Rodrigo Ramiro Muniz Junior

Estudo do protocolo de sinalização NSIS para reserva de recursos

São José - SC

Março / 2009

Rodrigo Ramiro Muniz Junior

Estudo do protocolo de sinalização NSIS para reserva de recursos

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Eraldo Silveira e Silva

Curso Superior de Tecnologia em Sistemas de Telecomunicações
Instituto Federal de Santa Catarina

São José - SC

Março / 2009

Monografia sob o título *"Estudo do protocolo de sinalização NSIS para reserva de recursos"*, defendida por Rodrigo Ramiro Muniz Junior e aprovada em 09 de março de 2009, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Ms. Eraldo Silveira e Silva
Orientador

Prof. Dr. Evandro Cantú
IF-SC

Prof. Ms. Odilson Tadeu Valle
IF-SC

Sempre que te perguntarem se podes fazer um trabalho,
respondas que sim e te ponhas em seguida a aprender como se faz.
F. Roosevelt

Agradecimentos

À minha mãe, pela base de educação que sempre me deu força para encarar a vida de frente. Por cumprir este papel magnificamente e pelo amor transmitido. Essa monografia é uma homenagem ao seu trabalho, a quem peço desculpas por não dar a devida atenção nos últimos meses para a conclusão deste.

Ao meu orientador Eraldo, por todo o conhecimento passado, pelas excelentes supervisões e orientação e por ter acreditado no meu trabalho. Obrigado por me proporcionar grande tempo de atuação ao seu lado, com serenidade. Torço pra que esta parceria continue de maneira íntegra.

Aos professores e professoras do IF-SC que fizeram parte dessa jornada em sala de aula, e na instituição. Aos servidores que sempre me ajudaram, direta e indiretamente. Parabenizo pela excelente qualidade do curso, visto pelo reconhecimento interno e externo.

Aos colegas do curso pela alegria e companhia na troca de informações e discussões que nos fizeram crescer, pela demonstração de amizade e solidariedade.

E, com chave de ouro, à minha namorada, pela amor, força e amizade. Pela companhia, sorriso e apoio, me iluminando com alegria e força de vontade. Pelas broncas que me fizeram manter o foco e manter a serenidade. Agradeço por ficar do meu lado, mesmo muita vezes não sendo fácil, me incentivando.

A Deus.

Resumo

Os protocolos de sinalização podem ser utilizados para a requisição de recursos em redes IP, ao longo de um caminho de comunicação. Desta forma, é possível reservar recursos para um determinado fluxo, desde o terminal emissor até o receptor, passando pelos roteadores intermediários. Modelos de qualidade de serviço com classes, tal como o *DiffServ*, também podem se beneficiar de tais protocolos para uma melhor otimização no uso de recursos.

Este trabalho irá discorrer a respeito do protocolo de sinalização NSIS, apontado como substituto do RSVP, mostrando suas características, utilização e vantagens. O NSIS ainda está em fase de discussão no âmbito da IETF, tendo algumas poucas implementações experimentais.

Os vários mecanismos do NSIS são explorados através da construção de cenários em máquinas virtuais, utilizando uma implementação real, da Universidade de Goettingen, na Alemanha. Entre outras características, são mostradas a sinalização de reserva a favor e contra o fluxo de dados e, a reserva de recursos propriamente dita, através do interfaceamento com uma função de gerenciamento de recursos.

Como contribuição adicional, foi realizado um primeiro estudo de uma implementação do NSIS, com fins de verificar a viabilidade na modificação de alguns mecanismos para utilização em modelos *DiffServ*.

Abstract

A signaling protocol may be used for resource reservation on communication paths in IP networks. In this sense, it is possible to signal for a given flow from the sender to a receiver locking resources in the intermediate routers on the path. Models of quality of service and classes, such as diffserv can also take advantage of such protocols to optimize the use of resources.

This work will explore the NSIS signaling protocol, considered as a substitute for RSVP, showing its characteristics, applications and advantages. The NSIS is still being discussed in the IETF and it has a few experimental implementations.

The mechanisms of the NSIS are exploited through the construction of scenarios in virtual machines, using a real implementation of the University of Goettingen. Among other features, it is shown the traces of reservation messages down and upstream to the data flow. In addition, a scenario shows the resource reservation carried out by a resource management function.

As an additional contribution, it was made a first study of an implementation of NSIS aiming to test the feasibility of the some mechanisms extensions for use in diffserv models.

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas p. 12

1 Introdução p. 14

1.1 Motivação . . . . . p. 14

1.2 Objetivo . . . . . p. 15

1.3 Organização do texto . . . . . p. 15

2 Qualidade de Serviço em redes IP p. 17

2.1 Histórico de QoS em Redes . . . . . p. 17

2.2 O Conceito e Parâmetros de QoS . . . . . p. 19

2.3 Modelo IntServ . . . . . p. 20

2.4 Modelo DiffServ . . . . . p. 21

2.5 Os protocolos de Sinalização e os Modelos de QoS . . . . . p. 23

2.5.1 O Protocolo de Sinalização como Elemento de Interligação de Modelos de QoS . . . . . p. 24

3 O protocolo NSIS p. 25

3.1 Visão Geral do projeto NSIS . . . . . p. 25

3.2 O Protocolo GIST . . . . . p. 26

3.2.1 O problema de mudança de rotas . . . . . p. 28

3.3 O protocolo QoS NSLP . . . . . p. 29

3.3.1 Associações de Mensagens . . . . . p. 31

3.3.2 Associação de Sessões e Reservas de Agregados . . . . . p. 32

3.3.3 Funcionamento com Estado Reduzido ou sem Estado . . . . . p. 33

3.3.4 Re-roteamento . . . . . p. 33

3.3.5 Interações com o Gist . . . . . p. 35

3.3.6 Considerações sobre QSPEC . . . . . p. 36

3.4 Outros Protocolos NSLP . . . . . p. 37

4 Cenários p. 39

4.1 Cenários estudados . . . . . p. 39

4.2 Aspectos da Plataforma de Testes e da Operação do protocolo de Goettingen . . . . . p. 39

4.2.1 Plataforma de Testes . . . . . p. 39

4.2.2 Estrutura e Configuração do NSIS . . . . . p. 40

4.2.3 O cliente do protocolo QoS-NSLP . . . . . p. 41

4.3 Cenário 1 . . . . . p. 42

4.4 Cenário 2 . . . . . p. 45

4.5 Cenário 3 . . . . . p. 48

4.5.1 Iniciando a função de RMF do NSIS . . . . . p. 49

4.5.2 Efetuando a reserva para um fluxo . . . . . p. 51

4.5.3 Análise da Saída do Depurador . . . . . p. 52

4.6 Cenário 4 . . . . . p. 55

4.6.1 Iniciando a função de RMF do NSIS . . . . . p. 56

4.6.2 Efetuando a reserva para um fluxo . . . . . p. 57

4.7 Cenário 5 . . . . . p. 60

5 O protocolo NSIS pela Universidade de Goettingen p. 63

5.1 O NSLP . . . . . p. 64

5.2 O GIST . . . . . p. 65

5.3 Modelo de mensagens e sugestão de alteração . . . . . p. 66

6 Conclusões e Perspectivas p. 68

Apêndice A – Montagem de Máquinas Virtuais p. 70

Apêndice B – Arquivo de configuração do NSIS p. 74

B.0.1 Arquivos Cenários 1 . . . . . p. 79

B.0.2 Arquivos Cenários 2 . . . . . p. 80

B.0.3 Arquivos Cenários 3 . . . . . p. 81

B.0.4 Arquivos Cenários 4 . . . . . p. 82

B.0.5 Arquivos Cenários 5 . . . . . p. 83

Referências Bibliográficas p. 85

Lista de Figuras

2.1 Modelos de redes . . . . . p. 18

2.2 Campo *DiffServ Code Point* . . . . . p. 22

3.1 Modelo Camada NSIS . . . . . p. 25

3.2 Modelo estrutura GIST *(draft-ietf-nsis-ntlp-19)* . . . . . p. 26

3.3 Modelo camadas e posicionamento do GIST . . . . . p. 27

3.4 Modelo Mensagem GIST . . . . . p. 28

3.5 Estrutura do NSLP (*draft-ietf-nsis-qos-nslp-16*) . . . . . p. 30

3.6 Modelo associação de Mensagens . . . . . p. 32

3.7 Modelo de mensagem em estado reduzido ou sem estado . . . . . p. 34

3.8 Exemplo de sinalização com objetos QSPEC . . . . . p. 36

3.9 Objetos da estrutura QSPEC . . . . . p. 37

4.1 Cenário 1 . . . . . p. 42

4.2 Sinalização de reserva . . . . . p. 45

4.3 Cenário 2 . . . . . p. 46

4.4 Fluxo de Sinalização de reserva com query . . . . . p. 48

4.5 Cenário com reserva de recurso . . . . . p. 49

4.6 Criação da estrutura de controle de tráfego . . . . . p. 50

4.7 Sinalização de reserva . . . . . p. 52

4.8 Cenário com reserva de recurso . . . . . p. 56

4.9 Classes de fluxo do controle de tráfego . . . . . p. 59

4.10 Sinalização com mudanca de rota . . . . . p. 60

5.1 Estrutura de camadas NSIS . . . . . p. 64

B.1 Cenário 1 . . . . . p. 79

B.2 Sinalização de reserva com query . . . . . p. 80

B.3 Sinalização de reserva . . . . . p. 81

B.4 Cenário com reserva de recurso . . . . . p. 82

B.5 Sinalização com mudanca de rota . . . . . p. 84

Lista de Abreviaturas e Siglas

AF *Assured forwarding*

API *Application Programming Interface*

DiffServ *Differentiated Services*

DSCP *Differentiated Services Code Point*

EF *Expedited forwarding*

EuQoS *End-to-end Quality of Service support over heterogeneous networks*

GIST *General Internet Signaling Transport*

IETF *Internet Engineering Task Force*

IntServ *Integrated Services*

IP *Internet Protocol*

ISO *International Organization for Standardization*

ITU *International Telecommunication Union*

ITU-T *ITU's Telecommunication Standardization Sector*

MRI *Message Routing Information*

NSIS *Next Steps in Signaling*

NTLP *NSIS Transport Layer Protocol*

NSLP *NSIS Signaling Layer Protocol*

NSLPID *Identification NSIS Signaling Layer Protocol*

PHB *Per Hop Behavior*

QDISC *Queueing Discipline*

QNE *QoS NSIS Entity*

QNI *QoS NSIS Initiator*

QNR *QoS NSIS Receiver*

QoS *Quality of Service*

QoS NSLP *QoS NSIS Signaling Layer Protocol*

QSPEC *QoS Specifications*

RFC *Request For Comments*

RMF *Resource Management Function*

RSPEC *Rate Specification*

RSVP *Resource Reservation Protocol*

SID *Session Identification*

SII *Source Identification Information*

TCP *Transmission Control Protocol*

TCP/IP *Transmission Control Protocol over Internet Protocol*

TSPEC *Traffic Specification*

UML *User Mode Linux*

1 *Introdução*

As redes IP, em particular a Internet, vem evoluindo no sentido de suportar fluxos de informação com diferentes requisitos de qualidade de serviço. Este tráfego associado a aplicações multimídia, que inclui a voz e o vídeo, é cada vez mais intenso. Neste contexto, a Qualidade de Serviço (*Quality of Service* (QoS)) fim-a-fim, colocada como propriedade emergente da concatenação de redes e domínios diferentes, se torna imperativa.

A implementação efetiva da QoS fim-a-fim depende de mecanismos das diversas camadas da arquitetura da Internet. Alguns modelos foram propostos pela *Internet Engineering Task Force* (IETF), tais como o *Integrated Services* (IntServ) e o *Differentiated Services* (DiffServ), para suprir a necessidade de QoS. Para suportar estes modelos e a interconexão entre os mesmos, se torna necessário (no caso do *IntServ*) ou desejável (no caso do *DiffServ*), o uso de protocolos de sinalização.

O protocolo *Next Steps in Signaling* (NSIS) é uma evolução do *Resource Reservation Protocol* (RSVP) e, surge como uma proposta genérica da IETF de protocolo de sinalização, aplicável em QoS e outras áreas tais como medição de recursos e *firewalls*. No caso específico de controle de QoS, o protocolo permite a especificação e transporte de objetos que permitirão a alocação de recursos ao longo do trajeto entre um emissor e um receptor de um fluxo ou agregado de fluxo. Este trajeto é constituído pelos diversos roteadores definidos pelo sistema de roteamento subjacente da arquitetura IP.

1.1 Motivação

O fato do autor deste trabalho atuar, na sua vida profissional, no setor de pesquisa e desenvolvimento de uma empresa de tecnologia, dentro da área de redes, fez despertar o interesse na pesquisa sobre a obtenção de QoS em redes IP. Isto porque o controle de QoS dentro da rede do usuário final é um dos interesses da empresa que trabalha, pois grande parte dos equipamentos comercializados pela mesma se aplicam a este segmento.

O estudo do protocolo NSIS, uma proposta relativamente recente, foi então considerada conjuntamente com o orientador, posto que ampliaria os horizontes na área de estudo, além da ampliação da visão das atividades dentro da empresa, permitindo-lhe contribuir na melhoria dos serviços e produtos ofertados. É interessante salientar que equipamentos de qualidade podem ser erroneamente considerados ineficientes em redes sem nenhum controle de QoS.

1.2 Objetivo

Os objetivos gerais do trabalho são, portanto, o estudo do protocolo NSIS e a verificação da sua aplicabilidade para o controle de QoS em redes. Mais especificamente, pretendeu-se:

- Estudar os *drafts* referentes aos protocolos *NSIS Transport Layer Protocol* (NTLP) (*General Internet Signaling Transport* (GIST)) e *QoS NSIS Signaling Layer Protocol* (QoS NSLP), integrantes da proposta NSIS, reconhecendo os diversos mecanismos que o fazem interessante para uso em modelos *IntServ* e *DiffServ*;
- Consolidar o aprendizado através da elaboração de alguns cenários de uso do protocolo QoS NSLP, utilizando-se de máquinas virtuais[1];
- Como contribuição, estudar uma implementação do protocolo com fins de permitir, futuras modificações experimentais no código. Em particular para extensões de uso para o modelo *DiffServ*;

1.3 Organização do texto

O texto está organizado da seguinte forma. O capítulo 2 revisa o conceito de QoS e os modelos *IntServ* e *DiffServ*. No capítulo 3 é apresentado o NSIS, explorando-se o protocolo genérico de sinalização - GIST e principalmente o protocolo específico para configuração de QoS, o QoS NSLP. Outras aplicações de sinalização serão brevemente apresentadas.

O capítulo 4 apresenta alguns cenários estudados, explorando principalmente a sinalização para a reserva de recursos para um fluxo individual, nas duas modalidades: reserva a partir do emissor e do receptor do fluxo. Em adição é explorado o mecanismo de *soft-state*

[1]Máquina virtual é o nome dado a uma máquina, implementada através de software, que executa programas como um computador real.

do protocolo e propriedade de atualização automática de rotas. Finalmente no capítulo 5 apresentamos de forma geral a estrutura da implementação do NSIS pela Universidade de Goettigen. O capítulo 6 conclui, apresentando as perspectivas de continuidade do trabalho.

2 Qualidade de Serviço em redes IP

As redes *Internet Protocol* (IP), originalmente baseadas em serviço de melhor esforço[1], vêm evoluindo para redes de comunicação capazes de suportar fluxos de pacotes com diferentes requisitos, em particular o tráfego multimídia. Surge então a necessidade de garantir níveis apropriados de Qualidade de Serviço nestas redes.

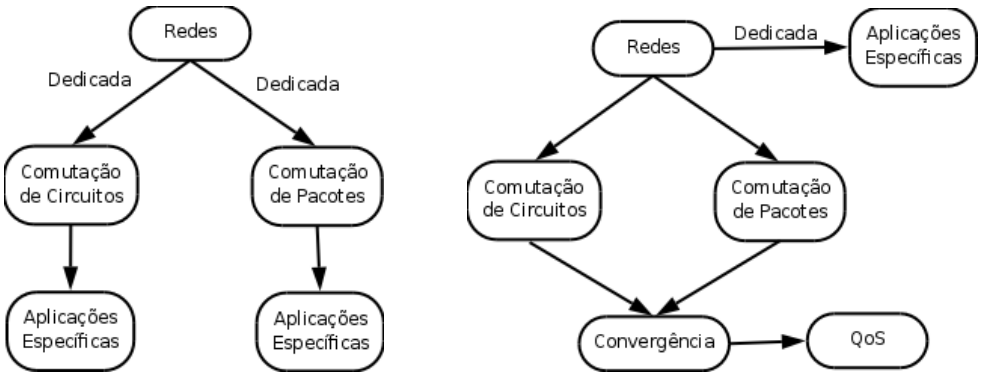
Neste capítulo o termo QoS é conceituado e faz-se uma rápida abordagem dos modelos clássicos de QoS para redes IP: o *DiffServ* e o *IntServ*. Ao final, os protocolos de sinalização são caracterizados como elementos de apoio a construção destes modelos e como ferramenta de interconexão entre os mesmos.

2.1 Histórico de QoS em Redes

Nos primórdios das telecomunicações a comunicação era feita dedicando circuitos para a transmissão de uma determinada mídia. Esta abordagem é a base das redes com comutação de circuitos, cujo principal exemplo são as redes de telefonia. Com a evolução dos computadores surgiram as redes de comutação de pacotes, dominadas na atualidade pela tecnologia de redes IP. No início, era possível determinar exatamente a aplicabilidade destas redes: as redes com comutação de circuitos basicamente para o transporte de voz e com comutação de pacotes para o transporte de dados. A figura 2.1(a) representa esta classificação.

