第10回 SIProp 勉強会, 2008年6月27日(金)

NAT 基礎講座

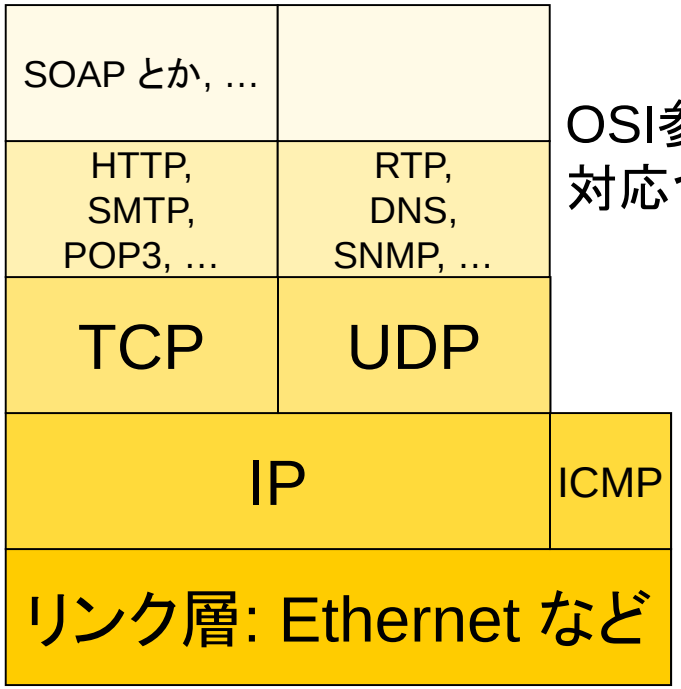
首藤一幸


Beijing
Lanzhou
Tianjin
Huang He
NORTH
Pyongyang
KOREA
Seoul
SOUTH
KOREA
Sea of Japan
Sapporo
Hokkaido
Honshu
JAPAN
Fuji-san
3776
Tokyo

TCP/IP

TCP/IP プロトコルスタック

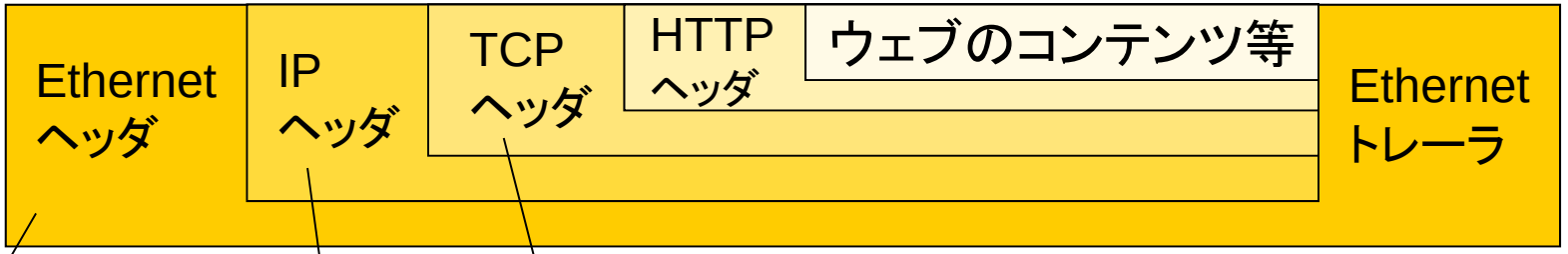
- ≠ TCP と IP の2つ
- インターネットの
  基本プロトコル集合
  - IP (Internet Protocol)
  - TCP (Transmission Control Protocol)
    - connection-oriented / 接続指向
  - UDP (User Datagram Protocol)
    - connectionless
  - ICMP (Internet Control Message Protocol)



L2, L3 という語は
ここから派生

TCP/IP

- 上位層のバイト列を、下位層の形式でくるんでいく。



IP データグラム

(Ethernet) フレーム

MAC アドレス

例: 00:15:58:0B:8D:00

世界中の機器で固有

IP アドレス

例: 61.120.101.31

2001:200:0:8002:203:47ff:fea5:3085

ポート番号

例: 80 とか 443 とか 25 とか

UDP も TCP と同様

TCP/IP に関係する識別子

- MAC アドレス
  - リンク層のアドレス。48 bit。
    - 上位 24 bit を IEEE が機器ベンダに割り当てている。
  - 世界中の機器に対する効率的な routing / forwarding は、無理 → IP の必要性
- IP アドレス
  - インターネット上のアドレス。IPv4: 32 bit、IPv6: 128 bit。
  - だいたい、機器ごとに振られる。
- ポート番号
  - 同一機器上で、通信のエンドポイントを識別する番号。16 bit。
    - 例: ウェブサーバは TCP の 80番、メールサーバは TCP の 25番で接続を待ち受ける。

TCP/IP のプログラミング

- BSD socket API (Berkeley sockets)
  - プロセス間通信のための C言語のプログラミングインタフェース。
    - 4.2 BSD (1983) 由来。
      - System V には Transport Layer Interface (TLI) API があったが、BSD socket が事実上標準となった。
    - TCP/IP プロトコルスタックで言うと、TCP, UDP を使うための API。
    - ネットワークをまたいだ通信だけではない。
      - /usr/include/socket.h

```
/* Protocol families. */
#define PF_LOCAL        1       同一マシン上 (パイプ等)
#define PF_INET         2       IPv4
#define PF_IPX          4       Novel Internet Protocol
#define PF_APPLETALK    5       Appletalk DDP
```

  - 使用例: 接続を受け付ける側

```
int32_t sock = socket(PF_INET, SOCK_STREAM, 0);
struct sockaddr_in sddr;            この構造体の中にポート番号 (&IP アドレス) を仕込む
  …
bind(sock, (struct sockaddr *)&saddr, sizeof(saddr));
listen(sock, 5);
…
int32_t fd = accept(sock, NULL, NULL);
```

