Engineer

CIDR

⊞ Adressblöcke über Präfixe beschreiben und routen

Was es ist

CIDR, kurz für Classless Inter-Domain Routing, ist das Verfahren, das einen Block von IP-Adressen durch eine Startadresse und die Zahl der führenden Bits beschreibt, die alle Adressen darin gemeinsam haben. Geschrieben ist es der bekannte Schrägstrich: 198.51.100.0/24 bezeichnet jede Adresse, deren erste 24 Bits mit 198.51.100.0 übereinstimmen. Das sind 256 Adressen von 198.51.100.0 bis 198.51.100.255. Die Zahl nach dem Schrägstrich heißt Präfixlänge, der Block selbst Präfix.

Klassenlos verweist auf den Vorgänger. Der ursprüngliche IPv4-Plan aus RFC 791 hatte drei Netzgrößen. RFC 4632, die heutige Beschreibung von CIDR, erklärt deren Scheitern: Klasse C mit 254 nutzbaren Hostadressen war für eine mittelgroße Organisation zu klein, Klasse B mit bis zu 65.534 viel zu groß. Trotzdem war Klasse B die passendste verfügbare Wahl. Als das Netz Ende der 1980er-Jahre kommerzialisiert wurde, wuchs die Nachfrage nach Klasse B scheinbar exponentiell.

Im Januar 1992 benannte die ROAD-Gruppe der IETF, Routing and Addressing, drei Probleme: zur Neige gehenden Klasse-B-Adressraum, Routingtabellen jenseits dessen, was Router und ihre Betreiber bewältigen konnten, und die spätere vollständige Erschöpfung von IPv4. Die ersten beiden sollten zwischen 1993 und 1995 kritisch werden. CIDR antwortete auf diese beiden Probleme. Es wurde zuerst im Juni 1992 in RFC 1338 und dann im September 1993 in RFC 1519 vorgeschlagen. Diesen ersetzte RFC 4632 im August 2006. Das dritte Problem sollte CIDR nie lösen, sondern nur Zeit gewinnen.

In CIDR-Notation schreibt RFC 4632 ein Präfix als übliche vierteilige Adresse, einen Schrägstrich und eine Zahl von 0 bis 32 für die maßgeblichen Bits. Die ältere Netzmaske drückt dasselbe aus, indem diese Bits auf eins und die übrigen

auf null gesetzt sind. /24 ist also 255.255.255.0. Die einzige beibehaltene Einschränkung des RFC ist, dass die auf eins gesetzten Bits der Maske zusammenhängen. Innerhalb dieser Grenze kann ein Präfix jeden Block mit einer Zweierpotenz als Größe beschreiben. Die Tabelle des RFC reicht von /32, einer Hostroute für eine Adresse, über /31 für eine Punkt-zu-Punkt-Verbindung mit zwei, /24 mit 256 und /16 mit 65.536 bis zu 0.0.0.0/0: allen 4.294.967.296 Adressen, als Standardroute verwendet.

Die Notation wird auch außerhalb der Adressfamilie verwendet, für die sie geschaffen wurde. RFC 4291, der den Aufbau von IPv6-Adressen festlegt, schreibt IPv6-Präfixe genauso. 2001:db8::/32 liest sich also genau wie seine IPv4-Verwandten.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist CIDR?“ 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

CIDR löste seine ursprünglichen Probleme durch Aggregation. Laut RFC 4632 ist das Zusammenfassen von Routen der einzige weithin verstandene Weg, den Umfang der Routinginformationen in einem paketvermittelten Netz zu verringern. Ein Anbieter, der einen großen Block besitzt und seinen Kunden kleine Teile davon zuweist, kann eine einzige Route für den gesamten Block bekanntgeben. Das übrige Internet braucht nicht mehr, um jeden Kunden darin zu erreichen. RFC 4632 ergänzt, dass sich dafür auch die Adressvergabe ändern musste. Auf den nicht aggregierbaren Teil des alten Klasse-C-Raums, als „swamp“ (Sumpf) bekannt, entfielen 2006 noch geschätzte 15.000 Routingeinträge.

Aggregation funktioniert durch eine Regel, der jeder Router folgt. RFC 4632 formuliert es so: „Die Weiterleitung im Internet erfolgt nach dem Prinzip der längsten Übereinstimmung.“ Decken mehrere Präfixe ein Ziel ab, gewinnt das spezifischste. So bleibt eine Organisation erreichbar, die ohne neue Adressierung den Anbieter wechselt: Der neue Anbieter kündigt ihr eigenes kleines Präfix an.

Laut RFC stanzt das ein Loch in das Aggregat des alten Anbieters. Dieses Präfix wird als längere Übereinstimmung gegenüber dem Aggregat bevorzugt. CIDR war für drei bis fünf Jahre geplant und laut RFC bis 2005 umfassend eingeführt.

