European Research
7th Framework Programme

Project title: Community Networks Testbed for the Future Internet.

Experimental research on testbeds for community networks (year 4)

Deliverable number: D.4.4
Abstract

This deliverable gives reports on the results of the research performed in CONFINE WP4 during the forth year (the last) mainly on the experimentation facilities deployed during the project execution. In the first section, we present the results of an empirical study that tries to give a better understanding on how (un)stable community networks behave.

Secondly, the results of a large-scale measurement campaign, where we specifically analyse the network performance as experienced by the end-user in community networks in comparison to other ISPs is presented. We also describe a study that focuses on link quality prediction by means of a time series analysis, as well as a study that explores end-to-end quality prediction in the routing layer of large-scale, distributed, and decentralized system.

The third section of this deliverable focuses on the topic of self-management. We first present mechanisms that increase the robustness of network connectivity. More specifically, firstly we address the detection of traffic forwarding faults. Secondly, we focus on one important self-management mechanism, the routing, and we study the scalability, performance, and stability of three proactive mesh routing protocols: OLSR, BMX6 and Babe, three common routing protocols in wireless community networks. Next, we experimentally evaluate the BMX6 mesh routing protocol in a production community network (QMPSU). The presentation of preliminary results from a performance comparison between the two multicast routing protocols, ODMRP and SMF, and a performance comparison between OLSR here called OLSRv1 (with the ETX metric) and OLSRv2 (with our novel DAT metric) follows. The next study applies some social mining techniques aiming to identify the roles of the individuals in the social network behind a community network, here Guifi.net, and measures the participatory involvement in the community network from 2003 to 2014. The following study in this section focuses on designing an open process to collect information, allowing each participating Community Network to shape up the topics addressed and then contribute details for a selection of them. Finally, we describe how Fraunhofer FKIE has improved and extended the OLSRv2 implementation of Olsr.org to provide Community Mesh networks with a better access to the second generation OLSR protocol – OLSRv2.
Contents

1 Scale, heterogeneity and limited resources in the infrastructure 7
  1.1 Introduction ............................................................. 7
  1.2 An Empirical Study on BGP ........................................... 7
    1.2.1 Introduction ...................................................... 7
    1.2.2 Border Gateway Protocol (BGP) ................................ 7
    1.2.3 Measurement Setup ............................................. 8
    1.2.4 Analysis .......................................................... 9
    1.2.5 Conclusions ..................................................... 12

2 Cross-Layer Interactions and Optimizations 20
  2.1 Measurement-Lab ..................................................... 20
    2.1.1 Measurement Tools .............................................. 20
    2.1.2 Results and Analysis ........................................... 22
    2.1.3 Implications on Community Networks for Development ......... 26
    2.1.4 Conclusions ..................................................... 29
  2.2 Predicting Link Quality ............................................. 34
    2.2.1 Experimental Methodology ...................................... 34
    2.2.2 Global Analysis of Links and Nodes ............................ 35
    2.2.3 Analysis of Individual Links and Nodes ....................... 38
    2.2.4 Lessons Learned and Conclusions .............................. 40
  2.3 Predicting Path Quality ............................................. 40
    2.3.1 Background ...................................................... 41
    2.3.2 Link Quality Prediction in Community Networks ............... 42
    2.3.3 Analysis of Results ............................................ 44
    2.3.4 Conclusions ..................................................... 48
  2.4 Minstrel Link Layer Rate Control Algorithm .................... 49

3 Self-management 50
  3.1 Routing robustness ............................................... 50
    3.1.1 Background ..................................................... 50
    3.1.2 Traffic summaries: Sketches ................................... 51
    3.1.3 Traffic Validation .............................................. 62
    3.1.4 Distributed detection .......................................... 66
    3.1.5 Conclusions ..................................................... 74
  3.2 Evaluation of mesh routing protocols for wireless community networks 74
    3.2.1 Mesh Routing Protocols ....................................... 74
    3.2.2 Evaluation Metrics ............................................. 78
    3.2.3 Emulation Experiments ........................................ 79
    3.2.4 Testbed Experiments .......................................... 83
    3.2.5 Discussion of Results and Conclusion ......................... 87
  3.3 Experimental Evaluation of BMX6 Routing Metrics ............... 92
3.3.1 Related Work .......................................................... 93
3.3.2 Theoretical Path Capacity ........................................... 93
3.3.3 BMX6 Performance ..................................................... 97
3.3.4 Conclusions ............................................................. 97
3.4 Multicast Performance Comparison of SMF and ODMRP ......... 97
  3.4.1 Introduction .......................................................... 98
  3.4.2 Related Work ......................................................... 99
  3.4.3 Testbed and Methodology .......................................... 100
  3.4.4 Evaluation ........................................................... 102
  3.4.5 Conclusion ........................................................... 106
3.5 Performance of OLSRv2 with DAT metric ......................... 107
  3.5.1 Physical Testbed at the Fraunhofer FKIE ..................... 107
  3.5.2 Evaluation of OLSRv2 Route Selection ....................... 108
  3.5.3 Conclusions ........................................................ 112
3.6 Analysis of the social effort in multiplex participatory networks . 113
  3.6.1 Community Network Member Participation ................... 114
  3.6.2 Experimental framework ......................................... 116
  3.6.3 Network evolution ................................................ 117
  3.6.4 Communities and multiplex analysis ......................... 119
  3.6.5 Conclusions and discussion .................................... 122
3.7 Socio-Economic qualitative analysis of community networks .... 123
3.8 Improvement of Community Network Software .................... 125
  3.8.1 Standard Implementation ........................................ 125
  3.8.2 OpenWRT Integration ............................................. 125
  3.8.3 Modularization and Documentation ............................ 126
  3.8.4 Acknowledgements ............................................... 126
4 Conclusions .................................................................. 127
List of Figures

1.1 Total updates and withdrawals in AWMN for November 10, 2013 ........................ 9
1.2 Cumulative updates and withdrawals per AS in AWMN for November 10, 2013 .... 10
1.3 Total updates and withdrawals per day in AWMN for November 2013 .................. 11
1.4 Median number of updates and withdrawals per AS per day in AWMN for November 12
1.5 Maximum number of updates and withdrawals per AS per day in AWMN for November 13
1.6 Monthly average number of updates and withdrawals per hour in AWMN for November 14
1.7 Total updates and withdrawals per day in Guifi for June 2014 ........................... 14
1.8 Median number of updates and withdrawals per AS per day in Guifi for June 2014 ... 15
1.9 Maximum number of updates and withdrawals per AS per day in Guifi for June 2014 15
1.10 Monthly average number of updates and withdrawals per hour in Guifi for November 13
1.11 AWMN RIB table size per day for November 2013 ...................................... 16
1.12 Hourly average update prefix rate per second for public Internet ........................ 17
1.13 Hourly average update prefix rate per second for AWMN ............................... 17
1.14 Hourly average update prefix rate per second for Guifi ................................. 18
1.15 Peak prefix update rate per second in the public Internet ............................... 18
1.16 Peak prefix update rate per second in AWMN .......................... 19
1.17 Peak prefix update rate per second in Guifi ........................................... 19
2.1 Nodes in the experiment .......................................................... 22
2.2 Cumulative distribution of RTT measurements, capped to 1s .......................... 23
2.3 RTT measurements, AWMN ....................................................... 24
2.4 RTT measurements, guifi.net .................................................. 25
2.5 RTT measurements, Ninux .................................................... 26
2.6 Cumulative distribution of download measurements, capped to 400 Mbps .......... 27
2.7 Cumulative distribution of upload measurements, capped to 400 Mbps ............ 28
2.8 Download speeds, Ninux ......................................................... 29
2.9 Upload speeds, Ninux ................................................................ 30
2.10 Median Download throughput (Mbps) per Country and ISP, sorted by Download 31
    speed (M-Lab July 2015) ................................................................ 31
2.11 Median Upload throughput (Mbps) per Country and ISP, sorted by Download speed 32
    (M-Lab July 2015) ......................................................................... 32
2.12 Median Latency (RTT in ms) per Country and ISP, sorted by Download speed (M-Lab 33
    July 2015) .................................................................................. 33
2.13 Average Mean Absolute Error (MAE) of the links ........................................... 36
2.14 RT average Mean Absolute Error of the LQ predictions .................................. 36
2.15 RT Mean Absolute Error of the whole network for several test dataset sizes ... 37
2.16 Average Mean Absolute Error versus Link Quality in steps of 0.1 .................... 38
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.17</td>
<td>Sample of LQ time series for each of the 4 clusters from K-means algorithm.</td>
<td>39</td>
</tr>
<tr>
<td>2.18</td>
<td>Prediction error of each method for each cluster, depicted as box plot.</td>
<td>39</td>
</tr>
<tr>
<td>2.19</td>
<td>Average Mean Absolute Error (MAE) of the paths.</td>
<td>45</td>
</tr>
<tr>
<td>2.20</td>
<td>Mean Absolute Error (MAE) of the EtEQ predictions as boxplot.</td>
<td>45</td>
</tr>
<tr>
<td>2.21</td>
<td>Distribution of RBR Mean Absolute Error (MAE) as boxplot.</td>
<td>46</td>
</tr>
<tr>
<td>2.22</td>
<td>Evolution of the average ETX, the average prediction and the average absolute error for: a)12 am-12 pm, b)12 pm-12 am</td>
<td>47</td>
</tr>
<tr>
<td>2.23</td>
<td>Evolution of the average ETX, the average prediction and the average absolute error</td>
<td>46</td>
</tr>
<tr>
<td>2.24</td>
<td>RBR average Mean Absolute Error (MAE) of the EtEQ predictions.</td>
<td>48</td>
</tr>
<tr>
<td>2.25</td>
<td>RBR Mean Absolute Error of EtEQ predictions, depicted as a boxplot.</td>
<td>48</td>
</tr>
<tr>
<td>3.1</td>
<td>Error distribution</td>
<td>54</td>
</tr>
<tr>
<td>3.2</td>
<td>99 percentile of the error</td>
<td>54</td>
</tr>
<tr>
<td>3.3</td>
<td>Error for the AGMS sketch</td>
<td>55</td>
</tr>
<tr>
<td>3.4</td>
<td>Coefficient for different percentiles</td>
<td>56</td>
</tr>
<tr>
<td>3.5</td>
<td>Error distribution</td>
<td>57</td>
</tr>
<tr>
<td>3.6</td>
<td>99 percentile of the error</td>
<td>57</td>
</tr>
<tr>
<td>3.7</td>
<td>Error for the FAGMS sketch</td>
<td>58</td>
</tr>
<tr>
<td>3.8</td>
<td>Coefficient for different percentiles</td>
<td>58</td>
</tr>
<tr>
<td>3.9</td>
<td>Error distribution</td>
<td>60</td>
</tr>
<tr>
<td>3.10</td>
<td>99 percentile of the error</td>
<td>61</td>
</tr>
<tr>
<td>3.11</td>
<td>Error for the TZ sketch</td>
<td>62</td>
</tr>
<tr>
<td>3.12</td>
<td>Example packets with unsynchronized clocks.</td>
<td>64</td>
</tr>
<tr>
<td>3.13</td>
<td>Proportion of detected nodes</td>
<td>65</td>
</tr>
<tr>
<td>3.14</td>
<td>Effects of the sketch size</td>
<td>66</td>
</tr>
<tr>
<td>3.15</td>
<td>Example network</td>
<td>67</td>
</tr>
<tr>
<td>3.16</td>
<td>Barcelones area backbone</td>
<td>73</td>
</tr>
<tr>
<td>3.17</td>
<td>Protocols’ cost</td>
<td>73</td>
</tr>
<tr>
<td>3.18</td>
<td>Topology dissemination mechanisms</td>
<td>78</td>
</tr>
<tr>
<td>3.19</td>
<td>The topology of the Barcelonés network</td>
<td>80</td>
</tr>
<tr>
<td>3.20</td>
<td>Network overhead in bytes</td>
<td>81</td>
</tr>
<tr>
<td>3.21</td>
<td>Network overhead in number of packets</td>
<td>81</td>
</tr>
<tr>
<td>3.22</td>
<td>Network stability and reactivity</td>
<td>82</td>
</tr>
<tr>
<td>3.23</td>
<td>Memory consumption</td>
<td>83</td>
</tr>
<tr>
<td>3.24</td>
<td>Topologies in different scenarios</td>
<td>85</td>
</tr>
<tr>
<td>3.25</td>
<td>Network densities and link qualities for different testbed configurations</td>
<td>85</td>
</tr>
<tr>
<td>3.26</td>
<td>End-to-end path length and TCP throughput depending on network density varying power and rate</td>
<td>86</td>
</tr>
<tr>
<td>3.27</td>
<td>Data overhead (bytes/s) depending on network size and density (power, rate), no TCP user traffic</td>
<td>87</td>
</tr>
<tr>
<td>3.28</td>
<td>Data overhead (bytes/s) depending on network size and density (power, rate)</td>
<td>87</td>
</tr>
<tr>
<td>3.29</td>
<td>Data overhead (packets/s) depending on network size and density (power, rate)</td>
<td>88</td>
</tr>
<tr>
<td>3.30</td>
<td>CPU consumption depending on network size and density (power, rate)</td>
<td>88</td>
</tr>
<tr>
<td>3.31</td>
<td>Memory consumption depending on network size and density (power, rate)</td>
<td>89</td>
</tr>
<tr>
<td>3.32</td>
<td>End-to-end delivery success depending on topology dynamics and link redundancy</td>
<td>89</td>
</tr>
</tbody>
</table>
3.33 Experimental $C_e(P)$ (exp), theoretical $C_t(P)$ (comp), and fitted $C_f(P)$ (corr) capacities (top); relative error of theoretical $e_t(P)$ and fitted $e_f(P)$ capacities (middle); and number of hops (bottom). ................................................................. 95
3.34 Fitted (corr), best (best), and worst and best SPF paths (spf.min, spf.max) capacities (top); relative error with respect to corrected capacities (middle); and number of hops of fitted, best and SPF paths (bottom). ................................................................. 96
3.35 Topology of the testbed at the Fraunhofer FKIE, each number representing a research node. ................................................................. 100
3.36 Packet loss ratios on individual links measured by ODMRP ................................................................. 101
3.37 End-to-end delay in the VCT testbed ................................................................. 104
3.38 Packet loss in the VCT testbed ................................................................. 104
3.39 Network load during a multicast measurement in the VCT testbed ................................................................. 105
3.40 Packet loss in the physical testbed ................................................................. 105
3.41 Network load during a multicast measurement in the physical testbed ................................................................. 106
3.42 Example of a hybrid node within the testbed ................................................................. 107
3.43 Subset of the testbed topology at the Fraunhofer FKIE used for measurements. Each number is representing a research node. The red dots indicate the approximate antenna positions. ................................................................. 108
3.44 Network topology with a long distance link as seen by OLSRv2 with DAT metric during TCP throughput measurements ................................................................. 108
3.45 Results of the TCP throughput measurement with 20 replications and a confidence level of 0.95 ................................................................. 110
3.46 Route selection with OLSRv1, destination node 139 ................................................................. 111
3.47 Route selection with OLSRv2, destination node 139 ................................................................. 111
3.48 Route selection with OLSRv1, destination node 135 ................................................................. 112
3.49 Gini coefficient of two participatory forums in Guifi.net ................................................................. 114
3.50 Guifi.net users participation measured as the number of messages posted in the mailing lists users-list and dev-list, and the number of new communication devices. ................................................................. 115
3.51 The initial set of data gathered and the multiplex graph generation process ................................................................. 118
3.52 Evolution of members’ participation in each participatory layer ................................................................. 118
3.53 Number of nodes by community. Comm., Dev. and Users refer to the communication, dev-list and users-list layers. ................................................................. 120
3.54 Links distribution between communities. Bridges are links between two participants, both members of the same community, but one of them member of another community, too. Others are links to nodes which do not belong to any community. ................................................................. 120
3.55 Empirical Cumulative Distribution Function (ECDF) of members’ centrality scores ................................................................. 121
3.56 Multiplex layers robustness. ................................................................. 122
## List of Tables

2.1 Values in Figures 2.11, 2.10, 2.12 ................................................................. 26
3.1 Periodic Messages ................................................................. 78
3.2 Network characteristics of the emulation experiments ........................................... 80
3.3 General W-ILab.T testbed and protocol characteristics and configuration ............... 84
3.4 Experimented parameters, defaults, and ranges .................................................. 84
3.5 Summary of observed and interpreted performance characteristics from Section 3.2.5 92
3.6 Configuration parameters used in the evaluation ................................................. 103
3.7 Parameterization of Testbed .............................................................................. 109
3.8 Summary of the basic properties of the graph layers ........................................... 117
1 Scale, heterogeneity and limited resources in the infrastructure

1.1 Introduction

In this section we present the results of an empirical study that gives a better understanding on how (un)stable community networks behave. The community networks under study use the BGP as routing protocol. Each node in the network is represented as being a separate Autonomous System (AS). By analysing the amount of update and withdrawal messages sent on those networks and comparing this to the amount of BGP messages on the Internet, we have determined that, as expected, community networks are indeed less stable than the Internet. In the following section, we will present the setup of this study and discuss the results from our analysis.

1.2 An Empirical Study on BGP

1.2.1 Introduction

Because of the do it yourself (DIY) approach in community networks, a robust routing protocol is required to cope with frequent outages and topology changes[142]. Moreover, the routing has to be highly distributed and scalable, because of the internal organisation of community networks. Some networks use ad-hoc network routing protocols such as Optimized Link State Routing Protocol (OLSR) for this[104], other use novel mesh firmwares like LibreMesh[44], while the largest community networks such as Athens Wireless Metropolitan Network (AWMN) and Guifi simply use BGP[7] with in most cases one BGP AS number corresponding to one node. It is the performance of these BGP networks that has been evaluated, as an initial assessment of the stability of community networks from a routing protocol perspective.

Extensive research has been performed on the stability of BGP in the public Internet, especially related to the stability of the BGP routing tables[125, 72]. In this work, we started from BGP dumps rather than from the routing table itself. Other researchers take an analytical view on BGP stability[130], while we use an empirical approach. To the best of our knowledge, this is the first study of community networks stability. While research in this field has been growing steadily, the overall stability of this kind of networks is still unknown. This is also the major contribution of this work, to present initial stability measurements of community networks.

The remainder of this section is organised as follows: in section 1.2.2 we introduce the most important concepts of BGP, followed by an overview of the measurement setup in section 1.2.3. Section 1.2.4 presents the analysis results and finally, the conclusions are presented in section 1.2.5.

1.2.2 Border Gateway Protocol (BGP)

BGP is a well studied exterior gateway routing protocol, already at version 4[123]. At a high level, the protocol exchanges routing information between different ASs to make path decisions based on path
1.2. An Empirical Study on BGP

length, peering policies and rules. BGP is best known for being the routing protocol on the Internet, connecting different entities around the world to each other, while adhering to peering agreements. Although the protocol comes with its own set of challenges and security issues[97], its distributed nature, well-known behaviour and broad support in even simple routing solutions led to a broad adoption in community networks.

The protocol has an extensive list of features and configuration possibilities[149]. In what follows we describe the most relevant functional components, focusing on network stability. The two most important messages for this analysis are updates (also called announcements) and withdrawals. While the former serves as a message to inform neighbours and eventually the entire network about newly available routes, the latter signals path deletion from the network. Notice that in a relatively stable network the amount of withdrawals is supposed to be smaller than the number of updates, which indicates a network with possibly multiple parallel routes and no permanent path deletion.

We have limited ourselves to an analysis of the update and withdrawal messages, as they give a direct indication of the dynamic nature of the networks. In section 1.2.4.3, we give an overview of how these update and withdrawal messages affect the overall size of the Routing Information Base (RIB) table.

1.2.3 Measurement Setup

To study the behaviour of BGP in community networks, a BGP monitor has been installed in the AWMN and Guifi networks. The resulting data sets are publicly available1. Because of connectivity issues some data sets are empty. We have focused on the data sets from November 2013 in the case of AWMN and June 2014 in the case of Guifi. Considering publicly available growth charts of Guifi2, we believe the behaviour in more recent periods will be comparable as the network growth seems to stabilise. Unfortunately no growth data from AWMN is available.

The data sets contain the daily update/withdrawal and RIB messages, dumped at intervals of 15 minutes. As AWMN is federated with other community networks over the FEDERICA European federation network3, BGP traffic from other networks is expected to appear in the dumps. However, active BGP filtering at the peering links is in place, reducing the number of entries from other community networks to a negligible amount.

For analysis of the data the update messages were parsed using the RIPE NCC libBGPDump tool4. The dumps were first uncompressed and then inserted in a MySQL database for further analysis. The following columns were taken into account: message timestamp, message type (announcement or withdrawal), originating IP and AS, AS path and announced subnet. In total 2,681,777 rows were inserted in the MySQL database for AWMN, for Guifi this amounted to 4,396,650 rows. These rows were then queried with SQL queries to generate the plots in the following sections. During the BGP capturing process, 819 distinct nodes could be identified in the AWMN network, while 1996 nodes were identified in Guifi.net.

To compare the behaviour of BGP in community networks to the behaviour of BGP in the public Internet, we used the publicly available BGP data from Potaroo.net[71], a website which maintains a list of analyses of Internet BGP behaviour.

---

2Growth chart or corba de creixement available at https://guifi.net/guifi/menu/stats/nodes
3See http://www.fp7-federica.eu/
4See https://bitbucket.org/ripencc/bgdump/wiki/Home

---
1.2.4 Analysis

Based on the measurement data gathered with the measurement setup described in the previous section, we first performed an analysis of community networks separately. Then we compared the observed behaviour against public Internet observations.

1.2.4.1 BGP in AWMN

To provide context for the analysis of the stability of the community networks under study, Figure 1.1 gives an overview of messages observed in AWMN during a single day, in this case November 10, 2013. It becomes clear immediately that community networks show strong variations in network stability over time, with peaks in this figure around 13:00 and 18:00 (UTC). As the number of announcements is much higher than the number of withdrawals, clearly multiple paths between ASs are present. This is common in wireless community networks as can be confirmed by public topology information of AWMN and e.g. Guifi. Although this indicates higher resilience in case of failures of single links, it also does imply a higher routing message load.

![Figure 1.1: Total updates and withdrawals in AWMN for November 10, 2013.](image)

To assess the influence of individual community network nodes on the overall network stability, for this same day in AWMN Figure 1.2 shows the cumulative updates and withdrawals, relative to the AS percentiles. Clearly less than 50% of all AS numbers is responsible for the largest number of updates and withdrawals, with a strong exponential distribution of the announcements and withdrawals. From this we conclude that a number of weak points in the networks are responsible for most routing traffic and, as a consequence, most instability. This is illustrated by the fact that the top 5% of nodes each generate more than 1000 announcements and 200 withdrawals in a single day.

After zooming out, Figure 1.3 gives the total number of announcements and withdrawals over a period...
1.2. An Empirical Study on BGP

of the entire month November 2013 in AWMN, grouped per day. This figure illustrates the unstable, time-independent and largely unpredictable network behaviour per day.

For the same month, figures 1.4 and 1.5 give the median number of updates per AS and maximum number of updates per AS respectively, showing how the distribution of the number of updates is not uniform. A small part of all ASs are generating a relatively large amount of announcements, indicating that only parts of the AWMN network are unstable.

When considering the monthly averages per hour of AWMN as illustrated by Figure 1.6, a weak diurnal pattern can be observed. During working day hours, from 6:00 until about 20:00, more announcements are generated and the number of withdrawals is slightly higher. This indicates a larger degree of network instability or at least of changes to the network topology. In this case we believe this results from the very nature of community networks, where members maintain the network during daytime. Especially, the high number of updates in the late afternoon and the evening is not expected in commercial Internet offerings, where most operations happen during low-traffic periods. However, the pattern is quite weak, possibly smoothed out by the stronger, time-independent variations in the network.

1.2.4.2 BGP in Guifi

The other community network that had its BGP traffic monitored is Guifi in Spain, which is larger than AWMN (1996 vs. 819 nodes in the measurement data). Again, the data from one month, in this case June 2014, is analysed. The same graphs are presented, except for the introductory figures 1.1 and 1.2.

Comparing the maximum number of announcements and withdrawals per day between AWMN (Figure 1.3) and Guifi (Figure 1.7), it is clear that both network behave similarly: the Guifi network also
shows an unpredictable, unstable maximum amount of updates per day. Interestingly, the number of withdrawals is significantly higher in Guifi than in AWMN.

When the statistics per AS are compared, some interesting differences can be observed. In the Guifi network, the maximum number of updates per AS per day, as shown in Figure 1.9, is higher than in AWMN (Figure 1.5). In both AWMN and Guifi, some high spikes can be observed, but on average, the values are higher in Guifi. This is certainly the case when the number of withdrawals are compared. On the other hand, if the median number of updates per AS per day are compared (in figures 1.4 and 1.8), the values in Guifi are approximately 100 times lower. This is an indication that, generally speaking, the Guifi network is mostly very stable, but does contain unstable sections that generate a disproportionately large amount of announcements and withdrawals. Finally, when the hourly values averaged over a month are compared, as shown in figures 1.6 and 1.10, no significant differences can be observed. Both networks follow a similar, weak diurnal pattern.

1.2.4.3 RIB

The number of announcements and withdrawals does not give a direct indication on whether the network is growing or shrinking: a large number of updates and withdrawals might just indicate that a certain, possibly small, part of the network is behaving erratically and continuously disconnects and reconnects to the rest of the network. In order to verify the stability of the overall size of the network, the size of the RIB table in the AWMN network in the observed month, November 2013, is plotted in Figure 1.11. In this graph, an entry represents a RIB table size snapshot, taken with an interval of 2 hours.
1.2. An Empirical Study on BGP

1.2.4.4 Comparing Community networks and the Internet

To analyse the similarities and differences between community networks and autonomous systems in the public Internet, we considered the publicly available data from Potaroo.

Figure 1.12 shows hourly average update prefix rate per second, as measured by Potaroo for the public Internet for a week in May 2015. For comparison, based on the data from AWMN and Guifi, we derived similar data shown in Figure 1.13 and Figure 1.14 respectively. For the public Internet the average withdrawal rate is between 0.1 and 1 per second, for Guifi it is around 1 per second and for AWMN this same rate shows more variation and is closer to 0.1 on average. For the update rate we see that the Internet shows a rate of 1 to 10 per second, while for Guifi this rate is closer to 1. Again, AWMN shows larger variation.

When considering the peak prefix update rate per second as depicted in figures 1.15, 1.16 and 1.17 for respectively the public Internet, AWMN and Guifi, the withdrawal rates are surprisingly similar for the public Internet and AWMN, for Guifi the averages are higher. On the public Internet the peak prefix update rate is significantly larger, which is unexpected given the total number of ASs. When comparing the overall behaviour, the peak rate is similar. We currently do not have a full explanation for this, we can only guess that this is caused by the consideration of only peak rates.

1.2.5 Conclusions

Community networks are formed by individuals connecting to an existing network, by pointing one or multiple antennas on their roofs. There is a similarity to starting an Internet service provider (ISP) where you begin peering with other (transit) providers, however at a different scale and cost. More
interesting for the work performed here is that in both cases you will start with a single AS number in which you announce your subnet over BGP.

From the initial data on only the AWMN and Guifi networks we can conclude that a strong variation is present, with a small number of nodes causing instability in the networks. A weak diurnal pattern in the data can be observed.

When comparing the data gathered on these networks to data regarding the public Internet however, it is clear that in community networks the update rates and withdrawal rates per node are higher. The ratio of the update rates over the withdrawal rates is larger for the public Internet, due to the larger number of routes.

In general, we can conclude that to a certain degree community networks behave as the public Internet and could well be the Future Internet, with a higher degree of instability. This forms an important challenge to tackle with future research.
1.2. An Empirical Study on BGP

Figure 1.6: Monthly average number of updates and withdrawals per hour in AWMN for November 2013

Figure 1.7: Total updates and withdrawals per day in Guifi for June 2014
1. Scale, heterogeneity and ...  

1.2. An Empirical Study on BGP

Figure 1.8: Median number of updates and withdrawals per AS per day in Guifi for June 2014

Figure 1.9: Maximum number of updates and withdrawals per AS per day in Guifi for June 2014
1.2. An Empirical Study on BGP

1. Scale, heterogeneity and...

Figure 1.10: Monthly average number of updates and withdrawals per hour in Guifi for November 2013

Figure 1.11: AWMN RIB table size per day for November 2013
1. Scale, heterogeneity and...

1.2. An Empirical Study on BGP

**Figure 1.12:** Hourly average update prefix rate per second for public Internet

**Figure 1.13:** Hourly average update prefix rate per second for AWMN
1.2. An Empirical Study on BGP

Figure 1.14: Hourly average update prefix rate per second for Guifi

Figure 1.15: Peak prefix update rate per second in the public Internet
1. Scale, heterogeneity and...

1.2 An Empirical Study on BGP

Figure 1.16: Peak prefix update rate per second in AWMN

Figure 1.17: Peak prefix update rate per second in Guifi
2 Cross-Layer Interactions and Optimizations

2.1 Measurement-Lab

In this section we present the results of a large-scale measurement campaign, where we specifically analyse the network performance as experienced by the end-user in community networks in comparison to other ISPs. To the best of our knowledge, this is the first end-to-end measurement study of community networks, as a validation of their deployment for development projects.

2.1.1 Measurement Tools

In order to evaluate the performance of a network, measurement data is necessary. In order to do this, we considered the following tools to be deployed in community networks.

From the existing measurement tools, RIPE ATLAS provides a widely deployed tool for measurement of end-user experience[13]. The project has deployed small hardware, called RIPE ATLAS probes, all over the world in thousands of locations. These provide an excellent vantage point within the network. However, the RIPE ATLAS project requires custom hardware by design, to provide very strict guarantees on measurements. For reasons of cost deploying the required amount of RIPE ATLAS probes within the community networks under study was not possible.

Project BISMark wants to measure home network performance, and it realizes this by means of custom gateway firmware[134]. It is however not feasible to deploy this specific firmware on all nodes in a community network.

NLNOG-RING is a non-profit software project designed to share shell access in participating ISP (core) networks to study and debug network behaviour[109]. The approach is elegant, however it requires connecting existing servers to this ring network in order to increase the sharing scope. Therefore, it was not an option for this study.

Finally, perfsonar-ps is a suite of measurement tools which can be deployed freely, containing a number of systems and techniques to study the performance of networks[27].

For support reasons we chose Measurement Lab as an alternative, combined with the Community-lab testbed for launching the measurements.

2.1.1.1 Measurement-Lab

We selected the Measurement-Lab[51] (M-Lab) platform to perform our measurements on. Measurement Lab is an open, distributed server platform on which researchers can deploy open source Internet measurement tools. The data collected by those tools is released in the public domain. M-Lab was founded by the New America Foundation’s Open Technology Institute (OTI), the PlanetLab Consortium, Google Inc. and academic researchers. M-Lab servers are distributed globally, but most of the servers are located in North America and Europe. The M-Lab platform offers a number of measurement tools, enabling its users to do different kinds of measurements, such as Paris and reverse traceroute[6, 78], testing for application-specific blocking or throttling, testing for traffic shaping, checking
2. Cross-Layer Interactions and Optimizations

2.1. Measurement-Lab

upload and download speeds and more. The tool we selected to use in our measurements is called “Network Diagnostic Test” (NDT) and is described in more detail below.

