Network Topology Visualization and Monitoring for Multi--Hop Wireless Networks

Roberto Riggio, Matteo Gerola, Antonio Francescon, Andrea Zanardi, Tinku Rasheed
CREATE-NET, via alla cascata, 56/D, 38123 Trento, Italy
(name.surname@create-net.org)

Francois Jan
France Telecom R&D, 2 Av Pierre Marzin, 22307 Lannion, France
(francois.jan2@orange-ftgroup.com)

Abstract

Multi-hop wireless systems represent a viable means for deploying access networks covering medium-size areas with limited investment, making use of commodity hardware and freely available software suites. At the same time, management of such networks represent an overly complex task, due to the joint effect of the time-varying nature of the radio channel, user mobility and the inherently distributed nature of the system.

A number of solutions are currently being researched, whereby network management functionalities get embedded within the wireless network itself simplifying both the initial network deployment and its subsequent maintenance by removing the need for a dedicated network controller which can be expensive in that it requires dedicated hosting facilities, power, and adequate cooling. A common building block of such approaches is represented by a monitoring framework able to bring the relevant network-level information to the decision points.

In this paper, we present a distributed network monitoring toolkit, specifically developed for wireless multi-hop networks. The toolkit allows network administrators to monitor the status of the network as well as to plan and execute active measurement campaigns. Information is stored in a distributed network-wide repository and is accessible through a web interface.

Motivation

- Networks cannot be considered static objects
  - They grow, change, evolve and (sometimes) disappear
- Legacy network management tackles the issue of network adaptation
  - Definition of policies that cope with minor changes in the system
- As networks become more complex there is a growing need for human tuning
  - Labor-intensive tasks
  - Intrinsically error-prone

Wireless Mesh Networks [1] (WMNs) provide many advantages over traditional wireless networks, such as robustness, greater coverage, low up–front costs and ease of deployment. One of the major challenges that needs to be addressed in today's WMN if they have to be turned into a commodity technology is related to their control and management. In particular, for some usage scenarios, dedicated network control and management appliances may prove impractical due to either cost and/or architectural reasons.

As a result, in the last few years, a tendency emerged to distribute network management functionalities within the network itself [2-7]. The effective deployment of such solutions requires a scalable signaling channel for gathering network status information and conveying it to the relevant decision points, as well as a controller–less network management paradigm where network control and management functionalities are embedded into the network elements themselves.

In this paper, we present OBELIX, a distributed network monitoring toolkit specifically tailored for infrastructure multi-hop wireless networks. The monitoring toolkit is designed to support domain specific knowledge and incorporates appropriate reasoning logic to detect and diagnose faulty network conditions.

This paper is structured as follow. Slide 2 introduces application scenario and motivations. Slides 3 through 5 describe the state of the art and the limitations of currently available solutions. System requirements are summarized in slide 6. Slide 7 introduces the toolkit's main features. Slides 8 and 9 sketch, respectively, the network and the system architecture, while slide 10 provides some technical details. Slides 11 through 14 provide an overview of the web-based dashboard. Finally, slide 15 draws the conclusions pointing out future research directions.

How it is done today

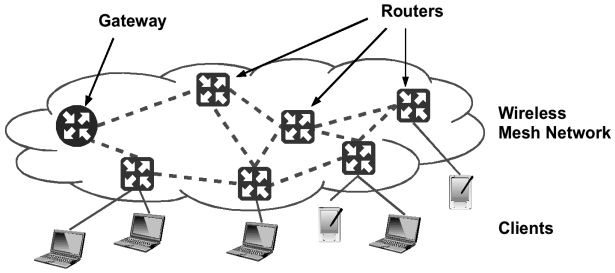
- Dedicated network controller, e.g. based on the Simple Network Management Protocol (SNMP)
  - Information gathering. Network devices are monitored by software agents to gather information on their status
  - Information Processing. Gathered information is sent to a network management system
  - Event notification. Devices can notify events to the network management system where suitable action can be defined
- Advantages
  - Provides a single point of consolidation
  - No need to touch every single network element
  - Small price tag for large deployments

A large set of protocols exists to support network and network devices management. Common solutions include SNMP [8,9], ICMP [10] netconf [11], and capwap [12]. However, most of such tools are designed around centralized architectures meaning that each node participating to the network runs a process which gathers information about the current network state. When a problem is recognized, the running process sends alerts to some management entities. Upon receiving these alerts, the management entities are programmed to react by taking some actions (e.g., operator notification, event logging, system reboot/shutdown, etc.). Management entities can also poll end-stations to check the values of certain variables.

