Distributed Algorithms for Real-Time Industrial Systems

DAVID KOZHAYA
PRINCIPAL SCIENTIST
ABB CORPORATE RESEARCH SWITZERLAND

ABB Corporate Research
Switzerland

112 Employees (**83** PhDs)
24 nationalities
29 students

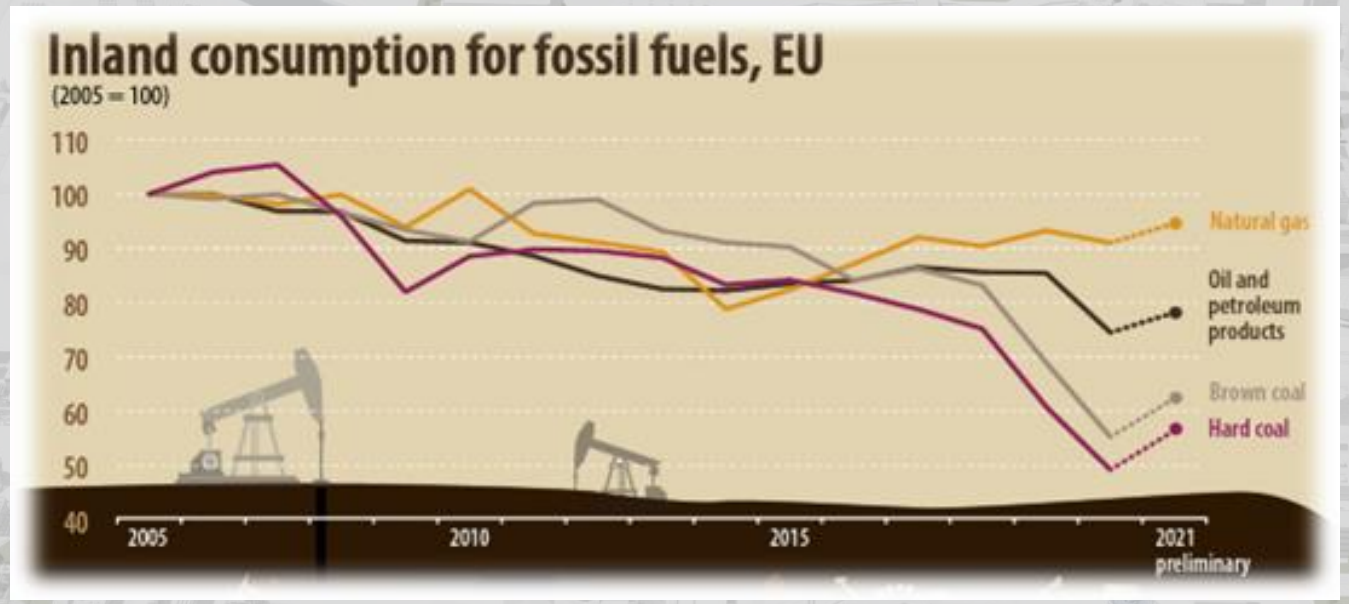
50 labs:
MV/LV Switching
Power Electronics
Sensing, Materials, Analytics