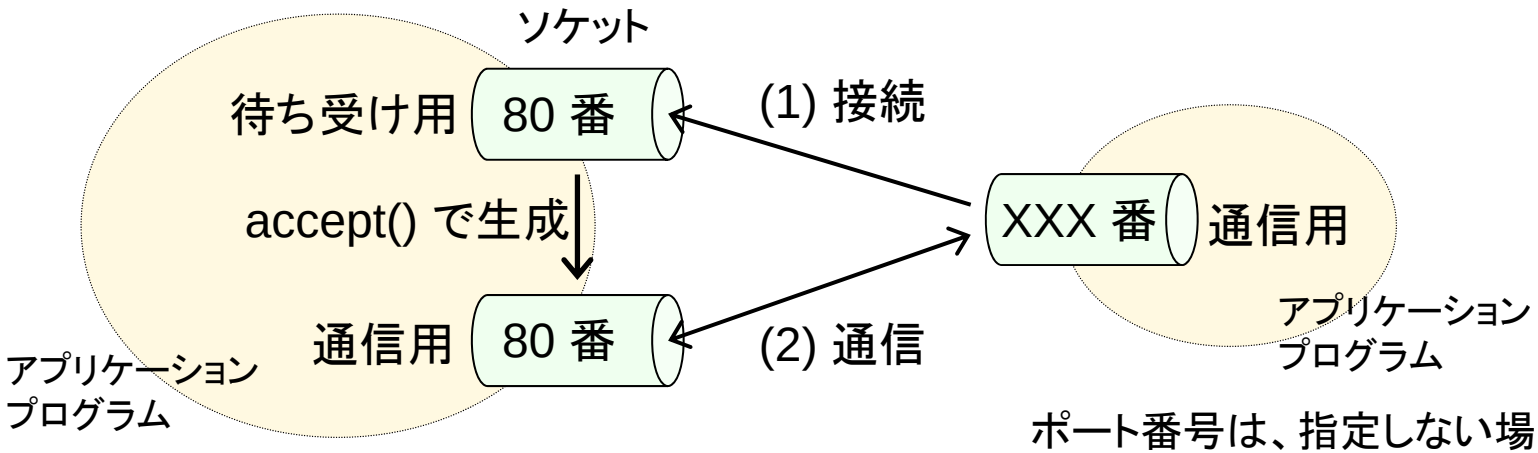
    - Java, Python, Ruby, … 等々では、もっと易しく使える。

ネットワークまわりの様子

- TCP まわり
  - netstat
    TCP fleece:47696 foxhound.shudo.net:ssh ESTABLISHED
  - netstat –n （数値表示）
    TCP 192.168.0.107:47696 218.42.146.77:22 ESTABLISHED
    - SSH（遠隔ログインのプロトコル）は TCP の 22番ポートで接続を待ち受ける。
- UDP まわり
  - netstat –an
    UDP 192.168.0.107:137 *:*
    UDP 192.168.0.107:138 *.*
    - NetBIOS over TCP/IP（NBT）関係の何か
    - ポート番号とサービスの対応表: /etc/services

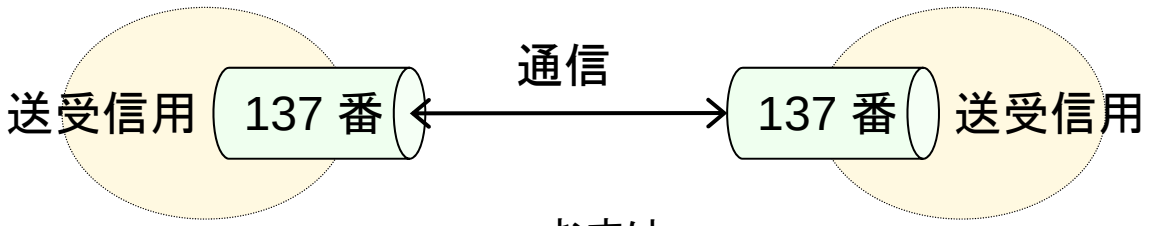
プログラムから見たポート番号

- TCP の場合



ポート番号は、指定しない場合は
乱数で (空いている番号に) 決められる。
TCP でも UDP でも同様。

- UDP の場合



おまけ:
ソケットはカーネル (or ライブラリ) 内のデータ構造。
アプリケーションからは整数値として見える。
この点、ファイルディスクリプタと同じ。


Beijing
Lanzhou
Tianjin
Huang He
NORTH
Pyongyang
KOREA
Seoul
SOUTH
KOREA
Sea of Japan
Sapporo
Hokkaido
Honshu
JAPAN
Fuji-san
3776
Tokyo

