

# Stateless Autoconfiguration, dynamische DNS-Updates und viele ungeahnte Möglichkeiten

Benedikt Stockebrand

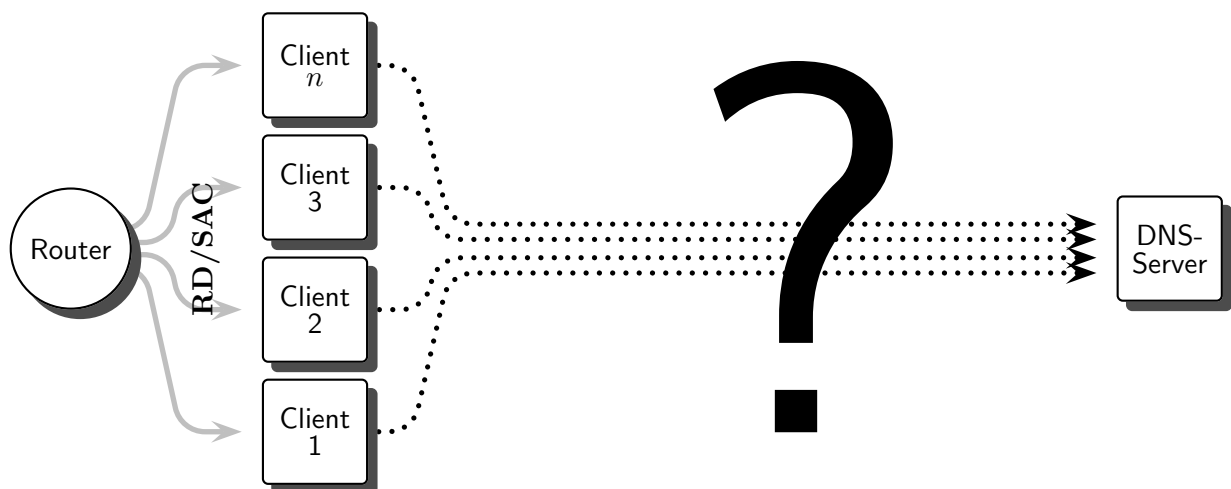
12. Oktober 2004

## Zusammenfassung

Auch wenn es grundsätzlich kein unlösbares Problem ist, per Stateless Autoconfiguration eingerichtete IPv6-Adressen sicher ins DNS einzutragen, gibt es bisher anscheinend keine produktions-taugliche Implementierung. In diesem Vortrag wird eine recht neue, nach heutigem Stand noch prototypische Implementierung vorgestellt. Die geplanten nächsten Schritte mit dem Ziel, die Implementierung auch für den unbeaufsichtigten Einsatz in produktiven Umgebungen vorzubereiten, werden aufgezeigt. Darauf aufbauend werden einige weiterführende Ideen vorgestellt, die das Problem zu einem universelleren Mechanismus generalisieren. Schließlich werden einige ursächliche Probleme herausgearbeitet, die zu dieser unschönen Situation geführt haben.

