Self-Organization in Communication Networks: Principles and Design Paradigms

Christian Prehofer and Christian Bettstetter, DoCoMo Euro-Labs

ABSTRACT

The trend toward ubiquitous wireless communication demands for a higher level of self-organization in networks. This article gives an introduction and overview on this topic and investigates the fundamental question: What are design paradigms for developing a self-organized network function? We propose four paradigms and show how they are reflected in current protocols: design local interactions that achieve global properties, exploit implicit coordination, minimize the maintained state, and design protocols that adapt to changes. Finally, we suggest a general design process for self-organized network functions.

INTRODUCTION

The phenomenon of self-organization is pervasive in many areas of our life. In nature, for example, fish organize themselves to swim in well structured swarms, ants find shortest routes to food sources, and fireflies emit light flashes in perfect synchrony. Other examples of self-organized behavior can be observed in economy, population dynamics, psychology, and brain theory ([1–4]). In all these examples, the participating entities establish an organizational structure that does not require any central coordination. Instead, the entities interact directly with each other and continuously react to changes in their local environment. Typically, such self-organizing systems are very flexible, adaptive, failure-robust, and scalable.

While many processes around us are self-organized, telecommunications networks have not exploited this potential in depth so far; many functions are still centrally organized and require significant manual configuration for deployment and operation. For example, cellular networks require thorough frequency planning and use centralized databases for mobility management. With the emergence of IP-based networks, however, we can see a trend toward self-organized control functions. The Internet Protocol (IP) suite exploits, for example, decentralized congestion control and address autoconfiguration. This trend will be further accelerated by the advent of ubiquitous computing, where wireless technologies interconnect an increasing number and diversity of devices, such as portable music and video players, wearable computers, electronic papers, sensors, and others. This leads to increased complexity and dynamics of communications networks, which might become a stumbling block for further development. We believe that a higher level of self-organization will help us to master these challenges. In particular, it will reduce the administration efforts of users and network operators.

The goal of this article is twofold. First, it gives an introduction to and overview of self-organization in the context of communications and computer networks; second, it elaborates on basic paradigms and a methodology for designing self-organized networking functions. We proceed as follows. We first give a definition of self-organization, along with its underlying principles and benefits. This is followed by some examples where self-organized functions can be found in current and evolving communications networks. Based on this, we identify four general design paradigms for self-organized networking. Specifically, we address the locality of interactions, perfectness of coordination, maintained state, and adaptability to changes. Finally, we put these four paradigms together and suggest a general guideline on how to engineer a self-organized network function.

DEFINITION AND BENEFITS OF SELF-ORGANIZATION

The term *self-organization* occurs in many branches of science, yet there is no commonly accepted definition in the literature. Researchers study existing phenomena of self-organization, but they take different viewpoints and analyze different aspects (see [1–4]). This makes self-organization a very interdisciplinary and heterogeneous field. Let us thus give a definition that is general enough to hold in different sciences but still specific enough to be useful in the area of communications networks.

0163-6804/05/$20.00 © 2005 IEEE

We regard a system consisting of several entities. We call this system *organized* if it has a certain *structure* and *functionality* [4]. Structure means that the entities are arranged in a particular manner and interact (communicate) with each other in some way. Functionality means that the overall system fulfills a certain purpose. For example, a school of small fish tries to achieve a group structure that protects the fish against enemies (Fig. 1).

A system is *self-organized* if it is organized without any external or central dedicated control entity. In other words, the individual entities interact directly with each other in a *distributed* peer-to-peer fashion. Interaction between the entities is usually *localized*. For example, in a school of fish each individual fish bases its behavior on its observation of the position and speed of its immediate neighbors, rather than on the behavior of a "central fish" or that of the whole school.

But self-organization is more than just distributed and localized control. It is about the relationship between the behavior of the individual entities (the microscopic level) and the resulting structure and functionality of the overall system (the macroscopic level). In self-organized systems, the application of rather simple behavior at the microscopic level leads to sophisticated organization of the overall system. This phenomenon is called *emergent behavior*. Why and how emergent behavior occurs is still not well understood.

Another important characteristic of self-organized systems is their *adaptability* with respect to changes in the system or environment. In fact, the entities continuously adapt to changes in a coordinated manner, such that the system always reorganizes as a reaction to different internal and external triggers for change. By doing so, it tries to converge toward desired beneficial structures while avoiding other structures. Combining this intrinsic adaptability with the distributed nature of self-organized systems leads to one of their major advantages: *robustness* against failure and damage. There is no single point of failure, and the system can repair or correct damage without external help. In this way, a good self-organizing system will degrade gracefully rather than break down suddenly. Another important property is that many self-organizing systems show a high level of *scalability*, which means that the system still works if the number of entities is very large. For instance, the self-organized group formation of fish works well in schools with thousands of fish.

In summary, as illustrated in Fig. 1, self-organization can be defined as the emergence of system-wide adaptive structure and functionality from simple local interactions between individual entities.

