**Article submitted to Computer Networks**

**Title:** *Matrix: Multihop Address allocation and dynamic any-To-any Routing for 6LoWPAN*

**Manuscript Number:** COMNET-D-17-999

Dear Editor and Reviewers

We would like to thank you and the anonymous reviewers for your careful reading and constructive comments on our submission. The feedback of the reviewers helped us to improve our manuscript. We have considered all issues raised by the Reviewers. In this version of our manuscript, we have tried our best to incorporate all comments. We hope that you and the Reviewers will find this updated version your satisfaction.

Sincerely yours,

The Authors

**Reviewer #1’s comments and Author’s response**

The paper has been published as a conference paper in the following link.

https://dl.acm.org/citation.cfm?id=2989139

with this title:

Matrix: Multihop Address Allocation and Dynamic Any-to-Any Routing for 6LoWPAN

The author must clearly describe the contribution of the journal version in comparison to the published conference version. The parameters, and some figures are them same.

**⯈ Response:** *We would like to thank the Reviewer for the feedback.* We note that the current manuscript extends our previous work with the same title, published at ACM MSWIM 2016. In the current work we present an extended and deeper analysis of the Matrix routing protocol.

In summary, the current manuscript extends our previous work by presenting the following additions:

1. In this work we describe in detail all the communication routines of MHCL, the Multihop Host Configuration for 6LoWPAN, which is an important building block of the Matrix protocol. Note that MHCL was not described in detail in the MSWiM paper, but a preliminary analysis of it was previously published in [2] (SBRC 2015).
2. We now consider collection protocols that perform reverse routing against our any-to-any protocol solution. All the experiments were re-executed from scratch and compared to a new baseline protocol that implements reverse routing, namely the XCTP [7]. Therefore, Figures 5, 6, 7 and 8 were updated and Figures 9 and 10 were newly added to the journal version of the paper. We performed an in-depth analysis of data traffic scenarios which favor different protocols. In particular, we characterize scenarios in which XCTP has severely degraded performance in top-down routing, whereas Matrix’s performance is unaffected.
3. A new data traffic pattern application was implemented and evaluated in the simulations, namely Any-to-Any routing.
4. We summarize the main (common and different) features of related protocols for 6LoWPAN in the Related Work Section.

We added a summary of the additional contributions of this journal version, relative the conference version, in the Related Work Section.

**Reviewer #3’s comments and Author’s response**

Summary: the paper presents an IPv6 address allocation routing protocol that are low overhead and with low memory footprint that support Wireless WAN static (not mobile) networks. The algorithm proposed is superior to existing schemes as it supports not just collection of data from leaves to root or distribution from root to leaves but also any-to-any communication (where every leaf can communicate with any other leaf). The paper is clear to read and provides enough evaluation. However, it seems to me that it can be extended to more efficient algorithms taking the structure of the network into account.

Specifically I would suggest the following 2 extensions of your algorithm:

1. The time based algorithms for address space collection could be improved by adding the notion of signaling from all ports of the switch. In other words: the switch knows how many active links it has. This knowledge could be used to avoid the timing uncertainty of the current proposed timing expiration approach. A protocol that clearly require some message to arrive on all inputs could be used. Such that the allocation algorithm can finish much faster and with lower probability of late comers.

**⯈ Response:** *We would like to thank the Reviewer for the feedback. We would like to point out that Matrix is designed to operate in a fully distributed (dynamic) tree topology, where each node only has knowledge about its direct children in the tree at each point in time. Since we implemented Matrix over the CTP, in the initial state of the network nodes do not have any information about neighboring links. That's why Matrix uses the Trickle timer to define what we call a "stable" network configuration.*

*To sum up, we agree with the Reviewer that such a mechanism could potentially improve throughput. However, the corresponding solution would have to rely on a more restricted communication model, where all nodes would store and keep updated information about all potential neighbors. We added we add a justification for using Trickle timer in Section 2.1.*

2. Since there is an explicit event of registering or un-registering every DCTree entry one would think that a more proactive approach than local broadcast could be used. At the event of DCTree registration, a traversal of the path from the common ancestor could be made and a single bit marking on the address table could mark the fact the packets should be forwarded up instead of down (following the IPTree). Such that eventually they reach the common ancestor with DCTree entry to that destination. This will avoid unnecessary reties, timeouts and broadcast. This scheme, if done correctly will also avoid the current proposal reliance on the "proximity" of the alternative links (most of them connect to a brother node - connected to the same parent) which looks like an undesirable limitation of your scheme.

**⯈ Response:** *We would like to thank the Reviewer for the feedback. Regarding the local broadcast mechanism, we have a trade-off between updating the routing tables from the new parent to the lowest common ancestor and the broadcast messages. Since we consider that links are dynamic, but nodes do not move and that eventually the node will re-establish its connection to its original parent, updating all tables is a more costly and slow process. This happens because, to forward the message, the* *common ancestor must know about the parent change. Therefore, all nodes in the path from the new parent and the common ancestor must communicate and update their routing tables.*

*To sum up, this is a valid suggestion, especially if Matrix were designed to work in a mobile environment, where nodes would actually change location over time. However, the memory footprint of such a solution would be linear with the number of mobile descendent nodes (or parent changes) in each node's subtree. Therefore, we chose to keep the approach that we had already implemented, because it is sufficient to handle link dynamics (without node mobility), while guaranteeing a constant memory footprint at each node.*

3. A short intro of the concepts of Collection Tree (CTree) and Reverse Collection Tree (RCTree) could be useful addition to the reader.

**⯈ Response:** *We would like to thank the Reviewer for the feedback. We have adopted the suggestion in our revised manuscript, and added a short intro of the concepts of Ctree, IPtree and RCtree in the Introduction Section:*

* *The Ctree is built and maintained by a collection protocol (in our case, CTP). It is a minimum cost tree to nodes that advertise themselves as tree roots.*
* *The IPtree is built by matrix over the first stable version of the Ctree in the reverse direction, i.e., nodes in the Ctree receive an hierarchical IPv6 address from root to leaves, originating a static structure.*
* *The RCtree is the reverse collection tree that reflect the dynamics. Since links might change due to link qualities, at some point in the execution the IPtree no longer corresponds to the reverse Ctree.*

Nits:

Page 2 line 8: "and links are dynamics" probably "and links are dynamic". Also I would say this sentence is not clear anyway. You probably want to say that nodes have static locations in the hierarchy however, links status may be dynamic… The preferred parent may change…

**⯈ Response:** *We would like to thank the Reviewer for the feedback. We have adopted the suggestion in our revised manuscript.*