

# An evaluation of BMX6 for Community Wireless Networks

Axel Neumann  
Pangea.org NGO  
Barcelona, Spain  
neumann@cgws.de

Ester López  
Computer Architecture Department  
Universitat Politècnica de Catalunya  
Barcelona, Spain  
esterl@ac.upc.edu

Leandro Navarro  
Computer Architecture Department  
Universitat Politècnica de Catalunya  
Barcelona, Spain  
leandro@ac.upc.edu

**Abstract**—Nowadays, a growing number of communities of citizens build, operate and own open IP-based community wireless networks with thousands of low capacity nodes actively participating in routing the data traffic. This article focuses on one of their concerns, routing and its scalability, by presenting BatMan-eXperimental Version 6 (BMX6) and evaluating its performance. BMX6 is a low overhead and scalable mesh network routing protocol inspired by human networks. Its performance is evaluated in comparison with OLSR in terms of overhead and convergence time as networks grow in number of nodes and diameter. The results show that the convergence time and protocol overhead per node in BMX6 is not significantly affected by the addition of new nodes in contrast with OLSR, where both parameters can grow super-linearly. This confirms the excellent scalability of BMX6.

## I. INTRODUCTION

The key role of Internet access as a fundamental infrastructure for individual and collective social participation in society coupled with the widespread availability of WiFi enabled devices, unlicensed spectrum and open-source software has enabled the growth of Community Wireless Networks. In this model communities of citizens can build, operate and own open IP-based networks. These networks grow organically and they are becoming very large, with thousands of nodes (e.g. Freifunk, FunkFeuer, AWMN, Guifi.net). Mesh networking [1] is a promising model where each node must not only receive and send its own data, but also serve as a relay for other nodes, helping to propagate data across the network with no need for manual reconfiguration as the network evolves. As a result messages propagate along a path, from node to node until the destination is reached. Ensuring the delivery of data to its intended destination is a huge challenge in these networks. The routing function must allow for continuous connections and reconfiguration around new, broken or congested paths, using self-adaptive algorithms and ensure its scalability as the network grows.

The contribution of this paper is the analysis and evaluation of the scalability of BMX6, an open-source mesh-routing protocol, in a network that grows organically with the addition of new nodes.

Previous work on mesh routing protocols does not differentiate between naturally dynamic (e.g. constantly changing route metrics) and rather static topology information (like the

IP configuration of nodes) nor differentiate between topology information that has only local relevance and other with global relevance. None of the previous approaches take advantage of optimizable direct communication between neighbouring nodes, which are usually few from the perspective of a single node compared to all nodes in the network. In contrast, BMX6 separates static from dynamic information in a node description which must be re-propagated only occasionally. Each BMX6 node keeps detailed information only about the relevant nodes and links in its neighbourhood and minimises the amount of protocol traffic by establishing a common understanding between neighbours (compact local node identifiers).

The evaluation of BMX6 is done through experimentation based on emulation over a virtualized network that replicates the topology of selected Guifi.net zones, one of the largest Community Wireless Networks, with around 17,000 nodes overall and around 100 nodes in the Barcelona core network graph. The analysis is based on a comparison with the well established OLSR protocol implementation `olsrd` version 0.6.2 from `olsr.org`, which is also available as open source.

The rest of this paper is structured as follows: Section II presents related work where OLSR is compared with other routing protocols; Section III describes BMX6 and OLSR mesh routing protocols. In Section IV we describe the methodology followed to evaluate both algorithms. Section V demonstrates our claims about BMX6 through an experimental evaluation based on emulation over a virtualized network running BMX6 and OLSR, and points out the lessons learnt and future work. Finally we summarise our contributions in Section VI.

## II. RELATED WORK

There have been several studies about the performance of different routing protocols in wireless mesh networks.

Johnsen in [2] compares the performance in terms of overhead, throughput, CPU and memory consumption of OLSR and BMXd by performing measurements on a real hardware using a 49-node indoor grid testbed.

Focusing on OLSR, [3] and [4] compare OLSR with AODV, DSDV and DSR in terms of routing overhead, average delay and throughput; those papers study the protocols performance by means of simulation of a grid like topology and considering

a portion of mobile nodes. [5] and [6] evaluate OLSR, DSDV, DSR and AODV on a real testbed; however, the number of nodes of the testbed is low (8) or even not mentioned, and scalability is not analysed.

As far as we are aware, none of the former work have investigated the performance consequences for mesh routing protocols when switching from IPv4 to IPv6.