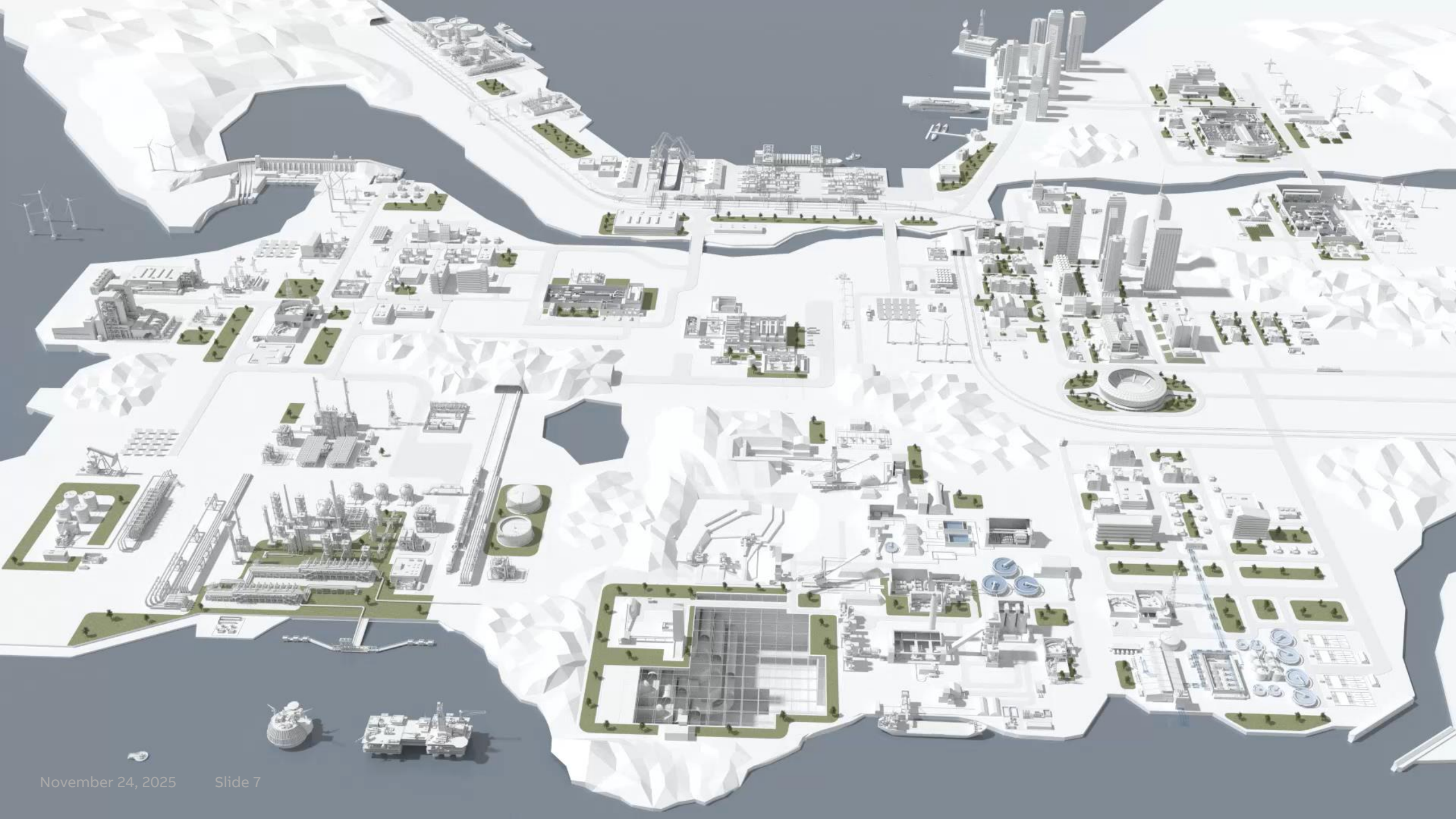
slido

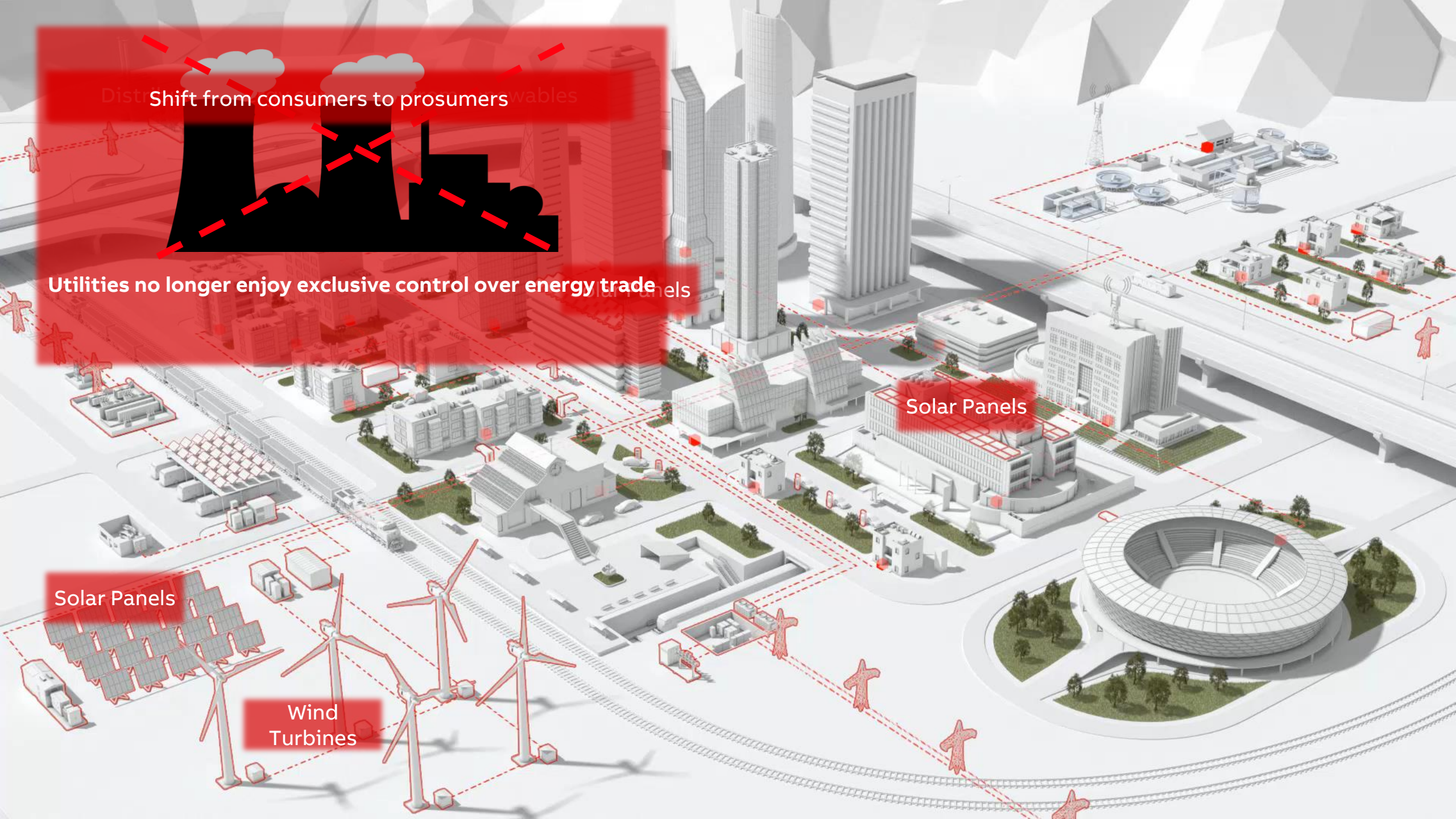
What are some examples of distributed systems you know about?

ⓘ Start presenting to display the poll results on this slide.

Why are distributed problems relevant to automation industry?

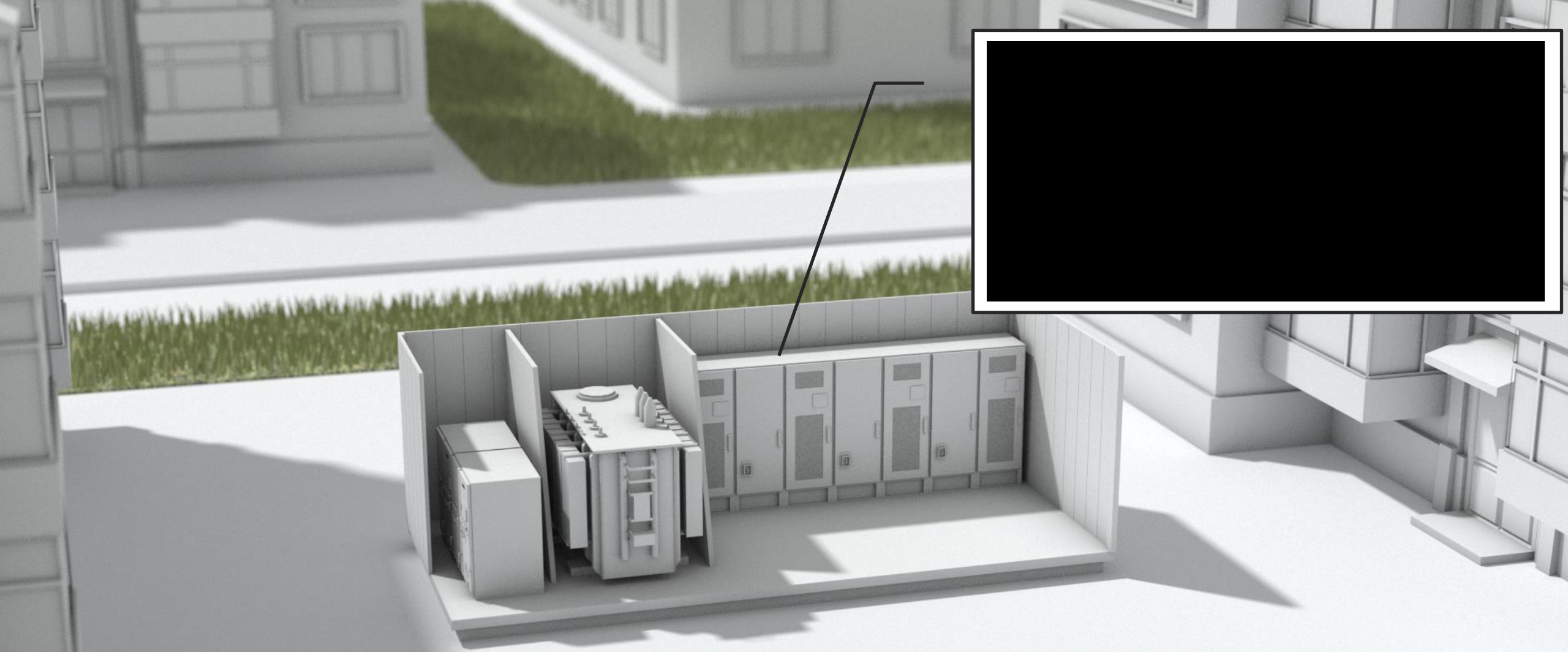

Inland consumption for fossil fuels, EU
(2005 = 100)
110
100
90
80
70
60
50
40
2005
2010
2015
2021
preliminary
Natural gas
Oil and petroleum products
Brown coal
Hard coal



