社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
電子情報通信学会 IA 研究会 2010-12-17

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
日立製作所 中央研究所
〒185-8601
東京都国分寺市東恋ヶ窪 1-280
E-mail: Yasusi.Kanada.yq@hitachi.com

中尾 彰宏*†
*東京大学大学院
情報学環・学際情報学府
〒113-0033 東京都文京区本郷 7-3-1
E-mail: nakao@iii.u-tokyo.ac.jp

†情報通信研究機構
〒113-0001
東京都文京区白山 1-33-16

あらまし 情報通信研究機構 (NICT) はネットワーク仮想化技術を使用して自由なフレーム形式をもつ非 IP プロトコルを実装可能にした 10 Gbps 級の仮想化ノード (VNode) の開発を推進している．このプロジェクトでは，VNode が研究開発用ネットワーク JGN2plus に導入され，開発者 (JGN2plus 使用者) のためのユーザビリティ向上が課題になっている．そこで我々は，課題解決のための第一歩として VNode を使用した実験用ネットワーク上で IPEC (IP-Ether-Chimera) という非 IP プロトコルを開発・実験してその手順や経験を記述し，そこからユーザビリティに関する問題やノウハウを抽出した．課題として開発者によるケアレスミスの防止策の必要性，ノウハウとして Linux PC による小規模な実験を VNode を使用した広域網の実験にスケールアップすることによって開発を容易化できることがあげられる．今回の実験には広帯域は必要でなかったが，今後 10 Gbps の帯域を活用した実験へのスケールアップも比較的容易に実現できるとかんがえられる．

キーワード ネットワーク仮想化，非 IP プロトコル，仮想化ノード，アドレス学習．

Development Method and Experience of A Non-IP Proocol Using the Virtualization Nodes

Yasusi Kanada
Central Research Laboratory,
Hitachi, Ltd.
Higashi-Koigakubo 1-280,
Kokubunji, Tokyo 185-8601
E-mail: Yasusi.Kanada.yq@hitachi.com

*Akihiro Nakao**†*
*The University of Tokyo Interfaculty Initiative in Information Studies, Graduate School of Interdisciplinary Information Studies
Bunkyo-ku Hongo 7-3-1, Tokyo 113-0033
E-mail: nakao@iii.u-tokyo.ac.jp

†National Institute of Information and Communications Technology
Bunkyo-ku Hakusan 1-33-16,
Tokyo 113-0001

Abstract In the National Institute of Information and Communications Technology (NICT), 10-Gbps-class virtualization nodes (VNodes) that enables implementing non-IP protocols with any frame format are developed using network-virtualization technology. In this project, because the VNodes have been introduced into the R & D test-bed network called JGN2plus, an important challenge is to improve usability for developers (JGN2plus users). Therefore, we developed and tested a non-IP protocol called IPEC (IP-Ether-Chimera) on the experimental network using the VNodes, described the procedure and experiments, and extracted the problems and knowhow concerning usability. A problem to solve is to develop methods for avoiding careless mistakes by developers, and an obtained knowledge is that a combination of a small-scale experiment using connected several Linux PCs and a scaled-up experiment on wide-area network reduced the complexity of the development. This experiment did not need wide bandwidth, but this method will enable scaling up the experiment utilizing 10-Gbps bandwidth relatively easier.

Keywords Network virtualization, Non-IP protocol, Virtualization node, Address learning.

1. はじめに

インターネットはもともと単純なネットワークをめざしてきたが，さまざまな用途につかわれるようになるのにしたがって複雑化してきた．インターネットにおいてはすべてのプロトコルがインターネット・プロトコル (IP) をベースとしているため，さまざまなプロトコル機能が干渉しあい，それらを共存させるために調整が必要になって複雑化するとともに，新機能を追加するのが困難になってきている．一方でクラウド・コンピューティングの普及などによってインターネットへの要求がさらに多様化・高度化するなかで，IPv4 や IPv6 などをベースとするプロトコルでは対応が困難になっている．

このような状況のなかで新世代ネットワーク・プロジェクトにおいては IP にとらわれない新プロトコルを開発し，そのうえで IP 上では困難だったさまざまなアプリケーションの実現をはかろうとしている．そのベースとなっているのが仮想化ノード・プロジェクトにおいて開発されているネットワーク仮想化技術と仮想化ノードである．このプロジェクトは情報通信研究機構 (NICT) を中心とし，東大，NTT，NEC，富士通，日立が協力してすすめている．このプロジェクトにおいては，独立かつ自由に設計された機能を実装した複数の仮想ネットワークがひとつの物理ネットワーク上で同時に動作できる環境を実現することをめざしている [Nak 10b]．実装されるべきネットワーク機能としては IP にかわるプロトコルやアプリケーション依存の機能などがある．ここで開発された仮想化ノードは 2010 年度に研究開発用テストベッド・ネットワーク JGN2plus に導入されたが，新プロトコルなどの研究開発にひろく使用できるようになる予定である．

この報告においては実験用の非 IP プロトコル IPEC (IP Ether

Chimera) の開発からえられた経験を報告する．このプロトコルの特徴や実験結果などについては別途，報告している [Kan 10].