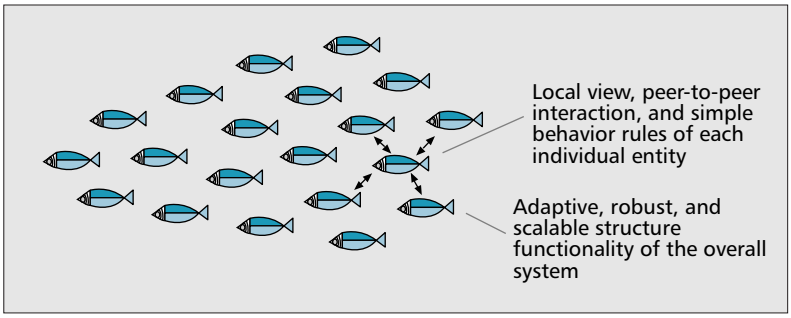

Figure 1. *Illustration of the main principles of a self-organizing system.*

Self-Organization in Communication Networks

Unsurprisingly, the notion of self-organization has also found its way into the area of communication and computer networks. The main goals are to minimize the need for configuration, develop protocols that facilitate network operation, and enable new types of communications networks, such as completely decentralized ad hoc and sensor networks [4–8].

A typical example of the emergence of self-organized functions is in the field of IP address allocation. Traditionally, an administrator configures each computer manually with an IP address from a specific address space that has been allocated to the administrator by a higher-level authority. This traditional approach has no elements of self-organization; instead, it requires significant human intervention and creates a very stiff address structure. A first step toward more self-organization was made with the introduction of the Dynamic Host Configuration Protocol (DHCP). Using this protocol, computers are able to automatically obtain an IP address from a server installed by the administrator in his or her domain. This allows computers to adapt to changes in their environment (e.g., to obtain a new IP address when they move to a different network). Although DHCP relieves users of configuring their IP settings, it still requires network administrators to install dedicated DHCP servers. This problem has been addressed with the standardization of IPv6 stateless autoconfiguration (self-configuration). Now a computer can query its current router for a subnet prefix and then form an IP address on its own: it simply combines the prefix with its lower-layer unique identifier (e.g., its medium access control [MAC] address). This state-of-the-art approach already includes many aspects of self-organization: no dedicated address server is needed, and communication is localized. The address allocation becomes more flexible and robust. Looking into the future, additional approaches to IP address autoconfiguration are under development. In ad hoc networks, for example, distributed autoconfiguration techniques are needed that deal with merging and separating networks caused by node mobility [9].

Another example of self-organization is the Transport Control Protocol (TCP), which implements a decentralized mechanism to handle congestion in the Internet. In simplified terms, TCP uses a control loop for the sending rate: it reduces the rate upon packet losses and increases it if packets arrive correctly. In this way, there is no explicit management of network resources, but these are shared among all participants in a reasonably fair manner. This example of traffic adaptation shows that decentralized control can be effective and scalable. Furthermore, it shows that

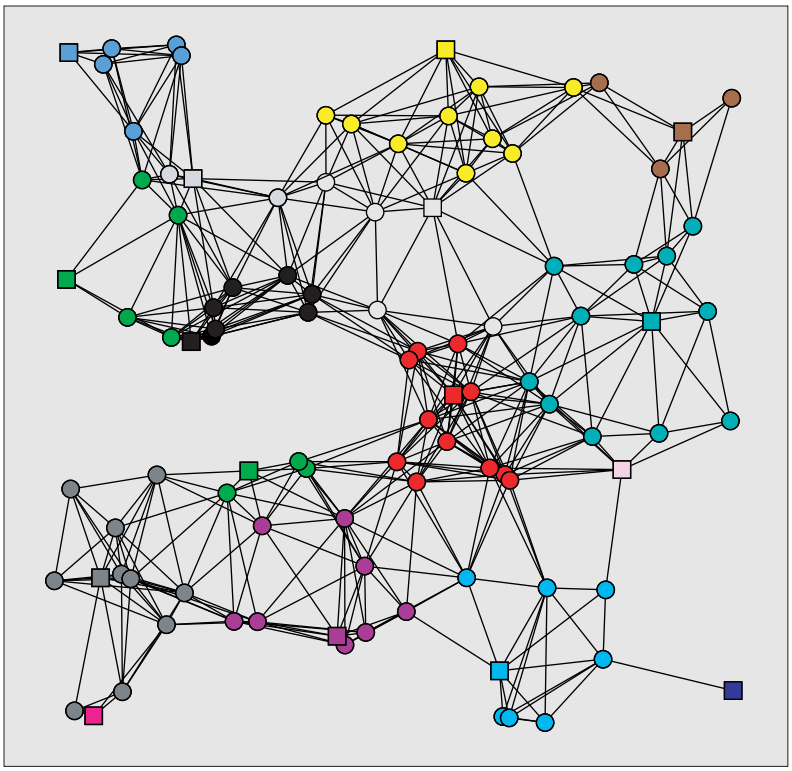

Figure 2. *Clustering of a network using Basagni's algorithm. Circles: ordinary nodes; squares: clusterheads; nodes of the same color belong to the same cluster.*

minimal information (local view of packet loss) can be sufficient. The fact that TCP does not behave well for sporadic packet loss on a wireless link does not diminish the overall concept.

