Crossroads
The ACM Student Magazine

Zero Configuration Networking

by ***David Stirling*** and ***Firas Al-Ali***

Introduction

The number of computers in use has increased dramatically in the last decade. Computers are now a crucial element in how we are entertained, educated, and how we do business. A natural result of this growth is a resultant increase in the desire and need for computers to communicate with each other. Network hosts and network infrastructures have historically been difficult to configure, requiring network services and relying on highly-trained network administrators, engineers, and consultants [**6**]. There are now situations in which administration and configuration is impractical or impossible, such as in the home or small office, embedded systems, and in impromptu networks. Emerging auto-configuration networking protocols promise to enable hosts to establish IP networks without prior configuration or network services. Even very simple devices with few computing resources will be able to communicate via standard protocols wherever they are attached. These efforts aim to make this form of networking both simple and inexpensive. Before we dive into the details of these proposed protocols, let's look first at past efforts in this area.

Previous Auto-configuration Efforts

For some time, network software and equipment vendors have been developing

proprietary protocol suites that stressed ease of use and deployment. The success of Apple's AppleTalk, Novell's IPX (Internetwork Packet eXchange), and Microsoft's NetBIOS/SMB (Network Basic Input Output System / Server Message Blocks), and especially NetBEUI (NetBios Enhanced User Interface), arose from their automatic address configuration, decentralized service discovery, and naming functions which facilitated local communication and sharing of resources such as files and printers [**9**]. These proprietary protocols continue to be used widely only because of their ease of configuration features. Adopting open protocol standards will let us retire proprietary networking - a move that has broad support. Even network equipment vendors uniformly accept that proprietary network protocols have seen their day and should be replaced by IP standards [**6**].

Rendezvous and Zeroconf

In the latest release of its operating system (Mac OS X 10.2), Apple Computer introduced an open set of protocols for zero-configuration networking over IP called **Rendezvous**. Rendezvous is essentially Apple's trade name for the protocols developed by the Zero Configuration Networking (Zeroconf) Working Group, which was chartered in 1999 by the Internet Engineering Task Force (IETF) [**7**, **2**].The goal is to enable networking in the absence of configuration and administration personnel. Zeroconf aims at making networking as easy as possible. In some cases, other considerations may dominate ease of use. For example, network security requires some configuration, which may not be as easy as the unacceptable alternative of no security. Existing network-based applications will work without modification over enhanced network service and application layers using standard interfaces. Indeed, users should not even be aware that the network service layer has been configured automatically [**8**].

The Zeroconf Working Group's requirements and proposed solutions cover three areas:

- Address auto-configuration (allocating IP addresses to hosts)
- Name-to-address translation (translating a human-understandable computer host name to its corresponding IP address, and vice versa)
- Service discovery (finding and using services on the network)

By having zero configuration solutions to these three problems, IP networking retains all its advantages over other networking standards while breaking new ground in ease of use [**8**]. For home networks and other small networks unlikely to have a system

administrator, users no longer have to assign IP addresses, assign host names, or even type in addresses to access services on the network. Ultimately, users should be able to just see what network services are available and use them. Each one of the above three areas is discussed in more detail below.

Address Auto-Configuration

The addressing problem has already been solved with self-assigned link-local addressing. This is an evolution from the currently popular and widely used Microsoft DHCP (Dynamic Host Configuration Protocol) in which a network administrator still needs to configure a network server to run the DHCP server service, but the clients get a DHCP-assigned IP address automatically; zero end-user effort! In the case of self-assigned link-local addressing, the need for a DHCP server is totally eliminated. Two hosts are considered to be on the same link if, when one host sends packets to the other, the entire link-layer payload (the content of the packet as represented in the physical network, such as Ethernet) arrives unmodified. In practice, on an Ethernet network this means that no IP router touches the packet between the two hosts. Self-assigned link-local addressing works by having hosts pick a pseudo-random IP address in the link-local range. If another device is already using the address, the host picks another address. Self-assigned link-local addressing has already shipped on IPv4 starting with Windows98 and Mac OS 8.5 [**1**].

Name-to-Address Translation

The proposed solution to host naming on a small isolated network is Multicast DNS (Domain Name Service/System), referring to DNS-format packets sent using IP Multicast. Users are free to pick a "local host name" for their computers, which is not administered by any global authority like conventional DNS host names, and the user does not have to pay anybody for it – it only has significance on that small isolated network. Other users on the network can use the local host name anywhere that they could normally use a conventional DNS host name [**4**].

