# An Approach to Quantify Reconfiguration Methods for PIM-SM Trees
# - A Network Complexity Perspective -

Nora Berg

Februar 28, 2015

## Contents

# 1 Introduction

High complexity in networks can lead to unpredictable behavior. Reasons for that lie in the number of network protocols and participants, which may interact in an unforeseeable manner. At local nodes, the control loops react on a defined set of events, and possibly trigger a local change of state or outgoing messages. With a growing number of states and varying reactions, the complete state space becomes hard to observe. The problem increases, if the protocols do not only react on their own set of messages, but also interact with other layers of the network stack.

Nevertheless, a certain amount of complexity is required to provide robust networks. Some exchanged information are required to maintain connectivity and failure states need to be balanced out. The typical network complexity phenomenon is that with increasing complexity, the robustness increases until a certain point the robustness does not increase further, or even decreases. This happens, if failures emerge from unforeseen combination of states at different locations in the network. In each protocol, the combination of states is different, therefore every protocol introduces its own level of complexity.

One of the reasons for unpredictable behavior is that all combinations of states cannot be thought through at the design time of a network algorithm. The reason is that analyzing all combination of states leads to a state explosion. For the same reason, it is not feasible to analyze network complexity of a network algorithm by analyzing all state combinations. Instead, we approach complexity by measuring its effects on *robustness* on the network. The reason is that changing robustness and resilience is a common effect of a large number of issues introduced under the term network complexity [13]. Resilience is the ability of a network to adapt to changes. Robustness is the ability of a network to persist despite changes and errors. These two aspects overlap insofar as robustness can be achieved by resilience. In our approach robustness is measured over the change introduced by repair mechanisms of the network algorithm. The change is measured in the sum of stated changes in the network. States hold the local view of a node over the network.

In the upcoming work, a goal is to find out, whether robustness can be approximated by calculating the change in the network which follows a failure. *Protocol Independent Multicast* (PIM-SM)[11] is chosen as a case study, because it is a popular multicast protocols and its shortest path tree can be approximated by the well understood graphs of uniform recursive tree.

We base our calculations on uniform recursive trees (URT) to approximate the change of the PIM-SM repair mechanisms in the network. Furthermore, we try to to approximate the number of failures it takes to render a network unusable. In the best case, the approximation over the state changes can be used to quantify robustness. A further goal is to check whether these calculations can be done for other network algorithms and whether they can be the basis of a meaningful metric for robustness.

Our approach to calculate the robustness of PIM-SM networks consist of four steps. First we approximate the number of nodes that are separated from the subtree if a random link fails. Second we quantify how much change in the network is caused by reconnecting them. Third, we check how much change it takes to render the network unusable. In the last step the approximations need to be evaluated with tests. The current state of the work is in the first step.

This work is structured as follows. Section 2 reviews the field of network complexity, its challenges, and the connection to our approach. In Section 3, some basics about network complexity and robustness from the network algorithm point of view are summarized which are the foundation of the following approach. An introduction to distribution trees in PIM-SM and their reconfiguration methods is given in Section 4. Theoretical fundamentals of this work, are shown in Section 5. Section 6 introduces the current ideas of quantifying the PIM-SM reconfiguration method. The outline for the upcoming work is shown in Section 7. Section 8 shows the risks, chances and the transferability of the approach and is followed by the summary and outlook in Section 9.

# 2 Challenges in Network Complexity and Related Work

Network complexity is a relatively wide spread research field. It includes many issues which concern different kinds of observers of a network. The Network Complexity Research Group (NCRG) of IETF base their approach to define network complexity on this observation [5]. The problem space of network complexity are shown by the listed aspects of complexity. These are for example the states and dependencies in a network, churn in a network, but also the total cost of ownership.

Furthermore the NCRG approached a metric for change in network by combining several sources within a multi-scale entropy [14]. It proposed to define several entropy metrics for the different sources of complexity and to apply them to a single network to get an overview over the complexity. For example this metrics would include the complexity of topology, as well as the complexity of the control plane state, the complexity for traffic and optimal forwarding state, and also the complexity for cost and human impact. This approach shows the wide range of complexity. Because of the size of the field, the NCRG made only very slow progress, until the group was shutdown last year.

Other research in this area is done by the European Network of Excellence in Internet Science (EINS), which emphasizes the importance of interdisciplinary aspects of network research [1]. The group addresses the wide range of the field by distinguishing the work into several working groups. The working groups cover for instance the theoretical background, critical infrastructure, or privacy and identity in the Internet.

For the theory of complex networks a roadmap document was published that summarizes several promising theoretical approaches for gaining insights over complex networks [13]. It emphasizes the relation of the network complexity research field to other fields such as biology, physics, economics, or sociology. These disciplines also work on complex networks and have their own approaches for different properties as emergent behavior or evolution of resilient structures. The document names as promising theoretical approaches, among other things, statistical physics, agent systems with autonomous decision making modeled by e.g. decentralized partial observable markov decision processes, and also graph theoretic approaches over large scale graphs. The latter highlights the result of scale free network being robust against random damage and vulnerability of targeted attacks. Furthermore, it names the influencing properties of large scale graphs on complex networks as an active domain of research and points out the modeling of graphs through statistically generating processes.

In the work outlined here, we concentrate on the special graphs of multicast distribution trees. One of the advantages is that the analysis of multicast algorithms over the change of the distribution tree is not done yet, to our knowledge. Furthermore, the subset of multicast networks is chosen, because several kinds of networks require several kinds of generating processes. In most cases multicast distribution trees are shortest path trees and can be modeled by URTs [17] which are described by a statistical generating process [10, 16]. Since most multicast algorithms maintain the same kind of tree, we can compare the complexity of their method of maintaining it. For that we analyze the change in the network which is introduced by the repair methods.

