WMM - Wireless Mesh Monitoring

(extended abstract of the MSc dissertation)

Ricardo Pinto
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Professor Luís Rodrigues

***Abstract*—This work presents a novel adaptive cluster-based monitoring scheme for wireless mesh networks. Nodes self-organize in semi-circular clusters and monitoring information from each cluster is aggregated by the cluster head and forwarded to the monitoring station. Clustering takes into account existing data-flows in the mesh network, to minimize the interference of the monitoring traffic on active streams.**

I. Introduction

Within the short span of a decade wireless networks have revolutionized the way we use our devices, bringing us cable-free mobility, always-on connectivity, and reduced infrastructure costs. Wireless Mesh Networks (WMNs) have emerged as a key technology to ease the deployment of wireless networks. A WMN is a multi-hop network, dynamically self-organized, self-configured, self-healing, resilient to device failures, and highly scalable; where all the nodes in the network assure the availability of one or more paths among each other[1], [2].

As in any other network, an important activity that needs to be supported in WMNs is network monitoring, in order to allow operators to gather information about the network operation and quickly detect anomalies or performance degradation. Therefore, it is necessary that every node reports its performance data, introducing extra traffic in the network, which can contribute to a significant increase in overhead. If performed incorrectly, network monitoring traffic may have a negative impact on network performance. Therefore it is important to use the most adequate monitoring solutions that minimize the usage of network resources.

This work proposes a novel strategy to perform network monitoring on WMNs. The key goal of our approach is to devise a monitoring strategy that reduces the impact of the monitoring traffic on the existing data streams that may be active in the network.

We employ a cluster based solution, i.e., nodes of the WMN self-organize into clusters such that cluster-heads can aggregate information before it is sent to one (or more) monitoring stations. The use of clusters for information aggregation is not new and has been used, for instance, in wireless sensor networks[3][4]. However, our solution embodies the following novel techniques:

- Clusters are semi-circular, such that information always flows in the direction of the monitoring station;
- Cluster formation is aware of existing data-flows in the WMN, to minimize the interference of the monitoring traffic on active streams.

The rest of this work is organized as follows. Section II provides a survey of the related work. Section III describes our architecture and the clustering algorithm. Section IV evaluates the performance of our solution. Finally, Section V concludes the paper.

II. Related Work

The Simple Network Management Protocol (SNMP) [5] is the de-facto protocol for management of most networks. SNMP was originally designed for static wired networks and uses a centralized approach for monitoring purposes. Devices run SNMP agents which are periodically polled and send information to a centralized device in the network. The use of this model in WMNs is limited due to the inherent poor scalability.

MeshMon is a multi-tiered framework [6] that only monitors a small subset of metrics (baseline metrics) when the network performance is satisfactory. The indication of a potential problem is perceived when those metrics cross a determined threshold, causing the system to transition to collect more detailed metrics. Overhead reduction is then achieved by only transmitting the necessary metrics for a specific problem set, limiting the amount of information available for a full network diagnosis at any given point.

MeshFlow [7] is a probe-based solution. Each node sends a special packet that contains a summary of properties of data packets passing through a mesh router. For each hop in the route the packet traverses more information is added and hence the growth of the packet size can affect scalability. Existing records in the packet can be shortened by functions like average or maximum values, but the detailed information is lost if such aggregations are performed. Furthermore, the system does not scale very well as node density rises.

MMAN [8] relies on multiple monitoring stations that collaborate and combine information. A number of these stations are deployed throughout the network and act as passive monitors. This solution requires the stations to be equipped with two radio interfaces: one for listening to the traffic and another to transmit monitoring information stations. Albeit not injecting additional traffic, the stations

increase deployment cost by requiring an extra radio interface and an extra network to transfer monitoring data.

DAMON [9] uses an agent-sink architecture for monitoring mobile networks. Agents in the nodes discover the sinks automatically through periodic beacons (initiated by the sinks). Beacons can also transport agent-instructions that update the nodes and enable the adaptation to new requirements. The proximity to a sink is determined by the hop count carried in the beacon. The system is only scalable if the number of sinks grows with the number of nodes, and that growth must be done to preserve the balance of node-to-sink numbers, in order to achieve load balancing.

JANUS [10] is a distributed framework that uses Pastry [11], a DHT (Distributed Hash Table) peer-to-peer overlay network, to make information available to all nodes in the system. JANUS also uses Scribe on top of Pastry, in order to build a multicast tree for distribution of publish-subscribe events. While peer-to-peer networks do scale well in an Internet paradigm, in a resource constraint environment such as a WMN, the scalability is poor.

