Infra-estrutura virtual nível 2 para interligação de máquinas virtuais em larga escala

Rui Ferraz
Centro ALGORITMI
Universidade do Minho
Email: a42846@alunos.uminho.pt

Maria João Nicolau
Centro ALGORITMI
Universidade do Minho
Email: joao@dsi.uminho.pt

António Costa
Centro ALGORITMI
Universidade do Minho
Email: costa@di.uminho.pt

***Abstract*— O conceito de virtualização de redes encontra-se em expansão. A capacidade de oferecer a uma infra-estrutura de rede a possibilidade de suportar várias redes virtuais distintas, sem as pressões impostas pelas redes de suporte, é um ponto de interesse tanto para operadoras como para académicos. Esta característica permite criar um "mundo" virtual, auto-contido e com parâmetros controlados e alteráveis onde podem ser realizadas experiências com novas tecnologias ou mesmo, olhando sob um ponto de vista empresarial, oferecer serviços específicos a clientes, onde os parâmetros da rede são contratualizados com um revendedor.**

Este artigo descreve uma proposta de implementação de rede virtual, onde se aborda a descoberta de recursos e as diferentes metodologias para a troca de informação entre os nós virtuais. As ligações criadas permitem virtualizar uma arquitectura nível 2 considerando cenários de conexão entre dois ou mais nós. São utilizadas várias abordagens de modo a criar a rede virtual, nomeadamente conexões baseadas em *Sockets* e *RawSockets*, assegurando uma comunicação controlada sobre uma infra-estrutura de rede operacional. Os parâmetros que estas ligações permitem oferecer foram avaliados numa situação real comprovando a possibilidade de disponibilização desta tecnologia sobre a infra-estrutura pública.

I. Introdução

A infra-estrutura de rede pública é considerada um suporte capaz de absorver as tecnologias que nela se colocam, contudo este pressuposto torna-se abstracto e difícil de compreender quando são testados novos protocolos e abordagens de comunicação. A inércia produzida pelos protocolos sedimentados leva a que muitas vezes seja difícil a incorporação ou o teste de novas características.

A necessidade de testar protocolos em arquitecturas distintas é, também, uma dificuldade real uma vez que a compra de novos equipamentos torna o processo demorado e dispendioso. É neste contexto que surge a virtualização de redes. A capacidade de disponibilizar uma rede parametrizada, com características e arquitectura esperadas permite transformar a variável rede num parâmetro controlado. Os nós intermédios de uma rede virtual, ao contrário dos equipamentos da rede são facilmente alteráveis podendo ser configurados de acordo com as necessidades do projecto.

A rede virtual permite uma utilização que ultrapassa o conceito académico e de investigação. Também num mercado empresarial esta pode apresentar mais valias, permitindo que um cliente usufrua de uma rede de características específicas e a possa alterar de acordo com o serviço que se pretende que ofereça.

O trabalho desenvolvido tenta dividir o desafio da criação de uma rede virtual em vários passos lógicos encadeados. A procura de recursos, a definição das conexões, a criação dos *links* virtuais, manutenção e portabilidade, são alguns dos passos que permitem o funcionamento do sistema como um todo, dando a liberdade ao utilizador de criar o cenário de teste pretendido.

Este encadeamento de passos permitirá solucionar alguns dos desafios propostos pela tecnologia: como descobrir nós dispostos a pertencer à rede virtual, fazer a sua configuração, interliga-los e implementar os mecanismos que permitem a multiplexação/desmultiplexação.

O artigo está dividido de acordo com os desafios propostos pela tecnologia: Serão introduzidas as soluções possíveis para os problemas relevantes, justificando a abordagem utilizada para a criação da rede virtual; depois são apresentados os resultados obtidos nos testes desenvolvidos, e finalmente a conclusão do trabalho e os aspectos a melhorar no futuro.

II. Redes virtuais

A virtualização de rede recorre a várias tecnologias que permitem a construção uma camada lógica que possibilita a troca de informação entre nós virtuais, sobre uma rede que disponibiliza os recursos físicos. Alguns projectos abordam este assunto, com implementações distintas mas com linhas de desenvolvimento similares: *X-bone*[17][16], *Vine*[1][18], o trabalho desenvolvido pela PT inovação e Universidade de Aveiro nesta área[12][7][13], o *Violin*[2][8] e ainda o projecto *4WARD*[2].

Estes projectos serão sumáriamente abordados, estudando as soluções que apresentam.

A. Procura de recursos

A procura dos recursos disponíveis para a criação da rede virtual é de especial importância, uma vez que marca o ponto inicial do algoritmo e permite descobrir os recursos disponíveis para a configuração e mapeamento de nós virtuais.

[1]Virtual Network Architecture for Grid Computing
[2]Virtual Internetworking on Overlay Infrastructure

Para esta fase da criação da rede virtual, os vários projectos apresentam abordagens similares tentando maximizar o alcance da procura com o mínimo de *overhead* e mensagens possível.

Na abordagem utilizada no projecto *X-bone*[16] cada recurso é controlado por um *RD*(*Resource Daemon*) que responde aos pedidos enviados pelos *OM*(*Overlay Manager*). Este, por sua vez, apresenta as respostas ao utilizador recorrendo a uma interface gráfica. O *OM* comunica com o *RD* recorrendo a uma técnica de procura com alcance variável recorrendo às capacidades do *multicast*. É enviado um pedido de recursos para um grupo *multicast*, onde os *RD* se encontram à escuta. Se o número de respostas não for suficiente para a criação da rede, é enviada uma nova mensagem com um *TTL* incrementado, de forma a abranger uma maior área. O algoritmo de procura é terminado no momento em que o número de respostas dos *RD* atinge o valor pedido para a criação da rede.

O projecto apresentado pela rede virtual da *PT* inovação, baseado no *framework 4WARD*, aborda o problema de uma óptica similar ao que acontece com o *X-bone*, apresentando um encadeamento de processos similar. O processo de procura de recursos está interligado, apresentando uma arquitectura em três módulos: o centro de controlo da rede virtual, que fará o contacto com o utilizador; o *manager* da rede virtual, que permitirá a agregação das informações da rede e os agentes da rede virtual que recolherão as informações. Os agentes ligam-se a um grupo *multicast* pré-definido e trocam mensagens de estado entre si. A agregação das informação de cada um dos agentes permite descobrir a topologia de rede.

