AutoMon

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

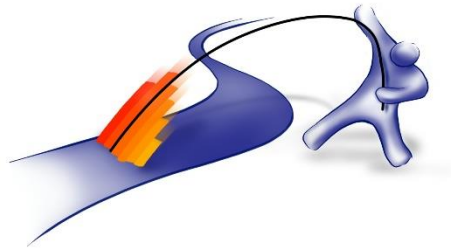
Closed-Loop Control in Network Monitoring

13. October 2017

KuVS „Network Softwarization“

isarNet
Software Solutions GmbH

jochen.koegel@isarnet.de



Agenda

- AutoMon project overview
- AutoMon Vision
- Concept of closed-loop control
- Use cases
- Conclusion and outlook

AutoMon Project – Facts

Project goal: Automated performance monitoring

Funded by the German government

- Innovation program for Small and Medium Enterprises (SME) „KMU-innovativ“
- Volume: 2.69 M€

Time frame: June 2016 … May 2019

https://automon-projekt.de/en

AutoMon Project – Partners

Application partners

DB Systel: Service provider

For German railway, global logistics

Problem statements, use cases, scenarios, labs

Suitable solutions, concepts and ideas for future plans

MultiNetwork WAN services

For airlines, global enterprises,…

Research partners

Technical University Munich

Chair of Network Architectures and Services, Prof. Carle

SME in Munich

- IsarFlow network monitoring
- Network consulting

SME in Dresden

- exply.io (data exploration)
- Contributor to Neos CMS

AutoMon – Problem Statement

Challenges in network operation

- network performance becomes more and more business critical
- fewer and fewer people operate increasingly large networks
- high dynamic in networks due to softwarization and automation

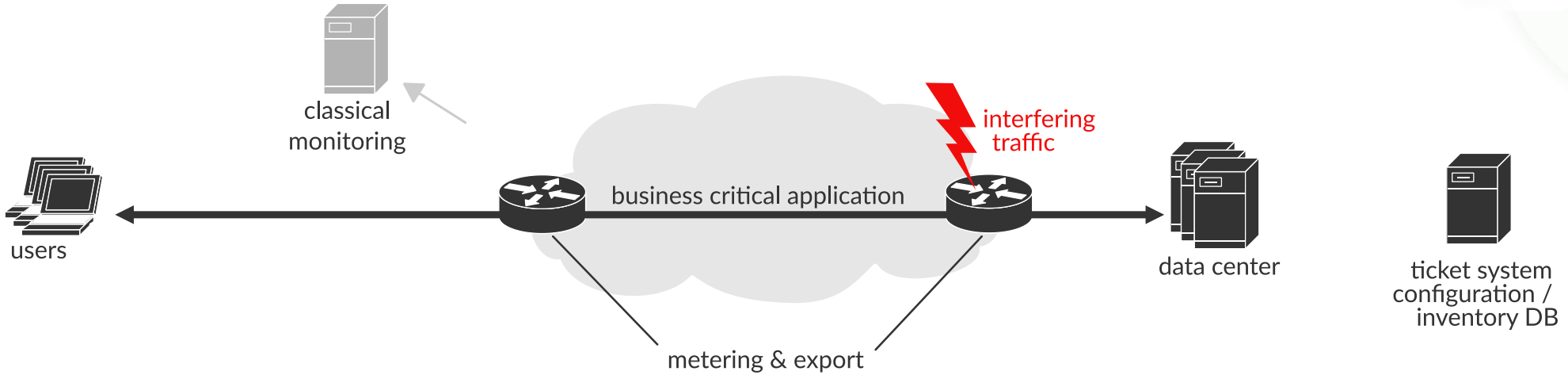
Operation = Monitoring + Control

→ Automation of network monitoring required

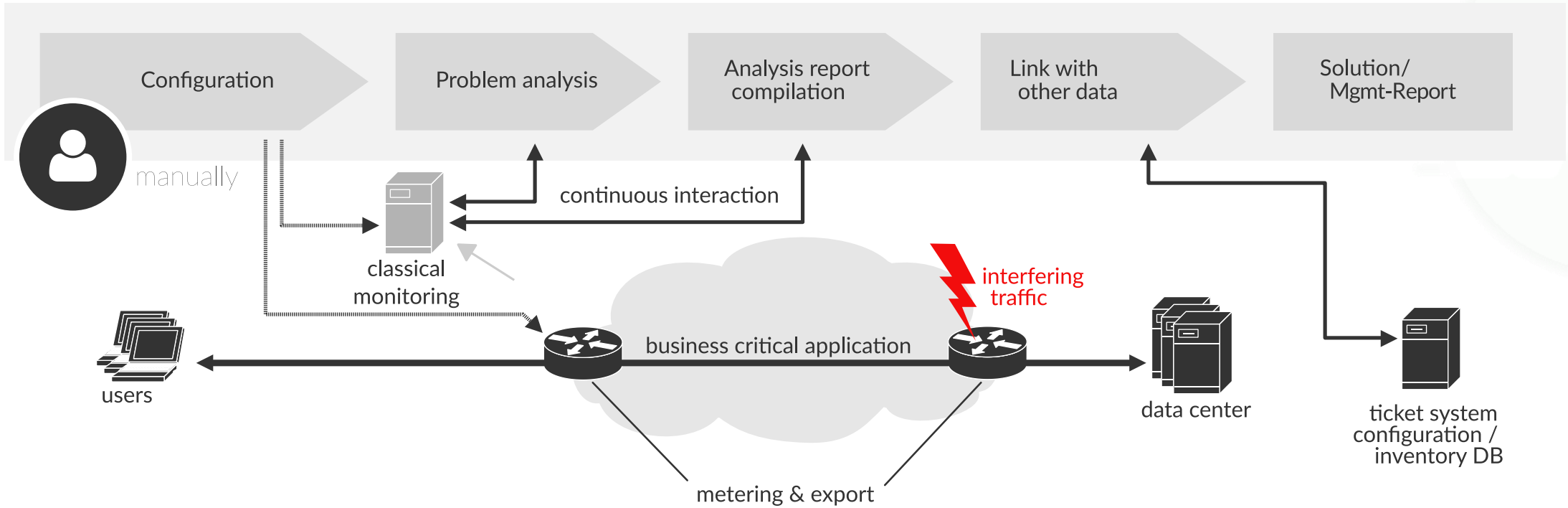
➔Utilize APIs of network equipment for softwarized monitoring approach