In contrast, our work presents a novel evaluation based on data of already deployed networks (Guifi.net), and therefore, considering more realistic network topologies with a larger number of nodes and in a IPv6 world. It also introduces BMX6, which even after being used in community networks, has never been studied and analysed by the research community.

### III. ROUTING PROTOCOLS

There are many manet routing protocols exploring a combination of different features [8] such as performance metrics beyond hop-count, cross-layer designs taking metrics from layer-2, scalability for large networks, robustness to mitigate service disruption due to link failures or congestion, etc.

Here, we describe BMX6, proposed and developed by Axel Neumann, in contrast with the Optimized Link State Routing Protocol (OLSR), based on a link-state algorithm. The scalability evaluation of BMX6, in terms of performance (convergence time) and cost (volume of protocol messages), also takes OLSR as a baseline.

#### A. BMX6

BatMan-eXperimental version 6 (BMX6) [10] is the successor of the BMX daemon (BMXd) which emerged as an independent branch from the BATMAN routing protocol [11] to explore and test new approaches for routing and context dissemination in mesh networks. The design and development of this new version was driven by the objective to better cope with the increased address space given by IPv6 addresses, enable node-individual configurations while clarifying the handling of conflicting node announcements (e.g. duplicate address allocations), and allow efficient state dissemination (thus reduced protocol overhead) through the strict distinction between local and global as well as static and dynamic state. BMX6 as well as BMXd are actively used in current mesh networking communities and projects such as Guifi.net (qMp [12] and Graciasensefils.net [14]), Freifunk [15], and Lugro-mesh [13].

BMX6 is a table-driven routing protocol for wireless mesh networks. As any table-driven routing protocol, its goal is to compose a path from source to destination by deciding on each node which will be the next hop. BMX6 is a distance-vector protocol, since the information each node manages is a list of tuples of nodes' identifiers and the cost of getting there when choosing a concrete link:  $\langle \text{destination node, next hop, cost} \rangle$ . The novelty in BMX6 is the dissemination mechanism it uses to propagate this information. The dissemination protocol is inspired by human social networks that are scalable as people tend to learn more about its neighbourhood and abstract and

filter out information about others. Topology knowledge in a node is optimised for itself and its neighbours by using local compact identifiers for a local compressed stateful dialogue.

This concept can be considered as a technique of stateful compression applied to a distributed system, which differentiates two different states for the routing protocol: (i) transient and (ii) steady. During the transient phase, neighbours exchange knowledge about their environment: nodes' descriptions, links, etc. and provide information about their internal identifiers (IID), which identify nodes in a compact way. With this information, each node sets up a dictionary table per neighbour that translates its IID values to the globally unique and non-ambiguous hashes of the full node description. On the steady state, each node has a local information state in the form of IID-to-hash dictionaries; and a global information state as hash-to-description dictionary. During this phase the protocol just exchanges small packets to keep track on the variation of link metrics and to monitor network changes. Thanks to the information state deployed during the transient phase, the fields of this periodically exchanged routing updates, which are usually given by a 128 bit IPv6 address, can be substituted by the much shorter IID value (16 bits), and thus it results in compressed messages. The separation in local and global state also pays off when a node moves, and therefore, its neighbourhood changes, because it only needs to re-establishment of IID-to-hash relations, whereas already existing knowledge about pairs between hashes and corresponding descriptions is still valid.

As a result, the control overhead will increase when there is a network change, stabilizing afterwards to a lower value. This can be seen in Figure 1. Figure 1 illustrates the total traffic in the network generated by OLSR and BMX6 when booting a network of 60 nodes at time 0; then 180 seconds later, a new node connects to the network. As we can see, regarding BMX6, there is an considerable peak at the very beginning, result from the exchange of nodes descriptions and local IID tables; later on, when the new node connects, it imposes a new peak of traffic, but clearly smaller than the first one, since there is less information to be exchanged.

It needs to be considered, that a typical situation would be small changes in the network, like connecting or disconnecting a node from the network, while simultaneous booting of all the nodes in the network is not common. However, a similar effect can be expected in case that two separate mesh clouds become one single network by the deployment of a new link that interconnects them. On this case, we can expect a high peak of traffic, since every node on the network needs to learn about all the nodes in the other cloud.

Consequently, there are two different types of messages on BMX6 depending on their nature: (i) periodic messages, that are periodically generated by the protocol on every node; and (ii) occasional messages, that are exchanged only when necessary because of a change in the network.

