Engineer

IPv6

Version des Internetprotokolls mit 128-Bit-Adressen

Was es ist

IPv6 steht für Internet Protocol version 6 und wurde als Nachfolger der in RFC 791 beschriebenen IP-Version IPv4 entworfen. Seine aktuelle Spezifikation ist RFC 8200 vom Juli 2017, ein Internetstandard (STD 86), der RFC 2460 ersetzte. Die auffälligste Änderung ist die Adressgröße: 128 Bits statt der 32 von IPv4. Ganz IPv4 umfasst 4.294.967.296 Adressen; ein einzelnes IPv6-Subnetz der üblichen Größe, ein /64, enthält das Quadrat dieser Zahl.

Auch der Rest des Pakets änderte sich. RFC 8200 streicht einige IPv4-Headerfelder oder macht sie optional, um den Verarbeitungsaufwand zu verringern. Optionale Informationen stehen in gesonderten Erweiterungsheadern hinter dem Hauptheader. Time to Live heißt nun Hop Limit und wird von jedem weiterleitenden Knoten um eins verringert. Eine Headerprüfsumme gibt es nicht. Router teilen ein Paket nie auf: Bei IPv6 fragmentiert nur die Quelle. Jeder Link muss Pakete von mindestens 1.280 Oktetten transportieren können. Sendern wird nachdrücklich empfohlen, mittels Path MTU Discovery herauszufinden, welche Paketgröße ein Pfad transportieren kann.

Eine Adresse wird als acht durch Doppelpunkte getrennte Gruppen von 16 Bits in Hexadezimaldarstellung geschrieben. RFC 4291 erlaubt, führende Nullen einer Gruppe wegzulassen und eine Folge reiner Nullgruppen durch einen doppelten Doppelpunkt zu ersetzen. Dieser darf nur einmal vorkommen. So lässt sich 2001:db8:0:0:8:800:200c:417a als 2001:db8::8:800:200c:417a schreiben. Weil damit viele Schreibweisen derselben Adresse möglich sind, legt RFC 5952 vom August 2010 eine fest: Kleinbuchstaben, keine führenden Nullen und der doppelte Doppelpunkt für die längste Folge von Nullgruppen, bei Gleichstand für die erste, niemals für eine einzelne Gruppe. Abgesehen von Sonderadressen wie

Loopback und link-lokalen Adressen stammen sämtliche Beispieladressen und Präfixe dieses Artikels aus einem Dokumentationsbereich: 2001:db8::/32, von RFC 3849 reserviert, oder für IPv4 aus den durch RFC 5737 reservierten Blöcken.

RFC 4291 ändert auch, woran eine Adresse gebunden ist. Adressen gehören zu Schnittstellen, nicht zu Maschinen, und eine Schnittstelle kann mehrere zugleich haben. Jede Schnittstelle muss eine link-lokale Adresse unter fe80::/10 besitzen, die nur auf ihrem eigenen Link gültig ist. Die Loopback-Adresse lautet ::1. Broadcast gibt es nicht; diese Aufgabe übernimmt Multicast, und ff02::1 erreicht alle Knoten auf dem Link. Unique Local Addresses unter fc00::/7, definiert in RFC 4193, dienen innerhalb eines Standorts. Globale Unicast-Adressen werden für die Kommunikation über den lokalen Standort hinaus verwendet. Bei den meisten Adressen sind die letzten 64 Bits die Schnittstellenkennung. Deshalb ist ein gewöhnliches Netzwerksegment ein /64.