NAT

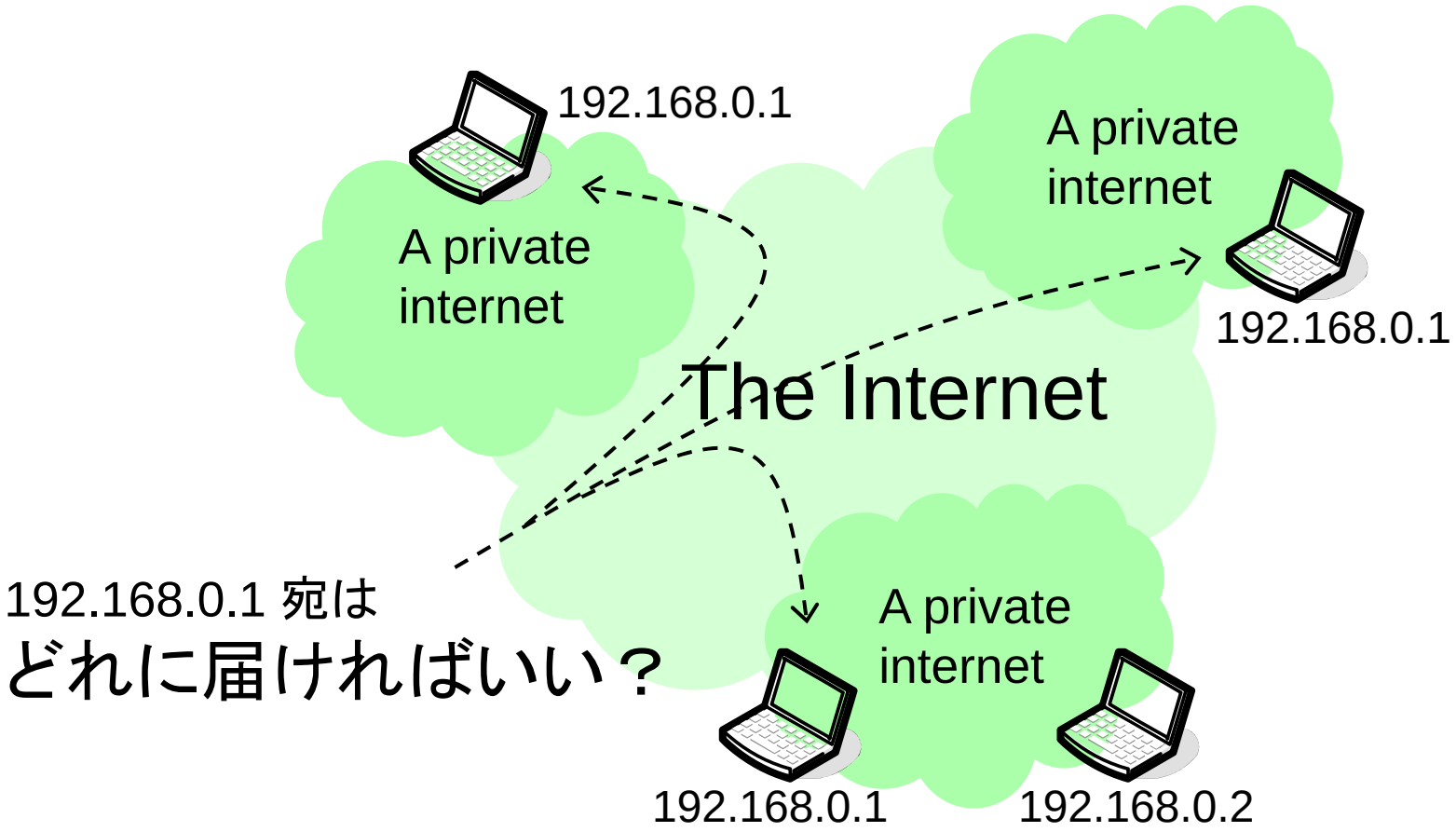
Private Address Space

- RFC1918: Address Allocation for Private Internets
  - 1996年 2月 発行。
  - 組織内で、外界 (the Internet) とは独立にネットワークを構築するために使える IP アドレスを挙げている。
    - 10.0.0.0 ～ 10.255.255.255（10.0.0.0/8）
    - 172.16.0.0 ～ 172.31.255.255（172.16.0.0/12）
    - 192.168.0.0 ～ 192.168.255.255（192.168.0.0/16）
  - public な Internet との相互接続性がない。けど…
  - public な IP アドレスの割り当てを受けずに使える！
    - ISP には福音だった。
      日本では、IP アドレスの割り当てがずいぶんと厳しい時期があった。
      NAT を使うことで、割り当てを受けた数よりも多くのお客をさばける。
    - NAT があれば、ウェブもメールも使えるし、private でいいじゃん！

RFC のページ: http://www.ietf.org/rfc

Private Address Space

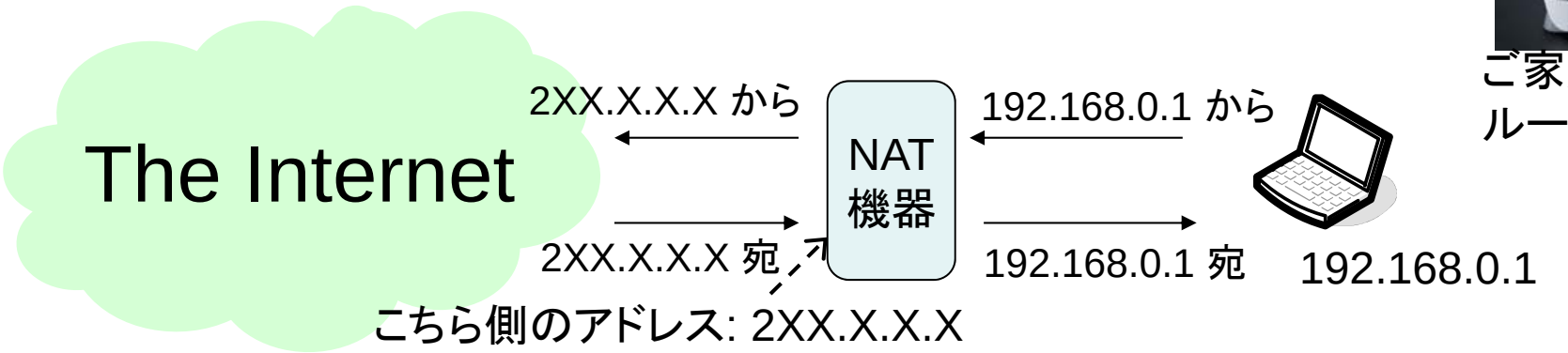
・ 外界との相互接続性は、ない。



Network Address Translation/or (NAT)

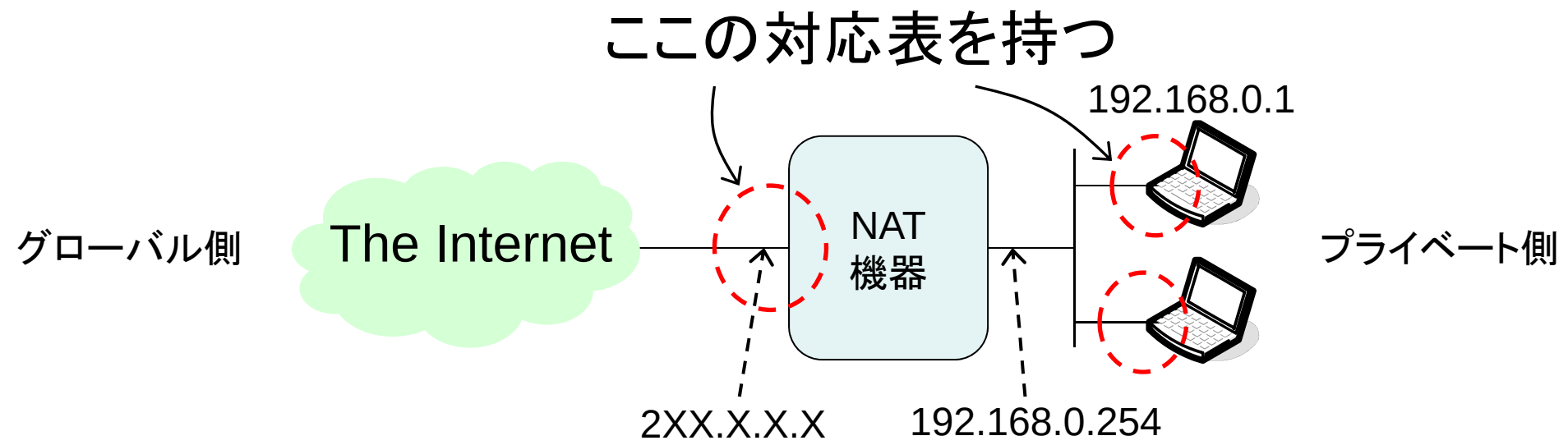
- 末端の機器 (PC) とインターネットの間で、通信パケット中の IP アドレス (やポート番号) を書き換える技術 / 機器。
  - プライベートアドレスを振られた機器が、外界 (インターネット) と双方向に通信できる。
  - 現在、ご家庭用を含めて、あらゆるルータが持っている機能。
  - 細かい分類
    - 静的 / 動的
    - 狭い意味の NAT / NAPT (IP masquerade)

ご家庭向けのルータ



NAT 機器のつくり

- IPアドレス / ポート番号の対応表を持つ。
  - NAT table と呼ばれる。
  - プライベート側機器のアドレス / ポート番号と、グローバル側の自分のアドレス / ポート番号の組
  - 内外対応エントリの例: 2XX.X.X.X : 22 ⇔ 192.168.0.1 : 47696