The goal is to gain insights over the change in a network, after a random link fails. The occurrence of random failures and its effect on the connectedness can be analyzed via percolation theory, which is also emphasized by [13] and was successfully used for analyzing the decay of other networks, e.g. interdependent networks in [7].

In contrast to the development of the pure theory as proposed in [13], this work concentrates on the usage of theoretical approaches to analyze the robustness of specific network algorithms. For that, we choose to analyze the repair mechanism and robustness of PIM-SM as a first study case. After this is done, the calculations can hopefully be transferred to other algorithms and at last an abstract way of calculating the robustness of multicast algorithms can be developed.

## 3 Network Complexity and Robustness

Network complexity is a generic term, which describes how fine grained the logic within a network algorithm is. It refers to the logic within a network node as well as to the interaction between different nodes, and the interaction between different layers in the network stack. The term network complexity needs to be distinguished from runtime complexity, which quantifies the efficiency of algorithms and how many resources (cputime or memory) it takes to accomplish a specific task.

Even if there is no generally accepted definition of network complexity yet, it is known to affect a variety of issues, e.g., robustness, scalability, or security [5]. Depending on the interest of the network observer, there exist several metrics for single aspects, for example software implementation complexity as number of classes. A common denominator for a large number of these aspects is, that they affect the robustness [13]. Since repair mechanisms are responsible for robust networks, we quantify robustness on the basis of reconfiguration methods of PIM-SM.

The behavior of a network algorithms is implemented in a local control loop of a node. As shown in Figure 1, a control loop includes several categories, to which the components of network algorithms comply [6]. A control

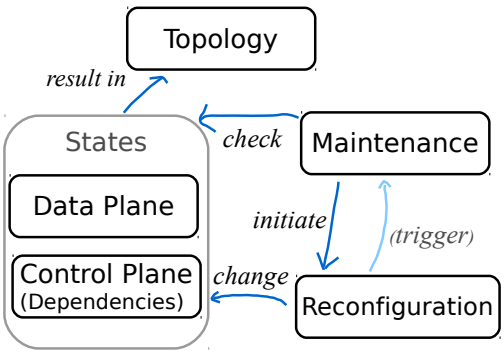



Figure 1: Control loops consist of several components

loop with all its components exist within one layer of the network stack. Sometimes, as in PIM-SM, they use also information from underlying layers.

The states can belong to the data plane or the control plane. Data plane states determine the data flow, on the routing layer these are they are the Forwarding Information Base (FIB) or its PIM-SM multicast equivalent, the multicast information base (MFIB). Control plane states hold configuration information over the network and are used to update data plane states. They include information which are necessary to configure the network or the network stack. On the routing layer the Routing information Base (RIB) includes all possible forwarding options. The multicast equivalent is the Multicast Routing Information Base (MRIB). To construct the data forwarding, the MFIB is generated from the best options in MRIB. Furthermore PIM-SM runs on the basis of the underlying unicast routing information. So a MRIB is created based on the respective RIB and contains in general neighbor routers that support PIM-SM.

The sum of all data plane states shows the application data flow in the network, which, in the case of multicast, are the distribution trees. The maintenance functions check for emerging irregularities within the states. For example, a data forwarding state from MFIB points over a link whose control plane state from MRIB says that the connection over this link is lost. If the maintenance functions discover such inconsistencies, they trigger a reconfiguration function. The reconfiguration functions are responsible for balancing out the irregularities. This includes gathering additional data, reacting on incoming messages, and changing the respective states. In this example, the triggered reconfiguration method would recalculate the best link for data forwarding and set the data plane state in the MFIB respectively.

Since robustness depends on the repair mechanisms of a network, we choose to analyze a PIM-SM repair mechanisms, which are part of PIM-SMs reconfiguration methods. The goal is to find out, whether a metric over robustness of a PIM-SM network can be derived. For that, it will be quantified, how much change is introduced into the network through the *rejoin* reconfiguration method of PIM-SM. The rejoin method is responsible to reconnect nodes back into the multicast network after they became separated, e.g., by a failing link. The change is measured in the sum of all changed states in the network. The number of states which are changed by a rejoin procedure affects the ability of a network to repair itself. The reason is that it takes some time until the affected nodes learn and react on change in the network because the change is distributed through the network by "slowly" spreading information. If there is a too large number of frequently failing links, or in network terms, high churn in the underlying network layers, then the network algorithms (e.g., PIM-SM) do not have enough time, to adjust to these failures because it takes too long to enable all the states. If this happens, the network cannot fulfill its purpose anymore. So the level of churn in the network that can be balanced out defines the robustness network we want to quantify.

High robustness requires that the reconfiguration methods converge efficiently and enable correctly working forwarding states at the routers. Here lies the connection to network complexity. If the complexity is too high, then the logic is too fine grained. A large number of small inconsistencies in states may trigger reconfiguration methods and lead to further change within the network, instead of repairing it. The level of robustness decreases. So the number of state changes seems an important factor which affects the robustness.

On the other side, if we have insufficient complexity, the mechanisms to overcome failures may be too rough and inefficient themselves. For example this is the case by distributing data through flooding. It helps to overcome a high rate of churn, but it's not robust towards the number of messages.

We quantify robustness towards the churn of the network by quantifying the reaction to change in the network.

