仮想化ノードを使用した
非IPプロトコル開発法と経験

金田 泰 (日立)
中尾 彰宏 (東大 / NICT)

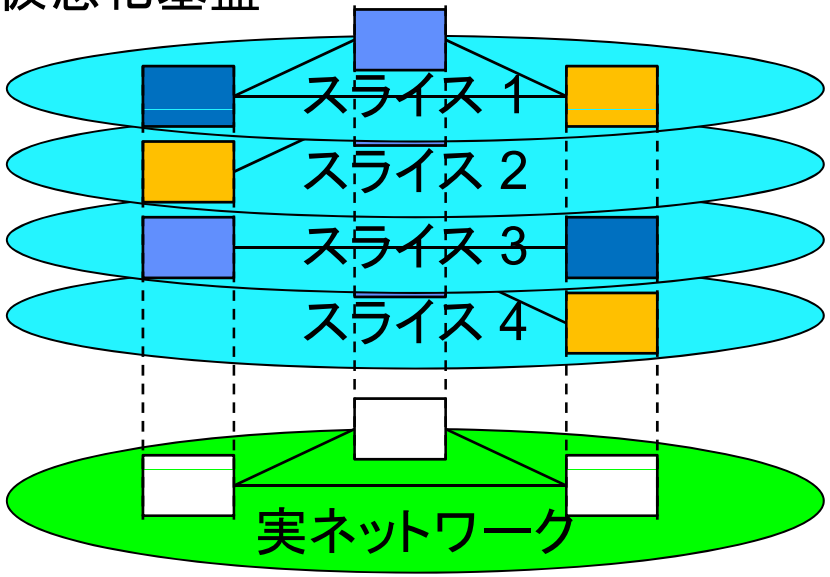
はじめに

- **この発表では “仮想化ノード” 試作機上で実験用非 IP プロトコル IPEC を開発した方法や経験について報告する.**
  - 今後，研究開発用テストベッド・ネットワーク **JGN2plus** に導入された “仮想化ノード” を使用する際などに参考にしていただきたい.
- **仮想化ノード・プロジェクトとネットワーク仮想化基盤について**
  - 情報通信研究機構 (NICT) 中心に，東大，NTT，富士通，NEC，日立の各社が**共同開発・共同研究**している.
  - 既存のインフラを利用して新世代ネットワークを研究するための**ネットワーク仮想化基盤**を開発している.
  - ひとつの物理ネットワーク上で独立かつ自由に設計された**複数の仮想ネットワークが同時に動作**する環境を実現する.
- **非 IP プロトコル IPEC** (IP-Ether-Chimera) **について**
  - 仮想化基盤上で単純で汎用性のある非 IP プロトコルを確立するための第 1 歩として開発した.
  - Ethernet と IP の利点をあわせもつ実験用プロトコルの開発をめざした.
  - プロトコルの機能や実験結果については IN 研究会にて発表ずみ.

