

Performance of Taroko, a Cluster-based Addressing and Routing scheme for Self-Organized Networks

Julien Ridoux^{*}, Meriem Kassar[†], Mathias Boc[†], Anne Fladenmuller[†], Yannis Viniotis[‡]

ABSTRACT

Self-Organized Networks (SONs) are a general description of autonomous networks without infrastructure, of which Ad Hoc, Sensor and Mesh networks are special cases. We had previously proposed a novel clustering architecture for SONs, named Taroko, that utilizes a fixed description of cluster routing tables to cope with SON's characteristics. Taroko uses trellis graphs to provide addressing/locating and routing capabilities for the SON, while providing a built-in multi-path routing scheme and redundancy in the addressing space. This paper focuses on comparing Taroko to AODV and OLSR in terms of both control and data plane metrics. The simulation results show that Taroko performs well, especially in the presence of large numbers of sources.

Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms

Algorithms, Performance

Keywords

Self-Organized Networks, regular structure, trellis graphs

1. INTRODUCTION

With the deployment of wireless technologies, mobile computing has become a definite challenge for the networking community. As they share common auto-configuration constraints and properties, mobile networks, such as Ad hoc,

^{*}EEE Dpt - CUBIN The University of Melbourne, Victoria 3010, Australia. j.ridoux@ee.unimelb.edu.au

[†]Lip6-UPMC, 8, rue du Capt. Scott, 75015, Paris, France. {meriem.kassar, mathias.boc, anne.fladenmuller}@lip6.fr

[‡]ECE Dpt - NCSU, Box 7911, Raleigh, NC 27695, USA. candice@ncsu.edu

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IWCMC'06, July 3–6, 2006, Vancouver, British Columbia, Canada.
Copyright 2006 ACM 1-59593-306-9/06/0007 ...\$5.00.

Wireless Mesh or Sensor networks, can all be classified under the generic description of Self-Organized Networks (SON).

The physical topology of a SON is dynamic due to node mobility, joins and leaves or the possibility for nodes to enter sleeping modes. The network topology can not be predicted and it has a significant impact on network layer protocols. As an example, the traditional IP address can no longer reflect both a node identity and location. Permitting a dynamic address allocation scheme while ensuring performance of the routing protocol has then become a challenge, centered around the description of the network topology.

A large number of proposals have been published in the context of Sensor and Ad Hoc networks to provide auto-configuration and efficient routing. A first set of proposals relies on the knowledge of node geographical location and requires the nodes to embed a geo-positioning system. Under the assumption that nodes have a unique identifier, routing protocols ([1, 2]) and location services ([3, 4]) allow nodes to exchange their location information and route packets, while ensuring good scalability for large networks. Based on neighborhood knowledge at each node, numerous Ad Hoc protocols have been proposed [5, 6]. These protocols rely on the use of IP addresses as node identifiers, but the process to “dynamically configure” nodes is not yet defined. Auto-configuration of nodes in a SON is crucial since the address allocation strategy can help network layer mechanisms, provided it reflects node location in the topology. A proper address allocation scheme for SON would help routing protocols by decreasing either the control plane load for proactive routing protocols or the route discovery latency of reactive routing protocols. Preliminary proposals have been made for such Ad Hoc routing protocols. They target the definition of an adaptation of a Duplicate Address Detection [7] involving wide flooding of the Ad Hoc network.

In this paper, we address the problem of providing an addressing and routing scheme with good properties to handle the peculiarities of SON's topology. Our approach, named Taroko, provides an original addressing space, topological location service and routing protocol exploiting the decoupling of address and identity of nodes in a SON. In this sense, our approach is closer to previous works stemming from Peer-To-Peer concepts ([8, 9]). However, contrary to these approaches, Taroko relies on a description of the network topology resulting from the mapping of the physical connectivity into a hierarchical structure of recursive clusters. The predefined virtual organization of nodes within each cluster imposes the addresses allocation scheme and a default routing strategy.

