Zero Configuration Networking

Daniel Siegel
Betreuer: Andreas Müller
Seminar Future Internet SS2009
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München
Email: siegel@in.tum.de

***Kurzfassung*— Eine Lampe wird an das Stromnetz angeschlossen, eingeschaltet und sie funktioniert. Es soll genauso einfach sein, ein Netzwerk ohne manuelle Konfiguration zu erstellen und zu benutzen. Das war das Ziel, als die Zeroconf Working Group ihre Arbeit in diesem Gebiet 1999 begann. Automatische Allokation von IP-Adressen ohne DHCP Server (IPv4 Link-Local Addressing), Übersetzung von Namen und IP-Adressen ohne einen DNS Server (Multicast DNS) und das Finden von Services im lokalen Netzwerk ohne einen Directory Server (DNS Service Discovery) war und ist die Grundlage von Zeroconf.**

In dieser Arbeit wird Zero Configuration Networking vorgestellt und auf den Aufbau, die Funktionsweise, bestehende Implementierungen und die Sicherheit von Zeroconf eingegangen. Es wird gezeigt, dass konfigurationslose Netzwerke kein Zukunftsszenario mehr darstellen und bereits jetzt im vollen Umfang nutzbar sind.

***Schlüsselworte*— Zero Configuration Networking, Zeroconf, IPv4LL, mDNS, DNS-SD, lokale Netzwerke, Zeroconf Working Group, Bonjour, Avahi**

I. Einleitung

Netzwerke einzurichten und zu konfigurieren galt schon immer als eine nicht sehr einfach zu lösende Aufgabe. In den meisten Fällen benötigt es gut ausgebildete Netzwerkadministratoren, die diese Aufgabe übernehmen. Hierbei wählen die Administratoren traditionellerweise zwischen statischer und dynamischer Adressierung: Statische Adressierung findet man vor allem bei fortwährend verbundenen Computern oder Netzen, wie z.B. bei Servern. Dynamische Adressierung hingegen wird vor allem bei begrenzter Anzahl von IP-Adressen, großer Teilnehmeranzahl oder schnell wechselnden Konfigurationen von Teilnehmern benutzt.

Bei allen oben genannten Möglichkeiten benötigt es einen Netzwerkadministrator, der entweder die statischen Adressen einrichtet oder Dienste, die dynamische Addressierung bewerkstelligen, verwaltet. Neue Teilnehmer zu diesen bereits bestehenden Netzwerken hinzuzufügen, bedeutet so einen Mehraufwand für den Administrator. Große Netzwerke einzurichten und zu verwalten ist zwar möglich, benötigt aber Manpower und Arbeitsaufwand.

Wenn nun aber nur wenige Teilnehmer schnell ein lokales Netzwerk aufbauen möchten, so ist in den meisten Fällen weder die Zeit, das Wissen oder ein Netzwerkadministrator zur Stelle, der dies bewerkstelligt. Wie ist es nun aber möglich, ein Ad-Hoc Netzwerk zu erstellen, das sich automatisch konfiguriert und nicht auf einen Netzwerkadministrator angewiesen ist? Genau diese Frage versucht Zero Configuration Networking zu lösen.

Als Analogie kann man sich folgendes Beispiel vorstellen: Man kauft sich eine Lampe bei einem Fachgeschäft. Zu Hause angekommen, steckt man nun die Lampe in eine Steckdose. Die Arbeit ist somit abgeschlossen. Wenn man nun aber nur schnell eine Datei über ein lokales Netzwerk auf einen Computer übertragen möchte, so scheitert es beispielsweise bereits an der korrekten Subnetzmaske. Eine Lampe besitzt so eine Subnetzmaske nicht. Zero Configuration Networking legt großen Wert auf die Einfachheit, auch wenn dies nicht das einzige Ziel ist [1].

Die Ausarbeitung ist wie folgt gegliedert: Zuallererst geht die Arbeit kurz auf die geschichtliche Entwicklung ein und zeigt ein Beispiel von Zeroconf auf. Kapitel II beschreibt die Anwendungsgebiete von Zeroconf, Kapitel III geht folgend auf die Funktionsweise ein. Einige Implementierungen werden in Kapitel IV vorgestellt, die Sicherheit von Zeroconf wird dann in Kapitel V beschrieben. Kapitel VI schließt die Arbeit mit einem Fazit und einem Ausblick ab.

A. Geschichte

Zur Zeit, als das Internet Protokoll (IP) entwickelt wurde, wurden von mehreren Unternehmen proprietäre Systeme erstellt, die ähnliche Ziele wie Zero Configuration Networking hatten. Dazu gehören AppleTalk von Apple, IPX von Novell und NetBIOS/SMB3 von Microsoft, die bereits automatische Adressierung, das Finden von Services im lokalen Netzwerk und teilweise Übersetzung von Namen mitbrachten und somit die Kommunikation und Benutzung von Services im lokalen Netzwerk ermöglichten. Somit war es bereits damals möglich, Drucker über das Netzwerk anzusprechen, wie in [2] beschrieben.

Diese Systeme wurden aber eingestellt bzw. kaum mehr benutzt, da IP eine robustere, bessere und offen dokumentierte Basis darstellt. So kann man Zeroconf als den Nachfolger von AppleTalk ansehen, jedoch auf der Basis von IP und vollständig funktionstüchtig [3].

