

RAFAEL RODRIGUES OBELHEIRO

**UM OVERLAY DE ROTEAMENTO
TOLERANTE A INTRUSÕES**

FLORIANÓPOLIS

2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UM OVERLAY DE ROTEAMENTO
TOLERANTE A INTRUSÕES

Tese submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Doutor em Engenharia Elétrica.

RAFAEL RODRIGUES OBELHEIRO

Florianópolis, novembro de 2006.

UM OVERLAY DE ROTEAMENTO TOLERANTE A INTRUSÕES

Rafael Rodrigues Obelheiro

Esta Tese foi julgada adequada para a obtenção do título de Doutor em Engenharia Elétrica, Área de Concentração em *Sistemas de Informação*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Prof. Joni da Silva Fraga, Dr.
Orientador

Prof. Nelson Sadowski, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Joni da Silva Fraga, Dr.
Presidente

Prof. Edmundo Roberto Mauro Madeira, Dr.

Prof. Antônio Marinho Pilla Barcellos, Dr.

Prof. Jean-Marie Farines, Dr.

Prof. Lau Cheuk Lung, Dr.

Prof. Leandro Buss Becker, Dr.

THE ROAD TO WISDOM

The road to wisdom? — Well, it's plain
and simple to express:

Err
and err
and err again
but less
and less
and less.

Piet Hein

À Tati e à minha família, uma vez mais

AGRADECIMENTOS

Ao meu orientador, Joni Fraga, pela sua amizade, dedicação, paciência, entusiasmo e pela confiança depositada, além das numerosas oportunidades que me proporcionou ao longo destes anos de trabalho conjunto. A sua perspectiva e objetividade foram fundamentais nos momentos em que o trabalho ameaçou desviar-se do seu caminho.

Aos colegas que integraram o grupo de pesquisa liderado pelo Joni, cuja visão crítica, incentivo, companheirismo e sobretudo amizade genuína foram fonte de reflexão e ajudaram a refinar o trabalho. Muito obrigado a Michelle Wangham, Lau Lung, Altair Santin, Fábio Favarim, Emerson de Mello, José Eduardo Brandão, Paulo Mafra, Wagner Dantas, Edson Camargo, Eduardo Alchieri, Luciano Costa, Fernando Pereira e Luiz Otávio Lento. Faço uma menção especial a Alysson Neves Bessani, com quem pude discutir muitas das idéias que fazem parte desta tese, e que foi uma fonte inesgotável de criatividade, otimismo, encorajamento e bom humor.

Aos colegas que minha longa estada no DAS me proporcionou conhecer e que se tornaram também grandes amigos. Em particular, menciono Luciana Frigo, Cássia Tatibana, Patrícia Pena, Fábio de la Rocha, Rodrigo Sumar, Marcos Vallim, André Leal, Jerusa Marchi, Eder Gonçalves, Luciano Rottava da Silva, Ricardo Schmidt, Cristiane Paim, Karina Barbosa, Augusto Loureiro e Carlos Brandão, além daqueles que porventura esqueci.

Aos professores do DAS, pelo seu comprometimento com a atividade acadêmica e pela sua amizade, especialmente a Jean-Marie Farines, Edson de Pieri e José Eduardo Cury.

A Wilson Costa e Marcelo Siqueira, da secretaria do PPGEEL, Marlos Gerber, da secretaria do DAS, e Luciano, Paulo, Róbson e Rodrigo, administradores da rede do DAS, por resolverem os mais variados problemas sempre com incansável presteza e cordialidade.

Aos professores que participaram da banca, pelas suas valiosas críticas e sugestões. Em especial, a Edmundo Madeira, que aceitou a tarefa de ser o relator da tese, e a Marinho Barcellos, que não apenas forneceu o simulador utilizado como esteve sempre pronto a me ajudar em meus experimentos.

A Jean-Marie Farines, que me deu a oportunidade de fazer um estágio sanduíche no LAAS-CNRS, em Toulouse (França). Aos companheiros da “Salle Carioca” *et les siens*, que nos ajudaram de inúmeras maneiras e se tornaram grandes amigos: Magnos e Roberta, Valentim e Alessandra, Eduardo e Karyn, Francisco e Cristina, Guillermo e David. À Mohamed Kaâniche, pour son accueil et son soutien amical pendant mon séjour au LAAS, qui sont allés bien au-delà de ses responsabilités avec moi.

À CAPES e ao CNPq, pelo apoio financeiro.

À minha família, que, de longe ou de perto, sempre me apoiou incondicionalmente.

À Tati, por me dar o privilégio incomensurável de partilhar a vida comigo.

A Deus, por me permitir chegar até aqui e viver tudo isso. Que Ele me permita seguir desfrutando de tantos amigos.

Resumo da Tese apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

UM OVERLAY DE ROTEAMENTO TOLERANTE A INTRUSÕES

Rafael Rodrigues Obelheiro

Novembro/2006

Orientador: Prof. Joni da Silva Fraga, Dr.

Área de Concentração: Sistemas de Informação Industrial

Palavras-chave: tolerância a intrusões, redes *overlay*, roteamento

Número de Páginas: xiii + 104

À medida em que pessoas e organizações se tornam cada vez mais dependentes de aplicações ligadas em rede, a disponibilidade da Internet se torna um aspecto importante. Diversos estudos demonstram que a Internet está sujeita a várias fontes de indisponibilidade, incluindo algumas limitações dos protocolos usados na sua camada de roteamento. Com base nessas observações, alguns pesquisadores propuseram o uso de *overlays* de roteamento como uma solução para melhorar a disponibilidade da rede. Estes *overlays* são redes lógicas que usam roteamento na camada de aplicação para encaminhar pacotes através de nós intermediários com o objetivo de contornar falhas em rotas. Entretanto, os *overlays* de roteamento existentes praticamente ignoram ameaças de segurança, o que os torna vulneráveis a atividades maliciosas que visam perturbar as comunicações na rede. Esta tese introduz um arquitetura de rede overlay tolerante a intrusões (ROTI). A ROTI é o primeiro *overlay* de roteamento que permite a comunicação de nós mesmo na presença de faltas e intrusões na rede. A arquitetura é composta por um novo protocolo de roteamento e mecanismos de encaminhamento de pacotes, detecção de falhas e recuperação de faltas, que juntos fornecem um serviço de comunicação tolerante a intrusões. É introduzido também um mecanismo inovador de recuperação de faltas baseado na reconfiguração da topologia da rede, que é uma de nossas principais contribuições. A solução é avaliada analiticamente, através de uma análise qualitativa da segurança oferecida e de uma análise de confiabilidade baseada em grafos que permite determinar o grau de tolerância a faltas e intrusões de um *overlay*, e também experimentalmente, através de um modelo de simulação. Os resultados obtidos demonstram que a ROTI consegue atingir seus objetivos de modo bastante eficiente, lidando com faltas e intrusões sem afetar significativamente o desempenho da rede.

Abstract of Thesis presented to UFSC as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

AN INTRUSION-TOLERANT ROUTING OVERLAY

Rafael Rodrigues Obelheiro

November/2006

Advisor: Prof. Joni da Silva Fraga, Dr.

Area of Concentration: Information Systems

Key words: intrusion tolerance, overlay networks, routing

Number of Pages: xiii + 104

As individuals and organizations become increasingly dependent on networked applications, Internet availability becomes a major issue. Various studies show that the Internet is subject to several sources of unavailability, including some shortcomings of the protocols used at the routing layer. Based on these observations, some researchers have proposed the use of routing overlays as a solution for improving network availability. Such overlays are logical networks that use application-level routing to forward packets across intermediate nodes in order to circumvent path outages. However, existing routing overlays mostly ignore security threats, which renders them vulnerable to malicious activities that aim to disrupt communications on the network. This thesis introduces an intrusion-tolerant overlay network architecture called ROTI. ROTI is the first routing overlay that enables nodes to communicate in spite of faults and intrusions in the network. The architecture comprises a novel routing protocol and mechanisms for packet forwarding, failure detection and fault recovery that together provide an intrusion-tolerant communications service. We also introduce an innovative fault recovery mechanism based on network topology reconfiguration, which is one of our main contributions. Our solution is evaluated analytically, through a qualitative analysis of the security provided and a graph-based reliability analysis that allows us to determine the degree of fault and intrusion tolerance provided by an overlay, as well as experimentally, through a simulation model. Our results demonstrate that ROTI is able to achieve its goals in a highly cost-effective manner, coping with faults and intrusions without strongly impacting network performance.

Sumário

1	Introdução	1
1.1	Motivação	2
1.2	Objetivos	5
1.3	Organização do Texto	5
2	Redes Overlay: Conceitos e Experiências	7
2.1	Roteamento na Internet	7
2.2	Redes Overlay	10
2.2.1	RON	12
2.2.2	SOSR	13
2.3	Ameaças a <i>Overlays</i> de Roteamento	14
2.4	Conclusões do Capítulo	16
3	Rede Overlay Tolerante a Intrusões (ROTI): Visão Geral	17
3.1	Contexto e Objetivos	17
3.2	Descrição da Arquitetura e Premissas	18
3.3	Subsistemas da ROTI	20
3.4	Conclusões do Capítulo	21
4	Roteamento e Encaminhamento de Pacotes na ROTI	22
4.1	Roteamento na ROTI	23
4.1.1	Fase de Roteamento Proativo	23

4.1.2	Fase de Roteamento Reativo	26
4.1.3	Seleção de Rotas	32
4.2	Encaminhamento de Pacotes na ROTI	33
4.2.1	Esquema Leve	34
4.2.2	Esquema Pesado	35
4.2.3	Comparação Entre os Esquemas	36
4.3	Trabalhos Relacionados	38
4.3.1	Roteamento	38
4.3.2	Encaminhamento de Pacotes	40
4.4	Conclusões do Capítulo	43
5	Detecção de Falhas e Recuperação de Faltas na ROTI	44
5.1	Detecção de Falhas	45
5.2	Recuperação de Faltas	48
5.2.1	Esquema Leve	48
5.2.2	Esquema Pesado	48
5.3	Reconfiguração de Topologia	49
5.3.1	Descrição dos Protocolos	50
5.3.2	Definição dos Parâmetros de Reconfiguração	54
5.3.3	Considerações Sobre a Reconfiguração de Topologia	57
5.4	Trabalhos Relacionados	59
5.5	Conclusões do Capítulo	63
6	Resultados	65
6.1	Análise de Segurança	66
6.1.1	Confidencialidade	66
6.1.2	Integridade	67
6.1.3	Disponibilidade	67

6.1.4	Considerações sobre a Análise	70
6.2	Análise de Confiabilidade	71
6.2.1	Modelo ROTI	71
6.2.2	Análise do Grau de Tolerância a Faltas	71
6.2.3	Considerações sobre a Análise	75
6.3	Resultados de Simulação	75
6.3.1	Ambiente de Simulação	76
6.3.2	Descrição da Simulação	77
6.3.3	Parâmetros da Simulação	81
6.3.4	Resultados Obtidos	82
6.3.5	<i>Overhead</i> do Roteamento	84
6.4	Trabalhos Relacionados	85
6.5	Conclusões do Capítulo	88
7	Conclusões	90
7.1	Revisão dos Objetivos	91
7.2	Contribuições e Resultados da Tese	92
7.3	Perspectivas Futuras	93
A	Conceitos de Teoria dos Grafos	95
A.1	Definições Básicas	95
A.2	Relações entre Grafos	95
A.3	Vizinhança e Grau	96
A.4	Caminhos	96
A.5	Conectividade	97

Lista de Figuras

2.1	Rede <i>overlay</i> sobreposta a uma rede física	10
3.1	Arquitetura de rede <i>overlay</i> tolerante a intrusões	19
3.2	Subsistemas em um nó ROTI	21
4.1	Exemplo de execução do protocolo de roteamento proativo	24
4.2	Exemplo de descoberta de rotas com o protocolo reativo básico	27
4.3	Roteamento reativo: funcionamento de CACHE-REPLY	28
4.4	Exemplo de descoberta de rotas com o protocolo reativo completo	30
4.5	Formato de um pacote ROTI usando o esquema leve	35
4.6	Formato de um pacote ROTI usando o esquema pesado	35
4.7	Construção de um pacote no esquema pesado	36
4.8	Um nó faltoso que atua como relé	40
5.1	Detecção de falhas com <i>pings</i>	45
5.2	Detecção de falhas com <i>heartbeats</i>	46
5.3	Localização de faltas com ACKs de nós intermediários	50
5.4	Execução do protocolo de reconfiguração: substituição do nó <i>e</i> pelo reserva <i>f</i> ($\phi = 3$, $\rho = 2$)	56
6.1	Exemplo de rede <i>overlay</i> sobreposta a uma rede física	72
6.2	Exemplo de projeção de rede <i>overlay</i>	73
6.3	Estrutura da simulação da ROTI	78
6.4	Distribuição da latência dos enlaces virtuais da simulação	81

6.5	Resultados de simulação: confiabilidade da ROTI	83
6.6	<i>Overhead</i> do protocolo de roteamento da ROTI	85
6.7	Tamanho médio do <i>cache</i> de rotas	86
6.8	Média de rotas por destino no <i>cache</i>	86

Lista de Tabelas

4.1	Primitivas usadas nos algoritmos do protocolo de roteamento	31
4.2	Resumo dos níveis de confiança para rotas em <i>cache</i>	33
4.3	<i>Overhead</i> de tempo dos esquemas de encaminhamento de pacotes	37
5.1	Funções usadas nos algoritmos 5.1 a 5.4	52
5.2	Variações da arquitetura ROTI	57
6.1	Resumo das contramedidas adotadas na ROTI	70

Lista de Algoritmos

4.1	Protocolo de roteamento reativo: início da descoberta de rotas	29
4.2	Protocolo de roteamento reativo: processamento de ROUTE-REQUEST	30
4.3	Protocolo de roteamento reativo: processamento de ROUTE-REPLY	30
4.4	Protocolo de roteamento reativo: processamento de CACHE-REPLY	31
5.1	Reconfiguração: PROPOSE-RECONF	52
5.2	Reconfiguração: processamento de proposições de reconfiguração	54
5.3	Reconfiguração: START-RECONF	55
5.4	Reconfiguração: protocolo do nó de reserva	55

Capítulo 1

Introdução

A Internet propiciou uma mudança importante na maneira com que pessoas e organizações interagem umas com as outras. À medida em que a rede global amadureceu, se consolidou e se tornou mais difundida, ela não apenas oportunizou o surgimento de novas aplicações, tais como comércio eletrônico, *e-business* e redes colaborativas, mas também ofereceu uma alternativa com melhor custo-benefício a organizações que dependiam de linhas dedicadas e redes privadas para satisfazer suas necessidades de comunicação. Esses novos padrões de uso impõem requisitos de segurança e tolerância a faltas cada vez mais severos, que estão além do que a rede foi originalmente projetada para oferecer. Tecnologias de segurança como IPsec [Kent e Seo, 2005], redes privadas virtuais (VPNs — *virtual private networks*) [Gleeson et al., 2000] e SSL/TLS (*Secure Sockets Layer/Transport Layer Security*) [Freier et al., 1996; Dierks e Rescorla, 2006] têm sido usadas para preencher essa lacuna no que diz respeito a confidencialidade, integridade e autenticação de comunicações. Entretanto, a **disponibilidade** é um aspecto ainda negligenciado.

Por um lado, a disponibilidade da Internet permanece abaixo, dentre outras, daquela observada na rede de telefonia fixa convencional, que é da ordem de 99,99% [Kuhn, 1997]. Experimentos realizados apontam que a Internet apresenta um índice de disponibilidade na faixa de 95% a 99,6% [Paxson, 1997; Labovitz et al., 1999; Andersen et al., 2001; Dahlin et al., 2003]. Esse índice parece bastante alto, mas mesmo essa aparentemente pequena ocorrência de indisponibilidade é suficiente para minar a confiança dos usuários em aplicações como correio eletrônico (para comunicações realmente importantes, as pessoas em geral ainda recorrem ao telefone em vez da Internet), para causar prejuízos consideráveis para empresas que trabalham com comércio eletrônico e para inviabilizar completamente o uso da rede para sistemas com características de tempo real.

Diversos estudos mostram que os protocolos de roteamento IP estão rotineiramente sujeitos a uma série de problemas capazes de perturbar seriamente comunicações fim a fim, levando à indisponibilidade de serviços. Por exemplo, Paxson [1997] constatou que de 1,5% a 3,3% das rotas enfrentam problemas sérios que levam um longo tempo para serem reparados. Estudos conduzidos por Labovitz com tráfego real descobriram que 10% das rotas estão disponíveis menos de 95% do tempo e que menos de 35% têm disponibilidade superior a 99,99% [Labovitz et al., 1999], e ainda que cerca de 40% das falhas em rotas levam 30 minutos ou mais para serem reparadas (e que o tempo de reparo

tem alta variância, isto é, há rotas que levam muito mais de 30 minutos) [Labovitz et al., 2001]. Em [Dahlin et al., 2003], os autores mostram que 5% das falhas em rotas levam mais de 10.000 segundos (quase três horas) para serem reparadas, sendo que o tempo de reparo pode, em alguns casos, chegar a 100 mil segundos (mais de um dia). Dentre as diversas causas que justificam esses números, as mais citadas incluem a lentidão com que o roteamento global na Internet recupera-se de falhas (o que é necessário para não provocar instabilidades nas tabelas globais de roteamento) e o uso de agregação de informações de roteamento, que é necessária para garantir escalabilidade mas ao mesmo tempo amplifica o efeito de falhas simples que poderiam ser corrigidas com alguma facilidade e rapidez.

Por outro lado, a infra-estrutura de roteamento corre o risco de ser alvo de ataques. Na verdade, o roteamento IP vem sendo apontado como uma vulnerabilidade crítica há mais de 15 anos [Bellovin, 1989; Schneider, 1999]. Isso levou ao desenvolvimento de protocolos de roteamento que não só fossem seguros mas que também pudessem ser implementados na Internet atual, como o S-BGP (*Secure Border Gateway Protocol*) [Kent et al., 2000]. Entretanto, nenhum protocolo com essas características foi amplamente adotado, por razões que incluem [Bellovin et al., 2005]:

- Os custos financeiros envolvidos na adoção de um novo protocolo;
- A falta de um mecanismo claro para a migração gradual dos protocolos antigos para o novo protocolo; e
- O receio de que um novo (e pouco testado) protocolo provoque um aumento na complexidade e na fragilidade da camada de roteamento.

Essas razões, que em última análise se traduzem em uma falta de incentivo para que os provedores de acesso verdadeiramente busquem uma solução para o problema, têm sido suficientemente fortes para impedir que a situação melhore, em que pesem relatos recentes de atividade maliciosa contra roteadores Internet (algo que tende a crescer à medida em que os invasores percebem as potencialidades de ataques nessa camada e adquirem o conhecimento técnico necessário a perpetrar esses ataques). Infelizmente, as perspectivas de que essa situação vá progredir em um futuro próximo não são animadoras [Bellovin et al., 2005].

As redes *overlay*, que são redes virtuais implementadas geralmente sobre a camada IP da Internet, têm sido propostas como um mecanismo para melhorar a disponibilidade das comunicações na Internet [Andersen et al., 2001; Amir e Danilov, 2003; Gummadi et al., 2004]. Entretanto, as experiências que tratam dessa problemática registradas na literatura se concentram em tolerar faltas “benignas” (de natureza acidental), sem considerar as faltas decorrentes de atividade intrusiva. A questão que permeia esta tese é, então, como usar redes *overlay* para oferecer disponibilidade considerando tanto faltas acidentais como intrusões, a um custo aceitável.

1.1 Motivação

Redes *overlay* são redes virtuais que usam a Internet como rede física subjacente e que oferecem funcionalidades e qualidades de serviço distintas. Um grande atrativo das redes *overlay* é que elas

permitem que novos protocolos e mecanismos sejam implementados sem nenhum suporte especial da infra-estrutura existente na rede física, pois todas as novas funcionalidades são implementadas em nível de aplicação pelos nós *overlay*. A agilidade e a flexibilidade oferecidas por essa abordagem trazem vários benefícios: uma rede *overlay* permite que novas funcionalidades sejam adotadas de forma gradual, e que apenas as redes interessadas precisem dar suporte a tais funcionalidades. Embora essas questões possam não ter tanta relevância teórica, na prática elas são determinantes para a viabilidade de uma proposta, conforme demonstra a experiência atual com protocolos seguros de roteamento [Bellovin et al., 2005].

Existem na literatura propostas que usam redes *overlay* visando melhorar a disponibilidade da Internet [Andersen et al., 2001; Gummadi et al., 2004]. Elas exploram a capacidade de ter nós *overlay* que executam funções de roteamento para possibilitar que pacotes sejam encaminhados através de nós intermediários quando o caminho direto entre dois nós se torna indisponível, oferecendo assim um mecanismo rápido de recuperação de faltas. Entretanto, essas redes não são explicitamente projetadas para lidar com comportamento malicioso ou bizantino [Lamport et al., 1982]. Em linhas gerais, elas geralmente presumem que os nós são cooperativos, e que rotas falham perdendo pacotes, devido a fatores como queda ou sobrecarga de nós ou enlaces físicos. A existência de nós maliciosos em uma rede exige que diversas ameaças, contra vários aspectos da arquitetura, sejam levadas em consideração, tais como:

- **Roteamento:** nós bizantinos podem atacar a disponibilidade da rede através do protocolo de roteamento de diversas formas. A mais simples de remediar é quando nós (internos ou externos à rede) tentam injetar pacotes de roteamento falsos como se fossem de outros nós, o que pode ser combatido através de mecanismos de autenticação. Possibilidades mais complexas dizem respeito a nós que geram pacotes de roteamento autênticos com informações falsas, omitindo nós vizinhos ou criando vizinhos inexistentes.
- **Encaminhamento:** quando pacotes devem ser encaminhados por nós intermediários que podem ter sido comprometidos, existem diversas possibilidades de ataque, que incluem a modificação de pacotes em trânsito, o descarte de pacotes ou a sua transmissão por rotas indevidas.
- **Detecção de falhas:** a maioria dos protocolos de detecção de falhas em redes se baseia em alguma espécie de *probe*, que são pacotes de controle usados para verificar se determinado nó permanece ativo e alcançável através da rede. Quando os nós podem ser bizantinos, porém, a utilidade dos *probes* é limitada. Por um lado, é perfeitamente possível que um nó faltoso forneça os *probes* necessários para provar que continua vivo mas se recuse a transportar dados quando solicitado. Outra dificuldade é quando uma falta ou ataque afeta apenas o subsistema responsável pela transmissão e encaminhamento de pacotes de dados (o chamado **plano de dados**), mas não o subsistema responsável pelas informações de controle, como roteamento e monitoramento de falhas (o chamado **plano de controle**).

A idéia de tornar redes de comunicação tolerantes à perda de nós e enlaces é antiga, tendo sido introduzida com os trabalhos pioneiros em comutação de pacotes, na década de 60 [Baran, 1964].

Entretanto, a idéia de tolerar nós que podem se comportar de maneira maliciosa com o intuito de prejudicar o funcionamento da rede só veio a ser explorada bem mais tarde, em 1988 [Perlman, 1988]. Apesar da existência de protocolos de rede tolerantes a faltas bizantinas, a adoção desses protocolos tem sido prejudicada por um número de razões, tais como:

- **Desempenho:** os mecanismos usados por esses protocolos (como algoritmos criptográficos) possuem um custo geralmente alto, que se traduz em perda de desempenho. Como a segurança tem sido um aspecto historicamente ignorado no desenvolvimento da Internet, tendo recebido pouca ênfase até meados da década de 90, nem sempre os usuários estão dispostos a sacrificar o desempenho em prol da segurança.
- **Impacto na infra-estrutura:** muitas propostas exigem que uma porção significativa dos nós da rede suportem os novos mecanismos, incluindo aí os roteadores que sustentam os principais *backbones* Internet. Isso praticamente inviabiliza tais propostas por vários motivos, dentre os quais:
 - exige que os fabricantes de equipamentos de rede implementem os mecanismos, algo que muitas vezes é difícil ou inviável devido à imaturidade e volatilidade das suas especificações;
 - exige que os operadores de rede atualizem seus equipamentos, instalando *software* muitas vezes pouco testado que pode causar instabilidade na rede.

Além das justificativas técnicas citadas acima, com frequência os fabricantes e operadores não têm incentivos econômicos para implementar as mudanças: eles precisam arcar com os custos para que seus clientes (em geral pouco dispostos a pagar a mais por segurança) possam usufruir dos benefícios associados. Essa conjuntura gera um imobilismo causado por uma dependência circular: os operadores adotam apenas os mecanismos implementados pelos fabricantes, e estes resistem a implementar novos mecanismos quando não há demanda técnico-econômica que justifique o investimento no seu desenvolvimento.

Como as redes *overlay* deslocam a implementação das suas funcionalidades para a camada de aplicação, elas não exigem nenhum suporte especial da camada de rede, o que praticamente elimina o impacto sobre a infra-estrutura. Quanto ao desempenho dos protocolos, ele depende do uso criterioso de mecanismos custosos, que devem ser reservados para os casos em que são realmente necessários.

As redes *overlay* ainda apresentam a vantagem de proporcionar flexibilidade na construção da topologia. O estabelecimento de um enlace virtual entre dois nós, por exemplo, depende apenas da existência de uma rota entre os nós na rede subjacente; não existe praticamente nenhum custo envolvido na criação e remoção de enlaces, diferente do que ocorre em uma rede convencional (onde um enlace só pode existir se houver uma infra-estrutura física de suporte). Isso abre a possibilidade de explorar essa flexibilidade para dotar a rede de uma capacidade adaptativa, modificando sua topologia de acordo com a situação.

1.2 Objetivos

O objetivo geral desta tese é propor um serviço de comunicação que ofereça alta disponibilidade, considerando um cenário em que os nós da rede possam ser maliciosos e não se possa confiar que eles irão cooperar uns com os outros. Isso é atingido através do desenvolvimento de uma arquitetura de rede *overlay* tolerante a intrusões. A arquitetura proposta aproveita as características desse tipo de rede para atingir esse objetivo a um custo aceitável.

Os objetivos específicos derivados do supracitado objetivo geral se traduzem inicialmente na proposição de um conjunto de mecanismos tolerantes a intrusões que implementem diversos subsistemas em uma rede *overlay*. Os subsistemas considerados nesta tese e as questões chave concernentes a cada um deles são os seguintes:

- **Roteamento:** como garantir a segurança e a veracidade das informações sobre rotas disponíveis na rede, como escolher rotas funcionais quando os nós que as compõem podem ter sido comprometidos e se comportar de maneira arbitrária;
- **Transmissão e encaminhamento de pacotes:** como garantir que os pacotes transmitidos por um nó de origem cheguem corretamente ao seu destino mesmo que ocorram faltas e intrusões na rede;
- **Deteção de falhas e recuperação de faltas:** como detectar a ocorrência de falhas e intrusões na rede, e como reagir a tais ocorrências.

Esses subsistemas são integrados e implementados na forma de um protótipo de simulação, que permite demonstrar experimentalmente a viabilidade da solução proposta. A solução também é avaliada analiticamente, através de uma análise qualitativa da segurança fornecida e de uma análise de confiabilidade que usa teoria de grafos para determinar o grau de tolerância a faltas e intrusões de uma determinada rede *overlay*.

1.3 Organização do Texto

Esta tese está estruturada em sete capítulos e um apêndice. Este capítulo inicial descreveu o contexto onde o trabalho está inserido, sua motivação e objetivos e um resumo de seus principais resultados.

O capítulo 2 contém uma revisão de conceitos e experiências envolvendo redes *overlay* cujo conhecimento se faz necessário para a compreensão do trabalho desenvolvido. Primeiramente, recapitulam-se os conceitos fundamentais de roteamento na Internet. A seguir são apresentados conceitos de redes *overlay*, e são discutidas em detalhes as principais experiências envolvendo o uso de *overlays* para oferecer confiabilidade a serviços de roteamento. O capítulo encerra com um apanhado de ameaças a que estão sujeitas redes *overlay* que implementam funcionalidades de roteamento.

O capítulo 3 apresenta uma visão geral da rede *overlay* tolerante a intrusões (ROTI) introduzida nesta tese. São discutidas a arquitetura da rede, as premissas adotadas no seu desenvolvimento e os diferentes subsistemas em que são decompostas as suas funcionalidades.

O capítulo 4 discute em detalhes os subsistemas de roteamento e encaminhamento da pacotes da ROTI. Esses subsistemas representam o cerne da funcionalidade da rede, sendo responsáveis pela disseminação das informações sobre a topologia do *overlay* e pela transmissão de dados propriamente dita.

O capítulo 5 apresenta os mecanismos usados na ROTI para detecção de falhas e localização e recuperação de faltas, que incluem o comportamento malicioso de elementos da rede. Este capítulo introduz também um protocolo tolerante a faltas bizantinas para reconfiguração da topologia da rede, uma das contribuições da tese.

O capítulo 6 mostra resultados analíticos e experimentais que demonstram a eficácia e a eficiência da ROTI. A parte analítica consiste em uma análise qualitativa da segurança oferecida pela rede e em uma análise quantitativa do seu grau de tolerância a faltas e intrusões. A parte experimental discute aspectos da implementação da ROTI em um simulador e apresenta resultados obtidos por simulação sobre a confiabilidade e o desempenho da rede.

O capítulo 7 apresenta as nossas conclusões e algumas perspectivas futuras para a continuação deste trabalho.

Finalmente, o apêndice A traz uma revisão de conceitos e terminologia de teoria de grafos que servem de base para a análise do grau de tolerância a faltas e intrusões da rede apresentada no capítulo 6.

Capítulo 2

Redes Overlay: Conceitos e Experiências

Este capítulo apresenta alguns conceitos preliminares que são necessários para a compreensão do restante da tese. Primeiramente, descreve-se de forma resumida o roteamento na Internet, enfocando os aspectos mais importantes do seu funcionamento e as limitações que decorrem da necessidade de prover um mecanismo de roteamento escalável para uma rede com milhões de nós. Em segundo lugar, são discutidos conceitos elementares de redes *overlay* e duas experiências que usam *overlays* para oferecer maior disponibilidade na Internet considerando semânticas de falhas mais simples, e que têm uma relação estreita com este trabalho. Finalmente, é feito um apanhado das ameaças de segurança a que estão sujeitas as redes *overlay*.

2.1 Roteamento na Internet

A função de um protocolo de roteamento é garantir a alcançabilidade entre dois nós de uma rede, sendo portanto a primeira linha de defesa da rede contra faltas em nós e enlaces. Os protocolos de roteamento propagam informações sobre a alcançabilidade de destinos através da rede, o que permite aos roteadores saber por onde encaminhar pacotes de modo que eles fluam de sua origem para seu destino. Quando um roteador detecta uma falha em um de seus enlaces ele atualiza as suas mensagens de roteamento de modo a refletir essa ocorrência; após um período de **convergência de rotas**, todos os roteadores da rede terão estabelecido novas rotas que evitam o enlace faltoso (supondo que haja pelo menos uma rota alternativa que permita contornar essa falta).

A Internet é um conjunto de redes individuais que se comunicam usando o protocolo IP. Essas redes se organizam em **domínios de roteamento**, que são conjuntos de roteadores interconectados sob uma mesma autoridade administrativa; na camada IP, um domínio de roteamento é mais conhecido como **sistema autônomo** (AS — *Autonomous System*). Cada AS tem alocado a si um ou mais blocos (faixas) de endereços IP. Tipicamente, detentores de AS são instituições de grande porte, como provedores Internet de âmbito regional ou nacional, operadores de telefonia e universidades. Usuários domésticos e organizações de menor porte geralmente usam endereços que pertencem ao AS do seu provedor Internet.

Considerando o tamanho atual da Internet — que engloba centenas de milhares de redes e mais de 100 milhões de *hosts* — a necessidade de um protocolo de roteamento escalável torna-se bastante evidente: seria inviável que roteadores mantivessem tabelas de roteamento com milhões de entradas. A solução adotada é o **roteamento hierárquico**. Dentro de um AS utiliza-se um protocolo que troca informações detalhadas de alcançabilidade, e que portanto possui rotas entre quaisquer pontos desse AS; entre os ASs utiliza-se um outro protocolo que usa apenas informações agregadas de alcançabilidade (essencialmente os blocos de endereços alocados a cada AS), e que fornece então rotas de um AS para o outro.

No **roteamento intradomínios** (dentro de um AS) são usados protocolos como RIP (*Routing Information Protocol*) [Hedrick, 1988], OSPF (*Open Shortest Path First*) [Moy, 1998] ou IS-IS (*Intermediate System to Intermediate System*) [ISO/IEC, 2002; Callon, 1990]). Cada protocolo utiliza um algoritmo para o cálculo de rotas, dos quais os mais conhecidos são:

- **Vetor de distâncias:** a tabela de roteamento armazena, para cada destino da rede, o endereço do próximo *hop* e a distância até esse destino; inicialmente, a tabela de um nó contém apenas o próprio nó (com distância zero) e seus vizinhos (cada um deles com distância um). Periodicamente, os nós enviam a seus vizinhos uma cópia da sua tabela de roteamento. Ao receber a tabela de roteamento do nó *i*, o nó *j* verifica se ela traz algum novo destino ou então alguma rota mais curta para um destino já conhecido; se houver, ele modifica sua própria tabela de roteamento, colocando *i* como próximo *hop* para esse destino e usando a distância anunciada por *i* acrescida de um. O RIP é um exemplo de protocolo baseado em vetor de distâncias.
- **Estado de enlaces:** cada nó divulga periodicamente uma lista de seus vizinhos e do custo do enlace com cada um deles (esse custo pode ser estimado a partir de diversas métricas, como a latência ou o tamanho da fila de saída do enlace). Essas atualizações são difundidas para todos os nós da rede através de inundação (*flooding*). Ao receber atualizações de todos os demais nós, cada nó pode construir um grafo valorado representando a topologia da rede, com o custo dos enlaces sendo o peso dos arcos, e usar um algoritmo de cálculo de menor caminho [Dijkstra, 1959] para determinar a melhor rota até os destinos de interesse. Esse algoritmo é considerado superior ao vetor de distâncias, e é usado no OSPF e no IS-IS.

No **roteamento interdomínios** (entre ASs) atualmente é usado o BGP (*Border Gateway Protocol*), versão 4 [Rekhter et al., 2006]. No BGP, os ASs enviam anúncios de rotas declarando que uma determinada rede pode ser alcançada através deles. Esses anúncios são propagados através da rede para todos os roteadores BGP. Após receber os anúncios de rotas, os ASs que participam do protocolo sabem como alcançar outras redes, diretamente ou percorrendo uma seqüência de outros ASs. As decisões de roteamento no BGP são baseadas essencialmente na minimização do número de ASs que um pacote deve atravessar, sem levar em consideração fatores como o desempenho das rotas, mesmo em situações de alta perda de pacotes. No BGP, a escalabilidade é o critério mais importante; ainda assim, uma tabela de roteamento BGP ultrapassa hoje as 180 mil entradas em média [Huston, 2006].

Tratar grandes conjuntos de subredes como se fossem entidades únicas para fins de roteamento global permite ao BGP resumir e agregar quantidades imensas de informações de roteamento em um formato capaz de escalar para centenas de milhões de nós. Em vez de fornecer informações detalhadas de roteamento para cada nó sob sua responsabilidade, um grande provedor dissemina apenas alguns pequenos anúncios de rotas para blocos inteiros de endereços, cada um contendo centenas ou milhares de nós. Embora essa agregação seja necessária para a escalabilidade da rede global, ela significa que os roteadores propagam atualizações apenas para o agregado completo — eles não anunciam falhas que afetam apenas um subconjunto dos nós ou redes dentro desse agregado. Isso faz com que outras partes do sistema de roteamento ignorem uma fração significativa das faltas que acontecem em outros ASs, não podendo assim reagir a elas. Esse problema é amplificado pelo fato de que os roteadores interdomínios podem levar dezenas de minutos para obter uma visão consistente da topologia da rede depois de uma falta, o que pode ser atribuído essencialmente a oscilações nas tabelas de roteamento durante o complexo processo de seleção de caminhos do BGP [Labovitz et al., 2001]. Durante esse período de “convergência atrasada”, a comunicação fim a fim fica prejudicada. Embora parte dos atrasos de convergência possa ser resolvida com mudanças nas implementações existentes de BGP, longos atrasos e oscilações temporárias são uma consequência fundamental do protocolo de roteamento por vetor de caminhos (*path vector*) do BGP.

Os protocolos de roteamento usados na Internet são ditos **proativos**, pois eles objetivam manter, em cada nó que participa do protocolo, informações de roteamento atualizadas para todos os demais nós (ou ASs, no caso do BGP). A principal vantagem desses protocolos é a latência zero, pois depois do período de convergência um nó pode se comunicar com qualquer outro. A maior desvantagem é que protocolos proativos geram um tráfego constante de mensagens de roteamento, independente das rotas disseminadas estarem sendo usadas ou não. Outro problema potencial com protocolos proativos é que em situações em que a topologia de rede se torna instável, com rotas sendo alternadamente criadas e removidas (um fenômeno conhecido como *flapping*), pode haver a geração de um número muito grande de mensagens de roteamento em um curto espaço de tempo.

Uma outra classe de protocolos de roteamento são os protocolos **reativos** ou **sob demanda**, que são muito usados em redes *ad hoc* móveis (MANETs — *Mobile Ad hoc NETWORKS*). Esses protocolos iniciam um processo de descoberta de rotas sempre que um nó de origem deseja enviar pacotes para um destino para o qual ele não possui nenhuma rota apropriada. Exemplos conhecidos de protocolos reativos incluem DSR (*Dynamic Source Routing*) [Johnson et al., 2001] e AODV (*Ad hoc On Demand Distance Vector*) [Perkins e Royer, 1999]. Protocolos reativos possuem geralmente custo mais baixo, pois só há tráfego de roteamento quando uma rota se faz necessária. A redução no custo é tão verdadeira quanto o número de nós que efetivamente se comunicam é pequeno em relação ao total de nós da rede e a topologia da rede evolui constantemente (o que tende a aumentar o número de mensagens de roteamento em um protocolo proativo). Além disso, os nós podem ter tabelas de roteamento menores, uma vez que não precisam armazenar rotas para todos os destinos. Por outro lado, um protocolo reativo impõe uma certa latência quando é necessário descobrir uma rota para um novo destino, o que constitui uma desvantagem.

2.2 Redes Overlay

Uma **rede overlay** é uma rede lógica construída sobre uma rede física existente. Este não é exatamente um conceito novo: a própria Internet foi inicialmente implementada sobre a rede pública de telefonia, e ainda hoje boa parte dos seus dados trafegam por circuitos telefônicos.

A figura 2.1 mostra um exemplo de como uma rede *overlay* se sobrepõe a uma rede física. Cada nó da rede *overlay* é também um nó da rede física. Tipicamente, a rede física é a própria Internet (geralmente a camada IP), e os nós *overlay* são implementados por *hosts*, não por roteadores. Uma conexão entre dois nós da rede *overlay* é chamada de enlace virtual, e corresponde à rota entre os respectivos nós na rede física. Cada nó é responsável por processar e rotear pacotes segundo critérios específicos da rede *overlay*. As conexões entre os nós do *overlay* são implementadas na rede física usando alguma forma de tunelamento (isto é, os pacotes da rede *overlay* são encapsulados em pacotes da rede subjacente), e não necessitam seguir nenhuma topologia predeterminada. Assim como em uma camada de rede típica, as funções principais de uma rede *overlay* são o encaminhamento de pacotes, que determina como os nós da rede processam um pacote em trânsito para que ele chegue ao seu destino, e o roteamento, que é o processo através do qual o conhecimento sobre as diferentes rotas entre nós da rede é calculado, armazenado e disseminado.

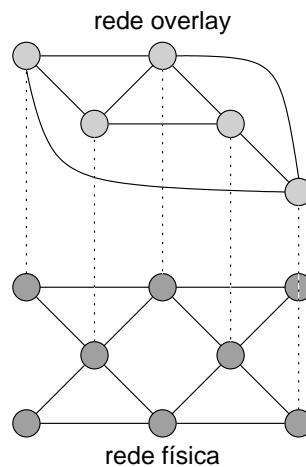


Figura 2.1: Rede *overlay* sobreposta a uma rede física

Recentemente, um número crescente de autores passou a defender a idéia de que a Internet está engessada em razão de seu próprio sucesso: a rede que um dia foi essencialmente acadêmica e por isso afeita a experiências tornou-se uma infra-estrutura cuja estabilidade hoje é crítica para um grande número de pessoas e organizações, e que portanto tornou-se refratária a experimentações fora da camada de aplicação [Peterson et al., 2002]. Exemplos claros desse fenômeno podem ser vistos na lentidão na transição do IPv4 para o IPv6 e na dificuldade de se implementar suporte nativo a IP *multicast* nos *backbones* da rede. As redes *overlay* oferecem uma solução para driblar esse engessamento: em vez de propor alterações aos protocolos fundamentais da Internet, pesquisadores e desenvolvedores podem implementar novas funcionalidades em um *overlay* construído sobre a própria Internet.

