Self-Organization in Autonomous Sensor/Actuator Networks [SelfOrg]

Dr.-Ing. Falko Dressler

Computer Networks and Communication Systems
Department of Computer Sciences
University of Erlangen-Nürnberg

http://www7.informatik.uni-erlangen.de/~dressler/
dressler@informatik.uni-erlangen.de

Overview

- □ **Self-Organization**
  Introduction; system management and control; principles and characteristics; natural self-organization; methods and techniques

- □ **Networking Aspects: Ad Hoc and Sensor Networks**
  Ad hoc and sensor networks; self-organization in sensor networks; evaluation criteria; medium access control; ad hoc routing; data-centric networking; clustering

- □ **Coordination and Control: Sensor and Actor Networks**
  Sensor and actor networks; communication and coordination; collaboration and task allocation

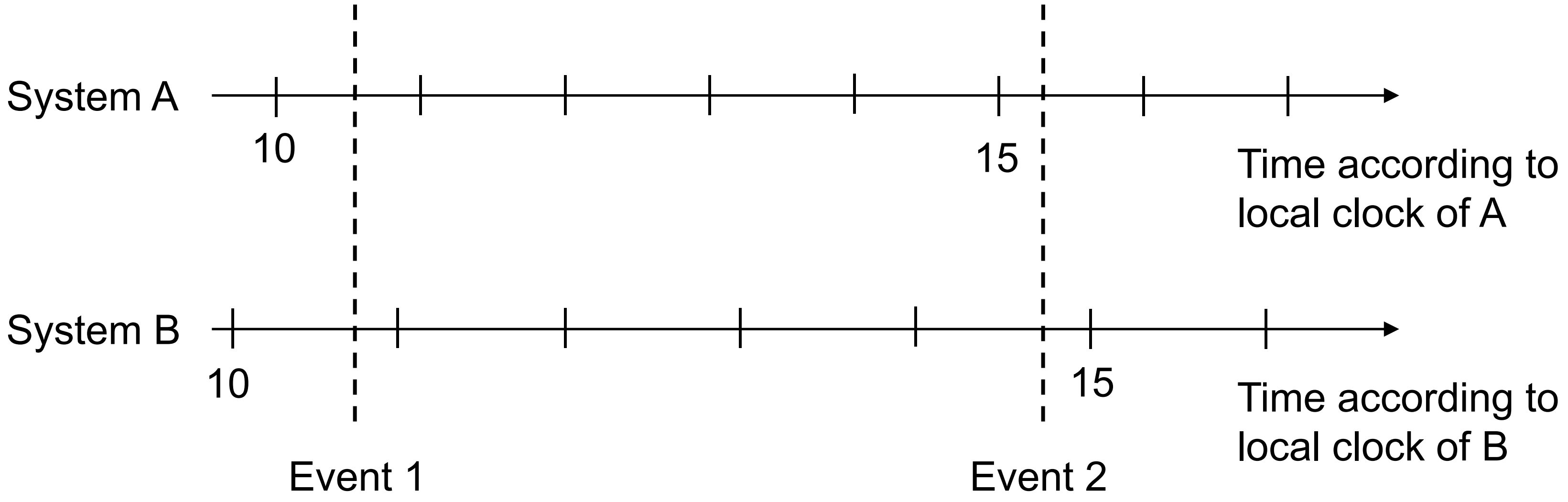
- □ **Bio-inspired Networking**
  Swarm intelligence; artificial immune system; cellular signaling pathways

Communication and Coordination

- Synchronization vs. coordination
- Time synchronization
- Distributed coordination
- In-network operation and control

Clocks Synchronization

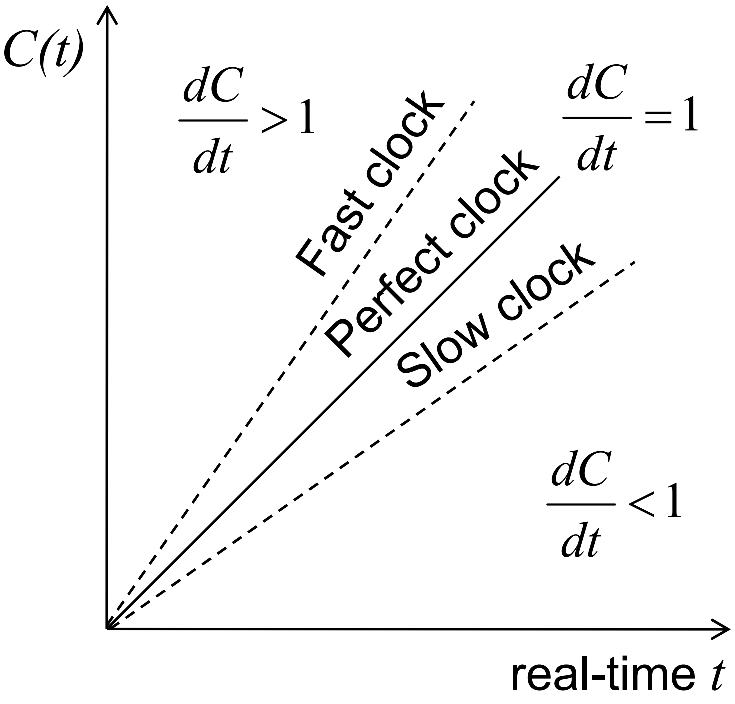
- Problem statement


System A
10
15
Time according to local clock of A
System B
10
15
Time according to local clock of B
Event 1
Event 2

- Differentiation
  - ***Absolute time*** – synchronization to a given globally unique clock source
  - ***Relative time*** – measured time difference between observable events

Synchronization in Distributed Systems

- ❑ The problem: clock drift
  - ❑ Maximum clock drift *ρ* is known and specified by the manufacturer
  - ❑ Clock drift: $1-\rho \le \frac{dC}{dt} \le 1+\rho$