The above systems do not perform well with a large number of nodes. Hierarchical systems should be designed to account for unbalanced distribution of nodes through the network. Clustering has been proposed as a technique to tackle the monitoring problem in WMNs through an organized hierarchy of clusters that dynamically and autonomously reconfigure the structure as the network topology changes. The cluster-head is a special node in the hierarchy that is elected to coordinate and publish information, and to build and maintain a local network view (aggregation and correlation of data) of its cluster members.

A clustering solution was proposed in [12]. The cluster formation is random and its preconfigured to j-hops, being j a configurable parameter. Cluster size has to be determined before deploying the nodes, which may raise problems if the network density increases.

Mesh-Mon [13] is another clustering solution which operates according to three principles: each mesh node must monitor itself, each mesh node must monitor its k-hop neighbors, and each node must help in forming a hierarchical overlay network for propagation of monitoring information. Periodically, nodes probe each other to measure bandwidth and latency among clients and mesh nodes, which may interfere with application traffic. Cluster-heads are selected by its k-hop neighbors. Assuming global topology is available, all the nodes can be ranked according to their degree of importance and thus elect the cluster-head.

We have identified some of the limitations of the monitoring systems for WMNs and will focus our solution in a cluster-based design with focus on: i) scalability; ii) routing-protocol independence; iii) minimization of monitoring traffic; and iv) less application traffic interference.

III. My Work

We propose a cluster-based hierarchical approach where nodes self-organize into k-hop clusters. In each cluster there is a self-elected leader (cluster-head) to which the monitoring information is routed from the other cluster nodes. The cluster-head aggregates and forwards such information to the gateway through one or several nodes as illustrated in Figure 1.

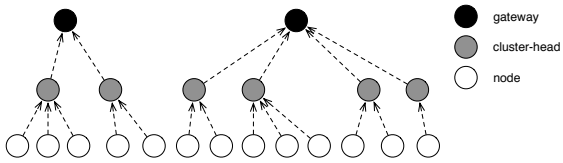


Figure 1. Architecture.

A. Routing

The routing of monitoring information to the gateway should be done independently of routing protocols used by applications to promote the non-interference of both monitoring and application traffic. A simple and pragmatic process to discover such information is to use similar mechanisms to those proposed by B.A.T.M.A.N. [14].

For this purpose, the gateway periodically sends a BEACON that is forwarded to the entire network using the protocol described below.

The BEACON message has three fields: the gateway address; an *epoch* number, that is incremented every-time the gateway sends a new beacon; and a *hopcount* value, which is initially set to zero by the gateway and is incremented by one unit every time a node forwards the BEACON.

When a node p receives a BEACON_q from node q it stores the beacon in a log and starts a *quarantine* timer, in order to wait for other possible retransmissions of the beacon. The goal of the quarantine period is to ensure that a node forwards the beacon with the correct hop count value. The log at each node keeps the record of all BEACON_q messages from the past e *epochs* (e is a configurable parameter of the protocol). At the end of the quarantine period, the node searches its log for the lowest hop count from all stable beacon sources. Let the *beacon count* for a source q of a BEACON message, denoted bc_q, be the number of *epochs* for which a BEACON_q appears in the log ($bc_q \leq e$). A source q is said to be stable if for every other source r in the log we have $bc_q \geq bc_r$. From all stable sources, the node p selects the source t that has sent the BEACON message with lower hop count. Finally, node p sets t as its next hop to the gateway, increases the hop count and forwards a new BEACON message to all its neighbors.

As a result of the procedure above all nodes collect the following information: i) distance in number of hops to the gateway and; ii) next hop node to be used when forwarding monitoring information to the gateway.

Route Stability: The algorithm above has the disadvantage that the omission of a single BEACON message may cause a node to change its next hop to the gateway. Let t be current next hop neighbor for routing messages to the gateway. In

order to promote route stability a node p only replaces t by another neighbor t' if the difference between their beacon counts, $bc_{t'} - bc_t$, is greater than a *stability threshold*. In all our experiments we have set the size of the log e to 10 *epochs* and the value of the stability threshold to 2.

B. Clustering

The goal of the clustering algorithm is to ensure that nodes self-organize in clusters with the following properties:

- All nodes belong to a single cluster;
- In each cluster, there is one and no more than one cluster-head.
- The shortest path between any two cluster-heads has at least $k+1$-hops.
- Gateways are always cluster-heads (minimizing the cost to route the monitoring information)

For this purpose, all network nodes execute the algorithm described below. In this algorithm, nodes can be in four possible states, namely:

- QUARANTINE;
- UNCLUSTERED;
- CLUSTERED;
- CLUSTER-HEAD.

