Zeroconf Networking

Brad Hards

Table of Contents

Abstract

Zeroconf networking is an emerging field of work from the Internet Engineering Task Force, providing IP level networking without needing a network administrator. It provides methods to assign an IP address to a host, discover network services by name or characteristics and to translate between names and addresses.

The starting point for Zeroconf networking is to take two hosts with ethernet ports, and a cross-over cable, and have them communicate without needing to manually configure network addresses (or have one of them run an DHCP server). This is currently possible with legacy protocols such as Appletalk or NETBIOS, but Zeroconf networking provides a standardised, inter-operable way to make easy networking available for any host. This is being adopted by Apple for OS X, under the marketing name of Rendezvous.

While the original scope is simple connections using ethernet cross-over cables or USB host-to-host devices, Zeroconf can be used for new applications that are simply not viable without such technology. For example, it becomes possible to move ethernet connected printers into the mass-consumer market, where the issues associated with configuring an IP address would normally have incurred excessive support costs for retailers.

This paper includes discussion of the zcip tool, which uses standard Linux interfaces to establish IP addresses for a group of hosts that are on the same network ("link local", traffic with Zeroconf source or destination addresses is never routed), which is available on kernel.org (and mirrors). This tool is (mostly) inter-operable with current Windows and Mac OS implementations, and aims at being a reference implementation for the Zeroconf IP assignment.

In addition to address assignment, options for "service discovery" are reviewed. These include using Service Location Protocol (SLPv2, currently supported by the OpenSLP tools) and use of existing DNS functions and record types.

This paper also covers DNS extensions to use multicast to identify the IP address for a particular host. This is a relatively simple idea, with two implementation options currently under consideration. There will also be a brief discussion of multicast group address assignment.

User Stories

User Stories are a Extreme Programming (XP) idea, which serve as a way to describe requirements. A *story* is a feature that a customer wants in their software, in terms relevant to that customer, a story that they would like to be able to tell their friends about this great system that they are using.[1].

To help understand what networking might be able to do in a Zeroconf world, this section provides a couple of stories of where we need to get to. These stories are phrased a little differently to normal XP story cards to make them easier to understand.

Exchange files

Two academics at a conference wish to exchange recent research papers. One of them has a USB host-to-host cable, and both have laptops with USB ports. They plug in each end of the cable to a port, and a suitable IP network is established over the cable, including compatible IP addresses and a suitable route. Each machine also pops up a file exchange GUI, and each user can see those files that the other has chosen to share. Copying files is simple a drag and drop activity.

Plug and Play Printing

A new user buys a printer and plugs it into the network. Each machine on the network adds the new printer to the list of available printers, and when the user selects the print function on any application, that printer is available for use.

What do we mean by Zeroconf Networking?

A zeroconf network is one that can exist without a central control element, and works without any kind of user configuration.

> draft-ietf-zeroconf-reqts-12.txt:
>
> A zeroconf protocol is able to operate correctly in the absence of configured information from either a user or infrastructure services such as conventional DHCP or DNS servers. Zeroconf protocols may use configured information, when it is available, but do not rely on it being present. For example, the use of MAC addresses (i.e. layer two addresses) as parameters in zeroconf protocols is generally acceptable because they are globally unique and readily available on most devices of interest.

A zeroconf network may require four functional elements:

- Network layer address assignment
- Translation between network names, and network addresses
- Location or discovery of services by name and protocol
- Multicast address assignment

The IETF has existing or developmental work in the first three element areas. There is a general acceptance that the need for multicast address assignment is very low, and while there is an equivalent to DHCP for multicast (known as MADCAP, for Multicast Address Dynamic Client Allocation Protocol[2]), its usage is very low. Work on multicast address assignment in a zeroconf environment has effectivelybeen abandoned.

Technology - Part 1 - Address Assignment

It is very rare to need to configure the physical layer (beyond plugging it in) or the link layer. The first area that usually needs configuration is the the network layer, for address assignment. This section discusses technical solutions for automatic configuration of the network layer - in particular, how to obtain a network address in both IPv4 and IPV6.

IP Address Autoconfiguration - IPv6

IPv6 actually has two different ways to automatically obtain a network address. One of these is known as *stateful* autoconfiguration[3], and roughly corresponds to DHCP, where you need a suitably configured server that will provide a unique address. The second type of autoconfiguration is called *stateless*, and is an effective zeroconf approach.

