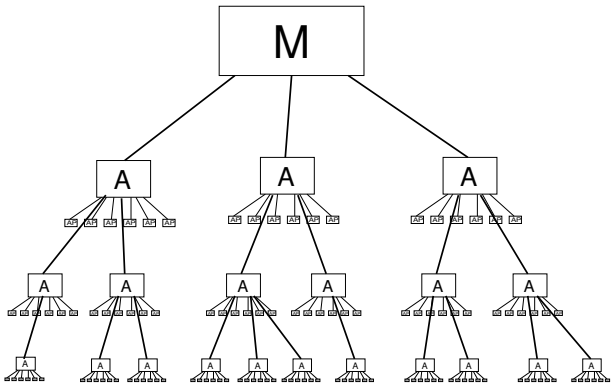
ROYAL INSTITUTE
OF TECHNOLOGY

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Daniel Grob, D-ITET

A Distributed Monitoring System for large, heterogeneous WLANs



Diploma Thesis DA-2003.30
7.4. - 8.8. 2003

Tutors:
Alberto Gonzalez (KTH)
Vincent Lenders (ETH)

Supervisors:
Prof. Dr. R. Stadler (KTH)
Prof. Dr. B. Plattner (ETH)

Acknowledgement

This report documents the work within the scope of my diploma thesis at the *Computer Engineering and Networks Laboratory (TIK)*[1] of the *Swiss Federal Institute of Technology Zurich (ETHZ)*[2]. I accomplished it as an excahnge student at the *Laboratory of Communication Networks (LCN)*[3] of the *Royal Institute of Technology (KTH)*[4] in Stockholm during summer 2003.

I would like to thank my tutors Alberto Gonzalez and Vincent Lenders, PhD students at LCN, resp. TIK, for their dedicated and competent support in various aspects. Moreover, I thank Prof. Dr. Stadler for his assistant inputs and Prof. Dr. B. Plattner for setting this exchange up for me.

Furthermore, my thank goes to all members of LCN, where I benefited from excellent support and infrastructures and was thus allowed to focus on the important tasks.

Zurich, 11th of August 2003

Daniel Grob

[1] tik.ee.ethz.ch
[2] www.ethz.ch
[3] www.imit.kth.se
[4] www.kth.se

Abstract

The goal of this thesis was the design and implementation of a monitoring system for large Wireless LANs (WLANs) composed by Access Points (AP) from different manufacturers. Furthermore, the monitoring system should consider WLAN specific data.

The heterogeneity introduced by the different monitoring interfaces (such as MIB or syslog) of the different types of AP had to be overcome by abstracting from different items to obtain the same monitoring information from the different types of APs.

The available monitoring interfaces and the monitoring data were investigated to select and abstract relevant data for the monitoring. The resulting data model provides the following attributes:

- *State:* The state of an AP.
- *Datarates:* The average datarates on the wireless interface of the AP.
- *Number of Stations*: the number of stations associated to an AP.
- *Stations:* The MAC address of all associated stations of an AP.
- *Mobility:* From the MAC addresses the mobility of all stations in the WLAN is inferable.

To provide the system with a good scalability, we applied a distributed architecture with a central Manager and several Agents. The Agents poll the APs and are in turn polled by the Manager, who thus collects the consolidated data.
To minimize the workload and the polling traffic of the Manager, we interconnected the Agents and arranged them in a tree topology, where the consolidated data is gathered branch-wise and passed on to the next level.

The final evaluations of the implementation illustrated that the system works very well in terms of performance and scalability. However, the distributing of the system entails additional expenses regarding maintenance.

Kurzfassung

Das Ziel dieser Diplomarbeit war der Entwurf und die Implementierung eines Überwachungssystems für grosse, drahtlose LANs (WLANs), die aus verschiedenen Access Points (AP) von unterschiedlichsten Herstellern bestehen. Ausserdem sollte das Überwachungssystem vor allem auch WLAN spezifische Daten berücksichtgen um allfällige neue Aspekte, die aus der neuen Technologie erwachsen, zu überwachen.

Die Heterogenität, die durch die unterschiedlichen Überwachungsschnittstellen der verschiedenen AP-Typen gegeben ist, konnte zum Teil durch Abstraktion der verschiedenen Daten ausgeglichen werden, so dass von allen APs die selben Daten erhältlich sind.

Die gegebenen Überwachungsschnittstellen und die entsprechenden Überwachungsdaten wurden untersucht und sinnvolle daten ausgesucht.

Das daraus resultierende Datenmodel beinhaltet die folgenden Informationen:

- *Status:* Der Status der APs.
- *Datenraten:* Die durchschnittlichen Datenraten der drahtlosen Schnittstelle.
- *Anzahl Stationen:* Die Anzahl der Stationen, die mit einem AP verbunden sind.
- *Stationen:* Die MAC-Addressen aller Stationen, die mit einem AP verbunden sind.
- *Mobilität:* Aus den MAC-Addressen kann die Mobilität aller Stationen im WLAN abstrahiert werden.

Um das System möglichst skalierbar zu machen, bedienen wir uns einer verteilten Architektur mit einem zentralen Manager und mehreren Agenten. Die Agenten sammeln die zu überwachenden Attribute von den APs und der Manager wiederum sammelt diese Daten in konsolidierter Form von den Agenten ein.

Um die Aufgaben und den Verkehr, der durch das Einsammeln der Daten entsteht gering zu halten, werden die Agenten untereinander zu einer Baumstruktur verbunden. Die konsolidierten Daten werden zweigweise gesammelt und zur nächsten Ebene im Baum weitergesandt.

Die Evaluation der Design-implemenation zeigt, dass das System gut funktioniert bezüglich Kapazität und Skalierbarkeit. Die Aufteilung des Systems in mehrere funktionale Einheiten bringt jedoch einen höheren Wartungsaufwand mit sich.

Thesis Schedule

	Tasks
Week 2	• Set up working environment • Acquaint with StockholmOpen network • Investigate on MIB's of AP and Switches and deducible Information • Studying & comparing of widely-used monitoring tools • Literature Studies • Simple application to fetch information from the devices
Week 6	• Studying & comparing of widely-used monitoring utilities • Study and investigate available management data, • investigate on addtional, WLAN specific data • System design • System implementation
Week 12	• System design • System implementation
Week 14	• System evaluation • System refinement
Week 16	• Finishing documenation • Submit Thesis at ETH
Extracurricular:	
Week 18	• System refinement • design & implementation of GUI • Commissioning

Contents

Chapter 1

Introduction

1.1 Motivation

Network operators and administrators have a substantial interest in the state and performance of their networks and network components.

Monitoring is thus a very important issue, which is also illustrated by the range of available monitoring systems and tools. See Chapter 3.3 for case studies.

With the increasing popularity of *Wireless LAN (WLAN)* in the last few years, a new type of network has started to supplement the existing network structures, providing untethered Internet access within buildings and frequented areas. WLAN has some novel features that distinguish it from traditional, wired networks, which are not considered by conventional monitoring tools, such as user mobility or fluctuating connectivity.

The StockholmOpen project is a cooperative effort of various *Internet Service Providers (ISPs)* to create a city-wide, operator-neutral WLAN. The vision is to create such a WLAN by connecting all *Access Points (APs)* to an open access network (see Section 4). The thereby deployed APs will cover the whole spectrum of available AP types.

1.2 Objectives

The goal of this thesis is to design and implement a monitoring system for large WLANs like StockholmOpen, which are formed by devices from various manufacturers. The system thus has to abstract from the introduced

heterogeneity among the monitoring interfaces of the different AP types and collect and process the available data.

Moreover, the novel features of WLAN should be considered in this monitoring system to provide a corresponding monitoring for these, e.g. the number of stations that are connected to an AP or the mobility of the stations.

As StockholmOpen is expected to expand considerably during the coming years, the system should provide a good scalability, with the potential to monitor scores of APs of various types.

Chapter 2

Technology Review

2.1 Wireless LAN

WLAN [1, 2] is a flexible data communication system implemented to extend or substitute a wired LAN within a building or a campus. Using electromagnetic waves rather than a cable infrastructure, it minimizes the need for wired connections and therefore drastically reduces the cost-intensive pulling of cables through walls and ceilings. Moreover, systems can be configured in a variety of topologies to meet the needs of specific applications and installations. Topologies are easily changed and range from peer-to-peer networks, suitable for a small number of users, to full infrastructure networks.

Due to considerable progresses in the fields of radio transmission and fast integrated electronics, WLAN has seen a remarkable performance increase concerning the data rate. It is now already competitive to its older wired predecessor, the 10Mbit-Ethernet. WLAN gives way for new applications adding a new flexibility to networks.

Today's working environment is characterized by an increasingly mobile workforce. Users are equipped with notebook computers and spend more of their time working in teams that cross functional, organizational and geographic boundaries. WLAN systems provide LAN users with seamless access to real-time information within a campus, regardless of location or hardware configuration. A comprehensive study of the usage and performance of a large WLAN is presented in [3].

2.1.1 WLAN, the 802.11 Standard

The IEEE 802 committee has established the 802 standards that have driven the LAN industry for the past two decades. In 1997, after seven years of work, the IEEE published 802.11, the first internationally sanctioned standard for WLAN. In September 1999 they ratified the 802.11b "High Rate" amendment to the standard, which added two higher data rates (5.5 and 11 Mbps) to 802.11.

Like all IEEE 802 standards, the 802.11 standards focus on the bottom levels of the ISO communication standard, the physical layer and data link layer (see Figure 2.1). The basic architecture, features, and services of 802.11b are defined by the original 802.11 standard. The 802.11b specification affects only the physical layer, improving data rates and providing more robust connectivity.

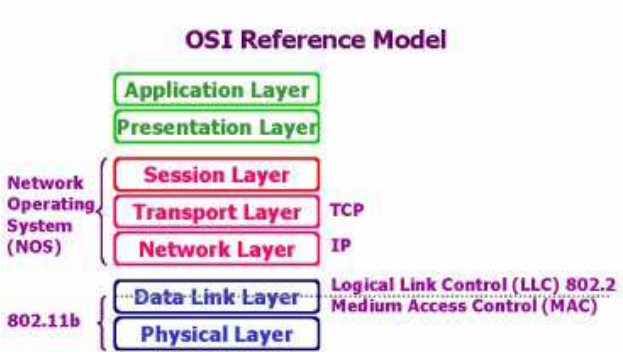


Figure 2.1: 802.11 Standards within the ISO Standard

Operating Modes

802.11 defines two pieces of equipment, a *wireless station*, which is usually a mobile device equipped with a wireless *Network Interface Card (NIC)*, and an AP, which acts as a bridge between the wireless and the wired network. An AP usually consists of a radio, a wired network interface (as defined e.g. in IEEE 802.3), and bridging software conforming to the 802.1d bridging standard. The AP acts as *Base Station (BS)* for the wireless network, aggregating access for multiple wireless stations onto the wired network.
The 802.11 standard defines two modes: *Infrastructure* mode and *Ad hoc* mode (see Figure 2.2). In the infrastructure mode, the wireless network consists of at least one AP connected to the wired network infrastructure and a set of wireless clients. This configuration is called *Basic Service Set (BSS)*. An *Extended Service Set (ESS)* is a set of two or more BSSs forming a single

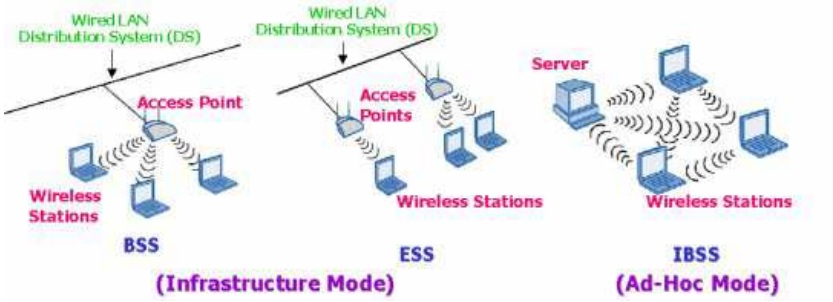


Figure 2.2: The Different Modes of 802.11

subnetwork. Since most corporate WLANs require access to the wired LAN for services they will operate on *Infrastructure* mode. The *Ad hoc* mode (also called peer-to-peer mode or *Independent Basic Service Set (IBSS)*) is simply a set of wireless stations that communicate directly with one another without using an AP or any connection to a wired network.