Logical Clocks

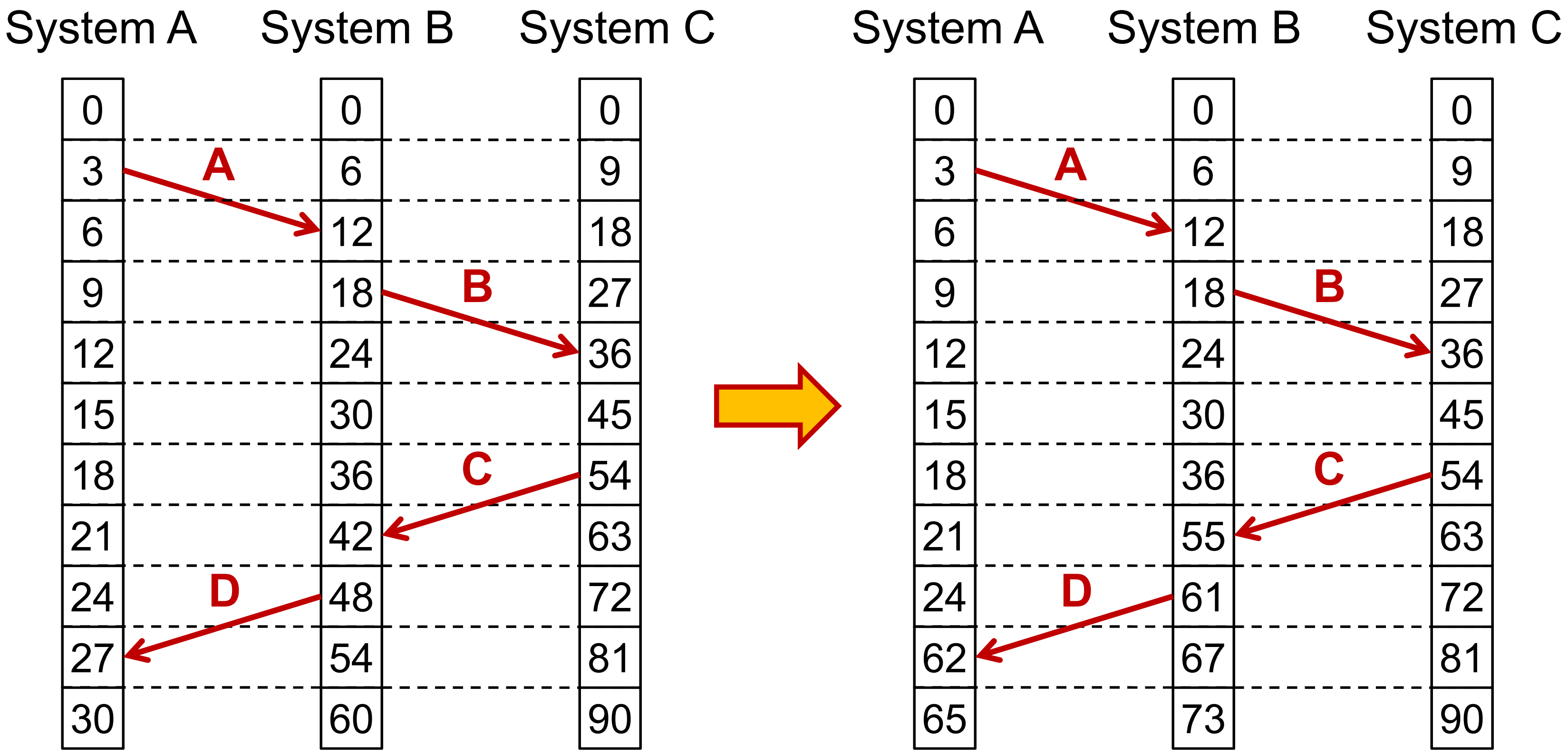
- Mostly, only the internal consistency of the clocks matters

  → **logical clocks**

  In a classic paper, Lamport (1978) showed that although clock synchronization is possible, it need not be absolute. If two processes do not interact, it is not necessary that their clocks be synchronized. Furthermore, he pointed out that what usually matters is not that all processes agree on exactly what time it is, but rather that they agree on the order in which events occur.

Lamport Timestamps

- ❑ Relation happens-before: a→b is read “a happens before b” and means that all processes agree that first event a occurs and than afterward, event b occurs



System A	System B	System C
0	0	0
3	6	9
6	12	18
9	18	27
12	24	36
15	30	45
18	36	54
21	42	63
24	48	72
27	54	81
30	60	90

System A	System B	System C
0	0	0
3	6	9
6	12	18
9	18	27
12	24	36
15	30	45
18	36	54
21	55	63
24	61	72
62	67	81
65	73	90

Lamport Timestamps

- Formal description of Lamport's timestamps
  - For all events a assign time value $C(a)$ to event a
  - Time values must have the property that if $a \rightarrow b$, then $C(a)<C(b)$

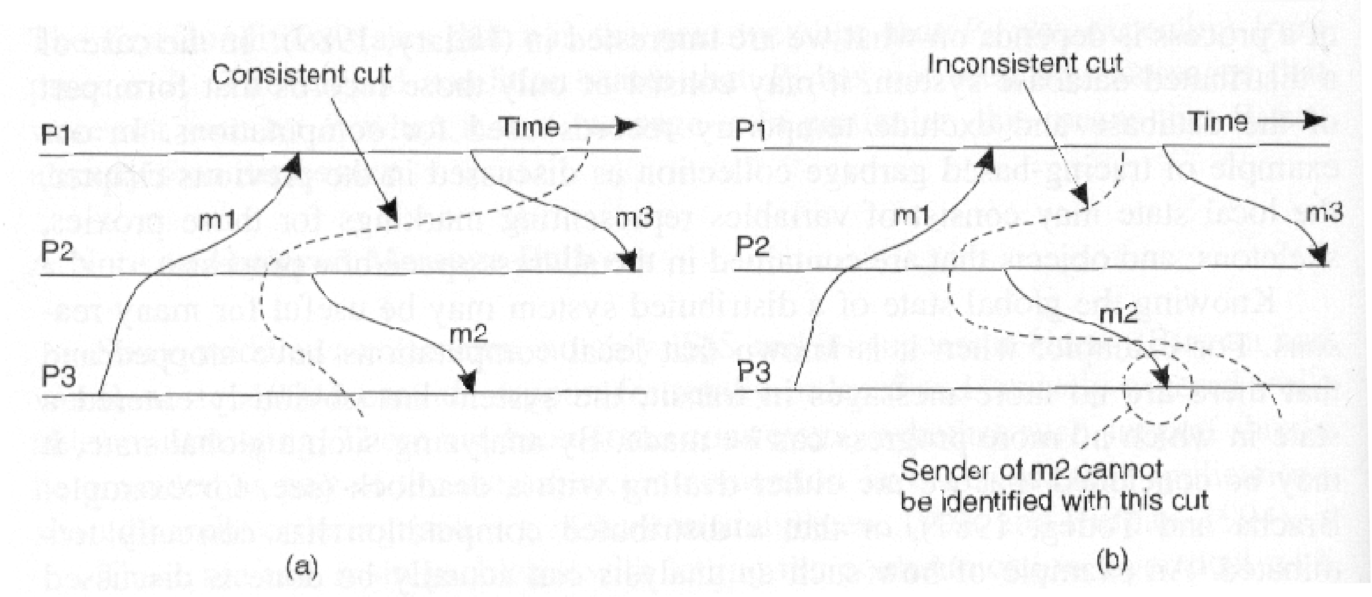
  - If a happens before b in the same process, $C(a)<C(b)$
  - If a and b represent the sending and receiving of a message, respectively, $C(a)<C(b)$
  - For all distinctive events a and b, $C(a) \neq C(b)$

- More information on clock synchronization and logical clocks

  → distributed systems

Global State

- Global state = local state of each process + messages currently in transmit (not yet delivered)
- Distributed snapshot (Chandy and Lamport)



- Application in distributed systems
  - e.g. termination detection

Coordination

- Weak synchronization
- Based on logical clocks and/or distributed snapshots
- Only the order of events becomes necessary
  - Except coordination issues in real-time systems
    → current research issue
- Prevention of global state information

- Coordination
  - Only between directly involved processes / systems
  - Sometimes using a coordinator (→ clustering)

- Application in
  - Autonomous sensor/actuator networks (→ see communication protocols in sensor networks)