As redes com circuitos dedicados garantem quase que completamente a integridade dos fluxos transportados, dado que existem recursos especificamente dedicados para eles. Já as redes por pacotes se utilizam de uma forma de multiplexação que pode eventualmente

[1] Melhor esforço é uma classe de tráfego utilizada em redes que envia fluxos de dados, ao mesmo tempos com largura de banda compartilhada com demais fluxos de dados enviados, tornando-os concorrentes entre si. Em caso de congestionamento de dados, os pacotes são descartados sem qualquer critério nem distinção, mas se propondo a fazer o melhor possível para a entrega de cada pacote.



(a) Antigo Modelo de rede (b) Atual Modelo de rede

Figura 2.1: Modelos de redes

comprometer a cadência e gerar tempo de atraso de pacotes de um fluxo e perda de pacotes.

A busca por qualidade de serviço remonta aos tempos dos sistemas telefônicos clássicos. Park (PARK, 2005) aponta que no início da rede de telefonia, haviam duas medidas chave de qualidade de serviço, probabilidade de bloqueio e a capacidade de percepção de usuários. A probabilidade de bloqueio, é a probabilidade de não conseguir fazer a chamada por conta da indisponibilidade de um tronco do circuito. A outra medida de qualidade foi a da percepção de voz, que dependia da qualidade de transmissão ponto a ponto durante uma chamada.

Com a evolução das tecnologias, a convergência digital passou a ser uma realidade. As redes para o transporte de dados e de voz e imagem tendem a ser integradas: são as redes redes convergentes, conforme mostrado na figura 2.1(b).

As redes convergentes, com interfaces abertas e com capacidade de transmitir dados multimídias, como a voz, exploram plenamente as tecnologias de ponta para oferecer serviços sofisticados. Em particular, as redes IPs surgem como uma solução madura e barata para esta finalidade. Entretanto, visto que o protocolo IP não foi originalmente concebido para suportar este tráfego multifacetado, é que surge a necessidade de incorporar soluções que forneçam alguma qualidade de serviço a estes fluxos. Dentro deste contexto a IETF propõe alguns modelos para o provimento de QoS. Nas seções subsequentes são apresentados os modelos *IntServ* e *DiffServ* e, uma breve discussão sobre o uso de protocolos de sinalização para a viabilização dos mesmos.

2.2 O Conceito e Parâmetros de QoS

Park (PARK, 2005) define QoS a partir de dois pontos de vista: a QoS experimentado pelo usuário e a QoS do ponto de vista da rede. Da perspectiva do usuário ou de uma aplicação, a QoS é a percepção da qualidade que ele recebe da rede para um serviço específico que envolva a transmissão da voz, vídeo e dados. Na perspectiva da rede, a QoS refere-se a capacidade desta, de fornecer a QoS percebida pelo usuário.

Park (PARK, 2005) cita ainda que a *International Organization for Standardization* (ISO), órgão que organiza padrões internacionais com sede na Suíça, define qualidade como "a soma das características de uma entidade como a capacidade de suportar e satisfazer necessidades implícitas dado por um conjunto de características intrínsecas". A *ITU's Telecommunication Standardization Sector* (ITU-T) (Recomendação E.800 (ITU-T, 1994)) define basicamente Qualidade de Serviço como "o efeito coletivo do desempenho de um serviço que irá determinar o grau de satisfação de um usuário deste serviço".

A IETF considera QoS (MARCHESE, 2007) como a capacidade de segmentar o tráfego ou diferenciar os tipos de tráfego para a rede, a fim de tratar determinados fluxos de tráfego de maneira diferente dos outros. QoS engloba tanto o serviço de categorização como também o desempenho global da rede para cada categoria. De forma similar, a ITU-T na resolução E-800 (ITU-T, 1994), destaca a necessidade da rede oferecer requisitos de performance e operabilidade para o provimento de QoS.

Para este trabalho será assumido que QoS será a garantia oferecida pela rede de que certos parâmetros serão mantidos entre níveis pré-acordados. Neste sentido, os principais parâmetros considerados serão a largura de banda, o retardo fim-a-fim, a variação do atraso entre pacotes (*jitter*) e as perdas de pacotes.

A largura de banda mede o número de *bits* por segundo em média que passam pela rede. O atraso fim-a-fim refere-se ao atraso médio dos pacotes de um fluxo desde a fonte até o destino. O *jitter* é a variação do atraso entre pacotes para um mesmo fluxo. Em geral, pacotes sofrem atraso diferenciado na rede, o que faz com que fluxos com cadência fixa, por exemplo originados pela digitalização da voz, sofrerem um total descompasso entre pacotes.

É interessante ainda observar que os parâmetros de QoS definidos pela *International Telecommunication Union* (ITU) (MARCHESE, 2007) são colocados de uma forma um pouco diferenciada, ou melhor, na forma de indicadores de desempenho conforme colocado abaixo:

- IPLR – IP *Packet Loss Ratio* (Taxa de perdas)
- IPTD – IP *Packet Transfer Delay* (Atraso)
- IPDV – IP *Packet Delay Variation* (Conhecido como Jitter)
- IPER – IP *Packet Error Ratio.* (Taxa de erro)

Os indicadores acima podem ser entendidos como métricas que classificam classes de QoS conforme as mensurações de QoS (atraso, variação do atraso, perda) e os requerimentos para minimizar a percepção de problemas.

Os mecanismos de QoS são aplicados, em geral, a saída de um nó da rede, onde a perda de pacotes é ocasionada, na maioria das vezes, por estouro de *buffers* associados as interfaces de roteadores. Os mecanismos envolvidos neste controle, a interação entre os mesmos, bem como as políticas e formas de gerenciamento neste processo constituem um modelo de QoS.

2.3 Modelo IntServ

Em redes de computadores, *IntServ*, ou serviços integrados, é uma arquitetura que especifica os elementos para garantir a qualidade de serviço em redes com requisitos estritos por fluxo. A idéia do *IntServ* é que cada roteador que o implementa, e todas as aplicações que requerem algum tipo de garantia, tem de fazer uma reserva individual por fluxo.

O *IntServ* provê, portanto, a garantia de qualidade de serviço para fluxos individuais, mantendo estados (incluindo reserva de estados) associados ao fluxo fim a fim, em todos os roteadores por onde este irá passar. Esta arquitetura foi projetada para a internet, porém o processo de alocação de recursos do *IntServ* tem um custo alto para os roteadores, pois cada roteador terá de manter as informações do fluxo (KUROSE; ROSS, 2006).

Para manter os estados da rede com fins de fornecer QoS para fluxos individuais, o *IntServ* se utiliza de uma entidade de sinalização, o RSVP (*Resource ReSerVation Protocol*). Este protocolo é capaz de iniciar e manter o estado da reserva do *IntServ* por fluxo, além de armazenar apontadores por fluxo, que permitem sinalizar na direção e contra o fluxo. O estado dos diferentes fluxos recebidos deve ser mantido e monitorado nos roteadores. Esta abordagem difere acentuadamente do serviço de melhor esforço fornecidos pela internet, pois o fluxo recebe uma garantia de banda dentro da rede.

O *IntServ* (BRADEN; CLARK; SHENKER, 1994) pode fornecer dois tipos de serviços: o serviço de carga controlada e os serviços garantidos. O serviço de carga controlada fornece a um fluxo uma aproximação de QoS que o mesmo fluxo receberia se a rede estivesse livre. Clientes que requisitam este serviço (WROCLAWSKI, 1997), fornecem aos componentes intermediários da rede a especificação de tráfego (*Traffic Specification* (TSPEC)): uma estimativa do tráfico que irão gerar. Os elementos da rede alocam recursos necessárias a manutenção deste tráfego. Um controle de admissão deve ser aplicado na requisição do serviço. Este controle pode envolver a verificação de recursos de banda no enlace, espaço em *buffers* internos, e até mesmo capacidade de processamento para o encaminhamento de pacotes.

O serviço garantido do *IntServ* (WROCLAWSKI, 1997) faz com que seja possível fornecer um serviço que garanta tanto o limite de atraso como a largura de banda para um fluxo singular. Este serviço objetiva atingir aplicações que necessitam garantir rigidamente que os pacotes de um fluxo não serão atrasados por um limite superior de tempo, após terem sidos transmitidos pela fonte. A invocação deste serviço deve ser acompanhada da especificação do fluxo, TSPEC, e dos requisitos de QoS do mesmo, *Rate Specification* (RSPEC).

Como mencionado anteriormente, *IntServ* depende do início e da manutenção da reserva de fluxo por estados, o que, obviamente, limita a sua escalabilidade e sua implantação no núcleo da Internet. Redes do núcleo da estrutura, que têm de lidar com milhões de fluxos simultaneamente, não podem tolerar as limitações de escalabilidade *IntServ*. Entretanto, o modelo *IntServ* puro pode ser aplicado em domínios *stub*[2], nas pontas da internet, ou mesmo, em redes IP dedicadas.

2.4 Modelo DiffServ

O *DiffServ* (*Differentiated Services*) (BLAKE et al., 1998), ou serviços diferenciados, é uma proposta da IETF para lidar com o problema da escalabilidade apresentado pelo *IntServ*. O *DiffServ* controla a agregação de tráfego através do campo DS (*Differentiated Services Code Point* (DSCP)) do cabeçalho IP, sem a necessidade de manter o estado e a sinalização do fluxo em todo roteador.

O *DiffServ* foi projetado com o intuito de agregar fluxos individuais em classes. Na

[2]Uma rede stub é um segmento de rede com uma faixa de endereço, que é um subconjunto da faixa de endereço de outro segmento (a rede pai)

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|         DSCP          |  CU   |
+---+---+---+---+---+---+---+---+

  DSCP: differentiated services codepoint
  CU:   Atualmente não utilizado
```

Figura 2.2: Campo *DiffServ Code Point*

borda de um domínio *DiffServ*, o roteador irá receber fluxos marcados com sinalização de QoS, classificá-lo e condicioná-lo em classes para encaminhar ao destino, passando por dentro deste domínio. No núcleo do Domínio, os roteadores adaptados a trabalharem com o *DiffServ*, encaminham o fluxo conforme tratamento, prioridade e roteamento.

Os roteadores com entidades *DiffServ* fazem o tratamento do fluxo através do *Per Hop Behavior* (PHB), que processam as classes *DiffServ* conforme o comportamento pré-configurado. Os Fluxos dentro do domínio *DiffServ* são definidos em 3 classes básicas: melhor esforço (*Best Effort*), encaminhamento acelerado (*Expedited forwarding* (EF)) e encaminhamento assegurado (*Assured forwarding* (AF)).

A diferenciação e marcação das classes é marcada no campo DSCP do cabeçalho IP, conforme a figura 2.2.

O *DiffServ* aceita uma combinação de 64 (0 a 63) classes de serviço (IANA, 2008), sendo uma faixa com 32 valores de classe de DSCP citado pela RFC2434 (NARTEN; ALVESTRAND, 1998) e demais ítens para teste e implantação futura. O modelo *DiffServ* fornece maior escalabilidade devido a menor quantidade na alocação de recursos. No modelo *DiffServ* não é necessário efetuar uma reserva específica para cada fluxo. Os fluxos são agregados em classes de serviços de modo que não há controle de estado individual.

Diferente do *IntServ*, o *DiffServ* não tem um protocolo de sinalização específico e que comporte a arquitetura em classes. Contudo, neste trabalho será mostrado que o protocolo de sinalização NSIS pode se apresentar como uma alternativa para sinalização que pode ser utilizada em domínio *DiffServ*, permitindo um melhor aproveitamento de recursos nos roteadores.

2.5 Os protocolos de Sinalização e os Modelos de QoS

Um dos principais componentes das infra-estruturas em todas as redes de telecomunicações, desde a rede telefônica, redes de computadores, ou a internet e suas aplicações, é o seu sistema de sinalização. Basicamente um protocolo de sinalização permite identificar uma requisição específica entre entidades, sincronizar a troca de informações entre pontos distintos além de outras aplicações.

Em redes de telefonia, a sinalização é feita para identificar troncos, armazenar estado no caminho da chamada e carregar informações referente ao chamador, originador de uma requisição. Em redes de computadores, o protocolo de sinalização pode ter funcionalidades similares de reserva de recursos, enviando os dados do requisitante até o destino para um devido encaminhamento de um fluxo.

O protocolo de sinalização define as opções de serviço como a solicitação do início da comunicação, a confirmação do pedido, a configuração da transmissão de dados, a resposta ao envio de informações e a desconexão definindo também subprotocolos responsáveis por controles específicos. Um protocolo de sinalização para especificação de fluxo e provimento de QoS pode especificar entre outros parâmetros a codificação de um tráfego, a configuração de rota, o transporte de dados, segurança, cabeçalho, endereçamento.

O protocolo de sinalização pode ser implementado em modelos fim-a-fim, de um ponto de uma rede a outro, não levando em conta os componentes intermediários da rota, ou ainda, o modelo *hop-by-hop*, ou seja, os pacotes são enviados de roteador a roteador, até ao destino final mantendo o estado nos hosts no meio do caminho.

Um protocolo de sinalização gerencia os estados de fluxo. Estes estados são armazenados em roteadores. Para isto, utiliza-se de mensagens de sinalização para reserva de estado, desconexão de estado e mensagens de alerta, como erro. Um estado é o conhecimento do roteador de determinado fluxo para tal protocolo. Um roteador somente armazena o estado de protocolos por ele conhecido, atráves das mensagens de sinalização deste.

Duas grandes abordagens para a sinalização podem ser identificadas: *hard-state* e *soft-state*. O termo *soft-state* foi cunhado por Clark (CLARK., 1988), que descreveu o conceito de atualização periódica de estados definidos inicialmente por uma requisição. Este procedimento de *refresh* (atualização) é transparente para o sistema final e garante, por exemplo, que no caso de um dos pontos finais seja removido, os estados mantidos pelas entidades envolvidas ao longo de uma rota sejam automaticamente removidos.

No caso de *hard-state* o estado da sinalização é mantido nos nós. Desde que o estado é instalado, a menos que explicitamente removido, pode manter-se durante um tempo previamente definido. Um protocolo *hard-state* requer um mecanismo para a remoção de estados já sem comunicação. Neste caso, um mecanismo a ser utilizado é ter um mecanismo de tempo de vida (*timeout*) e/ou uma mensagem explícita solicitando a remoção do estado (*teardown*).

2.5.1 O Protocolo de Sinalização como Elemento de Interligação de Modelos de QoS

A sinalização de QoS surgiu inicialmente como um componente do modelo *IntServ*. Entretanto, a sua aplicação na interconexão de domínios com diferentes modelos de QoS vem se tornando uma realidade, por exemplo, no projeto europeu *End-to-end Quality of Service support over heterogeneous networks* (EuQoS)(MASIP-BRUIN et al., 2007). Na própria IETF (BERNET et al., 2000) é discutido o uso do protocolo RSVP na interconexão de domínios com diferentes modelos de QoS e, como será visto adiante, o protocolo NSIS, apresenta mecanismos interessantes para este fim.

No sistema EuQoS, por exemplo, a sinalização fim-a-fim é desviada na entrada de roteadores de ingresso dos domínios atravessados, de forma a visitar as entidades de gerenciamento de recursos dos domínios (GR). O GR realiza tarefas de controle de admissão e de sinalização adicional dentro do domínio e, então, encaminha a sinalização para o roteador de egresso do domínio. As mensagens de sinalização são, por sua vez, encaminhadas para o roteador de ingresso do próximo domínio, onde o processo se repete até atingir o ponto final da comunicação.

Em adição, os modelos *DiffServ* podem se beneficiar da sinalização para obter um melhor aproveitamento de recursos. No capítulo que se segue, o protocolo NSIS, tido como sucessor do RSVP, será apresentado com maiores detalhes, sendo mostrado os mecanismos que o tornam interessante para as tarefas acima, além do comportamento mas comumente conhecido que é a reserva por fluxo.

3 O protocolo NSIS

Conforme apresentado no capítulo anterior, os protocolos de sinalização são fundamentais para a operacionalização de um modelo de QoS. Com fins de criar um protocolo de sinalização para atender os requisitos das redes de próxima geração, a IETF criou o grupo de trabalho do NSIS (*Next Steps in Signaling*) que resultou em uma série de *Request For Comments* (RFC) e *drafts* até o presente momento.

3.1 Visão Geral do projeto NSIS

A proposta NSIS é composta de duas camadas (Fig.3.1): a camada inferior, de transporte, chamada GIST ou NTLP, e a camada superior, com várias opções de protocolos de sinalização de aplicações, camada esta conhecida como *NSIS Signaling Layer Protocol* (NSLP).

O protocolo NTLP visa proporcionar funções que são comuns a todos protocolos, como por exemplo, a manutenção de estados ao longo dos roteadores de um caminho, além de funções especiais como a capacidade de sinalizar a partir do transmissor e do receptor de um fluxo (*upstream* e *downstream*).

Os protocolos de sinalização da aplicação se aplicam a diferentes propósitos: gerenciamento de *firewalls*, reserva de recursos (QoS NSLP) e até mesmo a medição em redes.

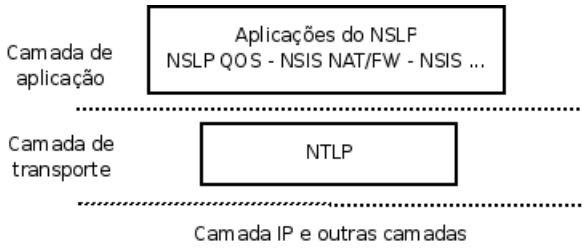


Figura 3.1: Modelo Camada NSIS

Na sequência serão apresentados o protocolo de transporte de sinalização GIST e,

com mais detalhes, o protocolo QoS NSLP, responsável pela sinalização de aplicação para QoS, e foco deste trabalho.

3.2 O Protocolo GIST

A sinalização da aplicação requer um conjunto de regras de gerenciamento de estado, bem como um protocolo de apoio para a troca de mensagens de dados ao longo do caminho. Vários aspectos deste protocolo de apoio são comuns a todos ou a um grande número de pedidos de sinalização da aplicação e, portanto, pode ser desenvolvido como um protocolo comum. O GIST (SCHULZRINNE, 2008) é o protocolo de transporte comum a todos os protocolos de sinalização, em nível de aplicação.

Importante ressaltar a *Application Programming Interface* (API) do GIST é uma interface entre a camada de transporte da arquitetura *Transmission Control Protocol over Internet Protocol* (TCP/IP) e a camada de sinalização de aplicação. As figuras 3.2 e 3.3 mostram isto.

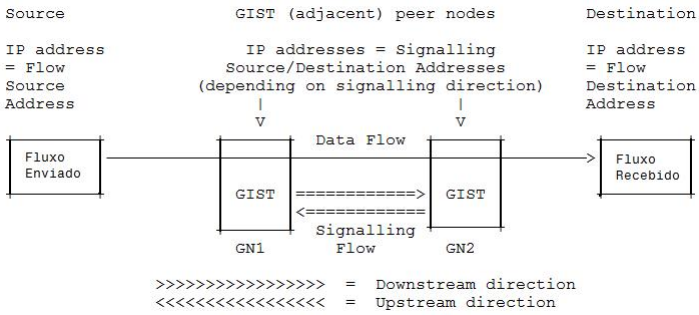


Figura 3.2: Modelo estrutura GIST *(draft-ietf-nsis-ntlp-19)*

O GIST é projetado, portanto, para suportar diversos tipos de aplicações de sinalização, pois sua funcionalidade é independente de uma aplicação específica. O comportamento do GIST, em funcionamento, é transparente para as camadas de transporte e aplicação. Este funcionamento se torna adequado para evitar interferências às mensagens de sinalização.

Uma das funções básicas do GIST é determinar o próximo nó a ser visitado pelo protocolo de sinalização. Para o caso de sinalização sobre os caminhos de dados, este próximo nó deve ser o nó visitado também pelo fluxo de dados. É interessante observar que mudança de rotas podem causar divergências temporárias entre o caminho de dados e o caminho da sinalização.

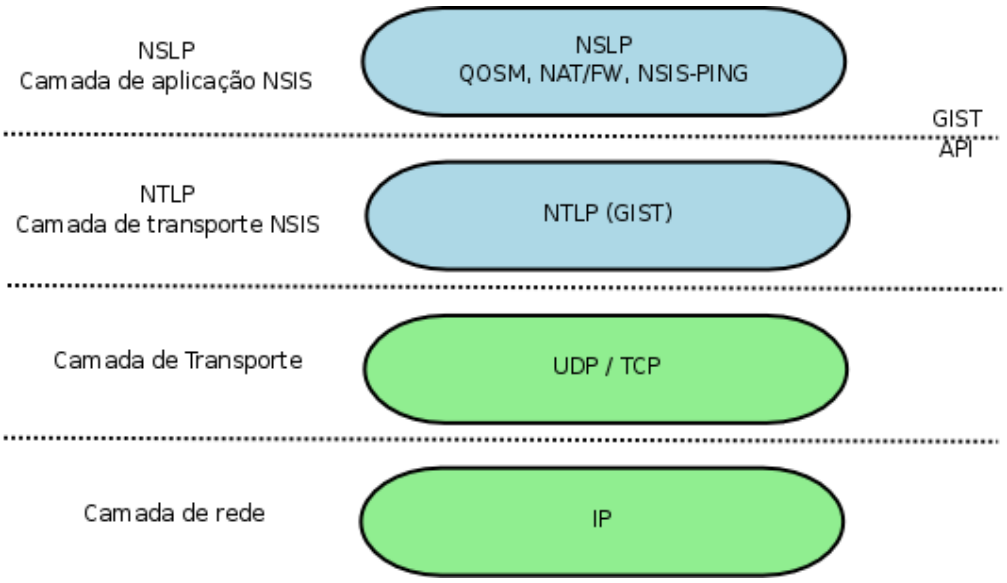


Figura 3.3: Modelo camadas e posicionamento do GIST

Quando o GIST recebe uma mensagem provinda da sinalização da aplicação, ele examina uma informação chamada *Message Routing Information* (MRI), fornecida junto com a mensagem. O MRI identifica, na sua forma mais simples, os pontos iniciais e finais do fluxo. A mensagem, também informa se a sinalização deve ser realizada em direção ou contra o fluxo. Uma vez determinado a direção e o destino, o GIST tenta observar se já existe uma associação entre o próximo nó e o nó atual. Esta associação, inerente ao modo *C-Mode*, modo de associação do nó GIST adjacente, é basicamente uma conexão *Transmission Control Protocol* (TCP) entre o nó atual e o próximo nó na rota da sinalização. Se já existe uma associação então ela pode ser utilizada para o transporte até o próximo nó. Note que o próximo nó pode variar de acordo com o protocolo de sinalização da aplicação.

No caso de não existir uma associação, então o GIST procede uma operação de descoberta do próximo nó. Usando um pacote sonda UDP e usando a facilidade de *Router Alert* da camada IP, o GIST envia um pacote QUERY na direção do destino final do fluxo, respeitando se a sinalização é contra ou a favor do fluxo. Cada roteador ao longo do caminho, processa o pacote, isto devido ao *Router Alert* setado. O pacote é repassado para o GIST local que verifica se existe um processo de sinalização de aplicação associado a mensagem (*Identification NSIS Signaling Layer Protocol* (NSLPID), identificador do NSLP). Se existir, a entidade GIST responde a entidade GIST par que enviou o pacote sonda através de uma mensagem RESPONSE. Neste momento, uma associação pode ser estabelecida. O processo de relacionar uma associação (no caso do modo *C-mode*) com a dupla (NSLPID,MRI) é o chamado armazenamento de estado no roteador. Na rea-

