Chapter 4

Distributed Sensor Networks

Deployment of Multi-Agent Systems in Distributed Sensor Networks

epubli, ISBN 9783746752228 (2018)

The growing complexity of computer networks and their heterogeneous composition of devices ranging from high-resources server to low-resource mobile devices demands for unique and standardized new data processing paradigms and methodologies. The Internet-of-Things is one major example rising in the past decade, strongly correlated with Cloud Computing and Big Data concepts.

4.1 Domains and Networks

The sensor network is partitioned into domains. Each network node creates a local domain that can bind multiple communication end-points. Multiple network nodes can be grouped in domains. Single nodes can act as bridges between domains. The network can be considered as being a global domain with independent sub-domains. Networks can be hierarchical, creating group- and graph-network domains.

4.2 The Sensor Node

All services of a sensor node are provided by stationary non-mobile service agents, shown in Figure *4.1*. There is a root agent, the node service agent, which is responsible to set up and start other service agents like the event manager and the sensor sampler agent.

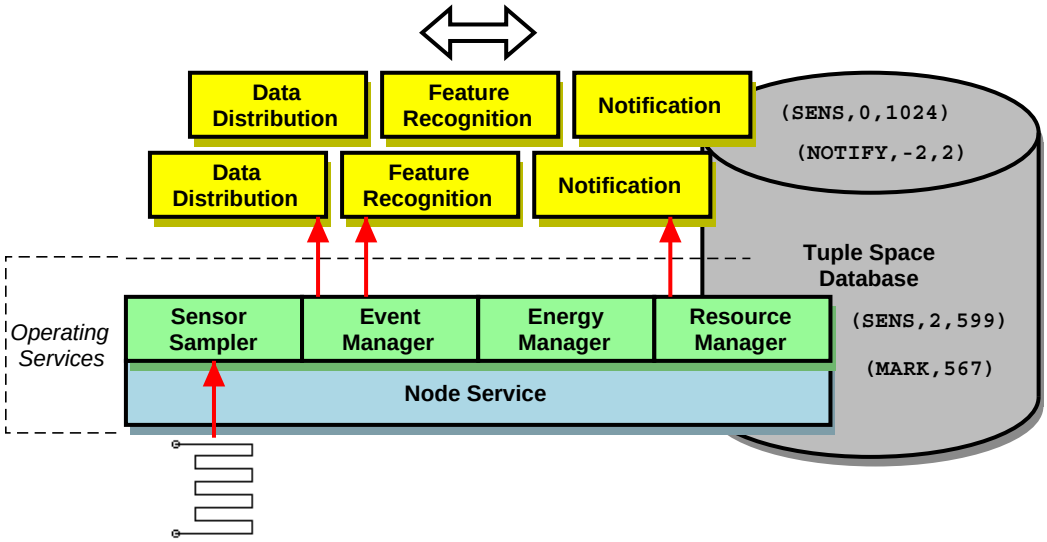

Fig. 4.1 *Stationary agents (green(on sensor node level providing node OS services interacting with mobile agents (yellow) using the tuple space data base*

The node service agent monitors the other node service agents. If they are in an unresponsive or failure state, the node service agent restarts the service agents to ensure node services availability.

4.3 The Sensor Network

A distributed sensor network as the central part of a sensing system consists of multiple active sensor nodes, a set N={n_1, n_2, ..}, connected and arranged in a network forming a graph $G(N,C)$ with edges C, a set C={c_1 , c_2, ..}, connecting nodes and providing communication (and eventually energy transfer) between nodes. Each sensor node provides signal and data processing for a set of sensors S={s_1, s_2, ..}, communication, some kind of energy supply and management, and sensor interfaces. The connectivity of the sensor nodes can be used for data and power exchange, too. Each sensor node should provide a minimal degree of autonomy and independence from other nodes in the network.

From the computer science point of view a sensor network can be composed of a set of processes P={p_1, p_2, ..} performing parallel data processing and interaction using messages M={m_1, m_2 ,..}. The processes communicate by exchanging data by using messages, and messages are exchanges by the nodes hosting the processes by using communication links. Message-based communication requires protocols for synchronization and message passing. Thus, in this sense and assuming the assumption of the above definition a distributed sensor network can be considered as being a distributed computer performing distributed data processing. Interaction between nodes is required to manage and distribute data and to compute information.

The properties and behaviour of a distributed system can be summarized as follows (see Figure 4.1):

1. Processing elements (PE) represent the physical resources and the nodes of the sensor network;
2. Processes are logical resources that perform the data processing for a specific task;
3. Cooperative communication and inter-process communication handled by message passing (instead of a centralized master-slave model);
4. Communication links (the interconnection network) are physical resources and connects PEs;
5. Processes are cooperative by interaction;
6. Communication delay does not affect the overall distributed program behaviour (immutability);

epubli, ISBN 9783746752228 (2018)

7. Robustness in the presence of resource failures (failures of single logical or physical resource do not affect overall system behaviour);
8. Run-time reconfiguration: adaptation of single resources or the whole system in the presence of resource failures.

