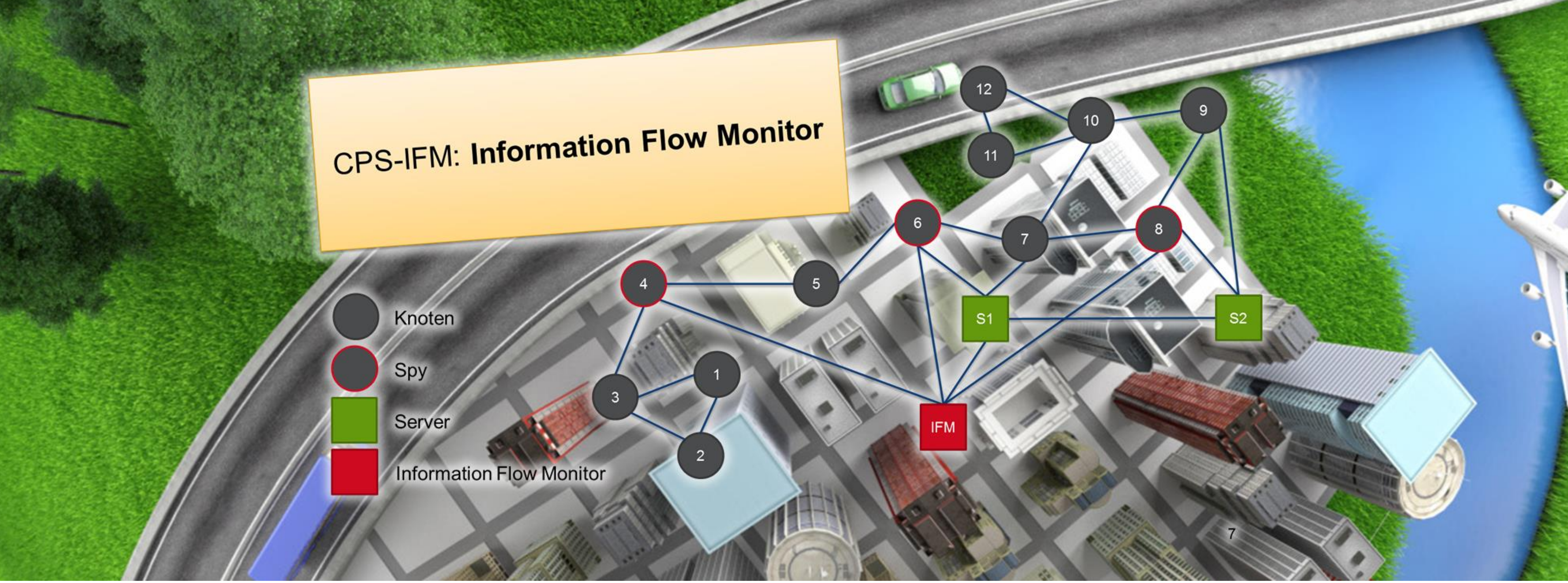


Using Blockchain Technology to Ensure Trustful Information Flow Monitoring in CPS

Stefan Gries, Ole Meyer, Florian Wessling, Marc Hesenius and Volker Gruhn

Seattle, April 2018

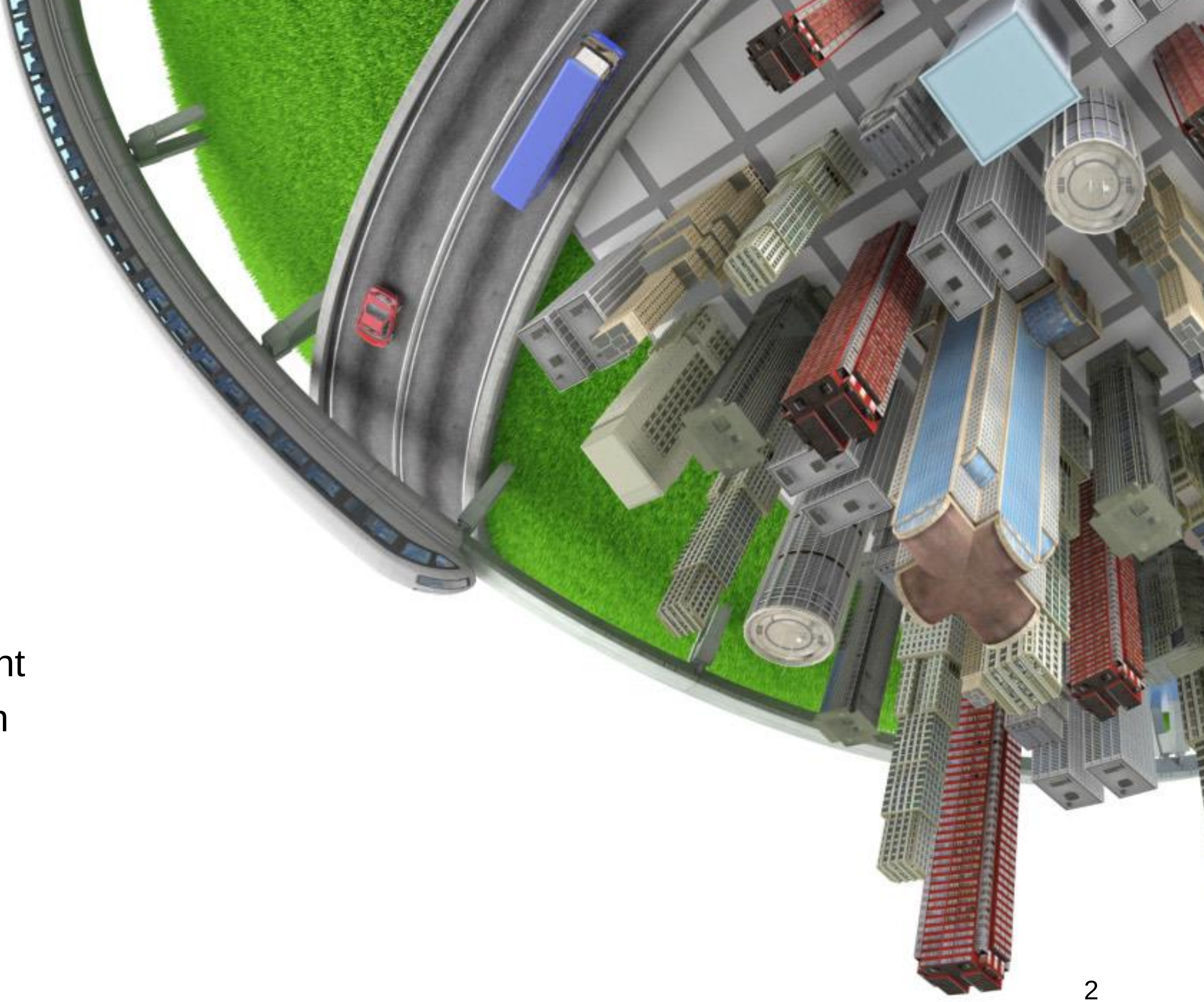
PALUNO The Ruhr Institute for Software Technology