```
+-------------------+-----------+
+ {MRI,SID,NSLPID}  | U ||  MA1 |
+-------------------------------+
```

Figura 3.4: Modelo Mensagem GIST

lidade, no NSIS existe também um identificador de sessão. Um ou mais fluxos podem estar associados a uma sessão e, de fato, a identificação de um estado se dá pela tríade (NSLPID,*Session Identification* (SID),MRI), que define respectivamente, a qual protocolo NSLP, a qual sessão e a que fluxo, uma mensagem de sinalização de aplicação se refere. O estado se apresenta então da forma da figura 3.4.Nesta figura, a associação MA1 (associação 1) é identificada através da da tríade (NSLPID,SID,MRI). Além disto, a direção do fluxo é identificado por U (*upstream*).

Existem alternativas às associações com TCP, usando o protocolo UDP, mas não será discutido neste trabalho. Também existem alternativas ao uso do *Router Alert* no processo de descoberta. Isto porque esta opção pode causar problemas de segurança.

O GIST, em si, é um protocolo ponto a ponto. Isto significa que ao longo de um caminho, um nó GIST só pode receber dados enviados pelos seus pares adjacentes, ainda que estes dados não sejam destinados a ele. Quando o GIST recebe uma mensagem de uma sinalização que não for destinado a ele, o GIST simplesmente encaminha a mensagem para o seu próximo nó adjacente, que irá entregá-lo para a camada GIST adequada. O transporte de uma mensagem de sinalização NSIS depende, portanto, da capacidade de um nó de encontrar ou localizar corretamente seu próximo nó adjacente.

3.2.1 O problema de mudança de rotas

O rearranjo topológico ou a mudança de rota pode impactar seriamente os estados armazenados pelo GIST. Isto porque novos nós vizinhos podem aparecer, por exemplo, mais próximos do que aqueles utilizados em uma determinada associação.

O GIST prevê alguns mecanismos para contornar este problema. Um deles, é o envio periódico de pacotes sondas para encontrar os nós vizinhos e refazer associações que estejam desatualizadas. Caso isto não ocorra, podem existir situações inusitadas onde um nó aparentemente vizinho continua ser utilizado, produzindo efeitos indesejáveis para os protocolos NSLPs.

3.3 O protocolo QoS NSLP

O QoS NSLP (NSLP - *NSIS Signaling Layer Protocol*) é um protocolo de QoS da camada de aplicação que atua na camada acima do GIST (figura 3.2). Este protocolo estabelece e mantém estados nos nós, ao longo do caminho de um fluxo de dados, com o objetivo de fornecer informações para a manutenção de recursos para esse fluxo na rede. O seu modelo de funcionamento consiste basicamente em receber os pacotes de sinalização provindos do GIST, processar esta sinalização e repassar ao gerenciador de recursos (ou devolver a sinalização ao GIST) para enviar a um outro nó. Esta sinalização tem um caráter fim-a-fim, ou seja, a sinalização é conduzida desde um ponto inicial até um ponto final, passando por roteadores intermediários.

Dentre as funcionalidades deste protocolo, podemos citar (MANNER G. KARAGIANNIS, 2008):

- Comportamento com *soft-state*: uso de mensagens de atualização de reserva periódicas. Caso nenhuma seja recebida dentro de um intervalo pré-estabelecido a reserva é removida;
- Sinalização de fluxo unidirecional e bidirecional, ou seja, suporte a mensagens de sinalização para fluxo em um só sentido e para dois fluxos em sentido contrários;
- Capacidade de trabalhar em domínios *DiffServ* usando recursos de armazenamento mínimo de estados no núcleo dos domínios;
- Combinação de várias mensagens de reserva simultânea (*Binding*);
- Capacidade de detecção de mudança de rota automaticamente, mantendo o estado dos fluxos da rota anterior para a nova rota.

O QoS NSLP, invoca o GIST para efetuar tarefas de envio e recebimento de mensagens através da API do GIST. O NSLP está na camada de aplicação, cuja arquitetura pode ser mostrada na figura 3.5.

Onde

- *QoS NSIS Entity* (QNE): nó intermediário de apoio ao QoS NSLP;
- *QoS NSIS Initiator* (QNI): o primeiro nó de uma sequência de QNEs que envia uma requisição de reserva para uma sessão;

```
                          QoS NSLP nodes
IP address                (QoS unaware NSIS nodes are          IP address
= Flow                     not shown)                          = Flow
Source                      |          |           |          Destination
Address                     |          |           |           Address
                            V          V           V
+--------+  Data +------+      +------+       +------+     +--------+
|  Flow  |-------|------|------|------|-------|------|---->|  Flow  |
| Sender |  Flow |      |      |      |       |      |     |Receiver|
+--------+       | QNI  |      | QNE  |       | QNR  |     +--------+
                 |      |      |      |       |      |
                 +------+      +------+       +------+
                        ====================>
                        <====================
                              Signaling
                                Flow
```

Figura 3.5: Estrutura do NSLP (*draft-ietf-nsis-qos-nslp-16*)

- *QoS NSIS Receiver* (QNR): o último nó de uma sequência de QNEs que recebe uma requisição de reserva para uma sessão.

As mensagens recebidas e capturadas pelo NSLP na entrada do nó são manipuladas pelo GIST. Apenas as mensagens relacionadas com QoS são passadas para o QoS NSLP. O GIST também pode disparar eventos ao QoS NSLP, como por exemplo, indícios de mudança de rota. Neste último caso é feita nova sinalização, para a nova rota, para manter a garantia dos recursos anteriormente estabelecidos.

O protocolo QoS NSLP trabalha utilizando poucas mensagens de comunicação, sendo basicamente a mensagem de reserva. Mensagens QoS NSLP são enviadas a cada nó da aplicação. Isto significa que um QNE considera seus pares adjacentes como fonte de cada mensagem.

Tipos de mensagens

O protocolo QoS NSLP (MANNER G. KARAGIANNIS, 2008) utiliza basicamente quatro tipos de mensagens:

- RESERVE: A mensagem de reserva é a única mensagem que manipula o estado da reserva do QoS NSLP. É utilizada para criar, atualizar, modificar ou ainda eliminar o estado na entidade;

- QUERY: Uma mensagem QUERY é utilizado para solicitar informações sobre o caminho de dados sem fazer uma reserva. As informações obtidas a partir de uma consulta pode ser utilizada no controle de admissão de um processo QNE (por

exemplo, no caso de medição baseada em admissão de controle). Esta mensagem não altera as reservas de estado;

- RESPONSE: A mensagem RESPONSE é utilizada para fornecer informações sobre o resultado de uma mensagem QoS NSLP recebida anteriormente. Isso inclui a confirmação explícita a uma mensagem de reserva, a resposta a uma mensagem QUERY ou uma mensagem de erro se o código QNE ou QNR for incapaz de fornecer as informações solicitadas ou se a resposta é negativa. A mensagem RESPONSE não pode realizar qualquer reserva de estado ou modificação de estado;

- NOTIFY: são utilizadas para transmitir informações a um QNE. Difere da mensagem RESPONSE na medida que as mensagens são enviadas de modo assíncrono e não precisa de um determinado estado ou mensagem recebida anteriormente. A informação de uma mensagem NOTIFY é normalmente relacionada às condições de erro.

De forma geral, estas mensagens podem conter três tipos de objetos:

1. *Control Information* (Informações de Controle): transporta informação para o QoS NSLP, tais como números de sequência, ou quando uma reposta é necessária;

2. *QoS specifications* (QSPECs): O objeto *QoS Specifications* (QSPEC) (ASH A. BADER, 2008) transporta informação sobre os recursos necessários, os recursos disponíveis, e outras informações exigidas pelo gerenciador de QoS do domínio da rede;

3. *Objetos de Política*: Contêm dados utilizados para autorizar a reserva de recursos.

3.3.1 Associações de Mensagens

O protocolo QoS NSLP oferece suporte a associação de mensagens, a fim de permitir expressar dependências entre diferentes mensagens. A associação de mensagem pode indicar dependência unidirecional ou bidirecional entre duas mensagens. Uma das mensagens da associação deve possuir um objeto de identificação MSG_ID *Mensage Identifier* (Identificador de mensagem) e a outra mensagem o objeto BOUND_MSG_ID *Bound Mensage Identifier* (identificador de salto/fronteira).

A dependência unidirecional significa que apenas mensagens de RESERVE estão vinculados um ao outro enquanto um dependência bidirecional significa que existe dependência também para mensagens de RESPONSE. A associação de mensagem pode ser utilizado

```
QNE/QNI'     QNE'    QNR'/QNE      QNR
  +-------------------->|          |
  |  RESERVE |          |          |
  |          |          |          |
  | RESERVE' |          |          |
  |   (2)    |          |          |
  +========>| RESERVE' |          |
  |          |   (2)    |          |
  |          +========>| RESERVE  |
  |          |          |   (3)    |
  |          |          +--------->|
  |          | RESPONSE'|          |
  | RESPONSE'|<=========+          |
  |<=========+          |          |
  |          |          | RESPONSE |
  |          | RESPONSE |<---------+
  |<--------------------+          |
```

Figura 3.6: Modelo associação de Mensagens

para acelerar a sinalização, iniciando a sinalização de dois intercâmbios simultaneamente e que são sincronizados depois usando mensagens de identificações.

Por exemplo, a associação de mensagem pode ser utilizado para acelerar a sinalização, onde, se necessário, algumas trocas de mensagens podem ser iniciadas de modo simultâneo e podem ser tratadas em paralelo até serem tratadas como uma sessão única. Isto é uma importante função do NSIS, que permite o tratamento em domínio por agregado, como pode ser visto na figura 3.6.

3.3.2 Associação de Sessões e Reservas de Agregados

Uma associação de sessão é definida como a relação entre diferentes sessões de QoS NSLP (ou seja, sinalizando fluxos com diferentes SESSION_ID *Session Identifier*), conforme definido no GIST. A associação de sessão indica uma relação entre duas ou mais sessões incluindo um objeto BOUND_SESSION_ID. Uma sessão com um identificador SID_A (associação da sessão) pode expressar sua relação unidirecional de dependência a outra sessão com SID_B por BOUND_SESSION_ID incluindo um objeto contendo SID_B nas suas mensagens.

Uma mensagem pode incluir mais de um objeto BOUND_SESSION_ID. Isto pode acontecer, por exemplo, em caso de agregação e reserva bi-direcional, onde uma sessão fim-

a-fim tem uma relação unidirecional de dependência com um agregado e, ao mesmo tempo, este objeto tem uma relação unidirecional de dependência com outra sessão utilizada para o caminho inverso.

Em alguns casos, por exemplo, em domínios *DiffServ*, é desejável a criação de reservas para um agregado, em vez de reserva por um fluxo, a fim de reduzir o montante das reservas nos roteadores, bem como a redução do processamento de mensagens de sinalização. Contudo, o QoS NSLP não especifica quantas reservas devem ser combinadas em um agregado, mas fornece sinalização para tal.

A diferença fundamental nesta abordagem é que a reserva precisa de duas seções diferentes, uma que descreve todo o tráfego transportado no agregado (por exemplo, um DSCP no caso de *DiffServ*) e a outra por fluxo simples, em cada roteador do caminho.

3.3.3 Funcionamento com Estado Reduzido ou sem Estado

Este exemplo usa um modelo diferente de sinalização dentro de um domínio. Esta funcionalidade evita o armazenamento do estado nos nós do interior do domínio. Como resultado, os nós do interior só armazenam o QSPEC relacionados com reservas direcionadas.

A sinalização neste modelo permite que mensagens sinalizem de ponta a ponta em um domínio, de modo transparente, enquanto a sinalização sem estado passa por todos os nós, do domínio, anunciando o fluxo necessário. Ao final, as mensagens são centralizadas e a resposta é enviada, cujo modelo pode ser melhor observado na figura 3.7.

3.3.4 Re-roteamento

O QoS NSLP se adapta a mudanças na rota do caminho de dados. Isto pressupõe a capacidade de detectar eventos de reencaminhamento, criar uma reserva de QoS sobre o novo caminho e, opcionalmente, derrubar as reservas existentes sobre o antigo caminho.

Para a camada de aplicação NSLP, o reencaminhamento pode ser realizado de duas maneiras. Pode vir através de uma notificação de mudança de rota do GIST, ou a partir de informações vistas através de mensagens do NSLP. Neste último caso, o nó QoS NSLP é capaz de detectar mudanças nos nós adjacentes através do *Source Identification Information* (SII). Quando uma mensagem com uma reserva existente SESSION_ID e um diferente SII é recebida, o QNE conhece que seu nó *upstream* ou *downstream* foi alterado.

