Engineer

CIDR

décrire et router des blocs IP avec des préfixes

Ce que c’est

CIDR abrège Classless Inter-Domain Routing, ou routage interdomaines sans classes : un schéma décrivant un bloc d’adresses IP par une adresse de départ et le nombre de bits initiaux partagés par toutes ses adresses. À l’écrit, c’est la barre oblique familière : 198.51.100.0/24 désigne toutes les adresses dont les 24 premiers bits correspondent à 198.51.100.0, soit 256 adresses de 198.51.100.0 à 198.51.100.255. Le nombre après la barre s’appelle la longueur de préfixe, et le bloc lui-même un préfixe.

Le terme sans classes renvoie à ce qui précédait. Le plan IPv4 initial de la RFC 791 avait trois tailles de réseau. La RFC 4632, description actuelle du CIDR, explique leur échec : la classe C, avec 254 adresses d’hôte utilisables, était trop petite pour une organisation moyenne, et la classe B, jusqu’à 65 534, bien trop grande. Pourtant, la classe B était le meilleur choix disponible. Avec la commercialisation du réseau à la fin des années 1980, la demande de classe B a augmenté à un rythme apparemment exponentiel.

En janvier 1992, le groupe ROAD de l’IETF, Routing and Addressing, a désigné trois problèmes : l’épuisement de l’espace de classe B, des tables de routage dépassant ce que les routeurs et leurs opérateurs pouvaient gérer, et l’épuisement à terme d’IPv4 lui-même. Les deux premiers devaient devenir critiques entre 1993 et 1995. Le CIDR y répondait, d’abord proposé dans la RFC 1338 en juin 1992, puis dans la RFC 1519 en septembre 1993, remplacée par la RFC 4632 en août 2006. Il n’a jamais été conçu pour résoudre le troisième, seulement pour gagner du temps.

En notation CIDR, la RFC 4632 écrit un préfixe sous forme d’adresse habituelle en quatre parties, d’une barre oblique et d’un nombre de 0 à 32 indiquant les bits significatifs. L’ancien masque réseau exprime la même chose avec ces bits à un et

les autres à zéro : /24 correspond ainsi à 255.255.255.0. La seule contrainte conservée par la RFC est que les bits du masque à un soient contigus. Dans cette limite, un préfixe décrit tout bloc dont la taille est une puissance de deux. La table de la RFC va de /32, route d'hôte pour une adresse, à /31 pour une liaison point à point de deux adresses, puis /24 avec 256 et /16 avec 65 536, jusqu'à 0.0.0.0/0, les 4 294 967 296 adresses, utilisé comme route de dernier recours.

La notation a dépassé la famille d'adresses pour laquelle elle avait été conçue. La RFC 4291, qui définit la structure des adresses IPv6, écrit les préfixes IPv6 de la même façon : 2001:db8::/32 se lit donc exactement comme ses équivalents IPv4.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que le CIDR ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Du Wi-Fi, des pare-feu et des accès à distance configurés pour travailler de partout, en laissant entrer uniquement les personnes autorisées.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 30 instances VPS Ubuntu Linux administrées — 30 serveurs Linux gérés, avec récupération prévue

30 instances VPS Ubuntu Linux administrées Contactez-nous →

Pourquoi c'est important

Le CIDR a résolu ses problèmes initiaux par l'agrégation. Selon la RFC 4632, résumer les routes est le seul moyen largement compris de réduire les informations de routage d'un réseau à commutation de paquets. Un fournisseur possédant un grand bloc et attribuant à ses clients de petits morceaux peut annoncer une seule route pour tout le bloc. Le reste d'Internet n'a besoin de rien d'autre pour joindre chaque client à l'intérieur. La RFC 4632 ajoute que l'attribution des adresses a elle-même dû changer pour le permettre. La partie de l'ancien espace de classe C impossible à agréger, surnommée swamp, ou marécage, représentait encore environ 15 000 entrées de routage en 2006.