AutoMon Vision

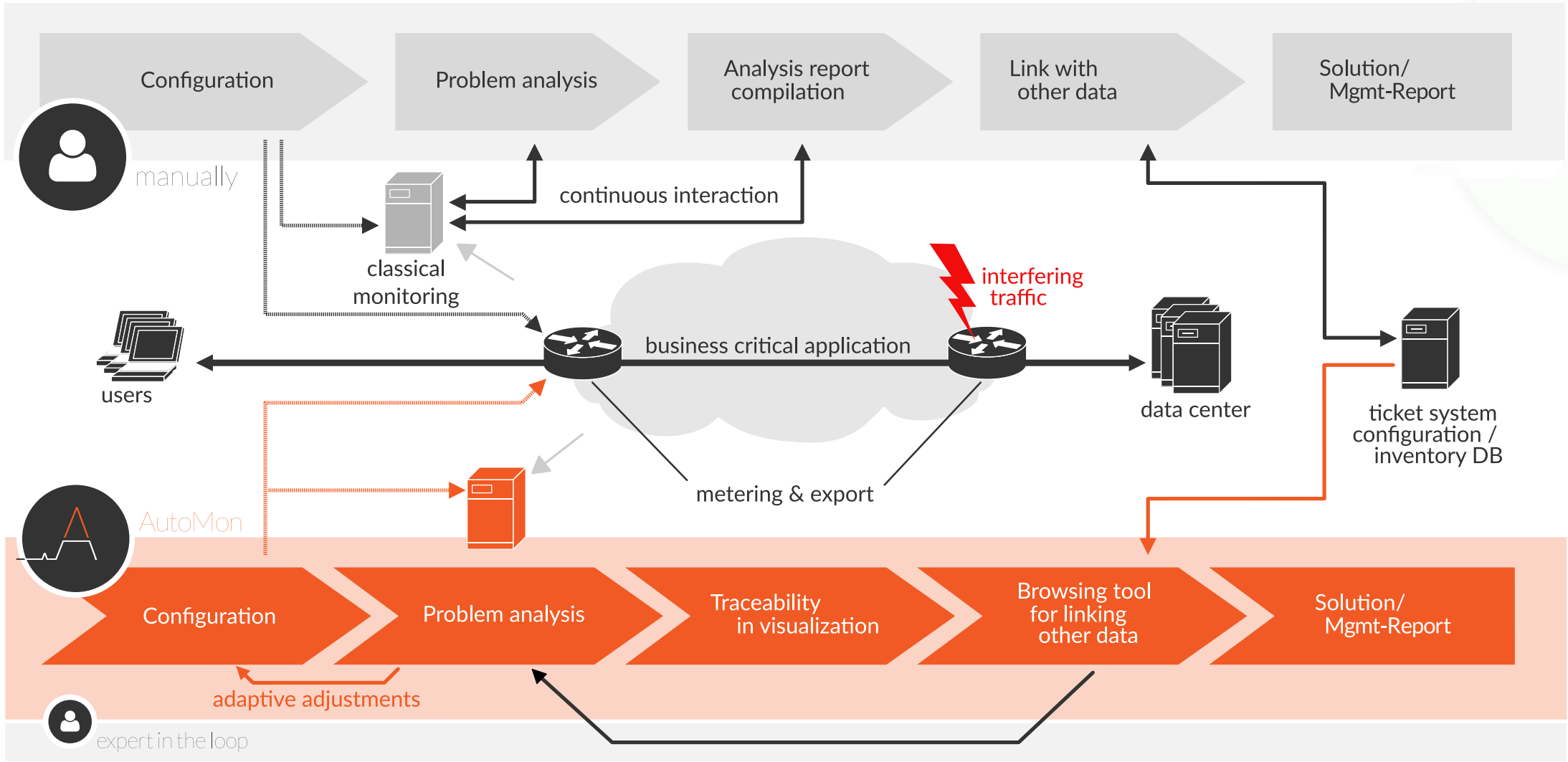
AutoMon Vision



AutoMon Vision

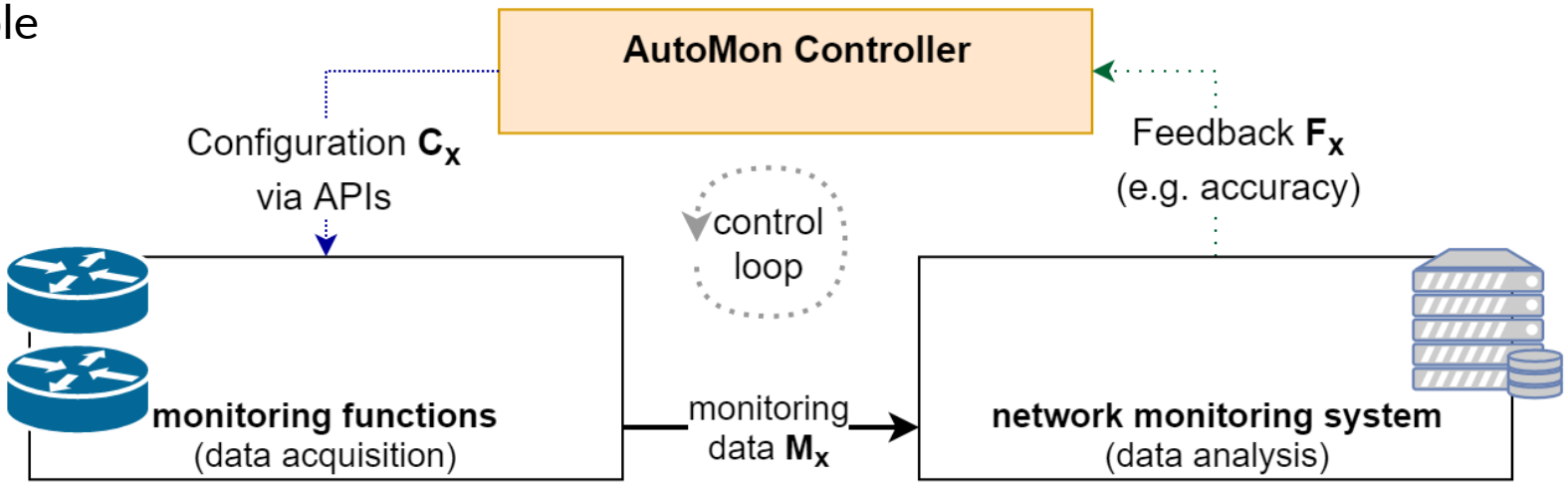


AutoMon Vision

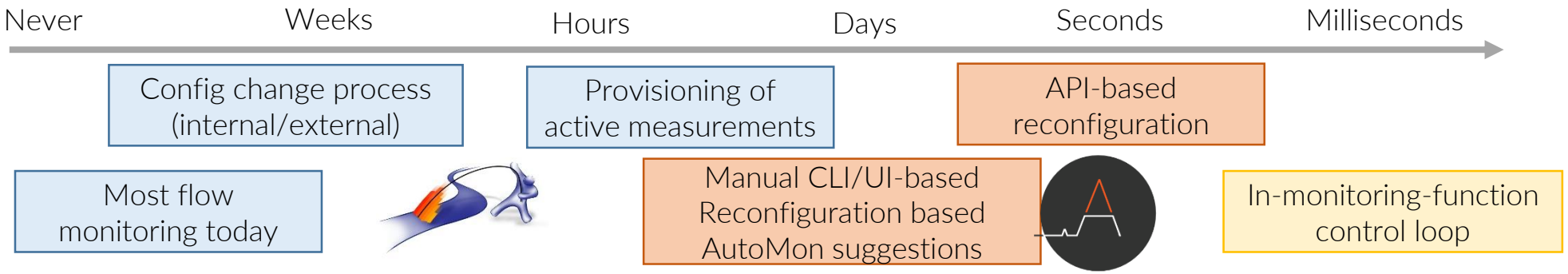


Concept of closed-loop control

General principle

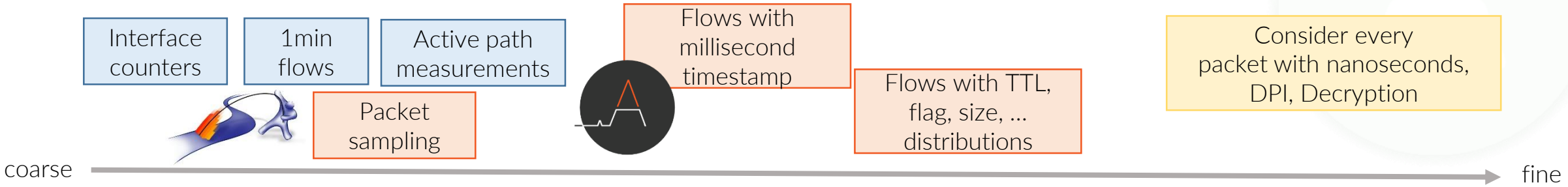


Time scale of control actions

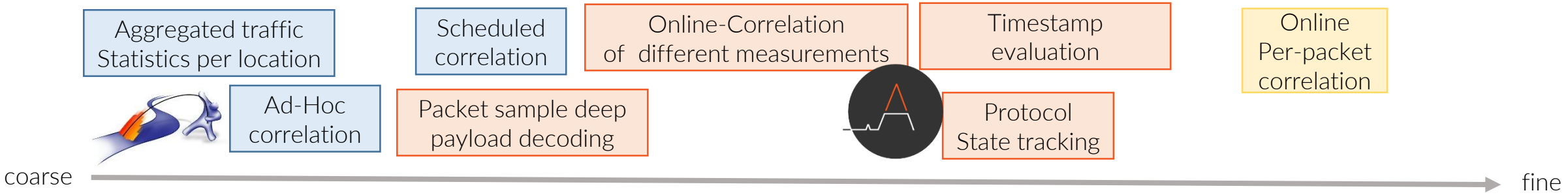


Concept of closed loop control

Monitoring degree of detail



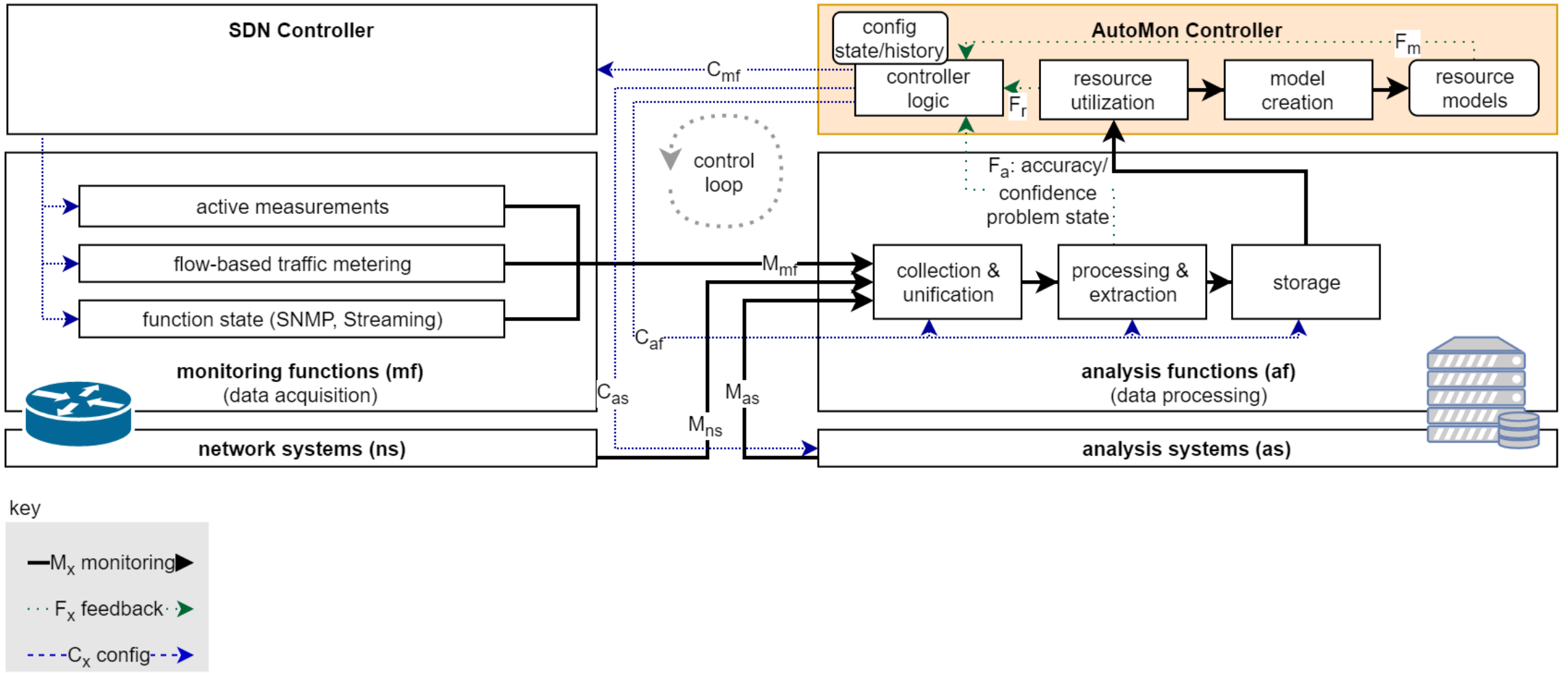
Data analysis degree of detail



Just do everything at the most fine grained level?

→Don‘t forget the available system resources (CPU, Mem, Disk, Network) in monitoring and analysis functions

Concept of closed loop control



Use case examples

Traffic peak analysis

- Normal operation: Automatically triggered detailed traffic analysis
- More analysis: Which hoststs, sessions, services
- More monitoring: Additional metrics like packet sizes, TCP flags, TTL, …