このプロトコルの開発目標はつぎのとおりである．第 1 に，仮想化ノードの開発に関連した目標は，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である．この目標はさらにつぎの 3 つの副目標にわけられる．

- 仮想化ノードによって構成されたネットワークの動作検証とユーザビリティの検証をおこなう．
- 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る (開発者にノウハウを提供する).
- 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デモする．

第 2 に，プロトコルの研究に関する目標は，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に踏みだすことである．すなわち，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し，Ethernet スイッチの学習アルゴリズムを拡張して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ことをめざしている．

この報告はこれらの目標のうち第 1 の，プロトコル開発のテストケースという面，そのなかでもとくにユーザビリティとノウハウ獲得に焦点をあてる．まず 2 章において NICT の仮想化ノードと仮想化ネットワークについてかんたんに説明し，3 章において実験用プロトコル IPEC についてかんたんに説明する．4 章において，仮想化ノードを使用した新プロトコル開発の一般的な手順を意識しながら，今回の IPEC 開発のながれを説明し，5 章においてそこでえられた経験的知識を記述し，6 章において結論をのべる．

2. 仮想化ネットワークと仮想化ノード

この章においては仮想化ネットワークとそれを構成する仮想化ノードについて説明する．なお，これらについては，もうひとつの報告 [Kan 10] においてよりくわしく記述している．

2.1 仮想化ネットワーク

仮想化技術はまずコンピュータ本体における記憶の仮想化や CPU などの分割使用 (タイムシェアリング) の技術として発展してきたが，最近は仮想マシン (VM) というかたちでコンピュータじたいが仮想化されている．ネットワークに関しては VPN による WAN の仮想化がすすめられてきた．最近ではデータセンタ内のネットワークも VM と連携しつつ仮想化されてきている．

新世代ネットワークの研究においては，こうしたネットワーク仮想化技術を発展させて，あたらしいプロトコルを他のユーザに影響をあたえることなしに開発できる環境 (ユーザの組織ごとにきちんとアイソレートされた環境) をつくることがもとめられている．仮想化ノード・プロジェクトは，このような状況のもとでまず研究者が自由にプロトコルを設計できる環境をつくることを目標としているが，さらにはそれを商業的展開も可能なようにすることもめざしている．

ネットワーク仮想化においては，仮想化前のネットワークと仮想化後のネットワークとが共存する．仮想化前の下層のネットワークを**仮想化ネットワーク** (virtualized network) とよび，仮想化後の上層のネットワークを**仮想ネットワーク** (virtual network) とよぶ．

2.2 仮想化ノード・プロジェクトにおける仮想化ネットワーク

仮想化ネットワークに関してはすでに多様な研究がおこなわれ，多様なモデルが提案されている．そのなかで PlanetLab [Pet 02] [Tur 07] がもっとも有名だが，GENI [GEN 09] は現在進行中の注目するべきプロジェクトである．仮想化ノード・プロジェクトのモデル [Nak 10b] は管理に重点をおいている．

このモデルにおいては，物理的なネットワーク (**図 2.1**) は 1 個または複数個のドメインによって構成される．各ドメインはドメイン・コントローラ (DC) によって管理される．ドメインのなかには仮想化機能をもつつぎの 2 種類のノードが存在する．

- **仮想化ノード** (VNode): 仮想ネットワークにおける中継機能をもつノード．
- **アクセス・ゲートウェイ** (AGW): 仮想ネットワークとユーザ端末やユーザのネットワークとのあいだの中継機能をもつノード．

VNode どうしは GRE (Generic Routing Encapsulation) のようなプロトコルによるトンネルによってむすばれるため，途中のルータやスイッチに依存せず自由なトポロジーをもつネットワークが構成できる．

各 VNode はつぎの 3 種類の構成要素によって構成されている．

- **プログラマ** (Programmer): パケットに対する処理をおこなう構成要素である．ハードウェアとソフトウェアとで構成される．パケットに対する処理を仮想ネットワークの開発者がプログラムできるため，プログラマとよばれる．1 個の仮想化ノードのなかに複数個存在することができる．
- **リダイレクタ** (Redirector): 通信データ (パケット) を他の仮想化ノードや他の種類のノードから受信したり，それらに通信データを送信したりする転送機能をもつ構成要素である．
- **VNode マネジャ** (VNode Manager): 仮想化ノード全体の管理をおこなう構成要素である．1 個の仮想化ノードのなかに 1 個だけ存在する．通常，ソフトウェアだけで構成される．ネットワーク全体を管理する管理サーバであるドメイン・コントローラ (DC) からの指示にもとづいて動作する．

