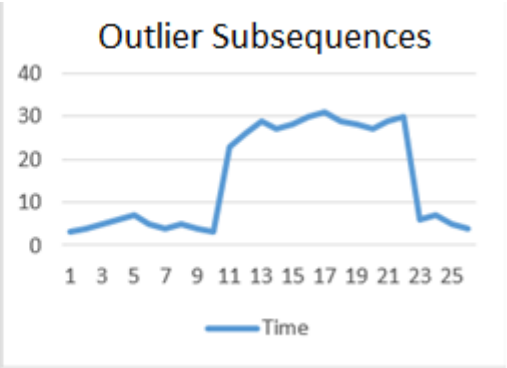
Monitoring and outliers





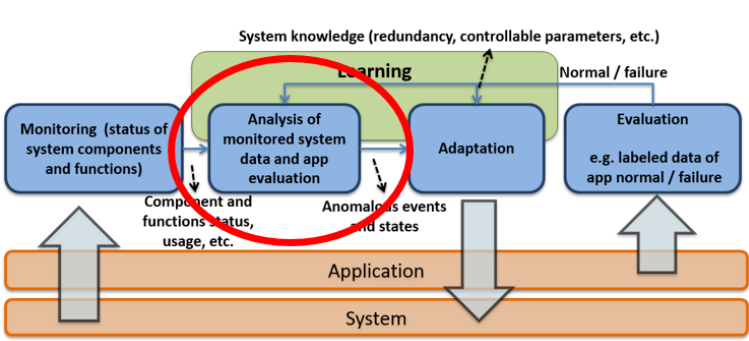
- Monitoring yields a time series
- Outliers: measurements falling in ‘Failure’
  - Point outliers
  - Outlier subsequences (Discords)
- Need sufficient evidence before taking measures
  - one outlier may not be enough
  - take history into account