Outro benefício da migração de funcionalidades para a camada de aplicação proporcionada pelas redes *overlay* é que os roteadores podem se ocupar apenas de tarefas básicas, como o encaminhamento

de pacotes e troca de informações de roteamento, deixando para os nós *overlay* funções mais custosas, como aquelas que envolvem interpretação do conteúdo dos pacotes. Isso é importante por várias razões, tais como:

- A interpretação do conteúdo de pacotes requer um conhecimento dos protocolos de aplicação, algo que muda com frequência e muitas vezes requer grande capacidade de processamento;
- A maioria dos roteadores implementa as funcionalidades mais usadas em circuitos dedicados de *hardware* (o chamado *fast path*), enquanto que as funções menos usuais (processamento de opções IP, por exemplo) são executadas pelo processador central do roteador (o *slow path*, assim chamado por ser bem mais lento do que o *fast path*);
- Nós *overlay* são implementados em *hosts* que em geral dispõem de capacidades de processamento e armazenamento (volátil e não volátil) bem maiores do que roteadores, o que torna esses nós mais aptos a desempenhar tarefas mais complexas, como operações criptográficas e armazenamento de arquivos.

Redes *overlay* vêm sendo utilizadas há algum tempo. Um dos primeiros exemplos (embora não adotasse o termo *overlay*) é o MBone [Eriksson, 1994], que criava túneis virtuais para conectar redes que suportavam IP *multicast* nativo para permitir o desenvolvimento e a utilização de aplicações com IP *multicast* sem o correspondente suporte em todos os roteadores IP. Outros contextos em que redes *overlay* têm sido adotadas incluem:

- Redes privadas virtuais (VPNs) [Gleeson et al., 2000];
- Sistemas de *multicast* [Jannotti et al., 2000; Chu et al., 2002];
- Serviços de comunicação anônima [Freedman e Morris, 2002; Dingedine et al., 2004];
- Implementações experimentais (*testbeds*) de novos protocolos, como o 6bone [6bone, 2006], para o IPv6, e o QBone [Teitelbaum et al., 1999], para qualidade de serviço (QoS) na Internet2;
- Tabelas *hash* distribuídas (DHTs — *Distributed Hash Tables*), que permitem o armazenamento distribuído de objetos e a sua subsequente localização usando um número pequeno (probabilisticamente limitado) de *hops*. Em uma DHT, cada nó da rede possui poucas entradas na sua tabela de roteamento, o que torna o esquema escalável mas implica uma certa complexidade na construção e manutenção do *overlay* para que ele sempre respeite uma topologia predeterminada. Exemplos conhecidos de DHTs incluem CAN [Ratnasamy et al., 2001], Chord [Stoica et al., 2003], Kademlia [Maymounkov e Mazières, 2002], Pastry [Rowstron e Druschel, 2001] e Tapestry [Zhao et al., 2004]. Aspectos de segurança em DHTs são discutidos em [Castro et al., 2002].

Uma classe particular de redes *overlay* são os *overlays* de roteamento, que visam substituir a camada de roteamento da Internet por um serviço com novas funcionalidades ou de melhor qualidade;

exemplos de *overlays* de roteamento incluem as VPNs e os sistemas de *multicast* citados acima. Este trabalho considera apenas os *overlays* de roteamento, sem se deter nos outros tipos.

As seções 2.2.1 e 2.2.2 a seguir examinam duas experiências de *overlays* de roteamento que enfatizam a disponibilidade do serviço de comunicação. Basicamente, estas experiências usam roteamento através de nós intermediários quando a rota direta entre dois nós não pode ser usada devido a uma falha. A eficácia dessa medida se baseia na observação — confirmada experimentalmente [Teixeira et al., 2003] — de que a redundância existente na camada IP é explorada de forma insuficiente pelos protocolos de roteamento, e no fato de que falhas em uma rota $a-b$ não necessariamente afetam as rotas $a-c$ e $c-b$ para algum nó intermediário c .

2.2.1 RON

O projeto RON (*Resilient Overlay Networks*) [Andersen et al., 2001] consiste em uma rede *overlay* que é capaz de tolerar falhas na camada de roteamento da Internet. A RON utiliza caminhos alternativos através do *overlay* para permitir a comunicação entre nós da rede quando o caminho direto através da Internet encontra-se interrompido por alguma razão (falhas em roteadores, quedas de enlaces, etc.) ou apresenta degradação no seu desempenho.

Cada nó de uma RON monitora o funcionamento e a qualidade das rotas entre ele e os demais nós *overlay*, e usa essa informação para decidir se pacotes devem ser enviados diretamente através da Internet ou se devem ser desviados por outros nós da RON, de modo a otimizar métricas de roteamento específicas da aplicação. Por *default*, para cada enlace virtual (caminho através da Internet) entre cada par de nós da RON são mantidas informações sobre três métricas: latência, taxa de perda de pacotes e *throughput*.¹ Essas métricas são calculadas a partir de amostras (de latência e perda de pacotes) obtidas através da troca de *probes* (mensagens de controle), e são armazenadas localmente em uma base de dados de desempenho. Periodicamente, cada nó envia aos demais $N - 1$ nós um resumo das métricas referentes ao enlace virtual compartilhado com cada um deles. Essas informações de roteamento são disseminadas através da própria RON, o que provê maior confiabilidade.

Cada nó da RON possui também um módulo de gerência de filiação (*membership*), que mantém a lista de nós que fazem parte da rede. Existem dois módulos de gerência de filiação predefinidos. O mais simples deles implementa filiação estática, com a lista de nós da rede sendo carregada a partir de um arquivo armazenado localmente em cada nó. O segundo módulo predefinido implementa filiação dinâmica, usando um protocolo tolerante a faltas de parada (*crash*). A cada cinco minutos, o módulo em cada nó difunde a sua visão da filiação (isto é, a lista de nós que ele acredita pertencerem à rede) para todos os demais nós da rede. Nós que não tenham sido incluídos em nenhuma visão por mais de uma hora são excluídos da rede.

Resultados experimentais demonstram que a RON é capaz de melhorar o desempenho da comunicação fim a fim entre dois nós, e de restabelecer rapidamente (18 segundos em média) a conecti-

¹A métrica de *throughput* corresponde a uma estimativa da taxa de transferência maciça (*bulk transfer*) de dados entre o par de nós usando o protocolo TCP. Ela é derivada a partir de uma equação que leva em consideração as estimativas de latência e taxa de perda de pacotes.

vidade quando ocorrem falhas. Esses resultados demonstram ainda que na grande maioria dos casos a comunicação entre dois nós é efetivada diretamente através do enlace virtual entre eles ou através de apenas um nó RON intermediário, o que permitiu otimizar alguns aspectos da RON partindo da premissa que no máximo um nó intermediário é utilizado.

O monitoramento de $N \times N$ nós permite à RON recuperar-se de falhas na rede com grande rapidez, mas limita a sua escalabilidade a algumas dezenas de nós. O uso de *probes* para determinar a taxa de perdas e a latência de cada enlace virtual é problemático em um cenário mais hostil, pois um nó malicioso pode manipular e falsificar esses dados com alguma facilidade, levando à construção de tabelas de roteamento que não condizem com o serviço de comunicação efetivamente oferecido às aplicações. Além disso, como os nós confiam incondicionalmente nas informações de roteamento fornecidas pelos demais nós, um único nó bizantino pode comprometer todo o funcionamento da rede, anunciando-se como o caminho mais curto para todos os nós e descartando ou corrompendo todo o tráfego subsequentemente por ele encaminhado.

2.2.2 SOSR

O SOSR (*Scalable One-Hop Source Routing*) é outra experiência que demonstra a eficácia do roteamento através de nós intermediários para aumentar a disponibilidade a baixo custo [Gummadi et al., 2004]. Essa experiência foi desenvolvida em duas fases. Na primeira delas, os autores fizeram uma série de medições para caracterizar falhas de rotas na Internet. A metodologia de detecção de falhas em uma rota é a seguinte:

1. *Probes* (segmentos TCP ACK) são enviados a cada 15 segundos;
2. Uma **perda** é registrada quando uma resposta (um segmento TCP RST) não é recebida dentro de três segundos; quando isso acontece, os *probes* passam a ser enviados a intervalos de cinco segundos, e é feita uma tentativa de localizar em que ponto da rota está ocorrendo a perda (usando um mecanismo similar ao *traceroute* [Jacobson, 1988]);
3. Uma **falha** é computada quando três *probes* consecutivos e o *traceroute* iniciado na primeira perda ficam sem resposta.

Essa fase de medições permitiu constatar, entre outras coisas, que 66% das falhas que afetavam rotas para servidores Web populares podiam ser contornadas usando roteamento através de um único nó intermediário; para um conjunto de endereços de destino escolhidos aleatoriamente, esse índice caía para 54%.

Essas observações levaram à segunda fase, de desenvolvimento de uma estratégia para recuperação de falhas de rotas. Essa estratégia, chamada de *random-4*, consiste em retransmitir um pacote perdido através da rota original e de outros quatro nós intermediários escolhidos aleatoriamente. A recuperação é iniciada assim que um único pacote é perdido e abandonada depois de quatro tentativas de transmissão através de cada nó intermediário, depois do que se aguarda que a rota direta se

restabeleça sozinha. O grande atrativo da estratégia *random-4* é o seu baixo custo: o estado que precisa ser mantido é mínimo (apenas no nó de origem), não há troca de mensagens adicionais envolvida na recuperação e o *overhead* de tráfego é baixo — em média 0,048 pacotes adicionais por conexão (cada conexão usa 23 pacotes IP quando não ocorrem falhas), número que sobe para 4,1 pacotes quando ocorre uma falha (pois a recuperação é abandonada depois de quatro tentativas). O *random-4* consegue recuperar cerca de 61% das falhas de rotas para servidores Web populares (do total de 66% de falhas recuperáveis) e de 48% das falhas de rotas para endereços escolhidos aleatoriamente (do total de 54% de falhas recuperáveis).

A exemplo da RON, o SOSR também foi projetado considerando apenas falhas simples. Em particular, o mecanismo de *probing* é vulnerável a respostas forjadas, o que pode fazer com que falhas não sejam detectadas. Mesmo que o mecanismo de detecção de perdas use informações do TCP em vez de *probes* separados, como não há autenticação na comunicação um nó malicioso pode falsificar respostas que indiquem que os pacotes estão sendo entregues no destino. Outro ponto de vulnerabilidade do SOSR é que ele considera apenas rotas com um único nó intermediário; em um ambiente em que os nós podem ser maliciosos, as chances de que essa característica venha a ser atacada aumentam significativamente em relação a ambientes sujeitos apenas a falhas simples.

2.3 Ameaças a *Overlays* de Roteamento

Overlays de roteamento implementam essencialmente o mesmo serviço de uma camada de rede, cujas principais funcionalidades são a troca de informações de roteamento e o encaminhamento de pacotes através da rede. Desta forma, as ameaças de segurança que afetam esses *overlays* podem ser caracterizadas como uma combinação das ameaças que afetam a camada de rede com ameaças específicas a *overlays*. Essas duas classes de ameaças são discutidas a seguir.

A camada de rede está sujeita a ameaças a todas as propriedades de segurança (confidencialidade, integridade e disponibilidade). No que diz respeito à **confidencialidade**, o conteúdo de pacotes pode ser inspecionado pelos nós por onde esses pacotes transitam, e revelado a entidades que não participam da comunicação. Essa ameaça é exacerbada quando um nó malicioso tem acesso a redes cujo suporte na camada de enlace utiliza difusão, como redes Ethernet ou sem fio, pois permite que esse nó capture pacotes mesmo sem fazer parte da rota entre origem e destino.

As ameaças à **integridade** na camada de rede podem ser agrupadas em três grandes classes:

- **Modificação não autorizada:** qualquer nó que encaminhe pacotes para outros nós pode modificar o conteúdo desses pacotes antes de repassá-los, tanto com o propósito de fraude (por exemplo, alterando a conta creditada em uma transferência bancária) ou com o simples propósito de prejudicar a comunicação (neste caso a modificação atentaria contra a disponibilidade da comunicação). Não apenas os mecanismos comumente usados para detectar essas modificações, como *checksums*, são ineficazes contra adversários inteligentes (que podem simplesmente recalculá-las antes de encaminhar o pacote) mas em geral se presume que a corrupção

de pacotes é causada acidentalmente por algum problema eventual nos enlaces subjacentes e não por um adversário ativo, e nenhuma medida é tomada pela camada de rede no sentido de contornar nós ou enlaces que sistematicamente provoquem tal corrupção.

- **Falsificação de mensagens:** na ausência de mecanismos de autenticação de origem, nós maliciosos podem injetar pacotes na rede com endereço de origem forjado, tentando se fazer passar por algum nó correto. Isso pode ser usado, por exemplo, para contornar mecanismos de controle de acesso ou para perpetrar fraudes.
- **Replay de mensagens:** um nó malicioso pode capturar pacotes que fazem parte de uma comunicação e posteriormente reinjetá-los na rede visando confundir ou iludir um dos nós comunicantes para obter algum benefício.

Além da já mencionada corrupção de pacotes, as ameaças à **disponibilidade** em uma camada de rede incluem:

- **Descarte de pacotes:** um nó que encaminha pacotes para outros nós pode descartar pacotes de forma a prejudicar a comunicação. Esse descarte pode ser total ou seletivo, dependendo de fatores como a identidade dos nós comunicantes ou o conteúdo dos pacotes.
- **Sobrecarga:** um nó malicioso pode tentar sobrecarregar um nó com vistas a impedir que outros nós consigam se comunicar com o nó atacado. Essa sobrecarga pode objetivar saturar um enlace crítico de acesso ao nó vítima ou esgotar algum dos seus recursos computacionais, como capacidade de processamento ou memória.
- **Desvio de pacotes:** um ataque semelhante ao descarte de pacotes, o desvio de pacotes (*mis-routing*) consiste no encaminhamento de pacotes por rotas indevidas, de forma que eles levem mais tempo para chegar ao seu destino ou nem cheguem (quando o desvio provoca um laço de roteamento, por exemplo). Para um atacante, uma vantagem do desvio sobre o descarte é que o desvio provoca o consumo de recursos de outros nós que terão que encaminhar pacotes desnecessariamente.

As ameaças de segurança citadas acima se aplicam tanto ao plano de dados (pacotes contendo dados de aplicação) quanto ao plano de controle (pacotes que contêm informações de controle, como roteamento e detecção de falhas), embora o seu impacto (e, por conseguinte, a necessidade de proteção) seja diferente em cada caso: por exemplo, o tráfego de controle tem geralmente menos requisitos de confidencialidade do que o tráfego de dados. Muitas dessas ameaças podem ser combatidas com o uso de mecanismos criptográficos, como cifragem e assinaturas digitais. Isso é particularmente verdadeiro para as ameaças à confidencialidade e à integridade. Entretanto, esses mecanismos costumam não ter a mesma eficácia contra ameaças à disponibilidade.

Com relação a ameaças de segurança específicas a redes *overlay*, não se pode afirmar que exista alguma fraqueza estrutural inerente a tais redes. Entretanto, nós *overlay* são normalmente implementados em *hosts* Internet, que executam uma certa quantidade de *software* sobre sistemas operacionais

de propósito geral. Por essa razão, eles podem ser considerados mais propensos a vulnerabilidades do que roteadores dedicados, que tendem a ter menos *software* executando em um sistema operacional especializado.

Por fim, um fenômeno que normalmente não é considerado referente à segurança mas que efetivamente representa uma ameaça ao funcionamento de uma camada de rede, especialmente à disponibilidade de alguns serviços, são os erros de configuração. Estudos mostram que eles são responsáveis por parte da indisponibilidade observada na rede, e que falhas provocadas por erro humano tendem a durar mais do que falhas devidas a outras causas [Mahajan et al., 2002; Oppenheimer et al., 2003].

2.4 Conclusões do Capítulo

O roteamento é uma funcionalidade central de qualquer rede de comunicação, sendo responsável pelo armazenamento e disseminação de informações topológicas que permitem que dois nós quaisquer dessa rede se comuniquem. A Internet se caracteriza por um subsistema de roteamento escalável e robusto face a falhas simples, tais como quedas de enlaces e roteadores. Entretanto, esse mesmo subsistema por vezes provoca indisponibilidade de regiões da rede devido a ocorrências que são contornáveis, mas diante das quais ele não consegue encontrar rotas alternativas em um espaço de tempo aceitável. Experiências com redes *overlay* demonstram que é possível, através de roteamento na camada de aplicação, resolver tais problemas de indisponibilidade sem que seja necessário modificar os roteadores que compõem o núcleo da rede.

As ameaças de segurança são uma classe de problemas que pode ter implicações sérias para a disponibilidade e estabilidade do subsistema de roteamento, podendo afetar tanto o tráfego de dados na rede como as trocas de mensagens de controle entre os nós que participam do roteamento. Atualmente, nem o roteamento Internet em si e nem as soluções baseadas em *overlays* de roteamento são capazes de resistir a uma série de ameaças de segurança ou à presença de roteadores comprometidos na rede. O próximo capítulo introduz uma arquitetura de rede *overlay* tolerante a intrusões, cujo objetivo é justamente a construção de *overlays* de roteamento que sejam robustos face a ameaças de segurança e outras manifestações de atividade maliciosa na rede.

Capítulo 3

Rede Overlay Tolerante a Intrusões (ROTI): Visão Geral

O capítulo anterior traçou um panorama de alguns aspectos relacionados ao roteamento na Internet, enfocando o problema de indisponibilidade das comunicações e algumas soluções baseadas em redes *overlay* que foram propostas para este problema. Essas soluções deixam uma lacuna importante no que tange a ameaças de segurança, que podem comprometer não apenas a disponibilidade do tráfego como também outras propriedades relevantes, como sua integridade e autenticidade.

Este capítulo apresenta uma visão geral da nossa proposta para preencher essa lacuna. Essa proposta, chamada **rede *overlay* tolerante a intrusões (ROTI)**, é o primeiro *overlay* de roteamento capaz de lidar com ameaças de segurança e elementos comprometidos na rede. O objetivo da ROTI é permitir que nós corretos se comuniquem através do *overlay* mesmo na presença de comportamento malicioso. Esse objetivo é perseguido pelo uso de mecanismos eficazes e que não prejudiquem o desempenho das comunicações de forma indevida.

A seção 3.1 apresenta em maiores detalhes o contexto e os objetivos da ROTI, incluindo potenciais usos da rede e o serviço oferecido às aplicações. A seção 3.2 descreve a arquitetura da ROTI e discute as premissas adotadas no seu desenvolvimento. A seção 3.3 mostra como as funcionalidades da ROTI são divididas em subsistemas com um certo grau de modularidade, o que simplifica a discussão e permite a construção de subsistemas especializados que podem ser facilmente substituídos em caso de necessidade.

3.1 Contexto e Objetivos

O roteamento na Internet está sujeito a diversas fontes de indisponibilidade. Em particular, os roteadores levam um tempo considerável para detectar e remediar falhas ocorridas em outros ASs, conforme discutido na seção 2.1. A dificuldade em interferir na infra-estrutura que suporta o núcleo da rede levou à busca por soluções alternativas para essas limitações, dentre as quais se destacam

as baseadas em redes *overlay*, como as apresentadas na seção 2.2. Embora eficazes para contornar faltas simples, essas propostas são suscetíveis a ameaças de segurança como as citadas na seção 2.3, e à presença de nós comprometidos que assumam um comportamento malicioso na rede. Essa conjuntura motivou o desenvolvimento deste trabalho de tese, cujo resultado é a **rede *overlay* tolerante a intrusões (ROTI)**. A ROTI é a primeira experiência envolvendo tolerância a intrusões em redes *overlay* registrada na literatura conhecida.

A ROTI é um *overlay* de roteamento que torna possível a comunicação entre nós mesmo na presença de faltas e intrusões, sejam elas no *overlay* ou na rede subjacente, rejeitando a premissa de confiança mútua entre os nós. Mais especificamente, deseja-se garantir que a comunicação entre dois nós corretos seja efetivada sempre que houver um caminho no *overlay* formado apenas por nós e enlaces corretos.

O objetivo da ROTI é fornecer um serviço de comunicação com alta disponibilidade para aplicações distribuídas que sejam formadas por grupos de entidades com algum grau de relacionamento preexistente, e não oferecer uma solução universal para o problema de disponibilidade na Internet. Exemplos de aplicações que se beneficiariam da ROTI incluem organizações virtuais, comércio eletrônico e redes privadas virtuais (VPNs) com alta disponibilidade. Outro campo emergente de aplicação é na proteção de infra-estruturas críticas [Stamp et al., 2003; Li et al., 2005], como sistemas de transmissão de energia e de abastecimento de água: a ROTI (eventualmente modificada de forma a oferecer algumas garantias temporais) poderia ser usada para a comunicação entre elementos nesse tipo de infra-estrutura.

A ROTI fornece às aplicações um serviço simples de comunicação ponto a ponto. A *interface* de serviço possui primitivas de comunicação entre os diferentes nós e funções para a configuração e o gerenciamento da rede.

3.2 Descrição da Arquitetura e Premissas

A arquitetura da rede *overlay* tolerante a intrusões é mostrada na figura 3.1. A rede *overlay* é construída usando a Internet como rede física subjacente (*underlay*); outras redes poderiam ser usadas como *underlay* com pouca ou nenhuma modificação na arquitetura. Um nó *overlay* é um nó (tipicamente um *host*) Internet que reside em um sistema autônomo (AS). Um enlace virtual ab é formado pelo caminho Internet entre dois nós *overlay* a e b , ou seja, a seqüência de nós e enlaces Internet percorrida por pacotes IP transmitidos de a para b .

Considera-se que existe um enlace virtual entre cada par de nós *overlay* que possuam alcançabilidade direta recíproca através da Internet: dois nós i e j têm um enlace virtual se i possui uma rota para j e j possui uma rota para i , na rede subjacente. Esses enlaces virtuais podem ser estabelecidos estaticamente (por exemplo, os nós a e b são configurados como vizinhos) ou dinamicamente (por exemplo, os nós a e d se tornam vizinhos quando d se junta ao *overlay*). Os enlaces virtuais são pressupostos bidirecionais, ainda que, por exemplo, a rota percorrida de b para a na rede subjacente não seja o reverso da rota de a para b .

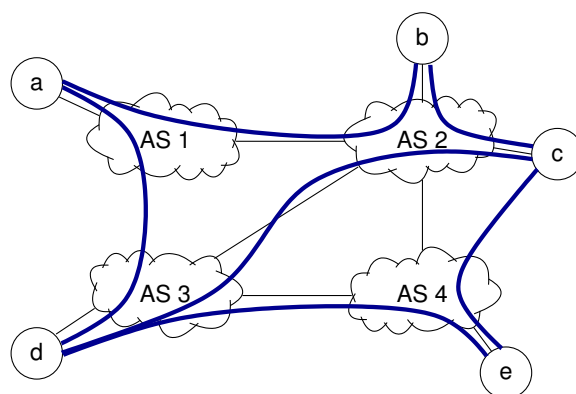


Figura 3.1: Arquitetura de rede overlay tolerante a intrusões

Uma premissa adotada é que a rede se comporta de forma síncrona na maior parte do tempo: é possível definir uma latência máxima δ para um enlace virtual correto (isto é, um enlace virtual concretizado por nós e enlaces corretos na rede subjacente) de tal forma que mensagens transmitidas por este enlace virtual sejam entregues dentro de δ com alta probabilidade. O que esta premissa significa na prática é que, em situações livres de falhas e congestionamento, a maioria dos pacotes chega no seu destino em um tempo δ após a sua transmissão, sendo aceitável que alguns pacotes cheguem atrasados ou nem cheguem (sejam perdidos); o seu propósito é possibilitar o uso de *timeouts* para detectar pacotes perdidos. Cabe observar que, embora a variabilidade das latências verificadas na Internet seja significativa, o uso de *timeouts* permeia todos os protocolos usados na rede sem que isso comprometa sua robustez. Além disso, essa mesma premissa é adotada por todos os trabalhos na literatura que lidam com o problema de roteadores maliciosos [Mizrak et al., 2005].

Em termos de semântica de falhas, considera-se que os nós, sejam eles do *overlay* ou da rede subjacente, podem exibir comportamento malicioso (semântica de falhas bizantinas com autenticação [Lamport et al., 1982]), agindo sozinhos ou formando conluios. Um nó faltoso pode interceptar, modificar ou injetar pacotes, descartar pacotes de forma seletiva, rotear pacotes através de caminhos sub-ótimos ou falsificar informações de controle (por exemplo, dados de roteamento, *probes* e suas respostas, e notificações de erros). Entretanto, considera-se que ele não pode quebrar os mecanismos criptográficos utilizados. Este trabalho não lida explicitamente com nós (do *overlay* ou da rede física) que disparam ataques de negação de serviço (DoS — *Denial of Service*) baseados em sobrecarga de tráfego (*traffic floods*) contra outros nós; não obstante, alguns ataques dessa natureza podem ser tolerados e ter seu impacto reduzido, como será mostrado no capítulo 5.

Uma infra-estrutura de chaves públicas (PKI — *Public Key Infrastructure*) é utilizada para dar suporte aos mecanismos criptográficos usados na ROTI. Cada nó *overlay* possui um certificado vinculando sua chave pública ao seu endereço virtual. Esses certificados não apenas permitem a distribuição segura de chaves públicas, mas eles também são usados para controlar a admissão de nós na rede: apenas nós com certificados válidos podem se juntar à rede. Mecanismos criptográficos de chave pública são usados tanto diretamente (para assinar mensagens, por exemplo) como também para estabelecer chaves simétricas compartilhadas entre pares de nós.

3.3 Subsistemas da ROTI

A funcionalidade de cada nó da rede *overlay* pode ser decomposta em uma série de subsistemas, conforme mostrado na figura 3.2. Essa estrutura é similar à adotada em outras experiências com redes *overlay*, como, por exemplo, a RON [Andersen et al., 2001].

A **API de usuário** provê a *interface* de serviço para as aplicações. Essa *interface* consiste em primitivas de comunicação do tipo *send/receive* e funções para gerenciar os demais subsistemas da rede. O subsistema de **encaminhamento de pacotes** atua na transmissão e recepção de pacotes de dados. Na transmissão, dados de aplicação passados pela primitiva *send* da API de usuário são encapsulados em pacotes da ROTI e enviados para o seu destino. Na recepção, pacotes ROTI são processados e o seu conteúdo é armazenado em *buffers* de leitura, de onde é extraído pela aplicação através da primitiva *receive*. Os pacotes são transmitidos através de rotas armazenadas em um **cache de rotas**, que mantém informações sobre as diversas rotas conhecidas na rede. O subsistema de **roteamento** gerencia os dados contidos no *cache* e é também responsável por trocar informações de roteamento com outros nós. Esse subsistema fornece rotas para destinos específicos e permite que rotas sejam invalidadas, em caráter temporário ou permanente.

Quando há dados a transmitir, o subsistema de encaminhamento consulta o *cache* de rotas para obter uma rota para o destino desejado. Caso não haja uma rota disponível, ele solicita ao subsistema de roteamento que providencie uma rota adequada. Em geral, é usada a rota direta através da Internet entre o nó de origem e o nó de destino; caso essa rota esteja indisponível, é usado roteamento indireto através de nós *overlay* intermediários. Os subsistemas de roteamento e o encaminhamento de pacotes são apresentados no capítulo 4.

O subsistema de **detecção de falhas e recuperação de faltas** monitora o funcionamento da rede, identificando eventuais falhas e atuando sobre os subsistemas de roteamento e encaminhamento para possibilitar a recuperação dos componentes faltosos ou maliciosos. A detecção de falhas se baseia em informações fornecidas pelos subsistemas de encaminhamento e roteamento, que incluem estatísticas sobre o funcionamento da transmissão (índice de perdas, latência na comunicação, etc.). A detecção de falhas e a recuperação de faltas são discutidas no capítulo 5.

Um subsistema que não está mostrado na figura é o de **gerência criptográfica**, que é o responsável por armazenar e gerenciar todo o material criptográfico usado pelos demais subsistemas. Mais especificamente, a gerência criptográfica se encarrega de validar e guardar as chaves públicas dos outros nós da rede, obtidas de seus certificados digitais, e de manter as chaves simétricas que são estabelecidas dinamicamente com esses nós. Além disso, esse subsistema implementa os algoritmos criptográficos utilizados para cifragem, assinaturas digitais e autenticação de mensagens, oferecendo essas operações como um serviço aos outros subsistemas do nó. Os aspectos de construção deste subsistema, embora importantes do ponto de vista prático, são amplamente conhecidos [Schneier, 1996; Ferguson e Schneier, 2003], de modo que ele não é discutido em maiores detalhes nesta tese.

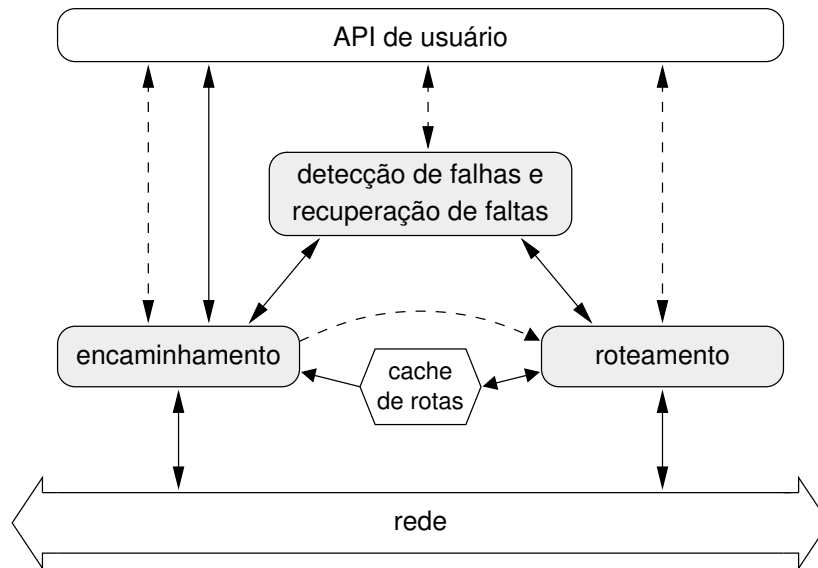


Figura 3.2: Subsistemas em um nó ROTI. Linhas contínuas representam fluxos de dados, e linhas tracejadas representam fluxos de controle.

3.4 Conclusões do Capítulo

Este capítulo apresentou uma visão geral da ROTI, um *overlay* de roteamento tolerante a intrusões. A ROTI possui dois objetivos básicos, sendo que o objetivo primário é possibilitar a comunicação entre nós corretos mesmo que haja elementos maliciosos no *overlay* ou na rede subjacente, e o objetivo secundário é tentar atingir o objetivo primário de forma eficiente, minimizando o uso de mecanismos que prejudiquem significativamente o desempenho da rede.

A decomposição funcional adotada na ROTI favorece a modularidade da arquitetura, permitindo que subsistemas individuais sejam eventualmente substituídos por outros que ofereçam as mesmas *interfaces* de operação. Assim, por exemplo, seria possível usar um protocolo de roteamento diferente causando uma mínima interferência nos demais subsistemas.

Os capítulos a seguir detalham os principais aspectos da arquitetura ROTI, começando pelos subsistemas de roteamento e encaminhamento de pacotes.

Capítulo 4

Roteamento e Encaminhamento de Pacotes na ROTI

O objetivo de uma camada de rede — e, por conseguinte, de um *overlay* de roteamento — é proporcionar a comunicação fim a fim entre os nós da rede, mesmo que eles não estejam diretamente conectados entre si através de um enlace. Para que esse objetivo seja atingido, existem duas funcionalidades básicas envolvidas: roteamento e encaminhamento de pacotes. O roteamento se encarrega de computar, armazenar e disseminar informações sobre a topologia da rede e as diferentes rotas passíveis de utilização nessa topologia. O encaminhamento de pacotes, por sua vez, é o responsável por efetivamente transmitir dados através da rede, se apoiando para isso nas informações de alcançabilidade fornecidas pelo roteamento.

Este capítulo examina como essas funcionalidades chaves podem ser implementadas em uma rede *overlay* tolerante a intrusões. A possibilidade de que os nós do *overlay* tenham sido comprometidos e se comportem de maneira arbitrária constitui um desafio a essa implementação, especialmente quando o impacto provocado pelos mecanismos de proteção adotados no desempenho da solução é levado em consideração.

Os subsistemas de roteamento e encaminhamento de pacotes na ROTI são guiados pelos seguintes princípios. A transmissão de pacotes na ROTI usa, sempre que possível, um enlace virtual direto entre os nós de origem e destino. A rota formada apenas por um enlace virtual é chamada de **rota direta**. Quando a rota direta não está disponível, os pacotes são encaminhados por nós intermediários, usando uma **rota indireta**. A ROTI usa **roteamento na origem** (*source routing*), com o nó de origem escolhendo uma rota para o destino que é incluída no cabeçalho dos pacotes. Quando se lida com comportamento malicioso, o roteamento na origem tem algumas vantagens sobre decisões de roteamento *hop a hop*. A primeira é que ele limita a possibilidade de que os pacotes sejam desviados de suas rotas, pois qualquer pacote encaminhado por uma rota diferente da especificada será notado por um nó correto. A segunda é que o roteamento na origem permite que um nó evite rotas que considera faltosas: como nós maliciosos podem descartar pacotes de forma seletiva, o mesmo caminho pode se comportar de maneira diferente dependendo de fatores como os endereços de origem e destino dos pacotes. A terceira é que o roteamento na origem acomoda com naturalidade nós que possuem

diferentes visões da topologia, pois toda a rota é computada com base na visão de um único nó. Por fim, o roteamento na origem permite que os nós usem estratégias diferentes para calcular as rotas, o que confere flexibilidade adicional em relação às decisões *hop a hop*. A principal desvantagem do roteamento na origem é que ele não permite o re-roteamento de pacotes; caso haja algum problema com uma rota selecionada, um nó intermediário não pode enviar os pacotes afetados por uma rota alternativa, cabendo ao nó de origem fazer os ajustes necessários — o que envolve a latência de propagação das informações de eventuais falhas — ao retransmitir os pacotes perdidos.

A seção 4.1 apresenta o protocolo de roteamento adotado na ROTI, enquanto a seção 4.2 discute duas estratégias que podem ser usadas para o encaminhamento de pacotes. A seção 4.3 compara as soluções propostas com experiências relacionadas na literatura.

4.1 Roteamento na ROTI

O objetivo primordial do roteamento na ROTI é garantir a integridade e a autenticidade das informações de roteamento. Mais especificamente, deseja-se assegurar que mensagens de roteamento sejam consideradas válidas apenas quando tenham sido geradas por nós *overlay* e não tenham sido manipuladas por elementos maliciosos, sejam eles do *overlay* ou da rede subjacente. O protocolo de roteamento em si não tenta garantir que as informações de alcançabilidade sejam inteiramente corretas, visto que esta é uma tarefa virtualmente impossível; ainda assim, ele incorpora mecanismos para dificultar a disseminação de informações incorretas. Como na maioria dos protocolos conhecidos, a confidencialidade das informações de roteamento não é considerada prioritária e, portanto, não é tratada pelo protocolo.

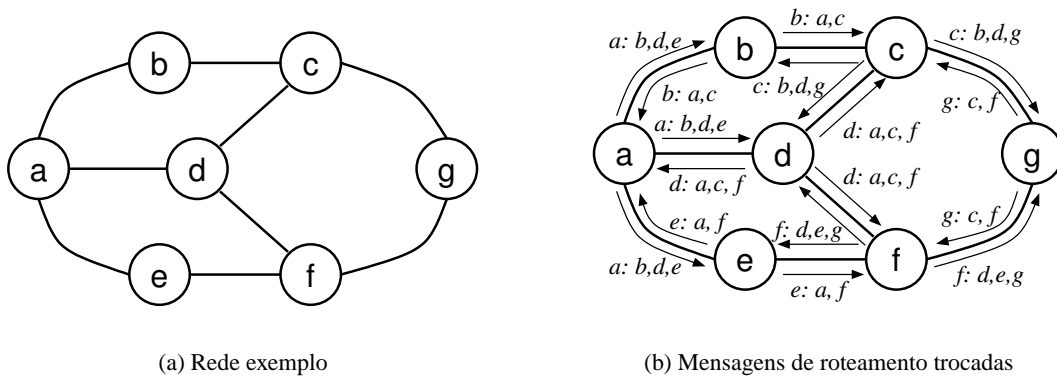
Para encontrar rotas no *overlay* foi desenvolvido um novo protocolo de roteamento que combina roteamento proativo e reativo. Basicamente, cada nó envia periodicamente a seus vizinhos uma lista de seus nós adjacentes (diferentemente de protocolos tradicionais de roteamento proativo, essa lista não é propagada pelos vizinhos). Estas difusões locais permitem que os nós *overlay* aprendam sobre todos os destinos que eles conseguem alcançar usando rotas com um único intermediário, chamadas aqui de **rotas (indiretas) simples**. Quando um nó deseja enviar pacotes para um destino que não é alcançável usando rotas diretas ou simples, ele usa roteamento sob demanda para encontrar rotas mais longas para esse destino. Cada nó mantém um *cache* de rotas local que é consultado a cada vez que o nó precisa de uma rota para algum destino. As seções seguintes descrevem as duas fases do protocolo de roteamento em detalhes, e de que forma as rotas são selecionadas no *cache*.

4.1.1 Fase de Roteamento Proativo

Experiências anteriores com disponibilidade e redes *overlay* demonstram que uma grande maioria dos nós podem ser alcançados diretamente ou usando um único nó intermediário [Savage et al., 1999; Andersen et al., 2001; Gummadi et al., 2004]. Isso acontece porque, na maioria dos casos, faltas de natureza não maliciosa que afetam uma rota são circunscritas a uma determinada região da rede, e

o uso de rotas alternativas que evitem a região afetada é suficiente para contornar o problema. Com base nessa observação, a ROTI usa roteamento proativo para possibilitar aos nós aprenderem sobre todos os destinos alcançáveis dentro de um raio de até dois hops. Nessa fase proativa, cada nó difunde localmente uma mensagem ROUTE-UPDATE contendo sua lista de vizinhos. Quando um nó recebe um ROUTE-UPDATE, ele atualiza todas as informações de rotas simples que passam pelo emissor dessa mensagem no seu *cache*, o que pode envolver a inclusão de novas rotas e a remoção de rotas antigas. Os ROUTE-UPDATES são reenviados periodicamente a não ser que haja alguma alteração na lista de vizinhos, o que faz com que um novo ROUTE-UPDATE contendo a lista modificada seja enviado.

A figura 4.1 mostra um exemplo de execução do protocolo proativo. A topologia considerada é apresentada na figura 4.1(a). A figura 4.1(b) mostra as mensagens de roteamento geradas pelo protocolo; um ROUTE-UPDATE é representado na forma $n: v_1, \dots, v_m$, onde n é o nó emissor do ROUTE-UPDATE e v_1, \dots, v_m é a sua lista de vizinhos. A figura 4.1(c) mostra como ficam os *caches* de rotas de cada nó da rede após as trocas de mensagens da figura 4.1(b) (para economizar espaço, os vizinhos de cada nó foram mostrados em um única linha, mas o *cache* não faz distinção entre rotas diretas e indiretas).



(a) Rede exemplo

(b) Mensagens de roteamento trocadas

Nó	Cache de rotas
a	vizinhos: b, d, e c: bc, dc f: df, ef
b	vizinhos: a, c d: ad, cd e: ae g: cg
c	vizinhos: b, d, g a: ba, da f: df, gf
d	vizinhos: a, c, f b: ab, cb e: ae, fe g: cg, fg

Nó	Cache de rotas
e	vizinhos: a, f b: ab d: ad, fd g: fg
f	vizinhos: d, e, g a: da, ca c: dc, gc
g	vizinhos: c, f b: cb d: cd, fd e: fe

(c) Caches de rotas após as mensagens de roteamento

Figura 4.1: Exemplo de execução do protocolo de roteamento proativo

A recepção de ROUTE-UPDATES serve também como confirmação de quais vizinhos continuam

ativos e alcançáveis. Entretanto, faltas transitórias podem fazer com que uma atualização seja ocasionalmente perdida, sem que isso signifique necessariamente que o nó deva ser considerado inalcançável. Para acomodar tais faltas, um vizinho só é considerado inalcançável quando U_{thr} atualizações consecutivas não são recebidas. Quando isso acontece, o nó registra esse vizinho como inalcançável e o exclui de seus próprios ROUTE-UPDATES. Caso o vizinho volte a ser alcançável, o nó espera receber $U'_{thr} < U_{thr}$ atualizações consecutivas antes de inseri-lo novamente nos seus ROUTE-UPDATES. O objetivo de esperar um tempo antes de incluir um vizinho anteriormente inalcançável nas atualizações é dar mais estabilidade às rotas, evitando que nós com alcançabilidade intermitente sejam anunciados como possíveis intermediários.

Para proteger contra ataques de *replay*, cada ROUTE-UPDATE contém um número de seqüência seq , que é incrementado cada vez que uma atualização é emitida. Na prática, porém, números de seqüência não podem crescer indefinidamente, pois são transmitidos em um campo de tamanho limitado contido no cabeçalho dos pacotes. Uma dificuldade que isso acarreta é como lidar com a necessidade de reaproveitar números de seqüência (isto é, reiniciar a contagem), que surge quando o valor máximo representável é atingido ou o nó é reinicializado. A solução adotada na ROTI é enviar um ROUTE-UPDATE contendo $seq = 0$ e um *timestamp* do instante de envio da mensagem. Os nós que recebem esse ROUTE-UPDATE removem de seus *caches* todas as rotas simples que passam pelo seu emissor. Os *caches* são preenchidos por uma atualização subsequente com $seq > 0$. Um ROUTE-UPDATE é aceito como válido se o seu seq é maior do que o anterior ou se $seq = 0$ e o *timestamp* é maior do que o *timestamp* anterior emitido pelo mesmo nó.