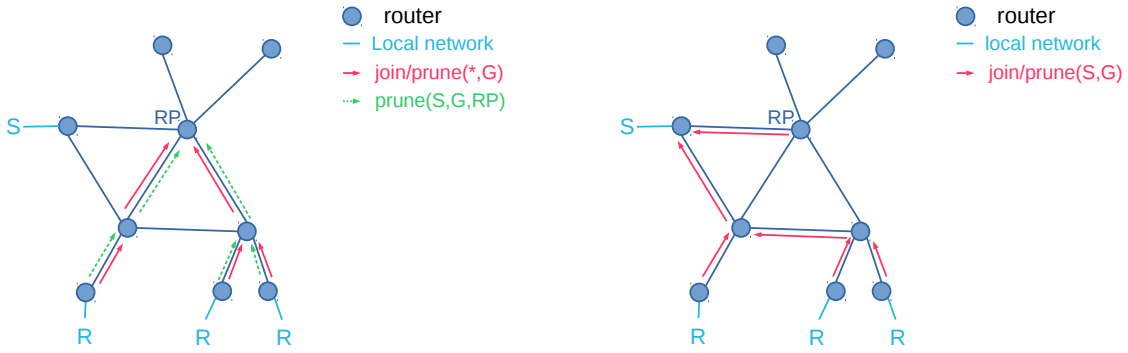
# 4 The Trees of PIM-SM

The Protocol Independent Multicast - Sparse Mode (PIM-SM) is a routing protocol for delivering data efficiently to multiple receivers [11]. It is a widely implemented IP multicast protocol and used between routers. Furthermore, it is designed to work independently from underlying routing protocols, so it can be applied across many different routing domains. Thereby it includes the information of the underlying unicast routing database.

The purpose of PIM-SM is to create efficient multicast data distribution tree, without any prior required knowledge about the senders at the receiver nodes. Therefore the protocol maintains two kinds of trees, which are created in three phases.

In the first phase a *shared tree* is created. Since a router has initially no information which sources belong to the group, it joins the shared tree. As shown in Fig. 2a, the receiver sends a `join(*,G)` message, which is forwarded to a well known Rendevouz Point (RP) of the group. At each hop, a state is enabled, to forward data to the joining receiver. The join message is only forwarded to the next hop, if the router did not already joined the source tree. A sender of multicast data just sends its packet with a multicast group address into the network. Every multicast enabled local network elects its *designated router* (DR). The DR responsible to encapsulate the message and send it as unicast packet to the RP. The RP drops the data into the shared tree. Then, the data is forwarded in the opposite direction of the join message via *reverse path forwarding* (RPF).

In the second phase of the protocol, the RP joins the source-specific tree of the sender. For that is sends a source-specific `join(S,G)` message into the network. Analog to the shared tree join of receivers in phase one, the




(a) the shared tree is necessary to initiate efficient data distribution through the source trees

(b) a source tree is always joined by the rendezvous point

Figure 2: PIM-SM maintains two kinds of trees

message will be forwarded in the direction of the source and enables reverse path forwarding states that point from the sender to the RP. When the first data arrive over the source-specific tree, it sends a `register-stop` message to the DR, so that the DR can stop sending messages in an encapsulated form.

In the third phase of the protocol the source-specific shortest path trees between the receivers and the sources are created. The DRs of at the receiver networks learned the sources over the data messages they received over the shared tree. Now, the DRs at the receiver networks join the source-specific trees the receivers are interested in. As shown in Fig. 2b, to join a source for a specific group, the DRs send a `join(S,G)` message in the direction of the source. The message is forwarded to the source analogously to the `join(*,G)` messages to the RP in the first phase and enable analog forwarding states. When the data start to arrive over the source-specific tree, the DR sends a source-specific `prune(S,G,RP)` message into the shared tree. This message travels up the shared tree and enables states which prevent that the data of this source is forwarded down the shared tree to the pruning router. So the amount of data which travels through the shared tree stays relatively small. The DRs stays in the shared tree, as long as one local receiver is interested in the group. This is independent from the number of sources which are pruned in the shared tree. The staying in the shared tree enables that sources can join at a later time and the DR will notice it, because its data will be delivered over the shared tree.

In PIM-SM, the reconfiguration in failure cases is done via join and prune messages. In addition, all the affected trees must be repaired. So the number of trees, as well as the join and prune behavior, is key for the complexity of PIM-SM. On this basis the overall change in the network will be approximated in the upcoming work.

## 5 Theoretical Basics for Approximations

Some theoretical basics are needed to quantify robustness. The following properties and ideas cover four aspects, which will be used in combination for our approach of quantification.

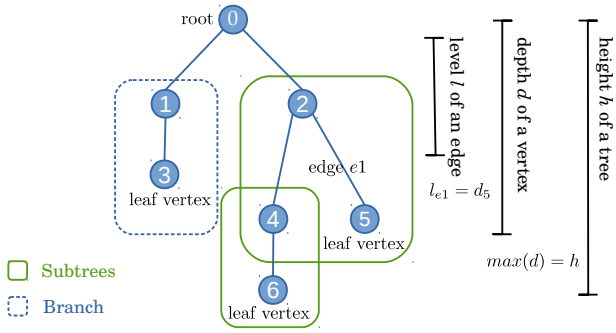



Figure 3: Reference for used terms about trees

Figure 3 gives an overview over the graph theoretic terms used in the following sections. A *rooted tree* is a graph theoretic tree with one root vertex [9]. The *depth* $d$ of a vertex in rooted tree, is the number of edges between the vertex and the root. We define the *level* $l$ of an edge, is the depth of the incident child vertex of the

edge. The *height* $h$ of a tree is the number of the maximum depth of a vertex. A subtree is a subgraph of a tree, which is also a valid tree. A *branch* is defined a subtree which is separated if an edge on level 1 is removed [10]. So all branches are subtrees, but not all subtrees are branches.