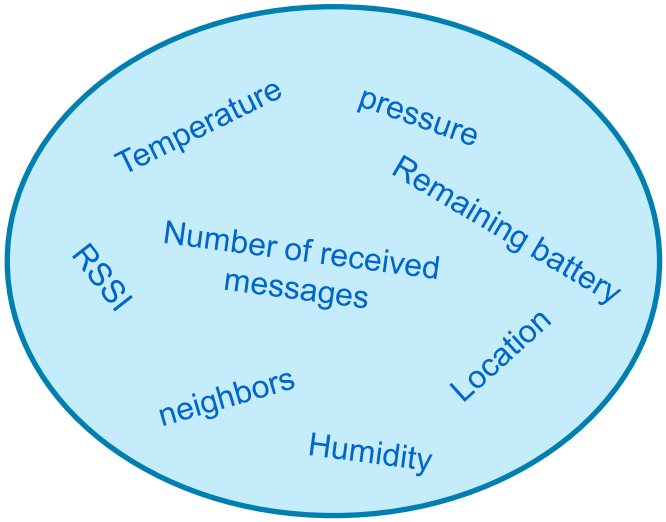


SAN TU/e

Model: failure detection

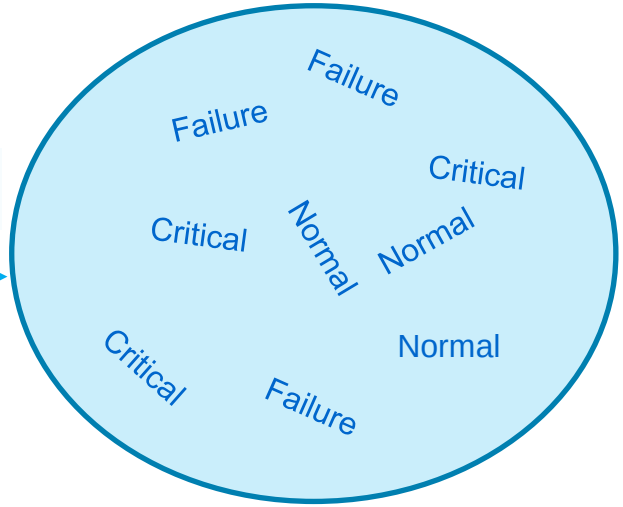


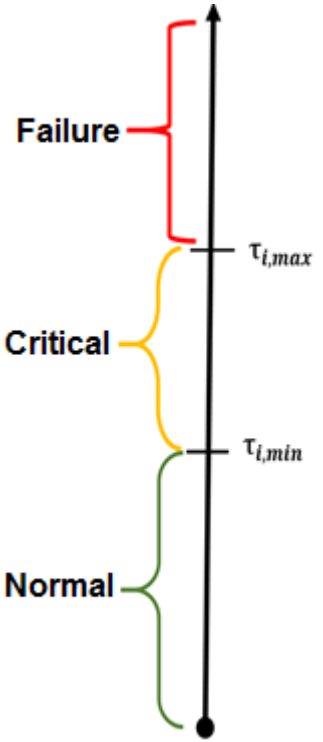
Observable Parameters



Outlier Scoring

Outlier Scores

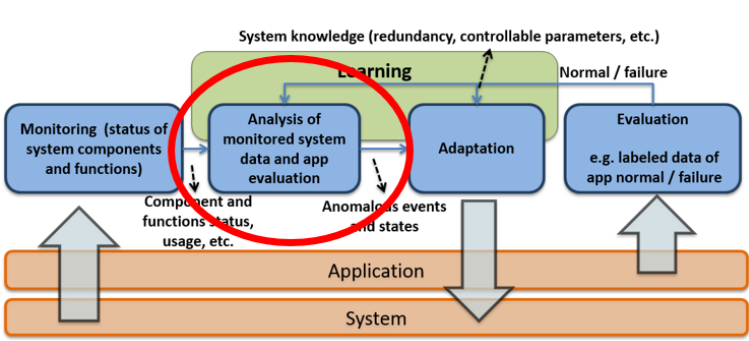




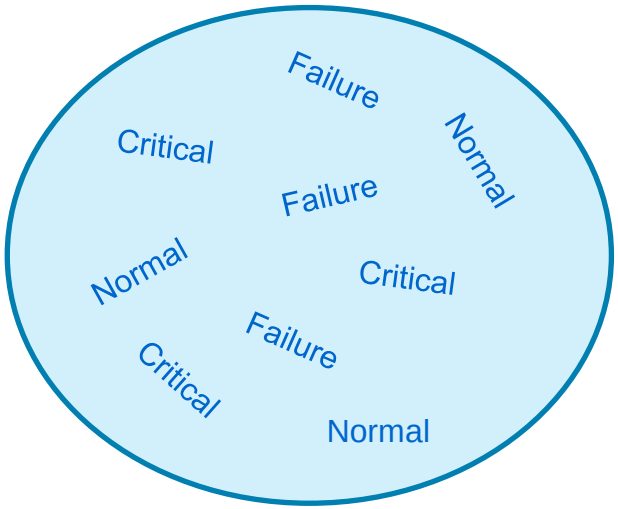
SAN

TU/e

Model: failure detection



Outlier Scores



Time series

[critical, failure, failure, normal, critical, normal, ……, failure, critical]

Subsequence Analysis

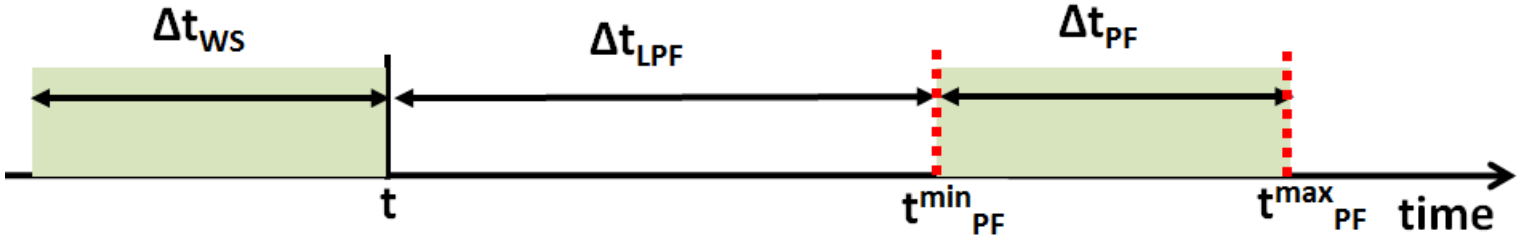
Rules to be used at runtime

SAN

TU/e

Model: failure/fault prediction

Fault-prediction: {F_{type}, t_{PF}^{min}, t_{PF}^{max}}



Where,

- F_{type} : type of predicted fault;
- $[t_{PF}^{min}, t_{PF}^{max}]$, the period in which a fault is predicted to take place;
- t: current time instance;
- Δt_{WS} is the window size (how far into the past we look)