Nur wenige Ingenieure konfigurieren domänenübergreifendes Routing. Fast alle schreiben jedoch Präfixe: in einer Firewallregel, die ein Büronetz zulässt, in einer Routingtabelle auf einem Linux-Server und in WireGuard. Dort entscheidet die Präfixliste AllowedIPs jedes Peers sowohl darüber, welche Quelladressen von ihm angenommen werden, als auch darüber, wohin der Verkehr für ihn gesendet wird. An jedem dieser Orte lässt eine falsche Ziffer nach dem Schrägstrich stillschweigend Tausende Hosts zu oder sperrt sie aus.

Wie es in der Praxis aussieht

Pythons Standardmodul `ipaddress` versteht die Notation und erlaubt eine schnelle Prüfung der Rechnung, bevor sie in eine Regel eingeht. Die folgende Sitzung ist ein Beispiel dieses Artikels, ausgeführt unter Python 3.15 mit einer Adresse aus dem Dokumentationsbereich 198.51.100.0/24:

```
>>> import ipaddress
>>> net = ipaddress.ip_network("198.51.100.0/24")
>>> net.netmask, net.num_addresses
(IPv4Address('255.255.255.0'), 256)
>>> for sub in net.subnets(new_prefix=26):
...     print(sub)
...
198.51.100.0/26
198.51.100.64/26
198.51.100.128/26
198.51.100.192/26
>>> ipaddress.ip_network("198.51.100.70/26")
Traceback (most recent call last):
  ...
ValueError: 198.51.100.70/26 has host bits set
>>> ipaddress.ip_network("198.51.100.70/26", strict=False)
IPv4Network('198.51.100.64/26')
>>> half = ipaddress.ip_network("198.51.100.128/25")
>>> ipaddress.ip_address("198.51.100.200") in half
True
>>> half.subnet_of(net)
True
```

Ein /24 zerfällt in vier /26-Blöcke mit je 64 Adressen. Jeder beginnt bei einem Vielfachen von 64. Die Anfrage nach 198.51.100.70/26 scheitert, weil der Block mit .70 bei .64 beginnt und die Bits nach dem 26. nicht null sind. Mit `strict=False` setzt das Modul sie auf null und gibt den Block zurück, zu dem .70 gehört. Die letzten beiden Zeilen stellen die Fragen, die ein Prüfer bei jedem Präfixpaar stellen sollte: Liegt diese Adresse darin, und liegt ein Block vollständig im anderen?

Die zweite Frage ist bei einem Paketfilter besonders wichtig. Er verwendet keine längste Übereinstimmung, sondern liest Regeln der Reihe nach und stoppt bei der ersten entscheidenden Regel. Das Handbuch nft(8) sagt, dass `accept` und `drop` die Auswertung der Kette beenden. `add` hängt eine Regel am Ende einer Kette an, `insert` setzt sie an den Anfang. Dieses zweite Beispiel des Artikels soll einem /24 den SSH-Zugriff erlauben und seine obere Hälfte ausschließen:

```
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 ip saddr 198.51.100.128/25 tcp dport 22 drop
```

Beide Regeln sehen richtig aus, und eine Auflistung zeigt beide. Doch jede Adresse im /25 liegt auch im /24. Die erste Regel akzeptiert deshalb diese Pakete. Die darunterstehende Verwerfungsregel wird nie erreicht. Beim Routing würde das spezifischere Präfix unabhängig von seiner Position gewinnen. Im Filter gewinnt es nur, wenn es zuerst kommt. Die beiden Zeilen zu tauschen oder die engere Regel mit `insert` einzufügen liefert das gewünschte Ergebnis. `nft -a list ruleset` gibt jede Regel mit ihrem Handle aus. Sowohl `add` als auch `insert` nehmen ein Handle entgegen, um eine neue Regel neben einer bestehenden zu platzieren.

Der Portfolioeintrag Firewallregeln anhand ihrer Position geprüft vom August 2026 beschreibt dieselbe Falle aus dem Betrieb. Ein Host lief ein Jahr lang mit einer Ratenbegrenzung, die kein Paket je erreichte, weil sie unter einer umfassenderen, zuerst passenden Regel stand. Jedes Werkzeug mit einer Zusammenfassung zeigte das Limit korrekt an. Die Prüfungen lesen jetzt die nummerierte Regelliste und die aktive Filterkette und verlangen die richtige Reihenfolge. Die Seite sagt nicht, wie die umfassendere Regel geschrieben war oder ob Adresspräfixe beteiligt waren. Die obigen Zeilen veranschaulichen deshalb das Prinzip, nicht die Regeln dieses Hosts.