The periodic messages generated by BMX6 are responsible for the little overhead during the steady phase, and they are:

- Hello advertisement (HELLO\_ADV) messages are broad-

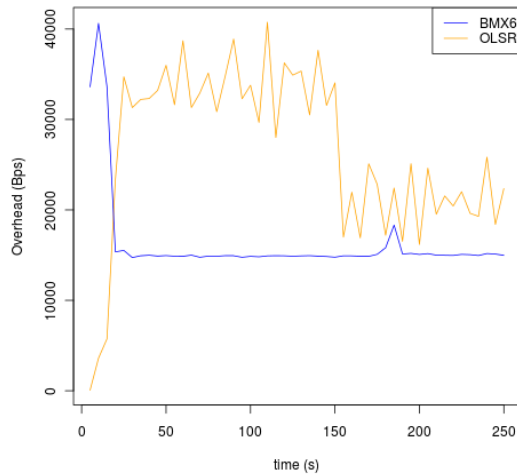


Fig. 1: Network overhead versus time on a 60 nodes network

casted every `HELLO_INTERVAL`, which by default is 0.5 seconds. They are used to measure the link quality (based on the number of received messages) and to know whether a link is alive or not.

- Similarly, report advertisement (`RP_ADV`) messages are periodically broadcasted as response to the `HELLO_ADV` messages, and therefore every `HELLO_INTERVAL`. They provide a summary of the received and lost hello messages from all neighbours and related links.
- `OGM_ADV`s or OriGinator Messages are sent every `OGM_INTERVAL` (which by default is 5 seconds) and propagated over the network. They are used to let nodes become aware of other nodes further than just one hop away and inform about the path metric to the originating node. However `OGM_ADV`s are not flooded indiscriminately through the network, but just through so called relevant links. A link is relevant whenever it is necessary to reach one of the nodes in the network, i.e. it is the next hop of at least one entry in the routing table.

In contrast, the occasional messages create a peak of traffic when there is a change on the network, allowing nodes to gain knowledge about their neighbourhood or learn about the full description of a formerly unknown node. These messages are:

- Link advertisement (`LINK_ADV`) and optional device advertisement (`DEV_ADV`) messages are broadcasted on demand (due to the reception of `LINK_REQ` or `DEV_REQ` messages) to describe the existence and further attributes of network devices and links from the perspective of an individual node. Each `LINK_ADV` message represents a link as perceived (due to previous received `HELLO_ADV`s) by the transmitting node to one of each neighbours. The order in which `LINK_ADV` messages are aggregated is further used as an implicit reference to a specific link of the node when creating or processing `RP_ADV`s messages.

- Description advertisement (`DESC_ADV`) messages are exchanged between nodes, providing a full description of a specific node, containing details such as their IP addresses, hostname, and protocol parameters. Description messages are requested via `DESC_REQ`s messages due to the receipt of an unknown description hash.
- A hash advertisement (`HASH_ADV`) message provides the relation of a node-specific IID value to the hash of a specific node description that is used for globally non-ambiguous node identification. By means of description's hashes BMX6 refers to already known nodes without having to send the full description of the node. `HASH_ADV` messages are requested whenever an unknown IID reference or a message from an unknown node is received.

In summary, BMX6 achieves to reduce its overhead by using two different mechanisms: first, it optimises the traffic transmitted periodically through the network by means of establishing a common understanding between neighbours using compact IIDs and description hashes; secondly, it controls the flooding of messages by analysing whether a link is relevant or not, and omits non-relevant links on the flooding of OGMs.

## B. OLSR

OLSR, as specified in RFC [16], is a proactive routing protocol that uses an optimised version of a pure link-state protocol. It is optimised in terms of overhead, since topology control messages are not purely flooded through the network, but selectively by the MultiPoint Relays (MPR). MPRs are selected in a distributed fashion, so each node selects a small set of immediate neighbours to be its set of MPR, which satisfy that every 2-hop away neighbour can be reached through one of the nodes on the MPR set.

However, its wide usage in existing mesh networks has shown that the MPR based optimisation is inefficient when faced with the dynamic changes and poor links that occur in real-life and self managed deployments. To overcome this, the MPR algorithm is disabled in the currently most used OLSR implementation from `olsr.org`. Instead, the OLSR fish-eye extension is activated by default to reduce the average protocol traffic overhead.

Currently, most existing community-mesh networks are using this implementation for the whole network (e.g. Freifunk [15], Funkfeuer [20]) or in parts of the network (e.g. Guifi.net [18], AWMN [19]). Since its first larger deployments in community networks in 2003, the code has constantly improved and become a very stable, mature, and future rich solution for small and large-scale mesh projects.

