INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

WMM: Wireless Mesh Monitoring

Ricardo Pinto

Dissertação para obtenção do Grau de Mestre em
Engenharia de Redes de Comunicações

Júri

Presidente:	Prof. Doutor Rui Jorge Morais Tomaz Valadas
Orientador:	Prof. Doutor Luís Eduardo Teixeira Rodrigues
Vogal:	Prof. Doutor Rodolfo Alexandre Duarte Oliveira

October 2010

Agradecimentos

I would like to thank my advisor Professor Luís Rodrigues for all the advices and guidance during the duration of this work. I would also like to thank José Mocito for all the fruitful discussions.

This work was peformed at INESC-ID and was partially funded by FCT under the project "Redico" (PTDC/EIA/71752/2006) and INESC-ID multiannual funding through the PIDDAC Program funds.

Lisboa, October 2010

Ricardo Pinto

To my parents,

Carlos and Mila.

Resumo

Neste trabalho propomos um novo algoritmo de monitorização para redes em malha sem fios, baseado em *clusters* adaptativos. Os nós organizam-se automaticamente numa malha de *clusters* semi-circulares e a informação de monitorização de cada *cluster* é agregada pelos líderes dos *clusters* e posteriormente reencaminhada para a estação de monitorização. O algoritmo tem em conta a existência de fluxos de dados na rede, e tenta minimizar a interferência do tráfego de monitorização nesses fluxos. Para avaliar o desempenho do sistema recorremos a simulações e a uma bancada experimental que desenvolvemos para esse fim.

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. The proposed solution was evaluated using simulations and an experimental testbed deployed for the effect.

Palavras Chave
Keywords

Palavras Chave

Redes Sem Fios em Malha

Monitorização de Redes

Agrupamento de Nós

Avaliação de Desempenho

Keywords

Wireless Mesh Networks

Network Monitoring

Clustering

Performance Evaluation

Índice

List of Figures

List of Tables

Acronyms

WMN Wireless Mesh Network

1 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(Akyildiz, Wang, & Wang 2005; Hamidian, Palazzi, Chong, Navarro, Korner, & Gerla 2009).

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

1.1 Motivation

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. Unfortunately, network monitoring requires the exchange of information in the network and therefore is also a source of 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.

As it will be shown later in the thesis, one of the problems in the monitoring activity of a WMN is the overhead that it induces. The more frequent the nodes refresh their status information to the gateway, the more network resources are consumed and less quality of service is provided to application traffic.

To mitigate the above problem, this work proposes a new clustering algorithm, along with

a mechanism that tries to create disjoint routing paths for both monitoring traffic and application traffic, minimizing the impact and interference that monitoring information causes on application's traffic, namely multimedia streams.

1.2 Contributions

This work addresses the problem of monitoring in large scale Wireless Mesh Networks. More precisely, the thesis analyzes, implements and evaluates techniques to reduce the overhead of monitoring traffic in the network and its impact on multimedia streams. As a result, the thesis makes the following contributions:

- a novel clustering scheme: semi-circular, that reduces the amount of monitoring traffic sent by the nodes;
- a routing protocol that allows nodes to contact their gateway;
- a routing-based mechanism that diverts monitoring traffic away from active multimedia streams.

1.3 Results

The results produced by this thesis can be enumerated as follows:

- A prototype that integrates the clustering algorithm, routing protocol and a multimedia stream avoidance mechanism.
- A simulated and experimental evaluation of the implemented clustering algorithm, routing protocol and routing-based mechanism that avoids paths where multimedia streams are active.

1.4 Research History

This work was performed in the context of the Redico project (PTDC/EIA/71752/2006) The general initial objective of this work was to design a monitoring protocol specifically for

WMNs. We have identified the problem of the resource consumption by the monitoring traffic as a key problem to address in the thesis. In this context we decided to experiment the idea of designing a clustering scheme that could help in this regard. During my work, I benefited from the fruitful collaboration with the remaining members of the GSD team working on Redico, in particular with José Mocito.

1.5 Structure of the Document

The rest of this document is organized as follows. Chapter 2 provides an introduction to the different technical areas related to this work. Chapter 3 introduces the Adaptive Semi-Circular Cluster-Based Monitoring for Wireless MESH Networks and Chapter 4 presents the results of the experimental evaluation study. Finally, Chapter 5 concludes this document by summarizing its main points and future work.

2 Related Work

2.1 Introduction

This thesis addresses the monitoring issues in wireless mesh networks, with focus on minimizing the amount of monitoring traffic generated and its interference with application traffic.

This section starts by introducing the basic concepts and characteristics of a WMN (Section 2.2), followed by some of the most used routing protocols (Section 2.3), then a brief description on how to evaluate the performance of algorithms (Section 2.4) and a comparison of experimental testbeds that were set to evaluate performance on real world (Section 2.5). Finally there is an extensive survey of the existing monitoring systems for WMNs (Section 2.7).

2.2 Wireless Mesh Networks

Mesh networking has its roots in tactical military networks, comprised of nodes with multiple interconnections that stored and forwarded packets(Bing 2007). Attracted by the inherent survivability and robustness of mesh networks, the US Defense research agency DARPA funded several projects that support troop deployment on the battlefield. PRNET(Jubin & Tornow 1987) project was started in 1973 and was a multi-hop Packet Radio NETwork system that reached a size of 50 nodes, allowing some to be mobile. More recently, the IEEE (its 802.11 Working Group) has tackled the standardization for wireless mesh networks. The 802.11s standard(Wang & Lim 2008; Hiertz, Max, Zhao, Denteneer, & Berlemann 2007) specifies the physical (PHY) and medium access control (MAC) layers and a default mandatory routing protocol: Hybrid Wireless Mesh Protocol (HWMP), although it allows alternate protocols to be used. With the broad availability of WLAN hardware and small-scale, low-cost portable devices in the late 1990's, interest in these networks increased dramatically.

2.2.1 Characteristics

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 different nodes in the network(Akyildiz, Wang, & Wang 2005)(Hamidian, Palazzi, Chong, Navarro, Korner, & Gerla 2009). In detail, WMN should have the following properties:

Multi-hop wireless network A WMN extends the coverage of current wireless networks without sacrificing channel capacity, because intermediate routers forward each other's traffic and provide non-line-of-sight (NLOS) connectivity. Also, by correctly placing the nodes it can achieve an efficient frequency re-use.

Dynamically self-organized, self-configured and self-healing Each node that joins the network, automatically establishes connections to other nodes without previous configuration needed. Adding new nodes or even relocating them is as simple as plugging them on. The mesh is self-healing precisely because no human intervention is necessary for rerouting messages.

Resilience to device failures This can be achieved by rerouting the packets around the failed nodes. Since for a given pair of nodes, the probability of having two or more routes interconnecting them is high, if one (intermediary) node fails, the network will adapt and route the packets through different paths.

Highly scalable The WMN should be expandable, allowing to add more routers in order to support more clients and cover a wider geographical region. The scalability depends on factors such as the size of the network, its architecture, topology, traffic pattern, node density, number of channels, and transmission power, among others. When the system grows it is desirable to have more gateways, given that the lack of the appropriate number of gateways may cause traffic bottlenecks and reduce network performance.

WMNs have the potential to help users to be always online, anywhere, anytime. Moreover, the gateway functionality enables the integration of WMNs with several existing networks such as cellular, wireless sensor, WiFi, WiMAX and wired networks. Conventional nodes such as

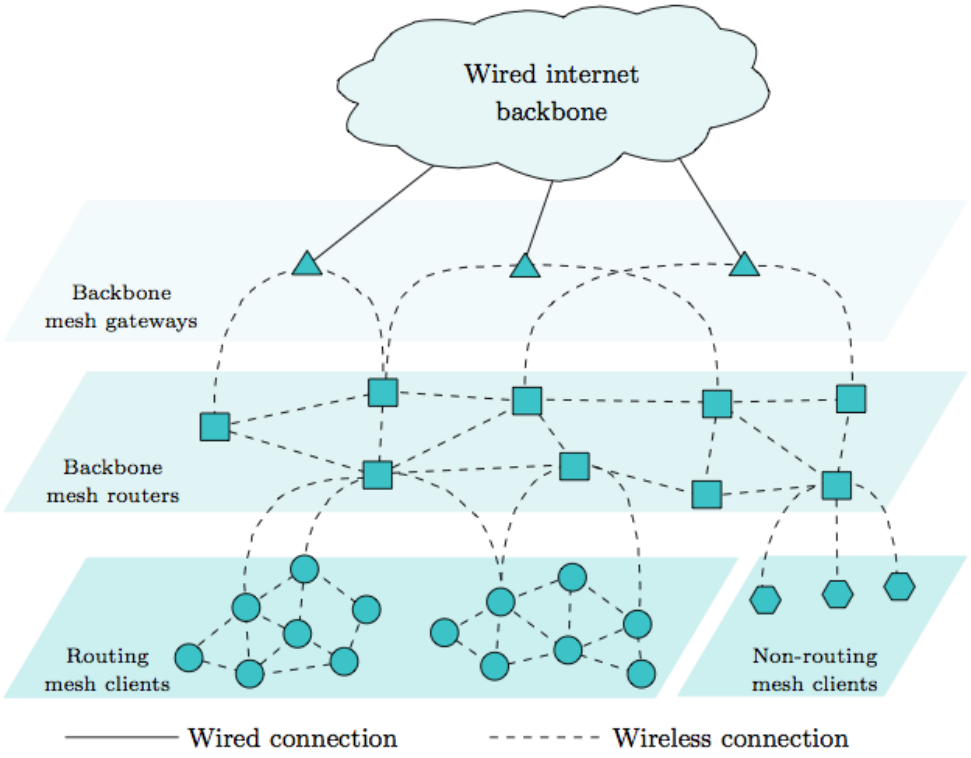


Figure 2.1: Layered WMN Architecture.

laptops, PDAs and phones equipped with wireless network interfaces can connect directly to the mesh network.

2.2.2 Network Architecture

Typical WMNs are comprised of two network components(Zimmermann, Schaffrath, Wenig, Hannemann, Gunes, & Makram 2007), as described below and illustrated by Figure 2.1:

Mesh Routers Mesh routers can be divided into gateways and backbone routers. Gateways are connected to the wired network and support transparent bridging and address learning. Backbone routers provide mesh services and may be a typical Access Point (AP) to which clients connect, or a dedicated infrastructure device that only enhances network coverage and capability (no client services).

Mesh Clients Mesh clients can be routing or non-routing capable. If the clients can forward network packets, then the network coverage can be further increased just by having more

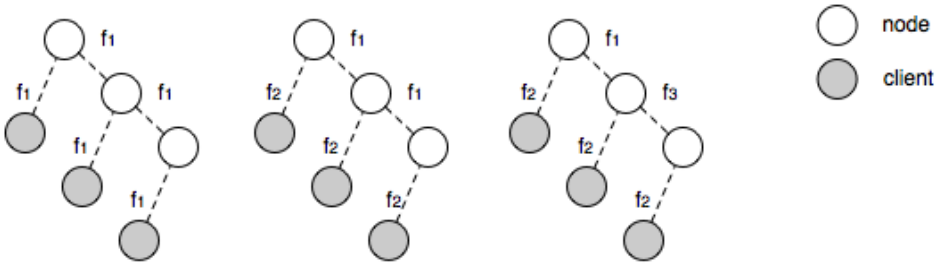


Figure 2.2: Mesh Configurations.

clients connected to the mesh routers and to each other.