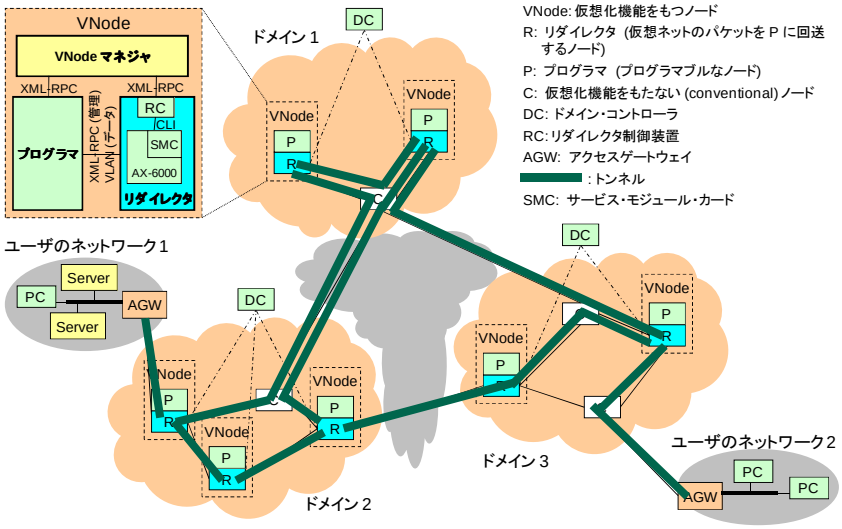


図 2.1 仮想化ノード・プロジェクトにおけるネットワークの物理構成

また，このモデルにおいては，PlanetLab にならって，仮想化ネットワーク上につくられる仮想ネットワーク (または仮想ネットワークの構成要素の集合) を**スライス** (slice) とよぶ．スライスは複数の仮想ノードとそれらをつなぐ仮想リンクとで構成される (**図 2.2** 参照) が，このモデルにおいてはこ

れらをつぎのようによぶ [Nak 10a].

- **ノードスリバー** (Node Sliver): 1 個の仮想化ノードのなか (プログラマのなか) に存在する計算資源である．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．大別すると，Linux (Ubuntu 9.1) を搭載した VM である**スローパス** (slow path) と，ネットワーク・プロセッサによる**ファストパス** (fast path) の 2 種類がある．
- **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を意味する．通常はことなる物理ノード内にあるノードスリバーを結合する．リンクスリバーは複数の VNode や AGW にまたがって存在し，リダイレクタが設定し資源管理するトンネルによって実現される．

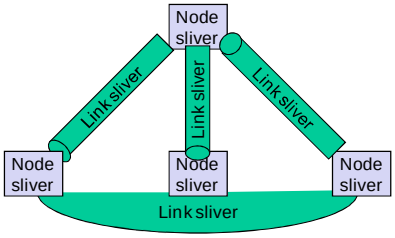


図 2.2 仮想ネットワークの論理構成 (スライスの構造)

3. 実験用プロトコル IPEC とその実験環境

実験用プロトコル IPEC についてはもうひとつの報告 [Kan 10] においてよりくわしく記述しているが，ここでは開発経験の記述に必要なだけの内容を記述する．

3.1 アドレスとフレームの形式

IPEC においてあつかうアドレスとフレーム (パケット) の形式は**図 3.1** のとおりとする．すなわち，まずアドレス (ながさ 8 バイト) に関してはつぎのとおりである．

- **ホスト ID**: アドレスの下位 (仕様上は可変長だが実装上は 4 バイトに固定している) はホスト ID をふくむ．
- **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 をふくむ．

グループ ID はロケータと解釈することもできる．

また，フレーム・ヘッダは 22 バイトあり，上位から順につぎのフィールドから構成されている．

- **受信者アドレス**: フレームを受信するべきホストのアドレスを指定する．上記のアドレス形式をしている．
- **送信者アドレス**: フレームを送信したホストのアドレスを指定する．上記のアドレス形式をしている．

