7
Referências Bibliográficas

ALAETTINOGLU, C. et al. Design and implementation of MaRS: A routing testbed. **Journal of Internetworking**: Research and Experience, 5(1): 17-41, 1994.

ALLEN,A.O. **Probability,statistics, and queueing theory: with computer science applications** – 2nd ed. San Diego: Academic Press, 1990.

BAE, S.H. et al. The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks. **IEEE Network**, vol. 14, 70-77. January 2000.

BAGRODIA, R. et al. News. Parsec: A parallel simulation environment for complex systems. **IEEE Computer**, 31(10): 77-85, oct. 1998.

BHATTI, N.T. et al. Coyote: a system for constructing fine-grain configurable communication services. **ACM Transactions on Computer Systems**, 16(4): 321-366, November 1998.

BLUETOOTH Special Interest Group (SIG). **The Official Bluetooth Wireless Info Site**. Disponível em <http://www.bluetooth.com>. Acesso em 26 jun 2002.

BROCH, J. et al. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (**MobiCom'98**), ACM, Dallas, TX, October 1998. Disponível em <http://www.monarch.cs.rice.edu/monarch-papers/mobicom98.ps>. Acesso em out 2002.

CHEN, T.; GERLA, M. Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks. **Proc. IEEE ICC'98**, 5 pages, jun 1998. Disponível em: <http://www.ics.uci.edu/~atm/adhoc/paper-collection/gerla-gsr-icc98.pdf>. Acesso em 1 nov 2001.

COMER, D.E. **Interligação em Rede com TCP/IP**: Princípios, protocolos e arquitetura. Volume I. Tradução da terceira edição de: *Internetworking with TCP/IP*. Rio de Janeiro: Campus, 1998. 672p.

COMER, D.E.; STEVENS, D.L. **Interligação em Rede com TCP/IP**: Projeto, implementação e detalhes internos. Volume II. Tradução da terceira edição de: *Internetworking with TCP/IP*. Rio de Janeiro: Campus, 1999. 592p.

CORDEIRO, C.M.; AGRAWAL, D.P. Mobile Ad hoc Networking. Minicurso 3 . **Simpósio Brasileiro de Redes de Computadores (SBRC 2002)**, 20, 2002, Búzios, RJ, 20 a 24 mai. 2002.

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, R.L. **Introduction to Algorithms.** MIT Presss, 1990.

CORSON, S.; MACKER, J. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. **IETF RFC 2501**, jan. 1999.

COWIE, J.H. Scalable Simulation Framework API Reference Manual - Version 1.0 - **Documentation Draft** - Revision March 25, 1999. Disponível em <http://www.ssfnet.org/SSFdocs/ssfapiManual.pdf>. Acesso em 01 out 2002.

CROW, B.P. et al. IEEE 802.11 Wireless Local Area Networks. In **IEEE Communications Magazine**, 116-126. September 1997.

DAS, S.R.; CASTAÑEDA R.; YAN, J. Simulation Based Performance Evaluation of Mobile, Ad hoc Network Routing Protocols. ACM/Baltzer **Mobile Networks and Applications (MONET) Journal**, pages 179-189, July 2000.

ETSI. Hiperlan Functional Specification. **ETSI Draft Standard**. July 1995.

FONTOURA, M.F.M.. **A Systematic Approach to Framework Development**. Tese de Doutorado, PUC-RJ, Rio de Janeiro, mai 2001.

GAMMA, E. at all. **Padrões de Projeto**: soluções reutilizáveis de software orientado a objetos/ Erich Gama, Richard Helm, Ralf Johnson e John Vlissides; trad. Luiz A Meirelles Salgado. – Porto Alegre: Bookman, 2000.

GLOMOSIM. **GloMoSim**: Simulador para Computação Móvel. Disponível em <http://pcl.cs.ucla.edu/projects/glomosim/>. Acesso em: 26 jun. 2002.

HORSTMAN, C.S.; CORNELL, G.**Core Java 2 – Volume1** – Fundamentos, tradução: João Eduardo Nóbrega Tortello, revisão técnica: Álvaro Rodrigues Antunes. São Paulo: Makron Books, 2001.

_____. **Core Java 2 – Volume2** – Recursos Avançados, tradução: João Eduardo Nóbrega Tortello, revisão técnica: Álvaro Rodrigues Antunes. São Paulo: Makron Books, 2001.

HUTCHINSON, N.C.; PETERSON, L. L. The x-Kernel: An architecture for implementing network protocols. **IEEE Transactions on Software Engineering**, 17(1):64-76, Jan. 1991. Disponível em <ftp://ftp.cs.arizona.edu/xkernel/Papers/architecture.ps>. Acesso em: 25 set 2002

IWATA, A. et al. Scalable Routing Strategies for Ad Hoc Wireless Networks. IEEE Journal on Selected Areas in **Communications, Special Issue on Ad-Hoc Networks**, Aug. 1999, pp.1369-1379. Disponível em: <http://www.cs.ucla.edu/NRL/wireless/PAPER/jsac99.ps.gz>. Acesso em 1 nov 2001.

JIANG, M.; LI, J.; TAY, Y.C. Cluster Based Routing Protocol, aug. 1999 **IETF Draft**, 27 pages. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-manet-cbrp-spec-01.txt>. Acesso em 1 nov 2001.

JOA-NG, M.; LU, I.-T. A Peer-to-Peer zone-based two-level link state routing for mobile Ad Hoc Networks. IEEE Journal on Selected Areas in **Communications, Special Issue on Ad-Hoc Networks**, aug. 1999, pp.1415-25.

JOHNSON, D.B.; MALTZ, D.A. Dynamic Source Routing in Ad Hoc Networks. **Mobile Computing**, T. Imielinski and H. Korth, Eds., Kulwer, 1996, pp. 152-81. Disponível em <http://www.ics.uci.edu/~atm/adhoc/paper-collection/johnson-dsr.pdf>. Acesso em 1 nov 2002.

_____. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, oct 1999. **IETF Draft**, 49 pages. Disponível em http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-03.txt>. Acesso em 1 nov 2001.

LOUREIRO, A.A.F.; CÂMARA, D. Roteamento em Redes Móveis Ad Hoc. Minicurso 4. In: **Workshop de Comunicação Sem Fio**, 1. 1999, Belo Horizonte, MG, 14 a 16 jul. 1999.