Hosts vergeben sich ihre Adressen üblicherweise selbst. Bei der zustandslosen Adressautokonfiguration nach RFC 4862 kündigen Router die Präfixe eines Links in regelmäßigen Router Advertisements an. Jeder Host kombiniert ein Präfix mit seiner eigenen Schnittstellenkennung. Bevor er das Ergebnis verwendet, führt er Duplicate Address Detection aus und prüft mit Neighbor-Solicitation-Nachrichten, ob ein anderer Knoten die Adresse bereits besitzt. DHCPv6 ist die Alternative, wenn ein Standort die konkreten Zuweisungen genauer steuern möchte. Beide Verfahren können nebeneinander laufen.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist IPv6?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. WLAN, Firewalls und Fernzugriff werden so eingerichtet, dass Mitarbeitende von überall arbeiten können und nur Berechtigte Zugang erhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 30 Ubuntu-Linux-VPS-Instanzen verwaltet — 30 verwaltete Linux-Server mit geplantem Wiederherstellungsweg

30 Ubuntu-Linux-VPS-Instanzen verwaltet Kontakt aufnehmen →

Warum es wichtig ist

Für viele Teams beginnt IPv6 nicht mit einem bewussten Projektstart. Der Sinn von RFC 4862 ist, dass niemand einem Host eine Adresse eintippen muss: Wird ein Server an einen Link angeschlossen, auf dem ein Router ein Präfix ankündigt,

kann er selbst eine globale IPv6-Adresse erhalten. Jede nur für IPv4 entworfene Positivliste, Firewallregel, Protokollauswertung oder Zustandsprüfung hat dann einen blinden Fleck von genau der Größe des neuen Adressraums.

Dieser blinde Fleck wird leicht übersehen, denn die beiden Versionen sind getrennte Protokolle, die auf der Leitung nichts gemeinsam haben. Ein Paketfilter gleicht ihre Header getrennt ab, eine auf IPv4-Adressen basierende Ratenbegrenzung sieht keinen IPv6-Client und ein Test, der nur zu 127.0.0.1 verbindet, sagt nichts über ::1 aus. Ein Host, der auf einer Version abgesichert und auf der anderen offen ist, hat praktisch keine Firewall.

IPv6 stützt sich außerdem stärker auf seine Steuernachrichten als IPv4. Nachbarerkennung, Router Advertisements und Path MTU Discovery verwenden sämtlich ICMPv6. Die aus IPv4 übernommene Gewohnheit, ICMP am Netzrand zu verwerfen, beschädigt IPv6 daher auf eine Weise, die vielleicht erst beim Datentransfer einer Verbindung auffällt. Und weil eine Adresse viele gültige Schreibweisen hat, widerspricht sich Code, der Adressen als Zeichenketten vergleicht, selbst.

Wie es in der Praxis aussieht

Pythons Standardmodul `ipaddress` liest jede gültige Schreibweise ein und gibt eine Kurzform aus, ohne führende Nullen und mit zusammengezogener längster Nullgruppenfolge. Damit lässt sich schnell ein Präfix planen und prüfen, ob zwei Schreibweisen dieselbe Adresse bezeichnen. Die folgende Sitzung dient als Beispiel dieses Artikels. Sie wurde mit Python 3.15 und Adressen aus dem Dokumentationspräfix ausgeführt:

```
>>> import ipaddress
>>> site = ipaddress.ip_network("2001:db8:42::/48")
>>> site.num_addresses == 2**80
True
>>> lans = site.subnets(new_prefix=64)
>>> next(lans), next(lans)
(IPv6Network('2001:db8:42::/64'), IPv6Network('2001:db8:42:1::/64'))
>>> len(list(site.subnets(new_prefix=64)))
65536
>>> a = ipaddress.ip_address("2001:0DB8:0000:0000:0001:0000:0000:0001")
>>> a
IPv6Address('2001:db8::1:0:0:1')
>>> a.exploded
'2001:0db8:0000:0000:0001:0000:0000:0001'
>>> a == ipaddress.ip_address("2001:db8:0:0:1::1")
```

```
True
>>> ipaddress.ip_address("fe80::1%eth0").scope_id
'eth0'
```

Ein /48 lässt 80 Bits frei. In /64-Segmente aufgeteilt ergibt es 65.536 Segmente, deren Nummer im vierten Block steht. Die mit Großbuchstaben und sämtlichen Nullen eingegebene Adresse kommt als Kurzform 2001:db8::1:0:0:1 zurück. Der doppelte Doppelpunkt umfasst dort die erste von zwei gleich langen Nullfolgen. Sie ist beim Vergleich gleich 2001:db8:0:0:1::1, einer anderen gültigen Schreibweise derselben 128 Bits. Die letzte Zeile zeigt eine link-lokale Adresse mit ihrer Zone: `%eth0` gibt den zugehörigen Link an.

