Network Address Translation

Seminar Innovative Internettechnologien und Mobilkommunikation WS2008

Florian Kaiser
Technische Universität München
Informatik VIII
Netzarchitekturen und Netzdienste
Email: kaiserf@in.tum.de

***Kurzfassung*— Aufgrund der beschränkten Anzahl an IPv4-Adressen hat es sich eingebürgert, den Adressraum in öffentliche, global geroutete, und private, nur lokal gültige, Adressen zu segmentieren und zwischen diesen zu übersetzen – Network Address Translation (NAT). Eine öffentliche Adresse wird unter mehreren Rechnern, die ihrerseits nur über private Adressen verfügen, aufgeteilt. Diese Vorgehensweise löst das Problem der Adressknappheit, bringt jedoch eine Reihe an Problemen mit sich – jegliche Kommunikation muss von hinter der NAT aus initiiert werden, echte P2P-Kommunikation ist dadurch nicht mehr möglich. In diesem Paper wird die Funktionsweise von NAT dargelegt und damit verbundene Probleme und Lösungsansätze aufgezeigt.**

***Schlüsselworte*— NAT, Network Address Translation, NAT Traversal, Networking, Internet Architecture, Peer to Peer computing**

I. Einleitung

> *What, exactly, is the Internet? Basically it is a global network exchanging digitized data in such a way that any computer, anywhere, that is equipped with a device called a "modem" can make a noise like a duck choking on a kazoo.*
>
> - Dave Barry

Analoge Modems sind heute in der Minderheit – die Idee des Internets als einem globalen Netzwerk, in dem jeder Rechner von jedem Punkt im Netz aus erreichbar ist, hat sich jedoch nicht verändert.

Zur Entstehungszeit des Internets war der heutige Boom der Computer nicht abzusehen, weshalb man im Internet Protokoll Version 4 (IPv4 oder auch nur IP, [**?**]) – dem Rückgrat unseres heutigen Internets – damals einen Adressraum von 32 Bit vorsah. Schienen die damit möglichen 4 Milliarden Adressen noch unglaublich viel, so sind wir heute an einem Punkt, an dem diese Anzahl nicht mehr ausreicht. Diese Tatsache hat IP-Adressen zu einem kostbaren Gut gemacht. Zwar existiert mit IPv6 [**?**] bereits seit geraumer Zeit ein Protokoll mit einem drastisch erweitertem Adressraum (128 Bit), allerdings ist die Umstellung eines derart riesigen, heterogenen Netzwerks wie des Internets eine äußerst langwierige Sache.

Unter anderem deshalb wurde das Internet in zwei Klassen unterteilt [**?**]: öffentliche Adressen, die entsprechend dem Internet-Gedanken global geroutet werden, und private Adressen, die nur für die Adressierung in einem autonomen IP-Netzwerk gedacht sind.

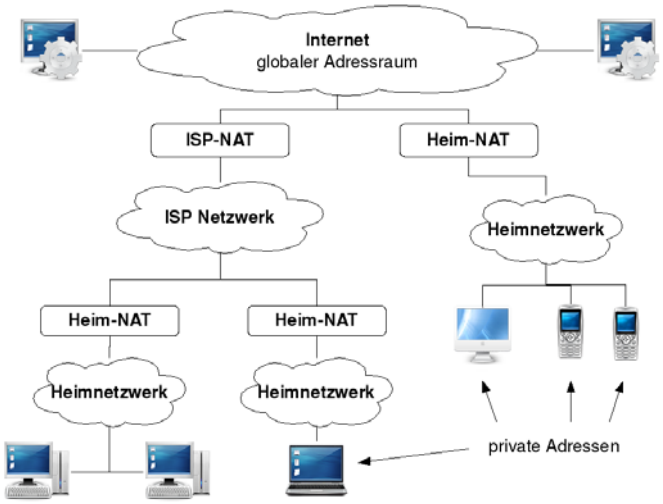


Abbildung 1. Netzwerk-Topologie mit NAT

Da diese privaten Netze unabhängig von einander sind, können Adressen mehrfach vergeben werden, was das Problem der Adressknappheit löst. Allerdings können Rechner in diesem privaten Netz per Definition erst einmal nicht mit Rechnern im Internet kommunizieren.

Um dies zu umgehen, hat es sich an vielerlei Stelle eingebürgert, eine öffentliche Adresse auf mehrere Rechner im lokalen Netz aufzuteilen. Damit sind zumindest ausgehende Verbindungen ins Internet wieder möglich (n:1-Multiplex). Diesen Vorgang bezeichnet man als Network Address Translation (zu deutsch etwa Netzwerkadressen-Übersetzung), kurz NAT.

