Uma Rede Overlay Tolerante a Intrusões: Arquitetura e Análise

Rafael R. Obelheiro* **e Joni da Silva Fraga†**

Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Caixa Postal 476 – 88040-900 – Florianópolis – SC – Brasil
Email: rro@das.ufsc.br, fraga@das.ufsc.br

Resumo. *A camada de roteamento da Internet é por vezes demasiadamente lenta na recuperação de faltas, e faz um uso sub-ótimo da redundância disponível na camada IP. Redes overlay podem contornar essas deficiências para implementar infra-estruturas de comunicação com maior disponibilidade e flexibilidade. A maior parte das experiências até o momento são capazes de tolerar apenas faltas simples (crash) da rede subjacente. Este artigo apresenta uma arquitetura de rede overlay que tem por objetivo oferecer comunicações com alta disponibilidade a despeito de faltas e intrusões na rede. Um mecanismo de adaptação permite que a rede reconfigure sua topologia em resposta a faltas e intrusões detectadas. Também é apresentado um modelo baseado em teoria de grafos que permite determinar o grau de tolerância a faltas de um dado overlay.*
Palavras chave*: redes overlay, tolerância a intrusões, tolerância a faltas bizantinas*

Abstract. *The Internet routing layer is at times too slow at recovering from faults and makes a sub-optimal use of redundancy available at the IP layer. Overlay networks can overcome these deficiencies to provide communication infrastructures with greater availability and flexibility. The majority of current experiences, however, are able to tolerate only simple (crash) failures of the underlying network. In this paper we present an overlay network architecture that aims at providing highly available communication in spite of faults and intrusions in the network. An adaptation mechanism enables the network to reconfigure its topology in response to detected faults and intrusions. We also present a graph-theoretic model that allows the degree of fault and intrusion tolerance of a given overlay to be determined.*
Keywords*: overlay networks, intrusion tolerance, Byzantine fault tolerance*

1. Introdução

A Internet tem sido usada para um número crescente de aplicações críticas, que se caracterizam por requisitos cada vez mais severos de tolerância a faltas e segurança. Estudos mostram, porém, que em muitos casos a comunicação fim a fim entre dois nós Internet pode ser interrompida por um longo período (o suficiente para causar a terminação de conexões TCP, por exemplo) devido a problemas na camada de roteamento [6, 11, 14, 15]. Isso é provocado por várias razões, que incluem a lentidão dos mecanismos de recuperação de faltas (necessária para não provocar grandes instabilidades nas tabelas globais de roteamento) e diferentes políticas de roteamento, que limitam o aproveitamento da redundância disponível na camada de rede. Independente das causas, o efeito percebido pelos usuários é de indisponibilidade do serviço de comunicação, a despeito da existência de caminhos alternativos ligando os nós afetados.

Experiências registradas na literatura demonstram que as redes *overlay* podem oferecer um serviço de comunicação fim a fim com maior disponibilidade e desempenho do que é usualmente atingido através da Internet [1, 3]. Entretanto, essas experiências consideram apenas a ocorrência de falhas acidentais (semântica de falhas de parada ou *crash*), sendo portanto

*Bolsista CAPES.
†Bolsista de Produtividade em Pesquisa do CNPq – Nível 2.

vulneráveis a nós maliciosos na rede.

O objetivo deste trabalho é investigar como construir redes *overlay* que ofereçam um serviço de comunicação com maior disponibilidade rejeitando a hipótese de confiança mútua entre os nós. Para isso, propõe-se uma arquitetura de rede *overlay* tolerante a faltas e intrusões que garante a comunicação fim a fim entre dois nós. Essa garantia é conseguida sempre que houver um caminho entre os pares comunicantes que contenha apenas nós e *links* corretos. Para aumentar a probabilidade de existência de um tal caminho, foram desenvolvidos protocolos que permitem a reconfiguração da topologia da rede *overlay* quando são detectadas falhas e intrusões. Além disso, uma análise baseada em teoria de grafos permite determinar o grau de confiabilidade da rede *overlay* proposta.

O restante deste artigo está organizado da seguinte forma: a seção 2 apresenta as redes *overlay* e analisa as principais experiências que inspiraram este trabalho. A seção 3 apresenta a arquitetura de rede *overlay* proposta, ao passo que a seção 4 detalha o protocolo de reconfiguração da topologia. A seção 5 faz uma análise do grau de tolerância a faltas e intrusões da rede. Na seção 6 são discutidos outros trabalhos que estão relacionados a este, e na seção 7 são apresentadas as conclusões e perspectivas futuras.

2. Redes Overlay

Uma rede *overlay* é uma rede lógica construída sobre uma rede física existente. Cada nó da rede *overlay* é também um nó da rede física. Uma conexão entre dois nós da rede *overlay* é chamada de *link* 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*; tais critérios normalmente dependem da aplicação a que a rede se destina. Cada rede *overlay* utiliza um esquema de endereçamento próprio, que pode ou não ser baseado no esquema de endereçamento da rede física. 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 (*forwarding*) 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.

As redes *overlay* são atualmente usadas para oferecer serviços seguros ou tolerantes a faltas. Na RON [3] a rede *overlay* é completamente conectada, e cada nó monitora o estado dos seus *links* virtuais, redirecionando pacotes através de nós intermediários em caso de falha de um *link*. O roteamento em nível de aplicação permite que a RON explore de forma inteligente a redundância que está disponível na Internet mas que é normalmente subutilizada. Por outro lado, a RON considera apenas faltas de *crash* de seus nós, além de não oferecer mecanismos para garantir a segurança das mensagens.

O Spines [2] propõe uma rede *overlay* com características de segurança e confiabilidade. Porém, os resultados publicados até o momento enfatizam um ganho no desempenho da comunicação usando ACKs *hop* a *hop* (em vez de fim a fim) para garantir a confiabilidade da comunicação [1]; os mecanismos de segurança e reconfiguração, embora mencionados em [2], não são definidos. Não obstante, essa experiência também mostra a capacidade das redes *overlay* de oferecer um serviço diferenciado sobre a Internet.

