Apache and Zeroconf Networking

Sander Temme
sander@temme.net
June 5, 2005

Abstract

This paper examines the set of IETF standards that comprise Zero Configuration Networking. It then discusses a Zero Configuration Networking plugin for the Apache 2.x web server.

1 Zero Configuration Networking

Networking is hard. It is sometimes surprising how much trouble even technically savvy people have setting up a TCP/IP network. Configuring a network requires us to worry about IP addresses, routing, name servers and sometimes even netmasks. A simple LAN party where a few friends get together to play some games requires a designated network guru. Fortunately, there is a technology that makes setting up a TCP/IP network much easier.

1.1 IETF

The Zero Configuration Networking working group of the Internet Engineering Task Force was formed in September 1999. Its members included engineers from companies like Sun, IBM and Apple. To quote from their charter:

The goal of the Zero Configuration Networking (ZEROCONF) Working Group is to enable networking in the absence of configuration and administration. Zero configuration networking is required for environments where administration is impractical or impossible, such as in the home or small office, embedded systems ‘plugged together’ as in an automobile, or to allow impromptu networks as between the devices of strangers on a train.

The Zeroconf working group wrapped up in July 2003. It has produced RFC 3927, titled Dynamic Configuration of IPv4 Link-Local Addresses and several Internet Drafts.

1http://www.ietf.org/html.charters/zeroconf-charter.html
2http://www.ietf.org/rfc/rfc3927.txt
3http://www.zeroconf.org/ under ‘Documents’
1.2 AppleTalk and the Olden Days

When Apple Computer released the Macintosh computer in the mid-eighties, it featured a proprietary network protocol named AppleTalk. The most important feature of this protocol was its ease of use: networked machines named themselves, figured out their own addresses and users found services – mainly printers – by using a simple network browser.

Of course AppleTalk had its issues: it didn’t scale very well, network administrators complained AppleTalk caused unnecessary network traffic and eventually Apple found themselves forced to switch to TCP/IP as their main networking protocol. TCP/IP, however, while scalable, did not have the ease of use that users had come to expect from AppleTalk. There was no name binding, and no automatic service discovery. The first IP-based version of Apple’s file server product still used AppleTalk to find servers, but when a user logged into the server, and client and server had a working IP link between them, TCP/IP was used for the actual connection. This solved the scalability issue, but not the chattiness.

It was clear that a new technology was needed to fully replace AppleTalk, so Apple got involved with the Zeroconf working group at a very early stage.

1.3 Technology Overview

The definition of Zeroconf Networking is fairly limited in scope: it is intended for small networks, ad-hoc networks or situations where devices need to communicate with each other over a direct connection. There is no provision for routing traffic in and out of the Zeroconf network.

The beauty of Zeroconf Networking is that the protocols it defines are small
extensions of existing network protocols, and they coexist peacefully with their centrally configured brethren. Also, the Zeroconf protocols do not depend on each other: one can function in the absence of others.

1.3.1 Link-local Addressing

The first problem is that of assigning IP addresses to hosts on the network. In the absence of a DHCP or Bootp server, hosts can select their IP addresses at random from the designated B-class network 169.254.0.0. When a host selects an address, it sends out an Address Resolution Protocol (ARP) broadcast to find out if anyone has previously selected the same address. If no other host answers, the address is available for use. On the other side, a host that has claimed an address can defend it against new hosts that try to supersede its claim. This form of addressing is called ‘link-local addressing’. It is tied to the local network because the ARP protocol is link-local. There is also no way to publish a default IP route out of the link-local network, so this form of addressing cannot be used to access the Internet. Link-local addressing can coexist with centrally configured addressing, since a host can have more than one address on any interface. However, in situations where a central address server is available, it’s often preferable to use that instead.

Link-local addressing was introduced in 1998 in Windows 98 and MacOS 8.5. For Linux systems, one can use the projects from the Zeroconf Sourceforge project described below. Mandrake Linux 9.1 includes both their zcip and tmdns packages to provide this functionality.
1.3.2 mDNS

The second problem to solve is that of naming hosts in the absence of a DNS server. The solution is to run a DNS-like responder on every host on the network. This responder speaks the normal DNS protocol, but listens on UDP port 5353 and binds to multicast address 224.0.0.251. The domain name `.local.' (no quotes) is reserved for mDNS hosts and every host on the network chooses its own name. Like the 169.254 addresses described above, these names are link-local to the network on which the host resides. When a query for a hostname arrives on the multicast socket, the host responds to that same multicast socket so all the hosts in the network get that information and can cache it if necessary.