Ein weiterer, nicht ganz so prominenter, Grund für den Einsatz von NAT kann das Bestreben sein, den Aufbau des internen Netzwerks zu verbergen (Topology Hiding). Durch NAT ist das gegeben: nach außen hin ist immer nur die Adresse des NAT-Gateways zu sehen; die Netzwerkadressen interner Hosts und damit die Netzwerk-Topologie bleiben verborgen.

Mit NAT gehen jedoch prinzipbedingte Probleme einher – Peer-to-Peer-Kommunikation ist nicht mehr möglich. Immer mehr Anwendungen setzen jedoch auf P2P-Mechanismen, z.B. VoIP (Voice over IP, "Telefonieren über das Internet"), Instant-Messenger (Kurznachrichten), Online-Spiele, Dateiübertragung und klassische Filesharing-Börsen.

Dieses Paper beschäftigt sich zunächst mit der grundlegenden Funktionsweise von NAT (II) und den damit verbundenen Problemen (III). Anschließend werden exemplarisch einige der wichtigsten Lösungsansätze (IV) zusammen mit Zahlen zu ihrer Unterstützung in derzeit eingesetzten NATs (IV-I) beschrieben. Zum Abschluss folgt eine kurze Zusammenfassung sowie ein Ausblick auf die Zukunft von NAT und NAT Traversal (V), gerade auch im Hinblick auf IPv6.

II. Funktionsweise

A. Idee

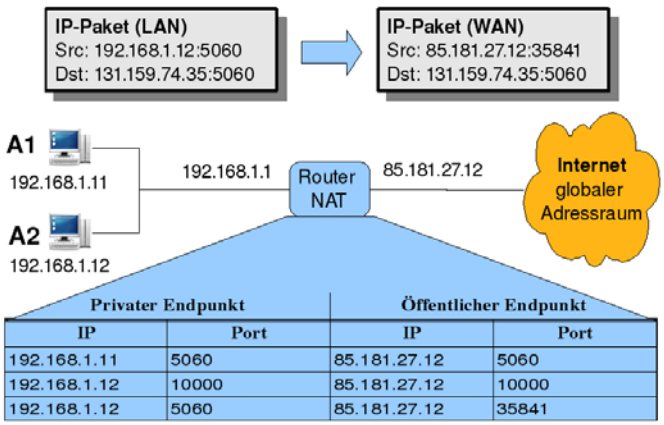
Die NAT zugrunde liegende Idee ist einfach: Jeder Host wird über eine IP-Adresse identifiziert. Damit auf einem Rechner jedoch mehrere Anwendungen gleichzeitig auf das Netzwerk zugreifen können, verwenden die beiden wichtigsten Transport-Protokolle, TCP und UDP, eine weitere Unterteilung in 65536 Ports pro Host – in der Praxis wird diese Anzahl jedoch nicht voll ausgeschöpft werden. Das legt die Idee Nahe, nur einen Host im lokalen Netzwerk über eine öffentliche IP-Adresse mit dem Internet zu verbinden und sämtliche Anfragen von Rechnern in diesem Netz über den Internet-Host zu leiten. In der Praxis wird diese Funktionalität oft von der gleichen Maschine übernommen, die auch das Routing in das Internet bereitstellt. Oft wird auch dieses Gerät, das die Netzwerkadressen-Übersetzung durchführt, als NAT bezeichnet.

In so gut wie jedem privaten Netzwerk, in dem mehr als ein Rechner mit dem Internet verbunden sein soll, wird dieser Mechanismus eingesetzt. Aufgrund der – historisch bedingten – weltweiten Verteilung von IPv4-Adressen bleibt in stark wachsenden Schwellenländern [**?**] auch manchen ISPs nichts anderes übrig, als private IPs hinter einer NAT an ihre Kunden zu vergeben. Natürlich kann weiterhin auch der Kunde selbst eine NAT betreiben, was zu mehreren Ebenen von NAT führt (siehe Abbildung 1: Netzwerk-Topologie mit kaskadierter NAT).

B. Realisierung

Eine Sitzung ist definiert durch die Endpunkte der beiden Kommunikationspartner. NAT geschieht auf Vermittlungs- und Transportschicht – für jedes Paket werden IP-Adresse und Port übersetzt.

Das NAT-Gerät befindet sich im Datenstrom (z.B. in Form eines Gateways) und übersetzt zwischen öffentlichen und privaten Endpunkten im Paket-Header: Für ein ausgehendes Paket wird der private Endpunkt nach einem gewissen Schema (siehe Abschnitt II-C) durch einen öffentlichen Endpunkt ersetzt und der Übersetzungsvorgang (Mapping) in einer Tabelle festgehalten. Man spricht hier vom Erstellen eines Bindings. Trifft nun eine an den öffentlichen Endpunkt gerichtete Antwort ein, so schlägt die NAT in dieser Tabelle das Mapping nach und überschreibt den öffentlichen Ziel-Endpunkt wieder durch den zugehörigen privaten Endpunkt, so dass das Paket im lokalen Netzwerk geroutet werden kann (grafische Darstellung des Mapping-Vorgangs: siehe Abbildung 2).