Performance metrics for fault prediction model

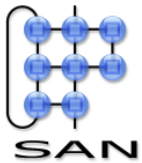
Performance Metrics	Formula
Mean Absolute Error (MAE)	$\frac{1}{n}\sum_{t=1}^{n}\lvert x_t - \hat{x}_t\rvert = \frac{1}{n}\sum_{t=1}^{n}\lvert e_t\rvert$
Root Mean Square Error (RMSE)	$\sqrt{\frac{\sum_{t=1}^{n}(x_t-\hat{x}_t)^2}{n}}$
Mean Square Error (MSE)	$\frac{\sum_{t=1}^{n}(x_t-\hat{x}_t)^2}{n}$
Relative Square Error (RSE)	$\frac{\sum_{t=1}^{n}(\hat{x}_t-x_t)^2}{\sum_{t=1}^{n}(\bar{x}-x_t)^2}$
Root Relative Square Error (RRSE)	$\sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t-x_t)^2}{\sum_{t=1}^{n}(\bar{x}-x_t)^2}}$
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F-measure	$\frac{2*precision*recall}{precision+recall}$

Case Study 1: detect communication faults

- MyriaNed WSN platform
  - 8 sensor nodes + sniffer node (sink)
- WSN in indoor environment

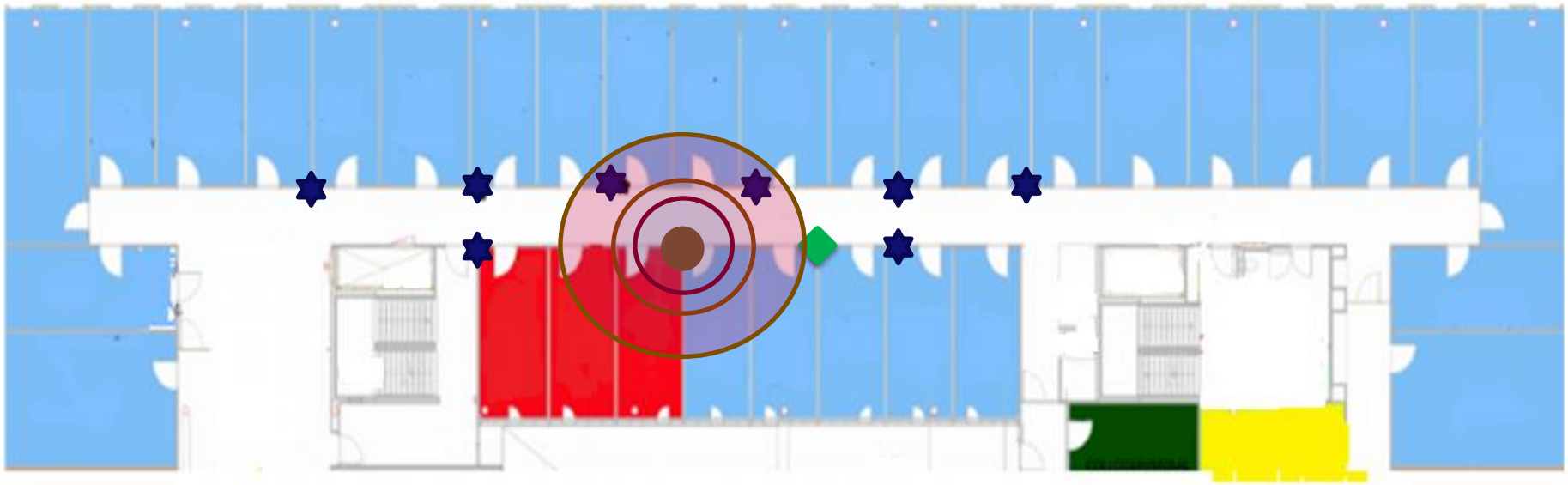
- Injected Communication disturbance:
  - jammer node causes packet loss

- Repeatedly, for days, alternate two different settings
  1. WSN with jammer node
  2. WSN without jammer node



