Available online at www.sciencedirect.com
SCIENCE DIRECT
Future Generation Computer Systems 22 (2006) 908–914
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

Using zero configuration technology for IP addressing in optical networks

Freek Dijkstra*, Jeroen J. van der Ham, Cees T.A.M. de Laat

Advanced Internet Research Group, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, Netherlands

Available online 3 May 2006

Abstract

Host configuration in optical networks is usually done by hand. In this paper we propose to use zero configuration techniques, including self-assigned IP addresses and multicast DNS to do this automatically. The proposed technology is designed for small networks without central control, and can be applied to optical private networks as long as there is no router in between the end-hosts.

Keywords: Zero configuration; Ad hoc networks; Optical networks; Hybrid networks

1. Introduction

Optical networks allow users to transfer data between a limited set of locations using dedicated switched circuits (lightpaths). Currently, most of the configuration of lightpaths is done manually, including the host configuration. Host configuration includes the assignment of an IP address, and possibly also modification of the routing table, configuration of the hostname and tuning of protocol stacks.

Widespread adoption of lightpaths benefits from the automation of the setup and configuration processes. Ideally, end-hosts automatically communicate with each other in the best possible way: via dedicated circuits if available, or the best-effort Internet if not. Preferably, applications do not need to know about the exact type of network available, and can transparently switch between dedicated and best-effort networks.

1.1. Requirements

An ideal architecture to handle automatic configuration of end-hosts for complex dynamic networks adheres to the following requirements.

- It supports a scenario where two remote LANs with multiple computers are connected together using a dedicated connection (not just host-to-host connections).
- The architecture scales well.
- It supports dynamic network joins and splits, without impacting stability.
- It handles graceful transition between dedicated networks and the best effort Internet.
- It supports both IPv4 and IPv6 addresses.

In addition, the following features are considered beneficial as well:

- There is a minimum amount of changes to applications, preferably none.
- It respects the administrative boundaries of separate domains. Decision points do not control hosts in other domains.
- It works well with name resolution (e.g. DNS), and service discovery.
- The solution is independent of other technologies (like those on other OSI layers).
- It supports complex host configurations (e.g. routing table changes, tuning of protocol stacks, instead of just assigning an IP address).
- It may support network scenarios where there is a router between the end hosts.

Note that some items may yield conflicting requirements. For example, complex configuration mentions two examples on different OSI layers (respectively the network layer for routing table and the transport layer for protocol stacks). This may conflict with the requirement that any solution is independent of other OSI layer technologies.

* Corresponding author. Tel.: +31 20 5257531.
E-mail address: fdijkstr@science.uva.nl (F. Dijkstra).

0167-739X/$ - see front matter © 2006 Published by Elsevier B.V.
doi:10.1016/j.future.2006.03.021

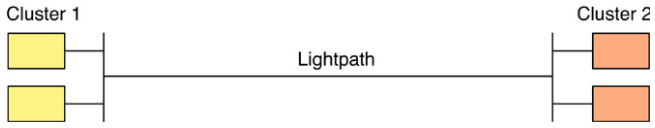

Fig. 1. A typical optical network scenario: two LANs, each with two hosts are joined together using a lightpath.

The last item also deserves a short note: in our opinion, routers on a "lightpath" are undesirable: the network should provide an end-host with either a regular Internet connection (possibly with guaranteed quality of service), or provide a dedicated circuit on layer 1 (or perhaps a layer 2 connection), without any layer 3 device.

Most of the technologies discussed in this paper explicitly assume a layer 2 connection, without routers, like the one shown in Fig. 1. The conclusions in Section 4 discuss the applicability of these techniques in more detail.

1.2. Solution models

A traditional way to solve a complex distributed configuration problem is to introduce a central authority that tells each component its configuration at any point in time. For host configurations, a central decision point alleviates the intelligence from the individual hosts and keeps track of the current network topology.

