Complexity of Multicast Communication Patterns and its Application in Smart Grid Networks

Nora Berg

February 27, 2014

Contents

1 Introduction

Complexity in a network can result in instabilities and unpredictability. Reasons for that is the human inability to predict all the situations which can emerge through cooperation of network participants. To avoid unpredictable behavior in a network it would be favorable to compare different network designs and their complexity on an abstract level before implementing them. However, putting network complexity into numbers is not easy. It emerges from various influences and can change over time. For example, failures due to complexity can arise after functions were added to initially lean network designs and from overloaded design goals. In these cases some of the design goals might influence or even object each other. Complexity is also increased by functions which are misplaced in the layering model. In this case goals are ensured on lower levels and higher level applications always pay the price for the increasing complexity, even if they do not need these functions. On the other hand, if important functions are not included on lower levels, the application might need to provide many functions, which do not originally belong to the design of the network application.

Since the complexity of all networks embrace a huge range of possible failures and influences, which are not even comparable, the focus in this work is on multicast networks. Multicast network algorithms have all the same goal in distributing data to a subset of nodes and are therefore comparable. To describe the complexity of a multicast network a definition of complexity is needed. However a definition of network complexity is dependent on the viewer. For example it is dependent on which level the network is observed. An application developer will define network complexity different from an network operator. A definition of network complexity aims at a method to put complexity into numbers. It distinguishes between views on a network and it declares what such a method should exactly measure. A method measuring complexity should distinguish between the different aspects which influence complexity and balance these aspects. Ideally this method would give an indication about the failure cases emerging through network complexity.

In this work an introduction to network complexity and its problems is given followed by the a section about the different aspects of network complexity. Subsequently some problems which occur when trying to quantify complexity are discussed. In the following section different forms of multicast distribution tree topologies and their influence on multicast complexity is shown. The influences of network operations are introduced more fine grained in section 6 and why they can be used to estimate complexity. In the last section an overview of smart grids is given and why smart grids are a promising environment to explore multicast complexity.

2 Network Complexity and Related Work

Analyzing network complexity means examining the characteristics which make a network unpredictable, its behavior hardly comprehensible and understandable, and lead to instability of the network itself. It is distinguished from code complexity which measures software code but not the underlying concept and to runtime complexity which measures the efficiency of algorithms achieving their specific goal. In distinction increased network complexity may result in suboptimal behavior because of the unpredictability of the network and therefore does not fit in one of these definitions. Although too much complexity has negative consequences some complexity is needed to achieve a certain robustness (compare figure 1 and [4]).

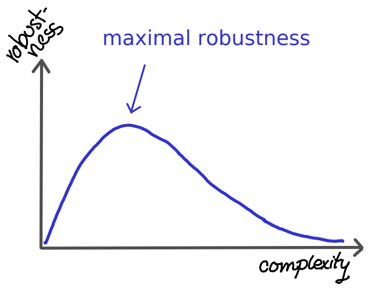


Figure 1: The robustness decreases when a certain complexity level is exceeded.

In the end we want a stable running network which is easy to maintain and to debug and a method for measuring complexity which should predict this kind of stability. For this goal we assume that network complexity is one of the keys. There is a number of key aspects which make a network complex. Which of these aspects are most important for defining complexity depends on the context of the network looked upon.

An ideal definition for network complexity should cover these expected key aspects in an abstract form and is applicable to a broad range of network applications. Until now there exist several definitions for network complexity:

Network Complexity is proportional to state, dependencies between components, and rate of change in a network. Too much complexity can cause unpredictable, non-linear behavior.

M. Behringer[2]

Network Complexity may be defined as the state of the network being hard to separate, analyze, or solve in case of a deviation from its intended, pre-determined behavior of its components interacting.

Burkhard[2]

The goal is to find a method to estimate the undesirable behavior described in these definitions. Such a method would help to avoid complex network design by predicting the complexity before implementation. The definitions above encompass a lot of different aspects and this makes them vague. From these definitions there is no easy way to decide whether a given network design is (too) complex or not. For finding such a method several steps are necessary. The first step is to extract the key aspects of networks complexity. Subsequently finding common factors of these aspects and as last step checking if these factors apply to other networks.

Approaches to grasp complexity often evolve in cooperation with research areas in mathematics, biology and chemistry and adopt their presumptions. So these results do not fit to computer networks completely but have common subsets. Nevertheless, there are some important conferences which include network complexity in their focus. The IRTF-Meetings with the Network Complexity Research Group (NCRG) are active on this topic, also the IFIP Networking Conference included network complexity as one of the topics in the call for papers and in the USENIX Symposium of Network Design and Implementation (NSDI) appeared some different papers.

Recent approaches to analyze network complexity can be roughly divided into three different classes, based on the aspects they observe. These classes are (i) complexity of the distributed control plane, (ii) complexity of the network topology, (iii) and complexity of the local stacks.

