\Orchestrating a brighter world NEC

EU IoT Week, Belgrade, 31/6 - 2/7 2016

Data Analytics at the Network Edge

Apostolos Papageorgiou
NEC Laboratories Europe
Heidelberg, Germany

Apostolos.Papageorgiou@neclab.eu

Outline

Background about Network-edge computing

- Technical Landscape and motivation
- Current limitations

Real-time per-item data reduction

- Differentiators and overview of our solution
- Way of operation of „exchangeable data handlers“
- „Streamification“ of data reduction algorithms
- Summary of evaluation

Edge deployment of IoT data streaming tasks

- Stream Processing Frameworks and their limitations
- Our solution for edge-aware streaming task deployment

Background

Technical landscape, motivation, and current limitations for Network-edge computing

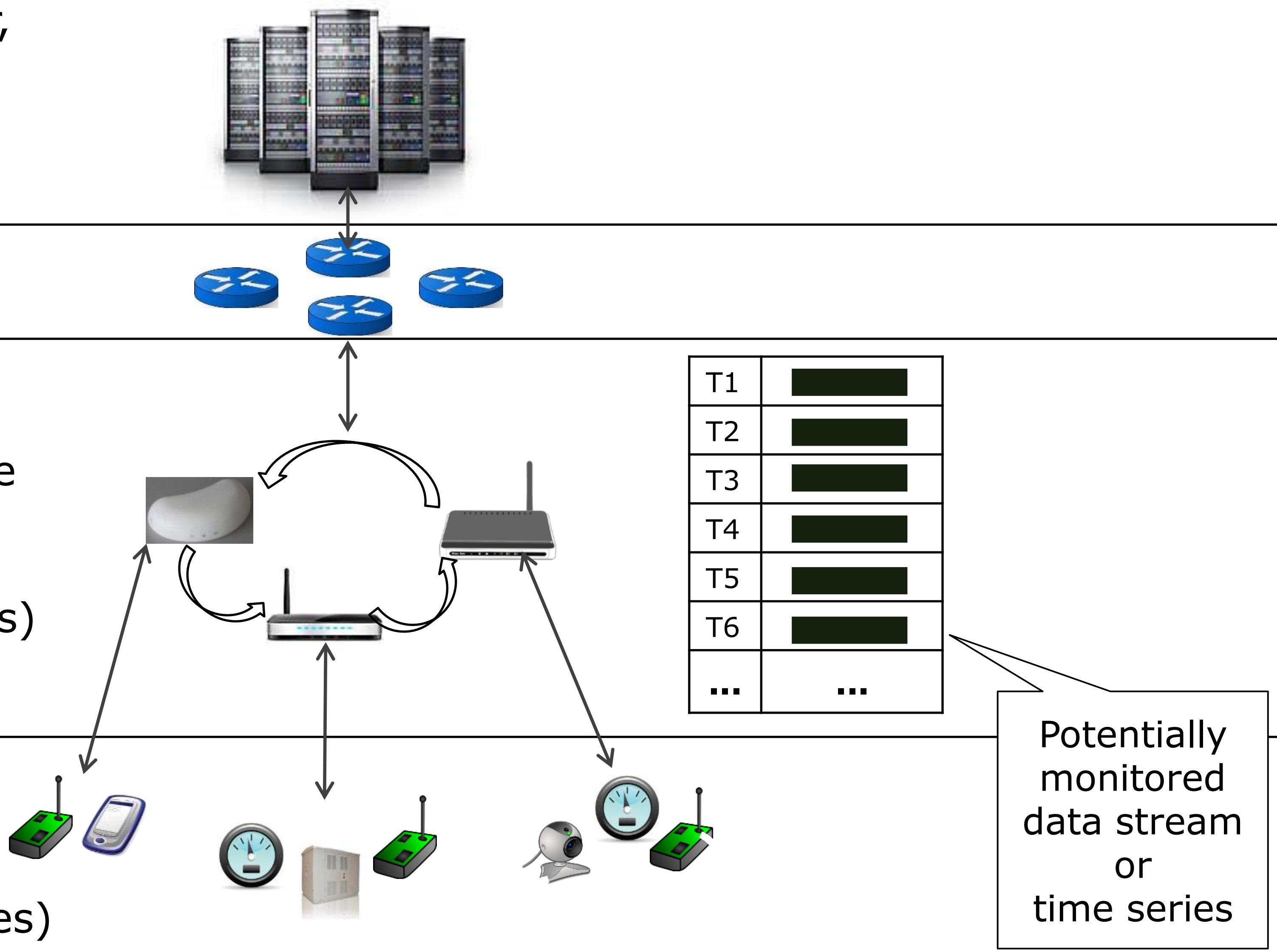
Network-edge computing

Data Center,
(Cloud)

Network
Core

Multi-service
Edge
(Gateways,
Edge routers)

Embedded
Systems &
Sensors
(M2M devices)



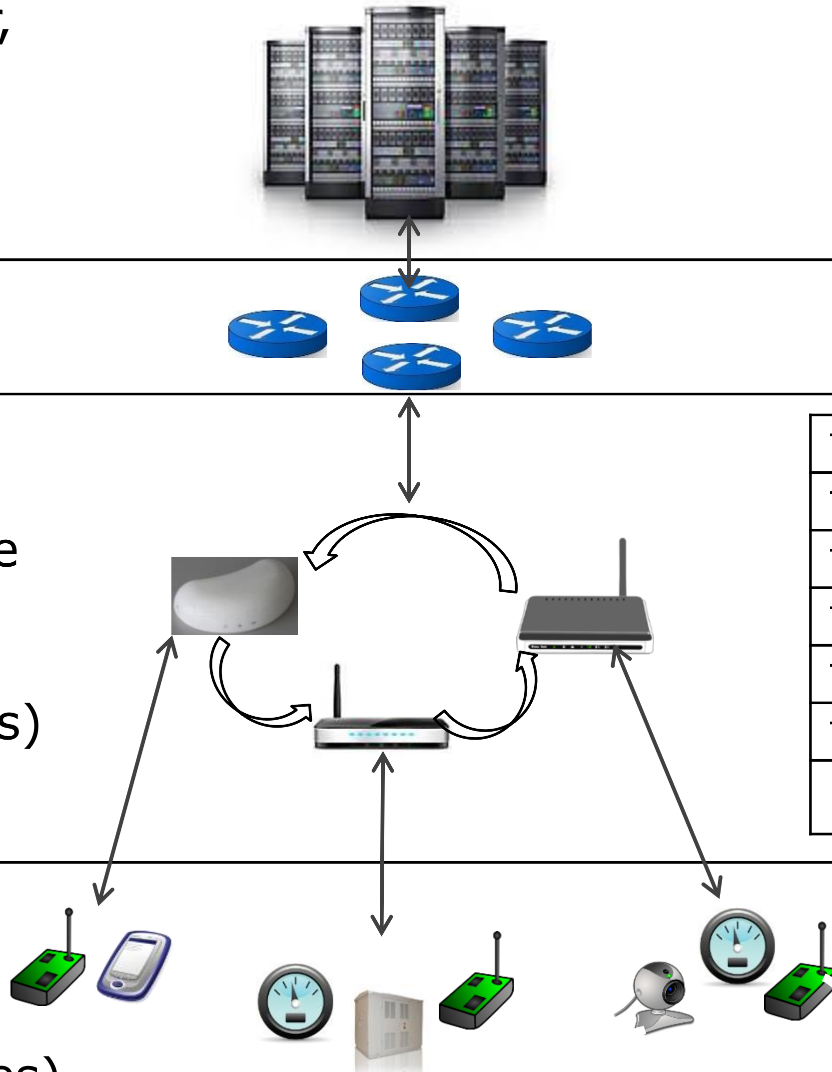
Network-edge computing

Data Center,
(Cloud)

Network
Core

Multi-service
Edge
(Gateways,
Edge routers)

Embedded
Systems &
Sensors
(M2M devices)



T1	[redacted]
T2	[redacted]
T3	[redacted]
T4	[redacted]
T5	[redacted]
T6	[redacted]
...	...

- Analyze and…

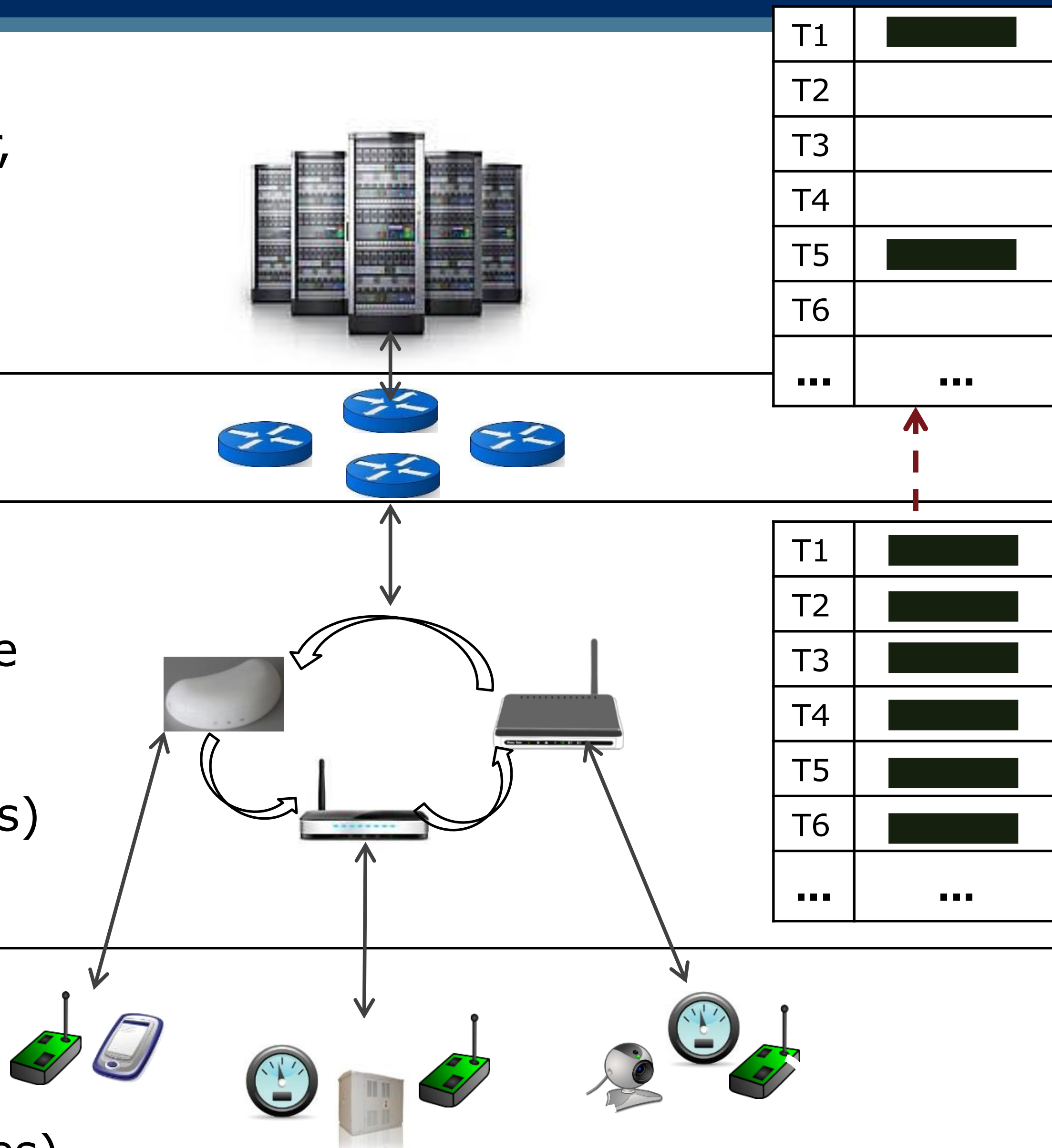
Network-edge computing

Data Center,
(Cloud)