Individual node mobility, as well as network splits and mergers are still open problems for most of the existing solutions. Based on the use of trellis graphs, Taroko introduces address redundancy to supply an addressing space robust to mobility. It also provides a built-in multi-path property that helps the routing protocol to handle the transient presence of nodes. Finally, the hierarchical, recursively-defined structure helps address the issue of network splits and mergers.

The details concerning Taroko’s addressing space construction, lookup mechanisms and routing strategy have been given in [10] and [11]. [12] gives insights about the parameters defining the best cluster size to build the addressing space. In this paper, we focus on the evaluation of the performance of our approach and compare it to two Ad Hoc routing protocols: OLSR [5] and AODV [6]. The present study does not investigate all the properties of Taroko but is a first effort for its overall evaluation.

The rest of the paper is organized as follows. In Section 2, we summarize the main characteristics of Taroko. In Section 3.2, we compare Taroko to OLSR and AODV, in terms of data plane metrics. In Section 3.3, we detail the behavior of our approach in the presence of node join and leave and the corresponding cost of the structure maintenance.

2. VIRTUAL STRUCTURE DESIGN

In this section, we describe the main characteristics of the Taroko proposal in terms of addressing space, location service and routing strategy. Similar to geographic approaches, Taroko relies on the assumption that nodes own a unique identifier; however, contrary to these approaches, the node locations are not defined by their geographical location. Based only on node neighborhood knowledge, the SON topology is represented as a virtual structure. This representation maps node locations into the structure, defining their address and routing strategies.

2.1 Virtual Structure

While the topology of a SON is basically flat (each node is equally in charge of routing functions), the virtual structure introduces a hierarchy of clusters. Figure 1 shows the topology of a SON composed of 11 nodes labelled from A to K. Figure 1 also shows one possible description of the topology as a virtual structure. It is worth noticing that the final virtual structure can be different depending on the node arrival sequence in the network. Based on such a representation of the topology, the virtual structure provides the means to support addressing, locating and routing mechanisms.

A cluster of the structure represents a subset of the physical connectivity existing among close nodes of the SON. The organization of each cluster is given by a predefined pattern known by all nodes. This pattern defines the subsequent topology description that results from the cluster construction. We use trellis graphs as the pattern defining cluster organization in our implementation of Taroko. As in convolutional encoders [13], a trellis is a regular directed graph which we will represent as a Finite State Machine (FSM). As an example, the top right part of figure 1 gives the organization of a cluster as a trellis of size 8. This pattern defines a maximum subset of neighbors. As a directed graph, the trellis pattern also imposes the routing strategy shared by all nodes of a common cluster.

As nodes are added to a cluster, they are mapped onto the trellis pattern. This mapping associates each node to

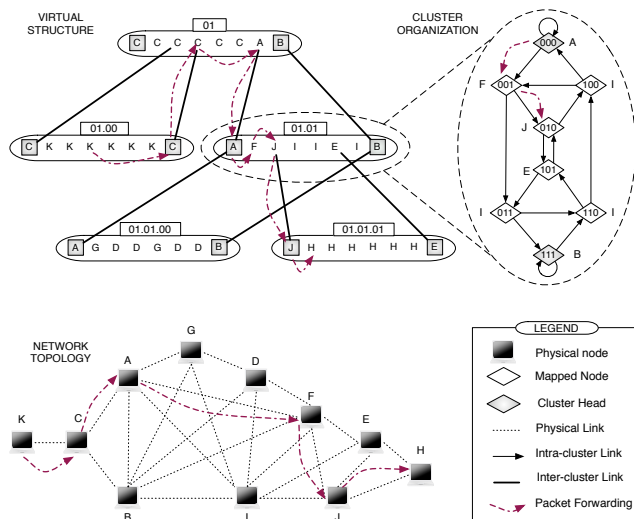


Figure 1: A physical topology, corresponding virtual structure and the detailed organization of one cluster based on a trellis graph of size 8.