In particular, Single Network Management Protocol (SNMP) represents the most widely used protocol for building monitoring applications. Formally, SNMP is an application layer protocol developed in order to standardize the exchange of management information between network devices. From an SNMP perspective, a network is constituted by a set of managed device (devices which are monitored to gather information on their status), agents (software running on managed devices) and a network management system (software running on managers, i.e., nodes managing the network). The network-level parameters and quantities monitored are termed Managed Object (MO). An example of a MO is the radio channel being used on a given wireless interface. While in v2 of the standard, an interface was specified for enabling communication among managers, SNMP is inherently centralized in nature.

What's wrong with that (in large scale Wireless Mesh Networks)

- Information is collected at a single decision point
  - Single point of failure
- Roles are statically assigned at deployment time
  - Human intervention is required in order to reconfigure the system
- Nodes are periodically polled by the network management system
  - Poor spatial reuse and robustness to node/link failures



In general wireless multi–hop/mesh network we can distinguish a number of logical roles, supported by the physical devices:

• Routers. They build and maintain the multi-hop wireless backhaul by establishing wireless links between nodes. Gateways. They interface the WMN with another network, typically the Internet.

• Access points. They provide wireless connectivity to the end-users' clients. In typical WMN deployments a single device combines access point, router, and (when a wired connection is available) gateway functionalities.

• Clients. They are used by the end-users to gain access to the Internet. Client nodes can be fixed, nomadic, or fully mobile (laptops, smart-phones etc.).

In centralized network monitoring and management solutions, the location of active network monitoring elements is statically selected at deployment time, using a variety of proposed algorithms and heuristics. In wireless multi–hop networks, on the other hand, the topology can change over time, due to a number of phenomena, including, e.g., node/link failures, channel fluctuations, addition of new nodes etc. In all such cases, monitoring frameworks should be able to adapt their configuration (including location of aggregation/analysis nodes) to match the features of the current topology in a transparent way.

Requirements

- **Flexibility and adaptivity.** The monitoring framework shall be able to automatically adapt the tasks performed by devices according to the actual operating conditions.
- **Spatial reuse.** The monitoring framework shall make an effective use of the available network resources.
- **Robustness and resilience.** The monitoring framework shall be robust with respect to node or link failures. No single point of failure shall be present in the system.
- **Ubiquitous network management.** The monitoring framework shall allow network administrators to manage the network using both legacy technologies or thorough a web interface

OBELIX has been designed taking into account the following requirements:

• Flexibility and adaptivity in the allocation of monitoring tasks. Different network scenarios and topologies require different configuration of the monitoring infrastructure. Nodes may just provide information on their locally measured performance or, instead, aggregate information gathered from nearby nodes. This depends on the amount of available resources (storage, computing) or on the node's location in the network topology. As in the wireless domain network conditions and topology may change over time (due to, e.g., node/links faults, new nodes joining etc.), the monitoring framework should be able to automatically adapt at run–time the tasks to be performed by devices.

• Efficient spatial reuse. The monitoring framework should make an effective use of the available network resources. In particular, it should avoid overloading certain portion of the networks and evenly distribute the traffic as much as possible.

• Robustness and resilience. The monitoring framework shall be robust with respect to node and/or link failures. In particular, the availability of information with global scope should be preserved in spite of failure of the nodes holding it. No single point of failure should be present in the system. Further, the system should possess self-healing properties, being able to automatically recover from the failure of nodes. This means that the monitoring system should be able to reconfigure itself in order to restore correct operations once failures have been detected.

• Ubiquitous network management. Network administrators should be able to both monitor and manage the network from anywhere using technologies such as an SNMP–compliant network management software or through a Web interface. This implies the ability to retrieve from any point in the network information on the whole network status.