Image: https://ec.europa.eu/digital-agenda/en/system-systems/

UNIVERSITÄT DUISBURG ESSEN

Agenda

- Motivation
- Problem Statement
- Solution Approach



PALUNO
The Ruhr Institute for Software Technology

What are Cyber-Physical Systems?

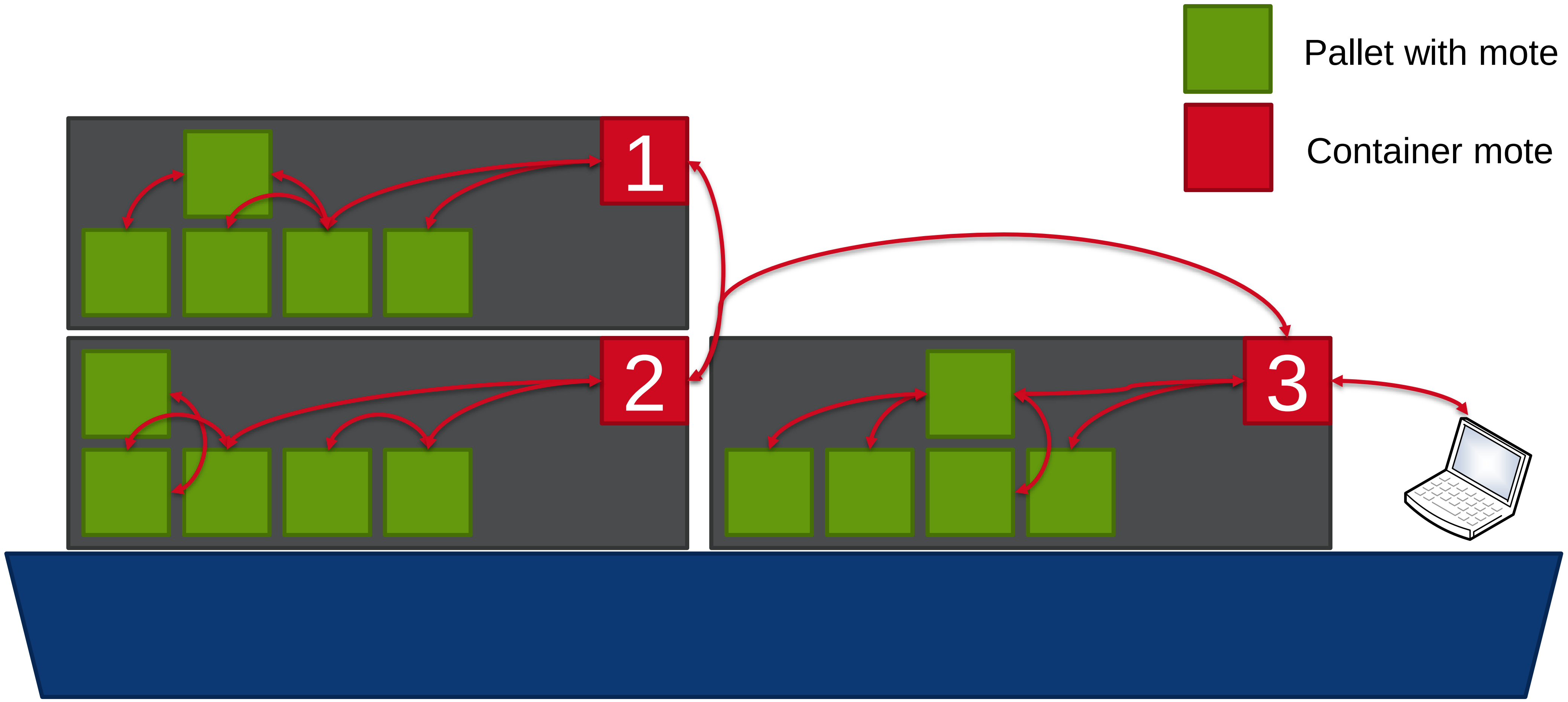
„CPS are **interconnected** Systems that observe and manipulate **real objects and processes**. They automatically adapt to the current real-world environment and its constraints. Furthermore, CPS may consist of **heterogeneous components** unknown at development time, thus allowing dynamic extension at runtime. CPS therefore show **emergent behavior** that leads to new problem solving strategies depending on the current setup of autonomous components." – CPS definition used at University of Duisburg-Essen

Our view on CPS

- CPS…
  - … bring the software engineering and traditional engineering domains closer together
  - … create new and interesting challenges and opportunities
  - … need multiple domain experts to create and engineer
  - … are the union of many different known and new technologies
- But: NEW problems arise from the usage of sensors, actuators, many nodes, software and hardware in one huge network.
- Engineering Process for CPS has to be rethought!
- My focus today:
  - new problems while **maintaining a CPS**
  - new problems in **trusting different nodes in CPS**

