

A Monitoring Framework for Hotspot enabled Mesh Networks

Case Study Report



A Monitoring Framework for Hotspot enabled Mesh Networks

Luis Gómez Ruiz
luis.gmz.rz@gmail.com
Andreas Kassler
andreas.kassler@kau.se
Computer Science Department
Karlstad University

Executive summary

In this case study report, we present an initial version of a monitoring framework for wireless access point based mesh networks that tries to centralize the management of some tasks that are typically performed by the clients. To get this centralization there are two main requirements: monitoring the network and communicate with the clients in order to transmit commands and performance data. In the developed infrastructure the clients and the access points of the network are monitored by a central entity that executes commands when a performance improvement is feasible like a handover or a channel reassignment

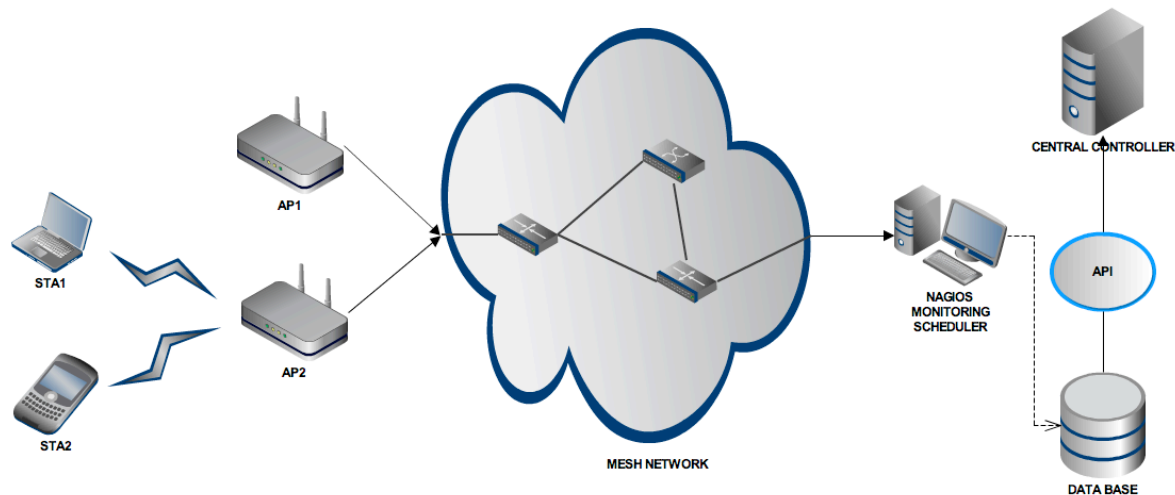
1. Introduction

Nowadays all the responsibility of associating to an access point of a 802.11-based network lies on the client. Typically, a client associates to the access point that provides the strongest signal. This behavior can lead to problems of low network performance when a big number of clients connect to the same access point while other access points can provide a better service despite of the low signal. What is proposed in the development of this project is to provide more intelligence to the network in order to improve the performance of the same by establishing a central controller that uses an API developed specifically for this purpose. The user of this API is usually a central controller that wants to get information from the nodes and send commands to the entities of the network, for example by reallocating a better distribution of the associations between clients and access points.

The developed infrastructure consists of:

- A central controller that stores information about the network through a periodic check that asks the clients of the network for monitoring data they have at any given time.
- Several access points that are queried by the central controller.
- Several mobile stations associated with access points that are also polled by the central controller to provide information on association opportunities

In the image below we can see an example of a network deployment where the developed solution can be useful. Two mobile nodes are associated to one of the two available access points. The access point forwards the traffic through the network. A Monitoring Scheduler is set to get data from APs and MNs and store it into a database.



The commands provided by the API are divided into two groups according to their objective. A first group provides to the central controller the ability to obtain necessary information from the network nodes. Some actions like getting the number of packets sent or received and obtaining the intensity of the signal, are used by the central controller to compile a database where a set of metrics is stored in order to take good decisions regarding the management of the network. This way, the central controller has the criteria to decide about the best possible configuration of the associations between Access Points and Mobile Stations.

To test this development a network scenario has been designed using KauMesh testbed trying to trigger the reaction of the system by changing the link quality between one Access Point and a Mobile Station. The expected behavior is the triggering of a re-association of the Mobile Station.

2. Infrastructure

A typical wireless network based on 802.11 is composed of a group of access points to which clients connect attending to the received signal strength. This behavior may cause agglomeration of mobile nodes in one Access Point turning this into a bottleneck scenario where a big amount of nodes want to communicate at the same time and resulting in a decrease in the performance of the network.

2.1 Design

In order to optimize such associations, we propose adding a central controller that monitors the status of the network and commands to improve the balancing of associations. Centralizing this task allows a central view on the network which makes it much easier to optimize its operation.