NAT の種類

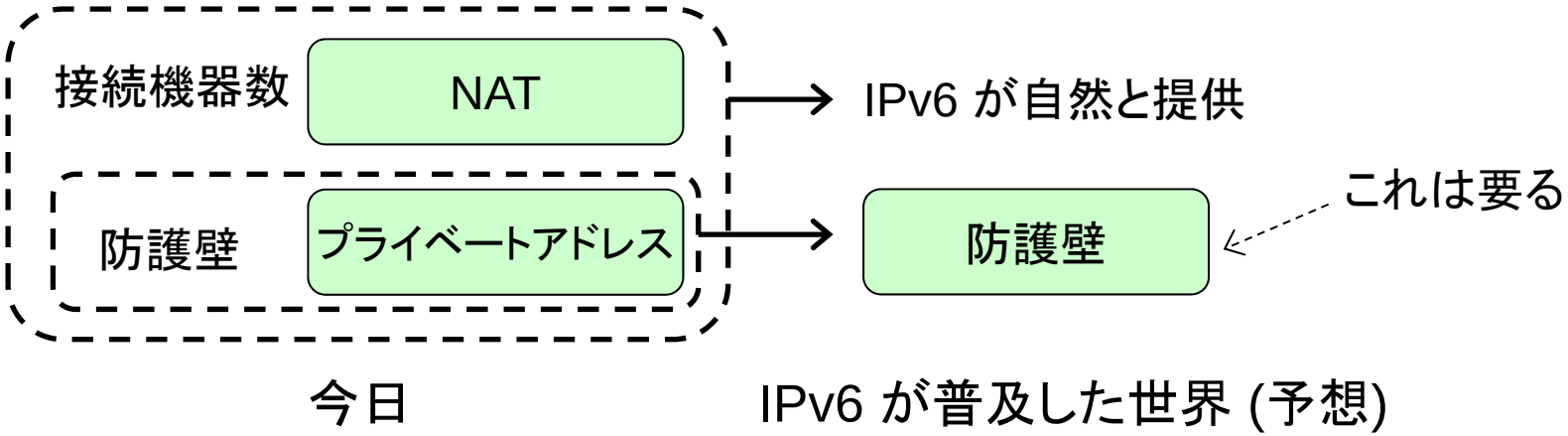
- 静的 / 動的
  - 静的: 手で事前に設定しておく
    - プライベート側にサーバを設置する際、や、特定のアプリケーション (ゲーム, IM) を使いたい場合など。
  - 動的: 対応表に自動的にエントリが追加されていく
    - プライベート側からグローバル側に向けて通信があった時点で、表にエントリを追加する。
      - その際、NAT 機器はグローバル側のポート番号を自動的に割り当てる。
        - ≫ 発展: 何番を割り当てるか？という方式によって、NAT 越えの難しさがまったく変わってくる。
    - 一度エントリが追加されれば、双方向に通信できる。
      - TCP: 双方向通信可能, UDP: 返信可能

NAT の種類

- 狭い意味の NAT と NAPT (IP masquerade)
  - 狭い意味の NAT
    - IP アドレスだけを置き換える。
    - グローバル側 IP アドレスの数と同数までの、プライベート側機器をサポートできる。
      → ご家庭ではあまり役に立たない。
  - NAPT: Network Address Port Translation/or
    - ポート番号も置き換える。
    - Linux に実装された IP masquerade 機能が有名になったので、当初は「NAT」に対して「IP masquerade」と区別された。
- 俗にはNAPTも含めて「NAT」と書かれる。
  念のため「NA(P)T」と書くことも多い。
  が「NAT」でいいのではないか？
- 以降の内容は、「動的 NAPT」が対象。

NAT と IPv6 の関係

- IPv6 が普及した世界では NAT は要らないのか？
  - NAT は、インターネットに接続できる機器を増やすために導入されてきた。
  - IPv6 では IP アドレスを潤沢に使えるため、NAT は不要？
  - いやいや、インターネット側からご家庭内に向けて通信 / 接続できてしまっては怖い。~~NAT~~ プライベートアドレスの使用が、ちょうど防護壁 (firewall) の役割を果たしている。
    - 注意: NAT は防護壁ではなく、逆に、通信を通すための手段。
- IPv6 が普及した世界でも、防護壁は要る。
  - ただのフィルタリングかもしれないが。

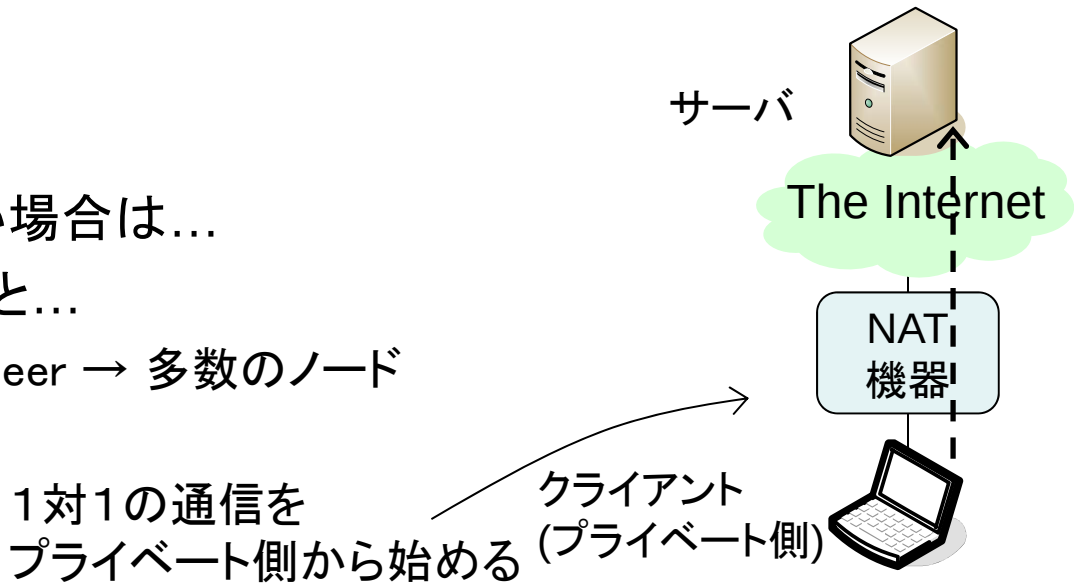


NAT traversal (NAT 越え)