## 5.1 k-ary Trees

In the ongoing work, perfect k-ary trees are used to test ideas and calculations before transferring them to URTs, whose structure is described in form of probability values. In addition, calculations based on perfect k-ary trees gives upper bounds for properties of trees with a maximum degree $k$ and the same height. K-ary trees are graphtheoretic rooted trees, whose vertices do not have more than $k$ child vertices [12].
Perfect k-ary trees are trees, where each vertex has zero or $k$ child vertices and each leaf vertex has the same depth [15]. An advantage of k-ary trees is, that calculation are relatively easy. Furthermore, calculations over perfect k-ary trees of height $h$ can give upper bounds for calculations trees over random trees with a maximum degree of $k$ and height $h$. For perfect k-ary trees the number of nodes in the network can be calculated with $N(h) = \frac{k^{h+1}-1}{k-1}$ in proportion to height $h$. The relation between height and number of nodes in a perfect k-ary tree is used in section 6.1 to approximate the size of subtree which is cut off if a random link fails.

## 5.2 Uniform Recursive Trees

The structure of shortest path trees that are derived over graphs with a large number of nodes can be mathematically modeled in form of Uniform Recursive Trees (URTs) [17, 10]. Since PIM-SM creates shortest path trees, we use URTs as underlying structure for further calculations. An advantage of URTs is that their properties have been widely studied and there is a broad range of insights about them. They are also used to model multicast shortest path trees in other works (e.g., [17]). The information in this section summarizes results from [10, 16].

A uniform recursive tree is generated by a specific process over several steps. As initialization, we have an empty graph. In the first step, the root vertex with label 1 is added to the graph. In each next step, a new vertex will be added and connected to one of the previous vertices, which is chosen with uniform probability. So in step 2, vertex 2 is added to the graph and connected to the root vertex. In the third step, vertex 3 is connected either to the root vertex or vertex 2. For each of these two vertices the probability of becoming the parent for vertex 3 is $\frac{1}{2}$.

In step $n$, the new vertex $n$ is connected to vertex $m \in \{1...n-1\}$ with probability $\frac{1}{n-1}$, whereby the new link is independent of the previous links. The generated tree is a URT and denoted as $\mathcal{T}_{\mathcal{N}}$.

A characteristic of the URT is that branches of a tree have the same structural properties as the complete tree. They are just smaller URTs. So the same calculations can be used for the branch as for the complete URT, with adjustment to the number of vertices. This is the *recursive* property of URTs, and it enables us to do calculation just for the root vertex and apply them at any vertex in the tree, independent of its depth.

## 5.3 Stability of a Multicast Tree

In "Stability of a Multicast Tree" [17] it is quantified, how many links are added or removed in a shortest path tree when the number of receivers changes by one. It introduces a formula which approximates the change of the multicast distribution tree $E[\Delta_N(m)]$ depending on the network size $N$ and number of receivers $m$. The advantage of this approximations is that it gives the *range* in the network which is affected by the join method. Furthermore it is shown that this change approaches a Poisson distribution for large N.

## 5.4 Percolation Thresholds in Network Decay

The robustness as a function of errors in the network is quantified by percolation theory. The percolation theory for networks [4, 7] describes the ability of networks to resist a relatively large number of failing nodes and remain a connected graph. But if the number of broken nodes exceeds a specific threshold, the network will not decay slowly. Instead, it will break into a large number of small unconnected subgraph very fast. This threshold is called *Percolation Threshold*. By the removal of vertices or edges it is distinguished between random failure and targeted attacks on important nodes.

For example, the shape of the graphs influence the percolation threshold due to the type of edge removal. Typical shapes of graphs are for example e.g. Erdös-Renyi random graphs (ER) or scale free networks. While ER graphs are completely random, scale free networks have an exponential node degree distribution [8]. Scale free networks are relatively resistant against random failures, since the probability that one of the high degree node which hold the networks together is relatively low. So scale free networks have a relatively high percolation threshold against simple random failure of nodes in comparison to ER graphs. In a scenario with targeted attacks a scale free networks has a relatively low percolation threshold, since there are only a few large degree nodes. If the high degree nodes are removed first, the network will decay fast.

So the percolation threshold is a descriptive characteristic of the robustness of a network depended on the network structure and the kind of failure. In the further work, random failures of node or links are assumed.

# 6 An Approach to Calculate the Size of Separated Subtrees

The information from the fields of the previous section enables us to gain insight in the range of failure and the cost of rejoining the multicast network. We want to calculate the sum of states which are changed by reconnecting a separated subtree after a link failed. To achieve this, we start by calculating how many nodes are separated if a random link fails. From this we get the the expected number of nodes that need to be reconnected. In the worst case, all of this nodes would start a rejoin procedure and all of them would rejoin on distinct paths. The next steps of the calculations are outlined in Section 7.

The first approach to calculate the expected number of rejoining nodes on the basis of k-ary trees and URTs is described in this section.