Of these physical and logical elements, specifically 1, 2 and 4 are prone to fail.

4.3.1 Communication and Network Topologies

Communication is a central part of distributed data processing in sensor networks, influencing performance and operation significantly. Figure 4.2 (top) gives an overview of different network topologies providing connectivity between sensor or computational nodes, covering heterogeneous networks with computation platforms differing in computational and storage resources (including generic computers), too.

A connection between two nodes is usually a bidirectional (serial) link, which is capable to transfer data encapsulated in messages. Message passing is the preferred communication method for large scale networks compared with switched networks requiring crossbar or multi-stage Banyan (butterfly) switches. Path finding and routing, the process of forwarding a message in the network with the goal to deliver messages from the sender to the receiver is required in message passing networks, and for economic reasons and the sake of simplicity there are usually no dedicated routers in sensor networks. Thus, a sensor or computational node is a service end-point and a message router, too. Traditional computer networks use unique node identification that ate unsuitable for (especially loosely or ad-hoc coupled) sensor networks. Path finding algorithms can be classified in those using routing tables managed by each network node and storing information about the network topology (usually limited to the neighbourhood of the respective node), and those not relying on routing tables. Routing tables provide information for path planning in advance. Managing routing tables can be critical in resource constrained and real-time systems, because they allocate a fairly high amount of local storage and require computational and communication activities, which should be minimized in sensor networks. Routing tables can only reflect an outdated view of the network (the network configuration within a certain time interval) and hence dealing with network changes is complicated by using routing tables.

Communication links can be classified fundamentally in wired and wireless technologies, and networks themselves in message passing (static) and switched (dynamic) networks with regular and irregular topologies of a specific dimensionality (see Figure 4.2, bottom).

epubli, ISBN 9783746752228 (2018)

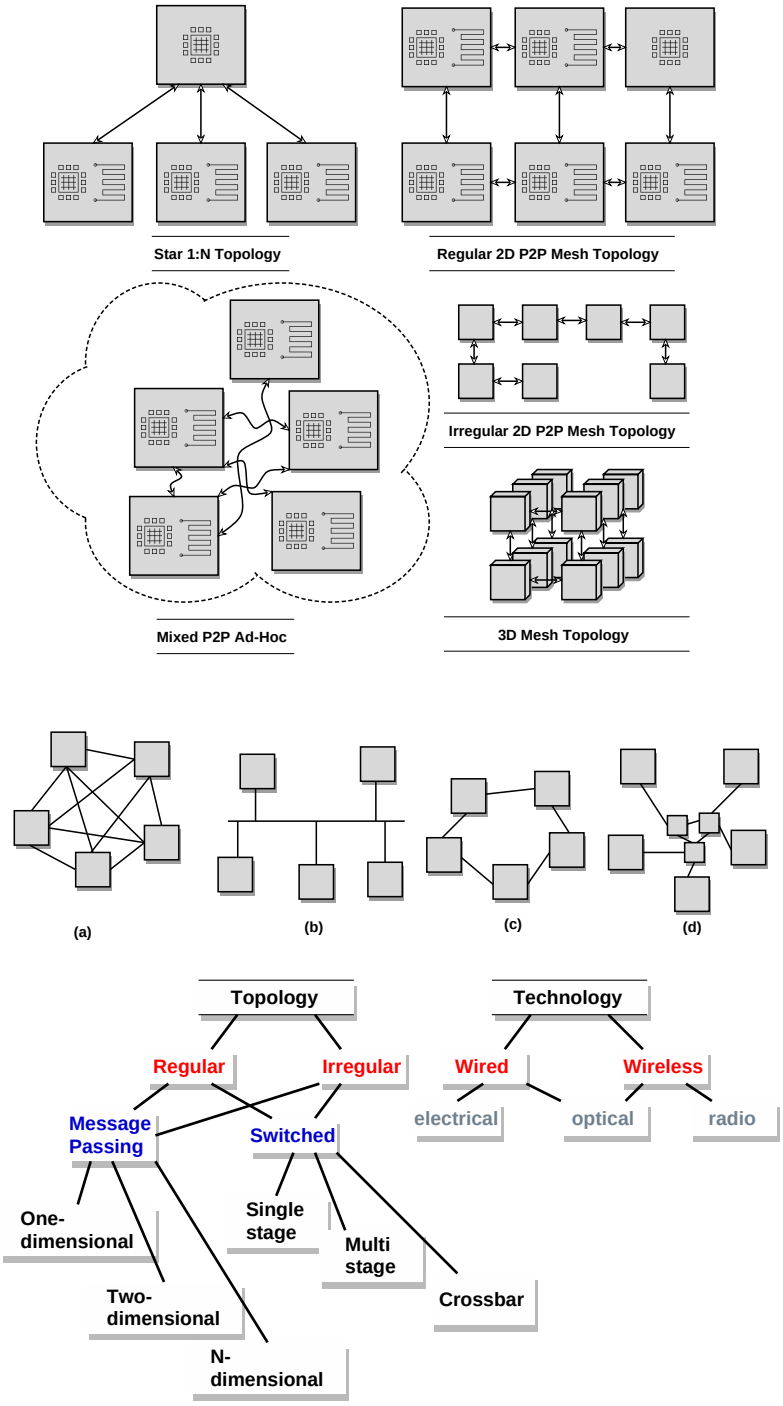

Fig. 4.2 *(Top) Network topologies and connectivity in DSN, (Bottom) Summary of network topology and technology classification*