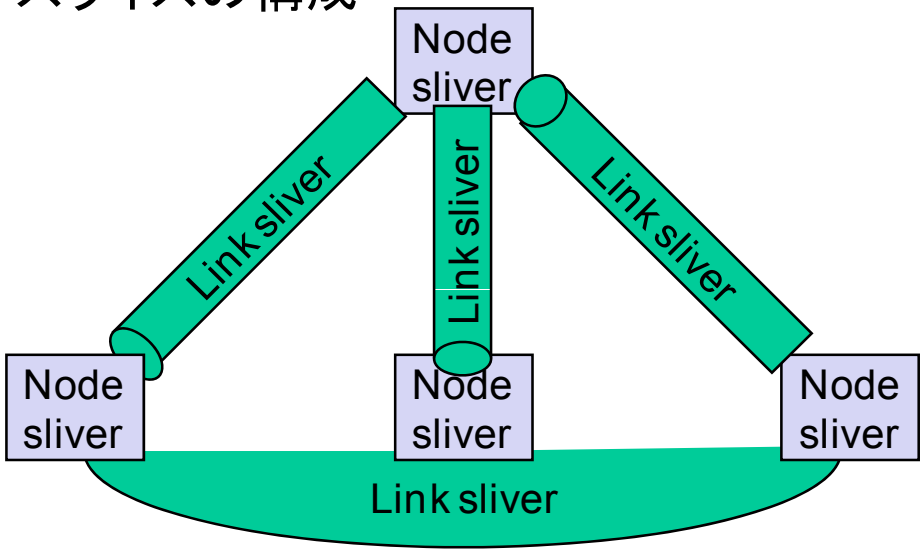
仮想化基盤におけるネットワークの論理構成

- **スライス (slice)**: 仮想化基盤上につくられる仮想ネットワーク.
- **ノードスリバー (node sliver)**: 1 個の仮想化ノードのなかに存在する計算資源.
  - ◆プロトコル処理やノード制御などを実行するのに使用する.
- **リンクスリバー (link sliver)**: ノードスリバー間を結合する仮想リンク. 通常はことなる物理ノード間を point-to-point でつなぐ.