## 6.1 Approximation of Subtree Sizes in k-ary Trees

In this section we calculate the number of separated nodes after the removal of a random link in a perfekt k-ary tree. For simplicity, we consider only failing links. Note, that if a node fails several subtrees get separated. However, the number of nodes which get separated if a *node* fails and the number of nodes which get separated if the incident *link* (in root direction) of the same node fails, just differ by one node. Since we are only interested in the number of separated nodes for now, we consider only failing links. We assume, that every link can fail with uniform probability.

Let $X_{k,h}$ be the random variable which maps from a random chosen edge in a perfect k-ary tree of height $h$ to its level $l$. The probability, that a link is damaged in a perfect k-ary tree of height $h$ in level $l$ is therefore

$$P(X_{k,h} = l) = \frac{k^l}{N(h) - 1} = \frac{(k-1)}{k^h - 1} \cdot k^{l-1} \tag{1}$$

whereby the well known function $N(h) = \frac{k^{h+1}-1}{k-1}$ gives the number of vertices in a perfect k-ary tree of height $h$. This probability is an exponential function with a static coefficient, since k and h are static numbers in a tree.

The height of the separated subtree $h_s$ is directly connected to the level $l$ of the failing link with $h_s = h - l$. So the random variable $Y_{k,h}$, which maps from a randomly chosen edge to the height of the separated subtree, has the same probabilities as (1). The probability for the height of a separated subtree $1 \leq h_s \leq h$ is therefore:

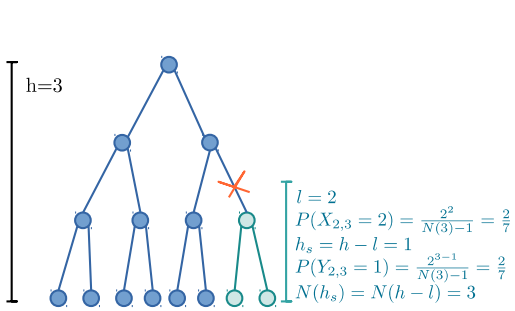
$$P(Y_{k,h} = h_s) = \frac{k^{h-h_s}}{N(h_s) - 1} = P(X_{k,h} = (h - l)) \tag{2}$$

Let $Z_{k,h}$ be the random variable that maps from a uniformly chosen edge to the number of vertices in the respective separated subtree. To get the expected value $E[Z_{k,h}]$, the function $N(h)$ can be applied to the expected value $E[Y_{k,h}]$

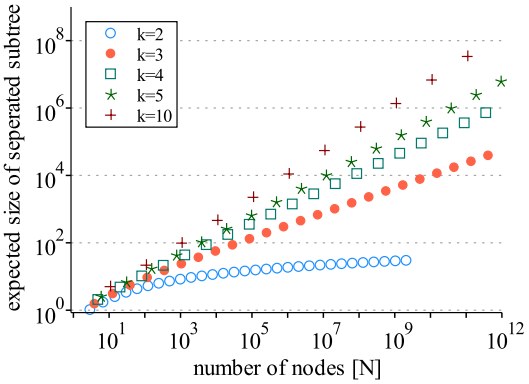
$$\begin{aligned} E\left[Z_{k,h}\right] &= E\left[N(Y_{k,h})\right] \\ &= \sum_{h_s=1}^{h-1} P(Y_{k,h} = h_s) \cdot N(h_s) \\ &= \sum_{l=1}^{h} P(X_{k,h} = l) \cdot N(h - l) \end{aligned} \tag{3}$$

The connection between $X_{k,h}$ and $Y_{k,h}$ is shown by example in Figure 4a. It shows the probability for the size of the separated subtree with $h = 4$, $k = 2$ and an removed edge in $l = 2$. The separated subtree has the height $h_s = h - l = 1$ and therefore consist of 3 vertices. The probability, that a link breaks in level 2 is $P(X_{k,h} = 2) = \frac{2}{7}$ which equals the probability that a tree of height 1 is created.

$E\left[Z_{k,h}\right]$ shows the expected number of vertices, which will be separated, if a random link breaks. In the worst case scenario all of the node would initiate a separate rejoin procedure. So $E\left[Z_{k,h}\right]$ gives us the worst case number for the initiated rejoin procedures after the random link failure. In the best case only the root vertex of the separated subtree rejoins. The approximated number of nodes that would rejoin in the worst case is plotted in Figure 4b. The calculation is done for trees between the height of one and thirty, and but shows only the results to $10^{10}$ nodes in the tree. After this, the lines increase just linearly. The figure shows, that with growing size of the trees, the size of the separated subtree can be approximiated linearly. With increasing $k$ the complete line gets quasi linear.




(a) The subtree which is separated by deleting one edge is of height $h_s = h - l$ which results into $N(1) = 3$ vertices.




(b) expected size of the subtree $E[Z_{k,h}]$ by removal of an random edge

Figure 4: Example calculations on perfect k-ary trees with $k = 2$

In this section the calculation base on perfect k-ary trees because they were used to develop the idea how to approximate the worst case number of separated and rejoining nodes in the first place. They are not supposed to show results which are valid for multicast distribution trees. In the next step, we do similar calculations for uniform recursive trees, which fit better to multicast trees.

## 6.2 First Approach to Subtree Sizes in URTs

Properties of vertices in URTs are often represented as a property of the root vertex. They can be applied to any vertex because of the recursive structure of the URTs. The following properties from [10, 16] are interesting for the calculation of the subtree sizes in URTs.