There are hybrid approaches in wireless networks. Though macro-scale sensor networks profit from wireless technologies due to reduced installation and deployment complexity, micro-scale network embedded in material structure can profit from wired technologies, as these are in general more energy efficient than their wireless counterparts (directed energy flow in the former in contrast to non- or semi-directed energy flow in the latter).

One of the simplest and still widely used topologies are master-slave 1:*N* star networks. These are mainly used in centralized processing architectures. There is only one connection from each client to a server node (connectivity degree 1, distance 1). The failure of the server is critical. A different situation is shown in the bottom row (b) using a broadcast medium (bus, Ethernet) which connects *N* individual nodes, with a connectivity degree (*N*-1) and a maximal distance of 1. Here a failure of the broadcast medium (the interconnection) is critical.

Two-dimensional mesh-like networks, either fully or partially connected and using point-to-point links, are the preferred topologies for wired planar material-integrated sensor networks having the lowest interconnect complexity but still providing robustness in the presence of missing or defective links due to path redundancy (there is more than one possible path from a source to a destination node). They have a connectivity degree of 4 and a maximal distance of $\log_2 N$. Furthermore, the logical network topology corresponds to the geometrical placement order of nodes.

In fully connected networks (a) each node is connected with each other node, leading to a connectivity degree of *N*-1 and a distance of 1 with the highest degree of robustness and lowest message passing latency, but requiring the highest resource demands. Cube networks (three-dimensional topology, middle row, right side) provide a good compromise between the afore mentioned networks, having a connectivity degree of 6 and a distance of $\log_3 N$ (resulting in a lower message passing latency compared with two-dimensional networks). They are a special case of a generic hypercube networks (n-dimensional), with very limited practical use in material-integrated sensor network due to the complex interconnect structure.

In chain or ring networks (c) the connectivity is 2, the maximal distance *N*-1, providing no (chain) or low (ring) robustness. Here, failure of a single node can already be critical: The chain is only as strong as its weakest link.

Hierarchical networks (d) provide node partitioning in spatial bounded subnetworks connected with each other by using dedicated routers (for example, sub-star networks arranged in chain networks applied in [GHE10]).

A chain network has no redundancy, in contrast to two- and three-dimensional mesh and cube networks (with four and six neighbour node connections, respectively). They provide an increasing number of alternative paths without introducing additional high connectivity complexity and costs (number of connections are in the order of nodes).

epubli, ISBN 9783746752228 (2018)

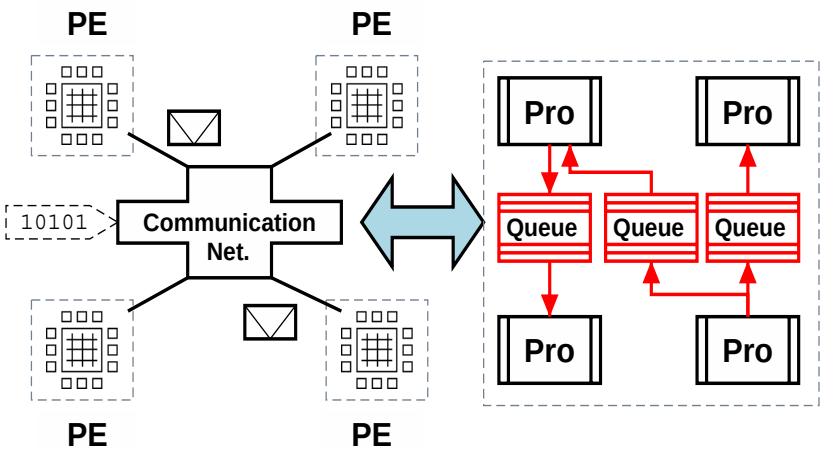

Fig. 4.3 *Abstraction of distributed computing in sensor networks with processes and message passing*

Distributed computing, i.e., in sensor networks, can be mapped on a communication processes model as outlined in Figure 4.3. Communication between processes is performed via network (or shared memory) message passing queues.

