

# Apache and Zeroconf Networking

By Sander Temme <sander@temme.net>

## ***What is Zeroconf?***

Zeroconf networking is a technology that enables the use of TCP/IP networking protocols without any configuration or preparation. Usually, using TCP/IP means configuring an IP address for each machine on the network, either manually or by setting up a DHCP server. Accessing those machines by name (instead of IP address), and finding out what services they offer takes yet more infrastructure, which can include hosts files, NetBIOS, master browsers, DNS or WINS servers.

For small networks or ad-hoc networks, setting up this infrastructure is often impractical. Not everyone has the expertise to set up a DHCP server, or to even know that they need one. *Zero Configuration Networking* is the name of a Working Group of the *Internet Engineering Task Force (IETF)*<sup>1</sup> that attempts to solve these problems.

## **Apple**

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.

## **IETF**

It was clear that a new technology was needed to fully replace AppleTalk, so Apple turned to the Internet Engineering Task Force (IETF). A Zeroconf IETF working group was formed in September 1999. Apple employees did a lot of the initial legwork, but if you look at the drafts this working group has produced, you see authors hailing from Sun, IBM and other companies.

The result, so far, of this working group is a collection of Internet Draft documents documenting the protocols summarized below. Starting with the release of version 10.2 of their MacOSX operating system, Apple has included support for the Zeroconf

---

<sup>1</sup> <http://www.ietf.org/html.charters/zeroconf-charter.html>

networking protocols under the marketing name Rendezvous.<sup>2</sup> Mandrake Linux version 9.1 also currently supports Zeroconf.

## Technology Overview

The application 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 co-exist peacefully with their centrally configured brethren. Also, the Zeroconf protocols do not depend on each other: one can function in the absence of others.

## 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 co-exist 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.

## 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.

---

<sup>2</sup> <http://www.apple.com/macosx/features/rendezvous/>

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<sup>3</sup> regarding this protocol.

## DNS-SD

DNS-based Service Discovery<sup>4</sup> (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, 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<sup>5</sup>.

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

---

<sup>3</sup> draft-cheshire-dnsext-multicastdns.txt, included on the CD

<sup>4</sup> draft-cheshire-dnsext-dns-sd.txt, included on the CD

<sup>5</sup> Periods in service names are escaped with backslashes. Backslashes are escaped with backslashes.

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.

### ***Service Namespace***

Giving names to services is discussed in RFC 2782<sup>6</sup>. The basic idea is that one takes the official service names as defined<sup>7</sup> 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, again prefixed with an underscore. This is how we end up with service names like ‘\_http.\_tcp’ or ‘\_nfs.\_udp’. The DNS-SD draft<sup>4</sup> 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.

## **Practical Uses of Zeroconf**

### ***Network Printing***

The examples in the Zeroconf IETF drafts tend to center around printers. This makes sense, because most of them were written by Apple and, traditionally, this is why Macs have been networked. However, the potential applications of Zeroconf are many and diverse.

Over the past couple of years, many applications and products have sprung up that use these protocols. To date, the only platform to fully implement Zeroconf networking is MacOSX, many of these products work primarily with, or on, this platform. Apple’s iTunes music player can stream music over the local network and uses Zeroconf to locate similarly equipped computers on the network. This way, you could play your co-worker’s shared music in your own cubicle. Another Apple application that uses Zeroconf is iChat. Besides connecting to the AOL Instant Messenger network, it browses the local network for other iChat users. This is great in meetings: you can send a message to your co-worker across the table from you to let him know he has lettuce in his teeth.

### ***LAN Gaming***

Another application area for Zeroconf networking is LAN gaming. Games on the PC side have long used IPX for local networking because it is easier to configure than TCP/IP. Zeroconf is an excellent cross-platform technology for this application and adopting it would enable gamers on Windows, Mac and Linux to play together (or against one another, as the case may be).

