# Receiver-driven routing for community mesh networks

Axel Neumann, Leandro Navarro
*Department of Computer Architecture*
*Universitat Politècnica de Catalunya*
*Barcelona, Spain*
*neumann@cgws.de, leandro@ac.upc.edu*

Roger Baig, Pau Escrich
*Fundació Privada per a la Xarxa Oberta,*
*Lliure i Neutral Guifi.net*
*Catalonia, Spain*
*{roger.baig, pau.escrich}@guifi.net*

***Abstract*—Community wireless mesh networks are decentralized and cooperative structures with participation rules that define their freedom, openness and neutrality. The operation of these networks require routing algorithms that may impose additional unnecessary technical restrictions in the determination of routes that can restrict the freedom of community users. We propose a receiver-driven discretionary routing mechanism where each receiver (the intended destination of the packet) can freely specify delivery objectives and remain compatible with the collaborative approach of community networks. Each node has a unique identifier and can announce the description of its offer and also the description of its routing policy with preferences to deliver traffic to it. BMX6 provides a "hash-based profile propagation mechanism" to disseminate descriptions. This receiver-driven routing can be applied to express preferences for desirable nodes and paths, or to restrict traffic to trusted nodes enabling trust and security aware routing. We validate our contributions with a proof of concept implementation of key concepts, as an extension of the BMX6 routing protocol, that confirms its feasibility and scalability.**

***Keywords*-routing; community mesh networks; metric policies; trust; security;**

## I. Introduction

Community wireless mesh networks are decentralized and cooperative structures thats grow organically in communities where network nodes are owned, contributed and administered by its members.

Routing in community wireless mesh networks is based on the principle of cooperation among the participants. Athough these communities usually have participation rules as a membership license or peering agreement that define their freedom, openness and neutrality. The implementation of these networks require additional technical agreements and restrictions for data transit among any pair of network nodes. That includes the use of a specific routing protocol where nodes can learn and inform about the state of the network and update their own routing tables.

Current mesh routing protocols [10], including OLSR [14], BATMAN [12] are based on running a common algorithm and sharing common metrics to determine the routes for all network nodes. However, the participants in the network and its diverse infrastructure should not impose technical restrictions limiting the freedom of the community users beyond the accepted participation rules defined by each community network agreement. The main challenge is to provide routing mechanisms that can respect and enhance community cooperation and individual freedom.

In this work we present a receiver-driven discretionary routing mechanism where the destination can freely specify delivery objectives while remaining compatible with the collaborative approach of community networks. Our proposal is based on the assumption that the owner of a packet is the intended receiver (destination) of it (not the sender), and that it should be able to control the policies for the delivery of the packets meant for it. To achieve this, we propose means for each node to announce the resources its willing to share and the implications to use them. Further, we let nodes (from a destination perspective) to select among the resources announced by others and specify the rules (policies) according to which nodes will handle its traffic.

Receiver-driven routing can be used by receivers to express preference for nodes and paths with desired characteristics. A receiver of interactive traffic may prefer to route traffic preferably through nodes with a track record of low latency or high availability. Another key application is enabling trust-driven routing. A receiver may want to get data only through trusted nodes or avoid unstrusted nodes that may affect his data such as for non-encrypted traffic.

We have experimented with the ideas as part of BMX6 [13], where each node can have a descriptive profile, and a global unique and non-ambiguous hash as identifier. BMX6 provides a "hash-based profile propagation mechanism" to disseminate profile information between nodes of a community network. Nodes can use this information to construct and announce its routing policy to any sender. Building on this our receiver-driven routing mechanism has been tested.

The rest of the paper presents the design of receiver-driven routing. We validate our contributions with a proof of concept implementation of the main idea.

## II. Receiver-Driven Routing

### A. Assumptions and Design Considerations

Routing in community networks is based on the principle of cooperation. To conceptually maximize the opportunities (in terms of network resources) for cooperation despite the usually diverse infrastructure and user heterogeneity in community networks, a principal cooperation should not depend on specific mandatory rules or policies.