Auf einem Linux-Host mit nftables besteht der zweite Teil des Beispiels aus einem Regelsatz, der SSH aus einem Büro über beide Versionen zulässt. Das Handbuch nft(8) beschreibt `inet` als hybride Familie, deren Tabellen sowohl IPv4- als auch IPv6-Pakete sehen. `ip saddr` liest dagegen den IPv4-Header und `ip6 saddr` den IPv6-Header:

```
add table inet filter
add chain inet filter input { type filter hook input priority filter; }
add rule inet filter input ip saddr 198.51.100.0/24 tcp dport 22 accept
add rule inet filter input ip6 saddr 2001:db8:42::/48 tcp dport 22 accept
add rule inet filter input tcp dport 22 drop
```

Ohne die vierte Zeile wird die Verbindung eines Administrators aus dem IPv6-Präfix des Büros von keiner der beiden Freigaberegeln erfasst und deshalb abgewiesen. Das scheitert wenigstens sichtbar. Der leisere Fehler ist die umgekehrte Situation: Ein allein in der Familie `ip` aufgebauter Regelsatz auf einem Host, der auch eine IPv6-Adresse besitzt, betrachtet IPv6-Pakete überhaupt nicht.

Der Portfolioeintrag Einen zweiten Server aus Code bereitgestellt vom September 2026 behandelt dieselbe Gefährdung aus Sicht der Bereitstellung. Die erste Maschine war von Hand angelegt und erst danach mit ihrer Firewall versehen worden. Dadurch war ein neuer Host mit einer Standardpasswortrichtlinie zeitweise aus dem Internet erreichbar. Die zweite wird erst nach ihrer Firewall erstellt: Das Firewall-Play läuft vor dem Server-Play, und die Firewalls beider Anbieter werden direkt über deren REST-APIs definiert. Die Seite erwähnt IPv6 nicht. Die obigen Regeln sind daher Beispiele dieses

Artikels, nicht die Regeln jenes Hosts. Der Artikel ergänzt den Grundsatz, dass die Firewall schon vor der Existenz des Hosts alle seine künftigen Adressen abdecken muss, einschließlich IPv6.

Wo es schiefgeht

Eine Firewall kennt nur IPv4. Regeln in der Familie `ip` oder Positivlisten mit ausschließlich IPv4-Präfixen überlassen jede IPv6-Adresse des Hosts dem, was danach kommt. Schreiben Sie den Regelsatz in der Familie `inet` oder halten Sie `ip` - und `ip6` -Tabellen synchron. Ergänzen Sie jede Regel mit `ip saddr` durch eine mit `ip6 saddr` und testen Sie jeden Dienst sowohl über seine IPv6- als auch über seine IPv4-Adresse.

ICMPv6 wird pauschal verworfen. RFC 4890 über das Filtern dieser Nachrichten in Firewalls sagt es deutlich: „ICMPv6 ist für das Funktionieren von IPv6 unverzichtbar.“ Kann ein Router ein Paket nicht weiterleiten, weil es für den nächsten Link zu groß ist, verwirft er es und sendet eine Packet-Too-Big-Nachricht zurück. Laut RFC 8201 erfährt die Quelle die Pfad-MTU nie, wenn ein Filter diese Nachrichten aufhält. Es entsteht eine dort so genannte Black-Hole-Verbindung: Der TCP-Handshake gelingt, danach hängt die Übertragung. RFC 4890 nennt außerdem Neighbor Solicitation und Advertisement sowie Router Solicitation und Advertisement als Nachrichten, die ein Knoten nicht verwerfen darf, wenn er an einem IPv6-Netz teilnehmen soll. Lassen Sie die von nft `packet-too-big` , `nd-neighbor-solicit` , `nd-neighbor-advert` , `nd-router-solicit` und `nd-router-advert` genannten Typen vor jeder allgemeineren Verwerfungsregel zu.