Projectos como o *Vine* abordam o problema de uma outra forma, deixando a procura de recursos de lado e fazendo com que todos os nós sejam pré-configurados com uma nova interface de rede e rotas especificas, permitindo-lhes comunicar com um *designated router*, que tratará do reenvio das informações recebidas para o destino.

B. Mapeamento dos recursos virtuais

Quando a rede de suporte é conhecida torna-se necessário pensar de que forma os recursos virtuais devem ser distribuídos pelos nós físicos. Alguns projectos abordam este problema de forma consciente[13][2], enquanto outros o referenciam como um problema a resolver[18] mas não lhe dão destaque.

Os nós que servem de suporte não possuem as mesmas especificações, alguns possuem melhor *hardware* e podem fornecer mais recursos à rede, não sendo linear o mapeamento de nós virtuais sobre a rede de suporte. Esta dificuldade é até vista como um problema *NP-completo*[4].

O mapeamento dos nós virtuais descrito em [13], baseia-se nos conceitos de *stress* nos nós de suporte e nos *links*, levando em consideração o efeito do *CPU* e memória na carga em cada um dos nós visados. O mesmo acontece com a carga sobre os *links* virtuais, que [19] apenas considera o número de *links* virtuais criados sobre a conexão física.

C. Criação dos links virtuais

Os canais de comunicação virtual permitirão a comunicação entre os nós da rede virtual, utilizando as conexões físicas como meio para a troca da informação. Este facto propõe vários desafios de forma a manter a transparência e a separação com o restante tráfego.

O *X-bone*[16][17] aborda o problema utilizando um encapsulamento recursivo da informação para cada rede de nível superior, permitindo a criação de uma rede virtual sobre uma outra rede virtual; criando um mecanismo de dois níveis de encapsulamento para cada nível de *overlay*, resultando num total de 3 níveis de cabeçalhos *IP* para casa *overlay*. O cabeçalho interno representa o *link* do *overlay* e permite indicar os seus extremos. O segundo funcionará como a camada de ligação[3] do *overlay*, indicando os pontos do túnel sobre o qual o pacote está a ser enviado. O último cabeçalho indica os extremos do túnel na rede de suporte que poderá também ser um *overlay*, criando uma rede recursiva. Os endereços utilizados na rede virtual pertencem a um conjunto separado, não interagindo com os que são utilizados na rede física.

O projecto *Vine* aborda o problema de uma outra forma, utilizando múltiplos endereços em cada *NIC (Network interface Controler)* físico, configurando uma rota estática que permite o encaminhamento dos pacotes *Vine* para os *VR* (*Virtual Routers*) que realizarão o reencaminhamento. Esta abordagem permite ultrapassar a utilização de *NAT* e *proxies*, uma vez que o *VR* servirá como ponto intermédio na ligação. Como a comunicação é iniciada pelo nó inicialmente incontactável, abrindo um canal de comunicação, permitirá que o *VR*, até ao momento armazenador, entregue todos os dados que estavam endereçados ao nó.

Uma outra abordagem é apresentada em *4WARD*[2]. Nesta, um protocolo de configuração de *link* virtual, aloca os recursos necessários no caminho entre os dois nós físicos, conectando as interfaces virtuais. O protocolo é baseado numa implementação do protocolo *Next Steps in Signalling* (*NSIS*[10]). Foi utilizado o *NSIS Signalling Layer Protocol* (*QoS NSLP*) que executa sobre o *General Internet Signalling Transport Protocol* (*GIST*[15]). O *QoS NSLP* é um protocolo de reserva de recursos que permite o controlo de admissão ao meio e alocação de recursos, para um encaminhamento de pacotes *IP* baseado no *QOS*. Foi desenvolvida uma extensão a este protocolo no âmbito do projecto *4Ward* que permite que este transporte as informações de endereços necessários para a criação do *link* virtual entre os nós virtuais.

III. Descrição da solução proposta

O trabalho desenvolvido assenta numa rede virtual que permite a conexão de nós virtuais e a troca de informações, sobre a forma de tramas nível 2.

[3]Simulando o nível 2 do modelo *OSI*

De forma a que qualquer sistema aspire a ser escalável é necessário que tome como bases do sistema as tecnologias que vigoram na rede pública. Protocolos como o *IP* no nível 3 do modelo *OSI* e o *ethernet* no nível 2, controlam a forma como a troca de informação na infra-estrutura da *Internet* é processada.

Foi abordada uma arquitectura que permite implementar as ligações entre os nós da mesma forma que uma topologia nível 2, sugerindo duas metodologias de criação de conexões virtuais: *links* ponto a ponto que simulam ligações dedicadas e arquitecturas como *token ring* e multi-ponto, que permitem criar topologias em estrela, barramentos e árvore.

A arquitectura proposta está representada na figura 1. Nesta, os nós são capazes de criar uma nova infra-estrutura de comunicação, tendo por base conexões ponto a ponto e multi-ponto.

Esta arquitectura permitirá a troca de informações entre os

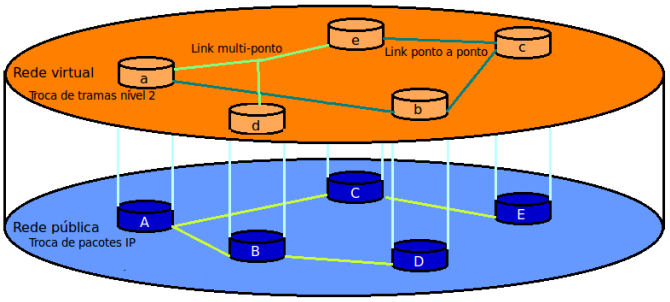

Fig. 1. Arquitectura lógica do projecto, onde os links virtuais criados, conectam dois nós, numa abordagem ponto a ponto, ou vários nós, ponto a multi-ponto

nós virtuais, onde as entidades criadas são isolados da rede física, operando transparentemente.

O problema foi dividido em 3 fases que se complementam: uma fase inicial de procura de *hosts*, a negociação de *links* e a sua criação e encapsulamento da informação.

Para além da criação de *links unicast*, recorrendo a *RawSockets*, foi testada a mesma abordagem recorrendo a *Sockets UDP*. A conexão multi-ponto pode ser implementada recorrendo a *MulticastSockets* e *RawSockets*. As implementações ponto a ponto têm em comum a procura de recursos que permite obter as características dos nós disponíveis.