Therefore, instead of enforcing the same behavior to all components of a community network by employing specific prevention mechanisms, the participation rules should be open and only demand a formal (and authenticated) announcement of offered resources and related implications. Users should be empowered to dynamically select the set of published resources and acceptable implications as needed for their individual objectives.

Practically, any infrastructure owner willing to (at least) partially share his resources should be able to identify himself and specify his offer and the implications or limitations that he associates with this offer in a node-specific profile (also called description in the following). Such a profile may include the following "offer" attributes:

- Routing topology and reachability information.
- Supported metric functions (e.g. ETX, ETT, hop-count).
- Locally reachable networks, offered gateway and proxy services.
- Traffic shaping implications for packets routed via this node.
- Identity information (eg: this node is owned by *X*, you may not trust me).
- Responsibility information (this node is managed by *Y* who certified its operation according to policy *Z*).

Being aware of the profiles of all nodes in a network, each community network user (the routing protocol in its node) could select among the available resources those he wants to accept for the forwarding of his own traffic (by adding forwarding requirements to his own profile). A list of typical requirements for a forwarding policy could be:

- End-to-end path establishment (route selection) according to its choice of metric function.
- Only via nodes that do not apply traffic shaping.
- Only using gateway services of node *A* and *B*.
- Only route my traffic via nodes owned or managed by *C*.

Following this approach an enhanced level of trust and security can be achieved simply by allowing nodes to waive some offered resources for the forwarding of their data (e.g. due to being untrusted). Maybe the resulting end-to-end paths are suboptimal or even not possible (compared to using all offered links and nodes of the network). However, we believe this is similar to real-life security: individually I may decide not to cross dark forests at night and sometimes I must accept a long way around or even that I can not reach the guy living in the middle of a forest.

From our experience with the guifi.net[26] community network we can report specific examples illustrating the advantages for community networks. For instance we are faced organizations reluctant to share their already existing network infrastructure and join the community simply because they have no means to control that their own traffic is not routed via unknown nodes (in particular the nodes of their competitors). Offering such companies the opportunity to exclude untrusted nodes as candidate nodes for relaying their traffic would indeed relieve the situation and create a win-win situation for the companies and the network community.

### B. Principles and Proposed Mechanism

Receiver-driven routing is targeted at enabling user-individual routing with respect to available network resources and each user's forwarding requirements. Our proposal is based on the assumption that the owner of a packet is the intended receiver of it (*not* the sender nor any intermediate node) and therefore the receiver should be able to control how the packet shall be forwarded. Further, we argue that this assumption naturally fits into today's IP architecture where the packet forwarding is typically destination driven as each node along the path is consulting it's routing table only for the destination address of a packet to identify the next hop. Thus, letting the destination node control how it's next hop in the routing tables should be selected already allows for a destination-specific directing of all traffic destinated to this node. In contrary, pursuing to support sender-individual (sender-driven) routing, given the destination-driven operation of routing-tables, would require either strict source routing or maintaining individual routing tables (policy routing) for all potential source addresses and in all nodes of a network, introducing performance penalties and new coordination challenges.

We propose a (i) "receiver-driven routing mechanism" that leverages a (ii) "hash-based description propagation mechanism" to disseminate requirements and profile information between nodes of a community network.

To enable the distributed application of node-individual forwarding policies we associate routing update messages with a specification of the forwarding requirements of destination nodes. All other nodes are requested to use this specification when identifying the next hop and configure its routing table entry towards this destination address accordingly. A consequence of this approach is that if intermediate nodes are following the forwarding policies defined by each destination node, then the traffic directed to any destination node will only be forwarded according to its own rules. Further, if the destination node is able to specify a set of trustable nodes and requests intermediate nodes to only forward its data via these trusted nodes, then, once a data packet is routed via any of these trusted nodes it will never leave the set of trusted nodes along its path until it is delivered to its final destination or dropped because an end-to-end path is not possible within the set of trusted nodes.

