

# Wireless Mesh Network Monitoring: Design, Implementation and Experiments

Françoise Sailhan<sup>1</sup>, Liam Fallon<sup>2</sup>, Karl Quinn<sup>2</sup>, Paddy Farrell<sup>2</sup>, Sandra Collins<sup>2</sup>,  
Daryl Parker<sup>2</sup>, Samir Ghamri-Doudane<sup>3</sup>, Yangcheng Huang<sup>2</sup>

Ericsson Ireland Research Center

sailhan@ieee.org, {FirstName.Surname}@ericsson.com<sup>1</sup>, <sup>3</sup> samir.ghamri-doudane@lip6.fr

LM Ericsson Ltd., Cornamaddy road, Athlone, Co. Westmeath, IRELAND

## Abstract

*Mesh networks conveniently complement infrastructure-based networks, allowing devices to spontaneously form a network and connect with other networks. However, effective service provisioning requires a network monitoring solution with adequate support for topology data dissemination and diagnosis, due to the underlying network dynamics and the absence of pre-existing network infrastructure. In addition, monitoring systems have to face a number of challenges relating to autonomy, robustness and scalability. To this end, we have created a self-organised management overlay that homogeneously and dynamically organises devices into a cluster-based hierarchy on which monitored data is disseminated. Scalability of the monitoring system is achieved through the minimisation of the generated traffic, as a result of the optimised design. We have implemented the proposed monitoring system and evaluated through experiments the resulting performance.*

## 1 Introduction

In the last decade, mesh networks have been drawing considerable attention from operators and service providers due to their potential for extending the coverage of public hot-spots, corporate buildings or large-scale urban areas; enabling savings on cabling, deployment and maintenance costs. In this case, mesh networks represent a continuation of the fixed/wireless networking infrastructure (Core and Radio Access Network), with users being expected to demand similar services, e.g., browsing, email, multimedia computing, collaborative networking applications. From a network management perspective, the challenge related to the deployment of mesh networks lies in providing effective service provisioning despite the unreliability caused by unpredictable addition, failure or removal of Network Elements (NEs), and also network merging or partitioning. This requires provision of adequate support for topology monitoring so as to enable applications, and/or network administrators, to react in a timely way to any topology change. Note that network monitoring is not restricted to this usage. On the contrary, network monitoring may also be used for (i) providing statistics to pinpoint the sources of network failure, (ii) verifying if the strategies adopted by (routing) protocols, applications or middleware perform well, and

(iii) locating potential bottlenecks so as to redimension the network. Traditional solutions for monitoring wire-line networks provide poor performance in mesh networks due to several reasons. First, unlike wired network, wireless mesh networks are characterised by the absence of underlying infrastructure; the network being maintained by the combined effort of the constituent hosts. Second, these hosts often operate under severe constraints such as limited bandwidth for example. Third, mesh networks experience significant signal quality fluctuation caused by unreliable physical medium, obstacles, interferences, hidden hosts and some varying conditions in the environment. To deal with the above limitations, we propose a low overhead monitoring architecture specifically customised for mesh networks. The key idea behind our approach is to automatically organise all nodes into a hierarchy of clusters dedicated to the delivery of monitoring data, and to dynamically and autonomically reconfigure the cluster-based structure in the presence of network dynamics (e.g., node failures). The rationale for introducing this novel self-configurable cluster-based monitoring system is twofold. First, this monitoring system does not necessitate any pre-existing network infrastructure; instead it is automatically deployed and requires minimal human intervention. Second, the adopted cluster-based hierarchical model for collecting data is both appropriate and message efficient since intermediate levels of the hierarchy can conveniently collate data (possibly producing a digest) before forwarding it to upper layers in the hierarchy. To achieve high availability and graceful degradation, we distribute the monitoring system functionality in a scalable way; nodes dynamically cooperate and form a monitoring overlay which adapts to the underlying network characteristics. In addition to the above, the monitoring requirements are complemented with a lightweight detection and diagnosis scheme which collects information related to the local network topology so as to keep to a minimum the traffic generated whilst also requiring minimal memory and computational capabilities. We further build our monitoring system upon an event notification service which supports an asynchronous one-to-one, one-to-many and many-to-many interaction model, which optimises the notification and monitoring system for inter-connecting and monitoring the loosely-coupled components which form mesh networks. We have both implemented the proposed monitoring system and demonstrated the robustness and scalability of our system through experiments. This

work is an important and novel component in designing automatic network management systems which reduce network operators' and service providers' OPEX (operating expenditure). The remainder of this paper is organised as follows. We first give an overview of related work (§ 2). We then introduce our monitoring architecture in more detail (§ 3) and demonstrate and evaluate its performance (§ 4). Finally, we conclude this paper with a summary of our results along with directions for future work (§ 5).