one or several vertices in the trellis, matching the node’s physical connectivity constraints. A node can be associated to a specific trellis vertex if it can physically contact the nodes associated to its adjacent trellis vertices. The resulting mapping represents nodes in a cluster and a subset of the physical links connecting them. When a new node joins the network and can not be inserted in an existing cluster, it triggers the construction of a new cluster. The node selects two of its neighbors and creates a cluster of lower hierarchical level in which the two selected neighbors become the cluster heads. Cluster heads belong to different clusters and ensure the connectivity between adjacent hierarchical levels.

The structure obtained by such a construction process possesses several properties. The creation of clusters, and thus the dynamic address allocation, is distributed and realized among neighbors only. There is no limit to the number of clusters formed, leading possibly to an addressing space with no size limitation. The hierarchical and locality properties of the virtual structure avoid network-wide flooding and limit the control plane load while dynamically allocating addresses to nodes. In terms of routing, the hierarchical clusters permit route aggregation and require routing tables of fixed size. Taroko’s addressing space introduces redundancy in several ways. Several trellis vertices in a cluster may be assigned to a node in a cluster if the physical connectivity requires it. Besides, if a node is a cluster head, it appears in several levels of the hierarchical structure. Each cluster maintains two cluster heads, leading to a more robust structure. Finally, the recursive and hierarchical cluster-based representation of the topology can help address the issue of network splits and mergers, by providing an adaptable addressing space.

2.2 Addressing space

The virtual structure defines the addressing space used by the network layer. Clusters are numbered as an n -ary tree, producing binary strings named cluster prefixes. Figure 1 shows the prefixes defined by the setup of the different clus-

ters. As an example, the prefix of the highest-level cluster of the structure is 01.

Within a cluster, nodes are associated with trellis vertices whose labels represent nodes’ relative addresses in their close neighborhood. To get a node’s absolute address in the virtual structure, one should associate the node’s relative address with the prefix of the cluster it belongs. Note that a node appearing several times in the virtual structure obtains several relative and absolute addresses. As an example, in cluster 01.01, node *A* is mapped to the vertex 000, defining its relative address. A physical node can also be mapped to several vertices of the trellis, leading to several relative addresses. In cluster 01.01, node *I* is mapped to the vertices 011, 100 and 110.

Cluster heads are compulsorily nodes associated to the vertices of lowest and highest label values, avoiding the need of any cluster head election procedure. The absolute address of a node identifies it uniquely, gives a straightforward information about its location in the virtual structure and so, the route to reach it. Finally, the recursive definition of cluster prefixes eases network splits and mergers as nodes’ addresses are only defined by the relative position of the clusters in the virtual structure.

2.3 Routing operations

Since the node addresses provide both their positions in the virtual structure and the way to reach them, routing operations are mostly confined to packet forwarding. To send packets, the source node first has to retrieve the address of the destination. Then, data packets are forwarded by relay nodes along an inter-cluster and an intra-cluster strategy.

As nodes join a cluster, they register their location information in the structure. At least one of their addresses is proactively propagated to the higher levels of the structure and stored at the cluster heads “on the way up”. When the source node initiates a communication, it reactively sends a request to the clusters of higher levels, until it reaches one cluster maintaining the destination’s location information. Once the destination address in the virtual structure is retrieved, the forwarding functions are initiated.

Intra-cluster forwarding function is realized thanks to the FSM representation of trellis graphs. Figure 1 shows how the packets sent to the final destination *H* are relayed within cluster 01.01. Based on the cluster prefix of the destination contained in the packet header, node *A* computes that *J* is one of the cluster heads of cluster 01.01.01 in cluster 01.01. The packets can then be forwarded to *J*, the gateway to the adjacent cluster 01.01.01 containing the destination *H*. The trellis pattern representing the topology connectivity indicates to *A* that a route to *J* can be found by relaying packets to *F*. This short example shows how the mapping of nodes into the trellis pattern implicitly defines default routes within a cluster.