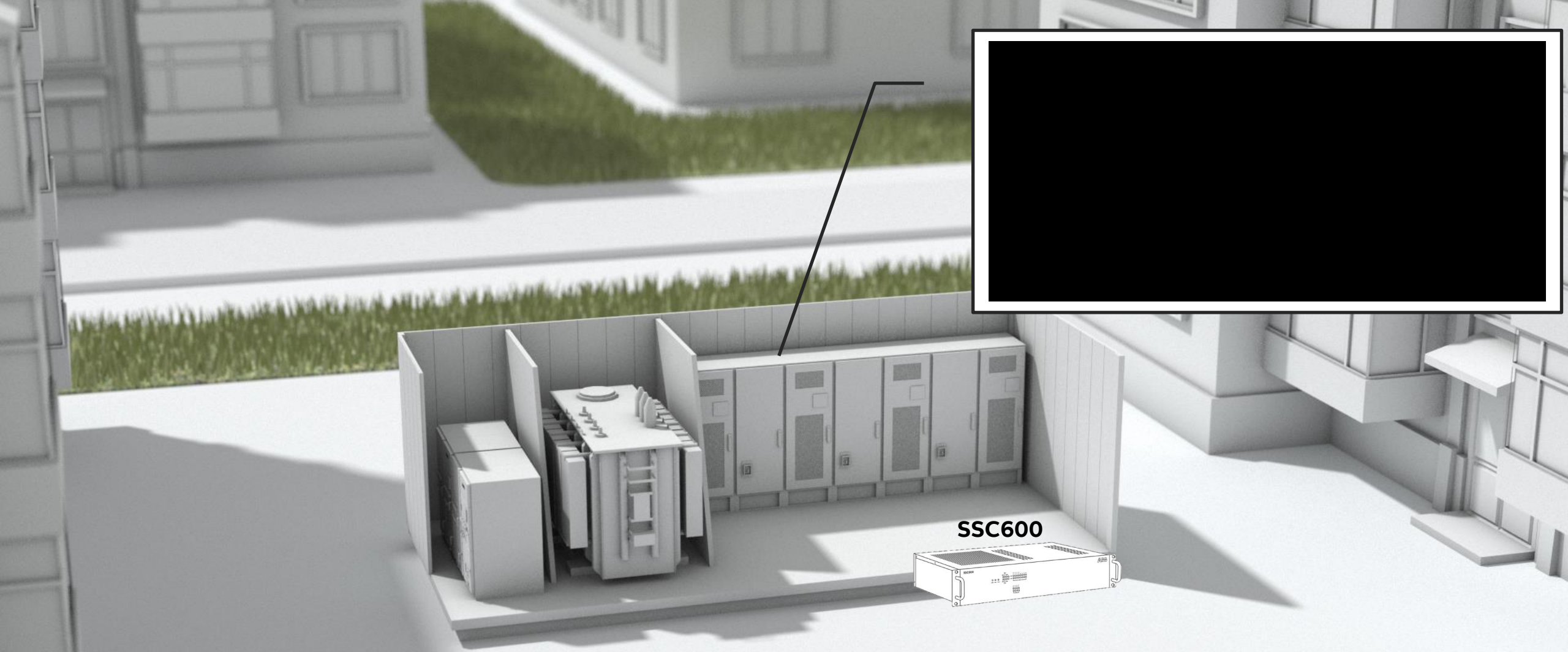

Shift from consumers to prosumers
Utilities no longer enjoy exclusive control over energy trade
Solar Panels
Solar Panels
Wind Turbines


In 2023, > 50% of net electricity in the EU came from renewable sources


Peer-to-peer energy markets
Free energy trade between prosumers



A **substation** transforms electrical voltage between transmission and distribution levels. Its main function is to **switch, protect, and control** power flow to ensure safe and reliable electricity delivery.



A **substation** transforms electrical voltage between transmission and distribution levels. Its main function is to **switch, protect, and control** power flow to ensure safe and reliable electricity delivery.

Importance of Distributed Systems to ABB

Distributed Systems Projects at ABB

Energy Trade/Redundant Protection

Description

- Determine energy production and consumption of all users
- Allow P2P trusted energy transactions regulated by dynamic real-time pricing
- Real-time reliable state information dissemination to multiple protection instances

Distributed problems relevant to system

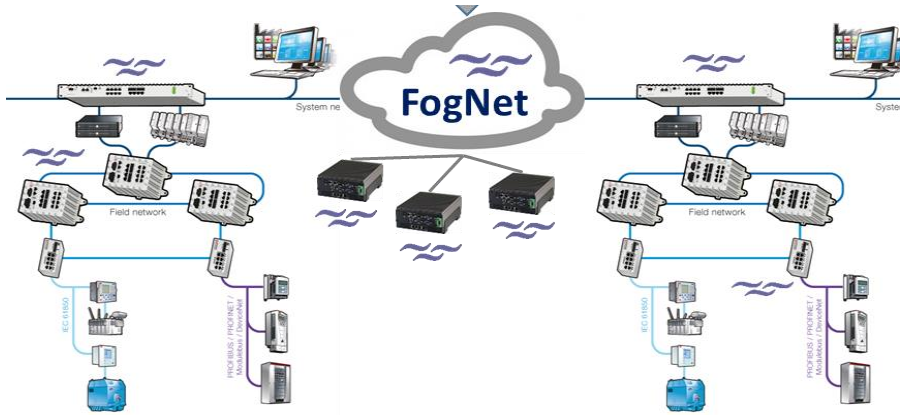
- Reliable Broadcast
- Consensus
- Distributed ledger

david.kozhaya@ch.abb.com

Importance of Distributed Systems to ABB

Distributed Systems Projects at ABB

Edge and Fog Computing



Description

- Run computations as close as possible to the industrial site (no cloud)
- Devices are overprovisioned and have at disposition spare resources (memory, CPU, network)
- Discover the dynamic available resources and attempt to deploy additional applications (analytics) on them

Distributed problems relevant to system

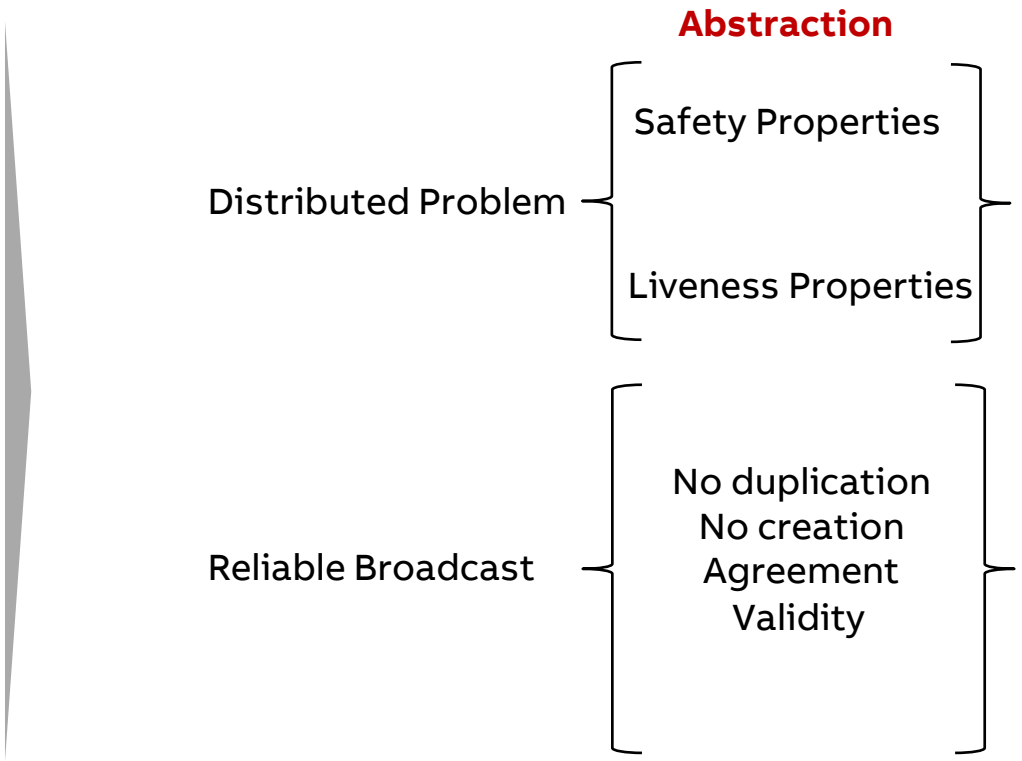
- Failure detection/ membership
- Reliable Broadcast
- Consensus