### ***File Server Location***

---

<sup>6</sup> <ftp://ftp.ietf.org/rfc/rfc2782.txt>

<sup>7</sup> <http://www.iana.org/assignments/port-numbers>

Zeroconf networking can also be used to browse for file servers. Currently, there is a browsing mechanism for Windows file and print services, based on NetBIOS over TCP/IP. This relies either on a centrally configured and administered Windows Internet Name Service (WINS). In the absence of a WINS server, as in a situation where multiple PCs share files on a peer-to-peer basis, one of the PCs is elected *Master Browser* and it maintains the list of available services. If that PC disappears from the network (for instance because it is a laptop and its owner took it home), it can take up to 15 minutes for a new *Master Browser* to be elected. Zeroconf is much better equipped for such dynamic environments. Every host on the network advertises its own services over Multicast. There is no central repository of information. Additionally, while Windows can mount FTP and WebDAV servers as file servers, finding them requires knowledge of their URLs since they don't show up in the *Network Neighborhood*. If Zeroconf were enabled, these services would appear automatically.

### ***Mail Server Identification***

Finally, consider the situation when a person arrives at a conference. She can easily log onto the conference wireless network: the network name is usually easily recognizable and configuration happens automatically through DHCP. However, when she wants to send mail, she needs to know the hostname of the outgoing SMTP server, and needs to know where that information goes.

Experts may know how to run a mailserver on their laptop and get the mail delivered directly, but even this approach is prone to difficulties as more and more client IP address pools get blacklisted. Why wouldn't there be a mailserver that advertises a `_smtp._tcp` service? Mail clients could browse for such a service and give the user the option of choosing any of the servers found. Currently, there are no mail clients or servers known to implement this, but it would be a very useful application of Zeroconf protocol.

### ***Security***

The above example brings with it the question of security. What would prevent a rogue conference attendee from advertising a mail service of their own and intercepting outgoing mail? In a small group, peer pressure can be used to deal with any rogue services springing up: an example of this happened at the 2003 O'Reilly Open Source convention, where a publicly editable page was kept to combat abuse of the wireless network<sup>8</sup>. The DNS protocol has security and authentication mechanisms that can be used to verify the origin of information like that mail service from the example. DNSSEC<sup>9</sup> can also be used on mDNS, but it remains to be seen how feasible it is to obtain a trusted zone key in an ad-hoc situation. At the service level, security and authentication mechanisms like SSL/TLS can be used to verify authenticity and encrypt data in-transit.

---

<sup>8</sup> <http://oscon.kwiki.org/index.cgi?WirelessTroublemakers>

<sup>9</sup> <ftp://ftp.ietf.org/rfc/rfc2535.txt>

## **Apple's Sample Code**

When Apple included Zeroconf in their OS under the marketing name *Rendezvous*, they made some code available<sup>10</sup> under the Apple Public Source License. There is a sample mDNS responder that runs on Windows, MacOS 9 and X, VxWorks and a number of POSIX platforms. This code is indented as a sample for developers who want to implement Zeroconf in their products. It is functional, but for instance the mDNS responder is configured through a configuration file or the command line, and lacks a programmable API.

## ***Zeroconf-enabling Apache httpd***

The Apache web server is an ideal server program to work with Zeroconf: it can publish any number of services for human users. Before discussing the requirements for making Apache Zeroconf-aware, let's look at the projects already under way.

## **Existing Efforts**

### **mod\_rendezvous**

This is an early effort by Eric Seidel<sup>11</sup>. He started mod\_rendezvous<sup>12</sup> in order to enable Apache on MacOSX/Darwin to register itself with the mDNS responder. This was before Apple released its own Apache module (see below), and it looks like development on mod\_rendezvous has stagnated. Perhaps this has to do with the fact that Eric began an internship at Apple in April 2003. This module works only on MacOSX, and is available under the Apache license.

### **mod\_rendezvous\_apple**

Apple released mod\_rendezvous\_apple as part of the MacOSX 10.2 operating system, which includes Apache 1.3 as 'personal web server'. It registers Apache VirtualHosts and user pages with the mDNS Responder. This module works only on MacOSX and is available under the Apple Public Source License and can be obtained from Apple's CVS<sup>13</sup>.

### **dotlocal.org**

This is a web site<sup>14</sup> that assembles some links to other projects working on or with Zeroconf. Jeremie Miller, who started the Jabber<sup>15</sup> project, maintains this site. He has a library package that applications can link in to register themselves on mDNS.

