



UNIVERSITEIT VAN AMSTERDAM

# SNE

System and Network Engineering

## Automatic end-host configuration

*Sebastian Dabkiewicz*

*February 12, 2012*

### **Abstract**

*In scientific environments inter-domain networks are spanned around the globe. These networks are usually built for a certain time-frame. Setting up of the network takes a huge amount of time. In the future it is desirable to build these networks more dynamically. An important issue is the configuration of the end-hosts attached to the network. They need a IP-address and the ability to find the other servers. Because no DHCP-server or DNS-server is present this has to be done manually. In this research project, research is done on automatic end-host configuration which is also known as Zeroconf. Zeroconf is a subset of link-local addressing, multicast DNS and DNS service discovery. Some operating systems and applications (like iTunes) have this functionality build in. Servers usually don't, but applications like Avahi adds this functionally also to servers. This research shows that automatic configuration of servers is feasible, and could be implemented in networks which are used in scientific environments.*



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Environment</b>	<b>6</b>
2.1	Automatic configuration . . . . .	6
2.2	Zeroconf . . . . .	6
2.2.1	IP-addressing . . . . .	7
2.2.1.1	IPv4 Link-Local addresses . . . . .	7
2.2.1.2	IPv6 Link-Local addresses . . . . .	7
2.2.2	Multicast DNS . . . . .	8
2.2.3	DNS service discovery . . . . .	8
<b>3</b>	<b>Experimental methods</b>	<b>10</b>
3.1	Test environment . . . . .	10
3.2	Initial installation . . . . .	11
3.2.1	avahi-daemon . . . . .	11
3.2.2	avahi-discover . . . . .	12
3.2.3	avahi-autoipd . . . . .	12
3.2.4	avahi-utils . . . . .	12
3.2.5	libnss-mdns . . . . .	14
<b>4</b>	<b>Results</b>	<b>15</b>
4.1	Interface configuration . . . . .	15
4.1.1	Address conflict . . . . .	15
4.2	Service Discovery . . . . .	16
4.3	IP lookup . . . . .	16
4.4	Cross platform . . . . .	17
4.4.1	Microsoft Windows 7 . . . . .	17
4.4.2	Apple Mac OS X . . . . .	18
4.5	Security . . . . .	19
4.6	Timing . . . . .	19
<b>5</b>	<b>Conclusion</b>	<b>21</b>
<b>6</b>	<b>Further research</b>	<b>22</b>
<b>A</b>	<b>Avahi-daemon configuration file</b>	<b>23</b>



# 1 Introduction

In scientific environments, there are circuit-based networks, which can span over the world. At the moment these networks are built for a longer time-frame (weeks or months).

In the future it is desirable that these networks become more dynamic so that they can be set up for a short time-frame (days or even minutes). Because there is no DHCP-server in those networks the configuration has to be done manually. This is a time-consuming task, which should be done in an automatic way.

Since the networks are on the same logical network segment there is no need to do routing. So link-local addressing is an option.

During the iGrid 2005 event in San Diego, USA the System and Network Engineering group from the University of Amsterdam has set up a network connection between San Diego and Amsterdam. The Goal was to do Zero configuration networking, and let people connect with their own equipment. Back then the software which was available had to be modified, and configuring it took more time than manually configuration [16].

Since then much has changed, Avahi an open source Zeroconf implementation has been developed. In this report I will have a look at this technology.

This situation gives the following research questions:

How can one create an automatic end-host configuration?

- What are the requirements for a fast establishment of the connection?
- What is the current situation?
- What kind of implementations are available?
- What kind of configuration is needed?
- Is there support for a cross platform solution?

These questions will be answered in this research report. We start with a global introduction of the environment and Zeroconf in chapter 2.

Chapter 3 describes the equipment and tools that are used for the experiment.

The results of the experiment will be discussed in chapter 4.

In chapter 5, a conclusion will be drawn based on the results from chapter 4.

Finally in chapter 6 ideas are given for additional research in this topic.

## 2 Environment

At this moment setting up a connection across the globe is a time intensive task. Because of the great amount of providers which are involved. The whole process can take months and a huge amount of email messages.

To minimise the time and emails that are needed to send, a project is started named Automated GOLE.

Automated GOLE is a project to let researchers create a connection using a Network Service Interface (NSI) to acquire unused network bandwidth which is made available by members of the Global Lambda Integrated Facility (GLIF), so there is no need to ask several organisations individually.

These connections can be used to transfer a huge amount of data or do calculations together and then the connection could be terminated.

### 2.1 Automatic configuration

Some time ago the configuration of IP-networks was a difficult task. Administrators must take care of IP-addresses, subnet-masks, gateways and so on.

With the introduction of the Dynamic Host Configuration Protocol (DHCP) [17] this became easier. The administrator can define a scope of IP-addresses which are leased to the clients automatically. Together with the IP-lease the client gets additional information like the default-gateway and DNS-server. But there is still need for an administrator to maintain the situation.

About 30 years ago Apple developed a protocol-suite named AppleTalk. Included in this suite was a protocol named Name Binding Protocol (NBP). NBP provided naming and service discovery functionalities. In the mid '90s there was a discussion on the Net-Thinkers mailing list [11] about the need for a simple automatic configuration method for small local area networks (LAN).

In 1999 the IETF Zeroconf Working Group was started, which developed the necessary techniques to archive zero configuration networking (Zeroconf).

The group worked on the Dynamic Configuration of IPv4 Link-Local Addresses. Other techniques which are still in development are Multicast DNS to resolve host names and DNS Service-Discovery to find services within the network.

### 2.2 Zeroconf

Zero configuration networking (Zeroconf) [12] is a collection of techniques which provide automatic network configuration. It is a subset of IP-Link-Local-addressing, multicast DNS (mDNS) and DNS service discovery (DNS-SD).

Developed by Apple as Bonjour (former known as Rendezvous) which is available for Apple and Windows systems, also an open source version has been developed, which named Avahi and is available for Linux.

## 2.2.1 IP-addressing

### 2.2.1.1 IPv4 Link-Local addresses

The working of IPv4 Link-Local addresses (IPv4ll) is described in RFC3927 [13].