4.3.2 Message-passing and Routing

The possible number of paths ***P*** connecting the (upper) left and (lower) right nodes and the connectivity costs C (number of all links in a regular configuration) of the network depend on the number of nodes (***n,m,o***) in each dimension respectively and can be calculated from Equation 4.1 (assuming a connecting path visits each node at most one time).

$$
\begin{aligned}
&P_1(n) = 1,\ P_2(n,m) = \frac{(n+m)!}{n!m!},\ P_3(n,m,o) = \frac{(n+m+o)!}{n!m!o!} \\
&C_1(n) = n-1,\ C_2(n,m) = 2nm - m - n, \\
&C_3(n,m,o) = 3mno - m - n - o
\end{aligned}
\tag{4.1}
$$

Figure 4.4 compares the redundancy of 1-, 2-, and 3-dimensional grid networks in relation to the number of nodes.

In macro-scale sensor networks nodes are often connected by using wireless technologies with dynamic ad-hoc network topologies. Protocol-based routing in changing networks (regarding communication connections and nodes) is one main challenge in the design of robust and energy-aware sensor networks.

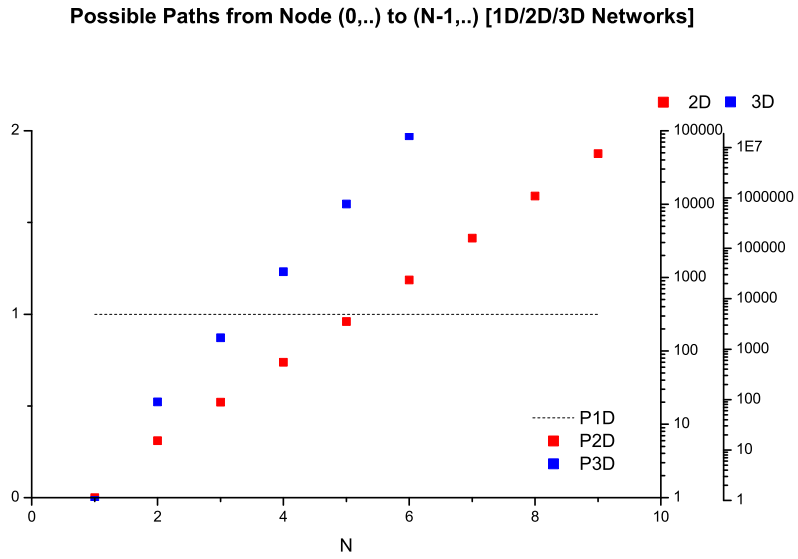

Fig. 4.4 *Redundancy in mesh-like networks (chain, grid, cube): possible paths from node (0,..) to (n-1,..) depending on the network size (n nodes in each dimension)*

Most traditional routing protocols rely on an address space uniquely identifying nodes as well as routing tables storing network information usually limited to a bounded spatial scope. Path finding is required for the process of forwarding of a message, and is the main goal of routing by minimizing the number of passed routers (e.g., nodes). Commonly there is a set of different paths connecting nodes. In the past many routing algorithms were developed (for wireless and wired networks) finding 1. Alternative paths in the case of missing or non-operating connectivity in parts of the networks, and by 2. Finding the shortest path (concerning the distance in hop counts and the delivery latency). One examples is junction based routing (discussed in detail in [BAD11]). Routing in large-scale wireless sensor networks is discussed in detail in [LI11].

Geographic routing and addressing bases on geometrical relations between nodes, for example, relative addressing specifying the relative distance between nodes, can be used to avoid absolute unique node addressing, which is not applicable in large scale miniaturized networks with an initially unknown configuration, like smart dust networks [WAR01].

The scaling of networking down to material-integrated level requires simplifying and robust approaches. Some of them capable for microchip level implementations satisfying low-resource constraints are summarized below.

They all not rely on routing tables (saving memory and computation) as well as they are able to adapt to network changes immediately.

4.3.3 Advanced Δ-Routing with Backtracking

Basics

Traditional Δ-distance routing can only be deployed in regular network structures existing in mesh or grid like networks. Incomplete networks with missing neighbour node connectivity and irregular networks with missing nodes (disturbing the spatial regularity) require adaptive Δ-distance routing strategies and backtracking to overcome these irregularities, for example, isolated traps, shown in Figure *4.1*.

The reliable communication protocol *SLIP* [BOS12A] implements smart and adaptive routing strategies suitable to be used in technical and material-integrated peer-to-peer networks with a similarity of the logical and physical network topology, that means, the neighbourhood connectivity of nodes can be uniquely assigned to a specific geometric direction.

The protocol message format is scalable with respect to the network size (address size class ***ASC***, ranging from 4 to 16 bit), maximal data payload (data size class ***DSC***), ranging from 4 to 16 bit length) and the network topology dimension size (address dimension class ***ADC***, ranging from 1 to 4).

Network nodes are connected using (serial) point-to-point links, and they are arranged along different metric axes of different geometrical dimensions: a one-dimensional network (***ADC***=1) implements chains and rings, a two-dimensional network (***ADC***=2) can implement mesh grids, a three-dimensional (***ADC***=3) can implement cubes, and so on. Both incomplete (missing links) and irregular networks (with missing nodes and links) are supported for each dimension class, shown in Figure *4.1*.