The connection between clusters of adjacent levels is physically maintained by the cluster heads. For a relay node, inter-cluster forwarding function consists then in selecting, among the clusters it belongs, the one it has to relay the packets in. When processing a packet’s header, the relay node basically runs a longest prefix match algorithm to compare the cluster prefix contained in the packet header to the prefix of the clusters it belongs. In the example given in figure 1, this corresponds to the forwarding of packets between clusters of adjacent hierarchical levels.

	Multiple Sources	Dynamic Topology
Number of nodes	100	100
Dimensions	1200*1200m	800*2000m
Simulation time	150s	300s
Traffic	CBR (UDP)	CBR (UDP)
Packet Size	512 Bytes	512 Bytes
Source rate	5 kBytes/s	50 kBytes/s
Number of sources	5 to 25	1
Transient nodes proport.	0%	40% to 80%

Table 1: Simulation parameters

The mapping of cluster nodes to the trellis pattern is shared by the nodes of a cluster and constitutes their forwarding table. The default route defined in a cluster may not be optimal, as the trellis pattern does not represent all existing links. Based on the physical node neighborhood, path optimization defines shortcuts in the default route, leading to near optimal paths within each cluster and avoiding routing loops. The set up of the shared routing table is realized in a proactive manner among close nodes, avoiding the need of network-wide flooding. As routes are directly given by node location information, no route discovery process is needed.

3. SIMULATION RESULTS

The construction of the optimal virtual structure is an NP-Complete problem; we provided a construction heuristic and evaluated it in [11]. That first study showed that the construction of the addressing space is always feasible and that the proposed heuristic scales well to large networks. In [12], we examined what cluster size is best, in terms of keeping a reasonable control plane load and maintaining data plane performance. This second study showed that clusters of size 16 or 32 meet these objectives. The study also showed that, when compared to the physical shortest paths, the path lengths in the data plane are sub-optimal, by a factor of at most 2 in more than 70% of the cases.

In this present study we aim to compare the data and control plane performance of our approach to two other well-known protocols, AODV and OLSR. For this purpose we set up two scenarios. The first one uses static nodes with a varying number of competing communication flows. The second one measures the impact dynamic changes in the topology would have on a single communication flow, in terms of route stability and convergence.

3.1 Environment and evaluation parameters

We implemented Taroko in the Network Simulator (NS-2.28). We use the AODV code distributed with NS-2.28 and a patch for OLSR from the University of Murcia [14]. Both of these protocol implementations correspond to the OLSR and AODV RFC ([5, 6]). The simulations use the CMU’s wireless extensions for the NS simulator with the IEEE 802.11 radio and MAC model provided [15]. Each node’s transmission range is set to 250 m. We generated CBR traffic on top of UDP that is preferred to TCP. OLSR and Taroko hello message frequency has been equally set to 2 seconds. The size of trellis graphs used by Taroko has been set to 16. Each set of scenario parameters has been run 15 times to provide meaningful averaged values of the evaluation metrics.

For each scenario we observe four metrics. The first metric measures the number of packets per node and per second, in