The configuration of a WMN has to be carefully planned and there are three possible scenarios[1], depicted in Figure 2.2, according to the characteristics of the hardware:

- The first generation of WMNs uses only one radio channel to provide client access and backhaul service. This is the worst of all options since both clients and backhaul compete for bandwidth, and nodes have to listen, then send, and then listen again; this intermittent behavior affects network performance.
- The second generation adds one more radio, separating backhaul and client service networks. The non-overlapping of both networks frequency-wise, improves performance when compared to the first generation. Still, a single radio frequency is servicing the backhaul, traffic destined to external networks shares bandwidth on each hop leading to network performance degradation (not as severe as first generation).
- The third generation dynamically manages channels of all radios in order to avoid channel interference. In this 3-radio configuration, two radios provide the up and downlink of the backhaul and the other radio provides service to the clients.

2.2.3 Application Scenarios

A good example of a mesh application is home networking. Nowadays, most wireless coverage is provided by 802.11 WLANs. In these networks, the location of the Access Points can

[1] http://www.dailywireless.org/2004/07/15/the-mesh-debate/

cause zones without service coverage. An approach based on a WMN can provide a full and flexible house coverage.

The 802.11 WLANs are also widely used in enterprise offices, and the Ethernet cabling is a key reason for the high cost of the wired infrastructure in small and medium enterprises. APs can be replaced by mesh routers that provide client connectivity and cable-free backhaul.

WMNs can also be applied to transportation systems where remote in-vehicle video and driver communications can be supported; domotics, where WMN can help reduce the cost of wired networks to manage lifts, power, lights and AC; and security surveillance of areas such as parking lots, shopping malls and grocery stores. In addition to the above scenarios, WMNs can also be applied to emergency situations where the simple placement of wireless mesh routers can quickly establish connectivity(Kiess & Mauve 2007a).

2.3 Wireless Routing Protocols

WMNs are unstructured networks, and routing protocols have to account for mobility, dynamic changes in topology, and the unreliability of the medium. WMN nodes communicate with each other and routes to non-neighboring nodes have to be established. Routing protocols are responsible for discovering, establishing and maintaining such routes. Routing protocols for WMN are mostly based on protocols designed for mobile ad hoc networks. These can be classified in the following categories(Abolhasan, Wysocki, & Dutkiewicz 2004):

Proactive protocols construct the routing table periodically. Each node maintains a table representing the entire network topology which is regularly updated in order to maintain the freshness of routing information. At any given time, any node knows how to reach another node of the network. This approach minimizes the route discovery delay at the cost of exchanging data periodically, that consumes network bandwidth.

Reactive protocols construct the routing table on-demand. Nodes are not aware of the network topology and find routes by flooding the network with route requests. This leads to higher latency due to the fact that the route has to be discovered, but minimizes control traffic overhead.

Usually, reactive protocols are better suited in networks with low node density and static traffic patterns. Since the traffic patterns are static, the first request encompasses the route discovery, while the subsequent use the previous discovery to route traffic. On the other hand, proactive protocols are more efficient in dense networks with bursty traffic, due to the continuous exchange of topology information, reducing route discovery delay.

Hybrid protocols are a mixed design of the two approaches mentioned above. These protocols typically use a proactive approach to keep routes to nodes in the vicinity of the source, but for nodes beyond that area, the protocol behaves like a reactive one. The challenge is to choose from what point the protocol changes from proactive to reactive.

In the following paragraphs, we briefly survey some of the most relevant routing protocols for WMNs.

2.3.1 OLSR

The Optimized Link State Routing(Jacquet, Muhlethaler, Clausen, Laouiti, Qayyum, & Viennot 2001) is a proactive link state protocol for mobile ad hoc networks. It includes a number of optimizations that aim at reducing the cost of forwarding information in the network. In particular, for each node, a subset of neighbors, called the multipoint relays (MPR), are elected to forward announcements. The key idea behind the multipoint relays is to reduce the duplicate retransmissions in the same region.

Algorithm: each node selects its multipoint relay set among its one-hop neighbors in order to cover all two-hop neighbor nodes. Having a bidirectional link towards each of those neighbors is imposed by OLSR. Each node in the network periodically broadcasts information about its one-hop neighbors which have selected it as a MPR. Upon reception of this MPR selectors list, each node calculates or updates its routes.

The route is then a sequence of hops through MPRs. In order to detect bidirectional links with neighbors, each node periodically broadcasts HELLO messages, containing a neighbor list and their link status. HELLO messages contain the list of addresses of the neighbors to whom the node has bidirectional connectivity and the list of neighbors that are heard by the node. The contents of these messages allow each node to know the existence of neighbors up to two

hops and the selection of its MPRs, which are also indicated in the HELLO message. With information extracted from HELLO messages, each node can construct its MPR Selector table.

Each node broadcasts specific control messages called Topology Control (TC), in order to build the routing table for forwarding purposes. TC messages are sent periodically by nodes to declare its MPR Selector set (empty MPR Selector sets are not sent). TC messages are used to maintain topology tables for each node.

2.3.2 B.A.T.M.A.N

The Better Approach To Mobile Ad Hoc Networks(Johnson, Ntlatlapa, & Aichele 2008) is another proactive protocol for establishing multi-hop routes in mobile ad-hoc networks. Each node only maintains information about the best next hop towards all other nodes, which avoids unnecessary knowledge about the global topology and reduces the signaling overhead.

Algorithm: each node n broadcasts originator messages (OGM) to inform neighbor nodes about its existence. The neighbors rebroadcast the OGMs to inform their neighbors about the existence of node n, and so on. The network is therefore flooded with these small packets that contain the address of the original node, the address of the node rebroadcasting the packet, a TTL and a sequence number. Each node rebroadcasts the OGM at most once and only if it is received by the current best next hop towards the original initiator of the OGM. Thus OGMs are selectively flooded through the mesh network. Route discovery and neighbor selection depend upon the number and reliability of received OGMs. Sequence numbers are used to perceive the OGM freshness, thus any message received with a lower sequence number than the previous one is dropped. Nodes may alter the TTL of their OGMs to limit the number of hops the message traverses. This is useful for backbone nodes that are deployed only for improved connectivity and coverage purposes.

BATMAN outperforms OLSR on almost all performance metrics, due to the simplistic approach. By not collecting more information than it can effectively use, and by only getting information about its neighbors, nodes can compute routes in a more efficient manner. Routing overhead is significantly lower than OLSR, proving that sometimes complex approaches lead to less overall performance.

2.3.3 AODV

The Ad hoc On Demand Distance Vector(Huhtonen 2004) is a reactive protocol that creates and maintains routes only when they are requested. On a given node, the routing table stores only information about the next hop to the desired destination and a sequence number received from the destination, preserving the freshness of the information stored.

Algorithm: on demand, route discovery is done by broadcasting a route request message to the neighbors with the destination and sequence number. Each node that receives that request, increases its hop metric and updates its own table. The destination node upon receiving the message, sends a route reply back to the requesting node.

Reactive protocols like AODV tend to reduce the control traffic messages overhead at the cost of increased latency in finding new routes.

2.3.4 Hybrid Wireless Mesh Protocol

The task group 802.11s set by IEEE has been working for the mesh amendment(Wang & Lim 2008; Hiertz, Max, Zhao, Denteneer, & Berlemann 2007) to the 802.11 standard.

IEEE 802.11s adds a third type of network topology called Mesh Basic Service Set or MBSS. An MBSS can have the following three different kind of entities: a Mesh Station, which is a normal 802.11 Station with added functionality of path discovery and packet forwarding; a Mesh Access Point, which is a Mesh Station that provides client connectivity services and; a Mesh Portal, that interconnects the WMN with other non-802.11 networks like 802.3. The 802.11s specifies the Hybrid Wireless Mesh Protocol (HWMP), that runs on the MAC layer, as mandatory for path selection.

Algorithm: Nodes can use two modes of operation:

- On Demand Mode;
- Proactive Tree Building Mode;

Both can be used simultaneously in a hybrid mode.

The On Demand Mode is based on AODV(Huhtonen 2004), and as stated before, it works at the MAC layer. It has three different control packets: Path Request (PREQ), Path Reply (PREP) and Path Error (PERR). When a node n wants to send information, it initiates a PREQ broadcast that floods the network. Every PREQ has a sequence number that allows nodes to perceive its freshness. When an intermediate node receives the PREQ, it either creates or updates the path to the source depending on the sequence number; if there is no path, it simply forwards the request until it reaches the destination. Once the path is established it is cached and subsequent PREQs are not flooded within a small time frame. When the destination node receives the PREQ, it sends a unicast PREP back to node n.

In the Proactive Tree Building mode of HWMP, one of the nodes acts as the ROOT node r. The node r periodically broadcasts proactive PREQs. The address field of such PREQs is the broadcast address, thus every node that receives them send PREP back to node r. In this way, a proactive tree is build, and node r has the routing table filled with all possible destinations within the network.

In hybrid mode, both proactive and reactive components act concurrently.

2.4 Evaluation of WMNs

There are several different techniques that can be used to evaluate algorithms and protocols for WMNs, namely: theoretical analysis, simulation, emulation, and real-world experiments.

Theoretical analysis uses mathematical models to derive performance metrics such as signaling cost, throughput, latency, etc. Unfortunately, the complexity of most systems makes them difficult to analyze in this manner for most realistic scenarios.

In a simulation, the algorithms are modeled and evaluated in an controlled artificial environment. This ensures that the evaluation is repeatable and that the user has a tight control on all the parameters that affect the results. Furthermore, it allows to experiment with very large topologies in a cost-effective manner. However there is also a downside to simulations: the lack of realism, since all effects must be simulated, there are many external factors that are not considered by the model (interference, reflection, etc), and the results may not be representative of the algorithm behavior in a real-world scenario. Most results are qualitative due to the reasons explained above.

In an emulation, both hardware and software are designed to run under controllable laboratory conditions. An emulator provides a translation layer (usually done by software) from the emulated computer to the computer it is running on. Network emulation is accomplished by introducing a device that mimics the behavior of the environment being emulated. This device may be a computer that incorporates a variety of network attributes into the emulation model such as: RTT (Round Trip Time), available bandwidth, packet loss, duplication of packets, and packet reordering. The advantages of using emulation are repeatability, control over the environment, and a certain degree of realism that the laboratory provides. The costs per test are higher than with simulation but there are scalability bounds to it.

With real-world experiments, all parts of the system are tested under the same operational conditions for which it has been designed to operate. Thus, this approach limits the possibilities of making erroneous or inaccurate assumptions about the impact of external factors. Real-world experiments provide more feedback than simulations or emulations. Furthermore, they are the best way to show that the tested system indeed works as intended. However these experiments lack the repeatability and their scalability is limited due to hardware costs and deployment manpower requirements(Kiess & Mauve 2007b).

2.5 WMN Testbeds

A testbed is a framework which supports testing, comparison and evaluation of algorithms and protocols in the real world. Below we refer to some examples of testbeds that have been deployed to study WMNs.

2.5.0.1 Roofnet

Roofnet(Aguayo, Bicket, Biswas, Couto, & Morris ; Bicket, Aguayo, Biswas, & Morris 2005) is a large scale WMN experiment from MIT that uses about 50 nodes in apartments (few nodes are gateways), scattered to ensure that the longest routes are four hops long. The mesh routers are small mini-itx motherboards with a 802.11b/g card. Roofnet uses SrcRR as the routing protocol. The goal of the project is to provide Internet access to the students - nodes are deployed at their apartments. All nodes are running on the same channel, hence the network is a first generation. Distance, SNR (Signal to Noise Ratio), transmission rate, and the packet

loss are measured by the software running on the mesh nodes. Roofnet is a good example of real-world experiment since it is widespread over Cambridge and provides Internet connectivity to about 50 households on a day to day basis. However since Roofnet's propagation environment is characterized by its strong Line-of-Sight (LOS) component, it does not model a typical WMN since it does not account for obstacles and NLOS environments. Some academic testbeds model this behavior precisely because they are deployed inside University buildings.