The client allocates an IP-address out of the 169.254/16 scope. The first 256 and last 256 addresses are reserved for future use and must not be selected. The selection of the IP-address is based on a pseudo-random number generation algorithm, which mainly uses the MAC-address of the network card to generate the IP-address. Since the Media Access Control (MAC)-address doesn't change the the host will usually choose the same IP-address.

After allocating the host will send an ARP (Address Resolution Protocol)-request to find out if the IP-address is already in use. If no answer is received the address is available and can be used. If the client gets a response back from another client it knows that the address is in use, it will select an other address and start over again.

Because of the big IP-range which is available in the IPv4ll address space the chance that a client pick up an unused IP-address on a link with 1300 other hosts is 98% on the first try. Within two tries the chance of getting a free IP-address is 99,96%. The chance that the client needs more than 10 tries is about 1 in  $10^{17}$ .

In our scientific-environment there will only be a few end-hosts in use, so the actual chance of an address conflict should be very low when using IPv4ll addresses.

### 2.2.1.2 IPv6 Link-Local addresses

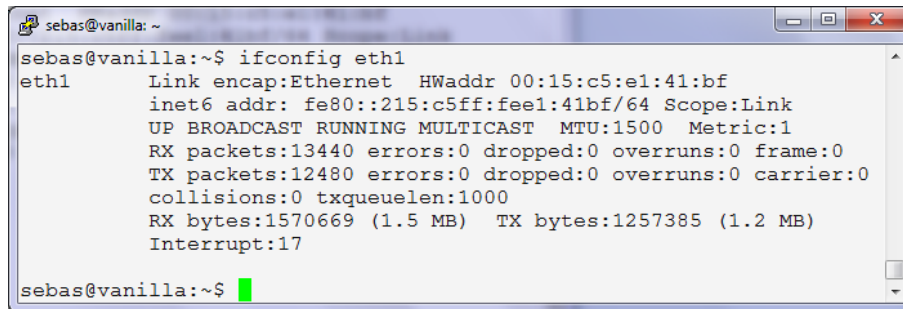
The representation and auto-configuration of IPv6 link-local (IPv6ll) addresses is described in RFC4291 [18] and RFC4862 [19].

At start-up each interface allocates an IPv6ll address by default. Even if there is a IPv6 address allocated by manual configuration or via a DHCP-server. The scope for the IPv6ll addresses is the `fe80::/64` scope.

The IPv6ll address is mostly derived from the MAC-address of the interface. In the middle of the 48 bits MAC-address `ff:fe` is inserted and the 7th significant bit is inverted to make it universal. So MAC-address `00:15:c5:e1:41:bf` becomes: `fe80::215:c5ff:fee1:41bf/64`.

IPv6ll addresses are not routable and can even as the IPv4 equivalent only used on the local segment.

After allocating an IPv6ll address the host has to check if the IP-address is available. This will be done using Neighbor Solicitation and Advertisement messages.



```
sebas@vanilla:~$ ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:15:c5:e1:41:bf
          inet6 addr: fe80::215:c5ff:fee1:41bf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13440 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1570669 (1.5 MB)  TX bytes:1257385 (1.2 MB)
          Interrupt:17
sebas@vanilla:~$
```

Figure 1: Interface with an IPv6 link-local address

### 2.2.2 Multicast DNS

Multicast DNS[15] (mDNS) is used to resolve host names into IP-addresses using multicast. mDNS uses 224.0.0.251[9] as IPv4 multicast address and ff02::0:0:0:0:0:0:0:fb[10] as IPv6 multicast address.

As destination MAC-address for the mDNS packets that are sent 01:00:5e:00:00:fb is used.

The UDP Port 5353 is reserved for the use of mDNS

The domain name which registers for the use of mDNS is `.local`. But if necessary this can be changed to an other domain.

The mDNS packet header contains a 2 byte field with flags. The first and sixth bit are used to identify if an packet is a request or a reply. If the packet is a request the first and de sixth bit are set to zero. If the packet is a reply these two bits are set to one. The other bits are always zero.

The data-field of the mDNS packet is UTF-8 encoded and contains DNS data, like A-records and PTR-records.

### 2.2.3 DNS service discovery

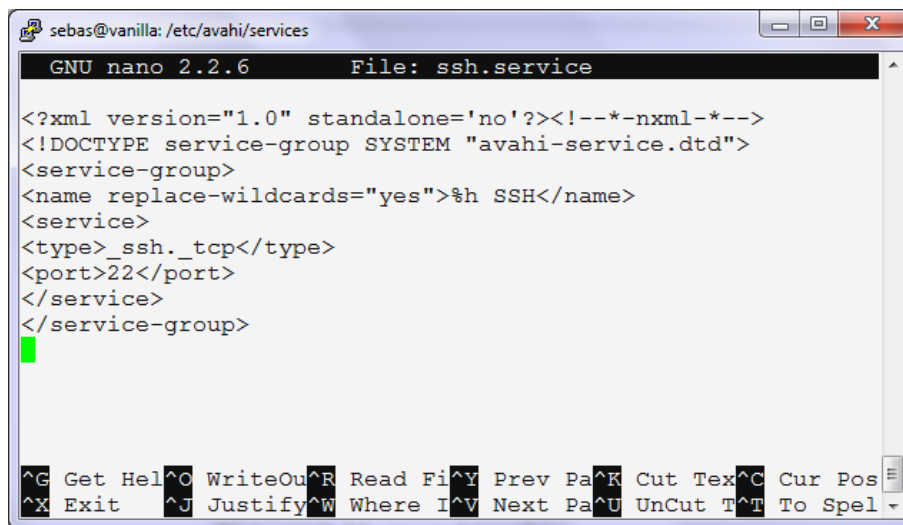
DNS service discovery[14] (DNS-SD) is a technique to discover services on other end-hosts.

DNS-SD uses DNS SRV, TXT and PTR records to advertise the services which are running on the host. A list of DNS SRV Service Types can be found on the dns-sd.org website[7]

For services who don't announce his existence by itself or via a plug-in, it is possible to create custom service announcement files. The XML-file format [5] is used to archive this.

Figure 2 shows an example-file for announcing an SSH-server.