OBELIX

- A clean slate approach to network monitoring consisting of…
  - A distributed architecture supporting run-time changes in the role performed by monitoring entities according to the actual conditions
  - A set of algorithms ensuring robustness of the monitoring infrastructure with respect to node and link failures
  - A prototypical implementation demonstrating the viability of the approach

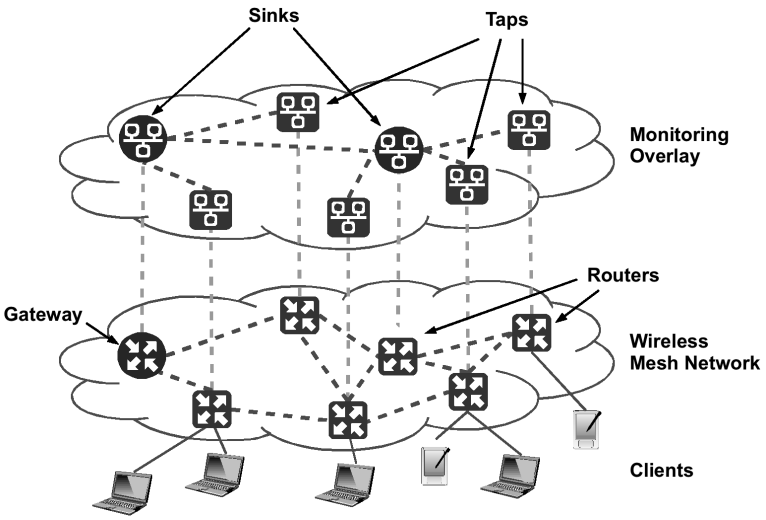
OBELIX has been designed around three pillars: adjustable level of pervasiveness, interoperability with legacy solutions, and ubiquitous network management.

Typical network management solutions, such as SNMP, rely on a centralized architecture, whereby network management tasks are carried out on a per-device. Such a solution results in poor spatial reuse of the wireless medium, congestion of routes to the network controller, and excessive load at the repository itself. OBELIX moves away from this highly centralized approach to network management by embedding management functionality (data analysis and aggregation) into the network itself and by distributing the network state information among the nodes participating in the monitoring efforts improving the availability of the managed information without disrupting network services.

Interoperability with other network management tools is provided by means of an SNMP-compatible interface. The SNMP protocol will be used to convey management information between Sinks and existing network management systems. An additional HTTP interface is supported in order to allow the web-based management dashboard to interact with the network Sinks.

The Web-based OBELIX Management Dashboard allows network administrators to both monitor and manage the network from anywhere using just a web browser. Such Dashboard supports a combined reactive/proactive approach to network monitoring. The reactive approach aims at detecting a fault only after it occurs by passively studying its effects on the network. Such an event will be logged and made available to the network administrator through the Dashboard. The reactive approach exploits temporal trends of the monitored properties in order to foresee potential failures and isolate fluctuating behaviors.

Network architecture



OBELIX supports different levels of participation in the monitoring efforts by the nodes in the WMN. Such participation can be conceptually classified into two categories:

•information gathering. Monitoring agents (Taps in the following) running within a mesh router gather the local network state either by sniffing the traffic flow in their neighborhood (passive approach) as well as by performing on-demand/periodic measurements (active approach).

•Information analysis. The local network state information gathered by the Taps is periodically sent to a set of management daemons (Sinks in the following) and exploited to maintain a global view of the network.

It is worth pointing out that information gathering and information analysis are to be considered as two separate, yet non-mutually exclusive functionalities. As a matter of fact, any node in the WMN can support a single functionality, both, or none at all. In the latter case, information about the state of a network devices can only by inferred through passive sniffing from neighboring Taps.

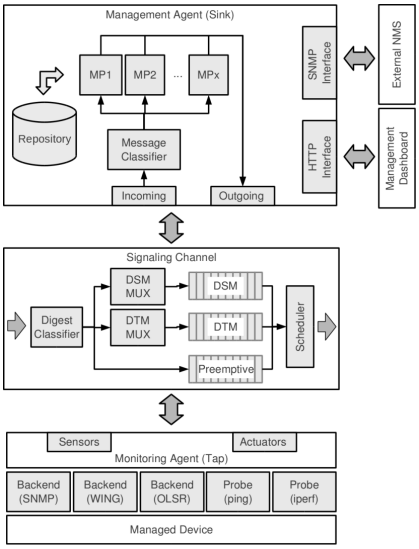
OBELIX builds a monitoring overlay in the network, whereby physical network devices are associated to a role that defines the monitoring functionalities they perform. We define two logical roles played by nodes in the monitoring framework: Taps and Sinks. As described in the previous section, Taps implement information gathering functionalities, while Sinks implement information analysis functionalities. At bootstrap, Taps selects one Master Sink for their normal operations and a number (possibly also none) Slave Sinks to be used if the Master Sink fails. During its normal operations each Tap is always associated to at least one Sink.