The Physical Layer

The three physical layers originally defined in the 802.11 standard included two spread-spectrum radio techniques and a diffuse infrared specification. Spread-spectrum techniques, in addition to increase reliability, boost throughput, and allow many unrelated products to share the spectrum without explicit corporation and with minimal interference.

The original 802.11 wireless standard defines data rates of 1 Mbps using *Frequency Hopping Spread Spectrum (FHSS)* or *Direct Sequence Spread Spectrum (DSSS)*. It is important to note that FHSS and DSSS are fundamentally different data transfer mechanisms and will not interoperate with one another.

Using FHSS, the 2.4 GHz band is divided into 75 1-MHz subchannels. Each conversation between a sender and a receiver within the 802.11 network occurs over a different hopping pattern, and the patterns are designed to minimize the chance of two senders using the same subchannel simultaneously. In contrast, the DSSS technique divides the 2.4 GHZ band into 14 22-MHz channels. Adjacent channels overlap one another partially, with three of the 14 being completely non-overlapping. Data is sent across one of these 22 channels.

The key contribution of the 802.11b addition to the WLAN standard was to standardize the physical layer support of two new speeds, 5.5 Mbps and 11 Mbps. To accomplish this, DSSS had to be selected as the sole physical layer technique for the standard.

To support very noisy environments as well as spatial range, 802.11b WLAN use *Dynamic Rate Shifting (DRS)*, allowing data rates to be automatically adjusted to compensate for the changing nature of the radio channel.

The Data Link Layer

The *Data Link Layer (DLL)* of 802.11 consists of two sublayers: *Logical Link Control (LLC)* and *Medium Access Control (MAC)*. 802.11 uses the same 802.2 LLC and 48-bit addressing as other 802 LANs, allowing for very simple bridging from wireless to wired 802 LANs, but the MAC is unique to WLAN.
The 802.11 MAC is very similar in concept to 802.3, in that it is designed to support multiple users on a shared medium by having the sender sense the medium before accessing it. 802.3 Ethernet LAN use *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)* as MAC.

In a 802.11 WLAN, collision detection is not possible due to antenna limitations; a station must be able to transmit and listen at the same time, therefore it can not hear a collision. To account for this difference, 802.11 uses a slightly modified protocol known as *Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)* or the *Distributed Coordination Function (DCF)*. CSMA/CA works as follows. A station wishing to transmit senses the air interface and, if no activity is detected, waits an additional, randomly selected period of time and then transmits if the medium is still free. If the packet is received intact, the receiving station issues an ACK frame that, once successfully received by the sender, completes the process. If the ACK frame is not detected by the sending station, a collision is assumed to have occurred and the data packet is transmitted again after waiting another random amount of time.

CSMA/CA thus provides a way of sharing access over the air. This explicit ACK mechanism also handles interference and other radio related problems very effectively. However, it does add some overhead to 802.11 that 802.3 does not have, so that an 802.11 LAN will always have a lower data rate than a wired LAN.
Another MAC-layer problem specific to wireless is the *hidden node* issue, in which two stations on the opposite sides of an access point can both sense activity from an AP, but not from each other, usually due to distance or an obstruction. To solve this problem, 802.11 specifies an optional *Request to Send/Clear to Send (RTS/CTS)* protocol at the MAC layer.
Finally, the 802.11 MAC layer provides two other robustness features: *Cyclic*

Redundancy Check (CRC) and packet fragmentation. Each Packet has CRC checksum calculated and attached to ensure that the data is not corrupted in transit.

Association and Roaming

When an 802.11 client enters the range of one or more APs, it chooses an AP to associate with, based on signal strength and observed packet error rates. Once accepted by the AP, the client tunes in to the radio channel to which the AP is set. Periodically, it surveys all 802.11 channels in order to assess whether a different AP would provide it with better performance characteristics. If it determines that this is the case, it reassociates with the AP, tuning to the radio channel to which that AP is set. If two APs are in range of one another and are set to use the same or partially overlapping channels, they may cause some interference for one another, thus lowering the total available bandwidth in the area of overlap.

Security

802.11 provides MAC layer access control and an encryption mechanism, known as *Wired Equivalent Privacy (WEP)*, with the objective of providing WLANs security equivalent to their wired counterparts. For the access control, the ESSID (also known as WLAN Service Area ID) is configured into each AP and is required knowledge in order for a wireless client to associate with an AP. In addition, there is provision for a table of MAC addresses called an *Access Control List* to be included in the AP, restricting access to clients whose MAC addresses are on the list.
For data encryption, the standard provides for optional encryption using a 40-bit shared-key algorithm from RSA Data Security[1]. Beyond Layer 2, 802.11 WLANs support the same security standards supported by other 802 LAN for access control (such as network operating system logins) and encryption (such as IPsec or application-level encryption).

[1] http://www.rsasecurity.com

2.2 Simple Network Management Protocol

To counteract the problem that arises with the management of an ever increasing number of different products by different vendors that nowadays form the Internet, the *International Engineering Task Force (IETF)* decided to specify a standard to simplify the management of large, heterogenous networks.

In August 1988, the first version of the *Simple Network Management Protocol (SNMP)* [4, 5] was published. Despite of its name, SNMP is more than just a protocol. It's a complete management framework consisting of four major components:

- *Structure of Managed Information(SMI):* a structure and identification of management information for TCP/IP-based networks. The SMI describes how managed objects contained in the MIB are defined.
- *Management Information Base (MIB):* an information base for network management. The MIB describes the managed objects in the MIB
- *Standard MIBs:* a standardized core set of management information and events for TCP/IP-based networks (defined by the IETF).
- *Simple Network Management Protocol (SNMP):* defines the protocol used to manage these objects.

The initial targets for this effort were routers and switches. However, the SNMP-based management approach is inherently generic, so that it may be used to manage various types of systems. Today, SNMP is deployed in virtually all network-enabled devices.

2.2.1 The Network Management Model

A typical network management system consists of a manager, a set of managed systems or devices, an information database and a protocol for the communication between manager and system.

The manager provides the interface between the network operator and the devices being managed by processing and representing the collected information.

Associated with the managed objects on the target system are various attributes, which may be statically defined (e.g. the speed of the interface), dynamic (e.g. entries in a routing table), or require ongoing measurement (e.g the number of packets transmitted in a given time period). These are stored

in the management information database, called *Management Information Base (MIB)*.

An agent process that is running on the managed system acts as intermediary between the manager and the MIB. As depicted in Figure 2.3 the agent basically performs three different operations on the MIB on the behalf of the manager:

- *SET:* setting values to configures the device.
- *GET:* fetching requested information and forwarding it to the manager.
- *TRAP:* sending trap message to the manager when a defined event occurs. This is done autonomously by the agent process to reduce polling by the manager.

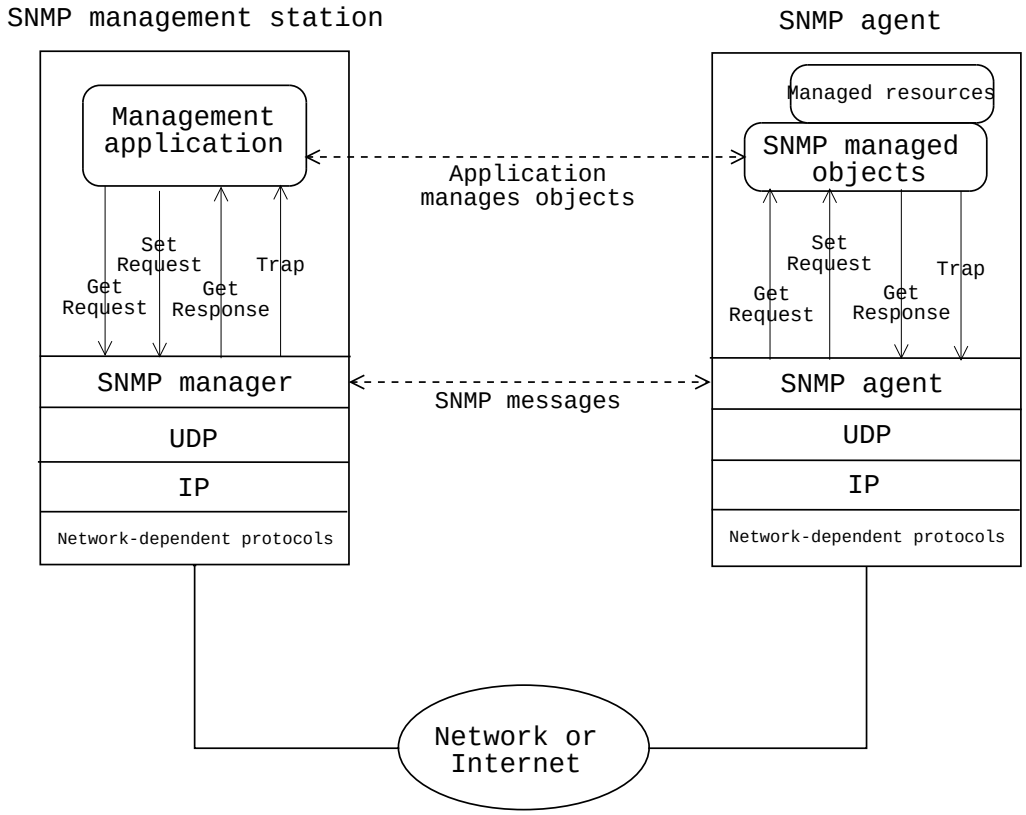


Figure 2.3: The SNMP Management Framework

The Structure of the Information Base

The SMI is a tree-like structure that organizes the managed objects by logical categories. The MIB represents the managed objects as leaves on the branches of the tree structure. Associated with each object in a MIB is a unique *Object IDentifier (OID)*. The OID is a vector of dot-separated integers that reflects the position of the object within the tree topology of the MIB.

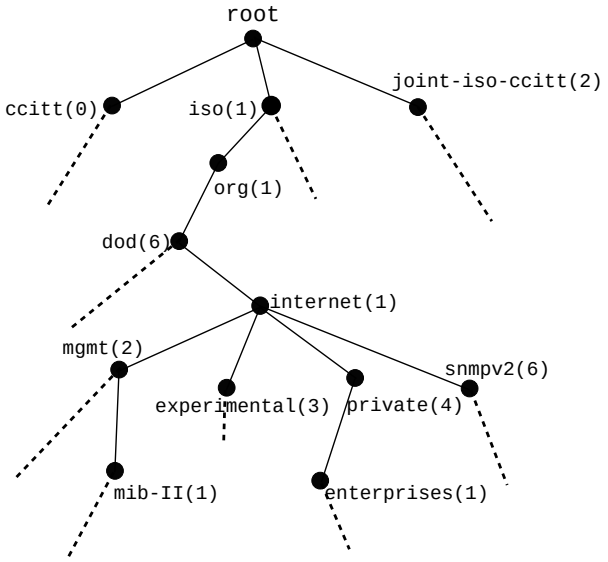


Figure 2.4: The OID tree with the standard SNMP assignments

- *MIB (.1.3.6.1.2.1):* A network management branch that contains the standardized core information about the system and its TCP/IP network interfaces. This is found on virtually all IP-enabled devices.
- *ENTERPRISES (.1.3.6.1.4.1):* This branch contains device-specific attributes, defined and implemented by the manufacturer of the device.However, how comprehensive and significant this information is depends on the manufacturer.