A screenshot of a terminal window titled 'sebas@vanilla: /etc/avahi/services'. The window shows the GNU nano 2.2.6 editor editing the file 'ssh.service'. The content of the file is XML-based, defining a service group for SSH. The XML includes a service group named 'SSH' with a service of type '\_ssh.\_tcp' on port 22. The terminal window also shows a status bar at the bottom with various keyboard shortcuts for nano editor operations like 'Get Help', 'Write Out', 'Read File', etc.

```
sebas@vanilla: /etc/avahi/services
GNU nano 2.2.6 File: ssh.service
<?xml version="1.0" standalone='no'?><!--*-nxml-*-->
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
<name replace-wildcards="yes">%h SSH</name>
<service>
<type>_ssh._tcp</type>
<port>22</port>
</service>
</service-group>
^G Get Hel^O WriteOu^R Read Fi^Y Prev Pa^K Cut Tex^C Cur Pos
^X Exit ^J Justify^W Where I^V Next Pa^U UnCut T^T To Spel
```

Figure 2: ssh.service file to announce a SSH-server

Important in the \*.service-file are the labels <type> and <port> which respectively describes the protocol used and port number of the service. The <name> describes the name of the service.

### 3 Experimental methods

In consultation with my supervisors, I decided to make use of two servers in the lab. This is because creating an inter-domain connection takes a long time and there are different other organisations involved then only the University of Amsterdam. Practically there is no difference between two servers connected by a simple network cable and a switch, or a light connection to a remote site. Because the work will be done within a broadcast domain.

Timing is important feature when connecting the end-hosts together. They should be fast online. Defining an entry in the interface configuration file as described in section 4.1 should make the host faster come online.

#### 3.1 Test environment

For the research I had two servers in use, installed with Ubuntu 11.10 server edition. These servers were attached to our public OS3 lab-network on eth0 and directly connected to a simple switch on eth1.

The eth0 was used to connect remotely to the server via SSH, while the eth1 connection was used for the Zeroconf.

Furthermore there where 2 clients, a laptop with Windows 7 Professional and a Mac Mini with Mac OS X. These two machines are used to test the cross-platform functionality.

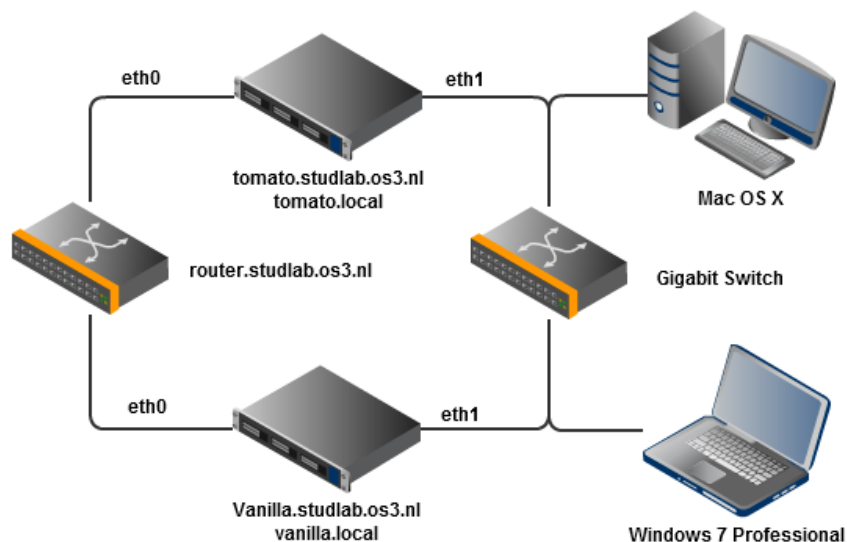


Figure 3: Network diagram

## 3.2 Initial installation

The initial installation of the server can be accomplished by installing the following standard Avahi-packages from the repository:

- avahi-daemon
- avahi-discover
- avahi-autoipd
- avahi-utils
- libnss-mdns

### 3.2.1 avahi-daemon

The avahi-daemon is the main part of Avahi. In the configuration file one can specify parameters to make sure the proper working of avahi. By default most settings are commend out and so they are set do the default value. The default settings works well in the most cases, since it was designed for home use.

In our environment we have advanced servers with multiple interfaces, native support for IPv4 and IPv6. So the we have to change some of the default values.

To restrict the use of IP-addresses one can specify to run Avahi only on IPv4 or IPv6, but also both at the same time is possible. For better readability I disabled IPv6 to create the screenshots for the document.

Our servers have two network interfaces, one connected to the internet and the other connected to the second server by a switch. Since we did not want to publish the services which our server offers on the internet, it is possible to restrict the use of avahi only to eth1, which is in our case the local interface.

Settings which I changed are:

```
use-ipv4=yes
use-ipv6=no
allow-interfaces=eth1
deny-interfaces=eth0
```

Other settings which may be important to use are:

```
[server]
host-name=foo
domain-name=local
browse-domains=example.org

[publish]
disable-publishing=no
disable-user-service-publishing=no
publish-workstation=yes
```

The Avahi configuration file of the Ubuntu server can be found in appendix A and additional information for the parameters here [3].

### 3.2.2 avahi-discover

Avahi-discover is a GUI tool to find services within the network. Due the fact that it is a GUI tool, it can not be used if only a command line connection to the server is available and avahi-discover is not installed on the client system.

### 3.2.3 avahi-autoipd

Avahi-autoipd provides an extra interface with a configured IPv4 link-local address. The name of the interface is as follows: ethX:avahi. In our environment it was eth1:avahi. So can, if the DHCP-server in the network fails or is not available, the host use the network connection with a link-local address.

Because every interface gets an IPv6ll-address automatically even if there is a DHCP-server who gives out IP-leases there is no need for an extra interface with an IPv6ll address.

Although Avahi works well with just installing avahi-autoipd, it is possible to run some commands directly [2].

To assign an IPv4ll address to interface eth1 one can run the following command: `sudo avahi-autoipd eth1 -D`

### 3.2.4 avahi-utils

Avahi-utils is a package with useful tools for use with Avahi. It is possible to browse the domain for available services or resolve an hostname to an IP-address.