OLSR has been described, analysed, and discussed extensively during previous work [16], [9], [7]. In the following we are just briefly reviewing the most important principles and messages of the OLSR implementation used for our evaluations and how they relate to protocol traffic overhead and convergence time.

OLSR periodically broadcasts two types of messages:

- `HELLO` messages are broadcasted every 2 seconds by default by every node and only travel one hop. `HELLO`

messages mainly contain the sender's IP, a list of its neighbours, and the link status. They are used to calculate the link qualities between nodes.

- Topology Control (TC) messages are flooded through all the network. In case of disabled MPR algorithm, these messages are originated by all nodes (otherwise TC messages are flooded selectively by the nodes that are selected as MPRs). TC messages have an originator address and a list of its neighbours with corresponding link qualities. TC messages are processed by each node to internally calculate the full topology graph of the network which provides the basis for calculating the best next hop to any given destination.

Like any link-state routing protocol, OLSR is conceptually vulnerable to routing loops resulting from non-synchronised topology graphs as calculated by different nodes on the forwarding path of a data packet. A trade off to this problem is given by flooding TC messages at a smaller interval, allowing nodes to recalculate their topology view more often at the cost of increased protocol traffic overhead and CPU load. The fish-eye extension for OLSR implements a third way to mitigate the problem. It is based on the finding that routing loops usually occur between nearby nodes (thus nodes at one or two hop distance). To achieve better synchronisation of topology graphs between nearby nodes while allowing less frequent synchronisation between distant nodes, TC messages are flooded with different TTL values. Specifically, the sequence of TTLs with active Fish-eye extension in the OLSR implementation used for our evaluations is 2,8,2,16,2,8,2,255. This means that only every even TC message is flooded beyond its two-hop neighbourhood.

Figure 1 shows the traffic overhead of OLSR under the same conditions as BMX6. As we can see, since the activation of the fish-eye extension is delayed, there is a transient state with higher traffic for the first 140 seconds, which reduces afterwards when the fish-eye extension becomes active. This delayed activation is the default behaviour of the used OLSR implementation with the objective to speed up the convergence time after the booting procedure of a new node.

#### IV. EXPERIMENTAL ENVIRONMENT

In this section, we describe our evaluation approach. Our experiments are using current implementations of BMX6 and OLSR and execute them in an emulated network environment. The two routing protocols are deployed on a virtualized mesh infrastructure using virtual machines (Linux Containers) interconnected following a real world topology from Guifi.net. A series of experiments are performed with graphs from 10 up to 90 nodes, measuring metrics related to performance (convergence time) and cost (protocol overhead).

##### A. The virtualized mesh infrastructure

Mesh Linux Containers (MLC) [21] is a collection of scripts based on Linux Containers (LXC) and Linux networking tools. MLC's goal is to provide the necessary tools and scripts to

quickly create emulated network topologies including link-specific packet loss and delay with up to hundreds of nodes.

Using MLC on a single testbed machine (a 2.4GHz Pentium I5 with 4 cores and 4GB of RAM) we could emulate up to 200 nodes running either OLSR or BMX6 in its default configuration and connected in a grid-like topology with perfect links before the system got overloaded. With 200 nodes the CPU load (as measured with top) exceeded 75 percent only during the protocol startup phases.

MLC creates a base LXC container with all the necessary software that will be run on the testbed. Afterwards, this container is replicated with the proper routing configuration files and network configuration depending on the desired amount of emulated nodes,

On the network side, each container has 3 interfaces which connect with the other containers through 3 different bridges. The first bridge is intended for controlling the containers, while the other two can be used for experimentation. MLC allows the creation of virtual links between the containers on the last two interfaces by controlling the forwarding probability and delay at link level. Within these restrictions it allows the definition of any target network topology. For emulating different link characteristics MLC defines different levels of link qualities in terms of delay and error probability and depending on unicast or broadcast traffic. For the evaluation of this work, we used only one experimentation interface per node and all emulated links were configured with no loss and no delay.

##### B. The test networks

The evaluation is based on the topology of Guifi.net, one of the largest Community Wireless Networks, with around 17,000 nodes. The topological information is extracted from the Guifi.net CNML table<sup>1</sup>, XML-formatted data that represents Guifi.net network. This table describes all the nodes with their interfaces and IP addresses, as well as information about the links in the network. However, this information might be slightly inconsistent due to unplanned links, or nodes losing visibility after the initial deployment, etc. Nevertheless, it does provide a fair approximation to how Guifi.net network looks like.

