Merging IPA and NetWatchBGP

OSCAR KENÉ

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2007
Merging IPA and NetWatchBGP

OSCAR KENÉ

Master’s Thesis in Computer Science (30 ECTS credits)
at the School of Computer Science and Engineering
Royal Institute of Technology year 2007
Supervisor at CSC was Henrik Eriksson
Examiner was Stefan Arnborg

TRITA-CSC-E 2007:120
ISRN-KTH/CSC/E--07/120--SE
ISSN-1653-5715
Abstract

This master’s project is aimed to present the optimal solution for re-forming NetWatchBGP, a tool that is developed by and used by Tele2 employees for monitoring their IP network. The reformation includes performance optimisations and alterations in how the data within NetWatchBGP is managed. The changes are illustrated and performance tests are done to show what kind of improvements were done to the initial solution, which was fully functional but rather slow.
Referat

Integrering av NetWatchBGP och IPA

## Contents

1 Background ........................................... 1
  1.1 Purpose and Goal .................................. 1
  1.2 Abbreviations .................................. 2
  1.3 Disposition of the report ..................... 3

2 Problem definition ................................. 5
  2.1 Risks ........................................... 6
  2.2 Problem Objective .............................. 6

3 Existing Systems and Protocols ................. 7
  3.1 BGP ........................................... 7
  3.2 IPA ........................................... 7
  3.3 T2Server ....................................... 8
  3.4 NetWatchBGP ................................... 9

4 Development .......................................... 13
  4.1 Database Merge ................................ 13
  4.2 Middle Tier - T2Server or SOAP .......... 13
    4.2.1 SOAP/XML ................................ 13
    4.2.2 T2-resource .............................. 15
  4.3 Developing NetWatchBGP for IPA .......... 15
  4.4 Interface .................................... 16

5 Results ............................................. 19
  5.1 Interface .................................... 19
  5.2 Middle Tier .................................. 19
    5.2.1 New NetWatchBGP module - nwIPA .... 19
    5.2.2 Load Testing ............................ 20
  5.3 Database .................................... 21
    5.3.1 Load Testing ............................ 22

6 Conclusions ......................................... 23

Bibliography .......................................... 25
Chapter 1

Background

Tele2 has telephony and internet customers across all of Europe. This requires systems to monitor equipment and backup crucial configuration. Such systems are already in place and working as intended. However the tasks performed by two different systems may in some cases overlap. One system may contain information also stored partially in another. This is the case of IPA and NetWatchBGP. IPA is a system, not fully developed but functional, to maintain Tele2’s customer IP-ranges. NetWatchBGP on the other hand monitors these networks up/down status in Tele2’s network as well as the core networks.

Clearly these two systems should be connected or merged, preferably the latter. The information which resides in both systems is customer information. Into NetWatchBGP this is fetched manually from a third system called SPAN. The connection from SPAN to IPA is somewhat automated but not to the extent that you would want it to be.

The link between NetWatchBGP and IPA does not exist. Both systems partially contain the same information and the lack of linkage will cause inconsistencies sooner or later. What is worse, the people working with NetWatchBGP are not the same as the people working with IPA. If one could relocate the information from one of the systems so that it only resides in one benefits are obvious. There will be no chance of inconsistencies and the insertion of this information from SPAN could be automated fully. See Fig. 1.1 for an illustration.

1.1 Purpose and Goal

The purpose of my master’s project is to change the fashion in which NetWatchBGP handles information with less room for errors and greater usability. Only the tasks that require human interaction should do so, the rest should be done automatically. I also changed the way that people at Tele2 work with automated alarms for lost/found networks.
My goal with this master’s project is to research and develop a function to make the process of setting alarms for the NetWatchBGP system more automated. The risk of inconsistency between systems must be eliminated.

In reality I relocated some or all of the information from NetWatchBGP to IPA and have NetWatchBGP gather required information from several sources before presenting them. The NetWatchBGP system itself do not require any manual labour to function properly.