IPv6 stateless autoconfiguration[4] takes advantage of the size of the IPv6 address space and the uniqueness of link layer addresses such as Ethernet MAC addresses.

Each interface generates an *link-local* address, which is never routed off the local link. This is done by taking `FE80` and appending the link layer address (typically, the ethernet MAC address), with zeros filling the remaining bytes between the link local identifier, and the link layer address. In IPv6 notation, this is `FE80::(mac address)`, where the `::` notation indicates that enough zeros are inserted to pad the resulting address out to the required 128 bits.

That address is then configured onto the network interface. At this stage, the address is considered *tentative*, and the host joins appropriate multicast groups.

One of the multicast groups (solicited-node) is then used to check that the tentative address is unique, by using a Neighbor Solicitation message, which is part of ICMPv6[5]. If a node on the link is already using the desired address (that is duplicated addresses are detected), then the host that is already using the address replies with a Neighbor Advertisement, and the tentative address is abandoned, and autoconfiguration stops on that host.

If there is no answer to the Neighbor Solicitation message, then the address is unique, and the interface transitions from tentative to *preferred*.

The host then sends a Router Solicitation to the all-routers multicast address. Each router will respond with a Router Advertisement message, and for each response with the autonomous bit set, an address is generated consisting of the prefix provided in the Router Advertisement, and the interface hardware address. These addresses are assigned to the appropriate interface, and are then available for use.

IP Address Autoconfiguration - IPv4

While IPv6 can use its address space to encompass the full range of hardware addresses, this is clearly not possible with IPv4. Instead of having a one-to-many mapping between hardware address and IP address, there is a many-to-one mapping between hardware address and IP address. That is, lots of hosts are selecting from the same limited pool of addresses. While the autoconfiguration approach in IPv4 is broadly similar to that of IPv6, the limited address space means that some changes are required.

IPv4 addresses are chosen from a pool of 65024 possible IP addresses in the range of 169.254.1.0 to 169.254.254.255 (that is 169.254/16, with the top 256 and bottom 256 addresses reserved for future purposes).

The algorithm to chose an address is relatively simple. The host seeds a random number generator with the hardware address (MAC address) of the interface, and randomly chooses an address in the required range.

The host then does an *ARP probe* for the address, and if there are any responses, then the host chooses another IP address at random, and tries the ARP probe again.

If there are no answers, then no-one is currently using the address on the local link, and the host is free to assign the selected address.

The host then does a couple of gratuitous ARPs to flush any ARP caches which may have old data, and can then use the address for further networking. The host must continue to monitor network traffic, in order to respond to any ARP probes that are for the address that the host is currently using.

It is worth noting that a "capable" host, such as a PC or workstation is expected to have at least two IP addresses - a routable address[6], and a zeroconf link-local address from the 169.254/16 space. This allows communication with device that only has a link-local address (such as a zeroconf printer), while also allowing communication with the rest of the internet, assuming such a connection is available.

Technology - Part 2 - Name to Address Translation

Having obtained a suitable network layer (IP layer) address, the transport layer (TCP or UDP, and now SCTP) normally does not need additional configuration. This leaves configuration at the application layer. This section and the next deal with key issues at the application layer.

While using IP addresses is obviously possible, most users have at least some need for use of abstract names. This is normally managed by the Domain Name System (DNS). However setting up DNS can never be considered zeroconf, since it requires both substantial server side work, and configuring of the resolver setup.

The zeroconf equivalent uses a little version of a DNS resolver that responds to DNS type queries received on a known multicast address (224.0.0.251), which provides a standardised way to resolve names to addresses, and addresses back to names.

There are two different proposals for how name to address translation might work:

- Link Local Mulicast Name Resolution (LLMNR), pushed by people from Microsoft[7].
- Multicast DNS (mDNS), pushed by people from Apple[8].

The general focus of both drafts is to use normal DNS Resource Records, transmitted over multicast to a known port (5353), with a multicast DNS responder running on each host on the network. Because there is no central authority for names, it is possible that there can be naming collisions, where two or more hosts have the same name. Both drafts attempt to avoid this occurring (with a similar claim and defend process to that used for IP addresses), but allow for duplicate responses, and make dealing with this an application layer responsibility.