MACHADO, M.V. et al. Avaliação de Classes de Nodos em Algoritmos de Roteamento para Rede Móveis Ad Hoc. **WCSF2002** – Outubro de 2002

MATEUS, G.R.; Loureiro, A.A.F. **Introdução à Computação Móvel**. Rio de Janeiro: DCC/IM, COPPE/Sistemas, NCE/UFRJ, 1998.

MATTSSON, M. Object-Oriented Frameworks: A Survey of Methodological Issues. 1996

MELE, A. **Protocolos de Roteamento para Redes Móveis Ad Hoc**. Rio de Janeiro, nov. 2001. Monografia (disciplina "Introdução à Computação Móvel", Prof. Dr. Markus Endler) - Programa de Mestrado em Informática, PUC-Rio.

MISRA, P. **Routing Protocols for Ad Hoc Mobile Wireless Networks**, nov 1999. Disponível em <http://www.cis.ohio-state.edu/~jain/cis788-99/adhoc_routing/index.html>. Acesso em 1 nov 2001.

MOBICS. **MobiCS**: Simulador para Computação Móvel. Disponível em <http://www.inf.puc-rio.br/~endler/MobiCS/>.Acesso em 26 jun.2002.

NS-2. **The Network Simulator**. Disponível em <http://www.isi.edu/nsnam/ns/>. Acesso em 26 jun. 2002.

NS-2. THE VINT PROJECT. **The ns Manual (formerly ns Notes and Documentation)**, 31 jan 2002. Disponível em <http://www.isi.edu/nsnam/ns/ns-documentation.html>. Acesso em 26 jun 2002.

PARK, S.; SAVVIDES, A.; SRIVASTAVA, M.B. **SensorSim**: a simulation framework for sensor networks. International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems - Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems - 2000, Boston, Massachusetts, United States - Pages: 104 – 111

PARK, V.D.; CORSON, M.S. A highly adaptive distributed routing algorithm for mobile wireless networks. **Proc. INFOCOM'97**, apr. 1997, 9 pages. Disponível em <http://www.ics.uci.edu/~atm/adhoc/paper-collection/corson-adaptive-routing-infocom97.pdf>. Acesso em 1 nov 2001.

PERKINS, C.E. **Ad Hoc Networking**. Addison-Wesley, ISBN: 0201309769, 2001.

PERKINS, C.E.; BHAGWAT, P. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. **Comp. Comm. Rev.**, oct. 1994, pp.234-244. Disponível em <http://www.svrloc.org/~charliep/txt/sigcomm94/paper.ps>. Acesso em 1 nov 2001.

PERKINS, C.E.; ROYER, E.M.; DAS, S.R. Ad Hoc On-demand Distance Vector Routing, oct. 1999 **IETF Draft**, 33 pages. Disponível em <http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-04.txt>. Acesso em 1 nov 2001.

ROCHA, R.A.; ENDLER, M. Flexible Simulation of Distributed Protocols for Mobile Computing. 3rd **Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)**, 123-126, Boston, August 11th, 2000. Disponível em <http://www-di.inf.puc-rio.br/~endler/paperlinks/mswim00.ps.gz>. Acesso em 6 jun 2002.

ROCHA, R.C.A. **Uma Arquitetura para Simulação Flexível de Protocolos para Computação Móvel**. Dissertação de Mestrado, IME-USP, São Paulo-SP, mai 2001.

ROYER, E. M.; LEE, S-J.; PERKINS,C.E. The Effects of MAC Protocols on Ad hoc Communication Protocols. In **Proceedings of IEEE WCNC 2000**, sep. 2000.

ROYER, E.M.; PERKINS, C.E. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In **ACM MOBICOM**, 207-218, aug. 1999.

ROYER, E.M.; TOH, C.-K. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. **IEEE Personal Communications**, v. 6, n. 2, pp. 46-55, apr. 1999. Disponível em <http://users.ece.gatech.edu/~cktoh/royer.html>. Acesso em 1 nov 2001.

ROYER, E.M.; TOH, C.-K. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. **IEEE Personal Communications**, v. 6, n. 2, pp. 46-55, apr. 1999 Disponível em <http://www.cs.ucsb.edu/~eroyer/txt/review.ps>. Acesso em 26 jun 2002.

SOARES, L.F.G **Modelagem e Simulação Discreta de Sistemas**. São Paulo: IME-USP, 1990. 250 p.

. **Redes de Computadores: das LANs, MANs e WANs às redes ATM** – 2. ed. rev. e ampliada. Rio de Janeiro: Campus, 1995.

SSF. **SSF Research Network**. Disponível em < http://www.ssfnet.org>. Acesso em 01 out 2002.

TANENBAUM, A.S. **Redes de Computadores**. Tradução da 3a edição, Ed. Campus, 1997.

ZENG, X.; BAGRODIA, R.; GERLA, M. GloMoSim: A library for parallel simulation of large-scale wireless networks. In **Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS-98)**, IEEE Computer Society, pages 154-161, Los Alamitos, may 26-29 1998.

8
Apêndice – Protocolos de Roteamento para MANET

Em redes tradicionais como a Internet, o roteamento por prefixo é baseado no endereço do destinatário. Em redes ad hoc o principal problema é a localização dos nós, pois estes não têm uma posição fixa ou pré-definida e podem mover-se livremente. Determinar a localização exata do nó não é simples ou até impossível. Isso torna o problema de roteamento em redes ad hoc fundamental, pois passa a ser a base de toda a comunicação da rede móvel (Loureiro & Câmara, 1999).

8.1.
Introdução

Alguns dos algoritmos clássicos de roteamento para redes fixas, como Vetor de Distância (*Distance Vector*), Estado de Enlace (*Link State*), Inundação (*Flooding*) e Difusão (*broadcasting*) têm as suas idéias incorporadas e adaptadas para as redes móveis ad hoc. Tanenbaum (1997) faz uma descrição detalhada destes e de outros algoritmos de roteamento para redes fixas.

É esperado que as redes móveis ad hoc tenham uma topologia dinâmica, com mudanças freqüentes e com reestruturação aleatória, e que esta topologia seja *multihop*. É também esperado que os seus enlaces sem fio tenham uma largura de banda restrita e de qualidade variável. Portanto, em relação às redes fixas, os protocolos de roteamento para MANET precisam ser mais dinâmicos para se adaptarem rapidamente às mudanças de topologia e de qualidade de transmissão do meio. Além disto os protocolos de roteamento para MANET precisam também levar em consideração o problema de energia restrita dos hosts móveis, o que não é um problema nas redes fixas.

8.1.1. Classificação

De forma geral, os protocolos de roteamento para redes móveis ad hoc podem ser classificados em dois tipos (Royer & Toh, 1999): Orientado a tabela (*table-driven*) e sob demanda (*on-demand*). Esta classificação é feita baseada em quando e como as rotas são descobertas.