Self-assigned names could of course lead to naming conflicts, but decades of experience with the venerable AppleTalk protocol which uses self-appointed names has proven that there are many human factors in naming hosts, so this is not really a problem. Besides, mDNS does have a defense mechanism that prevents the occurrence of conflicts.

While mDNS is DNS-like, there are some differences. For instance, there are no DNS or SOA records: there is only one zone, `.local.` with one DNS server, unchangeably designated 224.0.0.251, and since all hosts on the network cooperate to form a single, distributed DNS server, no serial number or central authority. Additionally, the protocol has been modernized to allow (and in fact, requires) UTF-8 (which is similar to ASCII) for hostnames and raise the maximum size of DNS messages. In the past, these were restricted to 512 bytes, but can now be up to the Maximum Transferable Unit (MTU) of the network.

The mDNS protocol can work in conjunction with link-local addressing, but since it uses its own multicast address, does not depend on it. It operates well on conventional, centrally administered networks. There is another Internet Draft regarding this protocol.

1.3.3 DNS-SD

DNS-based Service Discovery (DNS-SD) is the third and final piece to the Zeroconf puzzle. The user can browse a service namespace by sending DNS PTR queries for, for instance, "http._tcp.example.com.", and get a list of answers like "Larry's Personal Website._http._tcp.example.com.", "Corporate Intranet._http._tcp.example.com.". This information can then be used to query for a SRV record that will yield an actual DNS hostname and port, and TXT records that contain information such as a partial pathname. When the user is browsing the `.local.` domain, queries will generally go over multicast DNS. However, this scheme can be set up very well on a centrally administered DNS server: the record types mentioned above are all part of the normal DNS protocol. The service names like "_http._tcp" are discussed below. There are a couple of things worth noting about the above example. Firstly, the PTR records contain human-readable names for web site services. When, in an office setting,
Figure 3: Multicast DNS (mDNS) hostname assignment

a person is searching for someone’s personal web share, being able to identify it by the person’s first name makes this process much more accessible than it would be using cryptic numbering conventions. Any UTF-8 string can be used as service name. Secondly, the port number a service instance runs on is part of the SRV record. Since a client can dynamically and automatically, look up port numbers, it becomes less important to run services on well-known ports. In an mDNS situation, it’s even feasible for the server to bind to a random port number: clients will find the service through mDNS. Finally, the human-readable service name (Larry’s Personal Website) does not specify which host (for example “FINAPC023BA4”) said website actually runs on. The hostname is also part of the SRV record, and is resolved into an IP address through a regular DNS A record. The hostname may change, or the service may move to a different host, or a different port. Thus the user, who just sees the human-readable service name, may never be aware of such changes.

1.3.4 Service Namespace

Giving names to services is discussed in RFC 2782. The basic idea is that one takes the official service names as defined by the Internet Assigned Numbers Authority (IANA) and prefixes an underscore, for instance ‘http’ becomes ‘_http’. Then, take the protocol used for the service, tcp or udp and add that,
Figure 4: DNS Service Discovery (DNS-SD) service advertisement

again prefixed with an underscore. This is how we end up with service names like ‘.http.tcp’ or ‘.nfs.udp’. The DNS-SD draft\(^9\) also discusses the possibility of service subtypes: an ftp server that allows anonymous access could advertise the service ‘.anon.ftp.tcp’. In a similar fashion, file service clients could browse for services that advertise as ‘.dav.http.tcp’, but leave regular web servers alone. These subtypes, however, are not standardized but need to be defined by the individual services. There is a potential for incompatibilities as long as no standards are in place.

1.4 Applications of Zero Configuration Networking

There are a lot of applications for an automatically configured, easy to use service discovery method. The protocols are small and easily implemented through available libraries. Additionally, adopting Zeroconf for service discovery does not require radical changes in applications: it can exist concurrently with other discovery methods.

The only major operating system manufacturer that has included Zeroconf Networking support in their product to date has been Apple. This means that most applications for Zeroconf Networking have been happening on the Mac OS X platform. However, the Linux folks are catching up and developers can now find Zeroconf implementations available as part of the standard installation on a number of distributions. So far, a Zeroconf Networking implementations has not been part of the Windows operating system, which requires developers to

\(^9\)draft-cheshire-dnsextdns-sd.txt, included on the CD
supply their own Zeroconf implementation. This could make adoption on more
difficult on the Windows platform.