3. ROTI: Uma Rede Overlay Tolerante a Intrusões

3.1. Descrição da Arquitetura e Premissas Adotadas

A arquitetura ROTI é mostrada na figura 1. Nessa arquitetura, cada nó da rede *overlay* é um nó da rede física, sendo que cada nó dessa rede física pode abrigar apenas um nó *overlay*. Os nós do *overlay* dividem-se em nós ativos e nós de reserva; os nós ativos formam a topologia corrente da rede *overlay* em um dado momento, enquanto que os nós de reserva podem tornar-se ativos através de uma reconfiguração dessa topologia.

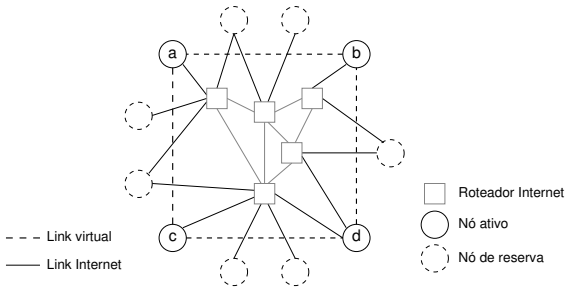


Figura 1: Arquitetura de rede *overlay* tolerante a intrusões

Em termos de semântica de falhas, considera-se que os nós, sejam eles do *overlay* ou da rede física subjacente, podem exibir comportamento arbitrário (semântica de falhas bizantinas com autenticação [12]), agindo sozinhos ou formando conluios. Um nó faltoso pode interceptar, modificar ou injetar pacotes, criar laços na rede, descartar pacotes de forma seletiva, rotear pacotes através de caminhos sub-ótimos ou fazer com que um caminho pareça mais curto ou mais longo do que realmente é.

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. Essas chaves públicas são usadas diretamente (para assinar mensagens, por exemplo) e também para estabelecer chaves simétricas compartilhadas (usadas, por exemplo, para cifragem de tráfego).

3.2. Encaminhamento de Pacotes

Na ROTI, o encaminhamento de pacotes da origem para o destino envolve autenticação, confidencialidade e confiabilidade. A autenticação garante a origem dos dados que estão sendo transmitidos, enquanto a confidencialidade garante que nenhum nó intermediário possa inspecionar o conteúdo dos pacotes transmitidos. A confiabilidade, por sua vez, garante que os pacotes sejam entregues mesmo na presença de faltas e intrusões na rede, desde que exista um caminho correto entre os nós comunicantes. Na discussão seguinte, considera-se que o nó de origem é capaz de obter uma rota através do *overlay* até o nó de destino; a seção seguinte detalha de que forma isso é feito.

A figura 2 mostra o formato de um pacote ROTI. Os campos SEQ e SRC contêm, respectivamente, o número de seqüência e o endereço de origem do pacote. O campo PAYLOAD contém os dados de aplicação que são enviados através da ROTI. Esse campo é cifrado com uma chave simétrica compartilhada entre o nó de origem e o nó de destino, o que garante a confidencialidade da comunicação.

A confiabilidade da comunicação é garantida pela detecção e recuperação de falhas na transmissão, que ocorre em dois níveis. O primeiro nível utiliza reconhecimentos positivos

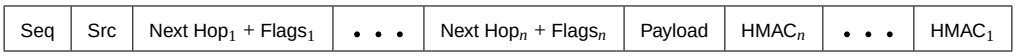


Figura 2: Formato de um pacote ROTI

(ACKs) para controlar a retransmissão de pacotes perdidos. Caso a rota em uso contenha um *link* que venha a ser declarado faltoso (o que pode ocorrer por detecção de uma intrusão ou por um índice excessivo de perdas), entra em ação o segundo mecanismo de recuperação, no qual a rota em uso é descartada e uma nova rota é solicitada ao protocolo de roteamento.

A transmissão dos pacotes utiliza *source routing*, com o nó de origem especificando a rota completa até o destino. Para cada nó que faz parte da rota, é adicionado um cabeçalho contendo dois campos, NEXT HOP e FLAGS. O primeiro contém o endereço do nó seguinte na rota (no caso do nó de destino, esse endereço é o seu próprio) e o segundo carrega algumas informações de controle (ACKs, por exemplo). Esse cabeçalho é cifrado com a chave simétrica compartilhada entre o nó de origem e o nó ao qual ele se destina. Um HMAC (*Keyed-Hash Message Authentication Code*) [10] cobrindo esse cabeçalho e o *payload*, e que usa a mesma chave simétrica, é adicionado ao final do pacote.

Os mecanismos criptográficos são aplicados de forma sucessiva, começando pelo nó de destino; assim, o que cada nó intermediário interpreta como sendo o seu *payload* é na verdade o conjunto cabeçalho+*payload*+HMAC do próximo nó na rota. Isso faz com que cada nó de uma rota só saiba quem é o próximo *hop*, mas não o destinatário final do pacote. Os HMACs permitem que qualquer alteração ou tentativa de falsificação de um pacote seja facilmente detectada, garantindo assim tanto a autenticação de origem quanto a integridade dos pacotes; além disso, eles são menos custosos do que assinaturas digitais para rotas com poucos *hops*.

Ao receber um pacote, um nó verifica primeiro se o seu HMAC está correto, descartando-o se não estiver. A seguir, ele retira o cabeçalho do pacote e o decifra, extraindo assim a informação de quem é o próximo *hop*. Se o endereço em NEXT HOP for o seu, ele repassa o conteúdo em PAYLOAD para a aplicação; caso contrário, ele envia o pacote (sem o seu cabeçalho e HMAC) para o nó apropriado.