Nos protocolos de roteamento orientados a tabela, cada nó mantém a informação de roteamento atualizada e consistente para todos os outros nós, enquanto que nos protocolos de roteamento sob demanda, as rotas são descobertas somente quando o host de origem as necessita.

Apesar destes protocolos de roteamento terem sido projetados para o mesmo tipo de rede, as suas características são bem distintas. As próximas seções descrevem de forma resumida as características e funcionalidades de alguns destes protocolos, organizados por suas categorias.

8.2. Orientado a tabela

Os protocolos de roteamento orientados a tabela tentam manter a informação de roteamento de cada nó consistente e atualizada para todos os outros nós da rede. Estes protocolos necessitam que cada nó mantenha uma ou mais tabelas para armazenar a informação de roteamento, e se adaptam a mudanças na topologia da rede através da propagação de mensagens de atualizações que visam manter uma visão consistente de caminhos na rede. Os aspectos nos quais estes protocolos diferenciam são o número de tabelas necessárias e os métodos para difundir as mudanças na estrutura da rede. Esta seção discute alguns dos protocolos de roteamento orientados a tabela propostos para redes móveis ad hoc.

8.2.1. Destination-Sequenced Distance-Vector Routing Algorithm (DSDV)

O *Destination-Sequenced Distance-Vector Routing Algorithm* (DSDV) (Perkins & Bhagwat, 1994) é baseado na idéia do algoritmo clássico de roteamento *Bellman-Ford*, com certos aperfeiçoamentos.

No DSDV, cada estação móvel mantém uma tabela de roteamento que lista todos os destinos disponíveis, o número de *hops* para alcançar o destino e o número seqüencial associado pelo nó destino. O número seqüencial é usado para distinguir rotas velhas de novas e assim evitar a formação de loops. As estações periodicamente transmitem suas tabelas de rotas para os seus vizinhos imediatos. Uma estação também transmite sua tabela de rota se uma mudança significativa tiver ocorrido em sua tabela desde a última transmissão de atualização. Portanto, a atualização é tanto disparada pela passagem do tempo como pela ocorrência de um evento. As atualizações da tabela de roteamento podem ser enviadas de duas formas: "*full dump*" ou atualização incremental. No *full dump* a estação envia a tabela de roteamento completa para os vizinhos e conseqüentemente pode envolver muitos pacotes, enquanto que em uma atualização incremental somente são enviadas aquelas entradas da tabela de roteamento que tiverem as suas métricas alteradas desde a última atualização, sendo que esta informação deve caber em um único pacote. Se houver espaço no pacote de atualização incremental então aquelas entradas que tiveram uma alteração somente no número seqüencial podem também ser incluídas. Quando a rede está relativamente estável, as atualizações incrementais são enviadas para evitar um tráfego extra e a atualização *full dump* é feita com menor freqüência. Em uma rede que muda de topologia rapidamente, o número de pacotes incrementais pode crescer muito, então serão feitas com mais freqüência as atualizações *full dump*, em vez de atualizações incrementais. Cada pacote de atualização de rota, além de conter a informação da tabela de roteamento, também inclui um número seqüencial único definido pelo transmissor. A rota rotulada com o maior número seqüencial (i.e. a mais recente) é usada. Se duas rotas têm o mesmo número seqüencial então a rota com a melhor métrica (p.ex., menor no. de *hops*) é usada.

8.2.2. Global State Routing (GSR)

O *Global State Routing* (GSR) (Chen & Gerla, 1998) é um protocolo de roteamento baseado no protocolo *link state*. Como no *link state*, este protocolo mantém uma visão global da rede e pode fazer cálculos precisos de rotas. O GSR aprimora o *link state* evitando a inundação (*flooding*) de mensagens de

roteamento. Uma das características mais interessantes do GSR é a sua capacidade de manipular parâmetros de QoS.

Neste protocolo, cada nó mantém uma Lista de Vizinhos, uma Tabela de Topologia, uma Tabela de Próximo *Hop* e uma Tabela de Distância. A Lista de Vizinhos de um nó contém a relação de todos os nós que podem ser alcançados diretamente por este nó sem a necessidade de roteamento. Para cada possível nó destino, a Tabela de Topologia contém a informação sobre o estado do enlace (*link state*) como reportado pelo destino e o *timestamp* da informação. Para cada destino, a Tabela de Próximo *Hop* contém o próximo *hop* para qual os pacotes para este destino devem ser encaminhados. E a Tabela de Distância contém a menor distância para cada nó destino.

As mensagens de roteamento são geradas na mudança de um enlace, como ocorre nos protocolos de estado de enlace (*link state*). Ao receber uma mensagem de roteamento, o nó atualiza a sua Tabela de Topologia se o número seqüencial da mensagem for mais novo do que o número seqüencial armazenado na tabela. Após isto, o nó reconstrói sua tabela de roteamento e difunde a informação aos seus vizinhos, mas apenas para os seus vizinhos, ao contrário do *link state* que envia para todos os nós da rede.

Existem duas técnicas para reduzir o tamanho das mensagens de atualização: *fresh update* e *fisheye* (olho de peixe). O *fisheye* será discutido na próxima seção (8.2.3).

No *fresh update* somente informações úteis aos vizinhos são disseminadas. A idéia é que se a informação sobre um nó é mais velha que a do meu vizinho, não faz sentido enviar esta informação para ele. Quando as informações obsoletas são eliminadas, a tabela de roteamento pode ser reduzida.

O *fresh update* ocorre em dois passos:

1. Primeiro cada nó envia um Vetor de Seqüências de Números (VSN) onde são indicados todos os destinos que estão na tabela e a idade de cada informação.
2. No segundo passo cada nó compara o seu VSN com os valores recebidos dos vizinhos. Se alguma entrada do seu VSN for mais antiga, então esta entrada é removida das mensagens de atualização. Após o segundo passo o resultado das atualizações de tabelas é disseminado.

