4rd試験実装技術解説
2012 WIDE 春合宿概要
(事前資料)

JANOG30「IPv6時代のIPv4を考える~第2章~」

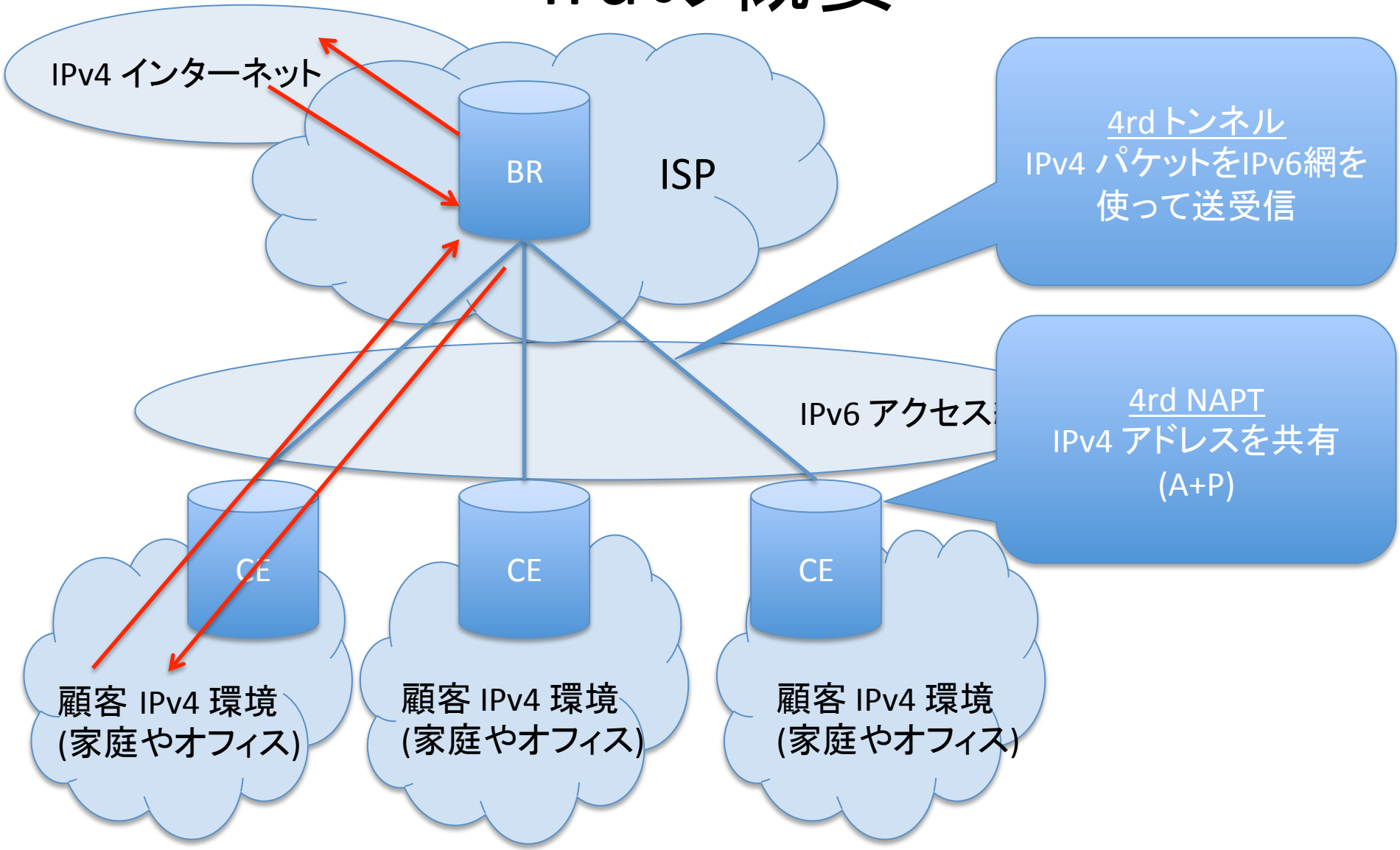
末永洋樹
hsuenaga@iij.ad.jp
株式会社インターネットイニシアティブ