B. Zero Configuration Networking an einem Beispiel

Zero Configuration Networking lässt sich am einfachsten an einem Beispiel erklären. Angenommen bei einer Präsentation

möchte der Computer des Vortragenden kein Bild anzeigen. Um nicht viel Zeit zu verlieren, sollen nun die Vortragsfolien auf einen anderen Präsentationslaptop übertragen werden. Um dies zu erreichen ist nur ein Ethernet-Kabel bzw. eine Wireless-Karte auf jedem der Computer nötig. Die Konfigurationen werden automatisch vorgenommen und sobald dieser Prozess abgeschlossen ist, können die Folien nun übertragen werden ohne weitere Einstellungen vorzunehmen. Dies funktioniert in dem Fall, dass bereits beide Computer mit einem anderen Netzwerk verbunden sind, z.B. Universität oder Internet, aber auch in dem Fall, wo kein solches Netzwerk verfügbar ist, ja nicht einmal ein Router oder Forwarder.

Die oben genannte automatische Konfiguration umfasst nun nicht nur die automatische Zuweisung einer IP-Adresse, sondern auch die Auffindung und Benutzung von Diensten, wie in den nächsten Kapiteln beschrieben wird.

II. Anwendungsgebiete

Zeroconf kann in sehr vielen Anwendungsgebieten benutzt werden, teilweise als Ersatz, teilweise als Erweiterung eines bereits bestehenden Netzwerkes. Grob kann man die Anwendungsgebiete in drei Szenarien einteilen [4]:

1) Ad-Hoc Netzwerke: Schnelle und kurzlebige Netzwerke, die meistens das Ziel haben, nur wenige bestimmte Aktionen, wie z.B. Dateitransfer, Präsentationen o.ä., auszuführen. Dabei können die Teilnehmer sehr schnell wechseln, sowie auch das Netzwerk wieder aufgelöst werden kann. Des weiteren sind hier meistens nur wenige Personen beteiligt.

2) Private Heimnetzwerke: Da immer mehr elektronische Geräte mit Netzwerktechnologien verfügbar sind und so sich in Wohnungen immer mehr solche Geräte befinden, kann eine Konfiguration eines Netzwerkes sehr viel Wissen und Zeit beanspruchen. Um dies zu umgehen, setzen bereits jetzt sehr viele Hersteller auf Zeroconf, um ein einfaches Einbinden der Geräte in das Netzwerk bereitzustellen.

Als Beispiele seien hier Netzwerkdrucker, Audiogeräte, Kameras u.v.m. genannt.

3) Große Netzwerke: Da ein herkömmliches Netzwerk oft sehr viel Aufwand mit sich bringt, verwenden Administratoren zunehmend Zeroconf, um den Aufwand zu minimieren. Besonders bei externen Mitarbeiter oder Gästen einer Firma, zahlen sich automatische Konfigurationen aus, da diese Personen oft nicht lange im Unternehmen verweilen.

III. Zero Configuration Networking

Zero Configuration Networking, in Kurzform auch Zeroconf genannt, wird am besten durch die Definition der Zero Configuration Working Group beschrieben [5]:

> 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.

Frei übersetzt bedeutet dies folgendes: Das Ziel von Zero Configuration Networking ist, Benutzern die Möglichkeit zu eröffnen, ein Netzwerk ohne Konfiguration und ohne Verwaltungsaufwand zu erstellen. Sei es nun, weil ein die Verwaltung und Erstellung eines Netzwerkes unmöglich ist, aber auch wenn es nicht praktikabel ist. Sozusagen soll Zero Configuration Networking die Erstellung von Ad-Hoc Netzen und die Verwendung derselben erleichtern, aber nicht nur darauf beschränkt werden.

Die Zeroconf Working Group hat dabei folgende drei Problembereiche identifiziert und bereits gelöst [6], [7]:

- Automatische Allokation von IP-Adressen ohne DHCP Server (IPv4 Link-Local Addressing)
- Übersetzung von Namen und IP-Adressen ohne einen DNS Server (Multicast DNS)
- Finden von Services im lokalen Netzwerk ohne einen Directory Server (DNS Service Discovery)

Des weiteren soll Zeroconf keine Störungen in bereits existierenden Netzwerken hervorrufen und für den Benutzer transparent erscheinen. Deshalb werden von der Zero Configuration Networking Working Group folgende Punkte gefordert [1], [5], [6]:

- Zeroconf darf keine Auswirkungen auf bereits bestehende Netzwerke haben, so muss beispielsweise die Adressauflösung auch bei einem bereits bestehenden DHCP Server funktionieren und das Netzwerk darf keinen Schaden davon ziehen.
- Zeroconf soll die Auswirkungen auf Anwendungen minimal halten und versuchen, so transparent wie nur möglich zu sein. So sollen bestehende Anwendungen immer noch korrekt funktionieren.
- Der Sicherheitsstandard der Zeroconf Protokolle darf nicht kleiner sein, als andere aktuelle ähnliche bzw. zugehörige IETF Protokolle, die dem IETF Standard angehören.

A. Automatische Allokation von IP-Adressen ohne DHCP Server

Der erste Problembereich von Zeroconf ist die automatische Zuweisung von IP-Adressen. Um in einem IP-basierenden Netzwerk Pakete verschicken zu können, benötigt jeder Teilnehmer eine eigene, im Netzwerk einmalige, IP-Adresse. In zentralen Netzwerken geschieht diese Zuweisung meist durch einen DHCP-Server oder durch bereits vorher für jeden Teilnehmer festgelegte Adressen.