On the evaluation of a routing protocol's performance there is a set of network characteristics we consider the most relevant:

- 1) Overall network size (number of nodes)
- 2) Network diameter (maximum number of hops between most distant nodes)
- 3) Node density (number of links per nodes)
- 4) Amount of announced interfaces and Host/Network Announcements (HNAs) per node
- 5) Link quality

This evaluation focuses on the first two: network size and diameter, restricting the network density to the one defined on the CNML. Because CNML data describing the growth of the

<sup>1</sup>Available online at <http://guifi.net/en/guifi/cnml/{zone}>

selected Guifi.net zones over the past years was not available, we have chosen a randomised approach to create topologies of smaller size and diameter based on a single recent CNML table<sup>2</sup>. Links have been emulated as perfect for the sake of simplicity, and the impact of announcements per node has been neglected during this evaluation where each node is only announcing the IPv6 address of a single interface.

As final consideration, we have only considered the nodes belonging to the core network, since the other nodes will typically be configured with a default gateway to access the network and will not run a routing daemon. In this case, Guifi.net point-to-point links belonging to the core network are assigned an IP address on the network range 172.25.0.0/16, so after retrieving the network topology from the CNML, it has been reduced to consider only the links attached to interfaces with an IP address on the mentioned range. Since our goal is to analyse how the routing protocols react to the size of the network, once we obtained the core-graph for several CNML zones, it has been sampled to generate realistic networks of different sizes in the following way:

- 1) One node is chosen randomly from the initial graph and added to the result graph.
- 2) Among all the neighbours of this node, another one is randomly chosen and added to the result graph.
- 3) This process is repeated, choosing one node randomly among the neighbours of the previously selected nodes until the network has the desired size.
- 4) All the existing links among the selected nodes are also added to the resulting graph.

Concretely, 100 different networks with sizes ranging from 10 to 90 nodes have been extracted following the mechanism explained above from the following CNML zones: Alt Empordà, Alt Penedès, Anoia, Bages, Barcelona, Castelló and Osona. Each of these networks has been simulated 20 times as explained in the following Section and the results present the averaged values for each considered variable (nodes or hops). An example of the extracted network topologies is given by Figure 2 showing the Barcelona case with 60 nodes.

### C. Evaluation metrics

Protocol overhead and the convergence time are the evaluation metrics used to compare the two routing protocols as networks vary in number of nodes.

- *Protocol overhead*: The overhead of a routing protocol is the quantity of control traffic required to be sent through the network in order to work properly. To measure the protocol overhead each network has been emulated during a period of 10 minutes while the traffic generated by the protocol of all the network was being captured. In order to reflect the overhead of the steady state (which follows the startup period with a transient increased overhead as illustrated in Figure 1 and 3) only the protocol data, captured 200 seconds after the last node has been started, is taken into account. Furthermore, only a period of 100

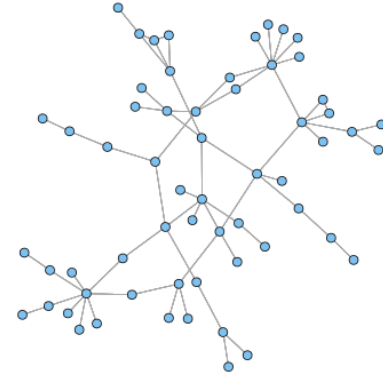


Fig. 2: Topology of a network extracted from Barcelona CNML with 60 nodes

seconds is considered for calculating average values. The selected period is illustrated with the filled areas in Figure 3.

- *Convergence time*: The convergence time measures the time it takes to the network to be aware of a change in it. Therefore, it depends not only on the network characteristics and routing protocol, but also on which is the change in question and the definition of awareness. In this paper, the change proposed is the connection of a new node to the network; whereas awareness implies reachability between all the nodes on the network. Concretely, we measure the convergence time as the time it takes to the farthest away node to reply a ping issued by the new node.

### D. The emulation process

In this section the process to evaluate the routing protocols on each test network is explained.

First, each node on the network is started as an LXC container with a single interface and it is assigned an IP address. Then, all the links of the graph are set up. The next step is to start the routing daemons. We consider that starting on every run the daemons immediately one after another might skew the results because periodic messages happen at the same time on every node. Therefore, we have decided to start the routing daemon after waiting a random interval since the previous one was started. The waiting interval is between 0 and 10 seconds, which is big enough to ensure independence between runs. As consequence, the transient state extends longer, as can be seen in Figure 3, and the initial peaks of the routing protocols are smoother. The overhead measurement's time interval goes from 200 seconds after the last node of the network starts its routing daemons and lasts for 100 seconds, which ensures that the measurement is done during the steady

<sup>2</sup>Retrieved the 05-06-2012.

phase. The overhead is measured on the MLC bridge interface, so it contains the messages generated by every node.