2.5.0.2 UCLA Testbed

In the University of California, Los Angeles a testbed comprised of one gateway, four mesh routers (only one of them provides wireless access to clients), and a variable number of clients has been deployed(Hamidian, Palazzi, Chong, Navarro, Korner, & Gerla 2009). The nodes are laptops, which increase the cost of deployment, and act as clients or mesh routers that communicate on the same channel (this is a first generation network). The gateway is connected to a FTP server and a streaming server, that are used for testing purposes. The technology used to build the WMN is the Mesh Connectivity Layer (MCL) from Microsoft, an open source tool that implements a modified version of the Dynamic Source Routing (DSR) protocol. The tests were done focusing on the performance of multimedia applications. Metrics were obtained when a flux of data traversed the nodes, such as the packet delivery ratio per hop and the latency caused by the number of hops between source and destination. Some experiments were also done to test the Quality of Service (QoS) provided by the 802.11e for wireless networks (parameters such as contention window values and inter-frame space number can be altered to differentiate service flows). The testbed is too small to correctly test scalability, since the tests were done solely for the deployed network topology.

2.5.0.3 UMIC-Mesh

The UMIC-Mesh(Zimmermann, Schaffrath, Wenig, Hannemann, Gunes, & Makram 2007) (RWTH Aachen University) is an alternative approach to study WMNs and it is characterized by a hybrid architecture, consisting of real and virtualized testbed. The virtual environment is used for development and validation of functionality. On the other hand, the real testbed is used for execution and evaluation purposes providing a high degree of realism that is needed for this step. The WMN consists of 21 mesh routers in one building and 12 in another, 2 routers are

used to interconnect both buildings. Each mesh router is equipped with two 802.11a/b/g NICs: one is used for router-to-client communication, another is used for mesh backhaul, creating effectively a second generation configuration. The testbed provides the option to run only two protocols: DYMO and OLSR (reactive and proactive routing). The chosen metrics for evaluating the protocols were: throughput, average hop count, and average packet loss. Tests were done with the purpose of showing that erroneous use of routing metrics (OLSR HELLO and TC) for wireless multi-hop networks can significantly reduce performance.

The above examples collect common evaluation metrics and all of them designed their own test and reporting tool. Evolution of WMN technology depends on the obtained results and laboratory environments can possibly catalyze the proliferation of such technology.

2.6 Clustering in Wireless Networks

In a clustering scheme(Yu & Chong 2005), the nodes in the network are divided in virtual groups, and are allocated adjacent to each other according to the heuristics of the clustering algorithm. A typical cluster structure contains one cluster-head, and one or more cluster members. A cluster-head normally serves as a local coordinator for its cluster members, performing intra-cluster functions such as: transmission coordination, data forwarding, aggregation, and so on.

Clustering has several advantages over simple network schemes. A cluster-head can coordinate transmission events within the cluster to reduce transmission collisions, it can also serve as a virtual backbone for inter-cluster routing, reducing the routing overhead outside the cluster.

When topology changes, only the nodes in the corresponding clusters need to update the topology information. Local changes are not propagated through the whole network, reducing the amount of information exchanged between nodes and thus greatly improving scalability.

2.6.1 Properties of Clustering Algorithms

The clustering algorithms are different across many attributes and categories (Abbasi & Younis 2007a):

Cluster proprieties : internal cluster characteristics can be divided into:

Cluster count : in some algorithms the number of *cluster-heads* is predefined in the beginning, while in other algorithms it can change dynamically, yielding a variable number of clusters;

Cluster stability : the number of members of a cluster can either be fixed, or nodes can adaptively change its membership across multiple clusters;

Intra-cluster topology : in case of multi-hop clusters, nodes can either change the node to which they forward information to the *cluster-head*, or use always the same node over time.

Cluster-head capabilities : *cluster-heads* can often provide different functionality to the network:

Mobility : they can be stationary or can relocate to other node, causing the clustering algorithm to execute again;

Node type : there may be only a subset of nodes that can be *cluster-heads*, or all nodes can have that capability;

Role : they can simply relay traffic, or perform other aggregating functions.

Clustering process : the entire clustering algorithm varies between systems:

Methodology : the algorithm either runs on every node, or a central node decides which nodes are *cluster-heads*;

Cluster-head selection : random nodes can become *cluster-heads* or they can be pre-assigned;

Algorithm complexity : the complexity and convergence rate of the algorithm can be constant or dependent on the number of nodes.

Formally, we can define the problem of building such structures (Chen, Liestman, & Liu 2004). Given the graph $G = (V, E)$ where the vertices are the nodes in the network and the edges are the communication links between them, the clustering process divides V into a collection of subsets $\{V_1, V_2, ..., V_k\}$, where $V = U_{i=1}^{k} V_i$, such that each subset V_i induces a connected subgraph of G. These subgraphs can overlap and each subset is a cluster G'. Typically a particular vertex V is elected to represent the cluster G' and it's denoted *cluster-head*.

Figure 2.3: Dominating set.

A way to cluster a WMN is to use the notion of graph domination. The members of a dominating set are chosen as *cluster-heads* and along with their neighborhood comprise the cluster.

2.6.2 Graph Domination

A *dominating set* of a graph $G = (V, E)$, is a subset $S \subseteq V$, such that every vertex $v \in V$ is either in S, or adjacent to a vertex of S. Figure 2.3 illustrates a *distance-1 dominating set*. Usually, a vertex subset S is called *distance-k dominating set*, if every vertex v is within a distance-k neighborhood of any vertex of S.

An *independent dominating set* is a *dominating set* except that no two vertices in the set are adjacent (non-overlapping sets).

Intuitively, the clustering process on a WMN should be executed with the heuristics of a *dominating set*, the use of *independent dominating sets* as clusters does not allow a flexible cluster election process due to its non-overlapping restrictions.

2.6.3 MOCA

Multi-hop Overlapping Clustering Algorithm (MOCA)(Abbasi & Younis 2007b) is a randomized clustering algorithm. It assumes that each node in the network becomes cluster-head with a probability p. Each cluster-head, when self-elected, transmits a message of its self-

election. All nodes receiving the message rebroadcast it up to k-hops away. The probability p is used to control the number of clusters and their degree of overlap.

2.6.4 FLOC

Fast Local Clustering service (FLOC)(Abbasi & Younis 2007b) is a clustering algorithm that classifies nodes based on their proximity to their cluster-head into inner (i-band) and outer (o-band).

A node stays idle and waits for a random duration to receive an invite from any cluster-head. If no invitation is received, the node becomes a candidate to cluster-head and broadcasts that message. When a node j receives that candidate message, and is already an i-band or o-band member of a cluster j, it replies back to inform the candidate of such membership. The candidate cluster-head will perceive the conflict and join cluster j as o-band node. If the candidate cluster-head receives no conflict messages, it becomes a cluster-head.

FLOC has self-healing capabilities since o-band nodes can switch to i-band in case they receive an invite from a closer cluster-head.

2.7 WMN Monitoring

The purpose of network monitoring is to extract information about the system current configuration, the current values of relevant performance metrics, to detect abnormal or faulty behavior, and forecast potential performance degradation scenarios.

The information obtained via network monitoring can be used by system administrators to solve existing problems, plan the maintenance and future upgrades of the system. The monitoring information may also be used by the protocols that run in the WMNs to optimize their own performance.

2.7.1 Examples of Monitored Values

Examples of configuration parameters and performance metrics that can be extracted from a network node are: CPU and memory usage; Uptime; RSSI and Noise (Received Signal Strength

Indication and Noise can be used to assess the medium quality); Bit rate; Wireless Channel; MAC Address and IP Address; Clients associated; MTU (the Maximum Transmission Unit is useful to know at which size the packet will fragment); TX, RX, FW packets; TX, RX, FW errors; TX, RX, FW traffic (Transmitted, Received, and Forwarded packet information is useful to understand how the network load is distributed); and Default Gateway.

2.7.2 Monitoring Steps

Network monitoring consists of two main steps (Sailhan, Fallon, Quinn, Farrell, Collins, Parker, Ghamri-Doudane, & Huang 2007): measurement phase and gathering phase. In the measurement phase, the state and performance of the nodes is evaluated, while in the gathering phase, the data is collected and analyzed with the purpose of inferring the overall network state. These phases can be implemented using different approaches.

2.7.2.1 Measurement Phase

The measurement phase can be passive or active. Passive measurement consists of capturing and examining individual packets passing through the node, whereas active measurement involves the injection of probe packets into the network. Active and passive measurement approaches have distinct advantages and drawbacks. Active monitoring causes application and measurement traffic competition, while passive monitoring avoids the contention problem. On the other hand, active monitoring improves fault tolerance, provides more up-to-date data and, more importantly, is application or protocol independent, in a sense that passive monitoring is tightly coupled with application or protocol specifications. Both approaches can be combined in a hybrid manner(Lowekamp 2003): when the network is saturated traffic-wise, a passive monitoring is used; when the network is on a non-traffic-intensive state, active measurements could be done without compromising the bandwidth offered to clients.

The active measurement phase can also be based on broadcast or unicast traffic. On broadcast-based measurements, each node broadcasts probes to all neighbors at an average period, introducing extra overhead as explained above. On the other hand, unicast-based measurements make use of the real unicast traffic as the natural probing packets without incurring extra overhead. Naturally, this approach only provides monitoring data when there is traffic

being routed through the nodes.

2.7.2.2 Gathering Phase

In turn, the gathering phase can be reactive or proactive. With reactive monitoring, the system gathers information only when it is requested (on-demand basis). The event driven monitoring is a particular category of reactive monitoring: data is only transmitted when a determined event occurs. To that end, a threshold-based monitoring is used. However, reactive monitoring does not provide the ability to predict future problems. With proactive monitoring, the systems actively collect and analyze the network on a regular basis to detect faults and predict potential states that compromise network performance. This is especially important if the network status snapshot has to be as up-to-date as possible due to time-critical traffic analysis. However, this approach suffers from high monitoring overheads. The reporting frequency should be selected appropriately, so as to not impact the desired functionality. Due to their complementary nature, proactive and reactive monitoring should not be seen as competitive(Gupta, Wu, Mohapatra, & Chuah 2009).

The gathering phase can also be classified in two categories, concerning who is in charge of collecting the measures, namely it can be centralized or distributed. On a centralized network monitoring approach, a unique data point collector gathers all the information regarding the network state from a set of agents that are limited to perform only data measurements. This concentration of data processing and analysis on a single node hinders the scalability of the system. In contrast, distributed network monitoring systems are comprised of a hierarchy of top and mid-level managers and bottom-level monitoring agents. Such top-down approach improves scalability and may be further enhanced by developing cooperation protocols between nodes located at the same hierarchy level.

In the following paragraphs we address some relevant monitoring protocols and systems.

2.7.3 SNMP

The Simple Network Management Protocol (SNMP)(Schoffstall, Fedor, Davin, & Case 1990) 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. The SNMP

architectural model is a set of devices that run SNMP agents, which collect local information as defined in the SNMP Management Information Base (MIB). MIBs contain the state of each node in the form of counters and variables.

Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management stations.

SNMP allows for periodic polling of variables in the MIB of each node as well as *traps*, which are triggered by events in the network. A network management system would periodically poll each node from a single location and provide a full network view to its administrator. SNMP traps could also be configured in the agents to respond to local changes. SNMP-based systems can either be proactive or reactive and are typically centralized, having high overheads due to periodic polling of MIBs.

Since SNMP uses a centralized model, its use to monitor WMNs is limited due to the inherent poor scalability. More efficient data exchange and decentralization has to be done in order to consume lesser network resources. One approach to enhance scalability is to limit the amount of monitoring information that is forwarded to the monitoring system.

2.7.4 MeshMon

MeshMon is a multi-tiered framework(Raghavendra, Acharya, Belding, & Almeroth 2009) 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. The biggest challenge in designing multi-tiered metric systems is to identify which metrics are strictly necessary to make decisions at each tier and which should be their threshold. When a baseline metric crosses its threshold, the node attempts to locally diagnose the problem, if unsuccessful, it contacts the gateway that will attempt to generate a diagnosis on every node of the node's upstream path. Overhead reduction is then achieved by only transmitting the necessary metrics for a specific problem set.

2.7.5 Scuba

Scuba(Suwannatat, Kevin, & Almeroth 2008) has an approach to monitoring similar to MeshMon, in the sense that limits the amount of observed metrics. It provides a focus and context visualization framework, in which the performance metrics are placed into several tiers or contexts. The topmost context provides the network administrator a holistic overview of the mesh network. This view can be narrowed to focus on the problem region, zooming the level of detail in order to reveal the underlying metrics that are not within their normal range. Contexts are divided in three zones: route, link, and client. The route context displays multi-hop routes between mesh routers and gateways and their metrics. The link context reveals the Expected Transmission Count (ETX) on each link. The client context provides metrics to diagnose causes of poor connection quality to mesh clients. SCUBA has a clear disadvantage that is the non-adaptation to network conditions, monitoring information transfer to the gateway should be aware of application data transfers.

2.7.6 Probing Systems

Probing or active measurement has also been researched for emergency scenarios(Naudts, Bouckaert, Bergs, Schoutteet, Blondia, Moerman, & Demeester 2007). During rescue interventions, it is important that the monitoring tool provides updated network state information without too much overhead. Two techniques are combined to monitor the network: end-to-end probing and bandwidth estimation. End-to-end capacity estimations can be obtained based on per-hop measurements, even under dynamic network conditions. Each node estimates the link capacity to each of its neighbors by sending packet probes. Two back-to-back packets are sent to each neighbor. First a small packet is sent as a trigger, followed by a probe packet with larger size. The time difference between the arrival of the first and second packet is measured and the result is sent back to the sending node. This solution works on top of the OLSR protocol, that transports the measured link capacity, together with link channel information and disseminates it throughout the network. The link capacity and channel information of each node on a path can be used to identify the bottleneck-link and to make an end-to-end bandwidth estimation of that path. If for each channel on a path, its capacity is calculated, an estimation of end-to-end bandwidth can be made by extracting the minimum capacity value, which is the bottleneck.

2.7.6.1 MeshFlow

MeshFlow(Huang, Yang, & He 2007) is another 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 made. MeshFlow records of each router are then exported to a collector, that constructs an entire view of the network. Probing is not always very accurate, since it ignores certain factors that affect the packet delivery time and path capacity in a WMN, for e.g.: cross talk between wireless interfaces or adjacent channel interference. Furthermore, the system does not scale very well as node density rises.

2.7.7 MMAN

MMAN(Kazemi, Hadjichristofi, & DaSilva 2008) runs on top of OLSR and 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, another to transmit that information out-of-band between stations and the management unit. Albeit not injecting additional traffic, the stations increase deployment cost by requiring an extra radio interface and an extra network to transfer monitoring data.

2.7.8 DAMON

DAMON(Ramach, Belding-royer, & Almeroth 2004) uses an agent-sink architecture for monitoring mobile networks on top of the AODV protocol. 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. Agents at the periphery of two or more sinks can receive beacons intermittently; agent association oscillation is prevented by replacing the primary sink only if a predetermined number of beacons are successively received. 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.

2.7.9 JANUS

JANUS(Scalabrino, Riggio, Miorandi, & Chlamtac 2007) is another distributed framework, running on top of Mesh Connectivity Layer (MCL) from Microsoft. It uses Pastry(Rowstron & Druschel 2001), a DHT (Distributed Hash Table) peer-to-peer overlay network, to make information available to all nodes in the system. Each node has a unique identifier (ID), which is the hash of its IP address. The routing algorithm works by resolving a single digit at a time. At each step, a node forwards the message to a node whose ID shares with the key a prefix that is at least one digit longer than the prefix that the key shares with the current node. If such node cannot be found, the message is delivered to the node with closest ID. JANUS also uses Scribe(Castro, Druschel, Kermarrec, & Rowstron 2002) 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 are source constraint environment such as a WMN, the scalability is poor.

2.7.10 Cluster Based Systems

Distributed systems should be designed to scale well according to the underlying structure. The above systems only perform well for a reduced number of nodes. Hierarchical systems should be designed to account for unbalanced distribution of nodes through the structure. 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. Nodes form a monitoring overlay that promotes collaboration and adapts itself to the underlying network dynamics. The intermediate levels of the hierarchy produce summaries of the collected data, in order to compress it, before transmitting data to the upper layers. The *cluster-head* plays a special node in the hierarchy that is elected to coordinate and publish information, it also builds and maintains a local network view (aggregation and correlation of data) of its cluster(s) members and outside connections to neighboring *cluster-heads.*

2.7.10.1 ANMP

ANMP(Chen, Jain, & Singh 1999) is a monitoring solution for ad hoc networks designed as an extension of SNMP. It uses the same structure and protocol for data collection through MIBs. Cluster-heads poll information from their cluster members, which creates unnecessary overhead. The clusters are not dynamic and this solution has not yet been implemented nor tested.

2.7.10.2 Cluster Monitoring

A more recent clustering solution was proposed in (Sailhan, Fallon, Quinn, Farrell, Collins, Parker, Ghamri-Doudane, & Huang 2007). The cluster formation is triggered by the addition of a new node at any time. When a node joins the network it broadcasts a *cluster-head* query, and its neighbors rebroadcast the message up to k-hops away, being k a configurable parameter. If after a determined time the node does not receive a reply, it promotes itself to *cluster-head*. A *cluster-head* may either accept or refuse the new node into its cluster, subjected to different criteria, for example, QoS, load or location. *Cluster-heads* periodically poll cluster member nodes to verify if they are alive. On the other hand, cluster member nodes expect to be polled, assuming that a *cluster-head* has disappeared in case the poll messages are not received. The absence of polling messages triggers a *cluster-head* promotion. Monitoring traffic interference with application traffic is not taken into account, and thus may disrupt normal network operation if both monitoring and application traffic are in the same area.

2.7.10.3 Mesh-Mon

Mesh-Mon(Nanda & Kotz 2008) is another clustering solution. It 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. The first principle states that each mesh node must measure its own local information, the second principle aims for a distributed analysis allowing nodes to cooperate and detect local problems, while the third principle is a common approach to achieve scalability through aggregation of data.

Mesh-Mon uses a combination of active and passive monitoring techniques, and a rule-based diagnosis engine. The information collected is concerned with the system configuration and measurements from the physical, link, and network layers. Periodically, nodes probe each other to measure bandwidth and latency among clients, mesh nodes, and external hosts. The measured information is summarized and disseminated to other nodes. To enhance scalability, more information is stored about local neighborhood than about nodes far away, however this constant measuring can affect network performance.

Mesh-Mon nodes can communicate using flooding if the routing protocol fails or is disabled. For networks with considerable size, the flooding is limited to k-hop neighbors, and thus forming a hierarchical overlay of *cluster-heads*, which are responsible for exchanging information between k-hop neighborhoods. A *cluster-head* is selected by its k-hop neighbors using a leader-election protocol. Nodes appoint themselves as *cluster-heads*, if none exists. Discovering of other *cluster-heads* is done through a beaconing process. The importance of a node in a mesh network can be characterized by the number of routing links the node shares with its neighbors. Assuming global topology is available, all the nodes can be ranked according to their degree of importance. A better connectivity rank can be calculated using eigenvector centrality (EVC). EVC is calculated using the network global topology and it is proportional to the sum of the centrality values of all neighboring nodes. A node with a high value of EVC is a strong candidate for *cluster-head*. EVC can be calculated using three proposed variants: binary adjacency matrix representing the global topology, ETX, and gateway EVC, in which the importance of Internet gateways is emphasized.

2.7.10.4 Astrolabe

Astrolabe(Renesse, Birman, & Vogels 2003) uses a gossip protocol as the method for dissemination queries and results. The key idea behind a gossip protocol is simple: periodically, each agent (running on every node) selects another agent at random and exchanges information with it. As time passes, the data will tend to converge (if agents are in different *zones*, then they exchange data associated with their least common ancestor *zone*). Each *zone* elects the subset of agents that gossip on its behalf. The election algorithm can either be arbitrary or deterministic, based on characteristics like load, uptime or even agent coverage (*zones* that the agent represents). When it is time to gossip, the agent picks at random one of the child *zones*,

System	Gathering	Structure	Implemented	Routing protocol dependent	Scalable
MMAN	proactive	plain	yes	yes (AODV)	no
DAMON	proactive	hierarchical	yes	yes (AODV)	yes
JANUS	hybrid	plain (DHT)	yes	yes (DSR variant)	no
ANMP	hybrid	hierarchical	no	yes	yes
Cluster Monitoring	hybrid	hierarchical	yes	yes (OLSR)	yes
Mesh-Mon	proactive	hierarchical	yes	no	yes
Astrolabe	hybrid	hierarchical	yes	yes	yes

Table 2.1: Comparison between distributed solutions.

other than its own. Next, the agent looks up for the contacts attribute for the selected *zone* and randomly picks another agent from the set of hosts in the list and proceeds to contact it and send attributes of all child zones at that level and for higher levels up to the root of the tree. The contacted agent compares the information received with his own and updates his out-of-date information and sends its own information back to the gossiper. Astrolabe adopts a weak notion of consistency, which means that updates will eventually be reflected in every node. Astrolabe also allows the use of SQL queries to search or subscribe to certain events. The queries provide more granularity in monitoring information access than the summarized information exchanged between nodes.

2.8 Discussion

Table 2.1 summarizes the differences between the distributed solutions previously analyzed. In a resource constraint environment such as a WMN, monitoring solutions will tend to be decentralized and distributed, passing some of the intelligence usually in the core of the monitoring system to each node that does not play a passive role anymore. The choice between proactive or reactive measurement depends on the resource or variable that requires monitoring, if one needs an historical chart of such resource, then a proactive approach must be taken, if the report can be done via events that only alert if something is wrong, then a reactive measurement can be chosen. Typical clustering systems have passive measurement capabilities and adopt a decentralized architecture to distribute the task of monitoring the network between their nodes, which gather data in a proactive or reactive manner.

The main challenges faced when developing a monitoring system are: minimize bandwidth consumption, minimize the size of monitoring information while providing important information for diagnostic of network health, adaptation of monitoring systems to underlying network conditions, resilience to cluster-head or gateway failure, minimize the resources consumed by mesh routers (CPU and memory).

All these challenges have to be faced while maintaining an up-to-date information of network conditions. An approach to WMN monitoring and testing that addresses these challenges is presented in the next chapter.