The algorithm supporting this approach may operate as follows:

- Require for each node:
  - A public and private key pair. The hash of the public key is used for non-ambiguous identification of the node.
  - A description (profile) of itself including its public key, offered resources and implications, and forwarding requirements (see above), signed with its private key. These descriptions can be unambiguously referenced using the hash of the overall description (*not* the public key hash).
- Each node description and the corresponding signatures are propagated over the network.
- Like any Destination Sequenced Distant Vector Protocol, each node periodically propagates routing update messages containing a sequence number, metric values, and (instead of a destination address) a reference to the description of the originator node (description hash).
- When node *A* receives a routing update originated by node *D* (and propagated) via neighbor *N* then:
  - *A* looks up the referenced description from *D* and checks whether *N* is in the set of nodes that *D* trusts and that *N* fulfils the forwarding requirements defined by *D*. Note that *A* itself must not necessarily be a trusted node of *D*. This allows even untrusted nodes to send packets to *D* as long as they can forward the packet via one of their neighboring nodes that is trusted by *D*.
  - If the condition is not met, then the routing update is silently dropped.
  - If the condition is met, *N* is a potential next hop for forwarding traffic towards *D*, and then:
    *A* uses the metric function defined via *D*'s description to calculate the current metric value to reach *D*, uses this value for ranking between alternative next hops towards *D*, and for the further propagation of *D*'s routing updates.
    *A* configures the routing table towards the addresses defined in *D*'s profile via the best ranking neighbor.

### *C. Technical Protocol Requirements*

Based on the above presented mechanism a number of protocol requirements can be identified. In the following we summarize these requirements and point to existing concepts and solutions that can be used for the implementation and validation of our ideas.

Although our proposed mechanisms may also be applicable (with conceptual adaptions) to link-state based routing protocols we believe that a (proactive) table-driven destination-sequenced distance-vector based protocol will fit best to allow an efficient and loop-free operation of our proposal. The main argument here is that for link-state based protocols our approach would require a continuous (re-)calculation of the complete end-to-end path between all nodes of a network and this for each node's (forwarding) policy. However, for distance-vector based protocols the calculation must only be done for the next hop towards each originator and each with respect to the policy specification of the originator.

An information propagation mechanism for disseminating node profiles in the network has to satisfy the following requirements:

- *The mechanism must be capable to deal with so called opportunistic networks [9]:* important to ensure that node profile information becomes available as soon as possible and despite the existence of unstable and temporary broken links or even disconnected fractions of the network because the availability of this information is a prerequisite for other nodes to instantly resolve and process node identities, attributes, and routing updates.
- *Independent propagation of profile information and routing updates:* needed since routing updates rely on previous local availability of the originator's profile.
- *Referencing:* to allow for an efficient (in terms of traffic and computational overhead) association of frequent and periodic routing updates with the originator's profile information it must be possible to non-ambiguously reference originator profiles with comparatively compact profile identifiers (e.g. the hash of a profile).
- *Dynamic profile updates and continuous synchronization of this information with other nodes:* required to, on the one hand, let nodes change their resource offers and implication announcements and, on the other hand, let nodes adapt their forwarding-policy specification to either changed resource availabilities or new traffic, security, or trust requirements.
- *Originator identification, authentication, information integrity, and non-repudiation:* needed by receiving nodes to validate the authenticity and integrity of profile information originated by other nodes.
- *Versioning:* profile information and referencing must allow receivers to distinguish old from new versions from the same originator. This is needed to ensure that processing of routing update messages can always be associated with the latest valid profile of originators.
- *Listing of nodes:* to support the traffic forwarding only via trusted nodes the routing protocol must allow dynamic white/black listing of particular intermediate nodes that could be used or must be ignored during the selection of potential next-hop nodes for the routing. Furthermore, the routing protocol must be capable to create and process a syntax for specifying and communicating those trusted nodes.
- *Scalable verification of authenticity and credibility:* since the users (node administrators) of a large community network can hardly know and maintain com-