A hierarchical solution with a central authority violates the end-to-end principle [1], one of the design principles for Internet protocols [2]. Our preferred solution would be one where all layer 3 devices, like end-hosts and routers, autonomously decide on the best configuration without consulting a central authority.

In optical networks, two communicating end-hosts typically fall within different administrative domains. This makes it even more prominent to use a solution which does not require negotiation of leadership, and to avoid a solution where a controller in one domain tells an end-host in another domain how to behave.

2. Zero configuration technologies

Zero configuration networking is a concept to *enable networking in the absence of configuration and administration* [3]. While this is different from our scenario, *network configuration with two or more administrative domains*, zero configuration technology can very well be applied to this scenario.

This section describes three specific zero configuration technologies. More details regarding technology and implementation can be found at [3–5]. The three problems discussed in this section are:

- Automatic assignment of link-local IPv4 addresses, without a DHCP server;
- Translation between hostname and address, without a DNS server;
- Service Discovery, without a directory server.

Link-local, in the context of zero configuration, means a logical (layer 2) network. Thus, a device can always reach all other link-local devices using Ethernet broadcast or multicast packets. A link-local IP address means an IP address which has only a valid meaning on the same link-local network.

2.1. Automatic configuration of IP addresses

There are two specifications for automatic generation and assignment of link-local IP addresses: one for IPv4 and one for IPv6.

The standard for IPv4 is defined in RFC 3927 [6]. Basically, a network device picks an address at random in the 169.254.0.0/16 range, and sends out an ARP request to check if another device is using it. If the device notices that the address is in use, it picks another IP address.

The standard for IPv6 is defined in RFC 2462 [7]. Self-Assigned IPv6 addresses fall in the range FE80::0/10. By default, network devices choose an address by appending their interface identifier (e.g. the MAC address) to the link prefix.

2.2. Name lookup

There are two competing standardization attempts for name lookup without an authoritative name server: Multicast DNS (mDNS) and Link-Local Multicast Name Resolution (LLMNR) [8,9]. LLMNR is under development in the IETF, but is not (yet) used in practice, and is more limited in functionality than mDNS. mDNS has a well-established user-base.

Normally, a host that want to lookup a DNS name consults a central DNS server. The idea implemented by mDNS and LLMNR is that all hosts listen to a specific IP multicast address. A host that wants to do a lookup broadcasts the query on the local-link to this multicast address. Whichever host(s) know(s) the answer to the query replies to the requesting host. This answer is only valid in the context of the local link. Instead of using fully qualified domain name (FQDN), hosts pick a hostname in the `.local` name space.[1]

While Multicast DNS and link-local IP addressing are often used in conjunction, they do not depend on each other.

2.3. Service discovery

There are multiple standards dealing with discovery of services, including:

- Service Location Protocol (SLP), as standardized in the IETF [10]
- DNS Service Discovery (DNS-SD) [11]
- Simple Service Discovery Protocol (SSDP) [12], which is part of Universal Plug and Play (UPnP).

[1] With LLMNR, hosts can use any namespace, which is considered a security risk.

Table 1
Computers in use for the demonstration

Hostname	Location	System	Purpose
VanGogh7	Amsterdam	Intel 32-bit, Red Hat	GridFTP server
VanGogh8	Amsterdam	Intel 32-bit, Red Hat	GridFTP server
Webcam	Amsterdam	G5, Mac OS X	HTTP webcam
igrid-demo05	San Diego	Opteron, Windows XP	Web browser
igrid-demo06	San Diego	Opteron, Fedora	Network visualization
pb-freek	San Diego	G4 (laptop), Mac OS X	GridFTP client
(none)	San Diego	(any)	Visitors laptop

SLP is the only IETF standard, but seems little used at the time of writing.