O trabalho foi desenvolvido tendo em conta os benefícios que a portabilidade possui num sistema deste tipo. Foram utilizadas ferramentas genéricas, suportadas por vários sistemas operativos possibilitando a troca do suporte facilmente. A escolha do sistema de virtualização foi também pensada, grande parte das abordagens estudadas utilizam sistemas de para-virtualização que necessitam de suporte do processador nativo para algumas tarefas. A escolha para o projecto foi baseada numa virtualização total, recorrendo à ferramenta de virtualização *VirtualBox*[3], uma vez que mesmo possuindo uma performance mais baixa que a para-virtualização, permite a utilização do sistema em todos os nós disponíveis.

A. *Procura de recursos*

A rede virtual vai ser composta por vários *hosts* virtuais. Cada um possuirá um suporte físico na rede, que terá de conseguir comunicar e dar a conhecer a sua presença de forma a possibilitar a troca de informações. Para isto foi implementado um mecanismo de procura de recursos, baseada na estudada em [16], que permite procurar e obter as informações sobre os nós interessados e preparados para participar na criação da infra-estrutura virtual.

Cada nó de suporte terá uma aplicação que o permitirá ligar-se a um endereço *multicast* variável, definido pelo utilizador.

Uma mensagem de procura é enviada inicialmente com um *TTL* de 1, fixando o limite de procura à rede local, onde apenas os *hosts* ligados directamente ao endereço *multicast* receberão esta mensagem, como está representado na figura 2.

O nó que requisita a procura fica à escuta do meio à

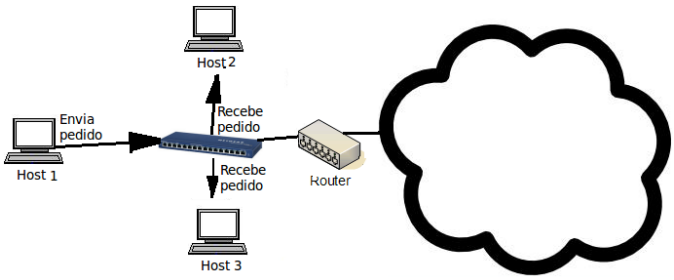

Fig. 2. Procura dos *hosts* com *TTL* inicial

espera das respostas. Nas respostas recebidas estarão os dados dos *hosts*: as informações das suas interfaces, a carga que o sistema neste momento comporta e as características das máquinas virtuais que tem ao dispor. A obtenção das informações das máquinas virtuais utilizará os recursos do *webservice* da *VirtualBox*[3].

A carga do sistema vai recair sobre vários parâmetros que podem ser estudados, desde a memória disponível, carga de *CPU* e número de máquinas virtuais em execução no sistema. Com estes valores é possível calcular localmente a carga em cada um dos *hosts*, usando a biblioteca *SIGAR*[1] para obter os valores. Este cálculo será a combinação dos recursos da máquina adaptando a nomenclatura apresentada em [13].

$$C_n = \frac{N_{vm}}{\delta + R_f \times F_{cpu} \times (N_{cpu} \times F_{cpu} - C_{cpu})}$$

- C_n - Carga de cada *host* físico;
- N_{vm} - Número de máquinas virtuais em funcionamento no *host*;
- δ - Variável usada para impedir que ocorra uma divisão por zero;
- R_f - *RAM* livre do sistema;

- F_{cpu} - Frequência do *CPU* em *MHz*;
- N_{cpu} - Número de *CPU* que o sistema possui;
- C_{cpu} - Carga que o *CPU* tem no momento, o valor pode variar entre 0 e $N_{cpu} \times F_{cpu}$.

O valor é compilado e colocado no Objecto que formará a mensagem de resposta. Em cada *host* é criada uma lista de interfaces que complementará a informação do nó. Neste Objecto ficarão guardadas, relativamente a cada interface, a nomenclatura que possui no sistema, o *IP*, o endereço *mac* que o identifica e o *gateway* que lhe permite enviar informação para fora da rede local. Serão também guardadas as informações sobre as máquinas virtuais disponíveis e as suas características.

Com o envio da primeira mensagem de procura, é iniciado um temporizador, que contará 2 segundos até ao envio de uma nova mensagem com um *TTL* de valor superior, até que se atinja o valor de *TTL* estipulado pelo utilizador ou o valor por defeito.

Cada nó ao receber uma mensagem de pedido de recursos, vai compilar a carga local, enviando a informação para o cliente que requisitou o pedido de criação de rede, que substituirá a informação caso esta exista.

As informações recebidas não serão actualizadas temporariamente, sendo dada a liberdade ao utilizador de pedir uma actualização dos dados quando entende que estes não estão de acordo com o cenário real. Esta medida pretende reduzir a quantidade de tráfego na rede, fornecendo uma maior liberdade ao utilizador.

B. Links virtuais

Num momento em que os recursos da rede são conhecidos é iniciada a criação das conexões virtuais, e a iniciação dos respectivos nós virtuais que servirão como extremos.

A criação dos *links* ponto a ponto seguirá uma metodologia de criação e mapeamento díspar da abordagem multi-ponto, uma vez que na primeira é necessário que os parâmetros do *link* virtual sejam negociados e acordados pelos extremos.

1) Links virtuais ponto a ponto utilizando RawSockets: Os *links* ponto a ponto utilizando *RawSockets* permitem realizar o envio da informação directamente sobre o meio de comunicação, necessitando de um mecanismo que permita diferenciar o tráfego gerado pela rede virtual. Estes parâmetros são acordados pelos extremos, e permitirão identificar o *link* virtual.

Negociação de *Links* virtuais

Com o conhecimento da rede disponível, são requisitados ao utilizador os parametros de construção do *link* virtual. A interface gráfica apresentada permite que este altere as seguintes características:

- *Ip* da interface física local;
- Endereço *ethernet* da máquina virtual que pretende na interface local;
- *Ip* da interface física remota;
- Endereço *ethernet* da interface virtual a criar na máquina virtual remota;
- Largura de banda do *Link* virtual;
- Compressão de dados nos extremos;
- Valor de *QoS* a colocar nos pacotes *IP* a serem trocados.

Estes dados são estudados pela aplicação que irá atestar a sua validade. Após a verificação é criado um Objecto representativo do *link* que irá conter as informações.