## 1 Was war

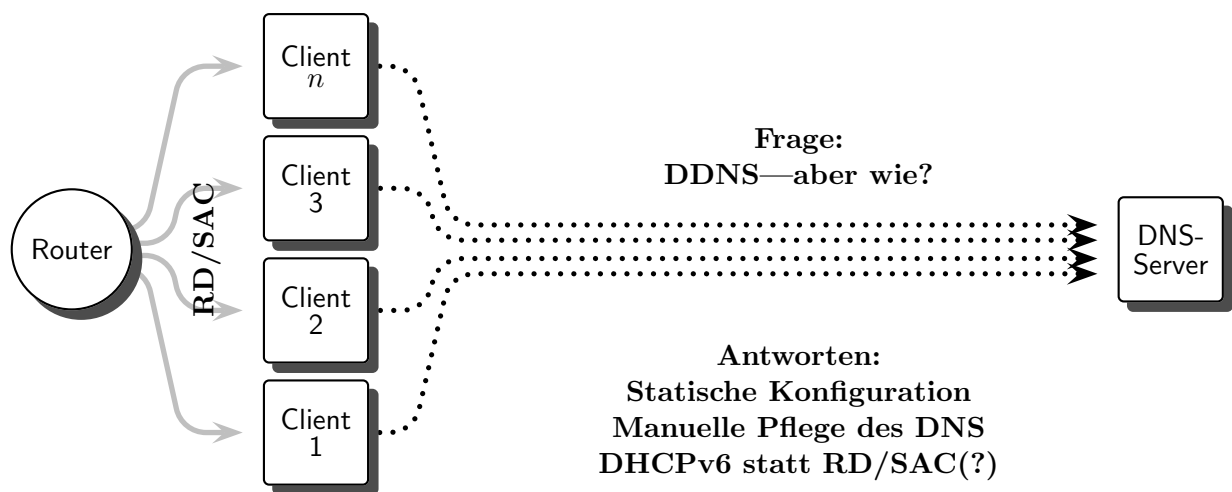
Vor einem knappen Jahr, während der Vorbereitung eines IPv6-Tutorials für das Frühjahrsfachgespräch 2004 der GUUG (German Unix User Group), fiel mir schmerzhaft auf, daß es anscheinend keinen etablierten Weg gibt, die per Stateless Autoconfiguration (SAC) konfigurierten IPv6-Adressen automatisch ins DNS einzutragen. Aus der Not des anstehenden Termins heraus entwickelte ich einige prototypische Scripts, die diese Aufgabe notdürftig implementieren.



Stateless Autoconfiguration, dynamische DNS-Updates und viele ungeahnte Möglichkeiten

Als ich im Februar 2004 noch immer weit und breit keine etablierte Lösung für dieses Problem gefunden hatte, beschloß ich, mich vor der versammelten JOIN-IPv6-Mailingliste zu blamieren, und fragte die Liste nach ihren „Best Practices“. Die Diskussion artete aus, von der Frage ob ich denn nicht doch lieber statisch konfigurierte Adressen verwenden oder die DNS-Zones von Hand nachpflegen wollte, über den Sinn und Unsinn von DHCPv6 zur Resolver- und NTP-Konfiguration bis zur Diskussion, wie weit RFC 3041-Adressen etwas im DNS verloren haben oder nicht. Aber eine zufriedenstellende Antwort konnte mir niemand geben.

Allmählich wurde klar: Daß ein Haufen Leute an mobilen Adhoc-Netzen herumspezifizierten bedeutet nicht, daß alle fundamentalen Basisfunktionalitäten rund um IPv6 schon implementiert sind.



## 2 Was ist

Alle Mechanismen, um das Problem zu lösen, sind seit langem spezifiziert, implementiert und allgemein verfügbar. Nur hat sie bisher niemand zu einer produktionstauglichen Lösung zusammengefaßt.

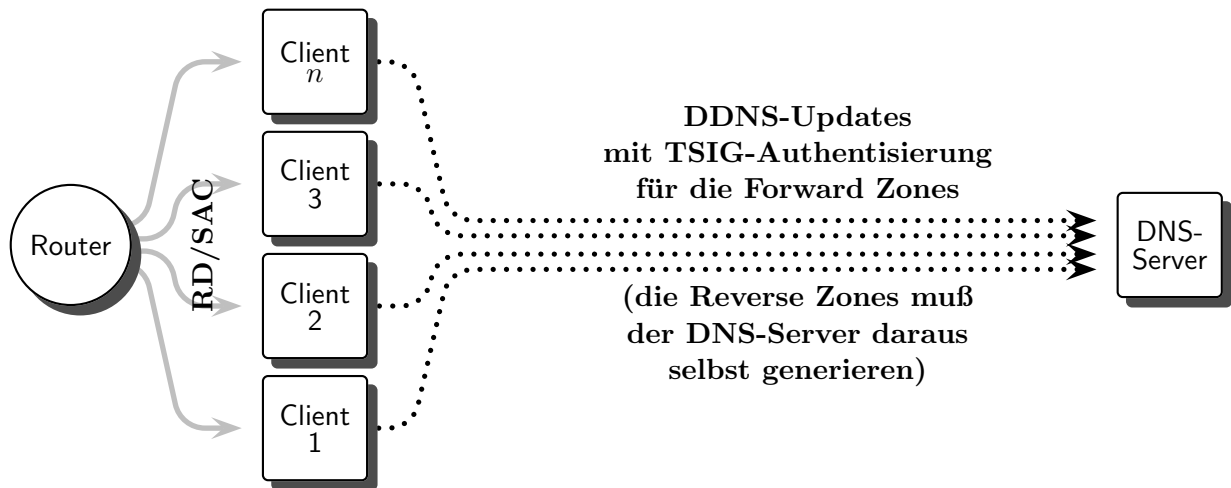
Der naheliegende Lösungsansatz ergibt sich so aus:

- Jeder Node überwacht, ob sich seine IPv6-Adressen ändern. (Mangels allgemein verfügbarer standardisierter Schnittstellen zum Kernel muß dazu bis auf weiteres per „`ifconfig -a`“ regelmäßig überprüft werden, ob sich die Konfiguration verändert hat.)
- Außerdem überwacht er „verdachtsunabhängig“, ob seine DNS-Einträge noch korrekt sind. (Auch hier ist es bisher nicht zu vermeiden, per Polling regelmäßig den aktuellen Stand abzufragen; ein entsprechendes Notify-Feature unterstützt DNS nicht.)
- Wenn der Node feststellt, daß seine DNS-Einträge nicht stimmen, aktualisiert er sie per dynamischem DNS-Update.

- Der entsprechende Name Server nimmt das Update an und aktualisiert seine Zones mit den Daten aus dem Update.

Auch wenn Microsoft diesen Ansatz anscheinend in dieser Form implementiert hat, bleibt an diesem Punkt noch ein nicht ganz unwesentliches Problem: Wie verhindere ich, daß ein gefälschtes Update dazu benutzt wird, einen DNS-Namen mit einer „feindlichen“ Adresse zu assoziieren? Denn die DNS-Updates werden per UDP verschickt und sind damit trivial zu spoofen.

Auch darauf gibt es eine Antwort, die allerdings Nebeneffekte hat: DDNS kann mit sogenannten „Transaction Signatures“ (TSIGs) Updates authentisieren. Diese TSIGs sind, anders als der Name vermuten läßt, Shared Secrets und müssen deshalb auf beiden Seiten eingerichtet werden. Mit einer entsprechend restriktiven Update-Policy kann jede TSIG nur den DNS-Record verändern, auf deren Namen sie ausgestellt ist.



Die Keys werden auf dem Name Server mit dem Befehl „`dnssec-keygen`“ und einer länglichen Folge von Kommandozeilen-Optionen generiert, müssen dann von Hand in der `named`-Konfiguration eingetragen werden und der `named` muß dazu bewegt werden, seine Konfiguration neu einzulesen. Mit einem `include`-Statement in der `named.conf` und einem recht simplen Shell Script, `maketsigkey`, läßt sich dieser ganze Vorgang recht einfach automatisieren und ein einfacher Aufruf „`maketsigkey www.example.com`“ reicht aus, um eine neue TSIG zu generieren und im Name Server zu hinterlegen.

Kopiert man jetzt die TSIG-Dateien auf den Client, kann der seine Adresse selbst aktualisieren, ohne darüber hinausgehenden Schaden im DNS anrichten zu können. Auch dazu reicht ein Shell Script aus, das regelmäßig per „`ifconfig -a`“ und „`dig www.example.com. AAAA`“ seine tatsächlichen Adressen mit denen im DNS vergleicht und wenn nötig mit der TSIG ein Update initiiert.

