

**A proposal for decentralized, scalable  
and automatic IP network  
configuration: SCNP**

*Thomas Delaet  
Thomas Clijsner*

*Peter Rigole*

*Wouter Joosen*

*Yolande Berbers*

*Report CW 408, April 2005*



**Katholieke Universiteit Leuven**  
Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

# A proposal for decentralized, scalable and automatic IP network configuration: SCNP

*Thomas Delaet*

*Thomas Clijsner*

*Peter Rigole*

*Wouter Joosen*

*Yolande Berbers*

*Report CW 408, April 2005*

Department of Computer Science, K.U.Leuven

## **Abstract**

The current generation of networking protocols uses the Internet Protocol as a ubiquitous layer for building protocols tailored for a specific kind of network. Some protocols focus on ad-hoc networks, other on typical enterprise networks. Home networks are a hybrid form of ad-hoc networks and typical networks. No currently existing protocol can solve the problem of network configuration for this kind of networks. Our solution bridges the gap between a protocol assigning IP addresses and managing routing information while guaranteeing IP address uniqueness. To realise this last part, we developed a conflict resolution algorithm. The combination of these three elements has resulted in the self-configuring network protocol, SCNP.

# A proposal for decentralized, scalable and automatic IP network configuration: SCNP

Thomas Delaet, Thomas Clijsner, Peter Rigole, Wouter Joosen, Yolande Berbers  
DistriNet, Department of Computer Science, K.U.Leuven  
Celestijnenlaan 200A  
3001 Leuven, Belgium  
{thomas,petteri,wouter,yolande}@cs.kuleuven.ac.be

## Abstract

*The current generation of networking protocols uses the Internet Protocol as a ubiquitous layer for building protocols tailored for a specific kind of network. Some protocols focus on ad-hoc networks, other on typical enterprise networks. Home networks are a hybrid form of ad-hoc networks and typical networks. No currently existing protocol can solve the problem of network configuration for this kind of networks. Our solution bridges the gap between a protocol assigning IP addresses and managing routing information while guaranteeing IP address uniqueness. To realise this last part, we developed a conflict resolution algorithm. The combination of these three elements has resulted in the self-configuring network protocol, SCNP.*

## 1. Introduction

In this paper, we describe a protocol that addresses the network configuration problem in home networks. Home networks are typically a mix of relatively stable and mobile parts. As such they can not be seen as ad-hoc networks, nor as fixed networks. An example will demonstrate this: devices like a TV, audio equipment and a router can be considered as stable parts of a home network, while devices like a laptop, an MP3 player and even a car (which can contain a whole network on its own) are mobile parts.

The problem is that home networks can also potentially contain a lot of different subnets. This can be easily demonstrated by the fact that the networking equipment by which devices are connected to the network is highly versatile. For example: an MP3 player connected with a USB cable to a personal computer can be considered as a separate subnet. Some devices are connected through a wireless network, others with a standard Ethernet network and still others with a Powerline network.

Home networks can be connected to the Internet (potentially with multiple connections), but this is not a requirement. This makes it impossible to rely on the presence of a central component, like a gateway.

We can identify three functional requirements in the problem of network configuration:

1. *Initial autoconfiguration*: When a router or host joins the network, each interface has to get a unique IP address. This implies generating a unique subnet identifier (the first part of an IP address) and interface identifier (the last part of an IP address).
2. *Routing*: A router - all hosts with more than one interface are considered routers - needs to know how to reach subnets other than those which are directly connected to the router. Therefore, each router runs a routing algorithm.
3. *Address Uniqueness Guarantee*: This requirement comes into play when for example two distinct networks (each with several subnets) get connected on-the-fly through two wireless devices. This merging scenario could imply the occurrence of duplicate subnet identifiers after the merge, i.e. one of the two subnets with a duplicate subnet identifier will become unreachable. Therefore, we have to support continuous duplicate detection for subnet identifiers.