The main issue concerning message-based communication is routing and thus addressing of nodes. Absolute and unique addressing of nodes in a high-density sensor network is not suitable. An alternative routing strategy is delta-distance routing, used by *SLIP*. A delta-distance vector Δ specifies the way from the source to a destination node counting the number of node hops in each dimension.

epubli, ISBN 9783746752228 (2018)

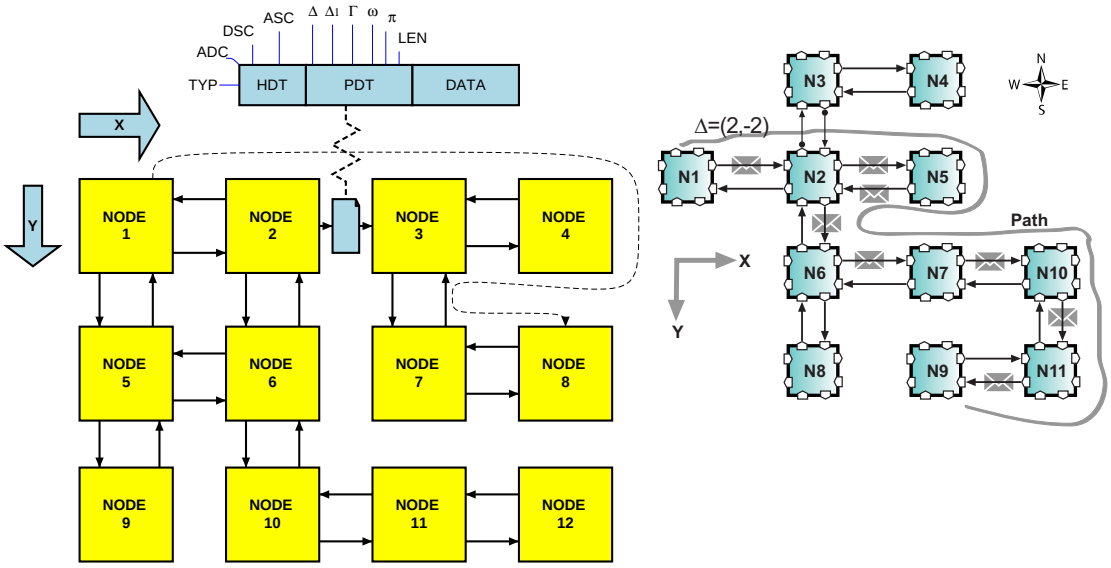

Fig. 4.5 *Message based communication in two-dimensional networks using delta-distance vector routing. Networks with incomplete (missing links) and irregular (missing nodes) topologies are supported by using smart routing rules.*

SLIP Message Format

A message packet contains a header descriptor specifying the type of the packet and the scalable parameters ***ASC***, ***DSC***, and ***ADC***, shown in Table *4.1*. The network address dimension ***ADC*** reflects the network topology, the size class ***ASC*** the upper size bounds of the network, and the data size class ***DSC*** the upper bound of the data payload. There are two different main message types: requests and replies.

A packet descriptor follows the header descriptor, containing: the actual delta-vector Δ, the original delta-vector Δ^0, a preferred routing direction ω, an application layer port π, a backward-propagation vector Γ, and the length of the following data part. The total bit length of the packet header depends on the {***ASC***, ***DSC***, ***ADC***} scalable parameter tuple setting, which optimises application specific the overhead and energy efficiency (spatially & temporarily). Each time a packet is forwarded (routed) in some direction, the delta-vector is decreased (magnitude) in the respective dimension entry. For example, routing in the x-direction results in: $\Delta_1=\Delta_1-1$. A message has reached the destination iff $\Delta=(0,0,..)$ and can be delivered to the application port π. There are different smart routing rules, applied in the order showed below until the packet can be routed (or discarded), shown in Algorithm *4.1*. First the normal XY routing is tried, where the packet is routed in each direction one after

another with the goal to minimize the delta count of each particular direction. The ω part of the packet descriptor selects a preferred starting direction. Default the x-direction is selected. If this routing strategy is impossible due to missing connectivity, the next routing strategy is tested. The packet is tried to send in the opposite direction, which will be marked in the gamma entry Γ part of the message packet descriptor. Opposite routing is used to escape small area traps, thereby backward routing is used to escape large area traps or to send the packet back to the source node indicating that the packet cannot be delivered. The routing decision is based on the actual message entries $\{\Delta, \Gamma, \omega\}$ and achieves adaptive routing reflecting the actual network topology and the path the message already had travelled, including back-end traps, resulting in routing of alternative paths by choosing different routing directions.