A node initiates the operation of the algorithm in the QUARANTINE state. In this state nodes first wait until they have acquired their distance to the gateway, according to the algorithm described in Section III-A. As soon as the distance to the gateway has been computed, nodes initiate a timer, with a value defined by Eq. 1 or Eq. 2, and set their state to UNCLUSTERED.

$$SelfElection_c(dist_{gw}) = \begin{cases} dist_{gw} + \lambda(s), \\ if dist_{gw}\%(2k+1) = 0 \\ \\ \alpha \times dist_{gw} + \lambda(s), \\ otherwise \end{cases} \quad (1)$$

In both Eq. 1 and Eq. 2 α is a constant that increases the time of self-election of non-optimal nodes and the λ is a random value between 0 and 1 that avoids multiple nodes to self-elect at the same time and thus reducing the convergence time. The cluster-head election equations aim at promoting to cluster-head the nodes that are more favorable to minimize the traffic costs when collecting monitoring information. In particular, it aims at ensuring that cluster-heads closer to the monitoring station are elected before the cluster-heads farther away, and that cluster-heads are within k hops from each other, as illustrated in Figure 2.

When the timer expires and the node is still in the UNCLUSTERED state, it self-elects as a cluster-head, setting its state to CLUSTER-HEAD, and starts broadcasting periodic HELLO messages which contain the cluster-head address, a TTL set to k and the distance (in hops) to its gateway.

If during the unclustered period a node receives a HELLO message from a cluster-head candidate c, the node aborts the timer and sets its state to CLUSTERED, and its cluster-head is set to c. Further, the node decrements the TTL of the HELLO message and retransmits it if the TTL values is still greater than zero. Nodes retransmit HELLO messages and select the best next hop towards their cluster-head executing the same procedures presented in Section III-A, for the processing of BEACON messages.

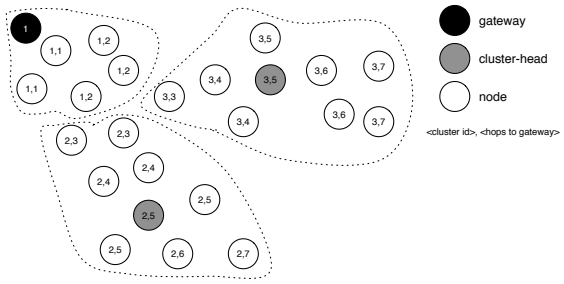


Figure 2. Circular Clustering.

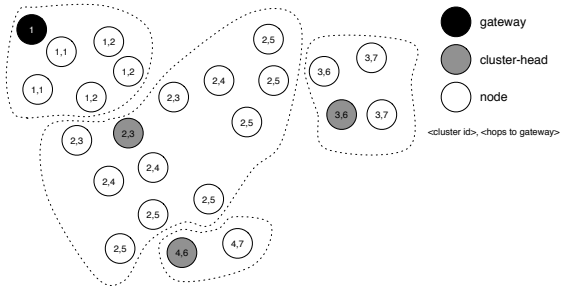


Figure 3. Semi-Circular Clustering.

Cluster-heads are responsible for aggregating the monitoring information sent periodically by the nodes and for sending that result to the closest gateway. The rate of aggregation is a multiple of the nodes' send rate (in our experiences we used double the rate) By introducing an aggregation layer between the nodes and the gateway, the system becomes more flexible and adaptable to the underlying network conditions. Cluster-heads can perform multiple operations to the data collected: averages, maximum or minimum, compress data or execute other complex operations.

Semi-circular Clustering: Most clustering algorithms, including the algorithm described above, create topologies where the cluster-head is in the center of the cluster. One problem of this configuration is that, since the information is first routed to the cluster-head and only later to the monitoring station, in some cases, the information may be initially routed in the opposite direction of the gateway, which results in suboptimal routing.

Therefore, we propose to use a variant of the algorithm above, that favors the construction of semi-circular clusters, as illustrated in Figure 3. In this topology, the cluster-head of a node is never farther away from the monitoring station than the source of the information.

To create a semi-circular clustering, nodes set their unclustered timer according to Eq. 2. Furthermore, nodes only rebroadcast HELLO messages if the distance to the sink of the source is lower than the node's distance to the gateway.

$$SelfElection(dist_{gw}) = \begin{cases} dist_{gw} + \lambda(s), \\ if dist_{gw}\%(k+1) = 0 \\ \\ \alpha \times dist_{gw} + \lambda(s), \\ otherwise \end{cases} \quad (2)$$

Gateways: The gateways, that serve as endpoints of monitoring information execute a slightly different algorithm from the remaining nodes. In particular, these nodes always start in the CLUSTER-HEAD state. This prevents monitoring information from nodes in the vicinity of the gateway to perform an additional hop to another cluster-head.

Optimized Routing: Nodes that are best next hop towards the cluster-heads wait for the reception of the monitoring message before sending their own messages, aggregating two monitoring messages in just one packet, reducing the monitoring overhead.