1.2 Abbreviations

- IPA/IMS - IP address management system for Tele2
- BGP - Border Gateway Protocol
- SPAN - Support for Planning And Network implementation
- SOAP/XML - Simple Object Access Protocol embedded in XML
- TCP - Transmission Control Protocol
- PL/SQL - Programming Language in SQL
1.3 Disposition of the report

The report describes the existing scenario, the assorted systems in use and their relations with each other. The problem definition is set from a finished product view, although all systems involved are constantly being developed and at a relatively high pace. It is important to set a specific point where your problem resides in the development process of the supporting systems.

Next I make room for viewing the supporting systems in functionality, dependencies and use. I see it as necessary to have some understanding about your surroundings before trying to alter them. My master’s project mainly revolves around NetWatchBGP and IPA. Currently IPA is interfaced/supported by the messaging software T2Server. It is therefore vital to also involve T2Server in the report as well as NetWatchBGP and IPA.

In the development part of the report, changes in database structure of NetWatchBGP are illustrated and due to the reallocation of information the interface currently in use in NetWatchBGP have been altered to some extent.

When all the previous steps have been taken, a finished result emerges and I will try to compare before and after snapshots of the system as a whole. Stress testing made to the system has also been presented within the normal usage limits.
Chapter 2

Problem definition

At the time of this master’s project, IPA is undergoing a radical change in development. The database will be restructured and interfaces towards other systems will be altered as well. This is not necessarily a bad thing from my point of view as my project will require IPA to be fundamentally developed. The challenging task, however, is to make my own desired changes be implemented in the process and at an early stage know exactly what changes are required.

Previously, NetWatchBGP contains some basic information about customer networks. This information was entered manually by the users. What is even worse is that the process of entering data to the system was made via a single user text file. Further explanation about the NetWatchBGP system is available in section 3.4.

NetWatchBGP is designed to handle events about lost/found networks received via the Border Gateway Protocol, BGP [1]. The information about customers assigned to these networks lost or found is also present at other locations apart from the NetWatchBGP system. So the task of maintaining this information in NetWatchBGP has to be done manually by multiple users. Things are bound to go wrong. I moved the information about customers from the NetWatchBGP system to a more suitable location, namely IPA. IPA contained some of the information contained in NetWatchBGP. NetWatchBGP contains information about customer and core networks whereas IPA only had information about customer networks. The framework for maintaining networks is already present in IPA so the information about core networks was easily inserted into IPA. There is however information associated with each network that only exists in the NetWatchBGP database and not in IPA.

My wish is that this responsibility of maintaining the information is to be added to the IPA system and removed from the NetWatchBGP system and the retrieval of this information into NetWatchBGP is done from IPA instead of from the NetWatchBGP database. The information is however originally fetched from the SPAN system, if it is a customer network, initially by the person entering it into
the NetWatchBGP system. What I gained from moving the information to IPA from NetWatchBGP is that SPAN and IPA are already connected. Also, SPAN only contains information about customer networks, not core networks. This will not be a problem as any non-customer networks in NetWatchBGP essentially have the same values apart from network address.

2.1 Risks

My work assumes that I will not be required to make the fundamental changes to the IPA system. I only plan to state my requirements and take part in the development process. The interfacing via T2Server however will be researched within this master’s project.

One of the most critical issues is that I am dependent on the near-future development of IPA. I can still develop a T2Server resource as long as I have the planned changes for IPA but testing will be difficult. This risk however has a low probability of occurring as the incentives are great and many in numbers and a group of people have finally been set to update IPA’s foundations.

2.2 Problem Objective

The issue at hand is not how to solve the problem or if there is a problem to be solved. Clearly information had to be relocated. My objective will be NetWatchBGP and IPA with stress on NetWatchBGP and information location. I did not work extensively with IPA but still I relied on IPA’s functionality.
Chapter 3

Existing Systems and Protocols

3.1 BGP

To allow for internet service providers to interconnect their networks a standard way of communicating is needed. This is provided by the Border Gateway Protocol, BGP. BGP peers are setup statically using TCP port 179, and from its peers a router periodically receives messages about other networks which these peers have knowledge of. Messages concern both found networks and lost networks.