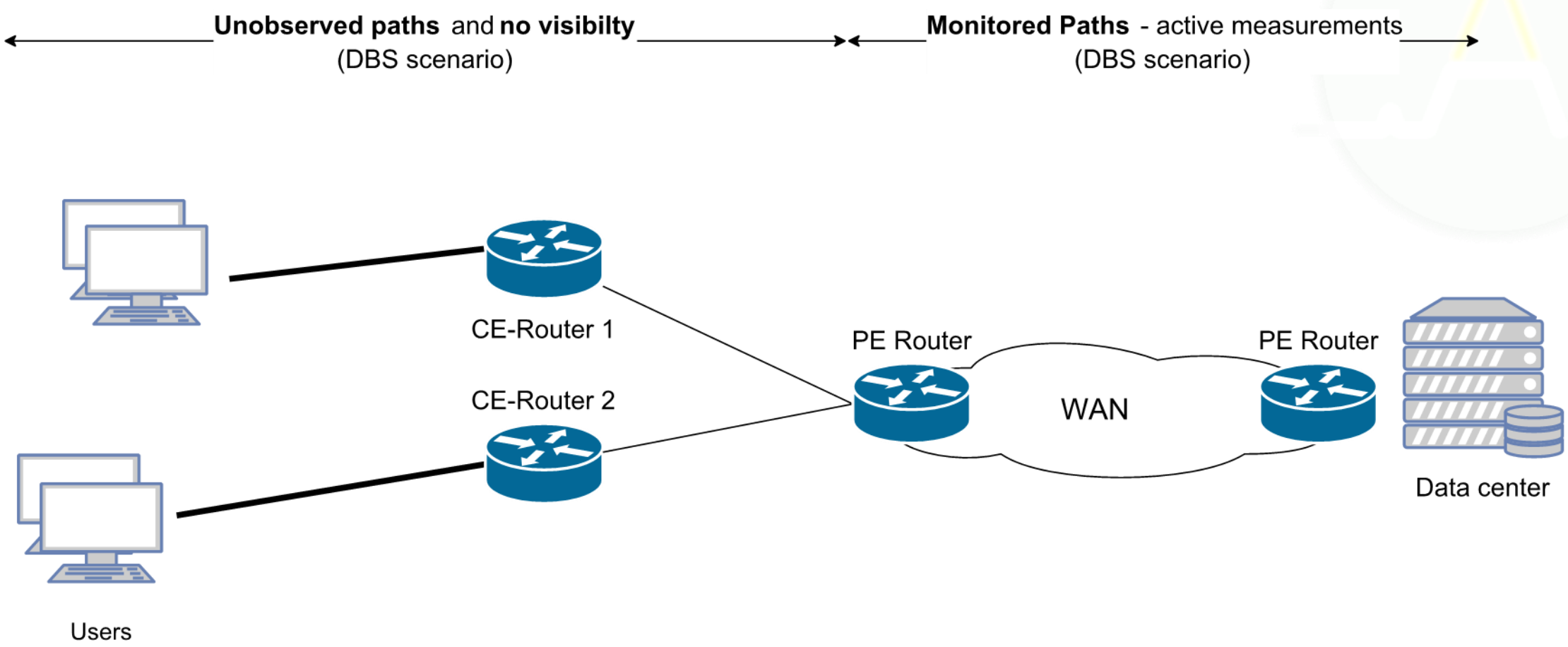
MTU/firewall issues

- Normal operation: Track max/mean packet sizes for all connections roughly
- More analysis: Track down suspicious packet size behavior to location, interface, link..
- More monitoring: Additional metrics like min/max packet size, payload capturing for TCP MTU options, active measurements

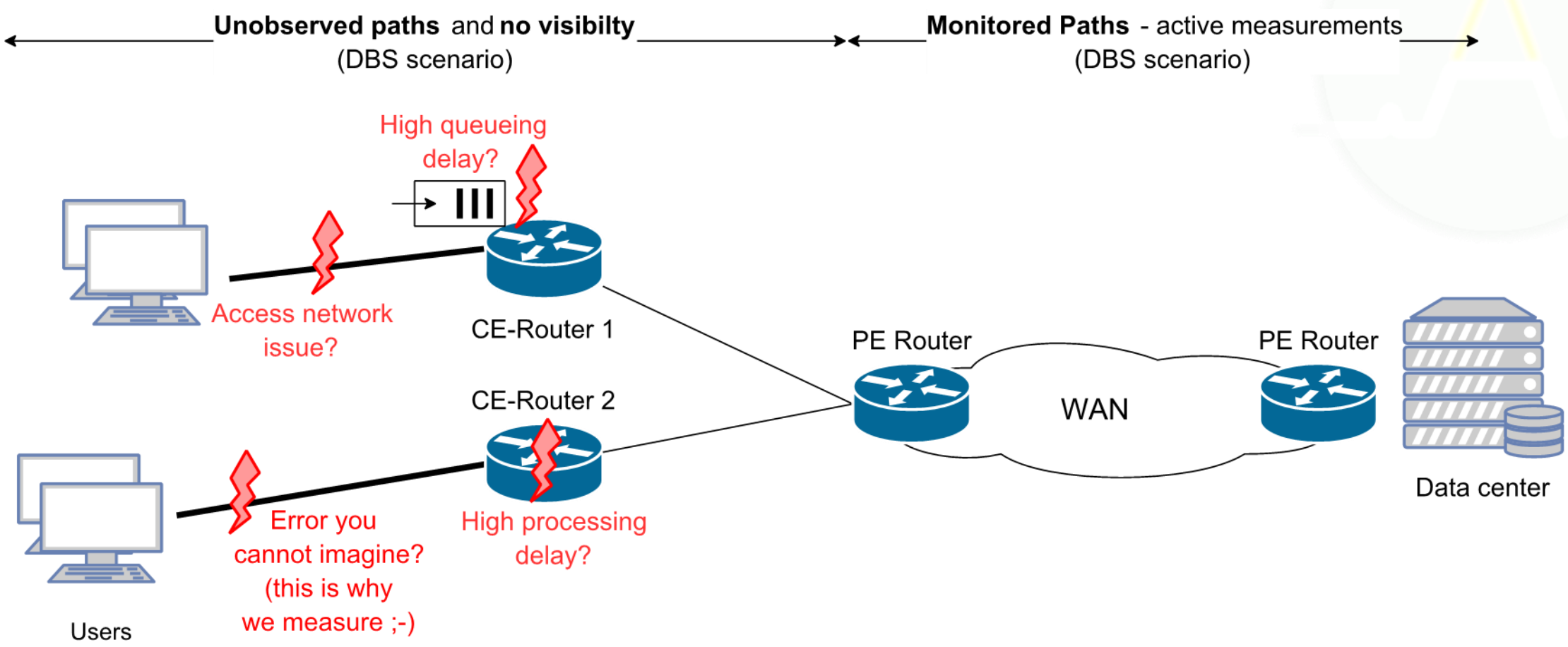
Delay Variation on unobserved paths

- Normal Operation: from TCP Timestamps
- → following slides

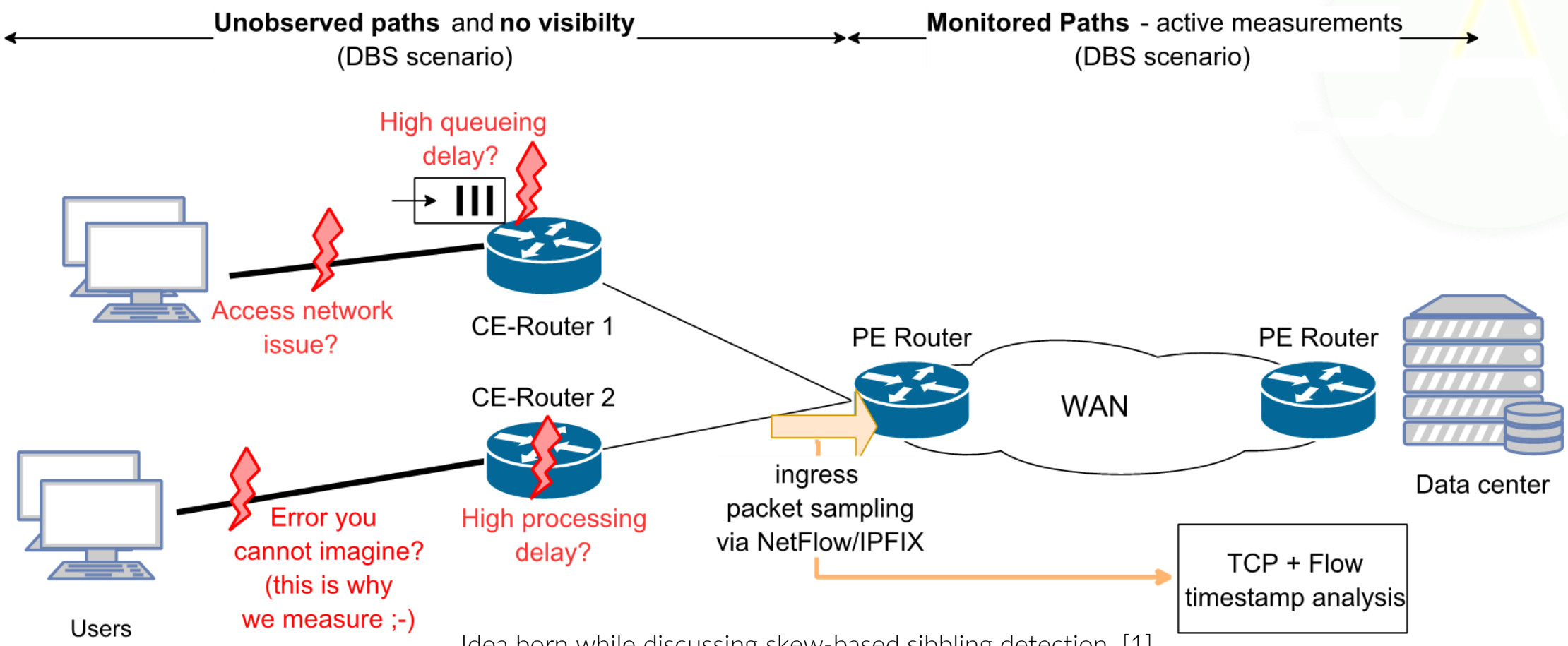
Example use case – unobserved paths



Example use case – unobserved paths



Example use case – unobserved paths



Idea born while discussing skew-based sibbling detection [1]

