Engineer

CIDR

⊞ 共通する先頭ビット数と開始アドレスでIPアドレスの範囲を表す方式

どのようなものか

CIDRはClassless Inter-Domain Routingの略で、IPアドレスのまとまりを、開始アドレスと、その範囲の全アドレスが共有する先頭ビット数で表す方式です。おなじみのスラッシュを使って記述します。198.51.100.0/24は、先頭24ビットが198.51.100.0と一致するすべてのアドレスを意味し、198.51.100.0から198.51.100.255までの256個になります。スラッシュの後の数字をプレフィックス長、範囲そのものをプレフィックスと呼びます。

classless（クラスを使わない）という語は、それ以前の方式に由来します。RFC 791の元のIPv4計画には三つのネットワークサイズがあり、現在のCIDRを記述するRFC 4632は、なぜうまくいかなかったかを説明しています。利用可能なホストアドレスが254個のクラスCは中規模の組織には小さすぎ、最大65,534個のクラスBは大きすぎました。それでも、選べる中ではクラスBが最も適していました。1980年代後半にネットワークの商用化が進むと、クラスBへの需要は指数関数的とも見える勢いで増えました。

1992年1月、IETFのROADグループ（Routing and Addressing）は、クラスB空間の枯渇、ルーターや運用者が扱える大きさを超えるルーティングテーブル、最終的なIPv4自体の枯渇、という三つの問題を挙げました。最初の二つは1993年から1995年の間に深刻化すると予測されていました。CIDRはその二つへの回答として、1992年6月のRFC 1338で初めて提案され、続いて1993年9月のRFC 1519に示されました。2006年8月には、RFC 4632がそれを置き換えました。三つ目を解決するためのものではなく、時間を稼ぐためのものでした。

CIDR表記について、RFC 4632はプレフィックスを、通常の四つの部分からなるアドレス、スラッシュ、有効なビット数を示す0から32の数字で記述します。従来のネットマスクは、そのビットを1、残りを0にして同じことを表し、/24なら255.255.255.0になります。RFCが維持する制約は、マスクで1になっているビットが連続していることです。その範囲で、プレフィックスは二の累乗個のアドレ

スを持つブロックを表せます。RFCの表は、一つのアドレスを持つホストルートの/32から、二つのアドレスのポイントツーポイント接続向けの/31、256個の/24、65,536個の/16を経て、既定の経路として使う、全4,294,967,296個のアドレスを含む0.0.0.0/0までを示しています。

この表記は、対象として生まれたアドレスファミリーを越えて使われています。IPv6アドレスの構造を定めるRFC 4291も、同じ方法でIPv6プレフィックスを記述します。そのため、2001:db8::/32もIPv4の場合とまったく同じように読み取れます。

概念を判断に生かす

「CIDRとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 Wi-Fi、ファイアウォール、リモートアクセスを整え、どこからでも働け、権限のある人だけが接続できるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Ubuntu Linux VPSインスタンス30台を管理 — 30 復旧計画とともに管理したLinuxサーバー

Ubuntu Linux VPSインスタンス30台を管理 お問い合わせ →

なぜ重要なのか

CIDRは、集約によって当初の問題を解決しました。RFC 4632の説明では、経路の要約は、パケット交換ネットワークが保持する経路情報を減らす方法として、唯一広く理解されているものです。一つの大きなブロックを持ち、その小さな部分から顧客へアドレスを割り当てるプロバイダーは、ブロック全体について一つの経路を広告できます。インターネットの他の部分は、それだけで中のすべての顧客へ到達できます。RFC 4632は、そのためにはアドレスの割り当て自体も変える必要があったと付け加えています。また、旧クラスC空間のうち集約できない部分はswamp（沼地）と呼ばれ、2006年にも推定15,000件の経路を占めていたとしています。