Network
Core

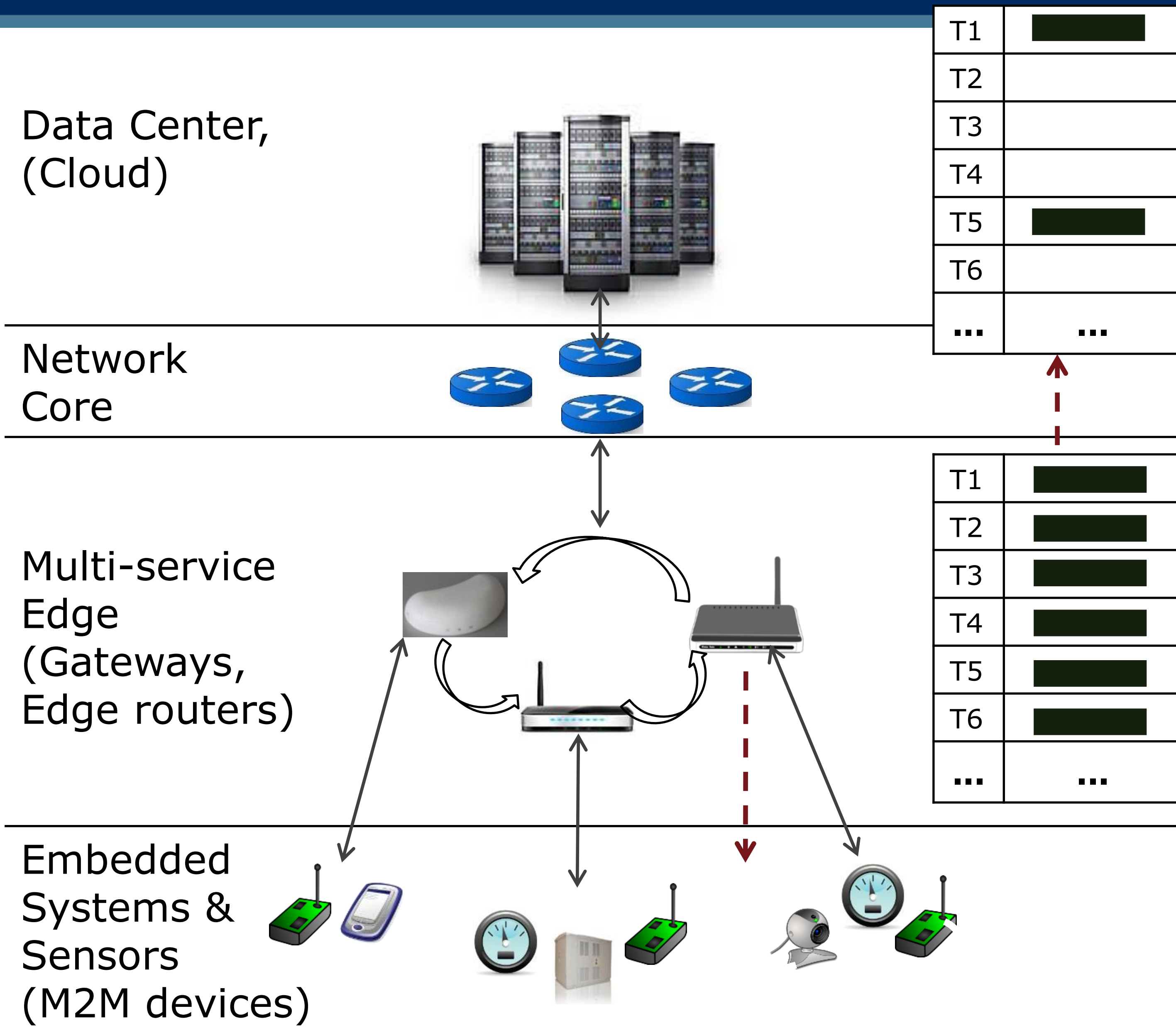
Multi-service
Edge
(Gateways,
Edge routers)

Embedded
Systems &
Sensors
(M2M devices)



- **Analyze and…**
  - Reduce

Network-edge computing



- Analyze and…
  - Reduce
  - React

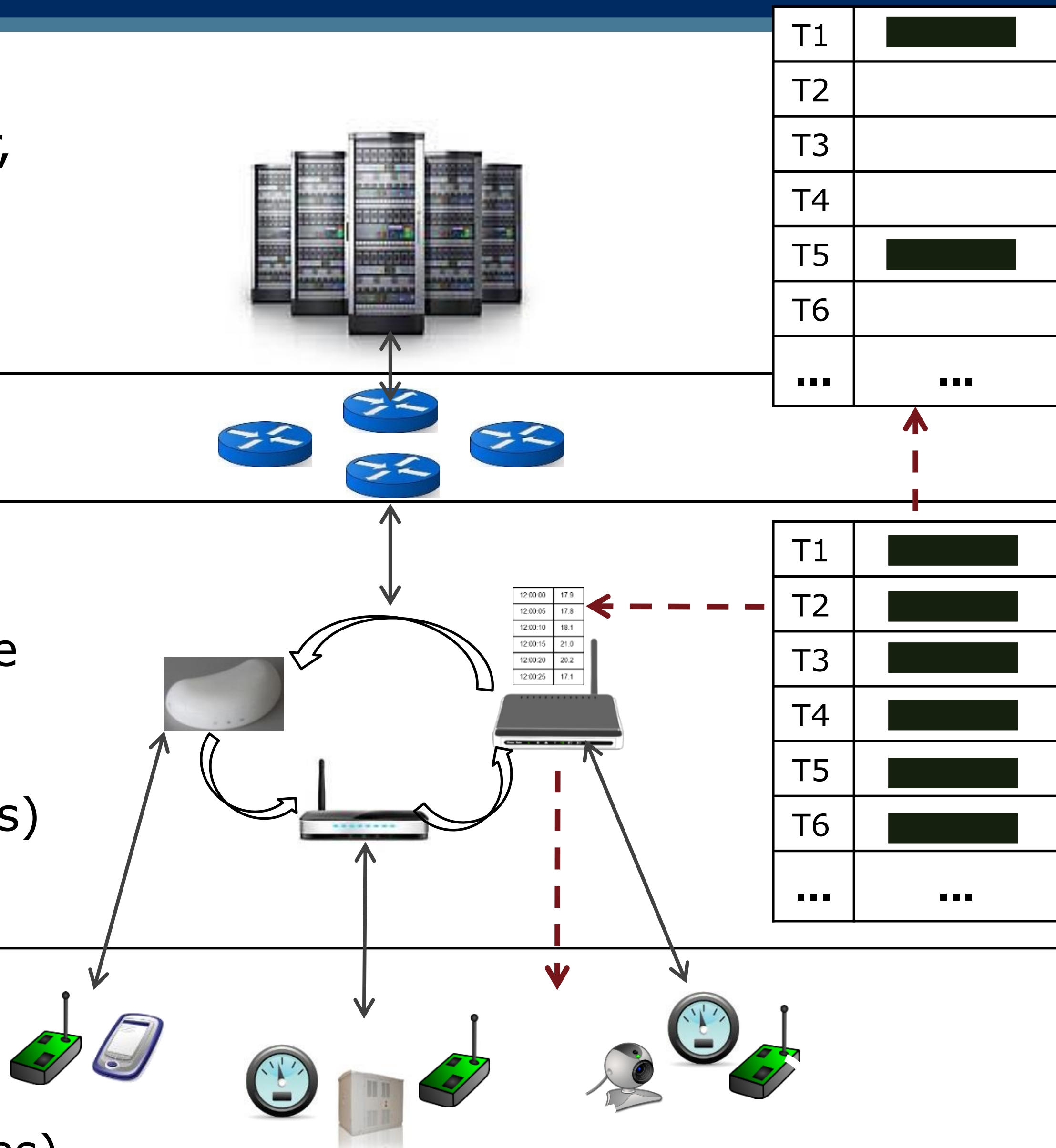
Network-edge computing

Data Center,
(Cloud)

Network
Core

Multi-service
Edge
(Gateways,
Edge routers)

Embedded
Systems &
Sensors
(M2M devices)



Analyze and…

- Reduce
- React
- Cache

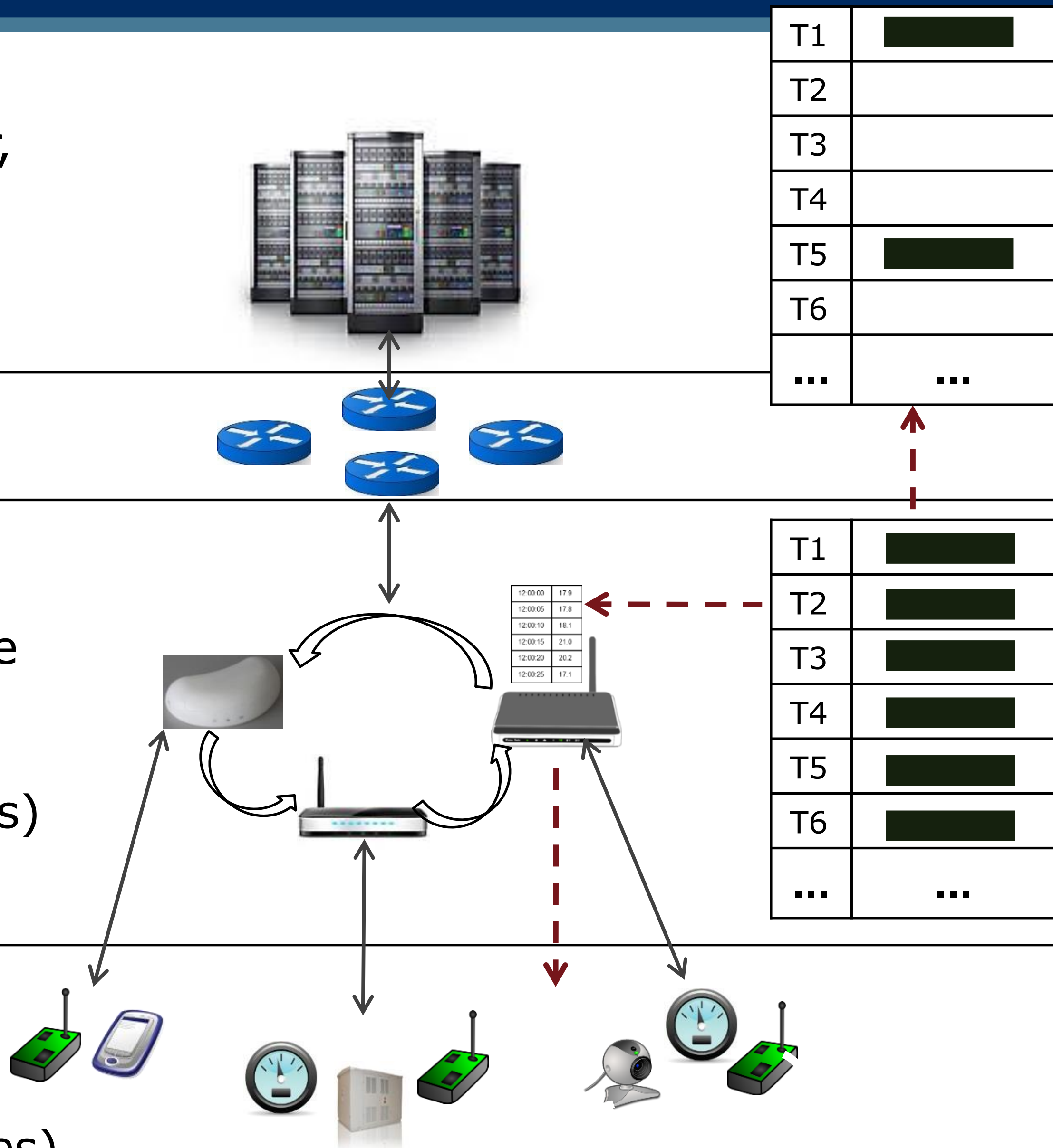
Network-edge computing

Data Center,
(Cloud)

Network
Core

Multi-service
Edge
(Gateways,
Edge routers)

Embedded
Systems &
Sensors
(M2M devices)