Privater Endpunkt		Öffentlicher Endpunkt	
IP	Port	IP	Port
192.168.1.11	5060	85.181.27.12	5060
192.168.1.12	10000	85.181.27.12	10000
192.168.1.12	5060	85.181.27.12	35841

Abbildung 2. Mapping-Vorgang

Bei dieser Methode, die in der Form heute in fast jeder NAT zum Einsatz kommt, handelt es sich eigentlich um Network Address Port Translation (NAPT). Es hat sich jedoch eingebürgert, auch in diesem Fall von NAT zu sprechen. In dem gesamten Artikel ist NAT immer als Synonym zu NAPT zu verstehen. Ebenfalls synonym wird oft auch der Begriff IP Masquerading verwendet.

Der Vorteil dieser Vorgehensweise ist, dass sich mit einer öffentlichen IP-Adresse mehrere öffentliche Endpunkte realisieren lassen – in der Theorie nur durch die Anzahl der verfügbaren Ports beschränkt.

Für Umgebungen, in denen ein Rechner die Übersetzung nicht bewältigen kann, existieren Ansätze für Distributed NAT [**?**]. In der Praxis wird jedoch meistens dennoch nur ein einziges NAT-Gerät mit einem öffentlichen Endpunkt eingesetzt.

C. Implementierungs-Typen

NAT kann auf verschiedene Arten implementiert werden. Es gibt keinen offiziellen Standard, so dass sich die Implementierungen von Hersteller zu Hersteller und sogar von Modell zu Modell unterscheiden.

Es existieren Versuche, das Verhalten von NATs zu standardisieren [**?**], allerdings kommt man aufgrund der vielen sich bereits im Einsatz befindenden Geräte – gerade bei Heimbenutzern – nicht umhin, sich mit den Eigenheiten verschiedener Implementierungen auseinander zu setzen.

RFC 3489 [**?**] teilt NAT-Implementierungen in vier Klassen auf (Symmetric, Port Restricted Cone, Restricted Cone, Full Cone) und definiert einen Mechanismus, um zu bestimmen, in welcher Klasse eine NAT liegt. Diese Klassen haben sich mittlerweile als Quasi-Standard zu Beschreibung des Verhaltens von NATs etabliert.

Wichtig ist es allerdings zu beachten, dass sich nicht alle Implementierungen genau in diese Klassen einteilen lassen – es gibt beispielsweise NATs, die sich in manchen Fällen wie eine Cone NAT verhalten, in anderen Fällen wie symmetrische NAT. Für eine genauere Untersuchung der Typen und Eigenschaften von NATs, die sich ebenfalls auf den in RFC 3489 definierten Klassen aufbaut, sei auf [**?**] verwiesen.

1) Cone NAT: Eine Cone NAT übersetzt den selben privaten Endpunkt immer in den selben öffentlichen Endpunkt, ungeachtet des Kommunikationspartners.

Eingehende Pakete an den öffentlichen Endpunkt werden von jedem Host (*Full Cone NAT*), nur von einem Host, an den bereits ein ausgehendes Paket verschickt wurde (*Address Restricted Cone NAT*) oder nur von einem Endpunkt (Adresse und Port), an den bereits ein Paket verschickt wird (*Port Restricted Cone NAT*) akzeptiert.

2) Symmetric NAT: Wie Port Restricted Cone NAT, zusätzlich wird jedoch für jede Sitzung ein neuer öffentlicher Endpunkt erzeugt, auch wenn der private Quell-Endpunkt identisch ist. Dies macht es unmöglich, den öffentlichen Endpunkt des Hosts hinter der NAT von einem zu einem anderen Rechner weiterzugeben. Genau das ist jedoch die Voraussetzung für Hole Punching (IV-A.2).

D. Hairpin Translation

Gerade bei einer durch den ISP eingesetzten NAT kann es vorkommen, dass sich zwei Rechner – ohne sich dessen bewusst zu sein – hinter der selben NAT befinden. Es ist im Allgemeinen für den Client nicht möglich, diese Situation zuverlässig festzustellen.

Deshalb ist es wichtig, dass die NAT auch ausgehende Pakete untersucht, erkennt, wenn diese an einen von der NAT verwalteten öffentlichen Endpunkt adressiert sind, und diese übersetzt und gleich wieder ins private Netzwerk zurück leitet (wie eine gekrümmte Haarnadel, daher der Name Hairpin Translation).