4rd の簡単な解説

4rd 始めました

- どんなもの？
  - 目的1: IPv6環境からIPv4環境へアクセスする技術
  - 目的2: IPv4 のアドレス共有を実現する技術
- どんなふうに？
  - IPv6アドレスとIPv4アドレスをアルゴリズムマッピング
    - v6Addr = f(v4Addr);
    - v4Addr = f’(v6Addr);
- なにがうれしい？
  - 事前に f() と f’() を決めれば、(ISPでは)NATが不要
    - いわゆるステートレス方式

4rdの概要


IPv4 インターネット
BR
ISP
4rd トンネル
IPv4 パケットをIPv6網を
使って送受信
IPv6 アクセス
4rd NAPT
IPv4 アドレスを共有
(A+P)
CE
CE
CE
顧客 IPv4 環境
(家庭やオフィス)
顧客 IPv4 環境
(家庭やオフィス)
顧客 IPv4 環境
(家庭やオフィス)

4rd トンネル

- IPv4 から IPv6 アドレスを生成して P2MP トンネルを構築
  - トンネルの終端アドレスは中身のパケットに応じて変わるので、P2MP
  - 計算するのは、
    - 顧客ネットワークからISPに出ていくとき(自明なので決めうちでも可)
    - IPv4インターネットからISPに入ってきたとき
  - 常に同じ計算結果が得られるようにルールを共有(4rdドメイン)
  - 顧客のIPv4アドレスがユニーク(=グローバルアドレス)であれば、IPv6アドレスもユニークなのでパケットは迷子にならない

v6Addr = f(v4Addr)

* 原理的にはカプセル化ではなく、トランスレーションでもOK

4rd NAPT

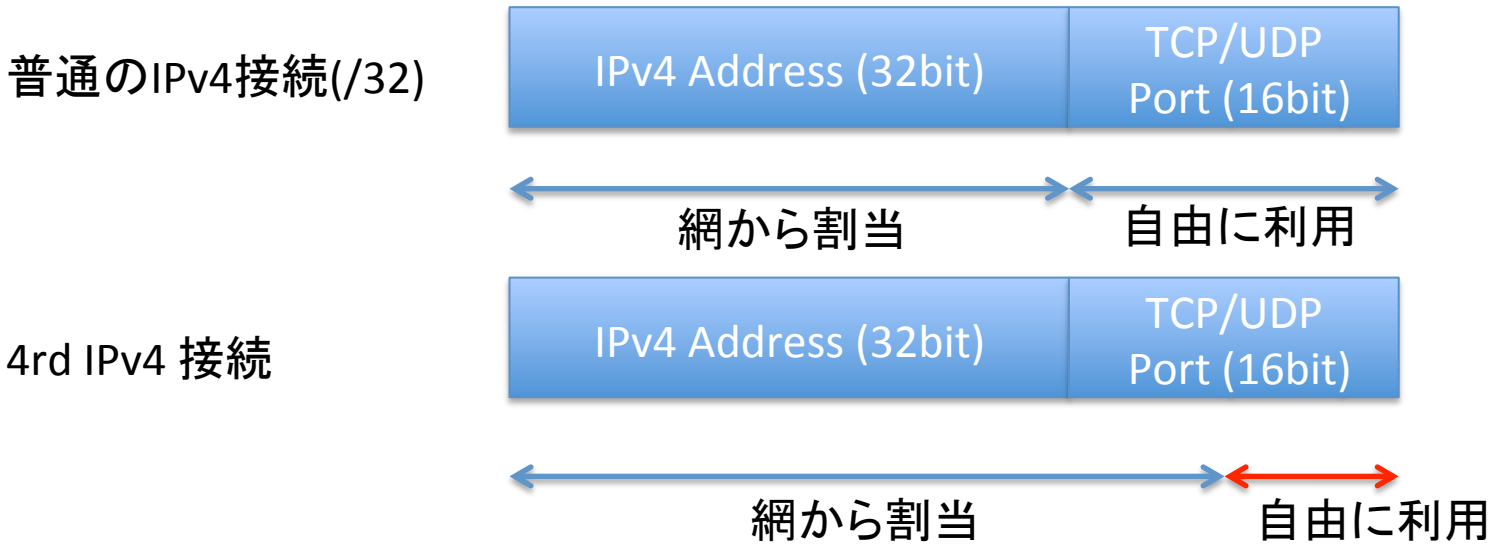
- IPv6 アドレスに対応するグローバル IPv4 アドレスが必要
  - でも、IPv4アドレスは枯渇していることが前提