```
        QNE             QNE             QNE            QNE
      Entrada        interior        interior         Saída
         |               |               |              |
 RESERVE |               |               |              |
-------->| RESERVE       |               |              |
         +------------------------------------------------>|
         | RESERVE'      |               |              |
         +-------------->|               |              |
         |               | RESERVE'      |              |
         |               +-------------->|              |
         |               |               | RESERVE'     |
         |               |               +------------->|
         |               |               |  RESPONSE'   |
         |<-----------------------------------------------+
         |               |               |              | RESERVE
         |               |               |              +-------->
         |               |               |              | RESPONSE
         |               |               |              |<--------
         |               |               |     RESPONSE |
         |<-----------------------------------------------+
 RESPONSE|               |               |              |
<--------|               |               |              |
```

Figura 3.7: Modelo de mensagem em estado reduzido ou sem estado

A nova reserva, sobre o novo caminho, acontece quando uma mensagem RESERVE chega ao QNE onde a antiga e a nova rota são divergentes. Se a QoS NSLP suspeita que tenha ocorrido mudança de rota, em seguida, uma mensagem de RESERVE com o QSPEC é enviado. Aliado a isto, a mensagem de atualização de estado será identificada com um erro pelo QNE que não tem a reserva instalada, no novo caminho.

A rápida recuperação na camada NSLP, portanto, requer atualização em períodos curtos e detecção antes que a próxima mensagem de RESERVE chegue. Esta informação, no entanto, só é possível de se obter na camada IP ou através do acompanhamento dos estados do GIST, isto através da relação entre o GIST e o NSLP.

Após a reserva sobre o novo caminho, o nó pode querer derrubar a reserva sobre o antigo caminho antes do tempo normal do protocolo *soft-state* ou do tempo de expiração da mensagem. Esta funcionalidade é suportada pelo acompanhamento dos antigos SII ao longo do caminho na API do GIST.

Se o antigo caminho é no sentido do fluxo (*downstream*), o QNE pode enviar um mensagem de RESERVE usando o antigo *SII-Handle*. Se o antigo caminho é no sentido *upstream*, o QNE pode enviar um NOTIFY com o código para alteração de rota *Route Change*. Este é transmitido até que encontre um nó QNE que possa emitir uma mensagem RESERVE para o fluxo correspondente.

3.3.5 Interações com o Gist

A QoS NSLP usa o GIST frequentemente para a entrega de suas mensagens. As mensagens são transmitidas a partir do NSLP para o GIST através de uma API, conforme figura 3.3, que identifica as informações como sessão de fluxo, direção, IP e porta. Na recepção, o GIST fornece as mesmas informações para a QoS NSLP.

A QoS NSLP mantém as mensagens e seus estados por sessão. A sessão é identificado por um SESSION_ID. O SESSION_ID é o principal índice de NSLP para armazenar o estado e deve ser único ao longo de um caminho da rede. Este valor é posteriormente utilizado pelo GIST e as aplicações do NSLP para se referir a uma sessão.

O GIST deve suportar as mensagens e operações do NSLP, sem interferir em seu processamento. Ao receber uma mensagem o GIST só envia esta para a camada de aplicação se for direcionado para o nó em questão, caso contrário, o GIST pode encaminhar para o próximo nó.

```
\+ Object Type       : InfoSpec
\++ InfoSpec         : 0x01 0x02 0x00 0x00
\+ Object Type       : Qspec
\++ V:Id:M_Seq:OC:l  : 0:0:0:3:20
\+++ QSPEC object    : QoS Desired
\+++ Object type     : Tmod1, length:4
\+++ Object type     : ExcessTreatment, length:1
\+++ QSPEC object    : QoS Available
\+++ Object type     : Tmod1, length:4
\+++ QSPEC object    : Minimum QoS
\+++ Object type     : Tmod1, length:4
(...)
\+++ QSPEC object    : QoS Model (QOSM)
(...)
\+++ QSPEC object    : QoS Reserved
```

Figura 3.8: Exemplo de sinalização com objetos QSPEC

3.3.6 Considerações sobre QSPEC

O QSPEC tem por objetivo prover uma linguagem comum para ser utilizada por gerenciadores de QoS podendo ser transportado por mensagens RESERVE, QUERY, RESPONSE e NOTIFY.

Um nó QNI inicia uma sinalização e adiciona um objeto QSPEC contendo os parâmetros desejados de QoS. Os parâmetros do QSPEC devem ser interpretados por todos os nós do interior do domínio QNE marcados para gerenciamento de recurso.

Sua estrutura é simples, e fornece uma descrição do tráfego para o qual os recursos são reservados. O cabeçalho do QSPEC pode ser visto na mensagem de sinalização do NSIS, figura 3.8, conforme fragmento do retorno do protocolo, retirado dos cenários montados:

Pode-se descrever os objetos de QSPEC e sua estrutura conforme mostrado na figura 3.9:

- QSPEC: QSPEC é o objeto de QoS NSLP contendo todas as informações específicas de QoS.
- QoS *Desired*: Objeto QSPEC que contém parâmetros descrevendo os parâmetros de QoS desejado para os quais o QNI recebeu tráfego.
- QoS *Available*: Objeto QSPEC que contém parâmetros descrevendo os recursos disponíveis. Eles são usados para recolher informação junto de uma reserva.

```
+---------------------------------------+
|            QSPEC Objects              |
+---------------------------------------+

\_________________ _______________________/
                  V
+----------+----------+---------+-------+
|QoS Desir.|QoS Avail.|QoS Rsrv.|Min QoS|
+----------+----------+---------+-------+

\____ ____/\___ _____/\___ ____/\__ ___/
     V         V          V        V

+-------------+...     +-------------+...
|QSPEC Para. 1|        |QSPEC Para. n|
+-------------+...     +-------------+...
```

Figura 3.9: Objetos da estrutura QSPEC

- QoS *Reserved*: Objeto QSPEC que contém parâmetros descrevendo os recursos reservados e os respectivos parâmetros de QoS.
- QoS *Model* (QOSM): Um método com o modelo de QoS desejado pelos domínios.
- QSPEC *parameter*: Parâmetros utilizados em uma especificação, como latência, modelo de tráfego (TMOD) e tratamento.
- QSPEC *Object*: Principais parâmetros de um QSPEC estabelecido tanto na entrada como na saída de um gerenciador de QoS.
- QSPEC *Type*: Identifica um determinado QOSM utilizado no QSPEC.

3.4 Outros Protocolos NSLP

Na seção anterior foi visto com mais detalhes o protocolo de sinalização, no nível de aplicação, QoS NSLP, voltado a sinalização de qualidade de serviço. Entretanto, já existem propostas na IETF de outros protocolos NSLP voltados ao controle de firewalls e de medição em redes.

O NSIS natfw é um exemplo de cliente NAT/*Firewall* NSLP cuja finalidade é limitada principalmente ao desenvolvimento de testes. A API destina-se a ser incluído em aplicações para falar diretamente com NAT/*Firewall* NSLP para configurar dispositivos

NAT ou *firewall* ao longo de um caminho. Ele pode ser acionado através do GIST na camada de transporte.

O NAT/*Firewall* NSLP pode ser acionado para enviar mensagens de vários tipos, dependendo do modelo escolhido, uma série de tipos específicos de opções pode ser definido, como *proxy*, simulação de processos, bloqueio ou permissão de fluxos de pacotes e regras de policiamento.

O NSIS apresenta também um modelo de gerenciamento por fluxo, o *Resource Management Function* (RMF) que gerencia o nó alocando recursos conforme sinalizado. O RMF utiliza ferramentas do linux para controle de fluxo, como o *tc* e o *iptables*, o que o torna simples e extremamente eficiente.

O RMF trabalha em separado do NSIS, contudo o RMF recebe as informações do daemon NSIS que verifica a existência de especificação do fluxo. Este especificação será lida somente pelo RMF, o nsis trata somente da sinalização.

4 Cenários

4.1 Cenários estudados

Após os estudos bibliográficos sobre o NSIS e com base nos documentos lidos, foram definidos alguns cenários de testes com fins de aprofundar o estudo dos mecanismos integrantes do referido protocolo, bem como os seus modos de operação.

Os seguintes cenários foram definidos para este trabalho:

1. Cenário 1: Sinalização por fluxo individual, com mensagem de reserva a partir do emissor do fluxo;
2. Cenário 2: Sinalização por fluxo individual, com mensagem de reserva a partir do receptor;
3. Cenário 3: Caso simples de sinalização com reserva real de recursos a partir do emissor do fluxo;
4. Cenário 4: Caso detalhado do mecanismo de reserva e de gerência de recursos usando várias reservas;
5. Cenário 5: Atualização de reserva face a mudanças de rota e o mecanismo de *soft-state*;

4.2 Aspectos da Plataforma de Testes e da Operação do protocolo de Goettingen

4.2.1 Plataforma de Testes

Para a análise do protocolo foram utilizadas máquinas virtuais *linux* UML. Uma máquina virtual é um software que simula um computador, possibilitando a instalação

de sistemas operacionais e programas. Estas máquinas facilitam o controle das diversas entidades dos protocolos e evitam a complexidade adicional no uso de diversas máquinas reais.

Um sistema de arquivos foi construído para o uso nas máquinas virtuais UML, conforme descrito no apêndice A, onde foi instalada a versão 0.6.0 do protocolo NSIS implementado na Universidade de Goettingen. Esta implementação foi a escolhida, dada a forma estruturada do projeto e das constantes atualizações a que vem sendo submetida.

Dentre os componentes do NSIS implementado pela universidade de Goettingen cita-se o GIST (NTLP) e os protocolos de aplicação: NSIS-QoSD (Servidor de gerenciamento de QoS) e o NAT/FW NSLP para testes e gerenciamento com *firewalls*. Além disto, um cliente do NSIS-QoS permite dar início as sinalizações, segundo várias opções possíveis.

Em adição, nos cenários construídos foram utilizadas ferramentas para análise e geração de tráfego, tais como o *tcpdump* e o *iperf*. No último cenário, para auxiliar no roteamento, foi instalado e configurado o protocolo de roteamento *rip* através do software *ZEBRA*[1].

4.2.2 Estrutura e Configuração do NSIS

Para o seu funcionamento, a implementação de Goettingen do NSIS requer que sua camada de transporte (GIST) e os protocolos de sinalização (NSLPs) sejam executados como processos independentes. Cada máquina, que se comporta como uma entidade NSIS, deve executar no mínimo o protocolo GIST e um protocolo NSLP.

O processo GIST possui uma API, baseada em soquetes, para acesso as funcionalidades de transporte. Os demais protocolos NSLP acessam a API do GIST para o transporte de suas mensagens. Cada protocolo NSLP é implementado como um processo. O protocolo QoS NSLP, foco de estudo deste trabalho, possui também uma interface via soquetes, para permitir o acesso as suas funcionalidades por parte de um programa cliente.

Dentro do diretório do NSIS, encontra-se o diretório *bin* com arquivos compilados do NSLP, entre eles os arquivos *nsis*, *nsis-qos* e o *nsis-ping*, além do arquivo de configuração *nsis.conf*.

O binário *nsis* é um dos principais arquivos, que funciona como um *daemon* de controle central: quando executado carrega as informações do arquivo de configuração *nsis.conf* e

[1]Software que implementa e gerencia protocolos e rotinas de roteamento

executa os demais componentes do protocolo. No caso deste estudo, os protocolos GIST e o protocolo QoS NSLP.

No arquivo de configuração *nsis.conf* é possível habilitar os *daemons* que se deseja iniciar junto com a execução do nsis, além de configuração de IP, NAT/FW e parâmetros da Função de Gerência de Recursos-RMF. Para os cenários citados, no arquivo *nsis.conf* de todas as entidades iniciadas, foi habilitado o parâmetro *nslp.startQoS = yes*, que faz com que o *daemon nsis-qosd* seja inicializado junto com o *nsis*.

Um importante detalhe da configuração do arquivo *nsis.conf* está na configuração de roteamento de rede através do parâmetro *readRoutingTable* que permite ao NSIS ler a tabela de roteamento do sistema ou carregar as configurações do arquivo, por interface.

A configuração utilizada na montagem do cenário, bem como a configuração resumida do arquivo *nsis.conf* de cada ponto está listada no respectivo ambiente de teste. Os detalhes destes arquivos pode ser vistos no apêndice B.

4.2.3 O cliente do protocolo QoS-NSLP

Para iniciar uma sinalização e efetuar uma reserva, o cliente do protocolo QoS NSLP, *nsis-qos*, é executado. Este cliente se comunica com o *daemon* do *nsis-qos*, iniciado nos nós que terão passagem do fluxo.

Para iniciar uma sinalização, associa-se a chamada do cliente *nsis-qos* com alguns parâmetros na linha de comando, conforme o modelo de mensagem a ser utilizado. Dentre os comandos, pode-se citar:

- -res: Envia uma mensagem de reserva para uma sessão;
- -rt: Envia uma reserva como *flag* T (*Teardown Session*) para remover uma sessão;
- -q: Envia uma mensagem de prova, sem manter estado nos nós;
- -rir: Faz a sinalização de reserva a partir do nó de destino.

Utilizando o modo de análise do protocolo NSIS de Goettingen, foram feitas as sinalizações e com base nas informações apresentadas pela depuração do protocolo e descrição no texto deste trabalho foram feitos os devidos comentários.

4.3 Cenário 1

O modelo básico de sinalização do cliente NSIS QoS é a sinalização por fluxo individual com reserva a partir do emissor do fluxo, ou seja, a sinalização de reserva (mensagem RESERVE) é feita no mesmo sentido do envio do fluxo. Esta sinalização também é chamada de sinalização no sentido sentido *downstream*.

Este cenário, conforme figura 4.1, é composto dos nós QNI, QNE e QNR, sendo que o nó QNI, neste caso, é o emissor do fluxo de dados. Não foi realizada a configuração de reserva de recursos, ou seja, as mensagens de reserva serão geradas normalmente, mas não existe função de gerenciamento de recursos nos nós. Para isto, é feita a marcação no arquivo *nsis.conf* na opção qos.rmf com NullRMF: `qos.rmf = NullRMF`. Esta restrição, permite mostrar somente a sinalização.

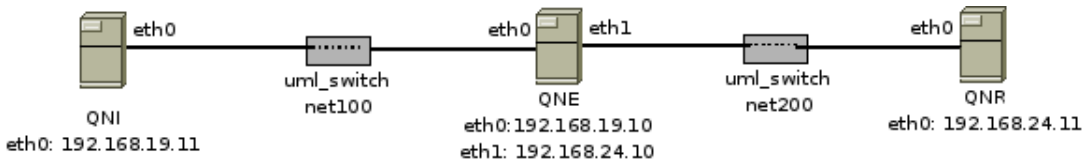


Figura 4.1: Cenário 1

Utilizou-se de um ambiente com as três unidade da rede, um nó QNE no meio, e nas pontas os nós QNI e QNR, cujas informações de endereço segue abaixo. Os detalhes da configuração podem ser verificados no anexo B.

- QNI IP: eth0=192.168.19.11/24; NSIS com NullRMF;
- QNE IP: eth0=192.168.19.10/24 IP: eth1=192.168.24.10/24 NSIS com NullRMF;
- QNR IP: eth0=192.168.24.11/24; NSIS como NullRMF;

Considerando o *daemon* nsis rodando em todos os nós, a partir do nó QNI, se efetua a sinalização com o comando:

```
./nsis-qos 192.168.24.11 -res
```

Neste caso aplicou-se o IP do nó de destino e a variável *-res* para sinalização de reserva. Note que neste cenário não é efetuada reserva, e a marcação de não efetuar reserva no nó QNE é respeitada.

Após a inicialização do comando, é gerado um fluxo de informações, obtidas utilizando o modo depurador do protocolo. O modo depurador é configurado através do arquivo

nsis.conf, no parâmetro *qos.debuglevel*, que recebe classificação de detalhes em ordem crescente de 0 à 7. Na sequência é discutida a saída de depuração em cada nó.

1. O nó QNI envia um mensagem de sinalização para frente e marca o tempo que aguarda por resposta (*RespTimer started* : 2s) e o tempo que ocorrerá a mensagem de atualização de reserva (*refresh*).

```
NullRMF: client connected
We are End Node (QNI)
SID: ffffffff ffffffff ffffffff ffffffff
QoSFsm::idle__rx_rmf_msg()
QoSFsm::inst__rx_rmf_msg()

QNI: outgoing msg -->
+ SII_Handle        :
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 120 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0
RespTimer started   : 2s
RefrTimer started   : 30s
```

2. O nó QNE recebe a mensagem de QNI e a encaminha para frente sem passar para a camada de aplicação (função RMF), pois não recebeu pedido de recurso. Aqui também permite que se verifique a marcação do tempo que ocorrerá a mensagem de atualização de reserva e o tempo de armazenamento de estado, por se tratar de um nó *statefull* (*StateTimer started* : 100s). O campo SII_Handle indica o endereço da interface a partir da qual a mensagem saiu.