Coordination vs. Synchronization

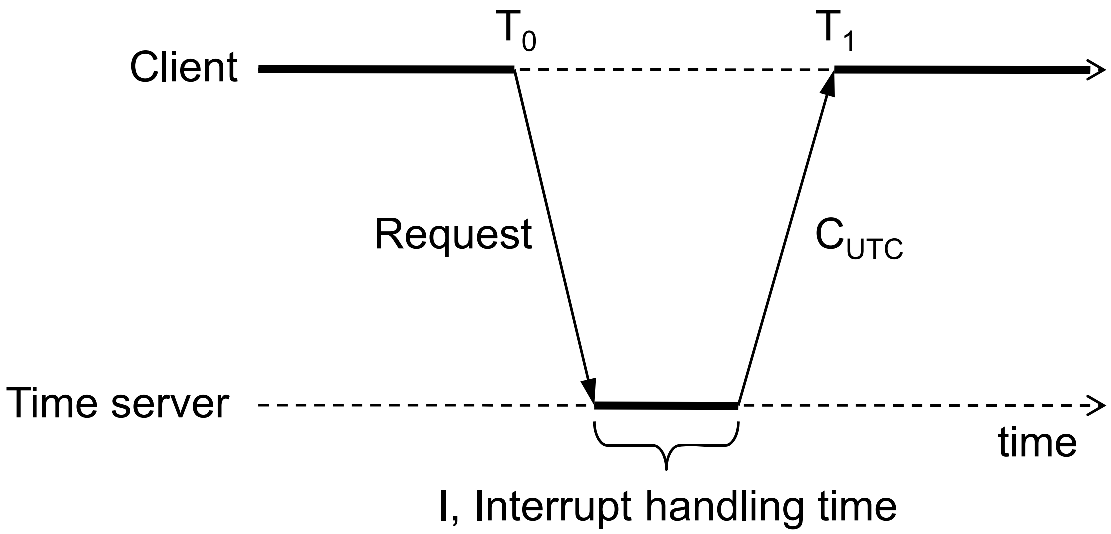
- Synchronization
  - Accurate synchronization to a given clock source, or
  - Agreement on a common (average) time
  - Pros: synchronized clocks are easy to use, provide capabilities for many distributed applications
  - Cons: message overhead (→ we tries to reduce the (global) state information in autonomous sensor/actuator networks), imprecise synchronization in large scale networks / in low bandwidth networks (→ inadequate for sensor networks)
- Coordination
  - Based on logical clocks and/or deterministic events
  - Agreement on the **order** of events (past **and** future)
  - Pros: usually low communication overhead, applicable in large-scale networks (→ scalability)
  - Cons: distributed snapshots (= global state) is hard to acquire, contradiction to energy-aware operation **or** quality of service requirements

Time Synchronization

- Characterization and requirements
  - *Precision* – either the dispersion among a group of peers, or maximum error with respect to an external standard
  - *Lifetime* – which can range from persistent synchronization that lasts as long as the network operates, to nearly instantaneous (useful, for example, if nodes want to compare the detection time of a single event)
  - *Scope and Availability* – the geographic span of nodes that are synchronized, and completeness of coverage within that region.
  - *Efficiency* – the time and energy expenditures needed to achieve synchronization.
  - *Cost and Form Factor* – which can become particularly important in wireless sensor networks that involve thousands of tiny, disposable sensor nodes.

Conventional approaches

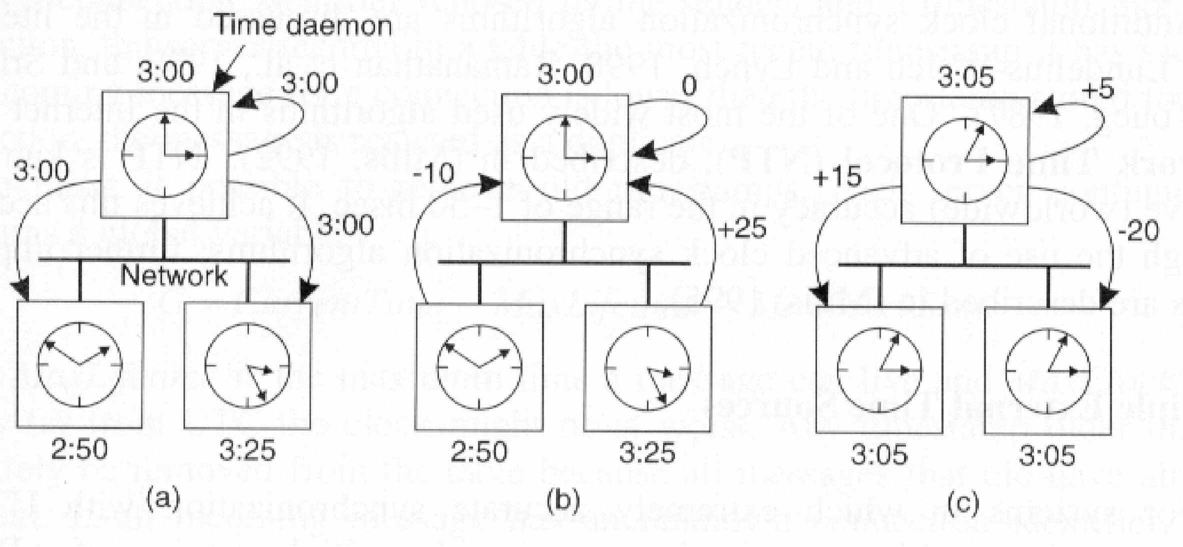
- ❑ Cristian's Algorithm



- ❑ First approximation: client sets its clock to C_{UTC}
- ❑ Major problem: the time must never run backward → gradual slowing down / advancing the clock, e.g. 1ms per 10ms
- ❑ Minor problem: transmission latency is nonzero → measurement of the transmission time: approx. $(T_1-T_0-I)/2$ → requires symmetric routes in terms of transmission latency

Conventional approaches

- Berkeley Algorithm
  - Active, periodically polling time daemon
  - Averaging algorithm



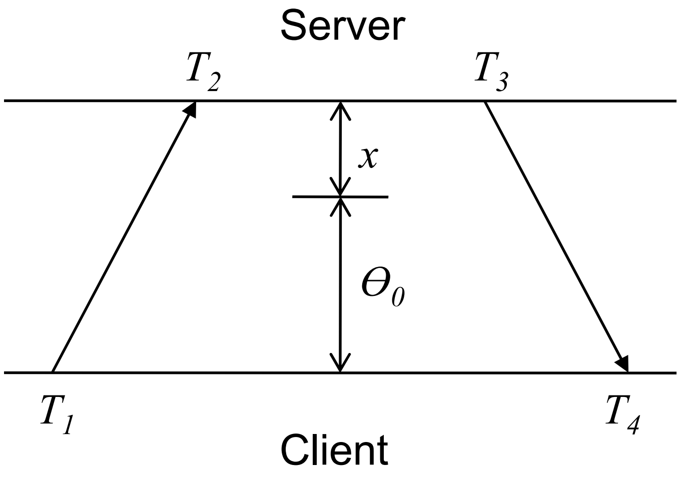
Conventional approaches

- NTP – Network Time Protocol
  - Similar to Critian's algorithm
- Estimation of
  - Round-trip delay