Summary

This chapter has introduced Wireless Mesh Networks and discussed a number of approaches to perform network monitoring in this type of systems. In the next section we will propose an alternative design for WMN monitoring, that aims at minimizing the impact of the monitoring traffic in the data flows the are established in the WMN.

3 Adaptive Semi-Circular Cluster-Based Monitoring

3.1 Introduction

This chapter introduces the Adaptive Semi-Circular Cluster-Based Monitoring for Wireless MESH Networks, that allow network operators to efficiently retrieve monitoring information from the network nodes with low interference on application traffic.

3.2 System Model

We consider a system model where it is assumed bidirectional communication with the same transmission power. We assume that network nodes always transmit with the same power, and if a node b can receive a transmission from a node a, then node b can also transmit information to node a.

3.3 Architecture Overview

The system is a cluster-based hierarchical solution where nodes self-organize into k-hop clusters.

Clusters emerge in the network as described in Section 3.4.2.

Every node's objective in the network is to report its status information to the gateway, where all monitoring information converges. Therefore, all nodes try to send their monitoring information to the gateway. Our clustering module, intercepts that information and diverts it to the node's cluster-head. The cluster-head then aggregates and forwards such information, using our routing protocol described in Section 3.4 to the gateway through one or several nodes as illustrated in Figure 3.1.

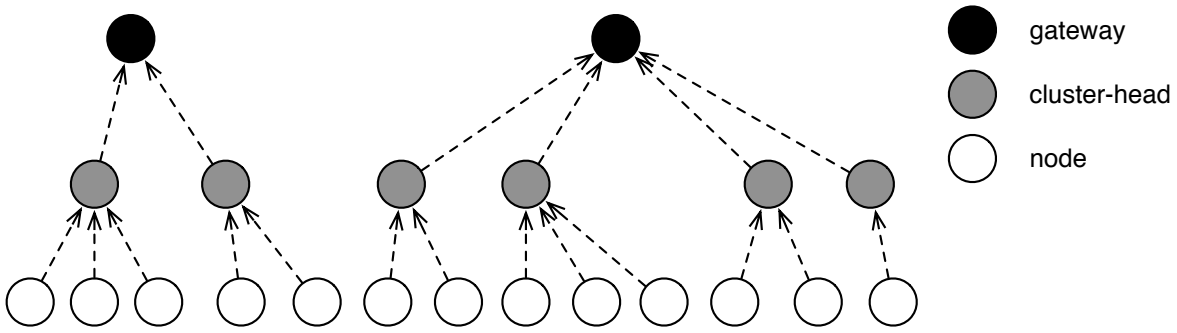


Figure 3.1: Architecture.

On top of these core modules runs the Adaptive Transfer Information module, described in Section 3.5, that perceives the network conditions in which the nodes execute, and tries to increase the performance of application traffic by diverting monitoring traffic away from its routes.

The system also supports multiple gateways, described in Section 3.6, and in that case, the network is logically segmented.

The system emulates a passive measurement monitoring approach, in which monitoring information is generated by capturing and examining packets that pass through a given node.

It is also based on proactive gathering monitoring: actively collecting and analyzing the network on a regular basis. As network status snapshots are being taken frequently, the time to detect faults and predict potential states is low. Such conclusions are possible through trend analysis, that provide a projection of network performance. This is especially important to reduce the reaction time of network operators to potential problems.

As the system is comprised of clusters, that classifies the approach as distributed architecture, that is a hierarchy of gateways, cluster-heads and nodes, respectively top, middle and bottom-level agents, that improve overall scalability and functionality flexibility through the addition of middle-level agents.

The network clusters yield a variable number depending on the network topology and the exit and entering of nodes in the network. The cluster stability is also adaptive, in which nodes can adaptively change its membership across multiple clusters as described in Section 3.4.2. The intra-cluster topology is also dynamic: nodes can change the next hop to which they forward their monitoring information to the cluster-head.

In terms of cluster-head capabilities, their mobility is fixed: once they are elected they cannot move; all nodes can become cluster-heads, with no restrictions imposed other than they being members of a cluster; and during our tests, in Section 4, the majority of the cluster-heads simply relay traffic.

The clustering algorithm runs on every node and it's not centralized, the cluster-head selection is totally random and follows one of the algorithms described in Section 3.4.2 that converge in O(1).

3.4 Routing of Monitoring Information

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. (Johnson, Ntlatlapa, & Aichele 2008).

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). This log is configured to clear all BEACON messages that are older than $e \times \text{BEACON}_{sendrate}$ and to shift the position of every Beacon message in the log according to a FIFO policy. 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.

3.4.1 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.

The *epochs* history value: 10, was selected so that all algorithms depending on that value are as up-to-date as possible. A large value of stored *epochs* would mean that if a node stopped sending BEACON messages, either because it went down or because they are colliding with other transmissions, its quality would be perceived as high during a large amount of time.

Following the same rationale, the *stability threshold* of 2 was selected to allow nodes to quickly react to network conditions. If a node's BEACON quality starts to decrease beyond 2, then a node using it as next hop should seek in his log another one to use as best next hop. Lower values of *stability threshold* would promote route flapping and cause nodes to constantly switch best next hops and route instability.

3.4.2 Clustering Algorithm

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 3.4. As soon as the distance to the gateway has been computed, nodes initiate a timer, with a value defined by Eq. 3.1 or Eq. 3.2, and set their state to UNCLUSTERED.

$$SelfElection_c(dist_{gw}) = \begin{cases} dist_{gw} + \lambda(s), \\ if dist_{gw}\%(2k+1) = 0 \\ \\ \alpha \times dist_{gw} + \lambda(s), \\ otherwise \end{cases} \tag{3.1}$$

In both Eq. 3.1 and Eq. 3.2 α is a constant that increases the time of self-election of non-optimal nodes and λ is a random value between 0 and 1 that avoids multiple nodes to self-elect at the same time, 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 3.2.

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

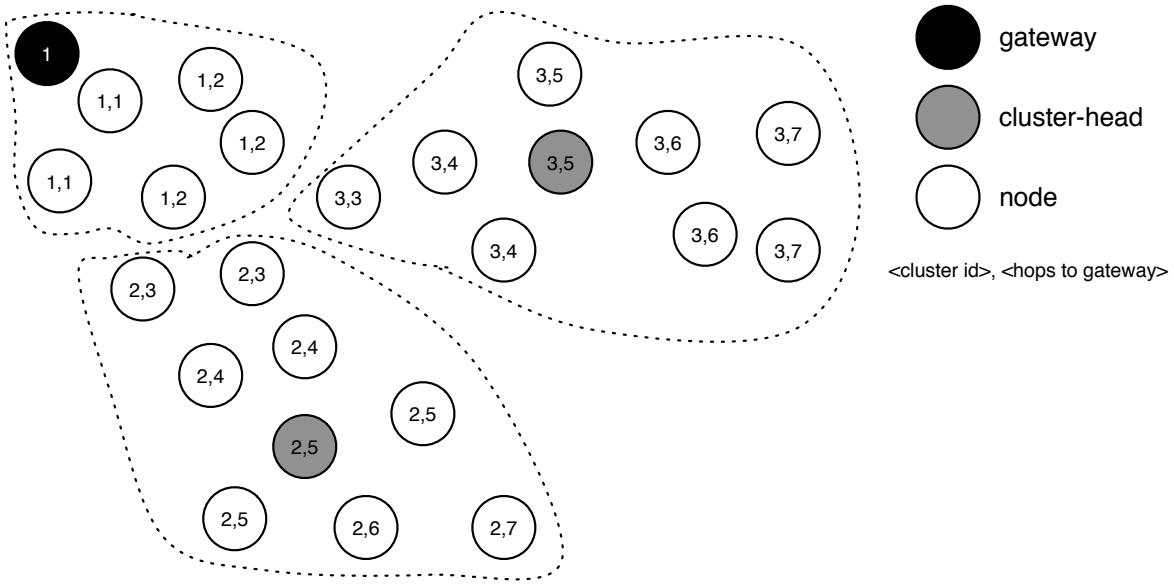


Figure 3.2: Circular Clustering.

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. The node sets again the timer to constantly check if it receives HELLO messages, if not, it passes to the CLUSTER-HEAD state. 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 3.4, for the processing of BEACON messages.

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 the double of the send rate). By introducing a decision layer between the nodes and the gateway, the system becomes more flexible and adaptable to the underlying network conditions. Cluster-heads can perform a myriad of operations to the data collected: averages, maximum or minimum, compress data or execute other complex operations.

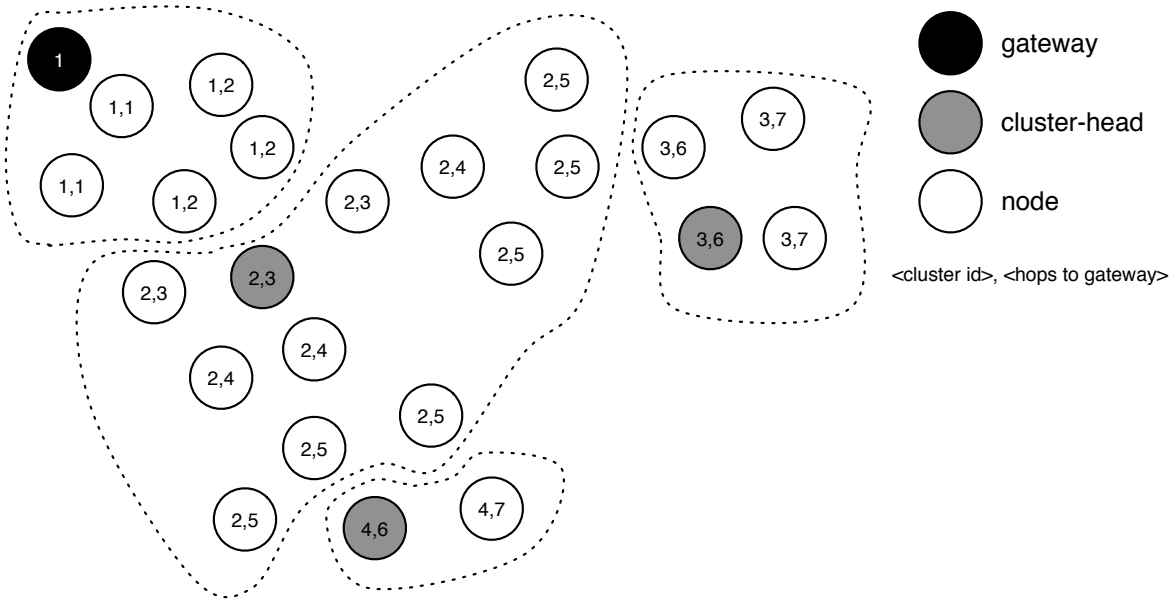


Figure 3.3: Semi-Circular Clustering.

3.4.3 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 redundant, 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 3.3. 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. 3.2. 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.

$$SelfElection(dist_{gw}) = \begin{cases} dist_{gw} + \lambda(s), \\ if dist_{gw}\%(k+1) = 0 \\ \\ \alpha \times dist_{gw} + \lambda(s), \\ otherwise \end{cases} \tag{3.2}$$

3.4.4 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.

3.4.5 Optimized Routing

Nodes that are best next hop towards the cluster-heads wait for the reception of the monitoring message before sending their own, aggregating two monitoring messages in just one packet, reducing the monitoring overhead. This promotes a more efficient use of the spectrum, since the message is transmitted in only one packet, saving not only the second header but also the transmission mechanisms in the 802.11 MAC.