A multiplexação e desmultiplexação dos *links* virtuais é um ponto importante na interligação das máquinas virtuais. Algumas abordagens utilizam o encapsulamento com estudo de cada um dos cabeçalhos de forma a saber qual o ponto final da ligação, colocando na ligação mais um nível de *overhead*[16]. A abordagem proposta está baseada num *link* virtual que utiliza encapsulamento marcado por um protocolo especifico, combinado pelos dois nós físicos. No momento em que o Objecto representativo do *link* virtual é criado, é calculado um esboço do protocolo a usar.

Este protocolo será colocado no pacote *IP* que servirá de cápsula. Para se compreender a alteração que vai ser efectuada no cabeçalho *IP* é necessário estudar os campos que o contêm.

O cabeçalho *IP* possui a forma apresentada na figura 3[14]. Neste ponto o campo protocolo é relevante para a aplicação

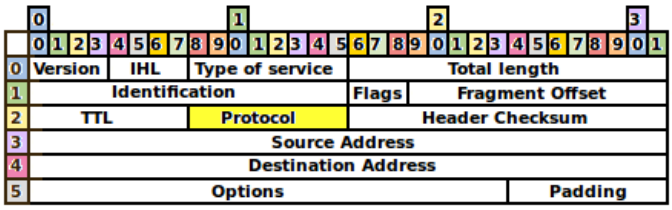

Fig. 3. Estrutura de um pacote *IP*

e será alvo de negociação entre os extremos. O campo possui 8 *bits* e normalmente indica o protocolo de próximo nível na porção de dados do pacote *IP*. Este varia para os diversos protocolos podendo ser consultado em [5], existindo ainda uma porção bastante grande de protocolos que ainda não estão usados.

O protocolo será calculado de entre os não usados, ou seja, de 143 a 252. Destes 109 valores possíveis, a aplicação escolhe um ao acaso, utilizando a função *random* e colocando-o no Objecto. É necessário que os extremos da aplicação sejam capazes de reconhecer os pacotes *IP* marcados com o protocolo como seus. Para isso, é enviado para o grupo *multicast* uma mensagem contendo o Objecto que descreve o *link* virtual que se pretende criar. A aplicação recorre ao valor de protocolo para enumerar os *links* virtuais criados localmente, onde os seus valores não podem ser repetidos, podendo, contudo, ser reutilizados num outro local da mesma rede virtual, permitindo que cada nó de suporte da rede tenha até 109 *links* distintos com protocolos diferenciados.

Cada *nó* que receba a mensagem irá verificar inicialmente se algum dos extremos indicados no Objecto recebido contém

um endereço seu. Se esta condição não for verificada, a mensagem recebida é descartada. Caso contrário, a informação recebida é importante e permitirá a criação do *link* virtual. O *host* verifica se algum outro *link* virtual partilha o protocolo com o que se pretende criar. Se sim, é computado um novo número de protocolo, da mesma forma que é feito no remetente da mensagem. Este novo número de protocolo irá substituir o computado originalmente, alterando o Objecto que descreve o *link* virtual recebido pela mensagem. Este Objecto alterado é colocado numa nova mensagem, colocando nos argumentos a *String "NOK"*.

O *host*, ao receber a mensagem e ao verificar que o argumento é *"NOK"*, testa se o novo protocolo recebido está livre localmente. Se o protocolo estiver livre envia uma nova mensagem de pedido de criação de *link*, com o Objecto contendo o protocolo sugerido. Se este não estiver livre o processo inicial é repetido, computando um novo número de protocolo e enviando para o nó remoto, numa mensagem com os campos enviados da primeira mensagem repetidos. O processo termina quando uma mensagem com argumento *"OK"* seja entregue. Esta mensagem terá como dados o protocolo final a ser usado.

Após esta mensagem ser enviada/recebida, ambos iniciam o processo de criação da conexão. A informação da conexão criada é inserida de imediato numa estrutura de dados que permite controlar os *links* virtuais criados.

O esquema temporal do acordo de protocolo está apresentado no diagrama temporal na figura 4 e permite compreender a forma como o protocolo é negociado entre os dois *hosts*.

O processo de numeração de *links* vem facilitar a

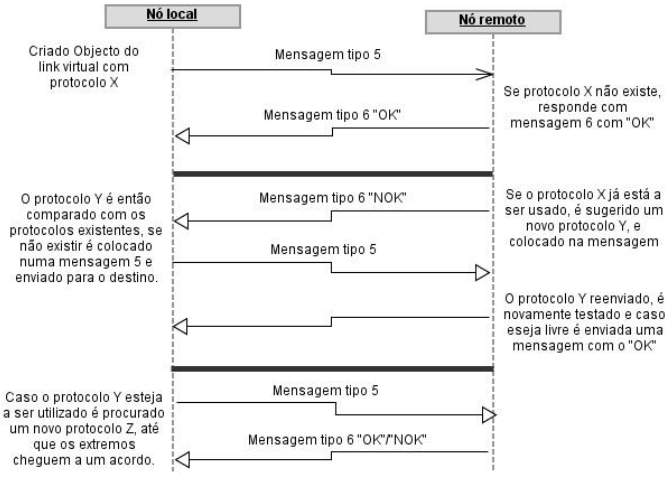

Fig. 4. Esquema temporal da negociação do protocolo

multiplexação e desmultiplexação das informações nos extremos, uma vez que uma conexão virtual será caracterizada pelo seu número de protocolo, fazendo com que os extremos possam ter vários *links* virtuais, cada um com um protocolo diferente, a funcionar sobre um físico, permitindo até ligar vários links virtuais a uma interface.

De forma a criar o *link* virtual é iniciada, em cada um dos extremos, a captura na interface do *host* virtual local, identificada pelo endereço *mac* que é passado na mensagem de criação de *link* virtual. Caso não seja passado um endereço *mac* válido, é utilizado aquele que está a ser utilizado pela máquina virtual, na interface visada.

Neste momento, é iniciada uma *Thread* que vai permitir controlar a máquina virtual local, assim como a captura de informação enviada pela interface virtual que foi passada pelo utilizador.