Additional examples can be found in the area of mobile ad hoc and sensor networks. For instance, the Internet Engineering Task Force (IETF) approaches to ad hoc routing [8] give solutions for self-organized packet delivery. These protocols react dynamically to failures and node mobility. Using an on-demand routing protocol, each node has only a view of its direct neighbors and performs route searches to discover paths to further-away destination nodes. There are no central location registers as in cellular networks, and no proactive routing tables as in the fixed Internet. Besides routing, distributed algorithms for adaptive clustering, medium access, topology control, and many other functions have also been proposed.

To give another example, the notion of self-organization also appears in the context of failure resilience and network restoration [6]. Here, the main goal is to design "self-healing" or "self-stabilizing" networks that react to link and node failures or physical damage. Protocols are needed that detect such events quickly and reroute the affected traffic in a self-organized manner.

Design Paradigms for Self-Organization in Networks

Despite the progress in particular aspects of self-organized networking during the past years, some fundamental questions have not been addressed properly. What general principles do self-organized network functions have in common? Which principles from self-organized systems in nature and other fields can we transfer to communications systems? What are the design paradigms to build a self-organized network function? Answers to these questions are far from obvious. Most research on self-organization follows a descriptive approach: people have observed self-organizing phenomena in dynamic complex systems, and try to understand why and how these systems evolve from the interaction of simple entities. Applying self-organization to communication networks, however, requires a constructive engineering approach: for a given complex networking problem, we need to design the rules and protocols for interactions among the nodes.

Our goal is to take a step in this direction by proposing four design paradigms for self-organized networking. We believe that these paradigms characterize the key enablers of self-organization in communication networks.

Paradigm #1: Design Local Behavior Rules that Achieve Global Properties

We consider the problem of designing a network function that establishes a global property, such as unique addresses or connectivity. A centralized solution is to introduce a dedicated entity that is in charge of establishing the global property (e.g., an address server). The paradigm of self-organization, however, is to distribute the responsibility among the individual entities: no single entity is "in charge" of the overall organization, but each contributes to a collective behavior. Following this paradigm, we must design localized behavior rules that, if applied in all entities, automatically lead to the desired global property (or at least approximate it). By "localized" we mean that entities have only a local view of the network and interact with their neighbors.

Such an approach can be taken if we are able to reduce the desired global property to a corresponding local property. An example illustrating this concept is topology control in wireless multihop networks [10]. Here, the desired global property is connectivity of all devices, keeping energy consumption and interference at low levels. It has been shown that this global property can be mapped into a local property: if each node maintains links to its kth nearest neighbors, where k depends on certain parameters, the resulting network topology will be connected with high probability. To achieve the local property, each node adjusts its transmission power accordingly.

Another area where localized behavior leads to the emergence of global properties is clustering. The goal is to partition the network into clusters. As shown in Fig. 2, a desired global property can be described as follows: each node belongs to one cluster, each cluster has one clusterhead, and each node is at most h hops away from the clusterhead (here $h = 1$). The clusterhead is in charge of specific tasks (e.g., all intercluster communication). The problem is to define such clusters with purely local communication and without central control. Using Basag-

ni's algorithm [11], the local behavior rules are defined as follows. A node decides to join a cluster if there is already a neighboring clusterhead with an address larger than its own address; otherwise, it decides to become a clusterhead itself. After making this decision, the node informs all its neighbors of its role. Using this simple protocol, the network converges within a finite time to an organization that achieves the above mentioned global properties.

The key challenge in the design of such networking functions is to find local behavior rules that lead ("emerge") to the desired global property. In several cases, we cannot reduce the global property to a local one, but may use a divide-and-conquer concept: information is collected locally, aggregated, and then exchanged with other nodes. The locally available information can be aggregated in such a way that a full view of the network is not needed to compute a solution for the overall network. Such an aggregation principle is applied in distance vector routing protocols (e.g., the Routing Information Protocol, RIP). Each node periodically asks all its neighbors for the delay ("distance") from them to other nodes. At the same time, it also determines the delay to its neighbors. Using the values from the other nodes and its own measurement, it can determine the shortest route to a node in the network. If each node performs this process, globally optimal routes are computed in a distributed manner without knowing the complete topology.

In summary, reduction to a local view and local interactions is the basis for most self-organized systems. One may argue that localized algorithms often may not result in a global optimum with respect to a certain property but only provide solutions close to the optimum. This suboptimality, however, is not a real disadvantage in a dynamic network. Here, optimum configurations are subject to change frequently anyway, and fast convergence to a stable configuration is important.