2.1.1.2 Network Diagnostic Test (NDT)

The Network Diagnostic Test (NDT) reports upload and download speeds, tries to determine the cause of limited speeds and checks for proxies, Network Address Translation (NAT) devices or middleboxes between the machine running the test and one M-Lab server [29]. Therefore, it can provide several objective indications of the user’s experience of an Internet connection. Below, we included the output of a typical run of the NDT tool:

```
Testing against host ndt.iupui.mlab1.ath02.measurement-lab.org
Testing network path for configuration and performance problems -- Using IPv4 address
checking for Middleboxes ........................................ Done
running 10s outbound test (client to server) ........ 4.45 Mb/s
running 10s inbound test (server to client) .......... 45.92 Mb/s
sending meta information to server ........ Done
The slowest link in the end-to-end path is a 10 Mbps Ethernet or WiFi 11b subnet
Information: Other network traffic is congesting the link
Server ‘ndt.iupui.mlab1.ath02.measurement-lab.org’ is not behind a firewall.
[Connection to the ephemeral port was successful]
Client is probably behind a firewall. [Connection to the ephemeral port failed]
Information: Network Middlebox is modifying MSS variable (changed to 1410)
Server IP addresses are preserved End-to-End
Information: Network Address Translation (NAT) box is modifying the Client’s IP address
Server says [79.131.35.128] but Client says [10.255.18.237]
```

The most important values in this output for the analysis here presented are the bitrates for the upload and download tests. What is not included in this output, is the Round-Trip Time (RTT) value. In addition to producing this output, the NDT tool logs all test data to M-Lab. This data can later be queried using Google BigQuery.

2.1.1.3 Google BigQuery

Google BigQuery is a tool to analyse big data in the cloud[137]. It offers an SQL-like interface to query data stored in the cloud. The NDT tool described above logs all results to M-Lab, which can be queried using BigQuery. The data logged by the NDT tool contains much more information than the output shown above. It contains RTT values, node identifiers, IP addresses, geolocation information and more. The node identifier can be manually specified by the user running the NDT test and can as such be used to group measurement results for each node.

2.1.1.4 Community-Lab

To gather the data required for the analysis performed in this work, we have used the Confine Community-Lab testbed. Three of the networks in this testbed have been used to preform the analysis: guifi.net, AWMN and Ninux.org. Figure 2.1 shows the Community-Lab nodes used in the experiment.

We have deployed the NDT tool on virtual machines running on all the available nodes (211) in the Community-Lab testbed. The tool was scheduled to run once every hour. The data of the test runs are logged to Measurement-Lab to allow access via Google’s BigQuery service.
2.1. Measurement-Lab 2. Cross-Layer Interactions and Optimizations

2.1.2 Results and Analysis

Out of a total of 211 nodes available in Community-Lab, 158 nodes effectively generated data. The large difference can be explained by the presence of test nodes and a large maintenance effort undertaken during the measurement campaign\(^1\).

To analyse the behaviour of nodes in a community network, we have used the measurement data on AWMN, guifi.net and Ninux.org from June 9, 2015 - June 30, 2015 (158 nodes in these three networks). We only have valid data starting from June 9, as the version of the NDT tool used prior to that date could not tag individual nodes with a unique identifier. We need to be able to do this, since we are running our tests across different community networks, which use the same (private) IP address ranges internally. This means that we cannot distinguish nodes based solely on their IP address, as different nodes (on different networks) might be using the same IP. As the Community-Lab testbed already uses internal unique identifiers for each node, we re-used those identifiers as the node identifier in the NDT tests, enabling us to uniquely match each measurement result to the node running the test. During the measurement period, 3769 download tests have been run on AWMN, 3500 on guifi.net and 3606 on Ninux. For the upload tests, we have 3917, 3500 and 3621 measurements, respectively.

2.1.2.1 Round-Trip Time (RTT) measurements

The RTT is one of the measures to assess the “quality” of a community network. Not only high RTTs are an indication of degraded Quality of Experience (QoE), but also the degree of variation in RTT is important to take into account.

Figure 2.2 shows the cumulative distribution of RTT values in the three community networks considered. The RTT measurements are taken from the download tests. The maximum RTT in the graph

\(^1\)See the WP3 deliverables for an extensive explanation of this.
is capped to 1000 ms. Figure 2.2 shows that the distribution of RTTs is very different depending on the community network considered. In the Ninux network, low RTTs are the norm, with 90% of the measurements being less than 200 ms. In AWMN, very low RTTs are uncommon. Most RTT measurements on this network are situated between 100 and 300 ms. The situation in guifi.net, however, is very different. Although a significant amount of measurements show low RTTs (less than 200 ms), the distribution has a very long tail, with 10% of the measurements taking more than 1 second.

This cumulative distribution is taken over all nodes during the entire period considered. As this only gives an indication of the RTTs one can expect in these networks, time information is not included in this graph. Therefore, figures 2.3, 2.4 and 2.5 each show the measured RTT values for two nodes in AWMN, guifi.net and Ninux, respectively. The nodes were chosen randomly amongst those that contained sufficient and realistic measurements. With this we mean that some of the nodes that ran the tests are located in data centers belonging to e.g. universities that participate in the community networks. These nodes skew the results, as they tend to have very fast and reliable Internet access, while a typical end-user node does not have this luxury. The graphs clearly show the unstable nature of the networks. In the AWMN network, both nodes behave similarly. From June 9 to June 16, they both experience a relatively high but constant RTT around 160 ms. However, on June 16, something changed in the AWMN network, causing the RTT values to become very unstable. When looking at the guifi.net network, it is clear that one node consistently has a very low RTT, whereas another node exhibits a very unpredictable behaviour. The same is true for the nodes in Ninux, in figure 2.5. Not only does this graph show that RTT values can be very different for different nodes in the network, but also that the behaviour can change over time: the first few days of the measurement period, both nodes experience constant, low RTT times, until something changes in the Ninux network, affecting only one of the two nodes.

These measurements indicate that the network quality in all three networks considered is variable,
2.1. Measurement-Lab 2. Cross-Layer Interactions and Optimizations

2.1.2.2 Throughput measurements

In addition to the RTT measurements, the NDT tool also performs throughput measurements, both in upload and download. Figures 2.6 and 2.7 show the cumulative distribution of download and upload measurements on the three community networks considered. Both graphs are capped to a maximum throughput of 400 Mbps, as they contain long tails.

Figure 2.6 shows that the AWMN network on average achieves slower download speeds than Ninux and guifi.net. However, looking at the lowest 50% of the results shows no real difference between the three networks. The Ninux network especially logs many very high-speed download tests, as 20% of the measurements exceed 400 Mbps. Although of course very promising results, we consider these to be measurement errors where the nodes were located in data centers or other less realistic locations, as explained above. Because of the setup and the configuration of Community-Lab combined with M-Lab NDT, these results are hard to filter out. Figure 2.7, on the other hand, shows that it is the AWMN network that has most high-speed upload measurements.

Figure 2.8 shows the download speed measurements on the same two nodes in the Ninux network as in figure 2.5. What is immediately visible, is that the download speeds are very variable. However, for the green node (the one with relatively stable RTT times), most measurements lie around the same average throughout the duration of the tests. The red node, on the other hand, shows more erratic behaviour. For a relatively long duration, the download speeds measured are very low. These are encountered during a time when the RTT values are very high and unstable. On the other hand, both periods (the period with erratic RTTs and the period with low throughput), do not coincide both in time (the same node may experience good or bad RTT values over time) and in place (different nodes might exhibit a different behaviour).
entirely. This shows that RTT measurement based monitoring of the quality of a community network is insufficient to assess the overall performance experienced by the end-user.

Figure 2.9 shows the upload speed measurements for the same nodes in the Ninux network. Again, the results for the red node change at a certain point in time, around the same time as when the RTT values rise in figure 2.5.

### 2.1.2.3 Comparison with other ISPs in the region

The experiment reported in this section using Community-Lab nodes embedded in these community networks has helped to raise the number of measurements contributed to M-Lab so that the Ninux (through the FusoLab AS), guifi.net (through the guifi.net Foundation, labelled as “Fundacio Privada per a la Xarxa Lliure” ...) and AWMN (as part of the LANCOM AS) can reach the threshold of 200 samples within the same month, and so enable the comparison with other ISPs in each country.

A comparison of results of our measurements in M-Lab with equivalent measurements from top ISPs in the same countries should show how well community networks can serve its users. The resulting performance, measured by M-Lab tools such as NDT, for the three community networks under evaluation is among the top eight ISPs in each country. The results for the most typical measurement of median upload and download speed and median latency are shown in Figure 2.11, Figure 2.10 and Figure 2.12 respectively. The three networks are among the top eight ISP in download speed. guifi.net is ranked first in Spain both in median upload speed and best median latency; Ninux (FusoLab) is ranked second in upload, and fourth in best latency; AWMN (part of LANCOM) is first in upload speed, 8th in best latency. In the area of Barcelona, where guifi.net has its connections to Internet carriers, the results are excellent: first in upload speed (guifi.net 7.82 Mbps, the Academic network 4.23 and Cableuropa ONO 3.31), third in download speed (Cableuropa-ONO 18.1 Mbps,
2.1. Measurement-Lab  

2. Cross-Layer Interactions and Optimizations

![Figure 2.5: RTT measurements, Ninux](image1)

**Table 2.1: Values in Figures 2.11, 2.10, 2.12**

<table>
<thead>
<tr>
<th>ISP in Greece</th>
<th>Down</th>
<th>Up</th>
<th>RTT</th>
<th>ISP in Italy</th>
<th>Down</th>
<th>Up</th>
<th>RTT</th>
<th>ISP in Spain</th>
<th>Down</th>
<th>Up</th>
<th>RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lancom (AWMN)</td>
<td>4.33</td>
<td>3.44</td>
<td>151</td>
<td>GARR</td>
<td>6.51</td>
<td>6.28</td>
<td>38</td>
<td>guifi.net</td>
<td>9.78</td>
<td>7.82</td>
<td>14</td>
</tr>
<tr>
<td>OF-Larissa</td>
<td>8.37</td>
<td>3.24</td>
<td>166</td>
<td>Fusolab (Ninux)</td>
<td>6.91</td>
<td>1.91</td>
<td>24</td>
<td>Telecable Ast.</td>
<td>14.7</td>
<td>6.69</td>
<td>18</td>
</tr>
<tr>
<td>TELLAS</td>
<td>6.69</td>
<td>0.61</td>
<td>26</td>
<td>Telecom IT</td>
<td>3.77</td>
<td>1.50</td>
<td>58</td>
<td>Euskaltel</td>
<td>15.51</td>
<td>4.55</td>
<td>15</td>
</tr>
<tr>
<td>FORTHNET</td>
<td>6.62</td>
<td>N/A</td>
<td>31</td>
<td>Convergenze</td>
<td>4.72</td>
<td>0.63</td>
<td>55</td>
<td>CESCA</td>
<td>9.79</td>
<td>4.23</td>
<td>48</td>
</tr>
<tr>
<td>Greek Research</td>
<td>8.85</td>
<td>N/A</td>
<td>20</td>
<td>UNIDATA</td>
<td>7.15</td>
<td>0.64</td>
<td>18</td>
<td>Cableeuropa-ONO</td>
<td>12.83</td>
<td>2.36</td>
<td>37</td>
</tr>
<tr>
<td>OTE</td>
<td>4.90</td>
<td>0.58</td>
<td>31</td>
<td>FastWeb</td>
<td>3.44</td>
<td>0.60</td>
<td>45</td>
<td>CableTel Galicia</td>
<td>10.97</td>
<td>1.82</td>
<td>50</td>
</tr>
<tr>
<td>CYPRUS TA</td>
<td>6.26</td>
<td>0.56</td>
<td>25</td>
<td>Vodafone O.</td>
<td>4.02</td>
<td>0.37</td>
<td>59</td>
<td>PROCONO</td>
<td>16.73</td>
<td>1.2</td>
<td>23</td>
</tr>
<tr>
<td>ON</td>
<td>7.08</td>
<td>0.55</td>
<td>35</td>
<td>NGI</td>
<td>3.58</td>
<td>0.33</td>
<td>23</td>
<td>Jazz Telecom</td>
<td>3.19</td>
<td>0.67</td>
<td>76</td>
</tr>
<tr>
<td>Hellas OnLine</td>
<td>5.59</td>
<td>0.53</td>
<td>28</td>
<td>Tiscali IT</td>
<td>4.54</td>
<td>0.31</td>
<td>48</td>
<td>Iberbanda</td>
<td>0.8</td>
<td>0.61</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Wind Tel</td>
<td>4.23</td>
<td>0.31</td>
<td>51</td>
<td>Telefonica</td>
<td>1.72</td>
<td>0.54</td>
<td>74</td>
</tr>
</tbody>
</table>

the Academic network 9.8, guifi.net 9.79) and first in best latency (guifi.net 14 ms, Vodafone 25, Cableeuropa-ONO 35).

2.1.3 Implications on Community Networks for Development

The results, although more favourable than we initially expected, all come from a well-know third party (M-Lab). Without going too far, we can easily say that the user experience perceived from users located where the probes are must be good to excellent. It is also relevant to see that, at country level, the three community networks are first or second in upload speed, with very symmetric transfer rates, a clear signal of good QoE and lack of the typical distinction between asymmetric client-oriented broadband connections versus much more expensive server-oriented connections.

Although it cannot be directly extrapolated to other countries, particularly in developing areas, the experience from many community networks around the world in all continents shows that community-driven cooperative initiatives can create network infrastructures run as a formal or informal network...
commons. Community network around the world have shown\cite{7} success in bringing connectivity to disconnected or under-served areas. As a result, these networks can satisfy the local needs of citizens, administration and businesses. Instead of extracting money from the community towards big Telecom operators, the money flow can go to local entrepreneurs and start-ups who do mainte-nance, troubleshooting, and computer help for local users, including local schools and community groups. This can create a sustainable ecosystem, even a competitive market, of local businesses that contributes to local development.

Regarding measurable impact in population from underserved areas guifi.net has collected evidences. guifi.net is mainly deployed in the region of Catalonia, starting from a rural area with very bad or nearly no connectivity \cite{12}. Statistical data is available from a large scale survey about penetration of the bandwidth and Internet access in the households of Catalonia in 2013, released by the public Catalan Statistics Institute (IDESCAT)\textsuperscript{2} detailed for each of the 42 counties in Catalonia. Despite the fact that Catalonia is about three points above the Spanish average, it is still seven points below the European average. The Catalan county with the best results and the only one above the EU average, is Osona, where guifi.net was born. It is surprising to see higher penetration than in Barcelona, the largest urban area in the region of study. Moreover, it is the only county where broadband access is above Internet access (showing that guifi.net is a local broadband infrastructure where most but not everyone use it to reach the Internet, and many also use it for internal communication). The indicators of other counties where guifi.net presence is significant, such as Bages and Baix Ebre, are also larger when compared to similar counties but where guifi.net presence is irrelevant. From additional public data sources and network statistics we estimate that about 22.4\% of the Osona inhabitants have guifi.net access: around 30,500 people.

\textsuperscript{2}http://www.idescat.cat Data source: http://www.idescat.cat/novetats/?id=1724&lang=en
While Ninux and AWMN are yet limited to their own countries, local initiatives following the guifi.net model and using its tools are starting or have developed in other regions of the world. In Africa: in the Canary Islands there are 3 operational nodes and 8 more planned, a few nodes are planned in Ethiopia, Morocco, Nigeria and Occidental Sahara. In Asia there is one operational node in Pakistan and 8 planned nodes in India. In America there are 5 operational nodes and 20 planned or under construction in Argentina, 5 operational in Colombia, for a total of 155 nodes in different states of development, including in Bolivia, Brazil, Chile, Cuba, Ecuador, El Salvador, Haiti, Mexico, Nicaragua, Paraguay, Peru, the Dominican Republic, the United States of America, Uruguay, and Venezuela.

The principles, governance mechanisms, and results, in terms of service (QoE) and coverage, shown by these community networks among many other around the world demonstrates the effectiveness of the model to serve the needs for connectivity and participation in the digital society for all, particularly for the underserved by other commercial or public offerings. These community networks become a collective good or a peer property in which participants contribute their efforts and contribute goods (routers, links, and servers) that are shared to build a computer network. Its development is a social production or a peer production because the participants work cooperatively, at local scale, to deploy an infrastructure and build network islands. The resulting infrastructure is governed as a common-pool resource (CPR) to avoid the tragedy of congestion or destruction by abuse and become a key resource for widespread participation and socio-economic development.

In fact, the International Telecommunication Union in its report[140] in 2008 proposed regulatory reforms to promote widespread, affordable broadband access, rooted in enabling and promoting diverse practices of sharing. Similarly as with competition, sharing is seen as very beneficial in multiple aspects such as passive infrastructure like ducts, civil works, towers, poles, rights of pass, radio spectrum, international gateways, undersea cables, mobile roaming, content distribution. A recent study[89] also confirms the opportunities, economic benefits, and its growth, with best practices iden-
Recent studies by the European Commission[41] show that many studies conclude that broadband has a significant and positive impact on economic growth (measured by GDP) through improvements in productivity, a positive relationship between broadband speed and GDP, with greater effects for countries and regions with lower income. The study arguments that high-speed networking service shares characteristics of a public good (such as street lighting) that can be supplied by different levels of public and private sector collaborations. Although this is part of the EU Broadband Vision, it applies globally, and community networks match with that vision.

2.1.4 Conclusions

The results from our measurements using NDT, from a user perspective, show promising network performance in general, however with high variability over time and over different nodes. This is clearly illustrated by measurement results which exceed multiple seconds.

The contribution of many more NDT measurements to M-Lab originating in community networks from our experiments has enabled a comparison with other ISPs of the performance of the Internet access service and the QoE from a user perspective. The country results show that each community network is among the top eight ISPs in his country in end-user network metrics, and top results for upload speeds, which shows the symmetry of connectivity compared to the typical asymmetric service from commercial ISPs.

Despite several community networks have proven their feasibility with sustainable local infrastructures and societal impact, our experimental results can not be considered conclusive so far. Therefore,
we believe more prolonged and extensive measurements are necessary. Additionally, it would make sense to correlate the end-to-end data with more information on the underlying network, including topology and e.g. routing algorithm knowledge. Given the requirements of real-time, correlated data, this might be particularly hard to realise.

Results from community networks and recommendations from global organisations such as ITU, Deloitte, APC and the European Commission show the great direct and indirect impact of cooperative (sharing) efforts to develop networking infrastructure and services. Supportive regulation and financial support of these initiatives can have a major impact in local socio-economic development and quality of life improvements, even with greater effects in developing areas.
Figure 2.10: Median Download throughput (Mbps) per Country and ISP, sorted by Download speed (M-Lab July 2015)
Figure 2.11: Median Upload throughput (Mbps) per Country and ISP, sorted by Download speed (M-Lab July 2015)
Figure 2.12: Median Latency (RTT in ms) per Country and ISP, sorted by Download speed (M-Lab July 2015)
2.2 Predicting Link Quality

2.2.2 Time Series Analysis to Predict Link Quality of Wireless Community Networks

In community networks, as in any other network that mixes wired and wireless links, the routing protocol must face several challenges that arise from the unreliable nature of the wireless medium. Link quality tracking helps the routing layer to select links that maximize the delivery rate and avoid traffic congestion. Moreover, link quality prediction could be a technique that surpasses link quality tracking by foreseeing which links are more likely to change its quality. Foreseeing link quality changes in advance allows routing layer to take the appropriate measures.

In this section, we focus on link quality prediction by means of a time series analysis. We apply this prediction technique on community networks, large-scale, heterogeneous, dynamic, and decentralized networks. We demonstrate that it is possible to accurately predict the Link Quality in 98% of the instances. Particularly, we analyse the behaviour of the links globally to identify the best prediction algorithm, the impact of lag windows in the results, the degradation of prediction over time, and the correlation of prediction with topological features. Moreover, we also analyse the behaviour of links individually to identify the variability of link quality prediction between links, and the variability of link quality prediction over time.

2.2.1 Experimental Methodology

2.2.1.1 FunkFeuer Network and Open Data Set

We used open data sets from the FunkFeuer network, available through the CONFINE Project open data platform. The data set is composed of OLSR information such as routing tables and network topology data of 404 nodes with 2095 links, collected during 7 days in the period from April 28th to May 4th, 2014. The data set instances were sampled every 5 minutes. It is important to notice that the total number of nodes is large. Every node has about 3.5 neighbours on average (degree) and that the largest of the shortest paths in the network (diameter) is 16. This means that there are several paths where packets have to go through a relatively high number of hops in order to reach their destination. The routing protocol must, therefore, react quickly to any change in the network topology since this will be critical to achieve high performance.

2.2.1.2 Link Quality

ETX [46] is a link metric that measures the expected number of data transmissions required to send a packet over that link, and is widely used in several mesh network protocols. The ETX of a particular link is calculated as: \( ETX = \frac{1}{LQ \times NLQ} \), where LQ and NLQ stand for the Link Quality and the Neighbour Link Quality of that link, respectively. The Optimized Link State Routing (OLSR) protocol uses the ETX to choose, for each device and packet, the next hop. The LQ assumed by OLSR is defined as the fraction of successful packets (HELLO) that were received by a node from a given neighbour within a certain time window, while the NLQ is the fraction of successful packets that were received by the neighbour within a time period. We focus on predicting the LQ, as the NLQ is directly derived from the neighbour-node LQ and the ETX can be easily calculated using both predictions.

For our simulations we only considered the links that have experienced some variations in the link quality \((LQ < 1)\) at any instant over time are represented, which represent approximately half of the
2. Cross-Layer Interactions and Optimizations

2.2. Predicting Link Quality

We consider the overall number of links (1068 of 2095). Moreover, we also discarded those links that did not have enough samples to perform the time series analysis. Therefore, our study only considered 1032 links.

### 2.2.1.3 Time Series Analysis

A time series is a set of data collected over time with a natural temporal ordering. It differs from typical data mining or machine learning applications where the ordering of data points within a data set is not important. Time series analysis is the process of using statistical techniques to model and explain a time-dependent series of data points. Similarly, time series forecasting is a method that uses a model to generate predictions (forecasts) of future events based on known past events. In our case, we used more than one prediction algorithm so that we do not rely on a specific learning technique. We applied four of the best well-known approaches [1]: Support Vector Machines (SVM), k-Nearest Neighbours (kNN), Regression Trees (RT), and Gaussian Processes for Regression (GPR).

The Support Vector Machines (SVM) algorithm has recently become one of the most popular and widely used methods in machine learning. It performs a linear or nonlinear division of the input space and builds a prediction model that assigns target values into one or another category. The k-Nearest Neighbours (kNN) algorithm is one of the most simple machine learning algorithms as it makes no assumptions on the underlying data distribution. This algorithm takes the k data-points closest to the target value and picks the most common one. Regression Tree (RT) is a type of decision tree algorithm where the target value can have continuous values. This method recursively partitions the data space and runs a simple prediction model within each partition. The main advantages of tree algorithms are that (1) they produce fast results, and (2) they are resistant to irrelevant values. Finally, the Gaussian Processes for Regression (GPR) algorithm is a very flexible approach that can easily deal with complex data sets. In this model the output is a normal distribution denoted by the mean and variance terms. The target value is represented by this mean value and the variance can be interpreted as a measure of its confidence.

Usually, classification studies assess the predictive power of their model by using Mean Absolute Error (MAE), widely used in related work. MAE is a common method to evaluate the performance of prediction approaches that gives the same weight to all individual differences. This metric is calculated through the following formula:

\[
MAE = \frac{\sum |\text{predicted} - \text{actual}|}{N}
\]

We applied a T-test to mean values for independent samples (at 95% confidence level) in order to compare the classification algorithms using the MAE. After this analysis, p-values smaller than 0.05 indicate that the means are significantly different, and therefore, we would reject...
2.2. Predicting Link Quality

2.2.2. Analysis of the Impact of the Lag Window Size

Lagged variables are the main mechanism by which we can capture relationships between past and current values of a series using propositional learning-algorithms. They create a "window" or "snapshot" over a time period. Basically, the number of lagged variables determines the window size (i.e. the number of previous samples used to make a new prediction, with samples every 5 minutes).

This analysis was performed to check the impact of the lag window in the prediction of the next link-quality value. Figure 2.14 shows the average MAE per link of the RT algorithm using the same experimental setup as in the previous test (1728 and 288 instances for training and testing, respectively) but in this experiment we used a lag window size ranging from 1 to 24 instances. We obtained good results using window sizes between 3 and 18 (Figure 2.14). The worst results were obtained for window sizes of 1 and 24. Nevertheless, these results are similar or even better than the results obtained by the other algorithms. Thus, we can sustain the claim that RT is the best candidate.
2. Cross-Layer Interactions and Optimizations

2.2. Predicting Link Quality

We also analysed the error variability for each window size and concluded that all window sizes achieved a similar performance for most of the links. Although the values for the median and the first quartile are similar for all window sizes, the values of the third quartile and the outliers slightly differ. These differences in the variability of errors lead to the differences in the average MAE. Finally, we tried to find the best lag window size for mean values using the T-test for independent samples (at 95% confidence level). After this analysis, we could not reject the null hypothesis at 95% of significance. Consequently, our results do not provide clear evidence about what is the best window size.

2.2.2.2 Degradation of the Regression-Tree Model over Time

This experiment was performed to evaluate the accuracy of the prediction models over time. Figure 2.15 shows the average MAE of the overall network and its approximation to a linear function. It shows the results of the RT algorithm using the same setup as the baseline experiment (a lag window size of 12 instances and a training dataset of 288 instances) but using a test dataset ranging from 144 (1/2 day) to 1728 (6 days) instances. We used linear regression to compute the parameters and estimate the goodness-of-fit, obtaining these parameters: slope = 0.0212 and b = 0.0132 (depicted by a line in Figure 2.15). Clearly, we can affirm that a linear function can be used to model the degradation of the RT over time.

![Figure 2.15: RT Mean Absolute Error of the whole network for several test dataset sizes.](image)

We have also observed that the variability of errors increases linearly with the number of instances of the test dataset. For this reason, it is important to train the model again after a certain period. Due to the fact that both the MAE and the error variability follow a linear function, we could easily determine a trade-off between the resulting error and the frequency of updates to the model.

The global analysis of nodes allowed us to determine the general behaviour of the LQ by considering the entire network as a whole. However, some individual groups of nodes and links seem to present a divergent behaviour. Therefore, it would be interesting to further analyse the behaviour of such groups, in order to be able to characterize the observed behaviour heterogeneity. Nonetheless, from the global analysis of nodes performed in the previous sections, we are able to draw some interesting lessons and conclusions:

It is also important to notice that any LQ value should lie in the range 0 to 1. Therefore, the predicted values that are greater than 1 or lower than 0, should be considered to be exactly 1 or 0 respectively. Moreover, if the LQ is very poor, it makes no sense to make a prediction (it would be better to disable the predictor). All these observations can be applied to improve the prediction results.

Deliverable D.4.4
2.2.3 Analysis of Individual Links and Nodes

In this section, we perform an analysis of individual links and nodes. Our aim is to identify and analyse data individually, in order to classify the nodes and links according to their behaviour. Thus, we try to answer questions such as: “Do some nodes and links show a similar performance?”, and “Is it possible to group nodes and links in clusters according to similarities in their behaviour?” The first part of this section analyses the variability of the LQ prediction among links. The last part, on the other hand, analyses the variability of the LQ prediction over time. The aim is to determine if this prediction remains constant with time or if it presents variations, and in the latter case, try to classify the behaviour of these variations.

The next analysis was intended to evaluate the behaviour of the four algorithms (SVM, RT, kNN, and GPR) for variations in the LQ in steps of 0.1, and therefore, to check if the behaviour changes for any particular range of LQ values (i.e., to verify if RT is not the best algorithm for a specific range). This study extends the global analysis presented in Figure 2.13. Figure 2.16 shows the results obtained.

We preformed a more detailed analysis of the data for RT and SVM for LQ values in the 0.9-1 range. These results show that SVM has better (lower values) median and 3rd quartile values, the mean values are almost the same for both methods, and SVM performs worse in the 1st quartile, maximum, and standard deviation results. This behaviour can be explained due to the fact that SVM presents some very high values that produce a significant increase in the mean and standard deviation values. Despite this, results in Figure 2.16 confirm that for 87.5% of the links SVM performs better than RT.

![Figure 2.16: Average Mean Absolute Error versus Link Quality in steps of 0.1.](image)

As we can observe RT is the best solution for LQ values below 0.9, but in the range from 0.9 to 1, SVM shows a slightly better performance.

2.2.3.1 Variability of Link Quality Prediction over Time

Our first task, in order to identify the different kinds of links (according to the temporal evolution of their LQ) was to determine the best number of clusters. To do so, we selected only those links that presented changes in the LQ (we excluded links with a constant value of $LQ = 1$). It is important to notice that there is not a clear criterion to decide the "best" number of clusters. However, we have to
take into consideration that we are not only interested in an analytical interpretation of the common features of the links within a cluster, but also in a human-readable interpretation, and when the number of clusters is large, the amount of elements belonging to some clusters are too small, making more difficult for humans to interpret the criteria used to classify the links that belong to each cluster. For this reason, we choose $K=4$.

The next step involved applying the K-means algorithm, with $K=4$ clusters. In Figure 2.17 we can observe a sample of the LQ time series for each one of the four clusters. The main features of the links in each cluster are listed below:

- Cluster 1 (618 links): links with LQ very close to 1 and small oscillations.
- Cluster 2 (148 links): links with large oscillations in LQ.
- Cluster 3 (206 links): links with small LQ oscillations, but LQ values far from 1.
- Cluster 4 (55 links): links with ON/OFF behaviour (with small oscillations).

![Figure 2.17: Sample of LQ time series for each of the 4 clusters from K-means algorithm.](image)

Once we identified the four clusters, we could analyse their behaviour. We have analysed how well each one of the four prediction methods (SVM, RT, kNN, and GPR) performs for each of the four cluster types. The results are available in Figure 18. Comparing these results with the ones depicted in Figure 2.13, we confirm that the global results obtained previously, where RT was the best prediction method, are now still valid for clusters 2 and 3. For cluster 1, nevertheless, both RT and SVM have a similar performance. A different result is obtained for cluster 4, where RT is the third best method, after kNN (the second best) and SVM (the best). Previously, in Figure 2.16, we had also observed a case with SVM performing better than RT, but now the difference is even more significant.

![Figure 2.18: Prediction error of each method for each cluster, depicted as box plot.](image)
2.2.4 Lessons Learned and Conclusions

This study demonstrates that time series analysis is a promising approach to accurately predict LQ values in community networks. This technique can be used to improve the performance of the routing protocol by providing information to make appropriate and timely decisions to maximize the delivery rate and minimize traffic congestion.

We analysed results from four learning algorithms (Support Vector Machine, k-Nearest Neighbours, Regression Tree and Gaussian Processes for Regression) that model time series. All algorithms achieved percentages of success between 95% and 98% when predicting the next future value of the LQ, with the Regression Tree being the best one. We also observed that the size of the training data set is a key factor to achieve high accuracy in the predictions. The bigger the size of the data set, the smaller the degradation of the error over time.