The Network Management Protocol

SNMP provides a way for the manager and the agents to communicate. To structure the communications process, the protocol defines specific messages, referred to as commands, responses and notifications. The manager uses these messages to request specific management information, and the agent uses them to respond. The building blocks of the messages are called *Protocol Data Units (PDU)*. For example, a manager sends a request PDU to retrieve information, and the agent responds with a response PDU.

SNMP was intended for use over a connectionless transport service, thus deploying the *User Datagram Protocol (UDP)*. The key reason for this is robustness. If SNMP would rely on the use of a transport connection, then the loss of that connection could impair the effectiveness of the SNMP exchanges.

Comment

As the name implies, SNMP is designed to be simple. It does this by means of three properties:

- reducing the development cost of the agent software. SNMP has decreased the burden on vendors who wish to support the protocol. This increases the acceptance of SNMP.
- SNMP is extendable since it allows vendors to add their own network management functions.
- it separates the management architecture from the architecture of network devices such as workstations and routers. This further widens the multivendor acceptance and support for this protocol.

SNMP is also referred to as simple because the agent requires minimal software. Most of the processing power and data storage resides on the management system, while a complimentary subset of those functions resides in the managed system. To achieve its goal of being simple, SNMP includes a limited set of management commands and responses.

Chapter 3

Network Monitoring

Network operators have a substantial interest in the state of their network components and their traffic volumina and they want to comprehend in which ways traffic flows through their networks.

Traffic information is vital for trouble shooting (detecting unusual behavior), for collecting usage data (logging, security, incident detection, billing), for capacity planning and to monitor the compliance of QoS requirements for *Service-Level Agreements (SLAs)*.

Monitoring tools are classified into two categories, according to type of functions and output they provide:

3.1 Fault Monitoring

Fault monitoring is a simple but pragmatic approach to network monitoring, as it focuses on the state of the various network elements only, i.e. whether an element is operational and whether it is accomplishing the services and tasks that are assigned to it. It doesn't provide any information on the performance or activity of the network and its nodes. Service faults are detected instantly whereupon the operator is alerted.
An example for a widely-used fault monitoring tool is Nagios [1], which performs scheduled tests of the state of hosts and checks whether services are operational. If a service is found to be not working properly, it is indicated on a web-interface and eventually a mail or pager message is sent out to inform responsible persons. Nagios is very common among ISPs, who want to be able to react quickly to possible service failures.

[1] www.nagios.org

3.2 Performance Monitoring

Performance Monitoring is a more sophisticated and more instructive concept, as it allows the measuring and tracing of the progression of the actual usage and performance of various network features, e.g. the output traffic of a router or the number of users logged onto a server.

Moreover, depending on the implementation, the progression of the network activity might be stored into a data base to allow further statistical processing and extraction. Such data is very interesting for Internet researchers for empirical validation of models and assumptions and for generating realistic input for network simulators.

Performance monitoring resp. measurement for large network structures is complex, and requires careful consideration of the following aspects (as pointed out in [6]), which are explained below:

- active vs. passive approach
- monitor placement within the network topology
- selection of useful metrics
- data collection and archiving

Active vs. Passive Measurement

Active measurements inject test packets into the network and trace their behavior, whereas passive measurements observe the actual traffic without generating additional traffic. However, passive monitors must process the entire load of a link to determine packets of interest, which might be problematic on high-speed links. Usually traffic flow data is collected either from routers and switches, or measured by stand-alone traffic meters.

Monitor Placement within the Network

Another crucial issue is where to place the meter(s) within the network infrastructure. One approach is to identify the links, which handle the biggest amount of traffic and place monitors there. Alternatively, one could begin by monitoring traffic at all border routers of the network.

Metric Selection

Popular network metrics are *throughput, latency, packet loss, link utilization* and as aforementioned *availability* (see the network measurement FAQ[2]).

Data Collection and Archiving

To allow statistical processing over a long period of time, the collected data must be archived. A common approach involves the building of a trace file repository that enables users to request a report on specific hosts, metrics or time intervals.

Once a measurement data repository is established, it is important to provide a clear, easy-to-use interface to the data. There is no point in collecting data if users can't access and interpret it easily.

[2] www.caida.org/outreach/metricswg/faq

3.3 Case studies

In the subsequent sections two widely-used network monitors are presented and studied: *Ntop (Network top)*[7] and *NeTraMet(Network Traffic Meter)*[8][9]. They are both performance monitors, that sniff the traffic on the transmission medium and count bytes and packets of different types of traffic flows. In the case of our monitoring tool, we have access to the MIB of the APs and can thus simply request these values from there.

3.3.1 Ntop

Ntop is an open-source web-based network usage monitor that enables users to track relevant network activities such as network utilization, established connections, network protocol usage and traffic classification. It is portable across various platforms and supports diverse network media. Moreover, it is user-friendly and its lightweight CPU utilization make it suitable for monitoring networks without deploying a sophisticated yet expensive management platform. Initially, it has been written for tackling performance problems of large networks.

The name Ntop was inspired by Unix' top tool, which reports and ranks processes according to their CPU usage on a running workstation. Its authors goal was to create a simple tool able to report the networks top users to quickly identifying those hosts that were currently using most of the available network resources. Ntop then evolved into a more flexible and powerful tool as people over the Internet downloaded it and reported problems and suggestions.
Ntop's main design goals are:

- portability across various platforms
- simple and efficient application kernel with low resource (memory and CPU) usage
- minimal system requirements
- user-friendly I/O-interfaces

The Ntop architecture basically consists of three components as illustrated in Figure 3.1.

The *Packet Sniffer (PS)* collects packets off the network that are then passed to the *Packet Analyzer (PA)* for processing. Whenever traffic information has to be displayed, the *Report Engine(RE)* renders the requested information appropriately.

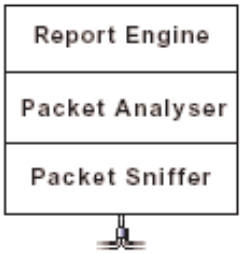


Figure 3.1: Ntop architecture.

The Packet Sniffer (PS)

The PS supports different network interface types including PPP, Ethernet and Token Ring and allows captured packets to be filtered before processed by the analyzer. Packet capture libraries usually have small internal buffers that prevent applications from handling bursty traffic. To overcome this problem Ntop buffers captured packets. This allows to decouple the PA from the PS and thereby prevent packet loss due to bursty traffic.

As modern switches allow global network traffic (virtual LANs) to be mirrored to a specific switch port, Ntop can operate on regular networks as well as on switched networks, if it is run on a host that is attached to such a switch port.

The Packet Analyzer (PA)

The PA successively processes the incoming packets. The Packet headers are analyzed according to the deployed network interface. Hosts information is stored in a large hash table whose key is the 48 bit hardware address (MAC address), which guarantees its uniqueness and allows different network protocols other than IP to be handled. Each entry contains several counters that keep track of the data that was sent and received by the host, sorted according to the supported network protocols. for each packet, the hash entry corresponding to packet source and destination is retrieved or created if not yet present. Periodically or if the size of the it exceeds a certain limit, Ntop purges the host table in order to avoid exhausting too much memory and creating huge tables that decrease the overall performance. Thus, it is guaranteed that Ntop's memory utilization does not grow indefinitely and that packet processing time does not increase linearly with the number of active hosts. If the received packet is not an IP packet, the protocol entry counters are updated and the packet discarded. Whereas, if the received packet is an IP packet, then further processing is performed.

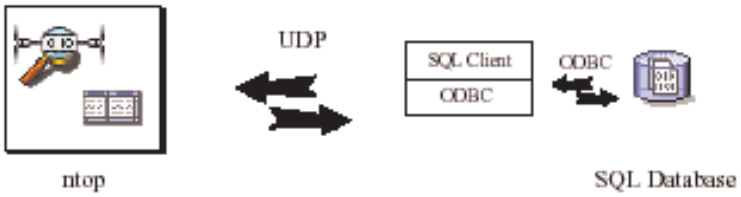


Figure 3.2: Ntop Caching Structure.

Ntop deploys a hierarchical caching with two layers (see Figure 3.2). The first level caching is semi-persistent and based upon GNU gdbm [3], a simple library for hashed databases.

The second level caching deploys an SQL database. The primary level caches locally semi-persistent information such as IP address resolution and remote host operating system. Network events (e.g. TCP sessions), performance data and other relevant information is stored permanently into the database. Storage happens either periodically or whenever the garbage collector has to purge data.

Ntop talks with the database by means of a client application. The client communicates with Ntop via UDP and with the database via the Open DataBase Connectivity protocol (ODBC). Whenever some network information is to be stored into the database, Ntop sends the client one or more UDP packets containing valid SQL statements. This architecture allows Ntop to be decoupled from a specific database and to be connected to the remote database (e.g. the main company database) while locally having a very simple and light database client.

The host entry shown below contains a counter for each of the user-specified IP protocols.

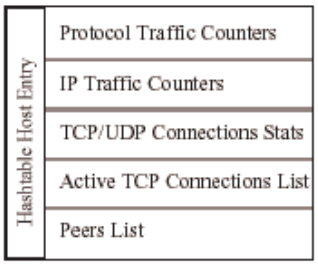


Figure 3.3: Ntop Data Struture.

For each IP packet, the appropriate protocol counter is updated. The

[3] www.gnu.org/software/gdbm/gdbm.html

host entry also contains a list of the host's active TCP connections. Ntop maintains the state of each TCP connection analyzing the IP flags. Hence, if the received packet is a TCP packet, then the host TCP connection list needs to be updated as well. Host traffic counters are convenient to analyze network traffic. However, in some cases it might be necessary to study specific traffic that flows through some specified hosts. Ntop allows users to specify network flows. Network flows can be very useful for debugging network problems, gathering statistical data or tracking suspicious access to some specified network resources.

Report Engine(RE)

The actual version of the RE of Ntop operates in two ways:

- *interactive mode:* Ntop runs as a character-based terminal application. The terminal is updated periodically as specified by the user. It shows general host information of all users and the according traffic that is generated.
- *web-mode:* Ntop acts as an HTTP server and allows users to analyze traffic statistics remotely by means of web browser. This mode has been designed as a long standing statistics gathering application able to provide users a detailed view of the current and past network activities. It turns Ntop into a full-scale web-based management application. Beside the information also shown in the interactive mode, the web mode contains additional statistics such as: IP multicast, traffic statistics, host information.

Conclusions

Ntop combines features otherwise only found in various tools that are not easy to integrate. It is easy to use and its unique user interface allows administrators to immediately take advantage of Ntop without the need to purchase and manage client applications that are necessary for tools such as NeTraMet. In addition, database support makes Ntop suitable not only for network problem debugging but also for long standing network monitoring. However, in its current form it does not provide any kind of fault monitoring and doesn't allow to monitor more than one network.

Especially interesting for this thesis is the concept of the hierarchical caching.

3.3.2 NeTraMet

NeTraMet is a tool for metering network traffic flows. It builds up packet and byte counts for traffic flows, which are defined by their end-point addresses. These counts are then fetched and aggregated by NeMaC.

It is the first implementation of the Realtime Traffic Flow Measurement (RTFM), the IETF's standard, generalized architecture for measuring traffic flows as outlined in RFC 1272 "Internet Accounting Background".
This architecture consists of four sets of distributed network entities:

- ***Meters***(NeTraMet), i.e. small hosts which are attached to a network segment and measure traffic flows and build tables of flow data. A meter is basically an SNMP agents who implements the RTFM meter MIB.
- ***Meter Readers***(NeMaC) retrieve flow data information from these meters using SNMP.
- ***Managers***(NeMaC) instruct meters which flows to measure and meter readers which meters they should collect data from, at what intervals.
- ***Analysis Applications***, which process the data from meter readers to produce useful, human-readable reports.