Nur ein Bruchteil der zurzeit eingesetzten NATs unterstützt jedoch Hairpin Translation (siehe IV-I).

III. Probleme mit NAT

Bedingt durch den n:1-Multiplex (n private Adressen werden auf eine öffentliche Adresse übersetzt) muss jede Sitzung von hinter der NAT aus initiiert werden – schließlich müssen in der NAT erst die Bindings erzeugt werden, bevor eingehende Pakete an die öffentliche Seite der NAT einem Host im lokalen Netz zugeordnet werden können. Aufgrund ihrer begrenzten Ressourcen kann die NAT Bindings nur endlich lange aufrecht erhalten, was auch einem eigentlich zustandslosen Protokoll wie UDP gewissermaßen einen Zustand aufzwingt: Findet in einer Sitzung eine gewisse Zeit lang keine Kommunikation statt, so wird in der NAT das Binding gelöscht – die Sitzung bricht zusammen.

IV. Lösungsansätze

Um das Problem der eingehenden Verbindungen zu umgehen, gibt es mehrere Möglichkeiten.

Wollen zwei Rechner, die sich potentiell jeweils hinter einer NAT befinden, kommunizieren, so ist eine Möglichkeit, dass sich beide Rechner zu einem dritten Server verbinden und dieser die Daten zwischen den beiden Clients weiterleitet (IV-B). Diese Variante funktioniert mit jeder Form von NAT, allerdings läuft sämtliche Kommunikation nicht direkt zwischen den Clients ab, sondern eben über den zusätzlichen Server. Dies kann die Verbindungsqualität (Geschwindigkeit, Latenz) durchaus beeinträchtigen. Desweiteren verursacht das Betreiben des Servers hohe Kosten, da dieser den gesamten Traffic durchschleusen muss.

Eine subtilere Lösung ist es, zwar einen zentralen Server zu betreiben, zu dem sich alle Clients eingangs verbinden, diesen aber nur dazu zu verwenden, den Clients jeweils den (öffentlichen) Endpunkt ihres Partners mitzuteilen und die eigentlichen Daten dann mit Hilfe von Hole Punching (IV-A.2) direkt zu übertragen (Rendezvous-Server).

Die bisher beschriebenen Lösungen setzen voraus, dass der Service Provider einen Server bereitstellt, der sich um NAT Traversal kümmert.

Es ist jedoch auch möglich, dass sich der Client selbst um NAT Traversal bemüht, indem er z.B. das Binding in der NAT manuell erzeugt (IV-F).

Der Vorteil einer serverseitigen Lösung ist, dass der Client nicht angepasst werden muss. Dafür muss im Server jede Anwendung individuell implementiert werden. Ein clientseitiger Ansatz hingegen erspart dem Service Provider viel zusätzlichen Aufwand, ist allerdings nicht immer praktikabel.

Im Folgenden eine Übersicht über die verbreitetsten NAT Traversal-Techniken.

In der Praxis ist natürlich eine Kombination der genannten Verfahren möglich und wünschenswert. Ein Beispiel für eine Anwendung, die eine ausgeklügelte Sammlung von NAT Traversal Techniken einsetzt ist das Skype-Protokoll. [**?**]

A. Rendezvous-Server

1) Connection Reversal: Befindet sich nur einer der Kommunikationspartner hinter einer NAT, so kann über einen Rendezvous-Server einfach die Umkehrung der Sitzungs-Richtung vereinbart werden:

Möchte beispielsweise Client A per VoIP Client B, der sich hinter einer NAT befindet, anrufen, so veranlasst er via Server S, an dem beide Kommunikationspartner registriert sind, einen Rückruf seitens B. Für B handelt es sich bei dem Anruf nun um eine ausgehende Sitzung, die problemlos durch die NAT möglich ist.

Zusammen mit Hole Punching ist auch eine Erweiterung des Szenarios auf den Fall, dass sich nur ein Client hinter einer symmetrischen NAT befindet, möglich.

2) Hole Punching: Ist der öffentliche Kommunikations-Endpunkt des Partners bekannt – z.B. durch einen Rendezvous-Server, so ist für viele Protokolle, darunter UDP und TCP, sogenanntes Hole Punching möglich. Das bedeutet, der Client versendet ein Paket an die Zieladresse. Dieses Paket wird möglicherweise von der NAT des Partners verworfen, passiert aber in jedem Fall zuvor die lokale NAT und führt dazu, dass ein neues Binding angelegt wird (ein “Loch” in der NAT). Der Partner führt genau den gleichen Vorgang durch und öffnet dadurch ein “Loch” in seiner NAT. Nun ist eine Kommunikation in beide Richtungen möglich (grafische Darstellung: Abbildung 3). Es muss lediglich noch darauf geachtet werden, dass periodisch Keep-Alive Pakete (Pakete, die keine sinnvollen Daten transportieren, sondern nur dem