Starting from the description of a typical home network in the beginning of this section, the following three non-functional requirements can be deduced:

1. *Scalability*: As a result of the hierarchical topology (two levels) of networks, two kinds of scalability can be identified: (1) scalability in one subnet and (2) scalability in the sense that the protocol can handle a network with a large number of subnets. We believe that the second kind of scalability will be more vital than the former. This assumption can be easily illustrated

with a simple example: in a home network, it is more likely to have different networking hardware (Wireless, Ethernet, PowerLine, etc.) which results in multiple subnets than having a few hundred devices in one subnet.

2. *Decentralization*: In a home networking context, every device can potentially be turned off at any moment in time. As a consequence, an algorithm should not be dependent on the presence of any (special) device to assure the correct functioning of the network. This means a good algorithm can not assume one or more central components.
3. *Self-configuration*: We can not assume the users of the network having the time nor the skills to care about configuring the network settings. This requires the protocol to be fully automatic and work without any user intervention.

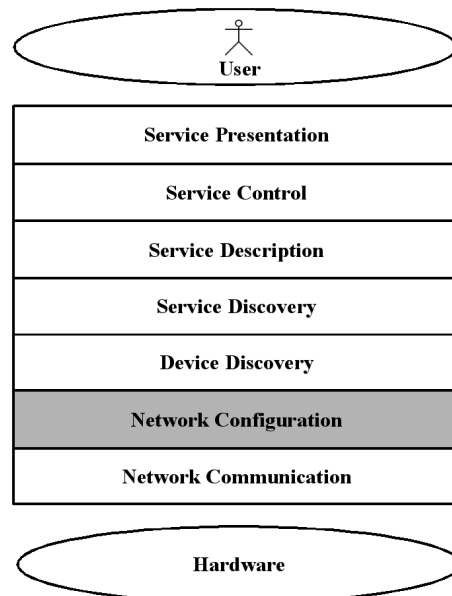
The biggest drawback of existing solutions (we will discuss them in section 2) is that most of them are not scalable enough to deal with more than one subnet. Our solution for the described functional and non-functional requirements is called the self-configuring network protocol (SCNP). This protocol is scalable, fully automatic and completely decentralized.

Automatic network configuration is only part of the problem of creating a ubiquitous computing environment (in the home or other contexts). Figure 1 represents our layered framework of the problems that need to be solved to create a ubiquitous computing environment. The division of the different problems and requirements that need to be solved is strongly inspired by the framework that UPnP [7] proposes.

As in figure 1 can be seen, users only get in contact with the presentation of services (*Service Presentation*). Underneath this layer, a service control mechanism and service descriptions take care of the functioning of these services. To be able to discover services on the network, a search strategy has to be used which is part of the *Service Discovery* layer. This requires that each device possesses a (virtual) view on the network, solved in the *Device Discovery* layer. The *Network Configuration* layer is the one for which we propose a solution in this paper (SCNP). Finally there is the need for a uniform communication protocol on top of the physical network connections. This is necessary because all devices need to be able to communicate with each other by using a standard message format and a unique addressing scheme. For a more detailed description, we refer to [2].

Our framework as described above differs from the UPnP framework [7] on two key issues:

1. UPnP has only one discovery layer. We splitted this up in *Device Discovery* and *Service Discovery*. The rea-



**Figure 1. Layered representation of the problems that need to be solved to create a ubiquitous computing environment.**

son for this is simple: there is a clear distinction between discovering other devices on one side and the execution of a query for a specific service on the other side.

2. We also added the *Network Communication* layer because we recognize a difference between (1) the protocol that offers a uniform packet format and a unique addressing schema (IPv6, for example) and (2) the kind of protocol for the *Network Configuration* layer (such as the one described in this paper) that defines how to generate unique addresses and exchange routing information.

The rest of this paper is organized as follows: in section 2 we will shortly present related work. We describe our solution in section 3 and evaluate it in section 4. Section 5 of this paper summarizes our protocol.

## 2. Related Work

There are some protocols, which address (at least partially) the same problem domain, namely: DHCP, IPv6 Stateless Autoconfiguration, AutoIP and Zeroconf. We will now briefly discuss them and verify how well they address

the functional and non-functional requirements mentioned in section 1.