TU/e

Experiment setting



Sensor nodes

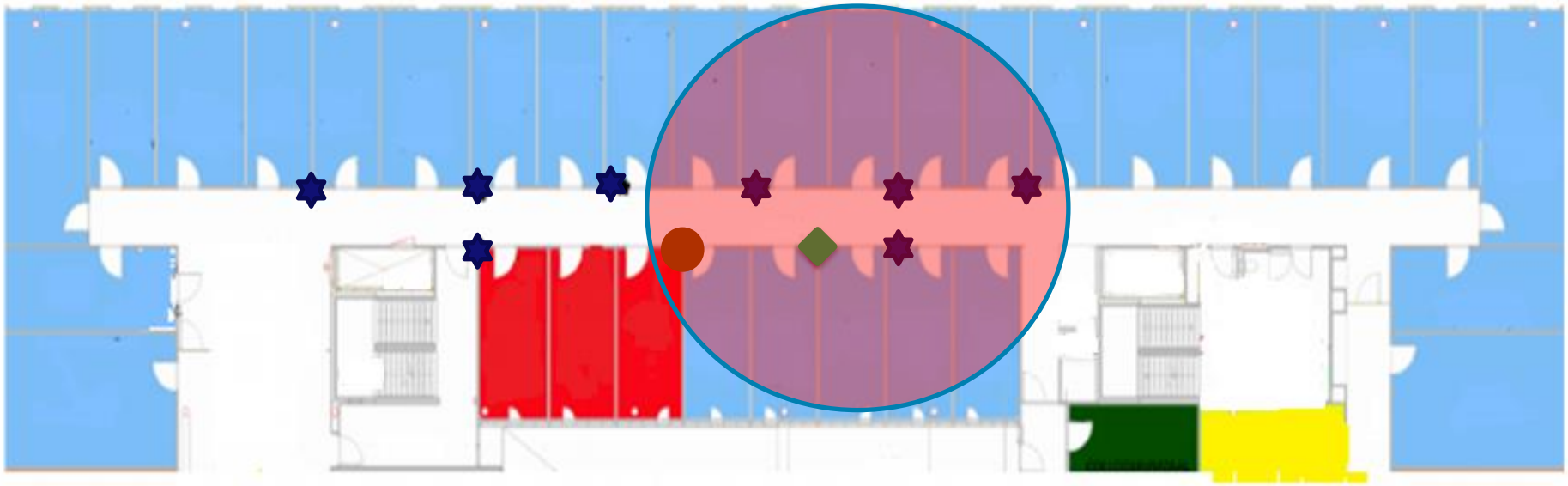
Sink

Jammer node

SAN

TU/e

Experiment setting

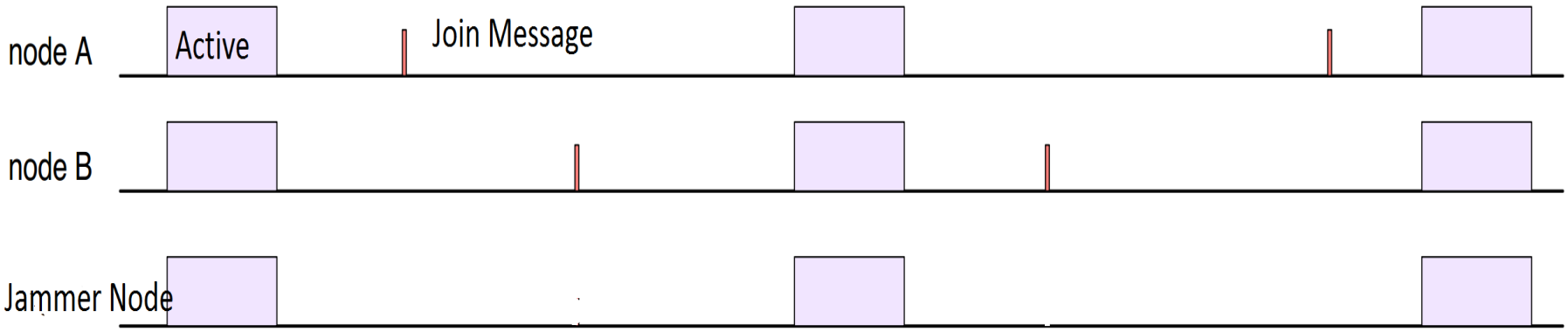


Reception
Range of
Sink

SAN

TU/e

Jamming



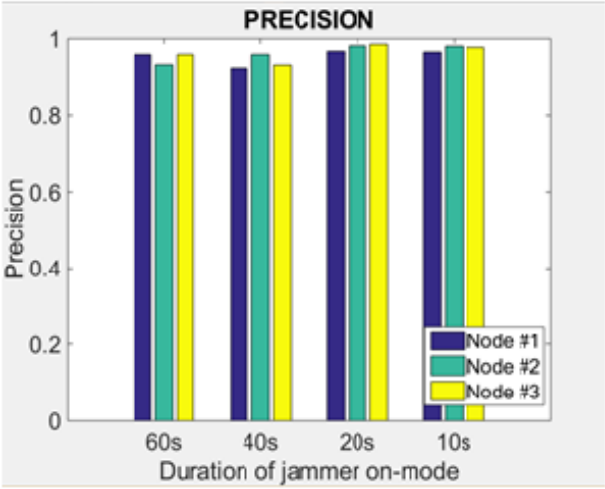
- Jammer Node
  - ✓ Jammer node is ‘on’ for a certain time, after which it is ‘off’ for the remainder of a period
  - ✓ When ‘on’, the jammer node continuously transmits a dummy message without channel checking, in order to cause packet loss.