集約が働くのは、すべてのルーターが従う一つのルールがあるためです。RFC 4632は、「インターネットの転送は最長一致に基づいて行われる」と述べています。宛先を含むプレフィックスが複数あれば、最も具体的なものが優先されます。これにより、組織がアドレスを振り直さずにプロバイダーを変えても、到達できる状態を保てます。新しいプロバイダーは組織独自の小さなプレフィックスを広告します。RFCはこれを、旧プロバイダーの集約経路に穴を開けると説明しており、一致するプレフィックスが長いため、集約経路より優先されます。当初の想定期間は三年から五年でしたが、RFCによれば、CIDRは2005年までに広く行き渡っていました。

ドメイン間ルーティングを設定する技術者は少数でも、ほぼ全員がプレフィックスを書きます。オフィスのネットワークを許可するファイアウォールルール、Linuxサーバーのルートテーブル、WireGuardなどです。WireGuardでは、各ピアのAllowedIPsというプレフィックスの一覧が、そのピアから受け入れる送信元アドレスと、そのピア向けトラフィックの送信先の両方を決めます。どの場面でも、スラッシュの後の一桁を間違えるだけで、気付かないうちに何千台ものホストを許可したり除外したりしてしまいます。

実際の活用例

Python標準の `ipaddress` モジュールはこの表記を理解し、ルールに組み込む前に計算を手早く確かめる方法になります。以下は、文書用の198.51.100.0/24の範囲からアドレスを使い、Python 3.15で実行した、この記事の説明用の例です。

```
>>> 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
```

/24は、64個のアドレスを持ち、それぞれ64の倍数から始まる四つの/26ブロックに分かれます。198.51.100.70/26を指定すると失敗します。.70を含むブロックは.64から始まり、26ビット目より後が0ではないためです。 `strict=False` にすると、モジュールはそのビットを0にし、.70が属するブロックを返します。最後の二つの行は、どのプレフィックスの組み合わせでもレビュー時に確認すべき問いです。このアドレスは範囲内にあるか、そして、一つのブロックはもう一つの中に完全に収まるか、を確認します。

二つ目の問いが特に重要なのは、最長一致を使わないパケットフィルターです。ルールを順番に読み、判定が決まった最初のルールで止まります。nft(8)のマニュアルは、`accept` と `drop` がチェーンの評価を終了し、`add` はルールを末尾へ追加し、`insert` は先頭へ入れると説明しています。次も説明用の例で、/24からSSHへの接続を認めながら、その上半分を除外することを目指しています。

```
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
```

二つのルールはどちらも正しく見え、一覧にも両方が表示されます。しかし、/25のすべてのアドレスは/24にも含まれるため、最初のルールがパケットを許可し、その下のdropには決して到達しません。ルーティングなら、どこに置かれていても、より具体的なプレフィックスが優先されます。フィルターでは、先にある場合だけ優先されます。二行を入れ替えるか、狭いほうのルールを `insert` で追加すれば、意図した結果になります。`nft -a list ruleset` は、各ルールをハンドルとともに出力します。`add` も `insert` もハンドルを受け取り、新しいルールを既存のルールの隣へ配置できます。

2026年8月のポートフォリオ記事位置に基づいてファイアウォールルールを検証は、同じ落とし穴を運用の側から説明しています。あるホストには一年間、どのパケットも到達しないレート制限が設定されていました。先に一致する、より広いルールの下に置かれていたためで、要約を表示するどのツールも、制限そのものは正しく示していました。現在のチェックは、番号付きのルール一覧と稼働中のフィルターチェーンを読み、順序を確認します。そのページは、広いルールがどう書かれていたか、アドレスのプレフィックスを使っていたかを記していません。そのため、上の行はそのホストの実際のルールではなく、原則を説明するためのものです。

問題が起きるところ

アドレスにホスト部のビットが残っています。 198.51.100.70/26はマスク付きのアドレスであり、ブロックではありません。Pythonは既定では拒否しますが、`strict=False` なら198.51.100.64/26へ切り下げます。それは、書いた人が意図したホスト群ではないかもしれません。ブロックのネットワークアドレスを書き、厳密に解析して、間違いがエラーとして現れるようにします。