8.2.3.
Fisheye State Routing (FSR)

Fisheye State Routing (FSR) (Iwata, 1999) é um aprimoramento do GSR. O grande tamanho das mensagens de atualização no GSR desperdiça uma quantidade considerável da largura de banda da rede. No FSR, cada mensagem de atualização não contém informação sobre todos os nós. Em vez disto, o FSR troca informação sobre os nós mais próximos mais freqüentemente do que faz para com os nós mais distantes, desta forma reduzindo o tamanho da mensagem de atualização. Assim cada nó tem uma informação precisa sobre seus vizinhos, sendo que o detalhe e exatidão da informação diminui quando aumenta a distância para o nó. A Figura 14 define o escopo do "olho de peixe" (*fisheye*) para o nó central (vermelho). O escopo é definido em termos dos nós que podem ser alcançados em um certo número de *hops*. O nó central tem uma informação mais precisa de todos os nós no círculo mais interno, e para cada área que mais afastada do centro a informação fica menos precisa. Apesar de um nó não ter uma informação precisa sobre os nós mais distantes, os pacotes são roteados corretamente porque a informação de roteamento se torna mais e mais precisa a medida que o pacote se aproxima mais do destino. O FSR é bem escalável para redes grandes, pois o *overhead* é controlado neste esquema.

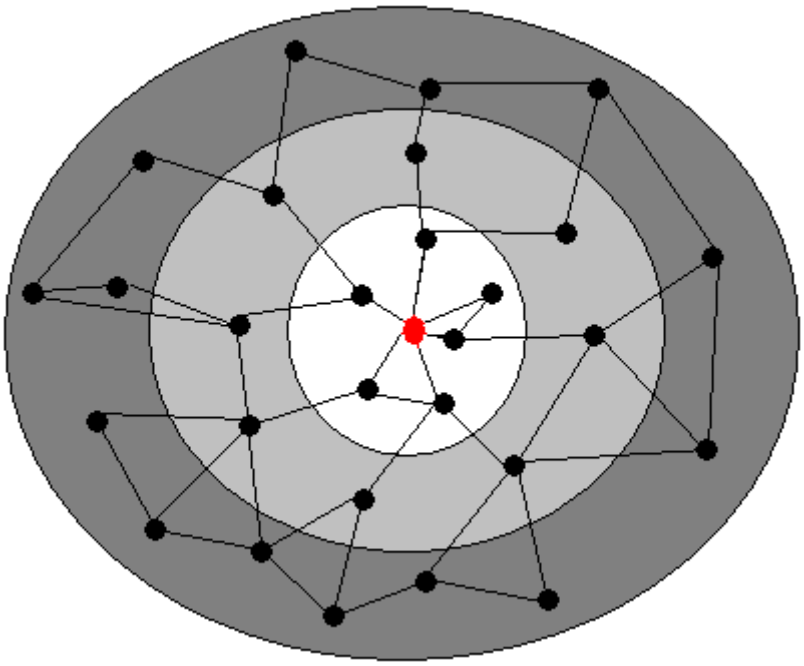


Figura 14 – Precisão da informação no FSR

8.2.4. Hierarchical State Routing (HSR)

A característica principal do *Hierarchical State Routing* (HSR) (Iwata, 1999) é o agrupamento em multiníveis (*multilevel clustering*) e o particionamento lógico dos nós móveis. A rede é particionada em clusters e é eleito um representante para cada cluster, chamado de *cluster-head*, baseado em algum algoritmo de eleição. Os nós de um cluster físico difundem entre si as suas informações dos seus enlaces. O *cluster-head* resume as informações do seu cluster e transmite-as para *cluster-heads* vizinhos através dos nós *gateway* (nós que pertencem a dois ou mais clusters vizinhos). Como mostrado na Figura 15, estes *cluster-heads* são membros do cluster em um nível mais alto e trocam entre si suas informações de seus enlaces, assim como a informação resumida do nível abaixo, e assim por diante. Um nó em cada nível difunde para seu nível inferior a informação que obteve dos nós de seu nível, redistribuindo desta forma a informação para todos os nós. Com isso, o nível inferior tem uma informação da topologia hierárquica. Cada nó possui um endereço hierárquico, sendo que uma forma de atribuir este endereço é usar os números dos clusters desde a raiz até o nó físico. Por exemplo na Figura 15, o nó 4 no cluster do nível físico tem o endereço hierárquico. Como dito anteriormente, um *gateway* pertence a mais de

um cluster, logo possui mais de um endereço hierárquico <1,2,4>. Um endereço hierárquico é suficiente para identificar unicamente o host na rede.