C. Adaptive Information Transfer

Our solution includes a module of *Adaptive Information Transfer* (AIT) that monitors the network conditions and reacts to them. This module estimates the amount of traffic being forwarded by each node and in order to minimize the interference that monitoring traffic may have on the applications running on the mesh. To this end, BEACON message propagation is either delayed proportionally to the traffic being forwarded by the node or delayed to a maximum time if the node is forwarding latency sensitive traffic. Indirectly, this forces the BEACON message quality from nodes that are forwarding traffic to drop and, consequently, nodes will not choose them as best next hops. Figure 4 illustrates this adaptive behavior. The cluster-head and subsequent nodes are going to choose the node with higher BEACON quality as its best next hop. In case the node is participating in the routing of a latency-sensitive stream, the monitoring traffic is going to interfere directly with the stream traffic. To avoid this, nodes that are actively forwarding application data (with a bold circle), will delay the BEACON propagation which causes a passive quality decrease in the nodes' routing table. The cluster-head and subsequent nodes will then choose another neighbor to send the monitoring data, forcing the monitoring traffic route to take a disjoint path from the application data traffic route.

D. Multiple Gateways

The presence of multiple gateways may cause the dissemination of redundant BEACON messages, given that nodes select a single gateway. To limit the flooding of redundant messages, every node receiving more than one BEACON message will only rebroadcast it if the received message has higher quality than other BEACON messages. In case of equal quality, the one with lowest hop count is preferred.

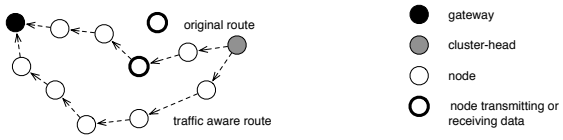


Figure 4. BEACON message delayed propagation.

IV. EVALUATION

To assess the performance of the monitoring system, we resort both to simulations and to a prototyoe of testbed that we developed to that purpose. We compared the performance of the mentioned system with a simple SNMP setup based on the OLSR routing protocol.

A. Simulation Environment

The *ns-2* simulator was used to evaluate the system's performance. The WMN is comprised of 100 static nodes, deployed randomly in a 500m x 800m space, with a transmission range of 100m. The propagation model used is the Two Ray Ground with the 802.11 MAC. For each test, we have generated 10 different scenarios using the *BonnMotion* tool[15], and each simulation was executed for 5 minutes. In each scenario, the gateway was placed in the bottom left corner of the space, in order to simulate a scenario where data traverses a large number of hops.

We conducted a series of experiments in order to assess the system's functionality and test the resulting:

- Clustering;
- Monitoring traffic and its delivery ratio;
- Route stability;
- Impact on multimedia streams;
- *Adaptive Information Transfer* (AIT).

1) Clustering: Both clustering methods (circular and semi-circular) were tested in the mentioned scenarios. We have measured the average number of clusters created and the average number of members per cluster (Figure 5). As expected, the circular method, which includes nodes in all of 2-hop neighborhood of the cluster-head, creates less clusters and those clusters have more members than the semi-clustering method, which creates more clusters due to the fact that each node only joins the cluster if it's cluster-head is closer to the gateway than the node.

The time of cluster establishment, measured as the interval between the election of the cluster-head and the last node to join the cluster, is relatively the same in both methods: 9.11ms for the circular and 8.15ms for the semi-circular clustering.

2) Monitoring traffic and delivery ratio: In order to assess the system's performance, we performed stress tests, by increasing the rate at which the nodes send the monitoring information to the gateways until the network is saturated. In these and the following tests that involve monitoring traffic, each node generates monitoring messages with 100 bytes.