Entry	*Size [bits]*	*Description*
`HDT:ADC`	2	Address Dimension Class
`HDT:ASC`	2	Address Size Class
`HDT:DSC`	2	Data Size Class
`HDT:TYP`	2	Message type = {`Request, Reply, Alive, Acknowledge`}
`PDT:`Δ	Num(*ADC*)*Bits(*ASC*)	Actual delta vector
`PDT:`Δ^0	Num(*ADC*)*Bits(*ASC*)	Original delta vector
`PDT:`Γ	2*Num(*ADC*)	Backward propagation vector
`PDT:`ρ	Bits(*ADC*)	Preferred routing dimension (x,y,..)
`PDT:`π	Bits(*ASC*)	Application layer port
`PDT:LEN`	Bits(*DSC*)	Length of packet
`DATA`	LEN*Bits(*DSC*)	Data

Tab. 4.1 *SLIP message format (HDT: header descriptor, PDT: packet descriptor,)*

A message is only send to a neighbour node using the particular link iff the connection to the neighbour node was negotiated and is fully operational concerning the sending and the receiving of messages to and from the neigh-

bour node. For this purpose, the communication controller sends periodically ALIVE messages to all direct surrounding nodes and waits for ACKNOWLEDGE messages send back from the neighbour node to check the state of a connection. Non-existing nodes can be detected this way, too.

The Routing Algorithm

Alg. 4.1 *Functional specification of the adaptive smart routing algorithm in pseudo notation (ω_{in}: direction in which the message has arrived,* moveto *sends the message in the given direction, finally applying the route function again, $\{\}^{\rho}$: ordered list starting with the preferred routing dimension ρ if contained, $\uparrow$ raises a signal terminating a set iteration, $?\Lambda$: test for link to a neighbour node in the given direction)*

```
Type M = Message(Δ,Δ⁰,Γ,ρ,π,Len,Data)message strcuture
ω = {NORTH,SOUTH,WEST,EAST,..)      set of directions
ω⁻ = {SOUTH,WEST,..}                set of 'negative' directions
ω⁺ = {NORTH,EAST,..}                set of 'positive' directions
DIM = {1,2,..}                      set of dimension numbers

DEF ω° = fun i →                 convert coordinate to topology direction
  case i of | -1 => WEST | 1 => EAST | -2 => SOUTH | 2 => NORTH | ..
DEF ω• = fun dir →               convert topology to coordinate direction
  case dir of | WEST => -1| EAST => 1 | SOUTH => -2 | NORTH => 2 | ..

DEF routeⁿᵒʳᵐᵃˡ = fun M,ωⁱⁿ →
  try
    ∀ { i ∈ DIM | M.Δᵢ ≠ 0 }^M.ρ do          iterate over all dimensions
     Gamma ignored for backward travel and if this direction is reached
      if M.Δᵢ > 0 ∧ ?Λ(ω°(i)) ∧ ¬(Γ ≠ 0) ∧ M.Δᵢ ≠ M.Δᵢ⁰) then
        moveto(ω°(i))
        route(M with Δᵢ=M.Δᵢ-1,ω°(i))
        ↑routed
      if M.Δᵢ < 0 ∧ ?Λ(ω°(-i)) ∧ M.Γᵢ=0 then
        moveto(ω°(-i))
        route(M with Δᵢ=M.Δᵢ+1,ω°(-i))
        ↑routed
    done
    false                            no success
  with routed → true                 success

DEF routeᵒᵖᵖᵒˢⁱᵗᵉ = fun M,ωⁱⁿ →
  try
    ∀ { i ∈ DIM }^M.ρ do                        iterate over all dimensions
      if M.Γᵢ = 0 ∧ ωⁱⁿ ≠ ω°(-i) ∧ ?Λ(ω°(-i)) then
        moveto(ω°(-i))
        route(M with [Δᵢ=M.Δᵢ+1, Γᵢ=-1],ω°(-i))
        ↑routed
      if M.Γᵢ = 0 ∧ ωⁱⁿ ≠ ω°(i) ∧ ?Λ(ω°(i)) then
```

```
        moveto(ω°(i))
        route(M with [Δi=M.Δi-1, Γi=-1],ω°(i))
        ↑routed
    done
    false                              no success
  with routed → true                   success


DEF route^backward = fun M,ω^in  →
  try
    ∀ { i ∈ DIM }^M.ρ do                          iterate over all dimensions
      if M.Γi = 0 ∧ ω^in ∈ ω⁻ ∧ ?Λ(ω°(-i)) then
        moveto(ω°(-i));
        route(M with Δi=M.Δi+1,Γi =-1)
        ↑routed
      if M.Γi = 0 ∧ ω^in ∈ ω⁺ ∧ ?Λ(ω°(i)) then
        moveto(ω°(i))
        route(M with Δi=M.Δi-1, Γi =-1)
        ↑routed
      if M.Γi = -1 ∧ ?Λ(ω°(-i)) then
        moveto(ω°(-i))
        route(M with Δi=M.Δi+1,ω°(-i))
        ↑routed
      if M.Γi = 1 ∧ ?Λ(ω°(i)) then
        moveto(ω°(i))
        route(M with Δi=M.Δi-1,ω°(i))
        ↑routed
    done
    false                              no success
  with routed → true                   success

DEF route = fun M,ω^in →
 ω^in : incoming-direction-of-message M
 if M.Δ=(0,0,..) then deliver(M.π,M)
 else
    if ¬route^normal(M,ω^in) then
    if ¬route^opposite(M,ω^in) then
    if ¬route^backward(M,ω^in) then
    discard(M)
```