$$\delta = (T_4 - T_1) - (T_3 - T_2)$$

  - Clock offset

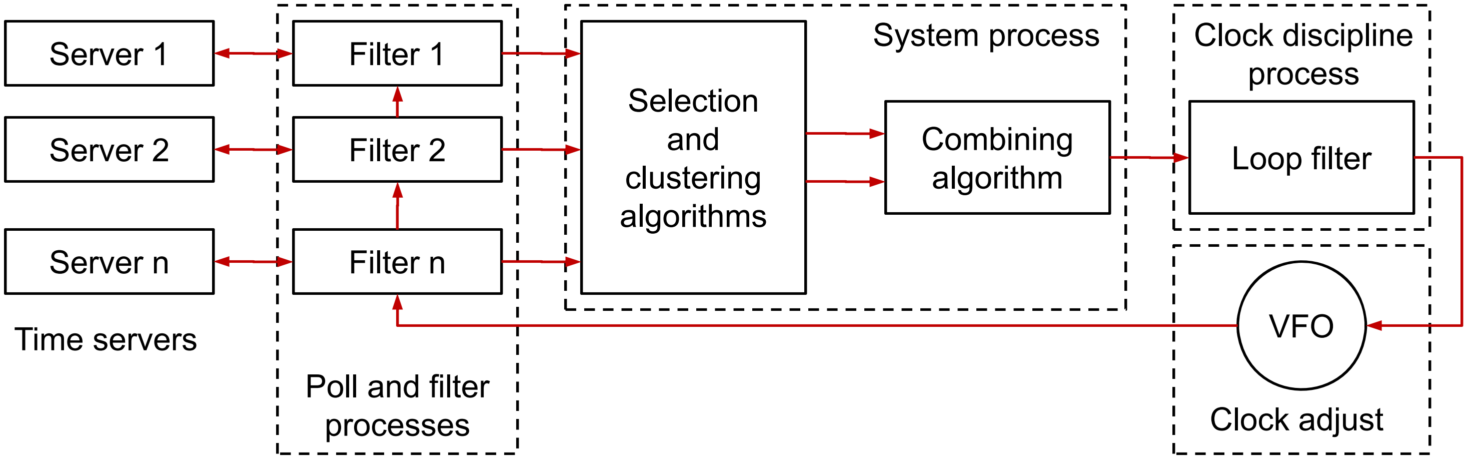
$$\theta = 1/2[(T_2 - T_1) + (T_3 - T_4)]$$



- Periodic calculation, δ_0 is estimated as the minimum of the last eight delay measurements
- the tuple (θ_0, δ_0) is used to update the local clock

NTP

- ❑ Major problems
  - ❑ System failures and unreliable data communication
  - ❑ Misbehavior
  - ❑ → may lead to time warps, i.e. unwanted jumps in time
- ❑ Solutions
  - ❑ Filters: phase-lock loops (PLLs)



Expected sources of error

- ***Skew in the receivers' local clocks*** – One way of reducing this error is to use NTP to discipline the frequency of each node's oscillator. Although running NTP all the time may lead to significant network utilization, it can still be useful for frequency discipline at very low duty cycles.
- ***Propagation delay of the synchronization pulse*** – Some methods assume that the synchronization pulse is an absolute time reference at the instant of its arrival - that is, that it arrives at every node at exactly the same time.
- ***Variable delays on the receivers*** – Even if the synchronization signal arrives at the same instant at all receivers, there is no guarantee that each receiver will detect the signal at the same instant. Nondeterminism in the detection hardware and operating system issues such as variable interrupt latency can contribute unpredictable delays that are inconsistent across receivers.

Time Synchronization in WSN

Design principle	Description
Energy efficiency	The amount of work needed for time synchronization should be as small as possible
Scalability	Large populations of nodes must be supported in unstructured topologies
Robustness	The service must continuously adapt to conditions inside the network, despite dynamics that lead to network partitions
Ad hoc deployment	Algorithms for time synchronization must work without *a priori* configuration settings

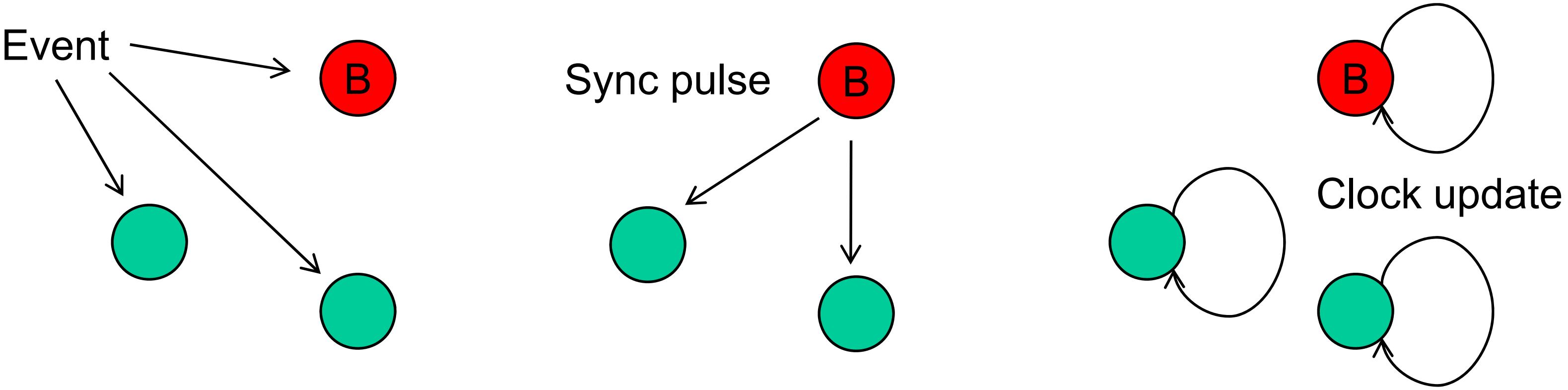
Time Synchronization in WSN

- *Virtual clocks*
  - represent the simplest type of synchronization algorithms
  - Based on the concept of logical clocks
  - Maintenance of the relative notion of time between nodes based on the temporal order of events without reference to the absolute time
- *Internal synchronization*
  - Maintains a common time in a single system or a group of nodes
  - Depending on the definition of ”internal”, this may include the notion of virtual clocks in WSNs
  - Cannot be extended to maintain clocks for distributed coordination actions
- *External synchronization*
  - Represents perhaps the most complex model
  - Every node maintains a local clock that is perfectly synchronized to a global and unique timescale
- *Hybrid synchronization*

Post-facto synchronization

- Principles
  - Assumes normally unsynchronized clocks
  - For each event, the node records the time of the stimulus with respect to the local clock
  - Immediately afterwards, a third party node broadcasts a synchronization pulse to all nodes in its radio broadcast range
  - All nodes that are receiving this broadcast use it as an instantaneous time reference and can normalize their stimulus timestamp with respect to that reference

→ mixture of logical clocks and time synchronization



Timing-sync Protocol for Sensor Networks (TPSN)

- Follows the sender-receiver model
- Basically, a two-way message exchange is used together with time stamping in the MAC layer of the radio stack

- *Level discovery phase*
  - The root node is assigned level 0 and initiates this phase by broadcasting a level discovery packet
  - Each node receiving this packet is assigned to level 1
  - These nodes rebroadcast the discovery packet and so on and so forth
- *Synchronization phase*
  - Pairwise synchronization is performed along the edges in the established tree based on sender-receiver synchronization
  - Two-way message exchange for estimating the propagation delay and the clock drift (similar to NTP)