On the control plane the NetComplex metric [8] argues that complexity arises directly from the effort to keep the control plane states in sync. This analytic metric distinguishes between network algorithms and estimates the propagated control messages of each.

The robust yet fragile behavior is often observed on the control plane. Robust yet fragile behavior emerges if a network is designed against one specific set of failures. These specific design results in fragile behavior against another set of failures. A simple example for the robust yet fragile behavior, is to design a network and using only UDP to distribute the data, to increase the network performance. It will be fast but not reliable. The stress impacted by changes in the network is another key point to observe, because the more configuration is necessary the more configuration can be wrong.

On a higher abstraction level the paper “Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures” [1] observes the robustness of a network as a main factor for network complexity. It includes the robust yet fragile paradigm as the main factor of increasing complexity and describes the complexity arising from the organization and constraints of a network. It argues, that the efforts of making a system robust introduces new concepts and ideas which work transparently but also introduce new failure probabilities which are hard to find. Furthermore the paper introduces a terminology for network complexity components as protocols and constraints within the network. The impact of the robust yet fragile paradigm as well as the terminology are explained with examples from technology and biology.

Within the Network Complexity Research Group (NCRG) of the IRTF, several tradeoffs at configuration level of a network are worked out [14]. These tradeoffs show how the goal of robustness often results in a contradiction to other network goals as e.g. optimal forwarding paths, failure domain separation or reactivity of the network.

More mathematical approaches observe complexity from the topological point of view. As a graph theoretical approach Van Mieghem [18] describes the complexity of graphs as the amount of shortest path trees which span a single graph. Furthermore he calculates the efficiency of using multicast over n-times-unicast by estimating the expected hopcount to distribute multicast data.

Another topological approach [10] uses graph characteristics (e.g. branching) as base for the Shannon entropy. With this approach it is possible to define classes of complexity of specific trees.

Complexity depends on many factors so another approach aims to include several complexity factors from topological point of view as well as from the control plane point of view and measure the overall network complexity with multiscale entropy [17] [9]. To achieve this, several elements are defined to represent the change of the network and observed over the time. The multiscale entropy is calculated upon these elements.

In the end a key for defining and measuring network complexity is to analyze certain aspects as mentioned in the definitions above (dependencies between components, rate of change, ...) in a more manageable environment. One of these more manageable environments are multicast networks, whose complexity can be seen as a case study for network complexity. In this context also the observation of the multicast algorithm itself is considered.

The approaches of the related work analyze mainly the control plane and topology of the network. In the multicast context another question arises: If all node behave to a specific control loop which failure cases can occur? And how do they differ due to distinct multicast algorithms? The NetComplex Metric has a comparable approach. It distinguishes between several algorithms and analyzes their control plane states. The problem with this approach is, that it only considers the messages which keep the network in sync. But complexity has more facets as e.g. which failures can occur and how to detect them. This clearly influences the algorithmic behavior but is not predictable, so a pure deterministic approach is not sufficient to analyze the behavior of an algorithm in a network. Furthermore an additional problem case is how to include the topological view of multicast distribution trees (e.g. represented with probabilistic approaches) into the multicast algorithm analysis.

In distinction this work introduces a differentiation between complexity aspects and their dependencies. It concentrates on control and forwarding states as well as control loop operations and dependencies between complexity aspects. It discusses these aspects and their dependencies in form of the rate of change and control loop complexity.

3 Aspects of Network Complexity

The complexity of a network can be observed from different points of view, where complexity emerges from different properties. Each of these perspectives affects the overall complexity and robustness, and additional interfere with each other. Furthermore they can result in different statements about complexity even if the same network is observed.

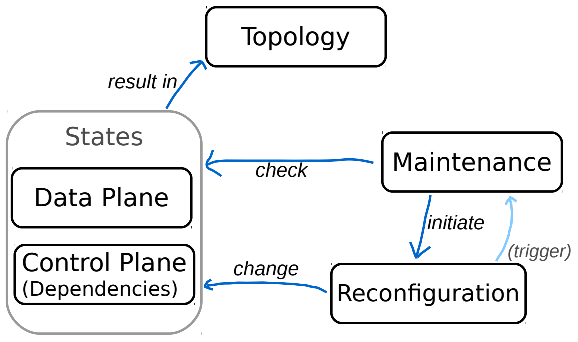


Figure 2: The informational dependencies between the complexity components

Figure 2 shows the dependencies between the complexity aspects and how they influence each other as follows:

Topology: The topological viewpoint observes the graph of a network and its graph theoretical properties. It focuses on how the graph looks like and which changes can occur without considering boundaries introduced by network algorithms. The topology of a network can be observed from two different angles. The forwarding states result in one topology, while the underlying network topology is the prerequisite for the observed network algorithm and can be completely different. Afterwards is referred to the former because it introduces its own set of complexity while the latter is assumed to be transparent and only visible as link characteristics. For example graph theoretical complexity can be measured by the number of shortest path trees in a graph [18].