L'agrégation fonctionne grâce à une seule règle suivie par tous les routeurs. La RFC 4632 l'exprime ainsi : « Le transfert sur Internet se fait sur la base de la correspondance la plus longue. » Si plusieurs préfixes couvrent une destination, le plus spécifique l'emporte. C'est ainsi qu'une organisation changeant de fournisseur sans renuméroter ses adresses reste joignable : le nouveau fournisseur annonce le petit préfixe propre à l'organisation. La RFC décrit cela

comme un trou dans l'agrégat de l'ancien fournisseur. Ce préfixe est préféré à l'agrégat parce que la correspondance est plus longue. Prévu pour durer de trois à cinq ans, le CIDR était, selon la RFC, largement déployé en 2005.

Peu d'ingénieurs configurent le routage interdomaines, mais presque tous écrivent des préfixes : dans une règle de pare-feu autorisant un réseau de bureaux, une table de routage Linux ou WireGuard. Dans WireGuard, la liste de préfixes AllowedIPs de chaque pair décide à la fois des adresses sources acceptées de ce pair et de la destination du trafic qui lui est destiné. Partout, un seul chiffre erroné après la barre peut autoriser ou exclure silencieusement des milliers d'hôtes.

À quoi cela ressemble en pratique

Le module standard Python `ipaddress` comprend cette notation et permet de vérifier rapidement les calculs avant de les placer dans une règle. La session ci-dessous illustre cet article. Elle a été exécutée sous Python 3.15 avec une adresse de la plage documentaire 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
```

Un /24 se divise en quatre blocs /26 de 64 adresses, chacun commençant sur un multiple de 64. Demander 198.51.100.70/26 échoue : le bloc contenant .70 commence à .64, et les bits après le 26e ne sont pas nuls. Avec `strict=False`, le module les met à zéro et renvoie le bloc de .70. Les deux dernières lignes posent les questions à vérifier pour toute paire de préfixes : cette adresse appartient-elle au bloc, et l'un des blocs est-il entièrement contenu dans l'autre ?

Cette seconde question compte surtout dans un filtre de paquets, qui n'utilise pas la correspondance la plus longue. Il lit ses règles dans l'ordre et s'arrête à la première qui décide. Le manuel nft(8) indique que `accept` et `drop` terminent l'évaluation de la chaîne, que `add` ajoute une règle en fin de chaîne et que `insert` la place au début. Dans cette seconde illustration, l'objectif est de laisser un /24 accéder à SSH tout en excluant sa moitié supérieure :