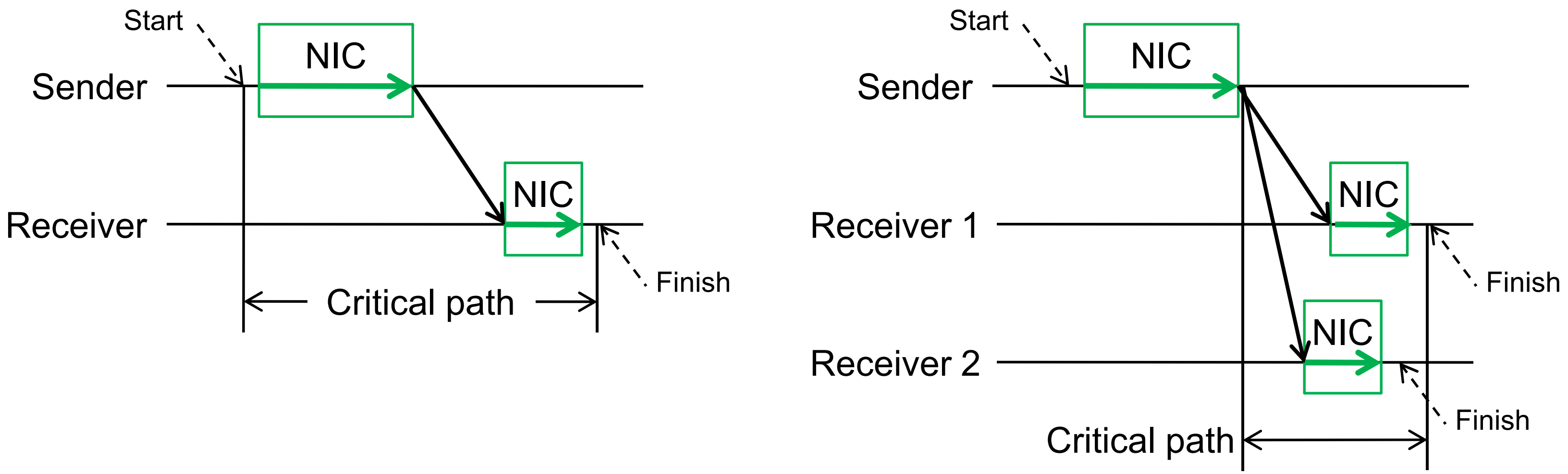
Reference Broadcast Synchronization (RBS)

- Observations in WSN
  - Communication in performed as local broadcasts rather than unicasts between arbitrary nodes
  - Radio ranges are short compared to the product of the speed of light times the synchronization precision
  - Delays between time-stamping and sending a packet are significantly more variable than the delays between receiving and time stamping (due to waiting for the free radio medium)

- Fundamental property of RBS is that it synchronizes a set of receivers with one another, as opposed to traditional protocols in which senders synchronize with receivers

Reference Broadcast Synchronization (RBS)

- RBS removes sender’s nondeterminism from the critical path and, in this way, produces high precision clock agreement



Distributed Coordination

Scalable coordination

- Primary requirements
  - The algorithms need to be designed to support ad hoc deployment of SANET nodes, which continuously adapt to the environmental conditions.
  - Untethered operation should be supported based on wireless radio communication.
  - The coordination need to be able to operate unattended as it might be infeasible to support continuous or periodic maintenance.

- Design choices
  - ***Data-centric communication*** – Sensor nodes may not have unique address identifiers. Therefore, pairs of attributes and values should be used to identify and to process received messages.
  - ***Application-specific operation*** – Traditional communication networks are supposed to support a wide variety of applications. In contrast, WSNs and SANETs are often designed for specific purposes or configured for a particular purpose.

Span

- Topology maintenance for energy efficient coordination
  - Similar to LEACH
  - Based on localized coordination instead of random election schemes

- Objectives
  - Span ensures that enough coordinators are elected to make sure that each node has a coordinator in its radio range
  - The coordinators are rotated to distribute workload
  - The algorithm aims at minimization of the number of coordinators in order to increase network lifetime
  - Span provides decentralized coordination relying on local state information

Span

- Protocol mechanisms
  - Proactive neighborship management using HELLO messages
  - Then, each non-coordinator node will become a coordinator if it discovers that two of its neighbors cannot reach each other either directly or via one or more coordinators

    → ensures connectivity but does not minimize the costs

  - Solution: optimized backoff delay

$$delay = \left(\underbrace{\left(1 - \frac{E_r}{E_m}\right)}_{\text{Remaining energy}} + \underbrace{\left(1 - \frac{C_i}{\binom{N_i}{2}}\right)}_{\text{Utility of node } i} + \underbrace{R}_{\text{Random value}} \right) \times \overset{\text{Number of neighbors}}{N_i} \times \underset{\text{Round-trip delay}}{T}$$