Os ROUTE-UPDATES são assinados digitalmente como forma de garantir sua autenticidade e evitar que sejam forjados. Cabe observar que assinar uma mensagem de roteamento oferece proteção apenas contra elementos externos que tentam se passar por nós do *overlay*; isso não evita que um nó *overlay* envie atualizações contendo dados inválidos, uma ameaça legítima quando os nós podem ter sido comprometidos (a seção 4.1.3 discute como dados de roteamento falsos são tratados).

Uma preocupação com o roteamento proativo são os problemas decorrentes da recepção de um grande número de atualizações de rotas em curtos espaços de tempo. Um número excessivo de mensagens de roteamento pode esgotar algum recurso computacional de um nó, como memória ou capacidade de processamento. Por exemplo, um roteador pode consumir toda a sua capacidade de processamento verificando assinaturas digitais, uma operação computacionalmente custosa. Embora o custo de geração de uma assinatura digital seja maior que o custo de verificação, um nó malicioso poderia simplesmente enviar uma grande quantidade de mensagens repetidas que teriam de ser processadas individualmente por um nó correto. Um excesso de atualizações pode ser também provocado por flutuações nas informações de alcançabilidade (*flapping*). Para proteger contra esses problemas, a frequência de emissão de atualizações de rotas na ROTI é limitada: um nó não deve gerar ROUTE-UPDATES a intervalos menores do que Δ_{upd} . Se a topologia ao redor de um nó muda a uma taxa maior do que essa, este nó deve esperar até Δ_{upd} depois do envio da sua última atualização antes de enviar uma nova. Para fazer valer essa limitação, os nós podem descartar ou — se houver espaço disponível — armazenar em um *buffer* a última atualização recebida menos de Δ_{upd} antes da atualização anterior. Essa restrição não se aplica quando o número de seqüência é reiniciado, conforme explicado anteriormente.

Este protocolo é relativamente eficiente, já que cada nó envia $O(n)$ cópias de uma atualização (uma para cada vizinho), o que corresponde a $O(n^2)$ atualizações para o total da rede. Para fins de comparação, isso é uma ordem de grandeza menor do que o custo de um protocolo tradicional de estado de enlaces. Isso ocorre porque em protocolos de estado de enlaces as atualizações de roteamento são enviadas para todos os nós da rede através de inundação, mecanismo que gera na ordem de $O(n^2)$ cópias de cada mensagem, enquanto que no protocolo discutido acima elas são difundidas apenas localmente (isto é, o nó que recebe uma atualização não a repassa aos seus vizinhos como em um protocolo de estado de enlaces).

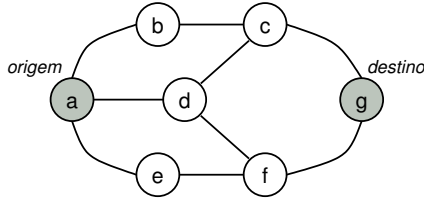
4.1.2 Fase de Roteamento Reativo

Embora a maioria dos nós possa ser atingida usando rotas diretas ou indiretas simples, alguns nós só podem ser alcançados usando rotas mais longas. Isso é mais provável de acontecer quando existe um grande número de nós faltosos descartando ou desviando pacotes. Portanto, para lidar com essas situações, quando um nó de origem não consegue alcançar um destino usando uma rota simples ele inicia um protocolo reativo de descoberta de rotas.

O protocolo de descoberta de rotas usado na ROTI é inspirado no DSR [Johnson et al., 2001], um protocolo de roteamento sob demanda para redes *ad hoc* móveis. A operação básica do protocolo reativo da ROTI é a seguinte: um nó s que deseja obter uma rota para um nó t constrói uma mensagem ROUTE-REQUEST contendo o endereço de s , o endereço de t , um número de seqüência e a rota acumulada entre s e t (que inicialmente contém apenas o endereço de s). Esse ROUTE-REQUEST é assinado e enviado para todos os vizinhos de s . Cada nó que recebe um ROUTE-REQUEST acrescenta o seu próprio endereço à rota acumulada e põe a sua assinatura à mensagem, que é então enviada para cada um dos seus vizinhos que não faz parte da rota acumulada. Quando um ROUTE-REQUEST chega ao nó de destino t , este adiciona seu endereço e assinatura à rota acumulada, que é então usada para construir uma mensagem ROUTE-REPLY que é enviada de volta para s através da rota acumulada reversa. Quando o nó de origem s recebe o ROUTE-REPLY, ele extrai a rota acumulada até o destino e verifica as assinaturas, adicionando a rota ao seu *cache* se as assinaturas estiverem corretas e se a rota for **disjunta** das outras rotas para o mesmo destino já no *cache* (duas rotas são consideradas disjuntas se não possuem nós intermediários em comum — apenas o nó de origem e o de destino coincidem). A figura 4.2 ilustra um processo de descoberta de rotas usando esse protocolo.

A confiança que pode ser atribuída às rotas obtidas através desse protocolo está ligada à idéia da **rota acumulada**. Uma rota acumulada é formada por uma cadeia de endereços de nós e assinaturas digitais. À medida em que um ROUTE-REQUEST vai sendo propagado na rede, cada nó que o recebe acrescenta à rota acumulada o seu endereço e uma assinatura digital cobrindo todo o conteúdo da requisição. Dessa forma, a rota acumulada oferece uma cadeia de assinaturas que atestam quais nós são percorridos pela requisição. Como a cadeia de assinaturas é verificada em cada nó antes da requisição ser processada, qualquer modificação na seqüência de nós pode ser detectada. Além disso, caso um nó faltoso não se inclua na rota acumulada, isso é detectado pelo nó correto subsequente (o último nó da rota acumulada não corresponde ao nó que enviou a mensagem).

A rota acumulada, porém, não é eficaz contra um **ataque de supressão**, que pode ser descrito como se segue. Seja uma rota acumulada $n_0n_1 \cdots n_m$, sendo n_0 o nó de origem e n_m o nó de destino. Um nó faltoso n_k , $1 < k < m$, pode suprimir dessa rota acumulada uma subrota $n_i \cdots n_j$ ($1 < i < j < k$) sem que isso seja detectado nem pela origem nem pelo destino. Isso acontece porque não há vinculação entre um nó da rota acumulada e o seu predecessor ou sucessor. O único nó capaz de detectar um problema com a rota alterada $n_0 \cdots n_{i-1}n_{j+1} \cdots n_k$ é n_{i-1} , que descartará pacotes enviados por essa rota por não ter um link virtual com n_{j+1} (supondo que n_{i-1} e n_{j+1} não sejam vizinhos). Para combater esse ataque, a rota acumulada poderia ser ampliada para incluir, além do endereço do nó que propaga a requisição, o endereço do seu sucessor, isto é, do nó para o qual essa requisição será propagada. A desvantagem dessa solução é que ela multiplica o custo de processamento das requisições: um nó que propaga um ROUTE-REQUEST para d vizinhos gera d rotas acumuladas distintas (uma para cada sucessor), o que implica d assinaturas digitais (em vez de uma, como no esquema original). Uma solução intermediária seria incluir na rota acumulada a lista de sucessores, o que tornaria o ataque de supressão ineficaz mas aumentaria o consumo de banda por mensagem de roteamento.



(a) Rede exemplo

a : ROUTE-REQUEST(a, g, a) $\rightarrow b, d, e$
 b : ROUTE-REQUEST(a, g, ab) $\rightarrow c$
 d : ROUTE-REQUEST(a, g, ad) $\rightarrow c, f$
 e : ROUTE-REQUEST(a, g, ae) $\rightarrow f$
 c : ROUTE-REQUEST(a, g, abc) $\rightarrow d, g$
 ROUTE-REQUEST(a, g, adc) $\rightarrow b, g$
 f : ROUTE-REQUEST(a, g, adf) $\rightarrow e, g$
 ROUTE-REQUEST(a, g, aef) $\rightarrow d, g$
 d : ROUTE-REQUEST($a, g, abcd$) $\rightarrow f$
 ROUTE-REQUEST($a, g, aefd$) $\rightarrow c$
 g : ROUTE-REPLY($a, g, abcg$) $\rightarrow a$ via $gcba$
 ROUTE-REPLY($a, g, adcg$) $\rightarrow a$ via $gcda$
 ROUTE-REPLY($a, g, adfg$) $\rightarrow a$ via $gfda$
 ROUTE-REPLY($a, g, aefg$) $\rightarrow a$ via $gfea$
 f : ROUTE-REQUEST($a, g, abcdf$) $\rightarrow e, g$
 c : ROUTE-REQUEST($a, g, aefdc$) $\rightarrow b, g$
 g : ROUTE-REPLY($a, g, abcdfg$) $\rightarrow a$ via $gfdcba$
 ROUTE-REPLY($a, g, aefdcg$) $\rightarrow a$ via $gcdfea$

(b) Mensagens de roteamento trocadas

Figura 4.2: Exemplo de descoberta de rotas com o protocolo reativo básico. O nó a é a origem, e o nó g é o destino. À esquerda é mostrada a topologia do *overlay*, e à direita são mostradas as mensagens de roteamento geradas pelo ROUTE-REQUEST original.

O armazenamento em *cache* de todas as rotas disjuntas recebidas é uma melhoria que foi introduzida em nosso trabalho (a maioria dos protocolos de roteamento sob demanda armazenam apenas a melhor rota recebida). Isso possibilita que, quando uma nova rota é solicitada porque a rota atual se torna faltosa, o subsistema de roteamento forneça imediatamente uma rota disjunta alternativa, eliminando a latência inicial geralmente imposta pelo processo de descoberta. O protocolo de descoberta de rotas é então reexecutado apenas quando o *cache* de rotas para um dado destino está vazio.

O protocolo conforme descrito acima é ideal do ponto de vista da diversidade de rotas: ele retorna

todas as rotas funcionais entre dois nós, dando máxima escolha para um nó de origem que precisa escolher uma rota para um destino. Infelizmente, como o número de rotas em uma rede cresce exponencialmente com o tamanho da rede, o protocolo pode ter desempenho sofrível, especialmente em *overlays* com alta conectividade. Em vista disso foram acrescentadas algumas modificações ao protocolo básico, que podem reduzir a diversidade de rotas mas melhoram o seu desempenho. Algumas dessas modificações também estão disponíveis no DSR, mas existem melhorias específicas ao nosso trabalho.

A modificação mais importante é permitir que os nós intermediários respondam a requisições de rotas a partir dos seus *caches*. Ela funciona da seguinte maneira: quando um nó intermediário x recebe um ROUTE-REQUEST para um nó t e x possui uma ou mais rotas para t no seu *cache*, x acrescenta seu endereço e sua assinatura à rota acumulada mas, em vez de enviar o ROUTE-REQUEST para os seus vizinhos, ele envia para s uma mensagem CACHE-REPLY assinada, contendo os endereços de origem (s) e de destino (t) e o número de seqüência contidos na requisição, o endereço de x , a rota $s-x$ acumulada na requisição e a lista de rotas disjuntas $x-t$ presente no *cache* de x . Ao receber um CACHE-REPLY, o nó de origem s verifica a cadeia de endereços e assinaturas na rota acumulada e combina essa rota $s-x$ com as rotas disjuntas $x-t$, tentando encontrar uma rota combinada que seja disjunta das rotas para t já no *cache* (e que possa, portanto, ser adicionada a este). O funcionamento do CACHE-REPLY é ilustrado na figura 4.3.

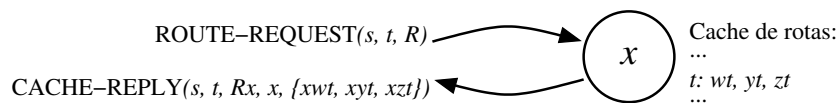


Figura 4.3: Roteamento reativo: funcionamento de CACHE-REPLY. R é a rota acumulada até o predecessor de x .

Outra modificação que foi introduzida foi a limitação do comprimento das rotas. Uma mensagem ROUTE-REQUEST possui um campo MAXLEN que pode ser usado para limitar o comprimento das rotas que serão aceitas como resposta; os nós intermediários descartam todas as requisições cuja rota acumulada é mais longa do que MAXLEN (se MAXLEN = 0, o comprimento da rota não é verificado). Esta modificação se baseia na observação de que rotas mais longas têm maior probabilidade de falharem (pois atravessam um maior número de nós e enlaces), e é bastante eficiente no controle da disseminação de ROUTE-REQUESTS. Por outro lado, ela diminui a diversidade de rotas, já que os nós terão menos rotas para escolher quando forem enviar pacotes. Uma solução de compromisso poderia ser começar o processo com um valor baixo para MAXLEN, e incrementá-lo sucessivamente (até um limite máximo) à medida em que não fossem encontradas rotas para o destino desejado.

A última melhoria incluída no protocolo de descoberta de rotas é o *caching* oportunista de rotas. Essa modificação permite que nós intermediários adicionem a seus *caches* informações sobre rotas obtidas de pacotes de roteamento que eles encaminham para outros nós,¹ na esperança de que essas informações possam ser úteis no futuro e evitem descobertas de rotas desnecessárias. Quando um nó

¹ Seria igualmente possível usar as informações sobre rotas contidas em pacotes de dados, mas optou-se por não fazê-lo para não impor um *overhead* ao encaminhamento de pacotes. O *caching* oportunista envolvendo pacotes de dados só ocorre em uma única situação: quando a rota reversa de um pacote recebido é armazenada como uma rota do destino para a origem.

intermediário x recebe um pacote de roteamento, ele tenta adicionar uma ou mais rotas ao seu *cache* de acordo com o tipo do pacote:

- Para ROUTE-REQUESTS, a rota acumulada da origem até x pode ser revertida e usada como uma rota de x para a origem. Por exemplo, se x recebe ROUTE-REQUEST($s, t, sn_1 \cdots n_k$), onde $sn_1 \cdots n_k x$ é a rota acumulada $s-x$, então $xn_k \cdots n_1 s$ é uma rota $x-s$ candidata a inclusão no *cache*.
- Para ROUTE-REPLYS, se a porção da rota acumulada que vai de x até o nó de destino for disjunta de outras rotas para o destino já presentes no *cache*, essa subrota pode ser adicionada. Por exemplo, se x recebe ROUTE-REPLY($s, t, sRxn_1 \cdots n_k t$), então $xn_1 \cdots n_k t$ é uma rota $x-t$ candidata a inclusão no *cache*.
- Para CACHE-REPLYS, há pelo menos duas rotas candidatas a inclusão no *cache*. A primeira é uma rota do nó x até o emissor e do CACHE-REPLY, e a segunda é uma rota de x até o destino passando por esse emissor (que pode ou não conter a primeira candidata). Por exemplo, se x recebe CACHE-REPLY($s, t, e, R_{sx}xn_1 \cdots n_k e, \mathcal{R}_{et}$), onde R_{sx} representa a rota acumulada $s-x$; $xn_1 \cdots n_k e$ representa a rota acumulada entre x e o emissor e do CACHE-REPLY; e \mathcal{R}_{et} representa um conjunto de rotas disjuntas $e-t$, então:
 - (i) $xn_1 \cdots n_k e$ é uma rota candidata para $x-e$;
 - (ii) a concatenação da melhor rota $x-e$ disponível em *cache* com cada uma das rotas $R \in \mathcal{R}_{et}$ gera um conjunto de rotas candidatas para $x-t$.

O protocolo de roteamento reativo, conforme descrito até aqui, é apresentado nos algoritmos 4.1 a 4.4; a tabela 4.1 descreve as primitivas usadas nesses algoritmos. O algoritmo 4.1 mostra como o processo de descoberta de rotas é iniciado, com a construção do ROUTE-REQUEST; a variável *myId* sempre representa o endereço do nó que executa o algoritmo. O algoritmo 4.2 na próxima página mostra como é processado um ROUTE-REQUEST recebido do nó x . O processamento de ROUTE-REPLYS é apresentado no algoritmo 4.3 na página seguinte, e o de CACHE-REPLYS no algoritmo 4.4 na página 31.

A figura 4.4 na página seguinte mostra um exemplo de descoberta de rotas usando esse protocolo. Comparando com o exemplo da figura 4.2 na página 27, o protocolo completo gera 7 mensagens de roteamento contra 27 do protocolo básico, uma redução de quase 75%.

Algoritmo 4.1 Protocolo de roteamento reativo: início da descoberta de rotas

```

1:  $seq \leftarrow 0$ 
2: procedure REQUEST-ROUTE(in  $dst$ : NodeId)
3:    $seq \leftarrow seq + 1$ 
4:    $accRoute \leftarrow \langle myId, \text{SIGN}(\text{ROUTE-REQUEST}, myId, dst, seq, maxLen, myId) \rangle$ 
5:    $req \leftarrow \langle \text{ROUTE-REQUEST}, myId, dst, seq, maxLen, accRoute \rangle$ 
6:   send  $req$  to all neighbors
7: end procedure

```

Algoritmo 4.2 Protocolo de roteamento reativo: processamento de ROUTE-REQUEST

```

1: upon receiving  $req = \langle \text{ROUTE-REQUEST}, src, dst, seq, maxLen, accRoute \rangle$  from  $x$  do
2:   if VERIFY-RREQ( $req$ ) then
3:      $returnRoute \leftarrow \text{REVERSE}(req.accRoute.nodes)$ 
4:     COSIGN( $req.accRoute, myId$ )
5:     if  $req.dst = myId$  then
6:        $sig \leftarrow \text{SIGN}(\text{ROUTE-REPLY}, req.src, req.dst, req.seq, returnRoute, req.accRoute)$ 
7:        $rep \leftarrow \langle \text{ROUTE-REPLY}, req.src, req.dst, req.seq, returnRoute, req.accRoute, sig \rangle$ 
8:        $n \leftarrow \text{NEXTHOP}(returnRoute)$ 
9:       send  $rep$  to  $n$ 
10:    else
11:       $Rset \leftarrow \text{GETALLROUTES}(req.dst)$ 
12:      if  $Rset \neq \emptyset$  then
13:         $sig \leftarrow \text{SIGN}(\text{CACHE-REPLY}, req.src, req.dst, req.seq, myId, returnRoute, req.accRoute,$ 
14:           $Rset)$ 
15:         $crep \leftarrow \langle \text{CACHE-REPLY}, req.src, req.dst, req.seq, myId, returnRoute, req.accRoute, Rset,$ 
16:           $sig \rangle$ 
17:         $n \leftarrow \text{NEXTHOP}(returnRoute)$ 
18:        send  $crep$  to  $n$ 
19:      else
20:        for all neighbor  $n$  such that  $n$  is not in  $req.accRoute$  do
21:          send  $req$  to  $n$ 
22:        TRYCACHE( $req.src, returnRoute$ ) { opportunistic caching }
23:    end do

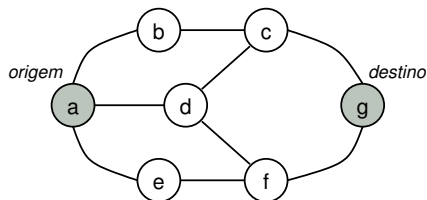
```

Algoritmo 4.3 Protocolo de roteamento reativo: processamento de ROUTE-REPLY

```

1: upon receiving  $rep = \langle \text{ROUTE-REPLY}, src, dst, seq, returnRoute, accRoute, sig \rangle$  from  $x$  do
2:   if VERIFY-RREPLY( $rep$ ) then
3:     if  $rep.src = myId$  then
4:        $R \leftarrow accRoute.nodes$ 
5:       TRYCACHE( $rep.dst, R$ )
6:     else
7:        $n \leftarrow \text{NEXTHOP}(rep.returnRoute)$ 
8:       if  $n$  is valid then
9:         send  $rep$  to  $n$ 
10:      TRYCACHE( $rep.dst, \text{SUBROUTE}(R, myId, rep.dst)$ ) { opportunistic caching }
11:   end do

```



(a) Rede exemplo

a : ROUTE-REQUEST(a, g, a) $\rightarrow b, d, e$
 b : CACHE-REPLY($a, g, b, ab, \{bcg\}$) $\rightarrow a$
 d : CACHE-REPLY($a, g, d, ad, \{dcg, dfg\}$) $\rightarrow a$
 e : ROUTE-REQUEST(a, g, a) $\rightarrow f$
 f : CACHE-REPLY($a, g, f, aef, \{fg\}$) $\rightarrow a$ via fea

(b) Mensagens de roteamento trocadas

Figura 4.4: Exemplo de descoberta de rotas com o protocolo reativo completo. O nó a é a origem, e o nó g é o destino. Os nós e e f conhecem apenas seus vizinhos (os seus caches não contêm rotas para outro nós além dos vizinhos). À esquerda é mostrada a topologia do overlay, e à direita são mostradas as mensagens de roteamento geradas pelo ROUTE-REQUEST original.

Algoritmo 4.4 Protocolo de roteamento reativo: processamento de CACHE-REPLY

```

1: upon receiving  $crep = \langle \text{CACHE-REPLY}, src, dst, seq, sender, returnRoute, accRoute, Rset, sig \rangle$  from  $x$  do
2:   if VERIFY-CREPLY( $crep$ ) then
3:     if  $crep.src = myId$  then
4:        $R_{s2i} \leftarrow crep.accRoute.nodes$  {  $R_{s2i}$  is the route from the source to the CACHE-REPLY sender }
5:       for all  $R_{i2d}$  in  $crep.Rset$  do {  $R_{i2d}$  is a route from the sender to the destination }
6:          $R \leftarrow MERGEROUTES(crep.dst, R_{s2i}, R_{i2d})$ 
7:         TRYCACHE( $crep.dst, R$ )
8:     else
9:        $n \leftarrow NEXTHOP(crep.returnRoute)$ 
10:      if  $n$  is valid then
11:        send  $crep$  to  $n$ 
12:      { opportunistic caching starts here }
13:       $R_i \leftarrow SUBROUTE(crep.accRoute.nodes, myId, crep.sender)$ 
14:      TRYCACHE( $crep.sender, R_i$ )
15:       $R_i \leftarrow GETBESTROUTE(crep.sender)$ 
16:      for all  $R_{i2d}$  in  $crep.Rset$  do
17:         $R \leftarrow MERGEROUTES(crep.dst, R_i, R_{i2d})$ 
18:        TRYCACHE( $crep.dst, R$ )
19:    end do

```

Primitiva	Descrição
SIGN	gera uma assinatura digital usando a chave privada do nó
COSIGN	apõe o endereço e a assinatura digital do nó a uma rota acumulada
REVERSE	retorna o caminho inverso $b-a$ de uma rota $a-b$
GETBESTROUTE	retorna a melhor rota disponível no <i>cache</i> para um dado destino
GETALLROUTES	retorna o conjunto de rotas disponíveis no <i>cache</i> para um dado destino
NEXTHOP	retorna o próximo <i>hop</i> em uma rota
MERGEROUTES	retorna uma rota para um dado destino combinando uma rota do nó chamador até um nó intermediário e uma rota desse intermediário até o destino, eliminando laços de roteamento
VERIFY-RREQ	verifica o número de seqüência, o comprimento máximo de rota e as assinaturas de um ROUTE-REQUEST
VERIFY-RREPLY	verifica o número de seqüência e as assinaturas de um ROUTE-REPLY
VERIFY-CREPLY	verifica o número de seqüência e as assinaturas de um CACHE-REPLY
TRYCACHE	adiciona uma rota para um dado destino ao <i>cache</i> se ela for disjunta das demais já presentes
SUBROUTE	retorna a porção de um rota compreendida entre um nó inicial e um nó final

Tabela 4.1: Primitivas usadas nos algoritmos do protocolo de roteamento

4.1.3 Seleção de Rotas

Com o protocolo de roteamento em duas fases (proativa e reativa) descrito nas seções anteriores, é bastante provável que um nó possua várias rotas em *cache* para qualquer outro. Como as mensagens de roteamento são autenticadas através de assinaturas digitais, nós que não pertencem ao *overlay* não podem injetar informações sobre rotas na rede. Entretanto, existe a possibilidade de que nós maliciosos forneçam informações falsas de roteamento, tentando atrair, repelir ou simplesmente perturbar o tráfego. Nesse contexto, como um nó deve selecionar a rota a ser usada? A resposta trivial — escolher a rota mais curta — permanece válida, mas ela não ajuda a escolher dentre um conjunto de rotas de mesmo comprimento, tais como as rotas simples aprendidas na fase proativa.

Antes que outras possibilidades sejam examinadas, faz-se necessário esclarecer dois pontos. O primeiro é que não existe nenhum método eficiente para evitar que nós *overlay* maliciosos falsifiquem suas próprias informações de roteamento. É impossível dizer se um nó está omitindo um ou mais nós da sua lista de vizinhos (um modo de repelir tráfego). É igualmente impossível dizer *a priori* se um nó está anunciando falsos vizinhos em uma tentativa de atrair tráfego; seria concebível estabelecer que um enlace virtual *ab* (ou *ba*) só seria considerado válido se *a* e *b* se anunciassem reciprocamente como vizinhos, mas isso ainda seria vulnerável a um conluio entre *a* e *b*. O segundo ponto que deve ser mencionado é que, mesmo que fosse possível garantir que todos os anúncios de rotas sejam corretos, os nós ainda poderiam escolher rotas não funcionais. Um exemplo disso seria quando um nó *x* falha e os nós que possuem rotas passando por *x* levam algum tempo para detectar essa falha e reagir a ela. Devido a esses dois pontos, um nó de origem tem que trabalhar com a possibilidade de que algumas das rotas que ele possui em *cache* para um dado destino sejam corretas enquanto outras podem ser faltosas.

Para minimizar o impacto de dados de roteamento falsos, associa-se um **nível de confiança** a cada rota em *cache*. São definidos quatro níveis de confiança: *oportunista*, *sem confiança*, *certificada* e *verificada* (em ordem crescente). Uma rota *oportunista*, como o próprio nome indica, é uma rota obtida através de *caching* oportunista. O nível mais baixo de confiança reflete o fato dessas rotas serem extraídas de informações de roteamento endereçadas a outros nós, sem que o nó que as obtém tenha como avaliar sua utilidade. Uma rota *sem confiança* é uma rota obtida a partir de um CACHE-REPLY ou uma rota indireta simples, enquanto uma rota *certificada* pode ser uma rota obtida de um ROUTE-REPLY enviado pelo nó de destino ou uma rota direta. A distinção entre esses dois níveis se dá porque as rotas *sem confiança* são obtidas de informações fornecidas por outros nós que não o destino, o que põe em dúvida sua validade, ao passo que uma rota *certificada* apresenta evidências de sua validade (a rota acumulada no caso de ROUTE-REPLYS e a relação de vizinhança no caso das rotas diretas). O nível mais alto de confiança é o de rota *verificada*, que só é atribuído quando uma rota (*oportunista*, *sem confiança* ou *certificada*) é usada com sucesso na transmissão de pacotes de dados. As rotas são então selecionadas no *cache* em ordem decrescente de nível de confiança, o que significa que um nó só irá tentar usar rotas obtidas de informações em *cache* de outros nós quando não tiver nenhuma outra rota obtida de uma fonte mais confiável (é preferível arriscar uma rota *oportunista* ou *sem confiança* do que não usar rota nenhuma). A tabela 4.2 mostra os diferentes níveis de confiança para as rotas em *cache* em ordem decrescente, e resume os critérios adotados na atribuição de níveis

às rotas.

Nível de Confiança	Descrição
<i>verificada</i>	rotas usadas com sucesso na transmissão de pacotes
<i>certificada</i>	rotas obtidas de ROUTE-REPLYS e rotas diretas
<i>sem confiança</i>	rotas obtidas de CACHE-REPLYS e rotas indiretas simples
<i>oportunista</i>	rotas obtidas de informações de roteamento endereçadas a outros nós

Tabela 4.2: Resumo dos níveis de confiança para rotas em *cache*

Embora os níveis de confiança sejam úteis, a questão de como selecionar uma rota quando existem várias delas com a mesma confiança ainda persiste. Existem aqui algumas possibilidades. A primeira delas é escolher uma das rotas ao acaso e usá-la para enviar pacotes, repetindo o processo caso esta rota falhe. No outro extremo do espectro de possibilidades, pacotes podem ser enviados em paralelo por todas as rotas disponíveis, com o nó transmissor avaliando quais delas funcionam e quais aparentam estar faltosas, e escolhendo a melhor dentre as funcionais para as transmissões subsequentes. A primeira abordagem é conservadora e otimista, usando apenas os recursos de rede necessários com base em uma premissa subjacente de que as rotas provavelmente irão funcionar; a penalidade a ser paga quando essa premissa é falsa é que pode levar muito tempo até que uma rota funcional seja encontrada, devido à demora inerente à detecção de falhas. A segunda abordagem, por sua vez, é agressiva e pessimista: ela parte do pressuposto de que a maioria das rotas deve ser faltosa e permite encontrar uma rota funcional com rapidez, mas correndo o risco de desperdiçar recursos de rede caso esse pressuposto não seja verdadeiro. Existe também uma abordagem intermediária, inspirada em [Gummadi et al., 2004], a qual foi adotada neste trabalho: quando existem várias rotas com o mesmo comprimento e nível de confiança, escolhem-se k dentre elas ao acaso e enviam-se pacotes por essas k rotas, sendo a mais rápida dentre aquelas que funcionarem escolhida para as transmissões subsequentes (se nenhuma das k rotas funcionar, escolhem-se k dentre as rotas não selecionadas inicialmente e o processo é repetido). O número de rotas escolhido nesse esquema pode ser fixo ou variável (ele pode ser uma função do número de rotas comparáveis disponíveis, por exemplo). Além disso, esse processo de teste de rotas em paralelo fornece informações úteis para o gerenciamento do *cache* de rotas: as rotas que transmitem pacotes com sucesso têm seu nível de confiança incrementado para *verificada*, enquanto aquelas que não funcionam são simplesmente removidas do *cache*.

4.2 Encaminhamento de Pacotes na ROTI

Enquanto o protocolo de roteamento se preocupa em trocar informações sobre as diferentes rotas na rede, o encaminhamento de pacotes consiste em efetivamente usar essas rotas para transmitir pacotes de uma origem para um destino. Embora esta aparente ser uma tarefa simples, ela se torna complicada à medida em que são levadas em consideração faltas e intrusões. No primeiro caso, a rota usada para uma transmissão pode passar por um nó ou enlace que acabou de sofrer uma falta ainda não percebida pelo subsistema de roteamento. No segundo caso, nós comprometidos podem tentar interferir nas comunicações que passam por eles, modificando pacotes ou desviando-os de suas rotas.

Em linhas gerais, os mecanismos empregados para conferir robustez ao encaminhamento de pacotes face a essa gama de ameaças podem adotar duas estratégias, prevenção ou recuperação. Os mecanismos de prevenção visam impedir ou dificultar ao máximo a ocorrência de problemas como os citados, ao passo que os mecanismos de recuperação consideram que tais problemas são inevitáveis, e se concentram em restaurar as condições normais de operação quando eles ocorrem. Mecanismos preventivos costumam impingir um custo ao desempenho do sistema mesmo em situações livres de faltas ou intrusões, enquanto os mecanismos baseados em recuperação tendem a ser mais leves em situações normais, mesmo que para isso imponham um custo significativo quando for necessária uma recuperação. A escolha da combinação mais apropriada de mecanismos para o encaminhamento de pacotes é ainda mais delicada porque essa é a funcionalidade mais usada da ROTI, e seu desempenho influi decisivamente no desempenho das aplicações que utilizam a rede.

Em vista das amplas possibilidades de escolha, e objetivando uma melhor compreensão das diferentes abordagens, foram desenvolvidos dois esquemas de encaminhamento de pacotes para a ROTI. O primeiro, denominado **esquema leve**, privilegia a recuperação em detrimento da prevenção, ao passo que o segundo, chamado de **esquema pesado**, possui um número maior de mecanismos preventivos. Além das diferenças em termos de objetivos e garantias, as distinções entre esses dois esquemas se refletem também na localização de faltas, conforme será discutido no capítulo 5. A seção 4.2.1 apresenta o esquema leve, e a seção 4.2.2 examina o esquema pesado. Uma comparação entre esses esquemas é mostrada na seção 4.2.3.

4.2.1 Esquema Leve

O esquema leve de encaminhamento de pacotes tem dois objetivos. O primeiro é garantir a integridade e a autenticidade dos dados transmitidos. O outro é fornecer suporte para os mecanismos de detecção de falhas e recuperação de faltas, que são os responsáveis por assegurar que os pacotes sejam entregues a despeito de faltas e intrusões na rede.

A figura 4.5 mostra o formato de um pacote ROTI usando o esquema leve. O campo SRC contém o endereço de origem, e ROUTE contém a rota fornecida pelo subsistema de roteamento, que é formada por uma seqüência de nós *overlay* dos quais o último é o nó de destino. O campo SEQ contém um número de seqüência, usado para evitar *replay* de mensagens e para permitir a detecção de perdas, enquanto ACK contém um número do reconhecimento (*acknowledgment*), que é usado na detecção de falhas (o uso de SEQ e ACK na detecção de falhas é discutido em detalhes no capítulo 5). O campo PAYLOAD contém os dados de aplicação que são enviados através da rede, e HMAC contém um código de autenticação de mensagem baseado em resumo com chave (*keyed-hash message authentication code*) [Krawczyk et al., 1997], que cobre o cabeçalho e o conteúdo do pacote. Esse HMAC usa uma chave simétrica compartilhada entre os nós de origem e destino que é estabelecida durante a primeira comunicação entre esses nós. HMACs permitem que tentativas de manipular ou forjar pacotes sejam facilmente detectadas, garantindo integridade e autenticação de origem para os pacotes, e são computacionalmente muito mais eficientes do que assinaturas digitais. A adoção do HMAC na ROTI se deve à sua popularidade (ele é usado, por exemplo, no IPsec [Kent e Seo, 2005]), mas

outros algoritmos que gerem um resumo criptográfico baseado em uma chave e que tenham o mesmo nível de segurança, como UMAC [Krovetz et al., 2006], poderiam substituí-lo sem prejuízos para a arquitetura.

SRC	ROUTE	SEQ	ACK	PAYLOAD	HMAC
-----	-------	-----	-----	---------	------

Figura 4.5: Formato de um pacote ROTI usando o esquema leve

O campo ROUTE contém a rota completa para o destino, escolhida pelo nó de origem. Ao receber um pacote do nó s , o nó t verifica o conteúdo de ROUTE para ver se (i) ele pertence à rota e (ii) s é o predecessor de t na rota. Se t for o destino (o último nó em ROUTE), ele verifica que o HMAC está correto e repassa o conteúdo de PAYLOAD para a camada superior, enviando a seguir um ACK para o nó de origem. Por outro lado, se t não for o destino, ele encaminha o pacote para o nó u que sucede t em ROUTE (desde que t tenha um enlace virtual com u — se tal enlace não existe, t descarta o pacote silenciosamente).

4.2.2 Esquema Pesado

Os objetivos do esquema pesado incluem, além dos objetivos do esquema leve, garantir a confidencialidade das comunicações, reduzir a capacidade de nós intermediários realizarem análise de tráfego com base no cabeçalho dos pacotes e limitar a propagação de pacotes inválidos na rede. Por trás de tais objetivos existe o propósito de restringir as oportunidades que elementos maliciosos têm para interferir de forma inteligente no tráfego que flui através deles.

A figura 4.6 mostra o formato de um pacote ROTI usando o esquema pesado. Os campos SRC e SEQ contêm, respectivamente, o endereço de origem e o número de seqüência do pacote. O campo PAYLOAD contém os dados de aplicação; no esquema pesado, este campo é cifrado com uma chave simétrica compartilhada entre os nós de origem e destino para garantir a sua confidencialidade.

SRC	SEQ	NEXT HOP ₁ + FLAGS ₁	...	NEXT HOP _{n} + FLAGS _{n}	PAYLOAD	HMAC _{n}	...	HMAC ₁
-----	-----	--	-----	--	---------	--------------------------------	-----	-------------------

Figura 4.6: Formato de um pacote ROTI usando o esquema pesado

A rota é codificada no pacote como uma seqüência de cabeçalhos, cada um contendo o endereço do próximo *hop* (NEXT HOP _{k}) e algumas *flags* de controle (FLAGS _{k}) para o k -ésimo nó da rota; além disso, um HMAC _{k} para cada nó da rota é inserido após o PAYLOAD. Os cabeçalhos são cifrados sucessivamente usando *onion encryption* [Goldschlag et al., 1996]. Os mecanismos criptográficos são aplicados recursivamente, começando pelo nó de destino; assim, o que cada nó intermediário interpreta como sendo seu *payload* é, na verdade, o conjunto cabeçalho+*payload*+HMAC para o próximo *hop*. Com isso, cada nó intermediário sabe apenas qual é o próximo *hop*, mas não quem é o nó de destino. Gerar um HMAC para cada nó pertencente à rota é computacionalmente mais eficiente do que gerar uma única assinatura digital (para rotas com um número pequeno de *hops*).

A figura 4.7 ilustra o processo de construção de um pacote ROTI transmitido do nó a para o nó

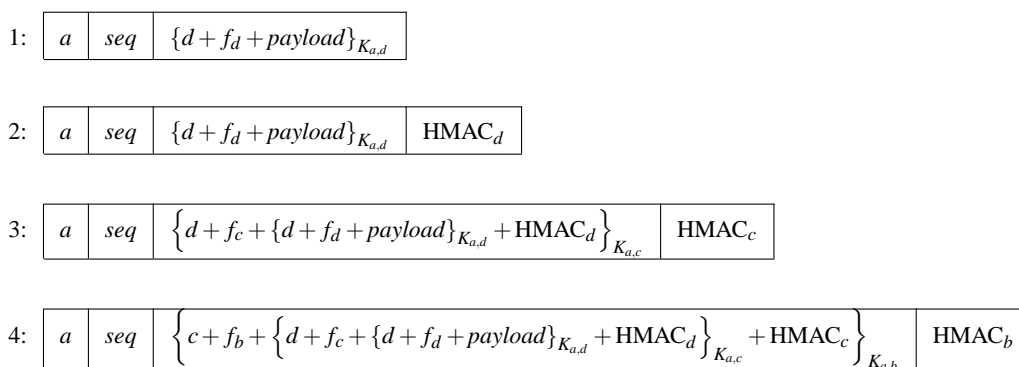


Figura 4.7: Construção de um pacote no esquema pesado

d usando a rota $abcd$ (na figura, $\{dado\}_{K_{i,j}}$ significa que $dado$ é cifrado com a chave simétrica $K_{i,j}$ compartilhada entre i e j , e f_i representa as *flags* para o nó i):

1. O cabeçalho referente ao nó d (NEXT HOP= d , FLAGS= f_d) e o *payload* são cifrados com $K_{a,d}$;
2. O HMAC cobrindo todos os campos (SRC, SEQ, o cabeçalho do nó d e PAYLOAD) é computado usando $K_{a,d}$ e agregado ao pacote;
3. O cabeçalho de c (NEXT HOP= d , FLAGS= f_c) e o seu *payload* (formado pelo cabeçalho de d , o *payload* e HMAC_d) são cifrados com $K_{a,c}$, e o HMAC correspondente é computado (este passo combina os passos 1 e 2 acima);
4. O cabeçalho de b (NEXT HOP= c , FLAGS= f_b) e o seu *payload* são cifrados com $K_{a,b}$, e o HMAC correspondente é computado.

Ao receber um pacote, um nó i primeiro verifica se HMAC _{i} está correto usando $K_{src,i}$, descartando o pacote em caso contrário. A seguir, o nó i decifra o cabeçalho e o *payload* do pacote, descobrindo quem é o próximo *hop*. Se NEXT HOP _{i} = i , ele repassa o conteúdo em PAYLOAD para a camada superior; caso contrário, ele envia o pacote (sem o seu próprio cabeçalho e HMAC) para NEXT HOP _{i} (se NEXT HOP _{i} não é um vizinho, i descarta o pacote).

4.2.3 Comparação Entre os Esquemas

O esquema leve faz um uso minimalista de mecanismos de prevenção visando impor um baixo *overhead* ao encaminhamento de pacotes e propiciando um melhor desempenho em situações livres de faltas e intrusões. Os mecanismos adotados (roteamento na origem e HMAC fim a fim) permitem que nós intermediários detectem apenas tentativas de desvio de pacotes (*misrouting*) que não envolvam a modificação do campo ROUTE. Como o HMAC contido no pacote só pode ser verificado pelo nó de destino (que é quem possui a chave criptográfica requerida), este é o único capaz de detectar outras alterações, o que possibilita que pacotes inválidos circulem na rede antes de chegar ao seu destino.

O esquema pesado, por sua vez, usa um conjunto maior de mecanismos de prevenção com o intuito de garantir a confidencialidade do conteúdo e do destino final do tráfego. Além do valor intrínseco da privacidade obtida, a cifragem do conteúdo e dos cabeçalhos limita os nós do *overlay* e da rede subjacente a conhecerem o endereço de origem e o próximo *hop* dos pacotes, reduzindo assim a capacidade de nós comprometidos interferirem de forma inteligente com o tráfego. O uso de um HMAC por *hop* permite que modificações não autorizadas sejam rapidamente detectadas pelo primeiro nó correto a processar o pacote corrompido; este nó pode descartar o pacote de forma a evitar sua propagação pela rede, reduzindo assim o consumo de recursos como banda passante e capacidade de processamento.