Finally, note that locality also means that changes in the network have initially only local consequences. On one hand, this helps the network to become stable and robust with respect to changes and failures, because it does not have single points of failures and abrupt changes of the entire network state. On the other hand, the localized decisions may introduce some inconsistencies in the network state. For example, using local address allocation, two nodes might have the same IP address. This discussion leads us to the second design paradigm for self-organized networking: to tolerate and handle imperfect coordination.

Paradigm #2: Do Not Aim for Perfect Coordination: Exploit Implicit Coordination

In any kind of communication system nodes share resources such as bandwidth and addresses. To enable efficient and fair access to these resources, the nodes have to be coordinated in some manner. One solution is to *avoid* conflicts and inconsistencies between the nodes. For instance, we can allocate radio resources exclusively to a certain mobile user for the duration of a call or traffic flow. This can be achieved using *explicit* coordination; that is, signaling messages are exchanged in a request-response manner to coordinate resources. This style of coordination is typically used in centralized systems. Examples include signaling channels and centralized address databases in cellular telecommunications.

Such a priori conflict avoidance and explicit signaling, however, may cause significant overhead if the network is very dynamic and asynchronous. Here, the idea is to *tolerate* conflicts if we can manage them in a contained manner (i.e., conflicts are acceptable if they are localized, restricted in time, or easily detected and resolved). A well-known example is carrier sense multiple access with collision detection (CSMA/CD) in local area networks. Another example is passive duplicate address detection in ad hoc networks [9], which relaxes the ideal property of globally unique addresses but assigns a locally unique address to a node, and then detects and resolves address conflicts locally.

Taking this approach, the use of *implicit* coordination can be exploited. Implicit coordination means that coordination information is not communicated explicitly by signaling messages, but is inferred from the local environment. A node observes other nodes in its neighborhood; based on these observations, it draws conclusions about the status of the network and reacts accordingly. A typical example can be found in the MAC layer of wireless ad hoc networks. Suppose node A sends a message to node B, which in turn forwards the message to node C. If A overhears the message of B to C, it knows that B has received it from A. In other words, the overhearing of a message can serve as an implicit acknowledgment. To be more general, implicit coordination means that nodes detect and analyze the communication messages exchanged in their vicinity. They might, for example, notice periodicities in time or other traffic characteristics of other nodes and then adjust their own traffic according to this observation. This concept was already addressed by Capetanakis [12] in 1979, who observed that a certain sequence of collision events gives a node some information about the status of the network; so the node can infer which other node may want to access the medium.

An extreme form of implicit coordination is the concept of *zero* coordination. It avoids coordinating among nodes at all. To illustrate this, let us again consider the above medium access scenario. If each node transmits without any coordination with other nodes, communication works fine until two nodes transmit in the same channel at the same time. Then a resource conflict occurs, which requires a mechanism for conflict resolution. This mechanism can be achieved in a simple manner through the application of *randomization* along with timers. Each node waits a certain random time or switches to a randomly selected channel and then performs a retransmission. This Aloha approach is simple and efficient in situations with low and medium traffic loads. In fact, many self-organized systems in nature apply randomization to initialize the system and recover from errors or deadlocks.

Note that locality also means that changes in the network have initially only local consequences. This helps the network to become stable and robust with respect to changes and failures, because it does not have single points of failures and abrupt changes of the entire network state.