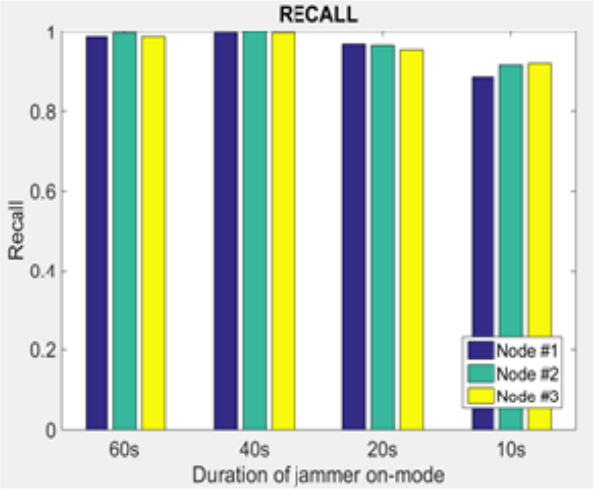
Parameters

- MyriaNed:

Observable parameters	Controllable parameters
RSSI on receiving node	TX power of transmitting node
Number of direct neighbors	TX power
	RX sensitivity
Number of received messages	number of TX schedules
	number of RX schedules (blocks)
Number of received new messages	round time

Case Study: results





- Used model decreases false alarm rate. (Precision results)
- Used model fails to detect some failures, while jammer on-mode decreases from 60s to 10s. (Recall results) → Shorter outlier subsequences

Case Study 2: battery lifetime

ProHeal

Observable parameter	Controllable parameters
Current energy consumption	sleep/wake intervals (defined by application)
	active local resources
	round time
Received messages frame timing	guard times used in data transmission

MyriaNed platform protocol components	Component state	Current consumption
Processor clock speed	14Mhz	low
	28Mhz	high
Tics per slot	28	high
	20	average
	14	low
Transmit power	0 dBm	11.3 mA
	-6 dBm	9 mA
	-12 dBm	7.5 mA
	-18 dBm	7.0 mA
Round time	8, 4, 2, 1, 0.5, 0.25, 0.125 sec	20μA, 23μA, 30μA, 54μA, 92μA, 158μA, 310μA