Tatsächlich stößt dieses Script aber an einigen Stellen auf unschöne Probleme: Zunächst ist die Frage, welche der gefundenen Adressen tatsächlich ins DNS eingetragen werden sollen. Die Loopback-Adresse ist offensichtlich irrelevant, aber Link-Local-Adressen wären ja vielleicht bei der Fehlersuche mit einem Packet Sniffer hilfreich, wenn sie in einem lokalen Name Server abgelegt würden. Site-Local-Adressen

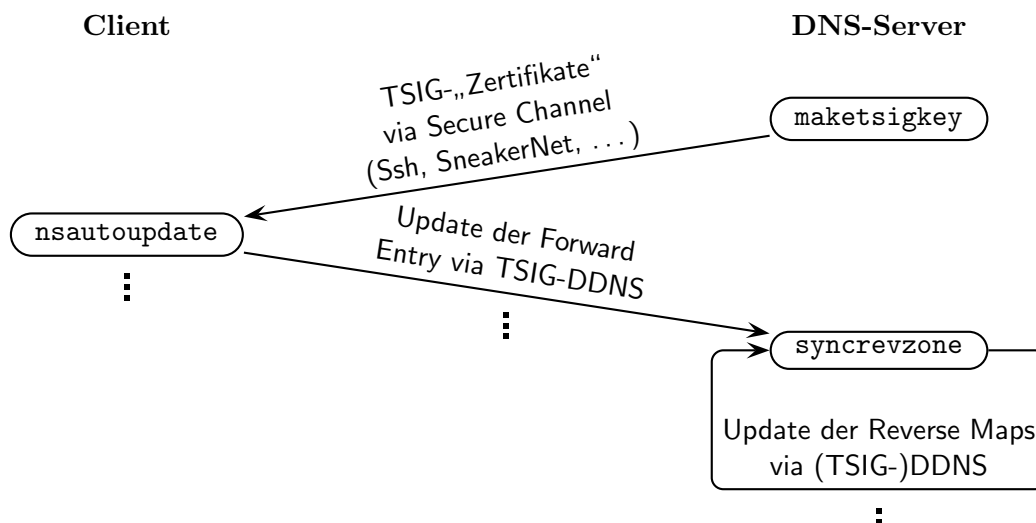
oder die hoffentlich bald kommenden Unique-Local-Adressen sind möglicherweise auch interessant, vor allem, wenn der entsprechende Node keine globalen Adressen hat, weil er nur intern sichtbar sein soll. Bei globalen Adressen, von denen es spätestens bei einem Renumbering mehrere gibt, stellt sich die Frage, welche eingetragen werden soll. Alle? Alle mit einer positiven Preferred Lifetime? Nur die mit der größten Preferred Lifetime? Aber wie komme ich dazu portabel und aus einem Shell Script an die Preferred Lifetime?

Eine pragmatische Natur und der näherkommende Konferenztermin ließen nur eine Entscheidung zu: Das Interface und der gewünschte Scope wird per Konfiguration festgelegt und die erste passende Adresse, die ein `ifconfig` liefert, trägt das Script, `nsautoupdate`, im DNS ein.

Alles andere wurde auf später vertagt.

Mit den restriktiven TSIG-authentisierten Updates handeln wir uns ein weiteres Problem ein: TSIGs werden immer auf einen festen Record Name ausgestellt. Für die Forward Zones ist das in unserem Sinn, aber für die Reverse Zones müßten wir die TSIGs auf die Adresse ausstellen—und genau die kann sich ja ändern. Damit ist der bisherige Ansatz für die Reverse Zones nicht verwendbar.

Wenn aber ein Node seine Adresse in der Forward Zone aktualisieren kann, können wir von einer zentralen Stelle aus die Forward und Reverse Zones miteinander abgleichen und Änderungen in die Reverse Zones übernehmen. Erste Versuche mit einem Shell Script `syncrevzone` führten schnell zur Einsicht, daß hier mindestens ein Perl Script nötig ist. Weitere Versuche mit Perl brachten neben einer notdürftig funktionierenden Implementierung die neue Erkenntnis, daß wohl eher C oder C++ das Mittel der Wahl ist. Die anschließenden Experimente mit C++ und der resultierenden, nicht mehr ganz so notdürftigen, Implementierung zeigten schließlich, daß die Resolver-Library keine brauchbaren Funktionen für Zone Transfers oder dynamische Updates mitbringt. Die Kombination aus `fork(2)` und `exec(2)`, um über den Umweg mit `dig` und `nsupdate` eine Neuimplementierung des Resolvers zu vermeiden zeigte weiter, daß das Ausgabeformat von `dig` sich anscheinend gelegentlich von einer Version zur anderen undokumentierterweise ändert, während `nsupdate` manche Probleme gar nicht und andere nur durch undokumentierte Exit Codes zurückmeldet.