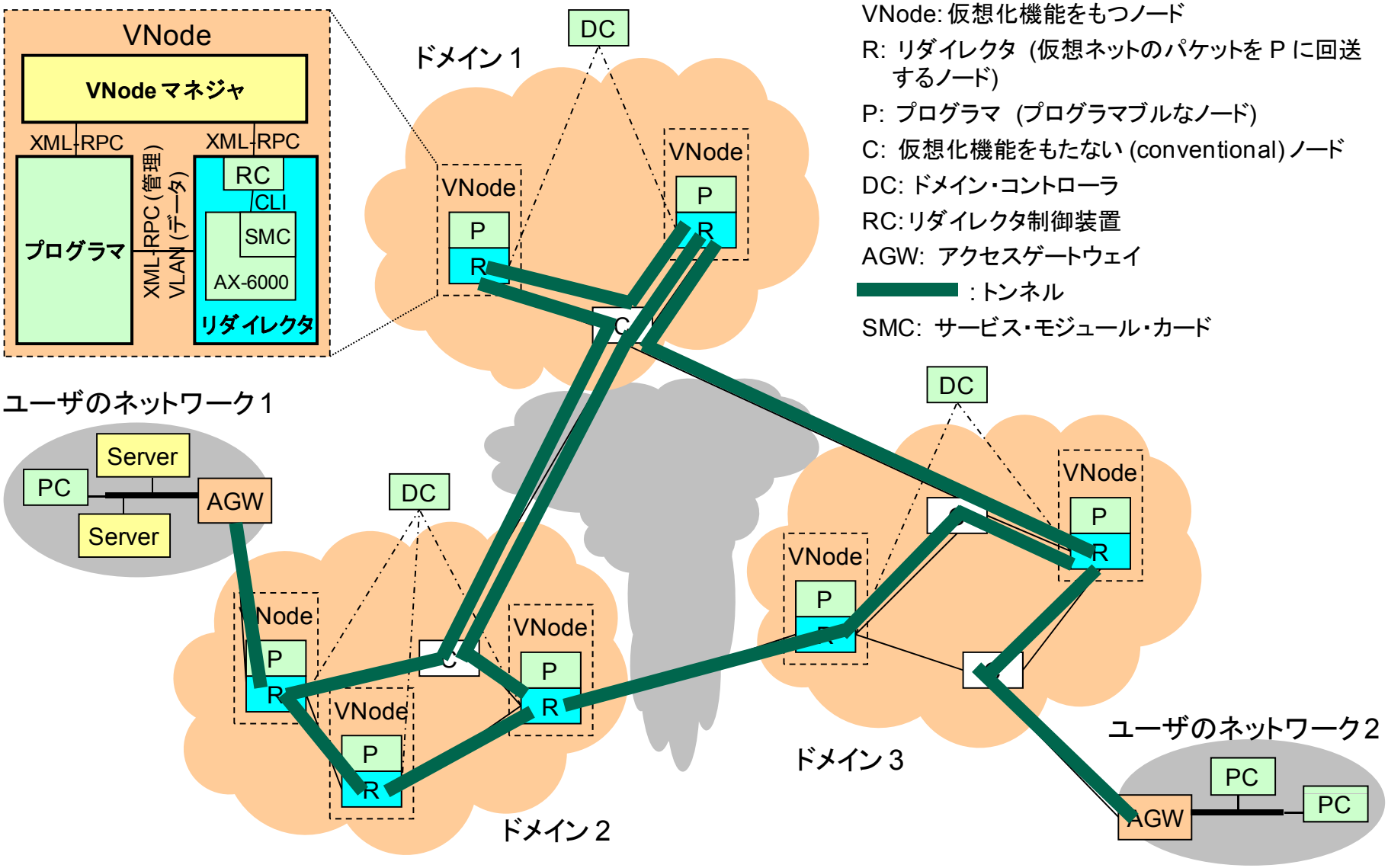
仮想化基盤



スライスの構成



仮想化基盤におけるネットワークの物理構成*

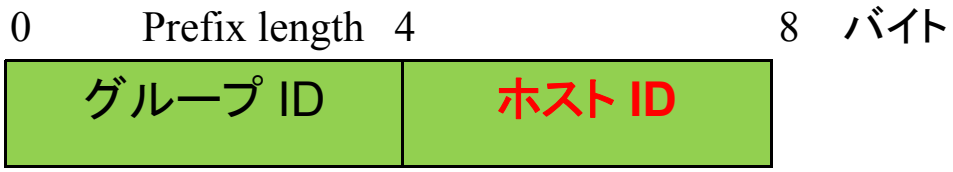


IPEC について 1: 開発目標

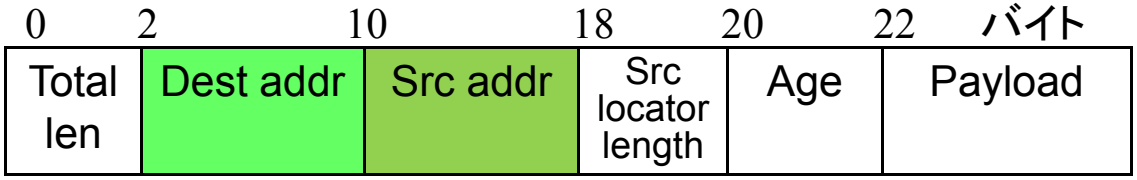
- **新プロトコルの研究:** 単純で汎用性のある非 IP プロトコルの確立をめざした研究への第 1 歩とすること
  - IP 上で実現すると多層化し複雑化する機能を，Ethernet や IP の長所をあわせもつ 1 層の単純な非 IP プロトコルにより実現する.
  - Ethernet スイッチの学習アルゴリズムを拡張し，ループをふくむ任意の構造のネットワークにおいて使用可能な転送アルゴリズムを実現する.
- **仮想化ノードの開発:** 仮想化ノード使用のネットワーク上で非 IP の新プロトコルが開発でき動作するのを実証すること
  - **ユーザビリティの検証**
  - 今後の仮想化ノード使用による新プロトコル開発のテストケースをつくり，そこから開発者に**ノウハウを提供**する.
  - スライスの動作検証
  - 仮想化ノードが新プロトコルの実験に適していることを確認

IPEC について 2: アドレス形式・パケット形式

- **IPEC は Ethernet と IP の利点をあわせもたせることを意図した非 IP プロトコル.**
  - ◆プロトコルの詳細は IN 研究会で発表ずみ.
- **アドレス形式**