Let $U_N$ denote the node degree of a root vertex in an URT of N vertices. $U_N$ becomes a random variable when we randomly select a tree from the set of all URTs of N nodes. According to [10], the expected value $E[U_N]$ for large $N$ reads:

$$E[U_N] = \sum_{j=1}^{N-1} \frac{1}{j} \tag{4}$$

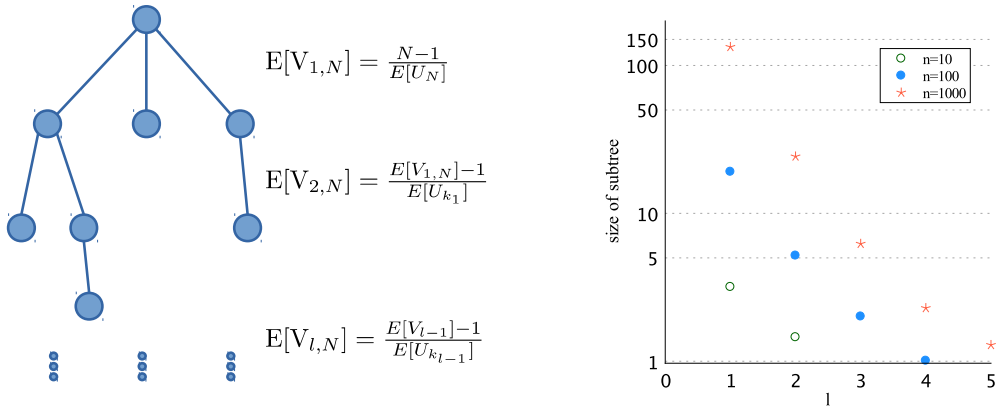
We are interested in the number of nodes which get separated if randomly chosen link fails, not only in the first level. So let the random variable $V_{l,N}$ map from a subtree with a failing link at level $l$ to the size of the separated subtree. We now derive the expected size of the subtree $E[V_{l,N}]$, depending on the level $l$.

Assume $U_N = i$, then we have $i$ branches with the total number of nodes $|T| = |T_1 + ... + T_i| = |T_1| + ... + |T_i|$. The expected value is additive so we have $E[|T|] = E[|T_1|] + ... + E[|T_i|]$. Therefore, the expected size of each branch rooted at depth $d = 1$ can be calculated as $E[V_{1,N}] = \frac{N-1}{i}$. This shows, that expected size of the subtrees depends on the concrete value of $i$.

Equation (4) gives us only the expected value $E[U_N]$. In the following approach we assume that this is the actual node degree of the root vertex, so that $i = E[U_N]$. Therefore, if an edge is removed in level $l = 1$ we expect a subtree size of $(N-1)/E[U_N]$. This is shown in the first level of Fig. 5a. Because of the recursive structure of URTs, we can apply the same idea on the next level $l = 2$. For simplicity, we assume that in the remaining subtrees the number of vertices is $k_l = E[V_{l,N}]$ even if normally other values are possible. Note that the following calculated values are only approximations to $E[V_{l,N}]$. As shown in Figure 5a, the number of edges in the second level is $E[U_{k_1}]$. So we can approximate the size of the separated subtrees $E[V_{l,N}]$ recursively as follows:

$$E[V_{l,N}] = \begin{cases} \frac{E[V_{l-1,N}]-1}{E[U_{k_{l-1}}]} & l > 1 \text{ and } E[U_{k_l}] > 0 \\ \frac{N-1}{E[U_N]} & l = 1 \text{ and } E[U_N] > 0 \end{cases} \tag{5}$$

In Fig. 5b, $E[V_{l,N}]$ is calculated for URTs of size 10, 100, and 1000. It shows, that the expected size of the subtree decreases very fast, and even trees with a large number of vertices, have a low expected height. The reason for that is the above mentioned assumption that every branch consist of the same number of nodes. $E[V_{l,N}]$ does not cover the possibility that branches of other sizes exist. There are two immediate next steps. The first is to include the possibilities of other branch sizes in the calculation. And the second is to use conditional probabilities to construct expectations over the subtree size dependent on the actual degree of a vertex.




(a) $E[V_{l,N}]$ is recursively approximated over the levels

(b) expected size of the subtree $E[V_{l,N}]$ as a function of $l$

Figure 5: $E[V_{l,N}]$ as the expected value for an separated subtree if a link fails on level $l$ in a URT

# 7 Outline for the Upcoming Work

The main goal is to find a general way of quantifying robustness as a function of the change of the network. A number which represents robustness in related contexts is the percolation threshold. So the idea is, to check if multicast networks as PIM-SM develop a percolation threshold that depends on the change of a network introduced by its repair mechanisms. PIM-SM is chosen as a first case study because it is widely used, creates URTs, which can be approximated, and uses calculable join and leave procedures as reconfiguration methods after a link fails.

To use the percolation threshold as indicator for robustness, several things need to be figured out. Therefore, the upcoming work consist of four steps. The first step is to approximate the sizes of subtrees that are separated from the complete tree if a random link fails. This number shows how many nodes need to be reconnected. In the second step, we use the size of the separated subtrees to approximate the number of new states in the multicast distribution tree. In the worst case, all of the separated nodes rejoin on distinct paths. In the best case, it is just the root vertex of the separated subtree, which rejoins. To approximate the number of new states in the distribution tree we use the formula for the expected change $E[\Delta_N(m)]$ introduced in "Stability of a Multicast Tree" [17]. In the third step, regularly failing nodes and the number of changed states in the network are used to observe the overall connectivity. Furthermore, it is interesting whether a percolation threshold emerges due to the interval of failing nodes, under which network decays and no solid data distribution for the receivers can be provided. In the last step, the outcomes of the calculations need to be tested against real world multicast distribution trees.