```
We are Entity  (QNE)

QNE: --> incoming msg
+ SII_Handle        : 192.168.19.11
+ srcAddr           : 192.168.19.11
```

```
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 120 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0

(...)

QNE: outgoing msg -->
+ SII_Handle        :
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 120 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0
RefrTimer started   : 30s
StateTimer started  : 100s
```

3. O nó QNR recebe a mensagem.

```
! We are End Node (QNR) !

QNR: --> incoming msg
+ SII_Handle        : 192.168.24.10
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 120 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0
```

De modo sequencial é possível ver o conjunto de mensagens e a sequência na figura 4.2.

Esta sinalização se mantém até que o *State Timer*, tempo de esgotamento (*timeout*) que o estado fique armazenado, termine ou até que o nó QNE receba uma mensagem com o *flag* T marcado. Este último serve para identificar a remoção do estado (*teardown*).

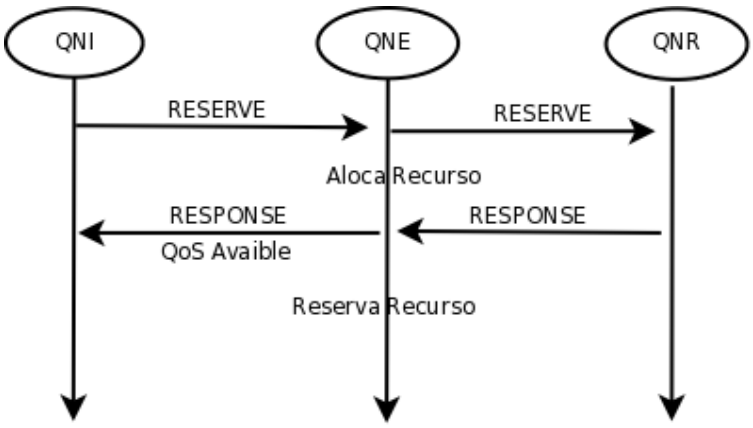


Figura 4.2: Sinalização de reserva

A partir do nó QNI, efetuou-se a solicitação de remoção de reserva com a mensagem `./nsis-qos 192.168.24.11 -rt`. Pode-se verificar o *flag* T marcado através da mensagem:

```
 QNI: outgoing msg -->
+ SII_Handle        :
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 24 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 1:0:0:0:0:0
```

Observe que outros meios de remoção do estado podem ocorrer, por exemplo, com a queda de um dos pontos do domínio. A mensagem transporta um tempo de vida, que é armazenado no nó e, caso dentro deste tempo não chegue uma atualização da reserva, a reserva é derrubada.

4.4 Cenário 2

O modelo anterior, com a reserva no sentido *downstream* é um modelo básico de sinalização, onde a sinalização é feita no sentido do fluxo. Contudo, o NSIS é capaz de gerar a reserva de fluxo a partir do nó destino, sentido *upstream*, para isto é feita a chamada do *nsis-qos* associada ao comando *-rir*.

Neste caso, utilizou-se o mesmo ambiente do cenário anterior, rearranjado conforme a figura 4.3 e com informações de endereço indicadas abaixo. Para visualizar detalhes da configuração, o leitor pode ser referenciar ao apêndice B.

- QNR IP: eth0=192.168.19.11/24; NSIS com NullRMF;
- QNE IP: eth0=192.168.19.10/24 IP: eth1=192.168.24.10/24 NSIS com NullRMF;
- QNI IP: eth0=192.168.24.11/24; NSIS como NullRMF;

Considerando o NSIS rodando em todos os nós, a partir do nó QNR, efetua-se a solicitação de verificação de recursos com o comando:

```
./nsis-qos 192.168.24.11 -rir
```

Onde *-rir* é o flag para a reserva a partir do receptor.

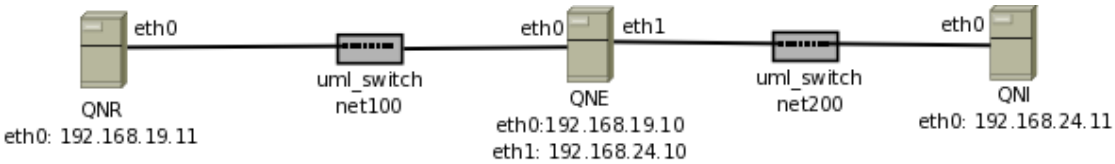


Figura 4.3: Cenário 2

Com o comando *-rir* (*Receiver Initiated Reservation*) ativo o nsis gera sinalização de reserva pelo lado oposto ao que o fluxo é recebido. Para fazer a reserva a partir do receptor do fluxo (*Receiver Initiated Reservation*), o QNR envia uma mensagem QUERY, que pode conter uma especificação de fluxo. A mensagem QUERY pode ser usado para coletar informações ao longo do caminho. Neste caso, será feita uma consulta de estado no caminho, para isto a mensagem deve ter o Flag R ativo e depois iniciado a reserva.

Analisando o tráfego a partir do nó QNE, que é o nó que fará a reserva, é possível verificar o fluxo de mensagens:

```
We are End Node (QNR)

QNR: outgoing msg -->
+ SII_Handle        :
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
```

```
+ msg length        : 68 bytes
+ QoS Message Type  : QUERY
+ T:R:B:A:P:S-Flag  : 0:1:0:0:0:0
```

O nó QNE recebe a sinalização de QUERY advinda do nó QNR (srcAddr=192.168.19.11) para o nó QNI (destAddr=192.168.24.11). A função `idle__rx_query()` marca o recebimento de uma query pelo GIST e a passagem para a máquina de estado, que estará esperando por uma resposta. Neste caso, o NSIS aciona a função `wr__rx_rmf_msg()` onde uma mensagem de reserva tenha sido acionada, conforme a mensagem de saída do nó QNE.

```
We are Entity  (QNE)

QNE: --> incoming msg
+ SII_Handle        : 192.168.19.11
+ srcAddr           : 192.168.24.11
+ destAddr          : 192.168.19.11
+ direction         : UPSTREAM
+ msg length        : 68 bytes
+ QoS Message Type  : QUERY
+ T:R:B:A:P:S-Flag  : 0:1:0:0:0:0

triggerEvent        : ST_IDLE::EV_RX_QUERY

QoSFsm::idle__rx_query()
StateTimer started  : 30s
transition to       : ST_WR
QoSFsm::wr__rx_rmf_msg()
```

Neste momento, o nó QNE recebe uma mensagem de RESERVE advinda do nó QNI, contudo, repare na condição das informações do cabeçalho:

- SII_Handle: 192.168.24.11: Observa-se abaixo que realmente a fonte (src) é advinda do IP do nó QNI, contudo, o cabeçalho SII_Handle vem marcado pelo originador da sinalização. SII é na, tradução para o português, a informação que identifica a fonte (*Source Identification Information*);

- srcAddr: 192.168.24.11: Originador da mensagem;
- destAddr: 192.168.19.11: Destino da mensagem;

Com isto, a mensagem de reserva é enviada para o nó QNE, que efetuará a reserva, a marcação é realizada de maneira correta, no sentido do fluxo.

```
QNE: --> incoming msg
+ SII_Handle        : 192.168.24.11
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 80 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0
```

O fluxo de mensagens é mostrado na Fig.4.4

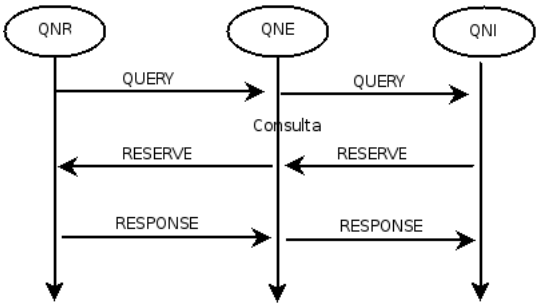


Figura 4.4: Fluxo de Sinalização de reserva com query

4.5 Cenário 3

Nos cenários anteriores, foi mostrado somente a sinalização, sem nenhuma alocação de recurso para o fluxo, isto porque o nó QNE estava marcado para não efetuar a reserva. Contudo, o NSIS permite, junto a sinalização, o envio da entidade QSPEC com a especificação de fluxo. Esta especificação será interpretada pela função de gerenciamento de recurso. Será realizada então uma sinalização por fluxo individual com reserva a partir do emissor do fluxo, ou seja, a sinalização é feita no mesmo sentido do envio do fluxo.

Este cenário, conforme figura 4.5, é composto dos nós QNI, QNE e QNR, com configuração de reserva de recursos para o nó central QNE. Para isto, altera-se no *nsis.conf*

o parâmetro *qos.rmf* para *SimpleRMF* (*qos.rmf = SimpleRMF*), conforme cenário apresentado:

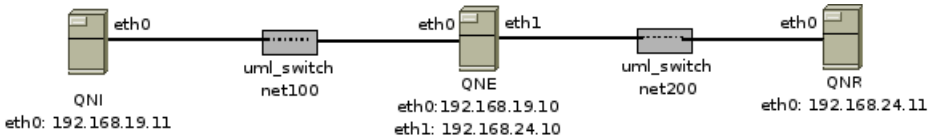


Figura 4.5: Cenário com reserva de recurso

Para este cenários, utilizou-se de um ambiente semelhante aos anteriores, com somente um nó QNE no *core* e este efetua a reserva, conforme abaixo. Caso queira visualizar detalhes da configuração, verifique o apêndice B.

- QNI IP: eth0=192.168.19.11/24; NSIS como NullRMF; Software iperf como cliente para fluxo priorizado;
- QNE IP: eth0=192.168.19.10/24 IP: eth1=192.168.24.10/24 NSIS como SimpleRMF, fazendo reserva;
- QNR IP: eth0=192.168.24.11/24; NSIS como NullRMF; Software iperf como servidor do fluxo priorizado;

4.5.1 Iniciando a função de RMF do NSIS

A função de reserva de recursos (RMF) do NSIS se utiliza de ferramentas do próprio *linux*, como o *iptables* e o *tc*. O *iptables* é uma ferramenta de filtro de pacotes baseado em regras pré determinadas. O *tc (traffic control)* é uma ferramenta de controle de tráfego e pode ser utilizada para gerenciamento de QoS em redes. O NSIS utiliza destas ferramentas como comandos, implementados na aplicação, requerendo que os programas *iptables* e *tc* estejam instalados.

O *tc* implementa o *Queueing Discipline* (QDISC) como algoritmo de classificação de tráfego na saída do fluxo. o QDISC é capaz de gerenciar o fluxo de uma interface, seja na entrada ou na saída do fluxo, criando associações de regras hierárquicas para a interface. O QDISC tem uma raiz (*root*) que é associada a uma interface.

No QDISC adicionam-se classes onde são implementadas as políticas de tratamento do fluxo. Dentro de uma classe podem haver outras classes além de outro qdisc, com outra disciplina de fluxo. Portanto, uma classe pode ter um QDISC como pai (*parent*)

ou uma outra classe. O QDISC precisa de um classificador para determinar qual a classe que precisa enviar um pacote. Isto é feito utilizando o *filter*.

Durante a inicialização do NSIS no nó QNE com a marcação para *SimpleRMF*, verifica-se o *tc* rodando, inicialmente apaga as classes que poderiam ter sido criadas e em seguida monta a estrutura de priorização:

```
(TC) Running "tc qdisc delete dev eth1 root handle 1:"
(IpTables) Running "iptables -t mangle -F PREROUTING"
(IpTables) Running "ip6tables -t mangle -F PREROUTING"
(TC) Running "tc qdisc add dev eth1 root handle 1: htb default 2"
(TC) Running "tc class add dev eth1 parent 1: classid 1:1 htb rate
 100kbit ceil 100kbit burst 100kbit"
(TC) Running "tc class add dev eth1 parent 1:1 classid 1:2 htb rate
 100kbit burst 100kbit ceil 0 mpu 0"
(TC) Running "tc qdisc add dev eth1 parent 1:2 handle 2: sfq perturb 10"
```

A estrutura criada pode ser visualizada conforme figura 4.6:

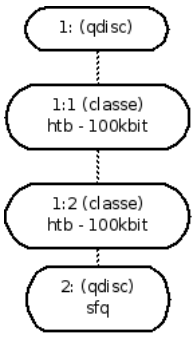


Figura 4.6: Criação da estrutura de controle de tráfego

Observa-se que é aplicado uma classe com taxa (*rate*) de 100kbit limitando o fluxo não priorizado, o que pode ser visto com o iperf:

```
(none):~# iperf -s -i 0
WARNING: interval too small, increasing from 0.00 to 0.5 seconds.
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

```
[  4] local 192.168.24.11 port 5001 connected with 192.168.19.11 port 36655
[  4]  0.0- 0.5 sec  9.90 KBytes    162 Kbits/sec
[  4]  0.5- 1.0 sec  4.24 KBytes  69.5 Kbits/sec
[  4]  1.0- 1.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  1.5- 2.0 sec  4.24 KBytes  69.5 Kbits/sec
[  4]  2.0- 2.5 sec  6.66 KBytes    86 Kbits/sec
[  4]  2.5- 3.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  3.0- 3.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  3.5- 4.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  4.0- 4.5 sec  4.24 KBytes  69.5 Kbits/sec
[  4]  4.5- 5.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  5.0- 5.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  5.5- 6.0 sec  5.66 KBytes  92.7 Kbits/sec
```

4.5.2 Efetuando a reserva para um fluxo

Considerando o nsis rodando em todos os nós, a partir do nó QNI, se efetua a solicitação de reserva de fluxo com o comando:

```
./nsis-qos 192.168.24.11 -res -bw 50 -p 6 -dpt 9876
```

Onde *-res* de reserva, *-bw* para especificar a quantidade de banda desejada em bytes por segundo, *-p 6* para utilizar protocolo TCP e *-dpt 9876* para especificar uma porta de destino para a reserva.

Neste momento é gerado um fluxo de informações, coletadas do modo *debug* do protocolo, que podem ser melhores descritas a partir do nó QNI.

1. O Nó QNI envia uma mensagem de reserva (RESERVE);
2. Após recebe uma mensagem NOTIFY confirmando que existe recurso: *RR supported.*
3. A partir da confirmação o nó QNI envia uma mensagem de resposta (RESPONSE) para informar o envio do fluxo.

De modo sequencial pode-se ver o conjunto de mensagens e a sequência na figura 4.7

- QNI sinaliza para QNR;

- QNE intermedia e verifica informações;
- QNE aloca recursos;

```
Reserving ...
(TC) Running "tc class add dev eth1 parent 1:1 classid 1:6 htb rate
 500bps burst 500 ceil 500bps mpu 64"
HTB: quantum of class 10006 is small. Consider r2q change.
(TC) Running "tc qdisc add dev eth1 parent 1:6 handle 6: sfq perturb 10"
(TC) Running "tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
 match ip
src 192.168.19.11/32 match ip dst 192.168.24.11/32 match ip protocol 6 0xff
match ip dport 9876 0xffff  flowid 1:6"
```

- As atualizações de máquina de estado são encaminhadas entre os nós `+++ QSPEC object : (`
  `+++ Object type : Tmod1, length:4` `+++ QSPEC object : Minimum QoS`
- QNR responde a sinalização
- QNE respode a QNI e reserva recursos.

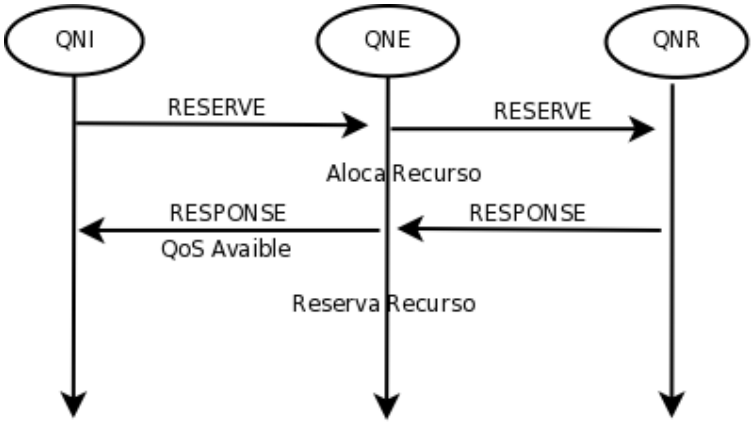


Figura 4.7: Sinalização de reserva

4.5.3 Análise da Saída do Depurador

Após a reserva de recursos, o estado da garantia de recursos se mantém por sinalização de estado, *soft-state*, gerada de tempos em tempos. Esta mensagem tem o intuito apenas de manter o estado e carrega informações de fonte e destino, ID da reserva e tipo de mensagem. O evento é gerado pelo nó QNI, que ao chegar ao nó QNE chama a função

`inst__rx_reserve()`, que compara o identificador da reserva com os atuais, se o valor for o mesmo que o armazenado, a reserva é atualizada. Um modelo pode ser visto abaixo:

```
----------------------------------------
QoSFsm: refresh_timeout()

triggerEvent        : ST_INST::EV_TIMEOUT_REFRESH

QoSFsm::inst__timeout_refresh()

QNE: outgoing msg -->
+ SII_Handle        :
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 16 bytes
+ QoS Message Type  : RESERVE
+ Object Type       : RSN
++ RSN              : 1
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0
RefrTimer restarted : 30s

transition to       : ST_INST

SID: ffffffff ffffffff ffffffff ffffffff
fsm                 : 0x80f4c28

QNE: --> incoming msg
+ SII_Handle        : 192.168.19.11
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 16 bytes
+ QoS Message Type  : RESERVE
+ Object Type       : RSN
++ RSN              : 1
```

```
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0

triggerEvent        : ST_INST::EV_RX_RESERVE

QoSFsm::inst__rx_reserve()
StateTimer restarted: 100s

transition to       : ST_INST
```

Para se remover uma reserva, basta utilizar o parâmetro *-t*, de derrubar (do inglês *teardown*). Este modelo ativa o *flag* T que associa um tempo para a reserva, se associado sem tempo determinado, remove a reserva. O procedimento de remoção de reserva também pode ser feito manualmente, como no exemplo:

```
./nsis-qos 192.168.24.11 -rt -bw 50 -p 6 -dpt 9876
```

Para o caso desta sinalização utilizou-se os mesmo parâmetros da reserva, contudo, o *flag* T foi enviado com o parâmetro *-rt*.