O *overhead* imposto pelos mecanismos criptográficos usados nos esquemas de encaminhamento pode ser quantificado de duas formas, tempo e espaço. A tabela 4.3 mostra as funções criptográficas que são executadas em cada nó pertencente a uma rota, para cada pacote transmitido; o *overhead* (criptográfico) de tempo por pacote é a soma do tempo gasto em cada nó para executar as funções indicadas. Na tabela, considera-se que a rota contém k nós (excluindo o nó de origem e incluindo o nó de destino); as funções de geração e verificação de HMAC, cifragem e decifragem são representadas, respectivamente, por H , H^{-1} , E e D . Um aspecto importante evidenciado na tabela é que o nó de origem despense mais esforço criptográfico do que os demais nós, o que funciona como um desestímulo a ataques de negação de serviço baseados no envio de mensagens inócuas cujo único propósito é desperdiçar capacidade de processamento dos nós que as recebem.

Nó na rota	Esquema leve	Esquema pesado
origem	$1 H$	$k E + k H$
intermediário	0	$1 H^{-1} + 1 D$
destino	$1 H^{-1}$	$1 H^{-1} + 1 D$
Total por pacote	$1 H + 1 H^{-1}$	$k \cdot (H + H^{-1} + E + D)$

Tabela 4.3: *Overhead* de tempo dos esquemas de encaminhamento de pacotes. As células da tabela representam o tempo despendido com funções criptográficas em cada nó para cada pacote encaminhado, considerando uma rota com k nós (excluindo o nó de origem e incluindo o nó de destino). H e H^{-1} representam a geração e a verificação de um HMAC, enquanto E e D representam a cifragem e a decifragem de um pacote.

O *overhead* de espaço devido aos mecanismos criptográficos é de um HMAC para o esquema leve e k HMACs para o esquema pesado; se for usado HMAC com SHA-1 [NIST, 2002] como algoritmo de *hash*, esse *overhead* é de 20 bytes (160 bits) por HMAC. Além disso, como as cifras simétricas costumam operar com blocos de tamanho fixo, o algoritmo criptográfico usado para *onion encryption* no esquema pesado pode impor um *overhead* na forma de *padding*, caso o tamanho dos dados a serem cifrados não seja um múltiplo do tamanho do bloco adotado. No pior caso, esse *overhead* é inferior a k vezes o tamanho do bloco; se o algoritmo de cifragem usado for o AES [NIST, 2001], os blocos são de 16 bytes (128 bits).

É importante observar que o custo dos mecanismos criptográficos é diretamente proporcional

à proteção oferecida em cada um dos esquemas. A escolha do esquema mais apropriado depende de alguns fatores, como a aplicação à qual se destina a rede, quais ameaças são importantes para essa aplicação (por exemplo, quão importante é a confidencialidade?) e o grau de hostilidade que é esperado dos adversários. Uma possibilidade promissora seria incluir um mecanismo para que os nós negociassem dinamicamente qual o esquema de encaminhamento usariam em uma comunicação; assim, em situações livres de faltas poderia ser usado o esquema leve, e em situações mais hostis poderia ser usado o esquema pesado. Um problema prático dessa negociação seria que a decisão dos nós comunicantes afetaria também os nós intermediários; esse problema precisa ser melhor analisado para que um mecanismo dessa natureza possa ser implementado.

4.3 Trabalhos Relacionados

Protocolos seguros de roteamento e encaminhamento de pacotes não são uma novidade. A literatura registra diversas propostas nesse sentido, especialmente relativas a roteamento. Uma boa parte dessas propostas, porém, se concentra em impedir ataques externos, como injeção de mensagens de roteamento, sem considerar a possibilidade de um roteador legítimo ser malicioso e tentar interferir de maneira indevida no processo de roteamento, visando, por exemplo, atrair ou repelir tráfego. Esta seção examina primeiramente as principais experiências relacionadas a roteamento seguro e em seguida alguns trabalhos envolvendo mecanismos para proteção do encaminhamento de pacotes. Para fins de comparação, são apresentados também os mecanismos de roteamento e encaminhamento usados nas redes *overlay* discutidas nas seções 2.2.1 e 2.2.2.

4.3.1 Roteamento

O trabalho pioneiro em redes tolerantes a faltas bizantinas é a tese de Perlman [1988]. Ela propõe dois mecanismos básicos: um protocolo de inundação e um protocolo de estado de enlaces, ambos tolerantes a faltas bizantinas. O protocolo de estado de enlaces usa assinaturas digitais para garantir a autenticidade e a integridade das atualizações, que são disseminadas na rede usando o protocolo de inundação. Conforme discutido na seção 4.1.1, protocolos baseados em inundação como o de Perlman têm *overhead* maior do que as difusões locais usadas na fase proativa do protocolo de roteamento da ROTI; porém, a diferença cai e pode até se inverter quando o protocolo reativo é muito utilizado. Diferentemente da ROTI, na proposta de Perlman não existe preocupação em controlar a frequência de emissão ou processamento de atualizações, o que pode levar a ataques de negação de serviço. Por outro lado, ela define um mecanismo que garante justiça (*fairness*) quando existe competição entre os nós da rede por acesso aos enlaces, algo que a ROTI não contempla.

A RON [Andersen et al., 2001] presume uma rede completamente conectada, onde cada um dos N nós tem $N - 1$ vizinhos. Ela usa um protocolo baseado em estado de enlaces para disseminar atualizações que contêm um resumo das diferentes métricas monitoradas (seção 2.2.1) para cada um dos seus $N - 1$ enlaces virtuais. Essas atualizações são disseminadas usando a própria RON, o que aumenta as chances delas serem entregues mesmo quando existem faltas na rede. Como a RON adota

a premissa de nós cooperativos, não existe nenhuma preocupação com a segurança do roteamento: as mensagens não são autenticadas e as informações de alcançabilidade são sempre consideradas corretas. Cada nó mantém várias tabelas de rotas, sendo uma tabela para cada métrica (pois a rota que minimiza as perdas para um destino pode não ser a mesma rota que minimiza a latência, por exemplo). As tabelas de rotas armazenam, para cada destino, o endereço do próximo *hop* da rota até esse destino segundo a métrica associada; por exemplo, a entrada para o nó x contém o endereço do primeiro *hop* na rota que apresenta o maior *throughput* ou a menor latência até x , dependendo da métrica associada. Como na proposta de Perlman, o *overhead* de roteamento da RON é maior que o da ROTI, ao menos enquanto é usado predominantemente o protocolo proativo. Em termos de critérios de seleção de rotas, a ROTI prioriza sempre a segurança e a confiabilidade, enquanto a RON permite às aplicações privilegiarem a confiabilidade (usando a métrica de perdas para escolher a melhor rota) ou o desempenho (usando as métricas de latência ou *throughput* na escolha).

O SOSR [Gummadi et al., 2004] não tem uma função explícita de roteamento: quando uma rota direta é considerada faltosa, o nó de origem envia os pacotes para o destino através de um conjunto de nós intermediários. Como esses nós são escolhidos aleatoriamente, não existe a necessidade de um protocolo de roteamento. Entretanto, essa estratégia exige um mecanismo de controle de filiação (*membership*), para saber quais nós executam o protocolo e podem ser usados como intermediários. A descrição do SOSR em [Gummadi et al., 2004], porém, é omissa em relação a esse mecanismo: os experimentos apresentados no artigo usam um conjunto fixo de nós devidamente instrumentados com o *software* necessário, sem discutir como um nó adquire conhecimento da existência dos demais. A ausência de uma funcionalidade específica de roteamento no SOSR dificulta a sua comparação com a ROTI neste quesito. Uma observação que pode ser feita, porém, é que a eficácia da escolha aleatória de nós intermediários para contornar faltas, especialmente bizantinas, é inversamente proporcional ao número de elementos faltosos na rede.

Existem diversas propostas de protocolos de roteamento seguro para a Internet. Um apanhado dessas propostas pode ser encontrado em [Papadimitratos e Haas, 2002]. A maioria delas usa mecanismos criptográficos para garantir autenticação de origem e integridade dos dados de roteamento, e não oferece meios de verificar se uma rota anunciada é legítima (isto é, se ela corresponde à topologia da rede) ou não, como o mecanismo de seleção de rotas da ROTI. Um exemplo dessas propostas é o S-BGP [Kent et al., 2000], que se preocupa com a autenticação dos anúncios de rotas (UPDATEs) no BGP. Ele usa duas PKIs (estritamente falando, duas hierarquias de certificação), uma para garantir que um AS pode anunciar um bloco de endereços IP e outra para garantir que um roteador é autorizado a representar um AS. Essas PKIs dão suporte a dois tipos de assinaturas, atestados de endereços e atestados de rotas. Um atestado de endereços garante que um roteador que se anuncia como origem de uma rota para um bloco de endereços é autorizado a fazê-lo; cada atualização BGP contém um atestado desse tipo. Os atestados de rotas validam a seqüência de ASs anunciada em uma rota; cada BGP UPDATE possui um atestado de rotas por AS pertencente à rota anunciada. Este esquema de atestação de rotas é similar ao esquema de rotas acumuladas usado no protocolo reativo da ROTI. As trocas de mensagens entre roteadores adjacentes usam IPsec para garantir integridade e autenticação de origem e proteger contra ataques de *replay* (no nível de rede). Como o S-BGP busca compatibilidade com o BGP, ele é vulnerável ao *replay* de UPDATEs, pois o BGP não usa números de seqüência

nas suas mensagens.

4.3.2 Encaminhamento de Pacotes

Para encaminhamento de pacotes em uma rede tolerante a faltas bizantinas, Perlman [1988] propõe dois esquemas. O primeiro é um esquema baseado em roteamento na origem, com a rota incluída no cabeçalho dos pacotes, tal como o adotado no esquema leve da ROTI. Entretanto, em vez de um HMAC verificado apenas no destino (como na ROTI), cada pacote de dados transmitido é assinado, e a assinatura é verificada em cada *hop*, o que aumenta sensivelmente o *overhead* da transmissão. No segundo esquema, antes de usar uma rota o nó de origem difunde uma mensagem ROUTE-SETUP, que especifica a rota a ser usada para um par origem-destino $s-d$; os nós que recebem o ROUTE-SETUP e pertencem a essa rota passam a usá-la para transmitir pacotes de s para d (isto é, pacotes com endereço de origem s e endereço de destino d que chegam são repassados ao próximo *hop* especificado na rota). Quando os pacotes são transmitidos, os nós intermediários certificam-se apenas de que eles procedem do antecessor correto antes de encaminhá-los ao próximo *hop*, reduzindo assim o *overhead* de transmissão. Porém, neste segundo esquema os pacotes de dados não são assinados, o que restringe a proteção oferecida à possibilidade de detecção de desvio de pacotes, eliminando a autenticação e proteção de integridade do tráfego. Ambos os esquemas utilizam ainda cifragem *hop a hop* e cálculo de caminhos disjuntos. Nós maliciosos podem simular a existência de um enlace entre dois nós corretos, atuando como relés (figura 4.8); o uso de cifragem *hop a hop* evita que um elemento bizantino, localizado no que a rede considera ser um enlace, interfira de forma inteligente no encaminhamento de pacotes. O outro mecanismo adotado, de cálculo de caminhos disjuntos, é usado para encontrar alternativas quando o melhor caminho fornecido pelo algoritmo de roteamento não funciona. Entretanto, Perlman não propõe nenhum critério para escolha de rotas que leve em conta aspectos de segurança, usando apenas métricas tradicionais baseadas em desempenho.

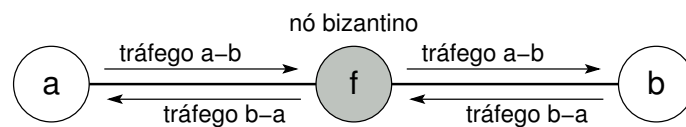


Figura 4.8: Um nó faltoso que atua como relé. O nó f age como relé entre a e b , enviando por um enlace o tráfego que chega pelo outro, criando assim a ilusão de um enlace inexistente entre os nós.

A RON [Andersen et al., 2001] emprega um esquema de encaminhamento de pacotes com decisões *hop a hop*. Diferente dos esquemas tradicionais, porém, aqui o nó de origem especifica qual métrica (perda, latência, *throughput* ou outra) deve ser usada pelos nós intermediários para determinar o próximo *hop*. A escolha da métrica fica a critério das aplicações que utilizam a RON, e depende dos requisitos dessas aplicações. Como não tem preocupações de segurança, a RON não usa nenhum mecanismo para garantir confidencialidade, integridade ou autenticidade do tráfego.

O SOSR [Gummadi et al., 2004] também usa decisão de roteamento *hop a hop*, com rotas limitadas a um único intermediário. Quando é usado roteamento indireto, os pacotes do nó de origem para o nó de destino são encapsulados em pacotes da origem para um intermediário; o nó intermediário

extrai o pacote interno e o envia ao destino. Nós intermediários usam um esquema de tradução de endereços (NAT — *Network Address Translation*) para manter um mapeamento entre o nó de origem e o nó de destino. O mapeamento é usado para que o intermediário saiba para qual nó deve enviar pacotes de resposta recebidos do nó de destino; essas respostas são encapsuladas em pacotes que são transmitidos do intermediário para o nó de origem correspondente. A exemplo da RON, o SOSR também não implementa proteção do tráfego por não considerar ameaças de segurança.

Um esquema de encaminhamento de pacotes similar aos adotados na ROTI foi proposto em [Avramopoulos et al., 2004]. Esse esquema também utiliza roteamento na origem, MACs (funcionalmente equivalentes a HMACs) para autenticar pacotes e reconhecimentos para detectar perdas (os aspectos de detecção de falhas deste esquema serão discutidos na seção 5.4). A autenticação é feita através de MACs encadeados: em uma rota $s \dots n_i n_{i+1} \dots d$, o MAC do roteador n_i é calculado usando não apenas o cabeçalho e o conteúdo do pacote como também os MACs dos roteadores n_{i+1}, \dots, d . A autenticação se aplica não só a pacotes de dados como a reconhecimentos (ACKs) e anúncios de faltas (FAs), que são pacotes de controle. Entretanto, a autenticação dessas mensagens de controle requer que, para cada pacote de dados enviado, o nó de origem gere, além dos MACs, uma cadeia de *hashes* de três elementos para cada nó da rota (logo, uma rota formada por m nós requer m cadeias de *hashes* de três elementos cada); o terceiro elemento de cada uma das cadeias é transmitido junto com o pacote (num total de m elementos por pacote). O esquema tenta descartar pacotes inválidos tão cedo quanto seja possível, reduzindo o tráfego espúrio: para isso, cada nó intermediário verifica não apenas o MAC a ele correspondente como também o número de seqüência do pacote, que deve estar dentro de uma faixa de valores admissíveis. Isso não apenas impõe *overhead* de processamento em cada nó intermediário como também significa que esses nós precisam manter informações de estado para todo o tráfego que encaminham, algo que limita a escalabilidade do esquema. O uso de MACs encadeados é similar ao que é feito no esquema pesado da ROTI, mas sem o uso de cifragem. O esquema de uso de cadeias de *hashes* para autenticar mensagens de controle é excessivamente complexo, e é justificado pela idéia de validar o tráfego de dados e de controle, que acrescenta *overhead* de processamento e de armazenamento de estado em todos os nós intermediários em troca de uma economia modesta de banda de comunicação.

A cifragem recursiva de pacotes (*onion routing*) é comumente usada em serviços que oferecem comunicação anônima através da Internet. Esse conceito é geralmente atribuído a Chaum, que criou as *mixnets*, ou redes de misturadores [Chaum, 1981]. De acordo com a proposta de Chaum, a correspondência entre emissores e receptores é mantida secreta encapsulando as mensagens em camadas de cifragem e transmitindo-as através de uma série de misturadores (*mixes*); em cada misturador, a mensagem é decifrada (removendo assim uma camada), atrasada e embaralhada com outras mensagens antes de ser transmitida ao próximo *mix*. Como a idéia é prevenir tanto a observação direta do conteúdo das mensagens como a análise de tráfego, frequentemente é usado tráfego de fundo para evitar que um adversário obtenha informações relevantes mesmo quando o tráfego real é baixo. Dentre os serviços que utilizam *onion routing* em redes *overlay*, podem ser citados:

- Tarzan [Freedman e Morris, 2002]: é um sistema P2P que permite a comunicação anônima através da Internet. Um nó que deseja anonimato seleciona um conjunto de nós que formam

um caminho no *overlay*. A seguir, esse nó de origem configura um túnel usando esses nós, de forma que cada nó saiba apenas o próximo *hop* para onde deve enviar pacotes, sem conhecer o destino final do tráfego. Nessa configuração são negociadas com cada nó pertencente ao túnel as chaves simétricas usadas pelos mecanismos criptográficos. Finalmente, o nó de origem envia os pacotes, cifrados recursivamente, através desse túnel. O último nó do túnel é chamado nó de saída, e é o responsável por enviar os pacotes recebidos através do túnel até o seu destino final (geralmente um endereço Internet externo ao Tarzan), receber os pacotes de resposta do destino e encaminhá-los de volta ao nó de origem, através do túnel. A seleção de nós para formar uma rota é restrita a pares de nós que usam tráfego de fundo para manter um nível de comunicação independente da taxa de transmissão de dados.

- Tor [Dingledine et al., 2004]: é uma rede *overlay* que tem por objetivo garantir a comunicação anônima para aplicações de baixa latência (tráfego interativo) através da Internet, evitando que adversários possam identificar pares de nós comunicantes ou de vincular uma ou mais comunicações a usuários específicos. O *overlay* do Tor é composto por um conjunto de *onion routers* (ORs), que formam uma rede completamente conectada cujos enlaces virtuais são implementados através de conexões TLS [Dierks e Rescorla, 2006]. Clientes que desejam se comunicar de forma anônima executam um *onion proxy* (OP), que se encarrega de obter a lista de ORs (mantida em servidores de diretórios replicados), estabelecer circuitos através do *overlay* e gerenciar conexões de aplicações. O estabelecimento de circuitos é similar ao estabelecimento de túneis no Tarzan; uma diferença é que, durante a transmissão de dados, um OP pode escolher qual OR pertencente ao circuito será usado como nó de saída para uma dada conexão. Além do estabelecimento sob demanda, os circuitos são também trocados periodicamente, o que inclui o estabelecimento de novas chaves criptográficas e a destruição das chaves antigas, garantindo assim que o conteúdo cifrado não poderá ser recuperado no futuro (a chamada *perfect forward secrecy*). Diferente do Tarzan, o Tor se baseia essencialmente em *onion encryption* para oferecer anonimato, não usando tráfego de fundo nem mecanismos como *padding* ou mistura de mensagens para esse propósito; seus autores argumentam que, enquanto o custo de tais mecanismos é bem conhecido, os benefícios de segurança que eles oferecem não estão suficientemente estabelecidos para justificar o seu uso.

A ROTI não chega a ter requisitos de anonimato propriamente ditos; o uso de *onion encryption* no esquema pesado de encaminhamento visa em primeiro lugar limitar a capacidade de nós maliciosos tomarem decisões inteligentes de filtragem ou desvio de tráfego, e em segundo lugar garantir a confidencialidade. Desta maneira, a ROTI emprega um conjunto mais leve de mecanismos do que os oferecidos por *overlays* como Tarzan e Tor, que são projetados especificamente para garantir anonimato e consideram um modelo de adversário mais preocupado em violar a confidencialidade do que a integridade ou disponibilidade do serviço de comunicação (o comprometimento dessas duas últimas propriedades só é perseguido quando colabora para o comprometimento da primeira).

4.4 Conclusões do Capítulo

Este capítulo descreveu os protocolos de roteamento e encaminhamento de pacotes na ROTI, que formam o cerne da funcionalidade da rede. Para o roteamento é adotado um protocolo híbrido, que tira vantagem das características das redes *overlay* para garantir um baixo *overhead* quando o número de faltas é pequeno mas é suficientemente flexível para encontrar rotas mais longas em situações mais adversas. Como nenhum protocolo de roteamento é capaz de garantir que as rotas propagadas são funcionais, foi criado um mecanismo que atribui níveis de confiança às rotas em *cache*, e que orienta a seleção de rotas na direção das rotas mais confiáveis.

O encaminhamento de pacotes se baseia em roteamento na origem e no uso de mecanismos criptográficos eficientes para garantir a integridade e a autenticidade do tráfego. Os dois esquemas de encaminhamento propostos oferecem diferentes garantias de segurança, e são adequados a situações distintas. O custo dos mecanismos usados nesses esquemas é proporcional à proteção oferecida.

Nem o roteamento na origem nem os mecanismos criptográficos de autenticação garantem que um pacote seja corretamente encaminhado pelos nós intermediários até o seu destino; a sua utilidade é basicamente garantir que o comportamento faltoso seja detectado. É perfeitamente possível que um nó malicioso execute o protocolo de roteamento corretamente e depois se recuse a encaminhar pacotes para outros nós. Uma rede que objetiva tolerar intrusões deve ser capaz de detectar e contornar esse tipo de comportamento, e os mecanismos usados na ROTI para esse propósito são discutidos no próximo capítulo.

Capítulo 5

Detecção de Falhas e Recuperação de Faltas na ROTI

Os subsistemas de roteamento e encaminhamento de pacotes, apresentados no capítulo anterior, implementam as funcionalidades necessárias para a comunicação entre nós da ROTI. Entretanto, esta é apenas parte da nossa solução: apesar de incorporarem mecanismos para proteção do tráfego e para uso de rotas redundantes, esses subsistemas precisam de informações sobre o funcionamento das rotas, para que possam tomar providências caso ocorram problemas. Este capítulo complementa então a arquitetura ROTI, introduzindo os mecanismos que são usados para detecção de falhas e recuperação de elementos faltosos, incluindo comportamento malicioso.

A idéia central do mecanismo de detecção é monitorar as rotas usadas para transmissão de dados para verificar se o seu comportamento é correto, conforme mostra a seção 5.1. Em princípio, a recuperação de elementos faltosos ou maliciosos a partir das falhas detectadas se dá em dois níveis. Falhas isoladas, como a perda ocasional de pacotes, são corrigidas através de retransmissões. Quando o nível de severidade das falhas aumenta de forma que as retransmissões não são suficientes para garantir a confiabilidade da rede, são tomadas medidas para que os nós removam da sua visão da topologia os elementos (enlaces ou rotas) que consideram faltosos. Como consequência dessa remoção, transmissões subsequentes de pacotes passam a utilizar rotas alternativas. Os mecanismos de localização e recuperação de elementos faltosos na ROTI são detalhados na seção 5.2.

A arquitetura composta por esses mecanismos considera que a topologia corrente da ROTI encerra toda a redundância disponível na rede. Embora eficaz no sentido de oferecer tolerância a intrusões, a remoção de elementos faltosos diminui, ao longo do tempo, essa redundância. A solução trivial para esse problema é superdimensionar o *overlay*, acrescentando nós para que mesmo que elementos faltosos sejam removidos ainda exista redundância suficiente para permitir a comunicação entre os nós corretos. Infelizmente, essa solução pode ter consequências negativas (especialmente em situações livres de falhas), tanto em termos do custo dos recursos computacionais que podem ficar ociosos como pelo impacto que os nós extras têm no desempenho dos protocolos da rede, em muitos casos sem que o benefício que esses nós oferecem seja efetivamente utilizado.

Em vista disso, este capítulo propõe ainda uma abordagem diferente e inovadora para o tratamento e recuperação de faltas, que consiste na reconfiguração da topologia da rede *overlay*. A idéia, descrita na seção 5.3, é que exista um conjunto de nós de reserva que possam ser agregados à rede sob demanda, quando a situação assim o exigir. O objetivo dessa ativação de nós de reserva é manter a redundância da rede dentro de um limite aceitável, aumentando assim a probabilidade de existência de rotas alternativas quando se torna necessário remover um elemento faltoso da topologia. O protocolo de ativação de nós de reserva é uma das principais contribuições desta tese, sendo o primeiro protocolo para reconfiguração de topologia em redes sujeitas a faltas bizantinas e intrusões.

5.1 Detecção de Falhas

A detecção de falhas de parada (*crash*) em sistemas distribuídos com alguma premissa de sincronismo é relativamente trivial. Para detectar se um nó *b* está correto ou falho, o nó *a* pode enviar para *b* uma mensagem de controle (*ping*); se *b* estiver correto (vivo), ele envia uma resposta (*pong*) para *a*. Ao receber a resposta de *b*, o nó *a* sabe que ele está correto (figura 5.1). Um mecanismo equivalente é o *heartbeat*: de tempos em tempos, um nó correto *c* envia uma mensagem I`M ALIVE para um ou mais nós, que interpretam essa mensagem como prova de que *c* está vivo (figura 5.2). A premissa de sincronismo nesse caso é importante para que os nós saibam por quanto tempo no máximo devem aguardar uma resposta ou *heartbeat*: se esse tempo expirar, o nó que não enviou a mensagem esperada pode ser considerado falho (supondo que a premissa de sincronismo não tenha sido violada). Muitos sistemas que adotam a semântica de falhas de parada permitem ainda que um nó informe aos demais sobre as falhas que este nó detecta; os nós que recebem uma notificação assumem-na como verdadeira e reagem de maneira correspondente. A detecção de falhas através de *pings* ou *heartbeats*, sem que seja admitida a possibilidade de equívocos, corresponde à semântica dos detectores de falhas perfeitos (classe \mathcal{P}) da hierarquia de Chandra e Toueg [1996].

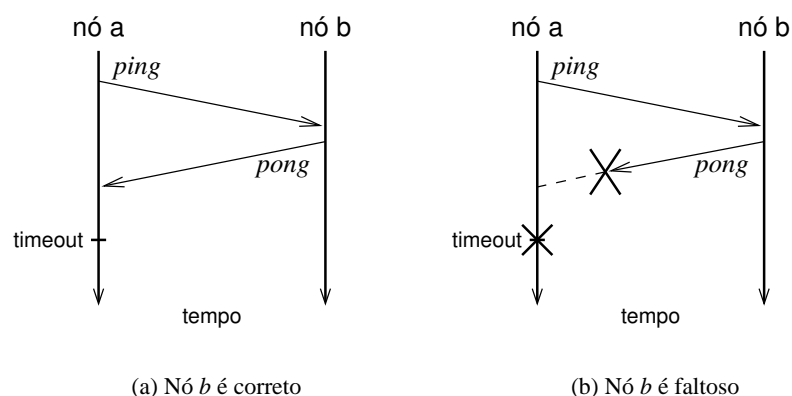


Figura 5.1: Detecção de falhas com *pings*

Entretanto, quando a semântica adotada é a de falhas bizantinas ou arbitrárias, como é o caso da ROTI, a detecção de falhas se torna uma tarefa muito mais complicada. Um nó não pode confiar em notificações individuais de erro emitidas por outros nós, já que o emissor da notificação pode ser um

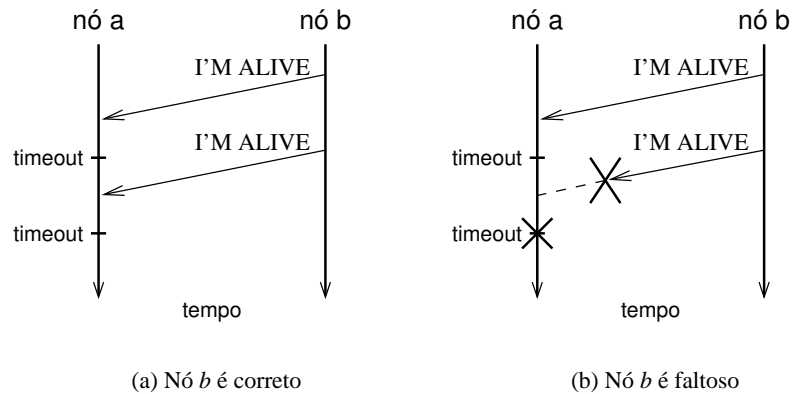


Figura 5.2: Detecção de falhas com *heartbeats*

nó falto tentando levar o nó notificado a tomar uma decisão equivocada. Mensagens específicas de detecção, como *pings* e *heartbeats*, também não são confiáveis, pois um nó falto pode enviar as mensagens necessárias para provar que ele está vivo enquanto se recusa a participar do roteamento ou do encaminhamento de pacotes (mensagens desse tipo ainda podem ser úteis para descobrir se uma rota deixou de funcionar devido a uma falha de parada, porém). O uso de mecanismos que tentam identificar comportamento malicioso, como sistemas de detecção de intrusões baseados em assinatura [Axelsson, 2000], esbarra na dificuldade de se conseguir antecipar todas as possíveis ações danosas de um nó comprometido.

Por outro lado, em uma rede como a ROTI é possível especificar o que é considerado como o comportamento correto da rede, pelo menos do ponto de vista das aplicações que a utilizam para transmitir dados: pode-se dizer que a rede é correta se todos os pacotes transmitidos são entregues aos seus respectivos destinos. Analisar o comportamento da rede como um todo, porém, não é muito útil, por várias razões. Em primeiro lugar, ocorrências isoladas, que afetem apenas uma região da rede, significariam que a rede como um todo seria considerada faltosa; dado que falhas ocasionais de transmissão são um evento relativamente comum, a rede provavelmente permaneceria faltosa o tempo todo. Em segundo lugar, um nó qualquer dificilmente dispõe de elementos para analisar o comportamento global da rede; um nó tem acesso em primeira mão apenas ao tráfego com o qual ele está envolvido, seja como origem, destino ou intermediário. A análise deve, então, ser individualizada para cada nó, que monitora o funcionamento das rotas que **ele** utiliza para transmitir dados. Se os dados forem entregues a rota é considerada correta, caso contrário ela é considerada faltosa e a recuperação se torna necessária. Usar o comportamento fim a fim de rotas como um mecanismo de detecção de falhas bizantinas não é uma idéia nova, já tendo sido previamente aplicada no contexto de redes *ad hoc* móveis [Xue e Nahrstedt, 2004]; sob certo ponto de vista, isso pode ser considerado uma forma simplificada de detecção de intrusões baseada em anomalia [Axelsson, 2000].

Na ROTI, o comportamento de uma rota é caracterizado em termos de medidas de perdas e latência (RTT — *round-trip time*). Para medir as perdas e a latência de uma rota, são usados reconhecimentos (ACKs). Conforme mencionado na seção 4.2, cada pacote possui um número de seqüência, contido no campo SEQ. A cada pacote transmitido para um nó de destino *d*, o nó de ori-

gem s incrementa SEQ e dispara um *timer*. Ao receber o pacote corretamente, o nó d reconhece esse número de seqüência, que é copiado no campo ACK de um pacote transmitido para a origem s (no esquema pesado de encaminhamento, descrito na seção 4.2.2, ACK faz parte das FLAGS de s). Como os reconhecimentos contêm o número de seqüência do pacote sendo reconhecido e um HMAC cuja chave é conhecida apenas por s e d , eles não podem ser forjados. Quando um pacote não é reconhecido antes que o seu *timeout* expire, registra-se uma perda para a rota atual e o pacote é retransmitido. Quando um ACK chega antes da expiração do *timeout* correspondente, a métrica de RTT para a rota é atualizada.

A taxa de perdas de uma rota R é obtida tomando-se a razão entre os pacotes perdidos e os pacotes transmitidos por essa rota:

$$\pi_R = \frac{p_R}{nt_R}$$

onde π_R é a taxa de perdas para R , p_R é o número de pacotes perdidos em R e nt_R é o número de pacotes transmitidos através de R . Para evitar que falhas levem muito tempo para serem detectadas, o cálculo da taxa de perdas considera apenas as estatísticas dos últimos 100 pacotes, e não todo o histórico da rota. Essa estratégia é a mesma adotada para medição de perdas na RON [Andersen et al., 2001].

O cálculo do RTT, por sua vez, usa uma média móvel exponencialmente amortecida (EWMA — *exponentially weighted moving average*) [Roberts, 1959]:

$$RTT_R^i = \alpha \cdot RTT_R^{i-1} + (1 - \alpha) \cdot RTT_{sample}$$

onde RTT_R^i é o i -ésimo RTT estimado para a rota R , RTT_{sample} é o novo RTT amostrado em R e α é o parâmetro de amortecimento (*smoothing*) que controla o quanto da estimativa é baseado na amostra mais recente. Seguindo [Andersen et al., 2001], adota-se $\alpha = 0,9$, ou seja, cada estimativa de RTT é composta por 90% do valor da estimativa anterior e 10% do valor de cada amostra. O uso de EWMA permite que o valor estimado de RTT convirja para o seu valor efetivo sem ser excessivamente influenciado por amostras discrepantes.

A detecção de falhas se baseia na definição do comportamento correto das rotas, que é expresso através de limiares de perdas (π_{thr}) e RTT (RTT_{thr}). Esses limiares são determinados a partir de medidas experimentais de desempenho e nos requisitos das aplicações que utilizam a rede. Quando um ou ambos destes limites são superados, ou seja, quando $\pi_R > \pi_{thr}$ ou $RTT_R > RTT_{thr}$, a rota R é considerada falha.

Cabe notar que nesse esquema uma rota é essencialmente avaliada de acordo com o seu desempenho. Se uma rota contendo nós ou enlaces faltosos entrega pacotes dentro de limites aceitáveis de perda e latência, ela é considerada correta. Por outro lado, uma rota que contenha apenas nós e enlaces corretos mas que tem desempenho ruim será apontada como falha. Essa aparente inconsistência é na verdade irrelevante, pois o que se faz é analisar se o comportamento fim a fim de uma rota satisfaz aos requisitos de aplicação. Esse mecanismo de detecção de falhas tem a interessante propriedade de encapsular várias manifestações de comportamento malicioso: se um pacote é descartado, atrasado, corrompido ou desviado da sua rota, isso será detectado como uma perda ou um RTT excessivo. Se

uma rota se comporta de forma inaceitável por um período suficientemente longo, ela acabará sendo considerada falha. O mesmo raciocínio se aplica a rotas que se tornam inutilizáveis porque um dos seus nós ou enlaces está saturado de tráfego, por exemplo devido a um ataque de negação de serviço baseado em sobrecarga de pacotes (*packet flood*).

A idéia chave por trás desses mecanismos de detecção de falhas é **permitir que cada nó aja individualmente, evitando rotas que ele considera faltosas**. Essa autonomia desempenha um papel central na ROTI, porque ela permite que a rede tolere comportamento malicioso mesmo quando uma maioria de nós é faltosa e sem exigir mecanismos distribuídos custosos, como protocolos de acordo bizantino, para excluir nós faltosos da rede.

5.2 Recuperação de Faltas

Com base nas estimativas de perda e RTT, considera-se que uma rota R é falha quando $\pi_R > \pi_{thr}$ ou $RTT_R > RTT_{thr}$, ou seja, quando pelo menos uma das estimativas supera os limites definidos como aceitáveis. Quando uma falha é detectada entram em ação os mecanismos de localização e recuperação de faltas, que dependem do esquema de encaminhamento de pacotes adotado, leve ou pesado.

5.2.1 Esquema Leve

No esquema leve (seção 4.2.1), quando se detecta a falha de uma rota em uso, a rota inteira é considerada faltosa, e uma nova rota é solicitada ao subsistema de roteamento (que pode fornecer uma rota disjunta do *cache* ou então iniciar um novo processo de descoberta de rotas, se o *cache* estiver vazio). A rota faltosa R entra em **quarentena** por um determinado período, durante o qual ela é considerada inválida pelo subsistema de roteamento. Quando termina a quarentena, a rota é reabilitada no *cache* (seu nível de confiança é rebaixado para *sem confiança*) e é novamente aceita como válida. Essa quarentena é útil para evitar que o subsistema de roteamento insista em escolher repetidamente uma rota ruim, ao mesmo tempo em que ameniza os efeitos de detecções equivocadas de falhas, como quando uma rota é considerada faltosa por estar temporariamente congestionada. Uma rota pode entrar e sair da quarentena até Q_{thr} vezes; quando esse limite é atingido, a rota é marcada como permanentemente inválida, e uma intervenção manual se faz necessária para que ela seja novamente aceita como válida. Como as rotas podem ter um longo tempo de vida, utiliza-se um mecanismo de envelhecimento (*aging*): o contador que registra o número de vezes que uma rota entrou em quarentena é decrementada de tempos em tempos de modo a reduzir a influência de quarentenas ocorridas em um passado distante.

5.2.2 Esquema Pesado

No esquema pesado (seção 4.2.2), quando é detectada uma falha em uma rota em uso, uma nova rota é solicitada ao subsistema de roteamento, e inicia-se um processo de localização do enlace virtual

faltoso. Esse processo consiste em retransmitir o pacote pela mesma rota, especificando no cabeçalho do pacote que todo nó intermediário deve acusar o recebimento desse pacote. O enlace situado entre o nó mais distante que envia um ACK intermediário (chamado IACK) e seu sucessor é declarado faltoso.

Para evitar que elementos faltosos manipulem os ACKs intermediários de forma a incriminar enlaces corretos, IACKs são encadeados. Ao receber um pacote que requisita um IACK, um nó intermediário espera pelo IACK de seu sucessor na rota; quando esse pacote chega, o nó acrescenta seu identificador e assinatura a uma lista de nós percorridos pelo IACK (um esquema semelhante ao da rota acumulada introduzida na seção 4.1.2), que é ainda cifrada com a chave compartilhada com a origem. Esse encadeamento de IACKs faz com que um nó faltoso tenha de incriminar um de seus próprios enlaces caso decida manipular ACKs intermediários. Para evitar que um nó espere indefinidamente pelo IACK do seu sucessor, é usado um *timeout*, definido pelo nó de origem e especificado nas *flags* do cabeçalho.

A figura 5.3 na página seguinte mostra um exemplo da localização de faltas usando IACKs. No exemplo, o nó *a* detecta uma perda na rota *a-f*, causada pelo enlace *ef*. O pacote é então retransmitido, e os nós intermediários que o recebem devem gerar IACKs para ele. Nessa situação, o nó bizantino *c* pode agir das seguintes formas:

- (a) Acrescentar sua assinatura a $IACK(e, d)$ e enviar $IACK(e, d, c)$ para *b*;
- (b) Descartar $IACK(e, d)$ e enviar $IACK(c)$ para *b*;
- (c) Descartar $IACK(e, d)$ e não enviar nada para *b* (que, esgotado o *timeout*, envia $IACK(b)$ para *a*).

No caso (a), o nó *c* não interfere na localização de faltas, e o enlace *ef* é declarado faltoso. Nos outros dois casos, o nó *c* interfere incriminando um de seus enlaces — *cd* no caso (b) e *bc* no caso (c). É importante notar que, como *c* é faltoso, essa localização de faltas é correta, ainda que o enlace apontado como faltoso não tenha relação direta com a falha de transmissão no enlace *ef*.

Uma vez identificado o enlace virtual faltoso, esse enlace entra em quarentena, juntamente com todas as rotas em *cache* que passam por ele. De maneira análoga ao que ocorre no esquema leve, durante a quarentena rotas que contenham esse enlace são consideradas inválidas pelo subsistema de roteamento, e uma vez encerrada a quarentena tanto o enlace como as rotas através dele são reabilitadas no *cache* (estas com nível *sem confiança*). Neste esquema, os enlaces também estão sujeitos ao limite de quarentenas Q_{thr} e ao mecanismo de envelhecimento.

5.3 Reconfiguração de Topologia

Os mecanismos apresentados na seção 5.2 tomam como pressuposto que a topologia corrente do *overlay* constitui toda a redundância disponível para recuperação de faltas: se o subsistema de roteamento é incapaz de encontrar uma rota alternativa para o nó de destino de uma rota falha é porque esse nó se encontra inalcançável. Uma outra característica desses mecanismos é que, em situações com alta

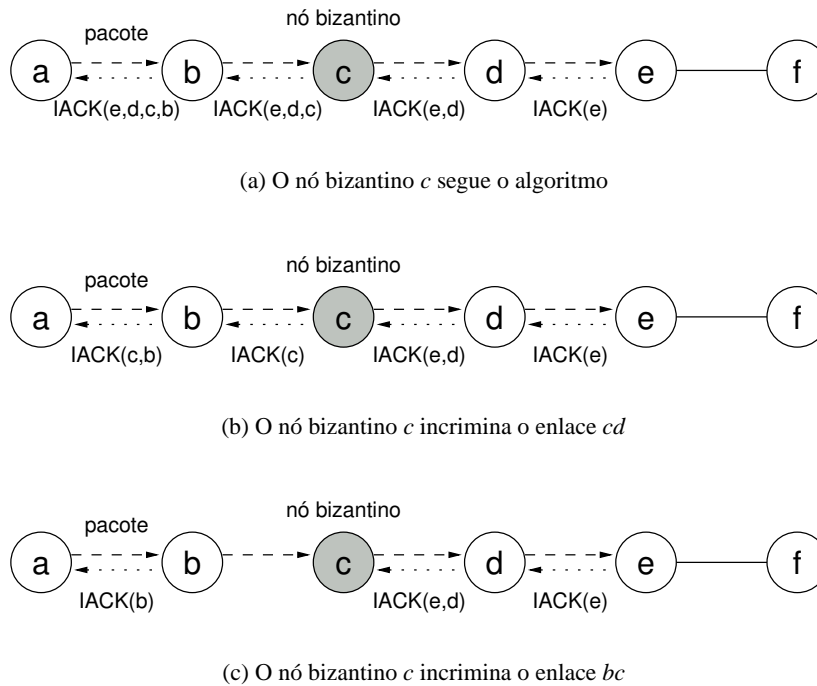


Figura 5.3: Localização de falhas com ACKs de nós intermediários. No exemplo, o nó a transmite dados para o nó f , mas os pacotes são perdidos no enlace ef . O nó c é bizantino.

incidência de falhas e intrusões um grande número de rotas ou enlaces pode ser declarado permanentemente inválido, o que diminui a redundância existente no *overlay*. A forma mais simples de combater esse tipo de ameaça é criar *overlays* com um grande número de nós (superdimensionamento), de tal sorte que a probabilidade de ocorrências dessa natureza seja reduzida a níveis aceitáveis. O problema dessa solução é que nem sempre se pode antecipar o que irá acontecer com a rede, e criar grandes *overlays* apenas para garantir a redundância pode representar um desperdício de recursos (mesmo os nós não utilizados participam dos protocolos de roteamento, por exemplo).