prehensive black or white lists, classifying all nodes of the network into trustable or not, mechanisms are desirable to let users rely on already existing chains or networks of trust. A natural candidate for this problem would be given by what is known as a *web of trust*.
- *The routing protocol must allow the dynamic (re-) configuration of metric functions and parametrizations:* although existing routing protocol implementations exist that allow the usage of different metric functions (for example the OLSR implementation of OLSR.org [20] allows this via metric plugins and Babel [11] comes with separate implementations for different metric flavors), to the best of our knowledge none of the existing open routing protocol implementations are currently able to change this behavior on the fly. However, this is needed to dynamically apply node-individual metric specifications without restarting the network. Further, the routing protocol must be capable to create and process a syntax for specifying metric functions and parametrizations as needed to express and communicate the desired route-selection behavior to other nodes.
- *Mechanisms for secure communication (authentication, integrity, confidentiality, non-repudiation) with neighboring (potential next hops) nodes:* necessary to ensure that protocol data is indeed exchanged between authentic neighboring nodes and that to-be forwarded data is forwarded via the intended neighbor. To support this, existing security solutions like IPSEC [21] can be used.
- *Suite of cryptographic tools for calculation of security attributes:* a variety of open libraries exist for solving the required cryptographic mechanisms using standardized APIs. We are focusing on the openSSL [22] API which is also supported by CyaSSL [23], a lightweight and efficient SSL library for embedded devices.

## III. Validation and Preliminary Results

As a proof of concept, key mechanisms of our proposal have been implemented based on existing work of the BMX6 routing protocol and validated by emulation of typical scenarios. In the following we quickly summarize the requirements already provided by the BMX6 routing protocol and the functional extensions that have been developed. Following the description of the methodology and tested scenario we summarize performance and scalability observations in terms of computation and communication overhead.

### A. Leveraged BMX6 mechanisms

Our current implementation builds on top of the BMX6 routing protocol [13] because its concepts and implementation [24] already leverage and provide key mechanisms for the requirements as discussed in Section II-C. In particular, it is a proactive, table-driven routing protocol for mesh networks, leveraging concepts of destination-sequenced, distance-vector protocols. Further, the current implementation is fully open-source (GPL licensed), and actively maintained and used (for example in various Guifi.net [26] community-network clouds and fully integrated in community network node-system distributions like qMp [25]). BMX6 also provides an opportunistic-network tolerant information propagation mechanisms operating independently of the routing-update mechanism and allowing efficient referencing of descriptions via hashes and individual identifiers, continuous and dynamic description updates, and versioning. Further, the mechanism allows to piggyback arbitrary additional content via description extensions and propagate them with the rest of the node's profile in a network. This feature has already been used by third-party applications to disseminate detailed topology information for visualization purposes, service announcements, and instant messaging data. In the scope of this proposal, this feature is used for the propagation of white/black lists of trustable nodes, security attributes (public keys, description signatures, trust chains), metric-functions, and policy specifications.

### B. Extensions toward Receiver/Preference Driven Routing

Our current extensions allow the concurrent application of different metric functions (including hop-count and expected transmit count (ETX) [17], a rudimentary expected transmit time (ETT) metric [16], and transmit quality (TQ), a multiplicative and link-asymmetry aware routing metric proposed by the B.A.T.M.A.N. [18] protocol. Because the TQ metric is used in our later validation it is briefly described here. The TQ metric function considers greater metric values as better than smaller ones and it's value decreases each time the containing routing update message propagates from one node to another. Each node calculates the end-to-end path metric value to a given destination by multiplying the metric value (as contained in the received routing update) with the link-transmit quality (being a value between 0 and 1) that represents the measured success rate for transmitting hello packets via the link via which the routing update message was received. If the link success rate is 1 then only the smallest possible value is substracted from the metric value.