## 2.1. DHCP

The Dynamic Host Configuration Protocol (DHCP) [6] supports the dynamic configuration of all kind of network settings. DHCP only takes care of the *Initial autoconfiguration* part from our functional requirements. When a host comes online it checks if a DHCP server exists and gets all its network settings from this server. DHCP is a scalable protocol: it is fairly lightweight and has the possibility to use relay agents to accumulate all the requests from different subnets. On the other hand, DHCP is not decentralized (it depends on one central server, which is a single point of failure) and is not self-configuring (although administration is reduced and centralized on the DHCP server). To accommodate the second functional requirement (*Routing*), a routing protocol needs to be used together with DHCP. Since all addresses are distributed by the central DHCP server the *Address Uniqueness Guarantee* is achieved, but DHCP can not deal with the merging of two networks.

## 2.2. AutoIP and Zeroconf

Both AutoIP [14] and Zeroconf [1] use very similar approaches. They both assign non-routable addresses on a single subnet. The technique used for checking uniqueness of an address is link-local multicast. Both AutoIP and Zeroconf use IPv4 addresses.

Since AutoIP and Zeroconf implement the *Initial autoconfiguration* requirement within one subnet, there is no need for a routing protocol (the second functional requirement) and the *Address Uniqueness Guarantee*. This limits the scalability a lot. Scalability within one subnet is typically achieved up to 30-50 devices in one subnet [8], which is a realistic assumption. But there is no scalability in the number of subnets (maximum one subnet). AutoIP and Zeroconf do exhibit the two other non-functional requirements (decentralized and self-configuring).

AutoIP is used in the UPnP protocol stack as an alternative for DHCP.

## 2.3. IPv6 Stateless Autoconfiguration

IPv6 Stateless Autoconfiguration [13, 12, 9, 5] works with ICMPv6 [4] messages to give each host a unique address. It is the responsibility of the routers to multicast the subnet identifier that each host has to use on a specific subnet. Each host can check uniqueness of the interface identifier (last part of address) by using link-local multicast. The IPv6 address that is constructed by combining the subnet identifier from the router and the interface identifier is a

routable IPv6 address. Additionally a non-routable IPv6 address is also generated.

For the non-routable IPv6 address, the same argumentation is valid as described in the previous section (not scalable, decentralized and self-configuring). The routable IPv6 address is not self-configuring (the subnet identifier has to be configured on the routers), decentralized (multiple routers on a subnet can provide the subnet identifiers) and scalable (works within multiple subnets). As with the DHCP protocol, a routing algorithm is necessary to complement the IPv6 Stateless Autoconfiguration protocol.

## 2.4. Evaluation

From the list of functional requirements, none of the protocols described above can handle the merging of networks which consists of different subnets (third functional requirement).

The three non-functional requirements are never simultaneously achieved in any of the protocols.

## 3. Proposed Solution

In this section, we explain the functionality of our protocol by describing the different scenarios that can occur in a typical home network. The reader will notice that we try to reuse existing protocols whenever this is appropriate.

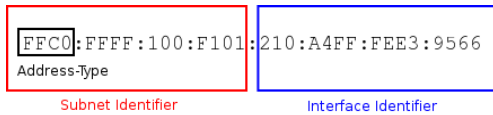
The presence of IPv6 [5] support is a requirement for our protocol. The primary reason for building our protocol on IPv6 is the large private address space that IPv6 offers. Private addresses are not routable on the Internet. The reason why they are used in networks such as home networks is because they are freely available and do not require any kind of registration at a central registration authority. Addresses in the IPv6 private address space are called site-local and link-local addresses [9]. Site-local addresses are routable between different subnets. Link-local addresses can only be used within one subnet.

### 3.1. Host startup

We define a host as a computing device with one network interface, as opposed to a router, which has more than one network interface.

When a host starts up, there are three tasks that need to be accomplished:

1. *Link-local address configuration*: This is defined by the IPv6 standard [5, 9]. An IPv6 address has a length of 128 bits. An address is divided in a subnet identifier (first part) and an interface identifier (last part). Since link-local addresses only work within one subnet, the subnet identifier is fixed. A link-local address always starts with the same fixed sequence of 64 bits. A host



**Figure 2. Site-local address parts.** The “Address-type”-bits are the bits which are fixed and denote a site-local address. The subnet identifier is unique between subnets. The interface identifier within one subnet

generates a value for the interface identifier (last 64 bits) and checks the uniqueness of this value by using link-local multicast messages.