3.3. Roteamento

A transmissão de pacotes na ROTI utiliza, preferencialmente, um *link* virtual direto entre o nó de origem e o nó de destino. Quando esse *link* não existe, é necessário obter uma rota através de uma seqüência de nós *overlay* que conecte uma ponta à outra da comunicação desejada. A ROTI utiliza um protocolo de roteamento sob demanda, no qual um processo de descoberta de rotas é executado sempre que um nó deseja obter uma nova rota, seja porque ele não possui nenhuma rota para um determinado destino, seja porque a rota atualmente em uso passa por um *link* que foi declarado faltoso. Esse tipo de protocolo foi escolhido visando a um melhor desempenho da rede quando não existem falhas. Como pacotes só são encaminhados através de nós intermediários quando o *link* virtual entre os nós de origem e destino não está disponível, a latência adicional imposta pelo processo de descoberta de rotas é mais vantajosa do que o *overhead* constante de um protocolo proativo, em que informações de roteamento são trocadas periodicamente entre os roteadores independente da utilização das rotas.

A estratégia de roteamento adotada na ROTI é baseada em uma proposta para roteamento seguro em redes *ad hoc* sem fio [4], adaptada para uso em redes *overlay*. Essa estratégia implementa descoberta de rotas, detecção de falhas e gerenciamento de métricas de roteamento.

O processo de descoberta de rotas funciona da seguinte maneira (algoritmo 1): um

nó *a* que deseja obter uma rota para um nó *b* monta uma requisição de rota assinada, que é disseminada na rede *overlay* através de *flooding*[1]. A requisição é propagada até *b*, onde a sua autenticidade é confirmada e uma resposta é enviada para *a*, também através de *flooding*. À medida em que a resposta vai se propagando através da rede, os nós intermediários vão calculando a rota entre *a* e *b*. A métrica usada para determinar a melhor rota é a contagem de *hops*: as rotas que passam por menos nós são escolhidas em detrimento das que passam por mais nós. Isso permite a minimização do custo associado aos mecanismos criptográficos usados no encaminhamento de pacotes. Todos os nós intermediários verificam a validade das assinaturas tanto de requisições como de respostas com o intuito de evitar que pacotes inválidos sejam disseminados através da rede. O *flooding* é usado nesse caso para garantir a máxima robustez, evitando que requisições ou respostas se percam ao atingirem um nó ou *link* faltoso.

Algoritmo 1 Algoritmo de descoberta de rotas

```
1:  seq ← 0; RC ← ∅; RQ ← ∅
2:  procedure REQUEST-ROUTE(in dst: NodeId)
3:      seq ← seq+1
4:      send SIGN(ROUTE-REQUEST, myid, dst, seq) to all neighbors        ▷ SIGN gera uma mensagem assinada
5:  end procedure
6:  upon receiving req = ROUTE-REQUEST(src, dst, seq) from x do
7:      if VERIFY-RREQ(req) and ⟨src, dst, seq⟩ is not in RQ then        ▷ VERIFY-RREQ checa a assinatura de req
8:          if dst=myid then
9:              reply ← SIGN(ROUTE-REPLY,src, dst, seq)
10:             send reply to all neighbors
11:         else
12:             for all neighbor n such that n is not x do
13:                 send req to n
14:             end for
15:         end if
16:         add ⟨src, dst, seq⟩ to RQ                                    ▷ RQ é um cache de requisições recebidas
17:     end if
18: end do
19: upon receiving rep = ROUTE-REPLY(src, dst, seq, R) from x do
20:     if VERIFY-RREPLY(rep) then                                       ▷ VERIFY-RREPLY checa as assinaturas de rep
21:         if src = myid and R doesn't include a removed link then
22:             if R is disjoint from all R′ such that ⟨dst, R′⟩ is in RC then   ▷ RC é o cache de rotas do nó
23:                 add ⟨dst, R⟩ to RC                                   ▷ adiciona rotas disjuntas ao cache
24:             else if exists R′ such that ⟨dst, R′⟩ is in RC and |R| < |R′| and R∩R′ ≠ {src, dst} then
25:                 replace ⟨dst, R′⟩ with ⟨dst, R⟩ in RC                ▷ esta rota é mais curta que uma anterior não disjunta
26:             end if
27:         else if myid is not in R then                                ▷ verifica se o nó já faz parte da rota para evitar loops
28:             newrep ← COSIGN(rep)                                     ▷ COSIGN adiciona o nó e sua assinatura à rota em rep
29:             for all neighbor n such that n is not in R do
30:                 send newrep to n
31:             end for
32:         end if
33:     end if
34: end do
```

Diferente da proposta original [4], na ROTI o nó de origem armazena em um *cache* não apenas a melhor rota para o destino mas todas as rotas disjuntas (isto é, que não compartilham nenhum nó intermediário ou *link* virtual) obtidas. Isso permite que, ao ser requisitada uma nova rota em função de faltas na rota atual, o subsistema de roteamento possa fornecer imediatamente uma rota alternativa disjunta enquanto o protocolo de descoberta de rotas é executado novamente, eliminando a latência inicial normalmente imposta pelo processo de descoberta.

Outro ponto importante é que o nó de origem descarta as rotas que incluam *links* faltosos, conforme será detalhado na seção 4. Com isso, cada nó pode agir individualmente, direcionando

[1]No *flooding*, cada nó retransmite um pacote recebido pela primeira vez a todos os seus vizinhos com exceção daquele que lhe enviou o pacote.

o seu tráfego para rotas que não passem por *links* que ele considera faltosos. Essa autonomia de ação permite que a rede tolere faltas bizantinas mesmo que exista uma maioria de nós faltosos e sem que seja necessário recorrer a protocolos de acordo [12] para decidir de forma distribuída quais nós são faltosos e devem ser excluídos da rede.