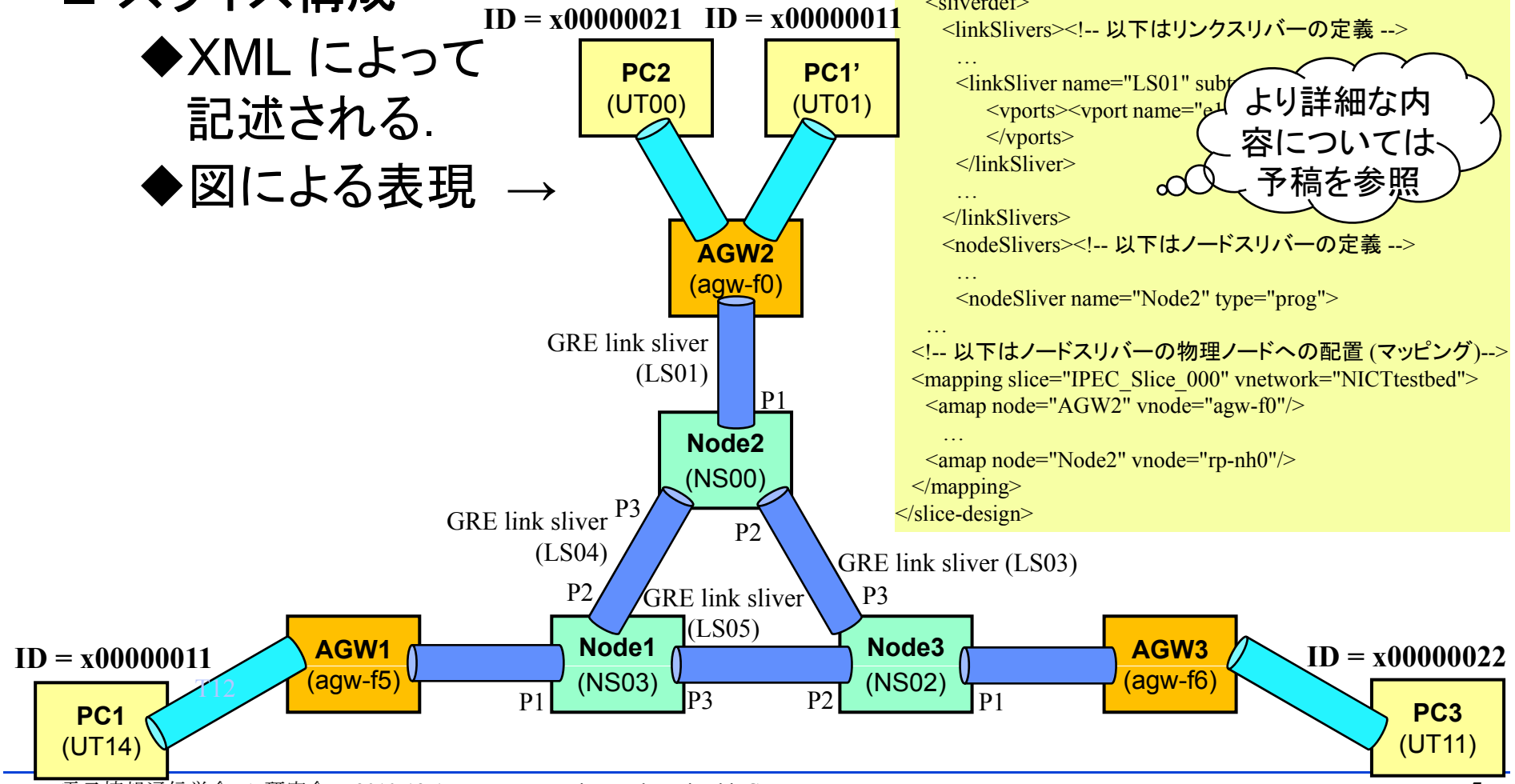
  - ◆**グループID**: アドレスの上位はグループIDすなわち複数個または1個のホストによって構成されるグループのID
- **パケット形式**



  - ◆**年齢:** スイッチ間でパケットが転送されるごとに，1ずつ増加する．ループの存在により重複したパケットの廃棄に使用される.