Technology - Part 3 - Service Location or Discovery

The goal of a networking system is to use services that are shared on the network. This might include printers, scanners, backups, mail servers, web services, file servers, and instant messaging. Normal IP networking practice is that you need to know the name (or IP address) of a service in order to use it. Automatic service location (or service discovery) is an important tool to ensuring that shared network resources can be found and used without requiring a significant amount of configuration work.

It is worth noting that service location is not a new idea - many systems already have some form of service location. However many of them suffer from serious design flaws, including excessive use of network bandwidth, and using broadcast (which means that service location cannot be scaled to larger networks).

Service Location Protocol

Service Location Protocol is an IETF protocol that provides automatic client configuration for applications and advertisement for network services. Activity continues, although the official IETF working group has concluded.

> RFC 2608:
>
> The Service Location Protocol provides a scalable framework for the discovery and selection of network services. Using this protocol, computers using the Internet need little or no static configuration of network services for network based applications. This is especially important as computers become more portable, and users less tolerant or able to fulfill the demands of network system administration.

Service Location Protocol uses three elements:

- a User Agent, which is essentially the client that is registering services or locating services.
- a Service Agent, which is essentially the daemon that holds registrations, and answers queries for service location.
- a Directory Agent, which acts as an intermediary, to provide improved scaling.

Service Location Protocol provides capabilities for applications to register and deregister services, and to locate services by type, by host, or by attribute. The ability to do service location by attribute (with server side filtering based on LDAPv3 style filtering strings) is key to the scalability of Service Location Protocol from zeroconf up to enterprise size networks.

DNS Service Discovery

DNS Service Discovery is a way of using existing DNS Resource Records to locate services. Since a typical zeroconf client will already have a multicast DNS responder (to provide the name to address translation), this type of service location design may not require much in the way of additional resources.

> draft-cheshire-dnsext-dns-sd.txt:
>
> Given a type of service that a client is looking for, and a domain in which the client is looking for that service, this convention allows clients to discover a list of named instances of a that desired service, using only standard DNS queries. In short, this is referred to as DNS-based Service Discovery, or DNS-SD.

DNS Service Discovery uses a combination of PTR, SRV and TXT records to locate services and their attributes.

PTR records are normally associated with reverse lookups (that is, given an IP address, PTR allows you to determine the name associated with that address). However PTR is really a generic mapping from one part of the DNS name space to one or more other parts of the DNS name space. PTR records are used to get from a generic class of services to a specific host.

SRV records are used to determine the port number that is to be used and relative weighting of each service

TXT records are used to convey attribute information

Consider the following example, loosely based on one provided in the Internet Draft:

To determine a list of FTP servers available on a network, you would issue a PTR query for `_ftp._tcp.example.com.`. This would likely return several Answers, such as `fileserver1.example.com.` and `Big Cool Guy’s oggs.example.com`

To figure out to connect to the server, you might do an ANY query on `Big Cool Guy’s oggs.example.com`, which might produce a SRV record that tells you to use port 2121, and to contact `pc323.example.com`. You may also get a TXT record, which would contain something like `path=/pub/ms/oggs`. So to connect, you need to contact pc323.example.com on port 2121, and change to directory /pub/ms/oggs.

If necessary, you would then do a normal DNS A query to determine the IP address of `pc323.example.com`.

While this is relatively simple, DNS Service Discovery does not have the deployment experience of Service Location Protocol. DNS Service Discovery also lacks some features that may make deployment in a larger situation difficult - in particular, it only supports client side filtering, so to find all the printers that can provide A2 size printing, you need to obtain the attributes (as TXT records) for every printer, and locally remove all those that cannot print on A2. As another limitation, the size of the attributes is limited by the TXT records.

Zeroconf in the home

While most of the focus on zeroconf network assumes an ad-hoc networking environment, zeroconf is a possible option for home networks, because it avoids the need for network management, which is likely to enable far more home users to use networking.

The home network is probably the principle case where the need for capable hosts to have multiple addresses configured on a single interface occurs. This allows the host to use an ADSL or ISDN gateway (which may offer DHCP services) to communicate with the wider internet, while not requiring a DHCP server to be present and active in order to print to a local printer (which is assumed to have a zeroconf IP address) or to play a networked game with a second, local, capable host.