3.4. Detecção de Falhas e Localização de Faltas

O principal mecanismo de detecção de falhas na ROTI são os ACKs, que detectam falhas na transmissão de pacotes. Quando um ACK não é recebido antes do seu *timeout* expirar, o nó de origem registra uma perda para a rota em uso e retransmite o pacote. Quando o número de pacotes perdidos ultrapassa um dado limiar, considera-se que há uma falta no caminho, e inicia-se um processo para localizar o *link* faltoso. Esse processo consiste em retransmitir o pacote pela mesma rota, especificando no cabeçalho (campo FLAGS) a todos os nós intermediários que eles devem enviar um ACK para o nó de origem confirmando o recebimento do pacote. O *link* situado entre o último ACK recebido e o primeiro ACK perdido é declarado faltoso. Por exemplo, na figura 3, se o nó *a* envia um pacote para o nó *e* e recebe ACKs de *b* e *c* (mas não de *d* ou *e*), ele considera o *link cd* faltoso.

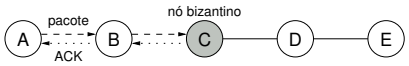


Figura 3: Localização de faltas com ACKs

É importante notar que o uso de ACKs oferece uma localização imprecisa de faltas. Seja o exemplo da figura 3, onde um nó *a* envia um pacote para um nó *e* e não recebe o ACK de *d* antes de expirar o respectivo *timer*. Isso pode acontecer por diversas razões: o pacote é perdido (caso ilustrado na figura), o ACK é perdido, *d* não envia o ACK, etc. Essas razões podem ser resumidas em uma ou mais de três causas: *c* é faltoso, *d* é faltoso e/ou o *link cd* é faltoso. A localização de faltas, nesse caso, pode reportar como suspeitos tanto o *link cd* como os nós *c* e *d*. É importante observar que ambos os nós têm que ser considerados suspeitos, uma vez que é impossível (usando apenas ACKs) determinar qual deles é efetivamente o faltoso caso apenas um deles o seja.

Em uma rede em que os nós e *links* podem apresentar comportamento bizantino, os ACKs podem ser manipulados pelos elementos faltosos. No exemplo da figura 3, o nó bizantino *c* pode deixar de transmitir ACKs do nó *e* em uma tentativa de incriminar o *link* correto *de*. A cifragem dos cabeçalhos e do conteúdo dos pacotes discutida na seção 3.2 tem justamente o propósito de limitar a capacidade de um nó intermediário analisar o tráfego passante, dificultando a identificação de pacotes com ACKs ou com requisições de ACKs e de pares origem-destino de nós comunicantes. Dessa forma, um nó bizantino não possui facilidade para interferir seletivamente na transmissão (o nó pode filtrar pacotes mesmo assim, mas isso acabaria por incriminar um de seus próprios *links*).

Os mecanismos criptográficos usados na ROTI também fornecem dados para a detecção de falhas. Voltando ao exemplo da figura 3, se *d* recebe um pacote com assinatura inválida de *c* ele registra uma falta para o *link cd*, uma vez que pacotes com assinatura inválida não devem ser propagados na rede. Certificados com assinaturas inválidas também representam uma falta do *link* por onde foram transmitidos.

4. Reconfiguração da Rede

Uma característica importante da ROTI é a sua capacidade de adaptação que consiste na reconfiguração dinâmica da topologia da rede. A reconfiguração envolve dois aspectos: exclusão de *links* virtuais e ativação de nós de reserva.

4.1. Exclusão de *Links* Virtuais

Basicamente, um *link* virtual é excluído após apresentar uma determinada taxa de faltas λ_{max}. A cada falta registrada pelos mecanismos de detecção e localização de faltas, a taxa de faltas λ_{ij} do *link* ij apontado como faltoso é analisada. Se $\lambda_{ij} \leq \lambda_{max}$, o *link* ij é posto em quarentena. Durante a quarentena, o *link* é ignorado, e qualquer rota que o contenha é descartada. Após um período Δ_q, ou após algum pacote válido chegar ao nó através de uma rota que passe pelo *link* em quarentena, este é reabilitado. A quarentena é usada para evitar que um *link* que esteja apenas temporariamente faltoso (devido a congestionamento na rede, por exemplo) seja excluído indevidamente por apresentar um grande número de faltas em um curto espaço de tempo.

Quando $\lambda_{ij} > \lambda_{max}$, o *link* ij é considerado permanentemente faltoso e deve ser excluído. Se ij é um *link* local (ou seja, faz parte do conjunto de *links* virtuais do nó que deseja excluí-lo), ele deixa de ser usado para as comunicações desse nó: nenhum pacote é enviado através de ij, pacotes eventualmente recebidos através de ij são descartados, e o *link* é ignorado no protocolo de roteamento. Se, por outro lado, ij não for local, ele simplesmente passa a ser ignorado no protocolo de roteamento (rotas que passam por ij são descartadas). Cabe ressaltar que a exclusão de um *link* virtual é uma decisão local: cada nó monitora os *links* da rede, e decide quando e quais excluir, de forma independente.

4.2. Ativação de Nós de Reserva

4.2.1. Descrição dos Protocolos

Quando um nó x detecta que um nó y ficou inacessível (isto é, todos os *links* virtuais para y foram removidos), ele propõe que um nó reserva y' seja ativado para substituir y. O novo nó é escolhido a partir de uma lista de nós de reserva. Essa lista é instalada manualmente nos primeiros nós da rede, e é assinada pelas chaves emissoras de certificados da PKI. Quando um nó reserva é ativado, os nós que estabelecem *links* virtuais com ele enviam-lhe as suas cópias da lista. Listas com assinatura inválida são descartadas, e correspondem a uma falta do *link* que as transmitiu.

A ativação de nós de reserva compreende dois protocolos, um de votação e outro de ativação propriamente dito. Para garantir propriedades de *safety* e *liveness*,[2] esses protocolos utilizam as seguintes premissas:

- *P1*. Se os nós e *links* faltosos forem removidos do *overlay*, a rede restante não é desconectada, ou possui no mínimo uma partição com pelo menos ϕ nós (esta segunda condição é mais fraca, mas não possibilita que as partições com menos de ϕ nós sejam reconectadas ao(s) componente(s) majoritário(s));
- *P2*. Os *links* da rede física têm uma probabilidade significativa de entregar mensagens dentro de um *timeout* δ.