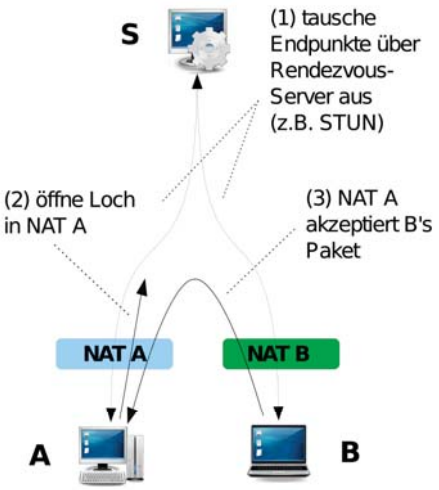


Abbildung 3. Hole Punching-Vorgang

Zweck dienen, die Verbindung aufrecht zu erhalten) verschickt werden, bevor eine der NATs die Bindings löscht.

Am weitesten verbreitet ist Hole Punching für das UDP-Protokoll, z.B. in Form von STUN (IV-A.3) für VoIP via SIP (*Session Initiation Protocol*, [**?**]). Prinzipiell lässt sich die Technik genauso auf andere Protokolle wie TCP anwenden, was allerdings in der Praxis relativ wenig eingesetzt wird und auch weniger zuverlässig funktioniert.

Hole Punching ist in vielen Fällen eine elegante und praktikable Lösung, funktioniert jedoch nicht mit allen Typen von NAT (siehe II-C, IV-I).

Für Näheres zu Hole Punching via UDP und TCP sei auf [**?**] verwiesen.

3) STUN: STUN (*Simple Traversal of UDP Networks*, [?]) ist ein von der IETF (*Internet Engineering Taskforce*) spezifiziertes Protokoll, das es einem Host ermöglicht, den Typ der NAT, hinter der er sich befindet, sowie den verwendeten öffentlichen Endpunkt zu ermitteln (NAT-Typen: siehe II-C). Das Protokoll definiert hierfür einen STUN-Server, der sich auf der anderen Seite der NAT – im Regelfall im öffentlichen Internet – befinden muss.

Im Falle von symmetrischen NATs ist der zurückgelieferte öffentliche Endpunkt jedoch unbrauchbar, da nur für die Kommunikation mit dem STUN-Server gültig.

STUN nach RFC 3489 ist also keine Komplettlösung für NAT Traversal, sondern lediglich ein Hilfsmittel. Dies wurde in der ursprünglichen Fassung von STUN jedoch nicht deutlich, so dass STUN in RFC 5389 [**?**] (Draft) neu formuliert und in "Session Traversal Utilities for NAT" umbenannt wurde.

STUN ist, wie der ursprüngliche Name bereits besagt, nur für den Aufbau von UDP-Sitzungen ausgelegt. Es gibt jedoch einen Versuch, den Ansatz auch auf TCP zu erweitern: STUNT (*Simple Traversal of UDP through NATs and TCP too*, [**?**]).

B. Relay-Server

1) SOCKS: Das SOCKS-Protokoll (Abkürzung für *SOCKetS*, [**?**]) wurde ursprünglich entwickelt, um eine Möglichkeit für Hosts hinter einer Firewall zu schaffen, mit externen Hosts zu kommunizieren, ist jedoch auch auf NATs anwendbar.

Sämtliche Daten werden von der SOCKS-fähigen Anwendung über einen Tunnel an den SOCKS-Server gesendet und von dort ins öffentliche Netz weiter geroutet.

SOCKS ist für Client/Server-Anwendungen ausgelegt, es unterstützt nur entweder eingehende oder ausgehende Verbindungen. Die aktuelle Version 5 bietet Unterstützung für die Protokolle TCP und UDP sowie Multicast und Authentifizierung.

Aufgrund der Client/Server-Beschränkung funktionieren nicht alle Protokolle mit SOCKS – und bei jenen, die es tun, würde oft auch reines NAT ausreichen.

Ein positiver Aspekt von SOCKS ist, dass sich der Einsatz eines SOCKS-Proxies durch eine Lösung wie tsocks [**?**] für die Anwendung völlig transparent gestalten lässt.

2) TURN: Einen IETF-Draft für einen Relay Server stellt das TURN-Protokoll (*Traversal using relay NAT*, [**?**]). Es erlaubt es einem Host hinter einer NAT, eingehende Verbindungen über TCP- oder UDP-Verbindungen zu erhalten. TURN ist allerdings nicht für den Betrieb eines Servers hinter einer NAT ausgelegt.