System Architecture

Autonomic roles selection. The network self-configure itself in order to adapt the task performed by each node to the actual network conditions

Data dissemination and replication. The global view of the network is disseminated and replicated across multiple repositories in order to improve spatial reuse and resilience

Scalable data look-up. Queries are routed to the closest node currently managing the requested object.



The Tap is a software process running in each managed device. A Tap has knowledge of the local parameters to be monitored. The local network state is collected using a modular information gathering back-end. The Tap process periodically sends the information collected to the Sink(s) it is associated to. The Tap also includes means to run measurement campaigns. This is achieved by including “Probe” components that can be triggered by means of control messages coming from Sinks for actively monitoring the network status.

The Sink is a software process running in a subset of the nodes composing the network. Sinks gather information from the Taps associated to them (communicated through a signaling channel), analyze, and store it. Information coming from the Taps is replicated in order to build a distributed base of monitoring information.

The signaling channel classifies the messages generated in response to the incoming information according to the related timing constraint; two multiplexer are included for delay-sensitive non-preemptive messages (DSM MUX) and for delay-tolerant messaged (DTM messages). The following messages are used:

(a) “Sink Hello”: messages broadcasted by Sink to signal their presence. Hello messages are flooded across the network by Tap nodes.

(b) “Tap Update”: messages sent by a Tap to the Sink(s) it is associated to, containing information on the monitored object.

(c) “Sink Update”: messages exchanged among Sinks containing updated values of the global managed objects.

Every sink has an HTTP interface through which network operators can access network information and perform management tasks. This implies the ability to look-up information, referring to a given Tap and having local scope, by redirecting the queries (coming from the network operator and injected at a given Sink) to the Sink managing that given Tap.

The prototype

- A prototype has been implemented and tested over a real-world wireless mesh network
- Monitoring back-end
  - Consists fo three software processes (Sink, Tap, Communication Channel)
  - Implemeted using the Python language
  - Runs on OpenWRT, Nokia Maemo, and Linux
- Monitoring dashboard
  - Implemented in AJAX
  - Accessible using any web browser
- The testbed:
  - Consisting of 15 mesh routers
  - Built using off the shelf-hardware and opensource software
  - Exploits multiple radio interfaces and link-quality routing

The OBELIX framework is implemented in Python, a lightweight interpreted programming language. Within a single OBELIX–monitored node, up to three distinct software processes can be active at any given time: the sink, the tap, and the communication channel.

All nodes in the OBELIX overlay run the communication channel process. Nodes that are also publishers and subscribers of monitoring information will also run the tap and the sink processes, respectively. Standard TCP and UDP sockets are used to enable communications among processes.

Messages being dispatched by the communication channel are composed by a header and a body. The header is used to indicate how and where a message should be delivered and the body provides information and commands to the destination entity. The header includes, among the other information, source and destination address, together with an indication of the type of message it carries (necessary for ensuring appropriate processing at the Sinks).

The software prototype has been experimentally evaluated over a real–world IEEE 802.11–based mesh testbed consisting of 15 multi–radio mesh routers deployed across three floors of a typical office building. Mesh routers are built around three different hardware platforms, namely the PCEngines ALIX 2C2, the PCEngines WRAP 1E and the Gateworks Cambria GW2358-4. Each node is equipped with two IEEE 802.11a/b/g wireless interfaces. Routers employ the OpenWRT operating system, a Linux distribution specifically tailored to embedded devices. Routing is implemented using the Click modular router and an implementation of the SrcRR protocol developed within the Roofnet project at MIT. The original protocol has been extended by the authors to add support for multiple radio interfaces, using WCETT as underlying routing metric.

The Dashboard

The OBELIX Dashboard is hosted by the nodes acting as mesh gateways. The network administrator can then connect to any of them using a regular web browser in order to monitor the status of the network and/or to perform administrative tasks.

This slide shows the Home Page of the Obelix Dashboard. As it can be seen, the interface is partitioned into three panes. The left pane lists the nodes currently available in the network indicating whenever the node is a mesh gateway (double-arrow marker) or a mesh router (round marker). Sink (S) and Tap (T) status is also indicated. The central pane is completely dedicated to the Google Map used to display the real-time network topology. Finally the right pane is used to display contextual information relative to the currently selected node.