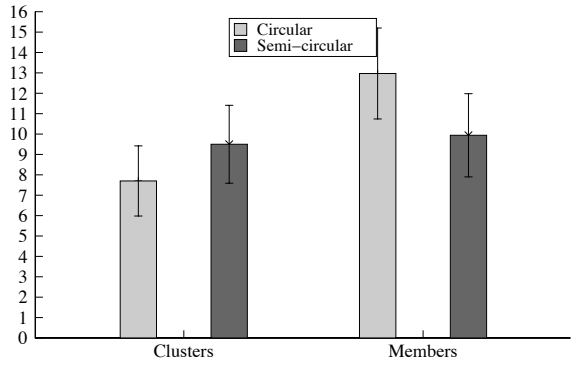
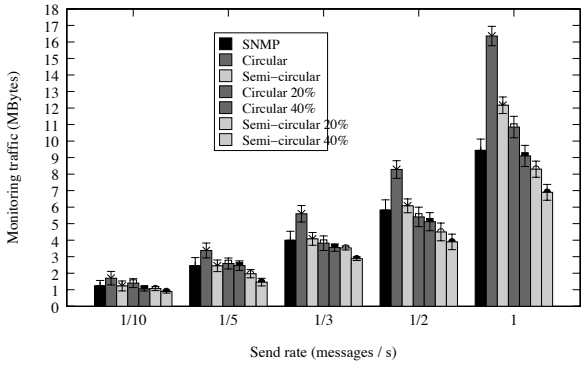
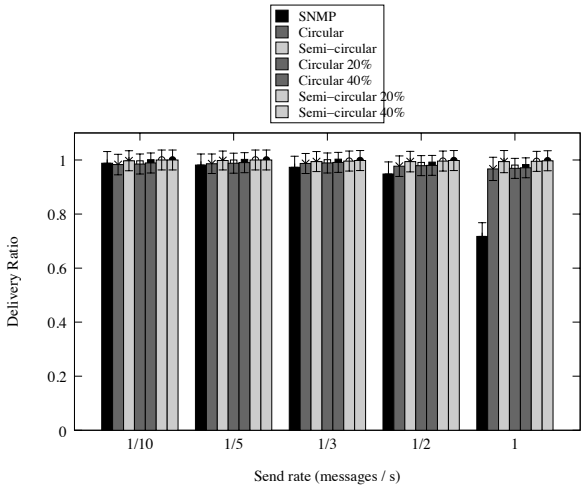


Figure 5. Clustering performance.



(a) Monitoring traffic.

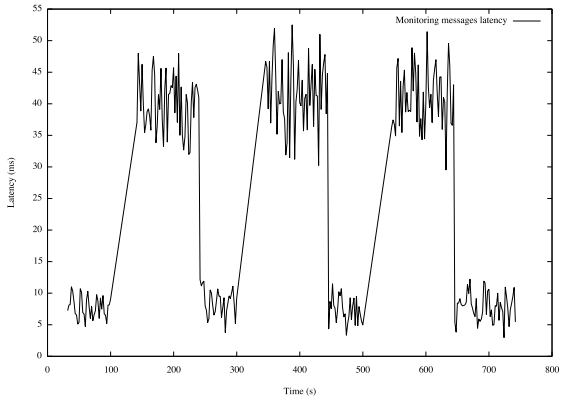


(b) Delivery ratio.

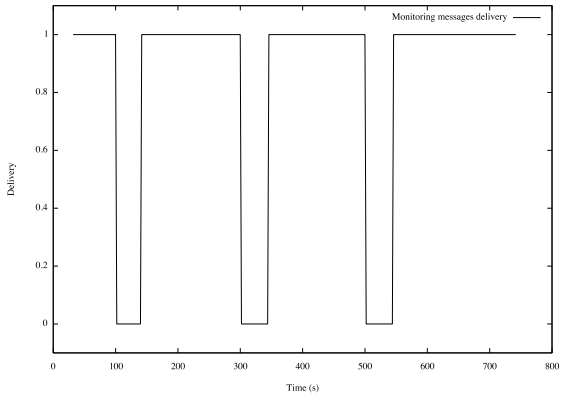
Figure 6. Monitoring traffic and delivery ratio comparison.

Figure 6(a) and Figure 6(b) show respectively, the monitoring traffic generated and the delivery ratio of monitoring messages for the three systems with variable send rates.

The delivery ratio remains high in both clustering methods, in the other hand, with SNMP, it drops significantly for a send rate of one message per second. Such drop is due to the fact that the monitoring traffic is not maintained locally,



(a) Monitoring messages' latency.



(b) Monitoring messages' delivery (*Note: 1 is delivered, 0 is not delivered*).

Figure 7. Route stability.

as it is with our system, and because OLSR generates more signaling traffic than the HELLO and BEACON messages, which in turn are going to occupy the channel and allow less information to be transmitted. There are differences between the traffic generated in both clustering methods, in the circular method, nodes that have cluster-heads further away from the gateway than themselves, have to send the monitoring information to the cluster-head, then the cluster-head sends it back to the gateway, creating a back and forth effect that consumes more traffic.

Although the metric of delivery ratio vs traffic is favorable to our system, this benefit comes at the cost of a higher end-to-end latency: the monitoring information generated in a node, proceeds to its cluster-head, which waits a period of time before sending the aggregated information towards the gateway (the default waiting period is a function of the node's send rate). Aggregation at the cluster-head level allows to apply compression schemes that further reduce the traffic sent to the gateway. Figure 6(a) also show the results when the aggregation function is able to reduce the size of monitoring information by 20% and 40%.