DNS-SD works particularly well with mDNS, since it also uses DNS records. It is based on the use of SRV records, as described in RFC 2782 [13], but uses another level of indirection (PTR records pointing to SRV and TXT records). DNS-SD is considered a lightweight protocol, and service type registration (e.g. `_http._tcp`) happens on an informal first-come-first-served basis.

SSDP is considered to be more complex than DNS-SD. SSDP uses HTTP notification announcements to discover services as identified by a unique combination of a service type URI and a Unique Service Name. The Device Control Protocols as used by SSDP are supervised by the Universal Plug and Play (UPnP) Steering Committee. UPnP is more formalized than DNS-SD, which may be an advantage or a disadvantage depending on your point of view.

The above protocols all support a scenario without a central directory server. There are more service discovery protocols, like UDDI for web services, Jini for Java objects, Salutation, and Service Discovery Protocol (SDP) for Bluetooth. These protocols can be considered as "higher layer" protocols, because they are application or technology specific.

3. iGrid 2005 demonstration

During previous demonstration events, like SuperComputing 2004, we noticed that IP configuration was a labour-intensive process, even for networks which do not require routable IP addresses. In our opinion this can be made easier using link-local IP addresses. We decided to demonstrate zero configuration networking during iGrid 2005, both to discover possible technical limitations (e.g. due to the long latency) as well as to test if it is feasible that every demonstration uses zero configuration networking on an event like iGrid.

3.1. Network setup

Our network setup is very similar to the one shown in Fig. 1, with three computers in Amsterdam and two computers in San Diego. In addition, we allowed visitors to plug their laptops into the network. The hosts in San Diego were connected together with a small consumer brand Ethernet switch. From there, a 182 ms round-trip link connected this switch to a high-end Ethernet switch in Amsterdam, which connected the three Amsterdam hosts. The whole connection between San Diego and Amsterdam consisted of private links, never touching any router along the way.

As shown by Table 1, we created a heterogeneous environment, by using a diversity of operating systems.

3.2. Implementations

We choose the following zero configuration technologies:

RFC 3927 for self-assigned IPv4 addresses;
mDNS for name lookup;
DNS-SD for service discovery.

For this to work, we needed four pieces of software:

IPv4 link-local software for auto IP configuration;
Kernel patches for auto IP configuration;
mDNS software for name lookup and discovery;
Socket library hooks for name lookup.

A more extensive overview of software, including implementations we did not use, can be found at our website [5].

3.2.1. Link-local IPv4 addresses software

There are two possible approaches to implement self-assigned IPv4 addresses. If the implementation is combined with a DHCP client, it allows for smooth transition between link-local IP addresses (when no DHCP server is found) and routable IP addresses (when a DHCP server is available). Another approach is to use it stand-alone and always pick a link-local IP address. The transition approach was not necessary for our situation, so we used the latter approach.

We used the following implementations:

Apple Darwin kernel where it is integrated with the bootp package [14];
Microsoft Windows Windows XP;
ZCIP (Zero-Conf IP) a stand-alone implementation for Linux that relies on the libnet and libpcap libraries [15];
Porchdog Howl a Linux implementation which includes the stand-alone autoipd daemon [17].

3.2.2. Kernel patches

Broadcasting of ARP replies as described in Section 2.5 and 4 of RFC 3927 aids in detection of IP conflicts after a network join. Network joins are a prime feature in optical networks.

ARP replies must be handled in kernel-space, rather then user-space. Since the default Linux kernel does not (yet) support this, we wrote a kernel-patch for this feature.

3.2.3. Multicast DNS software

Most software combines support for mDNS and DNS-SD. We used these three implementations:

mDNSResponder also known as Bonjour [16], an open-source implementation by Apple Computer, with interfaces for C and Java and is available on Mac OS X, Windows, and Linux;

tmDNS tiny multicast DNS, from the same project as ZCIP [15];

Porchdog Howl a Linux implementation of mDNS and DNS-SD [17].