1.4.1 Network Printing

The first application for Zeroconf Networking was probably network printing.
Printers are an ideal candidate for Zeroconf support: they have no user interface
of their own, limited code space in their firmware, offer one or multiple network
services and are often set up and maintained by regular office workers rather
than trained network administrators. Over the past couple of years, many
printer manufacturers have adopted Zeroconf networking and it is likely that
the networked printer you buy today has it on board.

What does this mean? It means that you can today take your printer out of
the box, connect it to the network, turn it on and it will automatically assign
itself an IP address, and start broadcasting the printing services it offers over
mDNS. Every client that has an mDNS browser built into their printing software
(like Mac OS X or Windows PCs with Apple’s Bonjour Printer Browser), will
be able to see these services and communicate directly to the printer. It can use
whatever protocol is a best match between both client and server. Meanwhile,
network administrators are free to set up printer queues on their servers for
clients that are not Zeroconf enabled. If a network admin wants to run all
printer traffic through the server queues, she can easily disable Zeroconf service
advertising.

1.4.2 Collaborative Document Editing

The shareware product SubEthaEdit provides a collaborative document editing
feature using Zeroconf Networking. This allows multiple participants on the
network to edit the same document concurrently. It is often used at conferences
to take notes during sessions.

1.4.3 Link-local Instant Messaging

iChat is Apple’s instant messenger application. Besides connecting to the AOL
and Jabber Instant Messenger networks, it browses the local network for other
iChat users. This is great in meetings: you can send a message to your coworker
across the table from you to let him know he has lettuce in his teeth. Other
instant messenger clients have also started to provide Zeroconf support for local
instant messaging.

1.4.4 Service and Server Management

Advertising a management interface on the local network using Zeroconf is a
very powerful and user-friendly way to make service and server configuration
available to users. For instance, besides its printing services, every networked
HP printer advertises a web-based management interface that can be accessed
by anyone with a web-browser. Since Zeroconf Networking includes automatic
IP address assignment, such a management interface may even be accessible before a device has been configured. When they first came out, Apple’s Airport base stations were pre-configured with a certain IP address and the user had to reconfigure their computer accordingly. Today, Airport base stations advertise their management service over Zeroconf and can be accessed by the configuration software before the base station has been configured with its own IP address. This bootstrap capability makes Zeroconf Networking very powerful for this application.

2 Apache httpd and Zeroconf

In a Zero Configuration world, the Apache web server would advertise its services on the Zeroconf network. Users could find servers by name using a Zeroconf service browser. This could be built into their web browser, but it’s also possible to have a standalone Zeroconf service browser that instructs the web browser to open the services it finds.

Having the web server advertise services on the Zeroconf network does not make sense in all cases: the Zeroconf service advertisement is by definition only visible on the link-local network, and many web servers are accessed from well beyond the local link. This makes Zeroconf support for web servers a fairly academic exercise, but in situations where a web server is accessed locally and should be easily found, it can be quite handy.

An interesting side effect of Zeroconf Networking support for web servers is that it can cause a paradigm shift in how TCP port numbers are used. Traditionally, the web server listens on the well-known port 80 for http and port 443 for SSL enabled servers. These port numbers are prescribed because web browsers know to find servers at this port, and require the user to type a longer, scarier, custom URL in the web browser address bar if the server runs at a different port. In the DNS-SD record that is part of Zeroconf Networking, the port number is specified along with the hostname and service name. This allows the server to bind to an arbitrary port number, and communicate this information to the client through Zeroconf. This effectively means the end of well-known ports.

Of course, for this to happen all clients that wish to access the server must support Zeroconf Networking. In the case of web browsers this is an unlikely assumption because there are so many varieties and versions, and no one has a grip on deploying new versions of popular web browsers. Fortunately, Zeroconf Networking can perfectly well publish services that run on the traditional well-known port and are defined elsewhere (like in regular DNS and on Google) for the benefit of Zeroconf-challenged web browsers.

2.1 Server Support for Zeroconf

To make Apache support Zeroconf Networking, it needs to either incorporate mDNS functionality, or use an existing mDNSResponder on the platform. It
would be quite easy to configure Apache httpd services for publication in the configuration file of an mDNSResponder daemon, but that would not be very flexible and would add a manual configuration step to setting up Apache. It would be much more desirable to have Apache automatically publish its services.