他のフィールドについてはここでは説明を省略する．

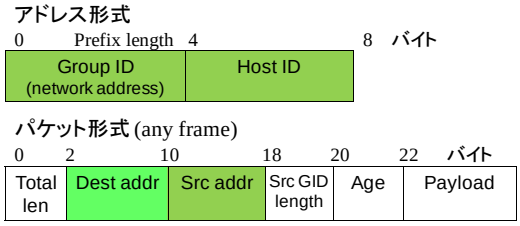


図 3.1 プロトコル・フォーマット

3.2 スライス構成

NICT 白山リサーチラボにおいて，IPEC に関する実験をおこなうために，3 個の VNode，3 個の AGW のうえに生成した**図 3.2** のような構成のスライス (仮想ネットワーク) を使用した．この図にはこのスライスに接続した端末 (Linux PC) も記述し，また VNode や AGW がもつテーブルの内容も記述している．これらのテーブルの意味についてはもうひとつの報告 [Kan 10] において説明している．

スライスの構造は物理ネットワークによって制約される一定の範囲のなかで，設計者が自由に選択することができる．すなわち，設計者は物理ノード数をこえない範囲で自由にスライスを構成するノードスリバー (仮想ノード) の個数をきめ，それらのあいだのリンクスリバー (仮想リンク) を自由に設定することができる．

現在の仮想化ネットワークにおいては，スライス構成は設計者が XML を使用して記述するようになっている．スライス定義の例を**図 3.3** にしめす．これは図 3.2 の構成をもつ IPEC_Slice_000 という名称のスライスの定義の一部だけをしめすものである．

スライス定義はおおきくわけてつぎの 4 つの部分からなっている: 1) リンクスリバー定義，2) ノードスリバー定義，3) ノードスリバーとリンクスリバーとの結合の定義，4) ノードスリバーの物理ノード (VNode, AGW) への配置 (マッピング) の定義．スライス定義には物理装置名と論理装置名の両方が記述され，それらの対応も記述される．図 3.2 にもその両方を記述している．たとえば，論理名 AGW1 は物理名 agw-f5 に対応している．また，ノードスリバー NS00 は物理ノード VNode 2 内にある．これらの対応は配置定義によって指定される．

現在はスライスを生成するために開発者が XML による定義を記述する必要がある．しかし，仮想化ノード・プロジェクトにおいては，ちかい将来 GUI などを使用してより容易にスライスを定義できるようにすることが検討されている．

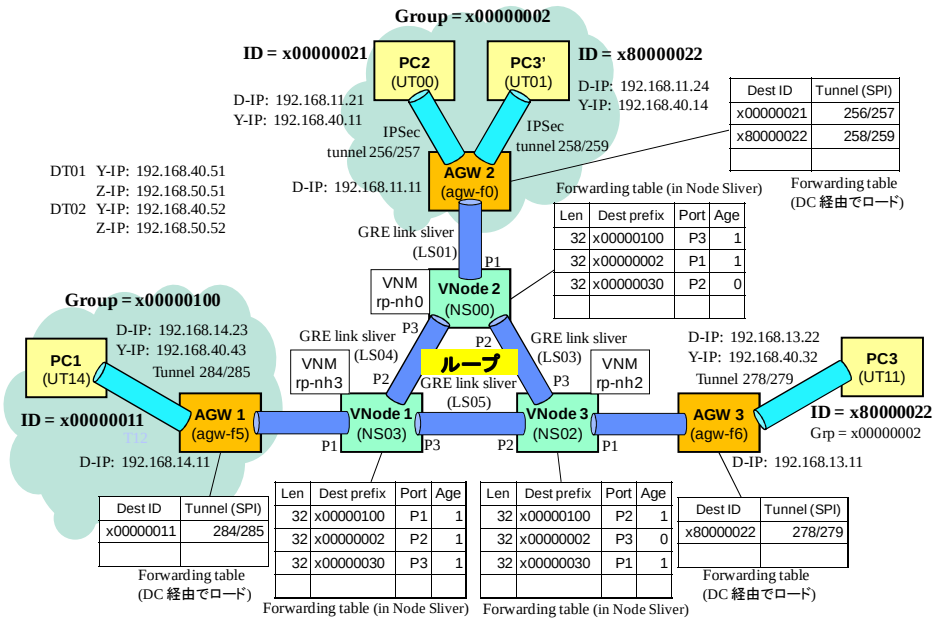


図 3.2 スライス構成 (学習後の状態)