**avahi-browse** Avahi-browse can be used to discover services on the network. Because of DNS-SD announcements the existence of a service shows up. It is also possible to resolve the services found. See figure

5

```

sebas@vanilla:~$ ifconfig eth1 && ifconfig eth1:avahi
eth1      Link encap:Ethernet  HWaddr 00:15:c5:e1:41:bf
          inet6 addr: fe80::215:c5ff:fee1:41bf/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:776 errors:0 dropped:0 overruns:0 frame:0
          TX packets:339 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57327 (57.3 KB)  TX bytes:58128 (58.1 KB)
          Interrupt:17

eth1:avahi Link encap:Ethernet  HWaddr 00:15:c5:e1:41:bf
          inet addr:169.254.100.115  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:17

sebas@vanilla:~$ █

```

Figure 4: interface eth1 and eth1:avahi

```

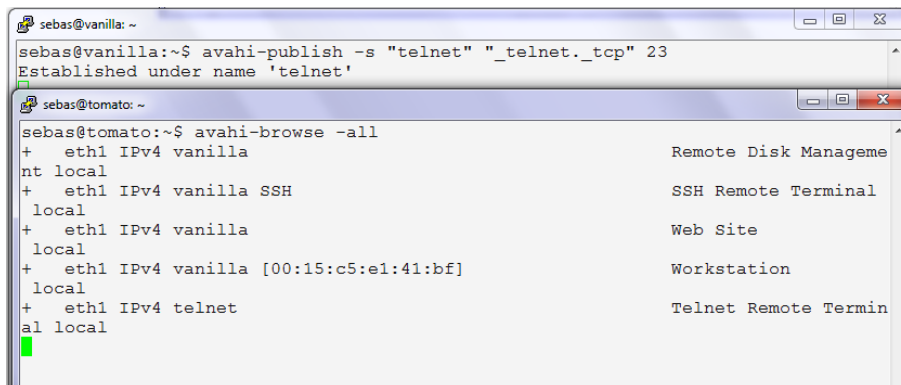
sebas@tomato:~$ avahi-browse -all -r -t
+ eth1 IPv4 vanilla                               Remote Disk Management local
+ eth1 IPv4 vanilla SSH                           SSH Remote Terminal local
+ eth1 IPv4 vanilla                               Web Site local
+ eth1 IPv4 vanilla [00:15:c5:e1:41:bf]           Workstation local
= eth1 IPv4 vanilla                               Remote Disk Management local
hostname = [vanilla.local]
address = [169.254.18.24]
port = [22]
txt = []
= eth1 IPv4 vanilla SSH                           SSH Remote Terminal local
hostname = [vanilla.local]
address = [169.254.18.24]
port = [22]
txt = []
= eth1 IPv4 vanilla                               Web Site local
hostname = [vanilla.local]
address = [169.254.18.24]
port = [80]
txt = []
= eth1 IPv4 vanilla [00:15:c5:e1:41:bf]           Workstation local
hostname = [vanilla.local]
address = [169.254.18.24]
port = [9]
txt = []
sebas@tomato:~$ █

```

Figure 5: The use of the avahi-browse tool

**avahi-publish** With `avahi-publish`<sup>[4]</sup> it is possible to announce the existence of a service within the network. For example an telnet server as seen in figure 6. After closing the `avahi-publish` command the service disappears.

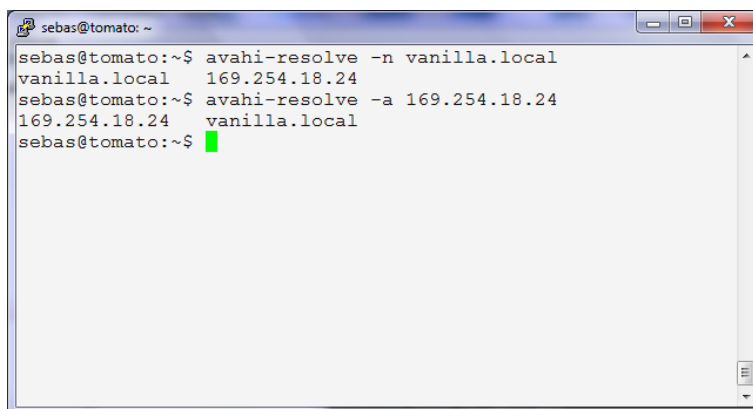
**avahi-resolve** Avahi-resolve can be used to resolve host names to IP-address and vice versa. `avahi-resolve -n HOSTNAME` gives out the IP-address and `avahi-resolve -a IP-address` gives the hostname of the client. See figure 7



The image shows two terminal windows. The top window is on a host named 'vanilla' and shows the command `avahi-publish -s "telnet" "_telnet._tcp" 23` being executed, with the output `Established under name 'telnet'`. The bottom window is on a host named 'tomato' and shows the command `avahi-browse -all` being executed, which lists several services including 'Remote Disk Manageme', 'SSH Remote Terminal', 'Web Site', 'Workstation', and 'Telnet Remote Termin'.

```
sebas@vanilla: ~  
sebas@vanilla:~$ avahi-publish -s "telnet" "_telnet._tcp" 23  
Established under name 'telnet'  
sebas@tomato: ~  
sebas@tomato:~$ avahi-browse -all  
+ eth1 IPv4 vanilla Remote Disk Manageme  
nt local  
+ eth1 IPv4 vanilla SSH SSH Remote Terminal  
local  
+ eth1 IPv4 vanilla Web Site  
local  
+ eth1 IPv4 vanilla [00:15:c5:e1:41:bf] Workstation  
local  
+ eth1 IPv4 telnet Telnet Remote Termin  
al local
```

Figure 6: Publish a service with avahi-publish



The image shows a terminal window on a host named 'tomato' demonstrating the `avahi-resolve` tool. The first command `avahi-resolve -n vanilla.local` returns the IP address `169.254.18.24`. The second command `avahi-resolve -a 169.254.18.24` returns the host name `vanilla.local`.

```
sebas@tomato: ~  
sebas@tomato:~$ avahi-resolve -n vanilla.local  
vanilla.local 169.254.18.24  
sebas@tomato:~$ avahi-resolve -a 169.254.18.24  
169.254.18.24 vanilla.local  
sebas@tomato:~$
```

Figure 7: The use of the avahi-resolve tool

### 3.2.5 libnss-mdns

Libnss.mdns is a Name Service Switch (NSS) module for Multicast DNS name resolution. It allows name resolution by programs in the `.local.` domain. [8]

## 4 Results

After installing the Avahi-packages described in section 3.2 and editing the configuration files the discovery of hosts and services worked well.