Essa situação levou ao desenvolvimento de um novo mecanismo de recuperação de falhas, que consiste na **reconfiguração dinâmica da topologia da rede**. Basicamente, esse mecanismo de reconfiguração permite que nós que tenham ficado isolados do restante do *overlay* sejam substituídos por nós de reserva, ativados dinamicamente. A ativação de um nó de reserva só é efetivada se esse nó possuir uma determinada conectividade, isto é, se ele conseguir estabelecer enlaces virtuais com um número mínimo de vizinhos.

As seções seguintes detalham o funcionamento dos protocolos de reconfiguração, discutem de que forma os parâmetros desses protocolos podem ser definidos, e oferecem algumas considerações sobre a reconfiguração.

5.3.1 Descrição dos Protocolos

A reconfiguração de topologia modifica a arquitetura ROTI apresentada na seção 3.2 criando uma divisão dos nós *overlay* entre **nós ativos**, que formam a topologia corrente da rede, e **nós de**

reserva, que podem se tornar ativos através da reconfiguração. A idéia é que a topologia do *overlay* seja formada por um número suficiente de nós, sem a necessidade de manter um conjunto grande de nós participando do *overlay* apenas para aumentar a redundância disponível; à medida em que essa redundância diminui, novos nós são ativados para suprir as ausências. A forma como é mantida a lista de nós de reserva não é especificada aqui; as possibilidades incluem carregar a lista manualmente em alguns nós, que se encarregariam de disseminá-la para os demais nós, ou então armazená-la em um serviço replicado tolerante a intrusões, que seria acessado pelos nós quando estes se juntassem à rede ou a determinados intervalos. Pressupõe-se que essa lista seja assinada por uma entidade confiável, como um dos emissores de certificados digitais usados na ROTI, e não possa ser forjada.

Quando um nó x exclui uma rota para o nó y ou um enlace local xy e o subsistema de roteamento é incapaz de fornecer uma rota alternativa para y , x conclui que y tornou-se inalcançável, e inicia um processo de reconfiguração propondo que um nó de reserva y' seja ativado para substituir y . O objetivo da reconfiguração é garantir que o nó sendo substituído não seja mesmo alcançável (para evitar que nós corretos sejam removidos da rede), e que o nó proposto como substituto tenha uma conectividade predeterminada (de modo a aumentar a probabilidade de que ele permaneça alcançável no futuro mesmo que ocorram novas faltas ou intrusões na rede).

A ativação de nós de reserva compreende dois protocolos, um de votação e outro de ativação propriamente dito. Esses protocolos, como todos os algoritmos distribuídos, devem satisfazer propriedades de *safety* e *liveness*, que podem ser enunciadas da seguinte forma:

Safety A substituição de um nó não pode ser causada por uma minoria de nós faltosos sem que haja a cooperação de nós corretos.

Liveness Uma proposição de reconfiguração que tem por objetivo substituir um nó isolado não pode ter o seu progresso impedido por uma minoria de nós ou enlances faltosos.

Para que essas propriedades possam ser satisfeitas, uma das duas premissas abaixo deve ser verdadeira:

- P1.** Se os nós e enlances faltosos forem removidos do *overlay*, a rede remanescente não é desconectada;
- P2.** Se os nós e enlances faltosos forem removidos do *overlay*, a rede remanescente possui no mínimo uma partição com pelo menos ϕ nós.

As premissas P1 ou P2 são necessárias para garantir tanto *safety* como *liveness*. Caso nenhuma delas seja verdadeira, é impossível garantir que os protocolos consigam ativar um nó de reserva. Das duas, P2 é evidentemente a mais fraca: ela garante que a partição com um mínimo de ϕ nós poderá se reconfigurar (no caso de haver mais de uma partição com o número mínimo de nós, o protocolo funcionará de maneira independente em todas elas), mas não permite que as partições com menos de ϕ nós sejam reconectadas aos componentes majoritários (aqueles com pelo menos ϕ nós).

A **fase de votação** utiliza um protocolo baseado em um algoritmo de consenso tolerante a faltas bizantinas com autenticação proposto por Cristian et al. [1995], e que está detalhado nos algoritmos 5.1 e 5.2. Os algoritmos 5.3 e 5.4, por sua vez, descrevem a **fase de ativação**. A tabela 5.1 descreve a semântica das funções primitivas usadas nesses algoritmos que não foram previamente descritas na tabela 4.1 na página 31.

Função	Semântica
ADDVOTE	adiciona o identificador do nó, o seu voto (YES ou NO) e a assinatura à lista de votação de uma mensagem de reconfiguração
VERIFY-RECONF	verifica se uma proposição de reconfiguração é válida (o emissor não foi forjado, as assinaturas digitais são corretas e não duplicadas, etc.)
COUNTVOTES	retorna o número de votos favoráveis (YES) em uma lista de votação
VALIDCERT	verifica se um certificado é válido para um determinado identificador de nó

Tabela 5.1: Funções usadas nos algoritmos 5.1 a 5.4

O protocolo de votação emprega o conceito de uma **lista de votação**, que registra os nós que já foram visitados por uma proposição de reconfiguração e o voto favorável (YES) ou contrário (NO) de cada um deles em relação a essa proposta. Essa lista de votação funciona de maneira análoga à rota acumulada usada no protocolo de roteamento reativo (discutida na seção 4.1.2), mas em vez de pares $\langle id, assinatura \rangle$ a lista de votação é formada por tuplas $\langle id, voto, assinatura \rangle$, sendo que a assinatura cobre toda a mensagem que contém a proposição de reconfiguração, da qual a lista de votação é o último campo.

Quando um nó decide que uma reconfiguração é necessária, ele chama o procedimento PROPOSE-RECONF (algoritmo 5.1). Esse procedimento é bastante simples: uma proposta de reconfiguração é formada a partir do nó a ser substituído e de seu possível substituto (linha 3), e o número de seqüência usado para reconfigurações é incrementado (linha 4). A seguir, a lista de votação é inicializada com o identificador, voto e assinatura do nó de origem (linha 5) e a mensagem de reconfiguração é construída (linha 6) e enviada para todos os vizinhos do nó (linha 7). O procedimento encerra com a inclusão da nova proposição no histórico de proposições H (linha 8). O nó que inicia uma reconfiguração é chamado de **emissor** dessa reconfiguração.

Algoritmo 5.1 Reconfiguração: PROPOSE-RECONF

```

1:  $seq \leftarrow 0; H \leftarrow \emptyset$ 
2: procedure PROPOSE-RECONF(in  $newNode, oldNode$ : NodeId)
3:    $R \leftarrow \langle newNode, oldNode \rangle$ 
4:    $seq \leftarrow seq + 1$ 
5:    $votes \leftarrow \langle myId, YES, SIGN(RECONF, myId, seq, R, myId, YES) \rangle$ 
6:    $rec \leftarrow \langle RECONF, myId, seq, R, votes \rangle$ 
7:   send  $rec$  to all neighbors
8:   add  $\langle myId, seq, R \rangle$  to  $H$ 
9: end procedure

```

O histórico de proposições é mantido para permitir a detecção de casos em que um nó bizantino inicia diversas reconfigurações com o mesmo número de seqüência. No histórico são mantidas tuplas da forma $\langle id, seq, R \rangle$, onde id é o identificador do nó, seq é o número de seqüência e R

é a reconfiguração proposta. Quando um nó recebe diferentes proposições de um emissor com o mesmo número de seqüência, ele considera que esse emissor é faltoso, e associa uma proposição nula (denotada pelo símbolo \perp , que não faz parte do conjunto de reconfigurações válidas) à entrada correspondente ao emissor e número de seqüência no histórico. Para evitar que esse histórico cresça indefinidamente, cada entrada tem um tempo de expiração Δ_H . Periodicamente, o histórico é varrido e as entradas expiradas são removidas.

O algoritmo 5.2 mostra o que acontece quando um nó recebe uma mensagem de reconfiguração m contendo uma proposta R . Primeiramente, verifica-se que a mensagem é correta (linha 2). Caso positivo, passa-se à análise da proposta de reconfiguração R : o histórico é inspecionado para ver se ele contém uma proposição R' do emissor $m.sender$ com o mesmo número de seqüência $m.seq$ (linha 4). Supondo que o emissor não tenha sido previamente identificado como faltoso (linhas 5–6) nem a mensagem seja uma duplicata (linhas 7–8), se existir uma proposição anterior $R' \neq R$, isso é um sinal inequívoco de que o emissor é faltoso, e o histórico é atualizado para sinalizar essa condição (linha 10). Por outro lado, se é a primeira vez que o nó está recebendo R , ele atualiza seu histórico (linha 12) e determina o seu voto (linhas 13–14). A seguir, ele atualiza a lista de votação contida em m com seu voto e assinatura, e difunde a mensagem para todos os seus vizinhos que ainda não estão nessa lista (linhas 15–17). Finalmente, o nó verifica se o quórum mínimo de votos ϕ foi atingido; caso afirmativo, ele invoca o procedimento START-RECONF, que dispara o processo de instalação da nova configuração (linhas 18–19).

O procedimento AGREE-WITH-RECONF, usado para determinar o voto de um nó com relação a uma reconfiguração proposta (linha 13), verifica duas condições:

1. O nó a ser substituído encontra-se inalcançável, o que significa que o nó que toma a decisão não possui nenhuma rota ativa — uma rota que esteja sendo usada para transmissão de pacotes e seja tida como correta — que inclua esse nó (como nó intermediário ou nó de destino) e que não passe pelo emissor da reconfiguração;
2. O nó proposto para ser ativado no lugar do original faz parte da lista de nós de reserva.

O nó vota a favor da reconfiguração proposta se, e somente se, ambas as condições forem satisfeitas.

A fase de ativação do nó de reserva é iniciada no procedimento START-RECONF (algoritmo 5.3). Essa fase exige que uma série de mensagens sejam trocadas entre os nós ativos e o nó de reserva:

1. Cada nó x que verifica a condição $COUNTVOTES(m.votes) \geq \phi$ (algoritmo 5.2, linha 18) envia uma mensagem $ACTIVATE(m)$ para o nó de reserva $r = m.R.newNode$ (algoritmo 5.3, linha 4), onde m é a mensagem de reconfiguração que comprova que r detém os votos necessários para ser ativado;
2. Ao receber uma mensagem $ACTIVATE(m)$ (algoritmo 5.4, linha 2), o nó de reserva r verifica que m contém pelo menos ϕ votos favoráveis à sua ativação (linha 4) e envia uma mensagem $ESTABLISH-LINK(m, myCert)$ para todos os nós que votaram a seu favor (linhas 6–9), onde $myCert$ é o certificado do nó de reserva;

Algoritmo 5.2 Reconfiguração: processamento de proposições de reconfiguração

```

1: upon receiving  $m = \langle \text{RECONF}, \text{sender}, \text{seq}, R, \text{votes} \rangle$  from  $x$  do
2:   if VERIFY-RECONF( $m$ ) then
3:      $\text{myVote} \leftarrow \text{NO}$ 
4:     if exists  $R'$  such that  $\langle m.\text{sender}, m.\text{seq}, R' \rangle$  is in  $H$  then
5:       if  $R' = \perp$  then
6:         { Faulty sender }
7:       else if  $R' = m.R$  and  $\text{myId}$  is in  $m.\text{votes}$  then
8:         { Duplicate message }
9:       else
10:        replace  $\langle m.\text{sender}, m.\text{seq}, R' \rangle$  with  $\langle m.\text{sender}, m.\text{seq}, \perp \rangle$  in  $H$ 
11:      else
12:        add  $\langle m.\text{sender}, m.\text{seq}, R \rangle$  to  $H$ 
13:        if AGREE-WITH-RECONF( $m.R$ ) then
14:           $\text{myVote} \leftarrow \text{YES}$ 
15:          ADDVOTE( $m, \text{myId}, \text{myVote}$ )
16:          for all neighbor  $n$  such that  $n$  is not in  $m.\text{votes}$  do
17:            send  $m$  to  $n$ 
18:          if COUNTVOTES( $m.\text{votes}$ )  $\geq \phi$  then
19:            START-RECONF( $m$ )
20:    end do

```

3. Cada nó s que recebe ESTABLISH-LINK(m, cert) de r confirma que a mensagem e o certificado que ela contém são válidos e envia uma mensagem ACK-LINK(s, r) para r (algoritmo 5.3, linhas 7–11);
4. Após ter recebido $n \geq \rho$ mensagens ACK-LINK, r difunde uma mensagem ACTIVE(AL), onde $AL = (\text{ACK-LINK}_1, \dots, \text{ACK-LINK}_n)$ é o conjunto de ACK-LINKs que prova que o nó conseguiu estabelecer um número suficiente de enlaces virtuais (algoritmo 5.4, linhas 11–16);
5. Ao receber uma mensagem ACTIVE válida, um nó x envia a sua cópia da lista de nós de reserva para r (caso tenha um enlace virtual com este) e retransmite a mensagem ACTIVE para os seus vizinhos que não fazem parte da lista de ACK-LINKs, encerrando a reconfiguração com a remoção do novo nó da lista de nós de reserva e o estabelecimento de chaves compartilhadas com r (algoritmo 5.3, linhas 12–21).

Um exemplo de execução bem sucedida do protocolo de reconfiguração (incluindo as fases de votação e de ativação) é mostrado na figura 5.4. A figura 5.4(b) mostra as mensagens que são trocadas pelos nó da rede *overlay* da figura 5.4(a) quando o nó e , isolado pela remoção de seus enlaces, é substituído pelo nó f . Os parâmetros de reconfiguração adotados no exemplo são $\phi = 3$ e $\rho = 2$. As mensagens da fase de votação são representadas por R, n_1, \dots, n_k , onde $R = \langle f, e \rangle$ e n_1, \dots, n_k são os nós que votam a favor de R . A figura não mostra as cópias da mensagem ACTIVE(AL) que são retransmitidas pelos nós a, c e d .

5.3.2 Definição dos Parâmetros de Reconfiguração

Basicamente, a reconfiguração da rede *overlay* é definida em função de dois parâmetros, ϕ e ρ . O parâmetro ϕ indica quantas mensagens de ativação são necessárias para que um nó de reserva possa

Algoritmo 5.3 Reconfiguração: START-RECONF

```

1:  $HA \leftarrow \emptyset; EM \leftarrow \emptyset; CN \leftarrow \emptyset$ 
2: procedure START-RECONF(in  $m$ : ReconfMsg)
3:    $r \leftarrow m.R.newNode$ 
4:   send ACTIVATE( $m$ ) to  $r$ 
5:   add  $\langle \text{ESTABLISH-LINK}(m, cert), r \rangle$  to  $EM$            {  $EM$  is the set of expected messages }
6: end procedure

7: upon receiving ESTABLISH-LINK( $m, cert$ ) from  $newNode$  do
8:   if ( $\langle \text{ESTABLISH-LINK}(m, cert), newNode \rangle$  is in  $EM$  or ( $\text{COUNTVOTES}(m.votes) \geq \phi$  and  $myId$  is in  $m.votes$ )) and  $\text{VALIDCERT}(cert, newNode)$  then
9:     send ACK-LINK( $myId, newNode, m.R, m.seq$ ) to  $newNode$ 
10:    add  $\langle \text{ACTIVE}(AL), newNode \rangle$  to  $EM$ 
11:  end do

12: upon receiving ACTIVE( $AL$ ) from  $x$  do
13:   if ACTIVE( $AL$ ) is not in  $HA$  and  $AL$  is valid then
14:     add ACTIVE( $AL$ ) to  $HA$                                {  $HA$  is the history of ACTIVE messages processed }
15:     add  $x$  to  $CN$                                          {  $CN$  is the set of contacted nodes }
16:     for all neighbor  $n$  such that  $n$  is not in  $CN$  do
17:       send ACTIVE( $AL$ ) to  $n$ 
18:       add  $n$  to  $CN$ 
19:     remove  $newNode$  from the list of backups
20:     set up keys with  $newNode$ 
21:  end do

```

Algoritmo 5.4 Reconfiguração: protocolo do nó de reserva

```

1:  $EM \leftarrow \emptyset; CN \leftarrow \emptyset; active \leftarrow \perp$ 
2: upon receiving ACTIVATE( $m$ ) from  $n$  do
3:   if VERIFY-RECONF( $m$ ) then
4:     if  $myId = m.R.newNode$  and ( $active = \perp$  or  $active = m.R$ ) and  $\text{COUNTVOTES}(m.votes) \geq \phi$  then
5:        $active \leftarrow m.R$ 
6:       for all  $s$  in  $m.votes$  such that  $s.vote = \text{YES}$  and  $s.id$  is not in  $CN$  do
7:         send ESTABLISH-LINK( $m, myCert$ ) to  $s$ 
8:         add  $s$  to  $CN$ 
9:         add  $\langle \text{ACK-LINK}(s, myId, m.R, seq), s \rangle$  to  $EM$ 
10:      end do

11:  upon receiving ACK-LINK( $n, id, R, seq$ ) from  $n$  do
12:    if  $\langle \text{ACK-LINK}(n, id, R, seq), n \rangle$  is in  $EM$  then
13:      add  $\langle \text{ACK-LINK}(n, id, R, seq), n \rangle$  to  $AL$            {  $AL$  is the set of ACK-LINKS }
14:      if  $|AL| \geq \rho$  then
15:        send ACTIVE( $AL$ ) to all neighbors
16:    end do

```

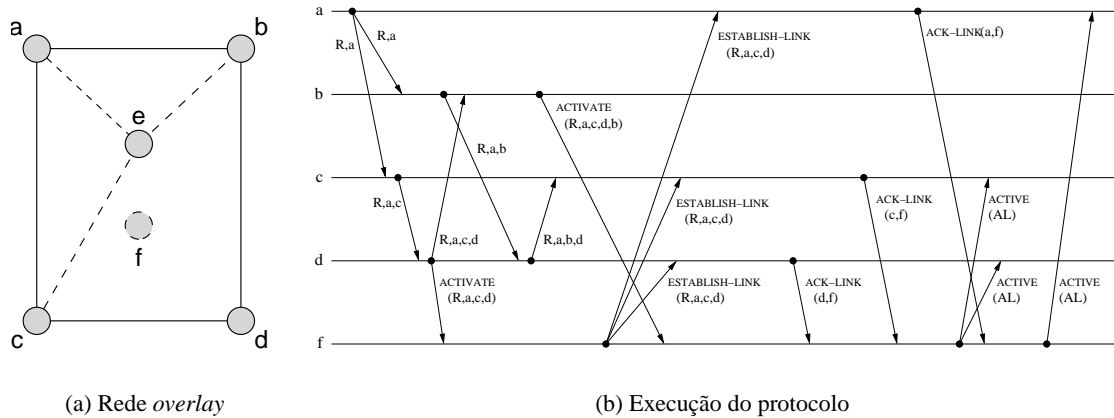


Figura 5.4: Execução do protocolo de reconfiguração: substituição do nó e pelo reserva f ($\phi = 3$, $\rho = 2$)

se tornar ativo no *overlay*, e o parâmetro $\rho \leq \phi$ determina com quantos nós, dentre os que enviam mensagens de ativação, o nó de reserva deve estabelecer enlaces virtuais.

Idealmente, não deve ser possível iniciar uma reconfiguração da rede exclusivamente pela ação de nós maliciosos (nós faltosos com comportamento bizantino), sem que nós corretos participem do processo. Devem ser igualmente evitadas situações em que um nó de reserva, ao tornar-se ativo, estabeleça enlaces virtuais apenas com nós faltosos, o que muito provavelmente viria a isolar o nó do restante da rede. A primeira condição está ligada ao parâmetro ϕ , enquanto que a segunda está ligada a ρ .

Evidentemente, não é possível definir parâmetros apropriados para que o protocolo de reconfiguração tolere um número arbitrário de faltas; os parâmetros só podem ser definidos em função de um número máximo de faltas. Considera-se então o problema de construir uma rede *overlay* que satisfaça as condições acima quando até f nós podem ser faltosos. Claramente, a segunda condição é satisfeita quando ρ é pelo menos $f + 1$. O teorema a seguir fornece um limitante inferior para ϕ .

Teorema 5.1. *Para que um nó de reserva não seja ativado apenas por nós faltosos e para que a ativação, uma vez iniciada, seja concluída, deve-se ter $\phi \geq 2f + 1$.*

Demonstração. Para determinar ϕ , é necessário lembrar que $\rho \leq \phi$; logo, já se sabe que o valor mínimo para ϕ é $f + 1$. Isso já garante que uma ativação não possa ser iniciada exclusivamente por nós faltosos, o que satisfaz a propriedade de *safety*. Por outro lado, deve-se considerar que os nós faltosos podem, em um primeiro momento, enviar mensagens de ativação para um nó de reserva r e, em um segundo momento, simplesmente se negar a estabelecer enlaces virtuais com r . Portanto, é necessário garantir que, dentre os ϕ nós que enviam mensagens de ativação, existam no mínimo ρ (ou $f + 1$) que certamente atenderão a requisições de estabelecimento de enlace virtual. Para que isso aconteça, devem ser exigidas pelo menos $\rho + f$ mensagens de ativação, o que resulta em $\phi \geq 2f + 1$ e satisfaz assim a propriedade de *liveness*. \square

5.3.3 Considerações Sobre a Reconfiguração de Topologia

5.3.3.1 Impacto Arquitetural da Reconfiguração

A reconfiguração de topologia é uma funcionalidade complementar aos mecanismos de detecção de falhas e recuperação de faltas apresentados nas seções 5.1 e 5.2. Esses mecanismos são os responsáveis por detectar falhas e remover elementos faltosos da rede, e a reconfiguração atua no sentido de restaurar a redundância da rede à medida em que esta vai sendo reduzida. Conjugando-se a possibilidade ou não de uso da reconfiguração com os dois esquemas de encaminhamento de pacotes descritos na seção 4.2, torna-se possível instanciar quatro variações da arquitetura, conforme mostra a tabela 5.2.

		Esquema de encaminhamento de pacotes	
		<i>leve</i>	<i>pesado</i>
Reconfiguração de topologia	<i>não</i>	arquitetura leve, sem reconfiguração	arquitetura pesada, sem reconfiguração
	<i>sim</i>	arquitetura leve, com reconfiguração	arquitetura pesada, com reconfiguração

Tabela 5.2: Variações da arquitetura ROTI

A arquitetura leve sem reconfiguração pode ser considerada a variante mais simples do ponto de vista conceitual, e aquela que utiliza os mecanismos menos custosos para proteção do tráfego (apenas um HMAC por pacote) e para localização de faltas (inexistente). Por outro lado, é a arquitetura que tende a exaurir com maior rapidez a redundância disponível na rede, pois descarta rotas inteiras em caso de falhas e não conta com nenhum mecanismo explícito para agregar nós à rede. Devido a isso, é a arquitetura que mais depende do superdimensionamento do *overlay* (o que reduz parte dos ganhos de desempenho pela sua simplicidade) e do ingresso voluntário de novos nós.

A arquitetura leve com reconfiguração é possivelmente a variante mais otimista, sendo a mais eficiente das quatro possibilidades em situações livres de falhas, pois o custo de proteção de tráfego é baixo e não há tanta dependência do superdimensionamento da topologia como na versão leve sem reconfiguração. Em contrapartida, em situações muito hostis essa arquitetura pode apresentar baixa confiabilidade, uma vez que invalida rotas inteiras quando detecta falhas e ainda precisa arcar com os custos do protocolo de reconfiguração, que compreendem não apenas as mensagens e assinaturas trocadas como também o tempo de execução do protocolo, que é imprevisível e pode não repor as redundâncias perdidas com a rapidez necessária.

A arquitetura pesada sem reconfiguração conta com uma detecção de falhas e recuperação de faltas mais precisa, uma vez que localiza e exclui enlaces faltosos no lugar de rotas inteiras. Em comparação com as arquiteturas leves, isso reduz a velocidade com que a redundância da rede é diminuída em função de falhas, o que implica menor dependência do superdimensionamento do *overlay*. Por outro lado, esta variante é aquela que apresenta maior custo fixo, mesmo em situações livres de faltas: além do custo da localização de faltas, o custo dos mecanismos criptográficos usados para

proteção do tráfego (ainda que condizente com o nível de proteção oferecido) e o custo de desempenho com os nós extras presentes na topologia para aumentar a redundância são maiores do que nas arquiteturas leves e na arquitetura pesada com reconfiguração. Por tais fatores, esta pode ser considerada a mais pessimista das quatro variantes da arquitetura ROTI.

A arquitetura pesada com reconfiguração é aquela que menos exige superdimensionamento da topologia, pois remove enlaces em vez de rotas e possui um mecanismo para repor a redundância da rede. Esta variante pode ser a mais indicada contra adversários perigosos que tenham alcance limitado (ou seja, adversários com grande capacidade de interferir no tráfego mas que não conseguem comprometer muitas rotas), ou quando o custo de nós adicionais é significativo. Por outro lado, a imprevisibilidade da latência do protocolo de reconfiguração pode constituir uma desvantagem para esta variante.

Essa análise das possibilidades de variação da arquitetura ROTI permite concluir que não existe nenhuma combinação de reconfiguração e esquema de encaminhamento de pacotes que se sobressaia perante as demais. Todas as variantes possuem virtudes e limitações que devem ser cuidadosamente avaliadas dentro do contexto em que se pretende implantar o *overlay*, não havendo uma arquitetura que seja universalmente melhor mas a alternativa mais indicada para uma determinada situação.

5.3.3.2 Aspectos do Protocolo de Reconfiguração

O protocolo de reconfiguração de topologia, devido ao seu caráter inovador (por ser o primeiro protocolo do gênero capaz de tolerar faltas bizantinas), possui alguns aspectos que ainda não estão bem amadurecidos e que precisam ser melhor aprofundados. Esta seção faz algumas breves considerações sobre dois desses aspectos, a seleção e a correção dos nós de reserva, e sobre o desempenho do protocolo expresso em termos de sua complexidade de mensagens.

Seleção de nós de reserva Nós substitutos são escolhidos ao acaso a partir de uma lista de nós de reserva. Essa lista é instalada manualmente nos primeiros nós do *overlay*, e é assinada pelos emissores de certificados da PKI. Quando um nó de reserva é ativado, os nós que estabelecem enlaces virtuais com ele lhe enviam suas cópias da lista. Não há como antecipar se um nó de reserva terá a conectividade necessária para ser ativado. Para resolver essa questão, pode-se definir um *timeout* de reconfiguração: se o nó de reserva escolhido não é ativado dentro do *timeout* especificado, a reconfiguração falha e uma nova reconfiguração (com um nó substituto diferente) é iniciada pelo nó que disparou a reconfiguração fracassada.

Correção dos nós de reserva O protocolo de reconfiguração pressupõe implicitamente que os nós de reserva são corretos quando eles são ativados. Garantir a validade dessa premissa em um contexto bizantino é difícil, e exigiria acrescentar um passo de verificação (que não pudesse ser ludibriado por um nó malicioso) ao protocolo de ativação de nós. Não obstante, ainda é melhor substituir um nó (comprovadamente) inalcançável por outro, mesmo que o novo nó venha a ser faltoso (o que acabará por redundar no seu isolamento do restante da rede e sua conseqüente substituição por outro nó).

Complexidade de mensagens Tanto o protocolo de votação como o protocolo de ativação de nós têm complexidade de mensagens $O(m)$, onde m é o número de enlaces da rede. Isto se deve ao fato de que uma mensagem qualquer passa no máximo duas vezes pelo mesmo enlace. Para uma rede completamente conectada com n nós, m é $n(n-1)/2$, e a complexidade de mensagens é $O(n^2)$.

5.4 Trabalhos Relacionados

A detecção de falhas e intrusões e a conseqüente recuperação de elementos faltosos em redes de computadores que satisfazem requisitos de segurança e confiabilidade é um tema abordado por diversos trabalhos na literatura. A exemplo do que foi feito no capítulo 4, esta seção examina as principais propostas sobre o assunto, revisando também os mecanismos adotados para esse fim nas redes *overlay* discutidas nas seções 2.2.1 e 2.2.2 (que, no entanto, não têm preocupações particulares com segurança).

A detecção de falhas na RON [Andersen et al., 2001] utiliza *pings*. Periodicamente, um nó a envia uma mensagem para um de seus vizinhos (por exemplo, o nó b), registrando o instante do envio. Ao receber o *ping*, b envia uma resposta para a e zera o *timer* de detecção de falhas (isto é, de envio de *pings*) associado a este nó. Quando a resposta chega ao nó a , este tem então uma amostra de RTT para a rota $a-b$ (comparando o instante de envio do *ping* com o instante de chegada da resposta), e envia uma réplica para b de modo que este também possa obter uma amostra de RTT para a rota $b-a$. Caso uma mensagem dessas não seja respondida dentro de um determinado *timeout*, uma perda é registrada, e um novo *ping* é enviado imediatamente. Considera-se que uma falha ocorre quando um número de *pings* em seqüência (quatro, por *default*) são dados como perdidos. A recuperação de faltas na RON é bastante simples: o processo de seleção de rotas utiliza apenas os caminhos considerados corretos no momento da seleção, ignorando aqueles que se encontram falhos. Quando um caminho volta a responder a *pings* ele é considerado restaurado e volta a ser usado na seleção de rotas.

O SOSR [Gummadi et al., 2004], a exemplo da RON, também usa *pings* para detectar falhas. A frequência de envio de *pings* é de um a cada 15 segundos, e o *timeout* para detectar uma perda é de três segundos. Uma falha é constituída por três perdas consecutivas (semelhante à RON, que considera quatro perdas consecutivas). Durante a ocorrência de uma falha, o intervalo entre *pings* para o vizinho inalcançável cai para cinco segundos, e o enlace virtual com esse vizinho só volta a ser considerado funcional após dez respostas corretas em seqüência. A recuperação de faltas consiste em rotear os pacotes através de um nó intermediário escolhido ao acaso, como mencionado na seção 4.3.1.

Conforme discutido na seção 5.1, a detecção de falhas baseada em *pings*, como a usada por RON e SOSR, é inadequada para situações em que os nós podem falhar de forma arbitrária. Por exemplo, como o tráfego de controle é separado do tráfego de dados, nós que prejudicam transmissões de dados mas respondem adequadamente a *pings* serão considerados corretos. Outro problema é que, como o tráfego de controle pode ser facilmente distinguido do tráfego de dados, um nó malicioso (do *overlay* ou da rede subjacente) pode interferir na detecção de falhas, descartando *pings* ou *pongs* que passem por ele de modo a fazer com que determinados nós, enlaces ou rotas sejam considerados

falhos. Na ausência de mecanismos criptográficos, um nó malicioso pode ainda forjar respostas a *pings* endereçados a nós que não estão ativos, fazendo com que outros nós acreditem que as rotas para esses nós permaneçam funcionais. Essas mesmas vulnerabilidades tornam as estimativas de RTT obtidas através de *pings*, como ocorre na RON, pouco confiáveis, já que elementos bizantinos podem influenciá-las com alguma facilidade: o tráfego de controle pode ser atrasado, aumentando o RTT, ou mesmo ser priorizado no encaminhamento de pacotes ou encaminhado através de um túnel de baixa latência, reduzindo assim o RTT. Essas vulnerabilidades não afetam a detecção de falhas com base no comportamento fim a fim das rotas, usada na ROTI: como não há tráfego específico de detecção de falhas, são detectados apenas os eventos que afetam a transmissão de dados, e um nó bizantino não pode prejudicar o tráfego de dados sem que isso seja refletido na detecção de falhas. Da mesma forma, as amostras de RTT e perda são extraídas do tráfego de dados, e refletem o comportamento das rotas percebido pelas aplicações.

Dos dois esquemas de roteamento introduzidos por Perlman [1988] (seção 4.3.1), a inundação robusta prescinde de detecção de falhas ou recuperação de faltas: sempre que houver um caminho livre de faltas entre nó de origem e nó de destino esse caminho acabará sendo usado para a transmissão de dados entre os nós. Quando é usado roteamento baseado em estado de enlaces, porém, torna-se necessário verificar se a rota usada é correta ou não. Perlman não define nenhum mecanismo específico para esse fim; ela presume que a camada superior irá detectar problemas com uma rota corrente e notificar isso à camada de rede. Ela examina ainda uma série de providências que podem ser tomadas quando uma notificação dessa natureza é recebida: (i) selecionar uma rota disjunta da rota faltosa; (ii) manter uma lista de roteadores que fazem parte de rotas previamente detectadas como faltosas e evitá-los, na medida do possível, durante a seleção de rotas; (iii) usar ACKs intermediários (semelhantes aos usados no esquema pesado da ROTI, conforme descrito na seção 5.2.2) para localizar roteadores faltosos; e (iv) recorrer à inundação robusta quando a rota escolhida não funciona. Embora discuta prós e contras de todas essas opções, ela não escolhe nenhuma delas em particular, deixando em aberto esse aspecto da arquitetura.

Uma abordagem intermediária entre detecção por *pings* e por análise do comportamento de rotas foi proposta por Cheung e Levitt [1997]. Nessa abordagem, os nós executam um protocolo para testar seus vizinhos, verificando se eles encaminham pacotes corretamente. O protocolo consiste em enviar ao nó testado um pacote de dados cujo destino é o próprio testador; se esse pacote for recebido pelo testador, o nó testado funciona corretamente. Esse protocolo depende de várias premissas, tais como: (i) o nó testado deve ser incapaz de distinguir o pacote de teste de tráfego regular; (ii) ele não pode identificar o endereço de destino do pacote como pertencendo ao testador; (iii) o pacote deve ser enviado através do mesmo enlace nos dois sentidos (do testador para o testado e vice-versa); e (iv) o nó testado, se for faltoso, não pode agir de forma discriminatória contra nós específicos. Essa abordagem representa um avanço em relação ao uso de *pings*, mas o grande número de premissas que devem ser satisfeitas a tornam uma alternativa inferior à análise do comportamento fim a fim das rotas.

Os mesmos autores propõem ainda um esquema mais elaborado chamado WATCHERS [Cheung e Levitt, 1997; Bradley et al., 1998]. O WATCHERS é baseado em análise de tráfego e no princípio de **conservação de tráfego**, que diz que o tráfego que chega em um roteador deve ser igual à soma do

tráfego que sai do roteador mais o tráfego destinado ao próprio roteador ou a uma das redes locais à qual ele está conectado. Um roteador que viola essa condição é considerado faltoso. Para detectar esse tipo de violação, cada roteador R_i mantém contadores de tráfego para três categorias de pacotes: pacotes em trânsito, pacotes destinados ao roteador R_j (ou a uma de suas redes locais) e pacotes originados no roteador R_j (ou em uma de suas redes). São usados contadores separados para cada roteador vizinho e para cada direção de tráfego de/para esse vizinho. Além disso, presume-se que os roteadores sejam capazes de mensurar o tráfego desviado (encaminhado através de uma rota sub-ótima) por seus vizinhos. Periodicamente, um roteador inicia o protocolo de testes inundando uma requisição, que deve ser confirmada pela maioria dos outros roteadores. A seguir, cada roteador troca os contadores de tráfego com os outros roteadores que se encontram a até dois *hops* de distância (ou seja, seus vizinhos e os vizinhos desses vizinhos), e compara os contadores recebidos com os seus próprios para verificar se existem roteadores violando a conservação de tráfego, que são então considerados faltosos. A exemplo do protocolo anterior, o funcionamento deste esquema também depende de um grande número de premissas para as quais é difícil garantir a cobertura, tais como: (i) os testadores devem conhecer as tabelas de roteamento de todos os seus vizinhos; (ii) roteadores corretos nunca descartam pacotes; (iii) todos os roteadores têm uma visão idêntica do estado da rede; e (iv) cada par de roteadores corretos está diretamente conectado. Essas diversas premissas comprometem a aplicabilidade desse esquema de detecção de falhas em uma rede real. Uma análise mais detalhada das vulnerabilidades do WATCHERS pode ser encontrada em [Hughes et al., 2000], onde são propostas também medidas para contornar essas vulnerabilidades ou minimizar seu impacto.

A proposta de Avramopoulos et al. [2004] usa, a exemplo da ROTI, reconhecimentos para detectar falhas, mas o esquema de detecção e de localização de faltas é mais complexo. Nesse esquema, ao encaminhar um pacote, cada roteador dispara um *timeout* para a recepção do ACK correspondente: caso esse ACK não chegue dentro do *timeout* estipulado, o nó envia para a origem um anúncio de falta (FA — *fault announcement*), que indica que o enlace com o seu sucessor na rota (pelo qual o pacote foi enviado para o destino) é faltoso. Ao receber o FA, o nó de origem remove o enlace faltoso da sua visão da topologia da rede, que é usada no cálculo de rotas (ou seja, a detecção de falhas é perfeita [Chandra e Toueg, 1996], não admitindo equívocos). Esse esquema funciona porque os *timeouts* dos nós sucessivos que compõem a rota são definidos em função do RTT de pior caso até o destino, de tal forma que, quando o *timeout* de um pacote expira, o ACK ou o FA correspondente já deveria ter sido recebido do próximo nó na rota (se isso não aconteceu, a falta pode ser atribuída ao enlace entre o nó que a detecta e o seu sucessor). A possibilidade de falsificação de FAs é minimizada por dois fatores:

1. Todos os roteadores que um FA percorre até chegar à origem verificam não só a autenticidade desse FA (através do MAC correspondente a cada roteador) como também a sua legitimidade — se o número de seqüência corresponde ao de um pacote de dados encaminhado previamente, se o FA não é uma duplicata, etc.¹
2. Quando um roteador i diz que um de seus enlaces ij é faltoso, um de dois casos pode acontecer:
 - i está falando a verdade (o FA é verdadeiro). Neste caso, ij deve mesmo ser evitado;

¹ACKs também passam por uma verificação de legitimidade análoga a esta.

- i está mentindo (o FA é falso). Neste caso, i é faltoso e seria desejável evitá-lo como parte de uma rota. Como o protocolo prevê a remoção de enlaces, ij é excluído de modo a reduzir os caminhos de acesso a i (a idéia é que todos os enlaces de i acabem sendo removidos ao longo do tempo).

Em princípio, os FAs só devem ser usados pelo nó de origem ao qual são destinados. Entretanto, os autores ainda propõem mecanismos para que um nó compartilhe os FAs que recebe com outros nós, e para bloquear tráfego vindo de roteadores faltosos. A principal desvantagem desses mecanismos é que, para lidar com a falta de confiabilidade das informações de detecção de falhas, os roteadores passam a ter que calcular rotas duas vezes para cada FA recebido. Comparando com a ROTI, a complexidade adicional desse esquema parece injustificada, uma vez que exige que roteadores intermediários mantenham uma grande quantidade de informações de estado sobre os pacotes que encaminham (para verificar a legitimidade de ACKs e FAs) sem melhorar significativamente a precisão ou a rapidez na detecção. Os mecanismos para compartilhamento de FAs e bloqueio de tráfego aparentam ser ainda menos recomendáveis, pois acrescentam complexidade ao cálculo de rotas sem oferecer grandes benefícios. Outra grande desvantagem é a inflexibilidade do esquema em lidar com equívocos na detecção de falhas, algo admitido pelos próprios autores [Avramopoulos et al., 2004].

Fatih [Mizrak et al., 2005] é um conjunto de protocolos para detecção e recuperação de comportamento bizantino de roteadores. Basicamente, a abordagem é a seguinte: durante um período, os roteadores coletam dados sobre o tráfego que flui pelos diferentes caminhos na rede. A intervalos previamente acordados, os roteadores trocam os dados coletados para identificar caminhos faltosos; se um caminho é considerado faltoso, essa informação é comunicada (usando um protocolo de difusão confiável [Hadzilacos e Toueg, 1993]) para todos os outros roteadores e esse caminho é removido das tabelas de rotas dos roteadores. As informações armazenadas sobre os pacotes que trafegam em cada direção de um caminho incluem, além de contadores de tráfego similares aos usados no WATCHERS, listas de *hashes* dos pacotes, que permitem detectar alterações de conteúdo; os autores propõem ainda manter essas listas ordenadas para detectar reordenação de pacotes, um fenômeno que pode prejudicar o desempenho do TCP [Bennett et al., 1999; Bellardo e Savage, 2002]. Eles não propõem nenhum mecanismo de roteamento, adotando como premissa a existência de um protocolo intra-domínios baseado em estado de enlaces; além disso, eles não usam roteamento na origem, mas pressupõem que seja possível prever o caminho pelo qual um pacote será encaminhado. Comparado à ROTI, esse trabalho não tenta coagir roteadores faltosos a se comportarem corretamente: os pacotes não são cifrados nem autenticados durante a transmissão, o que dá a roteadores maliciosos maior liberdade para interferir seletivamente no tráfego. A detecção de falhas nessa proposta requer consenso, enquanto na ROTI cada nó tem autonomia para escolher quais rotas ou enlaces evitar na rede. Além disso, no Fatih quando uma falha é detectada o caminho faltoso é sumariamente removido das tabelas de rotas, o que não permite acomodar falhas transientes (devidas a congestionamento, por exemplo) ou detecções incorretas — problemas que a quarentena adotada na ROTI foi explicitamente projetada para contornar.