Outros meios são a queda de um dos nós de um domínio. Cada reserva tem associada um tempo de vida, que é o tempo que ficará o estado armazenado em um roteador. Caso dentro deste tempo não chegue atualização da reserva, esta é derrubada.

```
SID: ffffffff ffffffff ffffffff ffffffff
fsm                 : 0x80f4c28

QNE: --> incoming msg
+ SII_Handle        : 192.168.19.11
+ srcAddr           : 192.168.21.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 24 bytes
+ QoS Message Type  : RESERVE
+ T:R:B:A:P:S-Flag  : 1:0:0:0:0:0

triggerEvent        : ST_INST::EV_RX_RESERVE
```

```
QoSFsm::inst__rx_reserve()
StateTimer started  : 30s

transition to       : ST_WR
```

Novamente a mensagem é recebida, é feita a chamada função `inst__rx_reserve()`, da máquina de estado, verifica o marcador *T-FLAG* setado (campo T:R:B:A:P:S-Flag:), e libera a reserva.

```
Refresh Timer stoped
Releasing reservation ...
(TC) Running "tc filter delete dev eth1 protocol ip parent 1:0 prio 1 u32
 match ip src 192.168.19.11/32 match ip dst 192.168.24.11/32 match ip
 protocol 6 0xff match ip dport 9876 0xffff  flowid 1:6"
(TC) Running "tc class delete dev eth1 parent 1:1 classid 1:6"
QoSFsm::wr__rx_rmf_msg()
```

4.6 Cenário 4

No cenário anterior podê-se verificar o mecanismo de reserva de recurso RMF atuando. Foi efetuada sinalização com reserva e o mecanismo de gerência de recursos efetuou esta reserva. Contudo foi feita apenas uma reserva para um fluxo individual.

Neste cenário o RMF efetua a gerência de recurso para várias requisições que entram e solicitam alocação. Para melhor visualizar as reservas é feita a redução do tamanho de alocação de banda para 100kbit e efetua-se então as várias reservas, para em seguida observar comportamento.

Para este cenários, utilizou-se de um ambiente semelhante aos anteriores, com somente um nó QNE no *meio* sendo que somente este efetua a reserva. Entretanto, será utilizado dois nós adjacentes além dos utilizados, pra fluxo não priorizado conforme indicado na Fig.4.8. Para maiores detalhes da configuração, verificar o ítem B:

A configuração básica do cenário é, portanto:

- QNI IP: eth0=192.168.19.11/24; NSIS como NullRMF; Software iperf como cliente para fluxo priorizado;

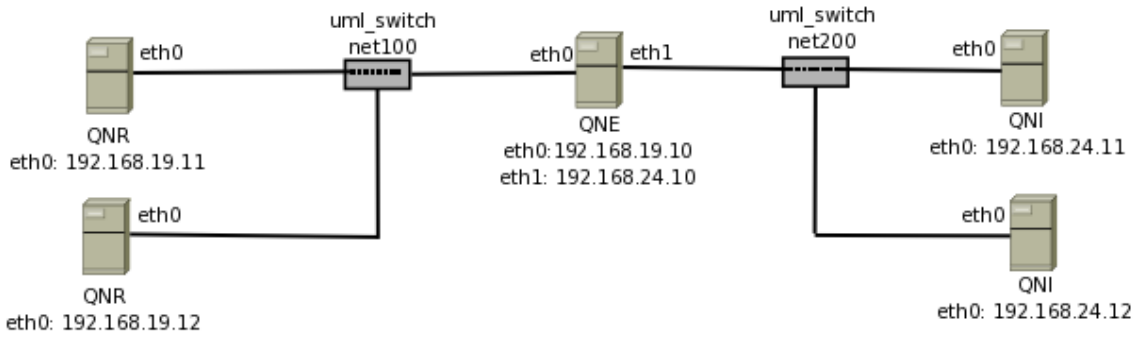


Figura 4.8: Cenário com reserva de recurso

- QNI IP: eth0=192.168.19.12/24; NSIS como NullRMF; Software iperf como cliente para fluxo não priorizado;
- QNE IP: eth0=192.168.19.10/24 IP: eth1=192.168.24.10/24 NSIS como SimpleRMF, fazendo reserva;
- QNR IP: eth0=192.168.24.11/24; NSIS como NullRMF; Software iperf como servidor do fluxo priorizado;
- QNR IP: eth0=192.168.24.12/24; NSIS como NullRMF; Software iperf como servidor do fluxo não priorizado;

4.6.1 Iniciando a função de RMF do NSIS

Na inicialização do nsis, é aplicado uma classe com taxa *rate* de 100kbit limitando o fluxo não priorizado, o que pode ser visto com o iperf:

```
(none):~# iperf -s -i 0
WARNING: interval too small, increasing from 0.00 to 0.5 seconds.
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.24.11 port 5001 connected with 192.168.19.11 port 35855
[  4]  0.0- 0.5 sec  9.90 KBytes    162 Kbits/sec
[  4]  0.5- 1.0 sec  4.24 KBytes  69.5 Kbits/sec
[  4]  1.0- 1.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  1.5- 2.0 sec  4.24 KBytes  69.5 Kbits/sec
```

```
[  4]  2.0- 2.5 sec  7.07 KBytes    116 Kbits/sec
[  4]  2.5- 3.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  3.0- 3.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  3.5- 4.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  4.0- 4.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  4.5- 5.0 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  5.0- 5.5 sec  5.66 KBytes  92.7 Kbits/sec
[  4]  5.5- 6.0 sec  5.66 KBytes  92.7 Kbits/sec
```

A cada reserva, será criada uma nova disciplina QDISC, que tratará o fluxo informado. O RMF tratará o fluxo de acordo com as informações passadas pela sinalização e fará os cálculos necessários para a agregação desta reserva.

Neste cenário será solicitado 4 reservas de fluxo, de 50Kbps, 40kbps, 10kpbs e 10kbps respectivamente. Contudo, deve-se atentar para um detalhe, o link foi limitado a 100kbps, ou seja, a última reserva, de 10kps não poderá ser feita, a menos que uma das anteriores seja retirada, devido a saturação do link.

4.6.2 Efetuando a reserva para um fluxo

Efetua-se então as reservas a partir do QNI:

- Para 50kbps

  ```
  ./nsis-qos 192.168.24.11 -res -bw 50000 -p 6 -sid 76 -dpt 9876
  ```

  E verifica-se a reserva no nó QNE:

  ```
   Reserving ...
  (TC) Running "tc class add dev eth1 parent 1:1 classid 1:6 htb rate
   48kbps burst 50000 ceil 48kbps mpu 64"
  (TC) Running "tc qdisc add dev eth1 parent 1:6 handle 6: sfq perturb 10"
  (TC) Running "tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
   match ip
  src 192.168.19.11/32 match ip dst 192.168.24.11/32 match ip protocol 6 0xff
  match ip dport 9876 0xffff  flowid 1:6"
  ```

- Em seguida, é feita a reserva de Para 40kbps

  ```
  ./nsis-qos 192.168.24.11 -res -bw 40000 -p 6 -sid 77 -dpt 9877
  ```

E verifica-se a reserva no nó QNE:

```
Reserving ...
(TC) Running "tc class add dev eth1 parent 1:1 classid 1:7 htb rate
 39kbps burst 40000 ceil 39kbps mpu 64"
(TC) Running "tc qdisc add dev eth1 parent 1:7 handle 7: sfq perturb 10"
(TC) Running "tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
 match ip
src 192.168.19.11/32 match ip dst 192.168.24.11/32 match ip protocol 6 0xff
match ip dport 9877 0xffff  flowid 1:7"
```

- Em seguida, a reserva de Para 10kbps

```
./nsis-qos 192.168.24.11 -res -bw 10000 -p 6 -sid 78 -dpt 9878
```

E verifica-se a reserva no nó QNE:

```
Reserving ...
(TC) Running "tc class add dev eth1 parent 1:1 classid 1:8 htb rate
 10000bps burst 10000 ceil 10000bps mpu 64"
(TC) Running "tc qdisc add dev eth1 parent 1:8 handle 8: sfq perturb 10"
(TC) Running "tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32
 match ip
src 192.168.19.11/32 match ip dst 192.168.24.11/32 match ip protocol 6 0xff
match ip dport 9877 0xffff  flowid 1:8"
```

- No entanto, conforme esperado, a última reserva não foi efetuada, pois não havia banda suficiente para alocação.

```
./nsis-qos 192.168.24.11 -res -bw 10000 -p 6 -sid 79 -dpt 9879
```

Com as alocações de banda efetuadas, cada fluxo priorizado tem a admissão de banda garantida sobre fluxos não priorizados. Isto pode ser visto utilizando o iperf, para isto foi efetuada a reserva de 50kbps e a de 40 kbps, restando 10kbps para o fluxo não priorizado. Em paralelo, é inserido tráfego na rede, conforme a reserva para averiguar a garantia de banda.

- Taxa de tráfego para o fluxo priorizado:

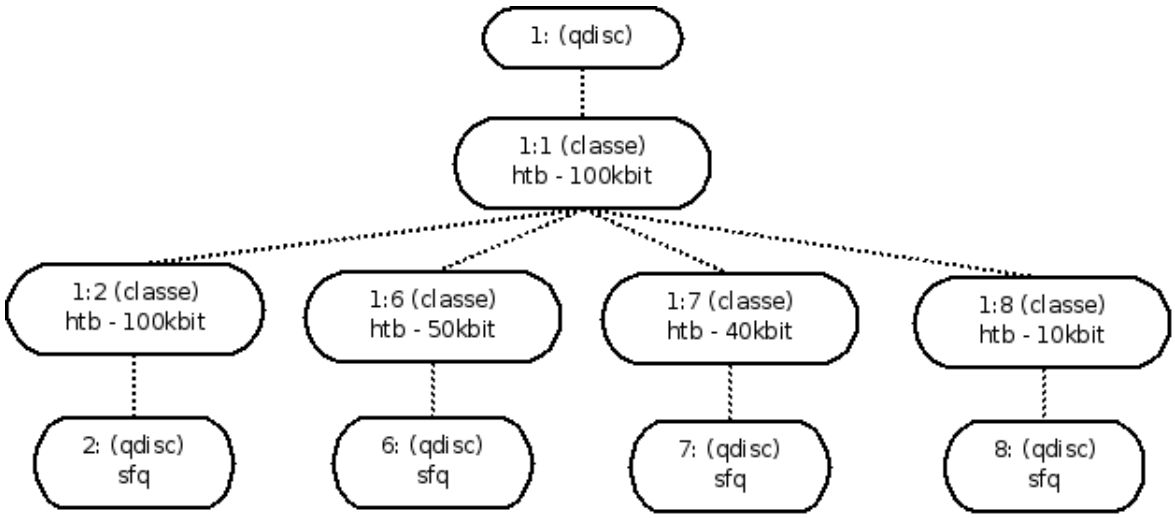


Figura 4.9: Classes de fluxo do controle de tráfego

```
WARNING: interval too small, increasing from 0.00 to 0.5 seconds.

------------------------------------------------------------
Server listening on TCP port 9877
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.24.11 port 9877 connected with 192.168.19.11 port 40751
[  4]  0.0- 0.5 sec  4.24 KBytes  69.5 Kbits/sec
[  4]  0.5- 1.0 sec  2.83 KBytes  46.3 Kbits/sec
[  4]  1.0- 1.5 sec  2.83 KBytes  46.3 Kbits/sec
[  4]  1.5- 2.0 sec  1.41 KBytes  23.2 Kbits/sec
[  4]  2.0- 2.5 sec  1.41 KBytes  23.2 Kbits/sec
[  4]  2.5- 3.0 sec  2.83 KBytes  46.3 Kbits/sec
```

- Taxa de tráfego para o fluxo não priorizado:

```
 (none):/NSIS/nsis-0.6.0/bin# iperf -s -i 0
WARNING: interval too small, increasing from 0.00 to 0.5 seconds.
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.24.12 port 5001 connected with 192.168.19.12 port 40718
```

```
[  4]  0.0- 0.5 sec  4.24 KBytes  46.3 Kbits/sec
[  4]  0.5- 1.0 sec  4.24 KBytes  0.00 Kbits/sec
[  4]  1.0- 1.5 sec  2.83 KBytes  0.00 Kbits/sec
[  4]  1.5- 2.0 sec  2.83 KBytes  0.00 Kbits/sec
[  4]  2.0- 2.5 sec  1.41 KBytes  23.2 Kbits/sec
[  4]  2.5- 3.0 sec  1.41 KBytes  23.2 Kbits/sec
[  4]  3.0- 3.5 sec  2.83 KBytes  0.00 Kbits/sec
[  4]  3.5- 4.0 sec  1.41 KBytes  0.00 Kbits/sec
```

4.7 Cenário 5

Neste cenário, observa-se um dos modelos do NSIS que o diferencia de modelos de sinalização, como o RSVP, por exemplo. A capacidade de reler a tabela de roteamento e refazer a reserva em caso de mudança de rota. Na mensagem de atualização de estado, verifica-se que o NSIS analisa se a reserva já existia, e caso nao exista, esta é efetuada.

Como o NSIS é um protocolo em desenvolvimento, este modelo ainda não foi implementado em sua totalidade, contudo, foram efetuadas as devidas análises.

No cenário utilizado deu-se preferência para que cada nó do cenário efetue uma tarefa, sendo que as bordas fazem e recebem a sinalização, o nó central faz a reserva deixando outros dois nós exclusivos ao roteamento.

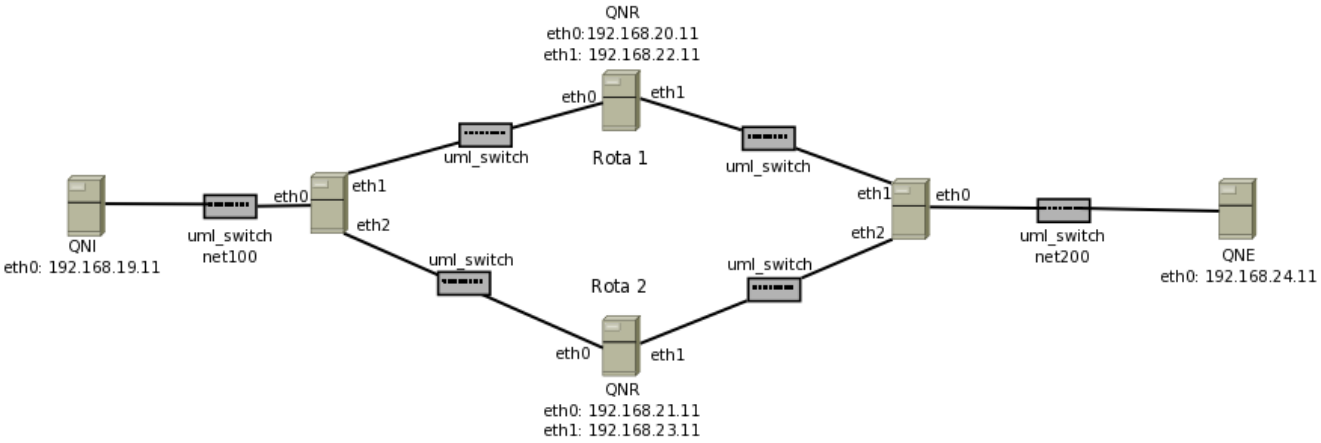


Figura 4.10: Sinalização com mudanca de rota

No cenário montado o *nsis* foi inicializado em todos os nós, e uma reserva foi feita de uma ponta a outra para o fluxo ser transmitido pela rede 192.168.21.0/24 e 192.168.22.0/24. Nos quatro nós do núcleo, está rodando algorítimos de roteamento, de modo que derru-

bando uma interface, a rota se refaz automaticamente.

Feita a sinalização, a reserva foi efetuada entre os nós do domínio. Em seguida as rotas do roteador ROT1 onde o fluxo estava passando foram removidas. Como o NSIS, através o *soft-state*, mantém mensagens de atualização da reserva, ao enviar a mensagem de *refresh*, a mudança é detectada.

Com este cenário, o comportamento esperado do NSIS, seria efetuar a reserva no novo caminho, contudo, em nossos testes, não foi possível efetuar a reserva no novo cenário.

Feita uma análise, diagnosticou-se que durante a inicialização do *daemon* nsis, a configuração de roteamento é carregada pelo nsis para o GIST, tanto se explícita no arquivo *nsis.conf*, como se lendo a tabela de roteamento. E a tabela de rotas do GIST não estava sendo atualizada.

A importância do protocolo de sinalização conhecer a tabela de roteamento se dá em conhecer o caminho e o destino do fluxo e justamente prover a capacidade de detectar a mudança por comparação de rota.