The global analysis of the LQ behaviour we performed in this work, allowed us to identify the best prediction algorithm (Regression Tree) and metric (Mean Absolute Error), and to understand the impact of lag windows in the prediction (although we were not able to determine the best lag window size). It also helped us to evaluate the accuracy of prediction some time steps ahead into the future (it seems possible to predict the LQ some steps ahead), and the degradation of prediction over time (the degradation follows a linear function: the larger the size of the training dataset, the smaller the error). We also analysed the behaviour of links individually to identify the variability of the LQ prediction between links (for LQ values within range 0.9-1, SVM performs better than RT in 87.5% of links) and over time (we identified four kinds of links according to the results of a time series clustering; and SVM continued performing better than RT in three of the four clusters).

The LQ prediction analysis we performed provided us with some interesting results. This helped us reach the following conclusions:

- In general, the RT predictor achieves the best results.
- The majority of the LQ values lie within the range from 0.9 to 1.
- For clusters 1, 3, and 4, the SVM predictor performs better than the RT for LQ values within the range from 0.9 to 1.
- For cluster 2, RT always produces the best results for the whole range of LQ values.
- The predictor generates LQ values out of the range from 0.0 to 1.0, which offers an opportunity to improve the predictor.

We also analysed the topological features, the node degree does not present a correlation with the LQ, but there are indications that the edge betweenness affects the LQ results. However, our results are not conclusive enough to allow us to apply this correlation in the prediction process.

2.3 Time Series Analysis to Predict Path Quality of Wireless Community Networks

Community Networks features (large, heterogeneous, dynamic, decentralized) raise challenges of interest for researchers [25]. One of the most important challenges is the effect of the asymmetrical features and unreliability of wireless communications on network performance and routing protocols. Many metric-based routing protocols for mesh networks that track Link Quality (LQ) and select higher-quality links have been proposed to minimize traffic congestion and maximize delivery rate [151], [52], [84], [122]. Hence, when routing packets through an unreliable network, LQ tracking is definitely a key method to apply. Moreover, routing algorithms should avoid weak links as soon
2. Cross-Layer Interactions and Optimizations  

2.3. Predicting Path Quality

as possible [126], and whenever possible [145]. LQ estimation [10] (or prediction [91]) approach increases the improvements achieved by LQ tracking in routing performance. Usually, real-time metrics do not provide enough information to detect the degradation or activation of a link at the right moment. Therefore, prediction is needed to foresee LQ changes and take the appropriate measures.

End-to-End Quality (EtEQ) or Path Quality extends the LQ concept to the full communication-path (between sender and receiver) and it is computed based on the quality (ETX) of the individual links that conform the communication path. In this work, we want to analyze if our previous work on LQ [104] is applicable to the full communication path (EtEQ tracking and prediction) and determine what differences exist between individual LQ and EtEQ. To the best of our knowledge, no previous work explores EtEQ prediction in the routing layer of large-scale, distributed, and decentralized systems.

2.3.1 Background

It is well known that selecting high quality links in real-world networks composed by wireless channels with unpredictable conditions is a big challenge for achieving high delivery rate and performance. Our research goal in this work is to assess if the improvements previously achieved by applying LQ tracking and prediction techniques, are also achievable when considering the full communication path (End-to-End Quality). To evaluate the potential benefits of this proposal, we first analyze the characteristics of a well known, and free Wireless Mesh Community Networks (WCN) that deals with the Optimized Link State Routing (OLSR) protocol to maintain the network topology.

FunkFeuer [57] is a non-commercial project maintained by computer enthusiasts that install Wi-Fi antennas across rooftops in several places of Austria that are relatively close to each other (Vienna, Graz, Weinviertel and Bad Ischl). Currently, there are around 2000 wired and wireless links and every week new antennas are added to the network. FunkFeuer uses the OLSR-NG routing protocol, which expands the capabilities of the OLSR protocol and makes it highly scalable. In fact, some members of the FunkFeuer network are actively involved in the olsr.org open source project as developers, testing the protocol in the network. Furthermore, the FunkFeuer network maintains open data sets, available also through the CONFINE Project open data platform [40], which were used in this work. The chosen data set is composed of OLSR information such as routing tables and network topology data, collected during 7 days in the period from April 28th to May 4th, 2014. Every node has about 3.5 neighbors on average (degree) and that the largest of the shortest paths in the network (diameter) is 18. This means that there are several paths where packets have to go through a relatively high number of hops in order to reach their destination. The routing protocol must, therefore, react quickly to any change in the network topology since this will be critical to achieve high performance.

As stated before, FunkFeuer assumes OLSR, a link-state routing protocol. The nodes in an OLSR network periodically exchange routing information to maintain a map of the network topology. The Multi Point Relays (MPRs) are the network nodes selected for propagating the topology information. The use of MPRs reduces the number and size of the messages to be flooded during the routing update process. The key issue is that every node maintains a connectivity map for all the network. Exploiting this OLSR property, FunkFeuer publishes its complete network information from the point of view of a single node (ego-network). While convenient for data collection, this method comes with the downside that the data set is biased and does not represent the real network state, since the time for event propagation throughout the network is not negligible. In other words, the higher the distance between a node and an event that happens in the network, the later this event will be present in the nodes global view. Therefore, prediction of path changes can improve local node routing decisions, since it can provide the node with an estimation about the future local and remote events.
ETX \([43]\) is an active-probing link metric, designed for MANETs and widely used in mesh protocols, based on estimating the bidirectional loss ratios of a link. The ETX value of a link is the number of expected transmissions needed to send a packet over the link and is calculated as follows: \(ETX = \frac{1}{(LQ \times NLQ)}\), where \(LQ\) and \(NLQ\) stand for the "Link Quality" and the "Neighbor Link Quality" of that link, respectively. The ETX of a path is defined as the sum of the ETX value of the links that form the path. As a result, ETX is always greater or equal to the actual number of Hops in the path. The difference between the path ETX and the number of Hops of the path is the expected number of losses.

The OLSR protocol uses ETX to choose, for each device and packet, the next hop. Concerning physical links, the LQ assumed by OLSR is defined as the fraction of successful packets (HELLO) that were received by a node from a given neighbor within a certain time window, while the NLQ is the fraction of successful packets that were received by the neighbor within a time period. Concerning paths, OLSR calculates the ETX of all the possible paths from the source to the destination, as described above, and chooses the one with minimum ETX value. That is to say, the ultimate decision to be made by OLSR will be about the selected paths; therefore, the final metric value that will be the subject of comparison will relate to the whole path. As a result, prediction of the path ETX will allow more efficient routing decisions in an unstable environment, taking also into account the ego-network measurement effect explained previously. It is important to point out here that LQ as defined by ETX and studied in this work ignores the parameters of transmitted-packets size as well as link transmission rate. Consequently, this work considers that the significant path quality parameter is packet loss.

### 2.3.2 Link Quality Prediction in Community Networks

Prediction is a very well-known technique that has been successfully applied in several areas of computer science. For instance, in computer microarchitecture it has been shown that is a key issue for achieving high performance. Web services is another topic that takes advantage of prediction to minimize the latency of accessing a web page. Computer networks have been also aware of prediction techniques such as routing traffic reduction \([102]\), \([103]\), \([68]\), energy efficient routing \([98]\), \([79]\), \([92]\) or LQ estimation.

LQ tracking has been previously applied in several scenarios in different ways \([151]\), \([52]\), \([84]\), \([122]\) to select higher quality links that maximize delivery rate and minimize traffic congestion. LQE (Link Quality Estimators) \([10]\), \([91]\) are in charge of measuring the quality of the links between nodes based on physical or logical metrics. Physical metrics focus on the received signal quality and logical metrics focus on the percentage of lost packets. LQE with metrics like LQI (Link Quality Indication) \([55]\), SNR (Signal-to-Noise Ratio) \([93]\) or RSSI (Received Signal Strength Indication) \([131]\) fit in the former category, whereas metrics like RNP (Required Number of Packets) \([34]\), ETX (Expected Transmission Count) \([43]\), \([34]\) or PSR (Packet Success Rate) \([151]\) fit in the latter. All these metrics can be used by LQE in isolation or as a combination of some of them \([10]\), \([91]\), \([124]\) to select the more suitable neighbor nodes when making routing decisions. LQ prediction is used in addition to LQ tracking to determine beforehand which links are more likely to change their behavior. Although LQ prediction is not identical with EtEQ prediction some of the above techniques can be very similar \([151]\), \([43]\), \([55]\), \([93]\), \([34]\), \([124]\). This relation is even more direct in the case of ETX EtEQ, studied in this work, which is a linear function of the ETX LQ. As a result, the routing layer can take better decisions at the appropriate moment.

End-to-End Quality has not been widely considered in the past. In fact, the main efforts have been
focused on determining the cost of a single link and then extended to the path by assuming the cost of a path as the sum of costs of several links. For instance, MARA [118] has been proposed as a method to make more accurate routing decisions by combining route quality evaluation and automatic rate selection. To do this, MARA computes ETX of every link at every available rate, estimating metrics as SNR, packet error rate, and probe packet size. End-to-End Retransmissions (EER) operation model [14] claims that the total cost of a path cannot be expressed as a linear sum of individual link costs. Therefore, variations of this simple formulation as incorporating error rate in the link cost are proposed. This led the authors to achieve significant energy savings compared to traditional minimum energy approaches. ETOP [75] has been proposed as a path metric to determine reliable end-to-end packet delivery. In the same way as EER, this work assumes that the cost of a path does not only rely on the quality of individual links but also on their relative position on the path. Finally, EED/WEED [86] is another approach that was designed as a link/path metric to select paths with minimum end-to-end delay and high network throughput but considering load balancing of routing. In any case, there is no work concerning prediction of path quality in a WMCN.

There are some relevant works that must be paid special attention as they are related to our study: Wang, et al [146], Maccari and Cigno [94], Cunha et al [45] and Millán et al [104]. Wang et al [146] introduces the MetricMap mechanism, that is is fundamentally a routing protocol for wireless sensor networks that uses a learning-enabled method for LQ assessment. Based on the observation that high traffic rates make tracking link qualities more difficult, this protocol uses prediction methods to estimate them in advance. In a first stage, a machine-learning algorithm is applied to classify link qualities. Two types of classifiers are evaluated: a decision tree and a rule-based classifier. The data used to train both classifiers were preclassified offline based on a LQ indicator and other metrics that represent some features of the nodes. In a second stage, the MetricMap routing protocol estimates the LQ at runtime by replacing the current traffic information with the rules collected offline from the classifiers. Results show that MetricMap can achieve a significant improvement on the data delivery rate in high traffic rate applications.

Maccari and Cigno [94] have considered the FunkFeuer network focusing on link layer properties, topological patterns and routing performance. They have analyzed the quality of the routes and proposed a couple of techniques to select the Multi-Point Relay (MPR) nodes in the Optimized Link State Routing (OLSR) protocol. Traditionally, routing algorithms assume that mesh networks are fairly stable but we have also observed that this is not completely true. Therefore, MPR selection should consider the path variability of a node instead of selecting them by agreement. We also analyze the quality of routes but focused on estimating its future quality, to improve the routing layer to select links that maximize the delivery rate and minimize traffic congestion.

Cunha et al [45] proposed a simple strategy for improving routing in the Internet domain that moves in two ways: first, it detects path changes (NN4 approach) and then it remaps these paths once a change is detected (DTRACK approach). Therefore, DTRACK adapts path sampling rates to minimize the number of missed changes based on NN4’s predictions. This predictor is based on Rule-Fit, a well-known machine learning technique, that takes into account inputs as route prevalence, route age, number of past route changes, and number of times a route appeared in the past. Results show that NN4 is not highly accurate but it demonstrates the potential of prediction to improve the routing layer when making routing decisions.

Finally, Millán et al [104] analyze the behavior of LQ prediction in the routing layer of large-scale, distributed and decentralized systems. In summary, the main contributions of this work are (1) the employ of time series analysis to estimate link quality in the routing layer for real-world WMCN, (2) the detailed evaluation of results, assuming several learning algorithms to show the potentiality of
2.3. Predicting Path Quality

2.3. Cross-Layer Interactions and Optimizations

2.3. Predicting Path Quality

2.3.3 Analysis of Results

2.3.3.1 Experimental Framework

As stated before, we deal with time series analysis to estimate link quality in the routing layer for real-world WMCN. To do this we assume the FunkFeuer experimental network and an OLSR data set of several nodes and links.

A time series is a set of data collected over time with a natural temporal ordering. It differs from typical data mining or machine learning applications, where the ordering of data points within a data set is not important. Time series analysis is the process of using statistical techniques to model and explain a time-dependent series of data points. Similarly, time series forecasting is a method that uses a model to generate predictions (forecasts) of future events based on known past events. In our case, we used more than one prediction algorithm so that we do not rely on a specific learning technique. We applied four of the best well-known approaches [1]: Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Regression Trees (RT) and Rule-Based Regression (RBR).

The Support Vector Machines (SVM) algorithm has recently become one of the most popular and widely used methods in machine learning. It performs a linear or nonlinear division of the input space and builds a prediction model that assigns target values into one or another category. The k-Nearest Neighbors (kNN) algorithm is one of the most simple machine learning algorithms as it makes no assumptions on the underlying data distribution. This algorithm takes the k data-points closest to the target value and picks the most common one. Regression Tree (RT) is a type of decision tree algorithm where the target value can have continuous values. This method recursively partitions the data space and runs a simple prediction model within each partition. Finally, the Rule-Based Regression (RBR) algorithm is similar to a decision tree approach but it is a stronger model that provides rules that are often potentially more predictive.

We applied training and test sets validation to evaluate the predictive accuracy of the models. After a model is processed using the training set, it is tested by making predictions against the test set. For this purpose, we used the Weka workbench system [66], a framework that incorporates a variety of learning algorithms and some tools for the evaluation and comparison of the results. Weka has a dedicated environment for time series analysis that allows forecasting models to be developed and evaluated. The Weka’s time series framework takes a machine learning or data mining approach to model time series by transforming the data into a form that can be processed by standard propositional learning algorithms. To do so, it removes the temporal ordering of individual inputs by encoding the time dependency via additional input fields.

Usually, classification studies assess the predictive power of their model by using Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE), both widely used in related work. We assume MAE in our experiments as it is a common method to evaluate the performance of prediction approaches, that gives the same weight to all individual differences. This metric is calculated through the following formula: 

\[
\text{MAE} = \frac{\text{sum}(\text{abs}(\text{predicted} - \text{actual}))}{N}.
\]
2.3.3.2 Comparison of Learning Algorithms Based on Time Series

As stated before, we want to explore whether time series analysis can be used to predict future End-to-End quality values. To do this, we applied four well-known approaches: Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Regression Trees (RT) and Rule-Based Regression (RBR).

![Figure 2.19: Average Mean Absolute Error (MAE) of the paths.](image)

Figure 2.19 shows the average Mean Absolute Error (MAE) per path using a training data set of 2016 instances (7 days), a test data set of 288 instances (1 day) and a lag window composed of the last 12 instances. This test was performed to verify whether time series learning algorithms could predict consecutive EtEQ values. These results show that we achieved the best accuracy for the Rule-Based Regression (RBR) and the worst for k-Nearest Neighbors (kNN). Regression Trees (RT) and Support Vector Machines (SVM) also moves very close to RBR results. The MAE per path is 0.24 for RBR and 0.5 for kNN. We applied a T-test to mean values for independent samples (at 95% confidence level) in order to compare the classification algorithms using the MAE. After this analysis, p-values smaller than 0.05 indicate that the means are significantly different, and therefore, we would reject the null hypothesis of no difference between the means. Consequently, we can claim that time-series analysis achieves high percentages of success and that among them, RBR seems to be the best candidate to make predictions.

![Figure 2.20: Mean Absolute Error (MAE) of the EtEQ predictions as boxplot.](image)

We also analyzed the error variability of each algorithm and represented the results using boxplots. Three of the four algorithms achieved similar performance for most of the links (RT, RBR and SVM), as shown in Figure 2.20. Although, RT may present some outliers, the differences among median, 1st quartile and 3rd quartile are minimal. On the other side, kNN presents different behavior compared
to the others. In this case outliers present larger errors that increase the average values and change the overall evaluation of the algorithm. The rest of the work, we assume RBR algorithm to show the potential benefits of predicting EtEQ by means of a time series analysis.

### 2.3.3 EtEQ Prediction with Rule-Based Regression

We proceed next to study more in depth the EtEQ using the RBR algorithm, in order to discover how can we reach a satisfactory level of prediction.

Figure 2.21 presents boxplots of the MAE of path ETX prediction with RBR versus the number of Hops corresponding to the paths. We used the same training data set as section IV.B. Even though ETX values for 10, 11 and 13 hop paths have a high dispersion, our prediction manages to successfully predict a big percentage of the fluctuations. For instance, the dispersion of 13 hop path ETX is 6, while the error of its prediction has a maximum value of 3. For the rest of the paths, the MAE has maximum value less than 1, resulting to a meaningful prediction.

![Figure 2.21: Distribution of RBR Mean Absolute Error (MAE) as boxplot.](image)

Figure 2.22 provides a more detailed analysis of the prediction accuracy. We can see that the average ETX value and the average prediction value are very close, even overlapping during the first half of prediction test. A better estimation for the deviation of the individual path values is given by the average absolute error line. Notice that the deviation remains less than 0.5 throughout the whole prediction, which is a great achievement. Nevertheless, the potential impact of this small error in routing decisions should be further studied.

![Figure 2.22: Evolution of the average ETX, the average prediction and the average absolute error.](image)
Another characteristic of the prediction revealed by Figure 2.22 is that after 100 steps of prediction (between 8 a.m. and 8:30 a.m.) the absolute error presents an increasing trend. In order to verify this assumption we performed two more prediction tests. From 12 a.m. to 12 p.m. (Figure 2.23.a) and from 12 p.m. to 12 a.m. (figure 2.23.b), using as training data set the 2016 more recent instances (7 days before prediction starts), as test data set 144 instances (half a day) and a lag window composed of the last 12 instances. The results obtained are almost identical to the results of Figure 2.22, leading to the conclusion that the ETX oscillations are indeed affecting the prediction. Therefore, we plan to explore in the future how accuracy could be increased by deploying two different predictors, for day and night.

![Graph](image)

**Figure 2.23:** Evolution of the average ETX, the average prediction and the average absolute error for: a)12 am-12 pm, b)12 pm-12 am

### 2.3.3.4 Prediction of Some Steps Ahead

This analysis was performed to explore if time series analysis and prediction can be used to predict the value of EtEQ some time steps ahead into the future.

Figure 2.24 shows the average MAE of paths. It shows the results of the RBR algorithm using the same setup that the baseline experiment (a lag window size of 12 instances, a training dataset of 2016 instances and a test dataset of 288 instances) and then predicting from 1 to 10 time steps into the future. The results obtained were good for the majority of the tests. As we can observe, the average MAE grows very slowly. It seems possible to affirm that we could predict successfully the EtEQ several steps ahead in time.

Once more, we analyzed the variability of errors for each number of steps ahead using a box plot, shown in Figure 2.25. Although the values for the median and the first quartile are similar for all
steps ahead considered, the values of third quartile and outliers (no depicted) grow with the number of steps. These differences in the variability of errors lead to the differences in the average MAE.

![Figure 2.24](image)

**Figure 2.24:** RBR average Mean Absolute Error (MAE) of the EtEQ predictions.

![Figure 2.25](image)

**Figure 2.25:** RBR Mean Absolute Error of EtEQ predictions, depicted as a boxplot.

### 2.3.4 Conclusions

This study demonstrates that time series analysis is a promising approach to accurately predict EtEQ values in community networks. This technique can be used to improve the performance of the routing protocol by providing information to make appropriate and timely decisions to maximize the delivery rate and minimize traffic congestion.

We have presented results from four well known learning algorithms that model time series. All of them achieved high percentages of success, with average Mean Absolute Error values per link between 2.4% and 5% when predicting the next value of the EtEQ. We also analyzed the error variability and found that three of them presented similar performance (RT, RBR and SVM), whereas kNN performs worse due to outliers with larger errors. A more detailed study of RBR prediction shows an average absolute error less than 1. We have also observed differences in the prediction behavior during day and during night, as it happens with actual ETX values.

As future work, we want to extend this analysis to other community networks [111], [5] to evaluate if the observed behaviour could be generalized. Moreover, we plan to identify which paths contribute most to the errors in the EtEQ prediction and to understand what factors make it more difficult to predict them. We also want to study the impact of errors in routing decisions, and study a solution with
two different predictors for day and night. Finally, we plan to improve the prediction process discarding those paths whose relation between EtEQ and prediction accuracy is above a certain threshold.

### 2.4 Minstrel Link Layer Rate Control Algorithm

The Linux Mac80211 Wi-Fi stacks mostly use an algorithm called Minstrel for selecting the outgoing unicast data rate. The Minstrel algorithm [152] has replaced most other rate control algorithms in the last years.

The algorithm keeps statistics about the throughput per neighbor for each available bit rate. It combines the raw bit rates with the measured frame success- and loss rates based on received and missing acknowledgments. Minstrel then uses this knowledge to decide on the best bit rates in terms of throughput. The first attempt to send a unicast frame is always done using the selected bit rate. Retransmissions are sent at lower bit rates with better success rates. The last retransmission is sent at the base rate. In order to update the bit rate statistics, one out of ten frames is sent with a higher initial bit rate than currently selected by the algorithm. This means that metrics that rely on link layer bit rates need to make sure that there is a certain amount of unicast traffic on every link in both directions. Otherwise, the data returned by the rate control algorithm will not reflect realistic conditions.

Thus, the Minstrel rate control algorithm directly influences the performance of the network and especially routing metrics like DAT, described in section 3.5. We observed that Minstrel also reduces the bit rate of the links when frame collisions occur. However, this increases the chances for further frame collisions. We believe that a well designed strategy would differentiate regarding the loss cause and ignore frame collisions. A modification of the Minstrel algorithm that is collision aware would be future work.
3 Self-management

3.1 Routing robustness

In this section we study and present mechanisms to increase the robustness of network connectivity. More specifically, we focus in the detection of traffic forwarding faults, following the approach proposed by Mizrak et al. [106]. Mizrak proposed to split the detection problem into three different sub-problems: traffic validation, distributed detection and response. And for each of those, we will see which solutions have been previously proposed and present new ones when those do not satisfy our needs.

3.1.1 Background

Connectivity is the result of a collaborative effort: to reach a far away node we need that every node in the path cooperates and forwards that packet to the next hop. However, connectivity maybe lost for many reasons, such as misconfiguration or faulty hardware; and once the connectivity is lost it is difficult to figure out the reason. Our goal is to present a solution that allows us to detect this problem taking into account the intrinsic characteristics of WCN: openness, diversity, nodes with limited capabilities and several antennas, and distributed management.

In the literature, the problem of routing robustness is usually faced from a resilience point of view or from a detection perspective. Solutions to achieve resilience usually modify the network stack to use some kind of redundancy to ensure that traffic is delivered. A classical example is robust flooding [119], which uses network redundancy and buffers to ensure packets are delivered. Another solution is ACR [148], which proposes a routing protocol that provides multiple paths, so that when a path does not perform as expected, an alternative path can be used. However, given the nature of WCN, which are managed in a distributed manner and are extremely diverse, changing the network stack does not seem reasonable, and because of this reason, we focus instead on detection-based solutions purely based on the analysis of traffic in the data-plane.

As mentioned before, detection solutions must address three different sub-problems:

- **Traffic Validation**: Which information should be collected and how it should be used to recognise a misbehaving path, node or link?

- **Distributed Detection**: How the information collected by Traffic Validation is shared through the network ensuring that there cannot be false accusation nor collusion?

- **Response**: How routing takes this information into account to improve traffic delivery?

On Traffic Validation, a simple mechanism to detect a node or area that does not forward traffic properly is counters: if the number of packets entering does not match the number of packets leaving (excluding its own traffic), then we can conclude that the node is faulty. This is the approach taken by WATCHERS [24] at node level and AudIt [4] at AS level. Counters can only detect packet dropping, while packet modification and corruption goes undetected.

If we are considering a path, another widely used mechanism is acknowledgements. Some solutions are purely end-to-end, where the destination acknowledges every packet [15, 127], or some of the
packets [15, 121] and the fault is assigned to the whole path. Others go further and when a failure is detected, a mechanism is triggered to pinpoint the specific faulty link [9, 115]. And others require the involvement of every node in the path [8], to directly detect the faulty link. From a more local perspective, acknowledgements have been also used in 2ACK [88] and NACK [133], where nodes monitor their next hop by requiring acknowledgments from the next-next hop. The main disadvantage of ACK-based solutions is that they require path information to localize the specific link or node that is failing, but that is not available for distance vector protocols.

In the context of wireless networks, many authors have proposed to make the most of the medium characteristics and determine if a node actually forwards traffic simply by overhearing [154, 100]. This solution, though attractive, makes the assumption that nodes do not have several antennas, on different directions and using different channels, a configuration that is quite common in WCN.

Finally, other summary mechanisms more robust than counters have been studied, so that by comparing them a node’s behavior can be characterized: sampling [48, 59], packet-fingerprinting [106], packet-reception bitmaps and homomorphic linear authenticator [132] and sketches [59, 156]. However, most of them require clock synchronization to agree on the time interval to monitor.

On the sub-problem of distributed detection, many solutions consider a reliable central entity, which establishes a secure channel with the involved parties and draws conclusions from the given evidence [48, 156, 132] and others flood the evidence through the network [24]; both cases do not scale on a wireless network of thousands of nodes and no central authority. Then, on some solutions only the source uses the learned evidence, such as [8] or [59], but that approach does not fit with the collaborative spirit of a WCN. Finally, Mizrak [106] proposes that each nodes monitors its neighborhood and then maintains network connectivity by disconnecting from faulty nodes.

To conclude, for the sub-problem of Response there are several solutions that propose to feed the routing protocol with the learnt evidence in the form of metrics [117, 101, 9, 121], whereas others prefer creating a trust or reputation system [100].

On the following subsections we will first study the accuracy of sketches, a traffic summary mechanism. Then we will see how they can be used as a traffic validation function even when there is no clock synchronization. Finally we will present a distributed detection protocol that works even when the path is not known, but still considering false accusation and collusion.

### 3.1.2 Traffic summaries: Sketches

Sketches first appear in the context of relational databases as a solution to estimate the size of joins in limited storage. A sketch is capable of summarizing a data stream while still being able to estimate the answer to certain queries about the stream, in our case the 2nd frequency moment. Consider a stream of elements \( S = [e_1, e_2, e_3, \ldots, e_s] \), where every element \( e_i \) belongs to \( I = \{1, \ldots, n\} \). We can define the frequency vector as \( f = [f_1, f_2, \ldots, f_n] \) where every element \( f_i \) is the number of occurrences of the element \( i \); formally, \( f_i = |\{j : e_j = i, \forall e_j \in S\}| \). Then, we can define the 2nd frequency moment as \( F_2 = \sum_i f_i^2 \).

The first sketches proposed are based on the Johnson-Lindenstrauss lemma: a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. Therefore we can project the frequency vector of the stream into a low dimension space and estimate its norm on that space instead of the original, i.e. every different element represents a dimension in the original space. That is the case of AGMS and Fast-AGMS, which use 4-way \( \pm 1 \) random variables for the projections. The second generation of sketches is based on a set of counters: every element updates a counter based on a hash function. In
this case the estimation function must compensate for the overestimation of the value given by the counters. This is the approach that follow the FastCount sketch and the CountMin sketch.

In any case, every sketch can be seen as a matrix of counters, where every row consist on a basic estimator for the 2nd frequency moment and by combining the estimation of every row, the variance of the error is reduced. The sketch is updated for every element of the stream, which results in a nice set of features that make sketches an attractive solution for traffic validation:

- Online updates: To compute a sketch there is no need to keep all the elements of the stream, but it can be created as the 0 sketch, and then updated whenever a new element is available.
- Distributed: Linear combination of streams results in linear combination of sketches, therefore, two nodes can compute each a partial sketch of a stream, and then they can be combined to represent the whole sketch.
- Reduced space: As said before, a sketch represents a summary of the data stream, therefore, nodes in the network can share sketches as summaries of the traffic, without causing much overhead.

Using Sketches for traffic validation instead of just plain counters gives us the advantage that we can detect not only packet dropping, but also packet corruption. And it has been already proven by Goldberg et al. [59] that sketches provide more accurate predictions than sampling.

### 3.1.2.1 AGMS Sketch

The AGMS sketch was proposed by Alon et al. [3] and it is inspired in the tug-of-war game: each stream element is assigned to a random team and the rope displacement is used as estimate. In more detail, every counter represents a tug-of-war game and has a $\pm 1$ 4-wise independent variable ($\xi$). When a new element, $e$, arrives, every counter, $c_i$, is updated:

$$c_i = c_i + \xi_i(e)$$

And the second frequency moment is estimated as:

$$\hat{F}_2 = \frac{1}{n} \sum_{i=1}^{n} x_k[i]^2$$

This results in an unbiased estimator with variance:

$$Var[\hat{F}_2] = 2 \times \frac{(F_2^2 - F_4)}{n}$$

Where $F_4$ is the 4th frequency moment. To reduce this variance, and therefore the error, Rusu and Dobra [128] propose combining several estimations using the median:

$$\hat{F}_2 = \text{median}_{1 \leq k \leq m} \left( \frac{1}{n} \sum_{i=1}^{n} x_k[i]^2 \right)$$

In such case, the quality of the estimation is given by:

$$\text{Prob}(|\hat{F}_2 - F_2| \leq \frac{4}{\sqrt{n}} \times F_2) \geq 1 - 2^{-m/2}$$
3. Self-management

3.1. Routing robustness

3.1.2.1 Mathematical approximation

To provide an analytical estimation of the probability mass function of the error we consider that the stream elements are unique. As we know, traffic packets are unique with high probability and if we use a 2-way hash function with a sufficiently big output space, so will be the sketched elements. Considering that assumption, after sketching \( N \) packets, the probability of obtaining \( k \) ‘+1’ from the \( \chi \) function follows a binomial distribution with 0.5 success probability and \( N \) trials. Therefore, the distribution of the counter value will also follow the shape of a binomial distribution, but its support will be \( 2k - N, k \in [0, N] \). The distribution of the squared value of the counter is:

\[
\text{pmf}_{c^2}(x) = \text{Prob}(c_i = \sqrt{x} + N/2) = \begin{cases} \frac{0.5^N}{2^N} & \text{if } x = 0 \\ \frac{2^N}{2^{(\sqrt{x}+N)/2}} & \text{if } x > 0 \end{cases} \tag{3.1}
\]

Finally, the pmf of the estimation will be the convolution of \( m \) \( \text{pmf}_{c^2} \) distributions:

\[
\text{pmf}_{\text{est}}(e) = \sum_{x \in R_x} \prod_{i=1}^{m} \text{pmf}_{c^2}(x_i) \quad R_x = \left\{ \overrightarrow{x} \mid \sum_{i=1}^{m} x_i = e \right\} \tag{3.2}
\]

The pmf of the error will be the same, but with a different support: \( \text{error} = e/N - N \). The error ranges from \(-N\) to \(N^2 - N\) and the discrete step between possible error values is \(4/m\).