3) Route stability: To test the route stability, we tailored a scenario where we selected a random node a (3-hops away from gateway) and identified its best next hop towards the gateway denoted as node b. Afterwards, we forced node b to go down, and observed that node a chose node c (10-hops away from gateway) as its best next hop. Later, we forced node b up again and after some time, observed node a switching back its best next hop to node b.

In node a's perspective, when node b went down (Figure 7(b)), its BEACON quality started dropping and it chose node c as its best next hop. As node c was far away from the gateway than node b, the monitoring messages latency increased, as seen in Figure 7(a). When node b went up again, its as BEACON messages started reaching node a again and when they reached equal quality as node c's, node a switched its best next hop to node b again, due to gateway proximity factor.

In a loss free scenario, when node b goes down, node a, using a *stability threshold* of 2, would need 3 *epochs* to switch its best next hop to node c (assuming all of node c's BEACON messages are delivered). However, in a medium with interferences, BEACON messages will collide and switch time could be a little higher as shown in Figure 7(a): around 35 seconds.

4) Impact on Multimedia streams: To test the impact of the monitoring information traffic on multimedia streams, we simulated a VoIP call in the network using the G.729 codec, which uses 20ms frames of 20 bytes each. The stream uses a route of 3 hops, traversing nodes near the gateway.

To measure the call quality the following metric[16] is used:

$$\begin{aligned} R &= 94.2 - 0.024d \\ &- 0.11\,(d - 177.3)\,H\,(d - 177.3) \\ &- 30ln\,(1 + 15e) \end{aligned} \tag{3}$$

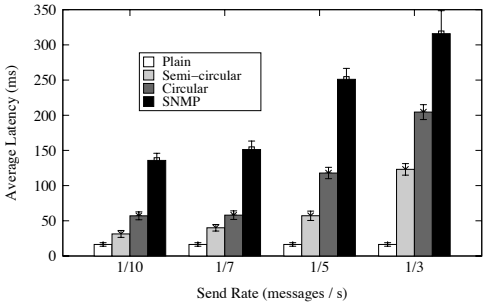
where:

- d = 25 + d_{buffer} + $d_{network}$ is the total ear to mouth delay (25ms), delay in the buffer and the network latency;
- e = $e_{network}$ + (1 - $e_{network}$) e_{buffer} is the total packet loss factored by a packet delay variation *buffer*) and;
- H(x) = 1 if x $>$ 0; 0 otherwise, is the Heaviside function.

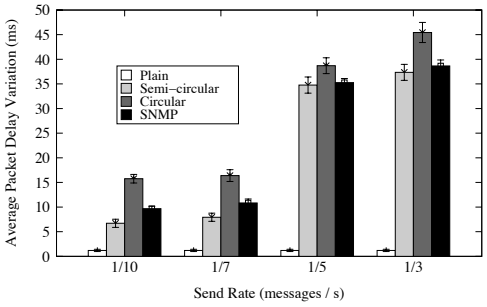
Quality is defined by the *R-score* (Eq. 3) which states that a score of 70 provides a medium call quality in VoIP.

We measured all relevant metrics for calculating the *R-score* of the stream in scenarios without monitoring traffic (*plain* scenario) and with variable send rates.

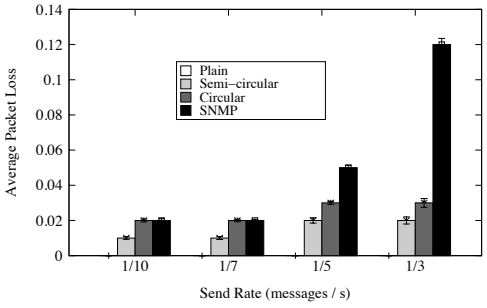
Depicted in Figure 8(a) is the latency degradation for various monitoring information send rates. Since OLSR exchanges far more information per second than our routing approach, the impact on latency is higher as send rates go higher, because the VoIP packets compete for transmission with both monitoring packets and routing packets. The semi-circular clustering provides the smallest negative impact on latency.



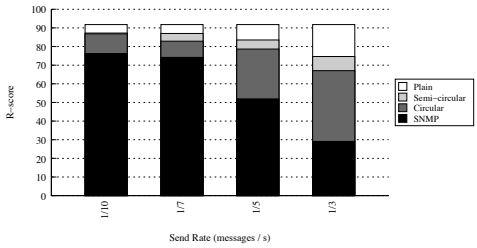
(a) Latency of VoIP call.



(b) Packet Delay Variation of VoIP call.



(c) Packet Loss of VoIP call.



(d) R-score of VoIP call.

Figure 8. Metrics of VoIP call.

In terms of packet delay variation (Figure 8(b)), the circular clustering method is the worst performer, since the burst of information sent by *cluster-heads* is higher due to having higher members per cluster.