- Analyze and…
  - Reduce
  - React
  - Cache
  - …

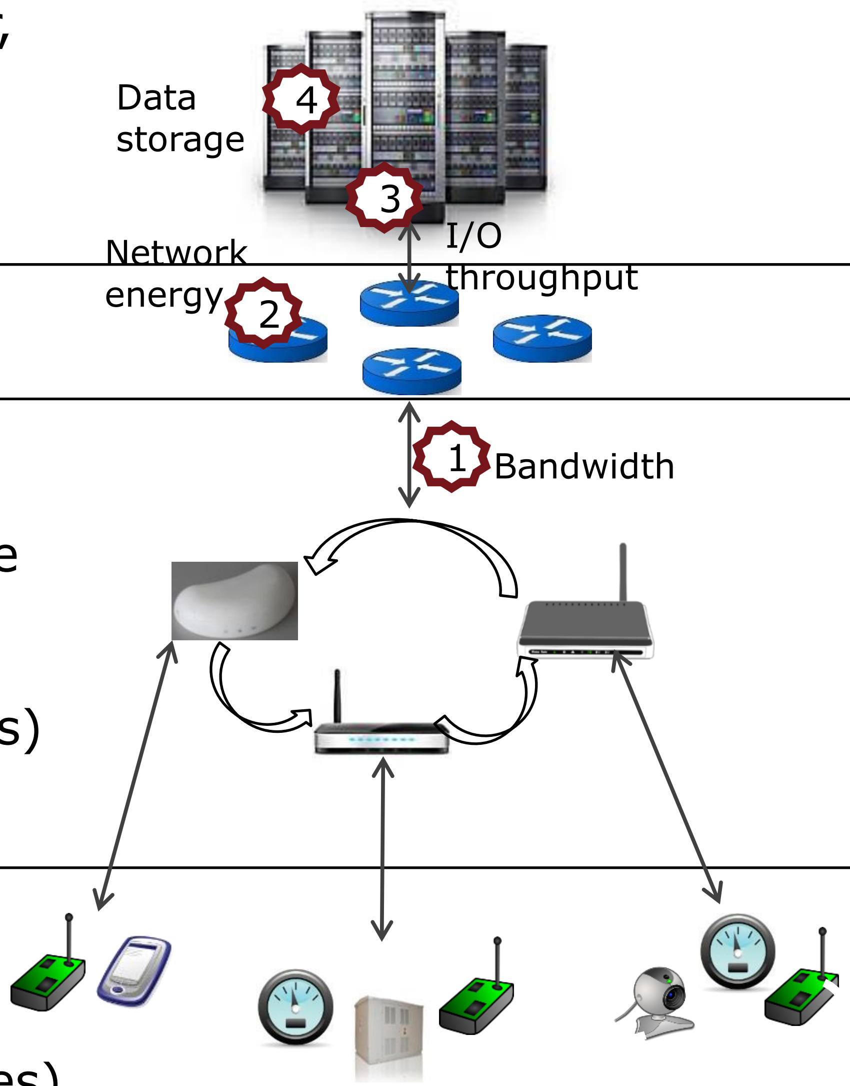
Why Network-edge computing?

Data Center, (Cloud)

Network Core

Multi-service Edge (Gateways, Edge routers)

Embedded Systems & Sensors (M2M devices)



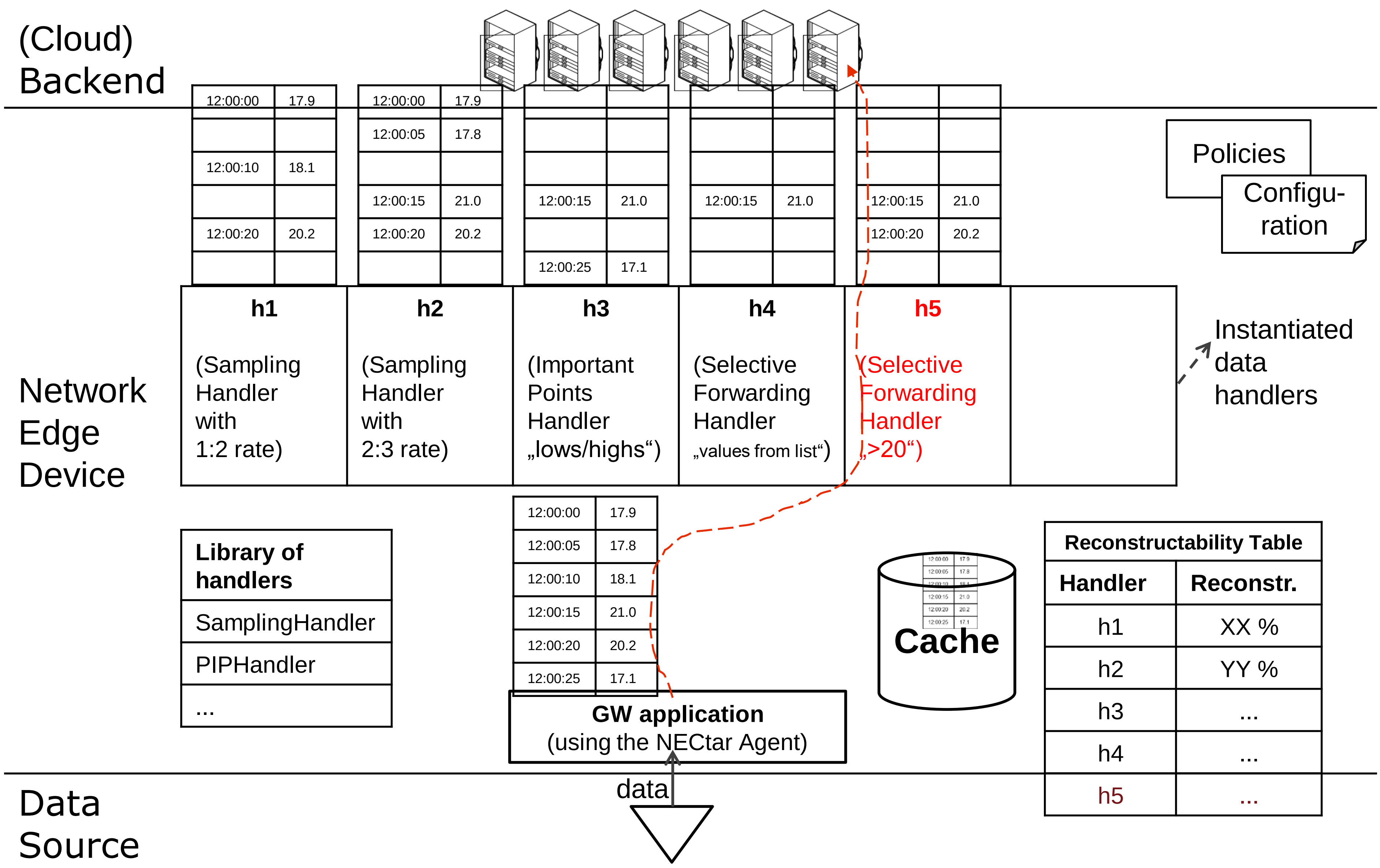
NECtar (Edge Data Handling/Filtering solution)

Our solution for real-time per-item data reduction based on exchangeable data handlers and „streamified" data reduction algorithms

What do we do differently?

- "Streamification"
  - Developed **data reduction solutions that work upon data streams**, i.e., "per incoming item", based on concepts of solutions that are currently designed to "compress" a posteriori, i.e., upon entire data sets
- Real-time aspect
  - **Reduced the "per item delay"** caused by the data handling at the edge by using cache reduction and cache projection techniques
- Reconstructability
  - Introduced "**reconstructability**" as data filtering criterion
- Exchangeable data handlers
  - Single-click **data handler instantiation** by implementing identical interfaces

NECtar Agent – Description of Operation



Reconstructability Table

Handler	Reconstr.
h1	XX %
h2	YY %
h3	...
h4	...
h5	...

NECtar Agent – Description of Operation

All Classes

Packages

eu.neclab.csst.apostolos.autoconfig.gw.fogagent
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.datahandlers
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.policies
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.reconstructability
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.util

eu.neclab.csst.apostolos.autoconfig.gw.fogagent.datahandlers

Classes

BaseHandler
FunctionalApproximationHandler
ImportantPointsHandler
LossyCompressionHandler
PiecewiseApproximationHandler
SamplingHandler
SelectiveForwardingHandler

Classes that impl. the same interfaces fulfilling internally one of the data reduction algorithms

Then we can apply and switch filtering logics as simply as...

```
BaseHandler h1
...
h1 = new SamplingHandler
(timeSeriesName, this, 2);
...
h1.handleData();
...
```

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

eu.neclab.csst.apostolos.autoconfig.gw.fogagent.datahandlers

Class SamplingHandler

java.lang.Object
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.datahandlers.BaseHa
eu.neclab.csst.apostolos.autoconfig.gw.fogagent.datahandlers.S

```
public class SamplingHandler
extends BaseHandler
```

SamplingHandler is a data reduction handler (i.e., extends BaseHandler) wh
time series) into the cache, but forwards only every n-th value to the Cloud (
instantiation or as a default value).

Author:

Apostolos Papageorgiou, NEC Laboratories Europe

Constructor Summary

Constructors

Constructor and Description
`SamplingHandler(java.lang.String name, DataSourceThread thre`   SamplingHandler Constructor that uses a default value for the dimension
`SamplingHandler(java.lang.String name, int dim, DataSourceTh`   Overloaded SamplingHandler Constructor that takes the dimension attrib

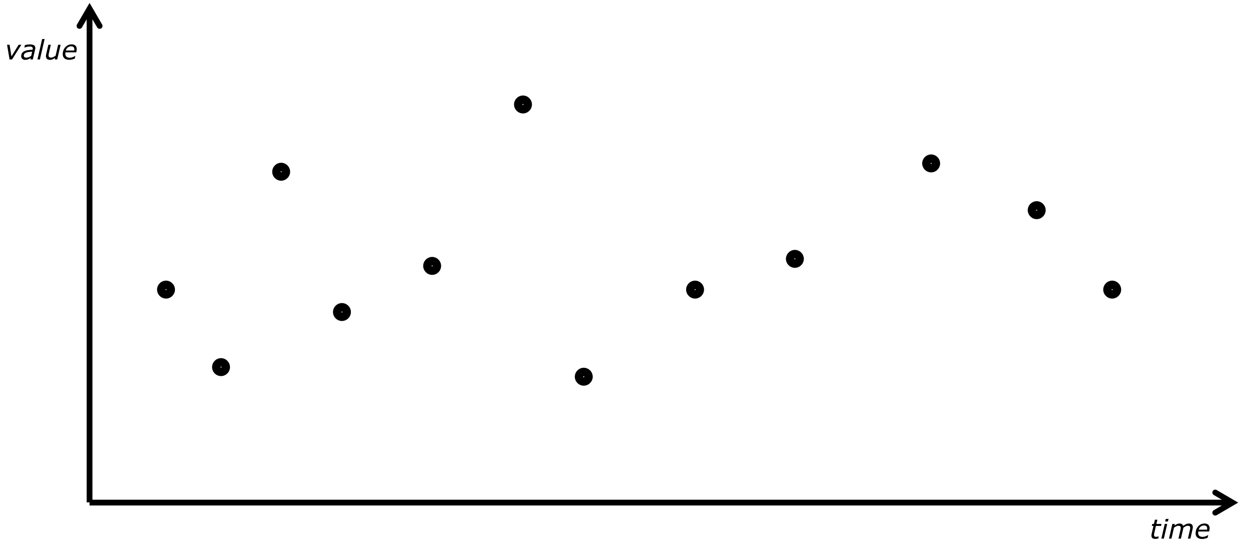
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