3.1.2.1.2 Experimental results

Basic Estimator

Our first experiments study the performance of a basic AGMS estimator. To do so, for each possible configuration we have run 100 experiments that initialise randomly an AGMS sketch and make 100 predictions by reading \( N \) packets from a CAIDA traffic capture [135] and estimate the number of sketched packets. Because the CAIDA traffic captures do not have complete packets, the missing bits have been randomly generated.

Our first goal was to observe the effect of the bit length of the 2-way hash function’s result. Figure 3.1(a) shows the error distribution for a sketch with 32 columns and 1 row, when sketching 50 packets and using \( \text{eh3} \) as \( \xi \) function. As we can see, if we take 16 or 32 bits of the 2-way hash function, the error is centered in 0 and the distribution is quite close to the one estimated by our mathematical approximation. However, if we only take 8 bits, the distribution is no longer centered in 0 since there are more collusions than expected. In other experiments, for bigger values of \( N \) (\( N=10000 \)) we observe how 16 bits starts being not enough and at least 32 bits are required.

Our second goal was to observe the effect of using different random functions and see which ones provide better results. As we can see in 3.1(b), all the random generators used provide a good estimation despite \( \text{cw2} \). Given this information, we can use the generator with best performance in terms of CPU or memory.

Then, we studied the effect of \( N \) and \( m \) in the accuracy of the sketch. Figure 3.2(a) shows the error bounds for the 99 percentile as the number of packets increases for a sketch of 32 columns by 1 row that uses a 32 bits 2-way hash function and \( \text{eh3} \) as \( \chi \). The figure is in log-log scale, and as we can see our mathematical approximation gives us a closer estimation on the error bounds than the Chebyshev bounds. The relation between the error bound and \( N \) is lineal. On the other hand, in Figure 3.2(b), we see how the error decreases as the number of columns increases for a sketch of the same characteristics but different number of columns when sketching 100 packets. Again, our

Deliverable D.4.4 53
3.1. Routing robustness

Figure 3.1: Error distribution

proposed estimation outperforms the Chebishev bounds and in this case, the error is proportional to \( 1/\sqrt{m} \). The results were similar for different percentile values.

Figure 3.2: 99 percentile of the error

Combining several estimators

Our next experiments measure the effect of averaging the results of several basic AGMS. First we measured the effect of using different mechanisms to average the basic estimations, namely the mean, the median and a trimmed version of the mean. Figure 3.3(a) shows the experimental CDF of the error when using the three different average functions. As we can see the mean provides the best results,
since its deviation is smaller than the trimmean and the median provides a biased estimation. Because
the optimal average function is the mean, that is equivalent of having just a larger basic estimator
rather than a collection of them.
Finally, we gathered many data points to estimate the 99 percentile of the error based on the number of
packets and the size of the sketch and we obtained the following approximation based on the number
of sketched packets \( N \) and size of the sketch \( \text{size} \):

\[ 99\text{percentile} = 3.7 \times \frac{N}{\sqrt{\text{size}}} \]

![Graph comparing average functions](image)

**Figure 3.3:** Error for the AGMS sketch

On Figure 3.3(b), we can see how the error is reduced when the number of rows increases for a sketch
of 32 columns and an estimated value of 50. As we can see the provided estimation comes really close
to the given experimental results \( R^2 = 997 \). Figure 3.4 shows how the \( N/\sqrt{m \times n} \) coefficient
changes for different percentile values.

### 3.1.2.2 Fast-AGMS Sketch

The Fast-AGMS sketch [42] is a refinement of the Count sketch [35] that reduces the number of
counters that need to be updated every time a new stream element arrives. This is achieved by using
a 2-way hash function that selects which counter will be updated:

\[ c_{h(e)} = c_{h(e)} + \xi(e) \]

The second frequency moment can be estimated as:

\[ \widehat{F_2} = \frac{1}{n} \sum_{i=1}^{n} x_k[i]^2 \]
As a result, we have an unbiased estimator with the same variance as the AGMS sketch, but fewer requirements on the computational cost.

### 3.1.2.2.1 Mathematical approximation

Providing a fair estimation for the Fast-AGMS sketch is slightly more complicated because it involves two random processes. First we randomly select which counter to update, and secondly whether to add or subtract 1. The first random process can be approximated with a multinomial variable that determines how many times each counter is selected, then a binomial variable measures the number of ±1 obtained for that counter. In this case, giving a closed mathematical formula is complicated, but we can still compute the results programmatically through the convolution of the multinomial PMF with a binomial with the given number of tries. As before, the error ranges from \(-N\) and \(N^2 - N\), but now the discrete step between possible error values is 2.

### 3.1.2.2.2 Experimental results

We did the same experiments as for the AGMS sketch, which show us similar conclusions: 3.5(a) proves that choosing a hash length of 8 bits is insufficient, but longer hash lengths are closely approximated by the mathematical approximation proposed; and 3.5(b) shows that every generator provides a good estimation except CW2.

On the relation of the error bounds with number of sketches is linear as before, and we can see that our proposed closely estimates the percentiles (see Figure 3.6(a)). The relation between the number of columns and the sketch accuracy is not as clean as for the AGMS sketch, but the tendency is the same: the percentile is proportional to \(1/\sqrt{m}\).

Figures 3.6(a) and 3.6(b) show the relation of the error bounds with the number of packets and columns. In this case, the Fast-AGMS sketch using a 2-way hash function as proposed [42] does not follow the proposed mathematical estimation, which under-estimates the error; however, if a 4-way hash function is used, then the proposed estimation is valid. Figures also show the bounds proposed by Goldberg et al. [60] which are slightly tighter than the Chebyshev bounds.
3. Self-management

3.1. Routing robustness

![Error distribution](image1)

Figure 3.5: Error distribution

![Error 99 percentile](image2)

Figure 3.6: 99 percentile of the error

Figure 3.7(a) compares again using either the median or mean as function to average the results from the different basic estimators. We observe the same results as for the AGMS sketch: using the mean provides a more accurate prediction without bias.

We also estimated the 99 percentile as a function of the number of sketched packets and size of the sketch and obtained the following formula:

\[ 99 \text{ percentile} = 4 \frac{N}{\sqrt{\text{size}}} \]
The precision of this estimation is quite high ($R^2 = 0.996$), as can be seen on Figure 3.7(b), and based on this estimation the FAGMS sketch provides the worst results among the studied sketches, though the difference is not big. Finally, Figure 3.8 shows the coefficient for different percentiles, and the tendency is as mentioned by Goldberg proportional to $\ln(1/\delta)$.

![Graph (a)](image1.png)

(a) Comparing average functions

![Graph (b)](image2.png)

(b) 99 Percentile estimation

**Figure 3.7:** Error for the FAGMS sketch

**Figure 3.8:** Coefficient for different percentiles

### 3.1.2.3 TZ Sketch

Finally, Thorup and Zhang [136] proposed to only increment by one the counter when a new stream element arrives by using a 4-way hash function ($h$) to choose which one:

\[ c_{h(e)} = c_{h(e)} + 1 \]

In this case, if we were to estimate \( F_2 \) as before, we would end up with a biased estimator so we need a more complex estimator:

\[
\hat{F}_2 = \frac{m}{m-1} \sum_{i \in [m]} c_i^2 - \frac{1}{m-1} (\sum_{i \in [m]} c_i)^2
\]

The variance in this case is slightly higher:

\[
\text{Var}[\hat{F}_2] = 2 \times \frac{(F_2^2 - F_4)}{n-1}
\]

But for large values of \( n \) the difference is negligible.

3.1.2.3.1 Mathematical approximation

The second frequency moment estimation can be also expressed as [136]:

\[
\hat{F}_2 = F_2 + \sum_{a \neq b} v_a v_b X_{a,b}
\]

Given that:

\[
X_{a,b} = \begin{cases} 
1 & \text{if } a \text{ b} \\
\frac{1}{m-1} & \text{otherwise}
\end{cases}
\]

Where, \( a \ b \) means that \( h(a) = h(b) \)

Because we assume network packets to be unique and the weight of each element to be either 1 or 0, we can simplify the equation further and express the error of the estimation as:

\[
\text{error} = \sum_{a \neq b} X_{a,b}
\]

For every element present on the data stream, \( a \), considering it colludes with \( C \) other elements, its contribution to the error will be:

\[
\text{error}_a = 1 \cdot C - (N - 1 - C) \cdot \frac{1}{m-1}
\]

Let’s define \( X \) as the multinomial variable that counts the number of times each counter is selected for a given data stream: \( X_i = |a|h(a) = i, a \in [a]| \). In that case, we have \( X_i \) elements that have the error given by \( C = X_i - 1 \) collisions, and so, we can define the error as:

\[
\text{error} = \sum_{i=1}^{m} X_i \times \left( 1 \cdot (X_i - 1) - (N - X_i) \cdot \frac{1}{m-1} \right)
\]

\[
= \frac{m}{m-1} \sum_{i=1}^{m} X_i^2 - N \cdot \left( 1 + \frac{N}{m-1} \right)
\]
Let’s define $\alpha$ as:
\[
\alpha = \sum_{i=1}^{m} X_i^2 = \frac{m - 1}{m} \left[ \text{error} + N \left( 1 + \frac{N}{m - 1} \right) \right]
\]

We can then express the probability of an error as:
\[
P(\text{error} = z) = \sum_{x \in R_x} P(x) = \sum_{x \in R_x} \frac{N!}{x_1!x_2!\cdots x_m!} \frac{1}{m^N}
\]

Where $R_x$ is:
\[
R_x = \left\{ x \in \{0, 1, 2 \cdots N\}^m : \sum_{i=1}^{m} x_i = N, \sum_{i=1}^{m} x_i^2 = \alpha \right\}
\]

As for the AGMS sketch, the error ranges between $-N$ and $N^2 - N$ and the the smallest difference between two possible values of the error is $\frac{m}{m-1}^2$.

### 3.1.2.3.2 Experimental results

#### Basic Estimator

We followed the same mechanisms to study the performance of the TZ sketch as for the AGMS sketch. As before, on Figure 3.9(a) we can see the distribution of the error for a sketch of 32 columns when the estimated value is 50; again, we need sufficient bits so that the estimator is not biased. Regarding the different possible implementations of the hash function (Figure 3.9(b)) both tabulated hashing [136] and CW4 perform as expected, but for CW2 there is a longer right tail.

![Figure 3.9: Error distribution](image)

Regarding the effect of $N$ and $m$, the results are similar to those discussed for the AGMS sketch. Figure 3.10(a) shows the relation between the 99 percentile and the number of packets sketched. The 99 percentile obtained from the experiments is accurately predicted by the mathematical model proposed and the relation between both variables is lineal. Regarding the number of columns, Figure
3.10(b), shows that the mathematical model provides also a good prediction and the error accuracy decreases with $1/\sqrt{m}$ as before, but the relationship is not as strong as for the AGMS sketch.

![Figure 3.10: 99 percentile of the error](image)

**Combining several estimators**

The regression for the TZ sketch is the following:

$$99\text{percentile} = 3.9 \frac{N}{\sqrt{\text{size}}}$$

Compared with the AGMS sketch, we have a slightly higher coefficient, which implies that we will need a bigger sketch to achieve the same precision. Also, because the relation between the size and the percentile was not as strong as for the AGMS sketch, the quality of the estimation is not as good ($R^2 = 0.988$), but is still pretty close (see Figure 3.11(a)).

Finally, Figure 3.11(b) shows the trend of the $N/\sqrt{\text{size}}$ coefficient, which follows a similar trend to the AGMS one, but grows quicker with higher percentiles.

### 3.1.2.4 Sketches Summary

We analyzed the accuracy of three different sketches as traffic summary mechanisms, finding that AGMS sketch is the most accurate among them (sketches of the same size produce more accurate predictions for the second frequency moment). However, given that updating the AGMS sketch is way more expensive than the TZ sketch, in some applications it may be more convenient to use this second sketch, which is almost as accurate as the AGMS sketch, but requires to update less counters for each packet.
3.1. Routing robustness

3.1.3 Traffic Validation

Traffic validation is the mechanism that determines which traffic information is collected and how that information helps us to determine that a network entity is faulty. Typically, traffic validation is built upon the conservation of the flow principle: traffic going into a network area should be (almost) the same to the traffic leaving it; as long as we do not consider the traffic destined to it and coming from it. We can apply traffic validation with different granularity in terms of the traffic being considered (individual flows or the whole traffic) and in terms of the area being monitored (link, node path, AS...). Here, we will explain our traffic validation mechanism in the specific case of monitoring the whole traffic going through a node so that it is easier to understand, but the traffic validation function could be applied to any other scenario where the conservation of the flow principle applies.

Consider the node being monitored, \( M \), and their neighborhood, \( \text{Neigh}(M) \), then by conservation of the flow we would expect in ideal conditions:

\[
\bigcup_{i \in \text{Neigh}(M)} T_{i \rightarrow M} = \bigcup_{i \in \text{Neigh}(M)} T_{i \leftarrow M}
\]

Where \( T_{i \rightarrow M} \) is the traffic from neighbor \( i \) to \( M \) without considering the traffic destined to \( M \) and \( T_{i \leftarrow M} \) is traffic from \( M \) to \( i \) without considering the traffic coming from \( M \), in both cases from \( i \)'s perspective. Of course, since networks suffer of packet losses due of congestion, collision, etc. both sides of the equation will not be exactly the same, but approximately.

As already mentioned before, we now need a traffic summary function, because sharing all the traffic already sent is terribly expensive. Here we will use sketches, which have been presented in the previous section. Because sketches can be linearly combined we have that if every neighbor keeps an sketch of the monitored node’s incoming and outgoing traffic; then all of them can be combined to obtain the sketch that represents the difference between the global incoming and outgoing traffic:
3. Self-management

3.1. Routing robustness

\[ S_{\text{diff}} = \sum_{i \in \text{Neigh}(M)} S(T_{i \rightarrow M}) - \sum_{i \in \text{Neigh}(M)} S(T_{i \leftarrow M}) \]

Where \( S(T) \) is the sketch of the traffic flow \( T \). Then, since sketches provide a good estimation for the second frequency moment of the traffic flow it summarizes, we know that:

\[ ||S_{\text{diff}}||^2 \approx ||T_{\ast \rightarrow M} - T_{\ast \leftarrow M}||^2 \leq ||T_{\ast \rightarrow M} - T_{\ast \leftarrow M}|| \]

Ideally, we would like the inequality to be an equality, which happens when the elements of \( T \) are unique. This is the case for IPv4 packets, because of the identification field, but not in IPv6. In the case there are duplicate elements sketched on \( S_{\text{diff}} \) the second frequency moment will be an overestimate of the difference between incoming and outgoing traffics. We will see in section 3.1.3.2 that this is a rare event, and so sketches can be used effectively for traffic validation.

In any case, there are a couple of considerations in order to use sketches for traffic validation:

- **Input space**: the original space of all possible packets is terribly big (\( I = \{2^{8 \times 1500}\} \) for Ethernet v2 and bigger with Jumbo Frames). As a result, computing the sketch over this space is very expensive in terms of computation, and therefore we require a space reduction function. In this work we use the last bytes of SHA-256 to reduce the space to a reasonably sized one.

- **Changeless elements**: to be able to compare the sketches produced in two different neighbors of \( M \), the elements sketched need to be the same. That implies that layers below the IP layer must be removed and that the TTL (or hop limit in IPv6) must be adapted by either having the sender reduce it by 2 before sketching the packet or simply setting its value to 0. Also, networks that perform packet fragmentation cannot use sketches for fault localization.

In essence, every node will keep a sketch of the traffic of each neighbor. Whenever a node receives a packet, it will remove its lower layers and adapt the TTL field as required. Then it will generate its SHA-256 hash and keep the last significant bytes as the element to sketch. Finally it will update the sketch of the source neighbor if that packet was not originated from it. Something similar will be done every time a message is sent.

To validate the forwarding behavior of node \( M \) we will estimate its loss probability as:

\[
\hat{\text{loss}}(t) = \frac{\text{diff}(t)}{\text{num\_pkts}(t)}
\]

\[
\hat{\text{loss}}_W = \sum_{t=0}^{W-1} \hat{\text{loss}}(t) \cdot \text{num\_pkts}(t) / \sum_{t=0}^{W-1} \text{num\_pkts}(t) \overset{?}{<} \text{threshold}
\]

Where \( \text{num\_pkts} \) is the estimated number of packets for each interval, \( \text{num\_pkts} = \max(||S_{\text{in}}||^2, ||S_{\text{out}}||^2, 1) \). To determine if the node is behaving properly, \( \hat{\text{loss}}_W \) is compared with \( \text{threshold}_2 \), calculated using the bounds of misbehavior of non-faulty nodes (\( \alpha \)) and faulty nodes (\( \beta \)) [59]:

\[
\text{threshold}_2 = \frac{2 \alpha \beta}{\alpha + \beta}
\]

Our proposal for the metric averages the node’s packet loss estimation during \( W \) intervals because traffic in WCN is not stable (some intervals will have a large amount of traffic while others will not). In section 3.1.3.2 we will show the effect of averaging or not the packet loss estimation.
3.1.3.1 Traffic Validation without clock synchronization

The previously proposed traffic validation mechanism measures the difference between the traffic flows captured by the sketches, but if the neighbors do not agree on the packets that are to be captured, the measured difference will be bigger than expected. Consider, for instance, that node M is connected to node A and B, and the traffic perspective from A and B is as shown in figure 3.12(a). In this case, because of the lack of synchronization, $|S_{\text{diff}}|^2 = |\{a_3, b_1, b_4\}| = 3$, however, M is forwarding packets properly.

To overcome this problem we propose to use the sketches sent on the previous and next interval. Consider $S_{\text{in}}(t)$ the sketch as result of summing every sketch related to the incoming traffic at time interval $t$ and $S_{\text{out}}(t)$ the one related to the outgoing traffic. As we have seen, some packets from $S_{\text{in}}(t)$ may not be on $S_{\text{out}}(t)$, but on $S_{\text{out}}(t-1)$ or $S_{\text{out}}(t+1)$. Similarly, some of the packets from $S_{\text{out}}(t)$, from a neighbor’s perspective may have been sent during interval $t-1$ or $t+1$, so we obtain:

\[
\hat{\text{diff}} = |S_{\text{in}}(t) - S_{\text{out}}(t)|^2 - S_{\text{in}}(t) \cdot S_{\text{out}}(t-1) - S_{\text{in}}(t) \cdot S_{\text{out}}(t+1) \\
- S_{\text{out}}(t) \cdot S_{\text{in}}(t-1) - S_{\text{out}}(t) \cdot S_{\text{in}}(t+1)
\]

Where $S_i \cdot S_j$ refers to sketches $i$ and $j$ inner product.

This estimation is reliable as long as $M$ behaves properly, but if it does not, it could replace some packets for others sent previously and avoid detection. Consider for instance the scenario on figure 3.12(b), $\hat{\text{diff}} = |\{a_2, a_3, b_4, b_1\}| - |\{b_2\}| - |\{a_1, a_2, a_3\}| - |\{b_2\}| - |\{\} | = 0$. Therefore, we cannot discount every packet in the previous and next sketches, but only the ones that are not duplicated from the current sketch:

\[
\hat{\text{diff}} = |S_{\text{in}}(t) - S_{\text{out}}(t)|^2 \\
- S_{\text{in}}(t) \cdot S_{\text{out}}(t-1) - S_{\text{in}}(t) \cdot S_{\text{out}}(t+1) \\
- S_{\text{out}}(t) \cdot S_{\text{in}}(t-1) - S_{\text{out}}(t) \cdot S_{\text{in}}(t+1) \\
+ S_{\text{in}}(t) \cdot S_{\text{in}}(t-1) + S_{\text{in}}(t) \cdot S_{\text{in}}(t+1) \\
+ S_{\text{out}}(t) \cdot S_{\text{out}}(t-1) + S_{\text{out}}(t) \cdot S_{\text{out}}(t+1)
\]

3.1.3.2 Evaluation of the traffic validation function

Our evaluation is based on a traffic capture from a wireless node member of the qMp Sants WCN[31]. Its main characteristics are summarized in table 3.1.3.2. At the moment of the capture, the node (M)
3. Self-management

3.1. Routing robustness

was connected to three other nodes, two of them, $A$ and $B$, had really good link quality (loss below $10^{-6}$) and another, node $C$, with bad link quality (50% packet loss measured with MTR). Because qMp Sants is an ad-hoc network, the routing protocols will rarely use the link with low quality (less than 0.1% of the traffic is routed through $C$). The traffic is quite variable in terms of packets per second. The $\alpha$ of this traffic capture is 0.058 (given by the periods with most traffic from $C$); therefore, for a $\beta$ of 0.1, threshold$_2$ is 0.73. On each experiment, we replay the traffic capture, simulating the link properties (link quality and delay) on each link and compute the expected sketches on each of the nodes with the given experiment parameters. $M$ will either behave properly or drop 10% of the packets, and we will study whether it is properly detected or not depending on the scenario.

<table>
<thead>
<tr>
<th>Number of packets</th>
<th>50000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>842.2 seconds</td>
</tr>
<tr>
<td>Average packets per second</td>
<td>59 p/s</td>
</tr>
<tr>
<td>Average bytes per second</td>
<td>45746.6 B/s</td>
</tr>
<tr>
<td>Duplicated packets</td>
<td>60</td>
</tr>
</tbody>
</table>

3.1.3.2.1 Synchronized scenario

First we will consider that neighbors have synchronized clocks, and so the sketches they share are perfectly aligned. In figure 3.13(a) we see the proportion of times faulty and non-faulty behaviors are classified as faulty. For very small intervals, it is difficult to decide whether a node is faulty or not, because there may be no packets or very few to determine the node’s behavior. As the interval becomes larger, detection becomes more accurate. Longer intervals will imply lower overhead and longer detection times. Another option is to average the estimation during $W$ intervals. On this scenario we have found no relevant difference on the results when using sketches of different sizes.

![Diagram](a) With different window lengths ![Diagram](b) Using or not intersection

Figure 3.13: Proportion of detected nodes
3.1.3.2 Not synchronized scenario

Then, to measure the effect of clock synchronization, we have advanced the clock of $A$ ($t = t_0 + \frac{skew}{2} \cdot Interval$) and delayed the clock of $B$ ($t = t_0 - \frac{skew}{2} \cdot Interval$) a proportion of the interval. Figure 3.13(b) shows the difference between the metric with intersection and without when the skew is 0.1 times the interval, sketch size is 64 by 64 and a window size of 10. However, because we are now using the sketches for predicting larger numbers (the difference between 2 sketches is always expected to be 0, whereas the intersection between 2 misaligned sketches is not), the size of the sketch matters on the precision of the prediction (see figure 3.14(a)): it has to be large enough to avoid false positives.

![Figure 3.13(b)](image)

**Figure 3.14:** Effects of the sketch size

3.1.4 Distributed detection

Our focus next is to present a solution, KDett, that provides a secure mechanism to exchange monitoring information between network routers so that faulty routers can be detected. In a network, compromised routers can lie on behalf of each other to avoid detection: e.g. in Figure 3.15, node A may cover for node B, reporting that it has received all the traffic that node B is dropping. The distributed detection algorithm must make sure that this situation is properly detected, usually by monitoring larger network areas.

Focusing on WCN there are some factors that must be considered while designing KDett:

- We cannot rely on a single routing protocol running on the network, neither that it will be a link-state routing protocol.
- We cannot rely on overhearing techniques, because a node may have several antennas.
- The detection protocol should be distributed, so that it scales up to thousands of nodes.

Before formalizing KDett’s goals and detailing the system model, some concepts that will be used to explain it:
A core is a connected set of nodes, that is, there is a path between each pair of nodes in the core that consists only of nodes form the core. A core’s boundary is the set of nodes that connects the core with the rest of the network, i.e., every neighbor of a core’s node that is not already part of the core.

A t-faulty node misbehaves in the traffic forwarding process, e.g., it may drop or corrupt packets instead of forwarding them. A p-faulty node does not participate properly in the detection protocol; for instance, it could provide false traffic summaries or simply not participate in the protocol. A faulty node is either p-faulty, t-faulty or both; and a core or boundary is p-faulty, t-faulty or faulty when at least one of its nodes is p-faulty, t-faulty or faulty, respectively.

3.1.4.1 KDet’s Specification

As a failure detector KDet’s correctness can be expressed in terms of accuracy and completeness. We define accuracy and completeness based on those presented by Mizrak et al. [105]:

- **a-Accuracy**: A failure detector is a-accurate if whenever a correct router detects a core as faulty, then there is at least one router \( r \in core \) that is faulty and \( |core| \leq a \).

- **a-Completeness**: A failure detector is a-complete if, whenever a router \( r \) is t-faulty, then eventually all correct routers will suspect a core such that \( r \in core \) and \( |core| \leq a \).

3.1.4.2 Assumptions

Next we present the required assumptions for KDet to work properly.

3.1.4.2.1 Network model

Our network model does not make any assumption regarding the routing protocol or the overhearing capabilities of nodes to be consistent with a WCN. However, we assume that only bidirectional links are considered and that neighboring nodes agree on the traffic they have exchanged.

We also assume that there are three services available to support KDet operation:

- A mechanism to verify the authenticity of KDet’s messages.
- A reliable mechanism to discover the neighbors of nodes up to \( k \) hops away.
- A set of trusted authorities (TA), responsible of collecting the core evaluations sent by KDet’s boundaries.

3.1.4.2.2 Traffic Validation function

KDet is a distributed detection algorithm independent of the mechanism used to summarize and validate traffic. Still, to achieve detection we need a traffic summary function, and a validation mechanism...
for network areas. We will use \( S(\cdot) \) to represent that summary function and \( S \) to represent the summary itself. If there is more than one summary involved, we will use \( S_{\text{traffic}}^{\text{link}} \) for the summary of some traffic going through a specific link. For example, node \( A \) may keep a summary for all the traffic sent to \( B \), \( S^{A\rightarrow B} \); and another for the received traffic, \( S^{A\leftarrow B} \). Sometimes, several links may exist between \( A \) and \( B \), and then, we may refer to an specific link as \( i \), having, for instance, the summary of the traffic sent with destination \( n \) through that link be \( S_{i\rightarrow}(n) \). Depending on the implementation strategy (subsection 3.1.4.5), it should be possible to add and subtract traffic summaries. Luckily many summary functions have defined \( + \) and \( - \) operations, such as sketches \([128, 59, 156]\), counters \([24]\), fingerprinting \([106]\) or sampling \([59]\); so they can be used for KDet.

For traffic validation we assume that there is a validation function, \( V(S_{\text{in}}, S_{\text{out}}) \), that evaluates a core given the summaries of the traffic entering and leaving that area. \( V \) equals \( \text{true} \) when the behavior of the core is as expected and \( \text{false} \) when it is under-performing.

\( V \) will be typically based on the Conservation of the Flow (CoF) principle: \textit{traffic entering a network area should be the same as traffic leaving, except traffic destined to and generated by it} \([24]\); and as such \( V \) will mainly compare the incoming and outgoing traffic of a given core:

\[
S(\text{traffic}_{\text{in}}) - S(\text{traffic}_{\text{to}}) \approx S(\text{traffic}_{\text{out}}) - S(\text{traffic}_{\text{from}})
\]  

\[ (3.3) \]

### 3.1.4.2.3 Threat model

As explained before, in our threat model we consider t-faulty and p-faulty nodes. We assume the behavior of the t-faulty nodes can always be detected by the traffic validation function; that is, given the real traffic summaries, \( V(\text{faulty,core}) = \text{true} \). No assumptions are made regarding how p-faulty nodes participate in the detection protocol. For instance, they may send false traffic summaries to avoid the detection of a colluding node, or corrupt the reports instead of flooding them.

As Mizrak et al. \([105]\), we measure the strength of the adversary as the largest set of connected faulty nodes, expressed by \( k \). For example, in Figure 3.15, \( k = 3 \).

We assume, that the network is sufficiently connected, so that even if the faulty nodes are removed from the routing fabric, the network is still connected and that terminal routers are not faulty with respect to the traffic they originate or consume.

Because the failure detector is defined in terms of intervals of time, we also assume that the misbehavior of faulty routers lasts long enough to be detected.

### 3.1.4.3 KDet description

KDet implements a detection protocol that monitors a set of nodes from its boundary with the rest of the network. KDet’s main goals are that if the core is faulty, the boundary can detect it, and if the boundary falsely accuses the core, the core can also detect it. This is achieved by designing KDet to satisfy two principles:

**Detection** If the core is t-faulty and there is no p-faulty node in the boundary, the core is detected as faulty.

**No false accusation** If the core is not faulty, then it is never detected as faulty.

The first principle is enforced through the \textit{boundary protocol} and the second principle is attained by detecting false accusation through the \textit{core protocol} and delegating detection to the \textit{coordinated detection phase}. 
3. Self-management

3.1. Routing robustness

3.1.4.3.1 Boundary protocol

The goal of the boundary protocol is to determine the forwarding behavior of the core. This is achieved by, first, monitoring the traffic entering and leaving the core and, later, using the traffic validation function to test the core’s behavior.

More formally, every node in the boundary monitors each link \((i)\) with the core and keeps a summary for the incoming traffic:

\[
S_{\text{in}}^{\rightarrow i}(T) = S(\text{traffic}_{\text{in}}(T) - \text{traffic}_{\text{to}}(T))
\]

and a summary for the outgoing traffic:

\[
S_{\text{out}}^{\leftarrow i}(T) = S(\text{traffic}_{\text{out}}(T) - \text{traffic}_{\text{from}}(T))
\]

Where \(\text{traffic}_{\text{in}}\) is the traffic sent through link \(i\) during period \(T\); \(\text{traffic}_{\text{to}}\) is the part of that traffic destined to nodes in the core. Similarly, \(\text{traffic}_{\text{out}}\) is the received traffic from link \(i\), and \(\text{traffic}_{\text{from}}\) is the part of that traffic whose source is a node in the core.

At the end of the period, each boundary node creates a report with the traffic summaries, signs it, and shares it with the rest of the boundary nodes by robustly flooding [119] it through the core.

After waiting a reasonable amount of time, each node expects to have the report of every other node in the boundary. If any report is missing, or a signature is not valid, the core is suspected. Otherwise, the boundary node evaluates the behavior of the core using the validation function:

\[
V_{\text{core}}(T) = V\left(\sum_{\forall i} S_{\text{in}}^{\rightarrow i}(T), \sum_{\forall i} S_{\text{out}}^{\leftarrow i}(T)\right)
\]

Then, every node emits its verdict by sharing with the corresponding TA a signed core evaluation, which consists of \(V_{\text{core}}(T)\) and a bitmap indicating which reports have been received.

3.1.4.3.2 Core protocol

In parallel, nodes in the core look for p-faulty nodes on the boundary by checking the reports and core evaluation, and disconnect from nodes detected as p-faulty. A correct boundary node behavior is characterized by:

1. There is a single and consistent report for each interval.
2. \(V_{\text{core}}(T)\) is consistent with the exchanged reports.

To assess the behavior of a boundary node \(A\), every node in the core \(B\), connected to it (via link \(i\)) will go through the steps that follow.

First, at the end of the monitoring interval, it expects a report from \(A\). In case there is no report, its signature is not valid, or there is more than one version, \(B\) will consider \(A\) p-faulty.

Then, the node will compare the traffic summaries in the report related to the link \(i\), \(S_{\text{in}}^{\rightarrow i}(T)\) and \(S_{\text{out}}^{\leftarrow i}(T)\), with its own local version, and if they are not the same, \(A\) is considered p-faulty as well.