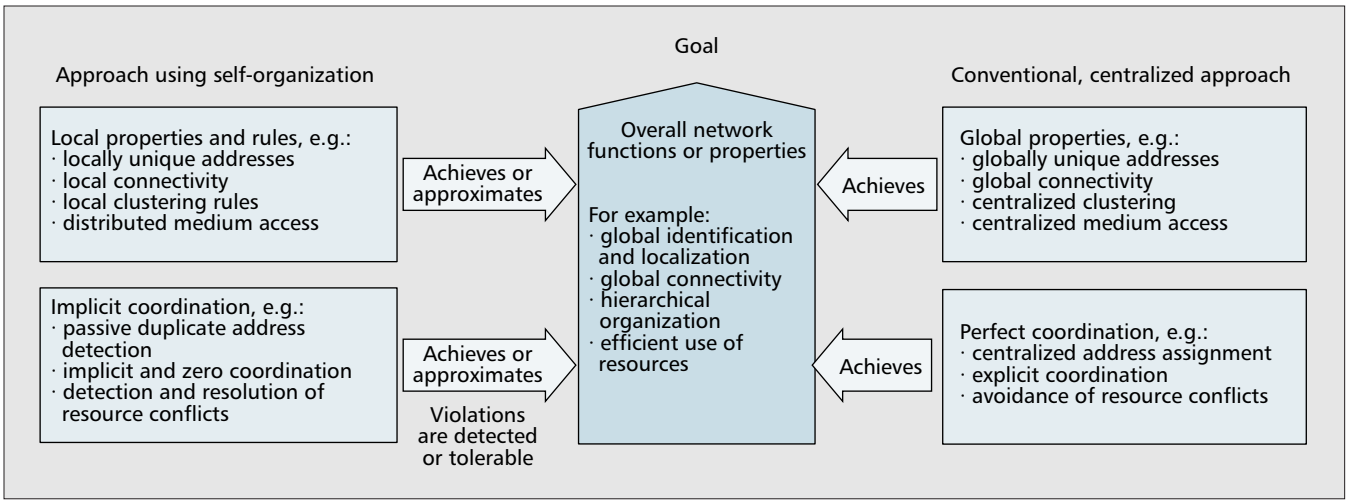

Figure 3. *Comparing self-organized and conventional networking.*

In summary, we propose to exploit implicit coordination as well as conflict detection and resolution methods to achieve resource- and time-efficient means of coordination among nodes in self-organized networks. Unlike perfect control, these methods can be performed in a very localized manner.

Relation between the First Two Paradigms

Before we proceed to the third design paradigm, let us put the first two paradigms in relation to each other (Fig. 3). Clearly, the goal of a communications network is to achieve efficient communication among its entities. This requires a number of network functions and properties, such as global identification and localization of entities, global connectivity, a hierarchical network structure, efficient use of resources, and others. In conventional telecommunications networks, these functions are usually achieved using global rules and perfect coordination. This approach is strongly motivated by a global view of the network, and thus implicitly favors centralized and ideal solutions. For example, the identification of a device is achieved by a globally unique address assigned by centralized address management. However, the same networking functions and properties can be achieved or approximated in a self-organized manner using the two design paradigms "local rules" and "implicit coordination." For example, global identification of a node can be achieved by a locally assigned, locally unique address if temporally and spatially restricted address conflicts are detected and resolved. In this way, a different technical problem is solved, but the ultimate networking goal is still achieved.

As shown in Fig. 4, it is instructive to position different protocols in a two-dimensional space whose axes represent the discussed paradigms: the level of locality (i.e., the spatial expansion of communication) and the degree of coordination among nodes. For instance, CSMA works with observation-based implicit coordination, which is restricted to the link local level. More coordination efforts are undertaken in wireless LANs and ad hoc networks, where explicit signaling messages are exchanged to avoid collisions (CSMA with collision avoidance, CSMA/CA). This exchange of coordination information also enlarges the vicinity in which nodes can overhear messages. It thus increases the spatial expansion of interactions. In time-division multiple access (TDMA), usually explicit coordination is performed by some control entity. Hence, both the level of coordination and spatial expansion of interaction are increased over CSMA/CA.

Paradigm #3: Minimize Long-Lived State Information

The third design paradigm relates to the state in the network. Many networking technologies require nodes and devices to store and maintain long-lived information about network state. For instance, each device in a cellular network has to store the addresses of dedicated network entities, such as location and security databases, voice boxes, and gateways to the Internet. This information must be configured manually by the user or administrator, or some network management. In addition to this *configured* state, which is individual for each device, there is also often a need to keep *synchronized* state, meaning that some information is stored consistently among several nodes and devices. Examples are routing tables or security data. Maintaining such consistency among nodes is difficult in dynamic networking environments, especially if nodes are frequently disconnected. Often some centralized entity is needed to give assurance that the state is up to date.

To achieve a higher level of self-organization, we should minimize the amount of long-lived state information. One approach is to employ *discovery mechanisms*, which nodes can use to obtain information about a certain network entity or service. We can distinguish between reactive (on-demand) and proactive discovery. In reactive discovery, a node sends a `Query` message via multicast or broadcast. If a searched entity receives this message, it responds via unicast. In proactive discovery, dedicated entities