Zeroconf vergibt die IP-Adressen jedoch zufällig und ohne Eingreifen eines Administrators. Das verfolgte Ziel hierbei ist zu bewerkstelligen, dass jeder Teilnehmer eines Netzwerkes über die jeweils anderen Teilnehmer im selben Netzwerk Bescheid weiß. Dies erfordert IP-Adressen mit besonderen Eigenschaften, die IPv4 Link-Local Adressen genannt werden [8]. Dazu gehören alle IP-Adressen im Bereich von 169.254/16 (169.254.xxx.xxx). Die ersten und letzten 256 Adressen (169.254.0.0 bis 169.254.1.0 und 169.254.254.255 bis 169.254.255.255) sind jedoch für zukünftigen Gebrauch

reserviert und dürfen nicht verwendet werden [8]. Des weiteren dürfen diese Adressen in einem Netzwerk nur einmalig vorkommen. Die Voraussetzungen für eine automatische Adressvergabe beinhaltet nun auch, dass es einem Teilnehmer möglich ist [2]

- selbstständig seine Netzwerkschnittstellen mit einmaligen Adressen zu konfigurieren
- festzustellen welche Subnetzmaske zu benutzen ist
- festzustellen, ob eine Adresse doppelt benutzt wird
- Kollisionen zu bewältigen

Die Adressvergabe wird nun mit einem auf dem Address Resolution Protocol (ARP) aufbauenden Verfahren umgesetzt. Der Teilnehmer wählt sich pseudo-zufällig aus dem oben genannten Adressraum eine IP-Adresse aus. Dabei werden rechnerspezifische Informationen, wie z.B. die MAC-Adresse der Netzwerkschnittstelle, berücksichtigt, um soweit möglich immer die gleiche IP-Adresse zu erhalten. Dies erhöht die Stabilität von Zeroconf [8].

Wenn nun aber identische Informationen benutzt werden, wie z.B. die Systemzeit zweier gleichzeitig eingeschalteter Systeme, können leicht Konflikte entstehen. Nachdem die IP-Adresse generiert wurde, muss nun also überprüft werden, ob diese nicht schon in Verwendung ist. Natürlich darf bis zu dem Punkt, wo feststeht, dass dieser Teilnehmer der einzige Besitzer dieser IP-Adresse ist, die gewählte IP-Adresse nicht veröffentlicht werden, beispielsweise durch IP- oder ARP-Pakete [9].

Des weiteren darf diese Überprüfung nur nach der Adressgenerierung durchgeführt werden und nicht wiederholt werden, um Netzwerkressourcen nicht zu verschwenden. Die Überprüfung geschieht mit Hilfe von ARP Probes. Eine ARP Probe ist ein ARP Paket in welchem die gewählte IP-Adresse als Empfänger und 0.0.0.0 als Absender eingetragen ist. Auch die Ziel-MAC-Adresse wird dabei ignoriert und mit Nullen aufgefüllt [9].

Tabelle I

IPV4 LINK-LOCAL KONSTANTEN, ENTNOMMEN AUS [8]

PROBE_WAIT	1 second	initial random delay
PROBE_NUM	3	number of probe packets
PROBE_MIN	1 second	minimum delay till repeated probe
PROBE_MAX	2 seconds	maximum delay till repeated probe
ANNOUNCE_WAIT	2 seconds	delay before announcing
ANNOUNCE_NUM	2	number of announcement packets
ANNOUNCE_INTERVAL	2 seconds	time between announcement packets
MAX_CONFLICTS	10	max conflicts before rate limiting
RATE_LIMIT_INTERVAL	60 seconds	delay between successive attempts
DEFEND_INTERVAL	10 seconds	minimum interval between defensive ARPs

Nach dem Versenden dieser ARP Probe, wartet der Teilnehmer eine zufällige Zeit zwischen 0 und PROBE_WAIT (diese und folgende Konstanten sind in Tabelle I aufgelistet) Sekunden. Nach der Wartezeit versendet der Teilnehmer PROBE_NUM ARP Probes, wobei der Zeitabstand zwischen den einzelnen Probes zwischen PROBE_MIN und PROBE_MAX betragen muss. Nach der letzten ARP Probe wartet der Teilnehmer noch ANNOUNCE_WAIT Sekunden. Hat der Teilnehmer nun zwischen dem Anfang der Überprüfung und dem Ende der Wartezeit von ANNOUNCE_WAIT eine ARP Probe empfangen, die die gewünschte IP-Adresse des Teilnehmers im Empfänger Feld des ARP Pakets enthält und dabei die MAC Adresse nicht die der Netzwerkschnittstelle des Teilnehmers entspricht, so tritt ein Konflikt auf. In diesem Fall muss nun eine neue IP-Adresse generiert werden und die Prozedur beginnt wieder von vorne. Wenn mehrere Teilnehmer die gleiche Adresse überprüfen oder ein Teilnehmer bereits diese Adresse besitzt, das, wie bereits oben erwähnt, passieren kann, so tritt dieses Szenario auf [8].

Hierbei kann nun eine Endlosschleife eintreten und das Netzwerk wird mit ARP Paketen überlastet. Um dies zu verhindern, muss jeder Teilnehmer nach MAX_CONFLICTS Konflikten seine Geschwindigkeit, mit der er seine IP-Adressen generiert und überprüft, drosseln. Reduziert wird auf eine Überprüfung pro RATE_LIMIT_INTERVAL [9].

Wurde nun jedoch kein entsprechendes ARP Paket empfangen und somit kein anderer Teilnehmer mit der gewählten IP-Adresse gefunden, so kann der Teilnehmer diese IP-Adresse für sich beanspruchen. Jetzt muss dieser noch den anderen Teilnehmern mit Hilfe von ARP Announcements bekannt geben, dass er diese Adresse für sich beansprucht. Ein ARP Announcement ist wiederum ein ARP Paket, in welchem die jeweilige beanspruchte IP-Adresse in das Empfänger- und Absenderfeld eingetragen wird. Er sendet nun ANNOUNCE_NUM ARP Announcements mit einem Abstand von ANNOUNCE_INTERVAL Sekunden. Jetzt kann und muss jeder Teilnehmer seinen Cache auffrischen und die neue Adresse eintragen. Ansonsten wäre es möglich, dass ein anderer Teilnehmer eine veraltete Adresse gespeichert hat [8].