Auch wenn diese drei Programme noch weit davon entfernt sind, als „produktionstaugliche“ Lösung eingesetzt zu werden, zeigen sie doch, daß das Problem lösbar ist.

### 3 Was wird

Als nächstes Ziel müssen also die letzten beiden Programme, `nsautoupdate` und `syncrevzone`, so weiterentwickelt werden, daß sie auch unbeaufsichtigt in einer produktiven Umgebung eingesetzt werden können.

Für `nsautoupdate` ist eine Neuimplementierung in C fast unvermeidbar. Erste Recherchen haben aber schon weitere unschöne Erkenntnisse zu Tage gebracht: Es gibt keine standardisierte Möglichkeit, an die Preferred und Valid Lifetimes einer Adresse zu kommen. Sofern nicht in den Tiefen von RFC 3484 eine Lösung versteckt ist, wird es sich nicht vermeiden lassen, hier die Eigenarten der jeweiligen Stack-Implementierung zu rekonstruieren und plattformabhängige Wege zu finden, wie man an diese Werte herankommt.

Darüber hinaus ist noch die Frage offen, welche Adressen tatsächlich eingetragen werden sollen, welche Auswahlvarianten praktisch relevant sind und wie man die entsprechende Konfiguration möglichst einfach gestaltet.

Das `syncrevzone`-Programm muß auf Dauer ohne den Umweg über `dig` und `nsupdate` direkt mit dem Name Server kommunizieren. Dazu ist es nötig, Teile des Resolvers neu zu implementieren. Schließlich sind noch einige Feinarbeiten am Logging und an den Debug-Möglichkeiten nötig. Eine weitere komplette Neuimplementierung läßt sich aber hoffentlich vermeiden.

### 4 Was werden könnte

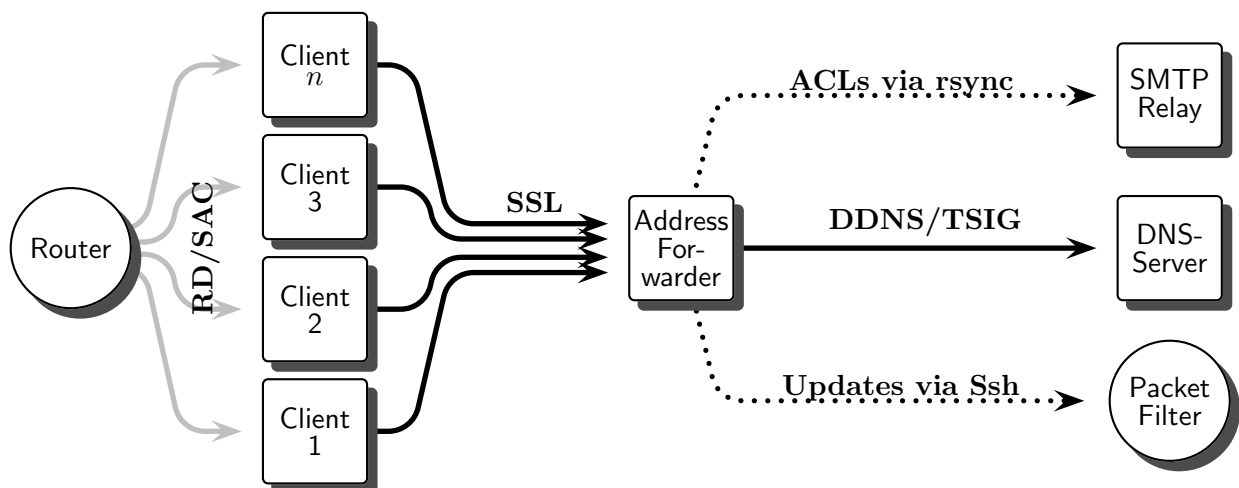
Der vorgestellte Ansatz hat meiner Meinung nach einige grundsätzliche Schwächen, die sich aus der Verwendung von DNS-spezifischen Verfahren einerseits und der Verwendung von Shared Secrets andererseits ergeben.