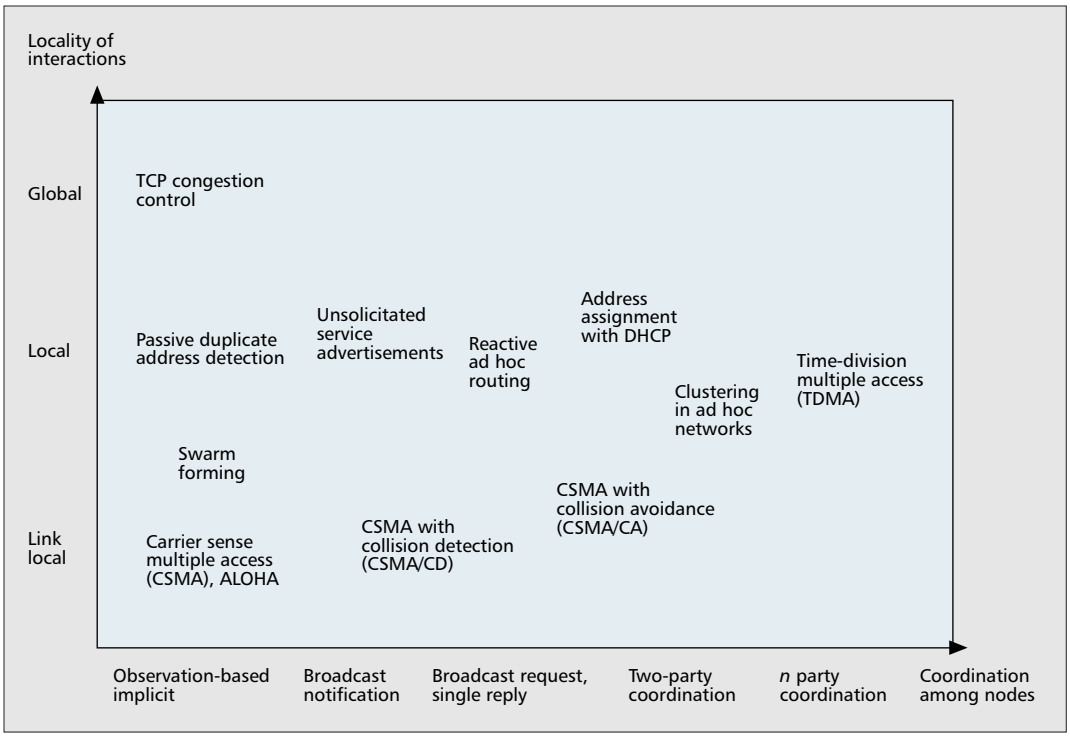

Figure 4. *Classification of protocols according to their level of locality and coordination among nodes.*

The more localized the interactions are and the less coordination is needed, the less state information must be maintained. For example, if nodes use implicit, observation-based coordination for medium access, they do not need to maintain long-lived state about the network.

send unsolicited `Advertisements` announcing their presence. If nodes receive such messages, they are informed of the existence, address, and type of service offered. Although state is maintained in such a discovery approach, it is refreshed regularly and is hence not long-lived. The network becomes more adaptive and more robust against changes and failures of entities. In addition, we reduce the assumptions made about other nodes in the network, which in turn removes dependencies between nodes.

The amount of maintained state information is related to the two above mentioned design paradigms. In general, the more localized the interactions are and the less coordination is needed, the less state information must be maintained. For example, if nodes use implicit observation-based coordination for medium access, they do not need to maintain long-lived state about the network. Also, if nodes of an ad hoc network use an on-demand routing protocol that employs a broadcast for route search and a unicast for reply, they only maintain short-lived state which is updated on a regular basis.

Paradigm #4: Design Protocols that Adapt to Changes

An important aspect is still missing in our set of design paradigms: the capability of nodes to react to changes in the network and its environment. The need for such adaptation typically arises from changed resource constraints, changed user requirements, node mobility, or node failures. Since there are no centralized entities that could notify the nodes about changes, each node has to continuously monitor its local environment and react in an appropriate manner.

Let us distinguish three levels of adaptation.

Level 1: A protocol is designed so that it can cope with changes, such as failure and mobility. An example of such a protocol is Basagni's mobility-adaptive clustering algorithm [11], described above.

Level 2: A protocol is designed to adapt its own parameters (e.g., value of timers, cluster size) as a reaction to changes in order to optimize system performance. An example of such a protocol is McDonald's clustering algorithm [13], which adapts the cluster size as a function of the observed level of mobility in the network.

Level 3: A protocol is designed so that it realizes if the changes are so severe that the currently employed mechanism is no longer suitable. For example, the dynamics of a network may be so high that a clustering algorithm no longer converges; it could then switch from cluster-based hierarchical routing to flooding. To detect such situations the environment is observed, and significant changes in major parameters trigger a fallback or alternative solution.

Typically, these levels of adaptation are combined using control loops. An example that covers two levels of adaptations is TCP. It adapts transmission windows and timeouts depending on round-trip time and delivery success. It reacts more drastically to consecutive packet loss and will stop transmission completely for some period of time. The basis is that packet loss indicates network congestion, and further sending will further aggravate this problem. After the timeout, it will resume normal operation.

There are also advanced levels of adaptation (e.g., learning algorithms) that effectively change the algorithms in the nodes. We do not include these algorithms here for self-organized networking, as their application in a dynamic net-

When it comes to the development of a networking function that should have a high degree of self-organization, it would be ideal to have a methodology describing this development in a step-by-step manner. It is very difficult to define such a design process in detail for the broad scope of self-organized network functions.