- NAPTによるIPv4アドレスの共有を最初から想定
  - v6Addr = 4rd(v4Addr, L4Port)　　*v4Addrを共有し、Port の割当を受ける
  - (v4Addr, L4Port) = 4rd'(v6Addr)
- アプリケーションでのサポートは非現実的なので、ゲートウェイがよしなに NAPT する

- IPv4インターネットへの到達確保とアドレス枯渇対策を同時に解決

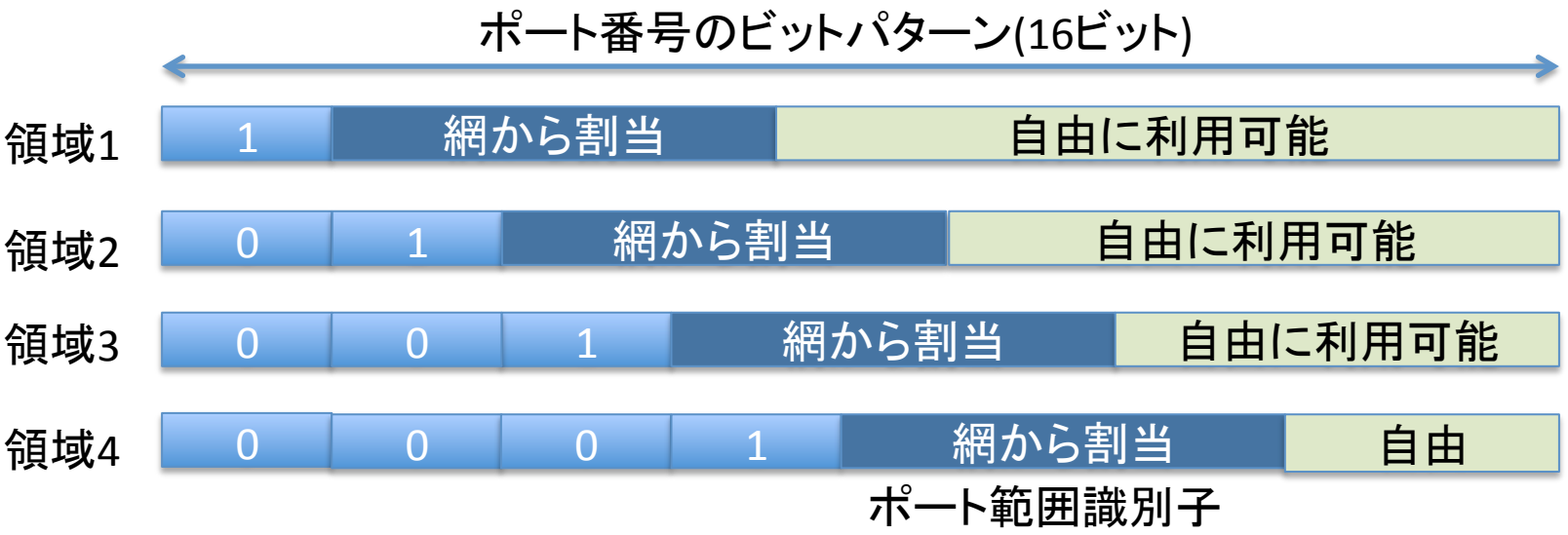
IPv4アドレス共有のアイデア

- ポート番号の一部を顧客の識別に利用しアドレスを共有
- IPv4 アドレス 32bit + ポート番号 16bit の 48bit アドレッシング
  - たとえば、プリフィックスが 40bit であれば、残り 8bit 分のポートを自由に使える。
  - アプリケーションで対応するのは難しいが、NAPTであれば容易。