Distributed Systems in Class

Solution approach

Distributed Abstractions

1. Define problem as an abstraction ✓
2.
3.



Distributed Systems in Class

Solution approach

Distributed Abstractions

1. Define problem as an abstraction ✓
2. Implement abstraction in system model
3.

No duplication
No creation
Agreement
Validity

Reliable Broadcast Abstraction (RB)

Algorithms to implement RB in

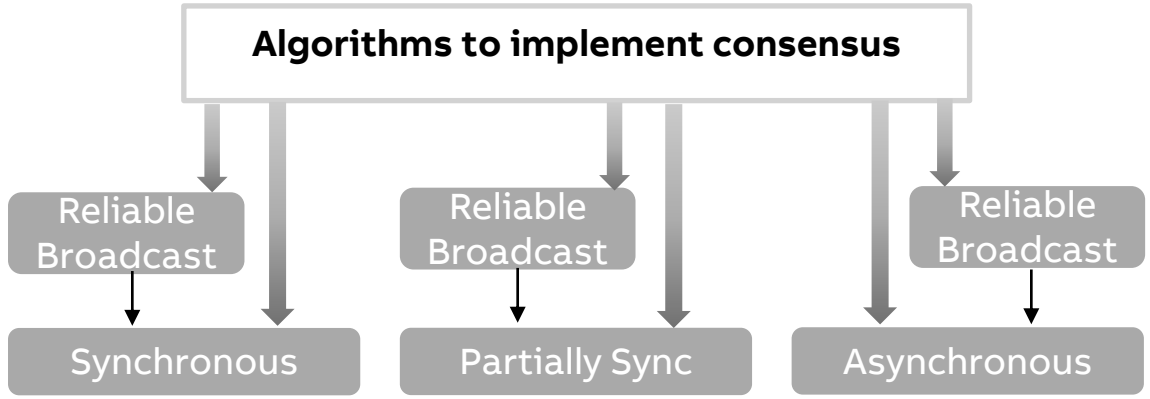
- Synchronous
- Partially Sync
- Asynchronous

Distributed Systems in Class

Solution approach

Distributed Abstractions

1. Define problem as an abstraction ✓
2. Implement abstraction in system model ✓
3. Use abstraction as building block ✓



Distributed Systems in Class

Solution approach

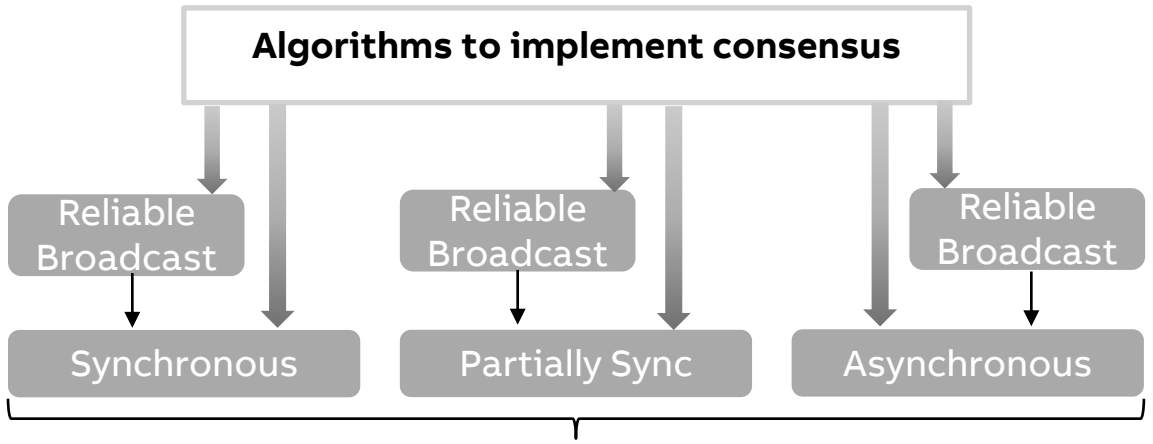
Modular Architecture

Importance:

- Easy to customize software to customer needs
- Better code maintainability
- Scale down across various types of hardware – based on needs
- …