To support node-individual metric functions we have defined a metric description syntax and added description extensions for the metrics plugin of BMX6 in a way that a destination node can select between several base metric functions and customize them with general and base-function specific parameters. For example it is possible to define the sliding window size for best next-hop ranking when averaging the metric values received via recent routing updates (originator messages in BMX6 terminology). A dynamic description update including the current metrics specification is triggered whenever the configuration is changed and instantly processed and applied by receiving nodes when selecting the best hop neighbor for a particular destination node.

### C. Validation via proof-of-concept

To validate the expected behavior of our implementation different networking scenarios have been tested (as described below) using the network emulation environment [15]. MLC facilitates Linux Containers (LXC) to instantiate (depending on host CPU and memory resources up to hundreds of) virtual nodes, each with an isolated networking environment and employs Linux networking tools (like ip, tq, tc, ebtables, brctl) to create virtual links between node interfaces, including per link and link-direction specific packet loss.

In order to validate the distributed application of node-individual metric functions, a purely hop-count based metric function was compared with the multiplicative and link-asymmetry aware TQ metric function. Resulting forwarding paths between nodes specifying different metric functions for the routing of their traffic have been compared using the ping record route option. To stimulate the functions for selecting different routes, a simple ring topology with a total of 5 nodes (e.g.: *A-B-C-D-E-A*) but highly asymmetric link characteristics between neighboring nodes has been set up. The link asymmetries were configured so that no loss occurs for packets transmitted from any node to its right neighbor but 20% of packet loss occurs in the opposite direction (to its left neighbor). Initially, all nodes were configured with the hop-count metric and as expected, any resulting path traversed at most two hops. After dynamically reconfiguring node *E* for selecting the TQ metric function all traffic meant for node *E* was routed clockwise along the ring topology. In particular, a ping record route from *A* to *A*'s left-hand neighboring node *E* showed that the packet was routed along *A-B-C-D-E* while a ping from *A* to *D* continued to traverse (in the opposite direction) along *A-E-D*.

In summary, the tested scenario proves the key idea for receiver-driven routing, being the feasibility to let destination nodes specify and achieve the application of their specific forwarding policies. As future work we will extend our implementation with the propagation and handling of white/black lists of trustable nodes and security attributes.

### D. Performance and scalability

The discussion of performance impacts is divided into typical operational phases of a routing protocol, namely the discovery phase (propagation and processing of node descriptions), the maintenance phase (propagation and processing of routing updates), and the forwarding phase (forwarding data according to the current state of the routing table).

Unless otherwise noted, all measurements were performed using the MLC emulation environment as introduced in Section III-C. However, instead of a ring-topology with asymmetric links, a grid-like torus topology with *N=XxY* nodes and perfect links (no packet loss) was configured with each node having one interface and four neighbors. The topology had *X=10* nodes per row and *Y* nodes per column

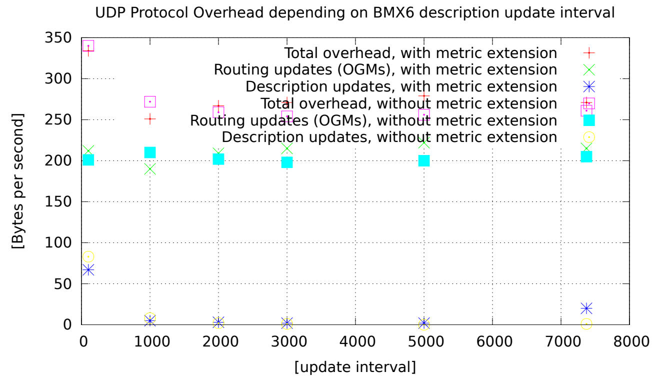


Figure 1. Protocol traffic overhead with and without our metric extensions for different description update intervals.

with *Y* being in the range [20,30,...200] to obtain the desired network size. Protocol-traffic overhead was captured using the traffic-statistics tool provided by BMX6 which allows to account for different message types, incoming, and outgoing traffic. CPU and memory consumption of BMX6 processes was captured using the unix *top* utility.