3.5 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 in order to minimize the interference that monitoring traffic may have on the applications running on the mesh. Traffic detection can be done either by identifying the well-known common destination ports used or by sniffing the payload of the transmitted packets and evaluating what type of traffic it is. 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

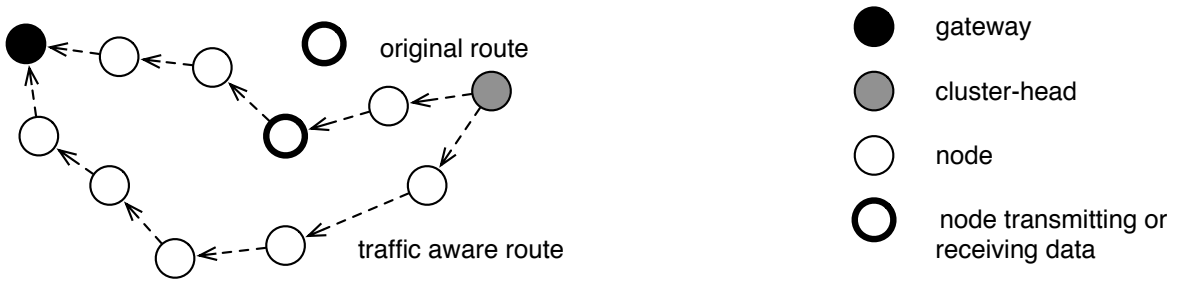


Figure 3.4: BEACON message delayed propagation.

quality from nodes that are forwarding traffic to drop and, consequently, nodes will not choose them as best next hops.

Figure 3.4 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.

3.6 Multiple Gateways

The presence of multiple gateways causes unnecessary BEACON messages that will not be used by nodes far away from them. To limit the flooding of such 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.

In borderline situations, a node a receiving a BEACON message from gateway a will rebroadcast it to node b, but since node b has selected gateway b as its gateway, it will receive the BEACON rebroadcasted by node a and only store it in it's log. The reverse process happens for node b when receiving a BEACON message from gateway a through node a.

3.7 ns-2 Implementation

3.7.1 The ns-2 Network Simulator

Ns-2 is a discrete event simulator for network protocols and scenario research. It provides support for simulation of TCP/IP or UDP, routing, multicast protocols and can be configured over wired and wireless networks using two languages due to the two separate tasks it needs to perform.

Protocol simulations require a programming language that can provide detailed and efficient manipulation of bytes, packet headers and iterations over large sets of data. For such tasks, run-time speed is crucial. On the other hand, network settings such as parameters, configurations or quickly deploying a large number of scenarios requires a simple language where iteration time (changing the model and re-run) is important.

Ns-2 uses C++ and OTcl to meet the above needs. C++ runs fast and takes more time to change and OTcl runs slower but it can be changed more quickly. Ns-2 internals connect both languages.

3.7.2 Implementation Details

In order to implement the Adaptive Semi-Circular Cluster-Based Monitoring protocol in ns-2, some changes had to be made.

An agent inheriting from the Agent class was created. This is the main class where we implemented our protocol. In addition, this class links with the OTcl interface trough scripts. The agent maintains internal states of the various modules of the protocol. Also, we had to define the packet types that are transmitted by our solution.

3.8 Experimental Prototype

The experimental prototype was developed in La Fonera+ devices running OpenWrt. The experimental testbed was deployed in the IST-Taguspark campus, comprising of eight La Fonera+(FON 2009), equipped with an Atheros processor with a clock of 183.50 MHz, 16 Mb of

Command	Description
gw_beacon_ival *val*	Sets the BEACON interval to *val* seconds.
cluster_head_hello_ival *val*	Sets the HELLO interval to *val* seconds.
cluster_head_hello_ttl_ *val*	Sets the HELLO TTL to *val* hops.
cluster_head_formation_ *val*	Sets the clustering formation to circular or semi-circular (when *val* is 0 or 1, respectively).
cluster_head_monitoring_ival_ *val*	Sets the monitoring send rate to 1 / *val* seconds.
cluster_head_compression_ *val*	Sets the cluster head compression rate to *val*.
cluster_head_ait_ *val*	Enables or disables the AIT module (when *val* is 1 or 0, respectively).

Table 3.1: TCL commands to configure the monitoring protocol.

RAM, 8 Mb of flash, one IEEE 802.11b/802.11g wireless card, one LAN port and one WAN port, running the OpenWrt(OpenWrt 2009) firmware.

The OpenWrt is a Linux distribution developed for embedded devices, providing a platform with a myriad of packages that suit the developer or user needs thus allowing the full customization of the system. Several packages were installed to create the mesh network and to add the functionality of our monitoring system, as described in detail below.

3.8.1 Implementation Details

We implemented the Adaptive Semi-Circular Cluster-Based Monitoring protocol as an NS-2 Agent, i.e. a functional module on top of the transport level in the communication stack. Therefore, to use our solution in NS-2 the user simply has to configure a wireless node with the developed agent on top. For the transport protocol we support UDP, and require unicast routing support (AODV or DSR are available in NS-2 and will suffice).

The developed agent inherits from the generic Agent class and implements the complete functionality of our protocol, including all its modules. It is configurable in TCL through a set of TCL commands, described in Table 3.1. Such commands are always pre-appended by: Agent/WMM set *command*. Additionally, a new packet type was created to encapsulate the control traffic required by our solution.

Within the OpenWrt system, our protocol was developed in user space. That allowed the system to have the flexibility of having other routing protocols to route application data, leaving

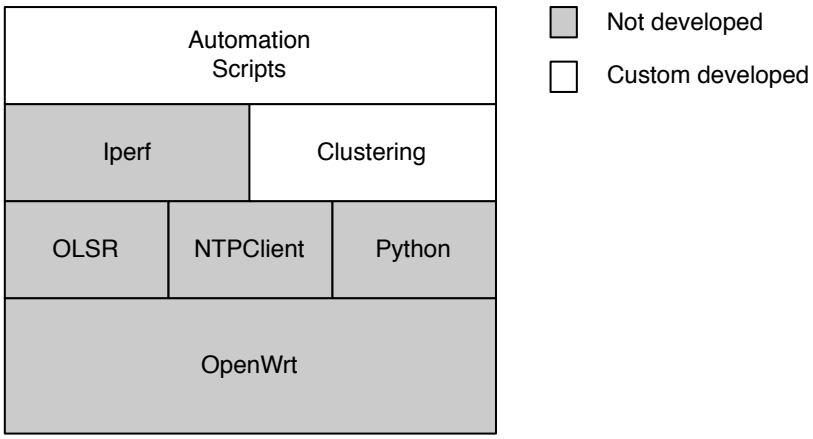


Figure 3.5: La Fonera system.

the monitoring data routing for our implementation to handle.

The OpenWrt firmware 8.09 version was installed with the following packages: *python*; *olsr*, *ntpclient* and *iperf*, which are represented in the La Fonera system architecture depicted in Figure 3.5.

The *python* package was installed in order to develop the core systems of the monitoring solution. The core systems were developed similarly as with the ns-2 tool and thus replicating all protocol behavior to the experimental prototype. Python was chosen due to the ability to quickly develop and test the solution without compiling the code.

The *olsr* package was installed to provide connectivity among the nodes. The *olsr* package is the most stable routing protocol available in the OpenWrt repository to provide mesh connectivity.

The *ntpclient* package was installed to synchronize the clocks among the nodes. Some difficulties were found in establishing the OLSR connectivity and the time synchronization enhanced it.

The *iperf* package was installed in order to provide a flexible traffic generation capability. The traffic generated was either monitoring information traffic or VoIP traffic.

Nodes were also configured to create an ad-hoc network and were attributed different IP Addresses in the 192.168.1.x range. The individual node firewall was also changed to allow

OLSR packets, NTP packets and monitoring packets to traverse between the nodes.

Summary

This chapter presented the main components of the Adaptive Semi-Circular Cluster-Based Monitoring architecture. It started with an introduction of the system model and the components of the monitoring system, namely the routing protocol; a typical clustering algorithm and the novel semi-circular one; and finally the Adaptive Transfer Information module.

In the following chapter, the presented modules of the system are evaluated using both simulations and an experimental testbed.

4 Evaluation

4.1 Introduction

To assess the performance of the monitoring system, we resort both to simulations and to a prototype 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.

4.2 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; the AODV protocol was used to route data packets when our system was used and the OLSR protocol was used to route data packets from the SNMP system. For each test, we have generated 10 different scenarios using the *BonnMotion* tool(University of Bonn 2009), 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. The default BEACON send rate is 1 message per 5 seconds; the default HELLO message send rate is 1 message per 2 seconds and the default QUARANTINE state period is 2 $\times$BEACON$_{sendrate}$ to account for the route stabilization parameters.

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;

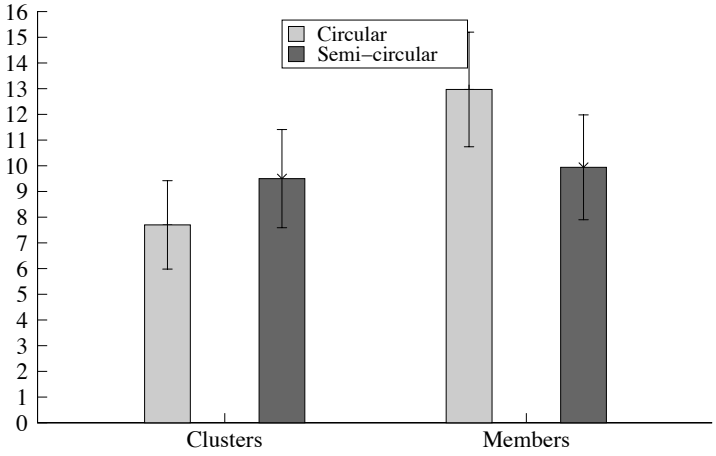


Figure 4.1: Clustering performance.

- Impact on multimedia streams;
- *Adaptive Information Transfer* (AIT).

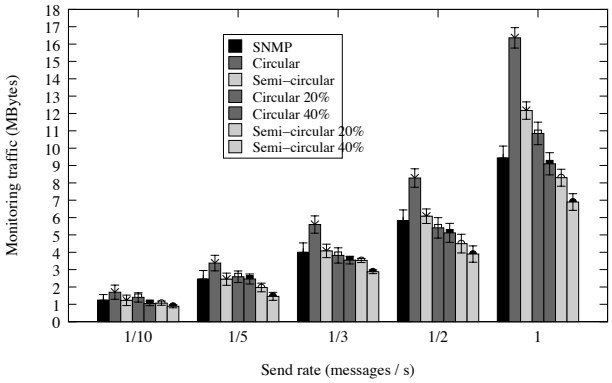
4.2.1 Clustering

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 4.1). 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.

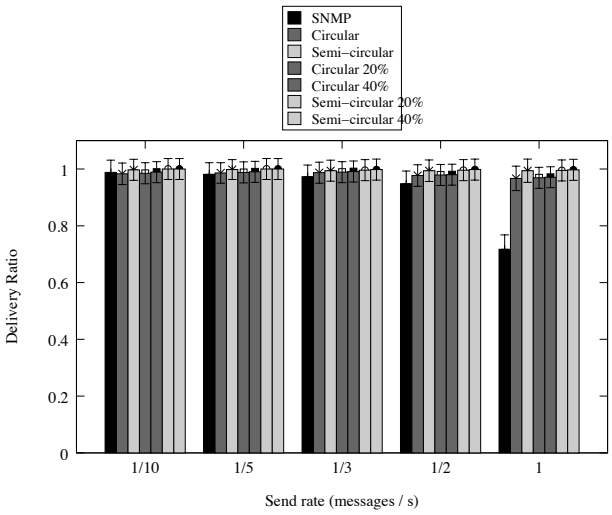
The time of cluster establishment, measured as the interval between the election of the cluster-head and the last node to join the cluster, is relatively the same in both methods: 9.11ms for the circular and 8.15ms for the semi-circular clustering.