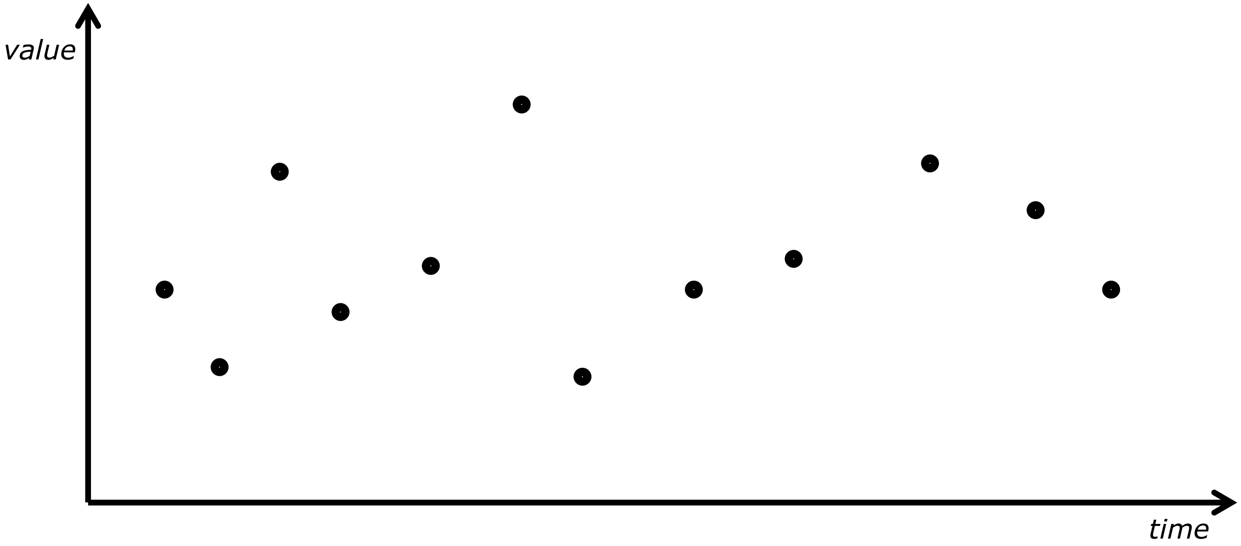
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

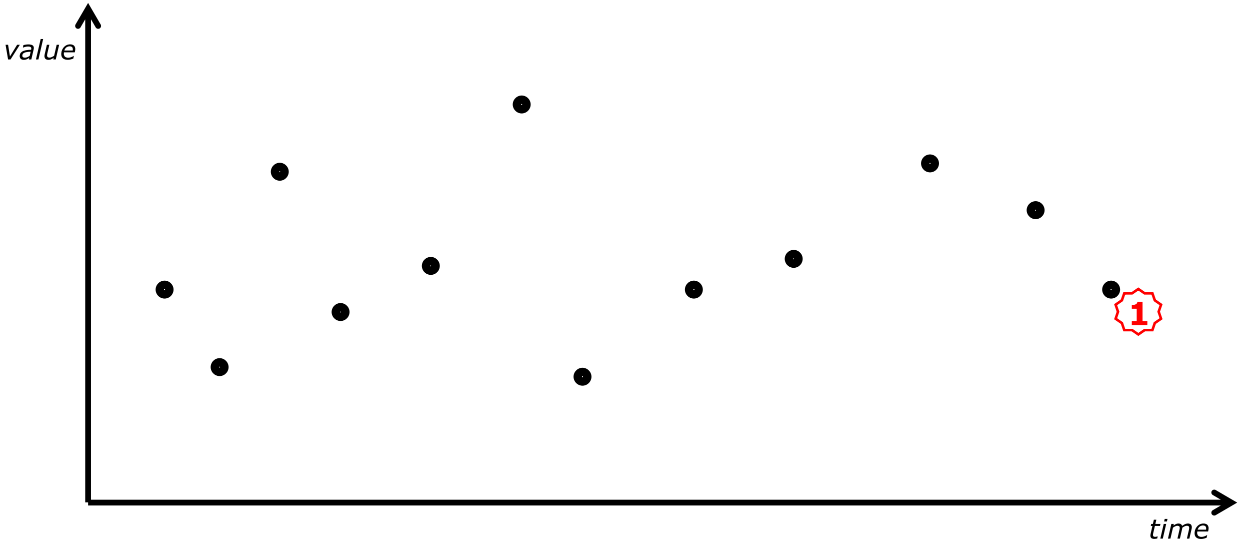
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

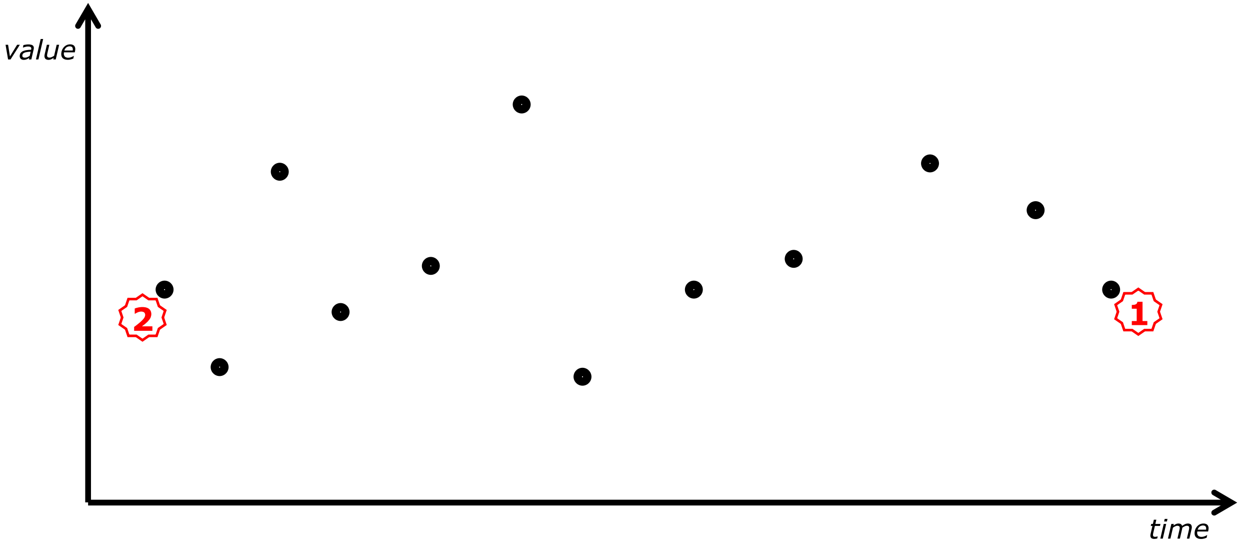
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

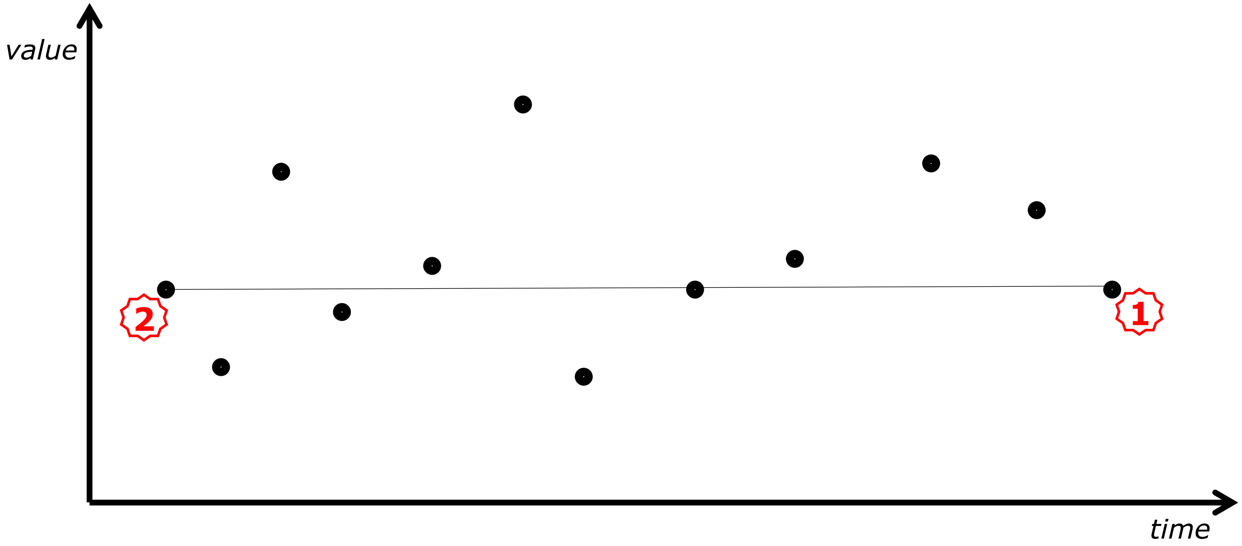
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

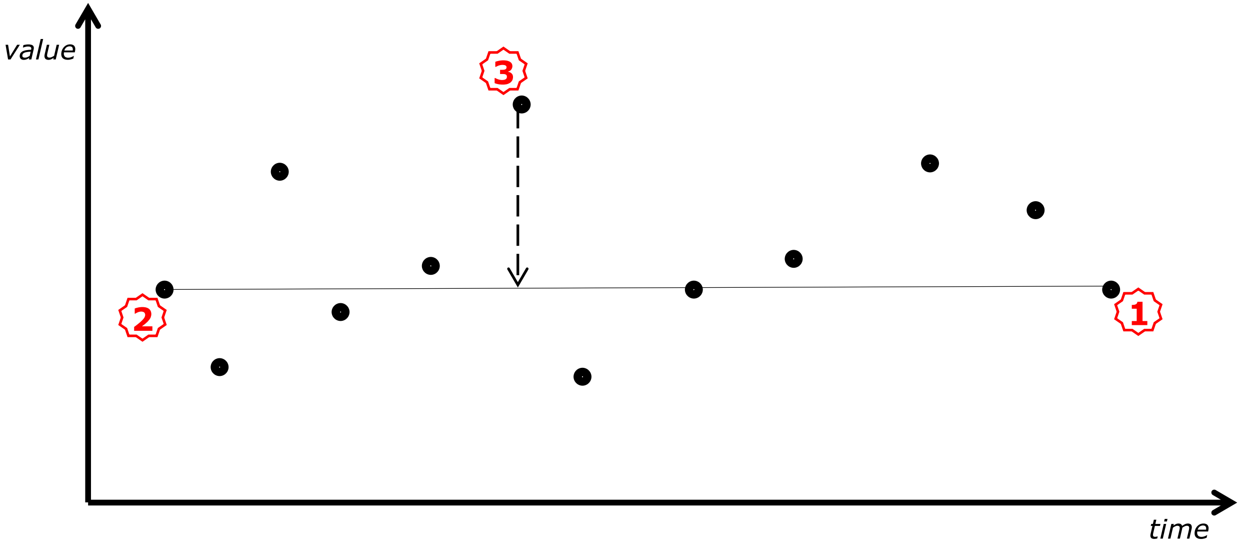
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

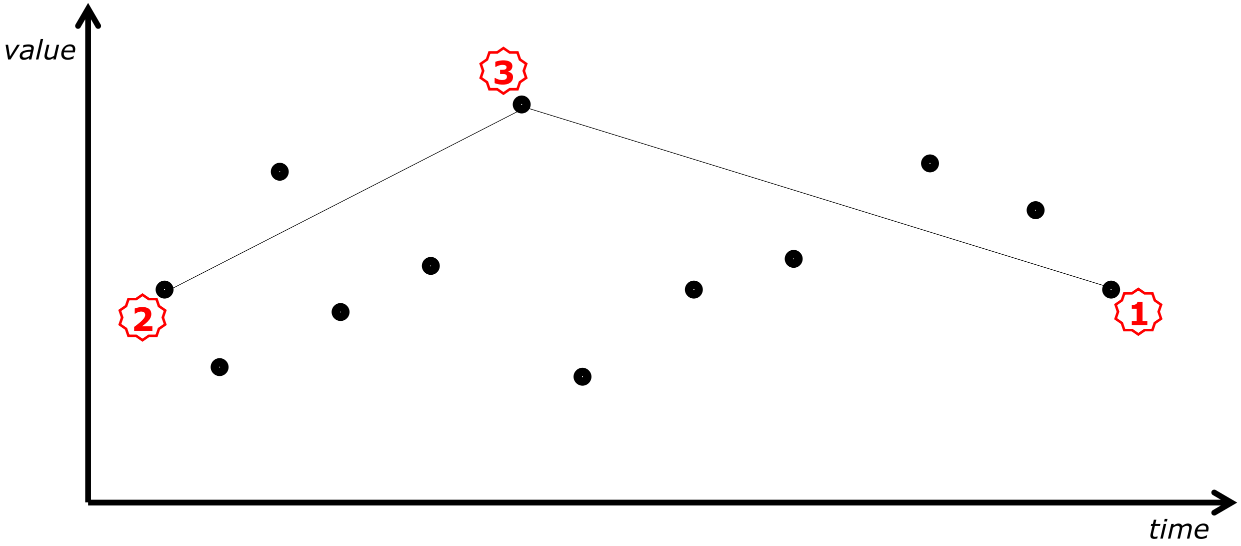
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