The next phase measures the convergence time. First, the new node, N, is connected to the network to a random node, A. To measure the convergence time, node N starts 2 ping requests: one to node A and another to node B, which is the furthest away node in the network. Then the convergence time is measured as the difference in time of each reply (Figure 4

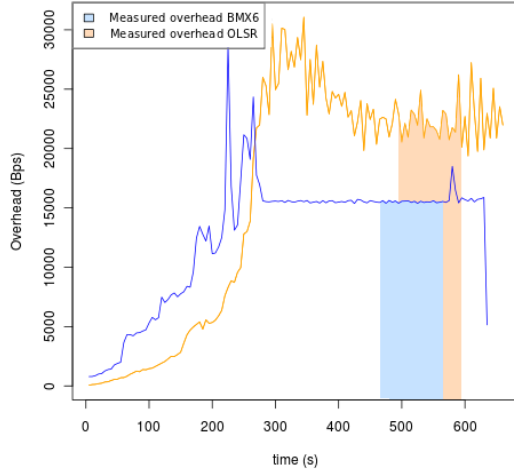


Fig. 3: Overhead when introducing delays on the start time of the routing daemons

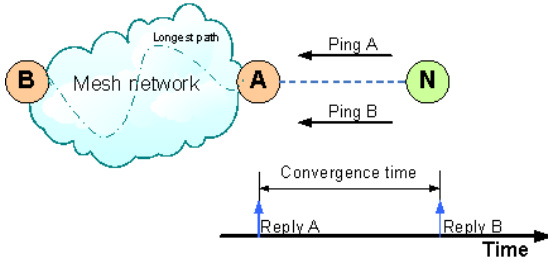


Fig. 4: How convergence time is measured

Used routing-protocols, as shown in table I, were parametrised with default values as predefined by its implementation.

TABLE I: Implementations used for evaluation

| protocol | project                                       | version   | git revision | date     |
|----------|---|-----------|--------------|----------|
| BMX6     | <a href="http://bmx6.net">http://bmx6.net</a> | 0.1-alpha | fcba5ac9     | Apr 2012 |
| OLSR     | <a href="http://olsr.org">http://olsr.org</a> | 0.6.2     | d14c696b     | Oct 2011 |

## V. EXPERIMENTAL RESULTS

As mentioned before, the evaluation of protocols performance is measured through their overhead and convergence time. The protocols' overhead in bytes per second is shown in Figure 5, where dots represent the value of a run and the

lines represent the average value for all the networks with the same size. As we can see BMX6's overhead grows slower than OLSR, obtaining better results for networks with more than 40 nodes.

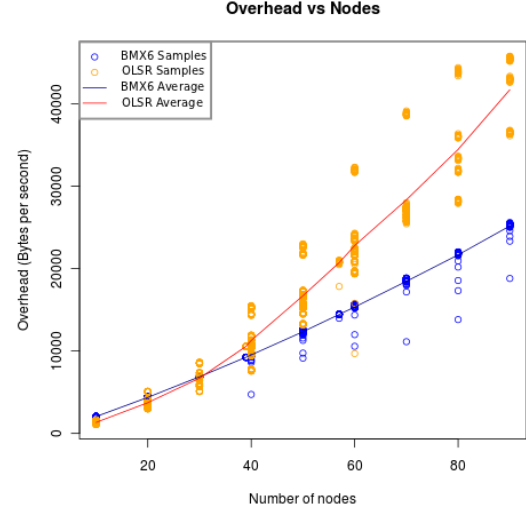


Fig. 5: Bytes per second depending on the number of network nodes

The growth rate of each routing protocol can be seen more clearly on Figure 6, which shows the overhead normalised by the number of nodes on the network. As we can see both algorithms grow super-linearly with the number of nodes, but BMX6 grows way slower than OLSR.