NAT があってもなお困ること

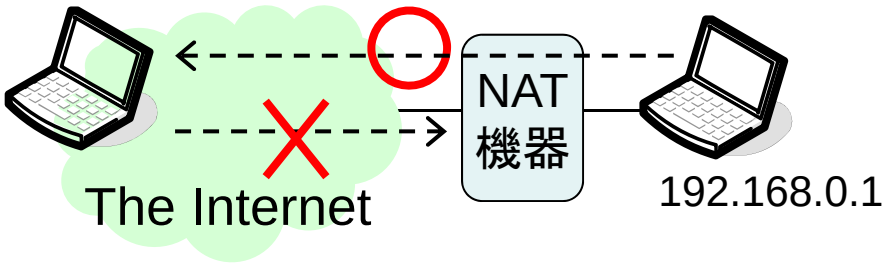
- １対１の通信をプライベート側から始める分には、動的 NAPT でほぼ事足りる。
  - サーバ-クライアント型アプリケーションは、この範疇。
    - 例: ウェブ, 電子メール 他、多くのインターネットアプリケーション
  - やりとりを始めた後は、双方向に通信できる。
  - アプリケーションとしては、自分から見える IP アドレス / ポート番号を信じていればいい。

- しかし…
  - 双方向に通信を始めたい場合は…
  - ３台以上が関係し始めると…
    - ユビキタスや peer-to-peer → 多数のノード



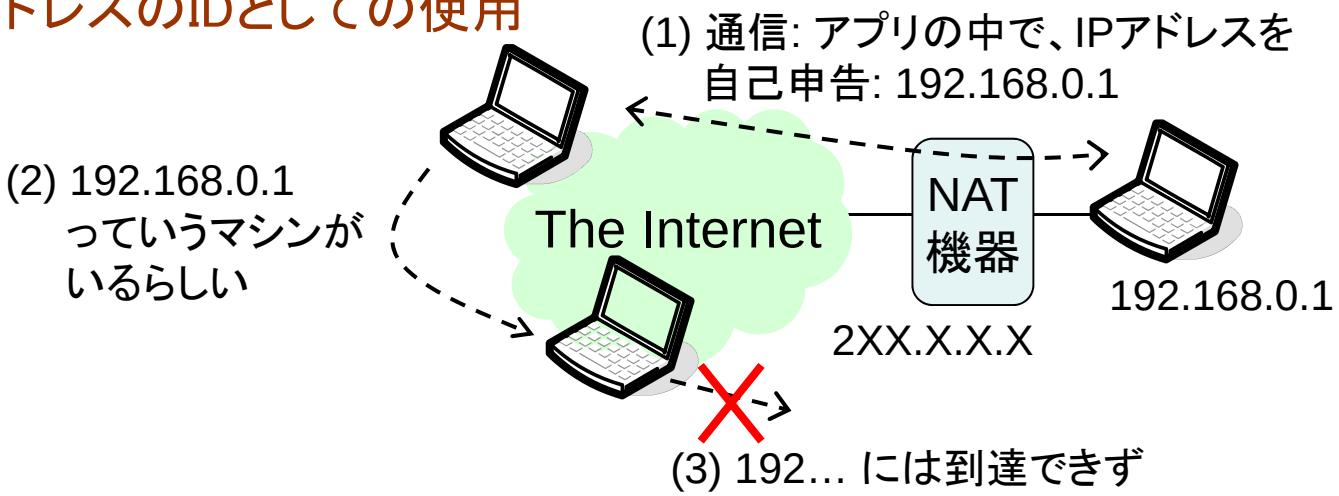
NAT があってもなお困ること

- 場合１ プライベート側に対する接続性
  - グローバルの側からプライベートの側に向けては、接続 / 送信ができない。
    - グローバルな IP アドレスがないため。
    - 静的 NAT（手での事前設定）で解決できることはできる。
  - peer-to-peer で困りがち。
    - peer-to-peer: 各ノードが対等で、サーバでありクライアント。
      → どのノードも接続を受け付ける。例: Instant Messenger

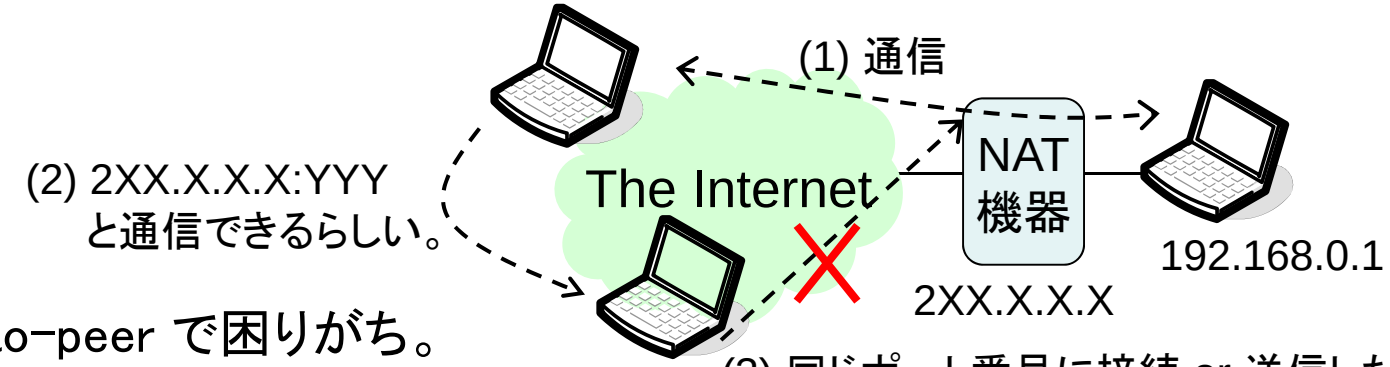


NAT があってもなお困ること

- 場合２ IPアドレスのIDとしての使用



- 場合３ NAT機器によるアクセス制限（UDP限定）



- やはり、peer-to-peer で困りがち。
  - １対１の通信だけでは済まないため。

NAT traversal の技術を使う以前に

- 場合2の反省
  - アプリケーションプログラムの中で IP アドレス (& ポート番号) を扱う場合、自身の IP アドレスを、実際のものとは別だと思い込む機能が要る。
    - 場合1: 実際の (ネットワークインタフェースに付いている) IPアドレスは 192.168.0.1 だったが、自身は 2XX.X.X.X だと思い込む必要がある。
    - とはいえ、どうやって、NAT機器のグローバル側のアドレスを調べるのか？
      → NAT traversal 技術の範疇 (後述)

NAT traversal（NAT 越え）

- 何ができるのか
  - プライベート側のアプリケーションが、自身NAT機器のグローバル側の IP アドレス & ポート番号を知ることができる。
    - 場合2の問題を解決。
  - グローバル側から、プライベート側のアプリケーションに対して接続 / 送信できる。
    - 場合1～3の問題を解決。

NAT traversal (NAT 越え) の手法

利用の優先順位 (およそ)