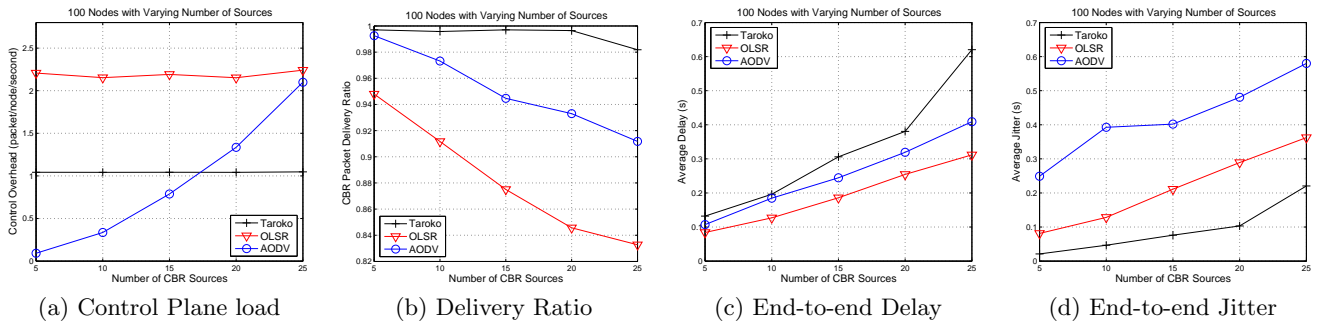


Figure 2: Averaged performance metrics for a varying number of sources in a static network of 100 nodes.

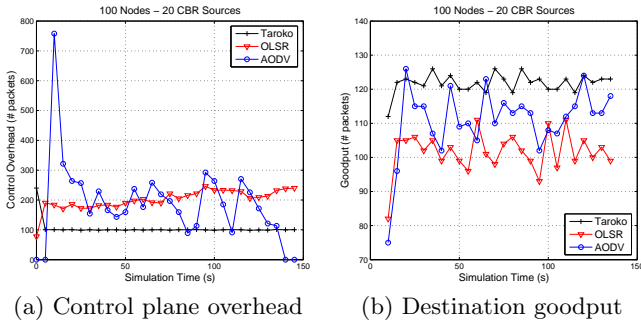


Figure 3: An example of control plane load and destination goodput with 20 active CBR sources.

order to evaluate the load each node has to handle continuously. As we use UDP flows that do not retransmit dropped packets, we observe the delivery ratio (i.e., the number of received packets over the number of sent packets) of each source destination pair. This second metric provides an indication of the quality of the routes defined, in the presence of channel access and competing flows issues. In addition to the delivery ratio, the average end-to-end delay and corresponding jitter are computed based on the set of packets effectively received by the destination nodes. The jitter is computed as the standard deviation of the end-to-end delay. Both delay and jitter depend on different parameters such as interference, congestion at relay nodes and length of the routes in number of hops. These two metrics then give an overall indication on the adaptation of the routing protocols to the wireless environment.

3.2 Increasing the communication load

The first scenario we set up aims at observing the reaction of the routing protocols to a varying communication load. For this purpose, scenario 1 consists in a topology of 100 static nodes placed randomly within a square area. Source and destination pairs are randomly chosen among nodes. Other parameter values for this scenario are shown in table 1. For each simulation run, we evaluate the four control and data plane metrics with a varying number of transmitting sources. Figure 2 shows the aggregated results for each of the protocols studied.

Figure 2(a) shows the control plane load for each of the protocols. As it was expected, OLSR shows the highest

control plane load as it periodically and proactively floods the network. On this plot, AODV shows the lowest but increasing control plane load. This result is explained when looking at the precise behavior of AODV during the simulation. Figure 3(a) shows an example of instantaneous control plane load of the three protocols during a specific run where 20 sources are on. As expected, most of the control plane load in AODV is concentrated at the nodes' communication initiation phase, a behavior common to the case where few sources are on. But more interestingly, when the number of sources increases, the load of AODV control plane is not negligible during the rest of the simulation time. This behavior is widely due to the interference among competing flows. In a comparable way as OLSR, figure 2(a) shows that the control plane load of Taroko remains stable as the number of sources increases. The clustered nature of the virtual structure used limits the propagation of control messages and makes it about half of OLSR, while having the same hello message emission frequency. As the number of sources increases, Taroko keeps a lower and stable control plane load on the network.