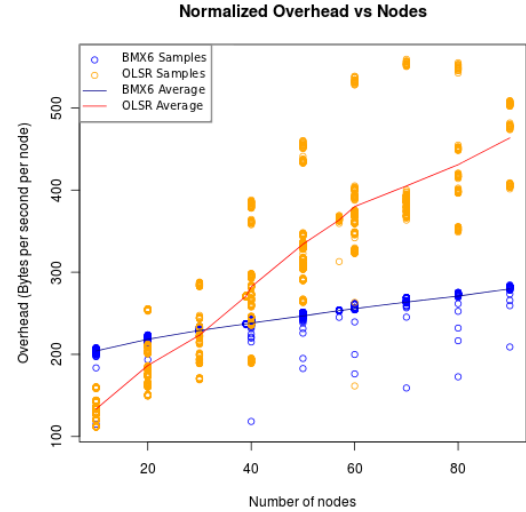


Fig. 6: Protocol overhead per node (Bytes/sec) depending on the number of network nodes

Regarding the convergence time, as can be seen on Figures 7 and 8, it remains practically constant for BMX6, while it increases with the number of hops for OLSR. Concretely, we can see in figure 7 that the average convergence time for



BMX6 stays below 10 seconds, while it grows from almost 20 seconds to 40 seconds for OLSR. This is caused by the fact that BMX6 immediately triggers the emission of Description Advertisement messages when a new node appears, since it is reasonable to consider that if a node is new for someone, it will also be new for its neighbours. For OLSR, on the contrary, the new node is notified on TC messages, which have to wait longer to be sent. The effects of the fish-eye extension can be clearly seen on the step from 8 to 9 hops, since only 2 out of 8 TC messages travel farther away than 8 nodes. As one could intuitively guess, convergence time after the addition of one random node is quite stable and nearly unaffected by the total number of nodes in the network graph, as can be seen in Figure 8.

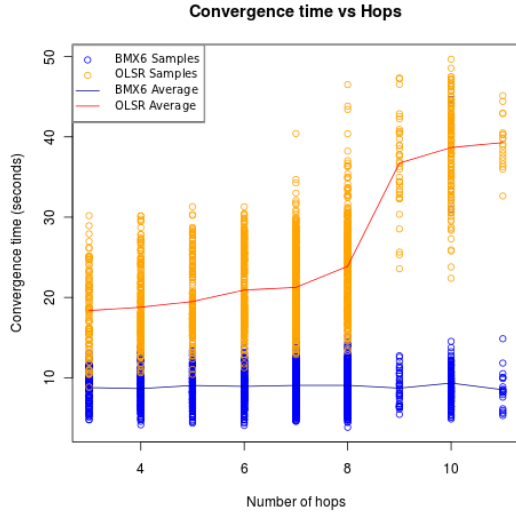


Fig. 7: Convergence time with different hops between A and B

These results confirm the scalability of BMX6 with respect to protocol overhead and convergence time is rather independent of the size of the network in terms of number of nodes or diameter of the graph. In contrast to OLSR with both parameters growing with the size of the network.

These results confirm the effectiveness of the design of BMX6 in reducing the protocol overhead and in the scalability of the mechanisms to deal with changes and growth of the network.

We are aware that the performance of routing protocols in typical mesh deployments may depend significantly on much more network characteristics than have been addressed in this work. In future work we want to extend our experimentation environment to also consider different link quality, the amount of interfaces per node that are involved in mesh routing, and the amount of network announcements per node. We are also planning to validate our findings by comparing our emulation results with measurements obtained from real-world deployments. Therefore, similar experiments shall be performed on testbeds as Community-Lab, being developed by

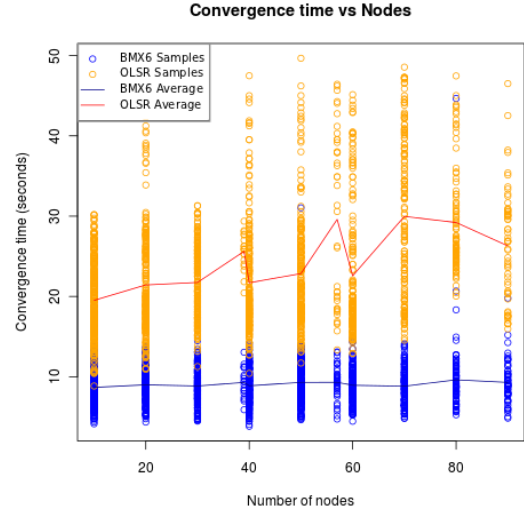


Fig. 8: Convergence time with different number of nodes on the network

the CONFINE project [22], which aims to provide a testbed embedded in production community networks.