1. UPnP NAT traversal (UPnP を使った NAT越え)
2. connection reversal (TCP での相手からの逆方向接続)
2. UDP hole punching
3. 中継
   - NAT よりも厳しいファイアウォールを越える際にも使う。例: HTTP プロキシ

- 適切な通信手段の選択:
  ICE (Interactive Connectivity Establishment)

UPnP NAT traversal

- UPnP: Universal Plug and Play
  - 機器を接続するだけでネットワークに参加できるようにする技術の総称。
    - IPアドレスの取得: DHCP → AutoIPでの自動決定
    - 他の機器の発見や利用、制御
  - 機器情報等はすべて XML 文書で表現される。
  - 1対1の要求・返答は、HTTP で行われる。
  - Microsoft 社が策定を主導。
    - 家電連携規格 DLNA も、UPnP を採用。

- UPnP NAT traversal
  - UPnP で、ネットワーク上のNAT機器（ルータ）を制御して、NAT 表にエントリを追加・削除する。
    - 手順: 発見 → 制御
    - ほぼすべてのご家庭用ルータが UPnP に対応している。

UPnP NAT traversal

- プログラミング
  - がんばるなら
    - 通信: IPマルチキャストと TCP, HTTP
    - 通信内容の扱い: XML
      - 適当なパーサ (ライブラリ) を使用するか、自前でがんばるか。
  - UPnP のライブラリを使う
    - Windows XP に内蔵
    - CyberLink for Java, C++, C, Perl by 今野氏
      - for Java を使い、ウェブで使い方を調べつつ、1日程度で開発できた。

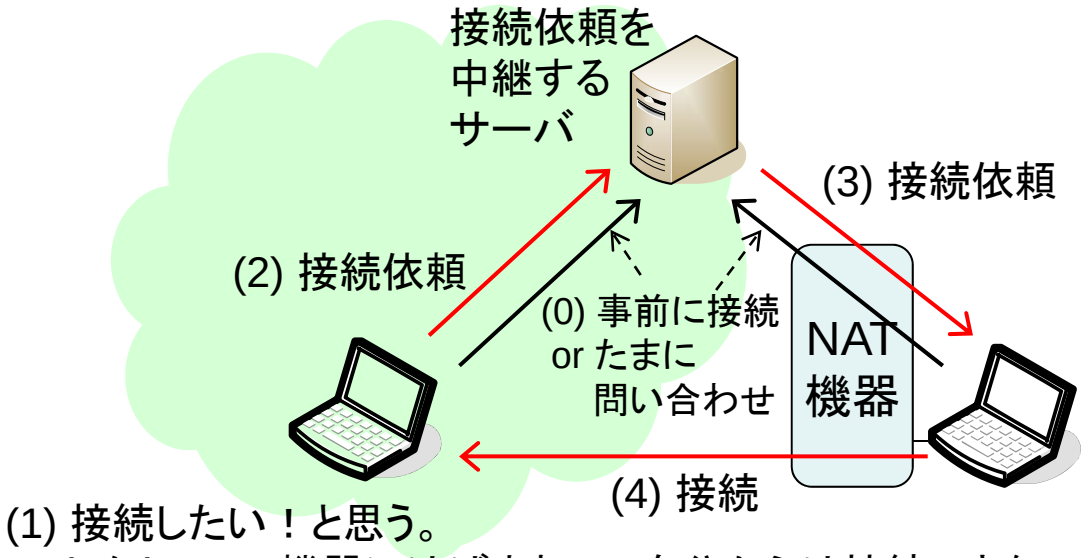
UPnP NAT traversal

- よい点
  - TCP と UDP、どちらに対しても使える。
  - 中継不要。
  - サーバ不要。
  - 一度設定が完了すれば、接続 / 通信ごとの処理が要らない。
  - 確実性が高い。
    - ルータの設定が完了しさえすれば、確実に双方向接続 / 通信できる。
  - 両側が NAT の内側でも通信できる。

- 残念な点
  - ルータが UPnP に対応している必要がある。
    - 現在、ほぼすべてのご家庭用ルータが対応している。
    - とはいえ、対応していても、機能が無効に設定されている場合も。
  - 多段になった NAT を越えられない。
  - 対応具合いのマズいルータがあり…
    - 下手をするとルータがハングアップする場合すらあった。
      → ソフトウェア製品を、あらゆるルータに対して試験する必要がある。
      とはいえ、いまどき Microsoft Messenger も行うので、Windows XP 内蔵の UPnP ライブラリを使えば安心？？？

Connection reversal

- TCP で、相手側（プライベート側）から接続してもらう。
  - 接続したい旨を、中継してもらう。



Connection reversal

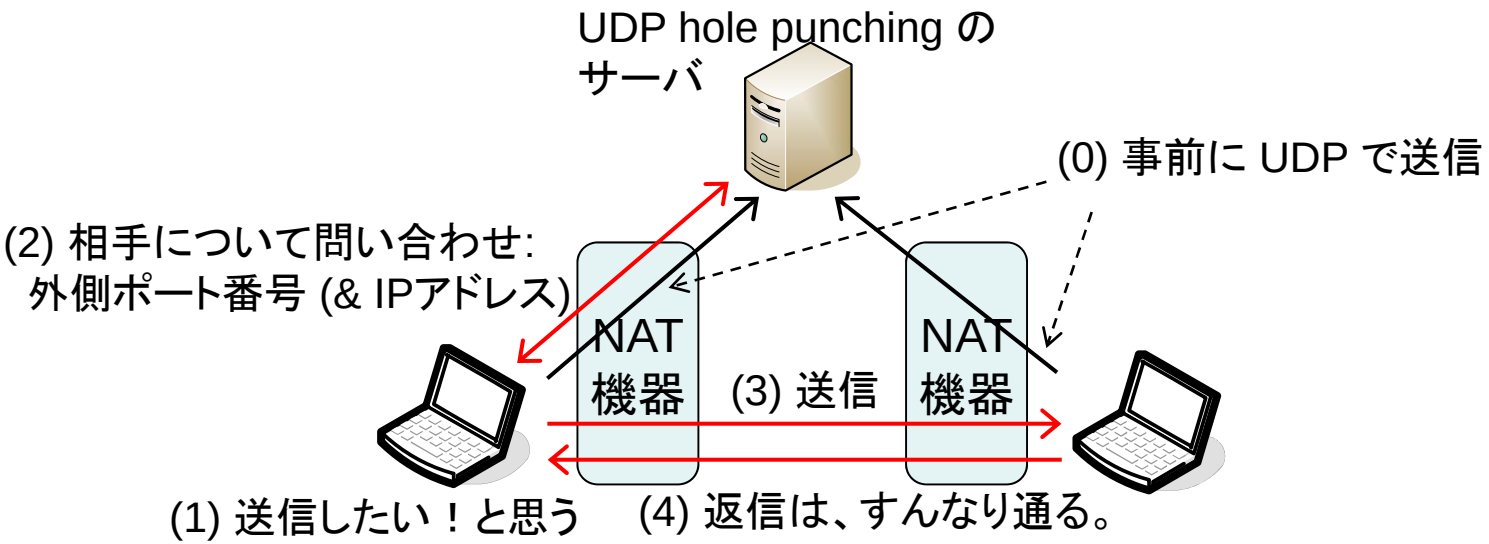
- よい点
  - 中継不要。
  - 越えられない NAT が (ほぼ) ない。
- 残念な点
  - 要 サーバ
    - 中継まではする必要がない。
  - 接続のたびにサーバへの依頼が発生する。
  - NAT エントリの expire (時間切れ) が起きる。
    - 間を空けずに通信し続ける場合はいいが、そうでない場合、エントリ維持に難しい点がある。
  - 両側が NAT 内側の場合、接続できない。
    - 注: UPnP NAT traversal できている場合は、その限りではない。
  - TCP 専用 (UDP に使えない)