More details – see [2]

Example use case – unobserved paths

Normal operation: detect large delay variations via passive TCP timestamp analysis

General control tasks

- Adapt sampling rate to get reasonable accuracy
- Adapt sample size to find enough TCP timestamps even for long IP/TCP headers with several options
- Determine whether delay variations can be obtained per subnet, host-IP, flow, traffic class

1st Control Action if issue found

- Enable additional measurment points closer to the problem source
- Passive path-based measurements

2nd Control Action if issue found

- Trigger active measurements to the hosts that experience the issue and hosts of same subnet
- IP-addresses learned from traffic

Conclusion and Outlook

Conclusion

- Softwarization demands for, but also enables more dynamic in network monitoring
- Closed loop control: finding the optimal monitoring system configuration
  → maximum insight for given resources
- Rough metrics for initial insights are promising (TCP timestamp analysis) in lab tests and experimental deployments with production traffic

Outlook

- Utilize the flexibility of AutoMon's data collection by adding more dynamic analysis methods initiated via API from the AutoMon controller
- Closing the control loop for allocating more resources on demand? Which criteria regarding cost and benefit to use?

References

[1] Q. Scheitle, O. Gasser, M. Rouhi and G. Carle:
Large-Scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew, 2017.

[2] S. Meier, J. Kögel:
Indirect passive measurement of network characteristics in the AutoMon project.
NMRG Workshop on Measurement-BasedNetwork Management at IETF 99, Prague, 2017.

Acknowledgement

This work was partly funded as part of the AutoMon project by the German Federal Ministry of Education and Research (BMBF) under contract No. 16KIS0408K. Responsibility for the information and views expressed in this publication lies entirely with the authors.

Backup slides

Example use case – unobserved paths

Focus: Larger scale delay variations

- not only packet-to-packet jitter (impacts Voice)
- but: generally worsening network conditions
  - impact interactive business applications
  - absolute delay values not required in the first place
- possible actions
  - bad condition: Trigger further automated investigation
  - good condition: Application performance issue ?
    →"Everything is fine in WAN – check DC"

Exemplary RTTs [3]

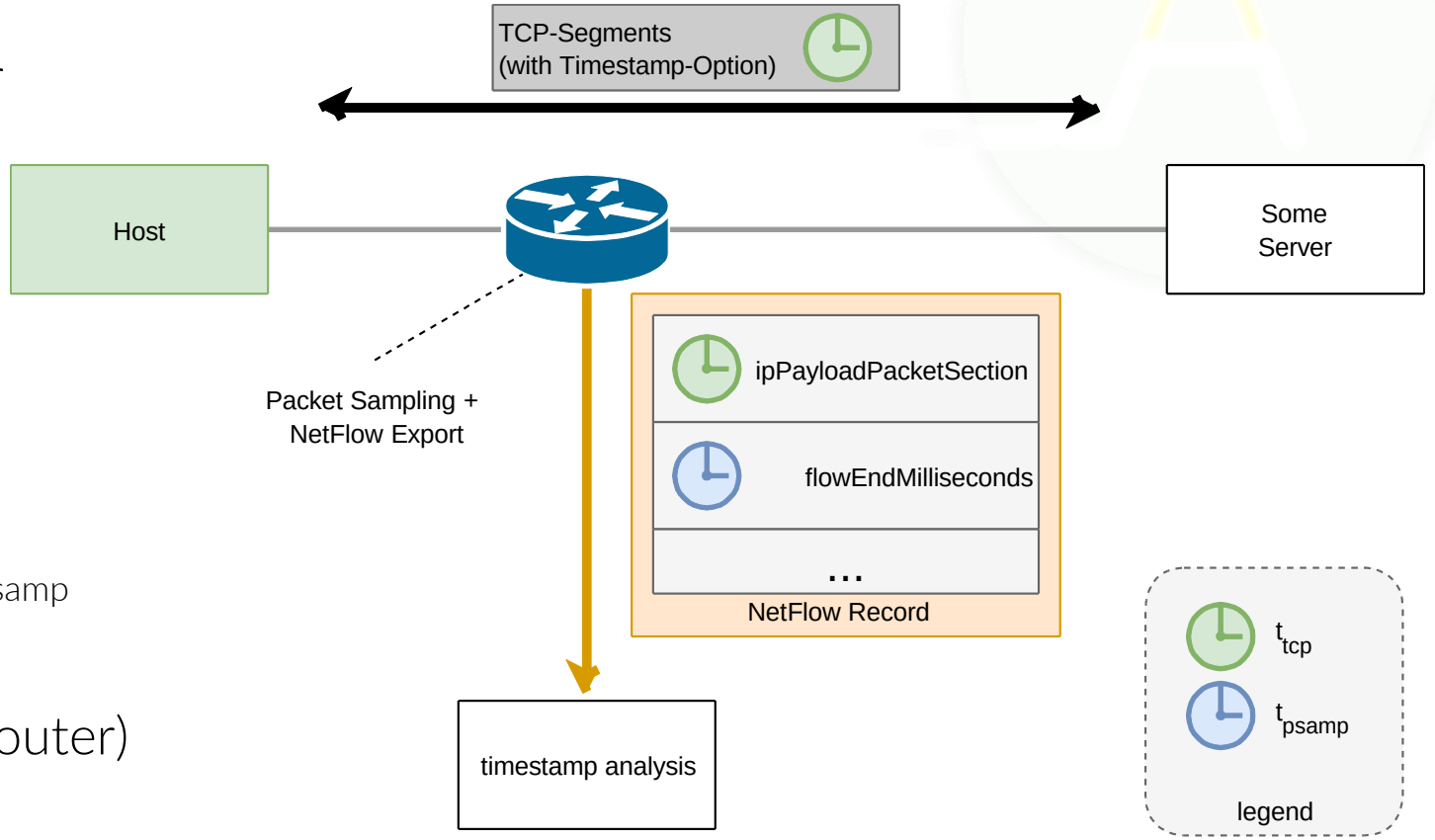
Timestamp sampling

Approach

- enable IP payload sampling on router
- export packet samples via NetFlow
- export two timestamps per packet sample
  - TCP timestamp (t_{tcp})
  - sampling timestamp (t_{psamp})
- establish relation between t_{tcp} and t_{psamp}

Challenges

- clock / timestamp accuracy (host & router)
- TCP timestamp availability
- suitable (per flow) sample size