A captura de informação é realizada utilizando os recursos disponíveis na biblioteca *Jpcap*[9], utilizando a classe *JpcapCaptor*, e recorrendo aos filtros que este permite criar. De forma a criar o mecanismo de captura, é necessário conectar a instância *JpcapCaptor* à interface indicada pelo utilizador fazendo com que este apenas capture informação da interface da máquina virtual que interessa. Esta procura da interface é realizada usando a biblioteca *vboxjws*, um recurso da aplicação de virtualização *VirtualBox* e a biblioteca *Jpcap*. Após ser criado o Objecto *JpcapCaptor*, que estará ligado à interface pedida, é criado um filtro que permite capturar os dados enviados da interface virtual. O filtro é conjugado com as informações fornecidas pelo utilizador, *"ether src [mac]"*. São ,assim, capturadas todas as tramas de informação que sejam enviadas com o endereço *ethernet* da máquina virtual. O filtro é muito importante, na medida em que assegura que não se capturem as tramas que são recebidas, evitando ciclos.

Cada fragmento de informação nível 2 recebido, dependendo das opções do utilizador, será comprimido e colocado dentro de um pacote *IP* que servirá de cápsula. Esta será marcada com o protocolo combinado entre os extremos.

Um Objecto da classe *JpcapSender* permitirá, com métodos nativos, recorrendo ao *JNI*[11], criar um *RawSocket* que facultará o envio do pacote *IP*.

De forma a permitir a escalabilidade do sistema, o envio das cápsulas até ao destino é facultado pela rede de suporte realizando o envio para o *router* indicado pelo sistema operativo como o primeiro salto. Sendo para isto necessária a descoberta do endereço *ethernet* do *gateway* recorrendo-se a ferramentas do sistema. É assim pedido que o sistema execute o comando *"arp -a"* e a informação obtida é estudada.

O processo iniciado aquando da recepção do datagrama será incumbido de o tratar e enviar para o destino, utilizando um mecanismo de encapsulamento, que permite que os dados que são recebidos das máquinas virtuais sejam enviados utilizando a rede subjacente, alterando as características da cápsula de acordo com a experiência que se pretende realizar. O datagrama recebido poderá ser comprimido e colocado no *payload* do pacote *IP* que funcionará como cápsula, tal como representado na figura 5. Esta é alterada de acordo com o que é pretendido pelo utilizador, colocando-lhe o protocolo que indicará univocamente o canal virtual de comunicação e os cabeçalhos de destino e origem do pacote.

Da mesma forma que foi necessário, na interface virtual, um mecanismo que permitisse o envio de informação, também

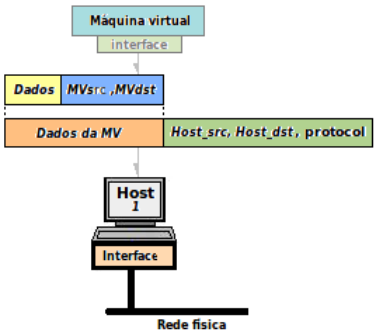

Fig. 5. Encapsulamento de mensagens para envio

na ligação com a interface física será necessária a criação de uma estrutura que estará ligada à interface externa e permitirá receber os pacotes *IP* e reenvia-los para a máquina virtual.

É criado um outro Objecto do tipo *JpcapCaptor* que será ligado à interface de comunicação com o exterior. Este permitirá a captura dos pacotes marcados com o protocolo acordado para o *link* virtual. Esta instância possuirá o filtro que permitirá separar o tráfego. O filtro terá a seguinte forma: "*ip proto [protocolo] and not src host* $[IP_{interface}]$". Os pacotes capturados serão tratados, retirando os dados do *payload*. É aqui que estarão as tramas que foram enviadas pela máquina virtual remota e que serão passadas para a máquina virtual local. Este pacote obtido do campo de dados da cápsula *IP* é então enviado para a máquina virtual.

Cada *link* virtual vai, então, ser representado por duas *Threads* locais que permitirão receber os pacotes marcados da ligação externa, enviando-os para a máquina virtual e a captura de informação da máquina virtual que após fazer o seu tratamento e encapsulamento, realiza o envio para a rede. A figura 6 apresenta o esquema lógico da comunicação entre dois nós.

Em cada *host* são criadas duas *Threads* (representadas

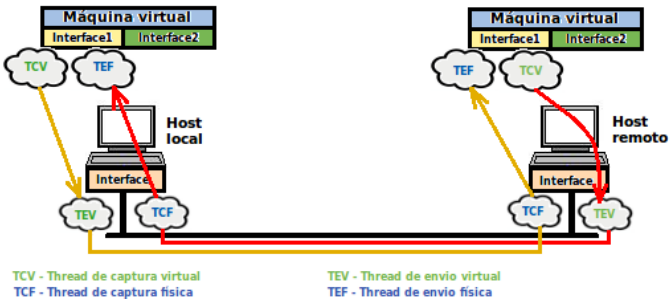

Fig. 6. Representação do esquema de comunicação entre dois nós utilizando um link ponto a ponto

em verde e azul). Cada uma destas será dividida em duas funções. A ligação à interface física, irá capturar as informações recebidas da rede[4], enviando-as para a máquina virtual[5] com a cor verde. A captura da máquina virtual[6] permite o envio destas informações pela rede física[7].

2) Links virtuais ponto a ponto usando Sockets: A ligação ponto a ponto poderá ser também criada recorrendo a *Sockets unicast* usando as capacidades que o *Java* pode oferecer. Neste tipo de abordagem não é necessária a criação de uma cápsula de transporte, pois todo o processo de transporte e recepção é processado usando *DatagramSockets*.

A aplicação requer um método de captura das informações enviadas pela máquina virtual, interagindo com a interface descrita pelo utilizador utilizando a interface gráfica.

O processo iniciado terá de recorrer a uma instância da classe *JpcapCaptor* que será conectada à interface da máquina virtual, capturando as informações enviadas.

Para que a captura se processe da forma pretendida, obtendo apenas as informações enviadas pela interface virtual, é necessário filtrar os datagramas que são trocados. É, assim, configurada a instância *JpcapCaptor* definindo os parâmetros de captura pretendidos, passando-lhe a *String* como filtro *"ether src [mac]"*, onde *[mac]* é o endereço passado pelo utilizador ou obtido da interface.

A informação recebida pela máquina virtual será comprimida, caso o utilizador assim o pretenda, colocada num *DatagramPacket* e enviada pelo *Socket*.

Uma interface externa estará responsável por capturar os pacotes enviados pelo nó remoto, colocando-os ao dispor da máquina virtual criada localmente que contém a interface com o endereço *mac* passado.

A *Thread* que irá tratar da interface externa instanciará um *DatagramSocket* que ficará à escuta e permitirá receber as informações enviadas pela máquina virtual remota.