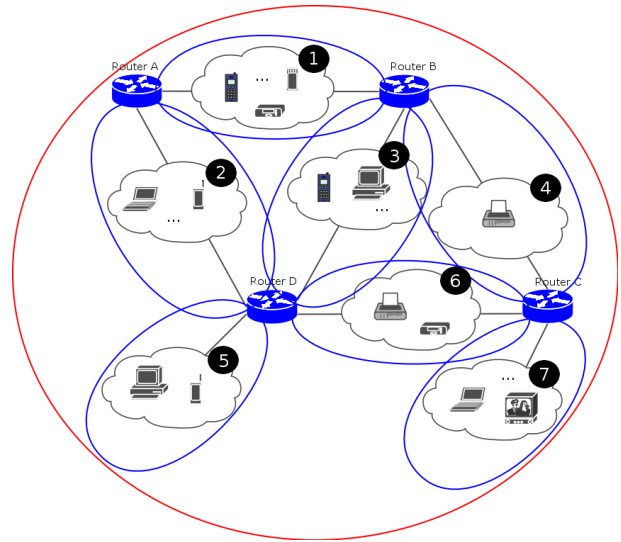
2. *Site-local address configuration:* Figure 2 shows an example site-local address. The “Address-type” box is the fixed part (first 10 bits). The rest of the subnet identifier can be freely used to distinguish different subnets. Notice the hierarchy imposed in the generation of unique site-local addresses: The subnet identifier is unique between the different subnets, the interface identifier is unique within one subnet. The combination results in site-wide unique addresses for every device. When constructing a site-local address, the unique interface identifier (last 64 bits) can be reused from the link-local address. This is also the reason why we imposed a subnet identifier length of 64 bits. The unique subnet identifier is requested by sending a *Router Solicitation* message to the link-local all-routers multicast address. A router answers this query with a *Router Advertisement* message. Notice that all routers are assumed to have unique subnet identifiers.
3. *Default route configuration:* Each host configures a set of default routers to use for communicating with hosts in other subnets. The routers to use can be (randomly) chosen from the routers who send out *Router Advertisement* messages.

When there are no routers present on the subnet, there is no need for configuring site-local addresses and a default route. The only hosts with which communication is possible are those on the local subnet and link-local addresses are being used to communicate within one subnet.

### 3.2. Router startup

The tasks a router needs to execute at startup are very similar to those of a host. Since a router is defined as a device with more than one network interface, each interface has to get a link-local and site-local address.

Take the startup of router C in figure 3 as an example. Router C will connect subnet 7 with the rest of the net-



**Figure 3. A possible graphical representation of a home network.** The clouds represent subnets, connected with routers. The straight lines are the network connections from the routers to the subnets. Uniqueness of the interface identifier has to be achieved in each subnet (inner ovals). Uniqueness of the subnet identifier has to be achieved in the whole network (outer circle).

work. All other subnets and routers are assumed to be already active. As a first task, router C needs to configure a link-local address on each of its three interfaces (to networks 4, 6 and 7). This is exactly the same as in the case of a host startup. Next, Router C sends *Router Solicitation* messages on all the three subnets to which it is connected. In subnet 6, router D replies with a *Router Advertisement* message containing the site-local prefix to use. The same is true for router B on subnet 4. Since router C is the only router in subnet 7, it will be impossible to acquire a subnet identifier. It is now the task of router C to generate a unique subnet identifier.

To check for uniqueness, we have identified two relevant possibilities:

1. Send a message to all possible routers in the whole network to see if they already use a particular subnet identifier.
2. Send only a message to the routers in the neighbouring subnets (routers B and D in this case) and expect them to have the correct information in their routing tables.

For scalability reasons, we have chosen the second option, because the number of messages that need to be sent

on the network is remarkably lower.

Two extra messages are necessary for checking uniqueness of a subnet identifier: a *Subnet Proposal* message and a *Subnet Collision* message. A router first sends a *Subnet Proposal* message. When a duplicate is detected, the router that detects the duplicate replies with a *Subnet Collision* message.