A premissa P1 é necessária para garantir *safety* e *liveness*, enquanto que P2 é necessária apenas para garantir *liveness*. A premissa P2 é uma condição de sincronia fraca: ela não exige que a rede física seja síncrona o tempo todo, mas que existam períodos durante os quais a comunicação ocorra sem que os *timeouts* expirem. Essa premissa permite que *timeouts* sejam usados para detectar falhas de transmissão; um *link* que a viole será considerado faltoso, o que pode não ser

[2]Neste caso, o *safety* diz que a substituição de um nó não pode ser causada pela ação exclusiva de uma minoria de nós ou *links* faltosos, e o *liveness* diz que uma proposição de reconfiguração iniciada por um nó correto e que tem por objetivo substituir um nó isolado não pode ter o seu progresso impedido por uma minoria de nós ou *links* faltosos.

estritamente verdadeiro mas, por outro lado, significa que o *link* apresenta um desempenho suficientemente ruim por um tempo longo o bastante para ser excluído sem impactar negativamente as aplicações.

Para iniciar o protocolo de votação,[3] o nó x difunde uma mensagem assinada m contendo a proposição de reconfiguração R = NEWNODE(y',y) e um voto favorável ("yes") a essa proposição. Cada nó n que recebe m verifica se concorda com R e adiciona o voto correspondente a essa verificação ("yes" ou "no") e a sua assinatura a m, repassando-a em seguida aos seus vizinhos (com exceção daqueles que já têm uma assinatura em m). Quando um nó verifica que uma determinada proposição possui um número mínimo de votos favoráveis ϕ (inferior ao número total de nós), esse nó inicia a segunda fase, que é a fase de ativação propriamente dita.

A fase de ativação do nó de reserva é iniciada no procedimento START-RECONF (algoritmo 2). 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 $votes \geq \phi$ envia uma mensagem ACTIVATE(m) para o nó de reserva r (algoritmo 2, linha 3), onde m é a mensagem que comprova que r detém os votos necessários para ser ativado;
2. Ao receber uma mensagem ACTIVATE(m) (algoritmo 3, linha 2), o nó de reserva r verifica que m contém $votes \geq \phi$ votos favoráveis à sua ativação (linha 5) e envia uma mensagem ESTABLISH-LINK$(m, cert)$ para todos os nós que votaram a seu favor, onde *cert* é o certificado do nó de reserva; (linhas 7–11)
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 2, linhas 5–10);
4. Após receber $n \geq \rho$ mensagens ACK-LINK, r difunde uma mensagem ACTIVE(AL), onde AL=(ACK-LINK$_1$, ..., ACK-LINK$_n$) é o conjunto de ACK-LINKs que prova que o nó conseguiu estabelecer um número suficiente de *links* virtuais (algoritmo 3, linhas 15–22);
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 *link* 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 o estabelecimento de uma chave compartilhada com r (algoritmo 2, linhas 11–24).

Um exemplo de execução bem sucedida do protocolo de reconfiguração (incluindo as fases de votação e de ativação) é mostrado na figura 4. A figura 4(b) mostra as mensagens que são trocadas pelos nó da rede *overlay* da figura 4(a) quando o nó e, isolado pela remoção de seus *links*, é substituído pelo nó f. As mensagens da fase de votação são representadas por $R, n_1, \ldots, n_k$, onde R=NEWNODE(f,e) e $n_1, \ldots, n_k$ 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.

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

[3] Por razões de espaço, os algoritmos contendo a especificação do protocolo de votação tiveram que ser omitidos deste artigo.

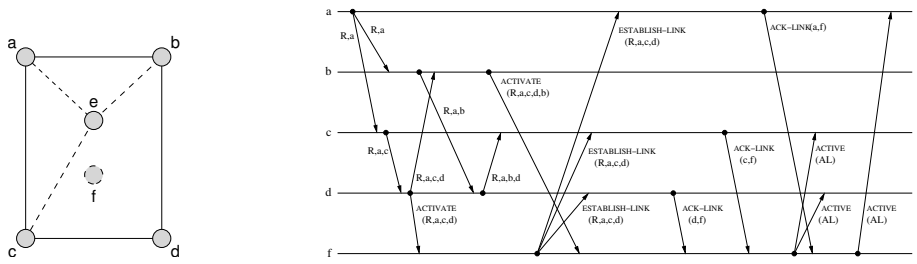
Algoritmo 2 Reconfiguração: START-RECONF

```
1: H ← ∅; EM ← ∅; CN ← ∅
2: procedure START-RECONF(in m: Smsg, R: Reconf)
3:     send ACTIVATE(m) to R.newnode
4:     add ⟨ESTABLISH-LINK(m, cert), R.newnode⟩ to EM                ▷ EM contém as mensagens esperadas
5:     upon receiving ESTABLISH-LINK(m, cert) from newnode do
6:         if (⟨ESTABLISH-LINK(m, cert), newnode⟩ is in EM or (COUNT-VOTES(m.S) ≥ ϕ and myid is in S)) and VALID-
               CERT(cert, newnode) then
7:             send ACK-LINK(myid, newnode, R, seq) to newnode
8:             add ⟨ACTIVE(AL), newnode⟩ to EM
9:         end if
10:    end do
11:    upon receiving ACTIVE(AL) from x do
12:        if ACTIVE(AL) is not in H and AL is valid then
13:            if ⟨ACTIVE(AL),newnode⟩ is in EM and x = newnode then
14:                send BackupNodeList to x                          ▷ BackupNodeList é a lista de nós de reserva
15:            end if
16:            add ACTIVE(AL) to H                                   ▷ H é o histórico de mensagens processadas
17:            add x to CN                                           ▷ CN é o conjunto de nós contactados
18:            for all neighbor n such that n is not in CN do
19:                send ACTIVE(AL) to n
20:                add n to CN
21:            end for
22:            establish shared key with newnode
23:        end if
24:    end do
25: end procedure
```