Interessant wird TURN vor allem im Zusammenhang mit ICE als letzter Ausweg, falls die Kommunikation mittels Hole Punching fehlschlägt.

C. ICE

ICE (*Interactive Connectivity Establishment*, [**?**]) kann als eine Art Meta-Protokoll zur Aushandlung von NAT Traversal Techniken angesehen werden.

Es setzt auf IETF-Standards wie STUN und TURN auf und ermöglicht eine stufenweise Eskalation der Traversal-Techniken. So kann z.B. zuerst eine direkte Verbindung zum privaten Endpunkt des Hosts versucht werden, anschließend Hole Punching und zu guter Letzt, falls auch dieses versagt, auf Relay via TURN ausgewichen werden.

Die ICE-Protokollsuite stellt ein vielversprechendes, standardisiertes Werkzeug im Umgang mit NATs jeglichen Typs dar. Der Preis dafür ist allerdings eine relativ hohe Komplexität.

Bis jetzt hat ICE den Status eines Internet-Drafts und ist erst in wenigen Anwendungen – Hardware wie Software – implementiert.

D. Application Layer Gateway

Ein Application Layer Gateway (ALG) könnte man als NAT für die Anwendungsschicht bezeichnen. Es inspiziert die Anwendungsschicht-PDU und übersetzt dort Endpunkte.

Dies setzt allerdings voraus, dass dem ALG das Protokoll bekannt ist.

In vielen NATs ist ein ALG für verbreitete Protokolle wie FTP (*File Transfer Protocol*, [**?**]) implementiert. Im Falle von FTP sucht das ALG beispielsweise nach einem PORT-Kommando. Entdeckt es ein solches, wird es den übergebenen privaten Endpunkt in einen freien öffentlichen Endpunkt übersetzen und für diesen in der NAT ein Binding erzeugen.

Versagen hingegen wird ein ALG bei neuen, noch nicht implementierten Protokollen. Weiterhin ist die Inspektion einer so hohen Schicht im OSI-Modell [**?**] mit erheblichem rechnerischen Aufwand verbunden. Für komplexe P2P-Protokolle, bei denen unter Umständen mehrere hundert Verbindungen pro Sekunde aufgebaut werden, übersteigt die Komplexität die Rechenkraft heutiger "Heim-NATs".

E. Port Prediction / Symmetric NAT Traversal

Viele NATs weisen Ports nach einem vorhersagbaren Schema zu – z.B. durch einfaches Inkrementieren des letzten benutzten Ports. Es gibt Ansätze, sich dies für die Kommunikation durch symmetrische NATs zunutze zu machen, in dem man versucht, den verwendeten öffentlichen Endpunkt – basierend auf zuvor zugewiesenen Endpunkten – zu "erraten".

Ein solches Verfahren, aufbauend auf STUN, wird in [**?**] beschrieben.

Allerdings sollte ein solch heuristisches Verfahren nur als letzter Ausweg betrachtet werden, um in einigen Fällen selbst eine symmetrische NAT umschiffen zu können. Je mehr mögliche Endpunkte vom Kommunikationspartner durchprobiert werden müssen, desto länger dauert der Verbindungsaufbau – und es gibt keinerlei Garantie, dass der richtige Endpunkt schlussendlich auch getroffen wird.

F. Port Forwarding

1) Manuelles Portforwarding: Fast jede NAT bietet heute die Möglichkeit, über ein Webinterface von Hand Bindings einzutragen. Dies wird als Portforwarding oder Static NAT (SNAT) bezeichnet. Diese statischen Bindings bleiben dauerhaft bestehen, bis sie manuell wieder entfernt werden.

Solange eine Anwendung nur bereits manuell weitergeleitete Ports verwendet, ist die NAT für sie weitgehend transparent. Allerdings ist diese Vorgehensweise in der Praxis nur für eine kleine Anzahl an Ports, die sich nicht (oft) verändern, praktikabel.

2) Automatisiertes Portforwarding: Portforwarding lässt sich natürlich auch automatisieren. Mit UPnP-IGD (*Universal Plug and Play – Internet Gateway Device*, [**?**]) und NAT-PMP (*NAT Portmapping Protocol*, [**?**]) / Bonjour existieren zwei Protokolle, die es Anwendungen ermöglichen, dynamisch Portforwardings einzurichten.

Allerdings sind diese Protokolle nur auf einem geringen Anteil der eingesetzten NATs aktiv. Weiterhin macht der Einsatz nur in einem privaten Heimnetzwerk Sinn – ein ISP wird kein Interesse daran haben, dass der Kunde seine Netzwerk-Infrastruktur beeinflussen kann.