Zeroconf in a managed network

While a managed network (in this context, meaning a larger network with a trained network administrator, albeit perhaps part time) would normally use DHCP to provide IP addresses, and DNS to provide name to address translation, it is possible that some of the features of a zeroconf network can still be effectively employed in managed network.

This is particular evident when considering peering protocols. For example, the Apple iChat system is effectively a local network peer-to-peer instant messenger[9]. The iChat client uses DNS Service Discovery (as discussed previously) to determine which other users are currently running iChat, and presents this as a list. This avoids the need to set up and run a Jabber server, for example.

Similarly, the KDE VNC application (KRFB) can use Service Location Protocol to provide a list of hosts currently providing RFB protocol[10] services. This avoids the need to determine and hand enter a host name or IP address.

The case against Zeroconf

While zeroconf networking is an exciting new technology capable of making many tasks much easier, and making some tasks possible at all, it is not without its problems. This section provides a brief overview of some of the more important issues.

Changing the way IP works

The way in which link-local networking works is subtly different to the way in which IP addresses can normally be treated, since link-local traffic is never routed. This can be a serious issue for hosts that are communicating using link-local addresses to some hosts, and using routable addresses to other hosts. In this case, it is quite possible that referals based on IP addresses may end up failing in unexpected ways.

The fact that IP addresses are only unique on a single link (at least in the IPv4 case) can be problematic for hosts that have more than one interface. As an example, consider two links (1 and 2), which have three hosts (A, B and C). A is on link 1, C is on link 2, and B is multi-homed on both links using two different interfaces. There is no reason why A and C cannot have the same link-local address. However this makes things complex for B, which cannot use IP address alone to identify a host. There is a workaround for this, however it needs to be implemented at the application layer.

Zeroconf as the antithesis of security

Zeroconf networks cannot provide protection against some important forms of information security attack. In particular, it is not possible to protect from "man-in-the-middle" attacks without some form of out-of-band configuration.

Social and Ethical Issues

As noted above, zeroconf is not a perfect technology as it currently exists. However it should be considered that any technical solution developed by the IETF Zeroconf Working Group can almost certainly be shown to modify the behaviour of at least one application. Some combination of protocol design (or protocol design error) and program design (or program design error) will ensure that this is the case.

So it is worth considering if the IETF should endorse a technical solution that may break applications.

I am personally not especially concerned if some applications break, however I would be concerned if one of protocols that breaks is a major one (say HTTP/IPP, or FTP, or SSH). If specialist applications break (say something that does IP based refers in a cluster environment), then the answer is likely to be "don't run that application and zeroconf technology at the same time". Zeroconf is a technology that can be run in major data centres, but the people with the specialist applications are the ones best able to handle the breakage and modify the configuration of the hosts/networks to compensate.

The progression of zeroconf should be judged on the balance of the advantages and disadvantages, not just the absence of disadvantages.

It is important to recognise that zeroconf can potentially contribute to social equity issues, to ease the division between technically able, and those who cannot access technology. At the moment, ability to use the internet (or local network) is basically the domain of the technical elite. If we pass up the opportunity to make this type of technology more accessible, just so someone's computing cluster doesn't need to be modified, then that would be sad. It might even be unethical.

Conversely, we should understand that the network administrators of the future may not have the experiences that today's administrators have, since network administration requirements for small networks may largely disappear. With a smaller group of people involved in network administration, we should also consider where the next generation of network designers and programmers will come from.

Work to Date

While Linux distributions do not currently support zeroconf networking very well, some progress has been made, and some of the required elements are now available. This section provides a sample overview, mainly to show it isn't all hype.

zcip

zcip is an implementation of the ad-hoc link-local IP autoconfiguration algorithm described in the IETF Draft "Dynamic Configuration of IPv4 link-local addresses"[11]. zcip is mostly conformant to the latest draft.

zcip uses libpcap and libnet (which in turn use fairly standardised socket interfaces provided by the Linux kernel, and also by the various BSD kernels) to provide the required functionality. It runs entirely in userspace, and most kernels are shipped with the required configuration options set[12].

CUPS

CUPS is an implementation of the Internet Printing Protocol, and is rapidly becoming the industry standard for network printing. Internet Printing Protocol does not specify a way of locating network printers, but CUPS has its own protocol. Unfortunately the CUPS printer discovery protocol is broadcast based, and uses a lot of network bandwidth, to the extent where some organisations turn it off.