4. 開発における関係者の役割と開発法

この章においては，はじめに仮想ネットワークの開発における関係者の 3 つの役割 [NTT 11] についてのべ，つづいて新プロトコル開発のながれにそって操作法

```
<?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" subtype="GRE" type="link">
            <vports><vport name="e1"/><vport name="e2"/></vports>
        </linkSliver>
        …
      </linkSlivers>
      <nodeSlivers><!-- 以下はノードスリバーの定義 -->
        …
        <nodeSliver name="Node2" type="prog">
          <vports><vport name="vp1"/><vport name="vp2"/>
              <vport name="vp3"/></vports>
          <hierarchy>
            <sliverdef>
              <nodeSlivers>
                <nodeSliver name="SP00">
                  <vports><vport name="vip1"/><vport name="vip2"/>
                      <vport name="vip3"/></vports>
                  <instance subtype="KVM" type="SlowPath_VM">
                    <resources><!-- ノードスリバーの計算資源 -->
                      <resource keyword="cpu" value="1"/><!-- CPUの個数 -->
                      <resource keyword="arch" value="x86_64"/>
                      <resource keyword="memory" value="2048"/>
                    </resources>
                    <params><!-- 以下はあらかじめ用意されたVMイメージ -->
                      <param keyword="bootImage"
                        value="http://…/KVM_Ubuntu910Server32.img"/>
                    </params>
                  </instance>
                </nodeSliver>
              </nodeSlivers>
              <linkSlivers>…</linkSlivers>
            </sliverdef>
            <structure>… </structure>
            <params>…</params>
          </hierarchy>
        </nodeSliver>
        …
        <nodeSliver name="AGW2" type="agw">
          <vports><vport name="vp1"/></vports>
        </nodeSliver>
      </nodeSlivers>
    </sliverdef>
    <structure><!-- 以下は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"><!-- w11 は AGW2 と LS01 を結合する -->
        <vport portname="vp1" slivername="AGW2"/>
        <vport portname="e1" slivername="LS01"/>
      </bind>
      …
    </structure>
  </slicespec>
  <!-- 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>
```

図 3.3 スライス定義の例 (一部省略)

[NTT 11] や開発法を説明する.

4.1 役割

仮想化ノード・プロジェクトにおいては, 仮想ネットワークの生成と運用にかかわる役割は, 運用者, 開発者, 一般ユーザという 3 者にわけられている.

- **運用者** (オペレータ): 実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): 仮想ネットワークを生成し管理する. すなわち, ノードスリバーのプログラムを開発し, スライスを設定し, 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. 現在は端末数と同数の一般ユーザ ID が使用される.

ただし, 今回の新プロトコル開発においてもそうだったが, 実験の場合はひとりの研究者がすべての役割をはたすことが多いとかんがえられる. 上記 3 者のいずれもが, **ポータル**とよばれる Web インターフェイスを介してこれらの登録・設定をおこなう[1].

4.2 スライスごとの操作

開発者はポータルにログインして, まず図 3.3 のようなスライス定義をファイルから入力してスライス予約 (Slice Reservation) する. そしてつぎのような一連のスライス操作 (Slice Operation) をおこなう.

- **結合** (Bind): ノードスリバーとリンクスリバーを結合する. これらはスライス予約時に生成されるが, 初期状態ではまだ結合されていないので, この操作によって結合する.
- **実行** (Run): 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をふくむ受信ホスト ID によってふりわける. そのため, 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設定する (Egress 抽出設定). IPEC においては受信者アドレスがパケット先頭から 2 〜 10 バイトめにあり, その後半 4 バイトが受信者 ID であるから (図 3.1), オフセットが 6, ながさは 4 である. この設定は開発プロトコルの形式にかかわる中核の設定である.
- **サービス開始** (Service Start): 上記の各設定が終了するとサービスを開始することができる. ただし, 後述する一般ユーザの設定がなされていなければ実際にはサービスされない.

4.3 一般ユーザごとの操作

一般ユーザごとの操作は開発者と一般ユーザとが連携しておこなう. まず, 開発者がポータルにてつぎの一連の操作をおこなう.

- **一般ユーザの登録**: まだ登録されていないときは, まずユーザ登録 (User Registration) する. 管理上必須の情報はユーザ ID とユーザが属する物理 AGW 名である. 現在のシステムにおいては各ユーザ (端末) はいずれか 1 個の AGW だけを使用する.
- **アクセス設定** (Access Configuration): スライス操作画面でボタンをおしてアクセス設定画面を表示し, スライスにアクセスできる一般ユーザを指定する. すなわち, 登録ずみ一般ユーザのリストのなかからスライスにアクセスしてよいユーザを選択する.
- **受信ホスト ID の登録**: Egress 中継登録画面において, AGW における端末へのパケットふりわけの際に使用する受信ホスト ID とともに, 物理 AGW 名, ユーザ ID を設定する. パケットから抽出されたデータにマスクをかける機能があるのでマスク値も指定するが, 通常は –1 (IPEC においては 4 バイトなので 0xFFFFFFFF) を指定する. 開発プロトコルの動作をきめる中核の設定である. IPEC では本来この情報を学習によって獲得したいが, 現在は固定的に登録する必要がある.
- **IPsec 情報の設定**: 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ので, そこで使用するモード (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 IPsec SA 設定において指定する. 端末の識別にはユーザ ID ではなく, ここで指定する SPI が使用される. そのため, SPI 以外の値は各ユーザが共通の値を使用できるが, SPI だけは一意な値を指定

[1] なお, 著者の使用経験からつぎのようなことがわかった: ひとりの開発者は複数のセッションをひらくことはできないので, ひとつのプロジェクトのために複数の開発者 ID を用意したほうがよい場合がある.