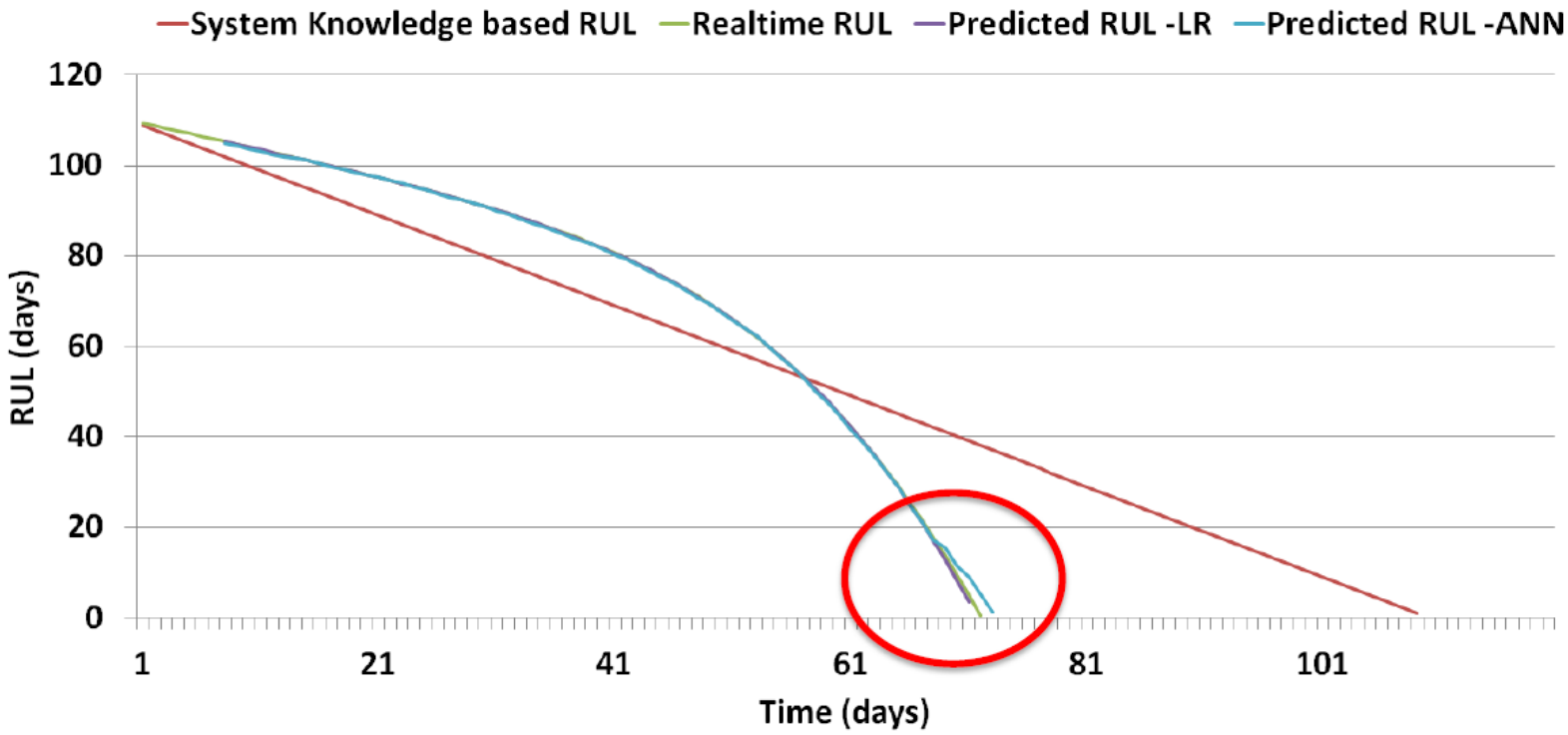
SAN

TU/e

Fault prediction (low battery)

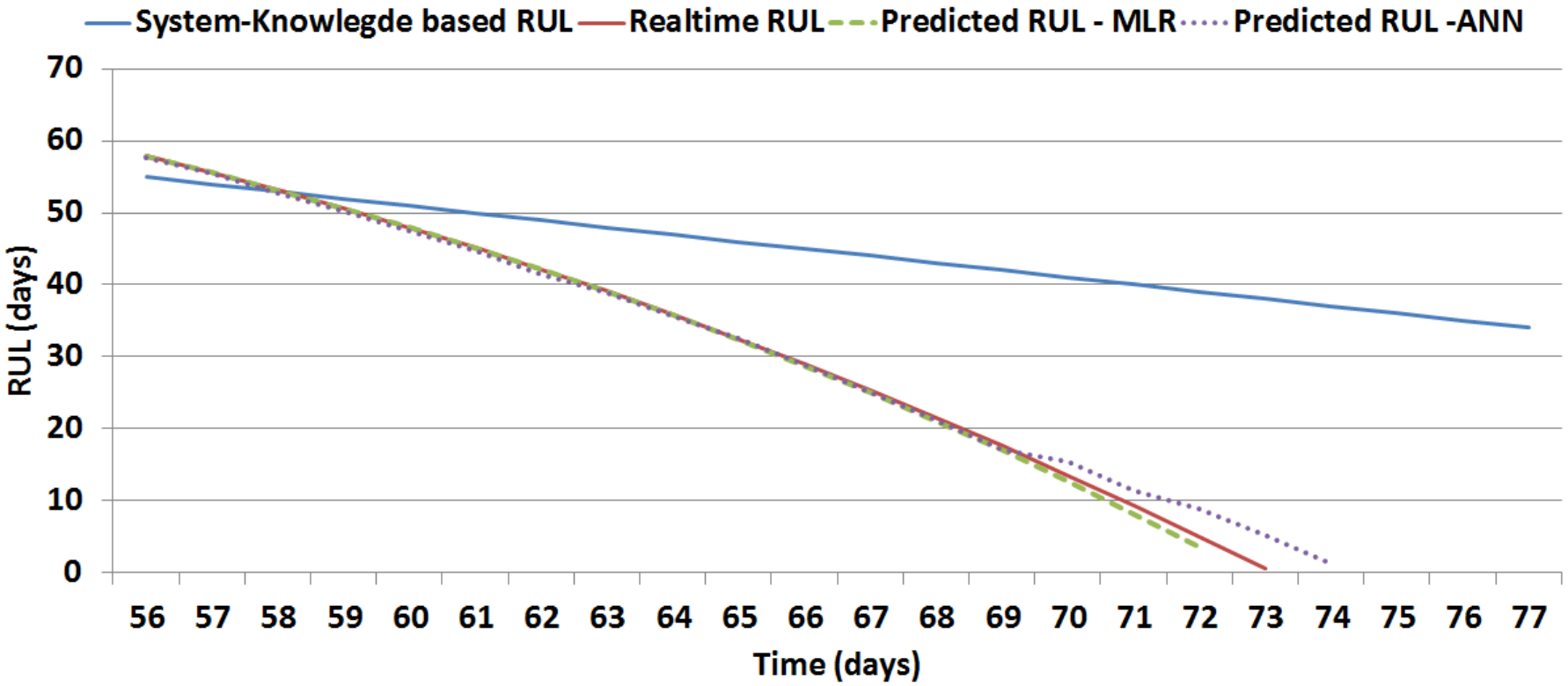
- Based on time series of real discharge we compared three algorithms for predicting the behavior of Remaining Useful Lifetime (RUL)
  - simple extrapolation of current system observation (system knowledge)
  - artificial neural network (ANN)
  - multiple linear regression (MLR)
- We compared errors in the piece-wise predictions