3.2.4. Socket library hook

Most applications have no knowledge of multicast DNS for name lookup. Instead of changing the applications, a better approach is to alter the behaviour of the UNIX socket library functions `gethostbyname()` and `gethostbyaddr()` to use multicast DNS, beside the regular DNS and the `/etc/hosts` file.

There are at least two ways to create these hooks.

On Linux, the cleanest method is to change the behaviour of the Name Service Switch (which handles `gethostby*()` calls). libnss_mdns is a library written by Andrew White to do exactly this. libnss_mdns is distributed with mDNSResponder from Apple Computer.

tmDNS, on the other hand, uses a different approach, and listens to both port 5353 (multicast DNS) as well as 53 (regular DNS) on the localhost interface. Localhost is specified as the first line in `/etc/resolv.conf`, so that all requests first reach the tmDNS daemon. If it gets a link-local specific request,[2] it will then forward the request to multicast DNS. For other requests, it returns a failure, so that the next (regular) name server is used.

3.3. Demonstration software

In our demonstration, we used two discovery mechanisms: one host discovery mechanism and a service discovery mechanism. Most applications only use service discovery, but we also wanted to detect laptops of visitors that did not offer any services, let alone advertise them.

For host discovery, we used a simple Python script to sent out an IP broadcast ping, and cache the results. We visualized the results with a Java applet, as shown in Fig. 2. By using the round-trip-time as a clue, we could guess the location of each host: Amsterdam or San Diego. Every host in the figure picked a link-local IP address in the 169.254/16 range, as well as a hostname in the `.local` domain. Only the host in San Diego did not announce its hostname, since that machine did not support multicast DNS (in this case, due to an bug in one of the software packages).

The software is available for download at our website [5].

[2] Any request for a name in `.local`, `.254.169.in-addr.arpa`, `.8.e.f.ip6.arpa`, `.9.e.f.ip6.arpa`, `.a.e.ip6.arpa` or `.b.e.f.ip6.arpa`.

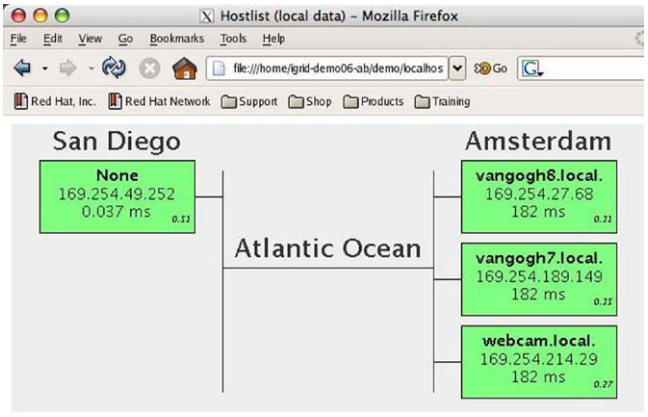

Fig. 2. Screenshot of host discovery using broadcast pings.

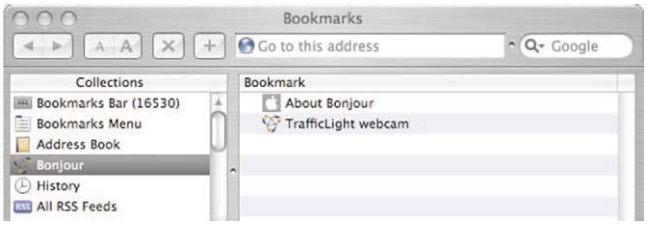

Fig. 3. Screenshot of DNS service discovery: the Safari web browser discovers a "`_http._tcp`" (web) service with the name "TrafficLight webcam".

Service discovery, thus finding out the hostname(s) of server applications, is an application-specific problem, rather than a host configuration problem. We demonstrated DNS-SD, which is an application-independent solution for service discovery, with off the shelf software. A screenshot is shown in Fig. 3. The use of DNS-SD for Grid applications was proposed by Maurizio Giordano and is implemented in the AccessGrid toolkit [18,19].