The Web Sharing feature of Apple’s Mac OS X operating system runs on Apache 1.3, and Apple supplies an Apache module that does Zeroconf service publishing. This module offers the dynamic publication service that we want, but it has some problems that prevent wider adoption:

- It is released under the Apple Public Source License, which requires that modifications be submitted back to Apple. Some open source developers have problems with clauses like this one.
- It runs only on Apache 1.3 and not Apache 2.0.
- It uses Mac OS X specific CoreFoundation APIs.
- It uses threads which may not be supported on all platforms.
- It does not seem to support virtual hosts.

All of the above concerns should be addressed by mod_zeroconf, which will be discussed in paragraph 3.

### 2.2 Client Support for Zeroconf

Server-side support for Zeroconf Networking is of course of limited value without client-side support. When the service is an HTTP web server, it stands to reason to build client support into web browser software. It’s theoretically possible to, on any given platform, to build a standalone Zeroconf Network Service Browser\(^\text{10}\) that uses an inter-program communications mechanism to open the user-selected service in a web browser, but having Zeroconf support built into the service client seems to offer the better user interface experience.

Currently, the Safari browser on the Mac OS X platform supports Zeroconf HTTP service browsing out of the box. Apple also makes available a Zeroconf plugin for Microsoft Internet Explorer on Windows. It would be really nice, of course, to have a Firefox or Mozilla plugin that would offer Zeroconf browsing in a sidebar to the web browser, but this has so far not emerged into the mainstream.

### 3  mod_zeroconf for Apache httpd 2.x

The mod_zeroconf module for Apache httpd 2.x aims to fulfill the following requirements:

\(^{10}\)Note the use of the word ‘browser’ for both finding Zeroconf Networking services and for viewing web pages. Author hopes that the distinction remains clear.
1. Publish Apache VirtualHost services through DNS-SD using a configurable service name and optionally a custom partial URL path.

2. Take (virtual) host configuration information as base for service publishing.


4. Support multiple mDNS implementations.

5. Published under the Apache License, version 2.0.

6. Be disabled by default for security reasons. To publish services, the user has to explicitly configure publication in the Apache configuration file.

The existing code supports all of the requirements above, with exception of item 4 and (partially) item 2. The latter mostly hinges on the difficulty to conclusively determine which IP address(es) the server is listening on, and to which port a given virtual host is bound.

### 3.1 Module Design

The mod_zeroconf module does most of its work in the post_config hook implementation. This hook is run after the server configuration file has been read, but before children are forked. Essentially, the post_config hook handler walks through the server_rec linked list it is passed. For every server that has the Zeroconf directive enabled and a ZeroconfRegister directive, it publishes the service name and all other information (see paragraph 3.3) with the mDNSResponder.

After it is done publishing, the post_config hook handler typically has to set up a callback construction. Depending on the Zeroconf implementation, the Apache server has to respond to various callbacks regarding the information it publishes. These callbacks typically concern registration conflicts, especially when the server loses such a conflict. Since the Apache httpd has no user interface and cannot put up a dialog window asking a user to, for instance, pick a different service name, it will have to derive its own unchallenged alternative and re-publish.

The main implementation difficulty for mod_zeroconf is to find out which IP address(es) and TCP port to register with the mDNSResponder. This information has to be obtained through several indirections inside the server_rec structure, and the Apache module API has not been designed with this sort of task in mind.

It should be noted that, in spite of its name, mod_zeroconf does not operate without manual configuration. It requires a server administrator to turn the module on in the Apache configuration file, and to explicitly publish services with their human-readable service names. This is not quite the spirit of Zeroconf, but it’s as close as we can get within the current module design. A pair of directives is a lot better than having to manually gather configuration details and having to enter them into an mDNSResponder configuration file.
3.2 Configuration Directives

**Zeroconf** Turn Zeroconf on or off for the entire server or a given virtual host. Parameter: on or off. Default: off. This directive should be inherited by VirtualHost containers and overridden by occurrence in a VirtualHost container.

**ZeroconfRegister** Register server or virtual host with Zeroconf mDNS Responder. Parameters: service name and optional partial URI. This directive should not be inherited. This directive can be used multiple times to register different services or web applications under a certain (virtual) host. If a registered service actually runs on a different server, its ZeroconfRegister directive can be paired with a Redirect directive to redirect incoming requests elsewhere.