Empfängt nun der Teilnehmer ein ARP Paket, und somit eine ARP Probe auf seine IP-Adresse, entsteht wieder ein Konflikt. Der Teilnehmer hat nun zwei Möglichkeiten: Entweder er verteidigt seine IP-Adresse oder er wählt eine neue. Sofern der Teilnehmer noch offene Verbindungen hat, beispielsweise einen Dateitransfer, wird er die Adresse meistens verteidigen. Die Verteidigung erfolgt durch das Verschicken eines ARP Announcements. Hat der Teilnehmer jedoch zuvor schon einen Konflikt feststellen können, so muss eine neue Adresse gewählt werden, um eine Endlosschleife zu vermeiden. Diese kann entstehen, wenn beide Teilnehmer versuchen ihre Adresse zu verteidigen [8].

Abbildung 1 zeigt ein beispielhaftes Zeroconf Netzwerk. Jede Netzwerkschnittstelle jedes Teilnehmers kann eine eigene, eindeutige IP-Adresse haben, obwohl sich im Netzwerk Wireless- und Kabelgebundene Schnittstellen befinden. Teilnehmer 1 und Teilnehmer 2 benutzen eine Wireless Verbindung und die Adressen A und B sind verschieden und eindeutig. Teilnehmer 2 und Teilnehmer 3 sind über ein Kabel verbunden und somit sind Adressen C und D eindeutig. Teilnehmer 2 wird nun aber auf keinen Fall bei der

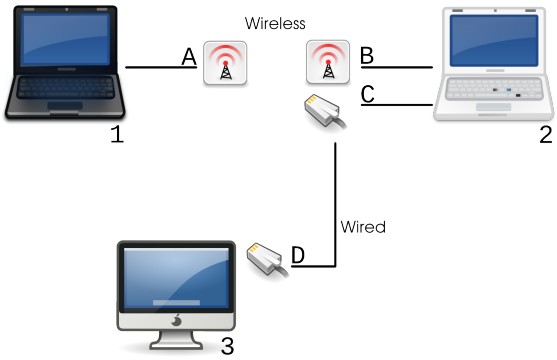

Abbildung 1. Automatische IP-Konfiguration. In diesem Schaubild besitzt jede Netzwerkschnittstelle A-D eine einmalige IP-Adresse für jede Wireless oder kabelgebundene Verbindung. [1]

kabelgebundenen Verbindung die Adresse A von Teilnehmer 1 verwenden. Allgemein wird ein Teilnehmer keine Adresse wählen, die mit irgendeinem anderen Teilnehmer auf irgendeiner Netzwerkschnittstelle kollidieren. Dies aus dem einfachen Grund, da Missverständnisse für diejenigen Teilnehmer auftreten könnten, die über mehrere Netzwerkschnittstellen mit dem Netzwerk verbunden sind.

Dass das Verfahren trotzdem gut skaliert, zeigen einige Experimente, wie z.B., dass die Wahrscheinlichkeit eine noch nicht benutzte Adresse in einem Netzwerk von 1300 Teilnehmer nach dem zweiten Versuch zu wählen, auf bis zu 99.96% anwachsen kann [10].

Bisher wurde nur IPv4 betrachtet, IPv6 bringt im Gegenteil zu IPv4 die automatische Adressierung von Haus aus schon mit. Dies gehört aber nicht explizit zu Zeroconf, deshalb wird nur kurz darauf eingegangen. Bei IPv6 befinden sich die Link-Local Adressen im Bereich fe80::/64. Die Adressen werden aus der MAC-Adresse der Netzwerkschnittstelle und dem fe80-Prefix erstellt. Des weiteren fällt die Konfliktüberprüfung weg, da jede MAC Adresse bereits eindeutig ist. Ein weiterer Punkt ist das Faktum, dass Link-Local IPv6-Adressen nicht geroutet werden dürfen, da sie ja schon eindeutig sind. Die automatische Adressierung wird in [11] genauer beschrieben.

B. Übersetzung von Namen und IP-Adressen ohne einen DNS Server

Da bekanntermaßen Namen besser gemerkt werden können als Zahlen, wäre es sehr unhandlich, andere Teilnehmer über deren IP-Adresse anzusprechen. Deshalb benutzt man das Domain Name System (DNS) um Namen auf IP-Adressen umzuwandeln und umgekehrt.

[1]Diese und folgende selbst erstellten Abbildungen benutzen Werke aus dem Tango Project (`http://tango.freedesktop.org/Tango_Desktop_Project`): *gnome-icon-theme-extras* unterliegt der GPL. *tango-icon-theme* ist unter Public Domain veröffentlicht.

In den meisten Netzwerken befinden sich aus diesem Grund auch DNS Server, die diese Tätigkeit ausüben. Aber gerade an Orten, wie z.B. bei Konferenzen oder Flughäfen ist es sehr unwahrscheinlich ein bereits konfiguriertes Netzwerk mit DNS Server vorzufinden. Das Fehlen solcher DNS Server führt natürlich zu dem Problem, dass eine Umwandlung von Namen zu IP-Adressen und umgekehrt nicht mehr möglich ist.