---

<sup>10</sup> <http://developer.apple.com/darwin/projects/rendezvous/>

<sup>11</sup> <http://homepage.mac.com/macdomeeu/>

<sup>12</sup> <http://sourceforge.net/projects/modrendezvous/>

<sup>13</sup> <http://developer.apple.com/darwin/tools/cvs/howto.html>

<sup>14</sup> <http://www.dotlocal.org/>

<sup>15</sup> <http://www.jabber.org/>

## **zeroconf.sourceforge.net**

This is a Sourceforge project<sup>16</sup> that aims to develop support for Zeroconf protocols on UNIX systems, specifically GNU/Linux and BSD. Projects include a Zeroconf autoconfiguration implementation named *zcip*, and their CVS repository contains an mDNS responder named *tmDNS*, where ‘t’ stands for either ‘tiny’ or ‘trivial’.

## **Swampwolf**

San Diego-based Swampwolf<sup>17</sup> publishes a product named *Howl* that implements Zeroconf on Linux, FreeBSD and Windows. *Howl* provides an mDNS responder, APIs for publishing and browsing services, and, on Windows, an Internet Explorer plug-in that lists http and ftp services published through DNS-SD right in your browser. Very cool. *Howl* is available from Swampwolf’s website. Early versions were licensed under the GPL, but more recent releases carry the BSD license.

## **mod\_zeroconf**

The `mod_zeroconf` module for Apache is currently in the design stage. Its objective is to register Apache virtual hosts with an external mDNS responder, just like the Apple module mentioned above. However, unlike Apple’s effort, `mod_zeroconf` should work on other operating systems. The design of the module is discussed in this paper.

## **Requirements**

### **Assumptions**

This module assumes that there is an mDNS responder available on the system, which can be addressed through an API. At the time of writing, the only freely available project that offers this functionality is *Howl* by Swampwolf, discussed above. The first design of this module will use the *Howl* library.

### **Architecture**

The architectural design for this module will be discussed at the Apachecon 2003 conference. Design documentation should also show up at the presentation home page<sup>18</sup>

### **Code**

Code is forthcoming and should be presented at the Apachecon 2003 conference.

## **Zeroconf-enabling Tomcat**

The design work for making Tomcat use Zeroconf has not started. . Since Tomcat supports dynamic deployment of webapps, the mDNS registration should support this and dynamically register deployed apps. In the case of deployed Web Services, the registration could include the WDSL information in the TXT record. . Probably use *jRendezvous* and embed that in Tomcat one way or the other.

---

<sup>16</sup> <http://zeroconf.sourceforge.net/>

<sup>17</sup> <http://www.swampwolf.com/>

<sup>18</sup> <http://apache.org/~sctemme/Apcon2003/TU10/>

## **Conclusion**

Zero Configuration Networking is a highly promising technology. Its protocols enable resolution of many configuration problems for TCP/IP networks. However, wide adoption and support for Zeroconf protocols is needed. The Rendezvous functionality in Apple's iChat is really effective, but it would be even more effective if it would work with other chat applications that do not run on Apple computers. The *gaim* chat client, a freely available open-source program that can communicate with the major chat applications like MSN, Yahoo Instant Messenger and AIM, should do Zeroconf too, but it cannot be expected to implement its own mDNS responder.

To allow Zeroconf-enabled applications to run on their platforms, Linux distribution vendors should incorporate Zeroconf networking in their products and make its functionality available through a standardized API. Microsoft, too, needs to offer support for Zeroconf protocols on the Win32 platform. True, there is a third-party mDNS responder available on the platform, but as long as this functionality is not available as part of the OS, it will not gain wide acceptance since most users do not wish to install additional software. They just want things to work. And that is what Zeroconf is all about.

## **Appendix A: *mod\_zeroconf* Requirements**

Note: the following list of requirements consists of rough notes and should be considered a work in progress.

- Module traverses the vhost configuration and takes note of hostname, port and any special features (SSL, DAV, ...).
- Module recognizes a directive in main server or vhost containers that en/disables Zeroconf registration
- Module recognizes a directive in main server or vhost containers that sets the Zeroconf service instance name (or goes to disk and get <title> tag of index page? What if there is no index page, or it is dynamic?)
- Module recognizes a directive in main server or vhost context that takes a partial URL as one parameter and a Zeroconf service instance name as the other parameter for registration of that partial URL as separate service instance with otherwise the same (host, port) parameters as its context
- Upon Apache startup, or restart, the module registers all known information with an external mDNS responder using some kind of IPC (TBD)
- The module listens for callbacks (for instance, upon hostname or service name conflicts where it needs to come up with alternative names). In case of a conflict, the module needs to log the problem, resolve the conflict and re-register the service.