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

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

Abstract

This paper presents SCNP, a self-configuring network protocol. Home networks conceptually are a hybrid form of ad-hoc networks and typical enterprise networks. No currently existing protocol can solve the problem of network configuration for home networks. SCNP bridges the gap between a protocol assigning IP addresses and managing routing information while guaranteeing IP address uniqueness. To realize the IP address uniqueness guarantee, we developed a conflict resolution algorithm. The combination of assigning IP addresses, managing routing information and guaranteeing IP address uniqueness has resulted in SCNP.

1. Introduction

The current generation of networking protocols uses the Internet Protocol as a ubiquitous layer for building protocols tailore d for a specific kind of network. Some protocols focus on ad-hoc networks, others on typical enterprise networks. Home networks are a hybrid form of ad-hoc networks and typical enterprise networks.

In this paper, we describe a protocol that addresses the network configuration problem in home networks. We will now first describe the typical characteristics of home networks before we give a more precise definition of the network configuration problem.

Home networks are typically a mix of mobile parts and relatively stable parts. As such they can be seen neither as ad-hoc networks, nor as fixed networks. An example will illustrate this: devices like a TV, audio equipment and a router can be considered stable parts of a home network, while devices like a laptop, an MP3 player and even a car (which can contain a whole network of its own) are mobile parts.

The problem is that home networks can also potentially contain a lot of different subnets. This can easily be 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 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, for example: 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. For this reason, 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, in which case 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 easily be illustrated with a simple example: in a home network, it is more likely to have different networking hardware - which results in multiple subnets - than to have 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 cannot assume one or more central components.
3. *Self-configuration*: We cannot assume that the users of the network will have either the time or the skills to care about configuring the network settings. This requires that the protocol be fully automatic and that it works 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 functional and non-functional requirements described is called the self-configuring network protocol (SCNP). This protocol is scalable, fully automatic and completely decentralized.

The rest of this paper is organized as follows: In section 2 we briefly present related work. In section 3 we describe our solution, and in section 4 we evaluate it.

2. Related Work

There are some protocols, which address (at least partially) the same problem domain, namely: the IPv6 Stateless Autoconfiguration protocol, 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. AutoIP and Zeroconf

Both AutoIP [9] and Zeroconf [1] use very similar approaches. They both assign non-routable addresses on a single subnet. The technique used for checking the 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 either for a routing protocol (the second functional requirement) or for the *Address Uniqueness Guarantee*. This limits the scalability a lot. Scalability within one subnet is typically achieved for up to 30-50 devices in one subnet [5], 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).

2.2. IPv6 Stateless Autoconfiguration

IPv6 Stateless Autoconfiguration [8, 6] works with ICMPv6 messages to give each host a unique address. It is the responsability of the routers to multicast the subnet identifier that each host has to use on a specific subnet. Each host can check the 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, but decentralized and self-configuring). The routable IPv6 address is not self-configuring (the subnet identifier has to be configured on the routers), but decentralized (multiple routers on a subnet can provide the subnet identifiers) and scalable (works within multiple subnets). A routing algorithm is necessary to complement the IPv6 Stateless Autoconfiguration protocol.

2.3. Evaluation

From the list of functional requirements, none of the protocols described above can handle the merging of networks, which involves the merging 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 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. Addresses in the IPv6 private address space are called site-local and link-local addresses [6]. Site-local addresses are routable between differ-

ent 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.

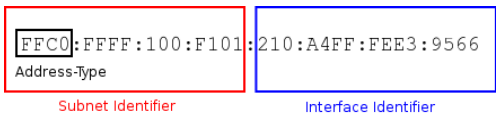

Figure 1. Site-local address parts. The subnet identifier is unique between subnets. The interface identifier is unique within one subnet.

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

1. *Link-local address configuration*: This is defined by the IPv6 standard [6]. An IPv6 address has a length of 128 bits. An address is divided into 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 host 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 1 shows an example of a 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. 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.