Container Network Example

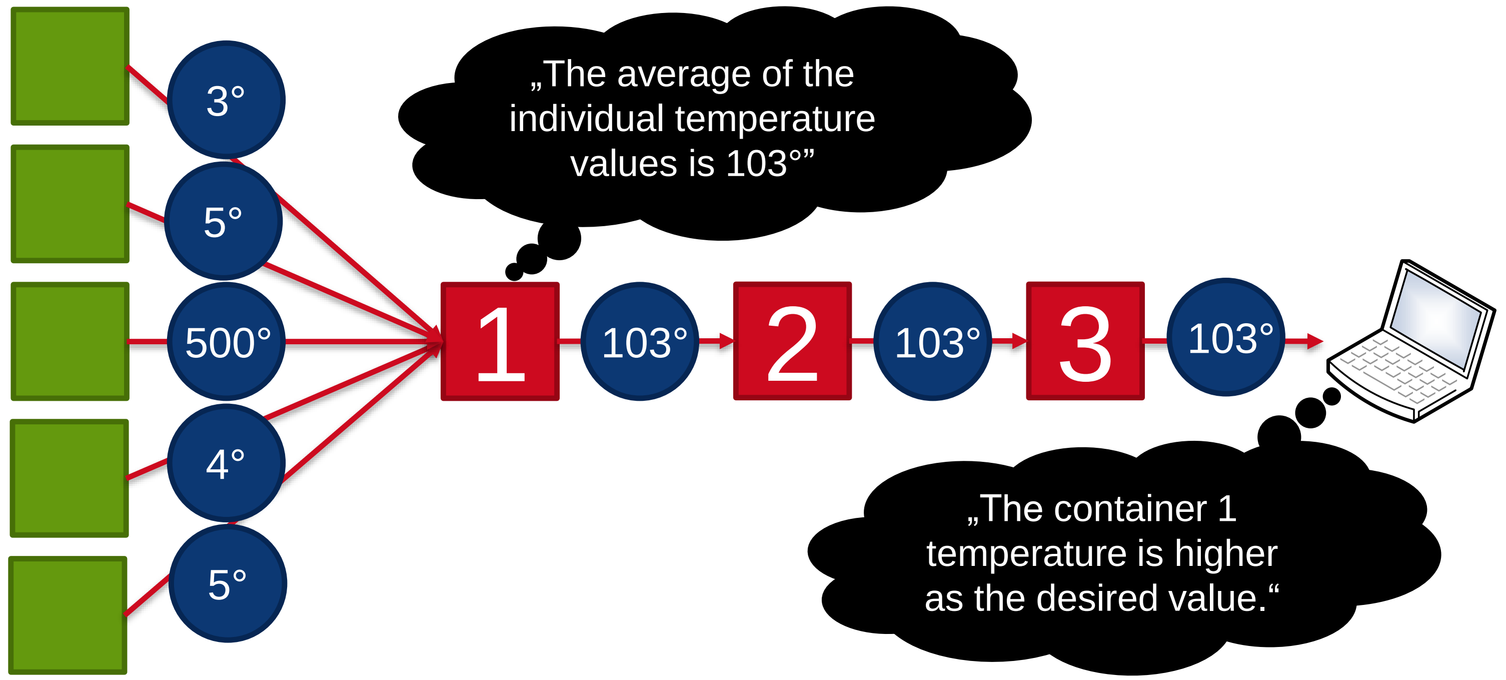
Scenario: Monitor the condition of various transported goods on a container ship.



PALUNO
The Ruhr Institute for Software Technology

Container Network Example

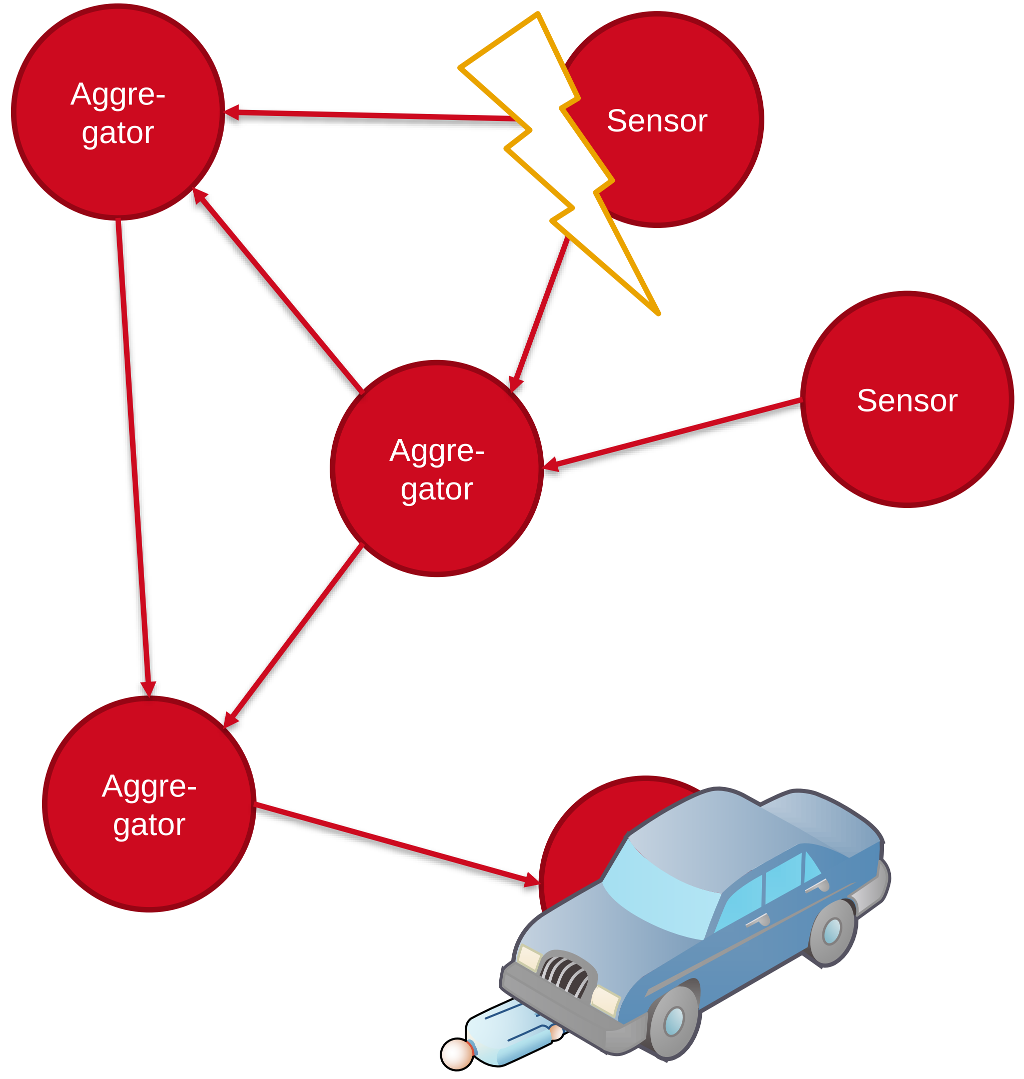
Problem Statement: It is hard to find the actual source of an error.