Fault prediction
(low battery)



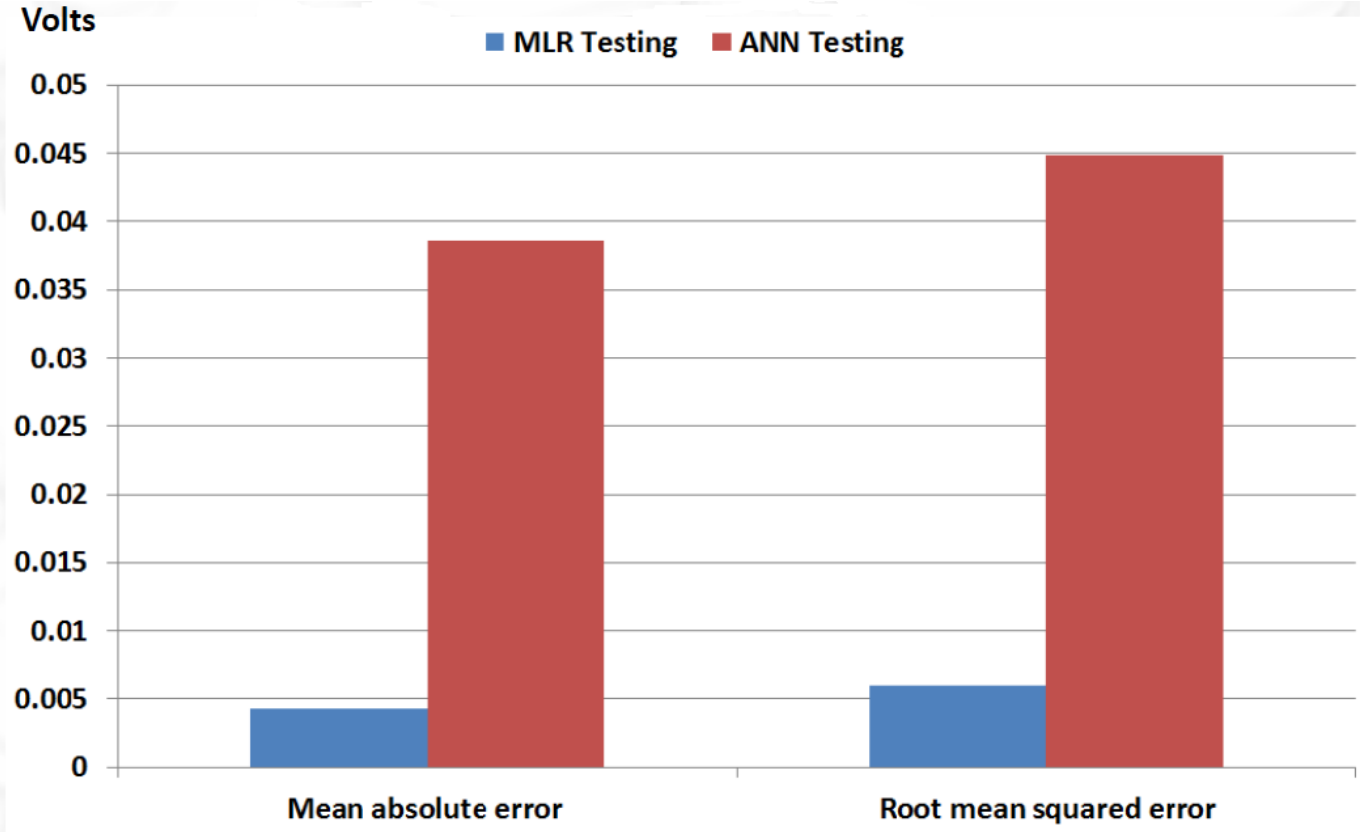
Predicted Remaining Useful Life (RUL) of the MyriaModem – MLR & ANN

Fault prediction - RUL graph (low battery)