IPEC の実験環境とスライス構成 1: 概要

- 実験環境は NICT 白山リサーチラボ内の 3 個の仮想化ノードを使用して構築し，動作確認した.
- スライス構成
  - ◆XML によって記述される.
  - ◆図による表現 →



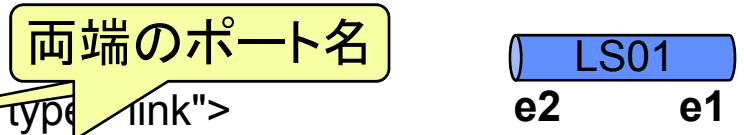

```
<?xml version="1.0" encoding="UTF-8"?>
<slice-design>
  <slicespec name="IPEC_Slice_000">
    <sliverdef>
      <linkSlivers><!-- 以下はリンクスリバーの定義 -->
        …
        <linkSliver name="LS01" subt
            <vports><vport name="e1
            </vports>
        </linkSliver>
        …
      </linkSlivers>
      <nodeSlivers><!-- 以下はノードスリバーの定義 -->
        …
        <nodeSliver name="Node2" type="prog">
    …
  <!-- 以下はノードスリバーの物理ノードへの配置 (マッピング)-->
  <mapping slice="IPEC_Slice_000" vnetwork="NICTtestbed">
    <amap node="AGW2" vnode="agw-f0"/>
      …
    <amap node="Node2" vnode="rp-nh0"/>
  </mapping>
</slice-design>
```

より詳細な内容については予稿を参照

IPEC の実験環境とスライス構成 2: XML による定義

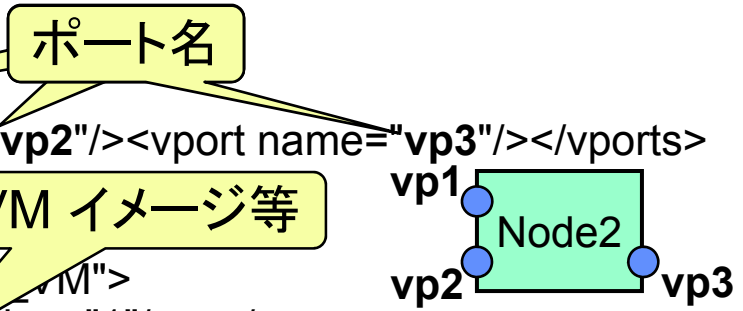
■ スライス定義の構成要素

◆リンクスリバー定義



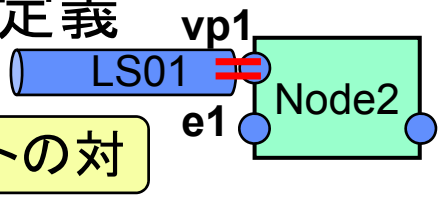

```
<linkSliver name="LS01" subtype="GRE" type="link">
  <vports><vport name="e1"/><vport name="e2"/></vports></linkSliver>
```

◆ノードスリバー定義




```
<nodeSliver name="Node2" type="prog" >
<vports><vport name="vp1"/><vport name="vp2"/><vport name="vp3"/></vports>
<hierarchy><sliverdef><nodeSlivers>
  <nodeSliver name="SP00">…
    <instance subtype="KVM" type="SlowPath_VM">
      <resources><resource keyword="cpu" value="1"/>…</resources>
      <params><param keyword="bootImage"  value="http://…"/></params>
    </instance></nodeSliver></nodeSlivers>…</sliverdef>…</hierarchy></nodeSliver>
```

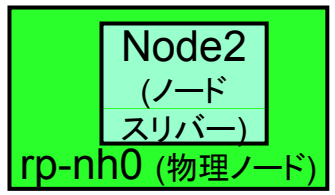
◆ノードスリバーとリンクスリバーとの結合の定義




```
<bind name="w11">
  <vport portname="vp1" slivername="AGW2"/>
  <vport portname="e1" slivername="LS01"/></bind>
```

◆ノードスリバーの物理ノード への配置の定義.

```
<amap node="Node2" vnode="rp-nh0"/>
```



■ スライス定義 GUI が開発されている.