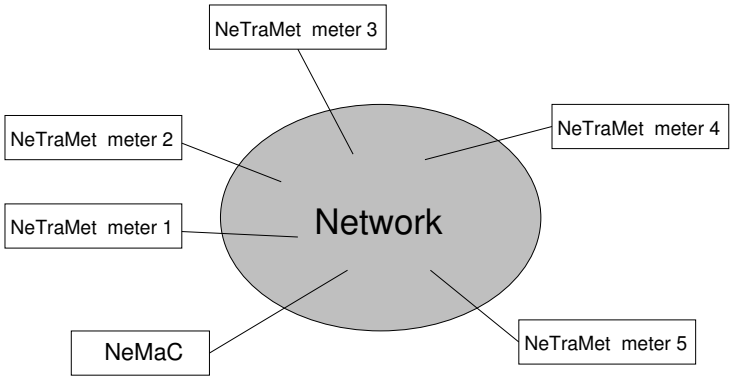


Figure 3.4: NeTraMet Monitoring Setup.

A meter reader can collect flow data from many meters, and each meter may have its data retrieved by several meter readers. Traffic flows of interest are defined by user-specified sets of rules.
A traffic flow is a stream of packets exchanged between two network host, which are referred to as flow's source and destination. For the definition of

traffic flow specifications, NeTraMet distinguishes the address attributes of a host by the following three kinds:

adjacent	link layer (MAC address)
peer	network layer (e.g. IP address)
transport	transport layer (e.g. port address)

NeMaC

As NeTraMet is only a meter, it is supplemented by *NeMaC (NeTraMet Manager/Collector)*. NeMaC is a combined manager and collector for NeTraMet. It's intended to provide control for NeTraMet, so as to make the initial NeTraMet implementation a useful and effective monitoring tool.

Implementation

Data Structures & Memory Management

The host addresses and the according masks are stored in structure referred to as key. Within the meter a flow is implemented as a larger data structure containing the keys of its source and destination, as well as its packet and byte counters, the time it was first and last observed, and other information used for control purposes.

As all times are measured relative to the uptime of the meter. Thus, a meter reader must read the current uptime of the meter as well as the time information for the flow analyzed.

All flows are stored in a flow table, which is implemented as an array of pointers to flows with corresponding flow indices into this array. The first time a flow occurs the meter allocates space for the flow, assigns it a flowindex and enters it in a count table that is implemented ass a single large hash table. NeTraMet maintains a table of these pointers to the count table.

To avoid a running out of memory as more and more flows are being set up, NeTraMet uses an incremental garbage collector. This searches through the flow table at regular intervals looking for flows which might be discarded. If a flow has not been active since the last interval, it is discarded.

Flow Selection

A rule-based selection process is applied to sort out the flows of interest. These rule sets are described in rule files and are basically table of rules, identified by their rule set number. The newer versions of NeTraMet introduced the *Simple Ruleset Language (SRL)* [10] and an according compiler. SRL is well-structured high-level language for creating rulesets. The SRL

compiler produces flow data files which NeMaC can read and download to a NeTraMet meter. Rule sets tend to evolve, one runs a ruleset for a time, analyzes the flow data it produces, then refines the ruleset as needed.

Each rule tests one attribute of a flow, using a mask to specify which bits are of interest. Each packet header has to traverse a tree of rules to be assigned to the according flow, where it is then counted. So when a packet reaches a meter, two key data structures are built, one for its source and one for its destination. The packet header is tested against the current set of rules with the keys in source-to-destination order. In case of a match the packet is counted. In case the match fails, the keys are reverted (destination-to-source) and the packet is tested against the rules again. If not, it is discarded.

A further possibility is that the packet matched a count rule, but its flow was not yet present in the count table. Since it might already have been observed travelling in the opposite direction the match is retried with the keys interchanged. If it fails the flow is added into the count table with its keys in source-to-destination order.

NeTraMet's outer Loop
NeTraMet's outer loop implements four asynchronous processes. These are (in decreasing priority order):

- Handle SNMP requests. Process these and send according SNMP responses.
- Monitor Ethernet packets. Test each against the active rule sets and counts required.
- Handle keyboard commands
- Memory Management. Attempt to recover memory, as described above.

Measuring traffic flows with NeTraMet/NeMaC
To observe traffic flows in a network, the NeTraMet meters have to be placed at points where traffic is to be measured, and one or more NeMaC managers/meter readers on convenient hosts within the network. NeMaC configures the meters by downloading the according rulesets to them, reads and collects data from meters at regular intervals and writes it to flow data files.

Conclusions

According to the RTFM concept, NeTraMet deploys freely parameterizable flow or stream specifications for monitoring, by using a high-level language to configure traffic meters. This flexibility allows large scale monitoring of extensive network structures in coexistence with a surveillance of specific single connection flows and streams. NeTraMet's ability to provide real-time distributions for any required flow in a flood of packets is particularly useful for production network monitoring. Interesting case studies are found in [6].

However, the downside is that it requires a sound knowledge of he network infrastructure and a thorough planning of the deployment of NeTraMet , i.e. what links to monitor, where to place meters and meter readers.

Interesting for this project is the idea of task sharing between NeTraMet meters and managers.

Chapter 4

The StockholmOpen Network

4.1 Introduction

StockholmOpen[1] is based on the idea of creating a city-wide WLAN as a cooperative effort of various ISPs. This project commenced in 1999 at the IT-University as a research project on wireless network with mobility support on the campus area.

The positive results of this project were motivation to extend the scope of the project to developing an infrastructure providing wireless network access, not limited to a private network inside the IT-University, but what was visioned to be a public operator neutral access network [11] for the entire Stockholm metropolitan area.

As a first step towards this ambitious goal, a new access mechanism and a new authentication mechanism were developed and deployed in August 2001. Thenceforward, the wireless access network has provided service to more than 600 users, including students and staff of the IT-University and subscribers of the various ISP's sponsoring the StockholmOpen project. Currently, nearly 80 AP's are deployed on the Kista campus and the city of Stockholm. For now, the AP's are based on the IEEE 802.11b standard. The access concept is applicable for any IEEE 802.2 compatible link level technology.

There is no restriction concerning type or vendor of the AP, the only requirement is WiFi certification, to guarantee high quality wireless connections and a high level of compatibility.

[1] www.stockholmopen.net

4.2 The Access Concept

To keep the level of mutual trust between the ISPs low, the common access network itself only provides two services:

- user self-registration to an ISP via web interface
- client IP address assignment via DHCP relay to the according ISP

The access network server relays the users configuration request to the chosen ISP who in turn provides the client with the correct network configuration. *Authentication, Authorization, and Accounting (AAA)* is conceded to the particular ISP. Thus, no confidential user data is managed by the shared infrastructure.

To enable new users to register, the access network provides basic services to any client, including users that may not have access to any of the participating ISPs, but still may want to access locally available services.

To meet these requirements, the access mechanism deploys the *Dynamic Host Configuration Protocol (DHCP)* and the *Bootstrap Protocol (BOOTP)*[2] in a new and innovative way. It uses a BOOTP relay service to serve the subscribers of the participating ISPs.

The novel aspect of the implemented solution is that the BOOTP relay agent is able to relay DHCP requests to the DHCP servers of the different ISPs depending on the requesting clients MAC address (see Figure 4.1).

Based on the information stored in the access networks local database, a clients DHCP request is relayed to the ISP where the client is registered. Thereby, the ISP maintains complete control over its DHCP server and there are no changes in the server or the protocol.

When a new client connects to an AP for the first time and sends out a DHCP request, a temporary private IP address is assigned. Thus, an unregistered user can access the registration service as well as other locally available services.

The registration service is used via a web interface. This web page is the user interface to the relay database, it allows user to select their own ISP from a given list. When a user selects an ISP, the MAC address of the client is registered in the database as belonging to the selected ISP.

If a user changes ISP, he will get a new entry in the relay database next time the so-called *lease time* expires. The lease time is kept short to minimize the adaption delay. A new client registering for the first time can get a new configuration in a few seconds without restarting the client network service.

[2]BOOTP allows a client machine to discover its own IP address, the address of a server host and the name of a file to be loaded into memory and executed

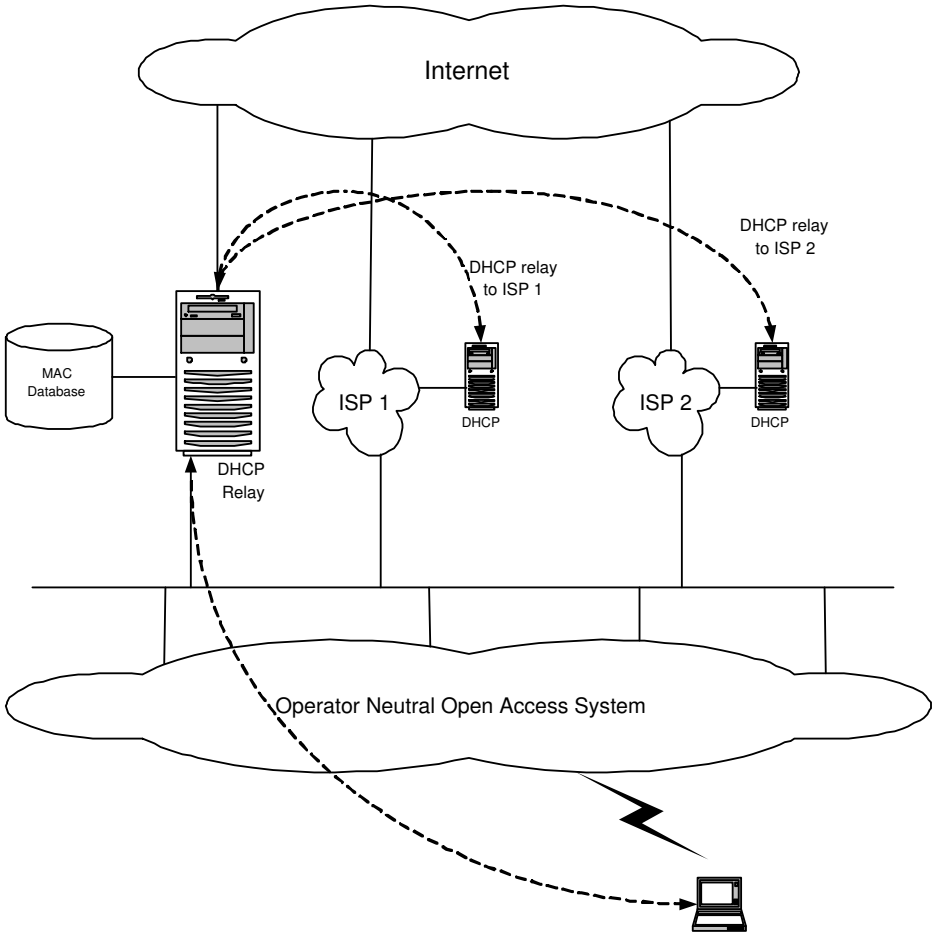


Figure 4.1: The Structure of Stockholm Open.

As an improvement of this access concept, a new AAA system has been developed to integrate any existing authentication mechanisms that are already in use. This AAA consists of two parts:

- an authentication/authorization service that provides an SSL secured, web based front for a pluggable back-end.
- a firewall/gateway

Once a user is authenticated and authorized, the system opens a specific rule in the associated firewall of the ISP to allow the users traffic to cross the firewall to access the actual upstream/downstream network. This traffic is identified by MAC address and IP address. The access is only valid as long as the client is present in the system. When an authorized client leaves the network, the AAA system closes the according rule in the firewall.

This means, that users have to log in each time they want to access the upstream/downstream network. The AAA system probes the presence of each client in the network by sending either ARP or ICMP echo requests to the client.

The StockholmOpen access network has been running successfully with hundreds of users. However, it's still evolving. Based on the gained experiences, the development continues into various directions in order to provide better scalability, improve the AAA system and make the network more transparent to the user.

4.3 The Access Points of StockholmOpen

In its current state, Stockholm Open is formed by 79 APs of the following types:

- 3COM AP-4111
- Breezecom BU-DS.11
- Cisco Aironet 1200
- Ericsson AP A11
- Lucent WavePOINT-II
- Orinoco Residential Gateway I
- Orinoco Residential Gateway II
- Orinoco AP-500
- Orinoco AP-1000

Proxim, which is the producer of the Orinoco Wireless line acquired the Agere line from Lucent in August 2002. Thus, all Orinoco APs and the Lucent WavePOINT-II have the same MIB, which is provided by Karlnet.