する．この値は端末で指定する値と一致する必要がある．

開発者による設定後，まずユーザごと (使用する端末ごと) にポータルにログインし，スライス設定 (Slice Configuration) により端末の IP アドレスと参加するべきスライスとを登録する．この登録の結果えられる SPI などの IPsec 関連の情報を端末 (Linux PC を仮定) において図 4.1(a) のように設定する [VNP 10]．ここでは IPsec の外側の IP アドレスとして AGW には 192.168.13.11，端末には 192.168.13.22 を使用し，ingress SPI, egress SPI としてそれぞれ 278 と 279 を使用することを仮定している．また，AGW-端末間には GRE トンネルが使用されるため，図 4.1(b) の設定が必要である[1]．

```
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
```

(a) IPsec の設定

```
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     -- gretap0 を link up させる必要がある．
[ ifconfig gretap0 mtu 1381 ]
```

(b) GRE の設定

図 4.1 端末における GRE と IPsec の設定

4.4 ノードスリバー用プログラムの開発とローディング

現在の IPEC 実装にはスローパス・ノードスリバーを使用している．ファストパス・ノードスリバーを開発する際には他におなじ環境がないので，最初から仮想化ノードを使用して開発する必要があるだろう．しかし，スローパス・ノードスリバーに関しては通常の Linux PC を数台使用すれば仮想ネットワークにちかい環境で開発しテストすることができるので，まずそうするのがよいだろう．IPEC に関する Linux マシン上での予備開発については 5 章においてのべる．

現在，スローパス・ノードスリバーは Ubuntu 上の KVM において動作する．非 IP による通信には Linux の promiscuous mode を使用する．通常の環境では promiscuous mode において Ethernet パケットをあつかうが，仮想化ノードにおいては Ethernet ヘッダは必要ない．すなわち，非 IP による通信においては，先頭から自由にきめたフォーマット，IPEC においては図 3.1 のフォーマットをあつかう．

プログラム上で Ethernet type を記述するときは，つねに ETH_P_ALL という値を指定すればよい (IP のときは ETH_P_IP という値が使用される)．この値はパケット上の Ethernet type の値が ETH_P_ALL であることをあらわすのではなく，パケットが Ethernet type フィールドをもたないことをあらわす．

これらのノードのあいだでは GRE トンネルが使用されているが，それによって任意のフォーマットを使用した通信が IP ネットワークを介した VNode 間および VNode-AGW 間でできるようになっている．

スローパス・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．図 3.2 の構成においてノードスリバーは 3 つのポートをもつので，eth1, eth2, eth3 を使用する．

開発者用端末などでプログラムをクロス・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 に転送して起動する．転送先 VM の IP アドレスとポートはポータルのスライス情報 (Slice Info) 画面において Login Info のメニュー項目にスライス名を入力すればわかる．この画面にはノードスリバーごとにその VM に関する情報が表示される．えられた情報を使用して slogin コマンドや ssh コマンドなどによってログインし，クロスコンパイルしたバイナリが存在する PC から scopy コマンドや sftp コマンドなどによってバイナリ・ファイルを VM に転送する．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，インターフェイスごとにつぎのようなコマンドを入力してからプログラムを起動する．

ifconfig eth*i* up

4.5 端末用プログラムの開発

端末間の通信を可能にするためには，ノードスリバーだけでなく，端末用のプログラムも開発する必要がある．端末用プログラムにおいてもノードスリバーと同様に promiscuous mode を使用し，Ethernet type として ETH_P_ALL を指定して通信する．ただし，ノードスリバーにおいてはインターフェイスとして eth*i* を使用したのに対して，端末においては gretap0 を使用する (4.3 節参照)．端末のデータ用インターフェイスは通常 1 個だけなので，gretap0 だけを使用する．

5. 開発上の話題と経験

この章においては，IPEC の開発経験のなかで前章までに記述していない特筆するべき事項を記述する．

5.1 Linux マシン上での仮想化ノード用プログラム開発法

4.4 節においてのべたように，IPEC のプログラム開発はまず Linux PC 上でおこなった．スライス上でのテストにおいてはプログラムの修正などにやや時間がかかるので，このように通常の Linux PC 上で予備開発しテストしておくのが有効である．