Listen [Subramanian et al., 2004] é um protocolo usado para detectar problemas de alcançabilidade no plano de dados (ou seja, rotas não funcionais) no contexto do BGP. A idéia básica do Listen

é monitorar fluxos TCP para determinar se a rota selecionada para um destino funciona ou não. Um fluxo é considerado completo se um SYN (tentativa de estabelecimento de conexão) é seguido por um DATA (segmento contendo dados); se um SYN não for seguido por DATA dentro de dois minutos, o fluxo é considerado incompleto. Para detectar falhas, um roteador examina os fluxos TCP cujos endereços de destino compartilham o mesmo prefixo (e, portanto, a mesma rota BGP): quando a fração de fluxos completos cai abaixo de um limiar aceitável, a rota para esse prefixo é considerada faltosa. Como essa estratégia de detecção de falhas é bastante simples e sujeita a equívocos (tanto devido a tráfego que ocorre naturalmente na rede como pela ação de nós maliciosos), são propostos alguns mecanismos para reduzir a probabilidade de falsos positivos e negativos. Por outro lado, os autores não propõem nenhuma forma de recuperação, presumindo que as falhas detectadas sejam informadas a um operador humano para que este tome providências. Comparado com a detecção de falhas da ROTI, o Listen testa apenas a conectividade de rotas, sem verificar ocorrências como o desvio ou a modificação de pacotes. Desta forma, uma fração significativa de comportamento malicioso não é detectada pelo protocolo, e as informações sobre detecção de falhas não são suficientemente confiáveis para serem usadas por algum mecanismo automatizado de recuperação ou resposta.

Existem ainda propostas relacionadas no âmbito de protocolos seguros de roteamento para redes *ad hoc* móveis (MANETs). O BFTR [Xue e Nahrstedt, 2004] é um protocolo de roteamento sob demanda baseado no DSR [Johnson et al., 2001]. A seleção de rotas usa o índice de perdas como métrica para determinar se uma rota é viável ou não (rotas viáveis são aquelas cujo índice de perdas encontra-se abaixo de um determinado limiar), escolhendo sempre a rota mais curta dentre as viáveis. A mesma métrica é usada para detectar rotas faltosas: quando uma rota em uso passa a ser considerada inviável, uma nova rota viável é selecionada. O esquema de detecção de falhas e recuperação de faltas nas arquiteturas leves da ROTI é bastante similar ao do BFTR, mas a ROTI utiliza também a latência como métrica de detecção de falhas (no BFTR, a latência é considerada apenas indiretamente, na medida em que pacotes que levem muito tempo para chegar ao destino sejam computados como perdas pelo nó de origem). Outra proposta relevante no contexto do MANETs é o OSDDBR [Awerbuch et al., 2002], que utiliza ACKs intermediários para localizar enlaces faltosos e um mecanismo de quarentena de enlaces semelhante ao usado nas arquiteturas pesadas da ROTI.

5.5 Conclusões do Capítulo

Este capítulo apresentou os mecanismos usados na ROTI para detectar falhas e restaurar a funcionalidade da rede quando elas ocorrem. A detecção de falhas baseia-se na análise do comportamento fim a fim das rotas mensurado através da sua taxa de perdas e sua latência, uma estratégia que permite detectar tanto falhas simples, como quedas de roteadores e enlaces congestionados, como comportamento intrusivo causado por elementos maliciosos internos ou externos ao *overlay*. Outra grande vantagem dessa estratégia é que ela impede que comportamento nocivo passe despercebido na rede, pois na medida em que ele seja prejudicial à comunicação entre os nós ele será detectado (um eventual comportamento malicioso que não prejudique as comunicações é, em termos de efeitos práticos, inofensivo, e portanto não exige uma reação dos demais nós).

A recuperação de faltas consiste, no nível mais básico, em evitar a rota ou o enlace detectado como faltoso. Para minimizar os efeitos de detecções errôneas e acomodar faltas transientes é adotado um mecanismo de quarentena. Um outro mecanismo de recuperação considerado envolve a reconfiguração da topologia do *overlay* através da substituição de nós inalcançáveis por outros nós que respeitem um determinado grau mínimo de conectividade. Esse protocolo de reconfiguração, que é uma das principais contribuições desta tese, permite que uma rede *overlay* seja formada por um grande número de nós que podem ser ativados sob demanda. Isso oportuniza a construção de uma ampla infra-estrutura, compartilhada por diversas organizações e espalhadas por vários provedores de acesso, onde os nós vão sendo incorporados a *overlays* conforme a necessidade provocada por faltas e intrusões na rede.

No capítulo a seguir, são mostrados diversos resultados obtidos no desenvolvimento da ROTI. Esses resultados compreendem uma análise da segurança oferecida pela rede, medidas de confiabilidade derivadas de um modelo analítico e dados extraídos de simulações dos protocolos da rede.

Capítulo 6

Resultados

Os capítulos anteriores apresentaram a arquitetura geral da ROTI, detalhando os seus subsistemas de roteamento, encaminhamento de pacotes e detecção de falhas e recuperação de faltas.

Recordando o que foi exposto no capítulo 3, o objetivo primário da ROTI é possibilitar a comunicação entre nós mesmo na presença de faltas e intrusões, quer no *overlay* ou na rede subjacente. Além disso, uma preocupação que permeia o desenvolvimento do trabalho — e que pode, portanto, ser considerada uma meta secundária — é com a relação custo-benefício dos mecanismos que são usados para se chegar a esse objetivo principal, que devem ser tão eficientes quanto possível. A principal questão que ainda precisa ser respondida no contexto desta tese diz respeito à eficácia da arquitetura proposta em atingir esses dois objetivos. O propósito deste capítulo é, então, oferecer respostas a essa questão.

O capítulo combina resultados analíticos e experimentais para mostrar como os mecanismos de prevenção e de tolerância adotados na ROTI concorrem para a realização dos propósitos da rede. Dada a complexidade inerente à avaliação de sistemas sujeitos a ameaças de segurança, a estratégia adotada é multifacetada, considerando aspectos complementares da arquitetura da rede.

Inicialmente, a seção 6.1 examina a eficácia dos mecanismos usados para garantir as propriedades de segurança na rede, verificando em que medida a ROTI resiste às ameaças de segurança apresentadas na seção 2.3.

A seção 6.2 usa uma formulação baseada em teoria de grafos para demonstrar que a confiabilidade da ROTI depende das topologias da rede *overlay* e da rede subjacente, e de como essas topologias interagem. Essa formulação é então usada para fazer uma análise de pior caso do grau de tolerância a faltas e intrusões de uma dada topologia de rede *overlay*.

A seção 6.3 discute a implementação da ROTI em um simulador, e apresenta resultados experimentais que demonstram a alta confiabilidade efetivamente obtida com a ROTI e o baixo *overhead* imposto pelo seu protocolo de roteamento. Na sequência, a seção 6.4 realiza uma comparação global da ROTI com outras experiências similares registradas na literatura.

6.1 Análise de Segurança

A segurança oferecida pela ROTI pode ser analisada de forma qualitativa, visando determinar a eficácia da rede em proteger a confidencialidade, a integridade e a disponibilidade das comunicações. A análise apresentada nesta seção foi conduzida da seguinte forma: as ameaças a *overlays* de roteamento identificadas na seção 2.3 foram revisitadas, verificando-se a sua relevância para o contexto da ROTI e em que medida essas ameaças são efetivamente combatidas pelos mecanismos adotados na rede. A análise considera tanto as ameaças que afetam o plano de controle — isto é, a segurança do roteamento — como aquelas que afetam o plano de dados, ou seja, a transmissão de dados de aplicação propriamente dita.

6.1.1 Confidencialidade

Das três propriedades fundamentais de segurança, a confidencialidade é a menos explorada na ROTI: mecanismos de cifragem são usados para garantir a confidencialidade apenas no plano de dados, e somente no esquema pesado de encaminhamento de pacotes. Essa decisão de projeto foi motivada pelo fato de que muitas aplicações possuem requisitos de integridade e disponibilidade, mas não de confidencialidade; como o uso de mecanismos de cifragem impacta negativamente o desempenho das comunicações, optou-se por não adotá-los sistematicamente na arquitetura leve. É importante observar que não existe nenhum tipo de dificuldade em implementar essa cifragem na arquitetura leve; considerando-se que chaves criptográficas simétricas já precisam ser estabelecidas para o uso de HMACs, seria trivial estender o processo de negociação de chaves para gerar uma chave adicional para cifragem dos dados usando um algoritmo simétrico (as boas práticas criptográficas condenam o uso da mesma chave simétrica para mais de um propósito). Uma opção seria aumentar a API de usuário de forma a permitir o uso facultativo de cifragem na transmissão de dados, seja através de um mecanismo de configuração ou usando uma primitiva diferente de `send`. Por outro lado, o uso sistemático de cifragem na arquitetura pesada, sob a forma de *onion encryption*, se justifica pelo fato dessa arquitetura ser mais indicada para uso em ambientes mais hostis, em que há preocupação com análise de tráfego.

Do ponto de vista do plano de controle, usualmente as informações de roteamento são consideradas de conhecimento público, o que dispensa o uso de mecanismos de cifragem. Isso ocorre porque, em geral, o comprometimento de um roteador é o suficiente para revelar um número significativo de informações sobre a topologia da rede, que estão contidas na tabela de rotas desse roteador. A quantidade efetiva de informações obtidas depende do protocolo de roteamento adotado; no caso da ROTI, todos os nós atuam como roteadores, e o comprometimento de um nó revela todas as rotas diretas e simples desse nó (ou seja, todas as rotas para destinos a até dois *hops* de distância) e mais algumas rotas eventualmente obtidas por roteamento reativo. Portanto, a ROTI não emprega nenhum mecanismo para garantir a confidencialidade no plano de controle.

6.1.2 Integridade

A seção 2.3 identifica três grandes ameaças à integridade de *overlays* de roteamento: modificação não autorizada, falsificação de mensagens e ataques de *replay*. No plano de dados, as duas primeiras são combatidas com o uso de códigos de autenticação de mensagens (HMACs). HMACs não evitam que um elemento malicioso efetue ataques dessa natureza (na verdade, não existe nenhum mecanismo que possa evitar tais ocorrências), mas garantem que esses ataques sejam detectados. Nesse ponto, a diferença entre a arquitetura leve e a arquitetura pesada reside em quem é capaz de realizar a detecção. No caso da arquitetura leve, onde cada pacote contém apenas um HMAC, somente o nó de destino tem como saber se um pacote foi modificado ou forjado. Na arquitetura pesada, como existem HMACs para cada nó pertencente à rota, qualquer nó intermediário é capaz de detectar essas ocorrências. O plano de controle, por sua vez, utiliza assinaturas digitais para essa mesma funcionalidade. O uso de assinaturas se justifica porque o número de nós que precisam verificar a integridade e a autenticidade das mensagens de roteamento é muito grande, no caso da fase proativa do protocolo, ou desconhecido, no caso da fase reativa. Essa imprevisibilidade torna o uso de HMACs contra-indicado (pois cada receptor exige um HMAC específico), mas não afeta o uso de assinaturas (pois qualquer receptor pode verificar uma assinatura).

Para combater ataques de *replay*, tanto o plano de controle como o plano de dados usam números de seqüência únicos para cada mensagem. Esses números permitem ao nó que recebe um pacote verificar se ele já o recebeu anteriormente ou não. No plano de dados, essa verificação cabe apenas ao nó de destino; no plano de controle, como uma mesma mensagem pode ser processada por vários nós, todos eles fazem essa verificação.

6.1.3 Disponibilidade

A disponibilidade de um *overlay* de roteamento pode ser ameaçada, segundo a seção 2.3, de três formas diferentes: descarte de pacotes, sobrecarga e desvio de pacotes. Cabe aqui frisar que essas ameaças estão entre as mais complicadas de serem combatidas, pois é virtualmente impossível forçar um nó bizantino a se comportar corretamente (o que seria necessário para evitar que pacotes fossem desviados ou descartados), e os ataques de negação de serviço por sobrecarga se baseiam na dificuldade de distinguir entre tráfego legítimo e tráfego malicioso antes que esse tráfego chegue até as vítimas dos ataques (quando é, em geral, tarde demais para tomar alguma providência). Por essas razões, os mecanismos usados contra esse tipo de ameaça são muito mais de natureza reativa (isto é, em resposta a ocorrências detectadas) do que de natureza preventiva.

O descarte de pacotes, que é possivelmente a forma mais eficaz de negação de serviço em uma rede, é tratado de forma diferente no plano de dados e no plano de controle. No plano de dados, os mecanismos de detecção de falhas e recuperação de faltas são responsáveis por detectar pacotes descartados ou perdidos e fazer com que sejam retransmitidos, pela mesma rota ou por uma rota alternativa. Nós que descartam um número grande de pacotes ficam então sujeitos a terem as rotas passando por eles identificadas como faltosas e substituídas por outras, neutralizando assim a sua ação

maliciosa. No plano de controle, por sua vez, não há persistência na transmissão de mensagens: um nó não controla se as mensagens de roteamento que ele envia são recebidas ou não. Na fase proativa do protocolo de roteamento, as atualizações de rotas são enviadas periodicamente. Sendo assim, se poucas atualizações forem perdidas as atualizações subsequentes suprirão a sua ausência. Caso ocorra um descarte sistemático de atualizações por parte de um nó da rede subjacente, por exemplo, a relação de vizinhança entre os nós afetados acabará por ser corretamente desfeita, uma vez que isso afigura comportamento faltoso do enlace virtual entre esses nós. Na fase reativa, o descarte intencional de ROUTE-REQUESTS, ROUTE-REPLYS ou CACHE-REPLYS não é considerado problemático, uma vez que a ocorrência desse descarte significa que existe um nó ou enlace faltoso na própria rota que está sendo propagada através dessas mensagens (o que implica uma boa chance de que a rota será faltosa). Por outro lado, o protocolo reativo pode sofrer com perda ocasional de pacotes: por exemplo, caso um ROUTE-REPLY seja perdido ao ser transmitido por uma rota correta (ou seja, cujo índice de perdas esteja abaixo do limiar de detecção de falhas), a rota não será propagada até o nó de origem que a requisitou. Isso pode ser resolvido acrescentando um mecanismo de retransmissão ponto a ponto ao protocolo de roteamento reativo: caso não receba um ACK dentro de um *timeout* especificado, um nó retransmite a mensagem de roteamento para o vizinho correspondente. Esse mecanismo é simples, eficaz contra as perdas acidentais que ocorrem na prática na Internet e não prejudica o funcionamento do protocolo; a desvantagem é que ele gera um pequeno tráfego adicional de controle.

O desvio de pacotes é um pouco mais simples de tratar do que o descarte. No plano de dados, como cada pacote possui no cabeçalho a rota pela qual deve ser transmitido, um nó correto que recebe um pacote desviado e identifica alguma discrepância com relação a essa rota especificada (ele não pertence à rota, o *hop* anterior não é o seu predecessor na rota ou o sucessor especificado não é um de seus vizinhos) conclui que se trata de um pacote desviado que pode ser descartado. Um ataque mais sutil consiste em modificar o campo ROUTE antes de enviar o pacote pela rota incorreta. Nesse caso, a detecção do ataque passa a ser feita pelos mecanismos de detecção de modificação não autorizada e não mais pelo mecanismo de verificação da rota especificada: na arquitetura leve, o nó de destino percebe que o pacote foi modificado e o descarta, e na arquitetura pesada o primeiro nó correto que recebe o pacote verifica que o seu HMAC é inválido e também elimina o pacote. No plano de controle, ataques de desvio de pacotes só se aplicam a ROUTE-REPLYS e CACHE-REPLYS, uma vez que ROUTE-UPDATES e ROUTE-REQUESTS são transmitidos por difusão. O mecanismo usado é o mesmo do plano de dados, de verificação da rota especificada nas mensagens. Embora o desvio de pacotes seja relativamente simples de detectar, como os pacotes desviados são descartados pelos nós que percebem o desvio, essa ocorrência se manifesta para o nó de origem como uma perda. O inconveniente disso é que a necessidade de recorrer a *timeouts* para detecção de perdas significa que esse tipo de comportamento malicioso pode levar um certo tempo até ser detectado e as providências apropriadas serem tomadas.

A última ameaça considerada nesta análise é a sobrecarga. No contexto da ROTI, os dois principais tipos de sobrecarga que precisam ser considerados são:

1. Sobrecarga de processador por excesso de operações criptográficas;
2. Sobrecarga de enlaces virtuais ou físicos por excesso de tráfego.

O primeiro tipo afeta muito mais o plano de controle do que o plano de dados. Isso ocorre porque os protocolos do plano de controle usam assinaturas digitais, que são computacionalmente muito mais dispendiosas do que os HMACs usados no plano de dados. Sendo assim, considera-se que no plano de dados a possibilidade de sobrecarga de processador devido à criptografia pode ser desprezada, e não são adotadas medidas específicas para tratar desse problema. No plano de controle, porém, ele não pode ser ignorado, e a medida adotada para aliviar o seu impacto é a limitação na taxa de emissão de mensagens de roteamento, que são aquelas que usam assinaturas digitais. Basicamente, a idéia é que um nó processe um número limitado de mensagens assinadas em um curto espaço de tempo, o que evitaria a sobrecarga do processador. Uma dificuldade de efetuar esse controle da taxa de emissão é que, devido à variabilidade na latência da rede, mensagens emitidas dentro da frequência permitida podem chegar a um nó em um intervalo menor (a primeira mensagem é atrasada e a segunda chega pouco tempo depois dela), ou o contrário (duas mensagens emitidas em curta seqüência podem chegar em um intervalo bem espaçado). Para evitar esses problemas, a idéia é que cada nó mantenha a mensagem mais recente de um emissor em um *buffer* até o instante em que ela pode ser legalmente processada. Isso elimina imediatamente o primeiro problema. Quanto ao segundo problema, se um nó mantiver apenas uma mensagem por emissor no *buffer*, ele terá somente essa mensagem para processar no momento adequado, não importando quantas mensagens ele recebeu desse emissor.

A sobrecarga de nós e enlaces por excesso de tráfego é mais difícil de resolver. Uma solução preconizada por Perlman [1988] é de usar mecanismos que garantam justiça (*fairness*) no acesso aos recursos de rede (basicamente *buffers* e enlaces). Esses mecanismos poderiam ser usados na ROTI, mas uma rede *overlay* possui uma diferença, em relação a uma rede convencional, que reduz a eficácia desses mecanismos. A diferença é que, enquanto em uma rede convencional um enlace é um canal ponto a ponto ou multiponto cujo acesso é restrito aos nós que compartilham esse enlace, em uma rede *overlay* os enlaces virtuais correspondem a rotas formadas por nós e enlaces que, a princípio, podem ser acessados de qualquer ponto da Internet. A consequência prática disso é que os nós e enlaces Internet que suportam a rede *overlay* estão sujeitos a ataques de negação de serviço distribuída (DDoS — *Distributed Denial of Service*), algo mais difícil de ocorrer em uma rede convencional. Essa suscetibilidade a DDoS reduz a utilidade de mecanismos de garantia de justiça no *overlay*, pois eles de pouco adiantam quando um nó ou enlace da rede subjacente usado pela rede *overlay* é vítima de um ataque de DDoS.

A literatura registra propostas que usam redes *overlay* para proteger contra ataques de negação de serviço (distribuída ou não) [Andersen, 2003; Keromytis et al., 2004]. Entretanto, o que essas propostas sugerem é usar uma rede *overlay* como passagem obrigatória do tráfego destinado a um serviço centralizado que se deseja proteger; o objetivo é garantir a disponibilidade do serviço centralizado, e não a disponibilidade da rede *overlay*.

Por conta de toda essa conjuntura, decidiu-se não implementar na ROTI nenhum mecanismo específico de proteção contra ataques de negação de serviço baseados em sobrecarga por excesso de tráfego. Uma ressalva importante é que, neste contexto, ataques de sobrecarga da rede subjacente são mais prováveis e perigosos do que ataques de sobrecarga no *overlay*; por conseguinte, a existência de mecanismos de proteção contra ataques dessa natureza na rede física aumenta a resistência de *overlays* construídos sobre ela.

6.1.4 Considerações sobre a Análise

A tabela 6.1 resume as contramedidas adotadas na ROTI para combater as ameaças de segurança a *overlays* de roteamento identificadas na seção 2.3. Essa tabela mostra os principais mecanismos usados para combater cada ameaça, que não necessariamente são todos os que têm alguma parcela de responsabilidade pelo sucesso da tarefa. Por exemplo, os HMACs e assinaturas digitais garantem a integridade dos números de seqüência usados para combater ataques de *replay*.

Ameaça	Contramedidas	
	Plano de Controle	Plano de Dados
Confidencialidade	não tratada	<i>onion encryption</i> (esq. pesado)
Integridade		
<i>modificação não autorizada</i>	assinaturas digitais	HMACs
<i>falsificação de mensagens</i>	assinaturas digitais	HMACs
<i>replay de mensagens</i>	números de seqüência	números de seqüência
Disponibilidade		
<i>descarte de pacotes</i>	retransmissões periódicas (roteamento proativo)	retransmissões, mudança de rotas, invalidação de rotas/enlaces faltosos
<i>desvio de pacotes</i>	verificação da rota para ROUTE-REPLYS e CACHE-REPLYS	verificação da rota, retransmissões, mudança de rotas, invalidação de rotas/enlaces faltosos
<i>sobrecarga</i>	limitação da taxa de emissão de mensagens de roteamento	não tratada

Tabela 6.1: Resumo das contramedidas adotadas na ROTI

Dois aspectos citados na seção 2.3 e que não foram mencionados na análise são a maior propensão a vulnerabilidades exibida pelos nós *overlay* em comparação a roteadores convencionais e a incidência de erros de configuração. O primeiro ponto foi omitido por se tratar de um aspecto de implementação, onde valem as recomendações tradicionais para garantir a segurança de *hosts* Internet [NBSO, 2003]. Com relação aos erros de configuração, eles podem se manifestar como faltas bizantinas, caracterizando um comportamento inesperado e fora da especificação por parte de elementos de rede. Como tal, eles são naturalmente tratados pelos mecanismos de tolerância a intrusões da ROTI, e na medida em que apresentarem comportamento bizantino esses elementos tendem a ser evitados pelos nós corretos da rede.

Outra ressalva importante é que essa análise é qualitativa, considerando apenas os aspectos conceituais da arquitetura ROTI. Na prática, erros de implementação ou configuração podem comprometer a segurança oferecida por um dos mecanismos adotados, minando assim sua eficácia. Nesse sentido, a **diversidade** [Deswarte et al., 1998; Hiltunen et al., 2003; Obelheiro et al., 2005], que consiste em utilizar diferentes implementações de uma mesma funcionalidade, pode constituir uma ferramenta poderosa para minimizar o impacto de tais erros.

6.2 Análise de Confiabilidade

A seção 6.1 examinou de que forma e em que medida a ROTI satisfaz os seus requisitos de segurança. Esta seção faz uma análise da confiabilidade oferecida pela rede. O objetivo inicial desta análise é investigar de que forma as topologias do *overlay* e da rede subjacente influenciam no número de faltas e intrusões que a rede é capaz de tolerar. Uma vez respondida essa questão, pode-se então determinar o grau de confiabilidade da rede, obtendo uma expressão para o número máximo de faltas e intrusões para o qual é possível garantir a confiabilidade das transmissões no pior caso.

A análise desta seção é fortemente baseada em teoria de grafos. O texto inclui apenas os principais resultados dessa teoria que são usados na análise; uma revisão mais completa da terminologia e notação adotadas pode ser encontrada no apêndice A.

6.2.1 Modelo ROTI

A rede física é modelada por um grafo $F = (V_F, E_F)$, onde V_F é o conjunto de vértices que representam os nós da rede física e E_F é o conjunto de arestas que representam os enlaces físicos.

Uma rota entre dois nós x e y na rede física é um caminho x - y em F . Define-se $R(x, y)$ como o conjunto de rotas entre x e y . Se não há rota entre x e y , $R(x, y) = \emptyset$. O conjunto de rotas possíveis na rede é definido por $R^* = \bigcup_{x, y \in V_F} R(x, y)$.

A rede *overlay* é modelada por um grafo $O = (V_O, E_O)$. Os nós do *overlay* são representados pelo conjunto de vértices $V_O \subseteq V_F$. Os enlaces virtuais, por sua vez, estão relacionados às rotas na rede física: um enlace virtual xy é implementado pelas rotas entre x e y na rede física que não passam por outros nós que pertençam ao *overlay*. Formalmente, isso é representado através da função $\psi: [V_O]^2 \rightarrow 2^{R^*}$, onde $\psi(x, y) = \{r \in R(x, y) \mid V(r) \cap V(O) = \{x, y\}\}$. Por exemplo, no caso da rede *overlay* mostrada na figura 6.1, a função ψ seria dada por:

$$\psi(a, b) = \{aeb, afb, afeb, aefb\}$$

$$\psi(a, c) = \{ac\}$$

$$\psi(b, c) = \{bdc\}$$

O conjunto de arestas E_O contém, portanto, todos os enlaces virtuais possíveis entre os nós do *overlay*. Formalmente, $E_O = \{xy \mid \psi(x, y) \neq \emptyset\}$.

6.2.2 Análise do Grau de Tolerância a Faltas

Uma rede pode ser considerada confiável se ela sempre possui um caminho formado por nós e enlaces corretos ligando dois nós quaisquer. Isso equivale a dizer que a rede deve permanecer conectada se forem removidos todos os nós e enlaces faltosos. Dado um grafo G , o mínimo número de nós que, se removidos, provocam a desconexão de G é dito a sua conectividade $\kappa(G)$; se $\kappa(G) = k$,

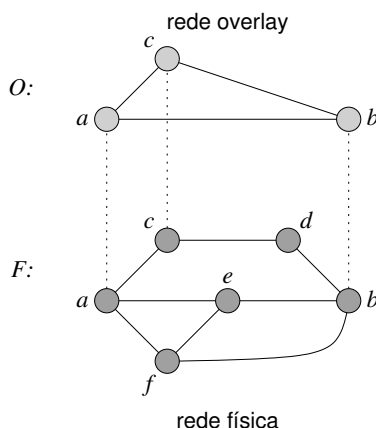


Figura 6.1: Exemplo de rede overlay sobreposta a uma rede física

G é dito k -conexo.¹ Logo, a confiabilidade de uma rede está ligada à conectividade do grafo que a representa, conforme exprime o seguinte lema [Harary, 1969]:

Lema 6.1. *Uma rede representada por grafo G k -conexo é capaz de tolerar até $f = k - 1$ faltas.*

Demonstração. Em uma rede representada por um grafo conexo, todos os nós conseguem se comunicar entre si (não há partição na rede). Se G é k -conexo, o número mínimo de vértices (nós da rede) que desconectam G é $k = \kappa(G)$. Portanto, após a remoção de $k - 1$ vértices, G continua sendo conexo. Por outro lado, seja $\ell = \lambda(G)$ o número mínimo de arestas (enlaces da rede) que desconectam G . Existe um teorema fundamental que diz que $\kappa(G) \leq \lambda(G)$ (vide teorema A.1 na página 97), e que portanto implica $\ell \geq k$. Logo, k é o limitante inferior para o número de faltas de nós ou enlaces que provocam uma partição na rede modelada por G . \square

Corolário. *Para determinar o limite máximo f de faltas toleradas por uma rede modelada por um grafo G é necessário analisar apenas a conectividade de vértices $\kappa(G)$, e não a conectividade de arestas $\lambda(G)$.*

Para determinar o grau de tolerância a faltas da ROTI, portanto, é suficiente determinar a conectividade do grafo que representa a rede. Todavia, como apresentado na seção 6.2.1, a rede é modelada através de dois grafos, um representando a rede física e outro representando a rede *overlay*. Esses dois grafos estão relacionados, uma vez que um mesmo enlace virtual pode ser implementado por diferentes rotas físicas. Tendo isso em vista, é necessário definir um modelo que possibilite a análise das propriedades desejadas. Esse modelo corresponde à projeção da rede *overlay* sobre a rede física que a implementa, e é representado por um grafo $\hat{O} = (V_{\hat{O}}, E_{\hat{O}})$. O conjunto de vértices $V_{\hat{O}}$ contém os nós *overlay* e os nós da rede física usados para implementar um ou mais enlaces virtuais, e o conjunto de arestas $E_{\hat{O}}$ contém os enlaces físicos usados para implementar um ou mais enlaces virtuais. Formalmente, tem-se:

¹A conectividade de arestas $\lambda(G)$ é definida de forma análoga a $\kappa(G)$, com arestas no lugar de nós; se $\lambda(G) = \ell$, G é dito ℓ -aresta-conexo.

$$\hat{O} : \begin{cases} V_{\hat{O}} = V(O) \cup \{v \in V(r) \mid \exists r \in R_{\hat{O}}^*\} \\ E_{\hat{O}} = \{l \in E(r) \mid \exists r \in R_{\hat{O}}^*\} \end{cases}$$

onde

$$R_{\hat{O}}^* = \bigcup_{x,y \in V(O)} R(x,y)$$

Seja o exemplo da figura 6.2(a), que representa uma rede física sobre a qual será construída uma rede *overlay* com quatro nós: c , j , o e s (representados com círculos brancos na figura). A figura 6.2(b) mostra o grafo \hat{O} que representa a projeção da rede *overlay* formada por esses nós sobre a rede física da figura 6.2(a). Observa-se que no grafo \hat{O} são eliminados da rede física todos os nós e enlaces que não são usados por nenhum enlace virtual, e que portanto não têm nenhuma influência sobre a rede *overlay*.

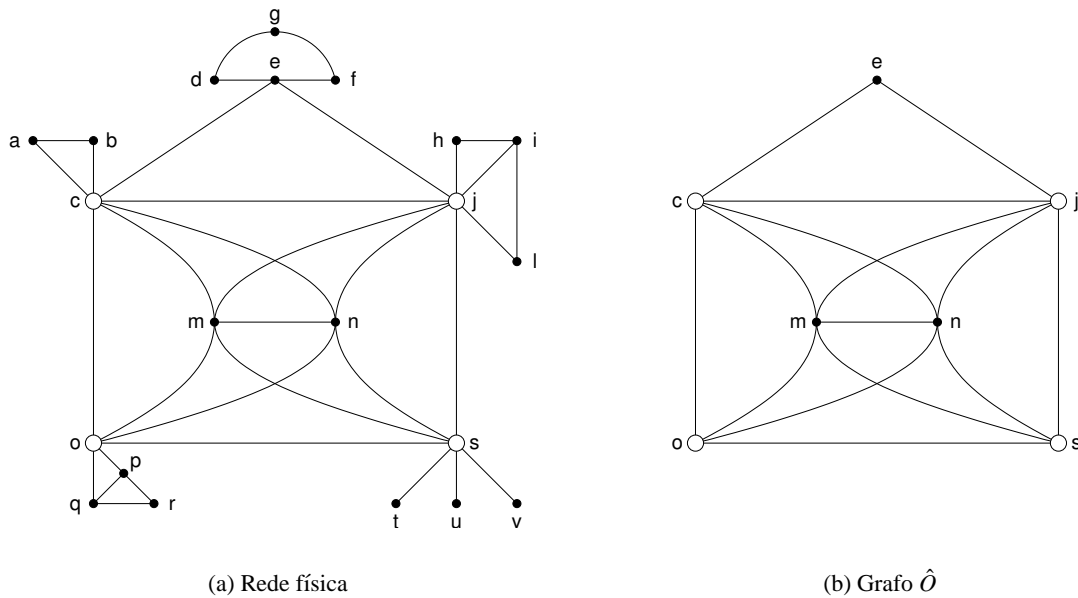


Figura 6.2: Exemplo de projeção de rede *overlay*

A análise da conectividade do grafo \hat{O} deve levar em consideração que o essencial é que os nós *overlay* permaneçam conectados entre si, ou seja, que a partir de um nó *overlay* seja possível atingir todos os outros. Na figura 6.2(b), por exemplo, a remoção das arestas ce e ej desconecta o grafo \hat{O} (o nó e fica isolado) sem desconectar nós *overlay*. Portanto, a propriedade de interesse para análise é a conectividade local² entre os pares de nós *overlay*, e não a conectividade do grafo \hat{O} como um todo. Como a conectividade local $\kappa(x,y)$ só é definida para vértices não adjacentes, utilizam-se os seguintes teoremas:

Teorema 6.2 (Teorema de Menger [Harary, 1969]). *O número mínimo de vértices que separam dois vértices não adjacentes s e t em um grafo G é o número máximo de caminhos disjuntos s - t em G .*

²A conectividade local de dois vértices não adjacentes a e b é o menor número de vértices cuja remoção separa a e b .

Teorema 6.3. *O número máximo de faltas tolerado em uma rede \hat{O} para que dois nós determinados x, y possam se comunicar é inferior ao número máximo de rotas disjuntas entre x e y .*

Demonstração. Caso x e y não sejam adjacentes, são necessários $k = \kappa_{\hat{O}}(x, y)$ vértices para separar x e y . Caso x e y sejam adjacentes é possível obter k removendo de \hat{O} a aresta xy , determinando $\kappa(x, y)$ no grafo resultante e somando um (correspondente à aresta removida) ao resultado; posto de outra forma, $k = \kappa_{\hat{O}-xy}(x, y) + 1$. De acordo com o teorema 6.2, k é igual ao número máximo de rotas disjuntas entre x e y . Como, pelo lema 6.1, o número de faltas f é inferior a k , completa-se a demonstração. \square

A informação sobre o mínimo número de faltas necessário para separar dois nós x, y na rede *overlay* pode ser capturada em um parâmetro $k_{\hat{O}}(x, y)$, definido por:

$$k_{\hat{O}}(x, y) = \begin{cases} \kappa_{\hat{O}}(x, y) & \text{se } x \text{ e } y \text{ não são adjacentes,} \\ 1 + \kappa_{\hat{O}-xy}(x, y) & \text{caso contrário.} \end{cases} \quad (6.1)$$

Teorema 6.4. *O número máximo de faltas tolerado em uma rede \hat{O} para que todos os seus nós possam se comunicar é inferior ao mínimo número de rotas disjuntas entre dois nós x, y quaisquer.*

Demonstração. Seja x', y' o par de vértices em \hat{O} com o menor número de rotas disjuntas entre si, e seja k esse número. Se $f \geq k$, é possível remover pelo menos o vértice adjacente a x' de cada uma das k rotas disjuntas, desconectando x' e y' . Portanto, como o número f de faltas toleradas significa que a rede não é desconectada, deve-se ter $f \leq k - 1$. \square

A condição do teorema 6.4 é que $f \leq k_{\hat{O}} - 1$, onde $k_{\hat{O}}$ é dado por:

$$k_{\hat{O}} = \min_{\substack{x, y \in V(\hat{O}) \\ x \neq y}} k_{\hat{O}}(x, y) \quad (6.2)$$

Além da condição do teorema 6.4, é importante observar que o número de faltas f está sujeito a uma condição adicional. Essa condição é que a comunicação em uma rede *overlay* só é possível quando existem pelo menos dois nós corretos nesse *overlay*; neste caso, tem-se $f \leq k_O$, onde k_O é dado por:

$$k_O = |O| - 2 \quad (6.3)$$

Combinando as condições $f \leq k_{\hat{O}} - 1$ e $f \leq k_O$, é possível obter uma expressão única para f :

$$f \leq \min(k_{\hat{O}} - 1, k_O) \quad (6.4)$$

onde $k_{\hat{O}}$ e k_O são dados, respectivamente, pelas expressões (6.2) e (6.3).

No exemplo da figura 6.2, a rede *overlay* possui quatro nós (c , j , s e o); portanto, $k_O = 2$. Para a análise da conectividade, obtém-se $k_{\hat{O}}$ usando as expressões (6.1) e (6.2):

$$\begin{aligned} k_{\hat{O}}(c, j) &= 5 & k_{\hat{O}}(c, o) &= 4 & k_{\hat{O}}(c, s) &= 4 \\ k_{\hat{O}}(j, o) &= 4 & k_{\hat{O}}(j, s) &= 4 & k_{\hat{O}}(o, s) &= 4 \end{aligned}$$

$$k_{\hat{O}} = \min_{\substack{x, y \in \{c, j, o, s\} \\ x \neq y}} k_{\hat{O}}(x, y) = 4$$

A partir daí, pela expressão (6.4), chega-se a

$$f \leq \min(2, 4 - 1)$$

$$\boxed{f \leq 2}$$

6.2.3 Considerações sobre a Análise

A análise baseada em teoria de grafos da rede *overlay* tolerante a faltas e intrusões nos permite determinar o seu grau de tolerância. É importante observar, porém, que o funcionamento correto da ROTI não depende de forma absoluta dos limites obtidos com essa análise. Com efeito, uma vez que esses limites não sejam respeitados, deixa de ser possível garantir de forma inequívoca a confiabilidade na comunicação, que passa a ser probabilística (no sentido em que passa a haver uma probabilidade significativa — e não mais uma garantia — de que dois nós quaisquer possam se comunicar através da rede). Por exemplo, para a rede da figura 6.2, determinou-se que $f \leq 2$; entretanto, se apenas os nós *overlay* (c , j , o e s) e o nó m — assim como os enlaces cm , jm , om e sm — forem corretos, a rede *overlay* permanece conectada, mesmo que os 16 nós e os 29 enlaces restantes sejam faltosos.

Outro ponto que é necessário considerar é que os limites determinados na análise são de caráter mais teórico do que prático, uma vez que é bastante difícil determinar o grafo \hat{O} caso os nós *overlay* estejam topologicamente distantes na rede física, devido à própria dificuldade de conhecer a topologia dessa rede física (ou seja, obter o grafo F). Uma hipótese alternativa de modelagem seria projetar o grafo O não diretamente sobre o grafo F , mas sobre um grafo representando as interconexões entre os diferentes ASs que compõem a camada de roteamento da Internet. Este modelo daria uma visão apenas aproximada do grau de tolerância a faltas e intrusões de uma rede *overlay*, mas seria mais fácil de ser obtido (por exemplo, a partir de dados sobre roteamento BGP disponibilizados pelo projeto RouteViews [RouteViews, 2006]) do que um grafo representando a rede física como um todo.

6.3 Resultados de Simulação

Para demonstrar a aplicabilidade da arquitetura proposta e avaliar seus custos e benefícios, foi desenvolvido um modelo de simulação da ROTI usando o Simmcast [Muhammad e Barcellos, 2002], um *framework* para simulação de redes baseado em Java. Os experimentos realizados têm por ob-

jetivo analisar o grau de confiabilidade efetivamente oferecido pela ROTI e mensurar o *overhead* do protocolo de roteamento. Esses experimentos são descritos a seguir.

6.3.1 Ambiente de Simulação

O Simmcast [Muhammad e Barcellos, 2002] é um *framework* de simulação por eventos discretos voltado para a modelagem e avaliação de redes e protocolos distribuídos. O Simmcast possui uma API simples e extensível, composta por um conjunto de classes Java que implementam blocos de construção para o desenvolvimento de modelos de simulação. Os modelos são facilmente elaborados a partir da combinação e da especialização dos blocos de construção fornecidos pelo Simmcast.

As principais classes fornecidas pelo Simmcast são:

- **Node:** representa um nó da rede, ao qual é associado um identificador numérico único. Cada Node possui uma ou mais *threads* de execução.
- **NodeThread:** uma *NodeThread* representa uma *thread* cooperativa, e corresponde à unidade básica de execução no Simmcast. A lógica de aplicação é implementada através de uma ou mais classes derivadas de *NodeThread*, e instâncias dessas classes derivadas são então associadas a um Node específico.
- **Link:** representa um enlace entre dois Nodes. Um enlace no Simmcast é caracterizado por sua largura de banda, latência e taxa de perdas. A taxa de perdas representa a probabilidade de que um pacote enviado por esse enlace seja perdido. A latência é dada por uma distribuição de probabilidade. O usuário pode definir sua própria distribuição, ou usar uma das distribuições predefinidas; atualmente estão disponíveis as distribuições normal, uniforme, exponencial, hiperexponencial e Erlang, além de uma distribuição simples que retorna apenas um valor constante.
- **Network:** representa a rede propriamente dita, formada por um conjunto de Nodes conectados através de Links. Um modelo contém apenas uma instância desta classe, que gerencia todo o fluxo de execução da simulação.
- **Packet:** representa um pacote, a unidade básica de comunicação entre nós na rede.