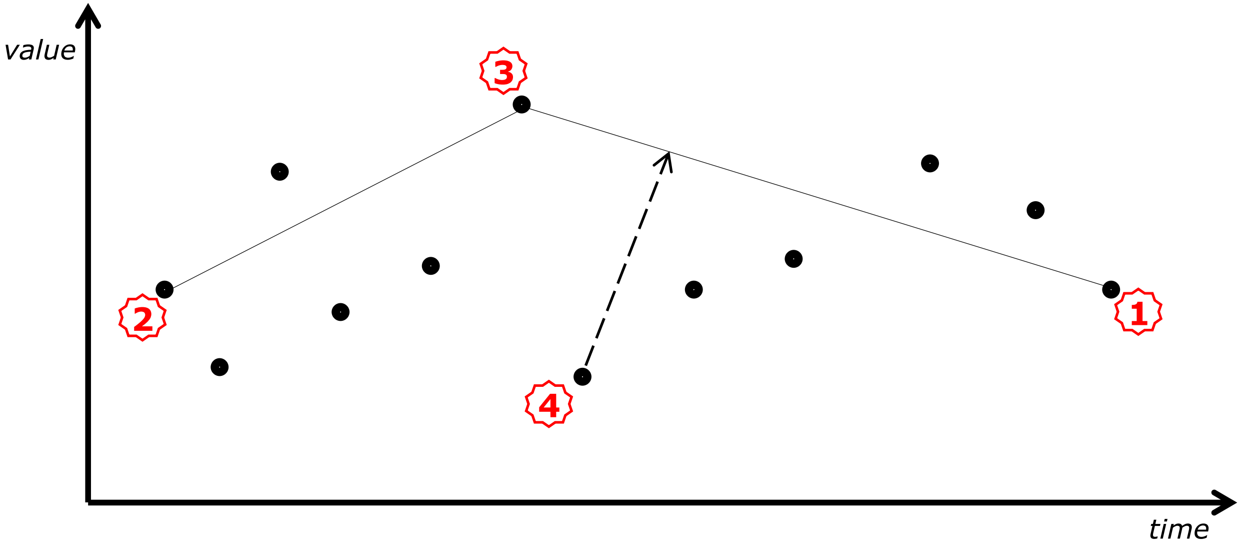
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

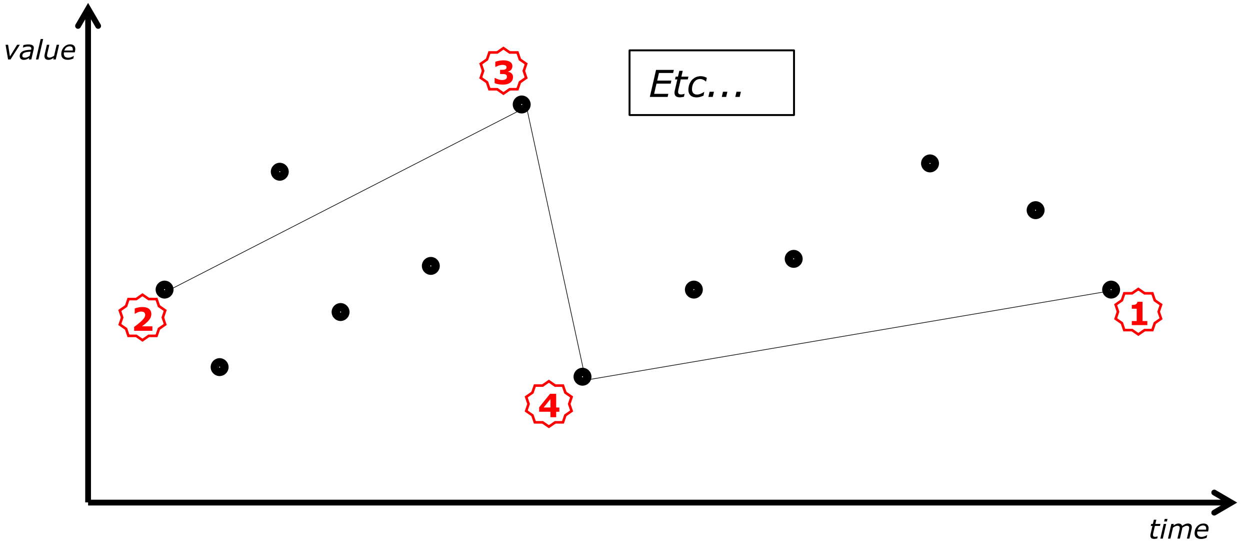
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



Numbering the items in order of importance ...

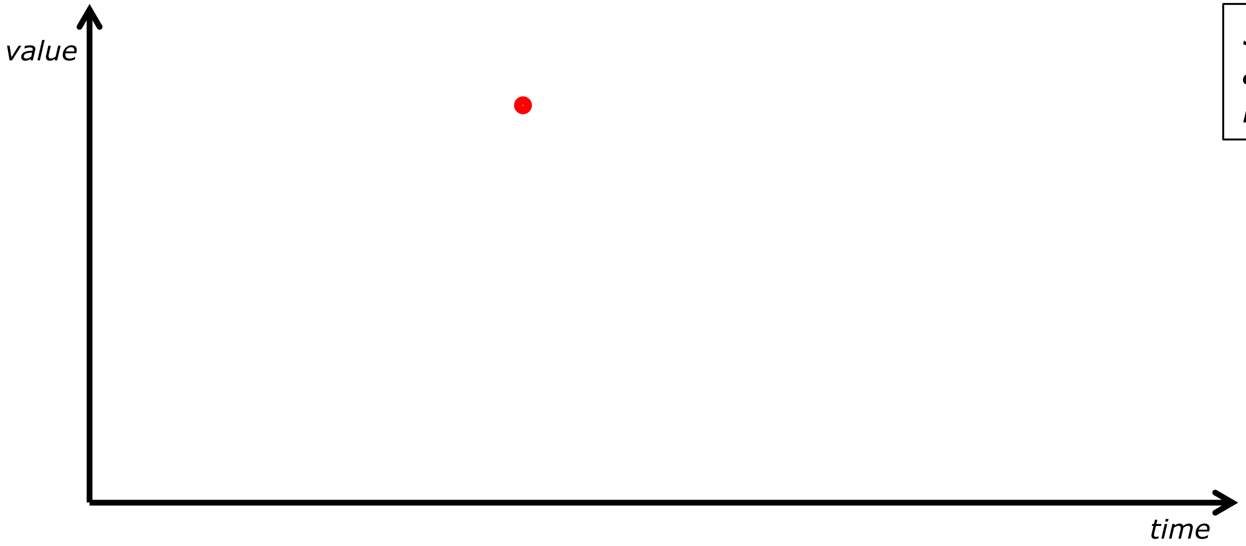
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

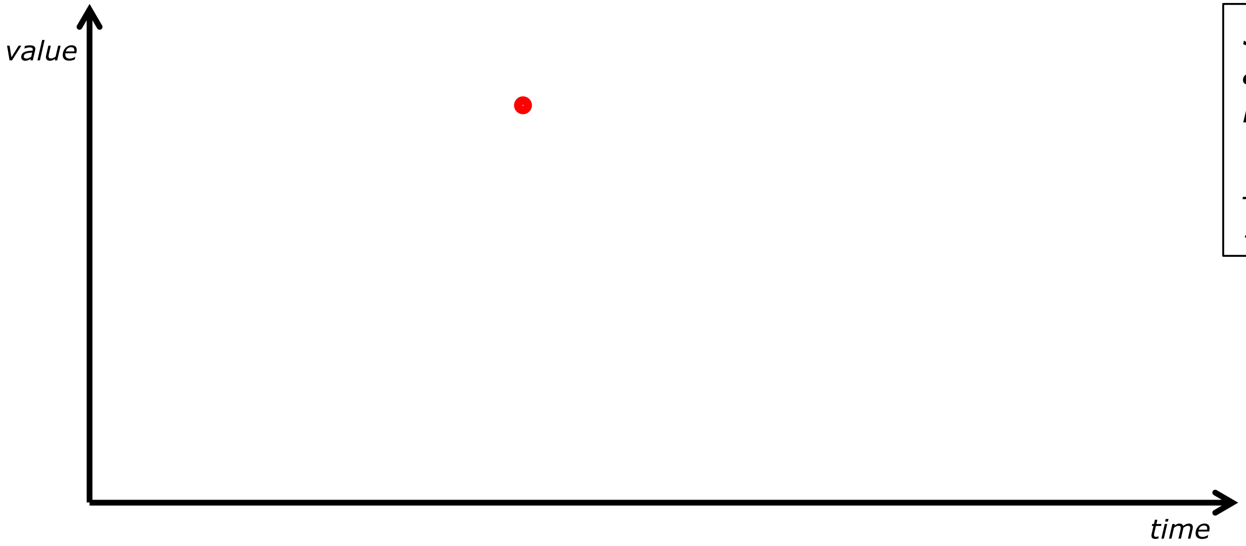
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:

1. *The data set is missing*

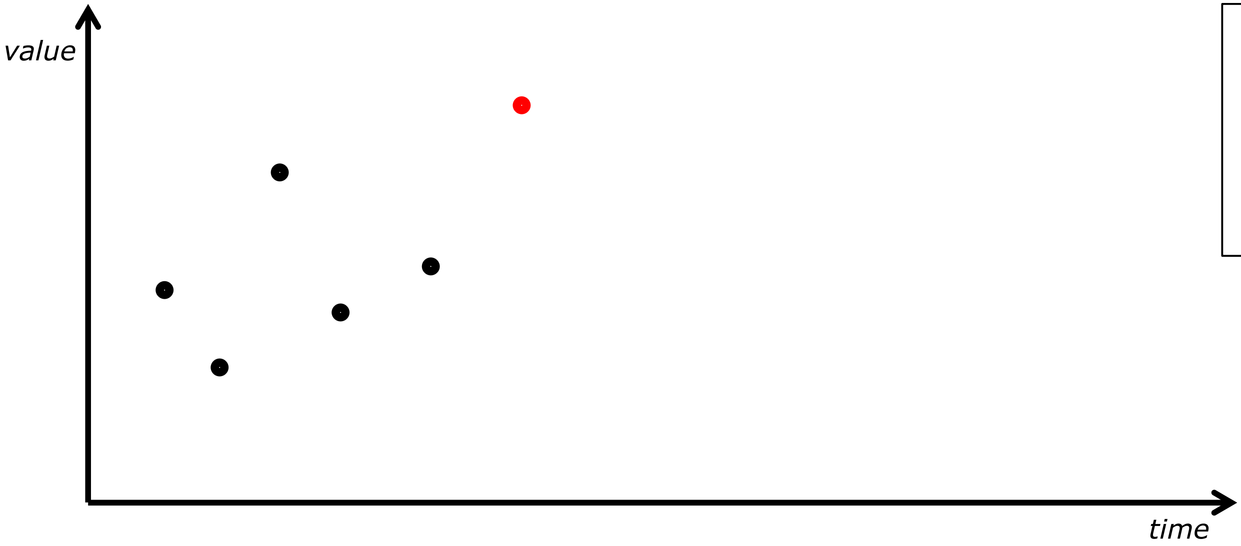
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:

1. The data set is missing

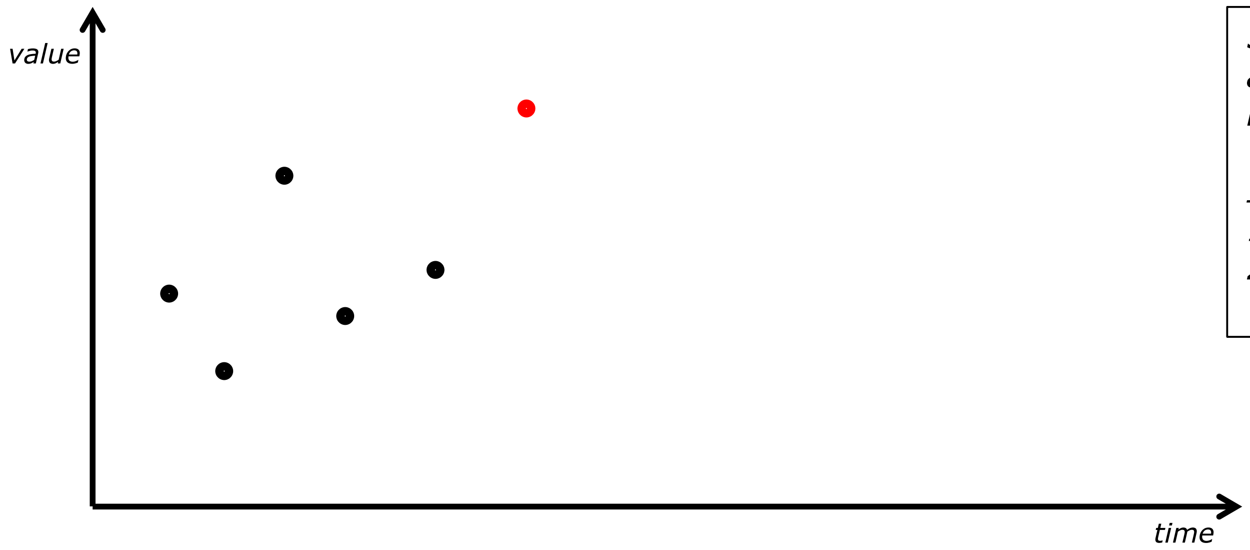
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:

1. *The data set is missing*
2. *Last item is always selected as most important*

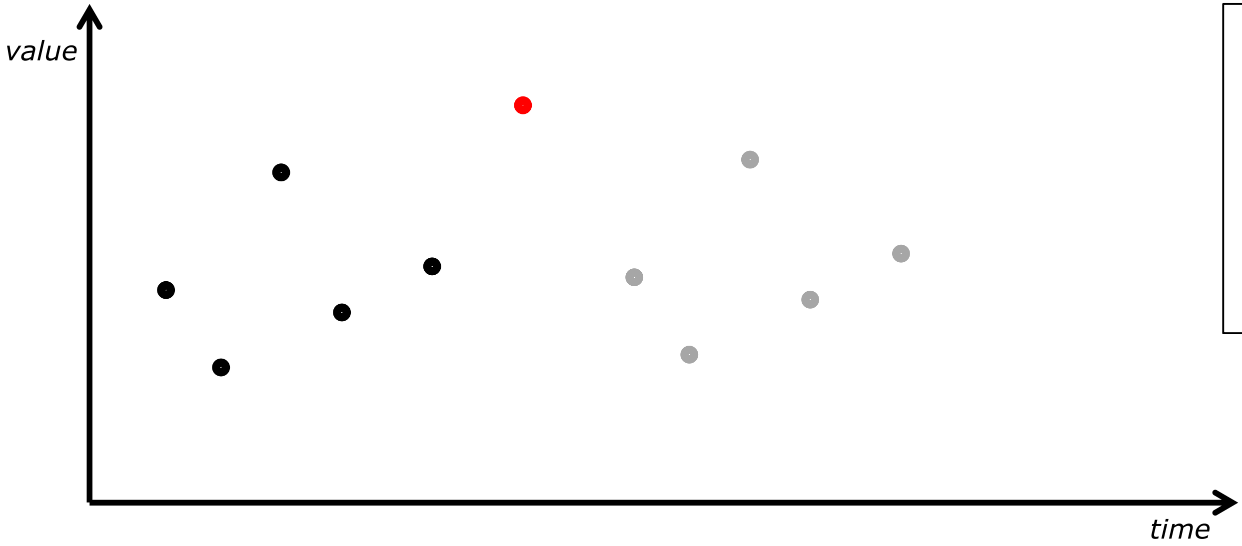
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. *The data set is missing*
2. *Last item is always selected as most important*

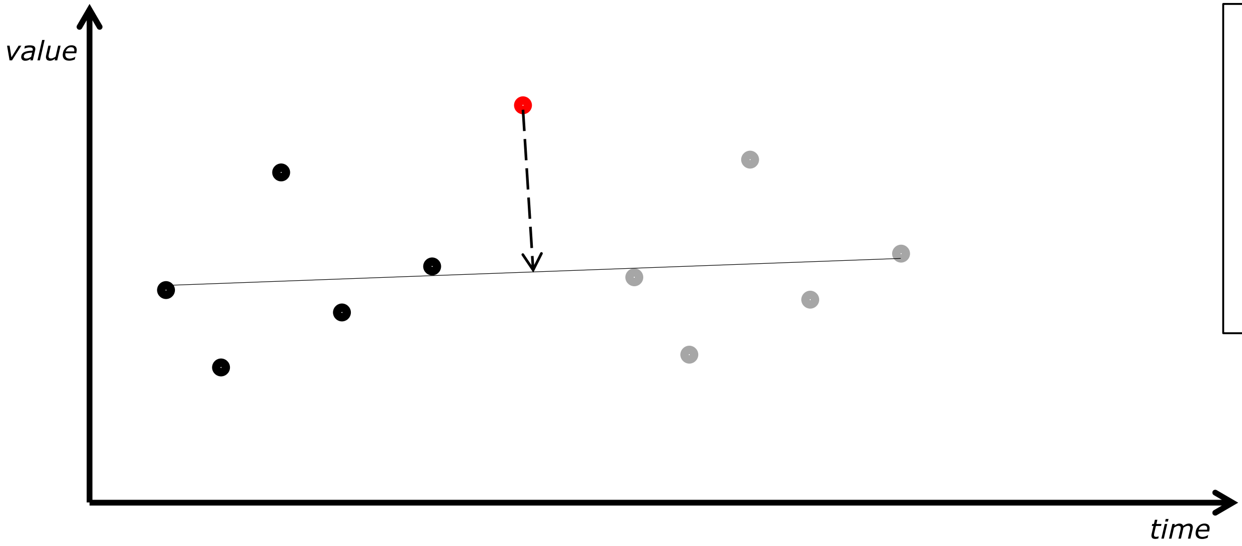
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item"...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:

1. *The data set is missing*
2. *Last item is always selected as most important*

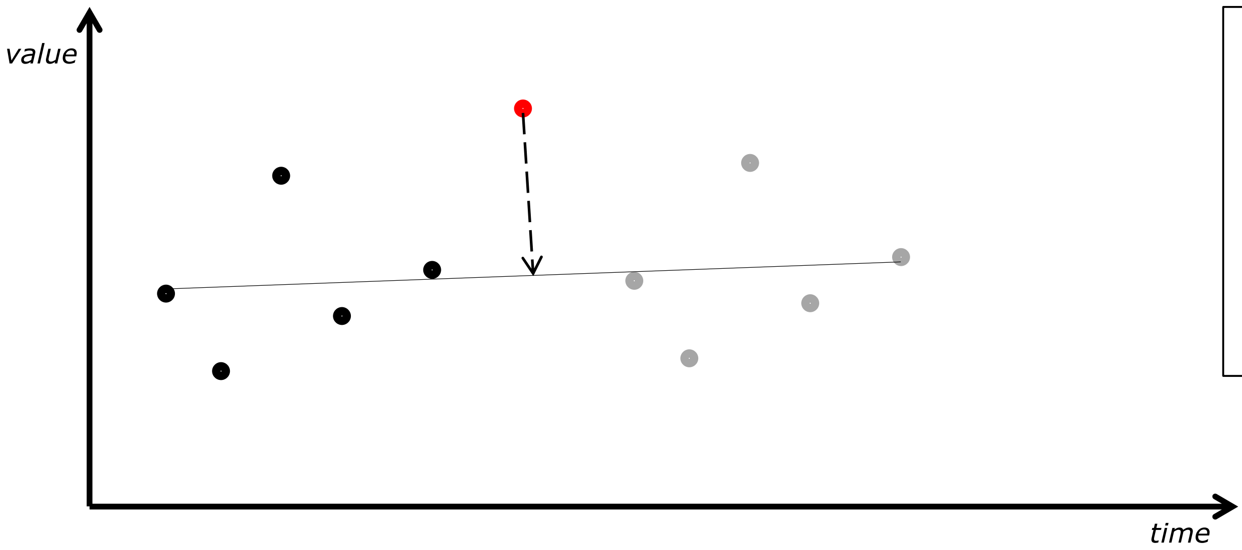
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:
1. *The data set is missing*
2. *Last item is always selected as most important*
3. *How will the future look like?*

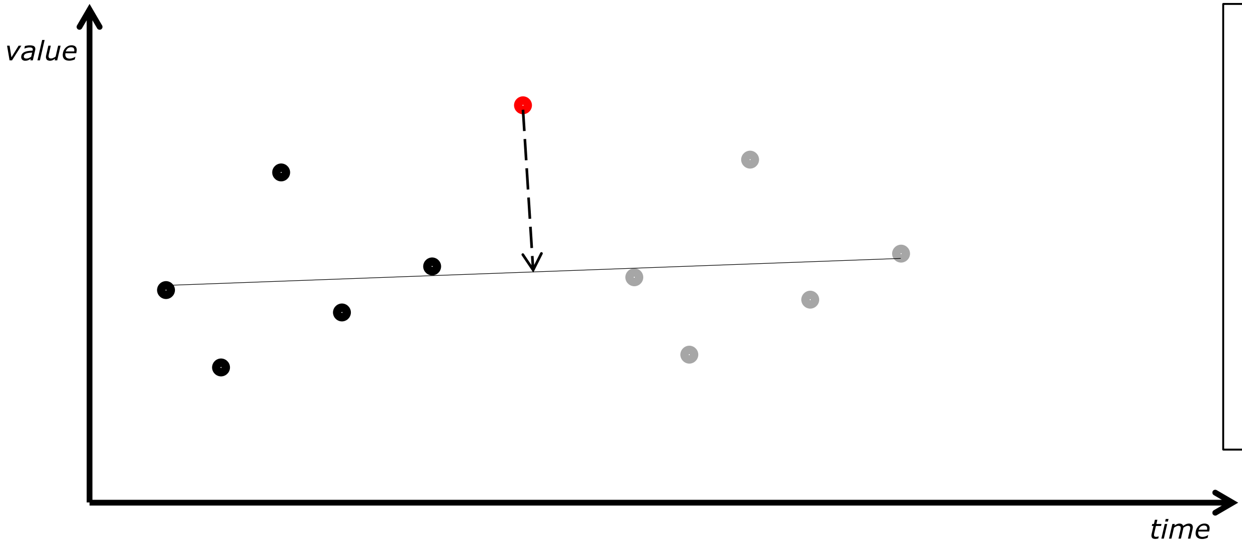
Streamification

What is the problem?

- It is straightforward to apply sampling or approximation „per incoming item“...
- ...BUT it is not possible to do this for sophisticated data reduction algorithms

Case Study: Perceptually Important Points (PIP) algorithm

- Simply explained:



So what happens when we try to apply this at the edge for an incoming item in real-time?