PC を 6 台ほど用意すればより実環境にちかいテストができるが，実際には 2 台だけでテストした．図 5.1 のように各 PC がもとからもつ物理インターフェイス (NIC) にくわえて 2 枚ずつ NIC を追加してクロスケーブルで接続し，1 台にはノードスリバーのプログラム vnode.c，もう 1 台には端末のプログラム term.c をのせてテストした．

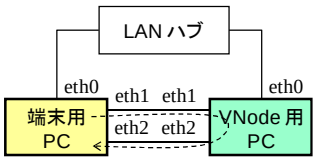
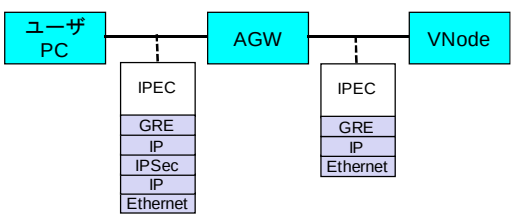


図 5.1 Linux マシン上での仮想化ノード用プログラム開発

3 枚の NIC のうち 1 枚 (図では eth0) は基本的に制御・管理用である．これらの PC 上または他の PC からこれらの PC にログインしたりファイル転送したりするために使用した．のこりの 2 枚はデータ転送用であり，promiscuous mode で使用する．Promiscuous mode においては IP 通信とはちがって NIC を直接指定するので，端末用 PC 上では eth1 からパケットを送信するプログラムと eth2 からパケットを受信するプログラムとを動作させ，VNode 用 PC 上では eth1-eth2 間で双方向にパケットを転送するプログラムを動作させれば，最低限の通信テストができる (図の矢印)．ただし，端末用のプログラムにおいては，実際の端末で使用する 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必要がある．また，1 台の PC 上

[1] 仮想化ノード・プロジェクトにおいては，このように AGW-端末間においても，また VNode 間や VNode-AGW 間においても GRE トンネルが使用されている．このようにやや複雑なパケット形式が使用されている理由は，これらすべてのリンクにおいて任意の非 IP フォーマットによる通信ができるようにすることである．それぞれのリンクにおけるパケット形式はつぎのとおりである．



で VNode 用のプログラムを複数個動作させたり VNode 用のプログラムと端末用のプログラムを同時に動作させるには，インターフェイス (eth1, eth2, …) の番号を自由選択できるようにする必要がある．

さらに NIC を増設すれば複数の VNode や 3 台以上の端末もシミュレートできる．ただし，複数のプログラムが動作できるためには CPU 時間を独占しないようにプログラムする必要がある．それでもなお，PC が 2 台だけではタイミングに依存するテストはやりにくい．

なお，このテストにおいては Ethernet の NIC をそのまま使用しているため，任意のプロトコルがそのまま使用できるとはかぎらない．Ethernet パケットの最初の 12 バイトには 2 個の MAC アドレスがふくまれ，つづく部分は type フィールドであるが，IPEC においてはそこに他のフィールドがふくまれ，アドレス長もことなっている．カードやドライバによっては想定外の動作をする可能性があるとかんがえられるが，IPEC のテストにおいてはこれまでのところとくに問題はおこっていない．いずれにしても，Ethernet にこのような形式のパケットをながすことは局所的なネットワークでなければできないし，このようなパケットを使用すると Ethernet スイッチを使用することもできない．

当初はこのような使用によって問題が発生することもかんがえて，VNode，端末共通のヘッダ・ファイル IPEC.h 上で IPECoverEthernet というスイッチをオンにすれば Ethernet ヘッダをつけるようにした．仮想化ノードを使用したテストにおいても最初は Ethernet ヘッダつきの形式を使用したが，すくなくとも今回はその必要はなかった．

5.2 失敗経験

まだ環境やマニュアルが整備されていないので，IP 以外による通信プログラムを記述した経験がほとんどない著者が IPEC の実験とデモのための環境を構築するときに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である．