4.2.2 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



(a) Monitoring traffic.



(b) Delivery ratio.

Figure 4.2: Monitoring traffic and delivery ratio comparison.

monitoring messages with 100 bytes. Figure 4.2(a) and Figure 4.2(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, 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 4.2(a) also show the results when the aggregation function is able to reduce the size of monitoring information by 20% and 40%.

4.2.3 Route stability

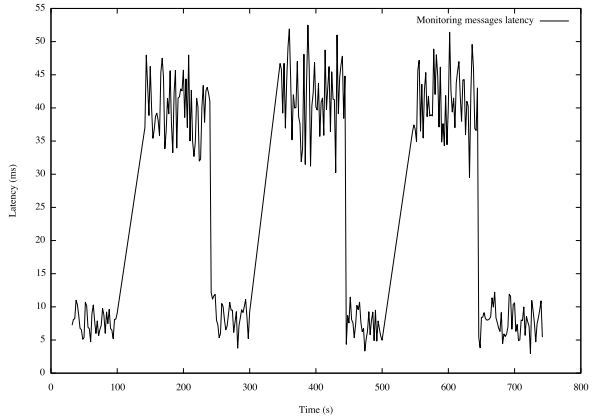
To test the route stability, we tailored a scenario where we selected a random node a (3-hops away from gateway) and identified its best next hop towards the gateway denoted as node b. Afterwards, we forced node b to go down, and observed that node a chose node c (10-hops away from gateway) as its best 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 4.3(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 4.3(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.

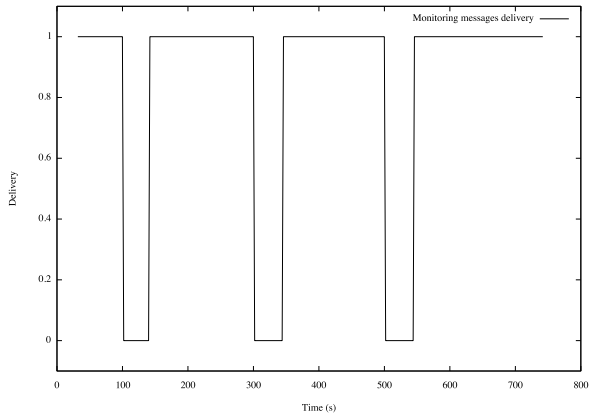
In a loss free scenario, when node b goes down, node a, using a *stability threshold* of 2, would need 3 *epochs* to switch its best next hop to node c (assuming all of node c's BEACON messages are delivered). However, in a medium with interferences, BEACON messages will collide and switch time could be a little higher as shown in Figure 4.3(a): around 35 seconds.

4.2.4 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



(a) Monitoring messages' latency.



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

Figure 4.3: Route stability.

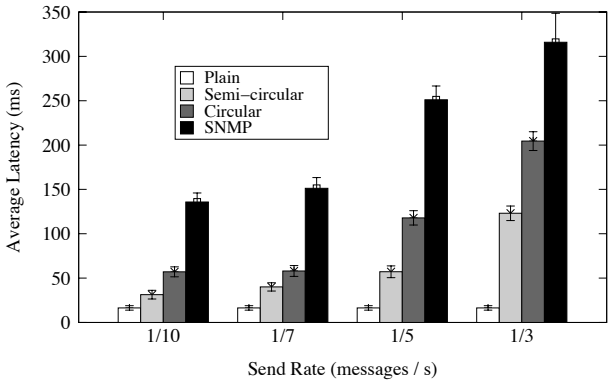
stream uses a route of 3 hops, traversing nodes near the gateway.

To measure the call quality the following metric(Cole & Rosenbluth 2001) is used:

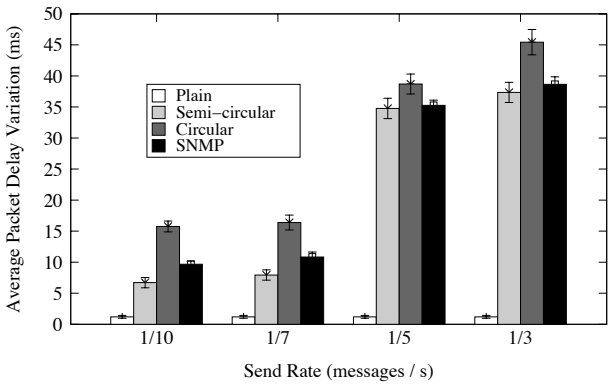
$$\begin{aligned} R &= 94.2 - 0.024d \\ &- 0.11\,(d - 177.3)\,H\,(d - 177.3) \\ &- 30ln\,(1 + 15e) \end{aligned} \tag{4.1}$$

where:

- $d = 25 + d_{buffer} + d_{network}$ is the total ear to mouth delay (25ms), delay in the buffer and the network latency;



(a) Latency of VoIP call.



(b) Packet Delay Variation of VoIP call.

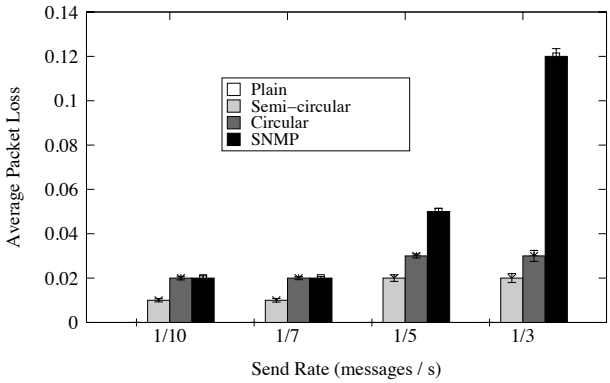
Figure 4.4: Metrics of VoIP call.

- $e = e_{network}$ + (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.

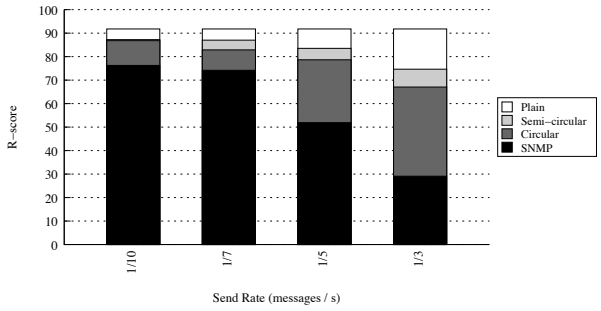
Quality is defined by the *R-score* (Eq. 4.1) which states that a score of 70 provides a medium call quality in VoIP.

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 4.4(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



(a) Packet Loss of VoIP call.



(b) R-score of VoIP call.

Figure 4.5: Metrics of VoIP call.

provides the smallest negative impact on latency.

In terms of packet delay variation (Figure 4.4(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 4.5(a)). In particular for SNMP, the packet loss is higher due to the fact that all nodes route the monitoring packets directly to the gateway, increasing neighborhood interference.

Figure 4.5(b) illustrates the combined effect of these factors, using the *R-score* metric previously described.

4.2.4.0.1 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.

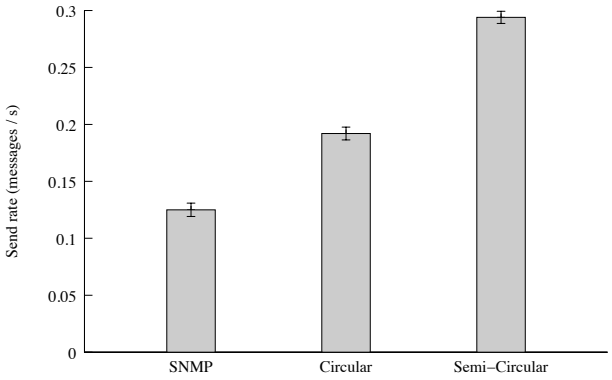


Figure 4.6: Maximum monitoring throughput comparison.

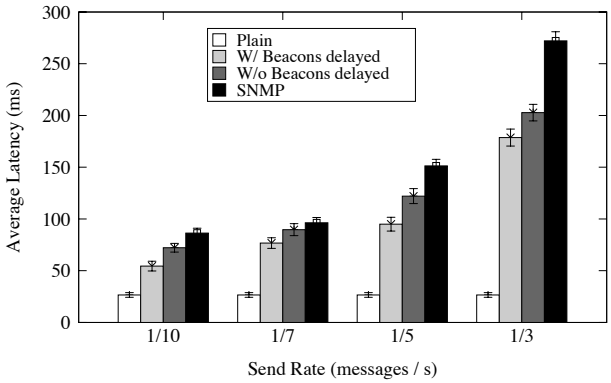
Analyzing Figure 4.6, we can see that a higher monitoring rate can be sustained by our architecture than with SNMP, without bringing the quality below the target threshold. The semi-circular clustering can achieve a send rate of one message every 3.4 seconds, without compromising call quality, while SNMP can only achieve 1 message every 8 seconds (a 42,5% increase in performance).

4.2.5 Adaptive Information Transfer

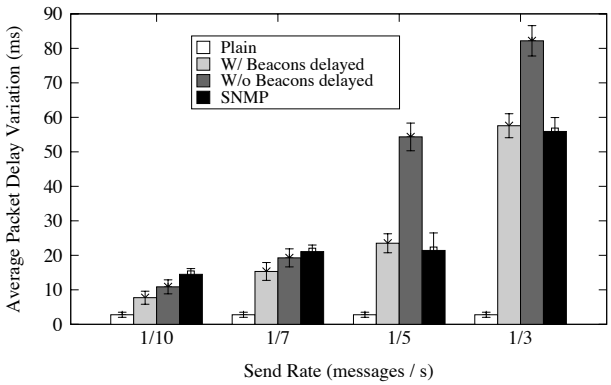
To test the Adaptive Information Transfer, a VoIP stream, with the same characteristics of the ones used before, was created near the *cluster-head* and between the nodes that were the best next hops of the *cluster-head* towards the gateway. For this test, besides fixing the gateway, the nodes that were mentioned above, were also fixed, leaving the rest of the nodes' position to the generator. Also, for this test we only tested the semi-circular clustering as it is well clear that it outperforms the circular clustering.

Measurements were done with the Adaptive Information Transfer mechanism on and off. The *cluster-head*'s monitoring information packets were observed, and in fact, when the mechanism was on, they took a different path than when BEACON propagation was not tempered with.

The stream's latency, in Figure 4.7(a), is lower when the BEACON messages are delayed and the *cluster-head* uses a different route due to the less traffic competition in the nodes participating in the routing of the stream.



(a) Latency of VoIP stream.



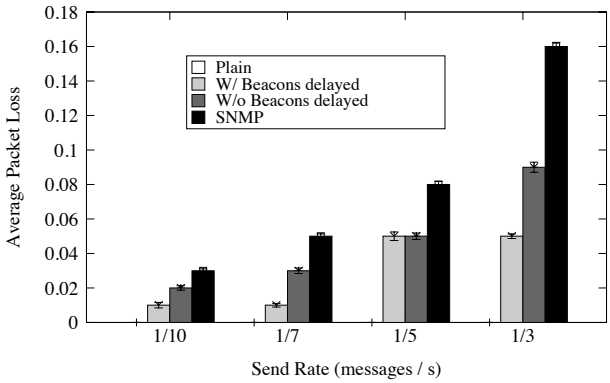
(b) Packet Delay Variation of VoIP stream.