## 2. Related Work

Network monitoring includes 2 main steps, consisting of:

- a measurement phase, which lies in evaluating the state of NEs along with their performances,
- a gathering phase, which corresponds to the collection of the measurement data with the intended purpose of inferring the state of the overall network.

Several design choices differentiate network monitoring systems with regard to the two above phases. During the gathering phase, they refer to the behaviour (proactive versus reactive) and the organisation (centralised versus distributed) whereas during the measurement phase, they relate to the swiftness (active versus passive), and the type of communication adopted (broadcast versus unicast). In the following, we describe the above approaches in turn. With proactive monitoring, the system actively collects and analyses network states to detect past events and predict future events so as to maintain network performance. Typically, information is collected on a regular basis, providing a partial or complete picture of the network. This is especially important for time-critical traffic. With reactive monitoring, the system collects information relating to the network states on-demand, i.e., only when it is requested. A particular category of reactive system refers to event driven monitoring systems whereby data is emitted when an event of interest occurs; information is then transmitted typically in a summarised format. Due to their complementary nature, proactive and reactive monitoring system should not be seen as competitive. Network monitoring systems can also be classified according to the network organisation: centralised or distributed. A centralised model is characterised by a unique data collecting point which gathers all the information from a set of agents limited to the role of dumb data collector. Due to the concentration of data processing and traffic on a single entity, the monitoring system scales poorly. In contrast, distributed monitoring systems are typically organised into a hierarchy (a multi-layer pyramid) comprising top-level and mid-level managers as well as monitoring agents at the bottom layers. Such top-down delegation improves the scalability of the monitoring system and may be enriched using vertical delegation which involves the cooperation between nodes located at the same level of the hierarchy. An alternative approach of handling network monitoring is built upon mobile code agents. However, this involves a non trivial overhead on routers [10].

Network measurements may be categorised according to their behaviour which is either active or passive, and

broadcast- or unicast-based. Passive measurement [2, 5] consists of analysing the network traffic by capturing and examining individual packets passing through the monitored NE, allowing for fine-grained operations such as deep packet inspections. In contrast, active measurement [8, 6] involves the injection of probe packets into the network. This requires active participation from all actors in the network and therefore can be used to monitor network sanity (i.e. the presence/absence of a resource and its behaviour can be detected regardless if this resource is used). Active and passive monitoring have distinct advantages and drawbacks. Whereas active monitoring causes competition between application traffic and measurement traffic, passive monitoring avoids the problem of contention and also avoids using stale data. On the other hand, contrary to passive monitoring, active monitoring improves fault tolerance, incurs minimal delays in obtaining measured data, and has a global relevance in the sense that it is independent of specific applications.

## 3 Proposed Monitoring Architecture

We organise the network nodes into a cluster-based hierarchical structure which is then used to implicitly deliver monitoring information; a cluster head being a node that is elected to coordinate and publish information relating to the topology of the cluster(s) and cluster elements that are under its supervision and to the neighbouring cluster heads. This approach keeps to a minimum the information disseminated across the network. The basic building block of the Monitoring Architecture is a Topology controller (TC). A TC corresponds to a cluster head which is responsible for building and maintaining a local view of its cluster(s) and the logical connections between itself and the neighbouring cluster heads. A TC further acts as a mediation point and may also aggregate and correlate data for the cluster. TCs co-operate with each other to build and maintain the network topology (i.e., the entire network, or a particular level or a sub-tree of the topology). A full topology consists of an aggregation of all local topology information for the TCs in that topology. Apart from monitoring, our platform is also required to support real time queries related to network topology. The monitoring service gathers measurement information from all, individual, or even components of NEs. When the service on a NE receives a request for information, it passes the request to all its subordinate nodes in parallel. Each subordinate node, in turn, passes the request down the tree until the bottom of the tree is reached. The data is then read by each node and passed up to the superior nodes where it is aggregated and again passed upwards until the top of the tree is reached. Users can specify a scope and filter on a topology request to limit the amount of data returned. Users of a topology such as an external system (e.g. a network management system or a particular application) may register for topological events. Notification subscriptions may vary in scope from an entire topology, to a sub-tree of a topology, or a single NE.