Hierfür gibt es zwei sehr ähnliche Lösungen:

- Link-Local Multicast Name Resolution (LLMNR) von Microsoft, das jedoch kaum bis keine Verwendung findet und erst kürzlich als RFC beschrieben wurde [12].
- Multicast DNS (mDNS) von Apple, das offen beschrieben wurde und Bestandteil von Zeroconf ist.

Aus diesen naheliegenden Gründen wird nur auf mDNS eingegangen. Multicast DNS schlägt, wie der Name bereits sagt eine geringfügige Änderung von DNS vor, nämlich Multicast. Dies bedeutet einfach eine andere Verfahrensweise wenn ein Teilnehmer eine DNS Abfrage schicken möchte, die gleich beschrieben werden. Multicast bedeutet hierbei, dass eine Nachricht von einem Teilnehmer eines Netzwerkes gesendet wird und von einer Gruppe von Teilnehmern des Netzwerkes empfangen wird. So kann jeder Teilnehmer, der Interesse an dieser Nachricht zeigt, diese empfangen. Um dies im Netzwerk zu realisieren, müssen solche Nachrichten an die IPv4-Adresse 224.0.0.251 (IPv6: ff02::fb) geschickt werden [13].

Grundsätzlich gibt es bei Zeroconf eine neue Top Level Domain (TLD), nämlich `.local`. Alle Domains mit dieser Endung sind frei verfügbar und können auch nicht, wie andere Domains, erworben werden. So kann sich jeder Teilnehmer eine Domain aus der `.local`-Domäne auswählen. Einzig zu beachten dabei ist, dass kein bereits vergebener Name benutzt werden soll. Dass dies zu Konflikten führen kann, ist offensichtlich. Von der Zeroconf Working Group wurde dieser Fall jedoch absichtlich nicht beschrieben, da es zum einen eher unwahrscheinlich scheint, dass zwei Teilnehmer den gleichen Namen auswählen. Zum anderen, auch weitaus wichtigeren Punkt, kann es für solche Mehrfachvergaben auch sinnvolle Anwendungen geben. Als Beispiel sei hier Load Balancing genannt [13].

Jeder Teilnehmer wählt nun also einen Fully Qualified Domain Name (FQDN), z.B. *macbook.local*, beim Betreten des Netzwerkes und kündigt diesen im Netzwerk an. Dadurch hat der Teilnehmer Besitzansprüche auf diese Domain. DNS Anfragen werden nun in das Netzwerk über Multicast geschickt und der Teilnehmer, der für die angefragte Domain zuständig ist, antwortet wiederum über Multicast. So kann nun jeder Teilnehmer im Netzwerk seinen Cache aktualisieren [10].

Auch Reverse-DNS Anfragen, die im Gegensatz zu DNS zu einer IP den Namen des Besitzers der IP erfahren möchten werden über oben genannte Multicast Adresse versandt [9].

Nun können solche DNS Anfragen eine hohe Netzwerklast einfordern und so wurden von der Zeroconf Working Group folgende Maßnahmen vorgeschlagen, welche zu einer Reduzierung des Traffics über das Netzwerk führen sollen. Dadurch sollen doppelte oder mehrfache DNS Anfragen verhindert werden [13].

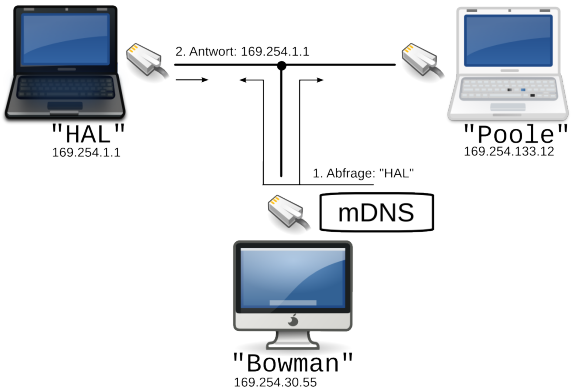

Abbildung 2. Multicast DNS. Der Teilnehmer „Bowman“ möchte die IP-Adresse von „HAL“ erfahren. Dazu schickt er eine Multicast DNS Anfrage an die bekannte Multicast Adresse, die dann jeder Teilnehmer im Netzwerk empfängt. Wenn „HAL“ im Netzwerk existiert, so antwortet dieser, was im Schaubild zu sehen ist. Gleichzeitig aktualisiert „Poole“ seinen Cache mit der Antwort von „HAL“. Aus Gründen der Übersichtlichkeit wurde der Suffix .local nicht an die Namen angefügt, jedoch endet jeder FQDN mit .local.

1) Known Answer Suppression: Wenn ein Teilnehmer eine Multicast DNS Anfrage verschicken möchte, zu der er schon einige Antworten in seinem Cache hat, so fügt er diese Antworten an seine Anfrage an.

Der Besitzer muss nun nicht auf diese Anfrage antworten, wenn die korrekte Antwort bereits in der DNS Abfrage enthalten ist und die Time To Live (TTL) dieser Antwort noch größer ist, als die Hälfte der üblichen TTL. Ist die TTL jedoch kleiner, so muss dieser die Anfrage beantworten, um die TTL der Caches zu aktualisieren.

Da der Besitzer eine Antwort verschicken muss, falls die TTL zu klein ist, ist dem Teilnehmer, der eine Abfrage verschickt, nicht erlaubt solche Einträge mitzuschicken, deren TTL schon kleiner als die Hälfte der üblichen TTL ist.

Des weiteren dürfen die restlichen Teilnehmer die Antworten dieser Multicast DNS Abfragen nicht in ihren Cache aufnehmen, da diese nicht vom Besitzer der Domain stammen und so möglicherweise schon veraltet oder falsch sein könnten.