In conjunction with receiving messages from other BGP enabled routers, the router itself calculates a routing table based on the information received and possibly information entered by the administrator of the router to determine the best path. As BGP operates between individual service providers there are options to allow or disallow certain routes and/or give certain routes a higher priority or to completely disable them. When setup correctly one has what we call the internet, multiple networks interconnected for full connectability to the rest of the world. For further information please examine RFC 1771 [1].

3.2 IPA

IPA is a system developed originally by Andreas Öhrvall in 2003 [4]. It has not been altered or upgraded much since then. By the time of writing this report (2006-2007) changes are done to the system and its database structure to reflect the requirements that has come up and the flaws that always have been there. IPA is a system to maintain and assign Tele2’s customer IP-ranges. It relies upon a T2Server resource and an Oracle database. IPA is mainly used by Tele2’s CNIS group, Core Network: Internet and Services, which in turn incorporates Tele2’s Registry, handling requests for ranges approved by other systems after a customer application.

The database is built around the IP ranges and not specific IP addresses, as is usual in IP routing. For example the IP addresses 130.244.10.0 .... 130.244.10.255
are grouped together with a network address and a network mask or prefix. So the previous example would look like 130.244.10.0/24 where /24 represents the number of significant bits in the bit representation of the address. The downside with this way of representing a block of addresses is the difficulty to ensure that one address is not present in two, or more, assigned blocks. In IPA the problem is solved through rather complicated database constraints. Here one can see the advantages of using an Oracle database as the scripting language PL/SQL becomes available. All but the simplest inserts towards the database is done through stored procedures.

3.3 T2Server

T2Server is an in-house developed middle-ware that is used in many different applications. The purpose of T2Server is to transport messages between applications. So an application which has support for T2Server automatically has the capability to communicate with any other application supported by T2Server.

T2Server is designed to be small and fast and is continuously being developed. It is currently used in RADIUS (Remote Authentication Dial In User Service) servers and various other content providers and their connection to the billing gateway.

T2Server is written in plain C but mimics some aspects of an object oriented language like structs declared in c-files instead of h-files. Also T2Server includes a quite extensive library of functions called libsw. It contains functions to handle memory management and database calls.

Central to T2Server calls are the key-value pairs along with their get and set methods. There is also the possibility to diversify your calls even more through namespaces and sub namespaces. There are two namespaces predefined, ID and Session. The ID namespace is a read-only namespace where for example identities are stored. The Session namespace contains information specific for the current T2Server session set up by the user such as expiration and client specifics. In defining your own namespace, simply design your own c-file according to the namespace definition. After compilation of your new namespace into T2Server, T2Server is ready to be started.

You can also define a different kind of namespace called authentication modules. They are similar to namespaces but specify other methods. The authentication modules are used to administrate the accesses of the namespaces.

All these namespaces are loaded into the T2Server nameserver process which upon execution loads configuration files listing where (IP and TCP port) each T2-resource is located.
3.4 NetWatchBGP

NetWatchBGP is a system to monitor the core and customer networks within Tele2’s net. If a network is lost or found this event is recorded by NetWatchBGP.

As one can see in Fig. 3.1, NetWatchBGP consists of many separate parts. The first in line is `nwbgp` which establishes connections with BGP routers, listed in its configuration file, receiving BGP messages as described in [1]. It also waits for connections from other client applications such as `nwbgpclient`. The connection between `nwbgp` and `nwbgpclient` is made via TCP and only BGP messages such as UPDATE or WITHDRAWN is forwarded in the following format:

Where NETF (net found) corresponds to a BGP UPDATE message and NETL (net lost) to a BGP WITHDRAWN message. The client then inserts this information into the NetWatchBGP database.