4rd でのIPv4アドレス共有

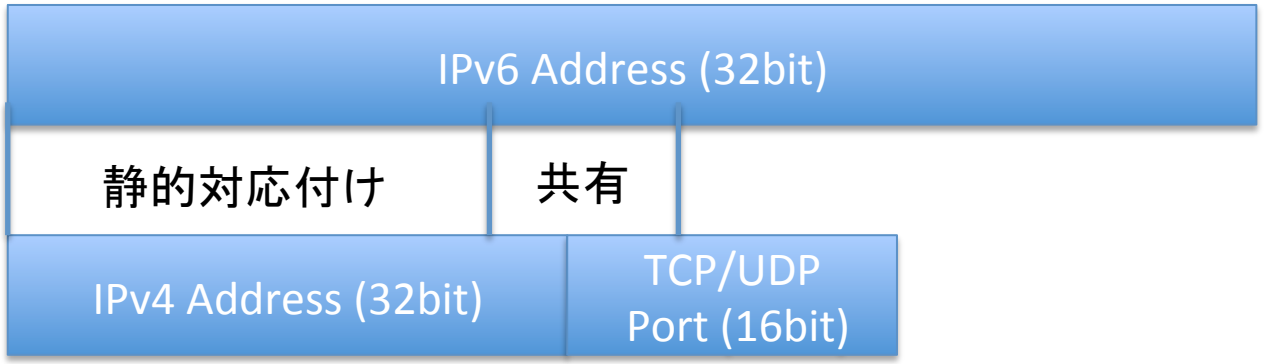
- Well-known ポートを使わせると不公平になるので、ポート番号が1024以下にならないように１ビット使います。
- IPv4 アドレス 32bit + ポート番号 15bit の 47bit アドレッシング
- ポート番号の領域を分割して、平等に覚えにくい(?)
  - 例として網からの割当が 0xEF だとすると、63360-63487、31680-31743、15840-15871、7920-7935 が使えます。



4rd の変換ルール

1. IPv6 のプリフィックスと IPv4 アドレス／ポート(48ビット)の対応を決めます(アドレスブロックの決定)。
   - 2008:db8:aaaa::/52 は 192.168.777.88/28 に対応
2. 上記の範囲ないからIPv6プリフィックスを切り出して顧客に払い出します。このとき、1.のプリフィックス以降のビット列は IPv6 と IPv4 で共通です。
   - 2008:db8:aaa:ff::/60 を払い出すと、IPv4 アドレスとポートの /28 以降の部分も 0xff になります。この場合、IPv4 アドレスは192.168.777.95/32, ポート範囲は識別子 0xf から算出されます。