The approach for the first step was introduced in Section 6. An overview about the steps two, three, and the four are given in the following paragraphs.

## 7.1 Approximating the Number of New States

On the basis of the number of separated nodes, the number of changed links in the networks are approximated in the second step. This is done by summing up the change in a network, which follows the rejoin reconfiguration method in the case of a link failure.

From the join procedure we see that PIM-SM maintains two kinds of shortest path trees. On the one side, is the shared tree, rooted at the RP. One the other side are the source-specific trees, each rooted at a source. So per group we have one shared tree and one tree per source. Failing links affect each tree that was routed over the link. Each of them need to be rejoined over a different link and prune the old links. Hence, the number of new states in the network can be approximated as a sum of emerging joins and prunes, each weighted with the expected number of new links $E[\Delta_N(m)]$ from "Stability of a Multicast Tree" [17]. Thereby is $N$ the total number of nodes in the network and $m$ the $m^{th}$ joining (or leaving) node in the multicast tree. A challenge hereby is to find a fitting $m$. It is approximated from the expected number of separated nodes from the first step introduced before.

Furthermore, the change of a network is increased with every source. Thus, another challenge in this approach is to figure out the effects of an increasing number of source trees and how to quantify them. Also, it is uncertain whether the `prune(S,G,RP)` procedure can be approximated in the same way as the other joins and prunes, because it is unclear if the presumptions of $E[\Delta_N(m)]$ fit.

## 7.2 Percolation Analysis of Distribution Trees

The goal of using percolation analysis is to check wether there is a percolation threshold which indicates the decay of a distibution tree due to churn in underlying network layers. When the multicast distribution tree decays a

dataflow to the receivers is not provided anymore. To analyze the decay of the distribution tree, the distribution of the new states in the network need to be compared against randomly occurring failing links. The basic idea is that the distribution of new states takes time, and if links fail frequently, the states cannot be established fast enough. We can apply a discrete time in which a specific number of links are repaired, at each step, and also a specific number of links fail within a network. The question is, at which number of failures the distribution tree decays.

The failing and repaired links over time can be modeled with stochastic processes. It follows the question at which fraction of to-establish-states (resp. number of failing links per time) in comparison to the number of receivers, the distribution tree decays. This number would represent the robustness of the multicast algorithm.

### 7.3 Tests

Any calculated number, which represents robustness, needs to be tested against reality. In this case, it is necessary to test the calculations against the changes in PIM-SM distribution trees. Since it is not feasible, to get complete real world snapshots of PIM-SM group topologies, two different approaches come to mind.

The first possibility is to use network simulator software. For that, a great a number of network participants need to be simulated as well as the PIM-SM functionality. The advantage is that it would give complete snapshots over multicast distribution trees. Using network simulators comes with the problem that PIM-SM is not implemented for prominent network simulators as Omnet++ [3] or Network Simulator 3 [2].

The other option is to use real network meta data to support the calculations. Global snapshots of group topologies cannot be created, and so the number of distinct rejoining paths cannot be counted directly. Therefore, it is necessary to extract meaningful snapshots of local configuration states. The most states are set by defined configuration messages and a large number of them can be observed at internet exchange points (IXPs). So what can be done, is to observe changing routing configuration messages and compare them against the number of following changing multicast configuration messages at certain IXPs. The advantage of getting this data from IXPs is, that there is a large number of routing information and additional multicast configuration messages to observe. What cannot be seen directly, is the number of receivers behind the root of the separated distribution tree. One way to deal with this is to extract an expected size of the subtree from the number of subtrees which have just been separated but there may be more sophisticated approaches.

## 8 Chances and Risks

The introduced idea of quantifying complexity with the percolation threshold comes with a number of risks. Nevertheless, if the quantification produces meaningful result, there is a chance, that similar quantification can be applied to other network algorithms and the complexity of network algorithms can become comparable. The main risks, chances, and possibilities to transfer the quantification to other algorithms are presented in this section.

### 8.1 Chances

The major chance of the upcoming work is the quantification of robustness in multicast network as a function of churn in the underlying network layers. Furthermore, the quantification can result in a meaningful metric of the complexity introduced by repair mechanism of the network algorithms. In other networks, the percolation threshold successfully described the vulnerability (e.g. [7]). So there is a good possibility to apply the percolation analysis successful on the PIM-SM networks. Furthermore, if similar quantifications are possible for other protocols, the complexity of different network algorithms becomes comparable.

Even if the quantification is not transferable to other protocols, weaknesses of the PIM-SM control loop can become visible and maybe transferred to other protocols. If one specific reconfiguration methods is followed by a repeatedly extensive change in the network, it would indicate which characteristics of a control loop is responsible for it (e.g. lacks a specific check, maybe for good reason). Other control loop could be analyzed if they have the same characteristic and result in a similar pattern.

If the quantification is tested against real configuration data in more than one IXP, there is a chance that cascading failures can be observed. For example, the rejoin procedure can be observed from the side of the separated subtree. In this case, we could see how at first the root vertex of the separated tree rejoins the multicast distribution tree. Subsequently, the connection could get too slow for the other nodes in the subtree, so that they rejoin over another connection. Thereafter the previous joined root node would leave again, because it is not responsible for the data transfer anymore. Patterns like this, and its likability of occurrence, show how effective the rejoin procedure works, and how much unnecessary link state change emerges.

## 8.2 Risks

There are two risks with this approach to quantify network complexity.

The first risk is, that the mathematical calculations do not reflect in the tests. Nevertheless, even if the mathematical calculations do not reflect in the test, we get insights about the difference between the model and the real world. The part of the theory that does not fit to reality becomes visible, and this results in the chance to create a more accurate model.