### 4.1 Interface configuration

Since Zeroconf should provide automatic configuration, it should not be necessary to configure the interface where one want to use Zeroconf. But sometimes it is good to have the ability to shut down an interface for some time, without disabling the whole network functionality. Therefore the use of the commands `ifdown` and `ifup` is sometimes desirable. To reach this, one can edit the `/etc/network/interface` file by adding the following lines:

```
auto eth1
iface eth1 inet ipv4ll
```

In our environment the `eth1` interface is used for Zeroconf. So it should get an `inet` (IPv4) address and more specific an IPv4ll-address. Avahi-autoipd then takes care of it and creates the `eth1:avahi` interface as described in section 3.2.3

Also with configuring an interface speeds up the establishment of the connection. Because the interface does not search for an DHCP-server before it changes to link-local addressing.

When using `dhcp` instead of `ipv4ll` at the interface configuration line, the host tries to first get an IP-address from a DHCP-server in the network. In our test this took about five minutes to time out, where after the interface gets an IPv4ll address within 10 seconds.

Without interface configuration, the command `avahi-autoipd -D eth1` is needed to create an IPv4ll address for the interface.

#### 4.1.1 Address conflict

To see what happens when an address conflict occurs, `vanilla.local` was manually configured with the IPv4ll address of `tomato.local`.

After configuring the IP-address on `vanilla.local`, the interface at `tomato.local` is restarted (packet 11). During the restart `tomato.local` wants to claim the IP-address again and probes with ARP-requests if the address is in use (packet 12). `vanilla.local` sends a reply that it has allocated the IP-address (packet 13). So `tomato.local` chooses an other IP-address and ask again (packet 14, 16, 18). No answer was received and `tomato.local` allocated the address (packet 19). See figure 8.

No.	VLA# Time	Source	Destination	Protocol	Info
11	7.731713	145.100.104.16	224.0.0.22	IGMP	V3 Membership Report / Leave group 224.0.0.251
12	8.531046	De11_e1:3f:bb	Broadcast	ARP	who has 169.254.170.77? Tell 0.0.0.0
13	8.531191	De11_e1:41:bf	De11_e1:3f:bb	ARP	169.254.170.77 is at 00:15:c5:e1:41:bf
14	8.839604	De11_e1:3f:bb	Broadcast	ARP	who has 169.254.9.234? Tell 0.0.0.0
15	9.015602	169.254.170.77	224.0.0.22	IGMP	V3 Membership Report / Join group 224.0.0.251 for any sources
16	10.617415	De11_e1:3f:bb	Broadcast	ARP	who has 169.254.9.234? Tell 0.0.0.0
17	11.759704	145.100.104.16	224.0.0.22	IGMP	V3 Membership Report / Leave group 224.0.0.251
18	12.332165	De11_e1:3f:bb	Broadcast	ARP	who has 169.254.9.234? Tell 0.0.0.0
19	14.333272	De11_e1:3f:bb	Broadcast	ARP	Gratuitous ARP for 169.254.9.234 (Request)
20	14.343713	169.254.9.234	224.0.0.22	IGMP	V3 Membership Report / Join group 224.0.0.251 for any sources

Figure 8: Wireshark screenshot of an address conflict packetdump

## 4.2 Service Discovery

The discovery of services is an essential part of Zeroconf. During the test it seems that not every software package advertise their service/existence by default. For some packages like the Apache2 webserver a plugin is available named libapache2-mod-dnssd which enables the announcement of the http-service.

Other packages like SSH or vsftpd does not have such a plugin. But it is still possible to announce the existence of a SSH or FTP-server within the network. This can be accomplished with a \*.service file placed in:

```
/etc/avahi/service
```

How these \*.service-files are set up is described in section 2.2.3

## 4.3 IP lookup

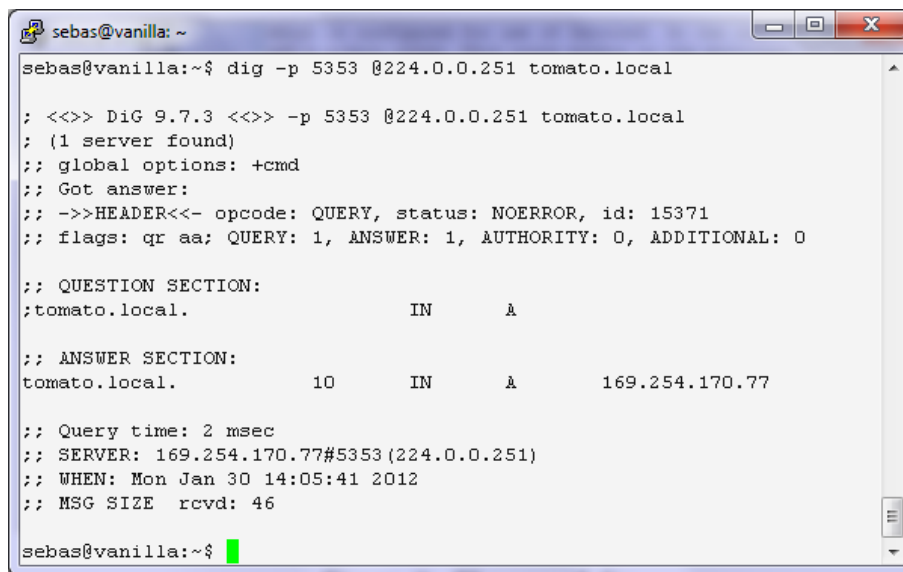
Using the multicast IP-address to resolve a host name to a IP-address with the DIG-tool may fail when using two interfaces. Because the multicast is send on the primary interface and not the interface which is configured for use with Zeroconf. So the request will timeout. To solve this one can add a static route which points to the multicast IP-address on eth1.

```
sudo route add 224.0.0.251 eth1
```

After adding the route, resolving a host name with dig works well, see figure 9.