Finally, \(B\) compares the \(V_{\text{core}}(T)\) announced by \(A\) with the one that results from combining the reports that have been exchanged (available to any node in the core because of robust flooding). Again, if the results do not match \(A\) will be considered p-faulty.

If in any case \(A\) is considered p-faulty, then \(B\) disconnects from it.
3.1.4.3.3 Coordinated detection

At the end of each interval, the TA gathers a core evaluation from every node in the boundary that consists of a bitmap indicating which reports were received and $V_{\text{core}}(T)$. If there is any missing report or $V_{\text{core}}(T) = \text{false}$ the TA will add the core to the list of suspected cores ($\text{core} \in \text{suspected}(T))$. Then, if in the next intervals the core is still suspected and every core node is still connected to the same nodes in the boundary, the core is detected as faulty ($\text{core} \in \text{detected}(T')$). Formally:

$$\text{core} \in \text{suspected}[T_1, T_2] \land \exists T' \in (T_1, T_2) \mid \text{boundary}(\text{core}, T') \subseteq \text{boundary}(\text{core}, T_1) \Rightarrow \text{core} \in \text{detected}(T')$$

Where $\text{core} \in \text{suspected}[T_1, T_2]$ means that the core was suspected on every interval from $T_1$ to $T_2$, and $\text{boundary}(\text{core}, T') \subseteq \text{boundary}(\text{core}, T_1)$ implies that for every link existing between the core and the boundary at time $T'$ the same link also exists at time $T_1$.

3.1.4.4 KDet Validation

We prove KDet’s correctness by proving that no matter what faulty nodes do, its principles are always respected or the faulty nodes become disconnected from the network. Then, we will prove that KDet is k-accurate and k-complete if we choose the proper set of cores to monitor.

We can re-state the first principle as:

$$\text{core} \in \text{faulty} \land \text{boundary} \in \text{correct} \Rightarrow \exists t \mid \text{core} \in \text{detected}(t) \lor \text{core} \in \text{disconnected}(t)$$

Because the core is faulty, if every traffic summary is available, the traffic validation function will detect the core:

$$V_{\text{core}}(T) = V \left( \sum_{\forall i} S_{\text{core}}^{\rightarrow i}(T), \sum_{\forall i} S_{\text{core}}^{\leftarrow i}(T) \right) = \text{false}$$

Because boundary nodes send signed reports, a core node cannot drop some traffic summaries, or the signature will be invalid and the report marked as not received. Therefore, core nodes can either robustly flood the boundary reports as expected and become suspected because $V_{\text{core}}(T) = \text{false}$ or not flood them and become suspected because the core evaluation will mark some reports as not received. In any case, the core cannot avoid being suspected.

Once the core is suspected, its only way of avoiding detection is by modifying its boundary, so that:

$$\text{boundary}(\text{core}, T') \nsubseteq \text{boundary}(\text{core}, T_0)$$

That is, on each interval, it needs to reduce its boundary by, at least, one link, and, because the number of links are finite, the core will eventually disconnected from the network or be detected.

**Lemma 1:** Eventually, every faulty core is either detected or it stops being part of the network.

Similarly, the second principle is equivalent to:

$$\text{core} \notin \text{faulty} \Rightarrow \text{core} \notin \text{detected}$$
3. Self-management

3.1. Routing robustness

A core can only be detected if it is first suspected and, moreover, its boundary does not change; therefore, we only need to prove that whenever the core is suspected, its boundary changes.

A core is suspected if there is a boundary node that claims either that not every report was received or that reports were received but \( V_{\text{core}}(T) = \text{false} \). In the first case, if boundary node \( A \) claims that it has not received the report from boundary node \( B \), since the core is not faulty and we use robust flooding, it can either be because \( B \) has not sent the report or because \( A \) is lying about not receiving the report. In any case, core nodes will know which one of the two nodes is p-faulty and disconnect from it, changing the boundary as expected. In the second case, because the core is not faulty we know that the traffic summaries satisfy:

\[
V_{\text{core}}(T) = \bigvee \left( \sum_{\forall i} S^{\rightarrow i}_{\text{core}}(T), \sum_{\forall i} S^{\leftarrow i}_{\text{core}}(T) \right) = \text{true}
\]

Thus, if boundary node \( B \) announces \( V_{\text{core}}(T) = \text{false} \), either one of the traffic summaries has been modified or \( B \) is lying about \( V_{\text{core}} \). If a node is misreporting the traffic summaries, the node in the core at the other end of the link will notice it and disconnect from it. And if \( B \) is lying about \( V_{\text{core}} \), every neighbor in the core will detect the inconsistency and disconnect from \( B \). Therefore, in any case the boundary changes when the core is suspected and so a correct core is never detected.

**Lemma 2:** A correct core is never detected as faulty.

Let’s now define the set of cores to monitor for a given \( k \) as every set of connected nodes that are smaller than or equal to \( k \):

\[
C = \{ c \mid |c| \leq k \land \text{connected}(c) \}
\]

In this case, every time a core is detected as faulty (\( \text{core} \in \text{detected} \) it has to be a faulty core or it will contradict Lemma 2. And given how \( C \) is defined, the core will always have a size equal or below \( k \).

Then, if the set of faulty nodes is \( F = \{ f_1 \cup f_2 \cup \cdots \cup f_N \} \) such that for every \( f_i \), \( |f_i| \leq k \) and there is no link between \( f_i \) and \( f_j \) if \( i \neq j \) (implicit because \( k \)), then, given the definition of \( C \) we can ensure that there will be a core \( c_i \) for every \( f_i \), such that \( c_i = f_i \) and \( c_i \) will have a correct boundary, because there are no direct connections between faulty sets of nodes. Therefore, applying Lemma 1, every \( f_i \) will be detected and since \( |c_i| = |f_i| \leq k \), KDet is k-complete.

3.1.4.5 KDet Analysis

Previous subsection proves KDet’s accuracy and completeness; here we will study what is the cost to achieve it. KDet’s cost is mainly determined by the memory required to store the state of the protocol and the network overhead required to share traffic summaries.

3.1.4.5.1 State size

The size of the state is determined by the number of summaries a node needs to keep. Given a summary function that supports the + and − operations, a node can follow two strategies to save the required summaries:

1. For every monitored core and every link, \( i \), that connects to that core, keep a summary, \( S_{\text{core}}^{\rightarrow i} \), of the traffic traversing the core.
2. Keep a summary of the incoming and outgoing traffic of every link, \( S_{\text{}}^{\rightarrow i} = S^{\rightarrow i} - S^{\leftarrow i}; \) and additionally, for each link, the traffic related to every node \( k \) hops away, \( S_{\text{from}(n)}^{\rightarrow i} = S_{\text{to}(n)}^{\rightarrow i} - S_{\text{from}(n)}^{\leftarrow i} \)
so that a core’s traffic summary can be computed as:

\[ S^i_{\text{core}} = \sum_{\forall n \in \text{core}} S^i_n \]  

(3.4)

For small values of \( k \), the number of cores to monitor will be small, and therefore, option 1 will be preferred. Conversely, as \( k \) grows bigger, the number of cores grows exponentially, and 2 becomes a better choice. In further detail, if \( N \) is the number of nodes in the network, and \( R \) is the maximum number of links per node, then the number of cores a node monitors is bounded by \( O(R^k) \) and the number of summaries \( O(R^{k+1}) \), since it can have up to \( R \) links to a core. In contrast, the second strategy keeps a summary per link and at most another per node, that is, the number of summaries will be bounded by \( O(\min(R^{k+1}, RN)) \), since for every link there will be at most \( R^k \) or \( N \) nodes at \( k \) hops. Compared with other strategies, Mizrak et al. [105] showed that WATCHERS cost is \( O(R \times N) \) and \( \Pi_{k+2} \), \( O(\min(R^{k+1}, N)) \); therefore, if we choose the optimal strategy based on our network, our solution will have a tendency that is between WATCHERS and \( \Pi_{k+2} \).

### 3.1.4.5.2 Network overhead

In the decision on how to propagate the traffic summaries, a node could take two similar options:

1. Robustly flood within every monitored core their traffic summaries for each link connected to it (\( S^i_{\text{core}} \)).

2. Robustly flood every summary maintained following the second strategy presented before (\( S^i \) for every link \( i \) and \( S^i_n \) for every node \( n \) in its \( k \)-hop neighborhood) with TTL = \( k + 1 \), so that they will reach every node that is also a part of the boundary of a monitored core.

Again, we expect the same behavior: a smaller \( k \) implies less cores to monitor and then strategy 1 causes less network overhead; whereas a larger \( k \) implies an exponential growth in the number of cores and therefore strategy 2 is more convenient. If we assume messages are flooded using broadcast, the first strategy will cause, that every summary stored will be sent by its owner and the \( k \) nodes in the core it relates to (overhead \( \in O(kNR^{k+1}) \)). Instead, the second strategy will broadcast every summary as many times as nodes \( k \) hops away plus one, i.e. overhead \( \in O(\min(N^2R^{2k+1}, N^3R)) \).

Equivalently, for \( \Pi_{k+2} \) every summary is sent through the monitored path, which is at most \( k + 1 \) hops long, so its overhead will be bounded by \( O(\min(kNR^{k+1}, kN^2)) \); and for WATCHERS, every summary is flooded, so it will be sent by every network node, \( O(R \times N^3) \). So as before, the solution proposed, if chosen to optimized the overhead will be within the bounds of \( \Pi_{k+2} \) and WATCHERS.

### 3.1.4.5.3 Example of KDet’s costs

However, the given trends are only an upper bound to the actual space and overhead consumed by the protocols. Figures 3.17(a) and 3.17(b) show which would be the real cost of each detection protocol for the guifi.net Barcelones backbone (Figure 3.16) [62], a network with 113 nodes and 134 links; assuming there is traffic between every pair of nodes.

Figure 3.17(a) shows the state size stored by each node for the guifi.net Barcelones area [62] assuming summaries of 500B [156, 59], the lines show us the average cost for each value of \( k \). As expected, the cost of the first strategy is exponential, and for any \( k \) bigger than 2 the second strategy already outperforms the first one. The second strategy has a state cost that is just slightly worse than \( \Pi_{k+2} \), and in any case always below 1MB, which sounds reasonable. Similar results have been found for different network topologies.

Figure 3.17(b) shows the data (MB) that is shared by each node in the network if we assume that KDet uses broadcast. As we can see, here the first strategy outperforms the second whenever \( k \) is smaller...
than four, because the second strategy floods traffic indiscriminately, whereas KDet-1 only shares them within the required cores. Moreover, the difference between KDet-2 and $\Pi_{k+2}$ is considerable because KDet requires flooding within cores, but $\Pi_{k+2}$ monitors paths, and as such there is no flooding required. To alleviate the effect of such overhead, techniques such as LEDBAT[90] can be used to
3.2 Evaluation of mesh routing protocols for wireless community networks

3.1.5 Conclusions

On this section we have presented and studied the problem of detecting forwarding faults to increase the routing robustness of a WCN. As proposed by Mizrak et al. [106], we have divided the problem into its different subproblems, and analyzed different traffic summary functions, traffic validation mechanisms and proposed a distributed detection protocol. On each case, we considered WCN properties and capabilities, to make sure that the proposed solutions were reasonable in that context. In summary, we have proposed a set of solutions that could be easily implemented for WCN as an independent monitoring daemon, without the need of modifying the current WCN network stack; and which requires little capabilities in terms of CPU and memory consumption, so it can be run in any low to middle range equipment.

3.2 Evaluation of mesh routing protocols for wireless community networks

In this section we focus on one important self-management mechanism, routing, and we study the scalability, performance, and stability of three proactive mesh routing protocols: OLSR[110], BMX6[21] and Babel[77], three common routing protocols in wireless community networks. We study different metrics on an emulation framework and on the W-ILab.T testbed at iMinds under different networks’ sizes and characteristics, making the most of the two worlds. Emulation allows us to have more control over the topology and more systematically repeat the experiments, whereas a testbed provides a realistic wireless medium and more reliable measurements, especially in terms of interference and resource consumption. These routing protocols have been further characterized by studying their control overhead, convergence delay, CPU and memory consumption, and stability. Our results show the relative merits, costs, and limitations of the three protocols.

3.2.1 Mesh Routing Protocols

Routing is a critical function in wireless mesh networks, since it decides the path any packet must follow to reach its destination. In a community network that grows organically, with several hops from the source to the gateway and where network management is not done by a single entity, but by many members of the community in a decentralized way, it is imperative that a routing protocol is able to continuously adapt to network changes. Routing protocols are usually classified as proactive or reactive, based on whether they learn routing paths proactively or just when needed (reactively). In [7], it can be seen that the vast majority of community networks use proactive routing protocols, since nodes do not have energy constraints. Additionally, proactive routing protocols are more efficient in terms of packet delay and outperform reactive protocols when the number of flows in the network increases [37].

One of the goals of our evaluation is to understand the consequences of choosing either a distance-vector or a link-state paradigm and how it affects scalability. Distance-vector routing protocols follow the Bellman-Ford algorithm, sharing only aggregated information about the path metrics, whereas link-state protocols share the whole view of the network, and the metric of every single link is known by every node. On the representation of link-state routing protocols, the choice is easy; OLSR is avoid clogging the network links with the protocol’s traffic.
the most studied and used link-stated routing protocol. On the distance-vector side, we have chosen BMX6 and Babel, which have been extensively compared with other mesh routing protocols in Battlemesh [150] workshops. Babel has been chosen because it is a clear implementation of a distance-vector protocol and BMX6 because of its recent popularity in existing community network projects (e.g., Guifi.net/qMp [32] and Libre-mesh [87]). In addition, BMX6 uses the Secure Hash Algorithm (SHA) hashes instead of IP addresses as node identifiers and implements a number of features to reduce the protocol overhead while keeping the protocol as reactive as possible. The other famous Layer-2 BATMAN [16] has not been included in our evaluation because of the difficulty of comparing it with the Layer-3 protocols.

In the following subsections, we explain how these three routing protocols work by describing the mechanisms used for neighbor discovery (How does a node know other mesh nodes in range?) and topology dissemination (How does a node learn about routes to nodes that are not directly reachable?).

3.2.1.1 Babel

Babel is a proactive, distance-vector routing protocol based on the Bellman-Ford protocol [77]. Its main concern is to limit routing pathologies as routing loops or black holes, which it achieves using a proper feasibility condition and attaching a sequence number to routing updates.

Babel’s feasibility condition determines which of the received routing updates should be considered and which should not; a routing update for a route is feasible only if its metric is smaller than any of the routing updates for the previously advertised route.

The sequence number attached to a routing update is generated by the destination node it announces and determines to which other routing updates the metric can be compared. Only information with the same sequence number is comparable.

**Neighbor discovery.** Babel nodes discover its neighborhood by exchanging two types of messages.

- *Hello* messages are sent to a multicast address by every Babel interface with a sequence number that is increased locally every time a new *Hello* message is sent. By listening to *Hello* messages, a node not only discovers its neighboring nodes, but it also estimates the reception cost (\(rxcost\)) of that link. By default Babel sends a *Hello* message every four seconds.

- *I heard you* (IHU) messages are used to determine the bidirectionality of a link and share the \(rxcost\) with the neighboring node. The IHU messages are conceptually unicast; however, they are sent to a multicast address to avoid address resolution protocol (ARP) exchanges and to aggregate multiple messages in a single packet. They are also sent periodically, but usually not as often as a *Hello* messages; by default they are sent every 12 seconds.

**Topology dissemination.** In Babel, nodes discover far away nodes by sharing their routing table in route update messages.

- A *route update* message announces a route and its associated cost, and every Babel node sends a periodic update for every node it can reach to a multicast address. Additionally, when there is a significant change in the network topology, such as a route retraction or a significant change in the metric, an unscheduled route update is sent, so that periodic updates do not need to happen as often (by default every 16 seconds).

When a node receives an update, first of all it checks its feasibility, and if feasible, it computes the accumulated metric by combining the metric on the update message plus the cost of the link from where the update is received.
3.2. Evaluation of mesh routing protocols for wireless community networks 3. Self-management

3.2.1.2 BMX6

The BMX6 protocol is also a proactive, destination-sequenced distance vector protocol whose main goal is to reduce the size of periodic messages to achieve low routing overhead while attaining high reactivity to network changes. The key concepts behind this are (i) using a stateful-compressed communication between neighbors and (ii) the context-specific propagation of local versus global and static versus dynamic information.

In a mesh network with flat addressing, reducing overhead using stateful communication translates largely to the use of compact (16 bit) local identifiers to refer to other nodes, since addresses (specially with IPv6) are very long. Therefore, every message sent by a node will use its own local identifiers instead of a global one, which has been previously shared.

On the categorization for information, static information refers to such addresses and other details about a node that are unlikely to change; those attributes are gathered together into the node’s description. On the other hand, dynamic information refers mainly to link and path costs estimations. The global versus local separation determines which information is kept within the neighborhood and which is flooded through the network; local identifiers and link costs are kept locally, while path costs and node descriptions are shared globally.

**Neighbor discovery.** Neighbors are discovered in a similar fashion to Babel.

- **Hello** messages are sent to a multicast address periodically with a sequence number that is locally increased every time a new Hello message is generated. By default, Hello messages are sent every 0.5 seconds.

- **Report (RP)** messages are sent with the same interval as Hello messages and report the number of Hello messages received. By counting the number of Hello messages received from a node and knowing the number of Hello messages that a node has received, a node can compute both the transmission and reception costs of a link.

**Topology dissemination.** Routes to other nodes in the network in BMX6 are obtained as a result of the flooding of originator messages.

- **OriGinator Messages (OGM)** are sent periodically by every node (originator) to announce its presence and then re-sent if appropriate by any node that receives it. An OGM contains the sender’s local identifier of the originator, a sequence number, and a metric that measures the cost of reaching it from the sender’s perspective. When a node receives an OGM, it computes the cost of reaching the originator by combining the metric announced in the OGM with the cost of the sender’s link; if this cost is smaller than the cost via any other neighbor, then the node will re-multicast the OGM, after updating it with its local identifier and the metric computed. By default, a node generates an OGM every five seconds.

Additionally, static information is shared on demand. When a node receives an OGM or a Hello message with an unknown local identifier, it will ask the sender for the node description’s hash. This hash allows the node to determine whether the local identifier refers to any of the known nodes, and if it is not the case, then it will request the node’s description and update its knowledge concerning the network.

3.2.1.3 OLSR

In contrast, OLSR, as its name points out, is an optimized link-state routing protocol. The optimized part comes from the optimization on the flooding mechanism; only nodes selected as multi-point relays (MPR) retransmit the node’s messages.
3. Self-management 3.2. Evaluation of mesh routing protocols for wireless community networks

Topology dissemination. As any link-state routing protocol, OLSR provides every node in the network with a (partial) view of the whole topology by flooding the network with Topology Control (TC) messages.

- A TC message describes all the nodes that are reachable from the message creator, as well as the quality of the involved links in both directions. The TC messages are generated periodically by every node in the network and are then retransmitted unchanged throughout the network. By default, the implementation used in our experiments transmits a TC message every five seconds.

Neighbor discovery Neighborhood sensing is performed in OLSR by periodically sending Hello messages.

- A Hello message consists of a locally increased sequence number and the list of known links to the sender’s neighbors as well as their quality and the quality from the neighbor’s perspective. Link quality is computed as a function of the number of received Hello messages from that neighbor, while the quality from the neighbor’s perspective is simply the quality reported in its Hello messages. Hello messages are sent to a multicast address periodically, by default every two seconds.

In this evaluation, we have used the popular implementation by olsrd.org[110], which implements three fundamental changes to the original RFC. First, the MPR optimization itself, originally designed for wired scenarios with loss-free links, is modified to require not only one but seven nodes to reach every two hop neighbor. Otherwise, when considering even the weakest detected link as a reliable resource for the dissemination of topology information, massive network instabilities must be expected. Second, to compensate for the overhead introduced by the increased MPR redundancy, the fish-eye extension has been introduced [2] where TC updates are exchanged less often between far away nodes than between nearby nodes. This is achieved by letting the originator of each TC message use different time to live (TTL) values with the consequence that only every second TC message propagates beyond the first hop. Third, the path metric used by the olsrd.org implementation is based on the expected transmit count (ETX) metric [47] which, compared to the originally proposed hop count metric, provides a better reflection of the real path cost for transmitting a packet via wireless links. Although controversy on the best parameterization of this protocol exists (such as the findings published by Johnson and Hancke in [76] on the performance of ETX and the hysteresis-based hop count metric), we decided to base our experiments on the defaults of this implementation because their current selection still represents a common ground that reflects the experience from its usage in several community networks over many years.

3.2.1.4 Summary

In essence, what differentiates these routing protocols is their topology dissemination mechanisms and how they solve and position themselves in the trade-off between convergence delay and overhead.

- In Babel, nodes only interact with their neighbors, sharing all the relevant information between them periodically. Routing updates are bigger because they contain the complete routing table, but are only shared locally. Overhead is reduced by retaining long intervals between periodic updates, while reactiveness is increased by sending unscheduled updates when the network changes considerably.

- Additionally, BMX6 floods small OGM messages through the network, but principally the information shared is the same as on Babel, except messages are split and triggered differently. Overhead is reduced by compacting periodic messages as much as possible using stateful
3.2. Evaluation of mesh routing protocols for wireless community networks

<table>
<thead>
<tr>
<th>Message</th>
<th>Size (B)</th>
<th>Interval</th>
<th>Messages/node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Babel</td>
<td>Hello</td>
<td>8</td>
<td>4 s</td>
</tr>
<tr>
<td></td>
<td>IHU</td>
<td>16</td>
<td>12 s</td>
</tr>
<tr>
<td></td>
<td>Update</td>
<td>[12, 28]</td>
<td>16 s</td>
</tr>
<tr>
<td>BMX6</td>
<td>Hello</td>
<td>[4, 6]</td>
<td>0.5 s</td>
</tr>
<tr>
<td></td>
<td>RP</td>
<td>1</td>
<td>0.5 s</td>
</tr>
<tr>
<td></td>
<td>OGM</td>
<td>4</td>
<td>5 s</td>
</tr>
<tr>
<td>OLSR</td>
<td>Hello</td>
<td>[28, 32]</td>
<td>2 s</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>$28 + 20 \times n$</td>
<td>5 s</td>
</tr>
</tbody>
</table>

communication between neighbors, whereas convergence delay is minimized by having a very frequent exchange of messages.

- Further, OLSR is a link-state protocol; therefore, during topology dissemination, information concerning every link is shared, instead of path-aggregated information. Overhead is reduced using the fish-eye extension; updates are shared more frequently with nearby nodes than with far away nodes, but to be adequately reactive, the time interval between updates is kept small.

Figure 3.18 summarizes the topology dissemination mechanism for each routing protocol. Table 3.1 lists the messages exchanged by each routing protocol, explaining its size and how many of them are exchanged depending on the number of neighbors ($n$) that a node has. The size given for each message type does not take into account the length of the headers.

3.2.2 Evaluation Metrics

There are several metrics to consider when evaluating the performance and overhead of a wireless mesh routing protocol.

The most common one is to measure its network efficiency, that is: how much routing traffic is necessary to be able to establish a connected network. Network efficiency is usually measured in terms of bytes/second or packets/second.
Given the relatively dynamic properties of wireless community networks we are interested in measuring how fast the routing protocol can adapt to these network changes or how stable the IP network is, given that the physical network is not stable. In terms of stability, we can measure the percentage of time the network is connected by pinging from some nodes to others or, similarly, measuring the longest time the network is not connected. From the perspective of reactivity, we can measure how long it takes to learn a new or better path, which is what we call convergence time. We are also interested in the cost in terms of the resource consumption of processing (CPU) and memory for each routing protocol. We must ensure the network nodes have the capabilities to run such protocols.

We measure the sensitivity of these metrics to the scale of the network, according to the variation of the size of the node neighborhood, variation of the total network size, variation of the length and number of hops of the network paths, and variation of the availability of links or the rate of changes in the network.

These measurements are performed and evaluated in different scenarios using container-based emulation and testbed-based experimentation. Both of them have different degrees of realism and flexibility. Emulation allows total control over environmental conditions, such as topology, availability, and changes but under limited realism. For instance, details taken from diverse real wireless community networks regarding the structure and events during a temporal period can be reproduced in a series of experiments and even be subjected to variations. In contrast, testbed experimentation provides a real environment, under stable environmental conditions, with a given set of nodes, radios, and locations, and control over a few aspects, such as transmit power and choice of nodes to use in an experiment among those available in the testbed.

### 3.2.3 Emulation Experiments

Using emulation, we can easily measure network overhead and convergence time. It is also possible to measure the cost in terms of memory, but the CPU cost is not reliable. Measuring the quality of the path is also complicated because the channels are not perfectly modeled. These metrics, therefore, are better studied in a testbed or real world deployment.

The emulation experiments presented in this section represent a summary of the results obtained during several experiments using the same emulation system. Table 3.2 shows the network characteristics used for each figure. The Barcelonès area corresponds to a large portion of the city and metropolitan area of Barcelona, and its topology was retrieved from Guifi.net Community Network Mark Up Language (CNML). Generator refers to topologies obtained using the generator from [30] with parameters from the Osona county, a representative semi-rural area where Guifi.net started. Each specific experiment with a given set of parameter values was repeated at least 20 times.

Our emulation system is based on mesh Linux containers (MLC) [107], which is a set of scripts based on Linux containers (LXC) and Linux networking tools, such as `ip` or `tc`. The MLC runs an LXC container for each node in the network and establishes the desired connections between them with a given link quality using `tc`, so that packets are randomly dropped and delayed with probabilities as configured. The system does not allow applying complex Wi-Fi models or reflecting the impact of interference between links of any kind.

### 3.2.3.1 Network Overhead

In our first set of experiments, we studied the effect of network size on the network overhead of each routing protocol. The topology emulated in this case corresponds with a representation of the
Table 3.2: Network characteristics of the emulation experiments

<table>
<thead>
<tr>
<th>Topology</th>
<th>Number of nodes</th>
<th>Number of links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figs. 3.20(a), 3.21(a)</td>
<td>Barcelonès 10,20, 30, 40, 50, 60, 66</td>
<td>9,21,31,43,54,65,72</td>
</tr>
<tr>
<td>Figs. 3.20(b), 3.21(b)</td>
<td>Barcelonès 50</td>
<td>54</td>
</tr>
<tr>
<td>Figs. 3.20(c), 3.21(c)</td>
<td>Barcelonès 69</td>
<td>75,100,150,250,500,750,1000</td>
</tr>
<tr>
<td>Fig. 3.22(a)</td>
<td>Barcelonès 66</td>
<td>72</td>
</tr>
<tr>
<td>Figs. 3.22(b), 3.22(c)</td>
<td>Generator 50</td>
<td>75</td>
</tr>
<tr>
<td>Fig. 3.23(a)</td>
<td>Generator 49</td>
<td>74</td>
</tr>
<tr>
<td>Fig. 3.23(b)</td>
<td>Generator 16,25,49,64,81,100</td>
<td>24,38,74,96,122,150</td>
</tr>
</tbody>
</table>

Barcelonès area of Guifi.net (shown in Figure 3.19 and more details in Table 3.2), where each link quality was determined by averaging the measurements of one hour [11].

Figure 3.19: The topology of the Barcelonès network

Figures 3.20(a) and 3.21(a) illustrate the network overhead in bytes and packets of each routing protocol on networks with different numbers of nodes. To obtain those networks, the original network was randomly sampled. As we can see, the number of bytes increases with the number of nodes, and OLSR seems to be the more heavily influenced protocol, Babel always has lower overhead but seems to increase at a faster rate than BMX6. Regarding the number of packets, all routing protocols seem quite stable except Babel, which has a step increase on 50 nodes.

Then for Figures 3.20(b) and 3.21(b), we run the experiment using the Barcelonès network without any modifications, and it shows the overhead of each node depending on the number of neighbors each node has. As we can see, OLSR is more heavily affected by the number of neighbors, whereas Babel and BMX6’s overhead in bytes only increases slightly with the number of neighbors. The number of packets looks stable in every case, except for a peak in Babel when there are five neighbors.

Finally, Figures 3.20(c) and 3.21(c) show the results when there is a fixed number of 69 nodes in the network, but the number of links is variable. As before, OLSR is the protocol that is more
heavily affected by the number of links, whereas Babel remains stable in every case. Further, BMX6 increases slightly as the number of links increases. Because every point represents the overhead for a node, nodes with higher numbers of links within the same network will have higher overhead, and this difference is greater for OLSR than for BMX6 and Babel. The results for the number of packets are similar.

3.2.3.2 Stability and Reactivity

Our second set of experiments attempts to characterize the stability and reactivity of the three routing protocols.

To measure the convergence time, we have looked for the longest path on the Barcelonès network. Then, we have measured how long it takes to discover new nodes that are attached to each of the nodes in the path from one of the endpoints. Figure 3.22(a) depicts our experiments’ results. The OLSR obtains the worst performance, and we can clearly see a step function. This is due to the fish-eye extension [2], which sends link updates more frequently to nearby nodes, and not that frequently to far away nodes. Both BMX6 and Babel have flat responses, with BMX6 outperforming Babel.
3.2. Evaluation of mesh routing protocols for wireless community networks

3.2.3.3 Memory Usage

The final metric studied through emulation is the cost in terms of memory. To measure the memory consumption, we have run the experiments on a computer with an Intel Core2 Duo E8400 processor running at 3.00 GHz with 4 GB of memory. We have retrieved statistics regarding memory usage using the `pmap` utility.

Our first experiment considers a network of 49 nodes and 74 links, and studies the memory consumed by each node based on the number of neighbors it has (Figure 3.23(a)). In this scenario, both OLSR and Babel show constant memory usage, independently of the number of neighbors; however, BMX6 requires more memory when the number of neighbors reaches 15. The results are similar in our next experiment (Figure 3.23(b)), which increases the size of the network (e.g., network with 16 nodes and 24 links, then 25 nodes and 38 links, etc.) and measures the average memory used by each node. Babel memory usage is stable, BMX6 increases with size and for OLSR, we observe an increase in one case for the biggest network. We believe that, for bigger networks, we would also see an increase in the memory use for OLSR and Babel, but this is not seen because memory is assigned in chunks, and, in contrast to BMX6, the OLSR and Babel protocols do not yet need the full memory provided by the current chunk. The assignment of memory in chunks also explains why memory usage of BMX6 seems non-linear. The measured usage remains unchanged until the last allocated memory chunk is exhausted and the probability for allocating the next chunk rises quickly and is non-linear in sections. The measured BMX6 memory usage in Figure 3.23(a) for 10 and 15 neighbors indicates such a section. In Figure 3.23(b), the exponential growth in memory requirements for BMX6 is also caused by an increasing number of links that come with an increasing number of nodes.
3. Self-management 3.2. Evaluation of mesh routing protocols for wireless community networks

3.2.4 Testbed Experiments

3.2.4.1 Experiment Setup

Protocol performance measurements based on real hardware have been performed in the W-ILab.T wireless testbed [23]. The facility consists of approximately 60 stationary and 15 mobile experimentation devices deployed in a 60 x 20 meter indoor location at iMinds. The grid-like deployment structure of stationary devices is principally given by six rows with 10 devices (columns) each and an inter row space of 3.6 meters and inter column space of six meters. The devices consist of off-the-shelf computer hardware, each equipped with two 802.11abgn WLAN cards, which can be freely programmed by the experimenters. Further characteristics and configurations are summarized in Table 3.3. The devices, also called nodes, were configured to run a Linux operating system (OS) based on OpenWRT, a Linux distribution optimized for embedded wireless devices, and several convenient OS and measurement tools to control and collect measurement data. For the experiment, 50 nodes (the upper five node rows) have been used with one radio each configured in IEEE 802.11a ad-hoc mode. All protocols were configured to run on IPv6 and to announce only their primary interface addresses. Given the physically dense node deployment with an average neighbor distance of less than five meters, a number of preliminary measurements were performed to understand the wireless characteristics of the testbed and to avoid a fully connected mesh where the broadcast-based link detection mechanisms of the routing protocols detect links to nearly all other nodes (note that even the two most distant nodes are less than 63 meters away from each other) and achieve the establishment of true multi-hop topologies. The result of this exercise can be seen in Figure 3.24, showing topology snapshots (as detected by the OLSR protocol) resulting from different transmit power configurations with and without background traffic.