仮想化ノード上のサービスにおける関係者の役割

- **3 つの役割** [NTT11]
  - **◆運用者 (オペレータ)**
    - 物理ネットワーク (仮想化ネットワーク) を管理する.
    - 開発者は運用者が登録する.
  - **◆開発者 (デベロッパ)**
    - 仮想ネットワーク (= サービス) を生成し管理する.
    - ノードスリバーのプログラムを開発し，スライスを定義し，一般ユーザを登録する.
  - **◆一般ユーザ**
    - 仮想ネットワークを使用する.
    - 端末の設定と端末に関する情報は一般ユーザがポータルに登録する.
- **実験の場合は 1 研究者が 3 つすべての役割をはたすことが多い (?)**
- **3 者ともポータル (Web インターフェイス) を介して登録・設定する.**

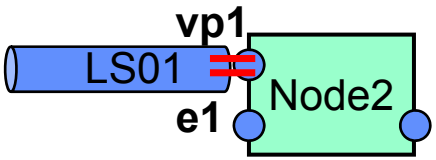
実験手順 1: スライスごとの操作

■ 開発者がポータルにログインして，スライス生成からサービス開始までの操作をする.

… IPEC 固有の部分

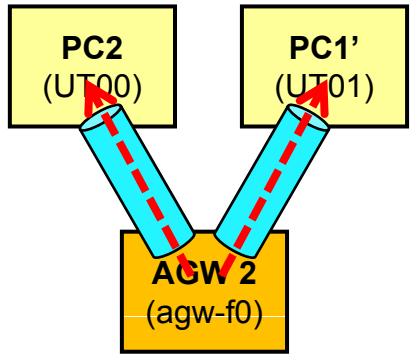
◆ スライス予約
- スライス定義をファイルから入力する.
- ノードスリバーとリンクスリバーが生成されるが，まだ結合されない.

◆ 結合
- ノードスリバーとリンクスリバーを結合する.



◆ 実行
- スライスを実行状態にして，ノードスリバーのプログラム起動やパラメタ設定などができるようにする.
- このときスローパス・ノードスリバーに関しては VM が起動される.

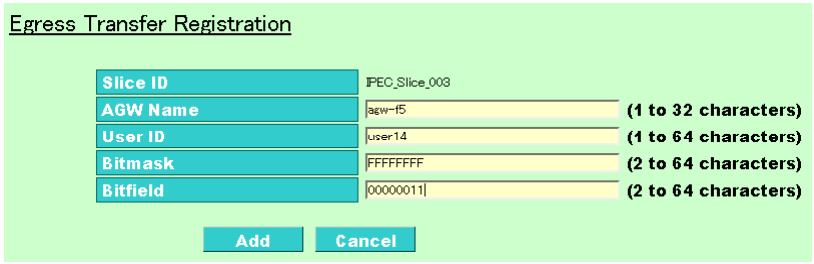
◆ 端末へのパケットふりわけのための設定
- AGW から端末へはパケット・ヘッダがふくむ情報 (IPEC においてはホスト ID) によってふりわける.
- **Egress 抽出設定**: パケットふりわけ用のデータを抽出するパケット上の位置を指定する.



◆ サービス開始
- 登録されたユーザがサービスをうけられるようにする.

実験手順 2: 一般ユーザごとの操作

- 開発者と一般ユーザとが連携してユーザ環境を設定する.
- まず開発者がポータルにてつぎの操作をする.
  - ◆ユーザ登録
  - ◆アクセス設定
    - ユーザのスライスへのアクセスを許可する.
  - ◆**ホスト ID 登録 (Egress 中継登録)**
    - AGW における端末へのパケットふりわけの際に使用するホスト ID とともに, 物理 AGW 名, ユーザ ID を設定する.
    - IPEC では本来この情報を学習したいが, 現在は固定的に登録する.
  - ◆**IPsec 情報設定 (IPsec SA 設定)**
    - AGW から端末へは認証とセキュリティのため IPsec を使用して通信する.
    - IPsec で使用するモード (トンネルなど), プロトコル, 暗号化モード, 暗号化キーなどを指定する.



実験手順 2: 一般ユーザごとの操作 (つづき)

■ つぎにユーザごと (使用する端末ごと) にポータルにログインして，つぎの操作をする.