Sempre que um datagrama remoto é recebido, o seu conteúdo é individualizado, descomprimido, caso seja necessário, e convertido num objecto do tipo *Packet*. O objecto é enviado para a máquina virtual utilizando uma instância do classe *JpcapSender*, previamente conectada à interface da máquina virtual.

3) Links virtuais multi-ponto utilizando RawSockets: Uma arquitectura de rede nível 2 poderá ser um meio partilhado entre vários nós, onde as mensagens enviadas para o meio de comunicação são partilhadas por todos os *hosts* na rede. É neste contexto que são usados os protocolos que poderão ser testados nas máquinas virtuais. Para que esta abordagem ao problema fosse possível foi projectada a melhor forma de virtualizar esta ligação, usando as máquinas virtuais como extremos de conexão da rede multi-ponto. Esta conexão virtual multi-ponto utilizará um paradigma de comunicação baseado no *multicast*.

É inicialmente implementado um grupo de controlo, utilizando uma instância *MulticastSocket*, que permitirá a

[4]TCF - *Thread* de captura da interface física

[5]TEF - *Thread* de envio da interface física

[6]TCV - *Thread* de captura na interface virtual

[7]TEV - *Thread* de envio da interface virtual

troca da informação de controlo entre os nós físicos, obtendo a informação dos *links* multi-ponto existentes. Por outro lado, os *links* de comunicação recorrem a *RawSockets* para o envio e recepção de informação. Para os criar é necessário simular o funcionamento do protocolo *multicast*, enviando as mensagens *IGMPv3* definidas em [6].

Quando o nó se liga a um endereço *multicast* são criadas as mensagens *IGMP* que permitem a junção ao grupo, estas são colocadas dentro de um pacote *IP* marcado com o campo de protocolo a 2 e enviadas para o endereço *224.0.0.22*. É também iniciada uma *Thread* que permite o envio de mensagens *Membership report* para o endereço, mantendo a subscrição no grupo *multicast* que serve de comunicação.

Neste momento o processo está ligado ao grupo *multicast*, preparado para passar para a segunda parte do algoritmo. É criada uma instância da máquina virtual e uma *Thread* de captura do tráfego que desta provém. Esta captura recorre ao *JpcapCaptor* configurado com um filtro que permite que apenas sejam capturados os datagramas que provém da máquina virtual criada. Os datagramas são então colocados dentro de um pacote *IP* e enviados para o endereço *multicast*, colocando no campo de destino o grupo indicado pelo utilizador. Os datagramas poderão ser ainda marcados com um campo protocolo especifico, que à imagem do que acontecia com os *links* virtuais *unicast*, permite a criação de várias conexões virtuais multi-ponto, utilizando o mesmo endereço *multicast*. Esta abordagem vem maximizar a utilização dos grupo *multicast*, permitindo várias redes virtuais multi-ponto sobre o mesmo grupo.

Ao mesmo tempo é iniciado o processo de captura da ligação externa que vai filtrar os datagramas que chegam ao *host*, permitindo capturar os que interessam à aplicação. O filtro, é representado pela *String* "*ip proto* $[IP_{protocol}]$ *and dst host* $[IP_{multicast}]$", isto fará com que apenas sejam disponibilizados para a aplicação datagramas que provêm do grupo *multicast* e estão marcados com o protocolo escolhido. A aplicação vai então analisar estes datagramas recebidos, retirando o *payload* que é enviado para a máquina virtual, usando uma instância do objecto *JpcapSender*.

4) Links virtuais multi-ponto recorrendo Sockets Multicast: Uma conexão multi-ponto, pode ser também criada usando uma instância do *MulticastSocket* presente no *Java*. Esta implementação será similar à solução que recorre ao *RawSocket*, executando o protocolo de ligação a um grupo *multicast*, de forma transparente ao utilizador e ao programador.

C. Guardar e carregar rede

A aplicação desenvolvida permite também guardar a rede criadas num ficheiro *XML*, permitindo altera-la e carregando-a numa outra infra-estrutura de suporte. Isto vem permitir remarcar o conceito de portabilidade do sistema, que aliado ao facto do sistema ser desenvolvido em *Java* e utilizando virtualização total, permite que execute em todos os dispositivos que possuam esta tecnologia.

O ficheiro *XML* é lido com a biblioteca *jdom*, que agrupará as informações, criando os objectos correspondentes, após o qual, comunicará com os *hosts* visados pelos *links* enviando os Objectos criados. As instâncias obtidas permitirão iniciar as entidades da rede virtual remotamente ou localmente trocando as informações obtidas do ficheiro utilizando *Sockets TCP*.

IV. Testes

O trabalho desenvolvido permite compor uma rede virtual sobre uma infra-estrutura física, criando *links* de forma a estabelecer uma arquitectura de comunicação virtual.

De forma a atestar a qualidade de cada abordagem utilizada para as criações das conexões, foram realizados vários testes, que permitem comprovar o *throughput* e perdas na ligação utilizando conexões *TCP* e *UDP* sobre os links virtuais.

Os resultados foram obtidos utilizando a ferramenta *Iperf*. O *Iperf* permite a obtenção dos valores de largura de banda efectiva em cada um dos *links*, permitindo compreender de que forma o *overhead* introduzido pela aplicação afecta a qualidade da ligação.

Foram usados para os testes três computadores portáteis instalados com *Ubuntu 10.10*. Cada nó virtualizado possui o sistema operativo *FreeBsd-8.2* que executa na ferramenta de virtualização *VirtualBox*.

A abordagem para o teste ponto a ponto contempla dois portáteis em redes distintas onde se testam as abordagens utilizando uma comunicação *TCP* e *UDP*. A primeira atesta a velocidade máxima que o *link* é capaz de oferecer numa comunicação sem perdas, enquanto que a segunda vai requisitando ao *link* virtual larguras de banda cada vez mais altas, controlando os valores que esta pode oferecer e as perdas registadas.

O teste multi-ponto utiliza os três portáteis de forma a controlar a chegada de datagramas aos três pontos onde serão controlados, à imagem do teste ponto a ponto, os valores de largura de banda real e a percentagem de perdas.