The average number of links per node is shown in Figure 8 for the number of nodes, transmit (TX) power, and TX rate grouped by their link quality and whether the links were captured with or without interference caused by TCP background traffic. It can be seen that the presence of TCP user traffic significantly decreases the perception of high-quality links (here those with an ETX rate of less than...
1.3), while the total number of detected links is much less affected. The figure also shows that, in our testbed scenario, the average number of links per node scales linearly with the selected transmit power or rate. Figure 3.26(a) demonstrates the path length established by each routing protocol to route from the leftmost node in the second upper row (node Id 0) to the rightmost node in the same row (node Id 9). As expected, with a low transmit power (or high TX rate) and consequently small transmit range the selected end-to-end path relies on many intermediate hops, while less relaying nodes are required with an increased TX power (or a more robust but lower rate). Figures 3.26(b) and 3.26(c) give an impression on the end-to-end throughput achievable with each routing protocol when varying the node density by increasing power or rate.

Based on these findings, our following experiments will be configured to use the parameters shown in Table 3.4. Primarily, we use 3 dBm of transmit power and disabled transmit rates below 36 Mbps to enforce the establishment of topologies with more than seven hops.

### 3.2.4.2 Measurements

In order to measure the impact of neighbor size (density) and network size on cost and performance, all protocols have been sequentially exposed to a variety of testbed configurations. These testbed configurations were given by the range of parameterization values for each studied parameter and the default value for all other parameters. Each exposure (experiment) consists of the following standard procedure. First, all currently active protocols were disabled on all nodes. Then, the interfaces, IPv6 addresses, wireless settings (channel, mode, TX power, and enabled rates), and the currently probed protocol (only one at a time) were configured and activated only on those nodes relevant for this test. A stabilization period of 100 seconds was applied before continuing the actual measurement to avoid capturing of atypical bootstrapping effects. Each following measurement lasted 60 seconds and relied

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Laboratory 16x60 meter</td>
</tr>
<tr>
<td>Deployment</td>
<td>Regular 5x10 nodes grid (see Fig. 3.24(a))</td>
</tr>
<tr>
<td>Operating system</td>
<td>Linux/OpenWRT BarrierBreaker rev41558</td>
</tr>
<tr>
<td>Protocol impl.</td>
<td>babeld v1.5.0, bmx6 rev8b0585e8, olsrd v0.6.6.2</td>
</tr>
<tr>
<td>Hardware</td>
<td>ZOTAC NM10-ITX</td>
</tr>
<tr>
<td>CPU model</td>
<td>Intel(R) Atom(TM) CPU D525 @ 1.80GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>903460 kB</td>
</tr>
<tr>
<td>Wireless</td>
<td>Atheros - AR928X 802.11a/b/g/n</td>
</tr>
<tr>
<td>Wireless mode</td>
<td>80211a, ad-hoc, channel 36 (5.18GHz)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit power [dBm]</td>
<td>3</td>
<td>3, 4, 5, 6, 7, 8</td>
</tr>
<tr>
<td>Transmit rate [Mbit]</td>
<td>36</td>
<td>6, 9, 12, 18, 24, 36, 48, 54</td>
</tr>
<tr>
<td>Used nodes</td>
<td>50</td>
<td>10, 20, 30, 40, 50</td>
</tr>
<tr>
<td>Network Load</td>
<td>active</td>
<td>none versus active (single TCP stream between most distant nodes of a row)</td>
</tr>
</tbody>
</table>
3.2. Evaluation of mesh routing protocols for wireless community networks

![Graphs showing topologies in different scenarios](image)

**Figure 3.24:** Topologies in different scenarios

(a) Links depending on number of nodes (3dBm, 36Mbit)
(b) Links depending on TX power (36Mbit)
(c) Links depending on TX rate (3dBm)

![Graphs showing network densities and link qualities](image)

**Figure 3.25:** Network densities and link qualities for different testbed configurations

on common Linux tools (such as ping6, iperf, top, and tcpdump) for active probing of end-to-end path characteristics and monitoring and capturing CPU, memory, and traffic overhead.

After each experiment, the measurement data were offloaded and, once all experiments of a particular scenario were executed, post processed into graphs illustrating the dependency of one characteristic depending on a particular testbed parameter. Each measurement point in any of the graphs shown in Figures 3.25 to 3.32 represents the averaged relevant measurement data captured during a single experiment run.

The very limited exclusive testbed usage slots that allow interference-free experiments in the often overbooked W-ILab.T infrastructure do not allow the systematic repetition of scenarios required for a statistical validity analysis. Instead, the goal has been covering a wide range of selected parameters. However, atypical measurements have been selectively repeated to avoid the consideration of exceptional outliers. This way, the resulting and purposely un-smoothed plots also include strongly varying behavior that is a typical characteristic of any real wireless network. Still, a number of protocol-typical characteristics and tendencies can be identified and generally acknowledge the findings made.

Deliverable D.4.4
3.2. Evaluation of mesh routing protocols for wireless community networks

The impact of node density has been studied by either varying the transmit power of each node (with results shown in Figures 3.28(b), 3.29(b), 3.30(b), and 3.31(b)) or by varying the minimal allowed transmit rate per node (see Figures 3.28(c), 3.29(c), 3.30(c), and 3.31(c)). For these scenarios, all 50 nodes were used from the beginning, but transmit power or rate was successively changed in each experiment round.

The repetition of the above scenarios without background user traffic revealed that CPU and memory consumption do not significantly differ in both cases: thus, only the resulting impact on protocol overhead is shown in Figure 3.27.

In order to obtain an experimentation-based picture of the self-healing capabilities of each protocol, we probed the end-to-end path between two nodes. This was realized using a 20-node subset (given by the upper two rows) of the original 50-node topology and an additional “mobile” node installed on a robot that was programmed to move with different speeds just below the second row of nodes in the W-ILab.T deployment from near the leftmost node column to the eighth column and back. Figure 3.24(d) illustrates this setup. The robot turning points were located 42 meters (or seven inter-column spaces) apart from each other. Path health to this moving node was probed using the ping6 command from the fourth node of the second row, located in the middle between the turning points. Figure 3.32(a) shows, based on a TX power and rate setting of 3 dBm and 36 Mbps, that for each protocol the average ping success rate to this moving node depending on its velocity, each reflects a scenario with differently fast changing links. It can be seen how the success rate, around 90% at velocities of 5 cm per second for all protocols, decreases with increasing destination velocities down to around 86 Deliverable D.4.4
60% for BMX6 and below 40% for OLSR and Babel. The repetition of this scenario with slightly increased power settings at 4 and 5 dBm (see Figure 3.32(b) and 3.32(c)) lead to a broader continuous coverage of the moving node, giving routing protocols more time to adapt to weakening links and narrow the performance gap between the three protocols.

![Figure 3.27](image1) **Figure 3.27:** Data overhead (bytes/s) depending on network size and density (power, rate), no TCP user traffic

![Figure 3.28](image2) **Figure 3.28:** Data overhead (bytes/s) depending on network size and density (power, rate)

### 3.2.5 Discussion of Results and Conclusion

In this section, we take a closer look at the measurement results obtained via emulation and experimentation with the objective to gain a general understanding of how the characteristics of a network influence the performance of various routing protocols. We also discuss potential discrepancies of the two different methodologies. Table 3.5 provides a high-level summary of the identified protocol-specific behaviors by grouping the studied performance characteristics (in terms of overhead and self-healing capability) and dependencies (in terms of density, size and topology dynamics) into rows with a few comparative words for each protocol.
3.2. Evaluation of mesh routing protocols for wireless community networks

3.2.5.1 Protocol Data Overhead

Protocol data overhead has been measured in bytes and packets per second and depending on network size, density, and dynamics.

Regarding byte overhead depending on network size, both, emulation- and experimentation-based measurements (Figures 3.27(a) and 3.20(a)) show (apart from one exception in experimental Babel measurements, which we will discuss later) consistent results of an essentially linear increase with a protocol specific slope and base load. In addition, BMX6 shows the highest base load but lowest slope, while Babel shows the lowest base load and a slightly greater slope and OLSR shows the greatest slope which, given the emulation-based results, raises up to 400 bps, about 120% more than Babel and 50% more than BMX6 for a network of 70 nodes. The different absolute numbers between experimentation- and measurement-based results can be explained by the greater average number of links (neighbors) per node in the different scenarios, with three versus up to five (compare Figure 3.25(a)).

A different picture arises when experimentally comparing the byte overhead in the presence of TCP user traffic as shown in Figure 3.28(a). Then, transmissions naturally cause interference and thereby affect the perception of link qualities between neighbors (Figure 3.25(a)) as well as the propagation of routing information. In this scenario, we see that the overhead of Babel increases dramatically because
3. Self-management 3.2. Evaluation of mesh routing protocols for wireless community networks

![Graph](image)

(a) Impact of Network size  (b) Impact of TX power  (c) Impact of TX rate

**Figure 3.31:** Memory consumption depending on network size and density (power, rate)

![Graph](image)

(a) Poor link redundancy (TX power 3dBm)  (b) Medium link redundancy (TX power 4dBm)  (c) High link redundancy (TX power 5dBm)

**Figure 3.32:** End-to-end delivery success depending on topology dynamics and link redundancy

Babel reacts to topology changes by sending unscheduled route updates. On the other hand, BMX6 and OLSR propagate routing updates periodically, independently of topology changes; therefore, this results in little effect on the overhead.

The slight increase in BMX6 can be explained by the requirement of acknowledgements when exchanging routing updates, which will cause overhead due to the retransmissions caused by traffic collisions. The opposite is the case for OLSR where the collision of link-information (TC messages) containing packets, if not successfully received via alternative links, are not further propagated and eventually result in an decreased overall overhead, an effect particularly likely in sparse networks with low link redundancy.

Another critical factor affecting protocol overhead is given by the network density where emulation- and experimentation-based measurements show quite different results. Looking at the former (Figure 3.20(c)), the observed shape of all protocols well matches with what one could expect from each protocol-dissemination algorithms. The highest, quite linear, slope for OLSR represents that of a non-optimized link-state protocol where information about all links in the network are propagated to all nodes for calculating a local view of the total topology. For this purpose, every node contributes to the propagation of this information by re-broadcasting new link state information once (via TC messages).
and thus causing respectively increasing transmission overhead by each node. This non-optimized link-state behavior could be explained by the non-standard behavior of the OLSR implementation using a default MPR selector set of seven (instead of one, see also Section 3.2.1.3). However, compared to the experimentation-based results in Figure 3.28(b), in the beginning the greatly increasing OLSR overhead quickly flattens for densities of around 10 or more links per node (corresponding to a TX power of 5dBm according Figure 3.25(b)), an effect which indicates that the OLSR-implementation specific MPR selector set value of seven still yields significant optimization in very dense networks.

The moderate overhead slope for BMX6 and the constant seeming slope for Babel from the emulation-based results correspond with the typical characteristics of distance-vector routing protocols, where link-quality information is only exchanged between neighboring nodes and thus affects each nodes’ overhead only by a few additional link-probing related messages and as far as the size of its local neighborhood increases but not beyond. Here the aggregation of many messages into much less eventually broadcasted packets help to even flatten the observable overhead on layer 2.

Interestingly, a significantly different picture could be observed for the Babel overhead when considering the measurement-based experiments where the amount of transmitted data increases by a factor of 10 when the node density (at a power level of 6 dBm) exceeds 15 links per node. We attribute this behavior to the high susceptibility of the Babel protocol to topology dynamics, which was already observed for the impact of interfering TCP traffic in Figure 3.28(a). In this case, such topology dynamics are caused by natural interference and consequent collisions due to the dense wireless deployments, a factor not existing in the emulation-based analysis. In fact, protocol performance instabilities, which are likely related to similar topology dynamics, were measured repeatedly in different scenarios. The exceptional experimental Babel measurements points mentioned earlier for byte overhead depending on network size are one example.

The measurement results for network overhead in terms of packets in Figures 3.21 and 3.29 illustrate on the one hand the dominance of protocol-specific link-probing and update intervals causing a constant and minimal packet transmission rate even in the simplest possible deployment. On the other hand, once protocol-stress factors (such as network size, density, or dynamics) cause a protocol to disseminate more data than could be aggregated into the packets sent at a minimal transmission rate, they show how packet overhead first increases in steps before scaling linearly with the byte overhead discussed earlier. In this sense, the high base rate of BMX6, at two packets/second compared to 0.5 packets/second for OLSR and even less for Babel, should only be considered relevant for deployments of rather stable and sparse networks. For more complex deployments the initial lowest packet rate of Babel can easily turn into a rate several times higher rate than that of OLSR and BMX6.

3.2.5.2 CPU and Memory Consumption

Experimentation-based measurements show (see Figures 3.30 and 3.31) that network size and density only have a very limited and generally non-critical impact on the CPU and memory consumption caused by the respective routing-protocol process. Given the embedded hardware and studied parameter space of up to 50 nodes and densities above 20 links per node, protocol-specific CPU usage always remained below 2% of the total CPU processing capacity, and virtual memory consumption (including all shared library objects mapped into the process) remained significantly below 1.5 MByte while showing only a very low increase over the studied ranges. Given these experimentation-based measurements, the memory consumption of Babel is the lowest and least increasing, while OLSR and BMX6 both show a very similar low linear increase depending on network size. Regarding density, BMX6 demonstrates a similar low slope as Babel, which matches what can be expected from any
distance-vector protocol that only has to maintain the next hop towards any distant node. However, the link-state based OLSR protocol, which must keep track of all relevant links in the overall network topology, shows only a slightly greater increase of memory usage depending on density.

In contrast, the emulation-based measurements in Figure 3.23 depict the writable memory requirements (instead of virtual) of OLSR and Babel as completely unaffected by network size and density and below those of BMX6, which also shows a more spread requirement of memory for networks with more than 50 nodes or densities with more than 14 links per node. Nonetheless, the total memory requirements of all protocols remain non-critically low, given memory provisioning, even of resource-constrained embedded devices.

### 3.2.5.3 Self-healing Performance

A number of cases have been studied based on emulation or experimentation to characterize the capabilities of the different protocols to react to topology changes in different scenarios.

Emulation-based results studying the average time needed by each protocol to fix an end-to-end path depending on its length (Figure 3.22(a)) show that the two distance-vector based protocols outperform the link-state based OLSR, particularly in end-to-end scenarios with many intermediate hops.

While the poor performance of OLSR could be explained by the implementation of fish-eye optimisation, the better convergence time of BMX6 compared to Babel is surprising given the reactive nature of the Babel algorithm that should encounter spontaneously detected topology changes on demand instead of delaying the propagation of corresponding routing updates for the next update period. However, the better performance of BMX6 regarding topology dynamics is consistently confirmed in all further measurements (emulation- or experimentation-based, such as shown in Figures 3.22(b), 3.22(c), and 3.32) and must be attributed to the prevailing of other protocol characteristics. One reason is certainly given by the different link-probing and route update intervals (see Table 3.1) used by each protocol that allow BMX6 (using a Hello interval of only 0.5 seconds) to detect and react to local topology changes much faster than Babel and OLSR. On the other hand, to enhance the self-healing performance of a protocol, such intervals cannot be decreased without introducing additional protocol overhead, which is already significantly higher for the latter two protocols in large and dense networks and which, if further increased, would also lead to further protocol instabilities due to self-caused collisions and interference.

### 3.2.5.4 Conclusion

This section presents an evaluation of mesh routing protocols for wireless community networks. We study the scalability, performance, and stability of BMX6, OLSR and Babel, three proactive routing protocols commonly used in these networks, through emulation and experimentation.

Our emulation and testbed-based experiments with various network conditions at different scales have provided several detailed results that compare the three protocols. In summary, we can say that Babel is the most lightweight protocol with the least memory, CPU, and control-traffic requirements as long as it is used in networks with stable links and low node densities.

However, if the protocol is used in large or dense wireless deployments with frequent link changes due to dynamic interference or nodes leaving or joining the network, then its reactive mechanisms to encounter topology changes by sending additional routing updates and route request messages turn into massive control-traffic and processing overhead. In such scenarios, OLSR and BMX6, with their
### Table 3.5: Summary of observed and interpreted performance characteristics from Section 3.2.5

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>OLSR</th>
<th>BMX6</th>
<th>Babel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase of protocol overhead depending on size, density, and dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size:</td>
<td>high linear</td>
<td>low linear</td>
<td>moderate linear</td>
</tr>
<tr>
<td>Density (low density):</td>
<td>high linear</td>
<td>low linear</td>
<td>lowest</td>
</tr>
<tr>
<td>Density (high wireless density):</td>
<td>logarithmic</td>
<td>low linear</td>
<td>in non-linear steps</td>
</tr>
<tr>
<td>Topology dynamics due to interference from TCP user traffic:</td>
<td>negative</td>
<td>unimpaired</td>
<td>highly susceptible with typical strong growth</td>
</tr>
<tr>
<td>Increase of memory usage depending on size and density</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size:</td>
<td>low linear</td>
<td>low linear</td>
<td>lowest, unaffected</td>
</tr>
<tr>
<td>Density:</td>
<td>linear, acceptable</td>
<td>low linear</td>
<td>low, unaffected</td>
</tr>
<tr>
<td>Comment: Non-critical given the studied range of size, density, and dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increase of CPU usage depending on size and density</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size:</td>
<td>low, total max &lt; 0.5%</td>
<td>low linear, total max &lt; 0.2%</td>
<td>low linear, total max &lt; 0.2%</td>
</tr>
<tr>
<td>Density:</td>
<td>varying, total max &lt; 1.5%</td>
<td>varying, total max &lt; 1%</td>
<td>varying, total max &lt; 2%</td>
</tr>
<tr>
<td>Comment: Non-critical given the studied range of size, density, and dynamics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End-to-end path-healing performance due to topology changes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off time versus path length:</td>
<td>high, increasing slope with jump between 7 and 9 hops, ( \text{avg} \sim 35s )</td>
<td>low, unaffected, ( \text{avg} \sim 8s )</td>
<td>medium, unaffected, ( \text{avg} \sim 16s )</td>
</tr>
<tr>
<td>Outage versus changing rate:</td>
<td>BMX6 shows least outage in highly changing environments</td>
<td>All protocols equally good at low changing rates</td>
<td></td>
</tr>
</tbody>
</table>

strictly constant rate for sending topology and routing update messages, outperform Babel in terms of overhead, stability, and even self-healing capabilities.

The OLSR protocol significantly benefits from the MPR mechanism that (despite the highly redundant parametrization used within our experiments) achieves only a logarithmically increasing overhead depending on network density.

The BMX6 protocol benefits from its generally low control overhead due to the usage of compact local identifiers and the hiding of local state (e.g. link qualities) from globally propagated information. It differentiates from OLSR with higher memory requirements but lower control overhead and a better reaction on dynamic link changes.

### 3.3 Experimental Evaluation of BMX6 Routing Metrics

In this section we experimentally evaluate a mesh routing protocol called BMX6 [21, 108]. This is the routing protocol used in the QMPSU (QMP Sants-UPC) production community network described in CONFINE D.4.3 [39, Chapter 1]. The results presented in this chapter are included in the paper [33].
We exploit the full access to the wireless routers of QMPSU to achieve two goals: The first is comparing the experimental measures of throughput on multi-hop paths that we perform on the network with the expected capacity estimation on the same paths derived using the well known conflict-graph model introduced in [74]. Our experiments show that even with an accurate knowledge of the network parameters the conflict-graph model introduces an overestimation of the available capacity. We discuss the possible causes for this error and propose a correction. The second goal is to test the capability of the BMX6 routing protocol used in the WCN to choose the path that can guarantee the highest throughput. We show that BMX6 is able to choose the best path in the large majority of the cases, which is a key feature for any routing protocol, enabled by the right combination of the protocol internals and the metric used for link and path quality estimation.

3.3.1 Related Work

The experimental evaluation of production-state wireless mesh network has been done only in a small number of papers in literature compared to the enormous amount of works that use simulations, a review of the experimental research papers can be found in [144]. Some of the works use a similar approach to this paper for the extraction of real measurement data [28, 18], but most of the networks analyzed are single-channel networks using omnidirectional antennas. In our case the use of multi-channel and directional antennas makes the analysis more challenging, since we neither assume interference between every couple of neighbor links (like in [18]) nor its absence and thus have to rely on a complex model for capacity estimation [74]. Moreover, to our best knowledge, this is the first empirical evaluation of a real IEEE 802.11n-based community network. Other empirical works use controlled scenarios [69] to compare routing metrics (like ETX [47] and ETT [53] metrics). Finally, some other works use off-line evaluation of available data to estimate various network properties [143], including routing performance [95] but can not really be compared to on-field experimentation.

3.3.2 Theoretical Path Capacity

In order to evaluate the performance of the routing protocols we need to estimate the capacity of selected paths. Accurate capacity estimation in wireless is challenging, and the Protocol Model proposed in [64] is typically used in 802.11 networks. With this model any couple of nodes using the same channel and in interference range can not simultaneously transmit. The protocol Model was used to define the concept of conflict graph in [74] to estimate the capacity of wireless networks as an LP optimization problem. Afterwards, the conflict graph has been extensively used in the literature to estimate the capacity of wireless networks in resource optimization problems, e.g channel allocation [83, 49, 153]. In the following we will recall the concept of conflict graphs and will use it not to formulate an optimization problem but instead to estimate the capacity of a multi-hop path once the capacity of the single hops has been measured.

Let $G(V, E)$ be a graph in which the set of vertices $V$ corresponds to the set of nodes in the network and the set of edges $E$ corresponds to the set of links. Let $N = |V|$ the number of nodes of the network denoted by $n_i$, $1 \leq i \leq N$. Take a generic path $P = \{n_1, \ldots, n_d\}$ as the ordered set of nodes chosen by the routing protocol to deliver a packet from the source node $n_1$ to the destination node $n_d$. Let $l_i$ be the link used to connect each node $n_i$ to the next node $n_{i+1}$ in $P$, $c_i$ the capacity (in bit per second) of link $l_i$ and $L = \{l_1, \ldots, l_{d-1}\}$ the set of links used in $P$.

Let $G_c(E, C)$ be the conflict graph of $G$. In $G_c$ vertices correspond to links in $G$, between two vertices there is an edge if the two links interfere and thus can not transmit simultaneously. Let $G_c(P)$ be the
induced sub-graph of $G_c$, where the vertices are the links $L$ in $P$, and the edges are the same as those that links $L$ have in $G_c$.

Now, let $N_i(P), \ i = 1, \cdots d-1$ be the sets formed by each vertex of $G_c(P)$ and its neighbors. Consider two links $l_i, l_j \in N_i(P)$ that require a time $\frac{1}{c_i}$ and $\frac{1}{c_j}$ respectively to send one bit on the link. Note that each set $N_i(P)$ is formed by links that need to schedule their transmissions in different time intervals, so if a bit has to travel over link $c_i$ and then $c_j$ it will require a total time of $\tau = \frac{1}{c_i} + \frac{1}{c_j}$.

The capacity of the path $l_i, l_j$ is thus given by $\tau$. Generalizing, for each sub-path formed by links belonging to $N_i(P)$ the expected capacity is:

$$C_i(P) = \frac{1}{\sum_{l_j \in N_i(P)} \frac{1}{c_j}}, \ i = 1, \cdots d - 1.$$ (3.5)

The theoretical capacity of the path, $C_t(P)$, is given by the most restrictive sub-path, thus:

$$C_t(P) = \frac{1}{t_b(P)}$$ (3.6)

where

$$t_b(P) = \max_i \sum_{l_j \in N_i(P)} \frac{1}{c_j}, \ i = 1, \cdots d - 1.$$ (3.7)

Note that $N_i=b(P)$ is the set of links of the path $P$ that minimize (3.5). Thus, we shall call $t_b(P)$ the bottleneck airtime of the path.

### 3.3.2.1 Validation

In order to validate equation (3.6) we have proceeded as follows: We have experimentally estimated the capacity $c_i$ of each link by measuring the throughput with netperf. The same has been done to estimate the capacity of the path to the gateway for each node. We shall refer to these measurements as the experimental capacities, and denote them as $C_e(P)$.

In order to compute the conflict graph $G_c(P)$ we proceeded as follows. First we defined the graph $G$ and we assigned to each link a value for the capacity $c_i$ that equals the measured one. Then we generated $G_c$, using as vertices the node links, and adding edges between neighbor links using the same channel. Thus, we assume that interference only occurs between WiFi neighbor interfaces using the same channel. For each node we computed the path used to reach the gateway by means of the routing tables and, on that path, we computed the theoretical capacity using (3.6). We shall refer as theoretical computed capacity, $C_t(P)$, to the capacity obtained by (3.6).

Figure 3.33 (top) shows the mean experimental ($C_e(P)$), and theoretical ($C_t(P)$) capacities to the gateway of each node. These are measured only for the most frequent route of each node. The means were obtained averaging more than 100 points in all cases. The resulting confidence intervals were rather small, less than 5% in most cases. In the same figure is shown a third curve ($C_f(P)$), which is a better estimation than $C_t(P)$ and will be explained in next section. Figure 3.33, middle, shows the relative error of $C_t(P)$ and $C_f(P)$ capacities with respect to the experimental ones, $C_e(P)$, computed as:

$$e_i(P) = \frac{C_t(P) - C_e(P)}{C_e(P)}, \ i = \{t, f\}.$$ (3.8)

Finally, Figure 3.33 bottom shows the number of hops of each route. Note that paths are sorted in
increasing order of hops, and capacity.

Figure 3.33 shows that the theoretical capacity overestimates significantly the experimental one. Indeed, the absolute relative error has an average around 34%. This result concerns the usage of the conflict graph as an accurate tool to estimate the capacity of a wireless network. In the following section we discuss this mismatch and propose a better fit of equation (3.6) to the experimental path capacity.

### 3.3.2.2 Path Capacity correction

Given the results of 3.3.2.1, we can say that the definition we use for the conflict graphs leads to an overestimation of the available capacity. To build the correct conflict graph we need to know all the links that interfere with each other, and can not transmit simultaneously. In $G_c$ we set an edge between two links only when the two links are in the same channel, and are separated by no more than one hop, we say that this approach describes only “direct interference”. This assumption is reasonable considering that the majority of the radios use directive antennas, but is probably optimistic, since there are a number of factor that produce what we call “collateral interference”. First we do not consider interference at a higher distance than one-hop, which instead can happen. The number of hops between two nodes depends on the way the radio are configured, and on the decision that the routing protocol takes. Two nodes can be close to each other, but configured with an incompatible MAC layer mode (for instance, both configured to be client of a third node) that prevents them to be direct neighbors. Second, neighbor-channel interference can happen when two radios are placed nearby [157] and even when directive antennas are used [85]. We can not capture this phenomenon with our abstraction so it is reasonable that this contributes to the overestimation of the available capacity.

Since it’s impossible to perfectly model a network operating in real conditions with an analytic approach we chose to apply an empirical approach using the experimental data we have.
3.3. Experimental Evaluation of BMX6 Routing Metrics

Thus, we propose to modify equation (3.6) to estimate the collateral interference in the QMPSU network, as:

\[ C_c(P) \approx C_f(P) = \frac{1}{t_b(P) + f(P)} \]  

(3.9)

We shall call **airtime bloat** the term \( f(P) \), which represents the increment on the bottleneck airtime induced by the interference that we cannot precisely model over path \( P \).

\[ f(P) = \theta \sum_{l_j \notin N_{obs}(P)} \frac{1}{C_j}, \quad 0 \leq \theta \leq 1 \]  

(3.10)

In order to estimate \( \theta \) we used the available experimental data to compute the mean square relative error of the mean capacities, i.e. by minimizing the cost function\(^1\):

\[ J(\theta) = \sum_P \left( \frac{C_f(P) - C_c(P)}{C_c(P)} \right)^2 \]  

(3.11)

From which we obtained that the most suitable value to approximate our data set is given by \( \theta \approx 0.5 \).

Figure 3.33, top, compares the experimental (exp) and computed capacities using equation (3.9) (corr). Figure 3.33, middle, reports the relative error. It can be observed that the capacity estimation is significantly improved. In fact, the absolute relative error has an average around 12\%, which is almost 3 times smaller than the 34\% error obtained with equation (3.6). Note that the value of \( \theta \) is a characteristic of the QMPSU network, so it cannot be simply re-used in other networks. Nevertheless, giving a reasonable good estimate for QMPSU, as discussed above, equation (3.9) will be used as reference to investigate the performance of BMX6 carried out in next section.

\(^1\)We have used the BFGS algorithm provided by the numerical tool R.
3.3.3 BMX6 Performance

In this section we compare the paths chosen by BMX6 with the best paths (having the highest capacity). For the sake of comparison we also use the paths obtained using the Shortest Path First (SPF) algorithm. Note that SPF correspond to hopcount metric. All capacities shown in this section are computed using the corrected equation (3.9). The best path to the gateway has been computed using an algorithm not provided here, for the sake of space. Basically, the algorithm performs a recursive search estimating the capacity of each path. In order to avoid a costly exhaustive search, it is first guessed a best path using a weighted SPF, with link airtimes as costs. Then, recursion is performed, stopping over paths that give worst bandwidth than the current best path estimate.

Figure 3.34 compares the capacity of the path chosen by BMX6 (bmx6); the best path (best); and paths yielding the maximum and minimum capacities using SPF (spf.max, and spf.min, respectively). Note that the points corresponding to bmx6 are the same than those marked as corr in Figure 3.33. For the same number of hops, there might be different paths, having different capacities. As in the previous section, these capacities are computed averaging over the most frequent paths chosen by BMX6, and the best and SPF paths obtained in the same captures. Figure 3.34, middle, reports the relative error of best and SPF paths with respect to BMX6 (see equation (3.8)). Thus, positive error means better paths than BMX6, and negative error means worst. Finally, Figure 3.34, bottom, shows the number of hops to the gateway for the paths chosen by BMX6, best and SPF.

Figure 3.34 shows that BMX6 Vector Metric behaves indeed very well: In most cases the best paths only give a slightly better capacity than BMX6. Only in 2 cases there exists a significantly better path (with relative increases of 400% and 40%, respectively), but having a larger number of hops. Regarding SPF, it was obtained that for the best choice (spf.max), only in 2 points SPF was slightly better, but less than 10%. While spf.min was always worse or equal than BMX6. Indeed, spf.min yielded 6 points (26% of the paths having more than 1 hop) with a relative reduction higher that 40% than BMX6.