3.4. Results

3.4.1. Technical feasibility

We have shown that self-assigned link-local IPv4 addresses can be used in optical networks. We found that the protocols we picked, IPv4 link-local addresses, mDNS and DNS-SD, are technically sound. They indeed work for any link-local network, even though they were designed for home networks and ad hoc wireless networks.

In summary, we believe that this technology is sufficiently mature to be rolled out to a larger set of applications that use optical networks. Sections 3.4.2–3.4.6 list considerations which need to be taken into account for a roll-out.

3.4.2. Robustness of current implementations

The implementation of link-local IP addresses in both Windows and Mac OS X seems robust. In addition, Mac OS X has a solid implementation of mDNS and DNS-SD, and Apple is pushing this technology. Apple has ported its software to Windows, and that version also worked flawlessly in our experiment.

The implementation of link-local IP addressing as well as mDNS and DNS-SD on Linux is not as solid. There are multiple

implementations, and Fedora core installs one of these by default. In general we experienced more problems on Linux, which is unfortunate, since Linux is the default OS for most scientific applications in optical networking. Currently, it takes slightly more time to install and configure these technologies on Linux (e.g. ZCIP with mDNSResponder), than it is to statically configure IP addresses by hand.

Some of the bugs we encountered were:

- Bugs in both socket library hooks: libnss_mdns had a socket leak, and tmDNS did not operate well with the mDNSResponder implementation. Shortly after we reported the issue, the socket leak in libnss_mdns was fixed.
- Hosts sometimes did not respond to broadcast pings. Windows XP never replied to broadcast pings, even with the firewall off. This affected our demo, but is not an implementation bug.
- Name lookup sometimes failed for unknown reasons.[3]

3.4.3. Security issues

One of the assumptions of link-local networks is to trust the data given by multicast DNS servers. LLMNR does support DNSSEC, but only if there is a pre-established trust relation.

Authenticity of hosts can be verified on the application layer. We demonstrated that it is possible to check the identity of a host using GridFTP.

3.4.4. Network joins

IP address conflicts can happen just after two remote LANs are connected to each other. In practice these conflicts are so rare that we never experienced one during our demonstration. When we artificially forced a conflict, we observed that hosts were able to detect IP address conflicts, and responded by reconfiguring their IP address. However, we also found that the stale information is cached on some computers in the ARP cache for up to 5 min.

3.4.5. Service discovery

Determining the name of a particular service is an application-specific problem. The hostname problem with link-local is slightly more complex, because `.local` names are not advertised in the global DNS system, and can change in the case of conflicts.

DNS-SD can be used to discover the hostname(s) of a particular services, though this still does not tell the application if it is running on the local cluster or on the remote cluster (or on which remote cluster). Our host discovery solution using round trip times is obviously no reliable mechanism.

A slightly better solution is to use `.local` names including a component specifying the domain name of the cluster (e.g. `vangogh7.uva.netherlight.nl.local.` instead of `vangogh7.local.`).

A third option is to advertise the names of the hosts on the link-local connection in another way, for example on a trusted server in the remote organisation or using regular DNS. This is possible using DNS dynamic updates, and Dynamic DNS update leases [20,21].

[3] Details can be found at our website [5].

3.4.6. Multi-homing

A known problem with link-local IPv4 addresses is that it does not support multi-homed network devices. That is, network devices with multiple network interfaces, all using link-local addresses. In the network scenario we have sketched so far, network devices are not multi-homed, since there is only one interface using link-local addressing, possibly connecting to multiple remote clusters. There may be a second (management) interface, but that will use routable IP addresses.

Multicast DNS has the same multi-homing problem, even with IPv6 link-local addressing, which allows applications to explicitly specify the network interface.

3.4.7. Timing performance

Technically, both RFC 3927 and mDNS are expected to scale up to latencies of 1000 ms round-trip time, though mDNS may already experience additional overhead traffic after 350 ms. That is about the RTT for a lightpath spanning the globe.