Data Plane States: The data plane states (also called forwarding states) hold the information to which node data is forwarded. It also includes the information about next hops used for forwarding a message whose destination is several hops away. It does not include how these information are distributed or calculated. The states represent the view a local node has about the network. The states are used by the data plane to forward the data messages and they determine the topology of the network. The complexity influencing factor is e.g. the amount of the states and they indicate the scalability of a network.

Control Plane States (Dependencies): The dependencies in a network are the information on the control plane which have to be in sync at a node to keep the network properly functional. As in [8] the complexity of an information which comes from other nodes requires intelligent forwarding and processing than local information. The dependencies determine which states have to be maintained but not necessarily how these information are processed and forwarded. Furthermore the control plane states are the view of a local node over a network. Complexity influencing factors from control plane states are e.g. the amount of distinct control information which a node receives and sends into the network, the knowledge of an rendezvous point or an calculation of the range from which nodes information is needed.

Maintenance Operations: The maintenance operations process control plane information locally. They check states and dependency information, send periodically maintenance information and can initiate reconfiguration operations in the network. At local nodes, the maintenance operations are implemented within control loops and the control loops decisions (e.g. forwarding of control information) depend on the control plane states (dependencies). Furthermore maintenance operations can be triggered by reconfiguration but are not dependent on them. A complexity influencing factor is how many possible different results a check of states (control or data plane) can have or how easy and reliable they can be determined

Reconfiguration: Configuration and reconfiguration operations are the operations which change the topology of a network and states of a node. They are initiated by maintenance operations and include the question how necessary information due to a specific reconfiguration operation is obtained. This depends on the states (data and control plane) and results in a number of changed states at the local node as well as at remote nodes. An example for a complexity influencing factor from reconfiguration operations is how many states are changed due to an received reconfiguration message.

Since all these aspects impact the overall complexity of a network they should be considered when aiming to measure complexity. However, a metric considering only one of these aspects can be used to predict weaknesses specific to their origin.

4 Problems of Quantifying Complexity

The goal of quantifying complexity is to make the network complexity measurable. For this reason, the complexity must be represented in an easier form, e.g. a number. A notation like this must satisfy certain critera, that the underlying complexity network is reasonable represented. One criteria is whether one network is more complex than another then the method should represent this. These form also have to match some criteria which are required from the mathematical point of view. A first approach to put a number to complexity could be a metric.

Metric

A mathematical metric is defined as follows:

Definition: $d : M \times M \to \mathbb{R}_{\geq 0}$
A metric maps two objects of a given set to a non-negative real number, which represents the distance. A metric fulfills several criteria:

Triangle Inequality: $d(x, z) \leq d(x, y) + d(y, z)$
The distance between two single objects can never be shortened by taking a detour over a third object.

Symmetry: $d(x, y) = d(y, x)$
The distance from one object x to another object y is the same distance as on the way back from y to x.

Positive definite: $d(x, y) = 0 \Rightarrow x = y$
If two objects have no distance, then they are the same object.

The first problem is the decision what the "object" in this problem space would be. Several approaches are thinkable as using the network topology which is created, or the amount of control loop information subject to a given underlying network topology. This also leads back to the question which are the main factors of a network that influence complexity and how they influence each other. Apart from these consideration there are some difficulties with the approach of mapping the complexity to a mathematical metric.

The first problem is that it is a measurement between two objects. This does not perfectly fit to the goal comparing two networks and to state which is more complex. A mathematical metric would rather describe how similar the complexity of the two networks are, but would have no statement how complex the networks really are. Nevertheless the calculation for one object could be implemented in a metric by using a mathematical norm. A mathematical norm is the distance from one object to a zero point.

So the questions arises how the most uncomplex network would look like to be a zero point network.

From a topological point of view, the behavior of complexity in simple graphs indicates how the simplest network should look like. One question is whether complexity increases with the count of links in a network. As in figure 3, a full mesh is obviously more complex than a graph network without any links. This indicates an increasing complexity with the number of links. But this is not necessarily the case from other aspects (cf. Sec. 3).

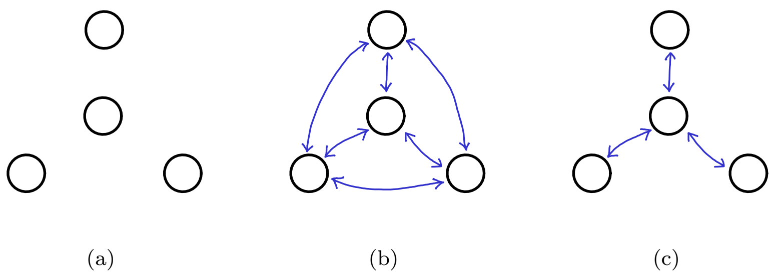


Figure 3: Although the complexity of a full mesh (b) is higher then of a network without links (a), a network which requires the distribution of forwarding information (c) is the most complex.