3.3.4 Conclusions

In this section we used experimental evidence to analyze the performance of the BMX6 routing protocol. In particular we focused on the capacity of BMX6 Vector Metric to select the route that can achieve the highest throughput and we verified that the combination of metric and protocol internals used by BMX6 is very efficient in selecting a path that is very close to the optimal one. To achieve this goal we performed experiments on the QMPSU network that showed that the model proposed in [74] with simple assumptions on the interference among links produces an overestimation of the achievable throughput.

3.4 Multicast Performance Comparison of SMF and ODMRP

In this section, we present preliminary results from a performance comparison between the two multicast routing protocols: ODMRP and SMF. While SMF uses a proactive selection of forwarding nodes, ODMRP is a reactive approach to multicast routing. For the evaluation, we use a two-step approach. First, we perform emulation-based measurements in a virtualized testbed utilizing the ns-3 IEEE 802.11 model. Then, we do real-world experiments with an actual IEEE 802.11 ad-hoc testbed in order to compare the results and increase the overall credibility of our analysis. The comparison highlights differences in the performance of ODMRP and SMF, both in the real-world measurements
3.4. Multicast Performance Comparison of SMF and ODMRP

3.4.1 Introduction

In scenarios which require spontaneous on-site deployment of communication means (e.g., disaster relief), but also in community networks, wireless multi-hop networks (WMNs) have proven to be a valuable approach. These networks facilitate the idea of infrastructureless communication using technologies such as the IEEE 802.11 ad-hoc mode. In combination with dynamic routing protocols, they enable multi-hop forwarding between a number of participating (potentially mobile) devices. Several protocols have been proposed which focus on the problem of unicast-routing (i.e., routing data between distinct endpoints) and various (mostly simulative) studies exist on their performance. For multicast routing, on the other hand, only few protocols have been proposed and there exist only few examples of real-world implementations and measurements.

It is our goal to gain more insight into the behavior of multicast routing protocols in static WMNs. We compare the performance of the Simplified Multicast Protocol (SMF) and the On Demand Multicast Routing Protocol (ODMRP). For this comparison, we use a novel approach combining emulation and measurements that has been developed within the CONFINE project.

The approach comprises a two-step workflow. In the first step, researchers can design and emulate their experiments in a virtualized environment – the Virtual CONFINE Testbed (VCT) – without affecting the community network. Afterwards, they can deploy their experiment in the physical testbed to obtain results from real-world measurements.

We extended VCT with modules for realistic wireless channel emulation. While the emulation allows for reproducibility, flexibility, and scalability, the physical testbed enables the comparison and validation of obtained results. In addition, certain metrics are easier to measure in the emulation, e.g., due to globally synchronized clocks. Thus, we believe that network emulation is a valuable addition to the workflow.

3.4.1.1 The Simplified Multicast Forwarding Protocol (SMF)

Simplified Multicast Forwarding (SMF) [96] is a multicast routing protocol primarily designed for wireless mesh and mobile ad-hoc networks. The protocol is based on broadcast-based forwarding of multicast packets by flooding (i.e., relaying with duplicate detection). The simplest approach is classical (or simple) flooding, where every node receiving a packet broadcasts it to its neighbors. In order to reduce the forwarding overhead caused by redundant transmissions, the authors suggest to reduce the set of forwarding nodes, e.g., by means of a multi-point relay (MPR) algorithm. This requires knowledge about the neighborhood, which can be gained using the neighborhood discovery protocol (NHDP). For our evaluation we use an implementation developed by the U.S. Naval Research Lab (NRL), which conforms to the standard specification. To the best of our knowledge, this is the only implementation of SMF, which is publicly available. In the initial evaluation presented in this section, we focus on the classical flooding mode without MPR-reduction, because a working implementation was unavailable. We are in contact with the authors and plan to perform enhanced measurements in the future.
3.4.1.2 The On-Demand Multicast Routing Protocol (ODMRP)

The On-Demand Multicast Routing Protocol (ODMRP) [155] is, like SMF, based on the idea of optimized flooding. However, it takes a reactive approach towards the selection of forwarding nodes. It uses the concept of forwarding groups, which are a subset of nodes selected to participate in the flooding for a particular multicast group. These are determined using a two-step discovery mechanism.

In the first step, a multicast sender originates a Join Query message that is flooded throughout the network. As duplicate transmissions are omitted, this process induces a spanning tree over the connected part of the network. The tree is recorded by storing the reverse route to the originator at every forwarding node.

In the second step, every multicast receiver responds to this query by generating a Join Reply message. This message is forwarded along the reverse route, until it reaches the multicast sender. Every intermediate hop on the reverse route joins the forwarding group for the requested address. In order to maintain a forwarding group, the membership is associated with a timeout and the discovery process has to be repeated regularly.

Since there does not exist a reference implementation of ODMRP, we use our own implementation (which is not publicly available at the moment). This implementation realizes a variant of the original protocol, including extensions (such as link-quality based routing) which we presented in our previous work [80].

The remainder of this section is structured as follows: In subsection 3.4.2, we present an overview of the related publications in the field of multicast performance evaluation focusing on ODMRP and SMF. In subsection 3.4.3, we describe the measurement approach used within the CONFINE project. Subsection 3.4.4 presents a selection of measurement results. Finally, we conclude our work in subsection 3.4.5.

3.4.2 Related Work

There are only a few publications that compare ODMRP and SMF. In [22], Bongartz et al. evaluate WNet, a uni- and multicast MANET routing protocol with focus on multimedia communications. The protocol was intended for a communication system in a setup with multiple mobile robots. It takes into account information from the link layer for the calculation of the routing metric. The protocol is evaluated in ns-2 simulations, using ODMRP and SMF as reference protocols. The authors use SMF with classical flooding (as described in section 3.4.1.1), as well as with MPRs based on NRL’s implementation the Optimized Link State Routing Protocol (OLSR). In a scenario with 25 mobile nodes and 3 concurrent multicast streams the loss rate of ODMRP was slightly higher than that of both variants of SMF. The results of other metrics like delay or jitter were dependent on the SMF variant.

In [54] Ernst et al. compare the performance of SMF and ODMRP in tactical underwater scenarios. They use a custom implementation of both protocols and extended SMF with MPRs. Two traffic patterns are evaluated in a chain-topology using ns-2 simulations. In the first pattern, data is sent from one end of the chain to the other end. In the second pattern, all nodes of the chain send data to a gateway node that is positioned alongside the chain. Ernst et al. concluded among others, that the throughput does not decrease significantly with the number of hops along a route. Moreover, the performance of ODMRP was superior to that of SMF if there was enough bandwidth for topology information. They also showed that under high load both routing approaches fail due to routing packet
loss.
In [17], Bauer et al. present an optimization for ODMRP for the deployment in underwater networks. The extension called Route-Discovery-Suppression (RDS) prevents a node from performing a second route discovery, while another discovery is already in progress. The paper presents a simulative evaluation of the approach within a 6x6 and a 9x9 grid and a performance comparison to SMF, simple flooding, and standard ODMRP. Besides demonstrating that RDS can significantly reduce the protocol overhead of ODMRP in the scenarios considered, their results show that SMF has a significantly higher overhead compared to standard ODMRP. This is caused by the proactive approach of SMF that generates more overhead than the scoped flooding scheme of ODMRP. Only at very low sender rates, SMF can outperform ODMRP concerning the overhead factor as ODMRP requires a certain amount of user traffic in order to compensate for the additional control traffic. Their results also indicate that ODMRP can yield to a lower packet delivery ratio at different sender rates.

### 3.4.3 Testbed and Methodology

As a basis for our measurements we use the *CommunityLab* testbed software that has been developed in the CONFINE project. The CONFINE project is active in building large scale testbeds that are integrated into real world wireless community networks. However, since our research is focused on dense ad-hoc deployments, we created an indoor testbed for dense mesh networks at our institute as part of the CONFINE project. For more details on the hardware components used in the testbed see [38]. Although our deployment is not part of an actual Community Network, we follow the same architecture used in other CONFINE deployments. Essentially, we distinguish between a *Community Network* (which acts as a backbone for management purposes) and a network for experiments.

#### 3.4.3.1 Physical Testbed at our Institute

Figure 3.35 illustrates a subset of the full testbed topology (c.f [38]) used in this work. The testbed spans across two floors of a large office building with a floor length of about 100 m. Here, we use nine nodes located on a single floor. Most of the nodes are placed in regular offices, while one node is located in a server room (and possibly exposed to more interference).

![Figure 3.35: Topology of the testbed at the Fraunhofer FKIE, each number representing a research node.](image)

We currently use IEEE 802.11n channels within the 5 GHz band around 5.7 GHz. This frequency range is not deregulated in Germany and may only be used with a proper license. By using this non-public frequency range, we can achieve a very low degree of interference with surrounding wireless networks. The radios are equipped with omni-directional antennas achieving a signal gain of 12dBi in all horizontal directions.
All of the antennas are mounted in a straight vertical orientation. Due to the antenna characteristics, this results in high signal strength regarding neighbors on the same floor but almost no connectivity to neighbors located directly above or below a node.

Figure 3.36 shows the average link quality measured by ODMRP over 500 samples (calculated in intervals of 10 seconds). The values reflect the broadcast packet delivery ratio in the direction indicated by the arrows. Each value is given with its standard deviation. It can be seen that the link quality is significantly affected by the number and structure of walls in between the nodes (as can be expected). High quality links (i.e., a ratio higher than 0.9) tend to be very stable, while links with low quality show a high degree of fluctuation. In addition, we can observe that several links are highly asymmetric. This should be due to specific characteristics of the environment and the antenna positioning, since the network is in (almost) idle state.

3.4.3.2 The Virtual CONFINE Testbed

A general problem of real-world testbeds is that it is difficult to create a controlled measurement environment independent from outside influences. In order to make the process of developing experiments more user friendly and to prevent disturbances by misbehaving experiments in an early stage of development, the Virtual Confin Testbed (VCT) has been developed. VCT allows for the easy creation of a virtual network of CONFINE nodes by providing a virtualization of the CONFINE hardware and networks. For the emulation of the network, different emulation modules can be plugged into VCT. As a simple approach, we use network emulation based on EBTables (a standard linux tool). This adds packet loss to ethernet links using a statistical model. As a more realistic approach, we integrate VCT with ns-3, which is able to emulate the radio devices on the Phy/MAC level.

3.4.3.3 Testbed Applicability

The choice of omni-directional antennas in the testbed is a worst case scenario in terms of community networks. Especially in densely populated cities, many Community Networks use point-to-point wireless links wherever possible. Other Community Networks use a mixture of omni-directional antennas for easy to set up short range mesh links and directional antennas only for long distance links [63]. In any case, one can not assume that all links are perfectly planned as they are also deployed by non technical expert community members. Therefore, and because we use typical hard-
and software also used in community networks as a basis, we believe that our measurements are at least qualitatively transferable to Community Networks with a different topology.

3.4.4 Evaluation

In this section, we present a selection of results from the measurements performed in our virtual and physical testbed. We first define the evaluation metrics and the set of parameters used for the protocols and then discuss the results.

3.4.4.1 Evaluation Metrics

Our comparison is based on common network performance metrics, which are defined as follows.

- **End-to-end delay** The time between the generation and the arrival of a packet at the application layer. Due to the global time synchronization required, this metric is only considered for the emulative measurements.
- **Network load** The sum of the size of all packets which are received or sent by a particular node. This metric is used as an indicator for the efficiency of a protocol.
- **Packet loss ratio (PLR)** For a given measurement, this metric is calculated as follows:

\[
\text{PLR} = 1 - \frac{\text{#received packets}}{\text{#expected packets}}
\]

3.4.4.2 Testbed and protocol configuration

For the emulated testbed, we chose a 3x3 grid topology with log-distance and Rician (small-scale) fading (parameters as listed in 3.5(a)). The nodes are positioned at a horizontal and vertical distance of 60 meters, resulting in rather low-quality links regarding vertical and horizontal distant (two-hop) neighbors. The nodes are numbered from 1 to 9 (such that the first and last node are located on opposite edges of the grid).

In the physical testbed, we use IEEE 802.11n with a channel that yields minimum interference with existing networks. Table 3.5(b) lists the most relevant parameters. In order to have a setup comparable to the emulation, we only use one floor of our physical testbed, consisting of ten nodes numbered from 130 to 139, as shown in figure 3.35.

SMF operates in a pure flooding mode with duplicate detection. For the configuration of ODMRP the parameters listed in Table 3.5(c) are used.

In both, VCT and the physical testbed, we use a single sender CBR traffic to generate multicast load. The receiver group is composed of all other nodes. The CBR generator is configured to send 1000 packets with 128 bytes of payload, at a rate of 10 packets per second.

3.4.4.3 VCT Measurements

Figure 3.37 shows the average end-to-end delays regarding each multicast receiver. As expected, the results are very similar for both protocols, since they use the same kind of broadcast transmissions. It can be seen that the delays achieved with SMF are slightly smaller. However, it has to be noted the difference is in the scale of a few milliseconds. Taking into account the (relatively) low amount of user traffic generated, this effect should be due to ODMRP’s control traffic.
3. Self-management
3.4. Multicast Performance Comparison of SMF and ODMRP

(a) Virtual testbed parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IEEE 802.11g</td>
</tr>
<tr>
<td>Frequency</td>
<td>2.412 GHz</td>
</tr>
<tr>
<td>Transmit Power</td>
<td>11.34 dBm</td>
</tr>
<tr>
<td>Antenna Gain</td>
<td>0 dB</td>
</tr>
<tr>
<td>EnergyDetectionThreshold</td>
<td>−90 dBm</td>
</tr>
<tr>
<td>CcaMode1Threshold</td>
<td>−99 dBm</td>
</tr>
<tr>
<td>Multicast Bitrate</td>
<td>1 Mbit</td>
</tr>
</tbody>
</table>

(b) Physical testbed parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>IEEE 802.11n</td>
</tr>
<tr>
<td>Frequency</td>
<td>5.745 GHz</td>
</tr>
<tr>
<td>Transmit Power</td>
<td>13 mW</td>
</tr>
<tr>
<td>Antenna Gain</td>
<td>12 dBi</td>
</tr>
<tr>
<td>Multicast Bitrate</td>
<td>6 Mbit</td>
</tr>
</tbody>
</table>

(c) ODMRP parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello Interval</td>
<td>10 s</td>
</tr>
<tr>
<td>Refresh Interval</td>
<td>3 s</td>
</tr>
<tr>
<td>Forwarding Group Timeout</td>
<td>10 s</td>
</tr>
<tr>
<td>Route Discovery Suppression</td>
<td>enabled</td>
</tr>
<tr>
<td>Link Quality</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>Overhead Reduction</td>
<td>enabled</td>
</tr>
</tbody>
</table>

Table 3.6: Configuration parameters used in the evaluation

In figure 3.38, the packet loss ratio is presented for each multicast receiver. The error bars indicate the 95% confidence interval calculated over ten replications. We can see that SMF is able to almost completely prevent any packet loss in this scenario, due to the redundancy caused by its flooding mechanism. In the case of ODMRP, however, the packet loss is still considerably low regarding most receivers. An exception is node nine, which is the most distant receiver (located at the other end of the grid topology). A closer analysis of ODMRP’s forwarding group selection revealed that the forwarding group selection is suboptimal in this scenario, as the protocol tends to prefer the use of diagonal links between nodes. This issue might be solved by a different parameterization of the link quality extension.

Considering the previous two metrics, we were able to examine that ODMRP performs well for most receivers in the grid. Nevertheless, in order to assess the overall performance, the network load needs to be taken into account as well. This metric is presented in figure 3.39. It shows the incoming, outgoing and total network load regarding both protocols, recorded during a single multicast measurement. We selected node 5 (the center node) as a single representative node for this part of the evaluation, since it is the most exposed node in the network. We can see that the amount of outgoing traffic is very similar for both, ODMRP and SMF. However, significant fluctuation can be seen in the case of ODMRP. The load may increase due to the periodic flooding of control messages (by sender as well as receivers), and decrease due to the selection of forwarding nodes (relative to the amount of offered load). The latter may fluctuate significantly over time. In addition, packet loss causes the load to decrease. Regarding total and outgoing network load, however, we can see that ODMRP...
3.4. Multicast Performance Comparison of SMF and ODMRP

3. Self-management

Figure 3.37: End-to-end delay in the VCT testbed

Figure 3.38: Packet loss in the VCT testbed

achieves a significantly lower load compared to SMF. Considering the low amount of packet loss (for most nodes), it can be seen that ODMRP is able to compensate for its overhead by means of reduced redundancy.

3.4.4.4 Testbed Measurements

Figure 3.40 shows the packet loss ratio measured in the physical testbed. Confirming previous results (see figure 3.38), we can see that ODMRP causes a higher amount of packet loss than SMF. However, in contrast to the emulated testbed, SMF is also affected by packet loss. Taking into account the topology shown in figure 3.36, this result is expected. Since almost all of the links in the network are affected by packet loss, broadcast redundancy is required to prevent the loss ratio from accumulating on a multi-hop path. With its link-quality extension, ODMRP can be expected to achieve a similar
3.4. Multicast Performance Comparison of SMF and ODMRP

The overall result is quite similar to the one presented in figure 3.39. However, SMF shows a similar degree of fluctuation to ODMRP in the outgoing load (although, in this case, the load can not exceed the offered load). This is caused by packet loss due to the signal fading and collisions on the physical layer, as we do not induce congestion (see figure 3.36). The fact that ODMRP shows a reduced overall amount of load suggests that it has a good chance to compensate for its additional

**Figure 3.39:** Network load during a multicast measurement in the VCT testbed

**Figure 3.40:** Packet loss in the physical testbed

degree of redundancy for most nodes, e.g., when sending data from node 130 to node 134, it may include both nodes 131 and 138 in the forwarding group. On the other hand, the chain-like topology limits the degree of redundancy in comparison to the emulated grid topology. In a high-density topology (with more possible paths between nodes), the difference between the protocols might therefore be more significant.

Again, we examined the load generated in the network in order to assess the performance of the protocols in terms of the overhead (see figure 3.41). This time, we selected node 136 (see figure 3.35 for reference). The overall result is quite similar to the one presented in figure 3.39. However, SMF shows a similar degree of fluctuation to ODMRP in the outgoing load (although, in this case, the load can not exceed the offered load). This is caused by packet loss due to the signal fading and collisions on the physical layer, as we do not induce congestion (see figure 3.36). The fact that ODMRP shows a reduced overall amount of load suggests that it has a good chance to compensate for its additional
load in this scenario.

3.4.5 Conclusion

We have presented initial measurement results for two very practical approaches to multicast routing in WMNs. The simple but yet extensible approach of SMF uses a large degree of redundancy, aiming at high reliability for many scenarios. ODMRP on the other hand, provides a reactive selection of forwarding nodes, in order to reduce the number of redundant transmissions in the network. For ODMRP, we used extensions presented in our previous work, which add features such as link-quality based routing and overhead reduction techniques.

We used a two-step evaluation process to compare the protocols. Using ns-3 based emulations, we were able to demonstrate that although ODMRP tends to achieve a higher packet loss ratio, it is likely to compensate for this by having a reduced overall network load. By means of real-world testbed measurements, we partially confirmed the observations made in the first step. However, we also observed that SMF did not perform as well in our physical testbed as it did in the emulation.

The limited scalability of flooding-based protocols like SMF makes it very difficult to use them in large WMN deployments like community networks. Even if a multicast receiver group only consists of a subset of the topology (e.g., when using MPRs), traffic still needs to be broadcasted through the whole network. We believe that the scoped flooding approach of ODMRP might increase the applicability of multicast routing in larger networks. Further performance enhancements might be integrated (such as route discovery in a limited topology subset). We therefore plan to perform measurements with larger topologies and within actual community networks in the future.

Using a solid environment for network emulation and a realistic testbed, we plan to perform more extensive evaluations and the optimization of routing approaches. In particular, we will be able to enhance the meaningfulness of our results by providing more real-world evidence for the performance

Figure 3.41: Network load during a multicast measurement in the physical testbed
of protocols.

3.5 Performance of OLSRv2 with DAT metric

In this section we present a performance comparison between OLSR here called OLSRv1 (with the *Expected Transmission Count* (ETX) metric) and OLSRv2 (with our novel *Directional Airtime* (DAT) metric). We propose the use of the DAT metric as it is suitable for the heterogeneous link characteristics often found in Community Networks. In our measurements performed using the physical testbed at FraunhoferFKIE institute, we achieve a more stable route selection process and improved throughput in comparison to the OLSR-based approach currently used in Community Networks.

3.5.1 Physical Testbed at the Fraunhofer FKIE

![Example of a hybrid node within the testbed](image)

*Figure 3.42: Example of a hybrid node within the testbed*

As part of the CONFINE project, we created an indoor testbed for dense mesh networks at our institute. It serves both as an environment for wireless experiments, as well as a prototype deployment of the hybrid node architecture described in [39]. The testbed spans across two floors of a large office building with a floor length of about 100 m. It currently consists of 22 hybrid nodes, each equipped with a single external IEEE 802.11n radio.

For the evaluation we use a subset of these nodes located on the same floor (see figure 3.43). Most of the nodes are placed in regular offices, similar to the setup shown in figure 3.42. An exception is node 137, which is located in a server room with higher attenuation and interference. All of the antennas used are mounted in a straight vertical orientation. Due to the antenna characteristics, this results in high signal strength regarding neighbors on the same floor, but almost no connectivity to neighbors located directly above or below a node.

Although our deployment is not part of an actual Community Network, we follow the same architecture used in other CONFINE deployments. Essentially, we distinguish between a *Community Network* (which acts as a backbone for management purposes) and a network for experiments. A detailed description of the testbed as well as a discussion on its applicability to Community Networks is included in [38].
3.5.2 Evaluation of OLSRv2 Route Selection

In this section we present a selection of results from measurements conducted in our physical testbed. We performed a basic performance comparison between OLSRv1 and OLSRv2 using the testbed described in section 3.5.1. Section 3.5.2.1 presents details about the testbed topology selected for the experiments while section 3.5.2.2 defines the parameters used for the measurements. In section 3.5.2.3 we discuss the results regarding throughput and route selection.

3.5.2.1 Topology

![Network topology](image)

**Figure 3.43:** Subset of the testbed topology at the Fraunhofer FKIE used for measurements. Each number is representing a research node. The red dots indicate the approximate antenna positions.

![Network topology](image)

**Figure 3.44:** Network topology with a long distance link as seen by OLSRv2 with DAT metric during TCP throughput measurements.

Figure 3.43 illustrates the physical testbed as described in 3.5.1. Each number corresponds to a hybrid node. The resulting network topology is depicted in figure 3.44. The directional labels provide the average inverse DAT link values normalized to Mbit/s as seen over 20 replications of TCP throughput measurements from 130 to all other nodes, including their standard deviation. These values provide a good indication of link qualities. The physical link speed as configured by the Minstrel algorithm [152] as well as the expected number of retransmissions estimated by the OLSRv2 loss monitoring are included in the calculation of the depicted values. The arrows indicate the directionality of the link and refer to the horizontal position of nodes in this figure. Note, that some walls are load-bearing and also provide higher attenuation. This is indicated via a thicker line in figure 3.43. An example is the connection between 134 and 133, which are in neighboring rooms but provide only half the speed of the links between 132 and 134 as well as between 133 and 135. The link between 131 and 137, depicted by a dashed line, is a special case and used only for special experiments. It represents a low
speed long distance link and is realized by a GRE Ethernet tunnel via the switched Ethernet network of the building. Traffic shaping is done via a token bucket filter. In general, one can notice that the standard deviations are relatively small.

### 3.5.2.2 Measurement Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PHY Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td>IEEE 802.11n</td>
</tr>
<tr>
<td>Frequency</td>
<td>5745 MHz</td>
</tr>
<tr>
<td>Transmit Power</td>
<td>13 mW</td>
</tr>
<tr>
<td>Antenna Gain</td>
<td>12 dBi</td>
</tr>
<tr>
<td>OLSRv1</td>
<td></td>
</tr>
<tr>
<td>Hello Interval</td>
<td>2 seconds</td>
</tr>
<tr>
<td>TC Interval</td>
<td>5 seconds</td>
</tr>
<tr>
<td>OLSRv2</td>
<td></td>
</tr>
<tr>
<td>Hello Interval</td>
<td>2 seconds</td>
</tr>
<tr>
<td>TC Interval</td>
<td>5 seconds</td>
</tr>
</tbody>
</table>

**Table 3.7**: Parameterization of Testbed

Table 3.7 lists the relevant parameters we used for the configuration of the physical layer as well as for OLSRv1 and OLSRv2, respectively. Due to the high-gain antennas used, we are able to use a fairly low transmission power level, thus reducing the degree of interference in the network. The link speeds between neighbors are determined by the Minstrel-algorithm. Since this algorithm requires unicast transmissions on each link, we use a periodic probing mechanism for OLSRv2. This ensures that the link speeds reflect the physical conditions of the network. For OLSRv1, such a mechanism is not required as the link speed is not taken into account for routing decisions. The throughput data were obtained using `wget` to transfer a 10 MB file.

### 3.5.2.3 Evaluation Results

For all performance metrics we performed measurements from a single source node to all other nodes. In the following, we present the results for source node number 130. This node is particularly interesting, because it is located at the border of the network. In order to reach the opposite end of the testbed, all nodes in between can act as intermediate hops, allowing for diverse routing decisions.

#### 3.5.2.3.1 TCP Throughput

Figure 3.45 (Standard Topology) shows a direct comparison between the TCP throughput achieved with OLSRv1 and OLSRv2 in a homogeneous Wi-Fi topology, respectively. The bars denote the average throughput measured over twenty replications. The first observation is that there is a strong correlation between the throughput and the spatial distance of a node to the source node (note, that the locations of node 134 and 133 are interchanged). This is due to the inherent self-interference caused by multi-hop transmissions, which leads to a degradation in throughput with an increasing number of hops. Nevertheless, we can see that for all of the target nodes, the throughput is equal to (within the level of confidence) or higher for OLSRv2 compared to OLSRv1.
3.5. Performance of OLSRv2 with DAT metric

The significance also increases with the distance. Especially at node 138, the difference is quite high. Here, the throughput with OLSRv2 is about three times higher than for OLSRv1. With OSLRv1, for all target nodes from 135 to 139 a similarly poor transmission rate of below 2.5 Mbit/s is achieved. With the exception of node 138, this is probably due to at least the temporary use of the slow link between 134 and 135. This link is assessed by OLSRv2 with about 1 Mbit/s on average (confer figure 3.44). OLSRv2 on the other hand achieved TCP throughput rates of between 3 and 4.5 Mbit/s for node 135, 136, and 139. For node 137, the throughput is slightly lower because it has in general relatively poor connectivity (remember that it is located in a server room). Node 138 is quite close to the source 130. Here, the difference between OLSRv1 and OLSRv2 is especially high, probably because OLSRv1 prefers the shorter route including link 131 to 138 and OLSRv2 takes the detour via node 132 with high speed links. This results in over 3 times the throughput to node 138 for OLSRv2 in comparison to OLSRv1.

A problem of long distance links is that they typically achieve relatively low data rates but also low

---

**Figure 3.45:** Results of the TCP throughput measurement with 20 replications and a confidence level of 0.95
3. Self-management

3.5. Performance of OLSRv2 with DAT metric

Loss rates. To compare OLSRv1 and OLSRv2 with the DAT metric in this scenario, we included the long distance link between nodes 131 and 137 (depicted as dashed line in figure 3.43). The results are shown in figure 3.45 (Heterogeneous Topology). For nodes 131 to 133 as well as for node 138, we observe the same performance in terms of throughput as in the scenario without the long distance link. Regarding the transmission to node 134 and to node 135, we observe a deterioration of the throughput while for nodes 136, 137, and 138, the throughput with OLSRv1 is about equal to that of the 200 kbit/s of the long distance link. The expected reason is that for OLSRv1 with the ETX metric, the long distance link attracts all routes to the neighborhood of node 137. Nodes 134 and 135 are expected to be in a break-even area, where it depends on the current loss rates of the links if OLSRv1 takes the long distance links or an alternative route.

The observations in this subsection represent a user centric, statistical perspective on the performance of OLSRv1 and OLSRv2. In the next subsection we will provide an in-depth analysis regarding the route selection strategies of OLSRv1 and OLSRv2. It supports many of the presumptions made in this subsection.

![Figure 3.46: Route selection with OLSRv1, destination node 139](image1)

![Figure 3.47: Route selection with OLSRv2, destination node 139](image2)

3.5.2.3.2 Route Selection Strategy The above measurement showed significant differences in the achieved throughput of OLSRv1 and OLSRv2 regarding several destination nodes. In order to further analyze the observed results, we examined the route selection during data transmission. Figures 3.46 and 3.47 illustrate the amount of data transmitted over each link in the network during...
the transmission of a 10 MB file from node 130 to node 139. The links are annotated with the amount of data sent in each direction, the red numbers showing the direction on the path towards the receiver and the blue color showing the reverse direction. Links not used for transmission are shown without annotation. We performed five measurements and calculated the average value for each link.

Figure 3.46 shows that OLSRv1 chooses the same three-hop route for all replications, which includes the long distance link between node 131 and 137. Since the long distance link is lossless, the value of 10.36 MB sent in the direction of the receiver comprises the data payload including TCP and IPv4 headers as well as TCP retransmissions. On the link between node 130 and 131, some MAC retransmissions occur, resulting in a slightly higher number of transmitted bytes.

In contrast, OLSRv2 selects paths not including the long distance link, as can be seen in figure 3.47. Instead, it prefers paths with more hops, yielding a smaller airtime than the three-hop path with the long distance link. We can see that the path length is five hops most of the time, since data is mostly transmitted over the link between node 131 and 134. In one single replication, the path included node 132 during the whole transmission, leading to the average value of 2.1 MB on the respective links. Similarly, some of the TCP acknowledgements sent on the reverse path took the extra hop over node 135.

Taking into account the results from the throughput measurements, it can be seen that OLSRv2 clearly outperforms OLSRv1 for this receiver node, despite the selection of longer routes. This also accounts for data transmissions to node 136 and 137. For receiver nodes 133, 134 and 135, which are located in the center of the topology, OLSRv1 fluctuates between routes including and not including the long distance link. This is due to the fact that routes of both types are similar in terms of ETX cost, but differ significantly in terms of bandwidth. Figure 3.48 illustrates this effect for receiver node 135. From the figure it can also be seen that the route selection of OLSRv1 appears to be less stable. The data is distributed over a larger set of links, from which the used routes are chosen.

In our measurements, we have demonstrated that the performance of OLSRv1 can be affected significantly in networks that use heterogeneous transmission technologies with different link characteristics. Our results indicate that OLSRv2 in combination with the DAT metric can increase the overall performance in such networks.