Contudo, neste mesmo cenário foi possível verificar o protocolo *soft-state* com mais detalhes. Uma vez que a rota não pode ser atualizada, o NSIS deve sinalizar a partir do nó adjacente a mudança para os nós seguintes para a remoção do status da sessão que foi aberta para reserva.

Conforme descrito no ítem 3.3.4, a sinalização é feita após o tempo de atualização de mensagems expirar, em seguida, sinaliza-se até encontrar o novo caminho, caso não se encontre, a reserva é removida. Conforme pode ser visto a partir do nós adjacente a antiga reserva:

- O nó detecta a mudança: *handleMessageStatus* (*GIST was unable to establish routing state with the peer*).
- O nó adjacente sinaliza para trás a informação de nova rota;

```
QNE: --> incoming msg
+ SII_Handle        : 192.168.20.10
+ srcAddr           : 192.168.19.11
+ destAddr          : 192.168.24.11
+ direction         : DOWNSTREAM
+ msg length        : 16 bytes
+ QoS Message Type  : RESERVE
```

```
+ Object Type       : RSN
++ RSN              : 1
++ EpochID          : 0x49ae3885
+ T:R:B:A:P:S-Flag  : 0:0:0:0:0:0

triggerEvent        : ST_INST::EV_RX_RESERVE

QoSFsm::inst__rx_reserve()
StateTimer restarted: 100s
```

– As mensagens reservas são derrubadas.

5 O protocolo NSIS pela Universidade de Goettingen

A IETF criou o grupo de trabalho para a criação de normas e definição do protocolo NSIS. Contudo, existem pelo menos 3 grupos de trabalho que implementam o NSIS, sendo duas implementações em C++ (Universidade de Karlsruhe e Goettingen, ambas na Alemanha) e uma em java (Universidade de Coimbra, Portugal).

Na Universidade de Goettingen, Alemanha, foi desenvolvida uma implementação do NSIS, o *Free Next Steps In Signaling* (FreeNSIS) inicialmente conhecido como OpenNSIS, tendo como integrantes da implementação Xiaoming Fu, Christian Dickmann, Bernd Schloer, Henning Peters, Ingo Juchem, Niklas Steinleitner, Hannes Tschofenig e Andreas Westermaier, além de contribuidores.

O grupo de trabalho do FreeNSIS mantém o site *http://user.informatik.uni-goettingen.de/ nsis/* que disponibiliza acesso a notícias, download das versões, informações do time do projeto e resposta a perguntas frequentes.

A implementação do NSIS pela Universidade de Goettingen, atualmente na versão 0.6.0, suporta algumas aplicações que podem ser oferecidas pelo NSIS, como as implementações de GIST, NSLP QoS, NAT / FW NSLP e aplicações para teste de comunicação e reserva. A maioria das aplicações são distribuídas sob a licença GPL[1]. No entanto, a API entre GIST e as NSLPs é liberado sob a licença LGPL[2].

Dentre os pacotes do NSIS implementado pela universidade de Goettingen, o NSIS-QoSD (Servidor de gerenciamento de QoS), NSIS-QoS (Cliente que faz requisições de QoS), NSIS-PING (análise da rede) e a implementação particular de NAT/FW NSLP.

[1]GNU General Public License (Licença Pública Geral), ou simplesmente GPL. A GPL é a licença com maior utilização por parte de projetos de software livre, em grande parte devido à sua adoção para o Linux.

[2]GNU Lesser General Public License: A principal diferença entre a GPL e a LGPL é que esta permite também a associação com programas que não estejam sob as licenças GPL, incluindo Software proprietário.

No pacote nsis contém arquivos para o desenvolvimento do protocolo, com todos os códigos fontes, nos diretórios:

- *gist*: Arquivos do GIST, da máquina de estados (FSM), da API do Gist;
- *library*: O *framework*, este diretório contém classes comuns e bibliotecas
- *nslp*: O diretório com os arquivos fonte do NSLP, contendo a API, as aplicações do NSIS e um modelo principal da confecção do NSIS.
- *applications*: aplicações de teste, como o nsis-ping;
- *bin*: diretório com os arquivos binários e o arquivo de configuração nsis.conf

Uma das funções mais importantes é a máquina de estados do protocolo, que gerencia as mensagens e identifica a sessão da sinalização. Cada nó QNI, QNE e QNR tem sua máquina de estado e uma não tem ligação direta com a outra. Para este modelo trataremos os três nós de maneira diferente, quando necessário for.

Conforme apresentado neste trabalho, o NSIS é divido basicamente em duas camadas, sendo a camada de transporte e a camada de aplicação, com seus aplicativos. Abriremos a camada de aplicação um pouco mais e a dividiremos em duas camadas, figura 5.1, o protocolo com a comunicação com a camada inferior e as aplicações que rodam sobre esta.

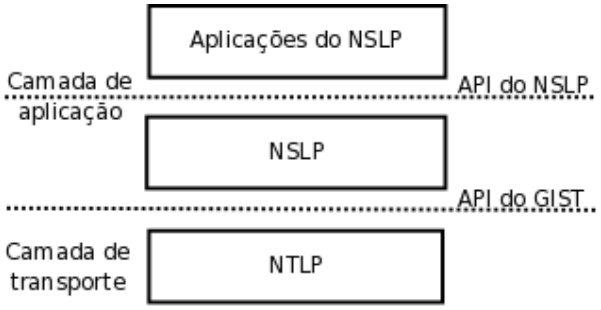


Figura 5.1: Estrutura de camadas NSIS

5.1 O NSLP

Subindo ao topo da arquitetura, mostramos o funcionamento do NSIS de Goettigen a partir da camada de aplicação até a camada de transporte. O QoS NSLP é executado através de um *daemon*, o nsis, que mantém suporte às aplicações do NSIS, que serão executadas.

Quando executado, o *daemon* do NSIS, o nsis-qosd fica escutando as mensagens recebidas a partir da aplicação cliente local nsis-qos e de nós diretamente ligadas a ele, partir de GIST. Ao receber uma mensagem, cria uma sessão com um identificador. A cada nova requisição de fluxo recebido, cria um identificador de sessão, denominado SID (*session id*) e uma máquina de estados FSM (*Finite State Machine*) associada a esta sessão.

Quando geradas as mensagens a partir do aplicativo cliente nsis-qos são marcadas para o tipo de mensagem (QUERY ou RESERVE) e para o SID. Este SID criado é pesquisado e se não for encontrado, é criada uma nova máquina de estados para esta sessão, relacionando o endereço com o SID, que será único durante toda a rede.

O NSLP trata eventos com estados próprios, e alternando entre os estados durante a troca de sinalização. Três estados estão definidos: ST_IDLE, ST_WR e ST_INST. Ao receber uma mensagem de sinalização, assume-se que o evento recebe o estado ST_IDLE. Neste modo a aplicação somente existe e está instanciada, não há ocorrência e registro de estado.

Em ST_WR a máquina de estado (FSM) espera uma resposta de uma mensagem enviada anteriormente, mas também não há registro de reserva. Contudo, em ST_INST o estado da reserva é instalado e mensagens recebidas serão tratadas. A partir da reserva, a maquina de estados gerência entre outras coisas, o tempo da reserva.

O tempo de resposta nos nós QNI e QNE é iniciada quando uma mensagem RESPONSE espera o envio de um QUERY ou RESERVE. Pode ser entendido como o intervalo da sinalização a manutenção do estado. Quando uma reserva é criada, o tempo de *Refresh* nos nós QNI e QNE são iniciados e os temporizadores *StateLife* são iniciados na QNE e QNR. Durante o período de reserva, este tempo é atualizado com mensagens de reserva, caso contrário, o tempo expira e o estado é removido.

5.2 O GIST

O GIST, ou *General Internet Signalling Transport*, assume que outros mecanismos são responsáveis pelo controle de roteamento dentro da rede. O GIST é um modelo padrão para a camada de transporte, que pode ser utilizado com os protocolos de sinalização como o RSVP e NSIS.

A Implementação do GIST por Goettingen consiste principalmente em seis partes:

- Eventos de *loop*, baseados nos soquetes de conexão;

- Mensagem de estado de roteamento (MRS);
- Associação de mensagem (MA);
- Máquina de estados (FSM - *Finite State Machine*)
- Mensagem de análise de composição;
- NSLP API

A sinalização da aplicação requer um conjunto de regras de gerenciamento de estado, bem como protocolo de apoio para a troca de mensagens de dados ao longo do caminho. Vários aspectos deste protocolo de apoio são comuns a todos ou um grande número de pedidos de sinalização e, portanto, pode ser desenvolvido como um protocolo comum.

5.3 Modelo de mensagens e sugestão de alteração

As mensagens do NSIS foram mostradas no capítulo 3 deste trabalho, contudo, relembra-se que o NSIS pode efetuar sinalização de modo *statefull*, mantendo o estado nos roteadores e *stateless*.

Em síntese o NSIS foi projetado para trabalhar com estas duas sinalizações, contudo, este dois modelos de mensagens poderiam ser utilizados juntos, onde uma mensagens sinaliza o agregado, de modo *statefull* de ponta a ponta e a outra mensagem sinalizando a cada nó, verificando disponibilidade do link.

O modelo com ambas sinalizações de modo concomitante foi uma proposta inicial para desenvolvimento de um modelo de sinalização a ser utilizado em domínios *DiffServ*. Esta proposta seria feita sobre o NSIS da Universidade de Goettingen. O NSIS oferece suporte nativo a mensagens em domínio *DiffServ*, permitindo sinalização por classe de serviço.

O NSIS poderia complementar a arquitetura *DiffServ*, aplicando por classe de classificação, onde nas bordas do domínio utilizaria um controle de admissão. O modelo permitiria a entidade no *DiffServ* sinalizar os pedidos de reserva na borda para o final. Em seguida uma segunda sessão de fluxo de tráfego é iniciada passando a especificação do fluxo em cada nó. Ao final, as mensagens se agregam e a sinalização continua.

Feito um estudo inicial, a implementação deste modelo exigira alteração nas classes do ciente NSIS, nsis-qosd, e na máquina de estados do NSLP, QoSFSM. Isto porque o protocolo oferece suporte para múltiplas sinalizações e agregação de mensagens.

Deste modo seria adicionado uma identificação da sessão destas sinalizações, a qual seria comparada no nó de borda, para identificação da mensagem. O servidor de QoS do NSIS deveria então entender e gerar uma identificação para esta mensagem, e a máquina de estados, entender e associar ao estado da mensagem, este novo campo.

A não utilização dos campos atuais de identificação de mensagem foi estudada, e, ao gerar dois fluxos de sinalização, o NSIS trata cada fluxo como uma sessão diferente. Esta proposta poderá ser utilizada em domínios *DiffServ*, com gerência de recurso. Cita-se este modelo como projeto futuro.

6 Conclusões e Perspectivas

O presente trabalho possibilitou o contato com o estado da arte da tecnologia, no que se refere a protocolo de sinalização em redes IP.

Através dos estudos efetuados, o protocolo de sinalização NSIS mostrou-se como uma solução robusta para trabalhar em domínios de rede, seja para sinalização fim-a-fim, seja para segmentos ou regiões específicas. No caso fim-a-fim, como elemento base para a interconexão de domínios heterogêneos, ou seja com diferentes tecnologias e modelos de QoS. Moderno em seu desenvolvimento, o NSIS atende a requisitos emergentes, como a gerência de recursos em domínios *DiffServ* e novos conceitos, como mobilidade em rede.

O objetivo primeiro deste trabalho foi estudar o protocolo NSIS e suas camadas, focando mais detalhadamente o QoS NSLP, a fim de entender sua concepção e aplicação, e comprovar experimentalmente algumas de suas características. Em alguns experimentos, como o de tratamento de mudança de rotas, a implementação utilizada não permitiu uma avaliação satisfatória.

Em um segundo objetivo, foi estudado os aspectos de implementação do protocolo visando a realização de futuras incorporações de mecanismos para o gerencimento de recursos em ambientes *DiffServ*. Os *drafts* do NSIS se referenciam a estes mecanismos, contudo a implementação utilizada não os disponibiza até o presente momento. Neste aspecto, o trabalho realizou um primeiro passo para futuros trabalhos.

Como perspectivas futuras pode-se citar:

1. um estudo mais detalhado da característica de atualização automática de reservas. Isto implica em um aprofundamento na camada GIST com fins de melhor entender o mecanismo de transporte genérico de mensagens;

2. um estudo das características associadas a mobilidade e a integração do protocolo NSIS com protocolos de gerenciamento de mobilidade;

3. uma implementação experimental de mecanismos para o gerenciamento de domínios

DiffServ;

Finalmente, o trabalho proporcionou ao autor, uma ampla visão de mecanismos e abordagens para o controle de QoS em redes, conhecimento este diretamente aplicável em seu ambiente de trabalho atual.

APÊNDICE A -- Montagem de Máquinas Virtuais

O ambiente de testes foi montado fazendo uso de máquinas virtuais. A *User Mode Linux* (UML) é nativa no *linux* e apresenta alto desempenho.

O uso das máquinas virtuais UML é muito fácil, simplesmente é necessário um *kernel* compilado para este tipo especifico de arquitetura e um *filesystem*. Neste trabalho, utilizamos tanto o *host* como a máquina virtual baseado na distribuição Debian.

Criando o Kernel

Para montar a máquina virtual é preciso um kernel compilado para a arquitetura UML. Para este ambiente foi utilizado a versão 2.6.25.6 do *kernel*. A criação de um kernel *linux* para uml é um procedimento relativamente simples, conforme descrito com os passos abaixo:

Obtendo o Kernel:

```
$ wget http://kernel.org/pub/linux/kernel/v2.6/linux-2.6.25.6.tar.bz2
$ tar jxvf linux-2.6.25.6.tar.bz2
$ cd linux-2.6.25.6
```

Como o ambiente de testes será utilizado máquinas virtuais devemos compilar o kernel para essa arquitetura:

```
$ export ARCH=um
```

Configurando kernel:

```
$ make defconfig $ make menuconfig
```

É interessante habilitar a opção *HOSTFS* para as máquinas virtuais terem acesso ao *filesystem* do *host* e as opções que estão em desuso do NETLINK. Para configurar o kernel

com suporte as opções deste trabalho, selecione no kernel as opções de *FIREWALL* e as tabelas (NAT, MANGLE, FILTER), NET, NETLINK, e opções de QoS. Estas opções devem ser setadas, no arquivo .config:

Após configurado o kernel deve ser compilado, os seus módulos serão instalados no diretório *uml-modules* para depois serem copiados para o *filesystem* da máquina virtual. O filesystem pode ser obtido no site *http://uml.nagafix.co.uk/*, que contém também alguns kernels do linux pré-compilados.

```
$ make
$ strip linux
$ make modules_install INSTALL_MOD_PATH=uml-modules
$ unset arch
```

Como se faz uso de máquinas virtuais é necessário copiar os módulos para o *filesystem* que será utilizado. Para isso:

```
# mkdir loop
# mount -o loop <seu_fs> loop
# cp -rf linux-2.6.25.6/uml-modules/lib/modules/* loop/lib/modules
```

Interface comunicação VM/Host

Para criar algumas funções de comunicação que podem ser úteis durante o uso da máquina virtual, aconselha-se a instalação dos seguintes pacotes:

```
#apt-get install uml-utilities openvpn openssl
```

Existem dois modos para criar interfaces de rede para a máquina virtual, através de interface virtual do *host* com o *tunctl* ou através do *uml_switch*.

```
# tunctl -u \textit{seu_usuario}
```

O comando acima cria um interface virtual, por exemplo, tap0. Se invocado novamente cria a tap1 e sucessivamente. Esta interface permite que a máquina virtual possa acessar a internet, o que facilita a instalação de pacotes dentro da mesma.

Configura-se a interface normalmente:

```
# ifconfig tap0 192.168.20.1 up
```

Para apagar a interface, utilize a opção -d:

```
# tunctl -d tap0
```

Ativar o roteamento de pacotes na máquina *host*:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Há outro meio de criar uma interface de rede para conectar as máquinas virtuais, que cria um *switch*, utilizando a *uml_switch*. Este modo foi o utilizado neste trabalho para a montagem de cenários, para isto cria-se um switch com o comando:

```
$ uml_switch -unix /tmp/net100.ctl
```

Onde net100.ctl é o nome do *switch* que será referenciado na inicialização da UML.

Utilizando a máquina Virtual

Após criado o *Kernel* e o *Filesystem* com os módulos do *kernel* compilado, pode-se invocar a máquina virtual, certificando-se de apontar os arquivos conforme diretório.

Para chamar a máquina virtual, altera-se a interface de comunicação conforme abaixo:

```
$ xterm -title Debian1 -e ./linux ubda=Debian-4.0-x86-root_fs
eth0=daemon,,,/tmp/net100.ctl
```

Ou para compartilhar através da interface tap0:

```
$ xterm -title Debian1 -e ./linux ubd0=Debian-4.0-x86-root_fs
eth0=tuntap,tap0
```

Ou Ainda...

```
$ xterm -title Debian1 -e ./linux-ro
ubd0=./cow-host1,./Debian-4.0-x86-root_fs eth0=tuntap,tap0
```

Neste último caso, cria-se um *filesystem* de uma máquina *virtual*, como *cow-host1*, que poderá ser invocado conforme o primeiro comando. Neste modo, as configurações feitas e salvas no *cow-host1* poderão ser utilizadas na próxima inicialização.