Service Discovery

The final element of Zeroconf is service discovery. Service discovery allows applications to find all available instances of a particular type of service, and to resolve a named instance of a service to an IP address and port number. The Rendezvous solution to this problem uses Multicast DNS to store service information in DNS resource records

for services on the local network link. Apple has said that a future release of Mac OS X's Rendezvous may support storing these records in a conventional DNS server using Dynamic DNS Update [**2**].

Discussing the issue of service discovery would not be complete without complementing it with coverage of how zero configuration networking is adopting a service-centric approach rather than a hardware-centric one when it comes to rendering the network services to the end users and clients - this is the topic of the next section.

Service Orientation

For someone who deals with network servers, network devices, and network programming, it is easy to make the mistake of thinking about services in terms of physical hardware. In the device-centric view, the network consists of a number of devices or hosts, each with a set of services. For example, the network might consist of a server machine and a client machine. In a device-centric browsing scheme, the client would query the server for what services it is running, get back a list (FTP, HTTP, network browsing, list of printers, etc.), and decide which service to use. In some sense, this mimics the actual layout of the network, in that each of these services is probably running on the same physical piece of hardware. The interface, thus, mimics the system.

From the user's standpoint, the device-centric view is not very useful. Users want to accomplish certain tasks, not query devices for what services they are running. It makes far more sense for a client to ask, "What printing services are available?" than to repeatedly query each available device with the question, "What services are you running?" and sift through the results. Additionally, services are not always tied to specific IP addresses or even host names. Often, a website is hosted by multiple servers with different addresses - in these cases, in a sense, the hostname represents a "service." Within an organization, network administrators may need to move a service from one server to another to help balance the load. If clients store the host name (as in most cases they now do), they will not be able to connect if the service changes hosts.

Zeroconf takes the service-oriented view. Queries are made according to the type of service needed, not the hosts carrying them. Applications store service names, not addresses, so even if the port number, IP address, or even host name has changed,

the application can still connect. By concentrating on services rather than devices, the user's browsing experience is made more useful and trouble-free.

Although the following two examples are not necessarily Zeroconf-compliant technologies, they still strongly represent the current shift towards service orientation rather than device orientation:

Server Clusters

A server cluster is normally a collection of separate actual physically-independent servers (server farm) grouped into one logical server entity with one host name, called the cluster name. Network users connect to this cluster name and use the services they require, which are offered by the cluster as a whole. If one server within the cluster goes down (fails), the services it runs are automatically ported over (failed over) to another server within the cluster where they start running again without having the user notice any noticeable service interruption (since the cluster name remains the same). Clusters nowadays are widely deployed in businesses and R&D facilities around the world, where issues like high availability and load balancing are of paramount importance. Examples of different clustering technologies include Linux's Beowulf, Novell's NetWare Cluster Services (NWCS), Microsoft's Cluster Server, NSI's Balance Suite, Tandem's ServerNet, HP's MC/ServiceGuard, etc.

Novell's Distributed Print Services (NDPS)

This printing architecture was designed by the partnership of Novell, HP, and Xerox. Although NDPS still requires a network administrator to centrally manage network printers, it is designed to handle the increased complexity of managing print devices in any type of network environment, ranging in size from small workgroups to enterprise-wide systems, by having the NDPS automatically configure a network printer newly plugged into the network. Another benefit from NDPS is that it facilitates network printer usage for end users by designating those network printers for automatic installation on user workstations without any action by the users [**11**].

Network Traffic

Server-free addressing, naming, and service discovery have the potential to create a significant amount of excess network traffic (sometimes called 'chattiness') [**2**]. Zeroconf takes a number of steps to reduce this traffic to a minimum including caching, suppression of duplicate responses, exponential back-off, and service announcement.

The multicast DNS responder maintains a cache of multicast packets to prevent requesting information that has already been requested. For example, when one host requests a list of local print spoolers, the list of printers comes back multicasted, so all local hosts can see it. The next time a host needs a list of print spoolers, it already has the list in its cache and does not need to reissue the query.

In order to prevent repeated answers to the same query, service queries include a list of known answers. For example, if a host is browsing for printers, the first query includes no currently known print services, and the query gets fifty available print services, the next time the host queries for print services, the query includes the fifty known print services. Only those hosts that provide print services but that are not listed in the query respond. If a host that is about to respond notices that another host has already responded with the same information, the first host suppresses its response.