Adressen werden als Text verglichen. RFC 5952 nennt das Beispiel einer Suche in einer Tabellenkalkulation nach 2001:db8:0:0:1::1, die einen Eintrag als 2001:db8::1:0:0:1 nicht findet. Das ist wichtig, wenn vor der Vergabe einer neuen Adresse in der Liste geprüft wird, ob sie bereits verwendet wird. Parsen Sie Adressen vor dem Vergleich. Verwenden Sie zum Speichern und zur Ausgabe eine kanonische Form, etwa die Kurzform, die Pythons `ipaddress` mit `str()` liefert.

Adresse und Port laufen ineinander. 2001:db8::1:80 könnte eine Adresse oder eine Adresse mit angehängtem Port 80 sein. RFC 5952 nennt diese Form mehrdeutig und empfiehlt wie in URLs eckige Klammern: [2001:db8::1]:80. Schreiben Sie die Klammern immer und trennen Sie im Code Host und Port an der schließenden Klammer statt am letzten Doppelpunkt.

Ein Linkpräfix ist kein /64. Für die Autokonfiguration müssen Präfix und Schnittstellenkennung zusammen 128 Bits ergeben. Bei 64-Bit-Kennungen lässt RFC 4862 einen Host angekündigte Präfixe jeder anderen Länge ignorieren. Hosts an einem Link mit /72 oder /56 in den Router Advertisements bilden daher keine Adresse aus diesem Präfix. Planen Sie jeden Link als /64 und teilen Sie größere Zuweisungen wie in der obigen Sitzung in /64-Netze auf.

Link-lokalen Adressen fehlt ihre Zone. Jede Schnittstelle hat eine Adresse in fe80::/10. RFC 4291 beschränkt diese Adressen auf einen einzelnen Link. Auf einem Host mit mehreren Schnittstellen sagt die Adresse allein daher nicht, welcher Link gemeint ist. Pythons Dokumentation hält fest, dass zwei Adressen, die sich nur durch ihre Zone unterscheiden, nicht gleich sind. Bewahren Sie die Zone bei der Adresse auf (fe80::1%eth0) oder geben Sie Diensten eine globale oder Unique-Local-Adresse und verwenden Sie link-lokale Adressen nur auf dem jeweiligen Link.

Weiterführende Lektüre

- RFC 8200: Spezifikation des Internet Protocol, Version 6 (IPv6) — Der aktuelle IPv6-Standard mit Headerformat, Erweiterungsheadern und den Regeln für Fragmentierung und Paketgröße.
- RFC 4291: Architektur der IPv6-Adressierung — Adresstypen, Textformen, link-lokale und Multicast-Adressen sowie 64-Bit-Schnittstellenkennungen.
- RFC 5952: Empfehlung zur Textdarstellung von IPv6-Adressen — Die kanonische Schreibweise einer Adresse und die Probleme zahlreicher Schreibweisen.
- RFC 4862: Zustandslose IPv6-Adressautokonfiguration — Wie Hosts aus angekündigten Präfixen eigene Adressen bilden und auf doppelte Vergabe prüfen.
- RFC 4890: Empfehlungen zum Filtern von ICMPv6-Nachrichten in Firewalls — Welche ICMPv6-Nachrichten eine Firewall durchlassen muss und warum.
- ipaddress: Bibliothek zur IPv4-/IPv6-Verarbeitung — Pythons Referenz zum Parsen, Vergleichen und Aufteilen von IPv6-Adressen und Netzwerken.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist IPv6?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/ipv6/