In our example: Router C sends its subnet identifier proposal to routers B and D. They both check their routing tables. When one (or both) of B and D find the proposed subnet identifier in the routing table, a *Subnet Collision* message is returned.

When there is no *Subnet Collision* message received during a certain time-interval, uniqueness of the subnet identifier is assumed. This time-interval is an estimate of the transfer time and processing time for a *Subnet Proposal* message to be sent and a *Subnet Collision* message to return on a particular link type.

Router C will periodically send out *Router Advertisement* messages on all directly connected subnets. It will also answer *Router Solicitation* messages. On subnet 4, Router C will advertise the same subnet identifier as Router B, the same is true for router D on subnet 6.

Router C will advertise the newly generated subnet identifier on subnet 7. The first time Router C sends this *Router Advertisement* message on subnet 7, all hosts will configure a site-local address. Before, the hosts on subnet 7 only had link-local addresses because subnet 7 was an isolated subnet.

Each router also has to run a routing protocol. This is necessary because hosts delegate the correct delivery of inter-subnets packets to the routers. The most important property for a routing protocol is that all routers get all the other subnet identifiers available in the network. This is necessary for duplicate detection of subnet identifiers.

Since we aim to realise a self-configuring network, all the configuration parameters of the routing protocol have to be determined dynamically.

The routing protocol that the SCNP uses is called OSPFv3 [3] protocol (v3 is the IPv6 variant of the OSPF protocol). OSPFv3 is the most widely used routing protocol in private networks. OSPFv3 allows the distribution of routing information throughout the network with minimal overhead and fast adaptation to changing network topologies. RIP [11] is another widely used protocol for private networks. The problem with RIP is that it takes more time to adapt to changes in the network topology (which we described as a property of home networks in section 1).

### 3.3. Two networks merge

When two networks merge, it is possible for duplicate subnet identifiers to appear in the network, since the uniqueness of the subnet identifiers was separately checked in the two networks (before they merged). Suppose  $n_1$  is the number of subnets on the smallest of the two networks involved in the merge.  $n_1$  is the maximum number of subnet identifiers that can cause collisions. In this case, every subnet identifier exists also in the other network.

To detect a conflict, a router monitors all the incoming link-state advertisement messages (this is the name that OSPFv3 uses for messages that contain routing information). A router checks all the subnet identifiers contained in a link-state advertisement message and compares them to the subnet identifiers on all its network interfaces. The link-state advertisement messages offer enough information to distinguish duplicate subnet identifiers and a message from one of the directly connected subnets.

As soon as a duplicate subnet identifier has been detected by a router, it discards this subnet identifier from the corresponding interface and floods the network with a *Subnet Collision* message. When the other router with the same subnet identifier gets this message, it too discards the (duplicate) subnet identifier. All routers also discard the subnet identifier from their routing tables.

When a new subnet identifier needs to be generated, the same procedure is used as described in the previous section (router startup).

When the subnet identifier changes, all hosts on that subnet need to change their site-local address. This is automatically achieved because a prefix advertised in a *Router Advertisement* message has a limited lifetime. This is also the reason why a router needs to send out those *Router Advertisements* periodically. When the subnet identifier changes, a host will not receive *Router Advertisement* messages for the old subnet identifier anymore. After the lifetime is expired, the old subnet identifier will not be used anymore.

### 3.4. Combinations

All possible scenarios that are possible can be described by one or a combination of the three cases we discussed in this section.

The example of a router startup that connects two existing networks (with each multiple subnets) is a combination of the “router startup” and the “two networks merge” cases. In this example, the router does not need to generate any new subnet identifiers, but it is possible that duplicates occur because two networks merge.

In the “router startup” case, we used the routing tables of the neighbouring routers to check for uniqueness of a subnet identifier. When the routing tables are not completely

up-to-date, it is possible for duplicates to occur. The resolution of these conflicts is exactly the same as described in the case when two networks merge.

## 4. Evaluation

We will evaluate our protocol based on the functional and non-functional requirements mentioned in section 1. The number of messages that need to be sent is evaluated quantitatively.