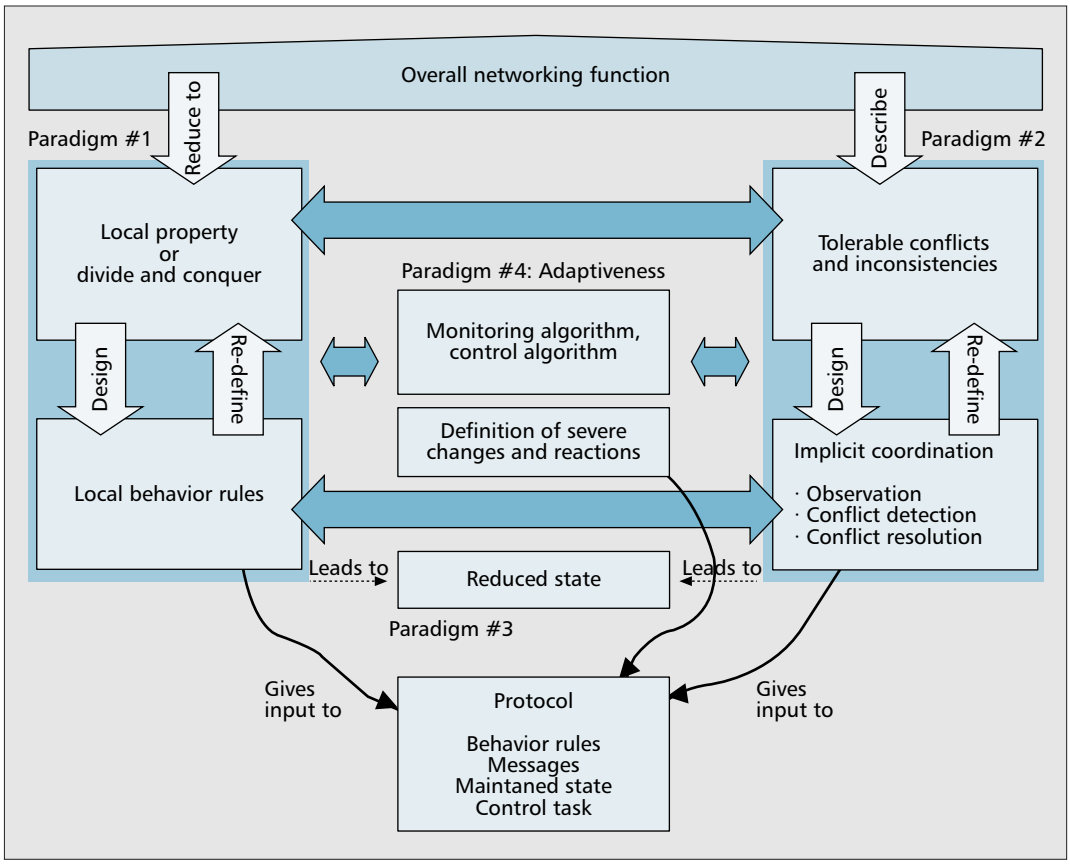

Figure 5. *A design process for a self-organized network function.*

work is typically quite complex and difficult to predict. In addition, dynamically changing algorithms might lead to difficulties in implicit communication, as the interpretation of the environment depends on the knowledge of their behavior.

Putting Things Together: How to Design a Self-Organized Network Function

When it comes to the development of a networking function that should have a high degree of self-organization, it would be ideal to have a methodology describing this development in a step-by-step manner. Clearly, it is very difficult — if not impossible — to define such a design process in detail for the broad scope of self-organized network functions. As shown in Fig. 5, we make a first attempt at such a process. It integrates the four design paradigms on a high level and applies them iteratively to obtain a protocol that implements the desired self-organized function. The following paragraphs explain this process and illustrate it for the development of a clustering scheme.

The protocol design process starts by describing the overall networking function to be achieved. We then apply paradigm #1 or #2. According to paradigm #1, we express the overall function in terms of a local property or a divide-and-conquer method. From this, we design a local behavior rule for each node. According to paradigm #2, we define a set of conflicts and inconsistencies that can be tolerated. Based on this, we design a method for implicit coordination, which should include methods for observation of the environment, conflict detection, and conflict resolution. It is likely that both paradigms are not designed independent of each other, but usually will require exchange and a number of iterations. Also, the question of which process step is performed first depends on the particular problem. In the design of clustering schemes, for example, we may start with paradigm #1: we first find a local property and clustering rules, such as those given in Fig. 2. Afterward, we employ paradigm #2, defining tolerable conflicts that improve the dynamic behavior of the algorithm. For instance, we may allow temporal inconsistencies in cluster membership. Finally, we adjust the local properties and rules according to these allowed conflicts. An example where the process might be performed in the opposite order is the design of wireless medium access. We first define tolerable collisions (paradigm #2), and then describe local properties and rules for collision detection and resolution (paradigm #1).

Paradigm #3, minimizing the maintained state, is not applied directly but is typically a consequence of applying the first two paradigms: locality reduces the globally synchronized state, and less coordination generally reduces the state needed for control. In the above distributed clustering scheme, for exam-

ple, nodes maintain state information about their direct neighbors only.