- この変換にセッション情報は含まれないのでステートレス方式と呼ばれることがあります。



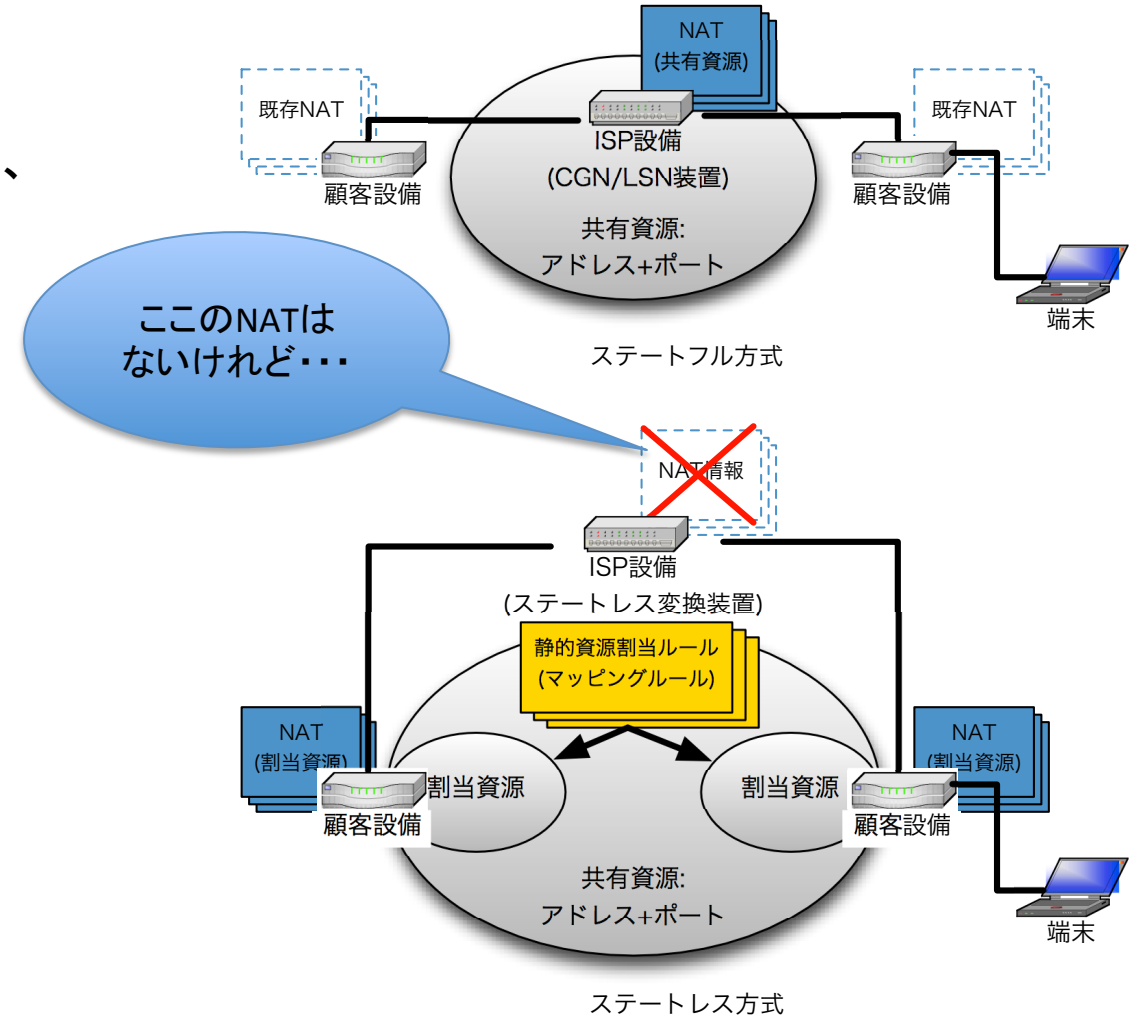
「ステートレス」でもNAPTに注意

4rd では ISP の資源(IPv4アドレスとポート番号)を変換ルールという形でざっくり分配します。この分配はステートレスなので、ISP はラクです。なにより通信ログの管理がラクです。

顧客側は割り当て資源の範囲内で通信をするために、事実上 NAPT が必須となります。

UPnPのようにポート番号を直接触るプロトコルはそのままでは動作しません。気をつけましょう。

新プロトコルへの対応する際は、NAPTと変換ルールの両方が対応しないといけません。



* 合わせ技も可能(464-XLAT など)

WIDE合宿での実験

WIDE合宿で利用した4rd実装

- 以下のI-Dを参考に 2010 年末から実装開始
  - draft-despres-softwire-sam-*
  - draft-despres-softwire-4rd-*