The interface is highly interactive, as for example, selecting one node on the left pane with a single click will center the map on the node and will display the local information about the same node on the right pane (no additional traffic is generated on the network for this operation). On the other hand, a double click on a node's marker will load both the local and the global information thus generating additional traffic over the network. Moving the pointer on a selected node displays the default route from the node to its gateway, on the other hand moving the pointer on a non selected node displays all the outgoing routes.

Tracking statistics

The Dashboard can be used in order to track the historical values for any monitored network object (i.e. transmitted or received packets, signal-to-noise ratio, CPU or memory load, etc) in a graphical form.

Network administrator can select the parameters to be visualized using the management dashboard. It is worth noticing that these parameters are constantly tracked by each Sink thus the management dashboard is not required to required to be constantly connected to the network being monitored.

The sampling period and the number of samples to be stored can be configured by the network administrator. For example, in the default configuration the CPU load is sampled every minute and the last 60 samples are stored by each Sink. On top of this statistic, both the daily and the weekly averages are computed.

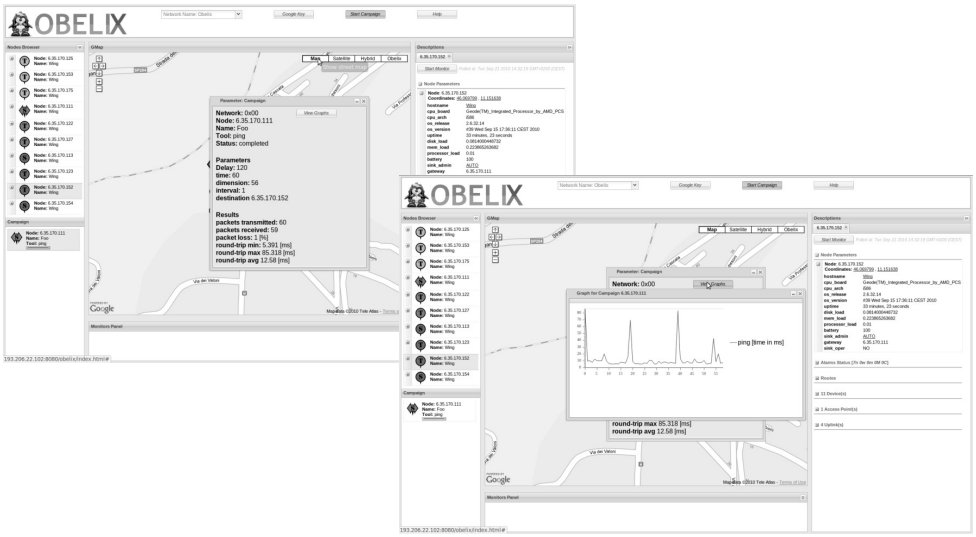
Running measurement campaigns (i)

OBELIX allows the network administrator to plan and execute network wide measurement campaigns using the embedded network probing facilities. The current version of OBELIX supports two types of network probes: ping and iperf.

The former tool (ping) allows the user to performs simple connectivity and round–trip–time measurements campaigns, while the latter tool (iperf) allows the network administrator to perform more complex measurements involving throughput, jitter and exploit ing either UDP or TCP as transport technology.

Measurement campaigns are planned using the “Start Campaign” button. Network administrator can specify the campaign type, an human readable label, and the date and hour at which the campaign shall begin. A number of probe-specific parameter can also be specified. For example, for the “ping” probe the packet size and number of sample can be specified. Scheduled and completed measurement campaign are listed in the left pane within the “Campaign” section.