Figure 2(b) shows that as the network load increases, OLSR and AODV deliver less and less traffic to the destinations (OLSR drops more than 15% with 25 sources). Taroko's delivery ratio remains stable and close to the optimum. This result is explained by two correlated arguments. As the number of sources increases, AODV control plane load increases and reaches OLSR level, creating more interference and a higher probability of packet collision on the wireless link. The second argument concerns the shortest path strategy in use by these two protocols. As OLSR and AODV provide routes that correspond to the physical shortest paths, data packets are relayed by a subset of nodes whose position in the network make them concentrate more traffic. Nodes that are in charge of relaying more packets drop more of them as the link capacity is reached. In this situation, both protocols find alternative paths based on their own strategy. OLSR nodes use other entries in their routing tables while AODV nodes send request messages as route timers expire. This search for alternative paths, while always searching for the shortest one, makes routes vary in time. Taroko performs better from this point of view by imposing at first its multi-path description to the routing mechanisms. The overall communication load is then more fairly distributed in Taroko. As packets are forwarded along different paths, and with a constant and localized control

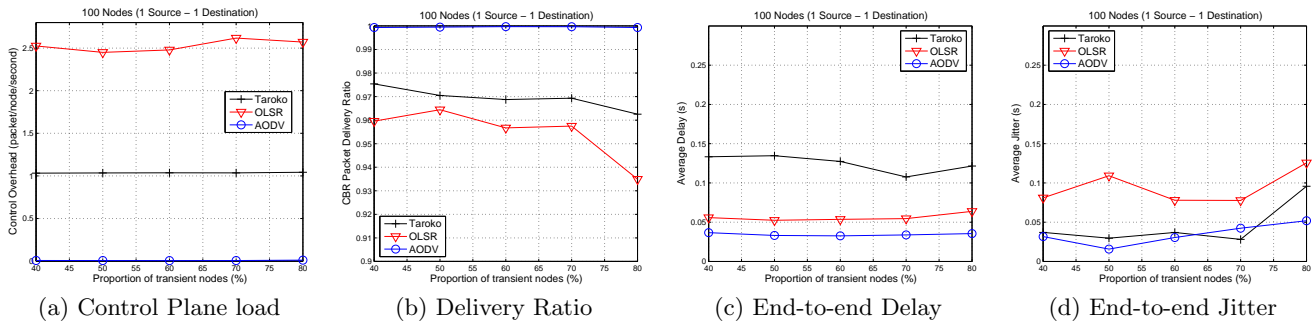


Figure 4: Average performance metrics for a varying proportion of transient nodes in a network of 100 nodes.

plane load, routes remain stable and performance is maintained.

This strength of our protocol has its own cost. The multi-path benefit mandates routes that do not follow the physical shortest path. As shown in figure 2(c), the average delay of packets received by the destinations is then larger in Taroko as the number of hops on the route increases. We then observe that OLSR provides the smallest delays as its routing algorithm converges to the physical shortest path. AODV performs a little bit worse, as its routing algorithm suffers from interferences in finding the optimal routes. Taroko performs the worst on this evaluation metric. Nevertheless, Taroko's delay is never higher than twice the delay of OLSR. It is important to notice that the delay depends on the delivery ratio. By delivering more packets, the delay computed for Taroko takes into account a higher proportion of packets undergoing higher delays. The average delay values with 25 sources are then much less significant than the ones with 5 sources. At a high communication load, a fair comparison would make Taroko incur a lower delay increase.

Figure 2(d) depicts the delay jitter metric and indicates that Taroko performs the best while AODV the worst. This again shows that Taroko does not provide the shortest path routes, but that the routes and the corresponding delay remain stable. Such a stability feature is also observed in figure 3(b) where the goodput observed at the destination with Taroko is the highest and the most stable. The stability of Taroko's route makes TCP flows reach even better performances by taking advantage of the retransmission process; due to space limitation, such results are not presented here. Nevertheless figure 2(d) shows that the jitter for AODV and OLSR is in the same order as the delay surely leading to instability.