In order to monitor the network, each node holds 4 main services (Figure 1): a measurement (3.2), directory (3.3), event notification (3.4) and grouping (3.5) service. The measure-

ment service gathers measurements (i.e., information relating to the topology) which are archived by the directory service. The grouping service organises the NEs in order to collect monitoring information; the event notification service is used to support the communication inherent to the grouping or monitoring of the nodes. Before delving into the design details of the above services, we firstly need to introduce the underlying network environment which we presuppose.

### 3.1 Network Architecture

We consider a mesh network whereby nodes are stationary and are connected with each other by multi-hop wireless links, as enabled by the IEEE 802.11 technologies. Each node in the network operates as a router, forwarding each others' data packets; with the discovery and the maintenance of the routes being handled by the OLSR routing protocol [3, 4]. The mesh network is attached to a wide area network *via* one or more on-line gateways which correspond to a node equipped with several radio interfaces. Apart from providing access to a WAN, gateways are expected to provide special configuration capabilities relating to the IP host's configuration (e.g., allocating unicast, broadcast and multicast address), and domain name resolution (mapping from address to names and vice versa). These functionalities are offered by a so-called *zero-conf* protocol, which generally provides a user-configurable network infrastructure. In addition to the above, a gateway should act as a Foreign Agent, implementing mobile IP functionalities so as to support user's macro-mobility. Further, in order to accommodate user's micro-mobility, we leverage the multi-hop communication facilities (multi-hop routing and forwarding techniques as enabled by the OLSR routing protocol), which remove radio coverage limitation. However, instability of the mesh network topology may lead users to experience significant quality fluctuation. This calls for a solution which adequately measures and monitors topology changes, enabling diagnosis and repair.

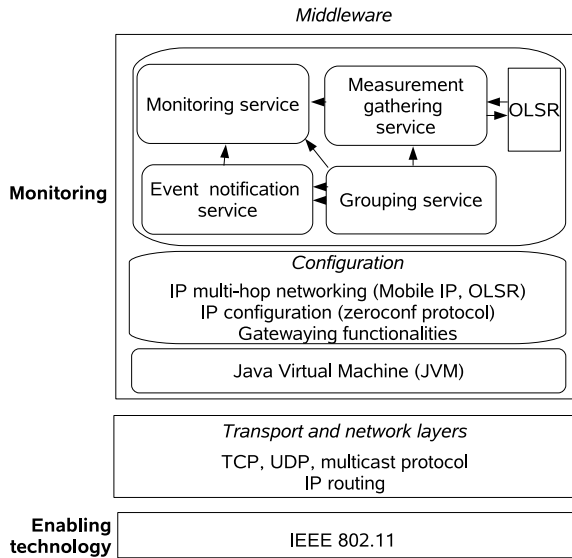


Figure 1: Host Architecture

### 3.2 Measurement gathering

We identify two basic mechanisms to monitor mesh networks and describe network changes. The first one stems from relying on the SNMP-inherited MIB [1, 9] and traps mechanism whereas the second relies on listening to the OLSR traffic. Recall that OLSR is a link state protocol optimised to operate in mesh network and that derives from OSPF [7]. Therefore we take the second approach, following the recommendations presented in [10], which showed the superiority, in term of reliability and robustness, of passively listening to the link state advertisements and updates, which are propagated by OSPF.

Remembering that the OLSR protocol :

- represents the topology as a graph,
- defines the concept of multipoint relays (MPR) as a set of selected nodes; those nodes forwarding link state information<sup>1</sup> intended for diffusion into the entire network on behalf of the other nodes.