Packet loss difference is negligible except for the send rate of one message every three seconds (Figure 8(c)). In particular for SNMP, the packet loss is higher due to the fact that all nodes route the monitoring packets directly to

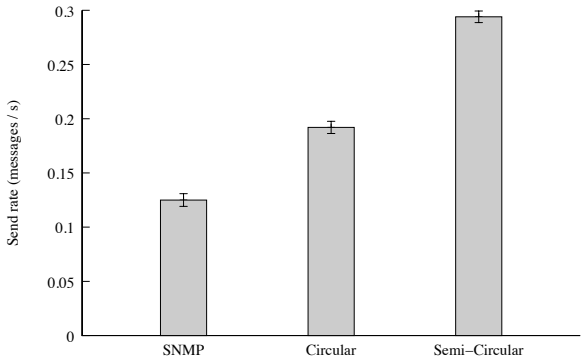


Figure 9. Maximum monitoring throughput comparison.

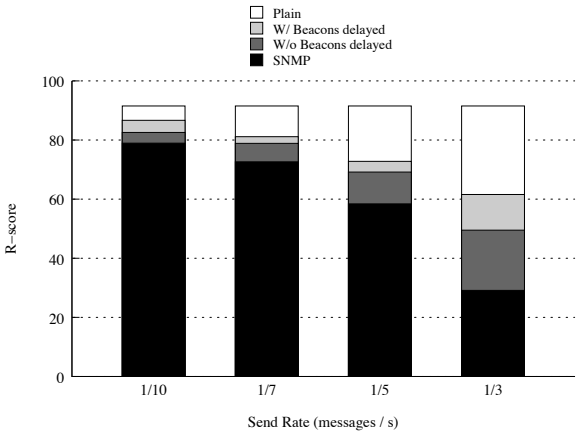


Figure 10. *R-score* of VoIP call.

the gateway, increasing neighborhood interference.

Figure 8(d) illustrates the combined effect of these factors, using the *R-score* metric previously described.

Maximum monitoring throughput: In this test, a VoIP call using the G.729 codec was placed horizontally in the middle of the network and we evaluated the maximum send rate possible without the call quality lowering the 70 threshold.

Analyzing Figure 9, we can see that a higher monitoring rate can be sustained by our architecture than with SNMP, without bringing the quality below the target threshold. The semi-circular clustering can achieve a send rate of one message every 3.4 seconds, without compromising call quality, while SNMP can only achieve 1 message every 8 seconds (a 42,5% increase in performance).

5) Adaptive Information Transfer: To test the Adaptive Information Transfer, a VoIP call, was created near the *cluster-head* with the purpose of interfering with the packets sent by it.

The adaptive mechanisms, based on the delay of BEACON propagation, changed the paths used by the *cluster-head* to reach the gateway, increasing call quality (measured mainly through the *R-score*), as we can see by the data presented

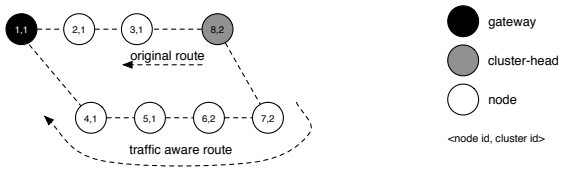


Figure 11. Testbed Topology for Adaptive Information Transfer tests.

Call Quality							
msg/s	AIT	Latency (ms)		Jitter (ms)		Loss	
	AIT	real	simul	real	simul	real	simul
1/10	off	12.10	(11.21)	5.42	(1.04)	0.015	(0)
	on	7.79	(10.96)	3.12	(0.95)	0.15	(0)
1/5	off	13.51	(11.12)	6.58	(1.04)	0.15	(0)
	on	11.99	(11.20)	3.58	(1.07)	0.15	(0)
1/3	off	26.66	(11.43)	7.334	(1.18)	1.1	(0)
	on	14.90	(11.25)	4.21	(1.09)	0.46	(0)
1	off	43.89	(11.68)	29.02	(1.4)	4.4	(0)
	on	20.86	(11.44)	6.50	(1.27)	1.1	(0)

Table I
PERFORMANCE OF THE ADAPTIVE MECHANISMS IN THE LA FONERA TESTBED.

in Figure IV-A5.

B. Experimental Testbed

The experimental testbed was deployed in the IST-Taguspark campus, comprising of eight La Fonera+[17], equipped with one EEE 802.11b/802.11g wireless card, one LAN port and one WAN port, running the OpenWrt[18] 8.09 firmware. Our system was developed in python and the exchanged traffic between nodes was done via iperf (both packages included in the OpenWrt repository).

All devices were scattered in order to achieve the maximum number of hops in the minimum amount of space. The reduced space for deployment restricted the amount of nodes that were possible to connect in the mesh network and the spectrum occupation by the IST-Taguspark own wireless infrastructure decreased the number of tests that could be performed in such conditions. To test the adaptive information transfer, we used the topology in Figure 11.