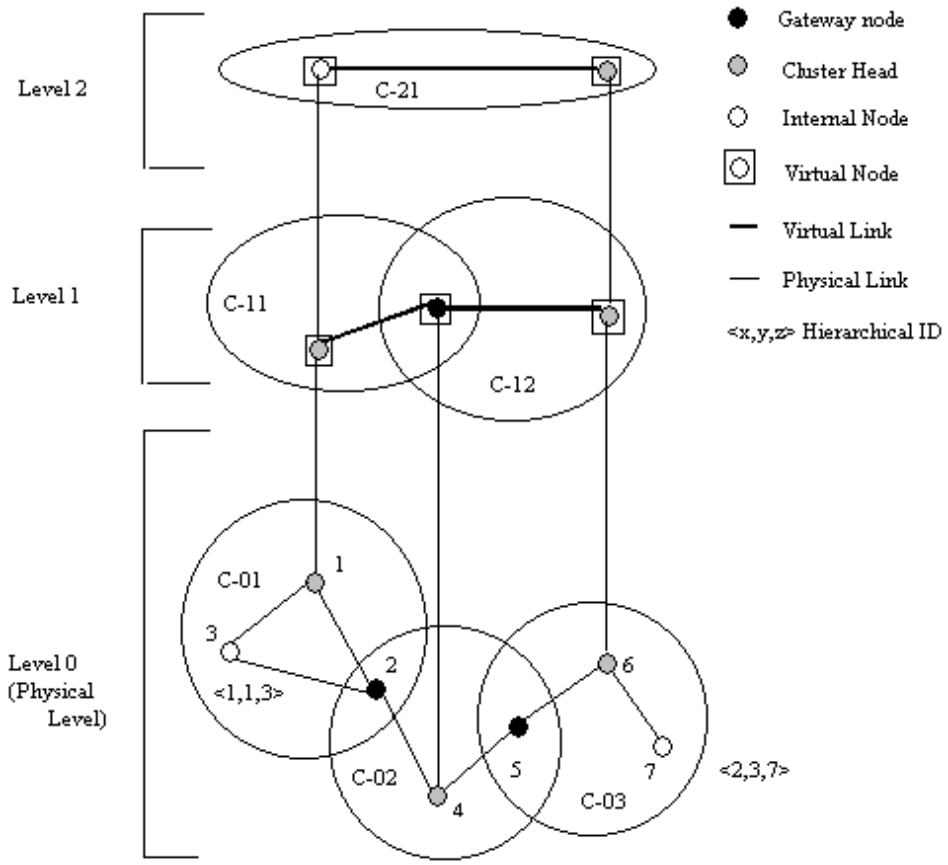


Figura 15 – Um exemplo de *clustering* no HSR

Os nós também são particionados em sub-redes lógicas e para cada nó é atribuído um endereço lógico <sub-rede, host>. Cada sub-rede possui um servidor de gerenciamento de localização (LMS – *Location Management Server*), onde todos os nós da sub-rede registram seus endereços lógicos. O LMS informa seu endereço hierárquico para os níveis mais altos e a informação é também enviada para baixo para todos os LMSs. A camada de transporte envia um pacote para a camada de rede usando o endereço lógico do destino. A camada de rede encontra o endereço hierárquico do LMS da sub-rede do destino, a partir do seu LMS, e então envia o pacote. Então o LMS do destino encaminha o pacote para o nó de destino. Uma vez que a fonte e o destino sabem seus endereços hierárquicos, eles não precisam mais dos LMSs e podem se comunicar

diretamente. Uma vez que tanto o endereço lógico como o endereço hierárquico é usado para roteamento, eles são adaptáveis às mudanças da rede.

8.2.5.
Zone-based Hierarchical Link State Routing Protocol (ZHLS)

No *Zone-based Hierarchical Link State Routing Protocol* (ZHLS) (Joa et al, 1999), a rede é dividida em zonas sem sobreposição. Ao contrário de outros protocolos hierárquicos, não existe um representante da zona (*zone-head*). O ZHLS define dois níveis de topologia: nível do nó e nível da zona. A topologia de nível do nó informa como os nós de uma zona estão conectados fisicamente entre si. Um enlace virtual entre duas zonas existe se pelo menos um nó de uma zona está fisicamente conectado a um nó de outra zona. A topologia de nível da zona informa como as zonas estão conectadas. Existem dois tipos de pacotes de *link state* (LSP – *Link State Packet*), que são o "*node* LSP" e "*zone* LSP". O *node* LSP de um nó contém a informação dos seus nós vizinhos e é propagado dentro da sua zona, enquanto que o *zone* LSP contém a informação da zona e é propagado globalmente. Desta forma cada nó possui um conhecimento total sobre a conectividade dos nós da sua zona e somente a informação da conectividade zonas do resto da rede. Então dado a identificação da zona e do nó de um destino, o pacote é roteado baseado na identificação da zona até que este chegue na zona correta, quando então é roteado baseado na identificação do nó. O par <id. zona, id. nó> do destino é suficiente para o roteamento, portanto é adaptável para mudanças de topologia.

8.3.
Sob demanda

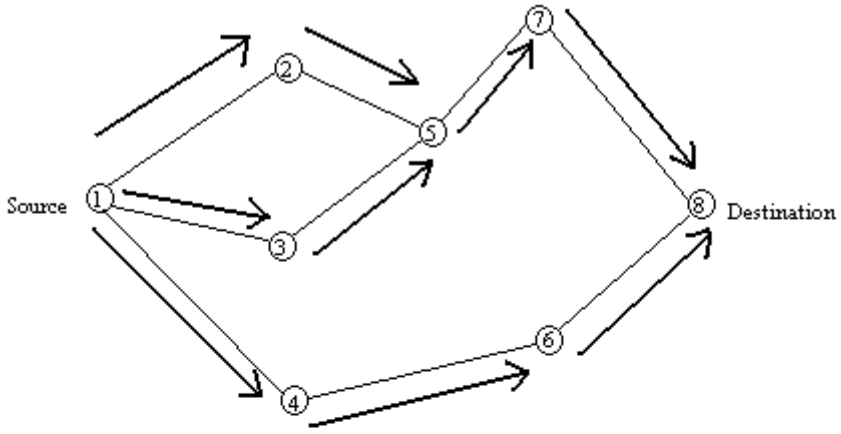
Uma abordagem diferente do roteamento orientado a tabela é o roteamento sob demanda iniciado pelo nó origem. Este tipo de roteamento só cria rotas quando estas são solicitadas pelo nó origem. Quando um nó requer uma rota para um destino, este inicia um processo de descoberta de rota dentro da rede. Este processo é completado se uma rota é encontrada ou se todas as possíveis permutações de rota tenham sido examinadas. Uma vez que uma rota tenha sido estabelecida, esta é mantida por um procedimento de manutenção de rota, até que

o destino se torne inalcançável através de todos os caminhos a partir da origem, ou até que a rota não seja mais desejada. Esta seção discute alguns dos protocolos de roteamento sob demanda propostos para redes móveis ad hoc.

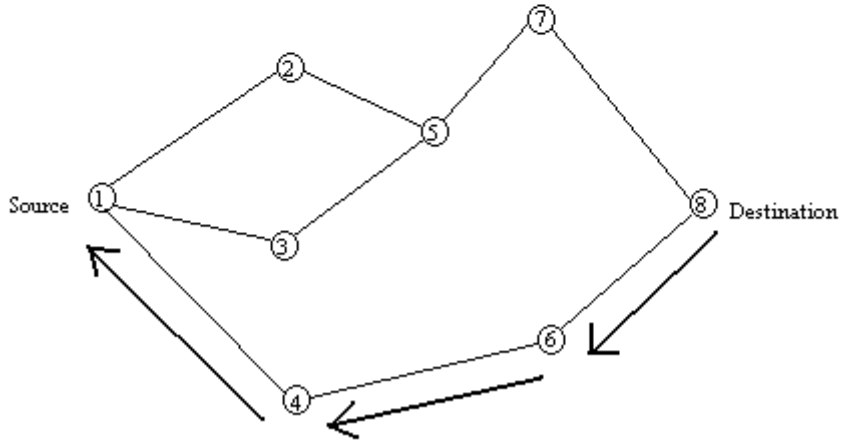
8.3.1. Ad Hoc On-Demand Distance Vector Routing (AODV)

O *Ad Hoc On-Demand Distance Vector Routing* (AODV) (Perkins et al, 1999) é um aperfeiçoamento do protocolo DSDV descrito na seção 8.2.1, sendo que o DSDV é orientado a tabela. De forma geral o AODV tenta minimizar o número de difusões (broadcast) criando rotas sob demanda, o oposto do DSDV que mantém a lista de todas as rotas, o que é o maior problema do DSDV limitando a sua escalabilidade. Outro ponto principal do AODV é tentar minimizar a latência, quando novas rotas são requisitadas.