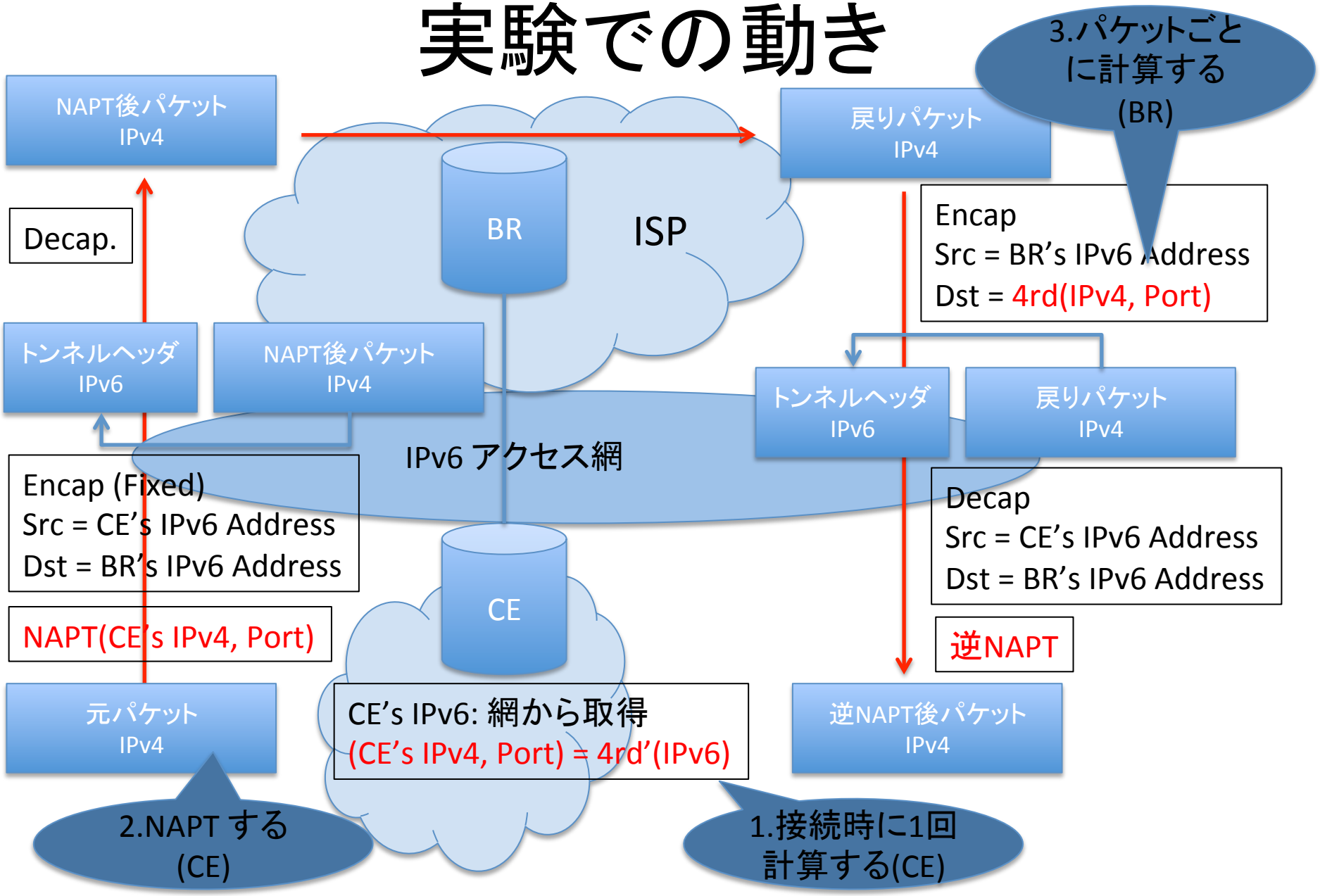
- 最新の 4rd (= 4rd-U) とは別のプロトコル
  - 良い名前がないのですが、「旧 4rd」とでも・・・