We let the clusters emerge, and the transfer of monitoring information start before node 8 transmitted during two minutes a 64kbit/s stream. The quality of the stream was measured during those two minutes either with the mechanism on and off, and with different monitoring information send rates. The results are presented in Table I, being the effects of the AIT mechanisms clearly visible.

We have also replicated the experiment on the simulator, using a scenario that mimics the deployment setting. The results for the simulation are depicted between parenthesis in the left column, for compairison purposes. Although experimental results differ from the simulated ones, reflecting the well known limitations of the simulation models, the relative performance is similar: the stream is less affected when traffic from the *cluster-head* is diverted to other paths.

V. Conclusions

In this work we have proposed an architecture to monitor the nodes of a WMN. Our solution combines different functionalities: it is based on semi-circular clusters that optimize the routing of monitoring information and implements adaptive mechanisms that minimize the impact of the exchange of monitoring information on the streams that are present in the network. We have evaluated our architecture and associated protocols running extensive simulations and using a experimental testbed with 8 La Fonera+ devices.

Results show that the proposed solution, in our experiments, can reach a 42,5% increase in monitoring information throughput without affecting the quality of service of an ongoing VoIP call. As future work it would be interesting to test several aggregation functions to reduce the amount of information that needs to be transferred in the network.

Acknowledgments

This work was performed at INESC-ID and partially supported by the Redico project (PTDC/EIA/71752/2006) and by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. Parts of this work have been performed in collaboration with other members of the Distributed Systems Group at INESC-ID, namely, José Mocito.

References

[1] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks ISDN Systems*, vol. 47, no. 4, pp. 445–487, 2005.

[2] A. Hamidian, C. Palazzi, T. Chong, J. Navarro, U. Korner, and M. Gerla, "Deployment and evaluation of a wireless mesh network," *Advances in Mesh Networks*, 2009.

[3] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Comput. Commun.*, vol. 30, no. 14-15, pp. 2826–2841, 2007.

[4] T. Anker, D. Bickson, D. Dolev, and B. Hod, "Efficient clustering for improving network performance in wireless sensor networks," in *EWSN'08: Proceedings of the 5th European conference on Wireless sensor networks*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 221–236.

[5] M. Schoffstall, M. Fedor, J. Davin, and J. Case, "Simple network management protocol (snmp)," United States, 1990.

[6] R. Raghavendra, P. Acharya, E. Belding, and K. Almeroth, "Meshmon: a multi-tiered framework for wireless mesh networkmonitoring," in *MobiHoc S3 '09: Proceedings of the 2009 MobiHoc S3 workshop on MobiHoc S3*. New York, NY, USA: ACM, 2009, pp. 45–48.

[7] F. Huang, Y. Yang, and L. He, "A flow-based network monitoring framework for wireless mesh networks," *Wireless Communications, IEEE*, 2007.

[8] H. Kazemi, G. Hadjichristofi, and L. A. DaSilva, "Mman - a monitor for mobile ad hoc networks: design, implementation, and experimental evaluation," in *WiNTECH '08: Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. New York, NY, USA: ACM, 2008.

[9] K. N. Ramach, E. M. Belding-royer, and K. C. Almeroth, "Damon: A distributed architecture for monitoring multi-hop mobile networks," in *In Proceedings of IEEE SECON*, 2004.

[10] N. Scalabrino, R. Riggio, D. Miorandi, and I. Chlamtac, "Janus: A framework for distributed management of wireless mesh networks," in *Proceedings of the 3rd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2007.

[11] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001, pp. 329–350.

[12] F. Sailhan, L. Fallon, K. Quinn, P. Farrell, S. Collins, D. Parker, S. Ghamri-Doudane, and Y. Huang, "Wireless mesh network monitoring: Design, implementation and experiments," in *Globecom Workshops, 2007 IEEE*, Nov. 2007, pp. 1–6.

[13] S. Nanda and D. Kotz, "Mesh-mon: A multi-radio mesh monitoring and management system," *Computer Communications*, vol. 31, no. 8, pp. 1588–1601, 2008.

[14] D. Johnson, N. Ntlatlapa, and C. Aichele, "A simple pragmatic approach to mesh routing using batman," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa*, 2008, p. 10.

[15] *BonnMotion a Mobility Scenario Generation and Analysis Tool*, University of Bonn, Jun. 2009.

[16] R. G. Cole and J. H. Rosenbluth, "Voice over ip performance monitoring," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2, pp. 9–24, 2001.

[17] "Fon," http://www.fon.com.

[18] "Openwrt," http://www.openwrt.org.