Para encontrar o caminho para o destino, a origem difunde um pacote de requisição de rota. Os vizinhos por sua vez também difundem o pacote para seus vizinhos até que o pacote chegue em um nó intermediário que tenha uma informação recente da rota para o destino, ou até que chegue no próprio destino (Figura 16a). Um nó descarta um pacote de requisição de rota que já tenha processado. O pacote de requisição de rota usa números seqüenciais para evitar que as rotas tenham *loops* e para garantir que os nós intermediários respondam a requisições de rotas, sempre com a informação mais recente.



(a) Propogation of Route Request (RREQ) Packet



(b) Path taken by the Route Reply (RREP) Packet

Figura 16 – Descobrimento de rota no AODV

Quando um nó encaminha um pacote de requisição de rota (RREQ) para seus vizinhos, este grava em suas tabelas o nó do qual a primeira cópia da requisição veio. Esta informação é usada para construir o caminho reverso para o pacote de resposta (RREP). O AODV usa somente enlaces bidirecionais porque o pacote de resposta da rota segue o caminho reverso do pacote de requisição de rota. Ao encaminhar o pacote de resposta da rota de volta para a origem (Figura 16b), os nós ao longo do caminho acrescentam a informação da rota de encaminhamento guardado por ocasião do recebimento de RREQ.

Se a origem se move então esta pode reiniciar o processo de descobrimento de rota para o destino. Se um dos nós intermediários mover então os vizinhos deste nó detectam a falha no enlace e enviam uma notificação de falha no enlace

para seus vizinhos no caminho reverso e assim por diante até que esta chegue na origem, que pode reiniciar o descobrimento de rota se necessário.

8.3.2. Dynamic Source Routing (DSR)

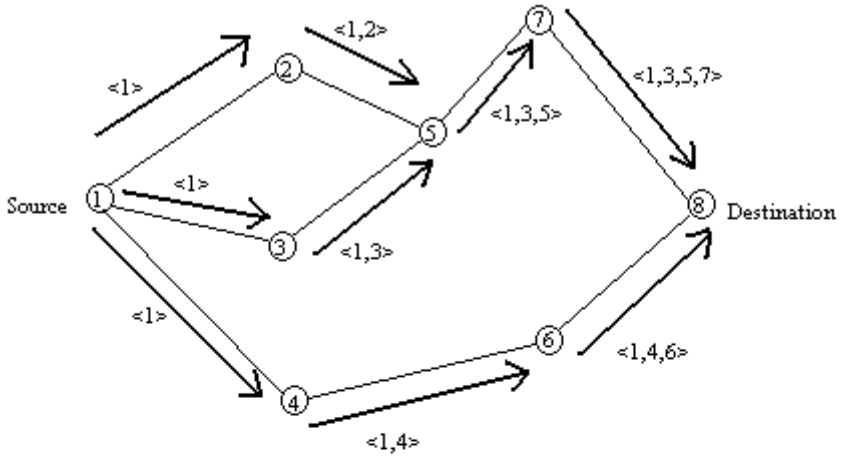
O *Dynamic Source Routing* (DSR) (Johnson & Maltz, 1999) é um protocolo de roteamento do tipo *source routing* sob demanda. Um nó mantém caches de rota contendo as rotas a partir da origem que o nó necessita. As atualizações no cache de rotas são feitas quando o nó descobre novas rotas.

As duas principais fases do protocolo são: descobrimento de rotas e manutenção de rotas. Quando o nó origem deseja enviar um pacote para um destino, este procura no seu cache de rotas para determinar se ele já possui uma rota para o destino. Se o nó origem descobre que possui uma rota para o destino e que o tempo de validade da entrada no cache ainda não expirou, então este usa esta rota para enviar o pacote. Mas se o nó não tem esta rota, ou se a validade no cache já expirou, então o nó origem inicia o processo de descobrimento de rotas, fazendo a difusão (broadcasting) de um pacote de requisição de rota. Este pacote contém o endereço da origem e do destino, e número de identificação único. Cada nó intermediário verifica se o mesmo conhece uma rota para o destino. Se não conhecer, o nó acrescenta seu endereço no registro de rota do pacote e o encaminha para seus vizinhos. Para limitar o número de requisições de rota propagando na rede, um nó só processa o pacote de requisição de rota se o pacote ainda não tiver passado por ele e seu endereço não estiver presente no registro de rota do pacote.

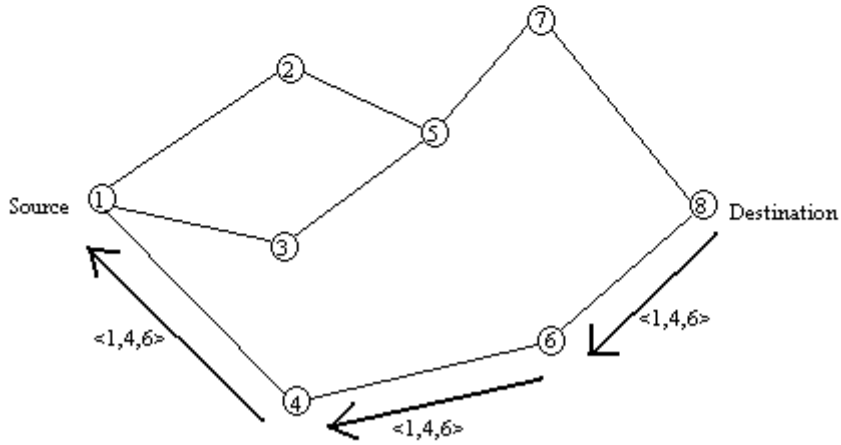
Uma resposta da rota é gerada quando o pacote de requisição de rota é recebido pelo destino ou por um nó intermediário que possua uma rota para o destino (Johnson & Maltz, 1996). Ao chegar em tal nó, um pacote de requisição de rota já contém no seu registro de rota a seqüência de *hops* que leva da origem até este nó.

A medida em que o pacote de requisição de rota se propaga através da rede, vai se formando o registro de rota, como é mostrado na Figura 17a. Se a resposta da rota é gerada pelo destino então este coloca o registro de rota que veio no

pacote de requisição de rota dentro do pacote de resposta da rota. Por outro lado, se o nó que gera a resposta da rota for um nó intermediário então ele acrescenta a rota para o destino que está no seu cache com o registro de rota do pacote de requisição de rota, e coloca tudo dentro do pacote de resposta da rota. Na Figura 17b é mostrado o pacote de resposta da rota sendo enviado pelo próprio destino. Para enviar o pacote de resposta da rota, o nó que responde deve ter uma rota para a origem, se o nó tiver esta rota no seu cache de rotas, esta pode ser usada. Caso contrário pode usar a rota reversa da que está no registro de rotas, mas isto só pode ser feito se os enlaces forem bidirecionais. Caso os enlaces sejam unidirecionais, o nó que responde pode iniciar um processo de descobrimento de rota para a origem, e enviar por "carona" a resposta da rota nesta nova requisição de rota.