ブロックのつもりで、アドレスだけを書きます。 Pythonの文書によれば、マスクなしのネットワークは/32として扱われます。そのため、オフィスのアドレスをプレフィックスなしで入力すると、許可するのは一台だけです。ホストルートなのか、スラッシュの書き忘れなのかが読み手に分かるよう、/32であっても、必ずプレフィックス長を書きます。

広いプレフィックスが、狭いものより上にあります。 例で示したように、最初の一致で判定するフィルターでは、先のルールが、含まれるプレフィックスを持つ後続ルールを隠してしまいます。最も具体的なルールから置きます。生成するルールでは、`subnet_of` や `overlaps` のような包含関係のチェックで各組み合わせをテストし、後のルールに到達できなければ、ビルドを失敗させます。

プレフィックス長が、数ビット短すぎます。 一ビット減るごとにブロックは倍になるため、/24のつもりで/16を書くと256倍のアドレスを許可し、/0ならすべてを許可します。RFC 4632は、過去に運用者の誤りなどにより、一部のネットワークで/1や/2の経路が観測されたと記しています。保有するブロックよりも短いルールは、すべてレビューの対象にします。0.0.0.0/0と::/0は、すべてに一致させる意図があるルールだけに使います。

アドレス数を、そのままホスト数と数えます。 /24には256個のアドレスがありますが、通常のネットワークでは最初がネットワークアドレス、最後がブロードキャストアドレスで、Pythonの `hosts()` は254個を返します。ポイントツーポイント接続では、この無駄が積み重なります。RFC 3021は、/30が二台のホストに四つのアドレスを使うと指摘し、代わりに/31を使えるようにしています。500本の接続で1,000個のアドレスを節約できます。必要なホスト数に予約済みのアドレス数を加えてブロックの大きさを決め、/31はRFC 3021に対応する機器だけで使います。

集約経路に破棄ルートがありません。 要約したブロックを広告するルーターが、その中の小さなプレフィックスの一つへの経路を失うと、その部分へのトラフィックを、デフォルトルートなどの具体性の低い経路へ送り返し、転送ループを作る場合があります。RFC 4632は、集約するルーターに対して、その集約経路には一致しても、より具体的な経路には一致しないパケットを破棄するよう求めています。実際には、広告する集約経路ごとに、破棄ルートまたはヌルルートを設定することになります。

許可リストがIPv4だけを対象にしています。 nftablesでは、`ip saddr` はIPv4ヘッダー、`ip6 saddr` はIPv6ヘッダーを読みます。Pythonで両バージョンを混ぜたリストを集約すると、組み合わせられないため `TypeError` になります。IPv4プレ

フィックスの一覧は、IPv6のクライアントについては何も定めず、そのクライアントは後続の処理へ進みます。すべてのルールでIPv4とIPv6のプレフィックスを並べて管理し、テスト時は両方を確認します。

参考資料

- RFC 4632：クラスレスドメイン間ルーティング（CIDR）—インターネットのアドレス割り当てと集約計画：歴史、プレフィックスの表、集約のルールを含む、現在のCIDRの説明です。
- RFC 3021：IPv4のポイントツーポイント接続での31ビットプレフィックスの使用：/30が接続でアドレスの半分を費やす理由と、/31でそれを避ける仕組みを説明します。
- RFC 4291：IPバージョン6のアドレス体系：同じスラッシュ表記のプレフィックスを含む、IPv6のアドレス規則を説明します。
- ipaddress：IPv4/IPv6操作ライブラリー：プレフィックスの解析、分割、比較、集約についてのPythonリファレンスです。
- nft(8)：nftablesマニュアル：ルールの順序と、チェーンの評価を終える判定を含む、nftablesのリファレンスです。
- wg(8)：WireGuardインターフェースの設定と取得：AllowedIPsをCIDRマスク付きアドレスの一覧と定義する、WireGuardのマニュアルです。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「CIDRとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/cidr/