Interesting remarks

Problems do not define deadlines



Never changing and/or never ending conditions

Industrial Systems

Differences with Distributed Systems Seen in Class

Energy Grids

Differences in Abstractions

- Real-time behavior
  - Hard timing deadlines

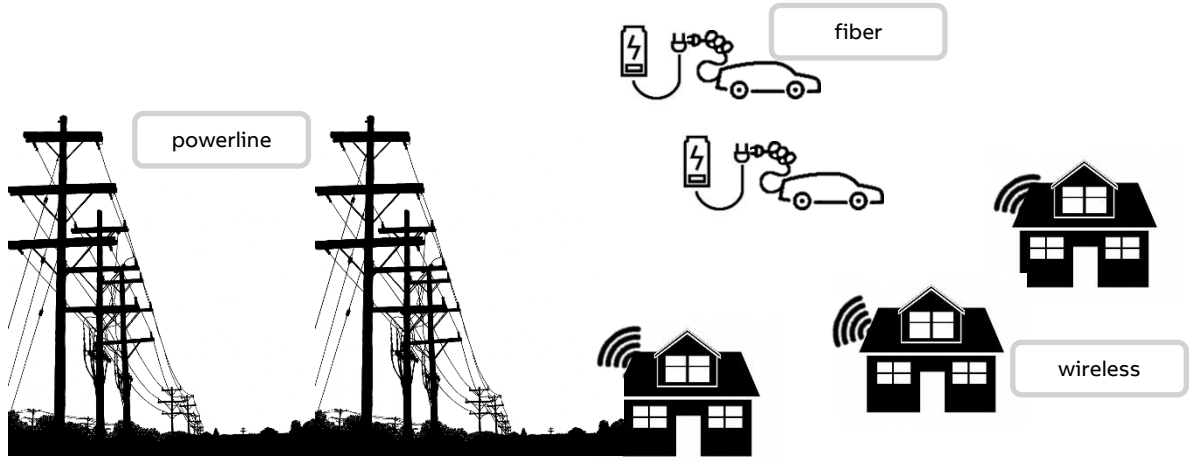
david.kozhaya@ch.abb.com

Industrial Systems

Differences with Distributed Systems Seen in Class

Energy Grids

Difference in System Models

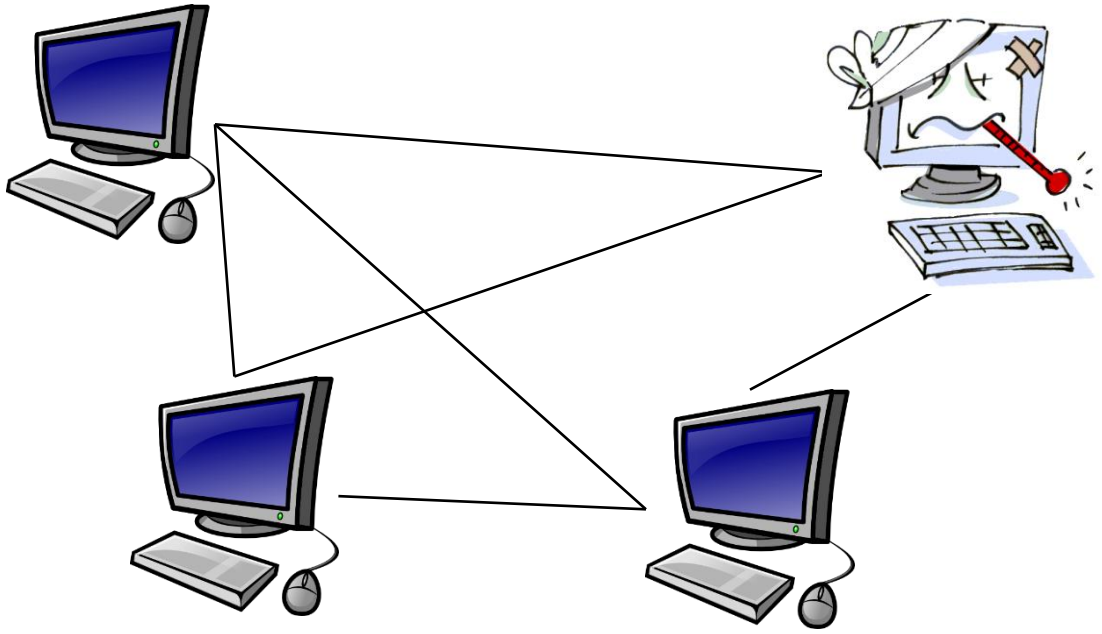


Time varying quality of communication links

- Links oscillate between synchrony and asynchrony phases

Walking Through an Example

Asynchronous Model



- Processes:
  - Synchronous
  - Some can fail
- Communication links:
  - A link is reliable but asynchronous
  - No guarantee when message get delivered

Broadcast Protocols

No duplication: No message is delivered more than once
No creation: If a process delivers message m with sender s, then s broadcast m
Agreement: if a correct process delivers m, then every correct process delivers m
Validity: If a correct broadcasts m, then some correct process eventually delivers m

Reliable Broadcast Abstraction (RB)

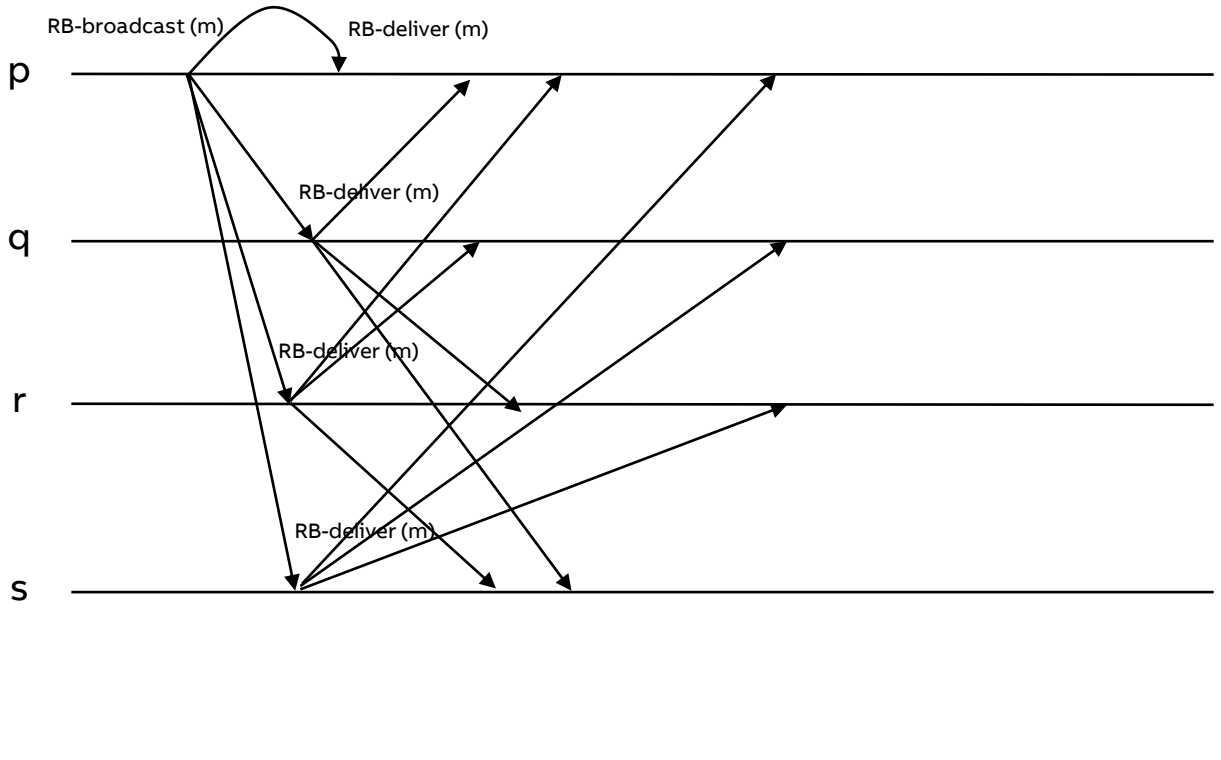
in

Asynchronous System with f<n

Reliable Broadcast (RB)

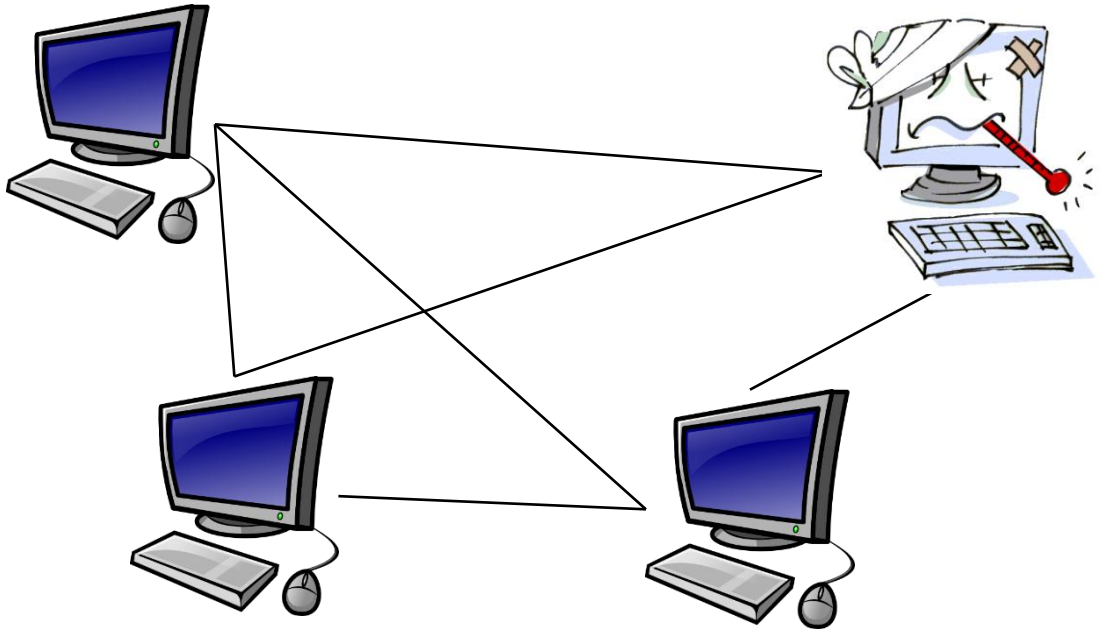
Algorithm example

```
@ process p
delivered={}

Upon event broadcast(m)
          ∀s ∈ Π :
                send(p, m) to s;


Upon event deliver(q, m)
          if (m ∉ delivered){
             RB-deliver(m);
             delivered=delivered U {m};
             if(q!=p){
                ∀s ∈ Π :
                   send(p, m) to s;
             }
          }
```