2) Multi-Packet Known Answer Suppression: Falls nun ein Teilnehmer zu einer Abfrage mehr Antworten besitzt, als in einer Multicast DNS Abfrage Platz finden, so muss dieser die Abfrage senden und so viele Antworten in das Paket geben, wie Platz verfügbar ist. Danach setzt dieser das Truncated (TC) Bit und schickt eine nächste DNS Abfrage jedoch ohne Interesse an einer Domain, aber mit den restlichen Antworten, die der Teilnehmer im Cache gespeichert hat. Dies macht er solange, bist alle Antworten verschickt wurden. Beim letzten Paket wird natürlich das TC Bit nicht mehr gesetzt.

Der Besitzer wartet nun eine zufällige Zeit von 400 bis 500ms zwischen den Paketen, um dann diese Abfrage zu beantworten. Ist nun eine der Antworten in den Paketen dieselbe, die der Besitzer geben würde, so löscht er diese aus der Abfrage und beantwortet diese nicht. Falsche oder fehlende Antworten werden aber trotzdem mit der richtigen Antwort beantwortet.

3) Duplicate Question Suppression: Wenn ein Teilnehmer eine Abfrage verschicken möchte und ein anderer Teilnehmer in diesem Moment die gleiche Abfrage bzw. eine ähnliche, die jedoch auf die gleiche Antwort hinausläuft, so soll dieser Teilnehmer die Abfrage des anderen Teilnehmers als seine eigene betrachten und so redundanten Netzwerkverkehr vermeiden. Er wartet also auf die Antwort, die über Multicast DNS ja sowieso alle Teilnehmer erreicht.

4) Duplicate Answer Suppression: Wenn ein Besitzer einer Domain gerade eine Antwort vorbereitet und eine Abfrage eines Teilnehmers erfolgt, die die selben Antworten und eine TTL, die jedoch mindestens gleich groß ist, wie die TTL in der Antwort des Besitzers beinhaltet, so versendet der Besitzer diese Antwort nicht und nimmt die Antwort als gegeben an.

C. Finden von Services im lokalen Netzwerk ohne einen Directory Server

Um einen Service benutzen zu können, muss ein Benutzer zuallererst wissen, wo sich der Service befindet und welches Protokoll dieser benutzt. Nun gibt es aber keine Directory Server in unserem Netzwerk, weshalb eine andere Lösung verwendet werden muss.

Die Zeroconf Working Group schlägt DNS Service Discovery (DNS-SD) als Lösungsansatz vor. DNS-SD setzt auf mDNS auf, kann jedoch auch auf „normalen“ DNS Servern operieren [14].

DNS-SD erweitert DNS, sodass es möglich ist, auch Services abfragbar zu machen. DNS bietet sich gerade deshalb an, weil bereits mit mDNS und DNS zwei Lösungen für lokale Netzwerke und solche mit DNS Servern existieren.

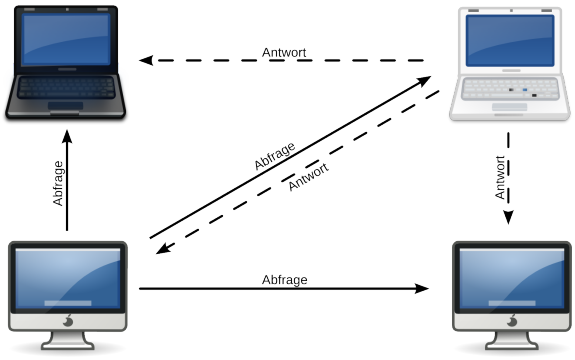

Abbildung 3. Der Teilnehmer (in der Abbildung links unten) stellt eine Anfrage für ein bestimmten Service, z.B. _http._tcp.local, die über Multicast an alle Teilnehmer versandt wird. Derjenige, der nun in seinen Resource Records diesen Service gespeichert hat, schickt eine Antwort, wiederum über Multicast, zurück, wo der Service zu finden ist.

Die Anforderungen sind hierbei, dass es zum einen möglich sein muss, Services eines bestimmten Typs in einer bestimmten Domain zu suchen. Diese Information soll nun dazu verwendet werden, die IP-Adresse und den Port des Services herauszufinden. Des weiteren müssen die Services auffindbar bleiben, auch wenn sich die IP-Adresse oder der Port verändern.

DNS-SD benutzt nun DNS SRV Resource Records (RR) [15], um Services zu finden. DNS SRV Resource Records propagieren per DNS Services, die unter der aktuellen Domain verfügbar sind. Dazu haben DNS SRV RR ein spezielles Format, das in Tabelle II dargestellt ist [15].

Tabelle II
AUFBAU SRV RESOURCE RECORDS [15]

```
_Service._Proto.Name TTL Class SRV Priority Weight Port
```

Service	Der symbolische Name des Services.
Proto	Das Protokoll des Services.
Name	Die Domain für diesen RR.
TTL	Gibt an, wie lange der RR im Cache gehalten werden darf.
Class	Die DNS Klasse, immer auf „IN“ für Internet.
Priority	Die Priorität dieses Services, falls mehrere identische Services angeboten werden. Je kleiner diese Zahl, desto höhere Priorität hat der Service.
Weight	Das Gewicht des Services, falls mehrere Services die gleiche Priorität haben. Je höher das Gewicht, desto wichtiger ist der Service.
Port	TCP oder UDP Port.
Target	Der Canonical Name (CNAME) des Rechners, der diesen Dienst anbietet.