To display the gathered information, there are two modules used, `nwperl` and `nwbgp-www`. The module `nwperl` generates HTML pages from the NetWatchBGP database. The module `nwbgp-www` is a web front end that generates status reports and alarm information via `nwperl`.

In the old days before NetWatchBGP, the customer information was kept in a simple text file which had strict formatting conventions but no error control. With the introduction of NetWatchBGP, the text file was injected into the NetWatchBGP database, but the text file still was the master copy and changes were done to this file and periodically injected into the database. This highly inefficient way of maintaining information will disappear with my development of NetWatchBGP as described in the next section.
Figure 3.1. **NetWatchBGP - Information flows in the previous system.**
3.4. NETWATCHBGP

```
 noc:NetWatchBGP-2.0/nwb\>telnet localhost 5052
 Encryption is verbose
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is ‘\’’.
 200 EANA NetWatchBGP Version 2.0. Your command?
 NETF193.15.224.16/28
 NETL130.244.127.104/29
 NETL193.13.23.0/24
 NETL193.15.224.16/28
 NETL130.244.124.164/30
 NETF192.68.221.0/24
 NETF130.244.124.164/30
 QUIT
 221 NetWatchBGP closing connection
```

**Figure 3.2.** Example session with nwbclient.
Chapter 4

Development

The development of a new version of NetWatchBGP has many aspects. The information has been moved, the connection towards this information has as a consequence been developed too. With these changes a new way of interacting with NetWatchBGP has been supplied.

4.1 Database Merge

The database merge, or more correctly moving the customer information from NetWatchBGP to IPA, has been done but not by me. The required changes have been accomplished by others who continuously develop IPA.

4.2 Middle Tier - T2Server or SOAP

The information gathering to be displayed to the user have to changed as the customer information is relocated. As the information will be available only through a T2Server resource, I have two options of accessing it. Either I can develop a T2Server resource wrapping around NetWatchBGP database or have NetWatchBGP formulate a SOAP/XML request and be able to parse the SOAP/XML answers. If a T2Server resource application is chosen, I open up for other T2Server resources to interact with NetWatchBGP in future aspects I have not considered, but may be advantageous for solving other issues. The SOAP/XML alternative is a simpler one and maintains NetWatchBGP’s stand alone nature.

4.2.1 SOAP/XML

SOAP is a protocol based on XML and consists of three parts. First there is an envelope that defines a framework for describing what is in a message and how to process it. Second there are encoding rules to express application-defined data types. Last in SOAP there are the actual conventions of representing remote procedure calls (RPC) and their return values.
Your application simply formulates an XML message based on the SOAP protocol and forwards it via HTTP to the nameserver described in 3.3. The sending application then expects a response in the same format and in the encoding of the sent message.

The advantages of using SOAP/XML in my case is that it is easy to modify the existing implementation of NetWatchBGP and this option maintains NetWatchBGP’s independence of other applications. The customer information that will be retrieved from IPA is not of vital nature for NetWatchBGP to function, the customer information is only a nice add on to the NetWatchBGP interface. The disadvantage however is that future implementation of new features into NetWatchBGP that requires interaction with other systems supported by T2Server will be tedious and require more programming done to NetWatchBGP.

![Diagram](image)

**Figure 4.1.** NetWatchBGP after SOAP implementation towards IPA.
4.3. DEVELOPING NETWATCHBGP FOR IPA

4.2.2 T2-resource

Developing a resource to accommodate NetWatchBGP into the T2Server environment may seem as a better alternative than SOAP/XML. By this, future development and/or integration with the T2Server environment will be easier and it is also a step towards unifying the assorted systems in place today.

In this scenario, the database tier of NetWatchBGP will be included into a T2Server resource and a totally separate application (NetWatchBGP interface) gathers information from several resources to assemble the information for display (Fig. 4.2).

This option though tempting cannot be realized due to the high dependency on T2Server. NetWatchBGP must function 100% of the time so keeping dependencies to a minimum is of highest importance. I have therefore chosen to discard this option of development and will focus on SOAP/XML described in section 4.2.1.