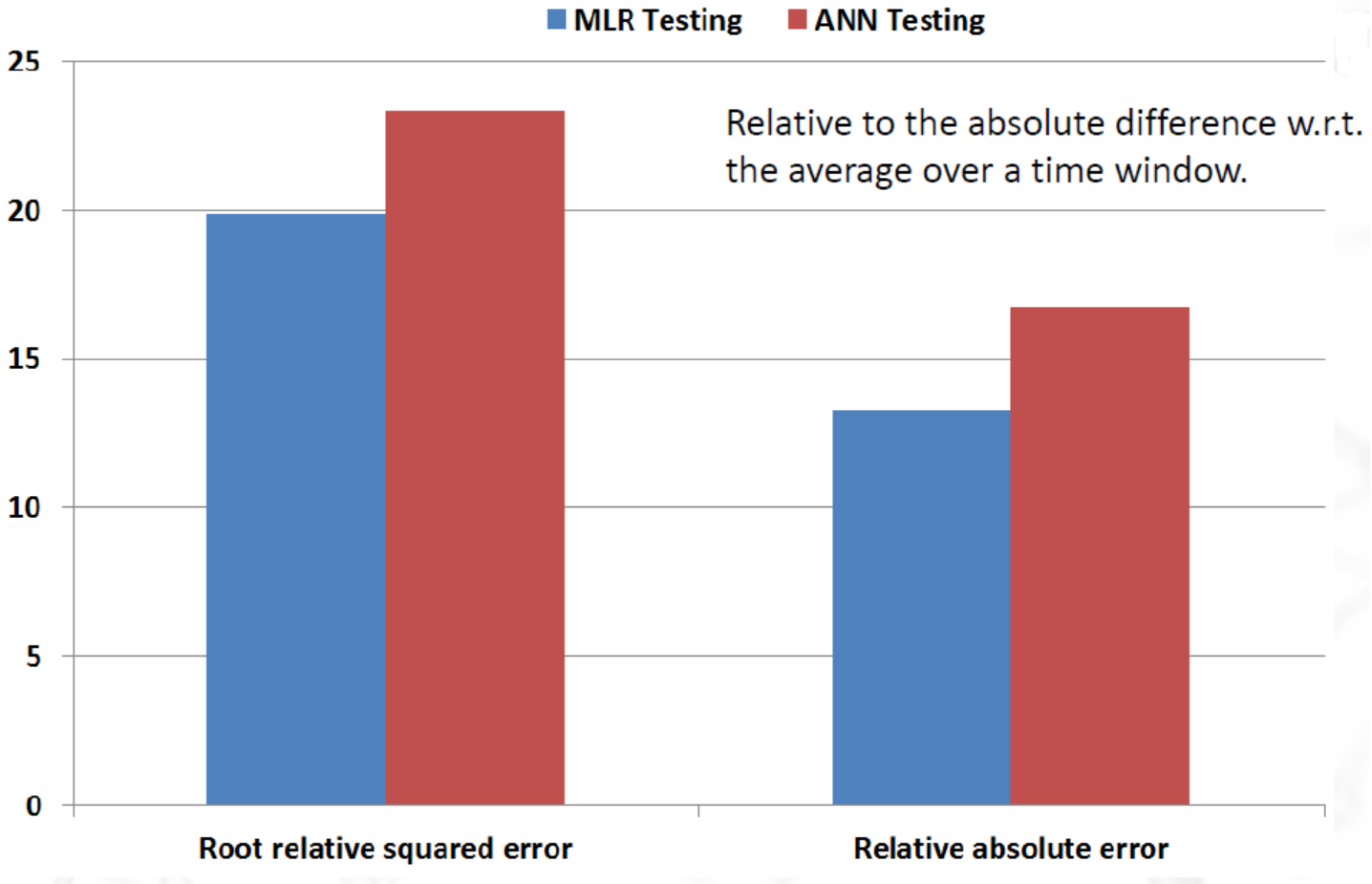


Predicted Remaining Useful Life (RUL) of the MyriaNed – MLR & ANN

Fault prediction
(low battery)

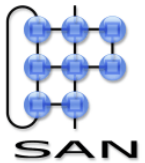


Fault prediction
(low battery)



Future Work

- Learn the normal and failure states' boundaries of the system resources and functions.
- Develop fault models
- Develop prediction models for system resources and functions.
- Develop adaptation policies to dynamically prevent the predicted faults using system knowledge
  - e.g., CPs state boundaries, redundancy, etc.
- Performance of each adaptation policy:
  - application availability and reliability
  - efficient resource management



TU/e

ProHeal

Questions?

SAN

TU/e