Timestamp relation

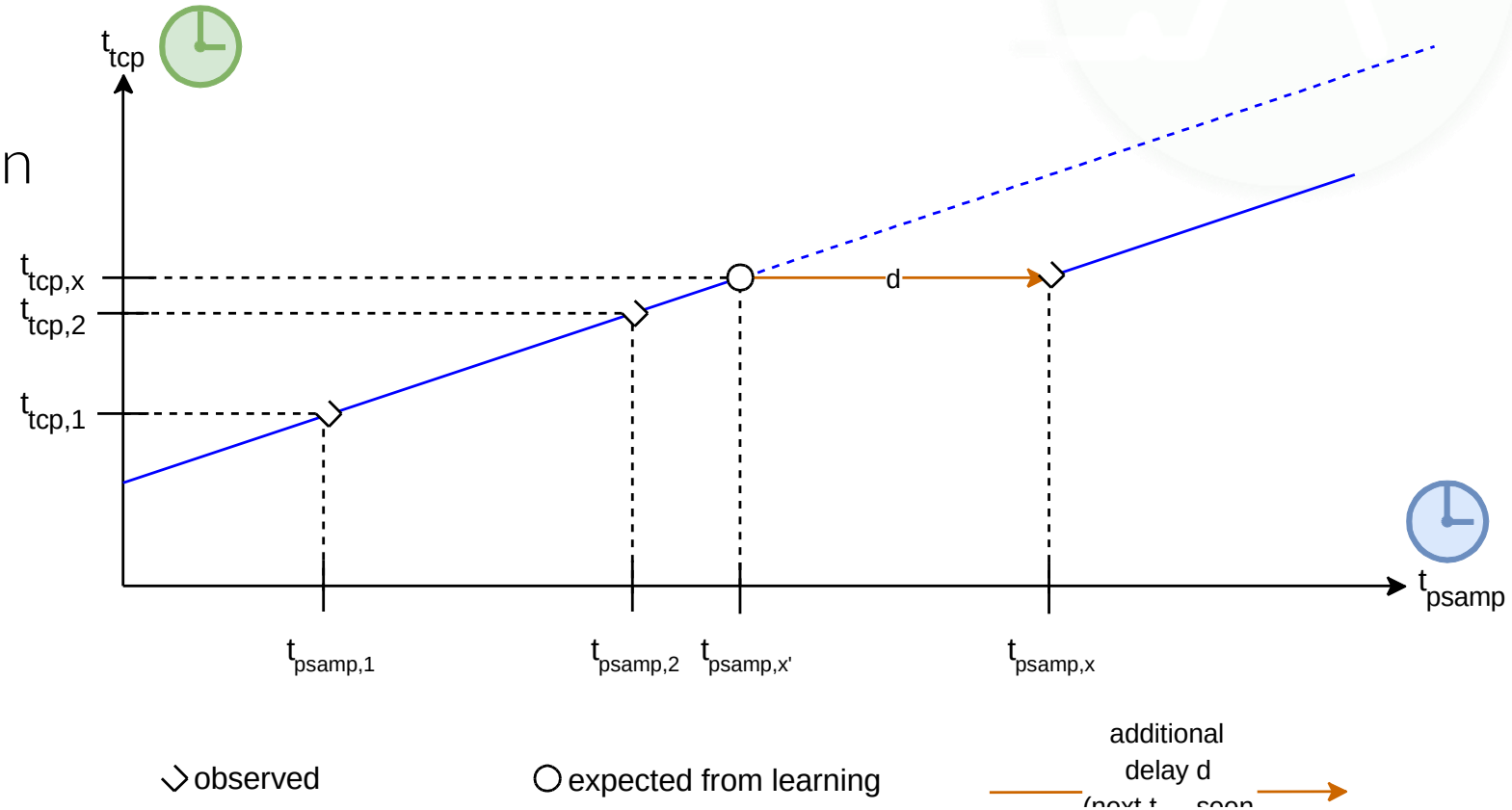
Assumptions

- clock drift negligible
- clocks do not jump

→ linear relation between t_{tcp} and t_{psamp}

Linear Algebra

- $y = m * x + b$
- $t_{tcp} = m * t_{psamp} + b$



Estimation of slope *m*

Slope

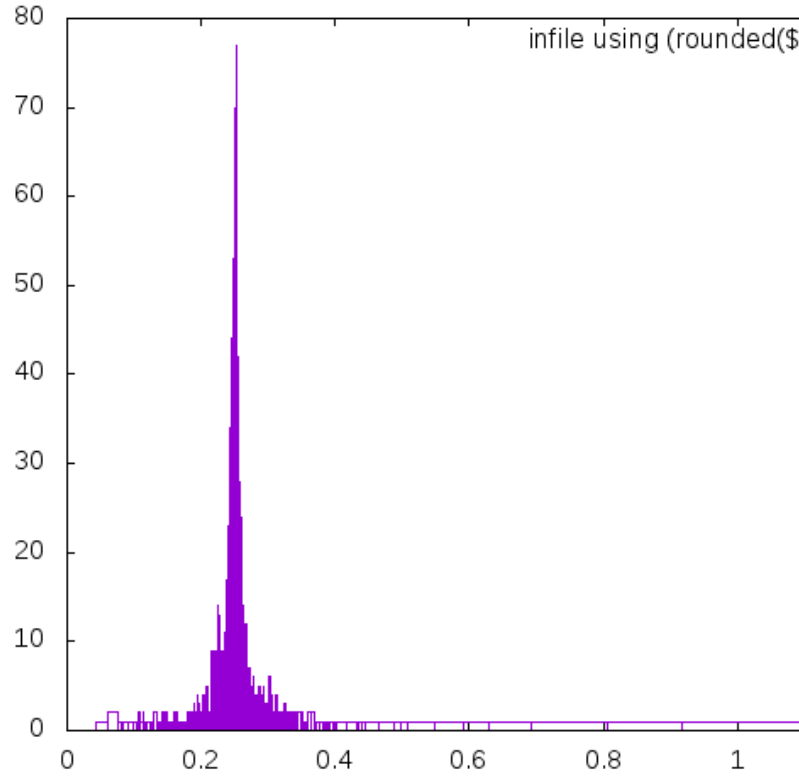
how fast advances time in router compared to time in host

Approach

- consider consecutive samples of same TCP flow
- for each pair: estimate slope m:

$$m = \frac{\Delta t_{tcp}}{\Delta t_{psamp}}$$

- „guess“ most likely slope after *n* slope estimations



Result

approach seems feasible (at least for lab setup)

Estimation of slope *m*

Slope

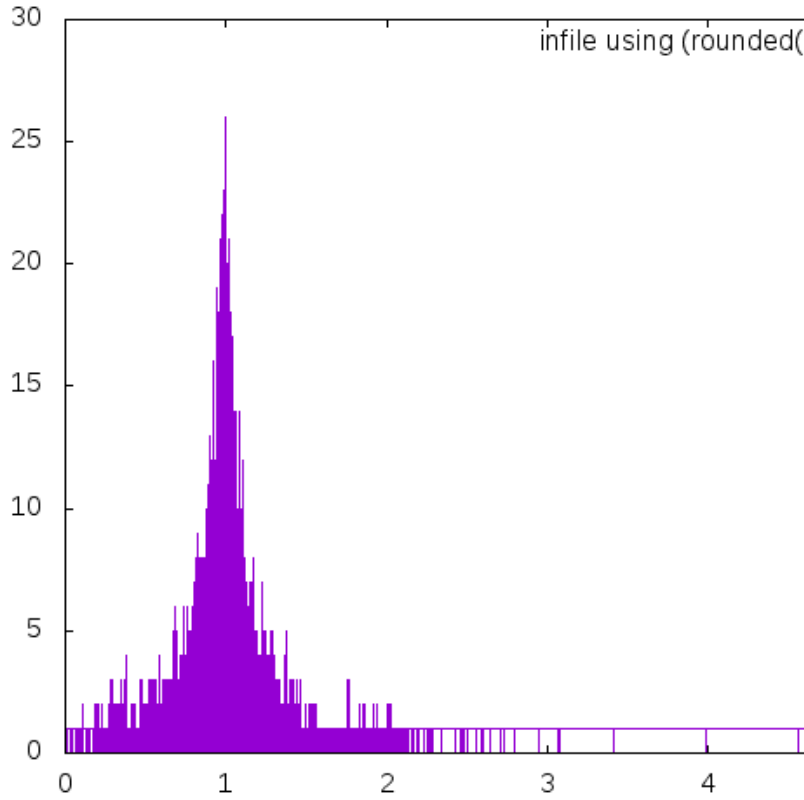
how fast advances time in router compared to time in host