New System Model

Probabilistically Synchronous



- Processes:
  - Synchronous
  - Some can fail
- Communication links:
  - During any transmission, a link is reliable and synchronous (timely) with probability p

Broadcast Protocols

No duplication: No message is delivered more than once
No creation: If a process delivers message m with sender s, then s broadcast m
Agreement: if a correct process delivers m, then every correct process delivers m
Validity: If a correct broadcasts m, then some correct process eventually delivers m

Reliable Broadcast Abstraction (RB)

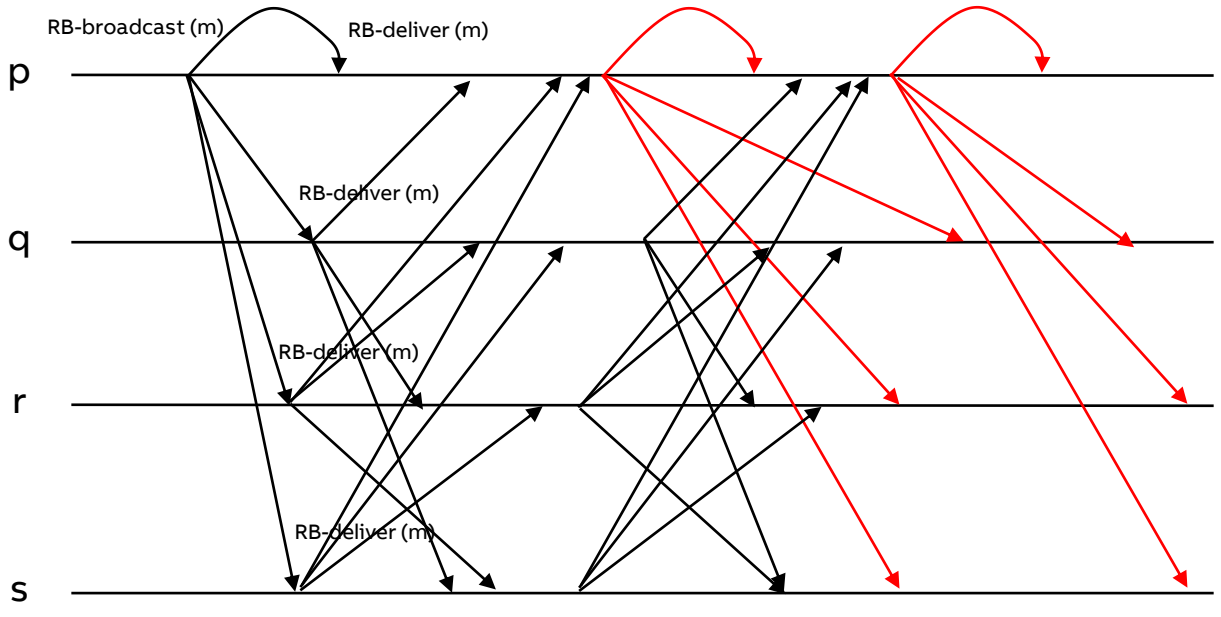
in

Probabilistically Synchronous System with f<n

Reliable Broadcast (RB)

Algorithm for Probabilistically Synchronous System

```
@ process p
delivered={};
Upon event broadcast(m)
            while (True){  wait (1 tu);
               ∀s ∈ Π :
                  send(p, m) to s;
            }
Upon event deliver(q, m)
            if (m ∉ delivered){
               RB-deliver(m);
               delivered=delivered U {m};
               while(True){
                  if(q!=p){
                     ∀s ∈ Π :
                        send(p, m) to s;
                  }
               }
            }
```



Network is congested!

Broadcast Protocols

No duplication: No message is delivered more than once
No creation: If a process delivers message m with sender s, then s broadcast m
Agreement: if a correct process delivers m, then every correct process delivers m
Validity: If a correct broadcasts m, then some correct process eventually delivers m

Reliable Broadcast Abstraction (RB)

in

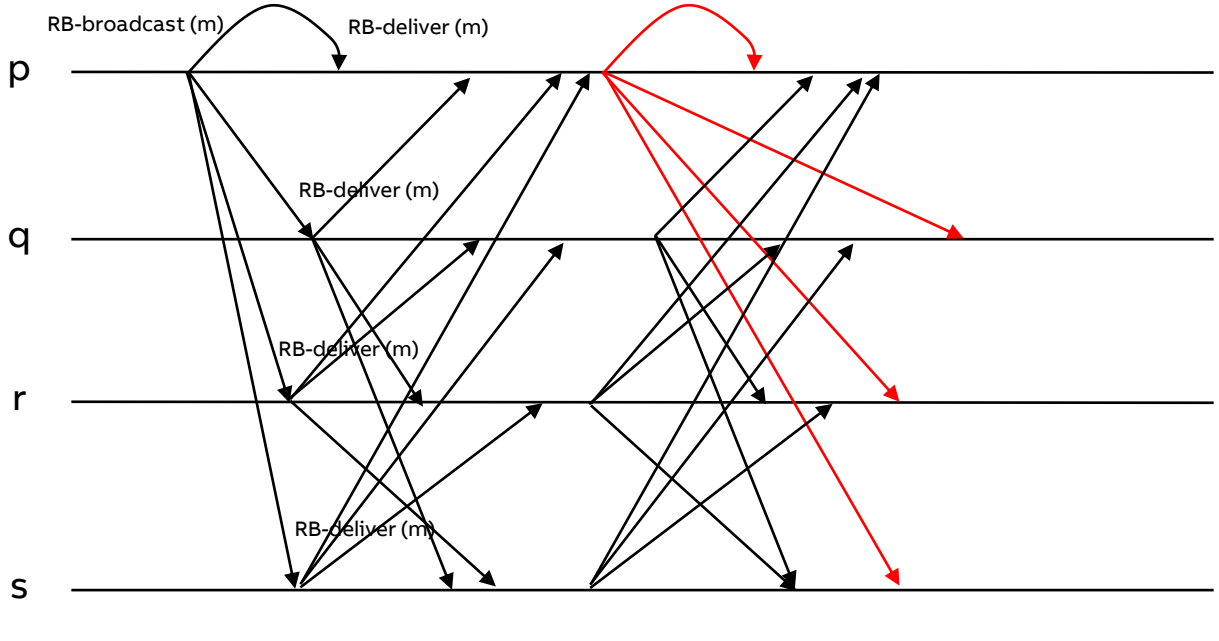
Probabilistically Synchronous System with f<n + P