O primeiro teste apresentado permite comprovar a largura de banda que é obtida em cada uma das conexões criadas. Desta forma, uma das máquinas virtuais funciona como servidor enquanto que outra, remotamente, cria ligações *TCP*. O tráfego *TCP* será sempre entregue, o que faz com que caso alguma porção de informação enviada seja perdida, é reenviada resultando numa degradação do valor de largura de banda efectiva. Inicialmente foi verificada a largura de banda que a rede de suporte é capaz de oferecer, este valor permite obter um termo de comparação em cada uma das abordagens. Os resultados obtido podem ser comprovados na figura 7:

Como é possível verificar, as abordagens implementadas utilizam uma percentagem da largura de banda reduzida em comparação com a rede de suporte. Esta situação pode ser explicada pela forma como o envio de informação é realizado. A biblioteca utilizada, *Jpcap*, necessita que os objectos se encontrem serializados aquando do envio e recepção. Por esta razão, não é possível aproveitar ao máximo o espaço disponível num datagrama *ethernet* uma vez que

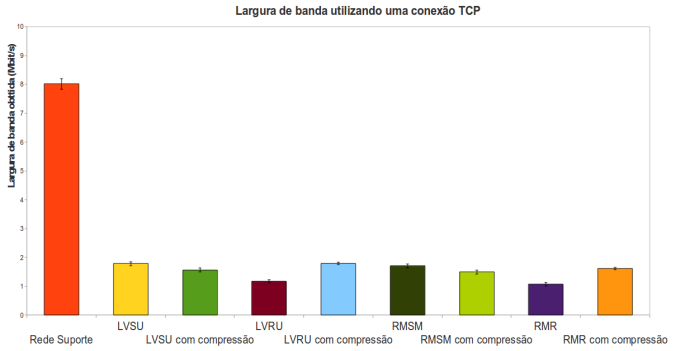

Fig. 7. Valores de largura de banda efectiva entre as máquinas virtuais, utilizando as diversas propostas de criação de *link*

alguma da informação que é transportada não é utilizável pela máquina virtual. Contudo, é possível verificar que a abordagem ponto a ponto que recorre a *RawSocket* com compressão apresenta uma largura de banda efectiva mais elevada, o que pode ser explicado pela forma como ocorre a multiplexação/desmultiplexação da informação, uma vez que a utilização do protocolo para identificar a máquina virtual a que deve ser entregue a trama presente no *payload* permite reduzir o *overhead* que seria gerado com a utilização de uma desmultiplexação baseada nos endereços.

O segundo teste utiliza uma comunicação *UDP* para envio de informação, este permite controlar a largura de banda efectiva máxima da conexão virtual criada testando-a com os parâmetros que são pedidos pela ligação *UDP* criada pelo *Iperf*. O *link* virtual atingirá um ponto a partir da qual começará a perder datagramas limitando o aumento da largura de banda. Os gráficos na figura 8 e figura 9 permitirão demonstrar o comportamento de cada uma das ligações criadas[8].

O gráfico 8 permite atestar a largura de banda máxima

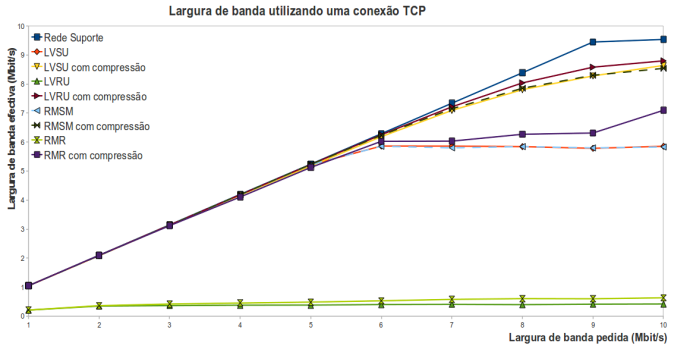

Fig. 8. Valores de largura de banda efectiva entre as máquinas virtuais utilizando uma conexão *UDP*

que cada uma das ligações é capaz de oferecer, uma vez que

[8]*LVSU* - *Link* virtual utilizando *sockets UnicastLVRU* - *Link* virtual utilizando *RawSockets*
RSMS - Rede Multiponto utilizando Sockets *Multicast*
RMR - Rede multi-ponto utilizando *RawSockets*

todos os gráficos tenderão para um valor máximo a partir do qual estabilizam, limitando o valor máximo de largura de banda e aumentando o valor de perdas registadas, que pode ser verificado pela figura 9.

Os dois gráficos baseados nas conexões *UDP* estão

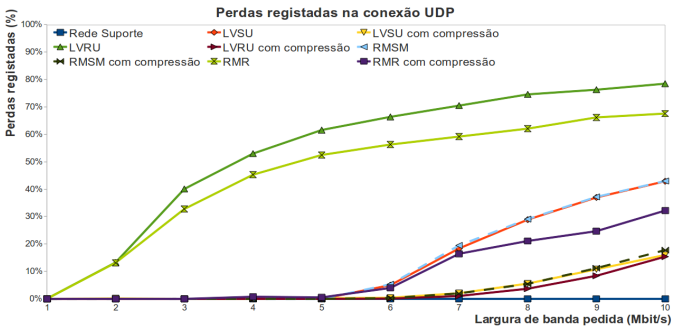

Fig. 9. Valores em percentagem das perdas efectivas de datagramas *UDP* na ligação

interligados, pois é possível verificar que quanto mais largura de banda é pedida, maior será a resposta do *link* virtual criado. Este comportamento regista-se até ao limite do *link*, a partir do qual a largura de banda efectiva estabiliza aumentando o valor das perdas.

A diferença principal entre as abordagens com *Sockets UDP* e *RawSockets* está na forma como a informação vai ser enviada, registando-se comportamentos distintos nas duas. Nas abordagens com *Sockets UDP* verifica-se que quando não existe compressão de dados as informações são normalmente fragmentadas, o que provoca um degradamento da largura de banda efectiva na conexão. Este comportamento é transposto para a abordagem *RawSocket*, onde para solucionar este problema, o *MTU* da conexão tem de ser reduzido, esta alteração do *MTU* é necessária para as ligações que recorrem ao *RawSocket* devido à necessidade de encapsulamento dos datagramas recebidos da máquina virtual em pacotes *IP* e posteriormente em datagramas *ethernet*. A necessidade da biblioteca de captura recorrer à serialização de Objectos para tratar os datagramas conduz a que muita da informação que é enviada seja supérflua, provocando uma quebra na performance das ligações. Contudo, a utilização da compressão permite solucionar estes dois problemas e obter larguras de banda perto da capacidade da rede de suporte e perdas reduzidas.