3°
5°
500°
4°
5°
„The average of the individual temperature values is 103°"
1
103°
2
103°
3
103°
„The container 1 temperature is higher as the desired value.“

Problem Statement

Cascading Data Corruption

- For any reason, data in a node of the network has been corrupted. This means, that there is faulty information spread between nodes.
- Faulty data spreads in aggregated form in the network without the knowledge of any maintainer.
- At some point of the network, the faulty information become visible / are noticed. The point of notice is often not the source of the error!
- Troubleshooting includes tracking to the source of error and awareness about the impact of the error
- It should also be noted that a source of error may have further effects, which have not yet occurred



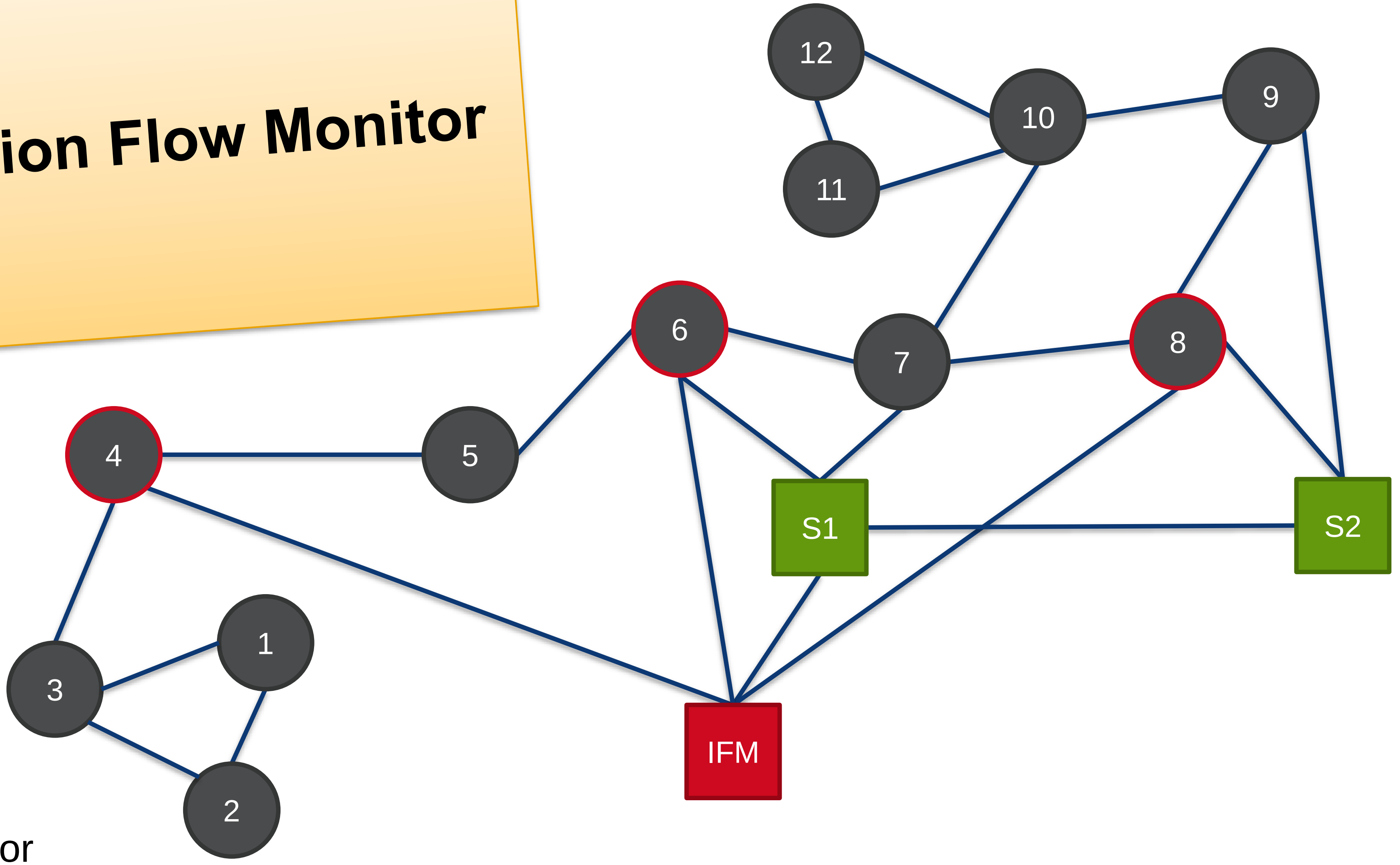
Two main problems to solve

- Understand cause and effect
  - CPSs are highly distributed systems that exchange and aggregate messages across multiple nodes
  - It is often not clear how an action of a node is achieved
  - decisions made by an CPS must be justifiable
  - If the dependencies of a decision are not comprehensible, reasons for a decision of a node cannot be determined

- Track errors to their source
  - Since it is not always clear how some actions are made, it is also not possible to determine which information is responsible for the error in the event of a fault
  - If erroneous information is inserted into aggregations, the cause cannot be determined at other nodes
  - There is no simple way of finding out which input originally caused the error at which node.