However CUPS also includes a Service Location Protocol option, which is built if the configuration process finds suitable system libraries (almost always OpenSLP). This can offer a standardised way of locating any printer on the network, without excessive network traffic, potentially including printer location across routers.

The CUPS implementation currently exposes limitations in OpenSLP. In particular, the CUPS daemon is not threaded, and OpenSLP doesn't support the asynchronous API at this stage, so the daemon essentially hangs waiting for the service discovery to time out each time it runs the service discovery loop (30 seconds, by default). Future changes to CUPS (to add a standalone monitoring server for SLP) and OpenSLP (to support the asynchronous API) should rectify these issues.

Demos

As an example, I have produced a simple demonstration based on rsync. The rsync server advertises each of its modules, and the client can identify all the rsync servers that are available.

The changes to rsync are mostly service location protocol boilerplate - two functions were inserted into the existing client and server routines, and a small change was made to the configuration file to allow variable timeouts.

Tools

Apart from OpenSLP and zcip (which were discussed previously), the main tool I use for Zeroconf development is Ethereal[13]. You need at least version 0.9.7 to have any SLPv2 dissection functionality, and version 0.9.8 to get improved handling of mDNS and DNS service discovery.

Future Activities

There are clearly many areas that could take advantage of easier networking, however there are two that really stand out - incorporating service location into widely used servers and clients, and multicast DNS.

Multicast DNS Resolver

A major planned activity is to develop changes against glibc to add a multicast DNS capability. This is effectively against the Bind 8.x resolver, which should be widely portable to other systems.

There are a number of aspects to this, including integration into a typical name server switch (NSS) style design. Among the design issues for this are how to handle the various name spaces, since it is *generally* accepted that the multicast DNS namespace (or Link-Local Multicast Name Resolution) is not the same namespace as the DNS namespace(s), but there rules for doing a lookup from the application are not well understood.

Development of a multicast DNS responder may also be required.

Service Advertisements

As of writing, Apple had just released their new web-browser (marketing name "Safari"), which can use DNS service discovery to locate web-servers. Adding this type of service advertisement capability to various servers, and suitable service discovery capabilities to major servers is clearly important for the on-going acceptance of open source software.

Service Location Switch

It would be very convenient to have a single API that allowed service location using whatever protocols are available, perhaps using some form of failover system like that currently provided by the Name Service Switch.

This would require adding DNS Service Discovery to any multicast DNS responder, and possibly creating a new API for service registration and location.

Notes

1. For more information on the reasoning behind Stories and their application to Extreme Programming, refer to *Planning Extreme Programming* by Kent Beck and Martin Fowler.
2. MADCAP is specified in RFC2730, with additional information in RFC2907.
3. For additional detail on how this works, refer to http://www.ietf.org/internet-drafts/draft-ietf-dhc-dhcpv6-28.txt, or the corresponding RFC which will likely be released by the time linux.conf.au 2003 occurs.
4. RFC2642 provides additional detail on this process.
5. ICMPv6 is a significant change from the ICMP of IPv4, and is a major aspect of IPv6 functionality. ICMP is a very simple protocol. ICMPv6 is a complex protocol. You can think of it as ICMP, souped up and lowered, and pumped full of steroids. Refer to RFC2643 for additional detail.
6. Note that this address may not be routable on the public internet - it could be an RFC1918 address such as the 10/8 or 192.168/16 address spaces.
7. At the time of writing, the current draft was available at http://www.ietf.org/internet-drafts/draft-ietf-dnsext-mdns-13.txt
8. At the time of writing, the current draft was available at http://www.ietf.org/internet-drafts/draft-cheshire-dnsext-multicastdns-01.txt
9. For completeness, it should be noted that this is not the only capability in the iChat application. For example, iChat can also be used as an AIM client. In this example, the "Rendezvous" mode is being discussed.
10. The protocol is really called RFB. VNC is the name of functionality provided by applications that used RFB.
11. At the time of writing, the current version was available as http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-07.txt
12. One important exception is that Linus' kernels come with a default `.config` that does not have the "Socket Filtering" option (`CONFIG_FILTER`) turned on. zcip now detects this, and warns you to reconfigure the kernel.

13. You can download Ethereal from http://www.ethereal.com. It is GPL'd software.