Approach

- consider consecutive samples of same TCP flow
- for each pair: estimate slope m:

$$m = \frac{\Delta t_{tcp}}{\Delta t_{psamp}}$$

- „guess“ most likely slope after *n* slope estimations



Result

approach seems feasible (at least for lab setup)

Estimation of offset *b*

Offset

(constant?) difference between t_{tcp} and t_{pcap} timestamp values

Approach

1. calculate initial offset *b* with first **obs**erved packet sample

$$b = t_{tcp,obs,1} - m * t_{psamp,obs,1}$$

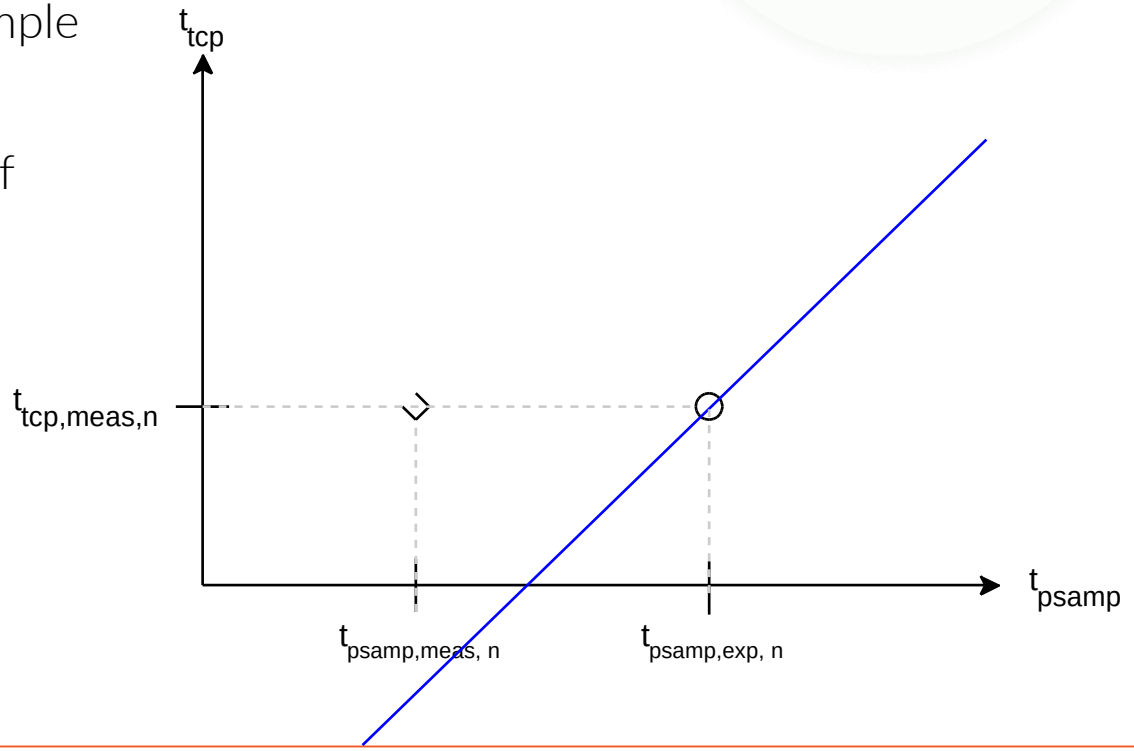
2. use initial offset for calculating **exp**ected timestamp of next sample

$$t_{psamp,exp,2} = \frac{t_{tcp,obs,2} - b}{m}$$

3. update *b* if $t_{psamp,exp,2} > t_{psamp,obs,2}$
4. repeat calculations for some/all subsequent samples to determine minimum/maximum offset

Open Issue

examine convergence behavior of offset



Estimation of offset *b*

Offset

(constant?) difference between t_{tcp} and t_{pcap} timestamp values

Approach

1. calculate initial offset *b* with first observed packet sample

$$b = t_{tcp,obs,1} - m * t_{psamp,obs,1}$$

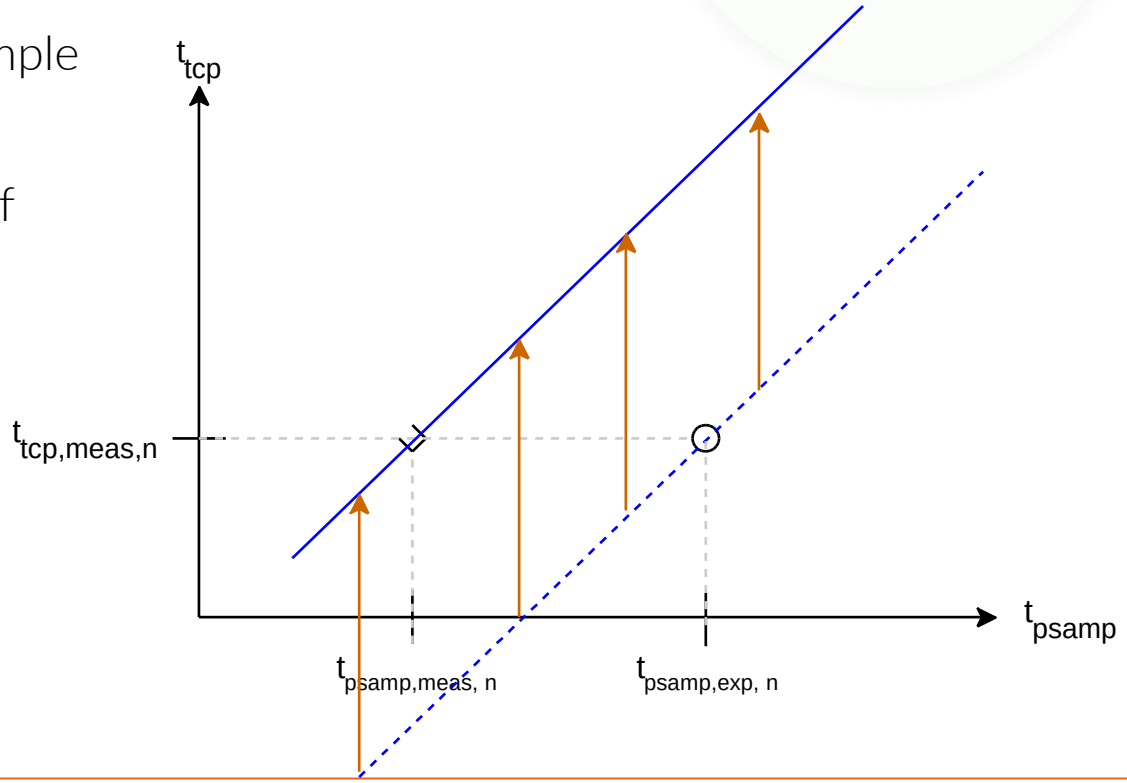
2. use initial offset for calculating expected timestamp of next sample

$$t_{psamp,exp,2} = \frac{t_{tcp,obs,2} - b}{m}$$

3. update *b* if $t_{psamp,exp,2} > t_{psamp,obs,2}$
4. repeat calculations for some/all subsequent samples to determine minimum/maximum offset