O Simmcast conta ainda com a classe *Group*, que serve para implementar grupos de *multicast*; esta funcionalidade não foi usada na simulação da ROTI.

Uma simulação com o Simmcast é construída em duas etapas. Na primeira etapa, desenvolve-se a lógica de aplicação, ou seja, o código Java que implementa o protocolo simulado; tipicamente, isso é feito derivando as classes *Node* e *NodeThread*, e combinando as classes derivadas com as outras já existentes. Na segunda etapa, monta-se um arquivo de descrição da simulação, que define a topologia da rede: quais são os nós, quais os enlaces e seus respectivos parâmetros. A descrição da simulação também comporta a invocação de métodos das instâncias de classes nela definidas (isso é usado, por exemplo, para configurar os elementos da rede). Esse arquivo descreve completamente a simulação a ser realizada, e é usado pelo Simmcast quando essa simulação é executada.

6.3.2 Descrição da Simulação

6.3.2.1 Estrutura da Simulação

A estrutura da simulação desenvolvida está mostrada na figura 6.3. Essa figura representa apenas as classes principais envolvidas com processamento de dados e eventos, omitindo algumas classes auxiliares e aquelas que basicamente encapsulam dados (como as classes que representam pacotes de dados e de roteamento). A base da simulação é a classe `ITONNode`, derivada de `Node`; cada nó *overlay* é uma instância de `ITONNode`. Um `ITONNode` possui um `ITONRouter`, que é a *interface* de serviço do subsistema de roteamento, e um `ITONForwarder`, que é a *interface* de serviço do subsistema de encaminhamento de pacotes. O `ITONRouter` é responsável por mediar o acesso ao *cache* de rotas (`RouteCache`) e controlar a emissão de `ROUTE-UPDATES` e `ROUTE-REQUESTS`; além do *cache* de rotas válidas, `RouteCache` também mantém informações sobre rotas e enlaces em quarentena ou permanentemente inválidos (a manipulação dessas informações fica a cargo de `ITONRouter`). O `ITONForwarder` implementa os métodos `send()` e `receive()` da API de usuário, além de interagir com `ConnectionState` e `ConnectionManager` para gerenciar reconhecimentos e *buffers* de transmissão e recepção.

Todos os pacotes recebidos por um nó são processados pela *thread* `ListenerThread`. Essa *thread* realiza uma tarefa de demultiplexação: na inicialização, `ITONNode` registra junto a `ListenerThread` uma associação entre determinadas *threads* (chamadas ***threads de processamento***) e os tipos específicos de pacotes que essas *threads* são capazes de processar. A `ListenerThread` classifica, então, os pacotes recebidos e os insere em uma fila de acordo com o seu tipo. As *threads* de processamento definidas na simulação são `DataThread`, `RouteUpdateThread`, `RouteRequestThread` e `RouteReplyThread`. A primeira *thread* processa pacotes de dados, enquanto as demais processam pacotes de roteamento (`RouteReplyThread` processa tanto `ROUTE-REPLYS` quanto `CACHE-REPLYS`). Outro tipo de *thread* empregado na simulação são as ***threads de transmissão***, que compreendem `TransmissionThread` e `ITONRouterThread`. A primeira transmite pacotes de dados, e a segunda transmite pacotes de roteamento disparados pelo `ITONRouter`. A última *thread* mostrada na figura é `SenderThread`, que é usada para simular um cliente que envia dados através da rede. Todas essas *threads* são derivadas de `NodeThread`. Algumas delas (especialmente as de transmissão) tiveram que ser criadas porque, no `Simicast`, apenas `NodeThread` e suas descendentes têm acesso a métodos de transmissão e recepção de pacotes. Essa separação de funções faz com que seja necessário gerenciar filas de pacotes para a comunicação entre as classes que contêm lógica de aplicação (como `ITONForwarder` e `ITONRouter`) e aquelas que têm acesso a rede; por outro lado, ela favorece a modularidade, reduzindo o acoplamento entre as classes e permitindo com que elas sejam facilmente substituídas.

Como na simulação não existe nenhum protocolo de transporte confiável como o TCP rodando sobre a ROTI, implementou-se um protocolo simples de janelas deslizantes com reconhecimentos seletivos para melhorar o desempenho da transmissão. A classe `ConnectionState` gerencia todo o estado da conexão com outro nó, como *buffers* de envio e recepção de dados e janelas de transmissão e recepção. Além disso, ela controla os reconhecimentos pendentes e recebidos; quando um

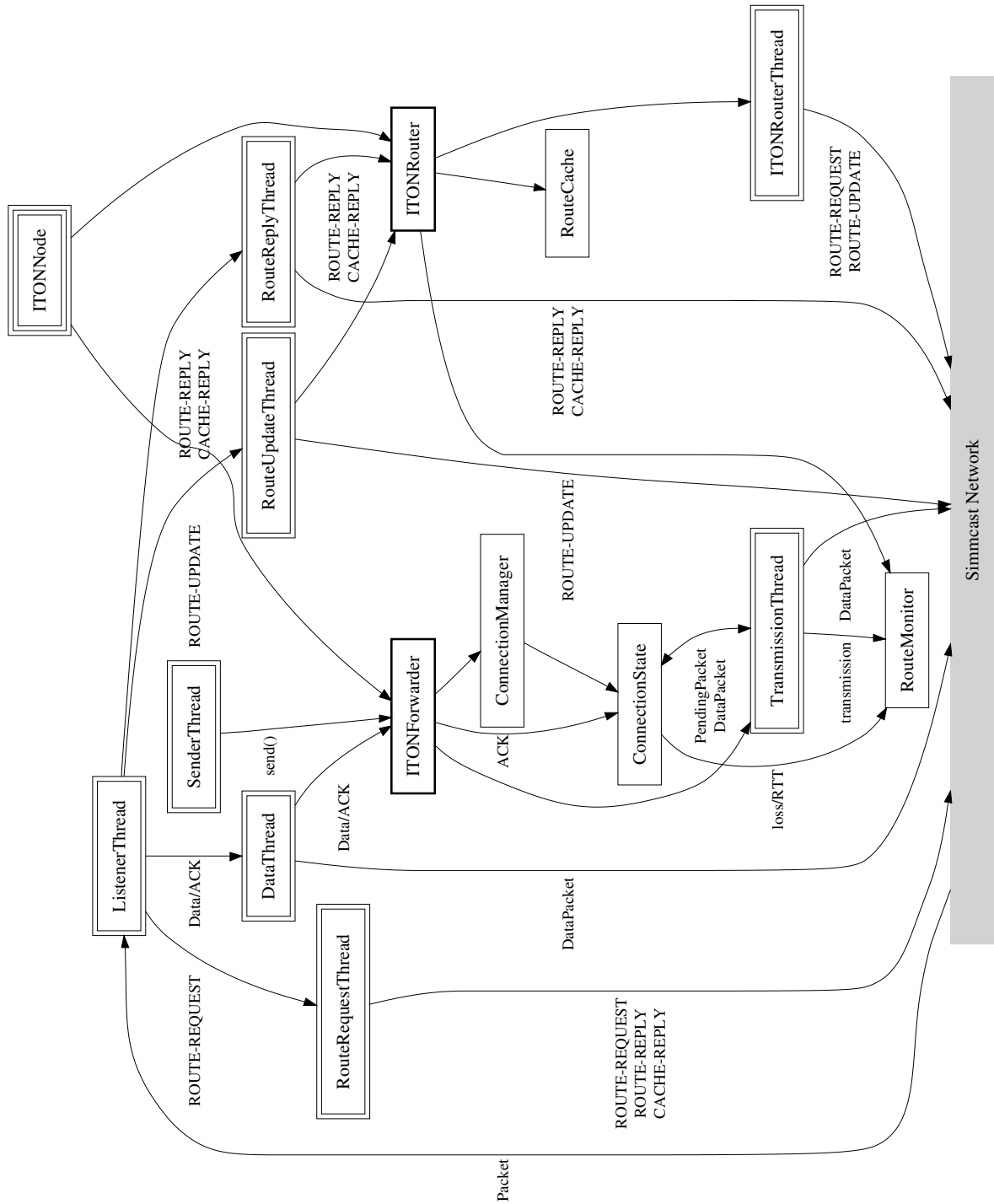


Figura 6.3: Estrutura da simulação da ROTI

ACK é recebido ou um *timeout* expira, ela passa a informação correspondente (perda ou RTT) para a classe `RouteMonitor`, que é responsável pela detecção de falhas. Quando uma falha é detectada, `RouteMonitor` interage com `ITONRouter` para invalidar (temporária ou permanentemente) a rota afetada. Um nó possui uma instância de `ConnectionState` para cada um dos demais nós com os quais se comunica; o gerenciamento dessas instâncias é responsabilidade da classe `ConnectionManager`.

Existem algumas classes de suporte que não são mostradas na figura 6.3. Um exemplo é a classe `StatsCollector`, que interage com diversas outras classes para coletar estatísticas sobre a simulação.

A simulação produz um registro dos eventos que ocorrem na rede. A cada evento é associado um nível de importância; os níveis atualmente definidos na classe `Log` são, em ordem crescente de importância, `DEBUG`, `INFO`, `WARNING`, `ERROR` e `REGULAR`. A classe `ITONNode` possui um método `setLogLevel` que permite definir o nível mínimo de importância dos eventos que serão incluídos no registro, o que facilita a filtragem dos eventos e permite que o tamanho do registro seja reduzido para experimentos de grande escala.

6.3.2.2 Protocolos Implementados

Foi implementado o protocolo de roteamento híbrido apresentado na seção 4.1. O intervalo de emissão de `ROUTE-UPDATES` é de 60 s, enquanto o intervalo mínimo entre atualizações consecutivas é $\Delta_{upd} = 10$ s. O número de atualizações perdidas consecutivas que fazem com que um vizinho seja excluído das atualizações é $U_{thr} = 3$. Quando existem diversas rotas com o mesmo comprimento e o mesmo nível de confiança, essas rotas são testadas em paralelo em grupos de $k = 4$ rotas, seguindo o esquema descrito na seção 4.1.3.

Foram implementados os esquemas leve e pesado de encaminhamento de pacotes descritos nas seções 4.2.1 e 4.2.2, com os respectivos mecanismos de detecção e recuperação de falhas. Conforme apresentado na seção 5.1, a detecção de falhas usa limiares de perdas $\pi_{thr} = 20\%$ e latência $RTT_{thr} = 350$ ms. O número máximo de vezes que uma rota ou enlace pode entrar em quarentena é $Q_{thr} = 10$. A simulação não contempla o protocolo de reconfiguração de topologia descrito na seção 5.3.

O suporte aos mecanismos criptográficos na simulação é limitado. Todos os pacotes possuem os campos reservados a HMACs e assinaturas, e funções que implementam algoritmos de teste são usados no lugar dos algoritmos reais para preencher esses campos (na transmissão) e verificar o seu conteúdo (na recepção). Essa simplificação foi feita para reduzir o tempo de desenvolvimento e execução das simulações, e não exerce nenhuma influência sobre os resultados obtidos.

6.3.2.3 Comportamento Malicioso dos Nós

Uma dificuldade de se obter resultados de simulação para redes sujeitas a falhas bizantinas ou intrusões diz respeito ao comportamento simulado dos elementos (nós ou enlaces) faltosos. Isso ocorre porque em tais redes justamente não se estabelece nenhum tipo de premissa sobre como esses elementos podem falhar. A estratégia adotada para definir como representar o comportamento malicioso

em nossas simulações leva em consideração o objetivo dos experimentos e de que forma um nó faltoso pode prejudicar a consecução desse objetivo. No nosso caso, os experimentos visam (i) avaliar a confiabilidade da rede e (ii) medir o *overhead* do protocolo de roteamento. Um nó faltoso só pode prejudicar a confiabilidade de transmissões das quais ele participe; portanto, ele deve executar corretamente o protocolo de roteamento, oferecendo-se como intermediário para outros nós. Se um nó faltoso não participar do protocolo de roteamento, ele não terá oportunidade de interferir no tráfego de nós corretos; além disso, ele reduzirá o número de mensagens de roteamento, melhorando assim o desempenho do protocolo para os nós corretos.

A segunda decisão envolve o que fazer com o tráfego que passa por um nó faltoso. Do ponto de vista de um adversário, a melhor estratégia é evitar ataques que sejam combatidos com mecanismos de prevenção — como a manipulação de pacotes, neutralizada pelo uso de HMACs —, jogando para os mecanismos de detecção de falhas e recuperação de faltas toda a responsabilidade de neutralizar as ações maliciosas. Essa estratégia é implementada na simulação através do descarte sistemático de pacotes por parte dos nós intermediários faltosos.

O último ponto referente ao comportamento dos nós faltosos diz respeito à sua atuação como nós de origem ou destino. Nos experimentos realizados, o comportamento de um nó faltoso como emissor não faz muita diferença, uma vez que não tem reflexos significativos na confiabilidade da rede ou no *overhead* de roteamento. Já o comportamento como receptor, por sua vez, parece decisivo à primeira vista: como os ACKs enviados pelo receptor são os dados usados pelo emissor para detecção de falhas, um receptor mal comportado pode fazer com que rotas corretas sejam consideradas falhas (caso não retorne os ACKs) ou mesmo que rotas falhas sejam consideradas corretas (fabricando ACKs — algo mais difícil de se conseguir na prática, pois exige conhecimento do número de seqüência correto, mas ainda assim possível). Entretanto, há que se considerar que o objetivo da ROTI é oferecer comunicação confiável para nós corretos, não necessariamente para nós faltosos. Portanto, optou-se por restringir o comportamento malicioso de nós faltosos à sua atuação como nós intermediários, fazendo com que eles sejam emissores e receptores bem comportados, transmitindo dados segundo o mesmo padrão de tráfego dos nós corretos e fornecendo os ACKs necessário aos nós de origem.

No modelo de simulação, o comportamento malicioso dos nós é encapsulado em uma classe chamada `ByzantineBehavior`. Essa classe possui um método `shouldDrop()`, que é invocado pelas *threads* de processamento sempre que elas vão repassar um pacote para um nó vizinho, seja ele de dados ou de roteamento; se o método retornar `true`, o pacote é descartado. O comportamento malicioso é configurável: um nó bizantino pode descartar pacotes de vítimas selecionadas e definir uma probabilidade de descarte de pacotes ou o instante em que o nó se torna faltoso. Nas simulações mostradas aqui, nós são faltosos durante todo o tempo de simulação, descartando pacotes de todos os outros nós com probabilidade igual a um. Alguns experimentos com outros valores para a probabilidade de descarte foram realizados, sendo que a única diferença observada foi no tempo para detectar as falhas, que não é medido nos resultados apresentados a seguir.

6.3.3 Parâmetros da Simulação

6.3.3.1 Topologia

Os experimentos foram realizados em redes *overlay* com 50 nós. Dez topologias foram geradas aleatoriamente usando o GT-ITM [Calvert et al., 1997]. O modelo adotado para as topologias é o chamado *random flat*, que gera topologias aleatórias com base em dois parâmetros, o tamanho da rede (50 nós) e a probabilidade de existência de enlaces. Este último determina a probabilidade de que dois nós quaisquer sejam vizinhos na topologia gerada. Para as simulações aqui apresentadas, adotou-se uma probabilidade de 50%; escolheu-se deliberadamente um valor baixo para que menos pares de nós possam se comunicar usando rotas diretas. Um menor número de vizinhos não apenas exercita mais as funcionalidades da ROTI como também significa que nós faltosos têm maiores possibilidades de perturbar o tráfego de nós corretos. Nas topologias geradas, o grau dos nós variou de 13 (observado em uma topologia) até 34 (observado em duas topologias).

6.3.3.2 Enlaces

Para aproximar o ambiente de simulação das condições reais observadas na Internet, é importante que os enlaces virtuais sejam modelados considerando perturbações, como variabilidade na latência e perdas causadas “naturalmente” (isto é, que não sejam provocadas intencionalmente por nós faltosos). Para satisfazer a esse requisito, os enlaces virtuais na simulação têm latência distribuída segundo uma curva normal com média $\mu = 20$ ms e desvio padrão $\sigma = 5$ ms; a figura 6.4 mostra o gráfico das funções densidade de probabilidade (6.4(a)) e distribuição acumulada (6.4(b)) da latência dos enlaces. Como no Simmcast a latência de cada pacote é determinada aleatoriamente e de maneira independente dos pacotes anteriores, essa variabilidade possibilita a ocorrência de reordenação de pacotes, outra condição que ocorre na prática na Internet e que pode influir no desempenho de protocolos de rede [Bennett et al., 1999; Bellardo e Savage, 2002].

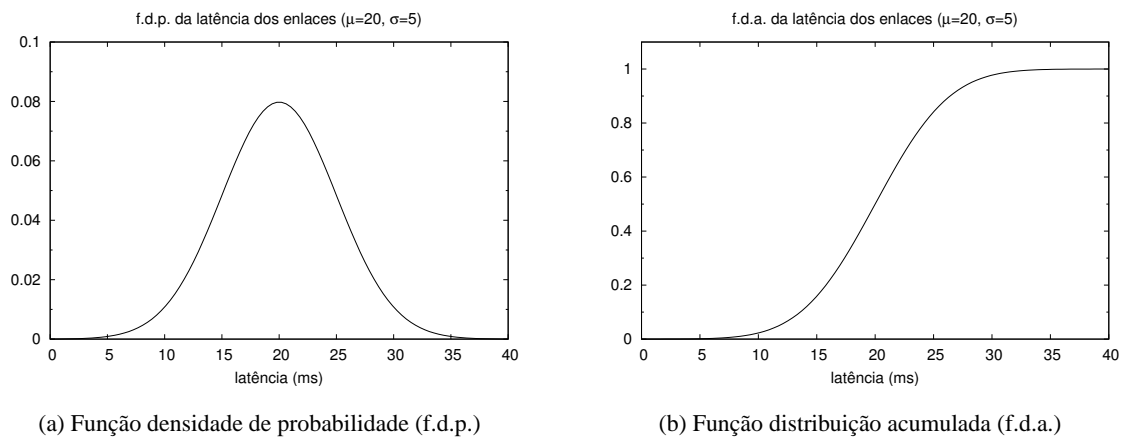


Figura 6.4: Distribuição da latência dos enlaces virtuais da simulação

Além da latência variável normalmente distribuída, os enlaces da simulação estão sujeitos a uma perda aleatória de 1% dos pacotes, e possuem largura de banda de 1 MByte/s.

6.3.3.3 Rodadas de Simulação

Cada experimento foi executado para 300 s de tempo simulado, e os resultados exibidos representam a média dos valores obtidos para as diferentes topologias. Cada simulação foi executada um número de vezes suficiente para que o erro padrão da média dos valores mensurados fosse inferior a 5% da média calculada. Esse erro padrão é calculado de acordo com a seguinte expressão [Spiegel, 1978]:

$$\sigma_{\bar{x}} = \frac{s}{\sqrt{n-1}}$$

onde n é o número de rodadas de simulação e s é o desvio padrão da média amostrada, dado por

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

sendo x_i os diferentes valores mensurados e \bar{x} a média desses valores.

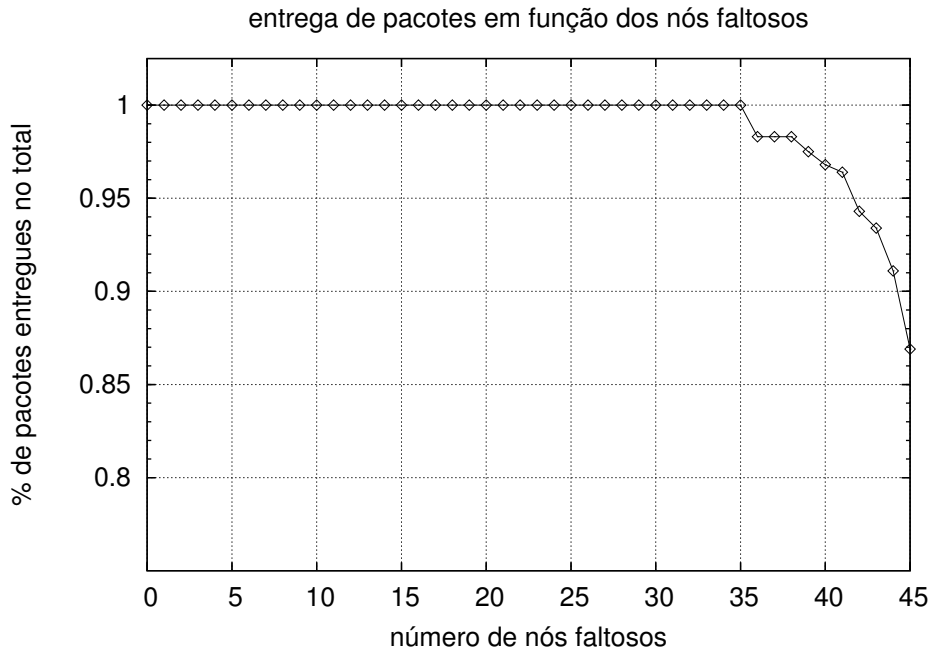
6.3.3.4 Padrão de Tráfego

Para cada topologia, 20 pares distintos de nós foram escolhidos ao acaso para formarem pares origem-destino de tráfego. Isso significa que 40 dos 50 nós da rede trocam tráfego de dados. Um nó de origem envia 100 pacotes por segundo para o nó de destino, cada pacote contendo um *payload* de 512 bytes, totalizando 51,2kbytes/s.

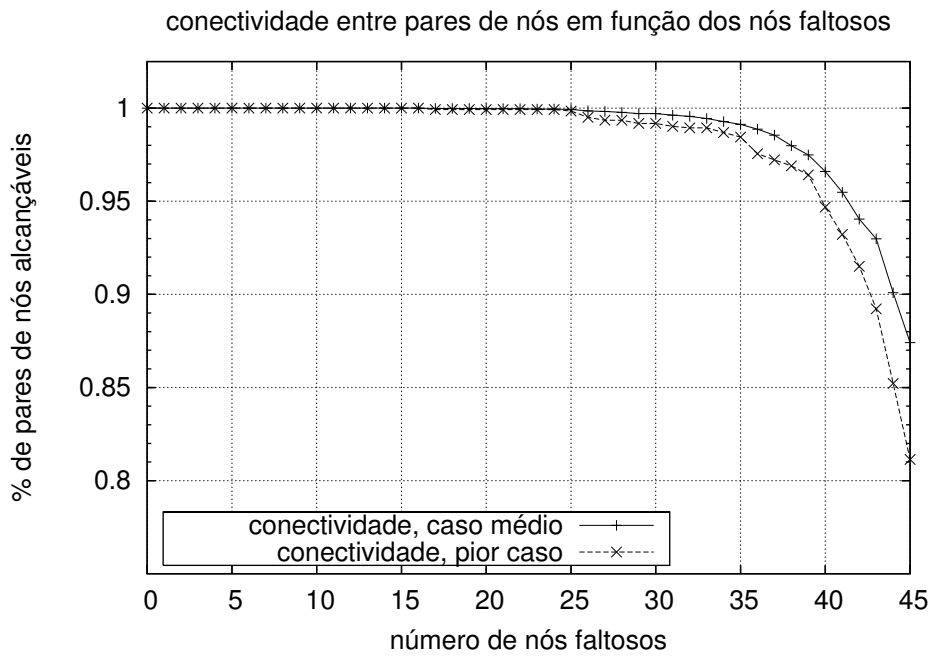
6.3.4 Resultados Obtidos

6.3.4.1 Análise da Confiabilidade

No primeiro experimento analisou-se a confiabilidade fornecida pela ROTI. Foram medidos o número de pacotes recebidos corretamente à medida em que se variava o número de nós faltosos, bem como a conectividade entre todos os pares de nós das topologias simuladas. A figura 6.5 mostra os resultados obtidos. Para o padrão simulado de tráfego (figura 6.5(a)), a taxa de entrega é bastante alta: mesmo com 35 nós faltosos (70% do total de nós da rede) todos os pacotes são entregues corretamente. Depois desse ponto a taxa de entrega de pacotes começa a cair, mas ainda é superior a 85% com 45 nós faltosos (o número máximo de faltas simulado). A alta confiabilidade exibida pela rede mesmo com um grande número de nós faltosos se deve primariamente ao alto nível de redundância disponível (o *overlay* é ricamente conectado), o que possibilita encontrar rotas corretas entre os nós.



(a) Taxa de entrega de pacotes com tráfego simulado



(b) Conectividade entre pares de nós

Figura 6.5: Resultados de simulação: confiabilidade da ROTI

Esses resultados com tráfego simulado medem a confiabilidade das comunicações entre 20 pares de nós que são escolhidos ao acaso. Para garantir que tais resultados são representativos da rede como um todo e não refletem apenas a situação específica dos pares de nós selecionados, foi analisada também a conectividade global entre os pares de nós da rede. A conectividade global foi obtida construindo-se um grafo representando a rede e medindo quantos pares de nós (dentro o total de pares da rede) possuem alcançabilidade **usando apenas rotas corretas**. O total de pares em uma rede com n nós é dado por $n(n-1)/2$; para uma rede com 50 nós, isso representa $(50 \times 49)/2 = 1225$ pares de nós. A figura 6.5(b) mostra duas curvas, uma com a média da conectividade entre todas as topologias e a outra (rotulada “pior caso”) com a conectividade mínima encontrada. A ROTI deve oferecer uma confiabilidade maior que ou igual à de pior caso e, idealmente, próxima ao caso médio (diferenças surgirão inevitavelmente como consequência dos padrões específicos de tráfego envolvidos). Comparando as figuras 6.5(a) e 6.5(b), verifica-se ser realmente este o caso.

Uma questão que surge quando se deseja medir a confiabilidade da rede é como lidar com nós maliciosos. Optou-se aqui por torná-los receptores bem comportados, pois do ponto de vista da rede o que um receptor faz com os pacotes a ele entregues (isto é, se eles são mesmo entregues à aplicação a que se destinam) é irrelevante. Se um nó malicioso tentasse atrapalhar um nó com o qual estivesse se comunicando se recusando a retornar os ACKs corretamente, o emissor acabaria por desistir da transmissão e considerar o receptor inalcançável (o que reduziria a taxa de entrega dos pacotes). Deve-se notar também que é impossível dar qualquer garantia sobre a confiabilidade fim a fim quando um dos nós comunicantes é bizantino. Mesmo assim, de qualquer forma os resultados de simulação aqui apresentados demonstram claramente que a ROTI é capaz de fornecer um bom nível de serviço aos nós corretos.

6.3.5 *Overhead* do Roteamento

O *overhead* imposto pelo protocolo de roteamento é uma importante medida na prática. A figura 6.6 mostra o *overhead* observado nas simulações, representado pela razão entre o tráfego de roteamento e o tráfego total. O *overhead* permanece bastante baixo (em torno de 2,5%) para até 36 nós faltosos, quando começa a subir para um pouco abaixo de 10% com 42 nós faltosos. A partir daí o *overhead* aumenta de maneira mais acentuada, chegando a 33% com 45 nós faltosos. Isso demonstra que o roteamento proativo fornece um número suficiente de rotas diversas mesmo para uma grande quantidade de nós faltosos; o roteamento reativo só passa a ser amplamente utilizado quando mais de 80% dos nós são maliciosos.

A alta eficácia demonstrada pelas rotas simples (fornecidas na ROTI pelo protocolo de roteamento proativo) não apenas está de acordo com os resultados de experiências anteriores na literatura [Savage et al., 1999; Andersen et al., 2001; Gummadi et al., 2004] mas também é corroborada por outras medidas obtidas nas simulações. Ao final de cada rodada de simulação, o *cache* de rotas em cada nó tinha uma média de 595 rotas para todos os 49 destinos possíveis (figura 6.7). Isso representa, na média, uma diversidade de rotas de 12,1 rotas por destino, como mostra a figura 6.8. Embora o tamanho médio do *cache* possa parecer alto, deve-se observar que ele representa menos de 25% do

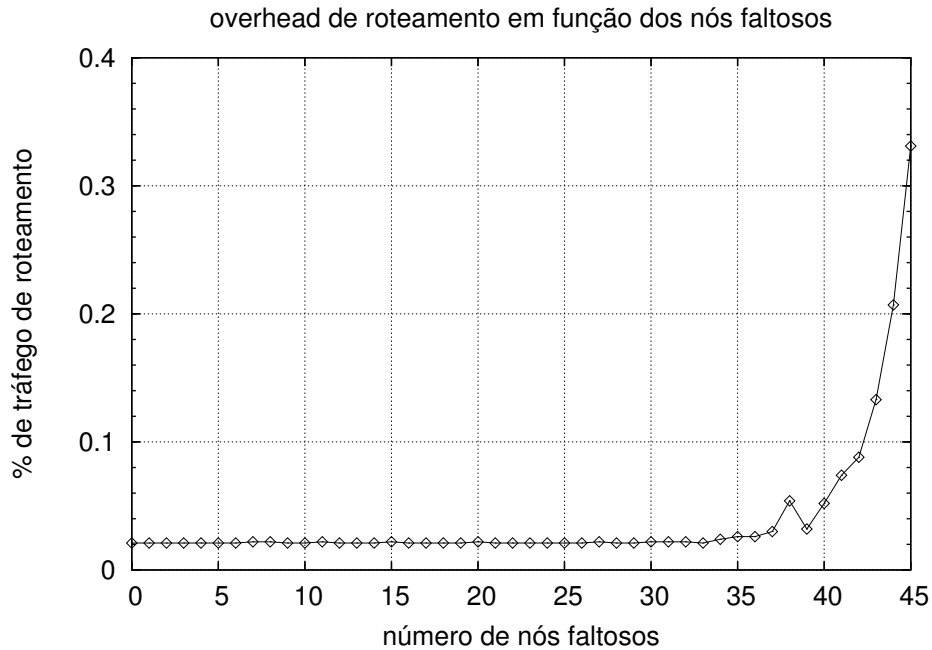


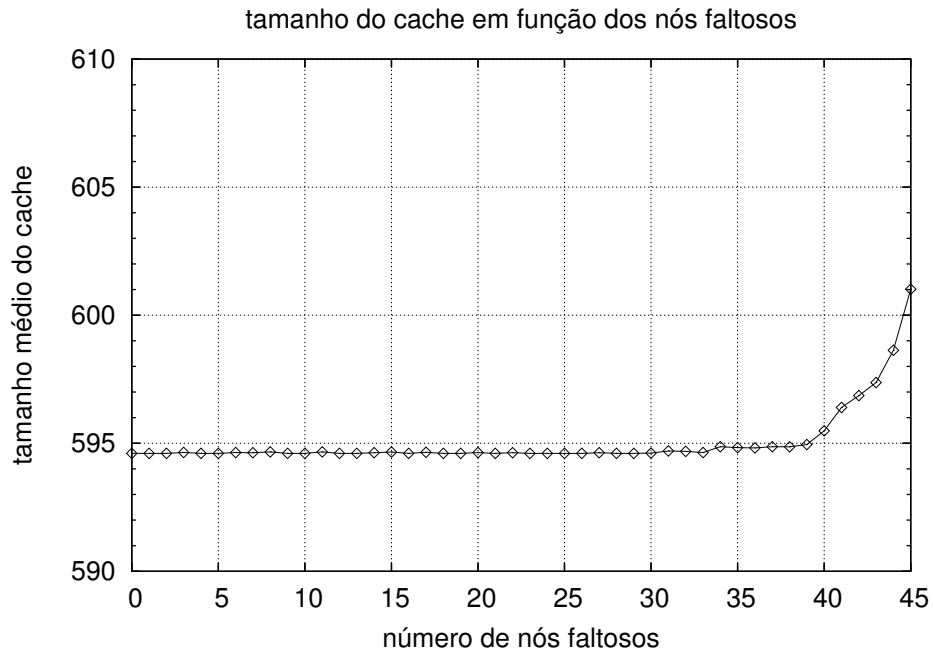
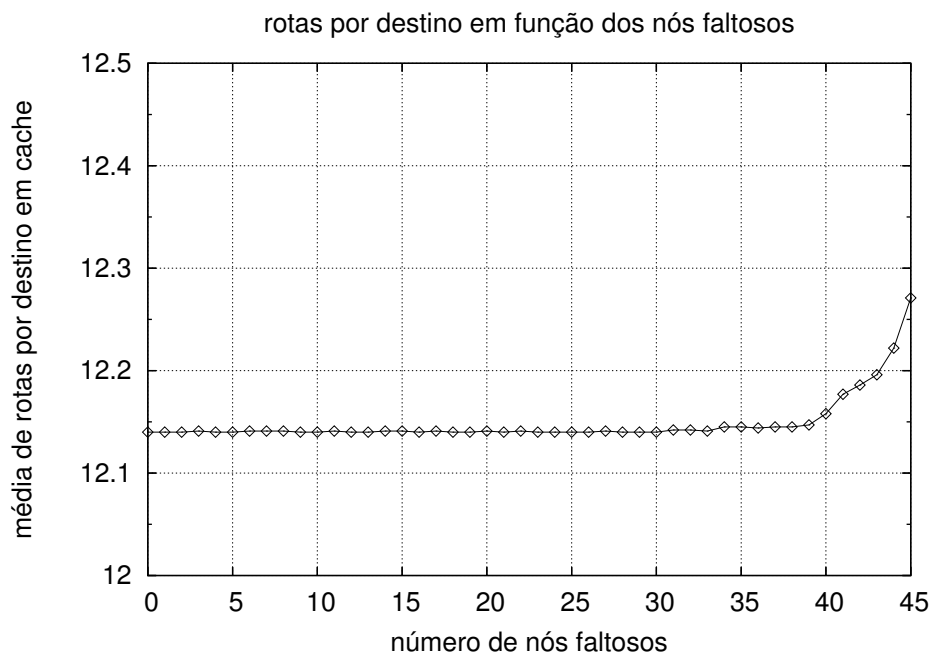
Figura 6.6: *Overhead* do protocolo de roteamento da ROTI

tamanho máximo possível para uma rede com 50 nós, que é de $49 \times 49 = 2401$ rotas (na verdade, essa média se aproxima do tamanho esperado para as topologias adotadas, onde cada nó é vizinho de cerca de metade dos demais). Esses números se mantêm praticamente inalterados quando o número de nós faltosos aumenta, o que significa que a maior parte do estado armazenado nos *caches* de rotas é facilmente (re)adquirido através do ROUTE-UPDATES quando um nó é (re)inicializado. Entretanto, se o tamanho do *cache* constituir um problema existem mecanismos que permitem reduzi-lo de forma inteligente, tais como:

- Armazenar um número máximo de rotas para cada destino;
- Descartar rotas antigas, não usadas ou menos confiáveis;
- Manter estatísticas sobre a qualidade das rotas fornecidas pelos vizinhos, descartando subsequentemente rotas que passam por nós que historicamente fornecem rotas ruins.

6.4 Trabalhos Relacionados

Os capítulos anteriores fizeram uma revisão de trabalhos relacionados aos mecanismos específicos da ROTI discutidos em cada capítulo (seções 4.3 e 5.4). Parte dos trabalhos discutidos nessas revisões limitam-se a alguns aspectos da segurança ou tolerância a faltas de redes, sem oferecer uma solução completa para o problema como a ROTI; outras referências usam mecanismos semelhantes ou mesmo idênticos aos usados na ROTI, mas em diferentes contextos. O objetivo desta seção é, então, fazer uma comparação global da ROTI com aqueles trabalhos que propõem um conjunto abrangente de

Figura 6.7: Tamanho médio do *cache* de rotasFigura 6.8: Média de rotas por destino no *cache*

soluções para oferecer disponibilidade e segurança em uma camada de rede, tentando se concentrar no todo e não em aspectos específicos. Quando se consideram apenas propostas completas, porém, o universo de trabalhos que podem ser referenciados fica bastante reduzido.

A primeira comparação que pode ser feita é com as experiências com redes *overlay* que motivaram o desenvolvimento da ROTI, que foram RON [Andersen et al., 2001] e SOSR [Gummadi et al., 2004]. Em um sentido amplo, o objetivo inicial do nosso trabalho era usar redes *overlay* para obter uma melhora na disponibilidade da rede (em relação à rede subjacente) comparável à dessas experiências, mas em um cenário muito mais hostil do que o que essas propostas consideram. Sob esse ponto de vista, os resultados alcançados são altamente satisfatórios: não apenas a ROTI atinge esse objetivo, como o faz sem degradar significativamente o desempenho da rede. Além disso, a ROTI melhora outro aspecto dessas experiências no que diz respeito à utilização de rotas com mais de um nó intermediário: enquanto RON e SOSR consideram somente rotas indiretas simples, a fase reativa do protocolo de roteamento da ROTI permite que os nós recorram a rotas mais longas quando essa necessidade se manifesta.

Passando para o campo dos trabalhos que têm como foco a segurança, a primeira e provavelmente mais influente proposta é a de Perlman [1988]. Essa proposta explorou de forma abrangente o problema de nós bizantinos em uma camada de rede e levantou uma série de soluções, ainda que nem todas tenham sido suficientemente exploradas. Comparando com a ROTI, pode-se dizer que as duas experiências compartilham os mesmos objetivos. Por outro lado, Perlman investigou soluções para uma camada de rede genérica, com base na experiência com a Internet de então, enquanto a ROTI busca os mesmos objetivos para redes *overlay* de menor escala.³ Os protocolos de roteamento podem ser considerados equivalentes, no sentido em que ambos garantem a autenticidade das informações de roteamento mas admitem a possibilidade de que essas informações sejam falsas ou incorretas, e possuem mecanismos que tratam essa possibilidade. Quanto ao encaminhamento de pacotes, na arquitetura leve da ROTI ele é bastante eficiente, com um pequeno *overhead* correspondente à geração de um HMAC no nó de origem e à verificação desse HMAC no nó de destino; Perlman propõe o uso de assinaturas digitais, que são bem menos eficientes do que HMACs. O esquema pesado tende a ser menos eficiente devido à cifragem em camadas, mas oferece maior proteção contra análise de tráfego. Em termos de detecção de falhas, a ROTI possui um mecanismo próprio, enquanto Perlman depende de notificações da camada superior, o que em termos práticos pode constituir um obstáculo à implementação da proposta. Do ponto de vista da negação de serviço, a ROTI possui mecanismos para limitar ataques que objetivam esgotar a capacidade de processamento de um nó, algo não previsto por Perlman. Por outro lado, Perlman propôs um esquema para satisfazer requisitos de justiça na rede, que garante que cada nó tenha acesso a uma porção dos recursos de rede disponíveis; um esquema desse tipo não é adotado na ROTI porque se considera que ele seria inócuo se o risco de ataques de negação de serviço contra a rede subjacente for significativo. Em linhas gerais, pode-se dizer que a ROTI tem um desempenho melhor do que a proposta de Perlman oferecendo um nível comparável de segurança e disponibilidade; essa vantagem em termos de desempenho é maior quando não existem

³Nesse ponto, é importante considerar que a Internet era muito menor na época em que Perlman desenvolveu o seu trabalho (meados da década de 80), e nem todas as soluções que ela propõe em sua tese teriam a escalabilidade necessária para serem aplicadas em uma camada de rede que substituisse a camada IP da Internet atual.

faltas na rede, sendo progressivamente reduzida à medida em que cresce o número de nós ou enlaces faltosos.

Embora a proposta de Avramopoulos et al. [2004] não inclua mecanismos para a segurança do processo de descoberta de rotas, é possível tecer uma comparação considerando o uso dos mecanismos de encaminhamento de pacotes, detecção de falhas e recuperação de faltas efetivamente propostos em conjunto com um protocolo de roteamento onde os anúncios de rotas são autenticados, como os discutidos em [Papadimitratos e Haas, 2002] (os autores pressupõem um protocolo desse tipo em [Avramopoulos et al., 2004]). Nessa proposta, a proteção do encaminhamento de pacotes pode ser considerada um meio termo entre o esquema leve e o esquema pesado da ROTI; por outro lado, as propriedades de segurança garantidas são as mesmas do esquema leve (autenticidade e integridade), a um custo consideravelmente maior, já que é usado um MAC para cada nó da rota (em vez de apenas um, como no esquema leve). O esquema pesado da ROTI, que também usa um (H)MAC por nó, conjuga essa medida com cifragem em camadas, o que é computacionalmente mais dispendioso mas não apenas garante confidencialidade como oferece uma proteção limitada contra análise de tráfego. A detecção de falhas e a recuperação de faltas propostas por Avramopoulos et al. incluem uma série de mecanismos para reduzir o tempo de detecção e ainda conter o tráfego de nós maliciosos. Embora a intenção seja boa, os mecanismos propostos deixam a desejar: o esquema de autenticação de ACKs e anúncios de faltas (FAs) é falho⁴, as propostas para compartilhamento de FAs e para contenção de tráfego malicioso tornam o processo de seleção de rotas mais complexo, e o fato de uma única perda provocar a remoção de um enlace da visão topológica de um nó (justificada pelo uso de um mecanismo de justiça similar ao de Perlman [1988]) torna o esquema excessivamente frágil. Além disso, a própria estratégia de compartilhamento de FAs e de remoção de enlaces faltosos a qualquer custo é pouco pragmática, podendo sacrificar desnecessariamente a disponibilidade da rede para manter um menor número de nós faltosos mesmo quando esses nós prejudicam apenas uma pequena parte do tráfego que passa por eles.