Algoritmo 3 Reconfiguração: protocolo do nó de reserva

```
1: EM ← ∅; CN ← ∅
2: upon receiving ACTIVATE(m) from n do
3:     ▷ VERIFY-MSG checa que m é válida e a decompõe em seq (no. seqüência), R (reconfiguração proposta) e S (lista de nós com seus
       respectivos votos)
4:     if VERIFY-MSG(m, seq, R, S) then
5:         if myid = R.newnode and (active = ⊥ or active = R) and COUNT-VOTES(S) ≥ ϕ then
6:             active ← R
7:             for all s in S such that s.vote = "yes"and s.id is not in CN do
8:                 send ESTABLISH-LINK(m, mycert) to s
9:                 add s to CN
10:                add ⟨ACK-LINK(s, myid, R, seq), s⟩ to EM
11:            end for
12:        end if
13:    end if
14: end do
15: upon receiving ACK-LINK(n, id, R, seq) from n do
16:    if ⟨ACK-LINK(n, id, R, seq), n⟩ is in EM then
17:        add ⟨ACK-LINK(n, id, R, seq), n⟩ to AL                    ▷ AL é o conjunto de ACK-LINKs
18:        if |AL| ≥ ρ then
19:            send ACTIVE(AL) to all neighbors
20:        end if
21:    end if
22: end do
```

Idealmente, não deve ser possível iniciar uma reconfiguração da rede exclusivamente pela ação de nós maliciosos (nós falhos com comportamento bizantino), sem que uma maioria de nós corretos participem do processo. Esta condição está relacionada ao parâmetro ϕ, e pode ser satisfeita fazendo $\phi \geq 2f + 1$, onde f é o número máximo de nós faltosos. Devem ser igualmente evitadas situações em que um nó de reserva, ao tornar-se ativo, estabeleça *links* virtuais apenas com nós faltosos, o que muito provavelmente viria a isolar o nó do restante da rede. Esta segunda condição está ligada ao parâmetro ρ, e pode ser satisfeita fazendo $\rho \geq f + 1$.



(a) *Overlay* (b) Execução do protocolo

Figura 4: Execução do protocolo de reconfiguração: substituição de e por f

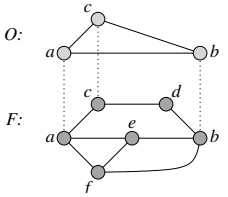
5. Análise de Confiabilidade

5.1. Modelo ROTI

O modelo usado na análise é baseado na arquitetura apresentada na seção 3.1, com uma rede lógica sobreposta a uma rede física. A rede física é modelada por um grafo não dirigido $F = (N_F, L_F)$, onde N_F é o conjunto de vértices que representam os nós da rede física e L_F é o conjunto de arestas que representam os *links* 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, $R(x,y) = \varnothing$. O conjunto de rotas possíveis na rede é definido por $R^* = \bigcup_{x,y \in N_F} R(x,y)$.

A rede *overlay* é modelada por um grafo $O = (N_O, L_O)$. Os nós do *overlay*, tanto ativos como de reserva, são representados pelo conjunto de vértices $N_O \subseteq N_F$. Os *links* virtuais, por sua vez, estão relacionados às rotas na rede física: um *link* virtual xy é implementado pelas rotas entre x e y que não passam por outros nós que pertençam ao *overlay*. Formalmente, isso é representado através da função $\psi\colon [N_O]^2 \to 2^{R^*}$, onde $\psi(x,y) = \{r \in R(x,y) \mid V(r) \cap V(O) = \{x,y\}\}$. A função ψ pode ser melhor compreendida através do seguinte exemplo:



$$\psi(a,b) = \{aeb, afb, afeb\}$$
$$\psi(a,c) = \{ac\}$$
$$\psi(b,c) = \{bdc\}$$

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

5.2. Análise da Tolerância a Faltas e Intrusões

As propriedades de segurança na ROTI são garantidas pelos mecanismos criptográficos utilizados. Esta seção faz uma análise da confiabilidade oferecida pela rede. O objetivo dessa análise é determinar o grau de tolerância da ROTI, obtendo uma expressão para o limite f de faltas e intrusões que a rede é capaz de tolerar.

Uma rede pode ser considerada confiável se ela sempre possui um caminho formado por nós e *links* corretos ligando dois nós quaisquer. Isso equivale a dizer que a rede deve permanecer conectada se forem removidos todos os nós e *links* faltosos. Dado um grafo G, o mínimo número de nós que, se removidos, provocam a desconexão de G é dito a sua conexidade $\kappa(G)$; se $\kappa(G) = k$, G é dito k-conexo.[4] Logo, a confiabilidade de uma rede está ligada à conexidade

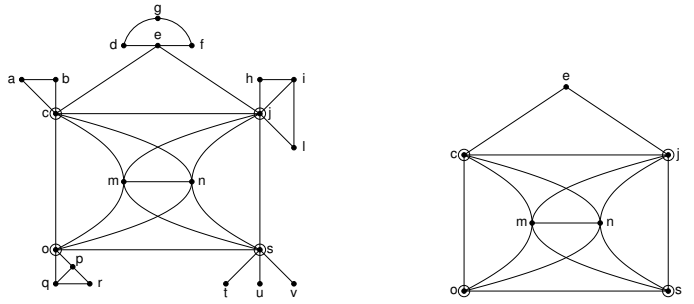
[4]A conexidade de arestas $\lambda(G)$ é definida de forma análoga a $\kappa(G)$, com arestas no lugar de nós. Entretanto,

do grafo que a representa, conforme exprime o seguinte lema:
Lema 5.1. *Uma rede modelada por um grafo G k-conexo é capaz de tolerar até* $f = k-1$ *faltas.*