- **端末設定において発生した問題**: VNode-端末 間の通信プロトコルには，認証・セキュリティと自由な非 IP 通信を可能にするため，GRE over IPSec という，やや複雑なプロトコルが使用される．そのため，図 4.1 に記述したように端末の設定がやや複雑になり，まちがえやすい．しかも，並行して進行中だった開発ではスライス上で IP を使用していたため，図 4.1 の設定にさらに IP 通信のための設定が追加されて複雑化していた．そのなかで非 IP プロトコル通信にどの部分が必要なのかが明確になっていなかったため，パラメタ誤記などで通信できないときの原因究明に時間がかかった．しかし，非 IP 通信に必要な設定が図 4.1 に記述したものだけであることが事前にわかっていれば，さらには適切なツールが使用できれば，もっと迅速にできたはずである．
- **スライス操作において発生した問題**: スライスの生成においては人間 (開発者) が介入するべき回数が多い．そこで誤操作などをするとエラーが発生することがある．著者も誤操作によってとまどうことになった．しかし，エラー発生時の対策がマニュアルに記述されれば，とまどう機会は減少するだろう．
- **インターフェイスのリンクアップ**: 4.4 節に記述したように，ノードスリバーのプログラムを動作させるまえにインターフェイスをリンクアップする必要があった．著者はそれをしなかったため，パケットがながれない理由がしばらくわからなかった．つまり，リンクアップしなくてもエラーが発生しないので，わかりにくい．リンクアップが必要なのは端末側も同様であり，図 4.1(b) にはその設定がふくまれているので，それを使用すれば問題ない．しかし著者はここでもその設定をいれていなかったためにつまずくことになった．
- **32 ビットと 64 ビット混在による問題**: 最近は Intel の CPU を搭載した PC でも 32 ビットのものと 64 ビットのもの (x86-64) が混在するようになっている．著者も，端末プログラムを 32 ビット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起こしてしまった．OS が 64 ビット版であることをあらかじめ知っていればすぐに気づいただろうが，知らなかったためにバグの原因がわかるまでに時間がかかった．

6. 結論

仮想化ノードを使用した IPEC の開発と実験をおこなってその手順や経験を記述し，そこからわかったユーザビリティに関する課題やノウハウ等を記述した．課題としては，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がある．また，えられたノウハウとして，Linux PC どうしをつなぐ実験からはじめて仮想化ノードを使用した広域実験にスケールアップすることによって，仮想化ノードのための開発が容易になることがある．局所的な環境で 1 Gbps 以下の通信を実験するだけならば Linux PC どうしをつなぐだけでもできるが，10 Gbps のリンクを使用した実験や広域での実験，ファストパスを使用する実験などは困難である．仮想化ノードを使用すれば，Linux PC 上で開始した実験を容易にそういった実験にスケールアップできるとかんがえられる．

IPEC の開発のなかで，いくつか軽度の失敗をかさねたが，JGN2plus に導入された仮想化ノードを一般ユーザが使用する際には，これらの点でユーザが失敗しないようにツールを開発し，スライス開発法をマニュアル化するべきである．プロジェクトは実際その方向にむかっている．とくに著者はこの開発からえられたノウハウ等を今後の新プロトコル開発にやくだつようにまとめていきたい．

謝辞

IPEC の開発と実験にあたり，NICT，東大，NTT，NEC，富士通研，日立による仮想化ノード・プロジェクトの参加者各位，とくにNEC の 元木 顕弘 氏，富士通の 渡辺 紀一 氏，NTT の 高橋 紀之 氏，アラクサラの 木谷 誠 氏に意見や助力をいただいたので感謝する．

参考文献

[GEN 09] The GENI Project, “Lifecycle of a GENI Experiment”, GENI-SE-SY-TS-UC-LC-01.2, April 2009, http://groups.geni.net/geni/attachment/wiki/ExperimentLifecycleDocument/ExperimentLifeCycle-v01.2.pdf?format=raw .

[Kan 10] 金田 泰, “仮想化ノードを使用した実験用非 IP プロトコルの開発”, 電子情報通信学会 IN 研究会, 2010-9.

[Nak 10a] Nakao, A., “Network Virtualization as Foundation for Enabling New Network Architectures and Applications”, *IEICE Trans. Commun.*, Vol. E93-B, No. 3, pp. 454–457, March 2010.

[Nak 10b] 中尾 彰宏, “ネットインフラを用途別に “スライス” 柔軟な機能拡張の実現に効果”, 日経コミュニケーション, June 2010.

[NTT 11] 山本 猛仁, 築島 幸男, 高橋 紀之, 中尾 彰浩, “プログラマブルな仮想化ネットワークにおける NW 管理モデルの提案”, 電子情報通信学会, 新世代ネットワーク研究会, 2011-1 (予定).

[Pet 02] Peterson, L., Anderson, T., Culler, D., and Roscoe, T., “A Blueprint for Introducing Disruptive Technology into the Internet”, *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 1, pp. 59–64, January 2003.

[Tur 07] Turner, J., Crowley, P., Dehart, J., Freestone, A., Heller, B., Kuhms, F., Kumar, S., Lockwood, J., Lu, J.,Wilson, M., Wiseman, C., and Zar, D., “Supercharging PlanetLab - High Performance, Multi-Application, Overlay Network Platform”, *ACM SIGCOMM Computer Communication Review*, Vol. 37, No. 4, pp. 85–96, October 2007.

[VNP 10] 仮想化ノードプロジェクト, “デベロッパーズマニュアル”, 未公開.