During the discovery phase, no significant impact could be observed for the computational overhead caused by the processing of our description extensions and only a marginal impact could be measured for the related protocol-traffic overhead due to its extended size for propagating the to-be-used metric functions and parametrizations. In fact our metric extensions increases the existing description by only 16 bytes which must be compared to the overall description size ranging (depending on amount of announced networks and other attributes and services) between 110 and several hundreds of bytes. Figure 1 shows the total, OGM, and description update related protocol traffic overhead with and without our metric extensions and for different description update intervals. Each measurement point shows a single measurement averaged over 100 seconds. It should be noted that small intervals (e.g. 1000 seconds and less) are very unlikely as a description update is only triggered due to the reconfiguration of a node. From the figure it could be seen that the additional overhead introduced by our metric extensions can be considered marginal.

For the outstanding validation of verifying authenticity and integrity of description updates, we indeed expect a significant performance hit. However, some preliminary tests using the Linux OpenWRT embedded OS and the OpenSSL Speedtest on a typical device like the ubiquity RouterStation (AR7161 - MIPS 24K CPU running at 680MHz) showed the feasibility of calculating a SHA1 hash for up to 8192 size blocks in less than 0.3 ms and signing/verifying data with 264 versus 3116 bps using a 512 bit RSA key pair (and signing/verifying with 50 versus 1040 bps using 1024 bit RSA key). Extrapolating these measurements for signing a BMX6 description of up to 1K means that a node (based on this hardware and *not* using any hardware-based crypto acceleration) would need less than 1.7 seconds for hashing

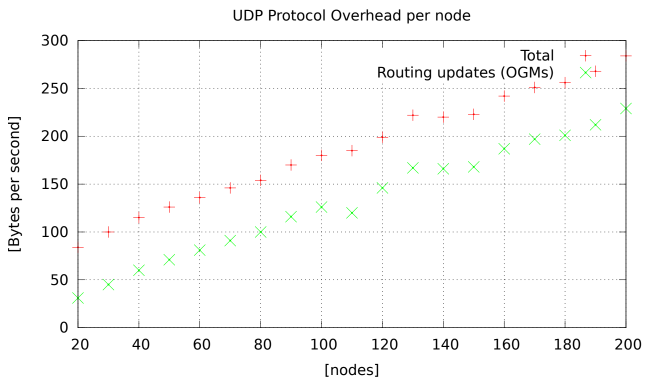


Figure 2. Total and OriGinator Message (OGM) related overhead versus number of nodes.

its own description and RSA512-based signing of the resulting 160 bit SHA1 hash (required only during startup and reconfiguration scenarios) and would require "only" 50 ms for hashing and verifying other nodes descriptions (which is required only for casual description update of other nodes).

Related to the maintenance phase, no impact was expected nor observed for the traffic overhead because none of the periodic messages needed for monitoring (probing) and for propagating link and end-to-end path variations are affected. In particular the routing update messages remain unchanged and are only processed differently. The impact, caused by the modified processing of destination-specific metric functions and differing only in using a function pointer instead of a hardcoded function, again results in non-measurable performance penalties. In fact, the only (still-hardly) measurable impact could be assigned to the overhead caused by specific functions simply because in our implementation different metric algorithms require different amounts of normalizations, multiplications, divisions, or even square-root calculations. However, this effect would be the same when supporting only a unique and hard-coded function.

It should be noted that the marginal impact of the maintenance phase can be considered most relevant for the overall overhead and scalability of a routing protocol because in large networks, the great majority of traffic and related processing is caused by the continuous propagation and processing of periodic routing updates. To justify this, we have measured the total and OGM-related overhead of BMX6 depending on the number of nodes in a network. Each measurement point, as shown in Figure 2, represents the UDP overhead averaged over 100 seconds during the maintenance phase of the protocol. It could be seen that OGM-related overhead increases with roughly the same gradient as the total protocol traffic overhead.