Therefore, we model topology dynamics as a sequence of changes to the underlying OLSR graph, wherein a change represents an addition/removal of a vertice(s)/edge(s) to this graph. A device is treated as a vertex; the backbone nodes, i.e., the nodes characterised by a high connectivity degree, designating MPRs. A link between two devices is identified as an edge. In order to reconstruct the routing table of any given set of routers at a given point of time, the measurement module communicates with the OLSR protocol to receive updates on the network topology, with the directory service archiving the communicated information.

### 3.3 Directory Service

The directory service aims to (i) archive information that has been collected relating to nodes and (ii) map nodes with their corresponding information structure, e.g., role, capabilities. The directory service further allows nodes to be looked up using a graphical interface. In order to allow an efficient lookup, each node is characterised by a permanent identifier that can trace each host lifecycle. This unique identifier is attributed either by a network administrator or generated using the MAC addresses, as layer 2 addresses are known to be globally unique and readily available in devices of interest. This prevents any inconvenience caused by the plausible re-attribution of an IP address by the zeroconf protocol due to the addition and removal of hosts, or the re-arrangement of network segments. In addition to its identifier, each node is characterised by its role (e.g., bridge, gateway, access point), capabilities, location (expressed as a number of hops), and up-time. A (management) user may rely on the directory service to obtain a snapshot of the topology enriched by the above information concerning the nodes. The directory service also provides a management user the ability to do historical analysis. This allows determination of the end-to-end paths in use within the OLSR domain at any given time,

<sup>1</sup>Link state information are limited to the 1-hop away nodes however, additional available link state information may be possibly utilised for redundancy purpose.

and determination of how those paths change in response to network events within a specified period of time.

### 3.4 Event Notification

The main reason that motivated our choice for an event-based communication mechanism is the asynchronous interaction model promoted by an event system, which renders these latter particularly suitable for interconnecting the loosely-coupled monitoring components that form a mesh network. Our event system derives from the well known publish/subscribe paradigm, in which:

- consumers express their monitoring demands to producers during a subscription process,
- producers transfer to subscribers the description of any event that has been triggered locally.

This service handles event notification (hereafter simply referred to as notification) internally between the components deployed on the host in an asynchronous manner with no guarantee on the notification ordering. The subscription

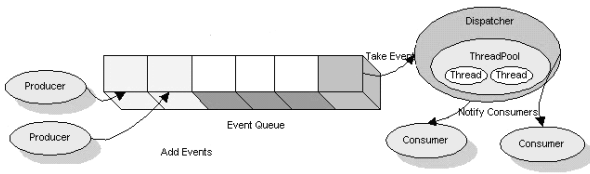


Figure 2: Event dispatcher using thread pool

and notification format derives from the Java event model<sup>2</sup>; any state associated with an event notification being encapsulated in a java object. As described in Figure 2, event notification is organised around an event notification queue manipulated by an event dispatcher. More precisely, this queue is populated with the produced notifications; each notification being extracted from that queue by the dispatcher with the intended purpose of notifying the subscribed Java-technology listeners. During the design phase, our main consideration was to ensure a *thread safe* queue, as the producer may be adding events from multiple threads. Our initial approach consisted of providing a single dispatch thread. However, this requires an event consumer to guarantee that the callback function completes as those functions are running in the same dispatcher thread. If the callback method blocks, then the dispatcher is blocked. Our improved mechanism which alleviates the chance of deadlocking and enables the support of a continuous flow of communication, lies in implementing the dispatcher as a pool of threads that handle events in a parallel manner. Applications or services that are using the notification service have to implement some event handlers and subscribe to the notification service. Our event notification service also handles notifications in a distributed way, i.e., between remote notification services. From a communication point of view, our distributed event notification consists of exchanging notifications and control messages (i.e., subscriptions and un-subscriptions) between

<sup>2</sup><http://java.sun.com/j2se/1.3/docs/guide/awt/designspec/events.html>

producers and subscribers through a collection of intermediate routers. A router is a device which holds the notification service. The notification system maintains consistent connection with adjacent routers (1-hop away nodes); these routers constituting the potential candidates for forwarding notifications. In practice, different way of communicating - one-to-one, one-to-many many-to-many<sup>3</sup> and reliable versus unreliable - may be selected by an event listener for notification depending on the requirements for a certain event. For the purpose of forwarding selectively notifications, each router holds a repository which encompasses a set of the consumer's subscriptions. This repository is also used to filter notification, i.e., to define if there exists a consumer that has subscribed for this notification.