CPS-IFM: **Information Flow Monitor**

Simplified Example



●	Node
●	Spy Node
🟩	Server
🟥	Information Flow Monitor

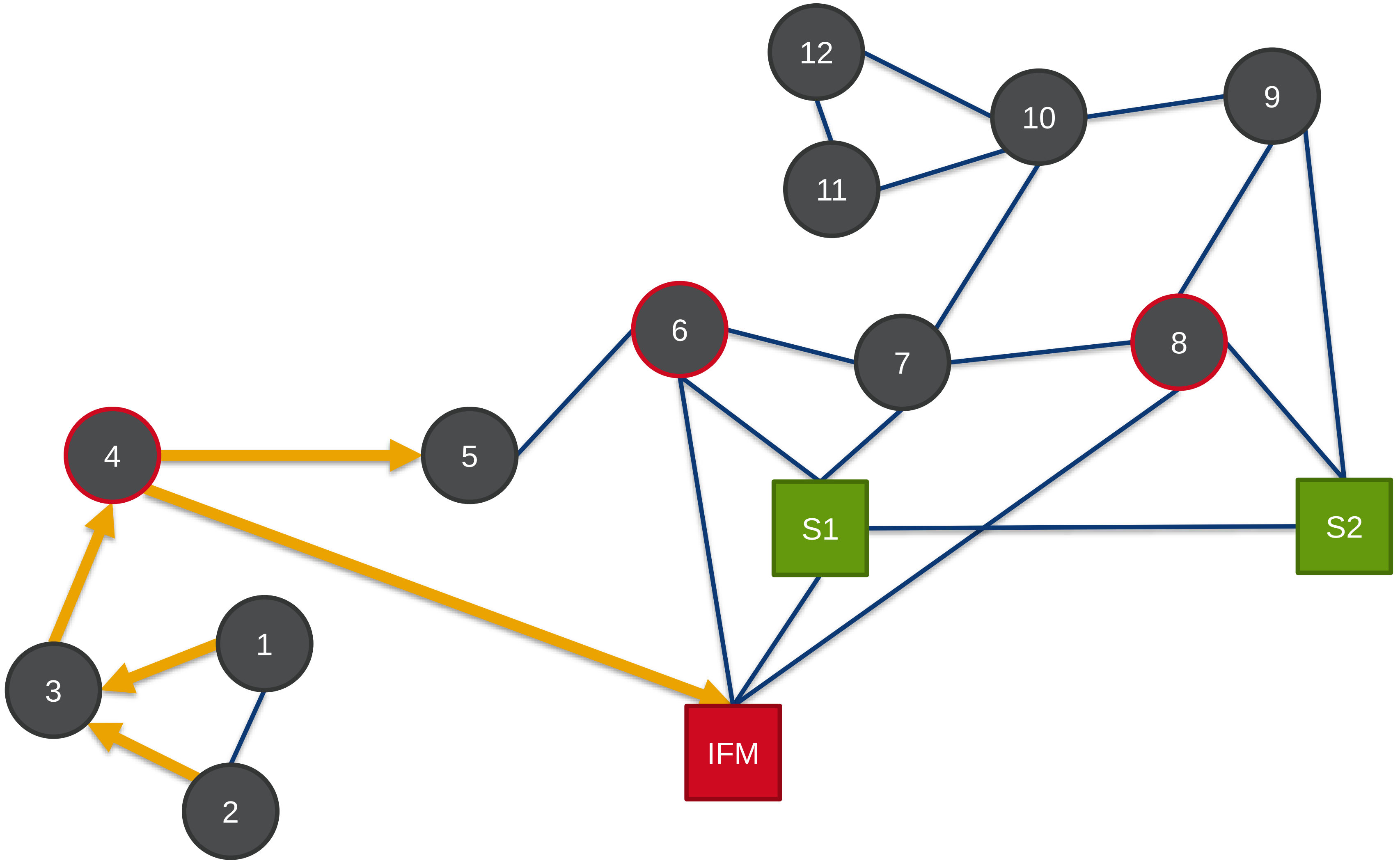
1. ❶ measures a sensor value and sends it to ❸. Same does ❷.

2. ❸ receives sensor data from ❶ and ❷.

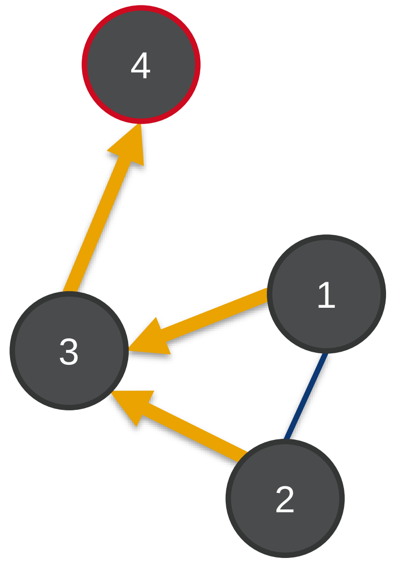
3. ❸ aggregates these data and sends the result to ❹.

4. ❹ receives those aggregated data and knows their original source as well, by the help of the IFM protocol.