Deshalb stellt sich langfristig die Frage, ob nicht ein anderer Ansatz sinnvoll ist, der ganz allgemein dafür sorgt, daß sich ändernde Adressen korrekt mitverwaltet werden.

- Die Verwendung von DNS-spezifischen „Zertifikaten“ führt zu einem unnötigen administrativen Aufwand, wenn der gleiche Client an anderer Stelle andere Zertifikate zur Authentisierung benutzt. Auf Dauer ist eine generalisierte Lösung erstrebenswert, mit der sich ein Client allgemein authentisieren kann. Offensichtlicher Kandidat für diesen Zweck ist ein X.509-Zertifikat, das ein Client auch für andere Aufgaben verwenden kann.
- Mit echten Zertifikaten, die von einer (lokalen) CA signiert sind, ist es nicht mehr nötig, Shared Secrets auf dem Name Server oder an anderer Stelle zu hinterlegen, was den administrativen Aufwand je nach Umgebung mehr oder weniger deutlich reduziert.

## Stateless Autoconfiguration, dynamische DNS-Updates und viele ungeahnte Möglichkeiten

- Außerdem können sich damit beide Seiten gegenseitig authentisieren, wenn das CA-Zertifikat mit auf den Clients hinterlegt wird.
- Der Konfigurationsaufwand für `nsautoupdate`, in dem festgelegt wird, welche Adressen wo eingetragen werden, ist unschön. Langfristig wäre es denkbar, einen Service zu definieren, dem ein Client nur mitteilt, welche seiner Adressen sich geändert haben. Dieser Service könnte dann nicht nur ein passendes Update an den Name Server schicken, sondern zum Beispiel auch die Konfiguration von Paketfiltern und anderen adressbasierten Konfigurationen wie adressbasierte ACLs eines SMTP-Relays aktualisieren.
- In der bisher vorgestellten Lösung ist es nicht möglich, sicherzustellen, daß ein fremdkontrollierter Client Adressen in der Reverse Zone für sich beansprucht, die tatsächlich einem anderen Rechner gehören. Der generalisierte Update-Service könnte an dieser Stelle noch per Callback überprüfen, daß er unter der beanspruchten Adresse tatsächlich den Client erreicht.



Die dynamische Konfiguration von Paketfiltern, ein zugegebenermaßen beunruhigendes Konzept, führt zu einer weiteren Frage: Läßt sich mit dem gleichen Mechanismus auch sicherstellen, daß nur bestimmte User auf bestimmten Maschinen dynamisch zusätzliche Rechte nutzen können?

Damit wären zwei große Schwächen heutiger Paketfilter möglicherweise behoben: Die adress- oder netzbasierte Konfiguration der Filterregeln, die immer separat zur aktuellen Netzkonfiguration manuell nachgepflegt werden muß, ließe sich möglicherweise sehr vereinfachen und ein benutzerbasiertes Filtern würde damit praktikabel möglich, ohne daß Application Level Gateways für die jeweilige Application ihren eigenen Authentisierungsmechanismus implementieren müßten.

Zumindest auf den ersten Blick ist die Pflege von ACLs unterschiedlicher Services, wie zum Beispiel bei SMTP-Relays, mit einem solchen Mechanismus deutlich unkritischer. Etablierte Services, die üblicherweise nicht über SSL-Verbindungen oder mit anderen Authentisierungsmechanismen angeboten werden, können möglicherweise auch von einem solchen Ansatz profitieren.

## 5 Grundsätzliche Aspekte

Hinter all diesen Schwierigkeiten und den Überlegungen, wie man sie bewältigen kann, zeichnen sich einige fundamentale Probleme als Ursache der momentanen Situation ab.

Die Name-Maschine-Relation, die jedem Rechner und jeder aktiven Netzkomponente einen festen Namen zuordnet und umgekehrt, ist traditionell über den Umweg der IP-Adresse hergestellt worden: Jeder Rechner hat ein oder mehrere Interfaces, die haben eine feste Adresse, die Adresse hat einen festen DNS-Eintrag.<sup>1</sup> Der DNS-Eintrag enthält nicht nur den Namen der Adresse, sondern damit auch den des Interfaces und damit des Rechners. Diese Kette bricht zusammen, wenn wir IPv6-Adressen vergleichsweise dynamisch durch einen Automatismus zuweisen.