Ein SRV RR für einen Webserver auf `www.example.com`, der auf Port 80 läuft mit Priorität 10 und Gewicht 0 ist beispielhaft in Tabelle III dargestellt.

Tabelle III
BEISPIELHAFTE DARSTELLUNG EINES SRV RESOURCE RECORDS

```
_http._tcp.example.com 3600 IN SRV 10 0 80 ↩
                                        www.example.com.
```

Es ist auch möglich, mehrere SRV RR für denselben Service einzutragen um beispielsweise Load Balancing durchzuführen. So kann man einfach eine Anfrage auf `_http._tcp.example.com` stellen und bekommt alle Webserver in dieser Domäne.

Wie bereits erwähnt wird jeder Service eindeutig durch `Service.Protocol.Domain` identifiziert. Hervorzuheben ist hierbei, dass man für jeden Service mehrere Instanzen haben kann, z.B. `laser_printer._ipp._tcp.example.com`. Dabei handelt es sich um einen UTF-8 kodierten Text, der im DNS Paket mitgeschickt werden kann. Aufgrund dessen und dem Punkt, dass bei der Entwicklung dieser Protokolle auf keine Kompatibilität Wert gelegt werden musste, entstehen keine Limitierungen bei der Namensvergebung [3]. Es kann also ein beliebiger Text, wie z.B. „Drucker im 2. Stock, pro Blatt 0,3€ in die Kasse!“ als Bezeichnung der Instanz dienen. Außerdem lautet die Domain in einem Ad-Hoc Netzwerk `.local`, d.h. `_service._prot.local` liefert alle Services von „service“ über das Protokoll „prot“ im lokalen Netzwerk zurück.

Weiters kann bei der Abfrage noch ein TXT Record abgefragt werden, der zusätzliche Informationen für den Service bereitstellt. Informationen werden darin in Key/Value Paaren abgelegt und der Eintrag kann bis zu einer Größe von 65535 Bytes befüllt werden. Die darin enthaltenen Informationen sind optional und natürlich für jeden Service unterschiedlich. Trotzdem muss ein TXT Record zu einem SRV RR vorhanden sein, der mindestens ein Byte beinhaltet, z.B. das Nullzeichen [14]. Beispielsweise würde man sich bei einem Druckservice wünschen, die Papiergröße, und die Telefonnummer des Supports zu kennen, bevor der Druckauftrag begonnen hat.

Neben DNS-SD gibt es noch mehrere andere Dienste, wie SLP (Service Location Protocol) oder SSDP (Simple Service Discovery Protocol), die jedoch nicht von der Zeroconf Working Group akzeptiert wurden und wenig Verwendung finden [16] beschreibt das SLP Protokoll und [17] SSDP.

IV. IMPLEMENTIERUNGEN

Da die Zeroconf Protokolle offen spezifiziert sind und jedem freien Zugang ermöglichen, gibt es zahlreiche Implementierungen. Es wird nun nur auf die beiden wichtigsten kurz eingegangen und so sind diese beiden in Tabelle IV dargestellt.

V. SICHERHEIT

Auto-Konfigurationsprotokolle, wie die Allokation von IP-Adressen, die Übersetzung von Namen in IP-Adressen und das Finden von Services im lokalen Netzwerk sind nicht sicher, da keine Sicherheitsmechanismen eingebaut wurden. [2] Aus diesem Grund konnten die Protokolle einfach gehalten werden, sind aber auch gegen Angriffe verwundbar.

Auch wenn Automatisierbarkeit eigentlich Zeroconf ausmacht und eines der Ziele davon ist, kann Sicherheit nicht automatisiert werden [2]. Wie bereits vorher erwähnt, dürfen aber Zeroconf Protokolle nicht weniger sicher sein, als vergleichbare IETF-Standards. Trotzdem kann Zeroconf einen weiteren Zugang zu einem Netzwerk bedeuten und soll besonders dann, wenn weitere Netzwerke daran grenzen abgesichert oder nicht benutzt werden.

Die Zeroconf Working Group ist sich diesem Problem bewusst und hat bereits mehrere Mechanismen für die Absicherung von Zeroconf besprochen, jedoch wurde nach [2] noch keine genaue Spezifikation vorgeschlagen.

Nun ist aber auch die Nichtbenutzung von Zeroconf kein Gewinn für die Sicherheit, denn Angreifer wissen bereits, wie man Teilnehmer und Services eines Netzwerkes auffinden kann, auch wenn kein Zeroconf benutzt wird [7]. So kann man sagen, dass Zeroconf nur dem Benutzer hilft, einfacher ein Netzwerk zu erstellen und zu benutzen.

VI. ZUSAMMENFASSUNG UND AUSBLICK

Die drei großen Bereiche von Zeroconf wurden vorgestellt, die Automatische Allokation von IP-Adressen (IPv4 Link-Local Addressing), die Übersetzung von Namen und IP-Adressen (Multicast DNS) und das Finden von Services im lokalen Netzwerk (DNS Service Discovery).

Tabelle IV
ZWEI DER WICHTIGSTEN ZEROCONF IMPLEMENTATIONEN

Bonjour

Autor:	Apple Inc.
Lizenz:	Apple Inc. - Proprietäre Freeware. Teile unter der Apache License, Version 2.0.
OS:	Mac OS X, Microsoft Windows
Webseite:	`http://developer.apple.com/bonjour`
Beschreibung:	Bonjour [18] (ehemals auch Rendezvous genannt) ist eine Zeroconf Implementierung, die von Apple Inc. entwickelt wird. Beteiligt ist dabei auch Stuart Cheshire, der einer der leitenden Köpfe der Zeroconf Working Group ist. Bonjour ist im Mac OS X Betriebssystem seit Version 10.2 enthalten und kann des weiteren auch auf Microsoft Windows Systemen installiert werden. Teile von Bonjour, wie der mDNSResponder sind auch unter *BSD und GNU/Linux verfügbar und lauffähig.