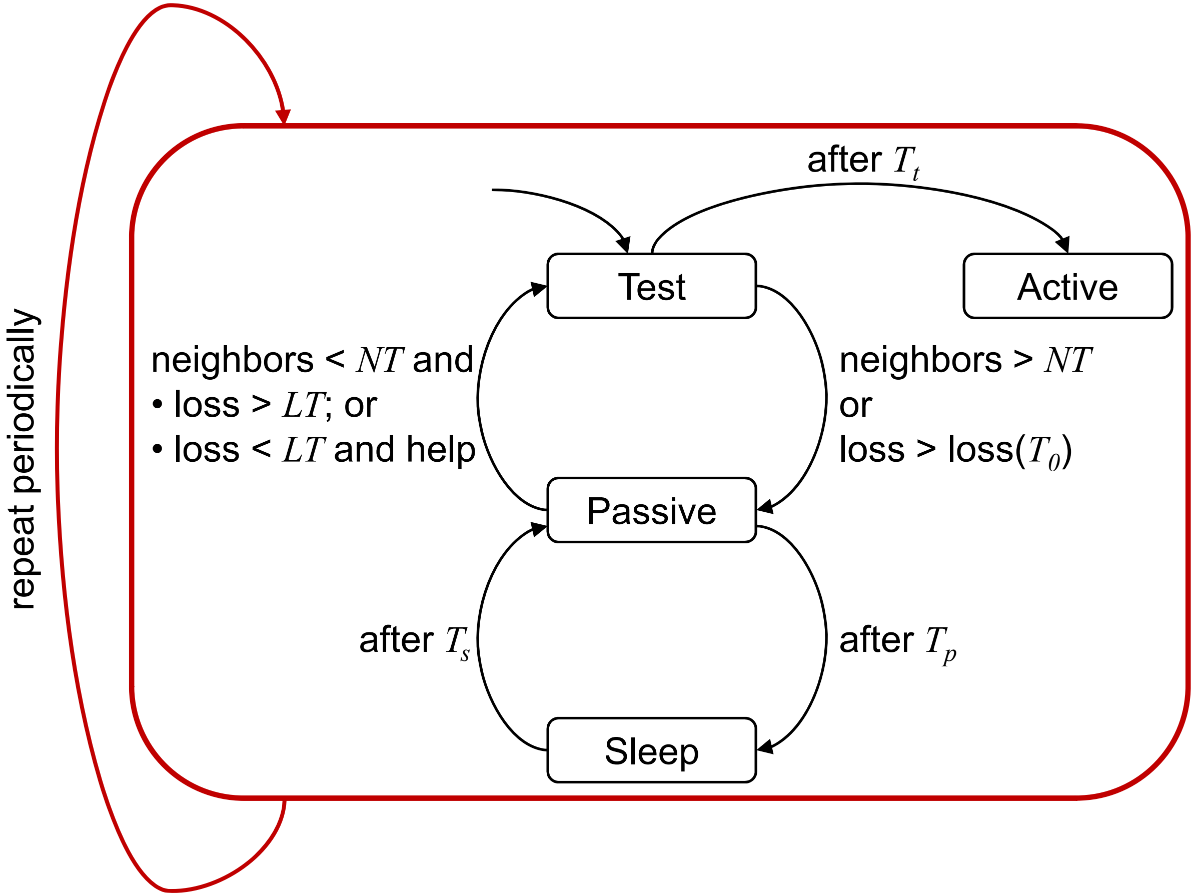
ASCENT

- Adaptive Self-Configuring Sensor Network Topologies
  - Topology maintenance similar to Span

- Based on three operations
  - A node signals when it detects high packet loss, requesting additional nodes in the region to join the network in order to relay messages
  - The node reduces its duty cycle if it detects losses due to collisions
  - Additionally, the node probes the local communication environment and does not join the multi-hop routing infrastructure until it is ”helpful” to do so

ASCENT

- Working behavior

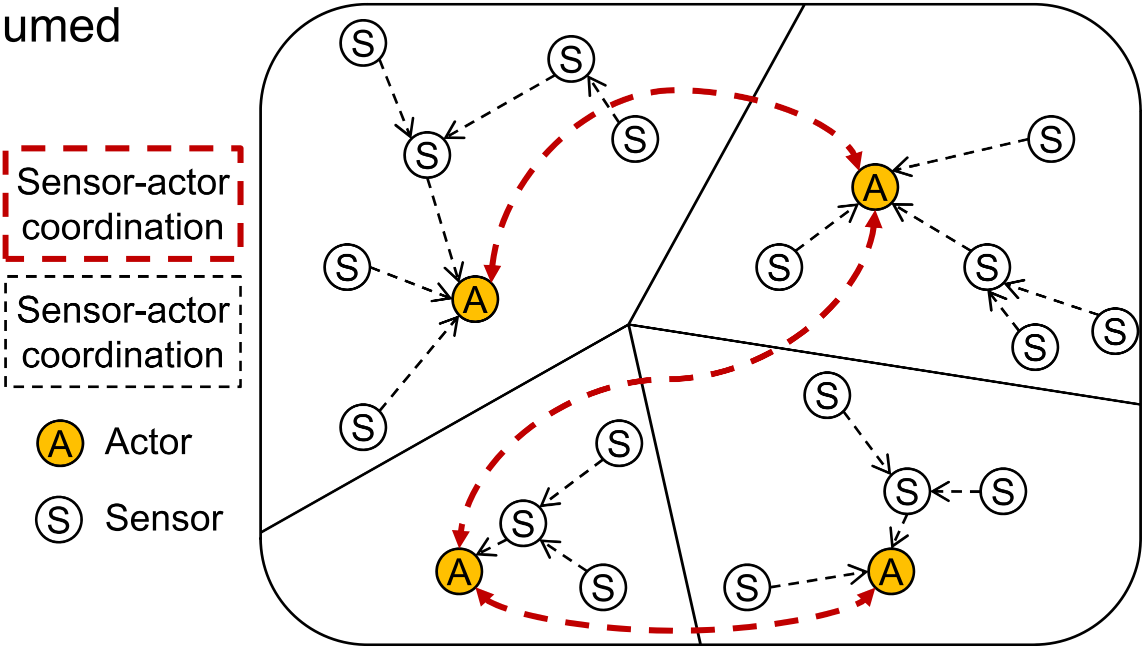


Sensor-actor coordination

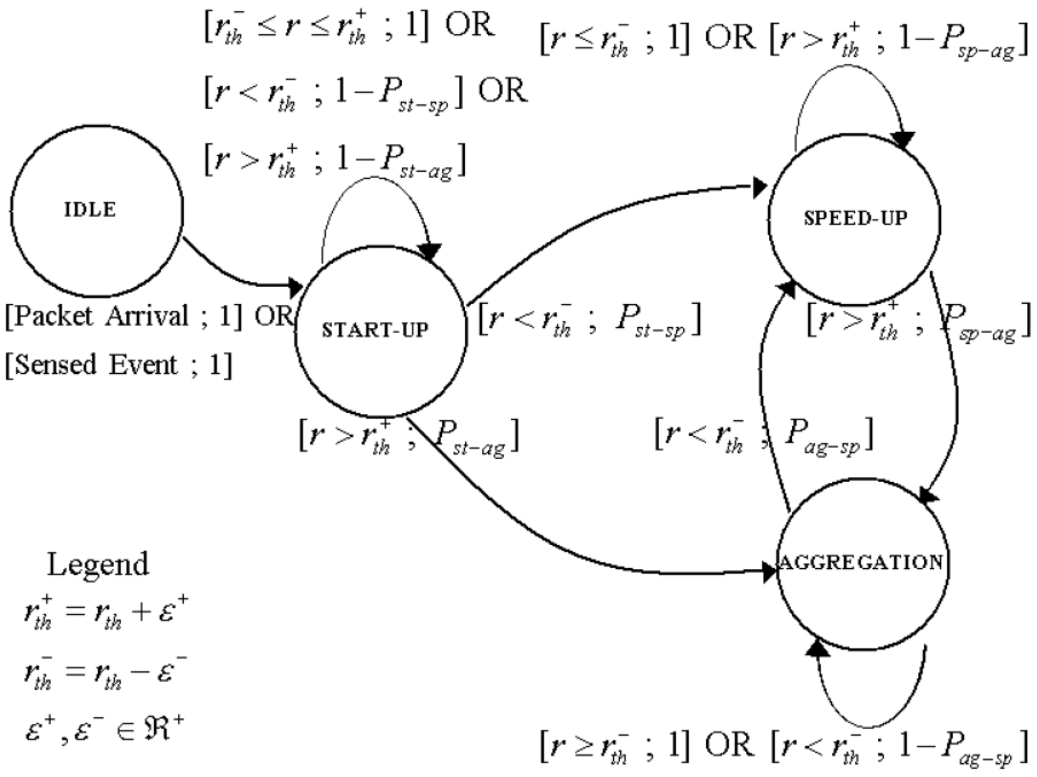
- Differentiation between sensor-actor and actor-actor coordination
  - First, associate sensors to actors (sink nodes)
  - Secondly, distribute application specified tasks among the actors

- Reliability *r* is the main measure
  - A latency bound can be depicted as “late” packets are assumed to be lost