According to the specification, the configuration process takes one round trip time. We did not perform extensive timing measurements in our demo, but our results seem to agree with the theory. The only time we experienced a delay longer then one second was when we forced an address conflict with a network join, as described in Section 3.4.4.

4. Conclusion

4.1. Applicability

If we look at the requirements posed in Section 1.1, we see that the technologies described are suitable for ad hoc optical networks connecting two or more clusters. The scalability of this technology is determined by the scalability of Ethernet. Optical networks are designed for connections from “few-to-few” hosts, and indeed this technology will be capable of handling a handful of geographically separated clusters. The IPv4 link-local protocol scales up to a few thousand hosts, before address conflicts become an issue. This is higher then what seems feasible using a single Ethernet network.

Most other features listed in Section 1.1 are also met. Two of them deserve some more explanation: first, this technology only configures IP address, but does not alter the routing table or support tuning of the protocol stack. Altering the routing table only need to be done once with the link-local IP addresses (since the link-local IP range does not change). Tuning of the protocol stack is not solved. In our opinion, this needs to be solved on the transport level, rather than on the network level. For example, using the improved TCP versions in recent Linux kernels.

4.2. Ethernet limitations

IP link-local addressing is built on top of an Ethernet network with a single logical link. This limits the applicability to networks without a router, such as a scenario with a limited number of clusters. In particular, Ethernet is very inefficient (unusable) for networks with loops, so a scenario with three

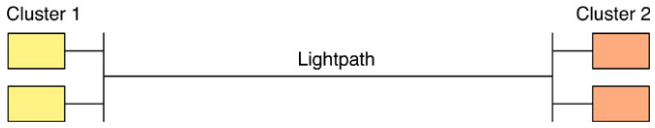

Fig. 4. A true lightpath: a layer 1 or layer connection without routers. It can use link-local IP addresses or a static (private) IP range.

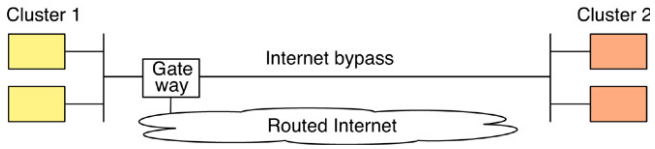

Fig. 5. A dedicated routed path. Though it contains one or more routers, the network connection is never shared with other traffic and has predictable and guaranteed properties. Also QoS paths through the routed Internet fall into this category.

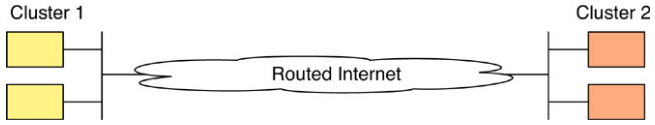

Fig. 6. A path through the routed Internet, giving a best-effort service.

clusters, interconnected with a triangle-shaped network, cannot be served using this technology.

If an optical private network gets too complex (i.e. consists of more then a few lightpaths), it is necessary to either use routers, or to physically split the network and use hosts with multiple "link-local" interfaces. Neither solution is supported by the zero configuration networking technologies as described in this article.

For complex networks, it may be better to look into routing protocols and address assignments in ad hoc networks [22,23].

4.3. iGrid networks

In our vision, each host in a simple "link-local" network as shown in Fig. 4 has two interfaces: one connected to the routed Internet for management purposes (not shown), and one on a dedicated LAN, connecting all nodes of the involved clusters together.

Conversely, each host in a "routed" network has one interface, on which every other host connected to the Internet could be reached.