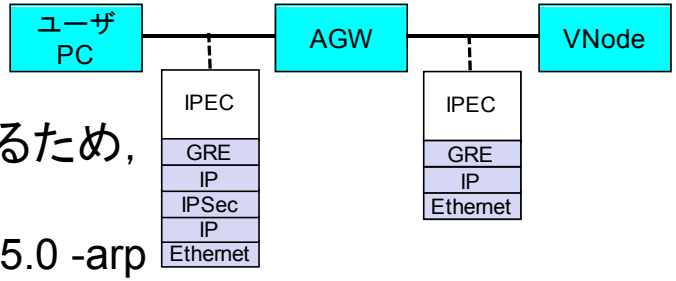
◆ スライスの指定

- 端末 (PC) の IP アドレスと参加するべきスライスとを登録する.

◆ 端末の IPsec 設定

- AGW-端末間の IPsec 関連の情報を端末に設定する.

```
flush;
spdflush;
add 192.168.13.22 192.168.13.11 esp 278 -m tunnel -E 3des-cbc
      "abcdefghijklmnopqrstuvwx" -A hmac-sha1
add 192.168.13.11 192.168.13.22 esp 279 -m tunnel -E 3des-cbc
      "abcdefghijklmnopqrstuvwx" -A hmac-sha1
```



◆ 端末の GRE トンネル設定

- AGW-端末間には GRE トンネルが使用されるため，それを設定する.

```
ifconfig eth0:1 192.168.13.1 netmask 255.25.255.0 -arp
iplink add gretap0 type gretap remote 192.168.13.11 local 192.168.13.1
ifconfig gretap0 up
[ ifconfig gretap0 mtu 1381 ]
```

gretap0 というインターフェースを link up させる必要がある

プログラム開発 1: ノードスリバー用プログラム

■ プロミスキャス・モードによる非 IP・非 Ethernet プログラミング

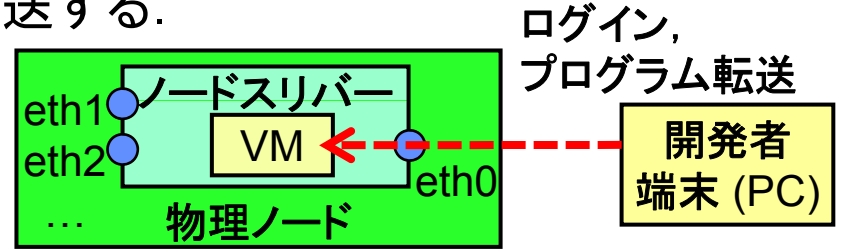
- ◆ **プロミスキャス・モード** (promiscuous mode) を使用して Linux 上でプログラムする (MAC アドレスによらずに転送する).
- ◆ 通常のプロミスキャス・モードとちがって, **Ethernet ヘッダはつけない** (そのかわりに今回は IPEC のヘッダをつける).

■ ノードスリバーのポートと論理インターフェース

- ◆ ノードスリバーのプログラムは論理インターフェイス eth1, eth2, … を使用して通信する (eth0 は制御用).

■ プログラムのクロス・コンパイルとロード

- ◆ 開発者端末などでプログラムをコンパイルし, バイナリ・ファイルをスローパス・ノードスリバー上の VM に転送する.
  - 転送に必要な情報はポータルからえる.

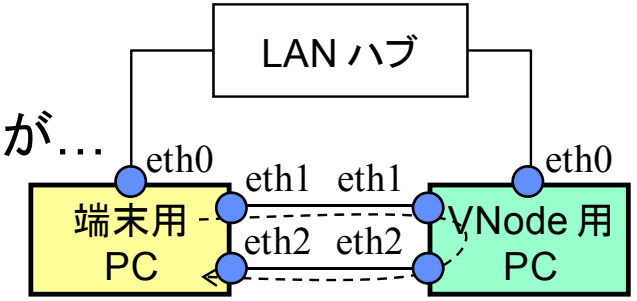


■ 実行の準備

- ◆ この状態では各インターフェイス (eth*i* とする) はリンクアップしていないので, 起動前につぎのコマンドを入力する: ifconfig eth*i* up

プログラム開発 2: Linux マシン上での開発法 [ノウハウ]