Thus, with the name Orinoco APs we henceforth refer to all Orinoco APs plus the Lucent WavePOINT-II, as all of these provide the same monitoring interface (MIB).

To simplify matters, we focus on the APs that cover the buildings of the IT-University in Kista. This subset consists of Orinoco and Cisco APs only and amounts to 53 APs. The reporting features of these two AP types are fairly different and representative for a large set of AP types.

4.3.1 Reporting Features

Besides the standard MIBs (defined by the IETF), the reporting features of APs differ considerably. In 1998 the IEEE published an official MIB for the 802.11 standard to counter this problem. This MIB is a very comprehensive complement to the standard MIBs and defines a lot of useful management data and notification events for WLANs, e.g. a list of authenticated or associated stations and according notifications.

Unfortunately, this MIB is not or only partly implemented on the present APs and thus scarcely useful for our purposes.

However, the standard MIBs for TCP/IP-devices is available on all APs and provides valuable information, such as various information about the

state and traffic of the present interfaces, the type of interface, the number of bytes that were received or sent, the number of corrupted packets received or the routing table of the AP.

The following two sections describe the monitoring information available from the investigated APs.

Orinoco AP

The enterprises branch of the MIB of the Orinoco APs is by Karlnet and contains a lot of useful information about all associated stations as for example the stations name, the MAC address or the number of packets sent and received. To make the AP refresh these values, it must be triggered[3], i.e. before the desired items may be requested, certain values must be set in the MIB.

Moreover, the MIB defines notifications for deauthentication and disassociation of stations, but none for authentication and association of the latter. These notifications are thus not particularly useful for the present monitoring purposes.

Cisco AP

The enterprises MIB of the Cisco APs is very comprehensive, yet useful information for monitoring is scarce. The only valuable item is a gauge with the number of the currently associated stations and a number counters for mobility mutations. In theory, there are objects for the MAC addresses of all associated stations. In practice, however, these fields are set to zero by default. The same practice was observed as well for other items in that MIB or in other MIBs on other APs. However, a list of the MAC addresses of all authenticated Stations can laboriously be extracted from the Bridge-MIB.

Besides the MIB, the Cisco APs provide a convenient and concise web interface for configuration and supervision. Oddly enough, this interface features all the information missing in the MIB, as e.g. the MAC and IP address of all authenticated or associated stations, the transmitted packets and bytes et cetera.

Moreover, the Cisco APs can be configured to send syslog messages to a specified remote host upon a specified events, e.g. the authentication/association respectively deauthentication/deassociation of stations or loss of connection to the backbone network. This feature is very useful, as the APs don't have to be polled for mobility mutations, but they report them autonomously. The drawback of this is

[3]http://edge.mcs.drexel.edu/GICL/people/sevy/airport/MIB.html

4.4 The existing Monitoring Tool

Currently, Stockholm Open is monitored by a self-made perl script that fetches traffic information and the number of associated stations from all APs and writes this values into a database. Furthermore, the collected data is processed and presented on a web page.

Although the monitoring tool runs on a workstation driven by a 1.4 GHz Pentium processor and with 256 MB RAM, it cannot comply with the scheduled 5-minutes update intervals. The reason for the latter is primarily that it deploys the net-snmp[4] shell tools to poll the APs. These are very resource-intensive, e.g. a simple request takes about 200 ms.

This shortcoming combined with the anticipated growth of Stockholm Open was one of the main motivations for this thesis.

[4] net-snmp.sourceforge.net

Chapter 5

The Monitoring System

The aspired monitoring system is suitable for large WLANs formed by various types of APs from different manufacturers. This kind of networks bear various challenges:

- Traditional monitoring concepts require a large amount of data to be sent around and a lot of processing power to handle it.
- The reporting capabilities of the APs are very limited and usually not alterable or extendable.
- Although there exist some standards, the provided monitoring interfaces vary considerably among the different manufacturers (see Section 4.3), i.e. different MIBs, different *Command Line Interfaces (CLIs)* or different proprietary protocols.

5.1 System Requirements

Firstly, the monitoring system should provide basic information about the APs and the WLAN, such as the traffic on the up- and downlink of the APs or the states of the APs.

But beyond that it should also provide the operator with WLAN-specific information, e.g. the number stations that are connected the an AP or their MAC addresses.

Moreover, it should store the resulting data into a database and provide an interface for the administrator to easily access and interpret the collected data.

Furthermore, to overcome the above-mentioned difficulties and to be apt for a provider-neutral WLAN like StockholmOpen, the system has to meet

some special requirements, apart from the classical monitoring requirements of fetching and processing data from the monitored devices.

These special requirements are the following:

- The system has to be very scalable, as Stockholm Open is expected to grow considerably to eventually serve the entire metropolitan area of Stockholm (see Chapter 4).
- The system should provide a hardware abstraction to overcome the heterogeneity introduced by the different types of APs.
- The system should be modular to support an easy integration of new models and types of APs.
- For practical reasons, the system should be reconfigurable *on-the-fly*, i.e. assigning a new AP to the monitoring should not a require a reinitizialisation of the entire system.

5.2 The Data Model

The various types of APs deployed in StockholmOpen are presented in Chapter 4.3. As mentioned, we focus on the Cisco APs and the group of APs we refer to as Orinoco APs. For a simultaneous monitoring of these two types we can only utilize monitoring information that is available on both, i.e. the intersection of both information sets.

In this case, this is only the number of bytes that crossed the WLAN interface and the number of associated stations. A counter of bytes that left, resp. arrived at the AP's wireless interface are found at .1.3.6.1.2.1.2.2.1.10, resp. .1.3.6.1.2.1.2.2.1.16 in the standard MIB of every AP. From the number of bytes we can derive the downlink and uplink datarate on the WLAN interface. Thus, the monitoring system would provide the same information as the existing monitoring tool (see Section 4.4). However, by abstracting from data that is different on both AP types, but refers to the same properties we can gain additional information.

Namely, we can derive information about association mutations from the list of associated stations in the MIB of the Orinoco APs by observing the changes in that list. By comparing this list at two different points in time, we can deduce what stations have left the AP respectively what stations have arrived between these two points in time.

As mentioned in Section 4.3, the Cisco APs provide notifications about association mutations, transferred by syslog. By introducing a *Hardware Abstraction Layer (HAL)* to hide these two different information sources (MIBs

and syslog messages), we can gain information about association mutations on both types of APs.

For the Orinoco APs, the HAL has to poll the list of associated stations in short time intervals and deduce mutations and for the Cisco APs, it has to interpret the notifications that are sent by the monitored APs. Mobility tracking is thus much less expensive on the Cisco APs than on the Orinoco APs, but as the Cisco APs don't actively poll their associated stations to check who is effectively around, there is no way to tell whether a station has left the WLAN or it is just inactive (if it has roamed, it sends out a an according message). After 30 min of inactivity the AP deauthenitcates the station and sends an according message. Thus, if a station that is connected to a Cisco AP leaves the WLAN, it is noted only half an hour later.

Collecting the association mutations of all APs in the entire network, we receive the information about the mobility of all stations that are associated to the network. We can thus provide mobility tracking of all stations. However, this requires a global view of the network, i.e. of all APs in the network. Moreover, the order of arrival of these mutations is crucial, as a false order provides a wrong picture of the association situation in the network. The association mutations are thus not superposable, but bound to certain timing restrictions.

The intersection of the available information on both AP types and the abstraction of the mobility is illustrated in Figure 5.1. Data abstractions are marked with dashed arrows.

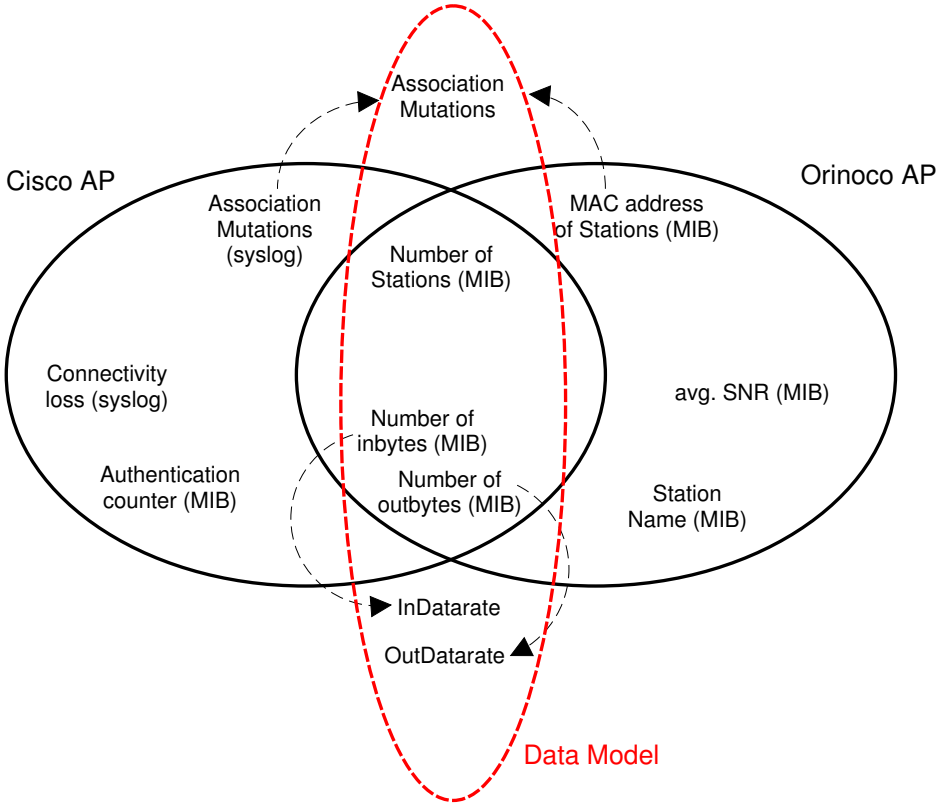


Figure 5.1: Abstraction and Derivation of the Data Model

Thus, we obtain the following information from both of the selected AP types:

- the state of the AP
- the datarate of the incoming traffic on the 802.11b interface
- the datarate of the outgoing traffic on the 802.11b interface
- the number of stations that are associated to it
- the MAC addresses of the associated stations, resp. their mobility

We can roughly categorize these information items into two classes of a very different kind, which require a different treatment as well:

Traffic Data

The information about the number of bytes that has crossed the wireless interface are simple counters that are usually polled in intervals of a few minutes, which allows to easily calculate the datarates on interface. This kind of information is not time critical.

Event Data

There are four different kind of events that can be deduced from the available data:

- a station has connected to an AP
- a station has disconnected from an AP
- an AP is not reachable any longer (is down)
- an AP is operational again (up again)

The occurrence of such an event is sporadic and should be discovered and reported as soon as possible.

5.3 System Design

The monitoring system that is implemented within the scope of this thesis aims at meeting the aforementioned requirements. Its design is presented in the following sections. The achieved improvements are subsequently discussed in Chapter 6.

5.3.1 System Architecture

One of the central questions that arises when designing a system, is how to structure the system. In this case, there are three possible architectures, which all have their advantages and drawbacks.

- *Centralized:* The centralized approach is based on one monolithic system that copes with all required tasks. Its main advantage is its simplicity and its presumably low maintenance expenses. However, its capacity is strictly limited by the processing power of its host station and can only be improved by upgrading the latter.

- *Distributed:* In a distributed system the workload is distributed among a number of agents, which are controlled and coordinated by a manager, who eventually collects and processes the resulting data. This architecture provides a much better scalability than the centralized, but requires more efforts concerning coordination and maintenance.