6.5 Conclusões do Capítulo

Este capítulo apresentou vários resultados que evidenciam a eficácia e a aplicabilidade da ROTI. A análise qualitativa de segurança demonstra que os mecanismos adotados, mesmo aqueles considerados simples conceitualmente, são capazes de oferecer proteção efetiva contra a maioria das ameaças de segurança que afetam camadas de rede.

A análise baseada em teoria de grafos revela de maneira precisa como a interação entre a rede *overlay* e a rede subjacente influencia na confiabilidade da primeira. Além disso, ela nos permite determinar o grau de tolerância a faltas e intrusões que a ROTI é capaz de garantir em uma situação de pior caso.

Finalmente, os resultados de simulação não apenas mostram que a ROTI pode ser realmente implementada na Internet atual como também comprovam que ela é capaz de oferecer um alto grau

⁴Fato reconhecido pelos autores, que propõem uma correção complexa e impraticável [Avramopoulos et al., 2004].

de confiabilidade mesmo em situações em que um grande número de nós da rede é comprometido, e que o seu protocolo de roteamento oferece uma boa diversidade de rotas aos nós da rede com baixo *overhead* mesmo com grandes números de nós comprometidos no *overlay*.

Capítulo 7

Conclusões

Esta tese apresentou uma arquitetura de rede *overlay* tolerante a intrusões como uma forma prática e eficiente de aumentar a disponibilidade das comunicações através da Internet diante de faltas acidentais e ameaças de segurança.

A arquitetura proposta usa um conjunto balanceado de mecanismos de segurança e tolerância a faltas bizantinas para atingir a qualidade de proteção pretendida. Uma preocupação que permeia o trabalho é evitar o uso indiscriminado de mecanismos custosos sem a devida apreciação da necessidade do seu uso e dos benefícios que eles aportam. Especial atenção é dispensada no sentido de minimizar o impacto sobre o desempenho da rede em situações livres de faltas.

Em termos mais concretos, o trabalho desenvolvido pode ser dividido em dois eixos complementares. O primeiro é o eixo de comunicação tolerante a intrusões, formado pelos subsistemas de roteamento e encaminhamento de pacotes. As funcionalidades implementadas por esses subsistemas, como troca de informações sobre rotas e proteção da segurança do tráfego, tornam possível a comunicação entre os nós *overlay*, mesmo em situações onde esses nós não conseguem se comunicar diretamente através da Internet.

O segundo eixo do trabalho é o de tratamento de elementos faltosos, seja a ocorrência de faltas acidentais simples ou a atividade maliciosa de elementos de rede. Neste eixo se enquadram os mecanismos de detecção de falhas e recuperação de faltas, que buscam identificar e remover da topologia elementos que perturbam excessivamente a comunicação de nós corretos, e o protocolo de reconfiguração de topologia, que visa restabelecer a redundância perdida na rede em decorrência de faltas e intrusões.

Juntos, esses dois eixos compõem uma arquitetura elegante e coesa, que teve sua robustez e eficiência demonstradas por meio de resultados tanto teóricos — através de uma análise de segurança e de uma análise de confiabilidade — quanto práticos, através de experimentos com um modelo de simulação.

7.1 Revisão dos Objetivos

Esta seção revisita os objetivos estabelecidos para a tese na seção 1.2, e relembra de que forma os diversos aspectos da arquitetura ROTI contribuem no sentido de satisfazer tais objetivos.

O objetivo geral da tese era o de propor um serviço de comunicação sobre a Internet que oferecesse alta disponibilidade mesmo na presença de faltas e intrusões, e que fosse suportado por mecanismos de baixo custo de modo a não degradar o desempenho de forma excessiva. A arquitetura da rede *overlay* tolerante a intrusões, introduzida no capítulo 3 e detalhada nos capítulos subsequentes, propõe um conjunto de subsistemas que possuem funcionalidades bem definidas e que juntos concorrem para que esse objetivo geral seja atingido.

O objetivo geral foi desdobrado em objetivos específicos, expressos como requisitos para a arquitetura proposta. A seguir, são enunciados esses objetivos específicos e são revisitados os mecanismos que são usados para atingir cada um deles:

1. Garantir a segurança e a veracidade das informações de roteamento.

O protocolo de roteamento híbrido apresentado na seção 4.1 garante a integridade e a autenticidade (e, conseqüentemente, a veracidade) das informações de roteamento fornecidas por nós corretos, tanto na fase proativa (seção 4.1.1) como na fase reativa (seção 4.1.2). A veracidade das informações de roteamento fornecidas por nós bizantinos é virtualmente impossível de ser garantida, até porque mesmo sem comportamento bizantino é possível que nem todos os nós tenham sempre a mesma visão da rede; informações falsas ou incorretas de roteamento são tratadas pelo mecanismo de seleção de rotas mostrado na seção 4.1.3. O *overhead* do protocolo de roteamento se mantém baixo mesmo com um grande número de nós maliciosos na rede, conforme demonstram os resultados de simulação apresentados na seção 6.3.5.

2. Possibilitar a escolha de rotas funcionais na presença de comportamento malicioso na rede.

O mecanismo de seleção de rotas apresentado na seção 4.1.3 associa um nível de confiança a cada rota em *cache*, priorizando as rotas que se mostram mais confiáveis no momento da escolha. Além disso, ele possui um esquema para testar rotas em paralelo quando várias rotas com mesmo comprimento e nível de confiança estão disponíveis no *cache*. O monitoramento do comportamento das rotas descrito na seção 5.1 permite identificar rotas em uso que não estão mais funcionais e devem ser substituídas, conforme mostrado na seção 5.2. Além disso, o protocolo de reconfiguração da topologia (seção 5.3) pode ser usado para aumentar o número de rotas disponíveis na rede, dando assim mais opções no momento da seleção.

Todos esses mecanismos (com exceção do protocolo de reconfiguração) permitem que cada nó aja individualmente em resposta a faltas e intrusões que afetem as suas comunicações com outros nós. Isso evita que um nó correto dependa da cooperação de outros nós para detectar e remover elementos faltosos da rede, e permite que nós continuem a se comunicar mesmo que possuam apenas um caminho correto que os conecte ao restante do *overlay* (conforme descrito na análise de confiabilidade da seção 6.2).

3. **Garantir que pacotes transmitidos por um nó de origem cheguem corretamente ao seu destino mesmo que ocorram faltas e intrusões na rede.**

O mecanismo de roteamento na origem (*source routing*) usado nos esquemas de encaminhamento de pacotes descritos na seção 4.2 permite que cada nó decida qual rota vai usar para transmitir seus pacotes, de maneira independente das decisões de roteamento dos demais nós. Esse mecanismo, conjugado aos esquemas de seleção de rotas e de tratamento de faltas e intrusões (conforme discutido no item 2 na página anterior), faz com que cada nó termine por encontrar os caminhos corretos que estão disponíveis para ele no *overlay*, garantindo assim que os pacotes transmitidos cheguem ao seu destino. Os dois esquemas de encaminhamento de pacotes propostos nas seções 4.2.1 (esquema leve) e 4.2.2 (esquema pesado), que são baseados nesses princípios, oferecem diferentes níveis de proteção para o tráfego de dados, permitindo controlar a quantidade de informações a que adversários tenham acesso e possam utilizar para tomar decisões inteligentes a respeito desse tráfego.

4. **Detectar a ocorrência de falhas e intrusões na rede.**

A detecção de falhas e intrusões na ROTI, apresentada na seção 5.1, se baseia no monitoramento e análise do comportamento fim a fim das rotas usadas para transmitir pacotes, mensurado através de seu índice de perdas e sua latência. A vinculação da detecção de falhas e intrusões à transmissão de dados assegura a precisão da detecção, por um lado, e evita tráfego de controle, por outro. A detecção de comportamento anômalo de rotas permite tratar diversas manifestações de falhas e intrusões de uma maneira unificada, baseada na forma com que essas manifestações prejudicam o tráfego de dados na rede, o que torna o esquema de detecção bastante simples e objetivo.

5. **Reagir à ocorrência de falhas e intrusões na rede.**

Como exposto no item anterior, falhas e intrusões são detectadas na medida em que provoquem as rotas a se comportarem fora das suas especificações (de taxa de perdas e de latência). Quando isso acontece, o elemento faltoso é localizado — em uma rota, no caso do esquema leve de encaminhamento de pacotes, ou em um enlace virtual, no caso do esquema pesado — e subsequentemente invalidado, deixando de ser usado por um período (quarentena) ou definitivamente; com isso, uma nova rota para o destino afetado é buscada no *cache*, conforme descrito nas seções 5.2.1 e 5.2.2. Esta estratégia é bastante pragmática: ela se preocupa em garantir que o serviço de comunicação funcione de forma aceitável para os seus usuários e evita desperdiçar esforço com ocorrências que não interferem negativamente no serviço fornecido.

O segundo mecanismo previsto para o tratamento de faltas e intrusões é a reconfiguração da topologia da rede, apresentado na seção 5.3, e que atua no sentido de recompor a redundância de rotas disponível na rede quando esta cai a um patamar abaixo do aceitável.

7.2 Contribuições e Resultados da Tese

O desenvolvimento deste trabalho de tese resultou em diversas contribuições, das quais podem ser destacadas:

- Introdução da primeira arquitetura de *overlay* de roteamento tolerante a intrusões. Experiências anteriores com *overlays* de roteamento limitavam-se a tolerar faltas simples, como queda de nós e de enlaces, sem ter nenhum tipo de preocupação com ameaças de segurança ou com o comportamento malicioso dos elementos de rede. Além disso, poucos trabalhos na literatura propõem arquiteturas completas de redes projetadas para lidar com incidentes de segurança como a apresentada nesta tese, geralmente se restringindo a abordar um aspecto específico da problemática de segurança.
- Definição de um protocolo de roteamento híbrido (proativo e reativo) para redes *overlay*. Este protocolo foi desenvolvido considerando as características peculiares a *overlays* de roteamento, apresentando um baixo *overhead* em situações com pequeno número de faltas na rede. Outro aspecto desta contribuição é que, diferente de outras experiências com redes *overlay*, que consideram apenas rotas com um único nó intermediário, esse protocolo híbrido possibilita que rotas mais longas sejam descobertas e utilizadas conforme a necessidade, aumentando assim a diversidade de rotas disponíveis para os nós *overlay*.
- Definição do primeiro protocolo de reconfiguração de topologia para redes *overlay* tolerante a faltas bizantinas. A reconfiguração de topologia aproveita a flexibilidade inerente aos *overlays* para recompor a redundância de caminhos em uma rede após a localização de elementos faltosos ou maliciosos, tentando manter um nível adequado de diversidade de rotas que permita aos nós se recuperarem rapidamente de futuros incidentes.

Algumas destas contribuições resultaram em publicações [Obelheiro e Fraga, 2004, 2005; Obelheiro et al., 2005; Obelheiro e Fraga, 2006].

7.3 Perspectivas Futuras

As contribuições e resultados apresentados neste trabalho de tese permitem antever diversas perspectivas para trabalhos futuros. Esses trabalhos podem ser divididos em dois grandes grupos: aplicação prática dos resultados e novos desenvolvimentos a partir das contribuições existentes.

No campo da aplicação dos resultados, a perspectiva mais imediata é a evolução do modelo de simulação para uma implementação real. A arquitetura ROTI proposta nesta tese se encontra amadurecida do ponto de vista teórico, e os resultados de simulação obtidos até o momento evidenciam que ela é perfeitamente aplicável à Internet atual. Uma implementação real forneceria dados experimentais que seriam inestimáveis para: (i) confirmar as hipóteses que ainda não foram demonstradas neste trabalho; (ii) levar ao aperfeiçoamento da rede, na medida em que ajudariam a identificar eventuais gargalos de desempenho e aspectos sub-especificados da proposta; (iii) permitir o ajuste dos parâmetros dos protocolos usados na rede, ou pelo menos a definição de um conjunto de valores *default* razoáveis, que servissem de ponto de partida para os usuários da rede. Essa implementação real poderia ser desenvolvida inicialmente usando a funcionalidade de *testbed* do Simmcast [Muhammad et al., 2004] e depois migrada para a Internet real.

Uma outra perspectiva de aplicação que poderia ser explorada é o uso da ROTI como suporte de comunicação para sistemas distribuídos tolerantes a intrusões. Uma funcionalidade que, se agregada à rede, poderia facilitar essa tarefa seria a comunicação multiponto (*multicast*). A experiência de uso da ROTI para a implementação de sistemas distribuídos poderia ainda levar à identificação de mecanismos da rede que poderiam ser tornados “plugáveis”, ou seja, que poderiam ser substituídos por mecanismos específicos de cada aplicação que seriam implementados segundo *interfaces* padronizadas. Uma primeira iniciativa neste sentido já se encontra em curso, com o desenvolvimento de um serviço de notificação de eventos tipo *publish/subscribe* para Web Services que usa a ROTI como suporte de comunicação.

Com relação a novos desenvolvimentos, uma primeira possibilidade seria o refinamento do protocolo de reconfiguração da topologia, buscando pesquisar mais a fundo os aspectos do protocolo ainda em aberto e propor novas soluções para eles. Nesse contexto, aspectos que merecem destaque são os critérios de seleção de nós de reserva, como garantir a correção desses nós no momento de sua ativação e — talvez o mais desafiador deles — como fazer com que a substituição de um nó restaure não apenas a conectividade perdida mas também as aplicações que estavam rodando no nó substituído. Este último aspecto talvez possa ser resolvido através de mecanismos de *checkpointing* [Elnozahy et al., 2002], que embora bastante conhecidos não são muito aplicados em sistemas sujeitos a falhas bizantinas (um exemplo de aplicação nesse contexto pode ser encontrado em [Castro e Liskov, 2002]).

Um outro aspecto que merece ser aprofundado é a defesa contra ataques de negação de serviço por sobrecarga de tráfego. Conforme discutido na seção 6.1.3, a ROTI possui mecanismos contra vários tipos de negação de serviço, como descarte de pacotes, desvio de pacotes e sobrecarga de processador por excesso de operações criptográficas. Entretanto, decidiu-se não implementar nenhum mecanismo de defesa contra ataques de sobrecarga de tráfego por se considerar que eles poderiam ser desferidos com similar facilidade contra a rede subjacente, e que então o custo de uma eventual defesa seria injustificado. Embora esta seja uma decisão válida do ponto de vista da engenharia da arquitetura ROTI, ela é insatisfatória do ponto de vista teórico. Uma solução mais adequada para este difícil problema talvez passe pela investigação de como o *overlay* e a rede subjacente podem agir em conjunto para resolvê-lo — por exemplo, identificar o suporte mínimo necessário na rede subjacente que justifique a implementação de mecanismos de proteção contra ataques dessa natureza na rede *overlay*.

Por fim, em um nível mais alto de abstração, podemos considerar que a arquitetura ROTI provê tolerância a intrusões se baseando em alguns princípios fundamentais, como a autonomia que cada nó possui para tomar decisões a partir de informações locais sobre faltas e o uso de um conjunto balanceado de mecanismos de prevenção e tolerância que minimiza o impacto sobre o desempenho em situações livres de faltas e faz com que esse impacto cresça à medida em que a situação vai se deteriorando. Uma perspectiva muito interessante de trabalho futuro seria, então, investigar a aplicação desse conjunto de princípios para prover tolerância a intrusões em outros contextos, considerando outros tipos de problemas e sistemas.

Apêndice A

Conceitos de Teoria dos Grafos

Como a terminologia da teoria de grafos não é universal, esta seção apresenta os conceitos adotados nesta tese. Maiores detalhes podem ser encontrados em [Harary, 1969; Bondy e Murty, 1976; Feofiloff et al., 2004; Diestel, 2005].

A.1 Definições Básicas

Um **grafo** é um par $G = (V, E)$ de conjuntos que satisfazem a relação $E \subseteq [V]^2$, isto é, os elementos de E são subconjuntos de dois elementos de V (G é um grafo não dirigido). Os elementos de V são os **vértices** (ou **nós**) do grafo e os elementos de E são as suas **arestas**. O conjunto de vértices de um grafo G é denotado $V(G)$, e o seu conjunto de arestas é denotado $E(G)$ (mesmo que os conjuntos respectivos não sejam V e E ; por exemplo, o conjunto de vértices do grafo $H = (X, Y)$ é dado por $V(H)$). O número de vértices de um grafo G (a sua **ordem**) é representado por $|G|$, e o seu número de arestas é denotado por $||G||$. Um grafo vazio ou que contém apenas um vértice é dito **trivial**.

Uma aresta $\{x, y\}$ é geralmente representada como xy (ou yx). Se $x \in X$ e $y \in Y$, então xy é uma aresta X – Y . O conjunto de todas as arestas X – Y em um conjunto E é representado por $E(X, Y)$. O conjunto de todas as arestas em E que incidem em um vértice v é representado por $E(v)$.

Dois vértices x, y de G são **adjacentes**, ou vizinhos, se xy é uma aresta de G ($xy \in E(G)$). Duas arestas $a \neq b$ são adjacentes se elas possuem um extremo (um vértice) em comum. Se todos os vértices de G são adjacentes entre si, diz-se que G é **completo**. O grafo completo de n vértices é representado por K^n .

A.2 Relações entre Grafos

Define-se $G \cup G' := (V \cup V', E \cup E')$ e $G \cap G' := (V \cap V', E \cap E')$. Se $G \cap G' = \emptyset$, diz-se que G e G' são **disjuntos**. Se $V' \subseteq V$ e $E' \subseteq E$, diz-se que G' é um **subgrafo** de G (e que G é um **supergrafo** de G'), denotado por $G' \subseteq G$. Menos formalmente, pode-se dizer que G contém G' .

Se $G' \subseteq G$ e G' contém todas as arestas $xy \in E$ com $x, y \in V'$, então G' é um **subgrafo induzido** de G , e pode-se dizer que V' **induz** G' em G , denotado por $G' := G[V']$. Portanto, para qualquer conjunto $U \subseteq V$ de vértices, $G[U]$ denota o grafo em U cujas arestas são exatamente as arestas de G que possuem ambos os extremos em U .

Se U é um conjunto qualquer de vértices (geralmente de G), $G - U$ representa $G[V \setminus U]$. Em outras palavras, $G - U$ é obtido de G removendo-se todos os vértices em $U \cap V$ e suas arestas incidentes. Em vez de $G - V(G')$, escreve-se apenas $G - G'$. Para um subconjunto F de $[V]^2$ define-se $G - F := (V, E \setminus F)$ e $G + F := (V, E \cup F)$. Se um dos conjuntos U ou F possui apenas um elemento e , representa-se apenas $G + e$ e $G - e$ em vez de $G + \{e\}$ e $G - \{e\}$.

A.3 Vizinhança e Grau

Se G é um grafo não vazio, o conjunto de vizinhos de um vértice v em G é representado por $N_G(v)$ (que pode ser abreviado $N(v)$ se o grafo for inequívoco). Em termos mais gerais, para $U \subseteq V$, os vizinhos em $V \setminus U$ de vértices em U são chamados de **vizinhos de U** , e o seu conjunto é denotado $N(U)$.

O **grau** $d(v)$ de um vértice v é o número $|E(v)|$ de arestas incidentes em v , que é igual ao número de vizinhos de v . O **grau mínimo** de G é dado por $\delta(G) := \min \{d(v) \mid v \in V\}$, e o seu **grau máximo** é dado por $\Delta(G) := \max \{d(v) \mid v \in V\}$. Se todos os vértices de G possuem o mesmo grau k , diz-se que G é **k -regular**, ou simplesmente regular. O **grau médio** de G é dado por

$$d(G) := \frac{1}{|V|} \sum_{v \in V} d(v).$$

É evidente que $\delta(G) \leq d(G) \leq \Delta(G)$.

A.4 Caminhos

Se v e v' são vértices de um grafo G e $k \geq 0$, diz-se que $C = v_0 v_1 \dots v_k$ é um **caminho** de comprimento k entre v e v' (chamado um **caminho $v-v'$**) se $v_0 = v$, v_i é adjacente a v_{i+1} para $0 \leq i \leq k$, $v_k = v'$ e $\forall i, j, v_i \neq v_j$ (ou seja, todos os vértices são distintos). Os vértices v e v' são chamados de **extremos**, e os vértices $v_1 \dots v_{k-1}$ são chamados de **vértices internos**. Os vértices que fazem parte de um caminho C são denotados por $V(C)$, e as arestas $e = v_i v_{i+1}$ desse caminho são denotadas por $E(C)$.

Dados dois conjuntos A, B de vértices, $C = v_0 \dots v_k$ é um **caminho $A-B$** se $V(C) \cap A = v_0$ e $V(C) \cap B = v_k$. Dois caminhos $a-b$ são **disjuntos** se eles não compartilham vértices internos, ou seja, $C_1 = av_1 \dots v_m b$ e $C_2 = av'_1 \dots v'_n b$ são disjuntos se $V(C_1) \cap V(C_2) = \{a, b\}$. Dois caminhos $a-b$ são **disjuntos nas arestas** se eles não possuem arestas em comum, ou seja, C_1 e C_2 são disjuntos nas arestas se $E(C_1) \cap E(C_2) = \emptyset$.

A.5 Conectividade

Um grafo G é **conexo** se para todo par x, y de vértices existe um caminho x - y em G . Se $U \subseteq V(G)$ e $G[U]$ é conexo, então U também é dito conexo (em G). Um conjunto X de vértices e arestas **separa** dois conjuntos de vértices A e B se todo caminho entre A e B contém pelo menos um elemento de X .

Um grafo G é **k -conexo** para $k \in \mathbb{N}$ se k é o tamanho do mínimo conjunto X que desconecta G , ou seja, $|G| > k$ e $G - X$ é conexo para todo $X \subseteq V(G)$ com $|X| < k$. A **conectividade** $\kappa(G)$ de um grafo G é o mínimo número de vértices cuja remoção resulta em um grafo desconexo ou trivial, ou seja, $\kappa(G)$ é o maior inteiro k tal que G é k -conexo. Um grafo completo K^n não pode ser desconectado pela remoção de qualquer número de vértices, mas a remoção de $n - 1$ vértices o torna um grafo trivial; logo, $\kappa(K^n) = n - 1$. A **conectividade local** $\kappa(a, b)$ de dois vértices não adjacentes a e b é o menor número de vértices cuja remoção separa a e b . G é **ℓ -aresta-conexo** se $|G| > 1$ e $G - F$ é conexo para todo conjunto $F \subseteq E(G)$ com menos de ℓ arestas. A **conectividade de arestas** $\lambda(G)$ de um grafo G é o maior inteiro ℓ tal que G é ℓ -aresta-conexo.

A conectividade, a conectividade de arestas e o grau mínimo de um grafo estão relacionados através de uma desigualdade, conforme mostra o teorema A.1 [Whitney, 1932].¹

Teorema A.1. *Para qualquer grafo G , $\kappa(G) \leq \lambda(G) \leq \delta(G)$.*

Menger mostrou que a conectividade de um grafo está relacionada ao número de caminhos disjuntos que ligam vértices distintos do grafo. A formulação original do teorema de Menger é a seguinte:

Teorema A.2 (Teorema de Menger). *O número mínimo de vértices que separam dois vértices não adjacentes s e t em um grafo G é o número máximo de caminhos disjuntos s - t em G .*

Uma versão global do teorema de Menger foi formulada por Whitney [1932]:

Teorema A.3 (Teorema de Menger, versão global).

- (i) *Um grafo é k -conexo se e somente se cada par de vértices for ligado por k ou mais caminhos disjuntos.*
- (ii) *Um grafo é ℓ -aresta-conexo se e somente se cada par de vértices for ligado por ℓ ou mais caminhos disjuntos nas arestas.*

¹Demonstrações para os teoremas usados fogem do escopo da tese e podem ser encontradas em [Harary, 1969; Bondy e Murty, 1976; Diestel, 2005].

Referências Bibliográficas

6bone. “6bone Home Page”. <http://www.6bone.net/>, 2006.

Amir, Y. e Danilov, C. “Reliable Communication in Overlay Networks”. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'2003)*, pp. 511–520, San Francisco, CA, USA, junho de 2003.

Andersen, D. G. “Mayday: Distributed Filtering for Internet Services”. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, Seattle, WA, USA, março de 2003.

Andersen, D. G., Balakrishnan, H., Kaashoek, F., e Morris, R. “Resilient Overlay Networks”. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pp. 131–145, Banff, AB, Canada, outubro de 2001.

Avramopoulos, I. C., Kobayashi, H., Wang, R., e Krishnamurthy, A. “Highly Secure and Efficient Routing”. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, março de 2004.

Awerbuch, B., Holmer, D., Nita-Rotaru, C., e Rubens, H. “An On-Demand Secure Routing Protocol Resilient to Byzantine Failures”. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, pp. 21–30, Atlanta, GA, USA, setembro de 2002.

Axelsson, S. “Intrusion-Detection Systems: A Taxonomy and Survey”. Technical Report 99–15, Department of Computer Engineering, Chalmers University of Technology, SE-412 96, Göteborg, Sweden, março de 2000. <http://www.cs.chalmers.se/~sax/pub/taxonomy.ps>.

Baran, P. “On Distributed Communications Networks”. *IEEE Transactions on Communications Systems*, 12(1):1–9, março de 1964.

Bellardo, J. e Savage, S. “Measuring Packet Reordering”. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pp. 97–105, Marseille, France, novembro de 2002.

Bellovin, S. M. “Security Problems in the TCP/IP Protocol Suite”. *Computer Communications Review*, 19(2):32–48, abril de 1989.

Bellovin, S. M., Ioannidis, J., e Bush, R. “Position Paper: Operational Requirements for Secured BGP”. In *DHS Secure Routing Workshop*, março de 2005. <http://www.cs.columbia.edu/~smb/papers/dhs-routing.pdf>.

- Bennett, J. C. R., Partridge, C., e Shectman, N. “Packet Reordering is Not Pathological Network Behavior”. *IEEE/ACM Transactions on Networking*, 7(6):789–798, dezembro de 1999.
- Bondy, J. A. e Murty, U. S. R. *Graph Theory with Applications*. North-Holland, New York, NY, USA, 1976.
- Bradley, K. A., Cheung, S., Puketza, N., Mukherjee, B., e Olsson, R. A. “Detecting Disruptive Routers: A Distributed Network Monitoring Approach”. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 115–124, Oakland, CA, USA, maio de 1998.
- Callon, R. “Use of OSI IS-IS for Routing in TCP/IP and Dual Environments”. RFC 1195, Internet Engineering Task Force, dezembro de 1990.
- Calvert, K., Doar, M., e Zegura, E. W. “Modeling Internet Topology”. *IEEE Communications Magazine*, 35(6):160–163, junho de 1997.
- Castro, M., Druschel, P., Ganesh, A., Rowstron, A., e Wallach, D. S. “Secure Routing for Structured Peer-to-Peer Overlay Networks”. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI’02)*, Boston, MA, USA, dezembro de 2002.
- Castro, M. e Liskov, B. “Practical Byzantine Fault Tolerance and Proactive Recovery”. *ACM Transactions on Computer Systems*, 20(4):398–461, novembro de 2002.
- Chandra, T. D. e Toueg, S. “Unreliable Failure Detectors for Reliable Distributed Systems”. *Journal of the ACM*, 43(2):225–267, março de 1996.
- Chaum, D. L. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms”. *Communications of the ACM*, 24(2):84–90, fevereiro de 1981.
- Cheung, S. e Levitt, K. N. “Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection”. In *Proceedings of the New Security Paradigms Workshop*, pp. 94–106, Cumbria, UK, setembro de 1997.
- Chu, Y., Rao, S. G., Seshan, S., e Zhang, H. “A Case for End System Multicast”. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, outubro de 2002.
- Cristian, F., Aghili, H., Strong, H. R., e Dolev, D. “Atomic Broadcast: From Simple Message Diffusion to Byzantine Agreement”. *Information and Computation*, 118(1):158–179, abril de 1995.
- Dahlin, M., Chandra, B. B. V., Gao, L., e Nayate, A. “End-to-end WAN Service Availability”. *IEEE/ACM Transactions on Networking*, 11(2):300–313, abril de 2003.
- Deswarte, Y., Kanoun, K., e Laprie, J.-C. “Diversity Against Accidental and Deliberate Faults”. In Ammann, P., Barnes, B. H., Jajodia, S., e Sibley, E. H., editors, *Computer Security, Dependability, and Assurance: From Needs to Solutions*, pp. 171–181, Williamsburg, VA, USA, novembro de 1998. IEEE Computer Press.
- Dierks, T. e Rescorla, E. “The Transport Layer Security (TLS) Protocol Version 1.1”. RFC 4346, Internet Engineering Task Force, abril de 2006.

- Diestel, R. *Graph Theory*, volume 173 de *Graduate Texts in Mathematics*. Springer-Verlag, New York, NY, USA, 3rd edition, 2005.
- Dijkstra, E. W. “A Note on Two Problems in Connexion with Graphs”. *Numerische Mathematik*, 1: 269–271, 1959.
- Dingledine, R., Mathewson, N., e Syverson, P. F. “Tor: The Second-Generation Onion Router”. In *Proceedings of the 13th USENIX Security Symposium*, pp. 303–320, San Diego, CA, USA, agosto de 2004.
- Elnozahy, E. N., Alvisi, L., Wang, Y.-M., e Johnson, D. B. “A Survey of Rollback-Recovery Protocols in Message-Passing Systems”. *ACM Computing Surveys*, 34(3):375–408, setembro de 2002.
- Eriksson, H. “MBone: the Multicast Backbone”. *Communications of the ACM*, 37(8):54–60, agosto de 1994.
- Feofiloff, P., Kohayakawa, Y., e Wakabayashi, Y. “Uma Introdução Sucinta à Teoria dos Grafos”. Livro-texto de minicurso apresentado na II Bienal da Sociedade Brasileira de Matemática, Salvador, BA, outubro de 2004. Disponível em <http://www.ime.usp.br/~pf/teoriadosgrafos/>.
- Ferguson, N. e Schneier, B. *Practical Cryptography*. John Wiley & Sons, Indiana, IN, USA, 2003.
- Freedman, M. J. e Morris, R. “Tarzan: a Peer-to-peer Anonymizing Network Layer”. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS-9)*, pp. 193–206, Washington, DC, USA, novembro de 2002.
- Freier, A. O., Karlton, P., e Kocher, P. C. “The SSL Protocol, Version 3.0”. Internet Draft, março de 1996. <http://wp.netscape.com/eng/ssl3/ssl-toc.html>.
- Gleeson, B., Lin, A., Heinanen, J., Armitage, G., e Malis, A. “A Framework for IP Based Virtual Private Networks”. RFC 2764, Internet Engineering Task Force, fevereiro de 2000.
- Goldschlag, D. M., Reed, M. G., e Syverson, P. F. “Hiding Routing Information”. In Anderson, R. J., editor, *Proceedings of the First International Workshop on Information Hiding*, LNCS 1174, pp. 137–150, Cambridge, UK, 1996. Springer-Verlag.
- Gummadi, K. P., Madhyastha, H. V., Gribble, S. D., Levy, H. M., e Wetherall, D. “Improving the Reliability of Internet Paths with One-hop Source Routing”. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, pp. 183–198, San Francisco, CA, USA, dezembro de 2004. USENIX.
- Hadzilacos, V. e Toueg, S. “Fault-Tolerant Broadcasts and Related Problems”. In Mullender, S. J., editor, *Distributed Systems*, pp. 97–145. ACM Press, New York, NY, USA, 2nd edition, 1993.
- Harary, F. *Graph Theory*. Addison-Wesley, Reading, MA, USA, 1969.
- Hedrick, C. L. “Routing Information Protocol”. RFC 1058, Internet Engineering Task Force, junho de 1988.

- Hiltunen, M. A., Schlichting, R. D., e Ugarte, C. A. “Building Survivable Services Using Redundancy and Adaptation”. *IEEE Transactions on Computers*, 52(2):181–194, fevereiro de 2003.
- Hughes, J. R., Aura, T., e Bishop, M. “Using Conservation of Flow as a Security Mechanism in Network Protocols”. In *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 132–141, Oakland, CA, USA, maio de 2000.
- Huston, G. “BGP Reports”. <http://bgp.potaroo.net/index-bgp.html>, 2006. Acessado em junho de 2006.
- ISO/IEC. “Intermediate System to Intermediate System Intra- Domain Routeing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)”. Standard 10589, 2002. 2nd Edition.
- Jacobson, V. “Traceroute”. <ftp://ftp.ee.lbl.gov/traceroute.tar.gz>, 1988.
- Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., e James W. O’Toole, J. “Overcast: Reliable Multicasting with an Overlay Network”. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation (OSDI’00)*, pp. 197–212, San Diego, CA, USA, outubro de 2000.
- Johnson, D. B., Maltz, D. A., e Broch, J. “DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks”. In Perkins, C. E., editor, *Ad Hoc Networking*, capítulo 5, pp. 139–172. Addison-Wesley, 2001.
- Kent, S., Lynn, C., e Seo, K. “Secure Border Gateway Protocol (S-BGP)”. *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, abril de 2000.
- Kent, S. e Seo, K. “Security Architecture for the Internet Protocol”. RFC 4301, Internet Engineering Task Force, dezembro de 2005.
- Keromytis, A. D., Misra, V., e Rubenstein, D. “SOS: An Architecture for Mitigating DDoS Attacks”. *IEEE Journal on Selected Areas in Communications*, 22(1):176–188, janeiro de 2004.
- Krawczyk, H., Bellare, M., e Canetti, R. “HMAC: Keyed-Hashing for Message Authentication”. RFC 2104, Internet Engineering Task Force, fevereiro de 1997.
- Krovetz, T., Black, J., Halevi, S., Hevia, A., Krawczyk, H., e Rogaway, P. “UMAC: Message Authentication Code Using Universal Hashing”. RFC 4418, Internet Engineering Task Force, março de 2006.
- Kuhn, D. R. “Sources of Failure in the Public Switched Telephone Network”. *IEEE Computer*, 30(4):31–36, abril de 1997.
- Labovitz, C., Ahuja, A., Bose, A., e Jahanian, F. “Delayed Internet Routing Convergence”. *IEEE/ACM Transactions on Networking*, 9(3):293–306, junho de 2001.
- Labovitz, C., Ahuja, A., e Jahanian, F. “Experimental Study of Internet Stability and Backbone Failures”. In *Proceedings of the International Symposium on Fault-Tolerant Computing (FTCS)*, pp. 278–285, Madison, WI, USA, junho de 1999.

- Lamport, L., Shostak, R., e Pease, M. “The Byzantine Generals Problem”. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, julho de 1982.
- Li, H., Rosenwald, G. W., Jung, J., e Liu, C.-C. “Strategic Power Infrastructure Defense”. *Proceedings of the IEEE*, 93(5):918–933, maio de 2005.
- Mahajan, R., Wetherall, D., e Anderson, T. “Understanding BGP Misconfiguration”. In *Proceedings of ACM SIGCOMM*, pp. 3–16, Pittsburgh, PA, USA, agosto de 2002.
- Maymoukov, P. e Mazières, D. “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric”. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, pp. 53–65, Cambridge, MA, USA, março de 2002.
- Mizrak, A. T., Cheng, Y.-C., Marzullo, K., e Savage, S. “Fatih: Detecting and Isolating Malicious Routers”. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN’2005)*, pp. 538–547, Yokohama, Japan, junho de 2005.
- Moy, J. “OSPF Version 2”. RFC 2328, Internet Engineering Task Force, abril de 1998.
- Muhammad, H. H. e Barcellos, M. P. “Simulating Group Communication Protocols Through an Object-Oriented Framework”. In *Proceedings of the 35th Annual Simulation Symposium (ANSS’02)*, pp. 143–150, San Diego, CA, USA, abril de 2002.
- Muhammad, H. M., Bedin, G. B., Facchini, G., e Barcellos, M. P. “Quebrando a Barreira entre Simulação e Experimentação Prática em Redes de Computadores”. In *Anais do 22º Simpósio Brasileiro de Redes de Computadores (SBRC’2004)*, volume I, pp. 87–100, Gramado, RS, maio de 2004.
- NBSO. “Práticas de Segurança para Administradores de Redes Internet”. Versão 1.3, NIC.br Security Office, maio de 2003. <http://www.cert.br/docs/seg-adm-redes/>.
- NIST. “Advanced Encryption Standard (AES)”. FIPS PUB 197, novembro de 2001.
- NIST. “Secure Hash Standard”. FIPS PUB 180-2, agosto de 2002.
- Obelheiro, R. R., Bessani, A. N., e Lung, L. C. “Analisando a Viabilidade da Implementação Prática de Sistemas Tolerantes a Intrusões”. In *V Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pp. 99–112, Florianópolis, SC, setembro de 2005.
- Obelheiro, R. R. e Fraga, J. S. “Uma Rede Overlay Tolerante a Intrusões”. In *V Workshop de Testes e Tolerância a Falhas (WTF’2004)*, pp. 91–102, Gramado, RS, maio de 2004.
- Obelheiro, R. R. e Fraga, J. S. “Uma Rede Overlay Tolerante a Intrusões: Arquitetura e Análise”. In *Anais do 23º Simpósio Brasileiro de Redes de Computadores (SBRC’2005)*, volume II, pp. 813–826, Fortaleza, CE, maio de 2005.
- Obelheiro, R. R. e Fraga, J. S. “A Lightweight Intrusion-Tolerant Overlay Network”. In *Proceedings of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pp. 496–503, Gyeongju, Korea, abril de 2006.

- Oppenheimer, D., Ganapathi, A., e Patterson, D. A. “Why Do Internet Services Fail, and What Can Be Done About It?”. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Seattle, WA, USA, março de 2003.
- Papadimitratos, P. e Haas, Z. J. “Securing the Internet Routing Infrastructure”. *IEEE Communications Magazine*, 40(10):60–68, outubro de 2002.
- Paxson, V. “End-to-End Routing Behavior in the Internet”. *IEEE/ACM Transactions on Networking*, 5(5):601–615, outubro de 1997.
- Perkins, C. E. e Royer, E. M. “Ad Hoc On-Demand Distance Vector Routing”. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, New Orleans, LA, USA, fevereiro de 1999.
- Perlman, R. J. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, janeiro de 1988. Available as MIT-LCS-TR-429.
- Peterson, L. L., Anderson, T., Culler, D. E., e Roscoe, T. “A Blueprint for Introducing Disruptive Technology into the Internet”. In *Proceedings of the 1st ACM Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, USA, outubro de 2002.
- Ratnasamy, S., Francis, P., Handley, M., Karp, R., e Schenker, S. “A Scalable Content-Addressable Network”. In *Proceedings of ACM SIGCOMM*, pp. 161–172, San Diego, CA, USA, agosto de 2001.
- Rekhter, Y., Li, T., e Hares, S. “A Border Gateway Protocol 4 (BGP-4)”. RFC 4271, Internet Engineering Task Force, janeiro de 2006.
- Roberts, S. W. “Control Chart Test Based on Geometric Moving Averages”. *Technometrics*, 1(3): 239–250, agosto de 1959.
- RouteViews. “University of Oregon Route Views Project”. <http://www.routeviews.org/>, 2006.
- Rowstron, A. e Druschel, P. “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems”. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Heidelberg, Germany, novembro de 2001.
- Savage, S., Collins, A., Hoffman, E., Snell, J., e Anderson, T. “The End-to-End Effects of Internet Path Selection”. In *Proceedings of ACM SIGCOMM*, pp. 289–299, Cambridge, MA, USA, agosto de 1999.
- Schneider, F. B., editor. *Trust in Cyberspace*. National Academy Press, Washington, DC, USA, 1999. <http://www.nap.edu/readingroom/books/trust/>.
- Schneier, B. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, New York, NY, USA, 2nd edition, 1996.
- Spiegel, M. R. *Probabilidade e Estatística*. Coleção Schaum. McGraw-Hill, São Paulo, SP, 1978.

- Stamp, J., Dillinger, J., Young, W., e DePoy, J. “Common Vulnerabilities in Critical Infrastructure Control Systems”. Technical Report SAND2003-1772C, Sandia National Laboratories, Albuquerque, NM, USA, maio de 2003.
- Stoica, I., Morris, R. T., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., e Balakrishnan, H. “Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications”. *IEEE/ACM Transactions on Networking*, 11(1):17–32, fevereiro de 2003.
- Subramanian, L., Roth, V., Stoica, I., Shenker, S., e Katz, R. H. “Listen and Whisper: Security Mechanisms for BGP”. In *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 127–140, San Francisco, CA, USA, março de 2004.
- Teitelbaum, B., Hares, S., Dunn, L., Narayan, V., Neilson, R., e Reichmeyer, F. “Internet2 QBone—Building a Testbed for Differentiated Services”. *IEEE Network*, 13(5):8–16, 1999.
- Teixeira, R., Marzullo, K., Savage, S., e Voelker, G. M. “Characterizing and Measuring Path Diversity of Internet Topologies (extended abstract)”. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurements and Modeling of Computer Systems*, pp. 304–305, San Diego, CA, USA, junho de 2003.
- Whitney, H. “Congruent Graphs and the Connectivity of Graphs”. *American Journal of Mathematics*, 54:150–168, 1932.
- Xue, Y. e Nahrstedt, K. “Providing Fault-Tolerant Ad hoc Routing Service in Adversarial Environments”. *Wireless Personal Communications*, 29:367–388, 2004.
- Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., e Kubiawicz, J. D. “Tapestry: A Resilient Global-scale Overlay for Service Deployment”. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, janeiro de 2004.