(a) Building Record Route during Route Discovery



(b) Propogation of Route Reply with the Route Record

Figura 17 – Criação do registro de rota no DSR

O DSR usa dois tipos de pacotes para manutenção de rotas: pacote de erro de rota e *Acknowledgements* (pacote de confirmação). Quando um nó encontra um problema de transmissão fatal na sua camada de enlace de dados, o nó então gera um pacote de erro de rota. Quando um nó recebe um pacote de erro de rota, este remove o *hop* em erro do seu cache de rotas, e todas as rotas que contém o *hop* em erro são podadas neste ponto. Pacotes de *Acknowledgment* são usados para verificar o funcionamento correto dos enlaces da rota. Isto também inclui *acknowledgments* nos quais um nó ouve (modo promíscuo) o próximo *hop* encaminhando o pacote através da rota.

8.3.3.
Cluster Based Routing Protocol (CBRP)

No *Cluster Based Routing Protocol* (CBRP) (Jiang et al, 1999), os nós são divididos em clusters. Para formar um cluster o seguinte algoritmo é usado: quando um novo nó surge, ele começa no estado "*undecided*", inicia um timer e difunde uma mensagem *Hello*; quando um *cluster-head* recebe esta mensagem *Hello*, ele envia imediatamente uma mensagem *Hello* de resposta; que, quando recebida pelo novo nó causa a sua mudança de estado para "*member*". Se estourar o tempo do nó indeciso sem que este receba uma resposta, então este nó torna-se um *cluster-head* se ele tiver um enlace bidirecional para algum vizinho. Caso contrário o nó permanece no seu estado indeciso e repete o procedimento novamente. As mudanças de *cluster-heads* são raras o quanto for possível.

Cada nó mantém uma tabela de vizinhos, onde para cada vizinho são armazenados o estado do enlace (uni ou bidirecional) e o estado do vizinho (*cluster-head* ou *member*). Um *cluster-head* mantém informação sobre os membros do seu cluster e também mantém uma tabela de clusters adjacentes que contém informação sobre os clusters vizinhos. Para cada cluster vizinho, a tabela tem a informação de quem é o seu *cluster-head* e do *gateway* através do qual o cluster pode ser alcançado.

Quando um nó origem tem que enviar dados para um destino, este nó inunda com pacotes de requisição de rota, os *cluster-heads* vizinhos. Quando um *cluster-head* recebe a requisição, este verifica se o destino é do seu cluster. Caso seja então o *cluster-head* envia a requisição diretamente para o destino, senão ele envia a requisição para todos os seus *cluster-heads* adjacentes. O endereço do *cluster-head* é gravado no pacote de modo que um *cluster-head* descarte um pacote de requisição que já tenha visto. Quando o nó destino recebe o pacote de requisição, este responde de volta com a rota que foi gravada no pacote de requisição. Se o nó origem não receber uma resposta dentro de um período de tempo, então ele inicia uma espera de período aleatória e exponencial, ou seja, aguarda por um tempo aleatório até um limite antes de tentar enviar uma requisição de rota novamente. Caso sejam necessárias mais retransmissões este limite vai sendo dobrado.

O CBRP é um algoritmo do tipo *source routing*. Além disto utiliza também uma técnica chamada *route shortening*, que é durante o envio de um pacote com a rota definida na origem, um nó ao receber o pacote tenta encontrar o nó mais distante na rota que seja seu vizinho (que pode acontecer devido a uma mudança na topologia) e enviar o pacote diretamente para este nó, reduzindo desta forma a rota. Enquanto encaminha um pacote, se um nó detecta a perda de um enlace, o nó então envia de volta uma mensagem de erro para a origem e usa um mecanismo de reparo local. Este mecanismo funciona da seguinte forma: quando um nó detecta que o próximo *hop* não está ao seu alcance, o nó verifica se o próximo *hop*, ou o *hop* após o próximo *hop*, pode ser alcançado através de qualquer um dos seus vizinhos; se qualquer um dos dois casos funcionar, o pacote pode ser enviado pelo caminho alternativo.

8.3.4. Temporally-Ordered Routing Algorithm (TORA)

O *Temporally-Ordered Routing Algorithm* (TORA) é um algoritmo de roteamento distribuído altamente adaptável, sem *loops*, baseado no conceito de reversão do enlace (*link reversal*) (Park & Corson, 1997). TORA é proposto para redes sem fio *multihop*, com mobilidade altamente dinâmica, sendo um protocolo de roteamento sob demanda iniciado pela origem (*source routing*). TORA é capaz de encontrar múltiplas rotas de um nó origem até um nó destino. A característica principal deste protocolo é que as mensagens de controle ficam restritas a um conjunto muito pequeno de nós próximos da ocorrência de uma mudança topológica. Para isto, os nós mantêm informação de roteamento sobre os nós adjacentes. O protocolo tem três funções básicas: criação de rotas, manutenção de rotas, e deleção de rotas.

A métrica do protocolo TORA é uma quíntupla englobando os seguintes elementos:

- Tempo lógico da falha de um enlace
- O identificador único do nó que definiu o novo nível de referência
- Bit indicador de reflexão
- Parâmetro de propagação de ordem
- O identificador único do nó

Os três primeiros elementos juntos representam o nível de referência. Um novo nível de referência é definido cada vez que um nó perde seu último enlace *downstream* devido a uma falha no enlace. Os últimos dois valores definem um delta com respeito ao nível de referência (Park & Corson, 1997).

A criação de rotas é feita usando pacotes QRY e UPD. O algoritmo de criação de rota inicia com a altura (parâmetro de propagação de ordem na quíntupla) do destino com o valor 0 e todas as alturas dos outros nós com o valor NULL (nulo, i.e. indefinido). A origem difunde um pacote QRY com o identificador do nó destino dentro dele. Um nó com uma altura não nula (i.e. definida) responde com um pacote UPD tendo sua altura dentro dele. Um nó recebendo um pacote UPD ajusta a sua altura para uma a mais do que a altura do nó que gerou o UPD. Um nó com uma altura maior é considerado *upstream* e um nó com altura menor é *downstream*. Desta forma um grafo acíclico dirigido (DAG – Directed Acyclic Graph) é construído da origem até o destino. A Figura 18 ilustra um processo de criação de rota no TORA. Como mostrado na Figura 18a, o nó 5 não propaga o QRY do nó 3 pois já foi processada e propagada uma mensagem QRY do nó 2. Na Figura 18b, a origem (i.e. nó 1) pode ter recebido UPD de vários nós (i.e. nós 2, 3 e 4), mas uma vez que o nó 4 dá a menor altura, a origem seleciona.