### 3.5 Grouping

The grouping service on a NE cooperates with the grouping service on other NEs to build a topological structure made up of hierarchical clusters composed of NEs. Clusters are formed as follows. In order to guarantee a loop-free hierarchical structure, a NE in a level  $i$  cluster (with  $i \in [0, n]$ ,  $n$  referring to the upper-bound of the number of layers) is the cluster head from a level  $i - 1$  cluster. NEs in a level  $i - 1$  cluster elect a cluster head which represents the level  $i - 1$  cluster in the level  $i$  cluster. The hierarchy is self-configuring and re-forms dynamically and seamlessly whenever NEs are added, moved, or removed or if any of the parameters (e.g., QOS parameter or location) that influence it are changed during network operation. The mechanisms for the formation and supervision of clusters, along with the election of cluster heads are detailed below.

**Cluster formation** Cluster formation is triggered by the addition of a new node at any time. When a node starts-up, the node waits for a parameterised random time period<sup>4</sup> and then sends a (multicast or broadcast) cluster head query. This query is sent to the neighbouring nodes, i.e., to the  $j$ -hops away nodes ( $j$  being set to 1 by default). In a parallel way, the new node initialises its topology data table and inserts a table entry at level 0 indicating that the node is a member of a cluster at that level and has no cluster head. If after a parameterised interval time no reply is received, the node promotes itself as cluster leader. Alternatively, after checking its parameters, a cluster head either refuses or admits the node in question into its cluster. In the second case, the cluster head reserves a place for the member and responds with an acceptance message. Note that different criteria (e.g., arrival order of the query/acceptance message, QOS parameters, load, location) may be used to select a cluster head or refuse the insertion of a member in a cluster. Note also that the clustering is not restricted to network monitoring. On the contrary, clustering can also be designed on a per-application basis, grouping NES with

<sup>3</sup>In practice, one-to-many and many-to-many delivery of control messages to producers, as well as notifications to consumers, is handled by a dedicated multicast protocol.

<sup>4</sup>Delaying the emission of the query avoids a message deluge if nodes join simultaneously.

common functionalities /application use, for e.g. efficient data or resource sharing.

**Cluster supervision** A cluster head periodically polls its members to ensure that they are still active. If any member does not reply within a specific time period, the cluster removes that member from its cluster, updating its member list. Similarly, a cluster member expects to be polled by its cluster head periodically. It therefore assumes its cluster head has disappeared in absence of receipt of polling messages; such assumption triggering a cluster head promotion. In addition to the above, when a cluster member or leader is expecting impending removal or stoppage, it sends a removal message to its cluster head (resp. all its cluster members).

**Cluster Head Promotion** When a cluster head is promoted, it initialises a list to hold the addresses of all its cluster members. Initially, this list includes just the address of the local host. This promotion may result from the initial configuration of the grouping structure, or when the failure (resp. leaving) of a cluster head is discovered (resp. announced), or at the receipt of a cluster query. On reception of a cluster query, and if the monitoring node is a cluster member but not a cluster head at that level, then it can be inferred that there are two cluster members at the same level with no cluster head. The node then checks its cluster parameters (e.g., load, localisation) and, if the check permits, promotes itself to cluster head.

## 4 Assessment

This section presents a set of simulation tests, results, and analyses that aim to evaluate the performance of the proposed monitoring architecture. We conducted our evaluation considering a variety of networking scenarios including: different network sizes (10, 30, 50, 70, and 100 nodes), cluster sizes (0, 3, 5, and 7 NEs per cluster), variable enabled services (grouping service enabled/disabled) and topology changes (0% and 10%); topology changes being measured at the percentage of nodes that are not available, or have restarted, on average over a one minute period. We performed multiple passes and then averaged the results to reduce the impact of any incidental effects. The approximate average resources available to each node are as follows; CPU: 186.2 KHz with 204Kb, virtual memory 50Mb, and network bus speed of 1Mb/s.