```
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
```

Les deux règles paraissent correctes et figurent toutes les deux dans la liste. Mais chaque adresse du /25 appartient aussi au /24 : la première règle accepte donc ces paquets, et le drop situé dessous n'est jamais atteint. En routage, le préfixe le plus spécifique gagnerait quelle que soit sa position ; dans un filtre, il ne gagne que s'il vient en premier. Échanger les deux lignes ou ajouter la règle la plus étroite avec `insert` donne le résultat voulu. `nft -a list ruleset` affiche chaque règle avec son handle. `add` comme `insert` acceptent un handle pour placer une nouvelle règle à côté d'une règle existante.

L'article du portfolio Règles de pare-feu vérifiées selon leur position, d'août 2026, décrit le même piège côté exploitation. Un hôte a fonctionné un an avec une limite de débit qu'aucun paquet n'atteignait : elle se trouvait sous une règle plus large correspondant en premier. Tous les outils de synthèse affichaient pourtant correctement la limite. Les contrôles lisent maintenant la liste numérotée des règles et la chaîne de filtrage active, et vérifient leur ordre. La page ne précise ni la forme de la règle plus large ni si elle utilisait des préfixes d'adresses. Les lignes ci-dessus illustrent donc le principe, pas les règles de cet hôte.

Où les problèmes surviennent

Des bits d'hôte dans l'adresse. 198.51.100.70/26 est une adresse avec masque, pas un bloc. Python la refuse sans réglage contraire, mais `strict=False` l'arrondit vers le bas à 198.51.100.64/26, ce qui peut ne pas représenter les hôtes voulus. Écrivez l'adresse réseau du bloc et analysez-la en mode strict pour que l'erreur soit signalée.

Une adresse seule au lieu d'un bloc. Selon la documentation Python, un réseau sans masque est pris comme /32 : une adresse de bureau sans préfixe autorise donc un seul hôte. Écrivez toujours la longueur de préfixe, même /32, pour distinguer une route d'hôte d'une barre oubliée.

Un préfixe large au-dessus d'un préfixe étroit. Comme dans l'illustration, un filtre à première correspondance laisse une règle antérieure masquer toute règle ultérieure dont elle contient le préfixe. Placez d'abord les règles les plus spécifiques. Si les règles sont générées, testez chaque paire par un contrôle d'inclusion comme `subnet_of` ou `overlaps` et faites échouer la compilation si une règle ultérieure est inaccessible.

Une longueur de préfixe trop courte de quelques bits. Chaque bit retiré double le bloc : saisir /16 à la place de /24 multiplie par 256 le nombre d'adresses autorisées, et /0 les autorise toutes. La RFC 4632 rapporte que des erreurs d'opérateur et d'autres événements ont fait apparaître par le passé des routes /1 et /2 dans certains réseaux. Toute règle plus courte que le bloc détenu mérite une revue. Réservez 0.0.0.0/0 et ::/0 aux règles censées tout couvrir.

Compter les hôtes comme les adresses. Un /24 a 256 adresses, mais sur un réseau ordinaire la première est l'adresse réseau et la dernière l'adresse de diffusion. Le `hosts()` de Python en renvoie donc 254. Sur les liaisons point à point, le gaspillage s'accumule : la RFC 3021 note qu'un /30 dépense quatre adresses pour deux hôtes et autorise un /31 à la place, économisant 1 000 adresses sur 500 liaisons. Dimensionnez le bloc pour les hôtes nécessaires plus les adresses réservées, et n'utilisez /31 que sur du matériel compatible avec la RFC 3021.

Un agrégat sans route de rejet. Un routeur annonçant un bloc résumé mais ayant perdu l'un des petits préfixes qu'il contient peut renvoyer le trafic de ce morceau manquant par une route moins spécifique, comme la route de dernier recours, et créer une boucle de transfert. La RFC 4632 exige du routeur générant

l'agrégat qu'il rejette les paquets qui lui correspondent sans correspondre à une route plus spécifique. En pratique, cela revient à installer une route de rejet ou nulle pour chaque agrégat annoncé.

Des listes d'autorisation limitées à IPv4. Dans nftables, `ip saddr` lit l'en-tête IPv4 et `ip6 saddr` l'en-tête IPv6. Dans Python, agréger une liste mixte lève une `TypeError`, car les deux versions ne se combinent pas. Une liste de préfixes IPv4 ne dit rien des clients IPv6, qui passent alors à ce qui suit. Gardez les préfixes IPv4 et IPv6 côte à côte pour chaque règle, et vérifiez les deux dans les tests.

Pour aller plus loin

- RFC 4632 : routage interdomaines sans classes (CIDR) et plan d'attribution et d'agrégation des adresses Internet — la description actuelle du CIDR, son histoire, la table des préfixes et les règles d'agrégation.
- RFC 3021 : préfixes de 31 bits sur les liaisons point à point IPv4 — pourquoi /30 consomme la moitié des adresses d'une liaison et comment /31 l'évite.
- RFC 4291 : architecture d'adressage IP version 6 — les règles d'adressage IPv6, dont la même notation des préfixes avec barre oblique.
- ipaddress : bibliothèque de manipulation IPv4/IPv6 — la référence Python pour analyser, diviser, comparer et agréger des préfixes.
- nft(8) : manuel nftables — la référence nftables, dont l'ordre des règles et les verdicts terminant une chaîne.
- wg(8) : définir et lire la configuration des interfaces WireGuard — le manuel WireGuard, où AllowedIPs est défini comme une liste d'adresses avec masques CIDR.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que le CIDR ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/cidr/