- Energy savings if $r > r_{th}$
- Greedy routing if $r < r_{th}$



Sensor-actor coordination – local state machine

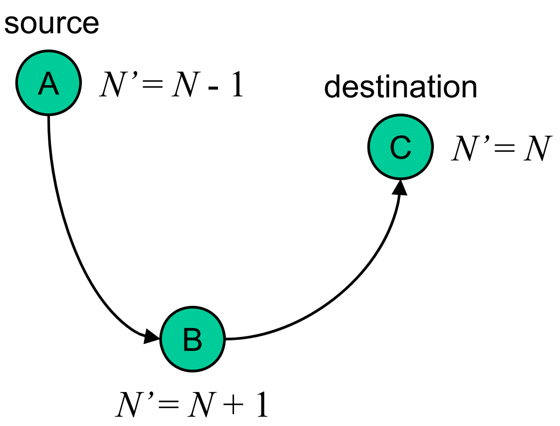

$[r_{th}^- \le r \le r_{th}^+ ; 1]$ OR
$[r < r_{th}^- ; 1-P_{st-sp}]$ OR
$[r > r_{th}^+ ; 1-P_{st-ag}]$
$[r \le r_{th}^- ; 1]$ OR $[r > r_{th}^+ ; 1-P_{sp-ag}]$
IDLE
SPEED-UP
START-UP
[Packet Arrival ; 1] OR
[Sensed Event ; 1]
$[r < r_{th}^- ; P_{st-sp}]$
$[r > r_{th}^+ ; P_{sp-ag}]$
$[r > r_{th}^+ ; P_{st-ag}]$
$[r < r_{th}^- ; P_{ag-sp}]$
AGGREGATION
Legend
$r_{th}^+ = r_{th} + \varepsilon^+$
$r_{th}^- = r_{th} - \varepsilon^-$
$\varepsilon^+, \varepsilon^- \in \Re^+$
$[r \ge r_{th}^- ; 1]$ OR $[r < r_{th}^- ; 1-P_{ag-sp}]$

- ❑ r^{+}_{th} is the high event reliability threshold
- ❑ r^{-}_{th} is the low event reliability threshold
- ❑ ε^+ and ε^- basically define a tolerance zone around the required reliability threshold to reduce oscillations

Problems in cooperative environments

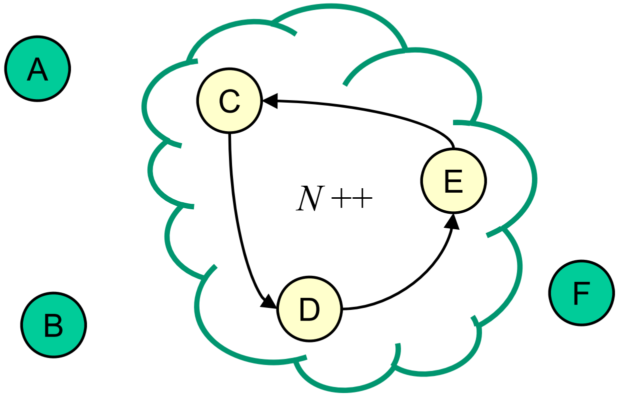
- ❑ Selfish nodes
  - ❑ Single nodes try to exploit available network resources
  - ❑ These nodes do not really participate in the network operation (actually, they will pretend to do so)

- ❑ Stimulating cooperation – for example based on a credit system
  - ❑ A security device (nuglet, must be tamper proof!) is used to maintain the credit
    - A node may send if it has enough credit, i.e. if its nuglet count is large enough; for an estimated n hop transmission, the node requires n credits from the nuglet (the nuglet must not become negative)
    - Whenever a node forwards a packet, its credit is increased by one



Stimulating cooperation

- Problems
  - The nuglet must be installed on each and every node and the nodes must be developed such that no bypassing is possible
  - Groups of nodes may still act selfish



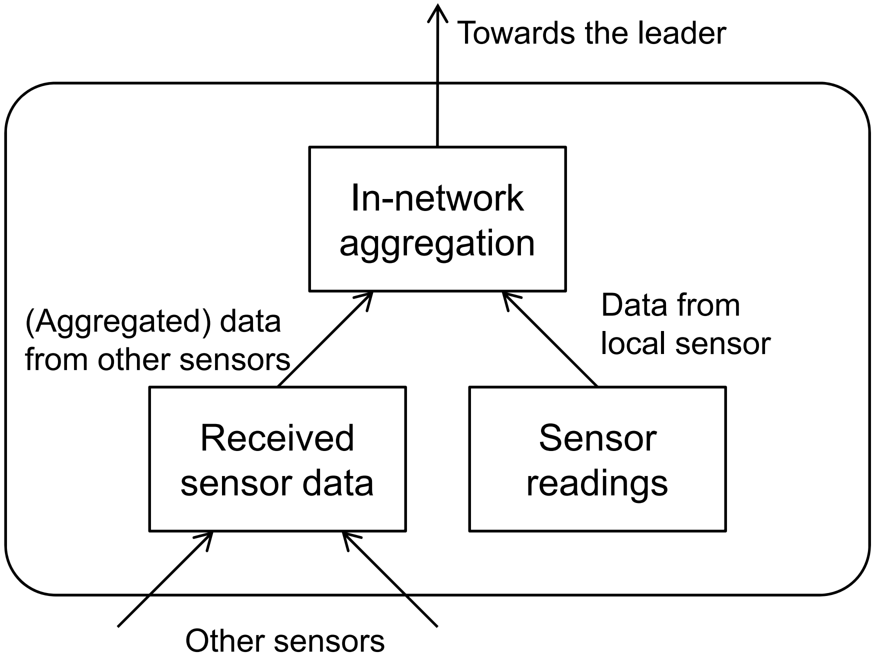
In-network operation and control

- Objectives
  - Energy-aware operation (high costs of wireless radio communication in comparison to computational efforts)
  - Two additional objectives have been identified that motivate the in-network operation in SANETs: *scalability and timeliness*

- Solutions
  - Network-centric data processing → data aggregation
  - In-network control → self-organization by cooperation