No caso da ROTI, todavia, 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 *link* virtual é implementado por uma ou mais rotas físicas. Essa relação corresponde à projeção da rede *overlay* sobre a rede física que a implementa, que é representada por um grafo $\hat{O} = (N_{\hat{O}}, L_{\hat{O}})$. O conjunto de vértices $N_{\hat{O}}$ contém os nós *overlay* e os nós da rede física usados para implementar um ou mais *links* virtuais, e o conjunto de arestas $L_{\hat{O}}$ contém os *links* físicos usados para implementar um ou mais *links* virtuais. Formalmente, $\hat{O}$ é dado por[5]

$$\hat{O}: \begin{cases} N_{\hat{O}} = N_O \cup \left\{ v \in V(r) \mid \exists r \in R^*_{\hat{O}} \right\} \\ L_{\hat{O}} = \left\{ l \in E(r) \mid \exists r \in R^*_{\hat{O}} \right\} \end{cases} \quad \text{onde} \quad R^*_{\hat{O}} = \bigcup_{x,y \in N_O} R(x,y)$$

Seja o exemplo da figura 5(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 concêntricos na figura). A figura 5(b) mostra o grafo $\hat{O}$ que representa a projeção da rede *overlay* formada por esses nós sobre a rede física da figura 5(a). Observa-se que no grafo $\hat{O}$ são eliminados da rede física todos os nós e *links* que não são usados por nenhum *link* virtual, e que portanto não têm nenhuma influência sobre a rede *overlay*.



(a) Rede física (b) Grafo $\hat{O}$

Figura 5: Exemplo de projeção de rede *overlay*

A análise da conexidade 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 5(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 conexidade local entre os pares de nós *overlay*,[6] e não a conexidade do grafo $\hat{O}$ como um todo. Como a conexidade local $\kappa(x,y)$ só é definida para vértices não adjacentes, utilizam-se os seguintes teoremas:
Teorema 5.2 (Teorema de Menger [8]). *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.*
Teorema 5.3. *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.*

---

um teorema básico diz que $\kappa(G) \leq \lambda(G)$ [8], de modo que a presente análise considera apenas $\kappa(G)$.

[5] $V(G)$ e $E(G)$ sendo, respectivamente, o conjunto de vértices e o conjunto de arestas do grafo *G*.

[6] A conexidade local $\kappa(x,y)$ de dois vértices não adjacentes *x* e *y* de um grafo *G* é o menor número de vértices cuja remoção separa *x* e *y* em *G* [8].

Demonstração. Caso x e y não sejam adjacentes, são necessárias $k = \kappa_{\hat{O}}(x,y)$ faltas 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 5.2, k é igual ao número máximo de rotas disjuntas entre x e y. Como, pelo lema 5.1, o número de faltas f é inferior a k, completa-se a demonstração. □

Essa 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:

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

Teorema 5.4. *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 $f < k$. □

A condição do teorema 5.4 é que $f < k_{\hat{O}}$, onde $k_{\hat{O}}$ é dado por:

$$k_{\hat{O}} = \min_{x,y \in V(O), x \neq y} k_{\hat{O}}(x,y) \quad (2)$$

Além da condição do teorema 5.4, é importante observar que o número de faltas f está sujeito a uma condição adicional. Essa condição é que a comunicação no *overlay* só faz sentido quando existem pelo menos dois nós corretos; nesse caso, tem-se $f \leq k_O$, onde k_O é dado por:

$$k_O = |O| - 2 \quad (3)$$

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

$$f \leq \min(k_O, k_{\hat{O}} - 1) \quad (4)$$

onde $k_{\hat{O}}$ e k_O são dados, respectivamente, pelas expressões (2) e (3).

No exemplo da figura 5, a rede *overlay* possui quatro nós (*c*, *j*, *s* e *o*); portanto, $k_O = 2$. Para a análise da conexidade, usando as expressões (1) e (2) obtém-se $k_{\hat{O}} = 4$. A partir daí, pela expressão (4), chega-se a $f \leq \min(2, 4-1) \Rightarrow \boxed{f \leq 2}$.

5.3. 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 probabilista (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 5, determinou-se que $f \leq 2$; entretanto, se apenas os nós *overlay* (*c*, *j*, *o* e *s*) e o nó *m*—assim como os *links cm*, *jm*, *om* e *sm*—forem corretos, a rede *overlay* permanece conectada, mesmo que os 16 nós e os 29 *links* 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 (quando não impossível) determinar o grafo $\hat{O}$ caso os nós *overlay* estejam topologicamente distantes na rede física. 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 (*Autonomous Systems*) que compõem a camada de roteamento da Internet. Esse 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 do que um grafo representando a rede física como um todo.

6. Trabalhos Relacionados

Diversos protocolos seguros de roteamento, com diferentes premissas e objetivos, são registrados na literatura. As principais propostas são analisadas por Papadimitratos e Haas em um *survey* concentrado em protocolos voltados à Internet [13].

O trabalho pioneiro sobre redes tolerantes a faltas bizantinas é a tese de Perlman [16], que utiliza uma combinação de *flooding* e roteamento *link state*; a influência desse trabalho pode ser constatada em muitas propostas atuais. O roteamento considerando faltas bizantinas adquire maior importância em redes *ad hoc* sem fio, onde a possibilidade dos nós estarem expostos a um ambiente hostil é maior do que em redes convencionais; propostas significativas nesse contexto incluem OSDBR [4] e Ariadne [9].

A FVPN [7] é uma VPN (*Virtual Private Network*) tolerante a faltas com objetivos similares aos da ROTI em termos de prover um serviço seguro de comunicação. Uma diferença básica entre a FVPN e a ROTI é que a primeira é projetada para uso em roteamento intradomínios, ao passo que a ROTI é projetada para contornar limitações do roteamento interdomínios. Outra diferença profunda é a semântica de falhas: a FVPN age apenas em reação a falhas sinalizadas pelo protocolo de roteamento *link state*, o que não inclui comportamento bizantino dos elementos de rede.