Issues:

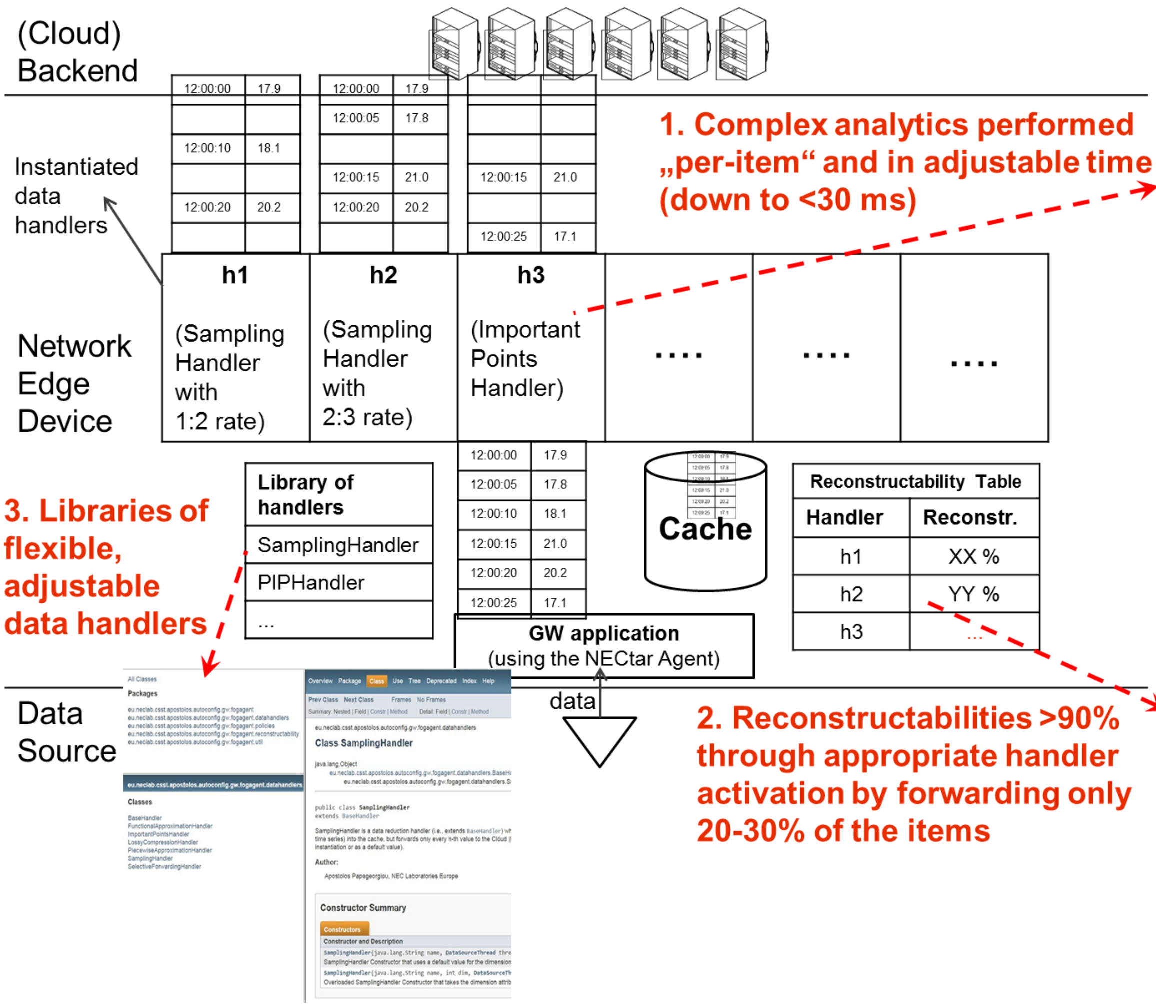
1. *The data set is missing*
2. *Last item is always selected as most important*
3. *How will the future look like?*
4. *How much time do I have for all this?*

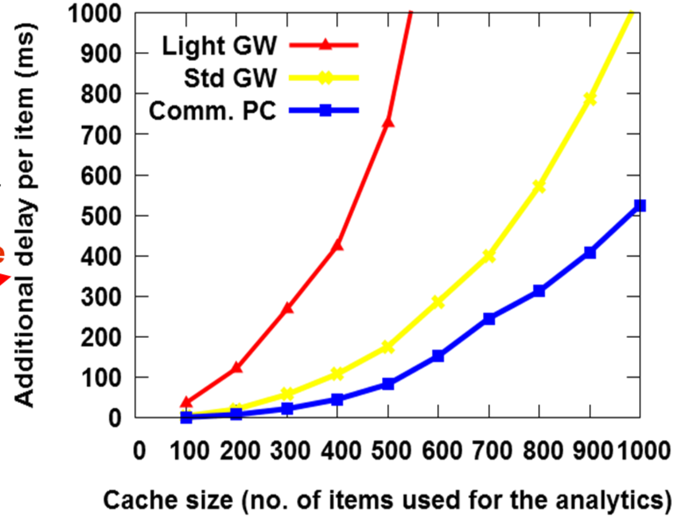
Streamification

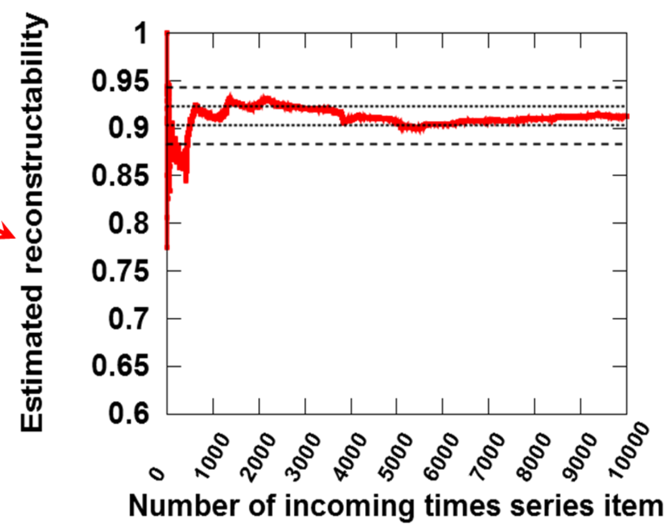
What we did:

- A „real-time" version of the PIP algorithm which
  - Uses a **cache with a delay-aware time window** as history
  - Uses **cache projection into the future** to add meaning to the measurement of important of the current item
  - Developed and evalauated three **different cache projection strategies**
    - CLONE: append a copy of the current item
    - TWIN: append a duplicate of the entire cache
    - AVG: append an item with an average value
  - Uses **cache reduction to make the "per item processing delay" negligible** compared to the transmission delay
  - Can be combined with a "**requested reconstructability degree**" in order to decide how important an item must be in order to be forwarded
  - (Please refer to our publications for details of the algorithms…)

Network-edge data filtering evaluation summary

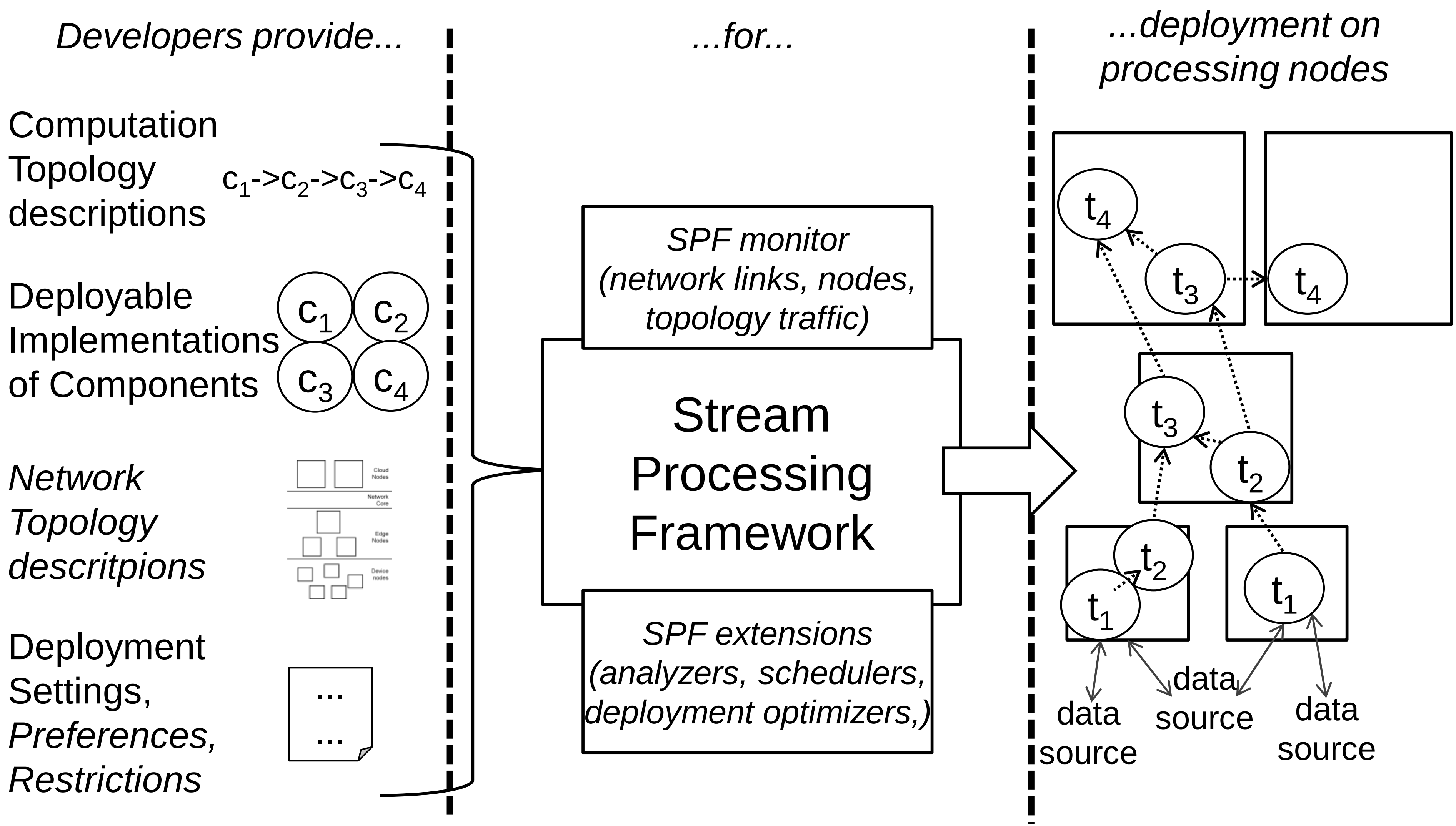






Edge deployment of IoT data streaming tasks

Stream Processing Frameworks functionality



Gap analysis

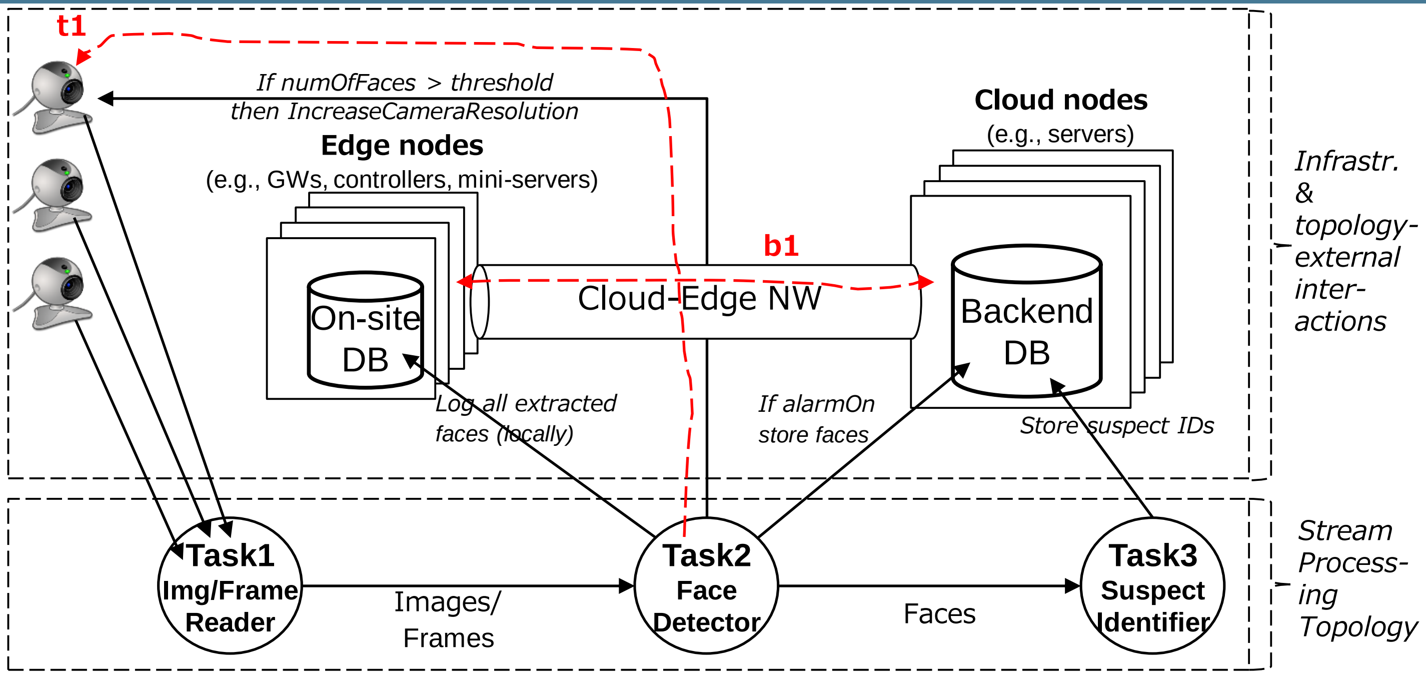
- SPFs are designed for performing stream processing in the Cloud

- In terms of task allocation and execution, standard SPFs ignore:
  - **node heterogeneity**
  - **geo-distributed nature** of IoT data sources
  - special **data traffic and delay** requirements
  - criticality of certain **sensors and actuators**

- In many cases edge computing can help, BUT this is not indicated by parameters that stream processing frameworks usually see

- For example...