![Figure 4.2. NetWatchBGP put into a T2Server resource.](image)

4.3 Developing NetWatchBGP for IPA

NetWatchBGP has a predefined structure as seen in Fig. 3.1. To allow IPA data to enter the system via SOAP/XML, there are two options as I see it. Either I can rewrite `nwperl` to fetch the required information from IPA (Fig. 4.3) or, to maintain the structure of NetWatchBGP, I can create a new part to fetch the information from IPA and insert it into the NetWatchBGP database (Fig.4.4). The second option duplicates information but there will not be any chance of inconsistencies as
this is done automatically through `crontab` or any similar process. This option is simpler as the existing code will require few if any alterations. Also the issue of load on the system can be controlled easily as the information gathering done from IPA can be done periodically and the actual presentation can be done on-demand i.e. in real-time.

Figure 4.3. `nwperl` does the job.

### 4.4 Interface

Altering information in IPA through SOAP/XML can be done just as easily as fetching information from the same source. Fetching information into NetWatch-BGP from IPA will be done through SOAP/XML. Regarding insertion/altering of information in IPA there are two options. The first is to build some interface in NetWatchBGP towards IPA that can insert or alter information concerning NetWatchBGP. The other option is to allow NetWatchBGP users to use the existing IPA interface. My final goal is to give the users of NetWatchBGP the possibility to add new customer information to the system by issuing a request for a new block via the IPA interface.
4.4. INTERFACE

Figure 4.4. A new part of NetWatchBGP inserts the information into the NetWatchBGP database.
Chapter 5

Results

The original NetWatchBGP code, both C code and Perl scripts, have received some attention from me performance-wise. My wishes were that updates requested via the web interface, showing current status of networks, could be done on demand. Before my alterations, another Perl script from the nwperl module, created a static HTML page once every minute initialised by crontab.

In order to allow a greater load on the system when users repeatedly request an updated view, load testing has been performed. This will be further detailed in sections 5.2.2 and 5.3.1. There I will outline the major bottle necks in the system and show what I have done to eliminate them or at least lessen their impact towards performance.

5.1 Interface

The NetWatchBGP interface was left virtually unchanged apart from a few minor tweaks. The single functional change that was made is the removal of a deletion link that if clicked would remove the monitoring of the affected network. This will in the new version be done via IPA setting a flag on the affected network. Also modperl was introduced in the web server Apache2 instance which improved performance somewhat.

5.2 Middle Tier

5.2.1 New NetWatchBGP module - nwIPA

After the decision to create a new module adding new functionality to NetWatch-BGP the question is how to accomplish this. I have two basic directions to go, either to spawn a new process when needed, which itself creates load, or to create a process that runs in the background periodically executing the task of fetching data from IPA.
I also have to consider to take advantage of the already existing processes present on the host. What comes to my mind is crontab which itself is a process executing other executables according to a schedule specified therein. As the spawning creates little load relative to the actual processing made when fetching and parsing the data, crontab seemed to be a very viable choice.

The choice of how to create the actual executable proved to be a more difficult choice. The C programming language is known for its widespread support and excellent speed. If the development of the new module would be done in a programming language, C would be the unchallenged choice. Another approach is to proceed with a scripting type language such as Python, Perl or even shell scripting. As the host where this will run is either a Sun Solaris variant or a Linux distribution, all the listed scripting options are equally supported. What it essentially comes down to is personal preference of the developer and the maintainer of the code.

With this in mind NetWatchBGP’s new module nwIPA has been written in the Perl scripting language. Perl was chosen for its simplicity and ease of small fast modification, as opposed to C which requires compiling after each change, no matter how small. As the SOAP/XML calls done are fairly static these can be kept in separate files shortening the actual script further. All that is really done is uploading a SOAP/XML question via HTTP and parse the answer for the relevant data where after the relevant entries in the NetWatchBGP database is updated or created. The script itself will be run via crontab once or a few times every day to reflect changes done to IPA’s customer database without stressing either IPA or NetWatchBGP too much.