### 4.1. Functional Requirements

We will now shortly recapitulate our protocol. This description is based on the three functional requirements mentioned in section 1.

1. *Initial autoconfiguration*: The IPv6 Stateless Autoconfiguration protocol [13] is used for generating unique interface identifiers and advertising subnet identifiers (*Router Advertisement* and *Router Solicitation* messages). We added the *Subnet Proposal* and *Subnet Collision* messages for generating unique subnet identifiers when a routers starts up.
2. *Routing*: The OSPFv3 [3] protocol is used for routing.
3. *Address Uniqueness Guarantee*: The Address Uniqueness Guarantee consists of two parts:
  - (a) A hook in the routing algorithm which checks each incoming link-state advertisement message for duplicate subnet identifiers.
  - (b) The flooding of *Subnet Collision* messages when a duplicate is detected. After the flooding, the same algorithm as in the *Initial autoconfiguration* phase is used for generating a new subnet identifier.

Our lazy conflict resolution approach is justified because the possibility of two subnet identifiers to collide is extremely low. Clearly, this depends on the number of subnets present. We distinguish two scenarios: (1) the possibility of a collision when generating a new subnet identifier and (2) when two networks merge.

1. *New subnet identifier*: Suppose  $n$  is the number of subnets present. Since 54 bits in our subnet identifier are free for use (the first 10 bits are fixed and denote that we work with site-local addresses), there are  $2^{54}$  possible subnet-values. When we assume random generation of subnet identifiers (this is the worst case, since it does not use any information about unique identifiers (e.g. manufacturer identifier) present on the router, the possibility  $p$  of a collision is  $\frac{n}{2^{54}}$ .

2. *Two networks merge*: Suppose  $n_1$  is the number of subnets in the first network and  $n_2$  in the second network. It is possible that each subnet identifier from the network with the fewest subnets collides with a subnet identifier from the network with the highest number of subnets. When we put this information in a formula, we get:  $p = \frac{n_1 * n_2}{2^{54}}$

### 4.2. Non-functional Requirements

We will now discuss how our protocol addresses the non-functional requirements, mentioned in section 1:

1. *Scalability*: In theory, the limit on the number of subnets is  $2^{54}$  because this is the number of bits that is reserved for the subnet identifier. We will validate the new parts of our protocol (duplicate detection of subnet identifiers) by calculating the message costs.

We calculated the cost of one collision in terms of number of messages. The number of extra messages that need to be sent when a collision occurs is  $n$  (the number of subnets). This is because the *Subnet Collision* message needs to be sent to all subnets (flooding).

The formula for the average number of messages that need to be sent when a collision occurs is recursive. When a new subnet identifier is generated, it is always possible that this is also a duplicate. The worst case is that  $n - 1$  values have to be generated before an unused subnet identifier is found.

The average number of messages  $M = p*(M+n)$ .  $p$  is the possibility of a collision and  $n$  is the number of *Subnet Collision* messages that need to be sent.

This formula can be written without recursion:  $M = \frac{p*n}{1-p}$ .

Our calculations take two simplifications into account:

- (a) We suppose that the possibilities for a collision to occur are independent from each other (in different rounds of the *Subnet Proposal* - *Subnet Collision* cyclis).
- (b) We do not take the *Subnet Proposal* messages into account, because they are only sent to the direct neighbours of the router which generates the subnet identifier.

The worst case is that th router has to try  $n - 1$  subnet identifier values before it has a unique subnet identifier. In this case,  $(n - 1) * n$  *Subnet Collision* messages have to be send out.

The most optimistic case is that, after a collision, the router directly generates a unique subnet identifier. The number of *Subnet Collision* messages that need to be sent is  $n$  in this case.

The complexity of our collision algorithm varies from linear (most optimistic case) in the number of subnets to quadratic (worst case). Since there is a huge difference between the number of possible subnet identifier ( $2^{54}$ ) and any realistic setting (maximum a few thousands subnets), in the vast majority of cases there will be no collisions. The average complexity of our algorithm will be less than linear.