O problema de nós bizantinos em redes *overlay* estruturadas, com aplicação no Pastry [17], é discutido em [5]. Redes *overlay* estruturadas são usadas em aplicações *peer-to-peer* (P2P) de larga escala, e possibilitam a localização de objetos em um número pequeno (probabilisticamente limitado) de *hops*, usando tabelas de roteamento com poucas entradas em cada nó. Isso é obtido através de uma certa complexidade na construção e manutenção do *overlay* para que ele sempre respeite uma topologia predeterminada. O ponto principal do trabalho de Castro *et al.* é um mecanismo de roteamento seguro, que possui três componentes: atribuição segura de identificadores de nó (endereços virtuais), manutenção segura da tabela de roteamento e encaminhamento seguro de mensagens. Comparando esse trabalho com a ROTI, observa-se que a atribuição de endereços virtuais é similar, usando certificados digitais para determinar quais nós fazem parte do *overlay*. A manutenção da tabela de roteamento e o encaminhamento de mensagens, por sua vez, possuem o mesmo espírito nas duas redes, embora as técnicas usadas sejam distintas por causa da diferença que existe entre uma rede *overlay* como a ROTI e um *overlay* estruturado, onde o conteúdo de uma mensagem é que determina como ela deve ser roteada. No caso da manutenção da tabela de roteamento, o conceito chave compartilhado entre as propostas é o de apenas atualizar a tabela de roteamento com informações confiáveis. Com relação ao encaminhamento de mensagens, Castro *et al.* usam um detector de falhas (não baseado em ACKs) para descobrir se uma mensagem chegou ao seu destino correto; caso haja indicação de uma falha, a mensagem é retransmitida por rotas diversas redundantes (embora não necessariamente disjuntas). Outra diferença entre a ROTI e o Pastry é que este último não possui preocupação com a confidencialidade das mensagens, deixada a critério da aplicação que utiliza o *overlay*.

7. Conclusões

Este artigo apresentou a ROTI, um exemplo de como é possível aproveitar a flexibilidade inerente às redes *overlay* para implementar um suporte de comunicação com alta confiabilidade mesmo na presença de faltas e intrusões. Dois pontos importantes no projeto da ROTI são a sua capacidade adaptativa e a autonomia que cada nó tem para detectar e contornar faltas e intrusões na rede.

Uma perspectiva que se abre é a obtenção de resultados experimentais que permitam comparar o desempenho da ROTI com o de uma rede *overlay* similar mas que tolere apenas faltas mais simples, como a RON. Isso permitiria mensurar o impacto dos mecanismos utilizados para garantir a segurança e a confiabilidade das comunicações. Outra possibilidade é trabalhar na caracterização do comportamento da rede quando o número de faltas e intrusões supera o limite que ela foi projetada para tolerar, o que permitiria compreender de que forma a rede se degrada sob condições extremamente adversas.

Do ponto de vista de análise, a teoria de grafos se revela uma técnica útil para determinar o grau de tolerância a faltas de uma rede *overlay*. Uma possível extensão do trabalho nessa direção seria modelar a rede através de um outro formalismo que leve em conta a evolução da topologia, como um dos modelos de grafos dinâmicos existentes.

Referências

[1] Y. Amir e C. Danilov. "Reliable Communication in Overlay Networks". In *Proc. Int. Conf. on Dependable Systems and Networks (DSN'2003)*, pp. 511–520, San Francisco, 2003.

[2] Y. Amir, C. Danilov e C. Nita-Rotaru. "High Performance, Robust, Secure and Transparent Overlay Network Service". In *Proc. Int. Workshop on Future Directions in Distributed Computing*, Bertinoro, Italy, 2002.

[3] D.G. Andersen, H. Balakrishnan, F. Kaashoek e R. Morris. "Resilient Overlay Networks". In *Proc. ACM Symp. on Operating Systems Principles*, pp. 131–145, Banff, Canada, 2001.

[4] B. Awerbuch, D. Holmer, C. Nita-Rotaru e H. Rubens. "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures". In *Proc. ACM Workshop on Wireless Security*, pp. 21–30, Atlanta, 2002.

[5] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, e D.S. Wallach. "Secure Routing for Structured Peer-to-Peer Overlay Networks". In *Proc. Symp. on Operating Systems Design and Implementation*, Boston, 2002.

[6] M. Dahlin, B. Chandra, L. Gao e A. Nayate. "End-to-end WAN Service Availability". *IEEE/ACM Trans. Networking*, 11(2):300–313, 2003.

[7] J. Han, G.R. Malan e F. Jahanian. "Fault-Tolerant Virtual Private Networks within An Autonomous System". In *Proc. Symp. on Reliable Distributed Systems*, pp. 41–50, Suita, Japan, 2002.

[8] F. Harary. *Graph Theory*. Addison-Wesley, Reading, USA, 1969.

[9] Y-C. Hu, A. Perrig e D.B. Johnson. "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks". In *Proc. MobiCom*, pp. 12–23, Atlanta, 2002.

[10] H. Krawczyk, M. Bellare e R. Canetti. "HMAC: Keyed-Hashing for Message Authentication". RFC 2104, Internet Engineering Task Force, 1997.

[11] C. Labovitz, A. Ahuja, A. Bose e F. Jahanian. "Delayed Internet Routing Convergence". *IEEE/ACM Trans. Networking*, 9(3):293–306, 2001.

[12] L. Lamport, R. Shostak e M. Pease. "The Byzantine Generals Problem". *ACM Trans. Programming Languages and Systems*, 4(3):382–401, 1982.

[13] P. Papadimitratos e Z.J. Haas. "Securing the Internet Routing Infrastructure". *IEEE Communications*, 40 (10):60–68, 2002.

[14] V. Paxson. "End-to-End Routing Behavior in the Internet". *IEEE/ACM Trans. Networking*, 5(5):601–615, 1997.

[15] V. Paxson. "End-to-End Internet Packet Dynamics". *IEEE/ACM Trans. Networking*, 7(3):277–292, 1999.

[16] R.J. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachusetts Institute of Technology, USA, 1988. Available as MIT-LCS-TR-429.

[17] A. Rowstron e P. Druschel. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems". In *Proc. IFIP/ACM Int. Conf. on Distributed Systems Platforms*, pp. 329–350, Heidelberg, Germany, 2001.