G. Tunneling

Manche Protokolle werden von NATs nicht übersetzt. Ein Grund kann sein, dass das Protokoll zu neu und deshalb nicht in der NAT implementiert ist (z.B. SCTP [**?**], DCCP [**?**]). Manche Protokolle wie IPSec (Internet Protocol Security, [**?**]) können aber auch prinzipbedingt nicht direkt übersetzt werden, da z.B. die Payload verschlüsselt ist.

Dieses Problem kann man umgehen, in dem man die zu übermittelnden Pakete durch ein bekanntes Protokoll, z.B. UDP, tunnelt. Jedes Paket wird also mit z.B. einem UDP-Header versehen, welcher von NATs übersetzt werden kann. Allerdings muss auch die Gegenstelle dieses Verfahren unterstützen und den UDP-Header wieder entfernen. Für IPSec ist dieses Vorgehen weit verbreitet und wird von vielen NATs unterstützt.

H. Realm Specific IP

RSIP ist ein experimentelles IETF-Framework [**?**] und -Protokoll [**?**] und soll eine Alternative zu NAT darstellen. Ein RSIP Host "leiht" sich (idr. öffentliche) Endpunkte von einem RSIP-Gateway. Möchte der Host Daten über einen geliehenen Endpunkt versenden, so überträgt er diese über einen Tunnel versehen mit privatem und geliehenem Absender-Endpunkt an das Gateway. Dieses entfernt den privaten Endpunkt und routet das Paket weiter. Antworten werden auf die gleiche Weise wieder zurück zum Host geleitet.

Vorteilhaft an diesem Verfahrens ist, dass die Ende-zu-Ende-Integrität der übermittelten Pakete erhalten bleibt.

Auch ein Ansatz für IPSec via RSIP existiert [**?**].

Allerdings scheint RSIP bis jetzt nicht über den experimentellen Status hinausgekommen zu sein.

I. Einige Zahlen zu NAT Traversal

Eine Untersuchung von Ford, Srisuresh und Kegel aus dem Jahr 2005 [?] führte zu dem folgenden Ergebnis:

Von den getesteten 380 NATs unterstützen

- 82% UDP Hole Punching
- 64% TCP Hole Punching
- 24% UDP Hairpin Translation
- 13% TCP Hairpin Translation

Getestet wurden NATs von 68 verschiedenen Herstellern sowie die eingebaute NAT-Funktionalität von 8 verschiedenen Betriebssystemen (bzw. Betriebssystem-Versionen).

Aus diesen Zahlen lässt sich ablesen, dass UDP Hole Punching eine interessante Technik ist, die in den meisten Fällen funktioniert – in der Tat wird sie auch an vielen Stellen wie z.B. VoIP (STUN) eingesetzt. Dennoch existiert mit 18 % ein nicht zu vernachlässigender Anteil an Clients, bei denen Hole Punching nicht möglich ist und deshalb auf andere Lösungen wie z.B. einen Relay-Server zurückgegriffen werden muss.

Aus einem Feldtest von A. Müller [**?**] mit 400 NATs geht weiterhin hervor, dass Portforwarding via UPnP-IGD nur in 13 % der Fälle unterstützt wird. Diese doch recht geringe Zahl dürfte allerdings auch daher rühren, dass in vielen Heim-Routern UPnP-Unterstützung zwar vorhanden, aus Sicherheitsgründen aber standardmäßig deaktiviert ist.

V. Zusammenfassung und Ausblick

NAT ist ein weit verbreitetes Verfahren, um eine öffentliche IP-Adresse unter mehreren Hosts zu teilen. Notwendig gemacht wird dieses Vorgehen in erster Linie durch die Adressknappheit bei IPv4. Ein weitere Grund für den Einsatz von NAT kann das Bestreben sein, die interne Netzwerk-Topologie des eigenen Netzwerks zu verbergen. Als Resultat des Übersetzungsvorgangs muss jede Sitzung von einem Host hinter der NAT initiiert werden – P2P-Kommunikation ist nicht möglich. Es existieren verschieden Lösungsansätze, um die mit NAT verbundenen Probleme zu umschiffen, die sich stark in Aufwand und Zuverlässigkeit unterscheiden. Prinzipbedingt kann es jedoch kein Verfahren geben, das NAT in jedem Fall völlig transparent macht.

Es muss daher individuell für jede Anwendung, die mit NATs funktionieren soll, abgewogen werden, welches bzw. welche Verfahren eingesetzt werden sollen. ICE [?] stellt einen vielversprechenden Ansatz zur Koordinierung der verschiedenen Traversal-Techniken dar, ist jedoch bis jetzt nur ein Internet-Draft und in Hardware wie Software wenig verbreitet.