3.3 Reaction to transient node presence

In this second scenario we aim at observing the effect of the transient presence of nodes on Taroko, AODV and OLSR. Depending on its geographical location, a node's join or leave may not affect any ongoing communication. In order to limit the scope of this investigation, we focus on a single communication between a source and a destination. The source and destination are placed at the extremities of a rectangular area of 800x2000 meters as shown in figure 5.

During each simulation run of 300 seconds, a proportion of the 100 relay nodes join and leave the network. The node identifiers and time at which nodes join or leave the network are chosen randomly. This join/leave process repeats

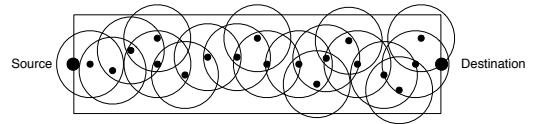


Figure 5: An example of nodes' initial positions for a single source and destination communication with a transient presence of relay nodes.

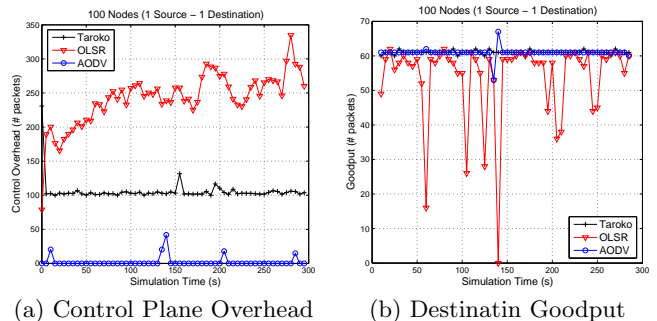


Figure 6: An example of control plane load and destination goodput with 60% of transient nodes.

throughout the simulation, causing changes in the topology and possibly changes in the route between the source and destination node. The other simulation parameters are indicated in table 1. Figure 4 shows the metrics observed for scenario 2.

As in scenario 1, figure 4(a) indicates that OLSR is the protocol that produces the largest control plane load on the network. In the case of OLSR, under this dynamic scenario, node join and leave events trigger more control messages to update routing tables. This scenario favors AODV since the route requests do not undergo interferences, making the control plane load negligible. Moreover, since the source and destination do not move, AODV can rely on local repairs to fix broken routes without having to flood the network. This is illustrated by the example of 60% of transient nodes presented in figure 6(a), where AODV control plane shows few and small load peaks. Taroko performs well even if based on the definition of virtual structure. The dynamic topology changes are handled while increasing the control plane load in the same order as AODV, that is, very few.

Since Taroko's structure relies on a local definition, the join and leave events are handled locally, resulting in a stable control plane load.

Figure 4(b) clearly indicates that AODV performs the best in delivering packets. Again, local route repairs help AODV in finding routes while dropping few packets. OLSR and Taroko nodes both rely on the proactive hellos to update their neighborhood and routing information. The proactive approach imposes some latency for routes to converge. Thus, Taroko and OLSR are likely to drop more packets. Nevertheless, Taroko performs slightly better than OLSR thanks to the local definition of the clusters, making routes converge faster. As an example, figure 6(b) shows the goodput observed at the destination for a particular realization of scenario 2. First it can be seen that OLSR may suffer a lot from node join and leave events and it may provide unstable delivery of packets. On the contrary, Taroko is able to maintain stable performance even by having to maintain the virtual structure.

Figure 4(c) shows that the average end-to-end delay for AODV, OLSR and Taroko remains stable as more nodes appear or disappear from the network. Taroko exhibits the highest delay again, because the route to reach the destination is sub-optimal in terms of number of hops. Figure 4(d) also shows that Taroko and AODV have comparable jitter values. Nevertheless, the jitter in AODV remains in the same order as the end-to-end delay, again forecasting more instability of TCP connections. The case of OLSR is even worse. Taroko's jitter remains lower than its end-to-end delay when the proportion of transient nodes increases. This again shows a higher stability of the connection. However, the route recovery for each of the three protocols induces variations of the delay in the same order of magnitude.