From the control plane viewpoint a network maintaining a tree structure between the nodes (fig.3(c)), is intuitively more complex than a full mesh. The reason for that are the control plane states (dependencies) in the network which have to be included. The amount of data plane states in a full mesh is at its maximum, however the messages reach their destination in one hop. No further forwarding information has to be distributed and maintained and therefore the complexity of a full mesh from the dependency point of view is considered as low. Most networks have a tree like topology to increase the scalability by reducing the states per node and to increase the efficiency of the data distribution. The control plane information about the location of a node has to be distributed and evaluated to create the forwarding state for every possible destination node in the network. The forwarding states at the nodes finally form the links in the network topology. Because of the additional control plane states which contain the information which hop to use to connect to another, the tree shaped form is considered the most complex from the control plane state point of view of these examples. These consideration would also be supported by other metrics about complexity based on configuration states [8]. However from the data plane state point of view the full mesh might be considered the most complex.

The effort to create and maintain the forwarding states, are introduced by the maintenance and reconfiguration point of view due to the specific network algorithm. Nevertheless it is obvious that maintaining and creating a tree requires more complexity than a full mesh. So from these points of view the tree is also more complex than the full mesh.

Hence simple facts such as number of nodes, links or node degrees from a graph theoretical network description are not sufficient as factor for a network complexity metric. The algorithms themselves have to be included, because they define how the forwarding states are established and therefore which topology emerges.

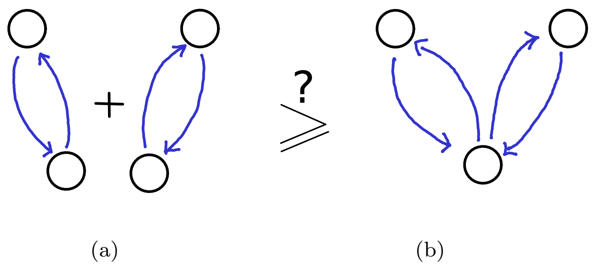


Figure 4: The triangle inequality is not necessarily matched

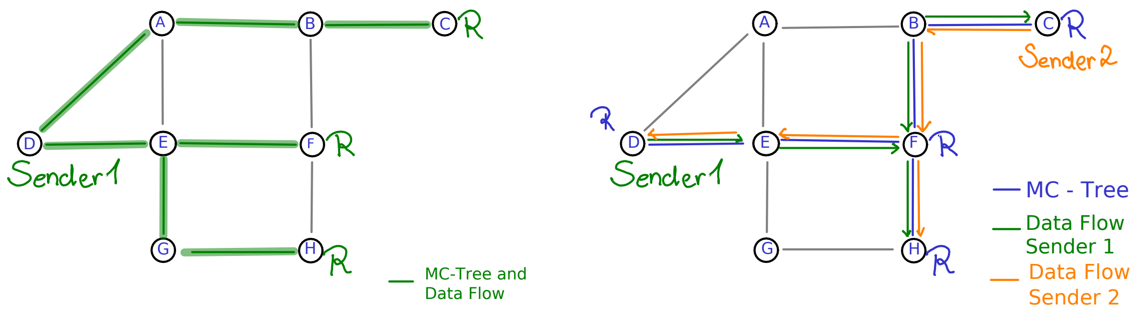
Furthermore the triangle inequality is not fulfilled (cf. [15]) by all complexity aspects . The example in figure 4 shows two networks combined. In figure 4(a) the lower nodes are the same node available in two different networks and in figure 4(b) it combines the two networks. The triangle inequality requires that 4(b) is less complex then the two networks in 4(a). But this is not the case, because forwarding states are required at the lower node in figure 4(b). As seen before tree like structures require more complexity. Thus in this case the triangle inequality is not necessarily satisfied.

These examples show that complexity cannot be easily expressed in a mathematical metric and other forms should be considered. This could be e.g. mapping network algorithms into classes with specific properties which produce a specific kind of failure.

An approach to cover differences in topology which are created by network algorithms can be to use expectation values similar to Van Mighem in [18]. To predict which failure cases can occur in networks further analysis of the network algorithms and their impact on the topology is required. Conferring to the complexity aspects in section 3 this means analyzing the maintenance and reconfiguration operations.

5 Multicast Tree Configuration

Multicast provides certain advantages as case study for network complexity. All multicast algorithms have the common goal to distribute data efficiently to a set of receivers. Nevertheless these algorithms differ in the way they build the distribution tree and the subgoals (e.g. reliability,...) they achieve. Based on the uniform multicast context, the common goal to distribute data and the common approach in building a distribution tree, complexity between multicast algorithms is comparable. The distribution tree itself is the topological view of multicast algorithms. There are two main classes of multicast distribution trees which differ in their tree creation and complexity.