Reliable Broadcast (RB)

Algorithm for Probabilistically Synchronous System with P

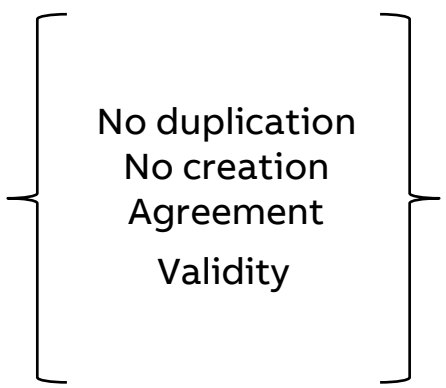
```
@ process p
delivered={}; acks={}; correct = Π;
Upon event broadcast(m)
            while (acks!=correct){
               ∀s ∈ correct :
                  send(p, m) to s;
            }
Upon event deliver(q, m) from sender a
            acks=acks U {a}
            if (m ∉ delivered){
               RB-deliver(m);
               delivered=delivered U {m};
               while(acks!=correct){
                  if(q!=p){
                     ∀s ∈ Π :
                        send(p, m) to s;
                  }
               }
            }
```

```
upon event crash(q)
            correct = correct - {q};
```

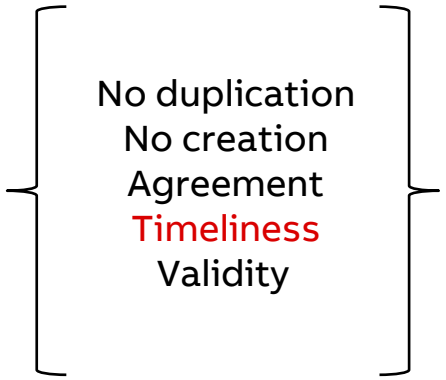


Adding Timeliness

Real-time Reliable Broadcast



Reliable Broadcast Abstraction (RB)



Real-time Reliable Broadcast Abstraction (RTRB)

Broadcast Protocols

No duplication: No message is delivered more than once
No creation: If a process delivers message m with sender s, then s broadcast m
Agreement: If a correct process delivers m, then every correct process delivers m
Timeliness: If a correct process broadcasts m at time t, then any correct process that delivers m does so by t+ Δr
Validity: If a correct broadcasts m, then some correct process eventually delivers m

Real-time Reliable Broadcast Abstraction (RTRB)

in

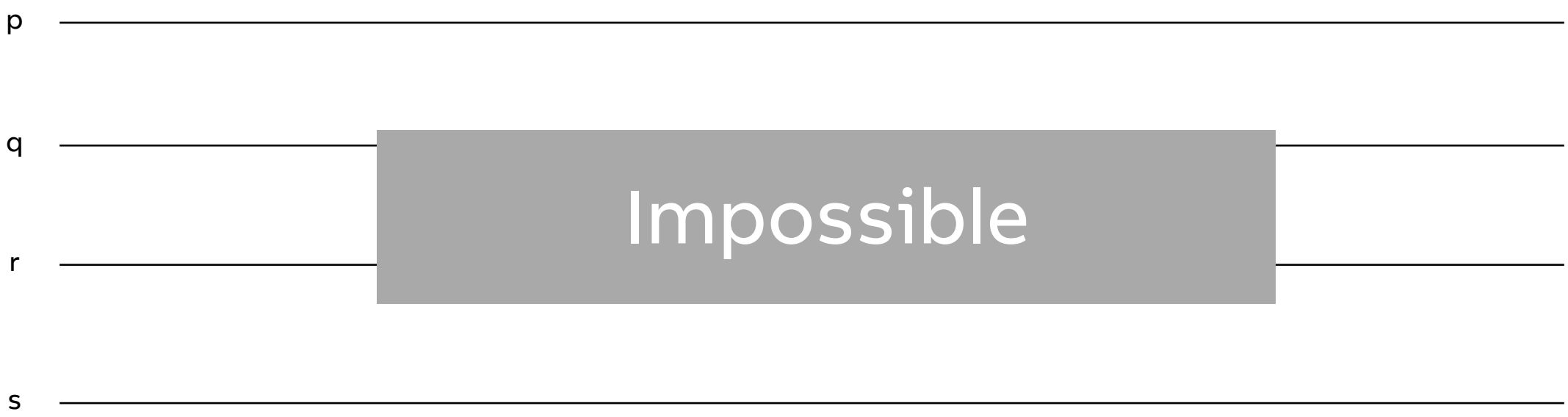
Probabilistically Synchronous System

+

P

Real-time Reliable Broadcast (RTRB)

Algorithm for Probabilistically Synchronous System with P

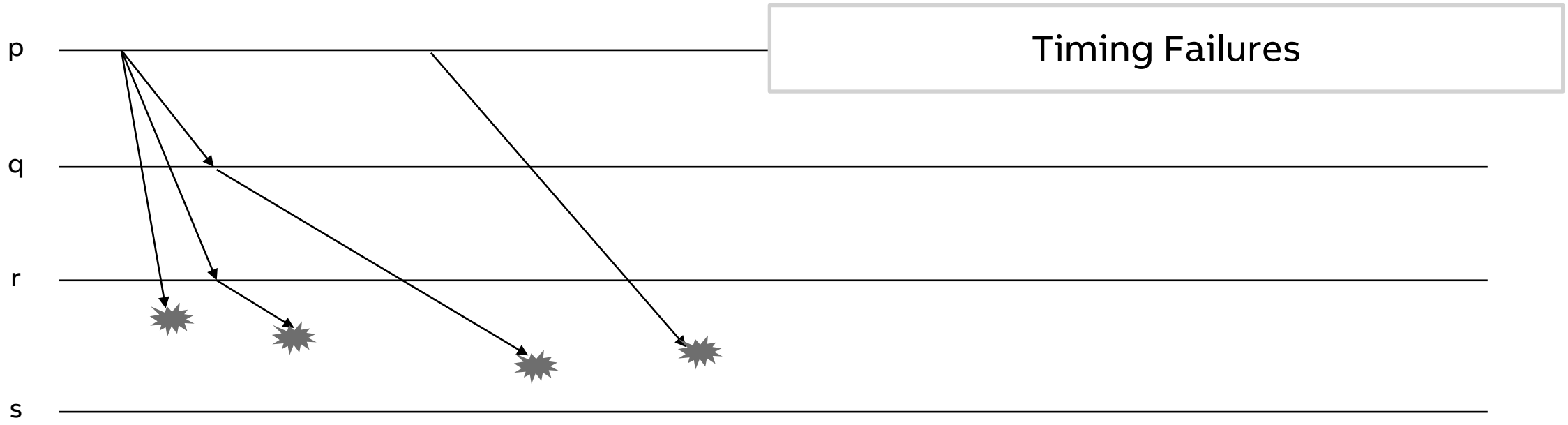


david.kozhaya@ch.abb.com

Real-time Reliable Broadcast (RTRB)