Wo es schiefgeht

Hostbits in der Adresse. 198.51.100.70/26 ist eine Adresse mit Maske, kein Block. Python weist sie standardmäßig zurück. Mit `strict=False` rundet es jedoch auf 198.51.100.64/26 ab, was womöglich nicht die vom Verfasser gemeinten Hosts sind. Schreiben Sie die Netzadresse des Blocks und parsen Sie strikt, damit der Fehler sichtbar wird.

Eine bloße Adresse statt eines Blocks. Laut Pythons Dokumentation gilt ein Netz ohne Maske als /32. Eine ohne Präfix eingegebene Büroadresse lässt deshalb einen Host zu. Schreiben Sie die Präfixlänge immer dazu, auch /32, damit Leser eine Hostroute von einem vergessenen Schrägstrich unterscheiden können.

Ein breites Präfix über einem engen. Wie das Beispiel zeigt, verdeckt in einem Filter mit erster passender Regel eine frühere Regel jede spätere, deren Präfix sie enthält. Stellen Sie die spezifischsten Regeln zuerst. Prüfen Sie bei generierten Regeln jedes Paar auf Enthaltensein, etwa mit `subnet_of` oder `overlaps`. Lassen Sie den Build fehlschlagen, wenn eine spätere Regel unerreichbar ist.

Eine um wenige Bits zu kurze Präfixlänge. Jedes entfernte Bit verdoppelt den Block. Ein statt /24 eingegebenes /16 lässt deshalb 256-mal so viele Adressen zu. /0 lässt jede existierende Adresse zu. RFC 4632 hält fest, dass durch Bedienfehler und andere Ereignisse früher /1- und /2-Routen in manchen Netzen beobachtet wurden. Behandeln Sie jede Regel, die kürzer als der eigene Block ist, als Prüfpunkt. Reservieren Sie 0.0.0.0/0 und ::/0 für Regeln, die alles erfassen sollen.

Hosts wie Adressen zählen. Ein /24 enthält 256 Adressen. In einem gewöhnlichen Netz ist die erste aber die Netzadresse und die letzte die Broadcastadresse. Pythons `hosts()` liefert deshalb 254 davon zurück. Auf Punkt-zu-Punkt-Verbindungen summiert sich diese Verschwendung. RFC 3021 merkt an, dass ein /30 vier Adressen für zwei Hosts verbraucht, und erlaubt stattdessen /31. Das spart über 500 Verbindungen hinweg 1.000 Adressen. Bemessen Sie einen Block für die benötigten Hosts zuzüglich dieser reservierten Adressen und verwenden Sie /31 nur auf Geräten mit Unterstützung für RFC 3021.

Ein Aggregat ohne Verwerfungsroute. Ein Router, der einen zusammengefassten Block ankündigt, aber eines der kleineren Präfixe darin verloren hat, kann Verkehr für den fehlenden Teil über eine weniger spezifische Route zurückschicken, etwa die Standardroute. So entsteht eine Weiterleitungsschleife. RFC 4632 verlangt vom erzeugenden Router, Pakete zu verwerfen, die auf das Aggregat, aber auf keine spezifischere Route passen. In der Praxis bedeutet das eine Discard- oder Nullroute für jedes angekündigte Aggregat.

Zulassungslisten nur für IPv4. In nftables liest `ip saddr` den IPv4-Header und `ip6 saddr` den IPv6-Header. In Python löst das Zusammenfassen einer gemischten Liste einen `TypeError` aus, weil sich beide Versionen nicht kombinieren lassen. Eine IPv4-Präfixliste sagt nichts über IPv6-Clients. Für sie greift dann, was als Nächstes folgt. Halten Sie für jede Regel IPv4- und IPv6-Präfixe nebeneinander und prüfen Sie beim Testen beide.

Weiterführende Lektüre

- RFC 4632: Klassenloses domänenübergreifendes Routing (CIDR): Plan zur Adressvergabe und Aggregation im Internet — die heutige Beschreibung von CIDR mit Geschichte, Präfixtabelle und Aggregationsregeln.
- RFC 3021: 31-Bit-Präfixe auf IPv4-Punkt-zu-Punkt-Verbindungen — warum ein /30 die Hälfte seiner Adressen auf einer Punkt-zu-Punkt-Verbindung verbraucht und wie ein /31 das vermeidet.
- RFC 4291: IPv6-Adressarchitektur — die IPv6-Adressierungsregeln einschließlich Präfixen in derselben Schrägstrichnotation.
- ipaddress: Bibliothek zur IPv4/IPv6-Verarbeitung — Pythons Referenz zum Parsen, Aufteilen, Vergleichen und Zusammenfassen von Präfixen.
- nft(8): nftables-Handbuch — die nftables-Referenz einschließlich Regelreihenfolge und kettenbeendenden Entscheidungen.
- wg(8): WireGuard-Schnittstellen konfigurieren und auslesen — das WireGuard-Handbuch, das AllowedIPs als Liste von Adressen mit CIDR-Masken definiert.

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 CIDR?“ 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/cidr/