- **IPEC のプログラムは途中まで Linux PC 上で開発した.**
  - ◆ 仮想化ノード使用のテストではプログラムの修正などに時間がかかる
    — このようなかたちで予備開発しテストしておくのが有効.
- **今回は PC 2 台だけでテストした.**
  - ◆ 6 台あればより実環境にちかいテストができたが…
  - ◆ 各 PC に NIC を 3 枚搭載した.
    - eth0 は基本的に制御・管理用 (IP で使用).
    - eth1, eth2 はデータ転送用 (promiscuous mode で使用).
  - ◆ 非 IP・非 Ethernet のプロトコルを動作させるには注意が必要.
    - 2 台のデータ転送用 I/F 間には Ethernet スイッチ / ハブは使用できない.



- **1 台の端末用 PC によって 2 台の端末をシミュレートした.**
  - ◆ データ転送用 NIC の数だけの端末が容易にシミュレートできる.
    - NIC を直接指定するので, 意図しない I/F にパケットがながれることがない.
  - ◆ 1 台の PC でノードスリバーと端末とをシミュレートすることもできる.
  - ◆ 注意するべき点はいくつかある.
    - gretap0 のかわりに eth1, eth2 などが使用できるようにプログラムする.
    - CPU 時間を独占しないようにプログラムする必要がある.
    - タイミングに依存するテストには向かない.

プログラム開発 3: 端末用プログラム

- 端末の OS は現在のところ Linux だけ.
- 端末間の通信を可能にするためには端末用のプログラムも開発する必要がある.
- Promiscuous mode によって通信する.
  - ノードスリバーと同様に Ethernet ヘッダなし.
- 端末においてはインターフェイスとして gretap0 を使用する.
  - ノードスリバーにおける eth*i* のかわり.

失敗経験 [ユーザビリティの検証]

- **開発・実験のなかでつまずいたいくつかの点を報告する.**
  - ただし, これらの問題点は近日中に解消されるはず.
- **端末設定において発生した問題**
  - **失敗**: GRE, IPsec のパラメタ誤記によって通信できなくなり, その原因究明に時間がかかった.
  - **対策案**: 非 IP 通信に必要な設定をマニュアルに明記し, 適切なツールを提供.
- **スライス操作において発生した問題**
  - **失敗**: 開発者にもとめられる操作回数や選択肢が多いため, 誤操作した.
  - **対策案**: 操作を単純化し, かつエラー発生時の対策をマニュアルに記述.
- **インターフェイスのリンクアップ**
  - **失敗**: I/F をリンクアップしなくてもエラーが発生しないため, パケットがながれない理由がしばらくわからなかった (ノードスリバーと端末).
  - **対策案**: 自動的にリンクアップするのがよい. できなければマニュアルに記述.
- **32 ビットと 64 ビット混在による問題**
  - **失敗**: 端末プログラムを 32 ビット CPU 専用に書いているのに, 気づかずに 64 ビットの Linux 上でコンパイルして異常な動作をおこした.
  - **対策案**: ノードスリバー・端末の両方について32 ビットか 64 ビットかを明記.

まとめ

- 仮想化ノードを使用した IPEC の開発・実験の手順や経験をのべ，ユーザビリティに関する課題やノウハウ等を示した.
  - **課題**: 開発者によるケアレスミスの防止策の必要性，スライス仕様やスライス操作の実装上の改善点など.
  - **ノウハウ**: Linux PC どうしをつなぐ実験を仮想化ノードを使用した広域実験にスケールアップすれば開発が容易になることなど.
- 仮想化ノードの利点
  - 非 IP プロトコルの 10 Gbps の実験や広域実験，ファストパスを使用する実験などが，仮想化ノードを使用すれば比較的容易に実現できる.
  - まず Linux PC をくみあわせて実験し，プログラムを仮想化ノードに移植すれば，比較的容易にそういった実験にスケールアップできるだろう.
- 失敗経験からまなぶべきこと
  - 失敗経験をシステムやマニュアルに反映させれば，JGN2plus 上でそれを一般ユーザが使用する際に失敗せずにすむだろう.
    - IPEC の開発経験だけでなく，他のアプリケーションについても.
  - プロジェクトは実際その方向にむかっている.