Cougar

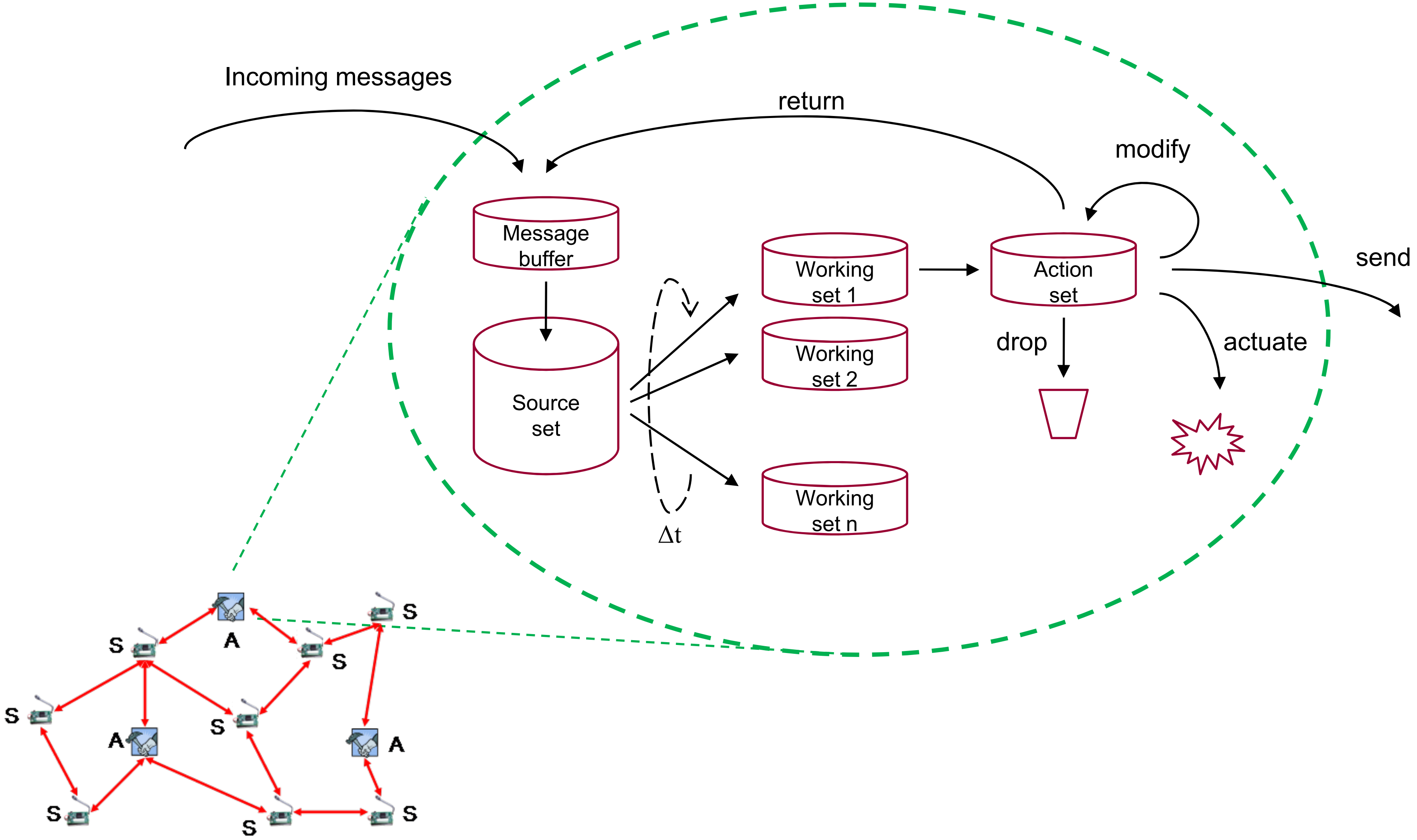
- In-network query processing and data aggregation
  - Computation is much cheaper compared to communication (energy)
  - Sensor readings might be failure-prone → validation is needed
    - As long as multiple sensor nodes measure the same physical phenomenon, their readings can be aggregated to construct a ”super-node” whose temperature readings have a much lower variance



Rule-based Sensor Network (RSN)

- Operation principles
  - ***Data-centric operation*** – Each message carries all necessary information to allow data specific handling and processing without further knowledge, e.g. on the network topology
  - ***Specific reaction on received data*** – A rule-based programming scheme is used to describe specific actions to be taken after the reception of particular information fragments
  - ***Simple local behavior control*** – We do not intend to control the overall system but focus on the operation of the individual node instead. Simple state machines have been designed, which control each node (being either sensor or actor)

RSN


Incoming messages
return
modify
Message buffer
Working set 1
Action set
send
Working set 2
drop
actuate
Source set
Working set n
Δt
S
A
S
S
S
S
A
S
A
S
S
S

RSN

- Possible actions
  - `modify` – A message or a set of messages can be modified, e.g. to fuse the carried information with locally available meta information.
  - `return` – Messages may be returned to the message buffer for later processing, e.g. for duplicate detection or improved aggregation.
  - `send` – Obviously, a node needs to be able to send messages. This can be a simple forwarding of messages that have been received or the creation of completely new messages needed to coordinate with neighboring nodes.
  - `actuate` – Local actuators can be controlled by received messages, e.g. to enable sensor-actor feedback loops.
  - `drop` – Finally, the node needs to be able to drop messages, which are no longer required, e.g. because they represent duplicates or because an aggregated message has already been created and forwarded.

RSN Example: Gossiping

- Each message is assumed to be encoded in the following way:

```
M := { type, hopCount, content }
```

- Then, the gossiping algorithm can be formulated as follows:

```
# infinite loop prevention
if $hopCount >= networkDiameter then {
   !drop;
}
# flooding for the first n hops
if $hopCount < n then {
   !sendAll;
   !drop;
}
# gossiping
if :random < p then {
   !sendAll;
   !drop;
}
# clean up
!drop;
```

RSN Example: Temperature monitoring + fire detection

- The message encoding is similar to the previous example:

```
M := { type, position, content, priority }
type := ( temperate || alarm )
```

- The complete algorithm can now be written as follows:

```
# test for exceeded threshold and generate an alarm message
if $type = temperature && $content > threshold then {
   !actuate(buzzerOn);
   !send($type := alarm, $priority = 1);
}
# perform data aggregation
if $type = temperature && :count > 1 then {
   !send($content := @media of $content, $priority := 1 - @product of
             $priority);
   !drop;
}
# message forwarding, e.g. according to the WPDD algorithm (simplified)
if :random < $priority then {
   !sendAll;
   !drop;
}
!drop;
```

Summary (what do I need to know)

- ***Synchronization vs. coordination***
  - Principles of Lamport timestamps
  - Weak synchronization

- ***Time synchronization***
  - Principles of classical solutions
  - Post-facto synchronization, sender-receiver synchronization (TPSN), receiver-receiver synchronization (RBS)

- ***Distributed coordination***
  - Objectives and principles
  - Span, ASCENT, Sensor-actor coordination

- ***In-network operation and control***
  - Cougar – aggregation and validation
  - RSN – rule-based sensor network programming

References

- N. Bulushu, D. Estrin, L. Girod, and J. Heidemann, "Scalable Coodination for Wireless Sensor Networks: Self-Configuring Localization Systems," Proceedings of 6th International Symposium on Communication Theory and Applications (ISCTA'01), Ambleside, Lake District, UK, July 2001.
- A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," *IEEE Transactions on Mobile Computing, vol. 3 (3), pp. 272-285, July/August 2004.*
- B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM Wireless Networks Journal, vol. 8 (5), September 2002.*
- F. Dressler, "Network-centric Actuation Control in Sensor/Actuator Networks based on Bio-inspired Technologies," Proceedings of 3rd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (IEEE MASS 2006): 2nd International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN 2006), Vancouver, Canada, October 2006, pp. 680-684.
- J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," Proceedings of 2001 International Parallel and Distributed Processing Symposium (IPDPS), San Francisco, CA, USA, April 2001.
- J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," Proceedings of Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, December 2002.
- S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync Protocol for Sensor Networks," Proceedings of ACM Conference on Embedded Networked Sensor Systems (Sensys 2003), Los Angeles, CA, November 2003.
- L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM, vol. 21 (4), pp. 558-565, July 1978.*
- T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "A Distributed Coordination Framework for Wireless Sensor and Actor Networks," Proceedings of 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2005), Urbana-Champaign, Il, USA, May 2005, pp. 99-110.
- Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *ACM SIGMOD Record, vol. 31 (3), pp. 9-18, 2002.*