(a) (b)

Figure 5: (a) source specific and (b) bidirectional shared data distribution trees

A source specific multicast tree anticipates only one sender (Figure 5(a)). In this case the distribution tree algorithm always creates the shortest paths from every node to the one sending root node. This kind of tree would be suboptimal if there are multiple sending node. Using the same tree would result in suboptimal elongated paths, overstress the root node and therefore decrease the scalability.

In contrast a bidirectional shared tree for data distribution from multiple sender will always aim to have the shortest available path length from every sender to every receiver without overstressing a certain node or a certain link (Picture 5(b)). The creation of a source specific distribution tree requires the information about the sender location to build an optimal tree. In contrast bidirectional shared trees need to distribute information about every sender to decide how to build an efficient tree. Additional the single nodes need forwarding states dependent on the sender to ensure an efficient data forwarding. This shows that the class of bidirectional shared tree algorithms are more complex than the source specific trees. Moreover it supports the idea of dependencies on other nodes information being a main factor for complexity in multicast networks (as in [8]).

Another question arising from this observation is how the dependencies on other nodes information and the information about sources and listeners are distributed. The decision of the local nodes what to do in the network is controlled by local control loops. They determine the behavior of a node, especially when to initiate certain network operations, how to react on certain received information and which control information to forward. These control loops are designed against a characteristic set of failures due to their specific multicast algorithm. The problem is, that these control loops cannot be designed against every possible failure case and unexpected behavior emerges. So another, more fine grained complexity influencing factor is the amount and behavior of the local control loops and how their decisions impact the multicast network.

6 Complexity Characteristics in Multicast Algorithms

The reconfiguration and maintenance of the distribution tree are important aspects to the complexity of a multicast network (cf. Sec. 3). To understand how complexity emerges it is useful to compare the (re)configuration and maintenance of existing distribution tree algorithms and evaluate in which cases they get complex or fragile.

Rate of Change

The rate of change is the average change of states on control and dataplane. Thereby network states are not only forwarding states but every assumption a local node holds in its view over the complete network. Different multicast algorithms react differently on specific events and this is what we wish to compare. Before estimating the overall rate of change in a network, it is necessary analyze the impact of a single operation in the multicast network. More precisely to identify the amount and location of state changes which follow single operations. The rate of change is not only limited to the impact of standard multicast operations (like *join* and *leave*) but also includes the initial triggering of maintenance operations by control loops and the subsequent change of states. In multicast several configuration operations provide the basic functionality (e.g. *join, leave, send, receive*). Additionally every multicast uses its own control operations as e.g. *, choose_cluster_leader, prune* and many more. The following example for the impact of an operation on the multicast network, uses the *join* operation on a basic IP-multicast level. *Join* is used to announce that a certain node is interested in the messages of a certain group. During the join operation a multicast distribution network establishes forwarding states at already joined neighbor nodes. This forwarding states point on which node a received message is forwarded. The change impacted by a single join operation is dependent on the location and the time a node joins the multicast network.

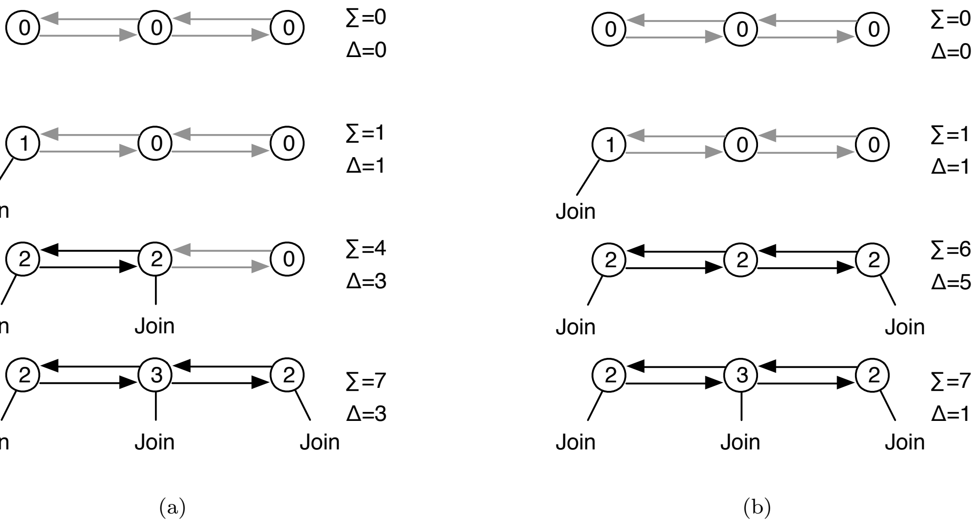


Figure 6: The change of data plane states impacted by a *join* operation is dependent on the location of the node.