Open Issue

examine convergence behavior of offset



Estimation of offset *b*

Offset

(constant?) difference between t_{tcp} and t_{pcap} timestamp values

Approach

1. calculate initial offset *b* with first observed packet sample

$$b = t_{tcp,obs,1} - m * t_{psamp,obs,1}$$

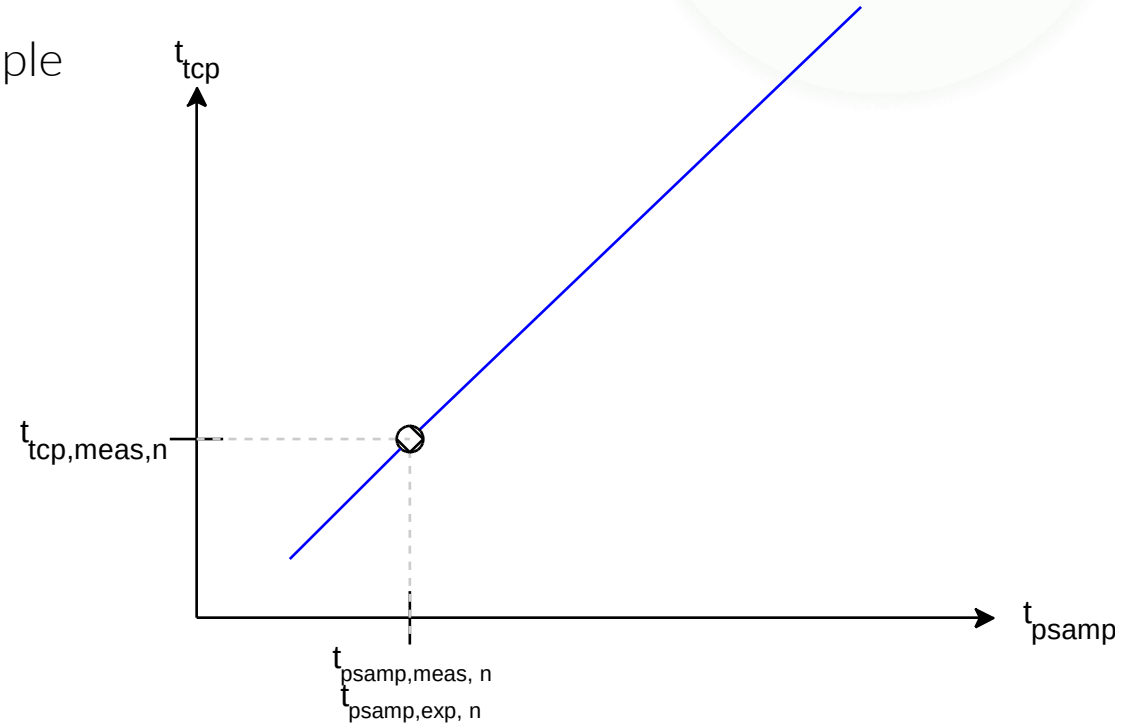
2. use initial offset for calculating expected timestamp of next sample

$$t_{psamp,exp,2} = \frac{t_{tcp,obs,2} - b}{m}$$

3. update *b* if $t_{psamp,exp,2} > t_{psamp,obs,2}$
4. repeat calculations for some/all subsequent samples to determine minimum/maximum offset

Open Issue

examine convergence behavior of offset



Preliminary Results

Measurement & processing setup

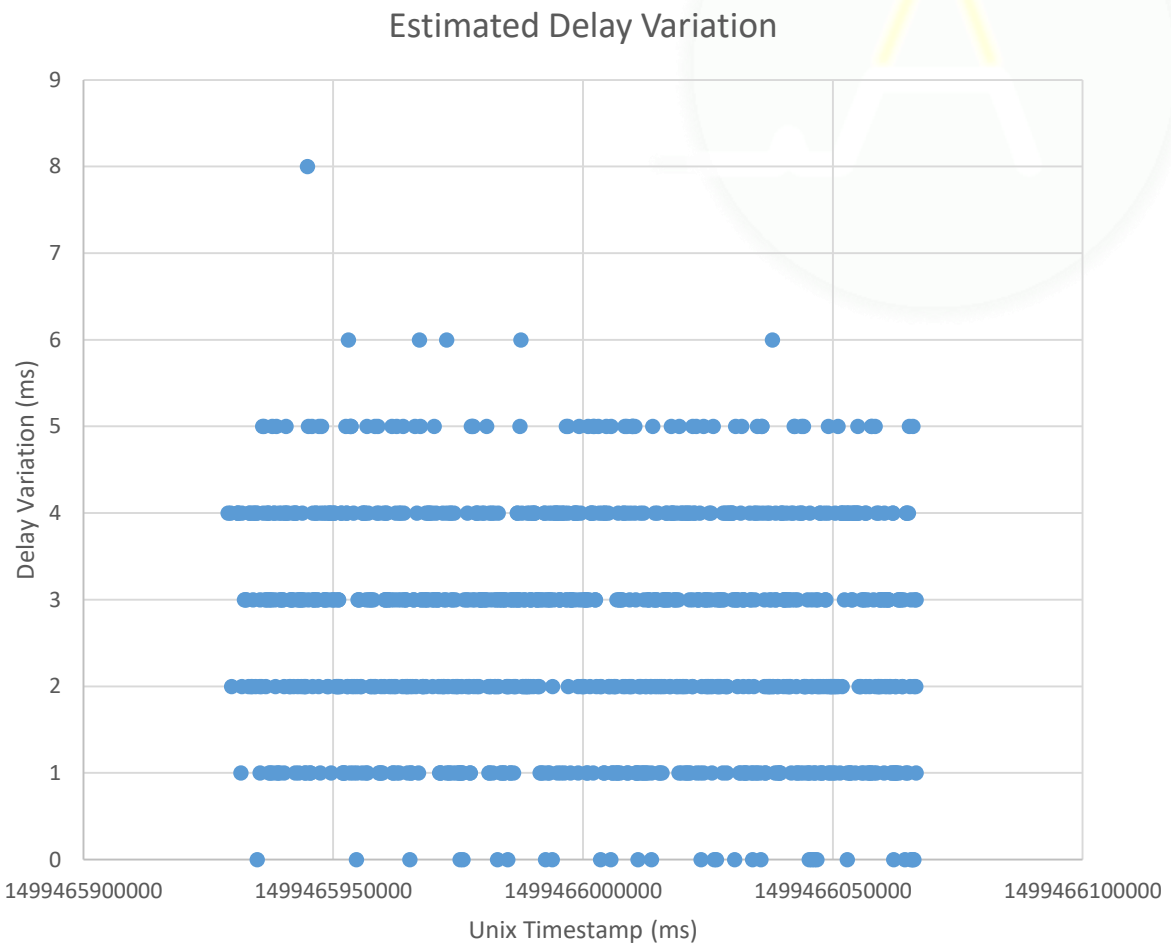
- packet sampling in IsarNet intranet
  - LAN + WAN traffic
  - no well-known test traffic
  - no well-known delay/jitter

  →no lab conditions
- offline processing

LAN-Traffic

- delay variation typically ~ 1-5ms
- at first glance no outliers

→measurement accuracy probably ~5ms



Preliminary Results

Measurement & processing setup

- packet sampling in IsarNet intranet
  - LAN + WAN traffic
  - no well-known test traffic
  - no well-known delay/jitter

  →no lab conditions
- offline processing

Other first observations

- **some** long lived flows (here: ~12h) show saw tooth pattern

  →probably clock drift in host
- might have to consider clock drift, and other clock effects in future work