## VI. CONCLUSION

Community wireless networks have an organic growth, with an already large number of nodes with no centralised or readily manual control. These networks can benefit from manet routing protocols, that self-adapt to network changes, to determine a path for the end-to-end delivery of messages across the network. BMX6 solves this challenging problem in a decentralised and scalable manner with a protocol overhead per node and a convergence time that is only marginally affected by the addition of a new nodes and the size of the network. In contrast, OLSR suffers from scalability issues for both metrics.

Looking at the scalability and the availability of an open-source implementation of BMX6, we believe that this mesh routing protocol solves the key challenges for routing in Community Wireless Networks and large wireless mesh networks.

## ACKNOWLEDGEMENT

This work is supported by the European Community Framework Programme 7 within the Future Internet Research and Experimentation Initiative (FIRE), Community Networks Testbed for the Future Internet (CONFINE), contract FP7-288535. Support is also provided by the Universitat Politècnica de Catalunya BarcelonaTECH and the Spanish Government through the Delfin project TIN2010-20140-C03-01.

## REFERENCES

- [1] Bruno, R.; Conti, M.; Gregori, E.; , "Mesh networks: commodity multihop ad hoc networks," Communications Magazine, IEEE , vol.43, no.3, pp. 123- 131, March 2005
- [2] D. Johnson, N. Ntlatlapa and C. Aichele, Simple pragmatic approach to mesh routing using BATMAN, 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa, 6-7 October 2008

- [3] Zakrzewska, A.; Koszalka, L.; Pozniak-Koszalka, I.; , "Performance Study of Routing Protocols for Wireless Mesh Networks," Systems Engineering, 2008. ICSENG '08. 19th International Conference on , vol., no., pp.331-336, 19-21 Aug. 2008
- [4] Ashraf, U.; Juanole, G.; Abdellatif, S.; , "Evaluating Routing Protocols for the Wireless Mesh Backbone", Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on , pp.40, 8-10 Oct. 2007
- [5] Abolhasan, M.; Hagelstein, B.; Wang, J.C.-P.; , "Real-world performance of current proactive multi-hop mesh protocols," Communications, 2009. APCC 2009. 15th Asia-Pacific Conference on , vol., no., pp.44-47, 8-10 Oct. 2009
- [6] Murray, D., Dixon, M.W. and Koziniec, T. (2010) An experimental comparison of routing protocols in multi hop ad hoc networks. In: Australasian Telecommunication Networks and Applications Conference, ATNAC, 31 October - 3 November, Auckland, New Zealand.
- [7] Kuppusamy, P.; Thirunavukkarasu, K.; Kalaavathi, B.; , "A study and comparison of OLSR, AODV and TORA routing protocols in ad hoc networks," Electronics Computer Technology (ICECT), 2011 3rd International Conference on , vol.5, no., pp.143-147, 8-10 April 2011
- [8] Ian F. Akyildiz, Xudong Wang, and Weilin Wang, "A Survey on Wireless Mesh Networks", Computer Networks and ISDN Systems archive, Volume 47 Issue 4, 15 March 2005
- [9] Cédric Adjih, Emmanuel Baccelli, Thomas Heide Clausen, Philippe Jacquet, and Georgios Rodolakis, "Fish Eye OLSR Scaling Properties", IEEE Journal of Communications and Networks, 2004.
- [10] BMX6 mesh networking protocol, <http://bmx6.net>
- [11] Axel Neumann, Corinna Aichele, Marek Lindner, Simon Wunderlich, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)", IETF Draft, October 2008.
- [12] QMP – quick Mesh project, <http://qmp.cat/>
- [13] Lugro Mesh, <http://www.lugro-mesh.org.at/en/>
- [14] Comunitat GraciaSensefils.net, <http://graciasensefils.net>
- [15] Freifunk, <http://start.freifunk.net/>
- [16] Thomas Clausen and Philippe Jacquet, "Optimized Link State Routing Protocol (OLSR)", IETF RFC 3626, October 2003.
- [17] OLSRd - an adhoc wireless mesh routing daemon, <http://olsr.org>
- [18] Guifi.net, <http://www.guifi.net/>
- [19] AWMN - Athens Wireless Metropolitan Network, <http://funkfeuer.at/>
- [20] FunkFeuer, <http://funkfeuer.at/>
- [21] Axel Neumann, Investigating Routing-Protocol Characteristics with Mesh Linux Containers (MLC), Workshop, UPC, Barcelona, Spain November 2011, [Online], Available: <https://raw.githubusercontent.com/axn/mlc/master/MeshLinuxContainers-x07.pdf>
- [22] CONFINe Project: Community Networks testbed for Future Internet, <http://confine-project.eu/>