However, a slightly increased performance penalty is expected for the requirement of secure communication of protocol data with neighboring nodes. But again, this overhead will be caused by the Linux kernel and the employed IPSEC implementation since the validation, calculation, and configuration of security attributes for the IPSEC stack will be done as an outcome of the discovery phase. Recent hardware measurements published in the OpenWRT wiki [28] report that reasonable IPSEC-encrypted throughput performance can be achieved with current embedded devices. For example using the AES256 cipher, encryption at up to 37Mbps is possible using a (AR7161) MIPS 24K CPU running at 680MHz. This is is enough to secure the comparatively small protocol traffic load between neighboring nodes (less than 3Kbps for a 200-nodes network) and even reasonable to encrypt user data.

For the forwarding phase no additional overhead was expected nor measured. This is because the protocol only configures the kernel's routing table (as the outcome of the maintenance phase) and relies on the lookup of routing table entries for each packet as performed by the Linux kernel.

## IV. Related work

Related to reliability (security) and responsibility: [1] provides interesting numbers on the awareness and counter measurements on routing security in the internet. [2] provides a survey on security issues in mesh networks presenting specific approaches to address secure/reliable routing in mesh networks. This work is aligned with these.

Related to discretionary routing and heterogeneous objectives, very little attention has been devoted to the simultaneous support of heterogeneous (routing) objectives in collaborative networking.

The pico-peering agreement [19] provides a minimum baseline template for defining an abstract interoperability agreement between owners of individual network nodes but leaves technical solutions to achieve an efficient multi-hop networking completely open. The guifi.net licence [26] is more specific in terms of expected conduct but often softens the main statement with exceptions (eg Section 3.2. "Whenever possible, the general criteria must be"). Also it demands that "network management criteria must be published" but leaves open how protocols could automatically consider the published criteria of specific nodes for routing.

In terms of technical solutions, [3] presents a routing-protocol coordinator that allows nodes to communicate with each other even though they are equipped with heterogeneous routing protocols. [4] approaches the interoperability problem in WMNs by proposing a cross-layer heterogeneous routing protocol.

To some extent, current routing protocol implementations have solved specific issues of the dilemma. For example OLSR [14], BMX6 [13] have integrated support to let each mesh-nodes select an individual gateway node to the internet. The community projects [25], [27] enabled multi-stack routing, using address-range separation and policy routing to allow simultaneous usage of different routing protocols.

To the best of our knowledge, no work has been published yet that aims to combine the two directions of reliability and discretionary routing policies into a coherent framework.

## V. Conclusion

We have presented a receiver-driven discretionary routing mechanism for community networks where each receiver can freely specify delivery objectives and remain compatible with the collaborative aim of community networks. This routing mechanism can be applied to express preferences for desirable nodes and paths, or to restrict traffic to trusted nodes enabling trust and security aware routing. A proof-of-concept implementation of key concepts, developed as an extension of the BMX6 routing protocol, confirms its feasibility and scalability.

In future work we plan to develop a complete implementation of the routing mechanism over BMX6, including syntax, authentication, signature and engine for policy specifications, definition of the bootstrapping procedure. In particular we are interested in further exploring the characteristics and limits of the protocol over larger and more realistic scenarios with the support of the Community-Lab [29] testbed for a detailed evaluation.

## Acknowledgment

The authors would like to thank Ester López and the whole qMp.cat team for their invaluable contributions in terms of advice, motivation, discussion, deploying, debugging, reviewing, and evaluating this work. This research is supported by the European Commission in the CONFINE [29] project.

## References

[1] Security Agency (ENISA), GNKS, NLnet Labs, July 01, 2010, *State-of-the-art Deployment and Impact on Network Resilience*.

[2] Muhammad S. Siddiqui and Choong Seon Hong, *Security Issues in Wireless Mesh Networks*, IEEE International Conference on Multimedia and Ubiquitous Engineering (MUE'07), 2007.