When a host is browsing for services, it does not continually pound on the network to see if new services are available. The host issues an initial query, and subsequent queries are sent exponentially less often - after one second, two seconds, four seconds, eight seconds, and so on, up to a maximum delay of 4096 seconds (a little over an hour) [**3**]. This does not mean that it takes over an hour for a service browser to see new services. When a service starts up on the network, it announces its presence with the same exponential back-off delay. This way, the background noise is kept to a minimum, but new services are seen very quickly.

Network Services Architecture

The network services architecture in Zeroconf enables an easy-to-use mechanism for publishing, discovering, and using IP-based services. The architecture supports three fundamental operations, each of which is a necessary part of zero-configuration network services:

- Publication (advertising a service)
- Discovery (browsing for available services)
- Resolution (translating service names to addresses and port numbers for use) [**7**]

To publish a service, an application or device must register the service with a multicast DNS responder, either through a high-level API or by communicating directly with the

responder. When a service is registered, two related DNS records are created: a service (SRV) record, and a pointer (PTR) record. In some cases, a third record, called a text (TXT) record, is created, containing additional data needed to resolve or use the service.

The service record maps the name of the service instance to the information needed by a client to actually use the service. Clients then store the service name as a persistent way to access the service, and perform a DNS query for the host name and port number when it's time to connect. This additional level of indirection enables two important features. First, the service is identified by a human-readable name instead of a domain name and port number. Second, clients can access the service even if its port number, IP address, or host name changes, as long as the service name remains the same.

The service record contains four pieces of information to identify a service: the host name, port, priority, and weight. The host name is the domain name where the service can currently be found. The reason a host name is given instead of a single IP address is that it could be a multi-homed host with more than one IP address, or it could have IPv6 addresses as well as IPv4 addresses. Identifying the host by name allows all these cases to be handled gracefully [**3**]. The port number identifies the UDP or TCP port for the service. The priority and weight fields are not currently used by most services [**2**].

Pointer records enable service discovery by mapping the type of the service to a list of names of specific instances of that type of service. This record adds yet another layer of indirection so that services can be found just by looking up PTR records labelled with the service type. The record contains just one piece of information, the name of the service (which is the same as the name of the SRV record).

The optional text record has the same name as the corresponding SRV record, and can contain a small amount of additional information about the service instance, typically no more than 100-200 bytes at most. For example, a multiplayer network game could advertise the name of the player, and a chat program could advertise the status of the user [**2**].

Service discovery makes use of the DNS records registered during service publication to find all named instances of a particular type of service. To do this, an application performs a query for PTR records matching a service type, usually through a higher-

level API. The multicast DNS responders running on each device return PTR records with service instance names.

Service discovery typically takes place only once in a while - for example, when a user first selects a printer. This operation saves the service instance name, the intended stable identifier for any given instance of a service. Port numbers, IP addresses, and even host names can change from day to day, but a user should not need to re-select a printer every time this happens. Accordingly, resolution from a service name to socket information does not happen until the service is actually used. To resolve a service, an application performs a DNS look-up for a SRV record with the name of the service. The multicast DNS responder responds with the SRV record containing the current information [**3**].

It is worthwhile to see where Zeroconf is positioned among other service discovery protocols, such as Salutation, SLP, Jini, and UPnP. Salutation, like Zeroconf, is an OS- and hardware-independent architecture with implementations that are already on the market, including developer tool kits, MFPs, fax devices, and Windows platform enablers. The SLP (Service Location Protocol) applies existing Internet standards to the service discovery problem. SLP relies on the use of a User Agent (UA) software to browse for a service, a Service Agent (SA) software to locate the service, and a Directory Agent(DA) software that acts as a centralized repository for the service location information. Sun's Jini is a more proprietary service discovery protocol for networks of Java-enabled devices. Microsoft's approach to the service discovery problem is the Universal Plug and Play (UPnP) protocol, which is more of an IP-based discovery protocol.

Security Considerations

The use of IPv4Link-Local Addresses may open a network host to new attacks. In particular, a host that previously did not have an IP address, and no IP stack running, was not susceptible to IP-based attacks. By configuring a working address, the host may now be vulnerable to IP-based attacks.

The Address Resolution Protocol (ARP) protocol is insecure. A malicious host may send fraudulent ARP packets on the network, interfering with the correct operation of other hosts. For example, it is easy for a host to answer all ARP requests with replies giving its own physical address, thereby claiming ownership of every address on the network [**3**].