- *Peer-to-Peer:* In this architecture the workload is distributed among various units, called peers. There is no manager to control or coordinate the peers, but they organize themselves. This sort of organization scales very well in terms of processing power, as there is virtually no limitation on the number of peers that can cooperate. On the other hand, the amount of coordination traffic grows exponential with the number of APs.

For the present task, the centralized approach is definitely not appropriate, as it does not comply with the stated scalability requirement and would sooner or later fail to handle the increasing number of APs of StockholmOpen.

The peer-to-peer structure on the other hand would definitively feature the necessary scalability, but network-wide time critical monitoring items, such as station mobility make its realization very difficult, even if the organization of the monitoring system is static.

One of the main problems is where to keep the data of stations that roam the network. Storing it on all peers leads to a vast amount of inter-peer update traffic and provides various synchronization and consistency problems. To have the station data distributed among the agents leads to expensive searches to find the agent the station is connected to and presumably to load disbalances among the agents.

The distributed architecture is much more appropriate for network-wide data like the above-mentioned station mobility, as it provides a central management unit, where the association mutations of all APs can be gathered to give a complete picture of the mobility of all stations in the network. Moreover, this structure is also favorable to archive the collected data, as all data of the Agents is transferred to the central Manager, who then continuously writes it do a database attached to it.

Furthermore, a distributed architecture may reflect the segmentation of the composite WLAN into its administrative domains, e.g. one agent for all APs of an ISP, and the Agents of all ISPs are connected to the central manager of the WLAN. This facilitates a clear structuring of the system. Furthermore, the traffic caused by the polling of the APs is kept within local network infrastructures.

The bottleneck in a distributed system is usually the Manager. Thus, to maximize performance and scalability of a distributed system, as much workload as possible should be deprived from the Manager and distributed among the agents, i.e. the manager should only have a minimal number of tasks assigned to maximize the number of agents he can manage.

For the present situation, it is evident to delegate the polling of the APs to the agents in the network, which are in turn polled by the manager to send him the collected information in a consolidated form.
Thus, every Agent has a number of APs assigned, which it autonomously polls. It then processes and consolidates the collected data to send it to the manager when polled by it. Thereby, the traffic to the manager is minimized and controlled by the latter, i.e. the risk of traffic congestion on the link to the manager is minimized.

5.3.2 System Topology

Up to now we used the terms manager and agent very general. From now on, when talking about the design of the envisioned monitoring system, we refer to them as Manager and Agent with capital initial letters.

As mentioned above, the traffic data and event data require a different treatment. The latter should be forwarded as fast as possible to the Manager, while the former is much less time critical. Therefore, we deploy two different logical system topologies for them, which is outlined in the following two sections.

Traffic Data

Traffic data is updated periodically and does not occur sporadically like event data. Thus, the corresponding items are stored on the Agents. To send it to the Manager, the Agents pack the updated data into strings that have the maximal size of the payload in a TCP packet, so that the number of packets, resp. the overhead is minimized. The Agents are polled by the Manager periodically, who thereby fetches the latest packets once per update interval. The processing power of the Manager, resp. how many Agents he can handle per unit is thus crucial for the performance of the system.

Traditional distributed architectures have a two-level topology where all Agents on the lower level are directly connected to the Manager.

However, to further disburden the Manager we introduce more levels by interconnecting the Agents and arranging them in a tree topology, as illustrated in Figure 5.2. We enumerate the levels, starting at the Manager which is on level 0 and the Agents directly below him are on level 1, et cetera. Thus, the manager is connected to the Agents on level 1 only, which in turn are connected to the Agents on level 2 and so on. An Agent has exactly one parent[1] and zero or more children[2].

Each Agent, upon receiving a request, first forwards the request to his children and waits for their responses, respectively the requested data. As soon as the transfer of the data from all children is completed, the Agent forwards his own data together with the data of his children to his parent, whence he originally got the poll request. Thus, the poll request "walks" the tree of Agents and the data of all Agents of a branch [3] is aggregated in the top-level Agent, who forwards it to the Manager to complete the request.

[1] For an Agent on level n the parent is the Agent on level $n-1$ he is connected to.

[2] As children we refer to all Agents on level $n+1$ an Agent on level n is connected to.

[3] A branch consists of a top-level Agent connected directly to the Manager and all Agents below it, which have to pass that top-level Agent to reach to Manager.

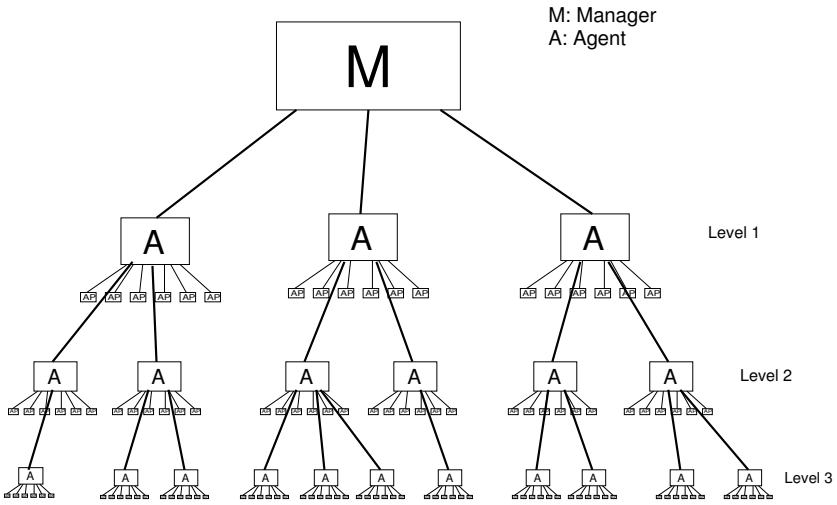


Figure 5.2: The Topology for Polling the Agents

The advantage of this topology is that the Manager only has to handle the top-level Agents, whence his requests are propagated through the topology of Agents,the required data gathered and then transferred to the Manager. Thus, also the task of polling the Agents is distributed among the Agents and the workload of the Manager further reduced. The Manager only sees one Agent per branch, all other Agents of a branch are hidden to him, which reduces complexity.

The drawback is that the delay introduced by every single Agent adds up during the polling of a branch. However, during the waiting time for a poll request to return, the Manager is largely idle and might handle other tasks.

Event Data

The event data is sporadic and should be processed as fast as possible. Thus, the Agent frequently has to check for events and, in case an event occurred, send a corresponding event message directly to the master.

As a consequence, the Agents are not interconnected, but the Agents are all on the same level (see Figure 5.3) and they don't have a permanent connection to the Manager, but just open a connection to it, when an event occurs. This connection is closed immediately after the transmission.

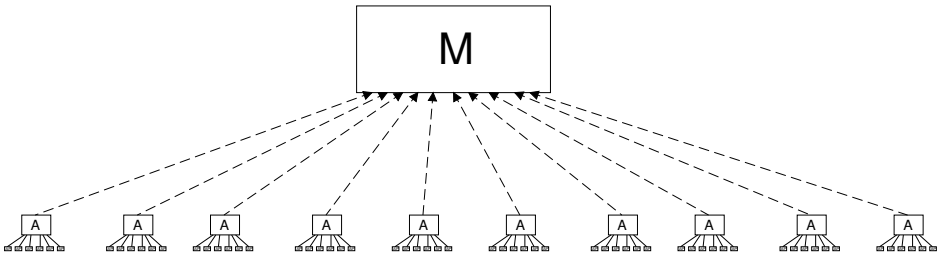


Figure 5.3: Topology for the Transmission of Event Messages

5.3.3 The Agent

In the subsequent subsections the design of the Agent as it has been implemented is presented.

Specifications

- The Agent has an internal data structure, where it caches specified data about all APs that are assigned to it.
- It provides a HAL (see Section 5.2) to allow monitoring of different types of APs.
- In regular time intervals, the Agent autonomously polls the assigned APs for specified information, processes the collected data and updates its internal structures.
- After all APs are updated, it fetches the updated data from the internal structure and fills it into a packet to send them to the parent when a poll request arrives.
- It opens a connection to its parent and a socket for his children to connect to. Poll requests of its parent are forwarded to its children. As soon as all children have transmitted their packets, all data is sent to the parent.
- It checks for events in short time intervals and, in case of occurrence, sends an event message to the Manager.

The Design of the Agent

A scheme of the design of the Agents is illustrated in Figure 5.4. At the beginning of every update interval the outer loop of the Agent sequentially invokes the *update_traffic* routine and the *update_mobility* routine.

The *update_traffic* routine fetches the current values for the traffic of the APs and overwrites the old values in the AP structure with it, whereupon the new values are packed up into packets.

The request handler runs in the background and checks incoming requests from the parent. If a request comes in, he first polls all children and then sends all packets up to the parent. To avoid concurrent access to the packets by the packet handler and the packer function, the packets are protected by a mutex semaphore.

If the mobility routine finds association mutations on one of the assigned APs, it sends an according event message to the Manager.

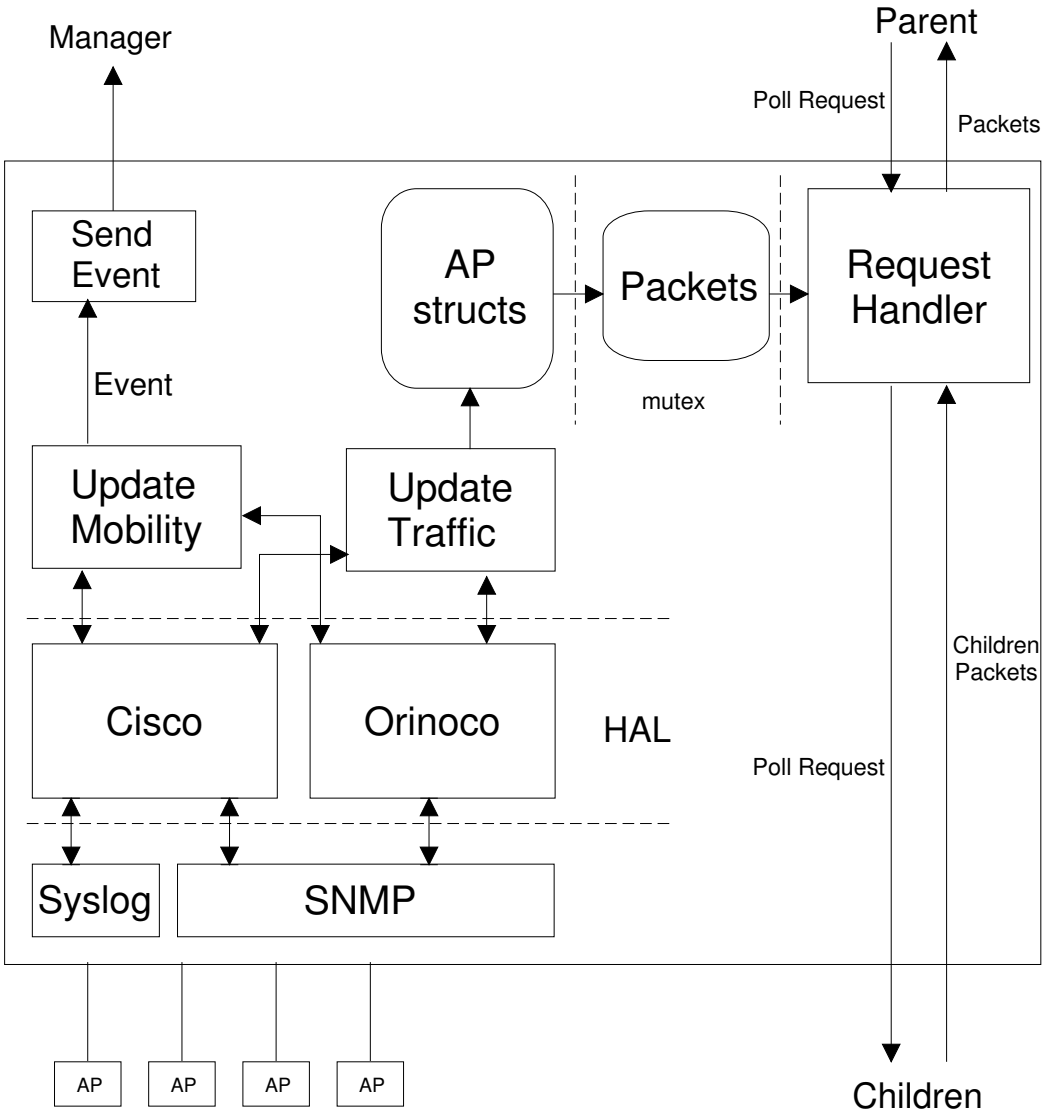


Figure 5.4: The Design of the Agent

5.3.4 The Manager

Specifications

- The Manager has to poll all branches in the specified update interval.
- It has to process all received data and cache it into internal structure, resp. into an attached database.
- It has to process the received data by computing network-wide and individual averages, maxima et cetera.
- Simultaneously, it has to receive and process incoming event messages to update the mobility situation within the network.