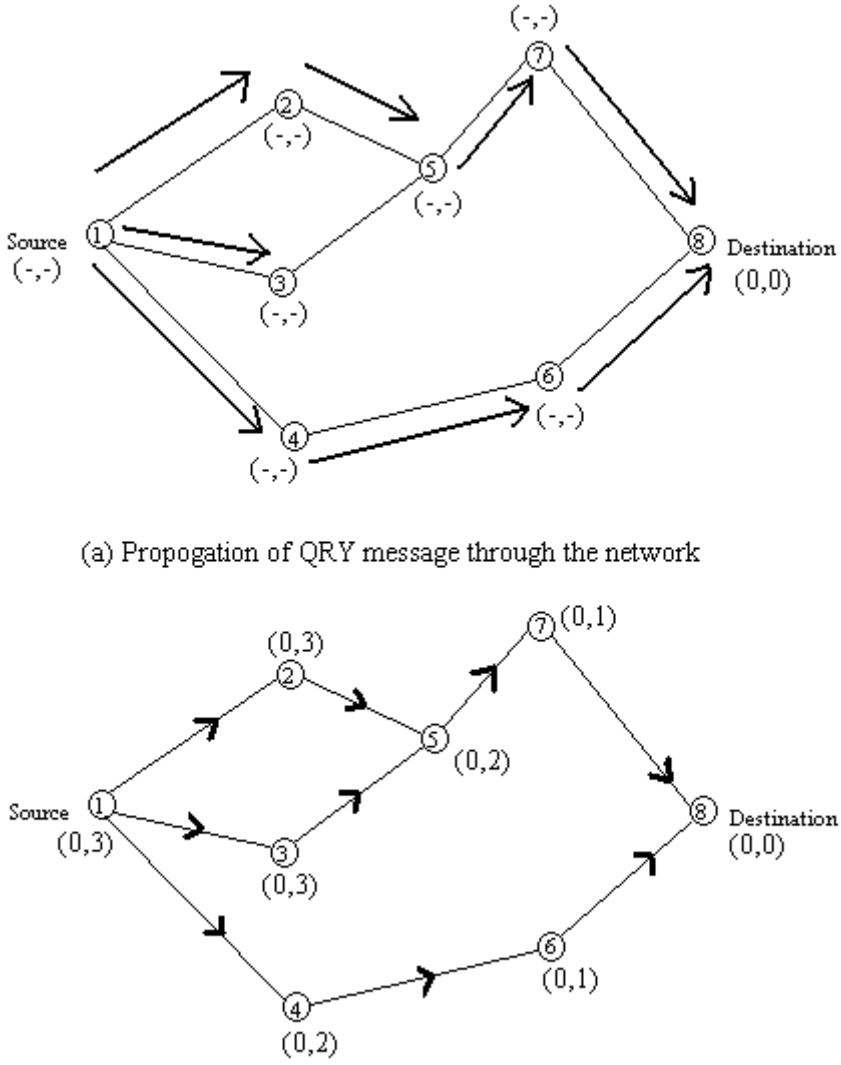


Figura 18 – Criação de rota no TORA (os números entre parênteses são: nível de referência e altura)

Quando um nó se move a rota do DAG é quebrada, e uma manutenção de rota é necessária para restabelecer um DAG para o mesmo destino. Quando o último enlace de *downstream* de um nó falha, este nó gera um novo nível de referência. Isto resulta na propagação deste nível de referência pelos nós vizinhos como mostrado na Figura 19. Os enlaces são revertidos para refletirem a mudança na adaptação para o novo nível de referência. Isto tem o mesmo efeito como se revertesse a direção de um ou mais enlaces quando um nó não tem enlaces de *downstream*.

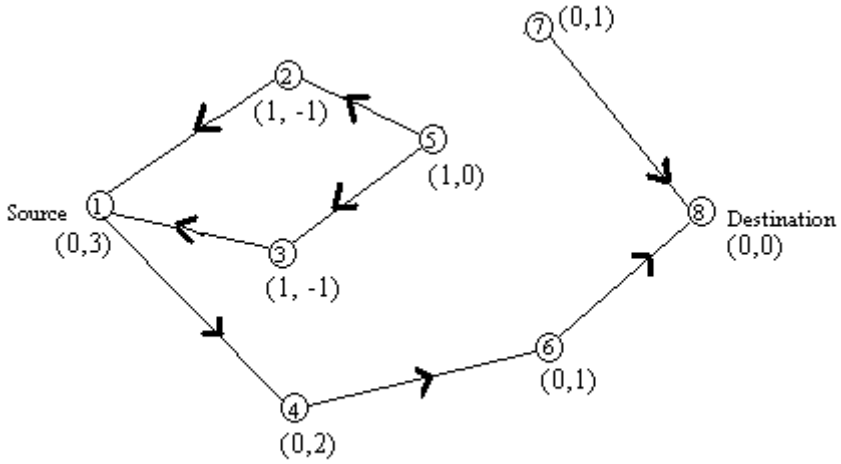


Figura 19 – Restabelecimento da rota devido uma falha no enlace 5-7. O novo nível de referência é o nó 5

Na fase de deleção de rotas, TORA inunda (*flooding*) a rede com pacote CLR (*clear packet*) para apagar rotas inválidas.

No TORA existe um potencial para ocorrer oscilações, especialmente quando múltiplos conjuntos de nós coordenadores estão concorrentemente detectando partições, deletando rotas, e criando novas rotas baseados um nos outros. Devido TORA usar coordenação internodal, seu problema de instabilidade é similar ao problema da "contagem até o infinito" do algoritmo de roteamento com Vetor de Distância (Tanenbaum, 1997), exceto que tais oscilações são temporárias e a convergência da rota irá certamente ocorrer.

O tempo é um fator importante para o protocolo TORA, porque a métrica da altura é dependente do tempo lógico da falha de um enlace (um dos elementos da quíntupla). TORA assume que todos os nós têm seus relógios sincronizados. Isto pode ser conseguido através de uma fonte externa de tempo, tal como GPS (*Global Positioning System*).