É possível verificar que a mesma abordagem, utilizando compressão, onde se retira a necessidade de alteração do *MTU*, possui o melhor valor registado, comprovando que a utilização do campo protocolo para multiplexação e desmultiplexação das conexões se apresenta como uma mais valia.

V. Conclusões e Trabalho Futuro

As virtualização de redes é, neste momento, vista como um catalisador para o desenvolvimento da *Internet* do futuro.

Sendo olhada tanto por operadoras como académicos como uma ferramenta promissora, capaz de oferecer a possibilidade de obter lucro com várias redes virtuais sobre a mesma infra-estrutura física. A ferramenta capacita também o amadurecimento e desenvolvimento de protocolos de rede sobre uma infra-estrutura controlada, permitindo a empresas e académicos o desenvolvimento de novos projectos. As possibilidades das redes virtuais são reais e participarão cada vez mais activamente no nosso dia a dia, oferecendo novos produtos e serviços que até agora estavam limitados.

Vários avanços foram realizados nos últimos anos nesta vertente. Projectos como o *4Ward* na Europa e o *VINI* na América, permitem comprovar o investimento e a importância que esta tecnologia pode apresentar no futuro. O próximo passo aparenta ser a passagem da virtualização de redes para o *backbone* das operadoras, criando várias redes com parâmetros distintos de forma a servir utilizadores com necessidades especificas.

Este artigo propõe uma abordagem de comunicação entre os nós virtuais onde estes trocam mensagens de nível 2, apresentando várias possibilidades de criação de um *link* consoante as necessidades do utilizador, criando uma arquitectura de rede controlada onde é possível testar novos protocolos e tecnologias de rede. Esta rede permite uma dinâmica de alteração, onde as características da máquina virtual e dos *links* pode ser alterada a pedido, permitindo comparar situações facilmente.

A solução proposta foi implementada num protótipo funcional, que permite a procura de recursos disponíveis para a criação da rede virtual, apresentando-os como alternativas para um *link* ponto a ponto. Os *links* ponto a ponto foram implementados recorrendo a um envio de informação utilizando *Sockets* e um outro onde o utilizador define os parâmetros da cápsula de transporte, onde se recorre a *RawSockets*. A mesma metodologia foi utilizada com a criação de *links* multi-ponto, entre vários nós virtuais. A capacidade de comprimir os *streams* de dados das máquinas virtuais, permite realizar uma troca entre largura de banda disponível e processamento. Os datagramas enviados são comprimidos de forma a gastar o mínimo de largura de banda, assegurando a comunicação com um débito efectivo superior ao que seria utilizável, em alguns casos, pela comunicação sem compressão.

A possibilidade de realizar a multiplexação/desmultiplexação dos canais de comunicação com recurso ao campo protocolo, colocando de parte o encapsulamento utilizado nos túneis *IP*, permite retirar um nível de *overhead* que normalmente é observado nestas implementações, resultando numa entrega da informação muito mais clara e simples entre as máquinas virtuais e mantendo o escalabilidade do sistema. As implementações estudadas propõem a criação de uma plataforma nível 3 virtual sobre uma infra-estrutura de rede do mesmo nível, mantendo as mesmas limitações de protocolos registadas na rede pública, a proposta apresentada tenta ultrapassar este pressuposto oferecendo um meio onde a comunicação virtual decresce um nível lógico, permitindo a virtualizar a camada de ligação, sobre uma infra-estrutura nível 3. Esta comunicação virtual ao nível 2 pode ainda ser estendida à topologia criada, criando uma topologia com conexões entre dois ou mais nós.

Os valores obtidos nos testes realizados permitem concluir que as ligações criadas, permitem uma largura de banda razoavelmente perto da oferecida pela rede de suporte mesmo considerando o *overhead* introduzido na ligação.

As ligações criadas são focadas na troca de informação, não sendo do âmbito do projecto a aposta na performance, contudo, foram analisados os pontos de *overhead* existentes no projecto. A serialização de objectos significou uma queda na performance bastante acentuada e por esta razão foi criada uma biblioteca simples de captura e envio de datagramas, similar ao *Jpcap*, capaz de obter os datagramas sobre a forma de uma sequência de *bytes*, e envia-los da mesma forma. A implementação desta nova biblioteca, é definida como um projecto futuro.

O desenvolvimento do projecto poderá recair sobre a proposta de uma abordagem para ultrapassar *proxies* e *NAT* permitindo que estes nós possam ligar-se a uma rede virtual. Alguns dos projectos estudados apresentam uma solução para este problema mas não foram testados por ultrapassarem o âmbito do que era esperado para o trabalho. Um outro ponto considerado para trabalho futuro é a incorporação de um mecanismo de segurança e reconhecimento dos nós que vão criar a rede.

REFERENCES

[1] SIGAR API (System information gatherer and reporter).

[2] 4ward – architecture and design for the future internet. *4WARD*, D-3.2.1, June 2010.

[3] *Oracle VM VirtualBox - Programming Guide and Reference*, 2011.

[4] David G. Andersen. Theoretical approaches to node assignment. January 2002.

[5] J. Arkko and S. Bradner. Protocol numbers (rfc5237), July 2011.

[6] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet group management protocol, version 3 (rfc3376), October 2002.

[7] J. Carapinha and J. Jiménez. Network virtualization - a view from the bottom. 2009.

[8] Xuxian Jiang Dongyan. Violin: Virtual internetworking on overlay infrastructure. 2003.

[9] K. Fujii. Jpcap tutorial, 2007.

[10] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch. Next steps in signaling (nsis): Framework (rfc4080), 2005.

[11] S. Liang. *The Java Native Interface - Programmer's Guide and Specification*, 1999.

[12] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. A distributed approach for virtual network discovery. 2009.

[13] J. Nogueira, M. Melo, J. Carapinha, and S. Sargento. Virtual network mapping into heterogeneous substrate networks. 2009.

[14] J. Postel. Internet protocol (rfc 791), September 1981.

[15] H. Schulzrinne and R. Hancock. Gist: General internet signalling transport (rfc5971), October 2010.

[16] J. Touch. Dynamic internet overlay deployment and management using the x-bone. 2000.

[17] J. Touch and S. Hotz. The x-bone. 1998.

[18] M. Tsugawa and J. A. Fortes. A virtual network (vine) architecture for grid computing. *In Proc. of 20th International Parallel and Distributed Processing Symposium (IPDPS-2006)*, 2006.

[19] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. 2006.