The Design of the Manager

The Manager has two processes running in parallel:

- *Traffic:* A process that polls all branches once per update interval. The thus acquired data is being cached in an internal structure. As soon as all APs are updated, the statistics of the single APs and of the entire network are updated before they are written into an attached database.
- *Events:* This process waits for event messages to come in and update the corresponding session[4] structure, i.e. if the station leaves the AP, the time is noted in the structure and if the station connects to a different AP, a new association entry is opened within the session structure. Because of the above mentioned timing issues for mobility, we introduce a *time-to-live (TTL)* for a session. This time to live is expressed in update intervals and starts to expire, if a station has disconnected from a station, but not yet reconnected to another station. The TTL is from then on decreased by one every update interval. If the TTL reaches zero before the station has reappeared in the network, the session is declared dead and the station starts a new session the next time it appears in the WLAN. If it reappears before the TTL is zero, the session is continued. Thus, the mobility tracking is more resilient and

[4]With the name session we refer to the time a particular station is connected to any AP within the WLAN. The session starts when the station first connects to the WLAN and ends when the station disconnects from the network and not reconnect for specified amount of time. If it reconnects later, it starts a new session. Thus all stations that are connected to the network are in a session. The session structure contains data about what stations the station has been connected to and the exact times if association and deassociation.

the aforementioned timing restrictions are partly mitigated. At the end of every update interval, dying sessions are "given the count", i.e. their TTL is decreased by one and dead sessions are written into the database.

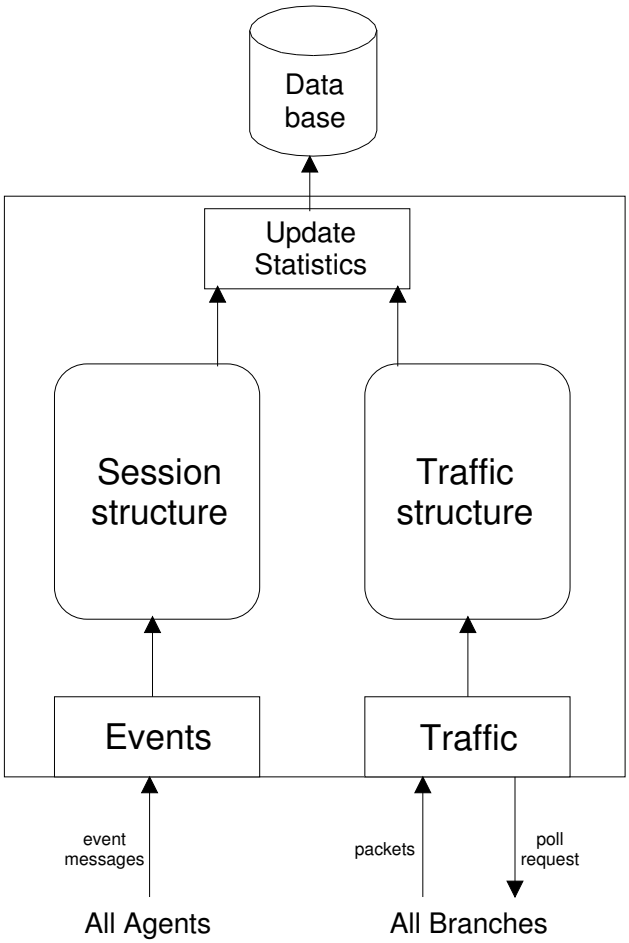


Figure 5.5: The Design of the Manager

Chapter 6

Evaluation

6.1 Proof of Concept

In the subsequent sections, the key ideas of the design are being discussed with regard to their implementation. These key ideas are namely the following three:

- distributing as many monitoring subtasks as possible
- interconnecting the Agents and arranging them in a tree topology
- introducing mobility monitoring

6.1.1 Distributed Monitoring

The main motivations to distribute the polling of the APs to several Agents are the following (as pointed out in Chapter 5.3):

- *Performance Scalability:* To assure a good scalability of the monitoring system, the central unit should be deprived of as much workload as possible. Therefore, the polling of the APs is delegated to Agents within the backbone network.

- *Traffic Scalability:* To avoid a local congestion towards the centralized monitoring unit. By delegating the polling to Agents the polling traffic can be kept local and thus low, if the Agents are deliberately placed within the backbone LAN.

- *Administrative Domains:* As a WLAN like StockholmOpen is a composite of numerous WLANs by different ISPs, the monitoring system should reflect this structure, so that the APs of an ISP are monitored by a number of Agents running in the ISPs network. Thus, the polling

traffic is kept local and has to cross only a minimal number of networks. The Agents of all ISPs are managed by a central manager, who collects and represents all data.

Centralized vs. Distributed

To demonstrate the benefits of distributing the monitoring task we compare our implementation to a notional centralized monitoring system that is based on the same components as our system, i.e. it needs the same processing resources and the same time to poll a single AP. To simplify matters, we focus in this section on traffic monitoring only.

In such a system, the dispatching of poll packets and the processing of the returning responses is clearly a considerable bottle neck. Therefore, we analyze the number of packets heading for the master and the according amount of data to process for both scenarios.

To monitor the traffic a monitoring system needs to poll the number of outbytes and the number of inbytes of the WLAN interface on each AP. This is accomplished with a single SNMP request. A request consists of a request packet and the according response packet (UDP). Such a SNMP request packet has a size of about 100 bytes, whereof about 30 bytes are relevant for the monitoring process. The response is about 110 bytes long, 10 of which are relevant. For the two different concepts this means the following:

- *Centralized:* The central monitoring unit has to poll every single AP, i.e. compose and dispatch a SNMP request, wait for the answer of the target AP and process the returning response packet, which contains only ten to twenty percent relevant information.

- *Distributed:* In the distributed system the polling of the APs is delegated to the Agents which in turn merge the traffic attributes of all their APs into one or more TCP packets. One TCP packet can hold the traffic data of about 28 AP (about 50 bytes per AP). These packets are then passed on through the tree topology of Agents to the Manager. Assuming an Agent topology with one single branch the latter ideally receives one TCP packet per 28 APs plus two small control packets to mark the initiation and the end of the transfer. These TCP packets consist of about 95% relevant data.

Figure 6.1 illustrates the number of packets that the Manager has to process in both scenarios as a function of the monitored APs.

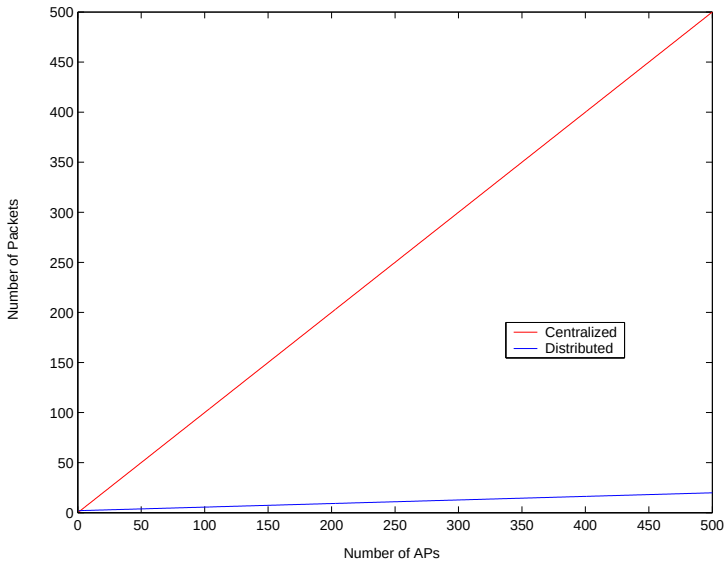


Figure 6.1: The Number of Packets sent to the Manager

6.1.2 Interconnecting the Agents

By interconnecting the Agents and arranging them in a tree topology, we reduce the number of direct connections to the Manager. This has the following advantages:

- The Manager has to deal with the top-level Agents only. Agents on lower levels are hidden to the Manager, and the work of polling the Agents is distributed among these, which further reduces the workload of the Manager.

- A further reduction of the traffic to and from the Manager, as the overhead is minimized by using the whole payload of the TCP packets. This minimizes the risk of congestion on the link to the Manager.

- The system allows for various topologies, i.e. depending on the available hardware the topology may be adapted to suite the processing power of the single hardware the Agent runs on, e.g. Agents running on a minimal hardware or as a daemon on a hardware with multiple purposes, may have no children connected to it (i.e. leaves of the tree),

while Agents running on a more powerful hardware may be placed close to the Manager in the topology, with a number of children connected to it.

6.1.3 Mobility Monitoring

The implementation of the concept, as outlined in 5.3.2, evinces the feasibility of mobility monitoring with the present APs. However, on the Orinocos the polling of all associated stations is very costly with the current implementation and reduces the overall performance of the Agents considerably (see Section 6.2).

The resolution of the mobility tracking (i.e. how often the associations of a all APs can be updated) is thus heavily dependent on the number of Orinoco APs that are assigned to an Agent.

With the Cisco APs, on the other hand, we have a dealy of half an hour for stations leaving the network, which impairs the system's accuracy for mobility tracking.

6.2 Performance Evaluation

In this section the evaluation of the implemented system is presented and discussed.

6.2.1 The Agent

The Agent is evaluated on the Cerfcube (see Appendix A), as it is planned to run on that hardware. Furthermore, it gives an indication about the performance of the Agent on a minimal, inexpensive hardware. The series of performed experiments provided very constant results and a very small deviation. The Cerfcube and all APs are located in the same LAN.

In this evaluation we disregard the initialization sequence of the Agent as it is not expected to be reinitialized often.

The performance of the Agent is determined by the number of APs whose monitoring data is updated per time interval, respectively the minimal time interval required to update the information of a given number of APs.

The time that is required by the agent to update the information of a given number of APs is referred to as t_{update}, which corresponds to the sum of the times required by the various updating subtasks for all APs. Table 6.1 gives an overview of the involved subtasks with the corresponding durations. These times are averages that resulted from 10000 test runs performed on the Cerfcube.

Updating Subtask		Orinoco AP	Cisco AP
Traffic information	t_t [ms]	10.4	15.2
Mobility information	t_m [ms]	552	<1 , $\forall$ APs
Pack updated information	t_{pack} [ms]	2.9	2.9
State check	t_{down} [ms]	521 (5)	521 (5)

Table 6.1: Average durations of the various monitoring subtasks

A few words about the most relevant items of this table:

- *Traffic Information:* Since the cube is located in the same LAN as the APs, the transmission times of the packets make up only a small fraction of this values $(0.2 - 0.3ms)$. It's interesting to observe that the average response time varies considerably among the different products, e.g. the Cisco APs takes about about 50% longer to for the same task than the Orinoco APs .