Example surveillance topology with topology-external interactions



NOTE: Tasks can be instantiated as many times as required and their instances can be deployed on any of the Edge or Cloud nodes

t1	Camera resolution increase Latency	Time required from the moment Task2 has received a frame with many (unclear) faces until the moment that Task2 has issued the „resolution increase command" to the IP camera
b1	Cloud-edge bandwidth consumption	Amount of data traversing the Cloud-edge NW (per second), e.g., the sum of Task2->BackendDB and the Task2-Task3 traffic if Task1 and Task2 run on edge nodes and Task3 runs on Cloud nodes (or the sum of Task1->Task2 and Task2->OnSiteDB traffic, if Task2 is moved to the Cloud etc.)

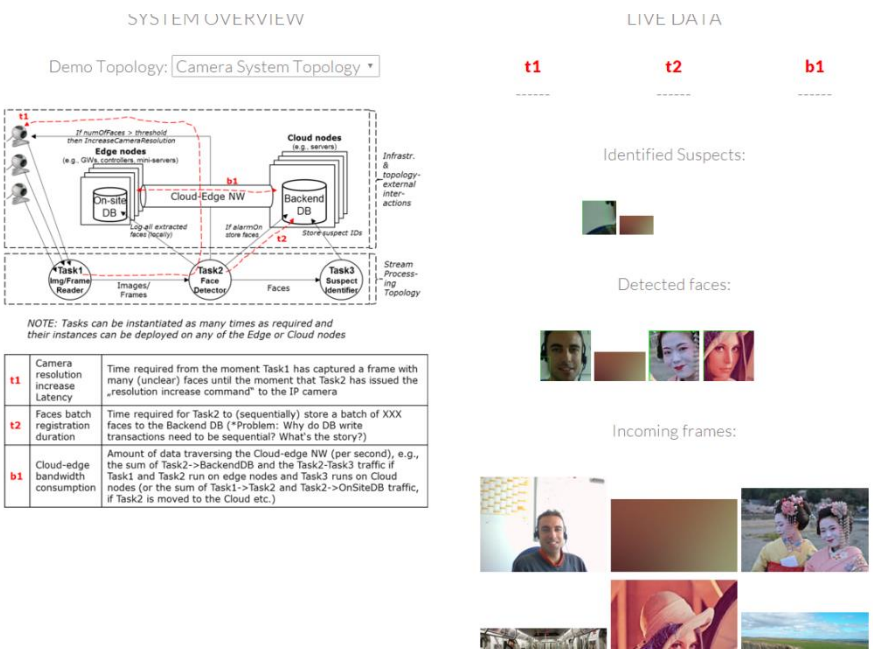
The key concept of Edge Computing Descriptors

- There are **three main things** (categories of characteristics) that shall **determine if a task is relevant to network edge computing** (and shall be executed at the edge) or not. These are:
  - The **interfaces of the task with the environment**, i.e., control of actuators, direct provision of intermediate results to users, event- or alarm-raising.
  - The characteristics of the **databases** with which the task interacts.
  - The **task computation characteristics**, namely its CPU- and data-intensity and security restrictions.

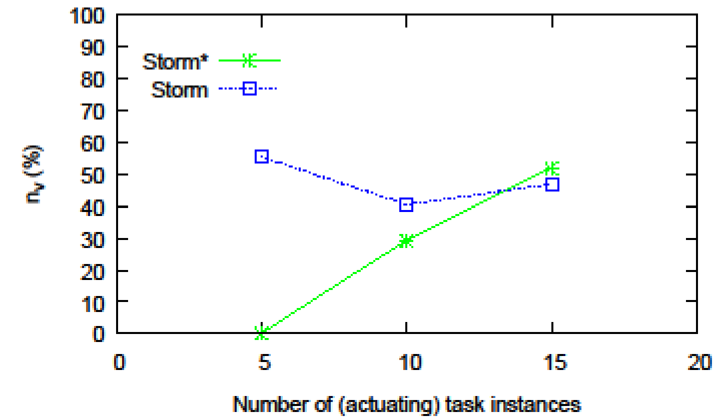

```
{
  "edgeInteractions": {
    "edgeExecutionRequired": "no",
    "controlledActuators": [
      {
        "name": "actuator1",
        "type": "switch",
        "area": "area1",
        "geo": "coordinates",
        "latencyRequirement": "low/medium/high"
      },
      {
        "name": "actuatorX",
        "type": "switch",
        "area": "areaX",
        "geo": "coordinates",
        "latencyRequirement": "low/medium/high"
      }
    ],
    "provisionedIntermediateResults": [
      {
        "result": "res1",
        "access": "pubSub",
        "mainUsersArea": "area1",
        "mainUsersLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      },
      {
        "result": "resX",
        "access": "pubSub/pull",
        "mainUsersArea": "areaX",
        "mainUsersLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      }
    ],
    "potentiallyRaisedEvents": [
      {
        "event": "event1",
        "propagationArea": "area1",
        "propagationLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      },
      {
        "event": "eventX",
        "propagationArea": "areaX",
        "propagationLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      }
    ]
  },
  "editedDatabases": [
    {
      "name": "db1",
      "type": "rdbms",
      "area": "area1",
      "dbLayer": "edge/core/cloud",
      "mainUsersLayer": "edge/core/cloud",
      "latencyRequirement": "low/medium/high"
    },
    {
      "name": "dbX",
      "type": "rdbms/nosql",
      "area": "areaX",
      "dbLayer": "edge/core/cloud",
      "mainUsersLayer": "edge/core/cloud",
      "latencyRequirement": "low/medium/high"
    }
  ],
  "computationCharacteristics": {
    "cpuIntensity": "low/medium/high",
    "expectedDataIOIntensity": "low/medium/high",
    "forbiddenAreas": ["forbiddenArea1", "forbiddenArea2", "forbiddenAreaX"],
    "forbiddenLayers": ["forbiddenLayer1", "forbiddenLayer2", "forbiddenLayerX"],
    "forbiddenDomains": ["forbiddenDomain1", "forbiddenDomain2", "forbiddenDomainX"],
    "specificDevice": "targetDeviceName"
  }
}
```

Implementation and evaluation summary

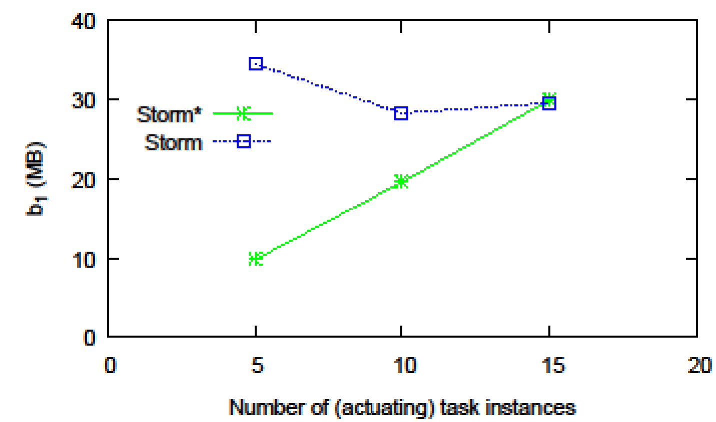
- We implemented our „edge-aware SPF“ concept as an extension of Apache Storm, evaluated it against Storm, and tested it with example topologies...



Latency violations:



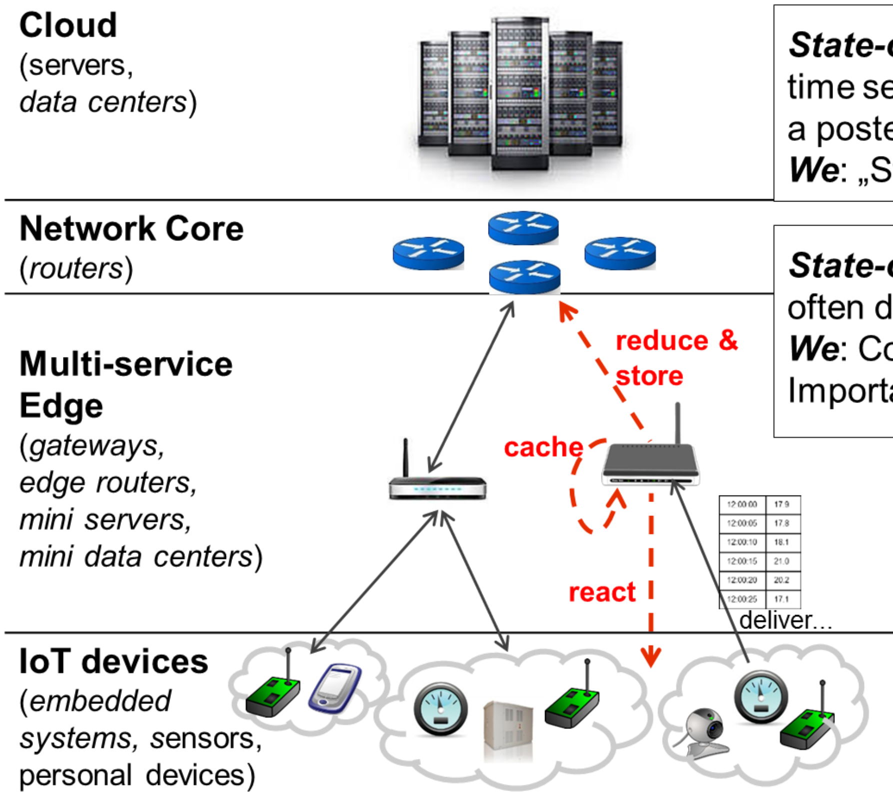
Used Cloud-Edge bandwidth:



Conclusion

Conclusion

Data Filtering



State-of-the-art challenge: The most efficient time series reduction algorithms can work only a posteriori upon complete data sets
We: „Streamify" such algorithms

State-of-the-art challenge: Information loss might often demotivate data reduction
We: Consider data „reconstructability" and Importance before reducing

State-of-the-art challenge: Dynamic switching between different types of network edge processing can be heavy or complex for gateway middleware
We: Develop flexible, adjustable data handlers enabling fast system adaptation

Edge-aware task deployment

- Consider edge computing characteristics such as…
  - Critical actuations, DB interactions, user locations, IoT node characteristics, system usage
- …in order to place tasks of IoT processing chains at the right "edges"

\Orchestrating a brighter world

NEC brings together and integrates technology and expertise to create
the ICT-enabled society of tomorrow.
We collaborate closely with partners and customers around the world,
orchestrating each project to ensure all its parts are fine-tuned to local needs.

Every day, our innovative solutions for society contribute to
greater safety, security, efficiency and equality,
and enable people to live brighter lives.