IP lookup using the avahi-resolve tool, works without adding a route for the multicast IP-address, and is preferred. Although sometimes using the avahi-resolve tool instead of the IPv4 address of the host the IPv6 is shown. This is caused because the standard Avahi configuration also publishes an AAAA-record via IPv4. This can be turned off using the





```
sebas@vanilla:~$ dig -p 5353 @224.0.0.251 tomato.local

; <<>> DiG 9.7.3 <<>> -p 5353 @224.0.0.251 tomato.local
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15371
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;tomato.local.                IN      A

;; ANSWER SECTION:
tomato.local.                10      IN      A      169.254.170.77

;; Query time: 2 msec
;; SERVER: 169.254.170.77#5353(224.0.0.251)
;; WHEN: Mon Jan 30 14:05:41 2012
;; MSG SIZE rcvd: 46

sebas@vanilla:~$
```

Figure 9: The use of dig

”publish-aaaa-on-ipv4=no” in the avahi configuration file. Running avahi-resolve a second time gives then the IPv4 address.

## 4.4 Cross platform

In situations where there is not always an internet or management connection is available there is a need for some kind of cross platform support. Like a Mac or Windows client who is connected to the network. In these situations we work with Avahi on the servers and Bonjour on the client, but this should not be a problem since they do practically the same.

In the test environment I used a Windows 7 Professional laptop and a Mac Mini with Mac OS X. After connecting both hosts to the network they got an IPv4ll address and could communicate with the other hosts.

### 4.4.1 Microsoft Windows 7

Windows has standard build in the functionality to use IPv4ll addresses if no other network configuration is available like manual configuration or a DHCP-server. Microsoft calls this Automatic Private Internet Protocol Addressing (APIPA) [1].

To make use of Zeroconf, Bonjour has to be installed. It can be installed with iTunes or the Safari Browser.

After installing Safari, the web server running on the `vanilla.local` server, was found using the Bonjour bookmark menu. See figure 10.

It is also possible to announce the existence of shared folders in present in Windows. Therefore is the Bonjour SDK for Windows [6] is needed. It

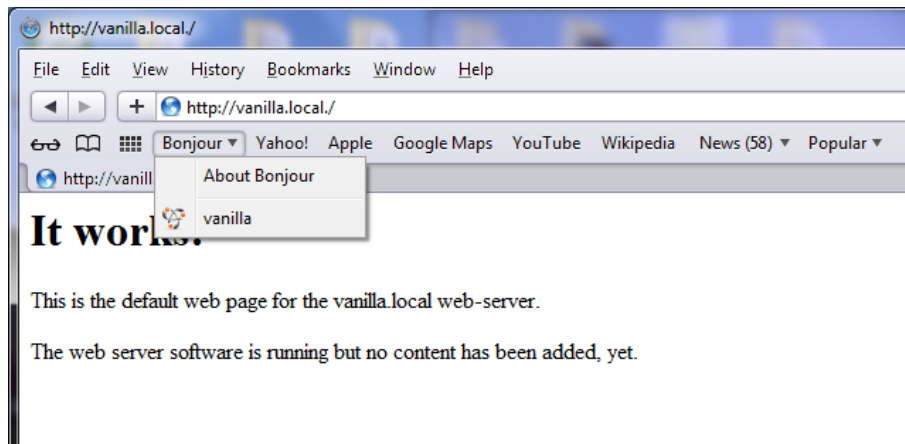


Figure 10: Screenshot of Safari with the `vanilla.local` webpage

installs a Control Panel program, in which one is able to select advertise shared folders in Bonjour, see figure 11.

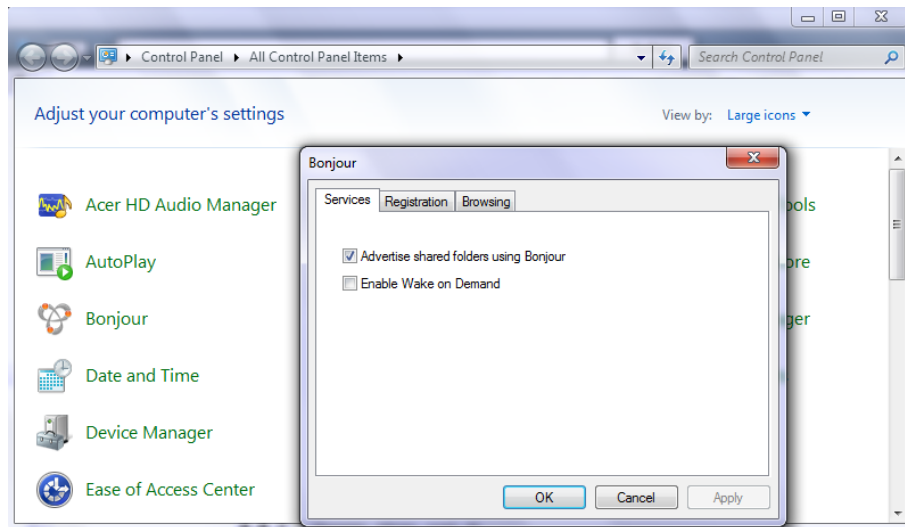


Figure 11: Screenshot of the Bonjour SDK

#### 4.4.2 Apple Mac OS X

Mac OS has built in Zeroconf support using Bonjour. (The Mac mini, which I used, is used to host a webcam using evocam on conferences. I didn't change the configuration to don't mess it up).

After I connected the Mac mini to the network and start it, the network configuration was done without any interaction of me. It was really Zeroconf.

The interface got a IPv4ll-address and was fully operational, ping as well

connection via SSH to the servers was no problem. Even ping using IPv6 worked.

Using the Safari browser, the Bonjour bookmark found the website running on `tomato.local`.