UDP hole punching

- 基本的なアイディア
  - NAT 機器内側（プライベート側）から外側に向けて通信しておくことで、**事前に、NAT 機器上に内外対応エントリを作っておく**。
  - NAT 機器外側のポート番号（& IPアドレス）を、UDP hole punching 用のサーバに知らせておく。送信したい場合、**このサーバに対して、相手先の外側ポート番号を問い合わせる**。
- STUN という仕様がある。RFC3489。

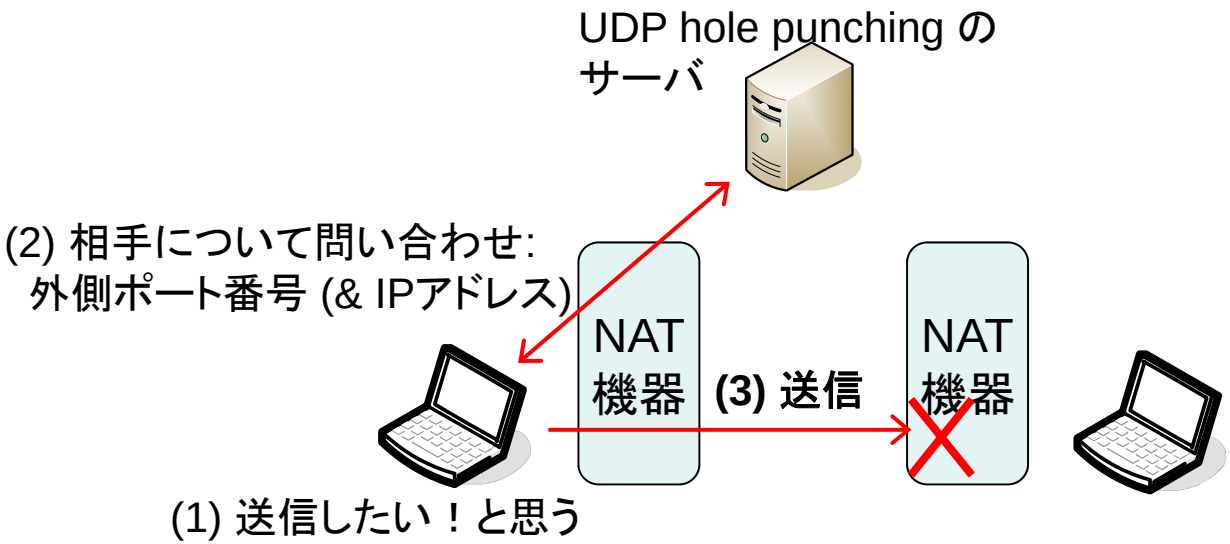
UDP hole punching

- 手順
  - (0) 事前に、UDP hole punching 用のサーバに対して、UDP データグラムを送りつけておく。これにより
    - NAT 機器上に内外対応エントリを作成される。
    - サーバは、NAT 機器外側の IP アドレス & ポート番号を把握することができる。
  - (2) サーバに対して、通信相手の、NAT 機器外側のポート番号 (& IP アドレス) を問い合わせ、返答を得る。
  - (3) 相手の、NAT 機器外側の IP アドレス & ポート番号に対して送信する。
    NAT 内部の相手に到達する。
  - (4) 返信は、すんなり通る。
    (3) の時点で、左側の NAT 機器上に内外対応エントリができているから。



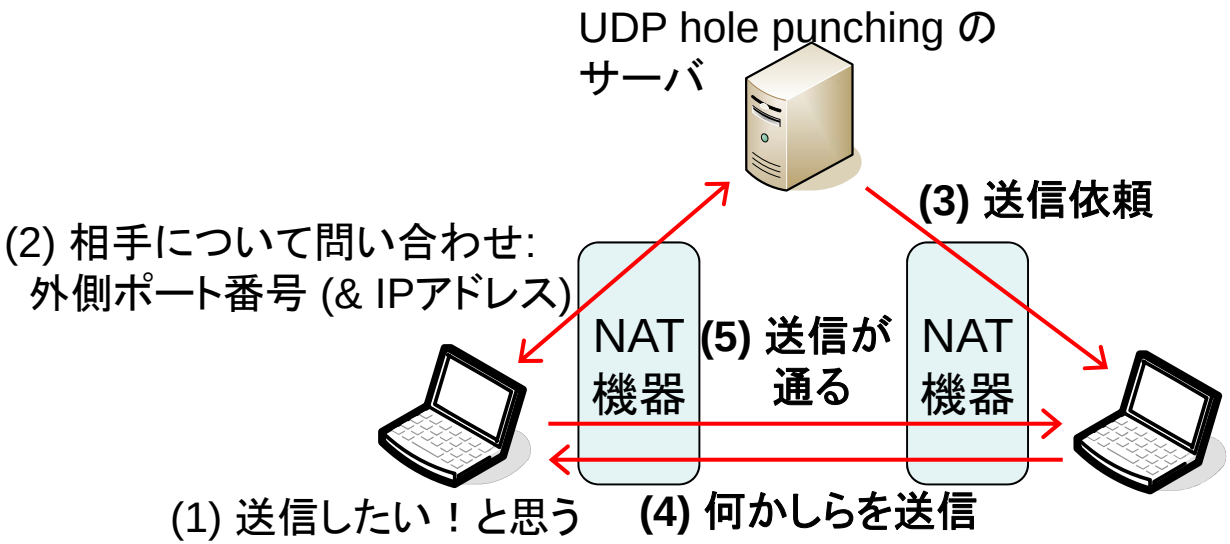
UDP hole punching

- …しかし実は、(3) の時点で、相手に UDP データグラムが到達しない場合がある。
  - 右側 NAT 機器上に、内外対応エントリはできているのだが…
  - 若干、NAT 機器が厳しい: (port) restricted cone NAT
    - 一度も NAT 機器内側から送信をしていない先からデータグラムがやってきた → 通すわけにはいかない
      (送信に対する返信だけを通すような制限を行っている。)