Figure 6 shows the sum of the forwarding states as $\sum$ and the change of the forwarding states as Δ. The circles represent routers which establish forwarding states and the string *join* indicates that they enable a forwarding state for a joining node. In figure 6(a) the second joining node is near a node which already has forwarding states for another joined node, and the count of new forwarding states is relatively small ($max(\Delta) = 3$). In figure 6(b) the second joining node is two hops away from a node which already has forwarding states. In this case all nodes on the path to the existing distribution tree add forwarding states for this joining node. The maximal change occuring in this scenario is $\Delta = 5$ when the second node joins. This example shows that a change impacts the node in a specific range. It is not only interesting how much impact a change has in sum, but also the minimum and the maximum change a operation can cause. Furthermore it is interesting how far the range of this change is, and therefore a distinction if a operation causes many changes in the neighborhood or fewer operations in a wider range. Both options can cause different kinds of failures even if they result in the same sum of changes. As changes in networks need time and entail a certain error probability the rate of change can be used to estimate how often a network gets in an inconsistent state and in which range the error impacts consequences.

Control Loop Complexity

Control loops at local nodes in a network check the local states at a node as a maintenance operation. As in Figure 6, specific states and state combinations lead to an reconfiguration operation, after which the states are checked again. The states represent the local view of the network and the control loop represent the behavior inside this network. To create a robust network an often used approach is to think of every error which can possibly occur and then design a method to solve this error. The robust yet fragile paradigm states that designing against one set of failures makes vulnerable against another set of failures [3]. A reason for this is that every control loop check which is performed locally is an assumption about the state of a different part of the network. More assumption and therefore more distinct solution strategies (implemented as reconfiguration operations) increase the amount of distinct errors which occur if this assumption is wrong.

Within the control loop two distinct kinds of error can occur. The first kind of error emerge on the layer where the multicast algorithm is implemented itself. In this case some inner state combinations were not considered at algorithm design time and lead to unpredictable behavior. Emerging from the algorithm itself these kind of failures are rare. The other kind of failures originate in the interaction of different layers and therefore different control loops. Robustness means that the multicast algorithm can deal with failures as lost

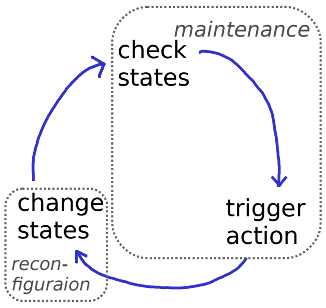


Figure 7: The control loop complexity includes maintenance and reconfiguration elements.

and delayed control messages, with a changing underlying topology and with lost connections to specific nodes. It leads to the question how many local states can be false until a specific rate of data messages cannot be delivered and the multicast functionality becomes useless for upper layers. Other implicit results which emerge through this kind of fragile behavior are splitted distribution trees or nodes which loose the connection to the whole tree.

Timing dependencies to underlying layers can lead to failures which are amplified by complexity. In multicast algorithms due to timers is decided which states a node obtains (e.g. membership query in IGMP [11]). In a network stack the lower layers are assumed to be transparent and therefore the time performance is not visible. The control plane states depend on the control loop timers to wait a considerable time, before the assumptions are justified. So we can assume that the amount of states which are not matching increases with the variation of the underlying network performance. This is amplified by the amount of synchronized control plane states.

Within the control loop there are two possible ways to cope with failures. The first one is to design an explicit error handling for specific failures and the second one is to create a fallback operation which resets the states (e.g. a new *join*). Which one is chosen is a question of detectability of the failures, cost and efficiency. Also the error handling clearly influence the rate of and should aim to keep the rate of change as low as possible.

So in summary two main question arise: How much changes the network due to the reconfiguration operation? And how many maintenance and reconfiguration messages can be delayed or lost until the multicast distribution gets useless? Answering these questions can lead to an estimation for the robustness of a multicast network in dependence to the control loop.

7 Smart Grid as a Case Study for Multicast Complexity

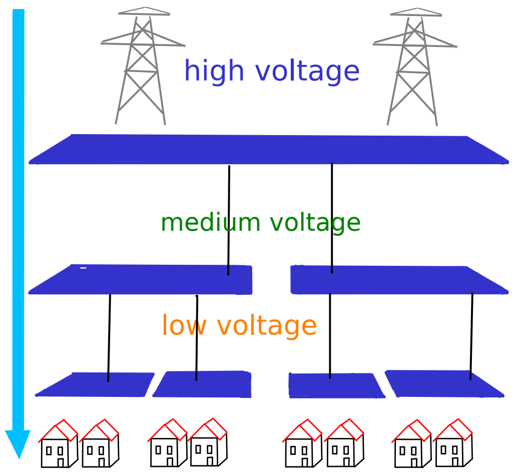


Figure 8: Classic electrical grids comply to a top down structure.

Smart Grids provide a promising environment for observation of multicast complexity. The following section consists of an short introduction to smart grids, how multicast can be used in smart grids and why smart grids are a promising field for observation of multicast complexity.