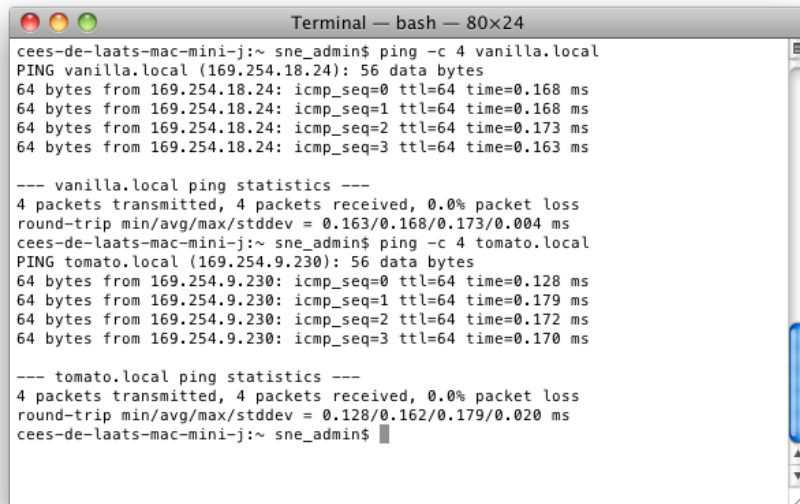
A screenshot of a macOS Terminal window titled "Terminal — bash — 80x24". The window shows the output of two ping commands. The first command is `ping -c 4 vanilla.local`, which shows four successful pings to 169.254.18.24 with times between 0.163 ms and 0.173 ms. The second command is `ping -c 4 tomato.local`, which shows four successful pings to 169.254.9.230 with times between 0.128 ms and 0.179 ms. Both commands also display summary statistics for each host, indicating 0.0% packet loss and round-trip times.

Figure 12: Pinging the servers from the Mac mini

## 4.5 Security

Zeroconf is designed to work on a small local area network or in ad-hoc networks, so there is no really security implemented. The techniques are kept very simple for ease of use.

However, in the configuration file as show in appendix A, there are some parameters which can add some basic kind of security, like disabling the publishing of services and names. So the host will be a bit invisible.

Since there is a dedicated circuit where only trusted hosts connected to this should not be a risk.

## 4.6 Timing

An other important factor is timing, the end-host should be online quickly. To test how long it takes, for a end host to come online, I did the following:

```
Ping from vanilla.local to tomato.local
Perform a "sudo ifdown eth1 && sudo ifup eth1" at inter-
face eth1 on tomato.local
```

While doing that, I ran tcpdump on eth1 at `vanilla.local` to capture the packets what are going over the line.

As seen in figure 13 `vanilla.local` performs first a IP lookup for the host name `tomato.local` via mDNS. After receiving the answer it starts with the ping. Four reply's are received, after which the restart of the interface is performed.

During the restart of the interface `tomato.local` leaves the mDNS multicast group (packet 13). After becoming online again, the interface checks if the IP-address it wants is available (see packet 15, 17 and 19). If no answer is received the `tomato.local` claims the IP-address and joins the multicast group again (packet 22 and 23).

During all the time `vanilla.local` pings and get no response. After `tomato.local` is online again it performs an ARP request to get the MAC-address of `vanilla.local` to reply to the ping request (packet 27 and 28). Finally packet 29 shows that `tomato.local` reply to the pings again.

Time needed to become online again is about 6 seconds from 3.35 to 9.00.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	169.254.18.24	224.0.0.251	MDNS	Standard query A tomato.local, "QM" question
2	0.000212	169.254.170.77	224.0.0.251	MDNS	Standard query response A, cache flush 169.254.170.77
3	0.000613	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
4	0.000712	169.254.170.77	169.254.18.24	ICMP	Echo (ping) reply
5	0.101229	169.254.18.24	224.0.0.251	MDNS	Standard query PTR 77.170.254.169.in-addr.arpa, "QM" question
6	0.101429	169.254.170.77	224.0.0.251	MDNS	Standard query response PTR, cache flush tomato.local
7	1.001593	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
8	1.001689	169.254.170.77	169.254.18.24	ICMP	Echo (ping) reply
9	2.000594	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
10	2.000690	169.254.170.77	169.254.18.24	ICMP	Echo (ping) reply
11	3.000417	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
12	3.000513	169.254.170.77	169.254.18.24	ICMP	Echo (ping) reply
13	3.347640	145.100.104.16	224.0.0.22	IGMP	V3 Membership Report / Leave group 224.0.0.251
14	4.000415	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
15	4.309588	Dell_e1:3f:bb	Broadcast	ARP	who has 169.254.170.77? Tell 0.0.0.0
16	5.000414	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
17	5.803118	Dell_e1:3f:bb	Broadcast	ARP	who has 169.254.170.77? Tell 0.0.0.0
18	6.000415	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
19	6.917208	Dell_e1:3f:bb	Broadcast	ARP	who has 169.254.170.77? Tell 0.0.0.0
20	7.000414	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
21	8.000413	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
22	8.918296	Dell_e1:3f:bb	Broadcast	ARP	Gratuitous ARP for 169.254.170.77 (Request)
23	8.927620	169.254.170.77	224.0.0.22	IGMP	V3 Membership Report / Join group 224.0.0.251 for any sources
24	8.950355	169.254.170.77	224.0.0.251	MDNS	Standard query response PTR _ftp_tcp.local PTR tomato [00:15:c5:e1:41:bf]
25	8.978784	169.254.170.77	224.0.0.251	MDNS	Standard query ANY b.b.f.3.1.e.e.f.f.5.c.5.1.2.0.0.0.0.0.0.0
26	9.000419	169.254.18.24	169.254.170.77	ICMP	Echo (ping) request
27	9.003633	Dell_e1:3f:bb	Broadcast	ARP	who has 169.254.18.24? Tell 169.254.170.77
28	9.003644	Dell_e1:41:bf	Dell_e1:3f:bb	ARP	169.254.18.24 is at 00:15:c5:e1:41:bf
29	9.003757	169.254.170.77	169.254.18.24	ICMP	Echo (ping) reply

Figure 13: Wireshark screenshot

## 5 Conclusion

As described in the previous section the automatic configuration worked well, and should be suitable for use in circuit-based networks.

The situation now is that setting up a inter-domain connection is time intensive and can take months and a huge amount of emails messages. in the future this can become easier with automated GOLE, described in chapter 2. There it is desirable to have almost instantly a connection when connecting the server to the network. With a proper configured interface this will be accomplished within 10 seconds, which fits in the requirement.