In order to assess the effectiveness of our solution, we use 2 performance metrics, the robustness (Figure 3 and Figure 4) and the scalability (Figure 5 and Figure 6) of the solution. Robustness is defined as the ability of the monitoring system to withstand various loads induced by external applications. Scalability refers to the ability of the monitoring system to grow in a graceful and functional manner as the constituent dependencies change (e.g. number of NEs, load). These two performance metrics have been investigated in terms of CPU and memory usage, time delay, and bandwidth usage.

**Robustness** Figure 3 illustrates how CPU usage alters with increasing packets per node and per second varying from 0 up to 50 packets per second. We measured the CPU

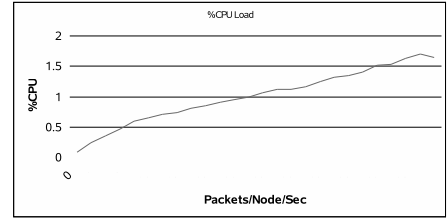


Figure 3: Average CPU Usages with different data rates

usage on each physical machine, with a sample rate of 1 second, and gathered and averaged the result among the 10 nodes that form the experimental networks. We can observe that the CPU usage increases linearly as the load increases; the change in CPU usage observed being approximately 1%, which is negligible. In addition, based on further measurement, we identified that the CPU increase levels out at a higher rate, and from that point<sup>5</sup> (approximately 1500 Packets per node and per second) onwards the performance slow down. Therefore, we can conclude that the system can withstand various loads and maintain robustness from a CPU perspective. We next complement this evaluation of the ro-

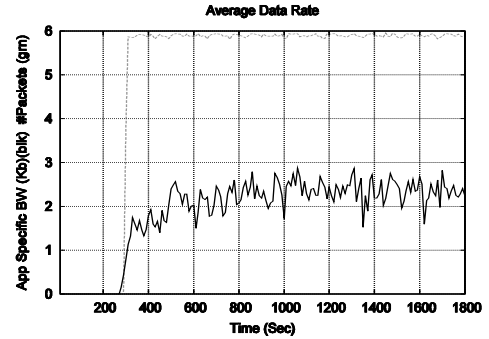


Figure 4: Data rate and packets number with 10-nodes network and 6 packets/sec/node

bustness with an analysis of the traffic in a 10 nodes experimental network for duration of 30 minutes with a sampling rate of 1 second. We observe a rapid increase in the traffic (average number of packets received by a node per second, and the size of the payload of those packets, see Figure 4) when considering the period [200 second, 350 second]. The explanation for this increase is that this region corresponds to the initialisation period; the node using bandwidth to join the cluster-based structure. Then a knee in the curve develops when the nodes grouping is completed and when the gathering of the monitoring information commences.

**Scalability** In order to highlight the effect on traffic of increasing the number of nodes in a network, we display in Figure 5 the average number of packets and payload bandwidth per node as the numbers of nodes increases for the combined full set of cluster groupings (0, 3, 5, and 7) and rate of topology changes (0%, 5%, and 10%). We observe that the traffic (numbers of packets received by a node and

<sup>5</sup>Such value is dependant on the limitations of the specification of the host machine.