To enable adaptive behavior (paradigm #4), we propose to develop monitoring and control algorithms. The monitoring algorithm defines how frequently the environment is observed and which parameters are taken as input to the control algorithm. Based on this input, the control algorithm adapts certain protocol parameters. Here, concepts from classical control theory, such as feedback loops and hysteresis, should be applied to optimize control and avoid too frequent adaptations. In a networking scenario, the change of a parameter in one node may trigger an adaptation in other nodes as well. Hence, it is especially important to guarantee that the transient behavior of the network is stable, and no deadlocks or improper terminations occur. In addition, the limits of each control algorithm should be evaluated to define a set of severe changes that trigger level 3 adaptability. Should such a severe change occur, the protocol will change to different local behavior rules and/or a different method of implicit coordination, or even change the local properties or tolerable conflicts.

Conclusions

Ubiquitous communication will bring an increasing heterogeneity of technologies and an increasing networking complexity and dynamics. To master these challenges, it is beneficial to exploit principles for self-organization. This enables spontaneous, autonomous networks among mobile devices, but it also helps conventional network operators reduce the administrative need and complexity in network installation, maintenance, and management. This is especially important since the increasing technological complexity might become a stumbling block for future development. In turn, the introduction of self-organized functions has the potential to reduce costs and improve the robustness of the network. These benefits are not free and require different thinking: self-organization is also about giving up control over the network and letting it organize itself as much as possible.

To come a step closer to this vision, this article gives an introduction to self-organization in communication networks and proposes four design paradigms. These paradigms were derived by looking at different protocols and extracting common features that contribute to self-organized networking. We believe that these paradigms represent basic building blocks to achieve self-organized functionality. Finally, we suggest a process for the design for self-organized functions. We hope that these contributions will stimulate further research in this evolving field.

References

[1] W. R. Ashby, "Principles of the Self-Organizing Dynamic System," *J. Gen. Psych.*, 37, 1947, pp. 125–28.
[2] H. Haken, *Synergetics: Introduction and Advanced Topics*, Springer, 2004.
[3] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities and Software*, Scribner, 2001.
[4] S. Staab, Ed., "Neurons, Viscose Fluids, Freshwater Polyp Hydra — and Self-Organizing Information Systems," *IEEE Intell. Sys.*, vol. 18, July 2003, pp. 72–86.
[5] T. Robertazzi and P. Sarachik, "Self-Organizing Communication Networks," *IEEE Commun. Mag.*, vol. 24, Jan. 1986, pp. 28–33.
[6] W. D. Grover, "Self-Organizing Broadband Transport Networks," *Proc. IEEE*, vol. 85, Oct. 1997, pp. 1582–1611.
[7] J. P. Hubaux *et al.*, "Towards Self-organizing Mobile Ad-hoc Networks: the Terminodes Project," *IEEE Commun. Mag.*, vol. 39, Jan. 2001, pp. 118–24.
[8] C. E. Perkins, Ed., *Ad Hoc Networking*, Addison-Wesley, 2001.
[9] K. Weniger and M. Zitterbart, "Address Autoconfiguration in Mobile Ad Hoc Networks: Current Approaches and Future Directions," *IEEE Network*, July 2004.
[10] D. M. Blough *et al.*, "The *k*-neighbors Protocol for Symmetric Topology Control in Ad Hoc Networks," *Proc. ACM MobiHoc*, Annapolis, MD, June 2003.
[11] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proc. Int'l. Symp. Parallel Architectures, Algorithms, and Nets.*, Perth, Australia, June 1999.
[12] J. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," *IEEE Trans. Info. Theory*, vol. 25, Sept. 1979, pp. 505–15.
[13] A. B. McDonald and T. Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks," *IEEE JSAC*, vol. 17, Aug. 1999, pp. 1466–87.

Biographies

Christian Prehofer (prehofer@docomolab-euro.com) is manager of the Self-Organized Ambient Networking group at DoCoMo Euro-Labs, Munich, Germany. He also lectures at the Technical University of Munich. He is working on QoS for IP networks, ad hoc networks, and self-organized networking technologies. He obtained his Ph.D. and habilitation in computer science from the Technical University of Munich in 1995 and 2000, respectively. From 1998 to 2001 he held positions as system engineer and software architect in the area of networking and mobile devices. He is the author of more than 60 publications and has contributed to several standardization bodies.

Christian Bettstetter (bettstetter@docomolab-euro.com) is a senior researcher at DoCoMo Euro-Labs, Munich, Germany. He works on networking aspects of ubiquitous mobile communication with focus on wireless ad hoc and sensor networks. He received his Dr.-Ing. degree summa cum laude from the Technical University of Munich in 2004, and his Dipl.-Ing. degree in electrical engineering and information technology from the same university in 1998. He co-authored the Wiley book *GSM: Switching, Services and Protocols*, and received the 2004 outstanding paper award from the German Information Technology Society. He is an editor of *ACM Mobile Computing and Communications Review* and serves on the organizational committees of ACM MobiCom and MobiHoc.

The introduction of self-organized functions has the potential to reduce costs and improve the robustness of the network. These benefits are not free and require a different thinking: self-organization is also about giving up control over the network and letting it organize itself as much as possible.