5.2.2 Load Testing

The module itself is fairly simple and straight forward. First and foremost the data retrieved from IPA is of considerable size, several megabytes of XML in a single file. This does not put any noticeable load on the system, the bottleneck is on the opposite side when fetching the data from its Oracle database and reformatting it onto SOAP/XML. After fetching the data all of it is contained within an XML formatted text file and is in need for parsing and insertion to the NetWatchBGP’s database.

Following is a breakdown of the total time consumed when fetching all nets from IPA, parsing and finally inserting them into the database.

What this illustrates is that improvements in the XML parsing is desirable. The other steps of operation is dependant on factors I have no real control over. Luckily these steps do not require as much computing power as the XML parsing does so
5.3. DATABASE

<table>
<thead>
<tr>
<th>Action</th>
<th>Time (s)</th>
<th>Time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetching data</td>
<td>31</td>
<td>1.3</td>
</tr>
<tr>
<td>Parsing data</td>
<td>2411</td>
<td>98.7</td>
</tr>
</tbody>
</table>

Figure 5.1. *nwIPA before improvements.*

there is room for improvement on my part. In the myriad of tools available to parse XML, Perls XML::Simple [8] module was used. This speeds up the processing of the retrieved XML file considerably instead of using a home made parser which essentially involves too much ineffective string handling.

<table>
<thead>
<tr>
<th>Action</th>
<th>Time (s)</th>
<th>Time (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetching data</td>
<td>31</td>
<td>60.8</td>
</tr>
<tr>
<td>Parsing data</td>
<td>20</td>
<td>39.2</td>
</tr>
</tbody>
</table>

Figure 5.2. *nwIPA after improvements.*

As shown, already existing tools for parsing XML are far superior to crude string parsing mechanisms. This is in my opinion good enough for the environment where it will operate so no further development will be done to nwIPA to improve it performance wise.

5.3 Database

The database remains more or less unchanged. It is a standard mySQL database consisting of a few tables. The structure of the customer table, the only part of the database interesting for us, can be seen in Fig. 5.3 . The database has information about customers, events and routes. There used to be a table called alert which was not used and therefore has been removed.

The customers table contains information about what network is concerned, customer name, last update, type of link and a few more fields seen in Fig. 5.3 . It is a what you get from a simple "describe table:" command in mySQL. This table is populated via the new nwIPA module gathering information via SOAP/XML from IPA.
5.3.1 Load Testing

Previously NetWatchBGP opened a new connection towards its database for each operation. This seemed very unnecessary and can be compared to opening a new TCP connection each time you would like to send a new packet. After altering the original code I could perform 8 times more database operations per second than before. As expected, the connection setup is where performance can be gained database-wise. With the small improvements made, I have gained the performance desired and will leave it as it is for now.
Chapter 6

Conclusions

The reworking of NetWatchBGP has been successful in its goal to improve performance and to eliminate the rather crude way of maintaining important information. My initial assumptions, that I had to incorporate NetWatchBGP or at least parts of it into a T2Server proved to be false, which is good, for in retrospect that would prove to be a very tedious task and would also take away NetWatchBGP’s stand-alone nature. After all, a surveillance system should be very robust and therefore not dependent on other resources to function. This has stayed as one of the benefits of NetWatchBGP. It has a fairly simple structure, separated into small modules, which in most cases do not explicitly rely on each other to function. This is also the case of nwIPA, which only gathers customer information which in turn is only cosmetic. NetWatchBGP will continue to function without the assistance of IPA. The only new restriction introduced is that one cannot add new nets to NetWatchBGP without IPA being present and able to interface with NetWatchBGP.
Bibliography

www.faqs.org/rfcs/rfc1771.html

www.faqs.org/rfcs/rfc793.html

[3] RFC 2616 (rfc2616) - Hypertext Transfer Protocol – HTTP/1.1  
www.faqs.org/rfcs/rfc2616.html


[8] XML::Simple -  