[3] N. Kang, S. Yoo, Y. Kim, S. Jung, and K. Hong, *Heterogeneous Routing Protocol Coordinator for Mobile Ad Hoc Networks*, in Proc. UCS, 2006, pp.384-397.

[4] Shih-Hao Shen and Yueh-Min Huang and Jen-Wen Ding, *A cross-layer design for heterogeneous routing in wireless mesh networks*, 2008 Int. J. Pervasive Computing and Communications.

[5] Parul Tomar, Prof. P.K. Suri, Dr. M. K. Soni, *A Comparative Study for Secure Routing in MANET*, International Journal of Computer Applications 4(5):17-22, July 2010.

[6] Wireless Mobile Network Security, Bing Wu, Jianmin Chen, Jie Wu, Mihaela Cardei, *A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks (Eds.) pp. c 2006 Springer Chapter 12*

[7] Loay Abusalah, Ashfaq Khokhar, and Mohsen Guizani, *A Survey of Secure Mobile Ad Hoc Routing Protocols*, IEEE Communications Surveys and Tutorials 10(1-4): 78-93 (2008).

[8] Siddhartha Gupte, Mukesh Singhal, *Secure routing in mobile wireless ad hoc networks* Proceedings of the 5th WSEAS international conference on Applied computer science (ACOS'06), USA, 996-1002.

[9] Chung-Ming Huang, Kun-chan Lan and Chang-Zhou Tsai, *A Survey of Opportunistic Networks*, 22nd International Conference on Advanced Information Networking and Applications - Workshops, 2008.

[10] 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.

[11] J. Chroboczek, *The Babel Routing Protocol*, IETF RFC 6126, April 2011.

[12] 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.

[13] Axel Neumann, Ester López, Leandro Navarro, *An evaluation of BMX6 for Community Wireless Networks* 1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB), IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, 2012.

[14] Thomas Clausen and Philippe Jacquet, *Optimized Link State Routing Protocol (OLSR)*, IETF RFC 3626, October 2003.

[15] Axel Neumann, *Investigating Routing-Protocol Characteristics with Mesh Linux Containers (MLC)*, Workshop, UPC, Barcelona, SpainNovember 2011, [Online], Available: https://raw.github.com/axn/mlc/master/MeshLinuxContainers-x07.pdf

[16] Georgios Parissidis, Merkourios Karaliopoulos, Rainer Baumann, Thrasyvoulos Spyropoulos, *Routing metrics for Wireless Mesh Networks*, Chapter in Guide to Wireless Mesh Networks, Springer London, 2008.

[17] Richard Draves and Jitendra Padhye and Brian Zill, *Comparison of routing metrics for static multi-hop wireless networks*, In ACM SIGCOMM, 2004.

[18] 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.

[19] Pico Peering Agreement v1.0, http://www.picopeer.net/PPA-en.html

[20] OLSRd - an adhoc wireless mesh routing daemon, http://olsr.org

[21] *strongSwan - the OpenSource IPsec-based VPN Solution*, March 2013, [Online], Available: http://www.strongswan.org

[22] *OpenSSL - Cryptographic and SSL/TLS Toolkit*, March 2013, [Online], Available: http://www.openssl.org/

[23] *CyaSSL Embedded SSL Library*, March 2013, [Online], Available: http://www.yassl.com/yaSSL/Products-cyassl.html

[24] BMX6 mesh networking protocol, http://bmx6.net

[25] QMP, quick Mesh project, http://qmp.cat

[26] Guifi.net, http://www.guifi.net

[27] http://wiki.leipzig.freifunk.net/Hauptseite

[28] OpenWRT public Wiki, *IPsec Basics, Hardware performance and IPsec tuning* March 2013, [Online], Available: http://wiki.openwrt.org/doc/howto/vpn.ipsec.basics

[29] European Commission FP7, Future Internet Research and Experimentation Initiative (FIRE), *CONFINE Project: Community Networks testbed for Future Internet*, contract FP7-288535, http://confine-project.eu