Based on earlier scientific experiments involving optical networks, we assume that these experiments either use the routed Internet or use simple "link-local" networks. To test our assumption, we examined the network setup of all demonstrations held during iGrid 2005 [24]. Besides link-local networks (Fig. 4) and regular Internet (Fig. 6) we discovered another common network setup: the one shown in Fig. 5. In this network setup, the hosts only have one network interface, which is used for both management purposes (the routed Internet) as well as large data transfers on a dedicated path. This is done by connecting that network interface to a router (gateway), which redirects all traffic for a fixed set of IP addresses to a dedicated link. The advantage of this setup is that there is no need to change the routing table on the host. The network complexity was handled by the network engineers of iGrid 2005, instead of the applications.

Table 2
Network setup per iGrid demonstration

Network set-up	Number of demos
True lightpath (Fig. 4)	6 (12%)
Dedicated path (Fig. 5)	16 (33%)
Routed Internet (Fig. 6)	14 (29%)
Unknown	12 (24%)
Other	1 (2%)

Table 2 shows the percentage of demonstrations that use a particular type of network. For twelve of the 49 demonstrations, we could not retrieve information about their network setup, for example because none of their lightpaths terminated at the iGrid show floor, or the consulted network engineers were not involved with a particular demonstration. One demonstration that used IP multicast is listed as "Other".

The number of demonstrators that used "true" lightpaths was lower than we expected. According to private conversations with the network engineers, this was mostly an engineering decision, which limited the number of network changes between time slots. If network changes occur more often (e.g. because a reliable controlplane becomes available that supports a complex network like iGrid [25]), true lightpaths will be used more frequently. A driving factor could be that moving from dedicated paths to true lightpaths would save ports on relatively expensive routers, at the cost of using more ports on relatively cheaper layer 2 switches.

4.4. Future research

Standardization work in this area has led to stable protocols. Most work in this area seems finished. Also, service discovery in scientific applications like Grid applications has been studied and implemented [18,19].

An issue that deserves more attention is debugging world-wide Ethernet networks. Unlike SONET/SDH, which carries a lot of monitoring information, Ethernet connections are extremely hard to debug. To illustrate, it took a skilled network engineer considerable time to debug our San Diego–Amsterdam network connection, even after the physical path was decided and devices were configured. It turned out that one interface on a switch was overlooked. The only debugging mechanism was looking in the ARP table of switches along the path. After these problems were supposedly fixed, it took us, the demonstrators, more time to figure out that we didn't get the desired result, simply because one host refused to respond to broadcast pings.

So in short, automatic configuration of end hosts may indeed save valuable time for network and system administrators. However, the gain is hardly significant as long as there are no automated tool for setting up lightpaths across multiple administrative domains, or for fault detection.

Acknowledgments

The authors would like to thank the GigaPort project and TNO for funding this research, and the network providers and hosts at iGrid for making the demonstration possible. The idea for using link-local IP addresses came from a discussion with Venkat Vishwanath and Eric He during an OptIPuter meeting in January 2005. A very special thanks goes to Stuart Cheshire for taking the time to visit our lab in Amsterdam, fighting a cold due to the Dutch weather. Last, thanks to Pieter de Boer and Paola Grosso for helping us classify the iGrid demonstrations and to Karst Koymans for proof reading this document.