- *Mobility Information:* The values confirm that the mobility tracking on the Orinocos is particularly costly in the present implementation (as mentioned above), whereas the mobility updating of the Cisco APs is very inexpensive, since the Agent doesn't need to poll any APs, but just has to interpret the syslog messages that were collected in a pipe. In a WLAN consisting of Cisco APs only, mobility tracking may thus be performed at very low expenses, because the Cisco APs can be configured to autonomously send association mutations to the Agent, whereas the Orinocos have to be polled individually. However, the drawback of the Cisco AP

- *State Check:* This value only applies if an AP is actually down, not reachable or its SNMP agent is malfunctioning. If the AP is working properly, the state check only takes about 5 ms, which is already included in the traffic information updating time t_t.

So the minimal update interval for a given number of each type of AP is determined as

$$t_{update}(n_c, n_o) = n_o \cdot \left(\rho t_{down} + (1-\rho) \cdot t_t^o + t_m^o\right) + n_c \cdot \left(\rho t_{down} + (1-\rho) \cdot t_t^o\right) + t_m^c,$$

where n_c denotes the number of Cisco APs, n_o the number of Orinoco APs and ρ the probability of an AP to be down.

The course of $t_{update}(n_c, n_o)$ is plotted in Figure 6.2. This Figure also shows the performance limits for WLAN that consist of only one type of AP. Thus, an Agent can either monitor up to approximately 100 Orinoco APs or up to 3900 Cisco APs in one-minute intervals.

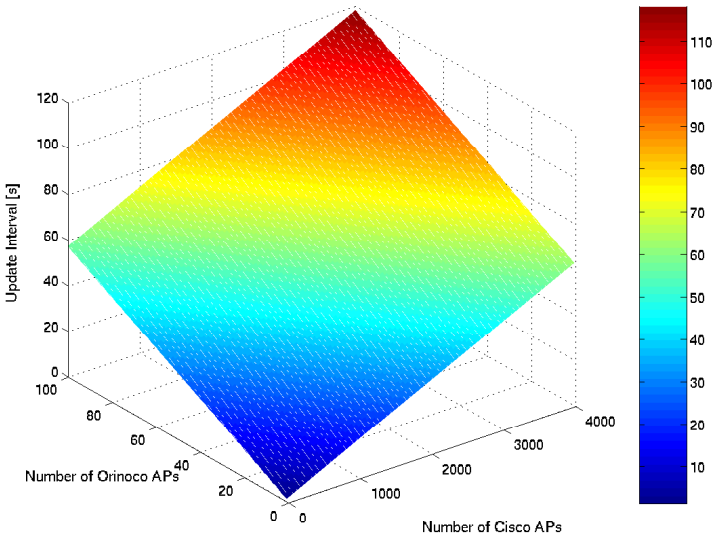


Figure 6.2: The minimal update interval for the Agents.

Not considered in these evaluations is the resources consumed to send an event message to the Manager. Tests showed that it takes the Agent about 200 μs to dispatch an event message, which is neglected for the present calculations.

6.2.2 The Manager

For the evaluations the Manager was run on a Fujitsu-Siemens Laptop with a 1.4 GHz Pentium processor and 256 MB RAM.

The performance of the Manager is determined by the amount of data (coming from the Agents) he is able to process during an update cycle before the next update cycle starts. As explained in chapter 5.3, the Manager polls all branches sequentially to get the current state of all APs.

After the polling of a branch, the Manager processes the received data and updates the databases. The processing time amounts to about a millisecond in a scenario with 118 APs. The polling time of a branch is depending on the total number of deployed Agents.

In the current implementation, the polling of all Agents is accomplished sequentially and not in parallel, i.e. the poll request walks the tree and thus the delays introduced by all Agents accumulate and the polling time of the entire system is proportional to the number of Agents deployed. The delay was found to amount to approximately $500 - 600ms$ per AP, depending on whether the Agent is just packing the data. The required time per branch is thus heavily dominated by the polling time. Figure 6.3 shows the minimal update interval for a given number of Agents. Thus, the current system should be able to process up to 100 Agents per minute.

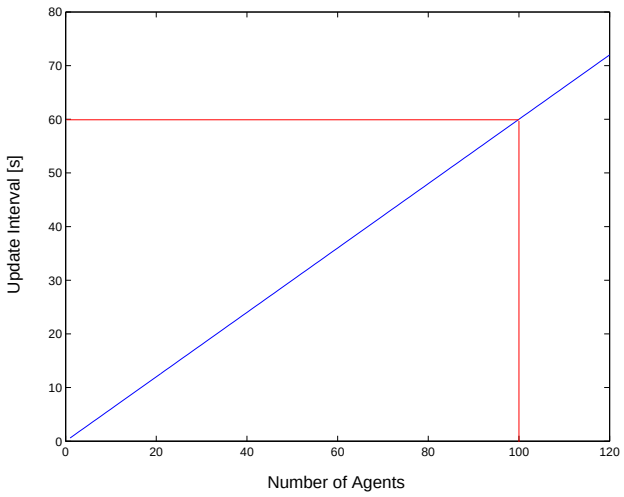


Figure 6.3: The minimal update interval for the Manager.

The receiving and processing of an event message takes about $50 - 80\ \mu s$ and is neglected in this evaluations.

6.2.3 Conclusions

The evaluations of the preceding sections allow to perform some back-of-the-envelope calculations to get a rough estimation of the capacity limits for the monitoring system in its current implementation.

An example: In a WLAN that deploys Orinoco APs only, an Agent can process about 100 of these per minute (see Figure 6.2), setting aside 3 seconds for circa 15′000 event messages . As found above, the Manager can process up to 100 Agents per minute. Thus, the system can notionally monitor up to 10000 Orinico APs in a one-minute intervals.

For a WLAN deploying Cisco APs only, we find that an Agent can handle up to approximately 3500 AP/min (setting aside 6 seconds for circa 30′000 event messages), and the whole system thus notionally up to 350′000 AP/min.

Thus, the scalability requirement for the implemented system is fulfilled.

Chapter 7

Conclusions

The efforts and investigations within the scope of this thesis, to design and implement a monitoring tool for large, heterogeneous WLANs, produced a working prototype, but also revealed difficulties and limitations, which are discussed in Section 6.1.

Comprehensive investigations of all types of AP that form StockholmOpen evinced that the available monitoring interfaces vary considerably among the different manufacturers (see Section 4.3). Especially since the aformentionned IEEE MIB for the 802.11b standard is only slowly gaining ground and is incorporated inconsistently among the manufacturers.

The set of common monitoring data of the investigated APs is meager (see Section 5.2). However, by abstracting from further data we gain information that is peculiar for wireless networks, e.g. the association mutations of stations.

To provide a high degree of scalability we adopted a distributed system architecture delegating the polling of the APs to Agents that are in turn polled by the Manager.

The performed evaluations of the implementation (See Section 6.2) show that the distribution of the workload boosts the performance and traffic scalability of the system substantially, allowing to monitor thousands of APs.

Moreover, the current implementation of the system is reconfigurable "on-the-fly", i.e. for the integration of a new AP into the monitoring system, only the corresponding Agent needs to be restarted with a new configuration. It will then automatically rejoin the system. New Agents can always be appended to the Agent tree during run-time.

However, the drawback are eventual higher costs, as several different platforms may be employed. This also leads to presumably higher maintenance expenses for the operator.

Chapter 8

Outlook

There are a number of tasks that could not be taken care of within the scope of this thesis due to lack of time.

- In parts of the code the shell programm *snmpwalk* is still deployed. This is one of the main reasons why the *mobility_update* routine performs badly. Replacing it with a function deploying the C API of net-snmp, should thus substantially increase the performance of Agents monitoring Orinco APs.
- The program currently just provides the data. To store it and to allow to collect statistics (e.g. about the performance of the WLAN or about the behavior patterns of WLAN users) all data should be written to database (according routines are defined) and by-and-by consolidated.
- To facilitate the access and interpretation of the collected data, it should be graphically edited and illustrated on a *Graphical User Interface (GUI)*. To make it available for a larger audience, the GUI may be presented on a web page.
- To extend the system for the integration of more AP types, the monitoring interfaces of these have to be studied and corresponding modules for the HAL have to be written.

Beyond these tasks there are some ideas for design amendments that could considerably improve performance and manageability of the system:

- *Parallel polling of APs:* In the current implementation, the information about APs are updated sequentially by requesting and processing items from them. These requests could be parallelized, which would boost performance of the Agents, as delays don't accumulate.

- *Parallel polling of Agents:* Instead of walking the tree of Agents (see Section 5.3.2), the Agents and the Manager could poll all their children in parallel. Thus, the response delays would not accumulate and the Manager could handle much more Agents per update interval.
- *Several Managers:* To boost the scalability of the system even more, several Managers could be employed all writing to a central database.
- *Monitoring the Agents:* To assure a proper operation of all Agents, the Manager should frequently check their state and report failures to the administrators.
- *FailSafe:* In case a Agents fails, its children should detect that and connect to a specified surrogate parent.
- *AP classes:* As not all AP types provide the same monitoring information, they should be classified into different classes, e.g. a class of AP, who support mobility monitoring and a second class of AP types, who don't.

Appendix A

The Intrinsyc Cerfcube

The Intrinsyc[1] Cerfcube is an inexpensive embedded hardware, ideal for server appliances, data collection devices or any other embedded application. Additionally, it is a multifunctional development platform for all kinds of applications.

[1] www.intrinsyc.com

A.1 Features

The CerfCube includes a high-performance, low power Intel StrongARM 1110 processor, lots of Intel StrataFlash^{TM} and fast SDRAM. The CerfCube's peripheral support includes Ethernet, one serial port, and flexible digital I/O. The CerfCube also includes a CompactFlash connector that can be used to add Bluetooth support, wireless WAN support with a digital phone card, or up to 6 GB of local storage with an IBM microdrive.

A.2 Software

CerfCube ships with Intrinsyc's I-Linux distribution including the Linux Kernel 2.4.9 and Familiar Distribution 0.5.1. Software includes both ipackage Tools for easy upgrades and the installation of thousands of modules found online, and Intrinsyc's bootloader software to manage flash memory and device configuration. The kit also includes a cross-compiler toolchain for doing x86 to ARM compiles.

A.3 Specifications

- *Processor:* Intel StrongARM 1110 microprocessor@206MHz
- *Memory:* 16 MB FLASH, 32 MB SDRAM (100 MHz)
- *CompactFlash:* Supports Type I and II, including IBM microdrive, CompactFlash memory cards, barcode reader and wireless modems.
- *Size:* 7.5cm x 7.5cm x 7.5cm.

List of Figures

Bibliography

[1] IEEE 802.11b Wireless LANs, Wireless Freedom at Ethernet Speeds. 3Com Corporation, 2000.

[2] Introduction to Wireless LANs, WLANA Inc, 1999.

[3] Analysis of a Campus-wide Wireless Network, David Kotz, Kobby Essien. In Proc of MOBICOM '02, September 2002, Atlanta USA.

[4] SNMP, SNMPv2, SNMPv3 and RMON1 and RMON2, William Stallings. Addison-Wesley, 1999 New York.

[5] Understanding SNMP MIBs, Davi Perkins, Evan McGinnis. Prentice Hall, Inc. 1997 New Jersey.

[6] Methodology for Passive Analysis of a Universitary Internet Link. Brownlee, kc. claffy, M. Murray and E. Nemeth, PAM2001.

[7] Ntop: beyond Ping and Traceroute, Luca Deri & Stefano Suin. Proceedings of DSOM '99, October 1999, Zurich, Switzerland.

[8] NeTraMet & NeMaC, Reference Manual, Nevil Brownlee.Information Technology Systems & Services, University of Auckland, June 1999, New Zeland.

[9] Using NeTraMet for Production Traffic Measurement, Nevil Brownlee. University of Auckland, June 2002, New Zealand.

[10] SRL Compiler and Language User's Guide, Nevil Brownlee.University of Auckland, August 1998, New Zealand.

[11] The design and implementation of an Operator Neutral Open Wireless Access Network at the Kista IT-University. E. Pelletta, F. Lilieblad, M. Hedenfalk, B. Pehrson, July 2002, Stockholm.