Avahi provides an ease to use Zeroconf solution for Linux-based hosts. It can run dual stack on both IPv4 and IPv6, this gives no problems in our environment. Although some configuration is needed, to fit in the environment. Like the interface on which the Avahi should listen or the IP version, if one don't want to run dual stack.

The automatic assignment of IP-addresses did give no problems at all. Also IP-address conflict detection, when two host have assigned the same IP-address works as expected and described in the RFC.

The cross-platform test using a Windows 7 client and Mac client running Bonjour in combination with Avahi on the Linux servers didn't give any problems. Although the test with the Mac works a bit better since MAC OS X has out of the box Bonjour support, which Windows not (yet) have. Maybe a feature which should be implemented in future versions of Windows.

## 6 Further research

In the environment where the System and Networking research group possibly wants to implement the Zeroconf implementation in a group of organisations, which share network equipment and servers could be Zeroconf a great opportunity to simplify the basic network configuration.

However it seems to be a difficult task to convince all cooperation's involved to do so. Since Zeroconf is implemented mostly in workstation operating systems, and system peripherals like printers, webcams. The configuration of servers takes some effort at the moment.

This document could be a starting point for that.

## A Avahi-daemon configuration file

```
# This file is part of avahi.
#
# avahi is free software; you can redistribute it and/or
# modify it
# under the terms of the GNU Lesser General Public License
# as
# published by the Free Software Foundation; either version
# 2 of the
# License, or (at your option) any later version.
#
# avahi is distributed in the hope that it will be useful,
# but WITHOUT
# ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
# or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
# Public
# License for more details.
#
# You should have received a copy of the GNU Lesser General
# Public
# License along with avahi; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston,
# MA 02111-1307
# USA.

# See avahi-daemon.conf(5) for more information on this
# configuration
# file!

[server]
#host-name=foo
#domain-name=local
#browse-domains=0pointer.de, zeroconf.org
use-ipv4=yes
use-ipv6=no
allow-interfaces=eth1
deny-interfaces=eth0
#check-response-ttl=no
#use-iff-running=no
#enable-dbus=yes
#disallow-other-stacks=no
#allow-point-to-point=no
```

```
#cache-entries-max=4096
#clients-max=4096
#objects-per-client-max=1024
#entries-per-entry-group-max=32
ratelimit-interval-usec=1000000
ratelimit-burst=1000
```

```
[wide-area]
enable-wide-area=yes
```

```
[publish]
#disable-publishing=no
#disable-user-service-publishing=no
#add-service-cookie=no
#publish-addresses=yes
#publish-hinfo=yes
#publish-workstation=yes
#publish-domain=yes
#publish-dns-servers=192.168.50.1, 192.168.50.2
#publish-resolv-conf-dns-servers=yes
#publish-aaaa-on-ipv4=yes
#publish-a-on-ipv6=no
```

```
[reflector]
#enable-reflector=no
#reflect-ipv=no
```

```
[rlimits]
#rlimit-as=
rlimit-core=0
rlimit-data=4194304
rlimit-fsize=0
rlimit-nofile=768
rlimit-stack=4194304
rlimit-nproc=3
```



## References

- [1] Apipa msdn. Website. available at <http://msdn.microsoft.com/en-us/library/aa505918.aspx>; on 30th January 2012.
- [2] avahi-autoipd(8) - linux man page. Website. available at <http://linux.die.net/man/8/avahi-autoipd>; on 30th January 2012.
- [3] avahi-daemon.conf(5) - linux man page. Website. available at <http://linux.die.net/man/5/avahi-daemon.conf>; on 30th January 2012.
- [4] avahi-publish-service(1) - linux man page. Website. available at <http://linux.die.net/man/1/avahi-publish-service>; on 31th January 2012.
- [5] avahi.service. Website. available at <http://avahi.org/download/avahi.service.5.xml>; on 24th January 2012.
- [6] Configuring clients to use wide-area bonjour. Website. available at <http://www.dns-sd.org/ClientSetup.html>; on 31th January 2012.
- [7] Dns srv (rfc 2782) service types. Website. available at <http://www.dns-sd.org/ServiceTypes.html>; on 25th January 2012.
- [8] libnss-mdns. Website. available at <http://0pointer.de/lennart/projects/nss-mdns>; on 30th January 2012.
- [9] List of ipv4 multicast addresses. Website. available at <http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xml>; on 25th January 2012.
- [10] List of ipv6 multicast addresses. Website. available at <http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml>; on 25th January 2012.
- [11] Net-thinkers mailing list. Website. available at <http://www.stuartcheshire.org/rants/NBPIP.html>; on 26th January 2012.
- [12] Zeroconf working group. Website. available at <http://www.zeroconf.org/>; on 04th January 2012.
- [13] S. Cheshire, B. Aboba, and E. Guttman. *Dynamic Configuration of IPv4 Link-Local Addresses RFC 3927*. available at <http://tools.ietf.org/html/rfc3927>; on 04th January 2012.
- [14] Stuart Cheshire and Marc Krochmal. *DNS-Based Service Discovery*. available at <http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt>; on 30h January 2012.

- [15] Stuart Cheshire and Marc Krochmal. *Multicast DNS draft*. available at <http://tools.ietf.org/html/draft-cheshire-dnsext-multicastdns-15>; on 25th January 2012.
- [16] Freek Dijkstra, Jeroen J. van der Ham, and Cees T.A.M. de Laat. Using zero configuration technology for ip addressing in optical networks. *Future Generation Computer Systems*, 22(8):908–914, May 2006.
- [17] Ralph Droms. *Dynamic Host Configuration Protocol RFC 2131*. available at <http://tools.ietf.org/html/rfc2131>; on 22th January 2012.
- [18] Robert M. Hinden and Stephen E. Deering. *IP Version 6 Addressing Architecture RFC 4291*. available at <http://tools.ietf.org/html/rfc4291>; on 25th January 2012.
- [19] Susan Thomson, Thomas Narten, and Tatuya Jinmei. *IPv6 Stateless Address Autoconfiguration RFC 4862*. available at <http://tools.ietf.org/html/rfc4862>; on 25th January 2012.