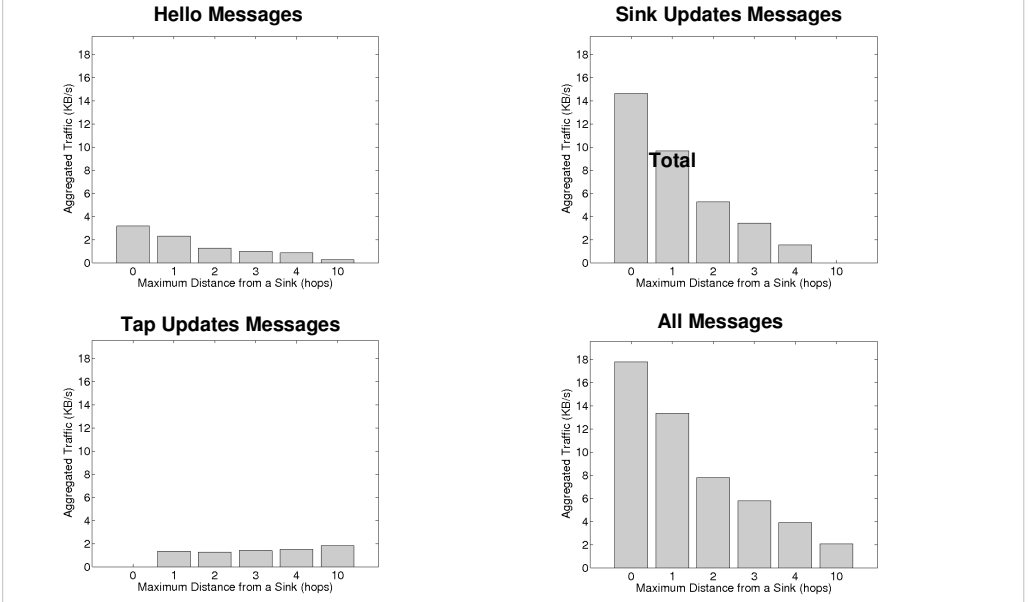
Running measurement campaigns (ii)



Once the measurement campaign has been scheduled, the network administrator can check its status by clicking on the corresponding item in the “Campaigns” section.

The same pop-up will allow the network administrator to access the synthetic results for a concluded campaign. As for example, for a campaign ran using the click probe the minimum, average, and maximum round-trip-time will be reported together with the number of samples lost. The single RTT samples can also be plotted and exported in CSV format.

Lessons learnt

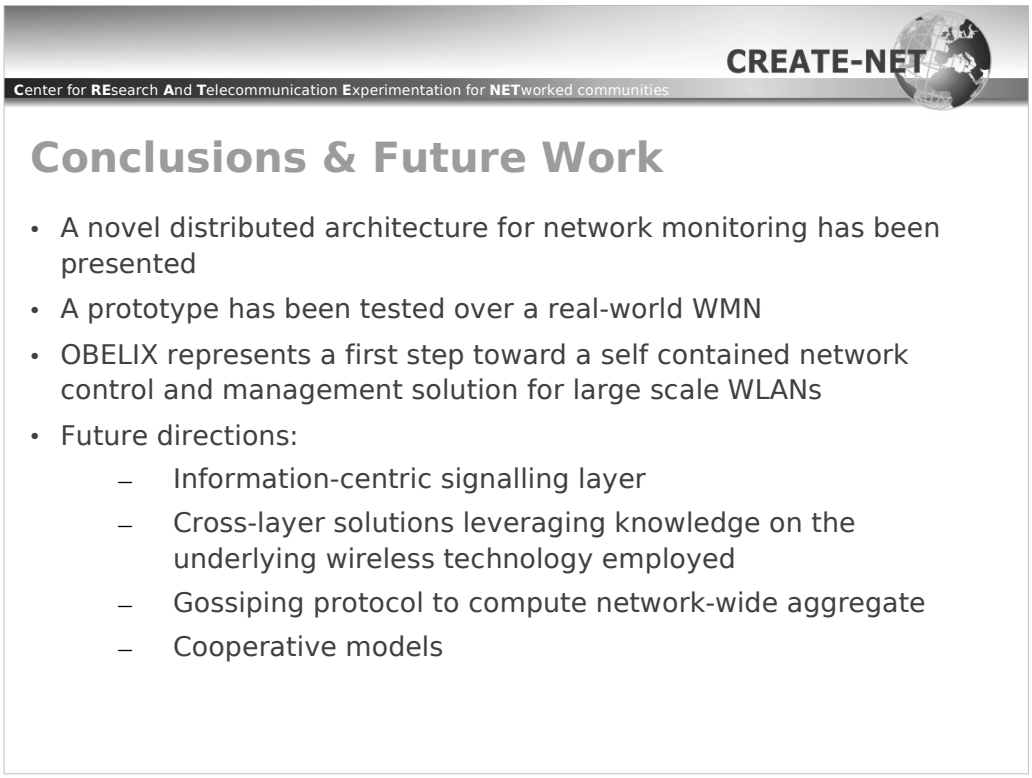


We analyzed the composition of the traffic generated, classifying it on the basis of the type of message for a varying number of Sink. The number of active Sinks in the network is controlled through the DT_H parameter in the autonomic roles assignment procedure. The DT_H parameter denotes the maximum distance of a Tap from the closest Sink.