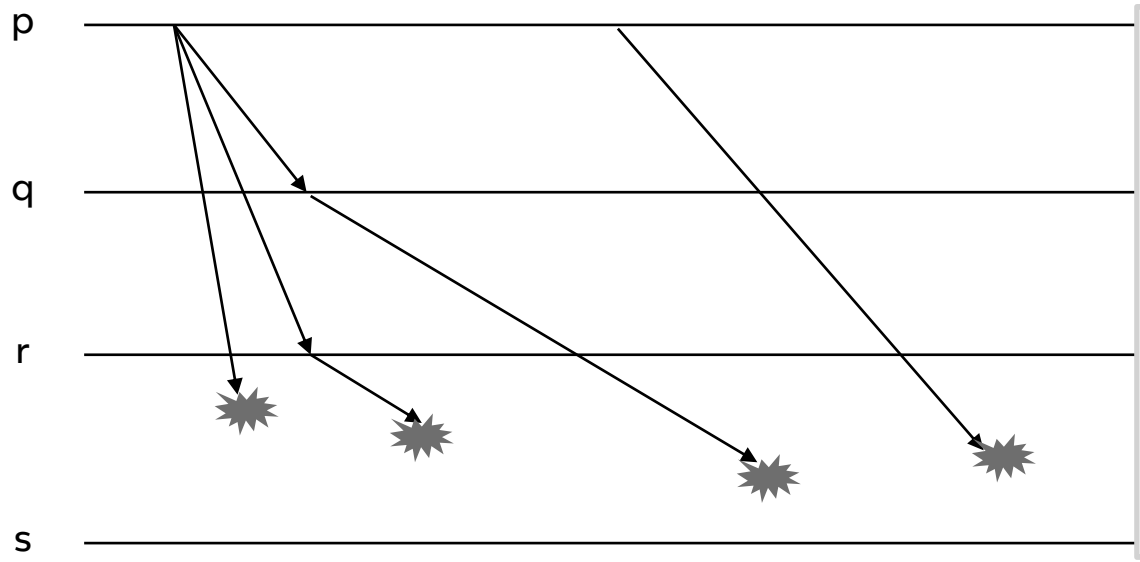
Algorithm for Probabilistically Synchronous System with P

For any predefined Δr



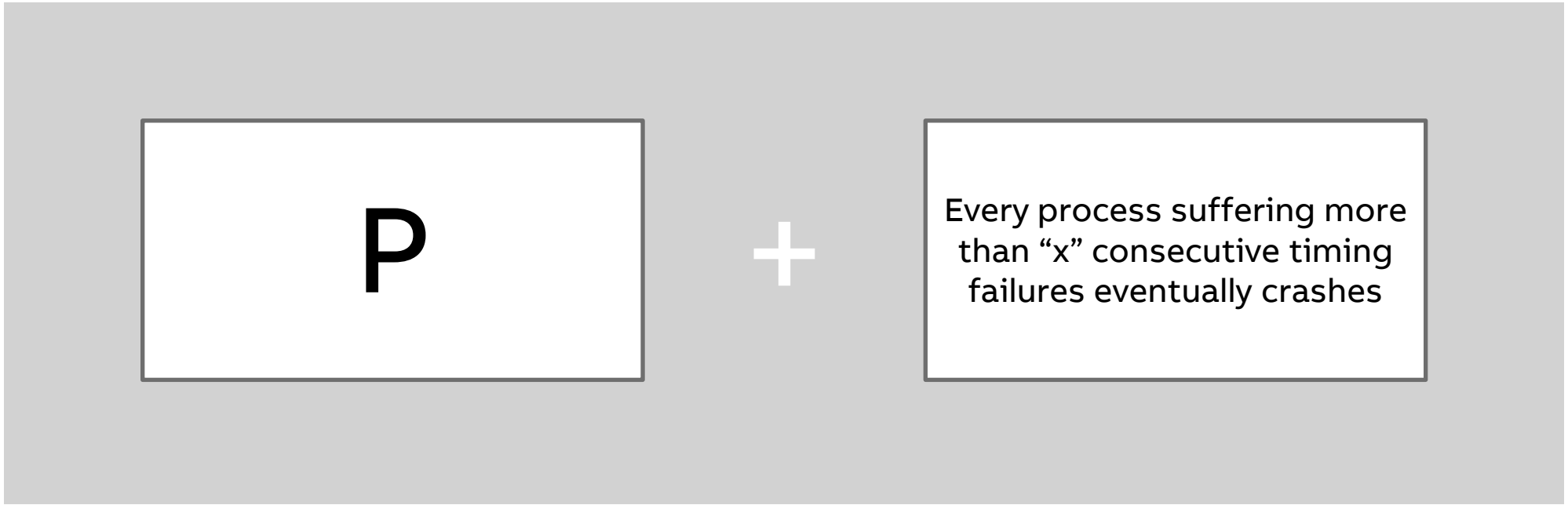
Transforming Timing Failures into Process Failures

For any predefined Δr



1. Detect Timing Failures
2. Force affected process to crash (Elastic self-aware systems)

Failure Detector TP



david.kozhaya@ch.abb.com

Process Faults Vs Timing Faults

How are they different?

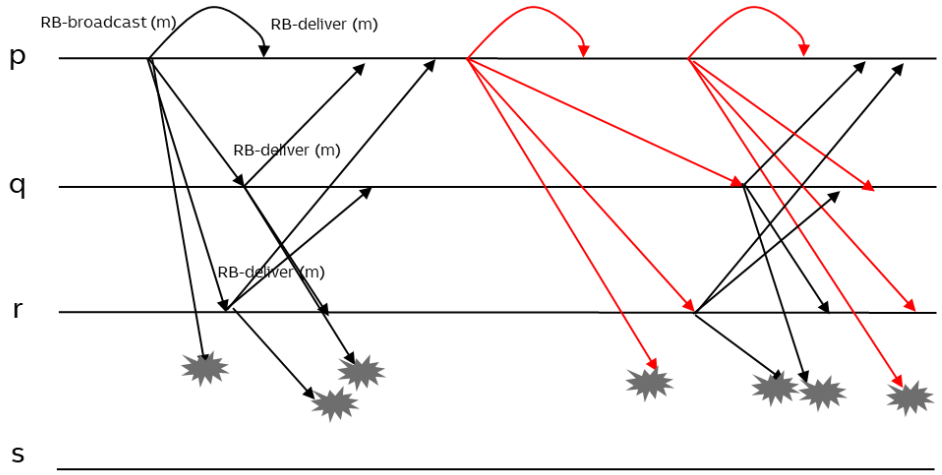
Process failures are system-wide

Timing failures are algorithm dependent

Modularity does not work in presence of timing failures

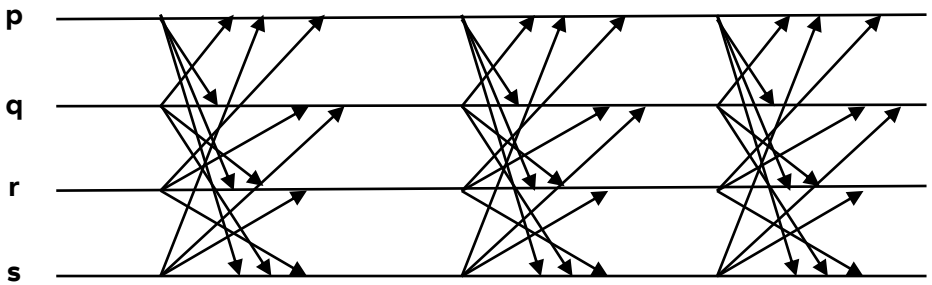
Modular Implementation

Broadcast Algorithm



Timing failures experienced

TP Algorithm



No timing failures experienced

Many Challenges and Opportunities in Industrial Distributed Systems

For Internships or Master Thesis at ABB send to david.kozhaya@ch.abb.com