In a first routing attempt, the normal routing strategy is tried with the goal to minimize the Δ-vector in each dimension of an ordered set of dimensions, starting with the preferred routing dimension ρ. If this is impossible due to a lack of required connectivity the message is tried to send in one opposite direction increasing the Δ-vector temporarily. Opposite travel is marked in the Γ entry of the message. Finally, if this strategy fails also, the message is sent backward, finally reaching the source node again if there are no other routing alternatives found on the back path. Some care must be taken to avoid sending a message back to the direction from which it originally comes from

(resulting in a ping-pong forwarding with a live lock). To summarize, this simple routing algorithm uses decision-making based on information stored in the message, and neighbourhood connectivity retrieved by the current node.

The hardware implementation (using *ConPro* and standard cell ASIC synthesis, details can be found in [BOS10A] and [BOS12A]) requires about 250k equivalent gates, including 15k Flip-Flop (FF) registers, which relates to a chip area of $\cong 2.5\text{mm}^2$ assuming ASIC standard cell technology 0.18µm. The design is partitioned on *ConPro* programming level in 34 processes, communicating by using 16 queues. There are parallel executed receiver and sender processes for each communication port.

Robustness and Stability Analysis

A simulation of a sensor network consisting nodes arranged in a two-dimensional matrix with 10 rows and 10 columns was performed by using a multi-agent model. Messages and sensor nodes were modelled with agents. A comparison of *XY* and smart routing using the routing rules introduced in Algorithm *4.1* is shown in Figure *4.6*. The diagram shows the analysis results of operational paths depending on the number of link failures. A path is operational (reachable) iff a node (device under test), for example node at position (2,2), can deliver a request message to a destination node at position (x, y) with $x \neq 2 \vee y \neq 2$, and a reply can be delivered back to the requesting node. A failure of a specific link and node results in a broken connection between two nodes. The right image in Figure *4.6* shows an incomplete network with 100 broken links.

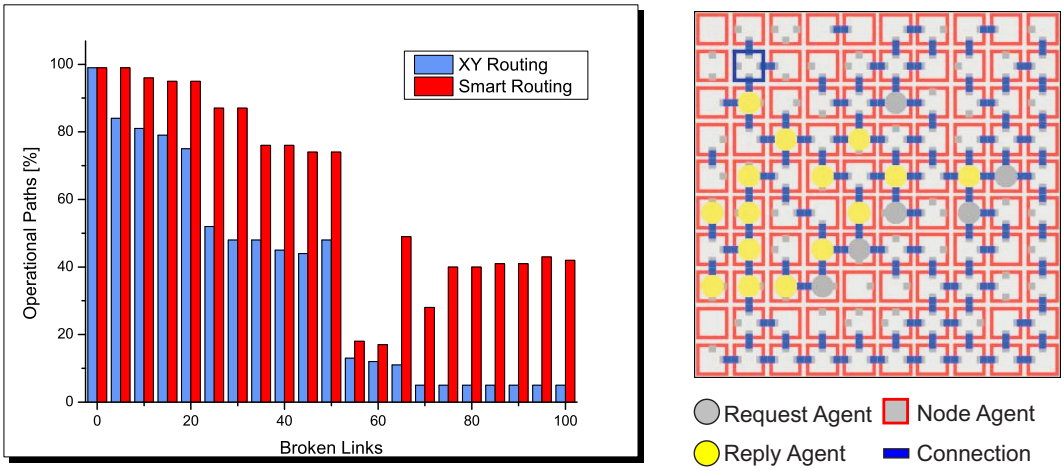

Fig. 4.6 *Robustness analysis with results obtained from simulation (left) and snapshot of sensor network (right)*

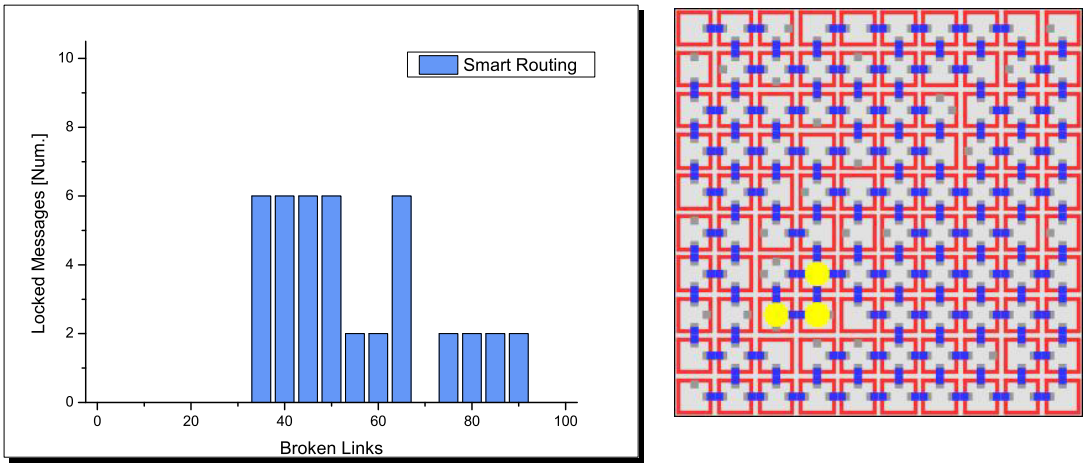

Fig. 4.7 *Stability analysis (live locked messages, left) and snapshot of sensor network with trapped messages (right)*