There can be two reasons why the calculations are not reflected in the tests. The first reason is, that the calculations generalize the structure of the multicast distribution tree too much. This can happen, if the calculations approximate the behavior too roughly. This problem can be approached by putting more precise values into the model, e.g. by only test a specific kind of trees. An example would be, to test only the source trees with exactly one sender. The other reason is, that the mathematical model bases on wrong assumptions. To sort them out, we need to find out, which assumptions are not fitting and change the mathematical model accordingly.

The second risk is not finding a test setup, which is actually able to evaluate the calculations. Since the effects of complexity arise from a high number of participating nodes, as well as a high number of protocols interacting, software network simulator could not be able to cover emerging phenomenons. This is why testing with real world network data seems a more promising idea. The problem of testing with real world data is that it only shows a local view. Extracting meaningful data out of a local view depends on the amount of visible data, as well as on the ability to interpret them in a global context. So the interpretation of the collected data is a main challenge to support the validity of the approximations.

## 8.3 Transferability of results

The results of the quantification of the robustness can be transferred to other algorithms, as long as they have comparable environments and fulfill the requirements of the mathematical foundations. This is one reason, why we choose to analyze multicast algorithms with shortest path trees. So many numbers, such as the number of separated nodes, can be transferred to other algorithms. What changes between the algorithms is, how the distribution trees are created and maintained.

In the reconfiguration methods lie another risk. The rejoin in PIM-SM is a number of joins and leaves, which can be described by the $E[\Delta_N(m)]$ value. In other algorithms the reconfiguration methods can be harder to approximate or it could be even impossible.

The basic idea of using change introduced by failures to check for the robustness is transferable to other network algorithms in general. A challenge, which arises from each analyzed algorithm, is to define what it means that a network is broken and detect its occurrence.

Another point which is transferable to the analysis of other network algorithm is the problem of evaluating calculations. For many network algorithms, it will be a challenge that global snapshots cannot be achieved. For example for the determination of a 'broken' network. From this arises the question for general experimentation methods and tools, which is another challenge in the field of network complexity.

# 9 Summary and Outlook

In this work, the reconfiguration method of the PIM-SM routing protocol is observed from a network complexity point of view. Network algorithms are implemented in the form of control loops, which run independently on nodes in a network. Network complexity describes the effects which emerge from interaction of different nodes and different components within the control loops or network stacks. While a certain level of complexity is needed to provide a robust network, too high complexity leads to unpredictable behavior. We reason why robustness is affected by the complexity of the repair mechanism of network algorithms.

The outlined approach in this work is about the quantification the effects of PIM-SM repair mechanism within a PIM-SM network. PIM-SM is chosen as a popular IP multicast protocol and because it maintains shortest path trees, which can be expressed as uniform recursive trees. Uniform recursive trees offer a well understood structure and calculable properties.

The quantification of rejoin methods for PIM-SM includes four steps. In the first step the expected size of the separated subtree is approximated. In the second step, the number of new states are approximated. In the third step, a percolation threshold is searched, emerging from regularly failing nodes in a network and in the last step the calculations need to be tested against real world multicast distribution trees. The current state of this work is in the first step.

These steps describe the upcoming work of the approach to quantify the rejoin methods of PIM-SM. If this kind of quantification is successful, it can be transferred to other network algorithms. In the future we want to calculate the robustness in a similar way on other algorithms so that the robustness of several algorithms can be compared against each other.

# References


[1] Network of Excellence in Internet Science (EINS), February 2015. http://www.internet-science.eu/.

[2] Network Simulator 3, February 2015. http://www.nsnam.org/.

[3] Omnet++ - Discrete Event Simulator, February 2015. http://www.omnetpp.org/.

[4] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[5] Michael Behringer and Geoff Huston. A Framework for Defining Network Complexity. Internet-Draft – work in progress 01, IETF, November 2013.

[6] Nora Berg, Sebastian Meiling, and Thomas C. Schmidt. Perspectives on Multicast Complexity. In *Proc. of WS on Understanding the inter-play between sustainability, resilience, and robustness in networks (USRR)*, Ghent, Belgium, April 2014. Network of Excellence in Internet Science (EINS).

[7] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[8] Reuven Cohen and Shlomo Havlin. *Complex Networks: Structure, Robustness and Function*. Cambridge University Press, 2010.

[9] Reinhard Diestel. *Graphentheorie*. Springer, 3rd edition, 2006.

[10] Qunqiang Feng, Chun Su, and Zhishui Hu. Branching structure of uniform recursive trees. *Science in China Series A: Mathematics*, 48(6):769–784, 2005.

[11] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601, IETF, August 2006.

[12] Peter Hartmann. Mathematik für Informatiker: Ein praxisbezogenes Lehrbuch. Vierte Auflage, 2006.

[13] D. Papadimitriou, P. Van Mieghem, K. Salamatian, and F. Griffths. D1. 5: Roadmap on developing the fundamental basis of autonomous and dynamic complex networks. 2014.

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

[15] James Andrew Storer. *An introduction to data structures and algorithms*. Springer Science & Business Media, 2001.

[16] Chun Su, Qunqiang Feng, and Zhishui Hu. Uniform recursive trees: Branching structure and simple random downward walk. *Journal of mathematical analysis and applications*, 315(1):225–243, 2006.

[17] Piet Van Mieghem and Milena Janic. Stability of a Multicast Tree. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 1099–1108. IEEE, 2002.