2. *Decentralization*: Our protocol does not depend on any central component of whatever kind. All routers keep information about the subnet identifiers present in the network and the advertising of subnet identifiers on a subnet is done by all the routers connected to that particular subnet.
3. *Self-configuration*: SCNP is self-configuring because (1) the IPv6 Stateless Autoconfiguration protocol takes care of the generation of unique interface identifiers, (2) subnet identifiers are automatically generated and (3) conflicts are triggered by the routing algorithm and automatically resolved.

In short, our protocol fully achieves the three functional requirements as described in 1 and at the same time is able to keep the number of messages that need to be sent less than linear in function of the number of subnets (scalability), is fully automatic (self-configuration) and decentralized.

## 5. Summary

In retrospective, it is clear that our solution adds the glue for combining the IPv6 Stateless [13] Autoconfiguration protocol and the OSPFv3 [3] routing protocol. The added value of our protocol is twofold:

1. The generation of unique subnet identifiers. These are advertised by the IPv6 Stateless Autoconfiguration protocol [13].
2. The continuously checking for duplicates by hooking into the routing algorithm and providing messages for duplicate notification.

We need two extra ICMPv6 [4] messages, namely a *Subnet Proposal* Message and a *Subnet Collision* Message. These messages follow the same format as the messages defined in the IPv6 Stateless Autoconfiguration protocol.

Our solution depends on IPv6. Nevertheless, we do not see any limitations in our protocol so that it could not be adapted to work with IPv4. The advantage of IPv6 over IPv4 is its large address-space and as a consequence fewer changes of duplicate addresses.

SCNP has a few drawbacks in its current form:

- There are no special security precautions taken. Since our algorithm is decentralized, a hostile host can possibly send incorrect information, alter or discard messages. DoS attacks are also easily done. These problems can be partly solved by requiring all hosts to use a security mechanism at the IP level (IPSec [10] comes to mind).
- When joining different networks, it can take some time to reach a stable state. If there are conflicts in the addressing scheme, some parts of the network can be temporarily unreachable. This is undesirable on some types of networks, but not really a problem in the home networking context.

In the near future, we will study (and eventually expand) existing protocols in the higher layers of the services framework, as mentioned in figure 1. An initial prototype of this protocol is described in [2]. We are now finishing a full implementation to be able to benchmark the protocol.

## Acknowledgments

We would like to thank Nico Janssens and Davy Preuveneers for reviewing this paper and provide us with helpful comments.

## References

- [1] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. <http://files.zeroconf.org/draft-ietf-zeroconf-ipv4-linklocal.txt>, 2004.
- [2] T. Clijsner and T. Delaet. ZASA: An architecture for ubiquitous networking. Master's thesis, K.U.Leuven, Dept. of Computer Science, Leuven, Belgium, May 2004.
- [3] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6. <http://www.ietf.org/rfc/rfc2740.txt>, 1999.
- [4] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. <http://www.ietf.org/rfc/rfc2463.txt>, 1998.
- [5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. <http://www.ietf.org/rfc/rfc2460.txt>, 1998.
- [6] R. Droms. Dynamic Host Configuration Protocol. <http://www.ietf.org/rfc/rfc2131.txt>, 1997.
- [7] U. Forum. UPnP Device Architecture. <http://www.upnp.org>, 2000.
- [8] Y. Goland. UPnP Security Flaws. [http://www.goland.org/Tech/upnp\\_security\\_flaws.htm](http://www.goland.org/Tech/upnp_security_flaws.htm), 2002.
- [9] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. <http://www.ietf.org/rfc/rfc2373.txt>, 1998.
- [10] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. <http://www.ietf.org/rfc/rfc2401.txt>, 1998.

- [11] G. Malkin and R. Minnear. RIPng for IPv6. <http://www.ietf.org/rfc/rfc2081.txt>, 1997.
- [12] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). <http://www.ietf.org/rfc/rfc2461.txt>, 1998.
- [13] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. <http://www.ietf.org/rfc/rfc2462.txt>, 1998.
- [14] R. Troll. Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network. <http://www.ietf.org/proceedings/99jul/I-D/draft-ietf-dhc-ipv4-autoconfig-04.txt>, 1999.