With traditional *XY* routing there is a strong decrease of operational paths, from a specific node (DUT) to any other node, if the number of broken links increases. Using smart routing increases the number of operational paths significantly, especially for considerable damaged networks, up to 50% compared with XY routing providing only 5% reachable paths any more.

Results from stability analysis shown in Figure *4.7* point out unstable behaviour under some particular network topologies. Though most situations are live lock free, there are some live locked messages circulating for ever in some isolated traps, shown for example in the snapshot on the right side of this figure.

The previous introduced simple adaptive routing strategy with an optimized scalable message header outperforms traditional table-based and diffusion-based routing strategies in terms of resource and path-search efficiency (details of common routing algorithms in DSN can be found in [LI11]). Live-locks can be avoided and the visited node count can be improved by adding a path-node list memory stored in the message, as proposed and evaluated in [BEN13]. But the improvement in delivery probability, delivery latency, and energy efficiency using a path history for the prediction and selection of alternative paths is compromised by a significantly increased size of the message header, and additionally the header has a non-fixed size, which makes the processing of messages more difficult.

4.3.4 From Passive Messages to Active Agents

The previously adaptive routing of messages is used in various mobile MAS presented in this book. The routing, originally part of the network node (operating) system is shifted towards agents and part of the agent behaviour, which has an additional routing goal, making routing active by the agent itself. Examples can be found in Chapter *9*. The nodes provide direct neighbourhood connectivity only.

Active mobile agents used for communication and data distribution in sensor networks has the advantages of adaptivity, ad-hoc behaviour, and forming of self-organizing communication structures between agents. This includes, e.g., the election of relay nodes in domains of networks and the propagation along dynamic travelling paths between multiple data source and sink nodes, shown in Fig *4.8*.

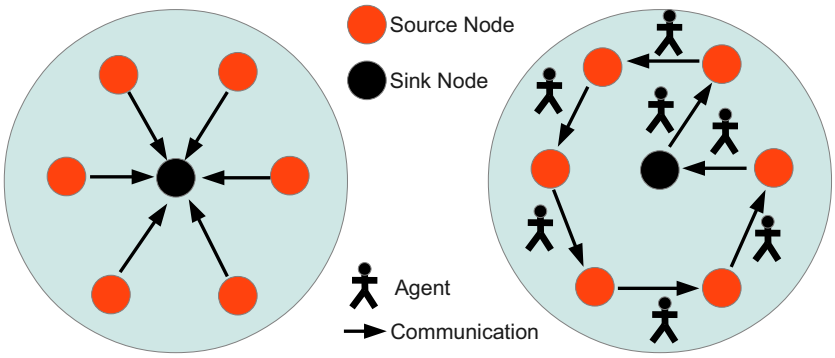

Fig. 4.8 *The comparison of traditional server-client communication (left) and mobile agent model (right)*

The mobile agent routing is a kind of complex combinatorial optimization problem that solves the optimal path according to a sequence of visited nodes and addressing energy efficiency in sensor networks, especially concerning wireless sensor networks (WSN). Ant colony optimization (ACO) is put forward to solve migration of mobile agent.

epubli, ISBN 9783746752228 (2018)

4.4 Further Reading

1. F. Hu and Q. Hao, *Intelligent Sensor Networks*, CRC Press, 2013, ISBN 9781420062212
2. M. J. McGrath and C. N. Scanaill, *Sensor Technologies Healthcare, Wellness, and Environmental Applications*, Apress Open, 2014, ISBN 9781430260134
3. R. Tynan, D. Marsh, D. O'Kane, and G. M. P. O'Hare, *Intelligent agents for wireless sensor networks*, in Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS'05, 2005, p. 1179.
4. A. Rogers, D. D. Corkill, and N. R. Jennings, *Agent Technologies for Sensor Networks*, IEEE Intelligent Systems, vol. 24, no. 2, 2009.
5. W. Liu, *Data Driven Intelligent Agent Networks for Adaptive Monitoring and Control*, Michigan Technological University, 2012.
6. S. S. Iyengar and R. R. Brooks, *Distributed Sensor Networks*. CRC Press, 2005.
7. Stefano Bonucelli, *Agent-Based Routing Algorithms for a Wired Sensor Network*, Università degli Studi di Genova Scuola Politecnica, 2014.