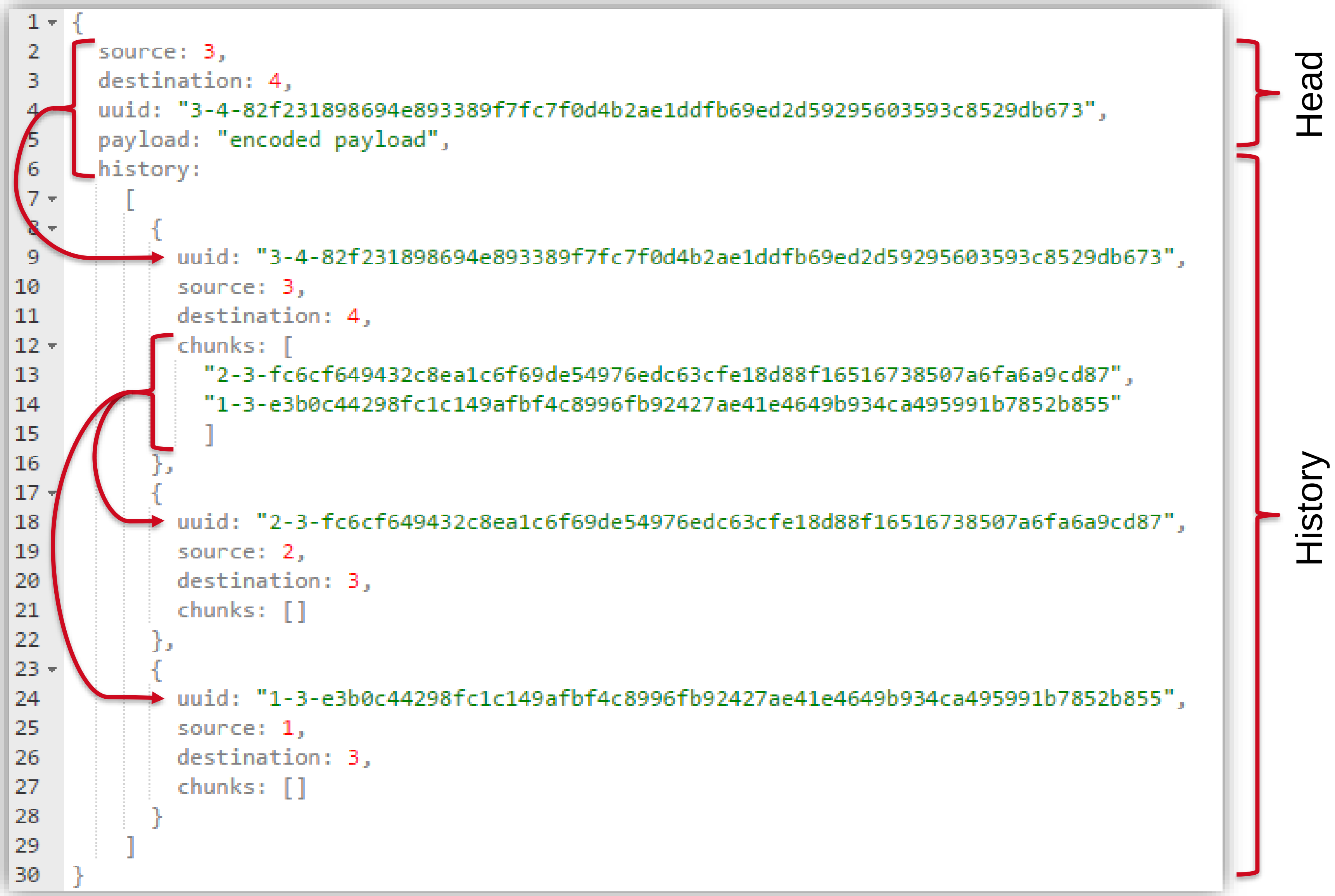
5. ❹ wants to edit those data and sends them to ❺. Because it is a spy node as well, it passes the history information to the IFM node.



IFM Protocol

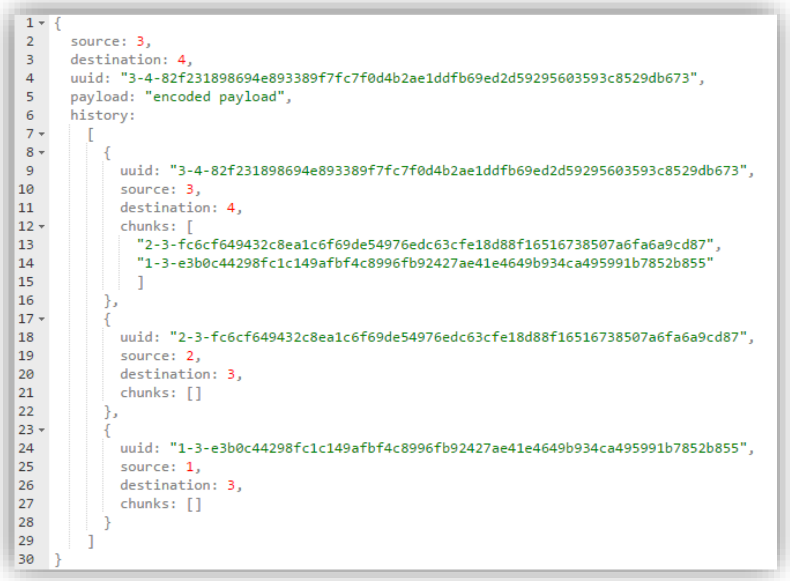


- ❸ sends data to ❹
- Information on the data from which the sent information was aggregated is contained as well