Figure 4.7: Metrics of VoIP stream.

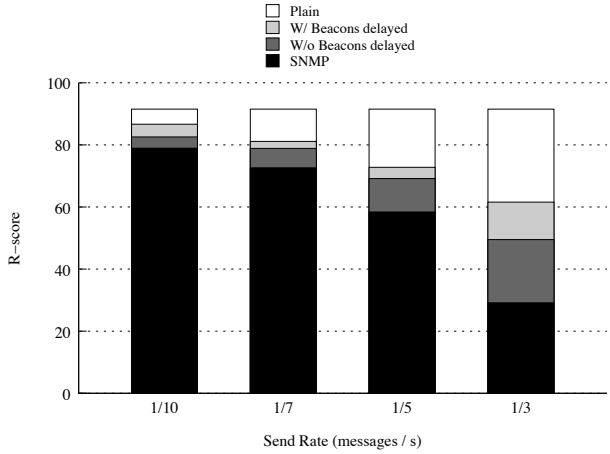
Due to *cluster-head* proximity (a source of bursty traffic), the stream has higher packet delay variation using the semi-circular clustering method without BEACON messages being delayed. Using the mechanism that delays them, we improve the stream's packet delay variation due to the fact that monitoring information no longer traverses the same nodes as the multimedia stream (Figure 4.7(b)).

The packet loss, in Figure 4.8(a), follows the same explanation, as the monitoring traffic is being diverted away from the path taken by the stream, the probability of losing a packet by collision lowers.

The change of path forced the monitoring information packets of the *cluster-head* to not interfere with the VoIP stream near it, causing the *R-score* to rise comparing to the other solutions.



(a) Packet Loss of VoIP stream.



(b) *R-score* of VoIP stream.

Figure 4.8: Metrics of VoIP stream.

4.3 Experimental Testbed

All devices in the experimental testbed 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.

We opted to perform tests regarding the:

- Clustering and;
- Adaptive Transfer Information.

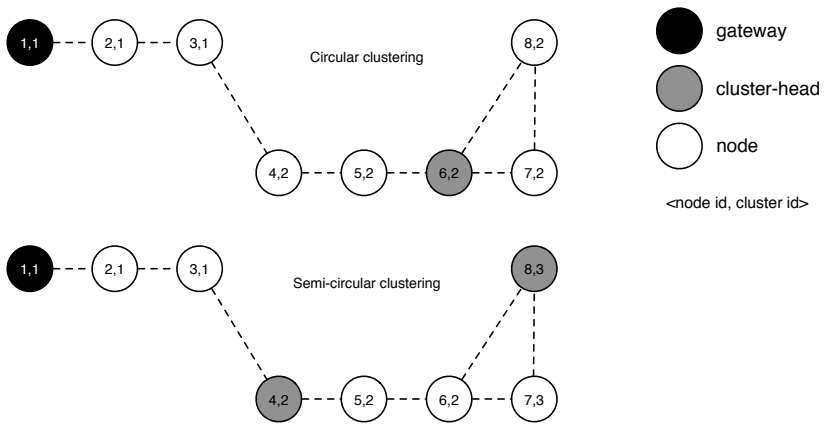


Figure 4.9: Testbed Topology for Clustering tests.

Clustering	Clusters	Nodes per Cluster	Cluster Establishment Time (ms)
Circular	2	3	13.72 (5.47)
Semi-circular	3	1.66	12.26 (2.68)

Table 4.1: Comparison between clustering methods.

4.3.1 Clustering

The two clustering methods were tested in the network topology depicted in Figure 4.9, they also were simulated in a replicated network in *ns-2*.

The results in Table 4.1 show the different times for cluster establishment that are similar to the ones obtained through simulation (indicated between parenthesis).

4.3.2 Adaptive Information Transfer

To test the adaptive information transfer, the topology was changed to the one depicted Figure 4.10.

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 4.2, being the effects of the AIT mechanisms clearly visible.

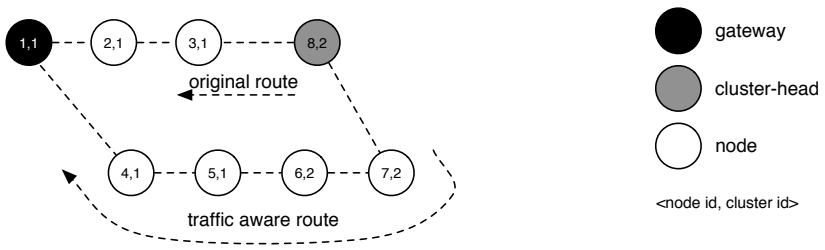


Figure 4.10: Testbed Topology for Adaptive Information Transfer tests.

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 4.2: Performance of the Adaptive Mechanisms in the La Fonera Testbed.

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 comparison 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.

Summary

In this Chapter, we have experimentally evaluated the operation of the Adaptive Semi-Circular Cluster-Based Monitoring for Wireless MESH Networks using simulations and an experimental testbed.

We have compared its performance against the most popular solution for monitoring networks.

Overall, the system proved to be a robust and efficient monitoring approach for mesh networks and proved to be adaptive to the underlying network conditions.

5 Conclusions and Future Work

In this work we proposed an architecture to monitor the nodes of a WMN. In such a dense wireless environment, the monitoring protocols should consume a minimal amount of network resources and avoid interfering with application traffic.

While designing the protocol, we focused on minimizing the amount of traffic exchanged between nodes, and designing mechanisms that prioritized application traffic over monitoring traffic without compromising too much its delivery and transmission delay.

Our solution combines different functionality: it is based on semi-circular clusters that optimize the routing of monitoring information and utilized adaptive mechanisms that minimize the impact of monitoring information on the streams that are present in the network. We evaluated our architecture and associated protocols running extensive simulations and using a experimental testbed with eight 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 a on-going VoIP call. Further tests show that our solution behaves even better when using compression schemes in the cluster-heads.

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. We would also like to develop a module that interacts with out Adaptive Transfer Information module to notify it of what type of application data is traversing the node and interact with a rule-based engine to classify and determine what to do when the node is forwarding certain types of data flows.

References

Abbasi, A. A. & M. Younis (2007a). A survey on clustering algorithms for wireless sensor networks. *Comput. Commun. 30*(14-15), 2826–2841.

Abbasi, A. A. & M. Younis (2007b). A survey on clustering algorithms for wireless sensor networks. *Comput. Commun. 30*(14-15), 2826–2841.

Abolhasan, M., T. Wysocki, & E. Dutkiewicz (2004, January). A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks 2*(1), 1–22.

Aguayo, D., J. Bicket, S. Biswas, D. Couto, & R. Morris. Mit roofnet implementation.

Akyildiz, I., X. Wang, & W. Wang (2005). Wireless mesh networks: a survey. *Computer Networks ISDN Systems 47*(4), 445–487.

Bicket, J., D. Aguayo, S. Biswas, & R. Morris (2005). Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, New York, NY, USA, pp. 31–42. ACM.

Bing, B. (2007). *Emerging Technologies in Wireless LANs: Theory, Design, and Deployment*. New York, NY, USA: Cambridge University Press.

Castro, M., P. Druschel, A.-M. Kermarrec, & A. Rowstron (2002). Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC 20*, 2002.

Chen, W., N. Jain, & S. Singh (1999). Anmp: ad hoc network management protocol. *IEEE Journal on Selected Areas in Communications*.

Chen, Y. P., A. L. Liestman, & J. Liu (2004). Clustering algorithms for ad hoc wireless networks. In *Ad Hoc and Sensor Networks. Nova Science Publishers.*

Cole, R. G. & J. H. Rosenbluth (2001). Voice over ip performance monitoring. *SIGCOMM Comput. Commun. Rev. 31*(2), 9–24.

FON (2009). Fon. `http://www.fon.com`.

Gupta, D., D. Wu, P. Mohapatra, & C. Chuah (2009). A study of overheads and accuracy for efficient monitoring of wireless mesh networks. *Pervasive and Mobile Computing*.

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

Hiertz, G., S. Max, R. Zhao, D. Denteneer, & L. Berlemann (2007, Aug.). Principles of ieee 802.11s. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pp. 1002–1007.

Huang, F., Y. Yang, & L. He (2007). A flow-based network monitoring framework for wireless mesh networks. *Wireless Communications, IEEE*.

Huhtonen, A. (2004). Comparing aodv and olsr routing protocols.

Jacquet, P., P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, & L. Viennot (2001). Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pp. 62–68.

Johnson, D., N. Ntlatlapa, & C. Aichele (2008). 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*, pp. 10.

Jubin, J. & J. Tornow (1987, Jan.). The darpa packet radio network protocols. *Proceedings of the IEEE 75*(1), 21–32.

Kazemi, H., G. Hadjichristofi, & L. A. DaSilva (2008). 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.

Kiess, W. & M. Mauve (2007a). A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks 5*(3), 324–339.

Kiess, W. & M. Mauve (2007b). A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks 5*(3), 324–339.

Lowekamp, B. (2003). Combining active and passive network measurements to build scalable monitoring systems on the grid. *SIGMETRICS Perform. Eval. Rev. 30*(4), 19–26.

Nanda, S. & D. Kotz (2008). Mesh-mon: A multi-radio mesh monitoring and management system. *Computer Communications 31*(8), 1588–1601.

Naudts, D., S. Bouckaert, J. Bergs, A. Schoutteet, C. Blondia, I. Moerman, & P. Demeester (2007, May). A wireless mesh monitoring and planning tool for emergency services. In *Proc. The 5th IEEE Workshop on End-to-End Monitoring Techniques and Services*.

OpenWrt (2009). Openwrt. `http://www.openwrt.org`.

Raghavendra, R., P. Acharya, E. Belding, & K. Almeroth (2009). 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, pp. 45–48. ACM.

Ramach, K. N., E. M. Belding-royer, & K. C. Almeroth (2004). Damon: A distributed architecture for monitoring multi-hop mobile networks. In *In Proceedings of IEEE SECON*.

Renesse, R. V., K. Birman, & W. Vogels (2003). Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst. 21*(2), 164–206.

Rowstron, A. & P. Druschel (2001). 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)*, pp. 329–350.

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

Scalabrino, N., R. Riggio, D. Miorandi, & I. Chlamtac (2007). 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*.

Schoffstall, M., M. Fedor, J. Davin, & J. Case (1990). Simple network management protocol (snmp).

Suwannatat, A., T. Kevin, & C. Almeroth (2008). Scuba: Focus and context for real-time mesh network health diagnosis. In *PAM*, Volume 4979 of *Lecture Notes in Computer Science*, pp. 162–171. Springer.

University of Bonn (2009, June). *BonnMotion a Mobility Scenario Generation and Analysis Tool*. University of Bonn.

Wang, X. & A. Lim (2008). Ieee 802.11s wireless mesh networks: Framework and challenges. *Ad Hoc Networks 6*(6), 970–984.

Yu, J. & P. Chong (2005). A survey of clustering schemes for mobile ad hoc networks. *Communications Surveys Tutorials, IEEE 7*(1), 32 –48.

Zimmermann, A., D. Schaffrath, M. Wenig, A. Hannemann, M. Gunes, & S. Makram (2007, Oct.). Performance evaluation of a hybrid testbed for wireless mesh networks. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pp. 1–10.