UDP hole punching

- では、どうするか
  - (2) の後、送信先マシンにも、逆方向の送信を依頼する。
  - 逆方向の送信によって、右側 NAT 機器に、内側からの送信の実績ができる → 外から内にも通る。



NAT の種類

- full cone NAT
  - 一度、内外対応エントリができると、外側のどこからの IP データグラムでも通す。
  - サーバが送信時に補助する必要すら、ない。
    NAT 内側のマシンが NAT 外側の IP アドレス & ポート番号さえ把握していれば充分。

UDP hole Punching の困難さ

- restricted cone NAT
  - 内側から送信した実績のある IP アドレスからの IP データグラムだけを、内側に通す。

- port restricted cone NAT
  - ほぼ同上。送信元のポート番号もチェックする。

- symmetric NAT　UDP hole punching できず！
  - 内側から外側への送信時、内外対応エントリを作る際、送信先によって異なるポート番号を割り当てる。
    → UDP hole punching が困難。(可能な場合もあるにはある)
  - ご家庭の 10% 程度？ 調査によって様々な割り合い。

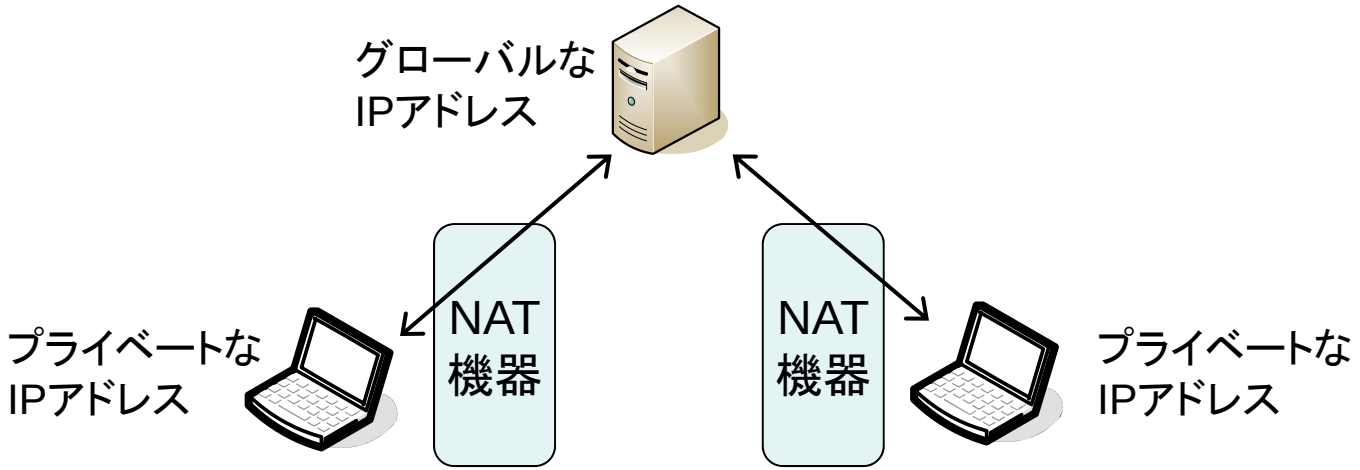
UDP hole punching

- よい点
  - 中継不要。
  - 両側が NAT の内側でも通信できる。
- 残念な点
  - 要サーバ。
    - 中継まではする必要がない。
  - ある相手と通信を始めようとするたびに、サーバへの依頼が発生する。
  - NAT エントリの expire（時間切れ）が起きる。
    - 間を空けずに通信し続ける場合はいいが、そうでない場合、エントリ維持に難しい点がある。
  - 越えられない NAT がある。
    - symmetric NAT
  - UDP 専用（TCP に使えない）

中継

- どうしようもなければ、グローバルなIPアドレスが振られたコンピュータに中継してもらう。
  - 中継機器のネットワーク帯域幅、処理能力を食う。
  - 2者の直接通信ができない場合は、中継するしかない。
    - UDP hole punching を行っていて、ルータが symmetric NAT だった場合
    - Connection reversal を行っていて、両端が NAT 内側だった場合

- TURN という仕様がある。
  - RFCにはなっておらず、Internet-Draftが何本かある。



ICE: Interactive Connectivity Establishment

- 両端で、適切な通信方法を交渉する。
  - IPv4, IPv6, STUN, TURN など、様々な通信手段でのコンタクト先を相手に提示できる。
- SIP, XMPP で ICE を行う仕様が提案されている。
  - SIP: IP電話などのセッション制御用プロトコル。HTTP にならったメッセージ形式。TCP and UDP上。
  - XMPP: チャットのプロトコル。Jabber (チャットソフト) 起源。Google Talk (IM, 電話ソフト) が XMPP での ICE を行う。
- SIP での ICE は、Internet-Draft が何本か出ている。

NAT traversal 手法の比較

・ 一長一短

	TCPとUDP両方	確実性	サーバ不要	対応の広さ	両端がNAT内	標準的な仕様
UPnP NAT traversal	○	○	○	△ UPnP非対応ルータ	○	UPnP
Connection reversal	× TCPのみ	△ エントリのexpire	△ 接続確立時に要サーバ	○	×	
UDP hole punching	× UDPのみ	△ エントリのexpire	△ 通信開始時に要サーバ	△ symmetric NAT	○	STUN: RFC3489
中継	○	○	× 通信自体がサーバ経由	◎	○	TURN

NAT Traversal 手法の選択

- 難しい。
  - 全手法を詳細に理解して、初めて妥当な選択ができる。
  - 目的や状況によって、手法の欠点が問題とならなくなる場合も。
    - 例: UDP が前提であれば、UDP のみのサポートで充分。

- Skype (P2Pインターネット電話), Joost (P2P IPTV) の場合
  - まずは UDP hole punching。だめなら中継。
    - UPnP 非対応ルータがある以上、どうせ他の方法が要る。しかし中継は極力避けたい。→ UDP hole punching
      - 音声トラフィックが常に流れるため (?)、エントリのexpireは起きない。
    - ルータが UDP hole punching に対応していない場合は、他のノードが中継する。
      - 中継はスーパーノードに行わせることで、Skype 運営側の負担はなし。
- ウタゴエ社 UG Live (P2Pストリーミング) の場合
  - UPnP NAT traversal。
    - 理由: TCP で通信すること。いったんルータを設定できた後の確実さ。
    - NAT の外側から内側の PC に対して接続できずとも、なんとかなる。→ 中継は不要。