For the monitoring we need a system where both the Mobile Nodes and Access Points (denoted as MN and AP) submit information about their status so that the central node is always aware of the status of the associations. The transmission of this kind of information is done periodically. Too much of this information would bring down the network performance due to the high control traffic. The so obtained monitoring information is stored into a database for further queries that allow an entity to make decisions regarding readjustment of associations. The data obtained is processed and used to take decisions and finally triggering readjustments when the scenario claims for it.








2.2 Implementation

Some specific tools have been used for the implementation. We provide a brief description below:

- Nagios is an open source monitoring system. We use it as the base for our development.
- NRPE is an addon designed to allow the execution of Nagios plugins on remote machines
- mySQL is a web based open source database software that allows to create and manage databases in a simple way.
- Perl is a highly capable programming language that runs on over 100 platforms and is suitable for both rapid prototyping and large scale development projects.

Using these tools the implementation can be described as follows. The main node of the infrastructure is the central controller. It sets up a Nagios server that runs a Perl script every 60 seconds and gets monitoring information about the network. This script launches a NRPE query to one of the nodes. This NRPE must be installed as a server on all nodes and as a client in the central controller. Once this query is passed as a parameter to another output plugin that inserts this information into a database. The database has been developed and installed MySQL in the central controller. The ultimate goal is to develop applications that analyze this database and fire commands so as to trigger re-associations or a change in the operational frequency of the AP.

Mobile Nodes will have the same functionality as the access point, but will also have to go through a registration process because the central controller lacks the necessary information to communicate with them. To do this, whenever a mobile node enters the network, it should run a script so that in another database we can store the correspondence between MAC address and IP address, very similar to what has been done by the ARP protocol. Below we show a caption of the IP translation database that associates an IP to the MAC so that the communication between Mobile Node and Central Controller is possible.

			index	IP Field for IP	MAC Field for MAC	TIMESTAMP Date
<input type="checkbox"/>			28	10.10.10.25	00:80:48:47:53:BC	0000-00-00 00:00:00
<input type="checkbox"/>			26	10.10.10.19	00:80:48:47:53:C4	0000-00-00 00:00:00
<input type="checkbox"/>			27	10.10.10.24	00:80:48:53:76:85	0000-00-00 00:00:00

3.API

In the following, we show the functionality of the API developed for network monitoring and the control of it. First, it shows the central controller commands used to obtain information about the network:

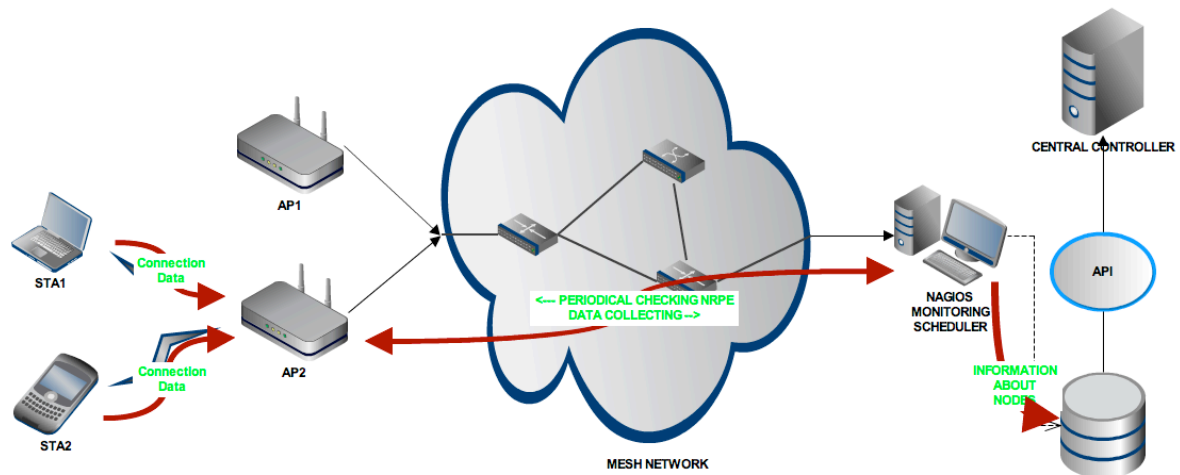
- **getaplist:** Gets the list of available access points on the network.
- **getassocsta:** Get mobile nodes associated with a specific accesspoint
- **getstarssi:** Get the RSSI, i.e. the intensity with which a signal is received.
- **getnpssent:** Gets the number of packets sent by a node.
- **getnpreceived:** Gets the number of packets received by a node.
- **getbytessent:** Gets the number of bytes sent by a node.
- **getbytesreceived:** Gets the number of bytes received by a node.
- **getChannel:** Gets the channel that operates a particular node.
- **getretransmissions:** Gets the number of retransmissions performed by a node.
- **getfailures:** Gets the number of erroneous transmissions have occurred

The next group of functions is used to make any changes to the network:

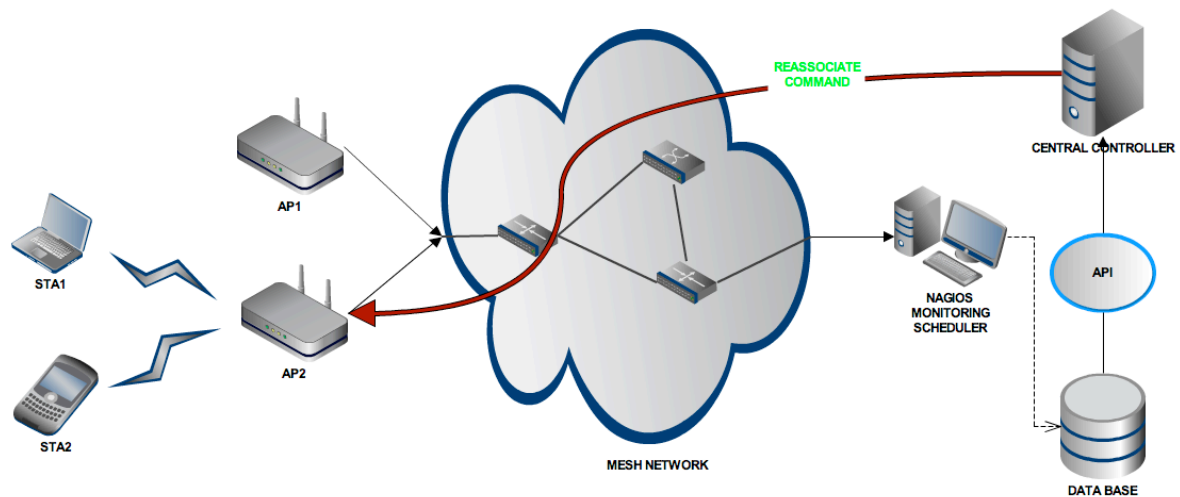
- **setchannel:** Sets the channel that is used by a node to transmit.
- **setrate:** Sets the operating speed of a node with values between 1 and 54Mbps
- **settxlevel:** Sets the transmit level in decibels.
- **handoff:** trigger a reassociation of a node and an access point.
- **get_aps:** Get access points a mobile station can potentially associate with. This will help to decide which access point the node will be reassigned at some point.

4.Scenario

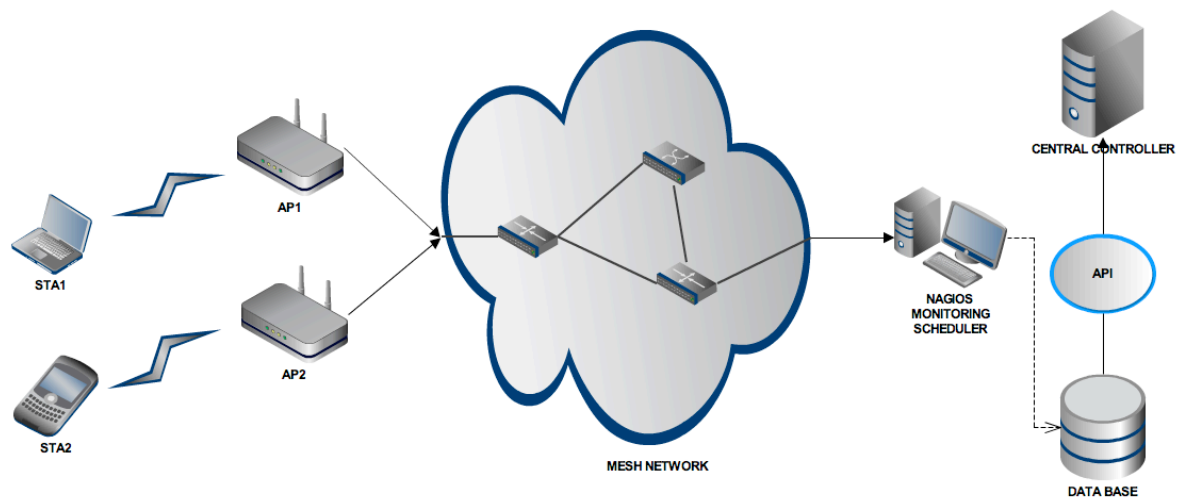
Here, we present an example of behavior of the developed infrastructure to have a better understanding of the purpose and the capabilities of the presented work. In the figure below we can see the normal behavior where data from the nodes and clients connected to the network is collected and being stored in the data base.



In the following, a overloaded AP is detected by the central controller and it is noticed that another AP is underutilized. So the central controller triggers a reassociate (handoff) command.



One of the clients is reassociated to the less used access point.



5. Conclusion

In this work, we have developed, implemented and tested a control framework for wireless LAN based Access Points. The system can also be used as is for wireless mesh networks. In current WLAN deployments, the wireless networks based on 802.11 can be inefficient in some scenarios due to the autonomous decisions of mobile terminals to associate to access points taking not into account network status. Using our implementation of a monitoring and control framework can provide a real enhancement especially in overloaded scenarios. Letting the central node be aware of the status of all of the network entities provides a new scope for wireless decision-making and for improving the actual performance of this type of networks.