**3.5.3 Conclusions**

We investigated user centric performance metrics like TCP throughput in our novel Community Network dense testbed at the Fraunhofer FKIE. The testbed uses the same hybrid node setup (radio and...
router separation) suggested for Community Network nodes with low attenuation between the radio and its antenna, allowing for good connectivity between different rooms despite of the use of 5 GHz Wi-Fi. Using omni-directional antennas on the one hand provides a worst case scenario for community networks, on the other hand, some Community Networks use this setup as it is easier to deploy and cheaper to build than carefully planned point-to-point links. However, we think that the results are transferable to point-to-point deployments in terms of quality. A central observation is that our proposed routing metric for OLSRv2 at least equals and in many cases outperforms OLSRv1 with ETX metric. DAT is particularly unsusceptible against a special problem of ETX. With the ETX metric, the performance of OLSRv1 can be significantly degraded in networks that use heterogeneous transmission technologies with different link characteristics. OLSRv2 with the DAT does not show this flaw and is thus also more robust in Community Networks where there is no central authority for planning the network topology.

For Community Networks, the most important question is if this is the right moment to switch from OLSRv1 to OLSRv2. The OLSRv2 implementation has matured up to a point that it can be deployed in larger networks and it outperforms OLSRv1 in many situations. Especially if the links are heterogeneous with a wide variety of link speeds, the new DAT metric produces good network paths more consistently. Naturally, a network-wide deployment might reveal new issues and smaller bugs in the implementation. However, without such deployment these issues will never be found. A reasonable migration path to a full deployment of OLSRv2 is to deploy it for IPv6 traffic right now. Here, OLSRv2 was especially efficient in terms of protocol overhead because of its address compression capabilities. When potential remaining issues have been fixed and the integration is done, switching from a parallel OLSRv1 (IPv4) and OLSRv2 (IPv6) deployment to OLSRv2 dualstack IPv4 and IPv6 mode will be just a change of configuration. Thus Community Networks should now start the migration to OLSRv2. OLSRv2 can easily be combined with our node architecture which is more scalable and provides additional benefits for example regarding maintainability.

3.6 Analysis of the social effort in multiplex participatory networks

Community networks are participatory connectivity solutions for citizens where all the resources are owned, managed and controlled by the participants. As a natural evolution, in recent years some initiatives have flourished to provide higher level services based on volunteer computing and resource sharing paradigms. A fundamental aspect of these paradigms is the user participation. In this section, we apply some social mining techniques aiming to identify the roles of the individuals in the social network behind a community network, here Guifi.net, and to measure the participatory involvement in the community network from 2003 to 2014.

We observed that community network participants generally dedicate their time and effort to a single participatory forum, generating several types of community structures. We analyzed such structures using a multiplex network formed by mailing list in Guifi.net and a relationship graph built pairwise of users that share a physical wireless link. We were able to distinguish between non-hierarchical participatory forums, where almost all users are part of the same big community and two-tier participatory forums leaded by a small number of users that act as social bridges between their members. Finally, by testing the impact of community leaders in all participatory layers, we profiled the utility of the members’ effort to the whole wireless community network.
3.6. Analysis of the social effort in multiplex participatory networks

3.6.1 Community Network Member Participation

Beyond Internet access provision, the community networks’ physical infrastructure is sometimes used by some members to provide applications (e.g. web servers, monitoring systems). As a natural evolution, some community networks are looking for ways to implement higher level applications [99], which would require mechanisms to regulate and normalize how their members interact with the computational resources [141]. The feasibility of implementing such contributory systems is highly dependent on the network participants’ ability to rank and evaluate members’ participation.

Typically, the deployment and management tasks are performed by the community network members, mostly volunteers. Avonts et.al. [7] reported that community networks members considered finding and keeping volunteers the largest organizational challenge, next to funding and finding devices maintainers the most second important challenge.

User participation can be measured in multiple ways and it is not limited to deploy and maintain physical devices or links. Community networks usually have other participatory forums where users can contribute to the growth and improvement of the network. Some community networks maintain online discussion forums while others use mailing lists or organize face-to-face meeting activities. These forums help users to organize and give support to new members to integrate into the network.

Figure 3.49 shows the empirical cumulative distribution function (ECDF) plotted as the Lorenz curve, of users participation in Guifi.net, the largest community network to the best of our knowledge. The participation is measured separately as the number of new devices created by users and the number of messages exchanged in one of its participatory mailing list. The Gini coefficient [58], measured as the area between the line of equality and each of the curves, is close to the absolute inequality in both participatory forums – 0.8358 in the devices creation and 0.8320 in the message exchange. The Lorenz distribution function also suggests that network members behave differently in terms of participation in the examined forums.

Figure 3.49: Gini coefficient of two participatory forums in Guifi.net.

As an example of individual different participatory involvement, we consider the number of messages and devices created by each user identified in both participatory forums in Figure 3.50. We observe that most of the users are selective and choose to collaborate only in one of the participatory forums, contributing with little or nothing to the other. For example, there is a high concentration of users
participating in the development mailing list, but these are users which contributed only with one or two devices to the physical communication network.

![Figure 3.50: Guifi.net users participation measured as the number of messages posted in the mailing lists users-list and dev-list, and the number of new communication devices.](image)

In this work we measure the level of user involvement in a wireless community network by applying social mining techniques to understand the differences observed among users of several participatory forums. We gather and process information from 13,407 registered users in the community network with more than 36,629 active communication devices, and a total of 10,045 threads and 49,355 messages in their most active mailing lists. We then study the evolution of participation and observe that participatory forums are currently in a mature state, which according to [73] suggests a relationship and subgroups analysis. Therefore, we conduct the analysis of interactions and community structure in each participatory forum separately. We find evidences of hierarchical community structures in the network’s mailing lists with strong ties among their members and poor communication with other communities. We also observe a lack of structure in the communication network. Finally, we measure the significance of the members in each participatory layer to the community network as a global entity. Our methodology is inspired by [70], which conducted work on a multiplex and multi-layer analysis of online and offline users’ interaction, where the authors discuss the existence of weak and strong ties among community members.

The contributions made in this work can be summarized as follows:

- We describe the evolution of members’ activities since the creation of the network until 2014. We observed that the network is currently in a mature stage and therefore the community and relationship analysis applies. Additionally we found evidence of seasonal participative patterns.
- We identify different community structures on each participatory layer, and observed that weak relations arise more in the communication layer, while strong relations are more common among members of the small communities that govern the mailing lists.
- We measure the participants’ social value as individuals using centrality measures, and observed that only a small portion of the participants can be considered of having a high social value.
- In terms of their impact on other layers, we simulate the robustness of the network against the disappearance of important members, and found that their activities in each participatory forum should be measured differently, since their impact on the community network is also different.
3.6.2 Experimental framework

In this section we present our framework for the analysis of the social interactions and the estimation of the users’ effort in Guifi.net. We first introduce the Guifi.net wireless community network. Then, we describe the current participatory forums and how we aggregate their data to build our analysis graphs. Finally, we explain the assembly process used to build the multiplex graph.

3.6.2.1 Guifi.net network and information gathering

The Guifi.net network started in 2004 and in 2014 it has reached more than 24,000 operational devices, most of them in Catalonia and nearly all of them in the Iberian peninsula. In 2008 their members created the Guifi.net Foundation, a non-profit organisation responsible to coordinate the volunteers and provide deployment support to its users. The Guifi.net foundation encourages the deployment of the network, but does not control it. Therefore, virtually all the decisions concerning the network growth and maintenance are up to the users.

Physical nodes and communication layer representation: The network consists of a set of nodes interconnected through mostly wireless equipment that users must install and maintain, typically on building rooftops. The network grows driven by the needs of individuals. New links only appear based on the needs for connectivity by their owners or indirect beneficiaries (e.g. users of a community network crowd-funded by municipalities). Deploying a new node, or improving the connectivity for an existing one, is difficult without the cooperation of the owner/manager of the communication device to connect with. The communication layer intends to capture such interaction between participants in a graph structure to ease the analysis.

Using the topological graph built previously [141] and a dump of the Guifi.net web database, we relate each owner with their active devices as we show in Figure 3.51. Then, we build an undirected graph where vertices represent the members of the network and edges represent a link in the topology graph that connects two nodes owned or modified by one of the users. Additionally, the weight of the vertices represents the number of nodes created, while the weight of the edges stands for the number of links between users.

Mailing lists and social layer representation: Social participation in Guifi.net changed during the community network lifetime. Nowadays, mailing lists are the only online and social participatory forums left. The Guifi.net Foundation maintains two general-purpose mailing lists to coordinate users and developers since 2006. The first of them, users-list, is mainly used to discuss general topics, issues on coordinating physical infrastructure creation and maintenance and to help new users. The other one, dev-list, serves as a communication channel between some of the most active members in Guifi.net, most of them developers. Each mailing list is currently managed independently and contains only a small subset of the users registered in the web page – i.e. the dev-list reports 401 different users registered, while the Guifi.net webpage reports 13,407.

We used web scraping to gather information from the mailings lists dump published on the Guifi.net web site. Following the same methodology proposed by [50], for each mailing list we built a tree for each thread (see Figure 3.51). Then, we used the sender email information to build a participants directed graph that includes as vertices the users in the mailing lists. While in the original framework the authors considered that connections between participants are reciprocal, we instead considered them directional. Therefore, in our graph an arc from user \( v \) to \( u \), represents an answer from \( v \) to a message previously sent by \( u \) in the same thread, while arc’s weight stands for the amount of messages ever sent by \( v \) to \( u \). We use vertices weight to keep track of the threads created by each user.
3.6.2.2 Homonymous detection

Homonymy is a characteristic of most distributed systems, like Peer-to-Peer applications, which implies the existence of users with multiple identifiers in the network [120]. Guifi.net homonymous exists because authentication is based on the email address of the participants and because the authentication for each participatory forum is independent. Homonymy was detected using the email similarity rule suggested by Bird et. al. [19], which is based on the Levenshtein edit distance between email address bases. Being as conservative as possible, we tag two identities as homonymous only if they have a distance of 1 or below.

3.6.2.3 Communities detection

Social relations in a group of participants can form a community if they are more willing to interact among them than with other members of the network. This is a well-known phenomena – called communities structure – that arises in most complex networks. The community’s size, structure or even members’ interactions outside and inside the communities are a good source of information to study the roles of the network users.

The community detection problem has been studied for a long time, and there are different algorithms and methods that can be applied, depending on the properties of the network and the properties of the targeted communities. In this work we apply two different methods, the clique percolation method [116] and the Louvain method [20] to detect two different community structures, and discuss the differences and the role of their members. In practical terms, the difference is that while the percolation method is based on the detection and aggregation of k-clique disjoints sets inside the graph – which will have maximum connectivity among their members –, the Louvain method is an optimized algorithm to find partitions providing that the modularity (the relationship between average degrees inside the community and intra communities) is minimized.

Multiplex graph: multiplex or multi-level graphs are abstract data structures which assemble the information of several graphs in such a way that each original graph is represented by a separated layer, meaning that each layer holds their original connectivity matrix [81]. The basic properties of each layer are summarized in Table 3.8. The structure of the multiplex enables to relate nodes with the same identifier between them. We built a 3-layer multiplex graph as the assemble of the participation layers (see Figure 3.51). We use it to discuss the impact of the most active members of each layer in the whole network.

3.6.3 Network evolution

In this section we study the activity performed by individuals on each participatory forum since the creation of the network. We analyze the activity of users by measuring the number of new physical communication devices registered in the network and the number of messages and threads sent in both mailing lists. Group activities are compared with the individual’s interest, which lead us to the

---

Table 3.8: Summary of the basic properties of the graph layers

<table>
<thead>
<tr>
<th>Network layer</th>
<th>Type</th>
<th>Nodes</th>
<th>Edges</th>
<th>Degree (max,min,avg.)</th>
<th>Components</th>
<th>Average distance</th>
<th>Diameter</th>
<th>Clustering coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>undirected</td>
<td>1919</td>
<td>3201</td>
<td>384,0,3.3861</td>
<td>250</td>
<td>3.238</td>
<td>7</td>
<td>0.1520</td>
</tr>
<tr>
<td>Users</td>
<td>directed</td>
<td>538</td>
<td>4607</td>
<td>521,0,17.1264</td>
<td>19</td>
<td>1.9675</td>
<td>2</td>
<td>0.751</td>
</tr>
<tr>
<td>Development</td>
<td>directed</td>
<td>401</td>
<td>3926</td>
<td>202,0,19.5810</td>
<td>13</td>
<td>2.3247</td>
<td>4</td>
<td>0.5614</td>
</tr>
</tbody>
</table>
conclusion that Guifi.net is in a mature state with evidence of lack of interest by non-members.

### 3.6.3.1 Network state detection

Figure 3.52 shows the activities done day by day by network participants in each participatory layer as the simple moving average at intervals of approximately one month (30), quarter of a year (90) and one year (360). We observe that users’ participation switches from long periods of high activity to short periods of less activity. We identify the periods of lower activity, as those that correspond to the last quarter of every year – Spanish winter season.

The last cycle of less activity has been extended in the communication layer after a long period of increasing activity which lasted for 77 months. This change of trend was predictable as the number of new registered users have been decreasing the past 4 years. Nowadays only 20% of the new working devices are installed and managed by users registered 62 days before or less.

Regarding the activity in both mailing lists, it is not comparable because the number of members is quite different (25.46% difference according Table 3.8). However we can observe that, on average, the participation by user in the development list used to outnumber the participation in the users list.
3.6.3.2 Interest generated

Users’ habits have changed over the time. While the interest in the communication network has decreased since 2010, the users interest in the discussions and information exchange prevailed one year and a half after. Nowadays, while developers activities are decreasing, the rest of the users contribute to the increase of the users mailing list (see Figure 3.52).

We conclude that voluntarism – measured as participation in the three layers – is stable in the network, and nowadays is concentrated around the senior members of the network. The network is still in a mature state, but is attracting fewer new users every day.

3.6.4 Communities and multiplex analysis

From the examination of the structure of the participatory forums in Guifi.net, we can consider the network mature enough to be subject of a community and interaction analysis. First, we focus on detecting possible independent communities which will capture the preferential interactions among members. Then, we address the detection of key actors in such communities to understand if there is some hierarchy. Finally, we address the multiplex analysis by understanding the role of layer authorities and well-connected members in the network.

3.6.4.1 Community structure

Community structure is a common characteristic shown by most complex networks, which allows us to discuss common properties among their members. We analyse the existence or not of community structures in the most important participatory forums, as a function of members interactions. Each boxplot in the Figure 3.53 summarises the nodes composition of communities detected in our layers when the 2 different detection techniques are applied (subsection 3.6.2.3). Members of a layer which do not belong to any community are not represented.

We find that it is possible to divide all layers into several disjoint communities, where 75% of them have between 2 and 147 users. The communities median size detected in the communication layer represents less than 1% of the users, while in the case of the development and users mailing lists it represents 14.4% and 7.11%, respectively, of the users.

In the communication layer 74.38% of the links are internal – between two members of the same community – and only 25.62% of the interactions are between members of two different communities, as shows the Figure 3.54, resulting in a two-tier structure with users geographically close showing high participation among them. Participants in the mailing lists, however, show no preference in answering to members of their community or members of other communities.

The analysis of the participatory layers using the clique percolation method reveals another community structure, enclosed by a core group of members. While the core group in the mailing list is formed by a small portion of users – from 17.28% in the users layer to 9.97% in the development layer, the core group of members 86.24% of the participants in the communication layer with 86.24% of the participants represents almost the entire network.

In contrast, the social layers show less ties among members of their communities than with other participants in the layer, as shown in Figure 3.54, where communities are detected using cliques. Communities are connected only through bridges, suggesting that both social layers face a two-tier
3.6. Analysis of the social effort in multiplex participatory networks

3.6. Self-management

Figure 3.53: Number of nodes by community. Comm., Dev. and Users refer to the communication, dev-list and users-list layers.

Figure 3.54: Links distribution between communities. Bridges are links between two participants, both members of the same community, but one of them member of another community, too. Others are links to nodes which do not belong to any community.

structure with several cores coordinated by some of the members. Finally, it is interesting to observe a stronger connectivity in the users mailing lists' communities, which suggests a large gap of involvement between core community members and the other members of the participatory forum.

3.6.4.2 Individual participation and social value

We have seen so far that in the participatory forums of Guifi.net, 72% of the members generate more than 91.2% of the contributions. Nevertheless, the social value of a community member for the whole network depends more on its connectivity and position towards the rest of the participants than on his or her individual contributions. For instance, if somebody sends a lot of messages to a single person, it does not imply that he or she is generating any social value.

We measured the individual impact of the users inside the network as their **closeness centrality, HITS**
3.6. Analysis of the social effort in multiplex participatory networks

**Figure 3.55:** Empirical Cumulative Distribution Function (ECDF) of members’ centrality scores.

**authorities and hubs.** Closeness centrality for a connected graph is defined as the inverse of the average distance to all other nodes. In our case, the distance is measured as the sum of the weights of the nodes managed for the communication layer and of the messages answered in the mailing lists layers of each link it has to traverse.

HITS [82] is a ranking algorithm used in the past to exploit the web’s hyperlink structure. As a result, we obtain two measures for each node, the authority and hubs. The first one is a measure of individuals as a source of information, while the second one ranks higher those nodes with high quality out-links. In order to apply the HITS algorithm directly to our mailing lists graphs, we transform them to undirected graphs.

Figure 3.55 shows as we expected the same distribution for both HITS values in the communication layer due to the fact that the graph is undirected. Indeed, the social graphs present small differences between the authorities and hubs distributions. It is important to note that the authorities distribution shows that only a small portion of the users are of high value because their messages generate more replies from others.

Individuals’ analysis revealed that there are only two users in common on all layers among the 10 higher ranked members, which were identified as the network founders. There are also two members in common in the mailing list which do not appear as top ranked in the communication layer. It highlights the higher affinity between members of the social participation layers compared with the communication layer.

**3.6.4.3 Global impact of participation**

In the previous section we discussed the social value generated by members in their own participatory layer. In this section, we turn our interest to understand their impact on the whole network by studying what happens when top layer authorities decide to leave the network. To this end, we firstly ordered the members of each participatory layer by the authorities score. Then, we proceeded recursively removing the top ranked node of the remaining ones and measuring the size of the largest component on each other layer in the multiplex.

Figure 3.56 shows the robustness of each participatory layer when the authority members are recur-
3.6. Analysis of the social effort in multiplex participatory networks

3.6.1 Self-management

Figure 3.56: Multiplex layers robustness.

sively removed in the multiplex. Ideally, when an authority is removed, we expect that the size of each layer to be reduced only by one member – the authority itself. However, if the authority removed is essential to maintain the connectivity, we expect some other members getting disconnected – and the size of the layer’s biggest component reduced.

In examining the multiplex robustness in case of the disappearance of members from any layer, we observe a huge impact on the dev-list layer, and a moderate impact on the communication layer. However, the graph structure of the users-list layer makes it indifferent to these removals, even of their authorities. Thus, we can argue that the communication and dev-list layers show a higher dependence on the particular contributors.

3.6.5 Conclusions and discussion

3.6.5.1 Summary of contributions

This work makes several contributions to the analysis of member participation in a community network. Using the historical data from users’ activities in the participatory forums, we explained why the communities’ structure and multiplex analysis are relevant to measure members’ participation, by showing that our target network is in a mature state. Therefore, our findings could be applicable to other community networks in the same life-cycle stage.

We show that the analysed community structures indicate the existence of a hierarchical structure in the governance of the on-line participatory layers. This structure is leaded by a small set of members who only have contact with the rest of the subscribers through a few bridges. These bridge members are ranked highly as the most valuable members of the participatory layers. The asymmetry in participation found in the community network makes these members unique and essential for the communication network survival. Furthermore, the communication layer is structured in a weak way as an effect of the demographic distribution of their members.

3.6.5.2 Limitation of the analysis

Studying participation in the context of building new devices is relevant for measuring the effort dedicated to increase and improve the communication network, and as we have observed in this work, not all users dedicate the same effort to this task. However, there exist other ways to improve or to help building the physical network that were not captured by our model, because they require a very deep understanding of users’ choices, and which could be more community network specific.
In general, the analysis of the social participatory forums allows detecting communities and users’ roles. In our work, we conducted the study using the two most important mailing lists in Guifi.net network. However, many other social interactions occur through other ways, like the general face-to-face assembly of the Guifi.net foundation, sporadic physical encounters among members, or region specific mailing lists. Including the data from other mailing lists would lead us to understand in more detail the heterogeneity of the network and to detect more relevant communities in such regions. Nevertheless, with our current analysis we were able to detect clusters of users, whose actions and connections have a high impact on the whole community network.

3.6.5.3 Implications of the analysis

Most previous works have addressed the resource sharing or service regulation problem in various scenarios and measure the users’ participation with physical resources. In a cooperative complex ecosystem like community networks, the effort dedicated to maintain and improve the network must be taken into account, too.

The social value analysis is a first step to measure the effort dedicated by community network members through their contributions. However, it does not capture the utility of members’ contributions. A more precise measure should include the interest of the members in participating with users from different clusters or communities, avoiding the rise of closed communities.

Contributory systems designers can take advantage of the results in order to design their regulation mechanisms, and adapt the effort evaluation of each member to the participatory layer robustness.

3.7 Socio-Economic qualitative analysis of community networks

The goal of this work is to present the diverse social, economic and legal challenges Community Networks (CNs) face, from their perspective, and identify implemented solutions that could be generalized or common obstacles. The information was collected through an online questionnaire. To include the CNs perspective, we designed a two-phase participatory process with the objective to allow CN members not only to answer a set of predefined questions but also decide which questions are important and should be answered based on their experience. At the same this served as a dialogue platform, where the members of each CN had the chance to propose topics and questions that members from other CNs could answer with their experiences or share their common concerns.

This work focuses on designing an open process to collect information, allowing each participating CN to shape up the topics addressed and then contribute details for a selection of them. We find it a good trade-off between the cost of participation in amount of time required per participant, the duration of the whole process, and the openness to allow shaping the topics of discussion. A total of 18 CNs from all over the world (4 different continents) participated to the process. In this section we present our findings from this work.

There are common enabling factors such as open wireless spectrum that has spurred a large market for inexpensive Wi-Fi devices. The Libre/Open source community has created many components and tools to lower the complexity barrier of creating new CNs. Despite the natural emergence of many diverse projects, there’s also a trend to consolidate developments into larger and more integrated software environments for CNs. This work complements other studies from the technological perspective [7].
CNs have proven to explore and find out diverse sustainable ways so that local communities can provide their own network connectivity to bring the benefits of the digital society to underserved areas. This collective community effort has multiple local effects in creating opportunities and expectations for local development through the definition of open rules for participation in developing the infrastructure and its services. CNs create opportunities for local businesses and local entrepreneurs, contribute to ensure and advance in the socio-economic development of the communities and even attract complementary market-driven services once the community reaches a more developed stage. However, the economic aspects are critical to sustainable growth and to the stability of CNs, essentially ensuring that costs and contributions are balanced every time, for every agent involved in the process in a fully transparent manner, and according to the expected external practices of any other organization in its social, economic and legal context. That is really challenging for organizations that typically grow in a very decentralized and organic manner, being redefined and formalized as they grow, and with so many expectations for so many diverse local participants.

As far as social issues are concerned, many of them remain unresolved. This is not necessarily problematic, since the environment and the motivations of each CN are very diverse. There exist, though, common local issues, like licensing, or challenges that need global initiatives where knowledge sharing and cooperation (as already done in some cases) can be very useful.

Organizational issues and approaching correctly the local community seem very challenging, but since CNs are social communities where social processes are taking place, the social challenges are obstacles that every healthy social community is facing. The governance of the community infrastructure, the rules of participation in the construction, decision making and the access to the connectivity services have local difference in all communities, but there is a common trait of an open infrastructure that is organized as a common-pool resource [114] to avoid congestion of the resource [67] with more or less formal rules and different levels of organizational complexity.

Legislation can be a serious obstacle for CNs, since they have no means to directly affect legislators. Peer pressuring and raising awareness are the only possible approaches, which in many cases are not effective enough. Moreover, legislation differs significantly between different locations and in many cases is very tricky to deal with. These issues prevent the formulation of a unique global strategy. Nevertheless, mature CNs have accumulated a lot of knowledge. Documentation and dissemination of this knowledge can create a knowledge base, which will be quite diverse, but certainly helpful for new communities.

Technology is evolving, quickly lowering the cost of optical communications, increasing the speed of Wi-Fi networks, and opening new spectrum opportunities. However, the socio-economic challenges are the most critical. Community-driven network infrastructures develop when there’s already a strong local community sense, and a leading team of local champion takes the lead. The environmental factors are also critical: such as the regulation for communications, conditions on spectrum usage and licensing, privacy and data retention, access to locations for deployments, competition for spectrum from mobile network operators. Different local environments result in different choices of organization, growth, community licenses, cooperation with local businesses that this paper describes. Several open problems have been identified by the CNs that the research, open source developers and entrepreneur communities can tackle. We can expect that mapping the scenario will help to better identify challenges, define requirements more precisely and, thanks to experimental testbeds such as Community-Lab, test the technical developments in realistic conditions or, for socio-economic challenges, share challenges, experiences and lessons.

In any case, initiatives learn from each other and exchange not just best practices but also software tools, knowledge and even people visiting each other or in inter-community meetings such as the
Self-management

3.8 Improvement of Community Network Software

One focus of the CONFINE project is to improve software tools of Community Mesh Networks. Fraunhofer FKIE has improved and extended the OLSRv2 implementation of Olsr.org to provide Community Mesh networks with a better access to the second generation OLSR protocol – OLSRv2.

3.8.1 Standard Implementation

The IETF MANET group is still working on additional standards extending the OLSRv2 protocol [139]. Within CONFINE, many of the extensions have been implemented into the OLSRd2 code base of Olsr.org [110].

The olsrd2 implementation has been used to test the Directional Airtime Metric [65] which the Fraunhofer FKIE submitted to the IETF for standardization as described in deliverable D5.6. The evaluation of the metric can be found in 3.5.

The metric code has been designed with a strong focus on self-configuration to make the deployment as easy as possible.

OLSRd2 also implements the OLSRv2 multi-topology draft [26] which allows to run multiple metric implementations simultaneously within the same protocol instance. This allows Community Networks to test different routing metrics in a single network without deploying a second instance of the protocol daemon.

The DLEP-radio and DLEP-router [129] implementations can now be used also without the OLSRd2 routing code. This simplifies the integration of OLSRd2 into DLEP-capable radios. The custom data types implemented in the Olsr.org DLEP implementation have been turned into an IETF standard compliant extension that can be switched on and off by negotiation between DLEP-radio and -router.

The IETF MANET group has released RFC7182, a generic security extension [138] for the RFC5444 packet format [36] used in OLSRv2. Within the project, we integrated an optional subsystem, the signature mechanism defined in RFC7182 with plugable support for multiple crypto libraries. In addition, we integrated experimental plugins that use this mechanism to provide a shared-key based security solution for Community Networks.

3.8.2 OpenWRT Integration

OpenWRT [113] is the primary Linux distribution used by Community Networks on embedded routers. A complete revision of the OpenWRT startup scripts and configuration handling has been realized within the project.

The Olsr.org code is now part of the official OpenWRT ’routing’ repository, which provides access to OLSRd2 and DLEP to every user of OpenWRT for OpenWRT Attitude Adjustment and higher versions.

OLSRd2 can now directly read the OpenWRT UCI configuration file format without conversion, which improves the integration into the OpenWRT configuration process.

The routing daemon also hooks into the OpenWRT hotplug mechanism to provide better handling of interface changes.
3.8.3 Modularization and Documentation

The Olsr.org Network Framework codebase has been restructured to provide a higher degree of modularization. This allows multiple protocols within the framework repository (e.g. DLEP and OLSRd2). This obsoletes the need for having multiple repositories. The existence of multiple repositories had been a confusing issue for multiple people working with the framework in the past.

The whole Olsr.org Network Framework has been documented in an open wiki [110] to simplify the usage by new users. Additional documentation about deployment examples and troubleshooting are now available to provide users with additional resources for setting up mesh networks without asking for direct help on the corresponding mailing lists.

3.8.4 Acknowledgements

Fraunhofer FKIE would like to acknowledge Bastian Bittorf from Freifunk Weimar [56] for the help on the OpenWRT integration of OLSRd2 and DLEP.
4 Conclusions

In this deliverable we presented results of the experimental research performed mainly on CONFINE testbeds during the fourth year of this project. The following studies have been described in this deliverable:

- An empirical study that gives a better understanding on how (un)stable community networks behave. By analysing the amount of update and withdrawal messages sent on those networks and comparing this to the amount of BGP messages on the Internet, we have determined that, as expected, community networks are indeed less stable than the Internet.

- A large-scale measurement campaign, where we specifically analyse the network performance as experienced by the end-user in community networks in comparison to standard ISPs networks. The results from our measurements, from a user perspective, show promising network performance in general, however with high variability over time and over different nodes.

- A link quality prediction by means of a time series analysis. We have demonstrated that it is possible to accurately predict the Link Quality in 98% of the instances.

- Based on that work, we have demonstrated that time series analysis is a promising approach to accurately predict end-to-end quality values in community networks. All of the four learning algorithms described achieved high percentages of success, with average Mean Absolute Error values per link between 2.4% and 5% when predicting the next value of the end-to-end quality.

- A study on mechanisms to increase the robustness of network connectivity by detecting forwarding faults. We have divided the problem into its different subproblems, and analyzed different traffic summary functions, traffic validation mechanisms and proposed a distributed detection protocol. In summary, we have proposed a set of solutions that could be easily implemented for wireless community networks as an independent monitoring daemon, without the need of modifying the current wireless community network stack.

- We have studied the scalability, performance, and stability of three proactive mesh routing protocols: OLSR, BMX6 and Babel, three common routing protocols in wireless community networks. Our emulation and testbed-based experiments with various network conditions at different scales have provided several detailed results that compare the three protocols.

- The performance of the BMX6 routing protocol has been experimentally evaluated. In particular we focused on the capacity of BMX6 Vector Metric to select the route that can achieve the highest throughput and we verified that the combination of metric and protocol internals used by BMX6 is very efficient in selecting a path that is very close to the optimal one. To achieve this goal we performed experiments on the QMPSU network.

- We made a performance comparison between the two multicast routing protocols: ODMRP and SMF. For the evaluation, we used a two-step approach. First, we performed emulation-based measurements in a virtualized testbed utilizing the ns-3 IEEE 802.11 model. Then, we did real-world experiments with an actual IEEE 802.11 ad-hoc testbed in order to compare the results and increase the overall credibility of our analysis. By means of this real-world testbed measurements, we partially confirmed the observations made in the first step. However, we also observed that SMF did not perform as well in our physical testbed as it did in the emulation.
4. Conclusions

- We made another performance comparison between OLSR here called OLSRv1 (with the Expected Transmission Count (ETX) metric) and OLSRv2 (with our novel Directional Airtime (DAT) metric). In our measurements performed using a physical testbed, we achieve a more stable route selection process and improved throughput in comparison to the OLSR-based approach currently used in Community Networks.

- We have measured the level of user involvement in a wireless community network by applying social mining techniques to understand the differences observed among users of several participatory forums. We have shown that the analysed community structures indicate the existence of a hierarchical structure in the governance of the on-line participatory layers.

- We have designed an open process to collect information, allowing each participating Community Network to shape up the topics addressed and then contribute details for a selection of them. Total of 18 CNs from all over the world (4 different continents) participated to the process.

- Finally, Fraunhofer FKIE has improved and extended the OLSRv2 implementation of Olsr.org to provide Community Mesh networks with a better access to the second generation OLSR protocol – OLSRv2.
Bibliography


Bibliography