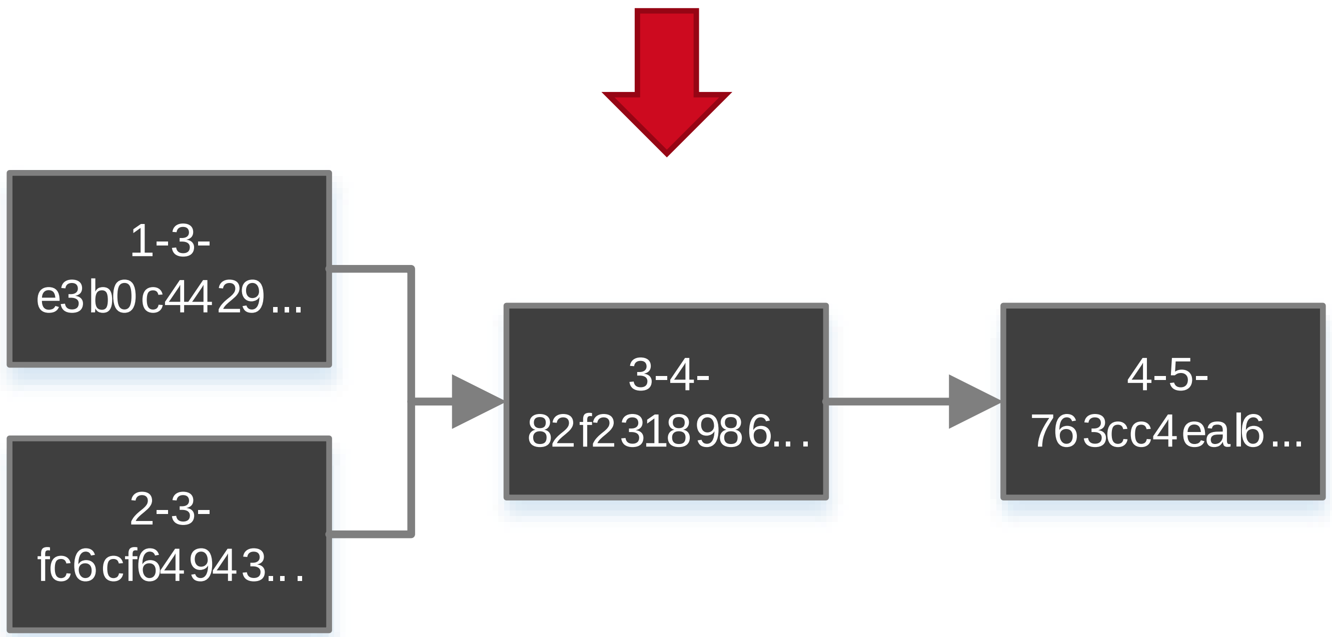


```
{
  source: 3,
  destination: 4,
  uuid: "3-4-82f231898694e893389f7fc7f0d4b2ae1ddfb69ed2d59295603593c8529db673",
  payload: "encoded payload",
  history:
    [
      {
        uuid: "3-4-82f231898694e893389f7fc7f0d4b2ae1ddfb69ed2d59295603593c8529db673",
        source: 3,
        destination: 4,
        chunks: [
          "2-3-fc6cf649432c8ea1c6f69de54976edc63cfe18d88f16516738507a6fa6a9cd87",
          "1-3-e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
          ]
      },
      {
        uuid: "2-3-fc6cf649432c8ea1c6f69de54976edc63cfe18d88f16516738507a6fa6a9cd87",
        source: 2,
        destination: 3,
        chunks: []
      },
      {
        uuid: "1-3-e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        source: 1,
        destination: 3,
        chunks: []
      }
    ]
}
```

IFM Protocol



```
{
  source: 3,
  destination: 4,
  uuid: "3-4-82f231898694e893389f7fc7f0d4b2ae1ddfb69ed2d59295603593c8529db673",
  payload: "encoded payload",
  history:
    [
      {
        uuid: "3-4-82f231898694e893389f7fc7f0d4b2ae1ddfb69ed2d59295603593c8529db673",
        source: 3,
        destination: 4,
        chunks: [
          "2-3-fc6cf649432c8ea1c6f69de54976edc63cfe18d88f16516738507a6fa6a9cd87",
          "1-3-e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855"
          ]
      },
      {
        uuid: "2-3-fc6cf649432c8ea1c6f69de54976edc63cfe18d88f16516738507a6fa6a9cd87",
        source: 2,
        destination: 3,
        chunks: []
      },
      {
        uuid: "1-3-e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        source: 1,
        destination: 3,
        chunks: []
      }
    ]
}
```



Trustful Information Flow Monitoring

- It has to be saved who passed on which data. Later it can be determined at which point information was corrupted.
- The parties do not have to trust each other. If a party lies, this lie has to be documented and persisted.
- Saved information should support non-repudiability. So everyone has to take responsibility for their data.

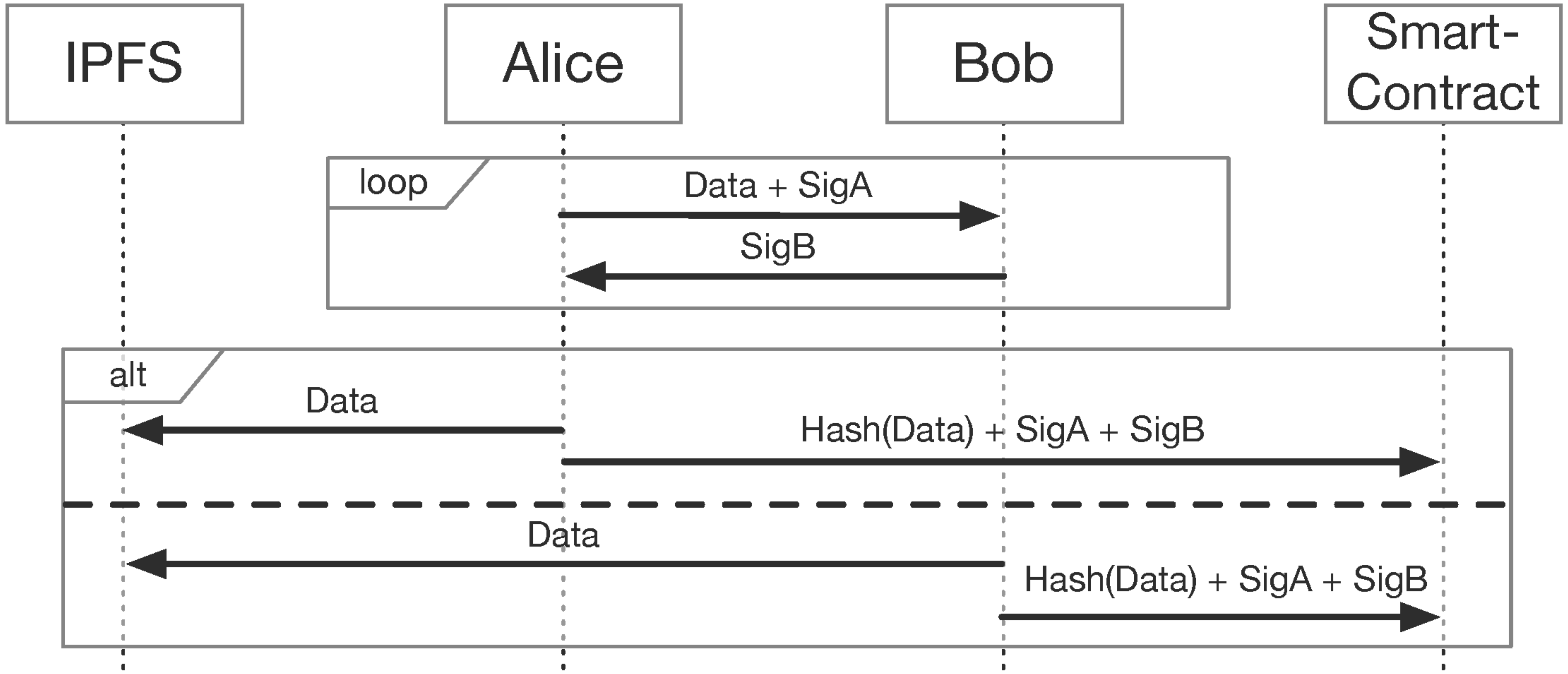
This leads to some technical requirements:

- History information should become unalterable once they are released for storage by a spy node
- Stored information should be verifiable
- There should be no central instances to trust

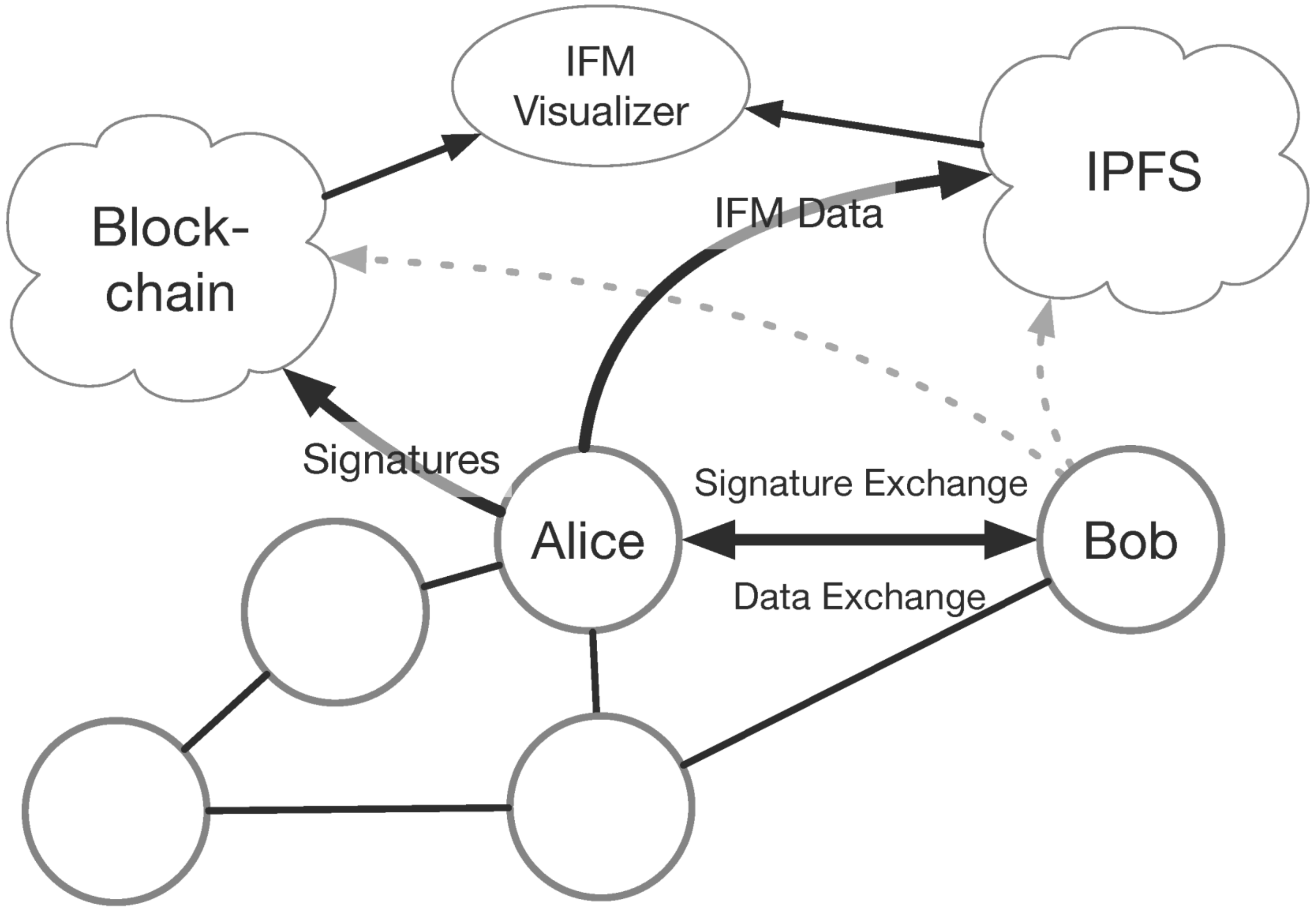
Trustful Information Flow Monitoring

1. Alice, wants to share data with Bob, who is a member of a different company. Alice and Bob do not trust each other. Therefore, the data to transfer is digitally signed by Alice and then transferred from Alice to Bob.
2. Bob can now use the received signature to prove that this data both originates from Alice and has not been altered.
3. Bob signs the data in turn and sends the signature back to Alice. Alice is now able to prove that the data has been correctly transferred to Bob.
4. Alice and Bob now are both able to prove that the file transfer has been successful and which data has been transmitted.
5. Either Alice or Bob stores the collected history information in the blockchain.
6. The blockchain can automatically check the signatures by a smart contract.
7. If the signatures are valid, the blockchain accepts the data and both nodes can delete the locally stored data.

Data Exchange & Storage Process



Architectural Overview



Conclusion

We presented our vision of a decentralized Information Flow Monitor where different parties can interact and share data without having to trust each other.

- The use of decentralized storage enables it to integrate multiple IFM instances that can access a common database.
- The scalability and trust of the IFM concept is thereby significantly increased, as there are no centralized instances that need to be trusted.

Thank you for your attention.

Vielen Dank!

Bildquelle: https://ec.europa.eu/digital-agenda/en/system-systems/