Avahi

Autor:	Lennart Poettering und Trent Lloyd
Lizenz:	LGPL
OS:	GNU/Linux, Solaris, Mac OS X, *BSD
Webseite:	`http://www.avahi.org`
Beschreibung:	Avahi ist eine freie Implementierung von Zeroconf, die unter der LGPL veröffentlicht ist. Seit 2004 wird Avahi entwickelt (vormals FlexMDNS) und ist momentan eine der besten Implementierungen von Zeroconf [3]. Es unterstützt IPv4LL, mDNS and DNS-SD und ist der de-facto Standard unter den meisten Linux und *BSD Distributionen [19].

Zeroconf eignet sich besonders für Ad-Hoc Netzwerke, Heimnetzwerke und kleine Firmennetzwerke. Besonders auch dann, wenn die beteiligten Personen wenig Fachwissen über dieses Gebiet mitbringen.

Aber auch aus dem Grund, da Zeroconf mit einem bereits bestehenden Netzwerk keine Probleme hat, kann sich der Administrator Arbeit sparen, wenn etwa einige neue Drucker und Dateiserver in das Netzwerk eingebunden werden sollen. Hier muss er etwa diese Geräte nur dem Netzwerk hinzufügen. Das Bereitstellen der Services geschieht dann automatisch. Zu beachten ist hier einzig, dass sich das Verkehrsaufkommen erhöhen wird.

Da Zeroconf keine wesentliche Sicherheitsmechanismen mitbringt, ist die Wahrscheinlichkeit sehr hoch, dass nicht befugte Benutzer diesem Netzwerk beitreten. Hier benötigt man zusätzliche Sicherheitsmechanismen, und sollte ohne diese keine kritischen Aufgaben erledigen.

Zeroconf ist nun bereits Standard für Auto-Konfigurationsnetzwerke, Apple besitzt eine sehr gute Integrierung von Zeroconf in seinen Produkten, GNU/Linux und restliche Unix-Systeme haben auch eine sehr gute Implementierung von Zeroconf, nämlich Avahi, und eine teilweise so gute Implementierung in bestimmten Applikationen. Windows hat nur Apple's Bonjour zur Verfügung und unterstützt Zeroconf nur teilweise von Haus aus. So bleibt abzuwarten, ob sich die Windows-Welt diesem Trend anschließt.

Ein weiterer Punkt ist, ob Sicherheitstechnologien in Zeroconf in naher Zukunft noch aufgenommen werden. Dies scheint aber wegen der Auflösung der Zeroconf Working Group [5] unwahrscheinlich.

ACKNOWLEDGMENT

Vielen Dank an Lennart Poettering, dem Entwickler von Avahi, für die Beantwortung zahlreicher Fragen.

LITERATUR

[1] Zeroconf Working Group, "Official website of the Zeroconf Working Group," http://www.zeroconf.org, March 2009.
[2] E. Guttman, "Autoconfiguration for IP Networking: Enabling Local Communication," *IEEE Internet Computing*, vol. 5, no. 3, pp. 81–86, May/June 2001.
[3] S. Cheshire, "Stuart Cheshire speaks about Zeroconf at Google," Google TechTalks, http://video.google.com/videoplay?docid=-7398680103951126462, November 2005.
[4] T. Kollbach and C. Beier, "Zero Configuration Networking," http://www2.informatik.hu-berlin.de/~beier/txts/2007-Rechnerkommunikation%20--%20Zeroconf%20Networking.pdf, July 2007.
[5] Zeroconf Working Group, "Description of the Zeroconf Working Group," http://www.zeroconf.org/zeroconf-charter.html, March 2009.
[6] A. Williams, "Requirements for Automatic Configuration of IP Hosts," IETF Internet-Draft, March 2003.
[7] IT-University of Gothenburg, "Zero Configuration Networking," Design of Complex Systems, http://pop.blandband.se/cv-files/ZeroConfArticle.pdf, 2005.
[8] S. Cheshire, B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses," RFC 3927 (Proposed Standard), May 2005.
[9] G. Stamboliev, "Zeroconf," http://wwwspies.informatik.tu-muenchen.de/lehre/seminare/SS05/hauptsem/Ausarbeitung03.pdf, 2005.
[10] T. Niemueller, "Zero Configuration Networking," *Lecture Notes in Informatics (LNI) - Seminars*, vol. S-3, pp. 143–146, 2006.
[11] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Proposed Standard), September 2007.
[12] B. Aboba, D. Thaler, and L. Esibov, "Link-Local Multicast Name Resolution (LLMNR)," RFC 4795 (Proposed Standard), January 2007.
[13] S. Cheshire and M. Krochmal, "Multicast DNS," IETF Internet-Draft, September 2008.
[14] ——, "DNS-Based Service Discovery," IETF Internet-Draft, September 2008.
[15] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," RFC 2782 (Proposed Standard), February 2000.
[16] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," RFC 2608 (Proposed Standard), June 1999.
[17] Y. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright, "Simple Service Discovery Protocol/1.0," IETF Internet-Draft, October 1999.
[18] Apple Inc., "Bonjour project site," http://developer.apple.com/networking/bonjour, March 2009.
[19] Avahi Project, "Avahi project site," http://www.avahi.org, March 2009.
[20] S. Cheshire, "IPv4 Address Conflict Detection," RFC 5227 (Proposed Standard), July 2008.