IPv6 könnte NAT, zumindest was den Adressraum betrifft, überflüssig machen. Dennoch ist der Einsatz von NAT z.B. zum Verbergen der internen Netzwerk-Topologie wahrscheinlich. Auch für IPv6 wurden nicht global geroutete Adressbereiche definiert [**?**]. Ein IETF-Draft für NAT mit IPv6 existiert ebenfalls bereits [**?**].

Literatur

[1] J. Postel, "Internet Protocol," RFC 791 (Standard), Sep. 1981, updated by RFC 1349. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Dec. 1998, updated by RFC 5095. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[3] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Feb. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1918.txt

[4] IP2Location.com, "IP2Location Internet IP Address 2008 Report." [Online]. Available: http://www.ip2location.com/ip2location-internet-ip-address-2008-report.aspx

[5] M. S. Borella, D. Grabelsky, I. Sidhu, and B. Petry, "Distributed Network Address Translation," in *Internet Draft*, 1998.

[6] F. Audet and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," RFC 4787 (Best Current Practice), Jan. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4787.txt

[7] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489 (Proposed Standard), Mar. 2003, obsoleted by RFC 5389. [Online]. Available: http://www.ietf.org/rfc/rfc3489.txt

[8] A. Müller, G. Carle, and A. Klenk, "Behavior and classification of NAT devices and implications for NAT traversal," *Network, IEEE*, vol. 22, no. 5, pp. 14–19, September-October 2008.

[9] S. A. Baset and H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol," 2004.

[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261 (Proposed Standard), Jun. 2002, updated by RFCs 3265, 3853, 4320, 4916, 5393. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[11] B. Ford, "Peer-to-peer communication across network address translators," in *In USENIX Annual Technical Conference*, 2005, pp. 179–192.

[12] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389 (Proposed Standard), Oct. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5389.txt

[13] S. Guha and P. Francis, "Simple traversal of UDP through NATs and TCP too (STUNT)." [Online]. Available: http://nutss.gforge.cis.cornell.edu

[14] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS Protocol Version 5," RFC 1928 (Proposed Standard), Mar. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1928.txt

[15] "tsocks, a transparent SOCKS proxying library." [Online]. Available: http://tsocks.sourceforge.net

[16] J. Rosenberg, C. Huitema, and R. Mahy, "Traversal using relay NAT (TURN)," 2003.

[17] J. Rosenberg, "Interactive connectivity establishment (ICE)," 2003.

[18] J. Postel and J. Reynolds, "File Transfer Protocol," RFC 959 (Standard), Oct. 1985, updated by RFCs 2228, 2640, 2773, 3659. [Online]. Available: http://www.ietf.org/rfc/rfc959.txt

[19] N. Budhiraja, K. Marzullo, F. B. Schneider, and S. Toueg, "CCITT Recommendation X.200, Reference Model of Open Systems Interconnection for CCITT Applications," in *Revision: 2.0.0 Page 124 August 20, 1991 OSI Work Group*, 1984, pp. 81–86.

[20] Y. Takeda, "Symmetric NAT Traversal using STUN," 2003. [Online]. Available: http://www.cs.cornell.edu/projects/stunt/draft-takeda-symmetric-nat-traversal-00.txt

[21] UPnP Forum, "Internet Gateway Device (IGD) V 1.0," 2001. [Online]. Available: http://upnp.org/standardizeddcps/igd.asp

[22] S. Cheshire, M. Krochmal, and K. Sekar, "NAT Port Mapping Protocol (NAT-PMP)," Internet-Draft / Standards Track, 2008. [Online]. Available: http://files.dns-sd.org/draft-cheshire-nat-pmp.txt

[23] R. Stewart, "Stream Control Transmission Protocol," RFC 4960 (Proposed Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4960.txt

[24] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340 (Proposed Standard), Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4340.txt

[25] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," RFC 2401 (Proposed Standard), Nov. 1998, obsoleted by RFC 4301, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc2401.txt

[26] M. Borella, J. Lo, D. Grabelsky, and G. Montenegro, "Realm Specific IP: Framework," RFC 3102 (Experimental), Oct. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3102.txt

[27] M. Borella, D. Grabelsky, J. Lo, and K. Taniguchi, "Realm Specific IP: Protocol Specification," RFC 3103 (Experimental), Oct. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3103.txt

[28] G. Montenegro and M. Borella, "RSIP Support for End-to-end IPsec," RFC 3104 (Experimental), Oct. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3104.txt

[29] A. Müller, "Network Address Translator - Tester." [Online]. Available: http://nattest.in.tum.de

[30] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 4291 (Draft Standard), Feb. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4291.txt

[31] M. Wassermann and F. Baker, "IPv6-to-IPv6 Network Address Translation (NAT66)," 2008. [Online]. Available: http://tools.ietf.org/html/draft-mrw-behave-nat66-01