The vast majority of the traffic is generated by "Sink Update" messages, which are used for data replication and dissemination. The amount of traffic generated by "Tap Update" messages is not correlated with the DT_H parameter value, in that those messages are always generated by each Tap and sent to the closest Sink (except in the extreme case of DT_H=0 where every node is a Sink).

Finally, the traffic carrying "Sink Hello" messages decreases with DT_H parameter. Indeed, the higher the number of hops, the lower the number of Sinks in the network and thus the amount of "Sink Hello" messages generated by the beaconing infrastructure.

The conclusion that these results allows us to draw is twofold. On the one hand, the amount of overhead generated by the monitoring overlay can be precisely controlled by the network administrator by appropriately setting the DT_H parameter. On the other hand, the overall amount of traffic generated by the monitoring overlay ranges from roughly 19 KBytes/s in the worst case (DT_H= 0) down to 2.5 KBytes/s when only one Sink is present (DT_H=10).



In this paper we have introduced an architecture and a set of protocols for building a distributed network monitoring framework. Design choices have been made to accommodate the peculiarities of wireless multi-hop networks, in terms of adaptivity, robustness and efficiency requirements. The proposed framework has been prototyped and experimentally evaluated on a 15-nodes wireless mesh network.

Directions for future research include the adoption of a more data-centric approach, whereby the whole monitoring framework becomes address-agnostic, and the use of optimised mechanisms for handling the replication of global monitoring information across sinks, leveraging -in a cross-layer perspective- knowledge on the underlying wireless technology employed.

References

[1] I. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: a survey,” Elsevier Computer Networks, vol. 47, no. 4, pp. 445 – 487, Mar. 2005.
[2] R. Mortier and E. Kiciman, “Autonomic network management: Some pragmatic considerations,” in Proc. of ACM SIGCOMMM, Pisa, Italy, 2006.
[3] A. Gonzalez Prieto, D. Dudkowski, C. Meirosu, C. Mingardi, G. Nunzi, M. Brunner, and R. Stadler, “Decentralized in-network management for the Future Internet,” in Proc. of IEEE ICC – Communications Workshops, Dresden, Germany, 2009, pp. 1–5.
[4] L. Li, M. Thottan, B. Yao, and S. Paul, “Distributed network monitoring with bounded link utilization in IP networks,” in Proc. of IEEE INFOCOM, San Francisco, CA, USA, 2003.
[5] Y. Bejerano and R. Rastogi, “Robust monitoring of link delays and faults in IP networks,” in Proc. of IEEE INFOCOM, San Francisco, CA, USA, 2003.
[6] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, “On the placement of Internet instrumentation,” in Proc. of IEEE INFOCOM, Tel–Aviv, Israel, 2000.
[7] S. Schuetz, K. Zimmermann, G. Nunzi, S. Schmid, and M. Brunner, “Autonomic and decentralized management of wireless access networks,” IEEE Trans. Netw. and Serv. Management, vol. 4, no. 2, pp. 96 –106, Sept. 2007.
[8] J. Case, M. Fedor, M. Schoffstall, and J. Davin, “A simple network management protocol,” IETF RFC 1157, May 1990, http://www.ietf.org/rfc/rfc1157.txt.
[9] W. Stallings, “Snmp and snmpv2: the infrastructure for network management,” IEEE Communications Magazine, vol. 36, no. 3, pp. 37 –43, mar. 1998.
[10] J. Postel, “Internet control message protocol,” IETF RFC 0792, Sep. 1981, http://www.ietf.org/rfc/rfc0792.txt.
[11] R. Enns, “Netconf configuration protocol,” IETF RFC 4741, Dec. 2006, http://www.ietf.org/rfc/rfc4741.txt.
[12] M. Montemurro and D. Stanley, “Control And Provisioning of Wireless Access Points (CAPWAP),” IETF RFC 5415, Mar. 2009, http://www.ietf.org/rfc/rfc5415.txt.