References

[1] J.H. Saltzer, D.P. Reed, D.D. Clark, End-To-End arguments in system design, ACM Transactions on Computer Systems 2 (4) (1984) 277–288.
[2] R. Bush, D. Meyer, Some Internet Architectural Guidelines and Philosophy, RFC 3439, December 2002.
[3] Charter of (now concluded) IETF ZeroConf working group. http://www.ietf.org/html.charters/OLD/zeroconf-charter.html, January 2004.
[4] E. Guttman, Autoconfiguration for IP Networking: Enabling Local Communication, IEEE Internet Computing 5 (3) (2001) 81–86.
[5] F. Dijkstra, Zero configuration technologies and software, technical documents. http://www.science.uva.nl/research/sne/wiki/CategoryZeroconf, October 2005.
[6] S. Cheshire, B. Aboba, E. Guttman, Dynamic Configuration of IPv4 Link-Local Addresses, RFC 3927, May 2005.
[7] S. Thomson, T. Narten, IPv6 Stateless Address Autoconfiguration, RFC 2462, December 1998.
[8] S. Cheshire, M. Krochmal, Multicast DNS, draft-cheshire-dnsext-multicastdns, June 2005. Work in progress.
[9] B. Aboba, L. Esibov, D. Thaler, Linklocal Multicast Name Resolution (LLMNR), October 2005. Work in progress.
[10] E. Guttman, C. Perkins, J. Veizades, M. Day, Service Location Protocol, Version 2, RFC 2608, June 1999.
[11] S. Cheshire, M. Krochmal, DNS-Based Service Discovery, draft-cheshire-dnsext-dns-sd, June 2005. Work in Progress.
[12] Y.Y. Goland, T. Cai, P. Leach, Y. Gu, S. Albright, Simple Service Discovery Protocol (SSDP), draft-cai-ssdp-v1, October 1999.
[13] A. Gulbrandsen, P. Vixie, L. Esibov, A DNS RR for specifying the location of services (DNS SRV), RFC 2782, February 2000.
[14] Apple Darwin kernel, http://developer.apple.com/darwin/.
[15] ZCIP (Zero-Conf IP) and tmDNS (tiny mDNS), available at http://zeroconf.sourceforge.net/.
[16] Apple Bonjour, http://developer.apple.com/opensource/internet/bonjour.html.
[17] Porchdog Howl, http://www.porchdogsoft.com/products/howl/.
[18] M. Giordano, DNS-Based discovery system in service oriented programming, in: Advances in Grid Computing—EGC 2005, 2005, pp. 840–850.
[19] AccessGrid, Python interface to mDNSResponder. http://www-unix.mcs.anl.gov/fl/research/accessgrid/bonjour-py/.
[20] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, Dynamic Updates in the Domain Name System (DNS UPDATE), RFC 2136, April 1997.
[21] K. Sekar, S. Cheshire, M. Krochmal, Dynamic DNS Update Leases, draft-sekar-dns-ul, June 2005. Work in progress.
[22] M. Thoppian, R. Prakash, A distributed protocol for dynamic address assignment in mobile ad hoc networks, IEEE Transactions on Mobile Computing 5 (1) (2006).
[23] K. Weniger, M. Zitterbart, Address autoconfiguration in mobile ad hoc networks: Current approaches and future directions, IEEE Network 18 (4) (2004).
[24] iGrid 2005, http://www.igrid2005.org/.
[25] P. Grosso, P. de Boer, L. Winkler, The network infrastructure at iGrid2005: lambda networking in action, Future Generation Computer Systems (in this issue), doi:10.1016/j.future.2006.03.013.

Freek Dijkstra received his M.Sc. in applied physics from the Universiteit Utrecht in 2002. He is researcher in System and Network Engineering group at the Universiteit van Amsterdam, where he pursues a Ph.D. degree. Freek's primary interest is Optical Networks, where he researches the multi-domain aspects.

Jeroen van der Ham received the M.Sc. degree in cognitive artificial intelligence from the Universiteit Utrecht in 2003, and the M.Sc. degree in system and network engineering from the Universiteit van Amsterdam in 2005. In 2005, he joined the System and Network Engineering group at the Universiteit van Amsterdam, where he is currently pursuing his Ph.D. degree. His primary research interests are the network control plane and its multi-domain aspects.

Cees de Laat is associate professor in the Informatics Institute at the Universiteit van Amsterdam. Current research includes hybrid networking, lambda switching and provisioning, policy-based networking and Authorization, Authentication and Accounting architecture. With SURFnet he implements projects in the GigaPort Research on Networks area. He serves as Grid Forum Steering Group (GFSG) Infrastructure Area Director and IETF Liaison. He is co-founder of the Global Lambda Integrated Facility (GLIF).