Smart grids are developed with the goal to optimize power providing and power consumption in electrical grids. The traditional electrical grid complies to a top down structure (as in figure 8). Few big power plants insert the generated power into high voltage transmission grids which are used to transport the electricity on long distances towards the cities. The electricity grid splits up into medium and low voltage areas which distribute the electricity to the households. In the last years the power generation moves from few large power plants to more small distributed power plants. Photovoltaik panels are attached to roofs, families get combined heat and power machines in the cellars and regenerative energy sources are combined to virtual power plants. These power sources are included into the electrical grid and provide additional power if the power is not used locally. For saving money and from rising ecological awareness there is an increasing amount of intelligent devices in the electrical networks. The demand side management in houses aims to minimize the electricity bill and performs energy expensive tasks if the energy price is low. Other additional devices come with the advanced metering infrastructure providing smart meter for communication between the household and the power company. Smart meter allow a more fine grained logging of the power consumption and more flexibility in pricing. Furthermore they are able to provide information from the households to improve the grid stability.

There are two main problems emerging from this increasing amount of small power plants. The distribution grid is created for a top down energy flow with few large power producers has now to deal with bidirectional energy flows. This leads to several electrical engineering instabilities which have to be handled. The second problem is that the small power plants are not as reliable as the traditional ones, because they depend on wind, sun and other natural sources. The main task of the smart grid is to adjust itself flexible to the current demand and distributed power supply and provide a stable electrical flow. Both of depend on different aspects as e.g. daytime, weather and day of the week. To achieve a stable grid the intelligent devices have to announce their demand and provided energy.

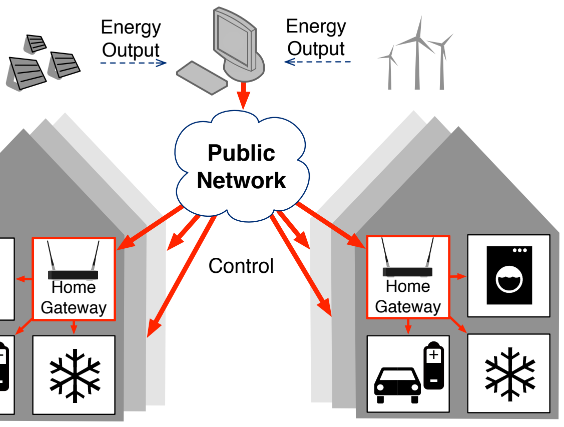


Figure 9: Intelligent devices can adjust their power consumption to the current power resources.

Nearly every household is connected to the internet and the devices get more intelligent and communicative. This can be used to handle the communication of maintenance and power demand information by already installed home gateways e.g. home routers [12]. Furthermore communication in smart grids benefit from multicast because nearly every thinkable communication is group communication. Source specific distribution trees appear at price updates from the power companies. They are supposed to reach every household in a specific area and come from one source (compare figure 9). Many-to-many communication and therefore a use case for bidirectional shared trees occurs at the announcement of available but unused energy from a local solar panel. Home gateways often already provide multicast functionality.

The efficiency of multicast data distribution is especially important in the smart grid environment because of the huge amount of devices and houses which communicate and cannot be handled with n-times-unicast. In smart grids the distinction of endpoints into specific groups results from the environment. Groups occur locally as e.g. a street or logical e.g. specific power sources or all households which need much energy at the moment. For example if a household is interested in charging its car at a cheap price and until 8.00 AM in the morning, the home gateway initiates the charging process at a cheap moment. The latter example shows the occurring of many constraints, considering the daily life of people, e.g. driving to work at 8:00 AM.

Technically in smart grids are two networks to consider. The household itself which optimizes its own energy consumption in dependency of the second network, the larger electrical grid outside which is also changing in a flexible way. Both of them have different constraints in availability of devices, security and available network communication.

The change from traditional power grids to smart grids also introduces security challenges. They base on two opposite main goals in the grid. The power companies want to sell energy and the energy consumers want to save money. The common goal between these two is to protect the stability of the grid. Another problem is not to reveal too much personal information of the household to the community. It is unfortunate, if the whole neighborhood can see from the distributed announcements if somebody is at home or not. So the devices in one households can trust each other more then the devices outside in the grid.

In conclusion in this example scenario exists much multicast communication between a huge amount of households, home devices and the electrical companies in form of the grid devices. Furthermore a need for both sorts of distribution trees (bidirectional any source and source specific) appear in this environment. The communication in smart grids has to be robust and secure because electricity is an elementary need of the daily life. One way to keep an multicast communication robust is to prevent it from growing too complex and becoming unpredictable. This smart grid multicast scenario is described in [12]. The main conference for communication in smart grids is the IEEE SmartGridComm. But also on other conferences the communication aspect ist present as in the IEEE Innovative Smart Grid Technologies Conference (ISGT) and on the ACM Symposium of Applied Computing (SAC). Furthermore smart grid technology is also discussed in several conferences on electrical engineering and has therefore a different focus.