Implementers of Zeroconf are being advised that the Internet Protocol architecture expects every networked device or host must implement security that is adequate to protect the resources to which the device or host has access, including the network itself, against known or credible threats. Even though use of link-local addresses may reduce the number of threats to which a device is exposed, implementers of devices supporting the Internet Protocol must not assume that a customer's local network is free from security risks [**7**].

While there maybe particular kinds of devices or environments for which the security provided by the network is adequate to protect the resources that are accessible by the device, it would be misleading to make a general statement to the effect that the requirement to provide security is reduced for devices using link-local addresses as a sole means of access. In all cases, whether or not link-local addresses are used, it is necessary for implementers of devices supporting the Internet Protocol to analyze the known and credible threats to which a specific host or device might be subjected, and to the extent that it is feasible, to provide security mechanisms which ameliorate or reduce the risks associated with such threats [**3**].

The Future

The Zeroconf Workgroup has stated that it is important to understand that the purpose of Zeroconf is not *solely* to make current personal computer networking easier to use, though this is certainly a useful benefit. The long-term goal of Zeroconf is to enable the creation of entirely new kinds of networked products, products that today would simply not be commercially viable because of the inconvenience and support costs involved in setting up, configuring, and maintaining a network to allow them to operate [**7**].

This goal may be realized sooner than originally-imagined. In September 2002, consumer electronics company Philips Electronics announced that it was signing up to use Rendezvous. Philips will begin shipping Zeroconf-compliant devices in 2003. Its CEO, Gerard Kleisterlee, is enthused about a near future when digital music or pictures stored on a home computer could be seamlessly delivered through a Philips stereo system or television [**5**].

In June 2002, Apple CEO Steve Jobs played out such a scenario on stage. Jobs showed two computers instantly finding each other across a wireless IEEE 802.11 "AirPort" network - one picking up MP3 playlists and files from Apple's iTunes music software on

the other's hard drive. The demonstration would have been much more interesting had the box on one end been not another computer but rather an inexpensive stereo receiver [**5**, **10**].

Whether the Zeroconf protocols will be adopted by the masses remains to be seen. The technology has the potential to herald the development of simple interoperable IP-enabled devices and greatly increase LAN stability and usability. We are excited about the prospect of entirely new kinds of networked products - products that could change the way we are entertained, educated and do business.

References

1

Aboba, B., et al., "Dynamic Configuration of IPv4 Link-Local Addresses", 23 August 2002, <**http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-07.txt**>, (8 October 2002).

2

Apple Computer Inc., *Rendezvous Developer Home Page*, 2002, <**http://developer.apple.com/macosx/rendezvous**>, (6 October 2002).

3

Brown, R., "Home Entertaining", *New Zealand Listener*, 28 September 2002, pp.38.

4

Cheshire, S., "DNS-Based Service Discovery", 20 December 2002, <**http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt**>, (14 January 2003).

5

Cheshire S., "Performing DNS queries via IP Multicast", 13 July 2001, <**http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt**> (7 October 2002).

6

Guttman, E., "Autoconfiguration for IP Networking", June 2001, <**http://www.Zeroconf.org/w3onwire-Zeroconf.pdf**>, (10 October 2002).

7

Internet Engineering Task Force, *Zero Configuration Networking (Zeroconf) Home Page.* <**http://www.zeroconf.org**> (6 October 2002).

8

Internet Engineering Task Force, "Zeroconf Charter", <**http://www.ietf.org/html.charters/Zeroconf-charter.html**> (6 October 2002).

9

Syngress Media, *CNE NetWare 5 Study Guide*, 1999, Osborne/McGraw-Hill U.S. A., ISBN 0-07-211923-3.

10

Tanenbaum, A., *Computer Networks*, 1996, Third Edition, Prentice Hall Inc.

11

White, J., "Rendezvous: It's Like a Backstage Pass to the Future", *The Idea Basket*, 5 July 2002, <**http://www.theideabasket.com/index.php/article/articleview/31/1/6/**> (12 October 2002).

Biographies

David Stirling (**djstirling@softhome.net**) is a 3rd year Bachelor of Information Sciences student at Massey University, Palmerston North, New Zealand. He is one of the students in the class for "Architecture & Networks," taught by Firas Al-Ali.

Firas Al-Ali (**f.m.al-ali@massey.ac.nz**) is a PhD Candidate and Assistant Lecturer with the Institute of Information Sciences & Technology, Massey University, Palmerston North, New Zealand. He currently lectures on the subjects of computer networks, architecture, and supercomputing.