As mentioned before, scenario 2 highly favors AODV because no data stream competes with the reference one whose end nodes are static. By comparing the two proactive protocols, we observe that Taroko performs better than OLSR on each of the metrics. This shows that the use of a virtual structure is compatible with a dynamically changing network where nodes can join and leave the network. The performance improvement of Taroko compared to OLSR is then due to the local neighborhood knowledge on which the virtual structure relies.

4. CONCLUSION AND FUTURE WORK

In this paper we have briefly presented the essential functions of Taroko, a novel architecture for SON. Taroko is designed to provide for dynamic, redundant address allocation, location service and multi-path routing functions.

The simulation results show that even though our approach uses sub-optimal routes, it still provides good performance by utilizing multipath routes. At the cost of a relatively minor delay increase, Taroko improves the overall performance of the network, ensuring packet delivery as the communication load increases. In a dynamic environment, Taroko performs slightly worse than AODV in delivering packets. Nevertheless, since Taroko's control plane load is small and constant, its data plane performances could be improved by increasing hello message frequency.

The simulations show the feasibility of the use of a virtual structure in a SON environment. While the scenarios defined allow a precise analysis of Taroko they do not exploit all of its properties. We strongly believe that the combina-

tion of local and recursive address definition provided by Taroko has the potential to address the thorny issue of network splits and mergers; this is the subject of future work.

5. REFERENCES

- [1] Y. Ko and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," in *Proc. of MOBICOM 1998*, October 1998, pp. 66–75.
- [2] B. Karp and H. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," in *Proc. of MOBICOM 2000*, October 2000, pp. 243–254.
- [3] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," in *Proc. of ACM MOBICOM 2000*, August 2000, pp. 120–130.
- [4] Y. Xue, B. Li, and K. Nahrstedt, "A Scalable Location Management Scheme in Mobile Ad-Hoc Networks," in *Proc. of IEEE Conference on Local Computer Networks (LCN)*, November 2001.
- [5] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol," IETF, RFC 3626, October 2003.
- [6] C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," IETF, RFC 3561, July 2003.
- [7] J. Jeong, H. Cha, J. Park, and H. Kim, "Ad-hoc IP Address Autoconfiguration," IETF, Internet-Draft, July 2004.
- [8] A. Viana, M. D. Amorim, S. Fdida, and J. D. Rezende, "An Underlay Strategy for Indirect Routing," *ACM Wireless Networks*, vol. 10, no. 6, pp. 747–758, November 2004.
- [9] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, "Scalable Ad Hoc Routing: The Case for Dynamic Addressing," in *Proc. of IEEE INFOCOM 2004*, March 2004.
- [10] J. Ridoux, A. Fladenmuller, Y. Viniotis, and K. Salamatian, "Trellis-Based Virtual Regular Addressing Structures in Self-Organized Networks," in *Proc. of IFIP Networking2005, Waterloo, Canada*, May 2005, pp. 511–522.
- [11] J. Ridoux, A. Fladenmuller, and Y. Viniotis, "Virtual Trellis Routing: how Regular Structures can ease Network Layer operations," in *Proc. of Med-Hoc-Net 2005, Ile de Porquerolles, France*, June 2005.
- [12] J. Ridoux, A. Fladenmuller, and Y. Viniotis, "Definition and Evaluation of a Trellis Structure for Self-Organized Networks," in *Proc. of LOCAN 2005 Workshop (in conjunction with MASS 2005), Washington, DC, USA*, November 2005.
- [13] S. Lin and D. Costello, *Error Control Coding: Fundamental and Applications*, ser. Electrical Engineering Series. Prentice-Hall, 1983.
- [14] F. Ros, "UM-OLSR patch for NS-2.28," March 2005, <http://ants.dif.um.es/masimum/um-olsr/html/>.
- [15] C. M. Group, "CMU Monarch Extensions to ns," <http://www.monarch.cs.cmu.edu/>.