8 Summary

Highly complex networks tend to result in unpredictable and instable behavior. Nevertheless some complexity is needed to build a robust network. At the moment there is no reliable method to which indicates how complex a network really is and which impacts results from the complexity. Several problems have looked upon to find such a method. A problem in finding such a method is that classical mathematical approaches like *metrics* might not work in the case of network complexity and other approaches should be considered. In addition it is hardly possible to measure complexity in a abstract way for all networks because there are too many influences.

Multicast is selected as an manageable case study to extract key aspects which influence the complexity. The goal is to use these key aspects to predict the robustness and failure probability of a multicast network. The rate of change and the control loop complexity seem to be observable in this context. A first approach to put the rate of change into numbers is the observation of changes in the forwarding states. Another approach is to analyze logical algorithmic decisions and therefore states of local control loops, which represent assumption over the rest of the network. One basic idea is, that the local view of a network being wrong is one key aspect producing failures. So the goal is to calculate the expected wrong control loop states and subsequently how many wrong states a multicast network can bear.

To observe the behavior of complex multicast networks with a certain level of diversity smart grid networks are chosen. The optimization of smart grids requires a lot of communication is required. In a specific smart grid communication approach multicast is used providing different scenarios and a large amount of members and groups. Hence this is a promising environment to explore the behavior of complex multicast algorithms.

References

[1] D.L. Alderson and J.C. Doyle. Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(4):839–852, 2010.

[2] Michael Behringer. Network Complexity - The Wiki. `http://networkcomplexity.org/`, December 2013.

[3] Michael Behringer and Geoff Huston. A Framework for Defining Network Complexity. Internet-Draft – work in progress 00, IETF, October 2012.

[4] Michael H. Behringer. Classifying Network Complexity. In *Proceedings of the 2009 Workshop on Re-architecting the Internet*, ReArch '09, pages 13–18, New York, NY, USA, 2009. ACM.

[5] Nora Berg, Sebastian Meiling, Thomas C. Schmidt, and Matthias Wählisch. Untersuchungen zur Komplexität komponierter Netzwerkarchitekturen im Future Internet. 7. GI/ITG Workshop Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und verteilten Systemen (MMB-net11), Hamburg, Germany, Sep 2013. Universität Hamburg, Dept. Informatik.

[6] R. Bush and D. Meyer. Some Internet Architectural Guidelines and Philosophy. RFC 3439, IETF, December 2002.

[7] Ross Callon. The Twelve Networking Truths. RFC 1925, IETF, April 1996.

[8] Byung-Gon Chun, Sylvia Ratnasamy, and Eddie Kohler. NetComplex: A Complexity Metric for Networked System Designs. In *5th USENIX Symposium on Networked Systems Design and Implementation*, USENIX NSDI'08, April 2008.

[9] Madalena Costa, Ary L Goldberger, and C-K Peng. Multiscale entropy analysis of biological signals. *Physical Review E*, 71(2):021906, 2005.

[10] Matthias Dehmer, Stephan Borgert, and Frank Emmert-Streib. Network Classes and Graph Complexity Measures. In *1st Int. Conf. on omplexity and Intelligence of the Artificial and Natural Complex Systems, Medical Applications of the Complex Systems, Biomedical Computing*, IEEE CANS'08, pages 77–84. IEEE, 2008.

[11] William C. Fenner. Internet Group Management Protocol, Version 2. RFC 2236, IETF, November 1997.

[12] Sebastian Meiling, Till Steinbach, Moritz Duge, and Thomas C. Schmidt. Consumer-Oriented Integration of Smart Homes and Smart Grids: A Case for Multicast-Enabled Home Gateways? In *Proc. of the 3rd IEEE Int. Conf. on Consumer Electronics - Berlin*, ICCE-Berlin'13, pages 279–283, Piscataway, NJ, USA, Sep. 2013. IEEE Press.

[13] A.-H. Mohsenian-Rad, V.W.S. Wong, J. Jatskevich, and R. Schober. Optimal and Autonomous Incentive-based Energy Consumption Scheduling Algorithm for Smart Grid. In *Innovative Smart Grid Technologies (ISGT), 2010*, pages 1–6, January 2010.

[14] Alvaro Retana and Russ White. Network Design Complexity Measurement and Tradeoffs. Internet-Draft – work in progress 00, IETF, August 2013.

[15] Matthew Roughan. Network Configuration Complexity. Presented at The Critical Internet Infrastructure (Dagstuhl Seminar 13322), August 2013.

[16] Thilo Sauter and Maksim Lobashov. End-to-End Communication Architecture for Smart Grids. *IEEE Transactions on Industrial Electronics*, 58(4):1218–1228, April 2011.

[17] Rana Sircar and Michael Behringer. Using Entropy as a Measure for Changes in Network Complexity. Internet-Draft – work in progress 00, IETF, October 2013.

[18] Piet Van Mieghem. *Performance Analysis of Communications Networks and Systems.* Cambridge University Press, Cambridge, New York, 2006.