- 実装の目的は未来の IPv4 環境の検証・実験
  - 製品・サービスとしてのリリースはもう少し先

実装の特徴

- draft-despres-softwire-4rd-* のマッピングルールを設定可能
- パケットは、IPv4-over-IPv6 トンネルで転送(旧 4rd)
- BR(ISP側), CE(顧客側) の両方で動作可能
- 顧客側でBR相当の動作をさせると、メッシュ型通信が可能
  - 顧客同士の通信(skypeとか)は BR を経由しない
- 既存のNAPT実装にポート制限ロジックをコンパクトに追加
  - TCP, UDP, ICMP に対応

実験での動き


NAPT後パケット
IPv4
戻りパケット
IPv4
3.パケットごと
に計算する
(BR)
BR
ISP
Decap.
Encap
Src = BR's IPv6 Address
Dst = 4rd(IPv4, Port)
トンネルヘッダ
IPv6
NAPT後パケット
IPv4
トンネルヘッダ
IPv6
戻りパケット
IPv4
IPv6 アクセス網
Encap (Fixed)
Src = CE's IPv6 Address
Dst = BR's IPv6 Address
Decap
Src = CE's IPv6 Address
Dst = BR's IPv6 Address
CE
NAPT(CE's IPv4, Port)
逆NAPT
元パケット
IPv4
CE's IPv6: 網から取得
(CE's IPv4, Port) = 4rd'(IPv6)
逆NAPT後パケット
IPv4
2.NAPT する
(CE)
1.接続時に1回
計算する(CE)

問題が発生したところ1

- フラグメント(MTU問題)への対処
  - IPv6 でカプセル化されたパケットへの Pkt Too Big
    - BR でフラグメント処理するの？
    - CE に押しつけてみる？
  - BRに対するIPv6フラグメントパケットの到着
    - リアセンブルする？どれだけためる？
  - BRに対するIPv4フラグメントパケットの到着
    - リアセンブルしてからカプセル化する？
    - フロー情報だけ覚えておく？
  - etc…

問題が発生したところ2

- NAPT の挙動ではまる
  - 普段は大丈夫だけど、フロー数が増えるとおかしくなる問題がでてくる
    - ポート数の制限が厳しいので問題にあたりやすい
  - FTP 対応などの L7 処理のことも忘れずに
  - 暗黙のフィルタ処理などの仕様が変わらないように注意
  - 実装が安直すぎた面もありますが・・・

- ちゃんとした実装とちゃんとしたテストをやりましょう
  - KDE さんの教えを参考に・・・

MAP-E じゃないんですか？

はい。MAP-E じゃないです、すみません。

最近の4rdは？

- 実装&実験しているうちに色々 I-D が出てました
  - draft-despres-intarea-4rd-*
    - 2011/3
  - draft-despres-softwire-4rd-addmapping-*
    - 2011/8
  - draft-despres-softwire-4rd-u-*
    - 2012/1
  - draft-mdt-softwire-mapping-address-and-port-00 (MAP)
    - 2012/4
  - draft-mdt-softwire-map-encapsulation-00
    - 2012/7
- 実験する上ではあまり困っていなかったので積極的には追従していません。

4rdからMAPへ・・・あれ？

- 系統1
  - 「4rd」を起源としつつも、より洗練された「MAP」
- 系統2
  - 4rd を直接的に進化させた 4rd-E/4rd-T/4rd-U
    - Encapsulation, Translation, Unified
  - 古い実装は 4rd-E が近いが、最新のドラフトは4rd-U のみに整理され、「4rd」=「4rd-U」となった
- 現在、どこに向かうのか様子見中
  - もはや我々の実装は根拠となるI-Dが存在しないです・・・
  - が、今後どうなるのか正直良く分からない・・・