Onde:

1. xterm: Programa para abrir a máquina virtual em janela do Ambiente Gráfico, com os seguintes parâmetros:
   - -title: Passa o nome para a janela do *xterm*
   - -e: Passa comando para ser executado dentro da janela do *xterm*. Aqui chamaremos a máquina virtual.
2. ./linux: O caminho do *kernel* da Máquina Virtual.
3. ubd0=: O caminho do *Filesystem* da Máquina Virtual.
4. eth0=: A interface de rede que a máquina virtual irá comunicar, exemplo: tuntap,tap0

Montando o diretório do Host na Máquina Virtual

Na máquina host, cria-se uma pasta com o comando:

```
# mkdir host
```

Com a máquina virtual funcionado, trabalhe como um sistema operacional igual.

Monte o diretório do *host* na máquina virtual:

```
# mount none /host(pasta VM) -t hostfs -o /home/rodrigo(pasta host)
```

Você também pode montar a unidade de mídia removível, conforme segue:

```
# mount none /cdrom -t hostfs -o /cdrom/
```

Acese o diretório correspondente ao *cdrom*.

```
# mount none /cdrom/ -t hostfs udf,iso9660 -o /cdrom/
```

APÊNDICE B -- Arquivo de configuração do NSIS

Para configuração dos cenários o arquivo nsis.conf recebe alterações conforme posição de cada nó no domínio da rede.

Um exemplo do arquivo nsis.conf segue abaixo, bem como, em seguida ítens específicos de cada nó.

Aquivo padrão: **nsis-0.6.0/bin/nsis.conf**

```
# This is an example configuration file for NSIS
# Please change it to fit your needs (i.e. IP address configuration
# of your host).
#
# For more information about the configuration, check the manual at
#   manual/Configuration

# ******************************************
# *****      General Configuration      *****
# ******************************************

# Start Ping and Diagnostics NSLP daemon together with GIST
nslp.startPing = yes
nslp.startQoS = yes
nslp.startNatFw = no
nslp.startDiag = no

# Accept explicitly routed messages (default is yes)
gist.acceptExplicitMessages = yes
```

```
# Accept GIST DATA messages that do not relate to any GIST state (default is
yes)
gist.acceptStatelessGistMessage = yes

# BSD only:
# List of interfaces where NSIS should listen on for GIST Queries
# Interfaces are separated by commas
#bsd.interfaces = ed0, ed1


# ******************************************
# *****  GIST Transport Configuration   *****
# ******************************************

## Policies which transport protocols are offered to peers.
# Offer SCTP as transport to peers?
gist.offerSCTP = no

# Offer TLS over TCP as transport to peers?
gist.offerTLS = no

## Policies
# Prefer SCTP over TCP as transfer protocol?
gist.useSCTP = no


# ******************************************
# *****    GIST Timer Configuration     *****
# ******************************************

## All Timeouts are measured in milliseconds

# How long do we wait for a Response to out initial Query?
# On retransmission, this value is doubled each time. (default: 10000 ms)
```

```
gist.timeout.waitForInitialResponse = 10000

# How long do we wait for a Confirm on the Receiver-Side? (default: 10000 ms)
gist.timeout.waitForConfirm = 10000

# How long do we wait between sending refreshing Queries? (default: 30000 ms)
gist.timeout.refreshInterval = 30000

# How long do we wait for a Response to a refreshing Query
# until state is removed? (default: 100000 ms)
gist.timeout.queryingNodeStateExpiration = 100000

# How long do we wait for a refreshing Query
# until state is removed? (default: 100000 ms)
gist.timeout.respondingNodeStateExpiration = 100000


# *****************************************
# ***** IP address/routing configurtion *****
# *****************************************

# If readRoutingTable is set to yes, all IP address configuration
# used by NSIS is derived from the local IP routing tables and
# interface information.
# NOTE: If readRoutingTable is set to yes, all remaining IP address
# configuration in this file is NOT used by NSIS.
readRoutingTable = yes

# CAUTION: The address configuration is like a routing table.

# This example IPv4 configuration contains a default route
# as well as special configuration for two network segments
# (i.e 192.168.0.0/24 and 192.168.1.0/24)
IPv4.entries = 3
```

```
# The first entry is meant as a default route. It is used when
# no subsequent entry matches.
IPv4[0].addr = 192.168.10.1
IPv4[0].net  = 192.168.10.0
IPv4[0].mask = 24
# Use this address as the external address for NatFW NSLP
IPv4[0].natfw.useAsExternalAddress = no
# Default network is public (i.e. the global internet)
IPv4[0].natfw.isPrivateNet = no


# This entry defines the outgoing local interface card
IPv4[1].addr = 192.168.10.1
# These entries define the network segment you want to reach
# with the previously defined outgoing interface card
IPv4[1].net  = 192.168.0.0
IPv4[1].mask = 24
# The network 192.168.0.0/24 is private
IPv4[1].natfw.isPrivateNet = yes

# The second network (192.168.1.0/24) uses the SAME outgoing
# interface again.
IPv4[2].addr = 192.168.20.1
IPv4[2].net  = 192.168.20.0
IPv4[2].mask = 24
# The network 192.168.1.0/24 is private
IPv4[2].natfw.isPrivateNet = yes

# The second network (192.168.1.0/24) uses the SAME outgoing
# interface again.
IPv4[3].addr = 192.168.30.1
IPv4[3].net  = 192.168.30.0
IPv4[3].mask = 24
# The network 192.168.1.0/24 is private
IPv4[3].natfw.isPrivateNet = yes
```

```
# The second network (192.168.1.0/24) uses the SAME outgoing
# interface again.
IPv4[4].addr = 192.168.40.1
IPv4[4].net  = 192.168.40.0
IPv4[4].mask = 24
# The network 192.168.1.0/24 is private
IPv4[4].natfw.isPrivateNet = yes

# IPv6 configuration with 1 address: (just remove this part completely
# if there is no IPv6 support on your maschine)
#IPv6.entries = 0
#IPv6[0].addr = fec0:1::4321:ff:ee12:3355
#IPv6[0].net  = fec0:1::4321:ff:ee12:3355
#IPv6[0].mask = 0



# *******************************************
# *****    NatFW NSLP Configuration     *****
# *******************************************

# This host runs a NAT, but no firewall
natfw.isNAT = no
natfw.isFW = no

# Hosts inside the private network can reserve external addresses/ports.
# As the above configuration shows, 10.0.0.1 is the only external address this
# router has to offer:
natfw.resources.IPv4.entries = 0
natfw.resources.IPv4[0].addr = 10.0.0.1


# *******************************************
# *****     QoS NSLP Configuration      *****
```

```
# ****************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = NullRMF
qos.debuglevel = 3
qos.reliable = true

## Configuration related to the SimpleRMF
# Use Traffic Control on these addresses if SimpleRMF is used:
qos.simplermf.interfaces = eth0

# Specify maximum overall bandwidth of the interface (default 100mbit)
qos.simplermf.interface[eth0].overall_bandwidth = 100mbit

# Specify mininum bandwidth, that unclassified traffic can use (default 100kbit)
# (Unclassified traffic is such traffic, that is not part of any known flow)
qos.simplermf.interface[eth0].min_unclassified_bandwidth = 100kbit

# Specify maximum bandwidth, that unclassified traffic can use
# (default is maximum overall bandwidth)
qos.simplermf.interface[eth0].max_unclassified_bandwidth = 100mbit
```

B.0.1 Arquivos Cenários 1

Os nós sofreram as seguintes alterações para funcionamento:

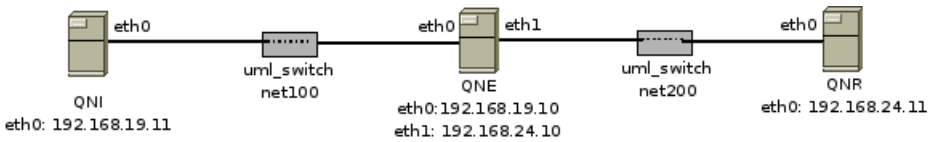


Figura B.1: Cenário 1

- Configuração do nó QNI (Que fará sinalização)

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração do nó QNI (Apenas para gerar tráfego) Não é necessário rodar o nsis neste host

- Configuração do nó QNE

```
readRoutingTable = yes

# ******************************************
# *****     QoS NSLP Configuration      *****
# ******************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = SimpleRMF
qos.debuglevel = 3
qos.reliable = true
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

B.0.2 Arquivos Cenários 2

Os nós sofreram as seguintes alterações para funcionamento:

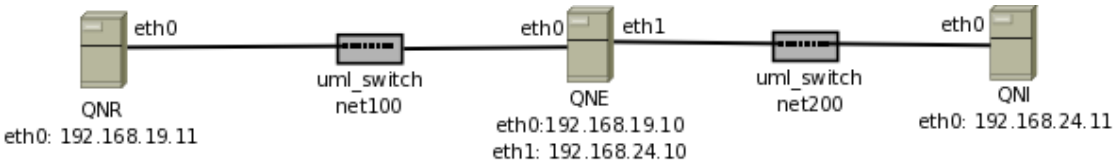


Figura B.2: Sinalização de reserva com query

- Configuração do nó QNI (Que fará sinalização)

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração do nó QNE

```
readRoutingTable = yes

# ******************************************
# *****     QoS NSLP Configuration      *****
# ******************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = SimpleRMF
qos.debuglevel = 3
qos.reliable = true
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

B.0.3 Arquivos Cenários 3

Os nós sofreram as seguintes alterações para funcionamento:

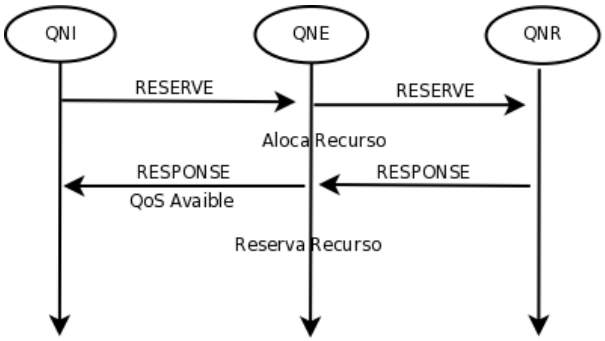


Figura B.3: Sinalização de reserva

- Configuração do nó QNI (Que fará sinalização)

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração do nó QNE

```
readRoutingTable = yes

# ******************************************
# *****     QoS NSLP Configuration      *****
# ******************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = SimpleRMF
qos.debuglevel = 3
qos.reliable = true
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

B.0.4 Arquivos Cenários 4

Os nós sofreram as seguintes alterações para funcionamento:

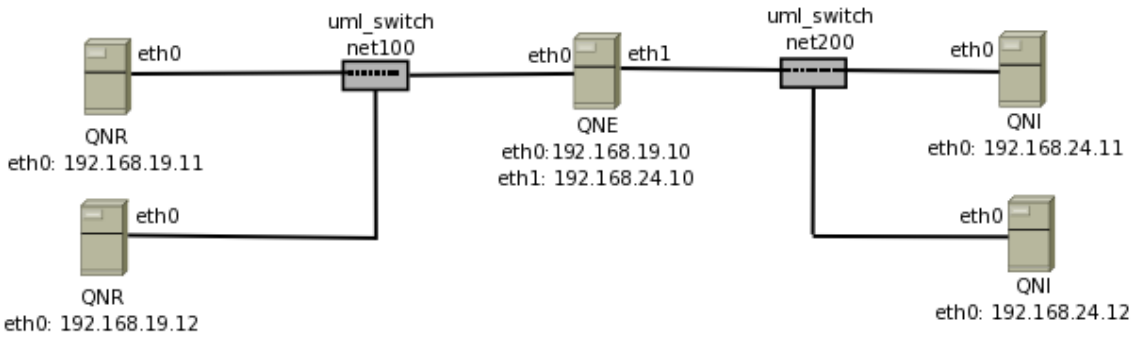


Figura B.4: Cenário com reserva de recurso

- Configuração do nó QNI (Que fará sinalização)

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração do nó QNI (Apenas para gerar tráfego) Não é necessário rodar o nsis neste host
- Configuração do nó QNE

```
readRoutingTable = yes

# ******************************************
# *****     QoS NSLP Configuration      *****
# ******************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = SimpleRMF
qos.debuglevel = 3
qos.reliable = true
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

B.0.5 Arquivos Cenários 5

Os nós sofreram as seguintes alterações para funcionamento:

- Configuração do nó QNI (Que fará sinalização)

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração do nó QNI (Apenas para gerar tráfego) Não é necessário rodar o nsis neste host

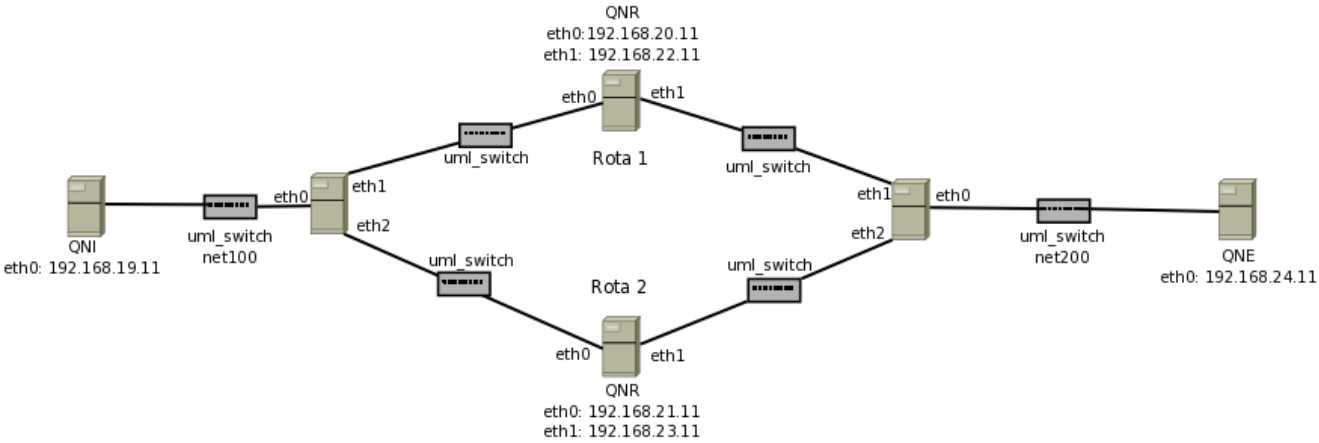


Figura B.5: Sinalização com mudanca de rota

- Configuração do nó QNE

```
readRoutingTable = yes

# ******************************************
# *****     QoS NSLP Configuration      *****
# ******************************************

# Choose the RMF: ClsRMF, SimpleRMF and NullRMF (default) are available.
# See man page on what they do. (case does not matter)
qos.rmf = SimpleRMF
qos.debuglevel = 3
qos.reliable = true
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

- Configuração QNR

```
readRoutingTable = yes
qos.rmf = NullRMF
```

Referências Bibliográficas

ASH A. BADER, C. K. D. O. G. *QoS NSLP QSPEC Template*. IETF, nov 2008. Ietf-nsis-qspec-21. (Draft of Request for Comments). Disponível em: <http://tools.ietf.org/html/draft-ietf-nsis-qspec-21.txt>.

BERNET, Y. et al. Rfc2998: A framework for integrated services operation over diffserv networks. *RFC Editor United States*, RFC Editor, United States, 2000.

BLAKE, S. et al. *An Architecture for Differentiated Service*. IETF, dec 1998. RFC 2475 (Informational). (Request for Comments, 2475). Updated by RFC 3260. Disponível em: <http://www.ietf.org/rfc/rfc2475.txt>.

BRADEN, R.; CLARK, D.; SHENKER, S. *Integrated Services in the Internet Architecture: an Overview*. IETF, jun 1994. RFC 1633 (Informational). (Request for Comments, 1633). Disponível em: <http://www.ietf.org/rfc/rfc1633.txt>.

CLARK., D. D. *The design philosophy of the DARPA internet protocols*. [S.l.]: Stanford, 1988.

IANA. *Differentiated Services Field Codepoints*. Mar 2008. Http://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml.

ITU-T. *Terms and Definitions Related to Quality of Service and Network Performance Including Dependability*. ITU-T, aug 1994. ITU-T E800. (ITU-T Recommendation, E800). Disponível em: <http://www.iso.org>.

KUROSE, J. F.; ROSS, K. *Redes de computadores e a Internet. Uma abordagem top-down*. 3. ed. São Paulo: Pearson Education, 2006.

MANNER G. KARAGIANNIS, A. M. J. *NSLP for Quality-of-Service Signaling*. IETF, fev 2008. Draft-ietf-nsis-qos-nslp-16. (Draft of Request for Comments). Disponível em: <http://tools.ietf.org/id/draft-ietf-nsis-qos-nslp-16.txt>.

MARCHESE, M. *QoS over Heterogeneous Networks*. [S.l.]: John Wiley & Sons Ltd, 2007.

MASIP-BRUIN, X. et al. The euqos system: a solution for qos routing in heterogeneous networks. *IEEE Communications Magazine*, IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, v. 45, n. 2, p. 96, 2007.

NARTEN, T.; ALVESTRAND, H. *Guidelines for Writing an IANA Considerations Section in RFCs*. IETF, oct 1998. RFC 2434 (Best Current Practice). (Request for Comments, 2434). Updated by RFC 3692. Disponível em: <http://www.ietf.org/rfc/rfc2434.txt>.

PARK, K. I. *QoS in packet networks*. [S.l.]: Springer, 2005.

SCHULZRINNE, R. H. H. *GIST: General Internet Signalling Transport*. IETF, nov 2008. Draft-ietf-nsis-ntlp-17. (Draft of Request for Comments). Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-nsis-ntlp-17.txt>.

WROCLAWSKI, J. *Specification of the Controlled-Load Network Element Service*. IETF, sep 1997. RFC 2211 (Proposed Standard). (Request for Comments, 2211). Disponível em: <http://www.ietf.org/rfc/rfc2211.txt>.