Ansätze wie DHCP oder das in der IETF diskutierte Host Identity Protocol (HIP) können diese langfristige Assoziation nicht vollständig wiederherstellen.

DNS ist nicht ausreichend sicher, um es als authoritative Verwaltung für die Name-Maschine-Relation zu verwenden. Auch mit DNSSEC läßt sich ein Sicherheitsniveau, wie es zum Beispiel in Verbindung mit der dynamischen Konfiguration von Paketfiltern nötig wird, nicht erreichen.

Über die Name-Maschine-Relation hinaus zeichnet sich ab, daß die betreffende Maschine und ihr Name für Aufgaben im Applikations-Umfeld nicht die nötige Auflösung bieten; auch der jeweilige User muß in einem schlüssigen Konzept möglicherweise schon auf dieser Ebene mit berücksichtigt werden.

## 6 Fazit

Auch wenn der DNS im Zusammenhang mit IPv6 an vielen Stellen eine notorische Problemquelle ist, zeigt dieses spezielle Problem doch, wo wir mit IPv6 stehen:

- Einer extensiven Spezifikation stehen stellenweise unvollständige, teilweise nur in einzelnen Bruchstücken funktionierende und in einigen Punkten realitätsfremde Implementierungen gegenüber, denen an vielen Stellen die verbindenden Elemente fehlen.
- Die Implementierungen zeigen teilweise erhebliche Schwächen in der Handhabbarkeit und damit der Zuverlässigkeit und Wirtschaftlichkeit im produktiven Einsatz.
- Statt weiter zu spezifizieren ist jetzt die Zeit, aufzubauen, in Betrieb zu nehmen, dabei Erfahrungen zu sammeln, die Implementierungen zu vervollständigen und in einen produktionsstauglichen Zustand zu bringen.
- Dabei werden wir zwangsläufig auf fundamentale Probleme stoßen, die wir erst dann lösen können, wenn wir im Betrieb mit ihnen ausreichende Erfahrungen gesammelt haben, um sie zu verstehen.

---

<sup>1</sup>Die Nuancen zwischen Host- und Interface-Name sind in diesem Zusammenhang nur ein technisches Detail, das vom Kernproblem ablenkt.

Stateless Autoconfiguration, dynamische DNS-Updates und viele ungeahnte Möglichkeiten

- Von den fundamentalen Problemen gibt es allem Anschein nach noch mehr als genug.

Bevor ich die Ideen rund um die X.509-Zertifikate oder darüber hinaus umsetze, werde ich deshalb als nächsten Schritt die direkte TSIG-Variante bis zur Produktionstauglichkeit weiterentwickeln und damit weitere Erfahrungen sammeln. Erst wenn das passiert ist, will ich die X.509-Ideen noch einmal anhand der gesammelten Erfahrungen überdenken und dann entscheiden, wie die weiteren Schritte aussehen können.

Für Anregungen, Kommentare, Kritik und tatkräftige Unterstützung bin ich dabei uneingeschränkt dankbar.

## Über dieses Manuskript

Dieses Manuskript ist als Begleitmaterial zu meinem gleichnamigen Vortrag als Gast der Betriebsagung des Deutschen Forschungsnetzes (DFN) am 12. Oktober 2004 in Berlin entstanden.

## Über den Autor



Der Autor ist Dipl.-Inform. und freischaffender Systemarchitekt und Trainer im Unix- und TCP/IP-Umfeld.

Mit Schulungen vermittelt er herstellerunabhängig Kenntnisse zu Unix, TCP/IP, zum Design von sicheren, zuverlässigen, effizienten und skalierbaren Systemen und vor allem zu seinem liebsten Thema, IPv6.

Wenn er nicht gerade tauchen geht oder mit dem Fahrrad Kontinente sammelt, ist er unter [stockebrand@guug.de](mailto:stockebrand@guug.de), [me@benedikt-stockebrand.de](mailto:me@benedikt-stockebrand.de) und <http://www.benedikt-stockebrand.de/> zu erreichen.