For example, the following snippet registers a company Intranet:

```html
<VirtualHost *:80>
  Zeroconf on
  ServerName intranet.example.com
  ZeroconfRegister "Example Inc. Corporate Intranet"
  # Additional configuration parameters
</VirtualHost>
```

The following stanza can be used to register a virtual host that offers a Bugzilla bug tracker:

```html
<VirtualHost *:80>
  Zeroconf on
  ServerName bugs.example.com
  ZeroconfRegister "Example Inc. Bug Tracker" /bugzilla
  # Additional configuration parameters
</VirtualHost>
```

The following stanza can be used to register a company Wiki that runs on an entirely different server:

```html
...
Zeroconf on
ZeroconfRegister "Example Inc. Company Wiki" /companywikiredirect
RedirectPermanent /companywikiredirect http://wiki.example.com/wiki
...
```

The partial path in this example is a complete artifact: as soon as the user’s web browser points to this partial URI, it receives a permanent (301) redirect to the server that is running the actual Wiki.
### 3.3 What Gets Registered

The goal of mod_zeroconf is to publish web server services on a local mDNS responder under a human-readable service name. A user should be able to select this name in a list in their Zeroconf Services Browser, and open the underlying web site in a web browser. However, for this to happen, the human-readable service name needs to resolve to a true DNS hostname, which in turn should resolve to an IP address that the server is listening on and that the client can reach. The intermediate step, the DNS hostname, cannot be skipped in this case, because the service may be running on a name-based virtual host and the HTTP client (web browser) must supply the correct hostname in the HTTP header.

Therefore, mod_zeroconf must for every service register the following information with the mDNSResponder:

- The human-readable service name
- The port that the (virtual) host is bound to
- The DNS hostname of the server (if different from the system hostname, which is already published)
- All IP addresses the (virtual) host listens on. These could be all IP addresses present on the host, a specific address for a (virtual) host, or a subset of listening addresses that applies to the port the (virtual) host is bound to.
- The DNS-SD service hostname. This will be `http.tcp` in the case of a plain-text web server, but should be `https.tcp` in the case of an SSL-enabled server.
- The path parameter. This is `path=/` for the entire web site, but can be a partial URI if the registration is more specific.

### 4 Conclusion

Zero Configuration Networking is a promising, lightweight, easy to implement technology for service publication and discovery. Its most exciting applications are in the realm of peer to peer, user to user networking, but it can very well be used to publish web server services offered by an Apache httpd. This paper has described the Zeroconf Networking technology as defined by the IETF Working Group, and has discussed an Apache module that automatically publishes Apache (virtual) hosts on the Zeroconf network.
A Implementations of Zeroconf

A.1 Apple’s Bonjour

The Bonjour\(^{11}\) technology that Apple supplies as part of their Mac OS X operating system is the most complete implementation of Zeroconf currently available. The Mac OS has offered support for IPv4 Link-local address assignment since version 8.5 in the late 1990s, and since version 10.2 it came with an mDNSResponder and a set of APIs for application development.

The availability of support for every development framework on the platform has made it very easy for Mac OS X application developers to incorporate Zeroconf support into their products. Many Zeroconf technology applications start their life on Mac OS X. Specifically, Apple offers support\(^{12}\) for their Carbon and Cocoa frameworks, as well as for Java and, through the dns_sd.h file, for regular C programs.

Apple makes its mDNSResponder source code available under the Apple Public Source License. They also make the dns_sd.h file available under a BSD style license and are trying to make this interface the de-facto standard for applications to register or browse Zeroconf services.

A.2 Howl

Howl\(^{13}\) is a Zeroconf implementation from Porchdog Software. It is mostly BSD licensed, but does include the mDNSResponder code from Apple so portions fall under the APSL. Howl recently had its 1.0 release and has been known to work on Linux, FreeBSD and Mac OS X. The Windows port has not received a lot of attention lately. The current version of mod_zeroconf uses Howl.

In the future, Howl author Scott Herscher plans to remove all APSL licensed code and change his API to the dns_sd.h interface published by Apple.

---

\(^{11}\)This technology started its life as *Rendezvous* but was recently renamed and extended.

\(^{12}\)http://developer.apple.com/networking/bonjour/

\(^{13}\)http://www.porchdogsoft.com/products/howl/