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

3.2. Router start-up

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

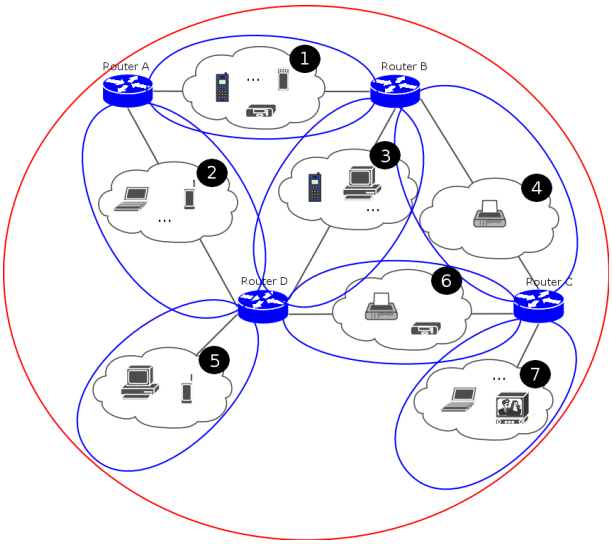

Figure 2. The clouds represent subnets. The straight lines are the network connections from the routers to the subnets. Uniqueness of the interface identifier has to be achieved in each subnet (inner ovals). Uniqueness of the subnet identifier has to be achieved in the whole network (outer circle).

with more than one network interface, each interface has to get a link-local and a site-local address.

Take the startup of router C in Figure 2 as an example. Router C will connect subnet 7 with the rest of the network. 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. 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.

Two extra messages are necessary for checking the uniqueness of a subnet identifier: a *Subnet Proposal* message and a *Subnet Collision* message. A router first sends a *Subnet Proposal* message to its neighbours. When a duplicate is detected, the router that detects the duplicate replies with a *Subnet Collision* message. The neighbours check for duplicates in their routing tables. 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 sub-

net identifier in the routing table, a *Subnet Collision* message is returned. When there is no *Subnet Collision* message received during a certain time-interval, the 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. Previously, 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-subnet 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 realize 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 the 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 [7] 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 checked separately in the two networks (before they merged).

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 between duplicate subnet identifiers on onse side, and a message from one of the directly connected subnets on the other side.

As soon as a duplicate subnet identifier has been detected by a router, the router 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 the other routers also discard the subnet identifier from their routing tables, otherwise the routing tables would get confused.

When a new subnet identifier needs to be generated, the same procedure is used as described in the previous section (Router start-up).

3.4. Combinations

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

The example of a router startup that connects two existing networks (each with 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 will occur because two networks have merged.

4. Evaluation

We will evaluate our protocol based on the functional and non-functional requirements mentioned in section 1. A more quantitative evaluation can be found in [4].

4.1. Functional Requirements

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

1. *Initial autoconfiguration*: The IPv6 Stateless Autoconfiguration protocol [8] 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 router 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.

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 are reserved for the subnet identifier. When there are more subnets, there is a greater possibility of conflicts and the routing algorithm has to send more packets to keep all routing tables up-to-date. It is clear that the possibility of a conflict is extremely low in any realistic setting (maximum a few thousand subnets).
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.

5. Summary

In retrospect, it is clear that our solution contributes the glue for combining the IPv6 Stateless [8] 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 [8].
2. The continuous checking for duplicates by means of hooking into the routing algorithm and providing messages for duplicate notification.

An initial prototype of this protocol is described in [2]. We are now finishing a full implementation so that we can move on to benchmarking the protocol.

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] T. Delaet, T. Clijsner, P. Rigole, W. Joosen, and Y. Berbers. A proposal for decentralized, scalable and automatic IP network configuration: SCNP. Technical report, K.U.Leuven, departement of Computer Science, 2005.

[5] Y. Goland. UPnP Security Flaws. http://www.goland.org/Tech/upnp_security_flaws.htm, 2002.

[6] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. http://www.ietf.org/rfc/rfc2373.txt, 1998.

[7] G. Malkin and R. Minnear. RIPng for IPv6. http://www.ietf.org/rfc/rfc2081.txt, 1997.

[8] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. http://www.ietf.org/rfc/rfc2462.txt, 1998.

[9] 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.