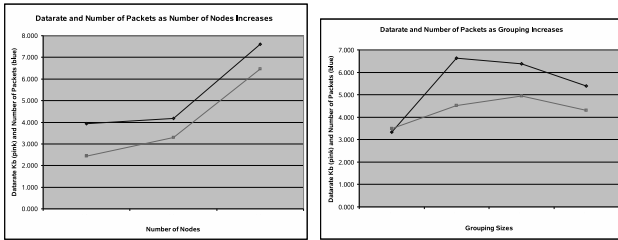


Figure 5: Data rate, packet number as number of nodes

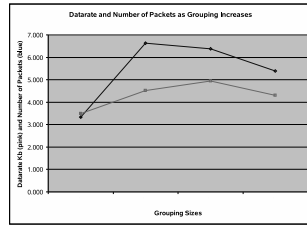


Figure 6: Data rate, packet number as the cluster size

the payload bandwidth) increases slowly; the fluctuations in data rate being approximately 3Kb and the fluctuations in numbers of packets being within 5. We next demonstrate in Figure 6 the effect of the grouping service (enabled or disabled<sup>6</sup>) and cluster size (0, 3, 5, and 7) on the traffic generated and throughput experienced by a node, with regards to a combination of number of nodes (30, 50, and 70) and disruption rates (0%, 5%, and 10%). There is an initial increase in both the data rate and number of packets when the grouping service is turned on, i.e. from 1 grouping to 3 grouping. When increasing the grouping size from 3 to 5 the data rate increases, however the number of packets reduces. Finally, further increasing the grouping from 5 to 7 reduces both the data rate and the number of packets. Further analysis into the effect of disruption rates with the grouping service disabled shows that the number of packets increases 4 fold when disruption rate increases from 0% to 10%. This 4 fold increase is derived from all samplings of number of nodes, i.e. 30, 50, and 70. When the grouping service is turned on under similar circumstances the effect of increasing the disruption rate from 0% to 10% is reduced. At best the activation of the grouping service can result in the number of packets for an average single node increasing 2 fold, which is half the number of packets without the grouping service. We can conclude that the grouping service not only provides a scalable solution for the proposed monitoring architecture but can also significantly reduce the effects of disruption. This reduction has an increased cost in terms of the number of packets sent, however it leads in turn to a more robust system.

## 5 Conclusion

This paper has presented a novel self-organising monitoring system for mesh networks which addresses the main requirements of those networks: scalability, autonomy and robustness. One of the key ideas behind this approach is the use of a cluster-based hierarchical structure created and maintained in a distributed manner by the proposed grouping service. This structure is then used to propagate efficiently the monitoring information, enabling that information to be filtered, aggregated and correlated so as to keep to a minimum the resulting bandwidth usage. The monitoring system is optimally based on an event-based communication mechanism which is suitable to interconnect nodes in a loosely-

<sup>6</sup>A grouping size equal to 0 signifies that the grouping service is disabled, i.e., that the network is flat.

coupled fashion. In addition, this paper has included simulation results for the proposed monitoring system which demonstrate its robustness and scalability. Indeed, CPU usage does not significantly alter as the packets per second is increased. In addition the traffic generated is very stable with regard to an increasing number of nodes or cluster size. Our experiments have also shown that activating the grouping service increases the data rate and number of packets, but that the grouping service can significantly reduce the effects of disruption within the network. This work has important application to the area of Network Management, in particular autonomic configuration, fault and performance management, leading to potential OPEX (operating expenditure) reduction for network operators.

**Acknowledgements** Authors would like to acknowledge the financial support provided by the Marie Curie Intra-European fellowship and the Celtic Initiative.

## References

- [1] J. Case, M. Fedor, M. Schoffstall, and al. A simple network management protocol (snmp), rfc 1157. <http://www.ietf.org>, May 1990.
- [2] A. Ciuffoletti and M. Polychronakis. Architecture of a network monitoring element. Technical Report TR-0033, CoreGRID Project, <http://dcs.ics.forth.gr/Activities/papers/netelement.coregrid-middleware06.pdf>, 2006.
- [3] T. Clausen and P. Jacquet. Optimized link state routing protocol (oslr). RFC 3626, <http://ietf.org>, October 2003.
- [4] T. Clausen, P. Jacquet, A. Laouti, and al. Optimized link state routing protocol, 2001.
- [5] Dressler, Falko, Nebel, and al. Distributed passive monitoring in sensor networks. In *IEEE INFOCOM*, 2007.
- [6] J. Moulrierac and M. Molnr. Active monitoring of link delays in case of asymmetric routes. In *IEEE ICN*, 2006.
- [7] J. Moy. Ospf v2, rfc 2328. <http://www.ietf.org>, April 1998.
- [8] H.X Nguyen and P. Thiran. Active measurement for multiple link failures diagnosis in ip networks. In *DAM workshop*, 2004.
- [9] R. Presuhn. Management information base (mib) for the simple network management protocol (snmp) networks, rfc 3418/std 0062. <http://www.ietf.org>, December 2002.
- [10] A. Shaikh, M. Goyal, A. Greenberg, and al. An ospf topology server: Design and evaluation. *IEEE Selected Areas in Communications*, 2002.