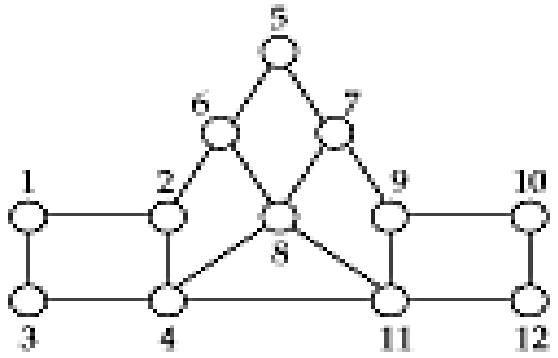
Routing Protocols

Traditional Routing

- A *routing protocol* sets up a *routing table* in *routers*



ROUTING TABLE AT 1

Destination	Next hop	Destination	Next hop
1	—	7	2
2	2□	8□	2□
3	3□	9□	2□
4	3□	10□	2□
5	2□	11□	3□
6	2	12	3

- A node makes a *local* choice depending on *global* topology

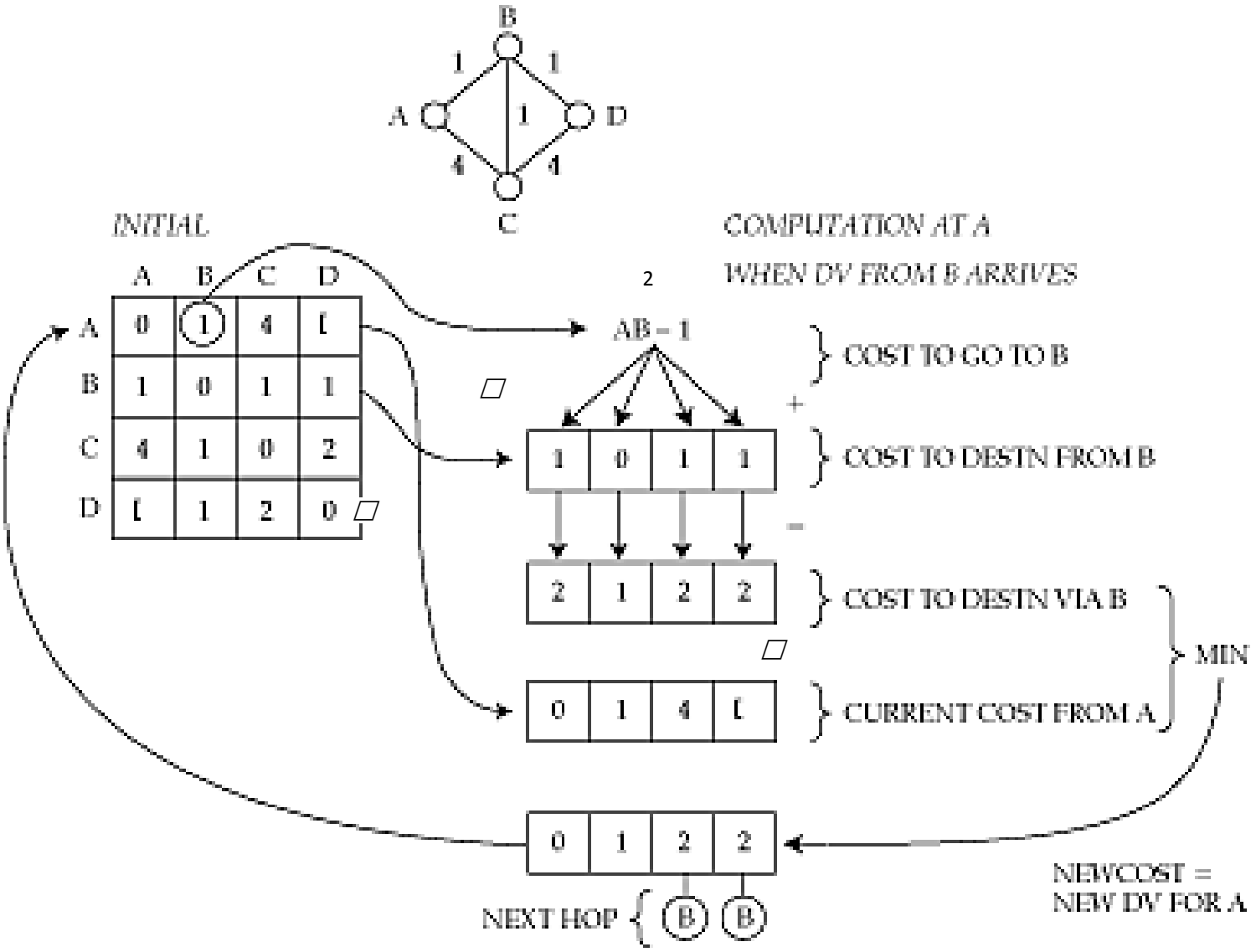
Distance-vector & Link-state Routing

- Both assume router knows
  - address of each neighbor
  - cost of reaching each neighbor
- Both allow a router to determine global routing information by talking to its neighbors

- Distance vector - router knows cost to each destination

- Link state - router knows entire network topology and computes shortest path

Distance Vector Routing: Example



Background

- LinkState Routing
  - Utilizado para obter os melhores caminhos de roteamento em uma rede
  - Realizado por todos os nós
  - Principal objetivo é que todos os nós possuam um mapa da conectividade da rede
  - Informação sobre as conexões entre nós é inundada através da rede

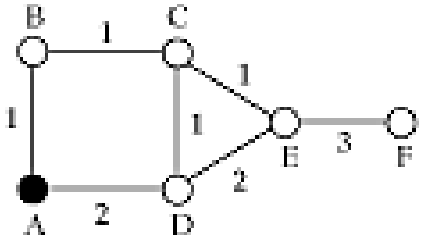
Background

- LinkState Routing
  - Determinar os vizinhos de cada nó
  - Distribuir a informaçao para a obtenção do mapa
    - Link-state advertisement:
      - Identifica o nó que a produziu
      - Identifica todos os nós àquele que é diretamente conectado
      - Inclui um número de sequência, que aumenta toda vez que uma mensagem nova é enviada
  - Criar o mapa

Background

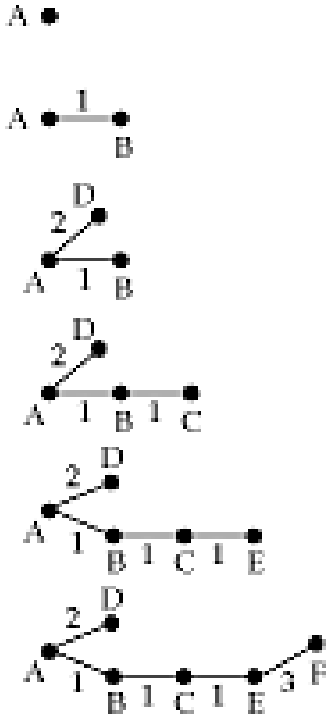
- LinkState Routing
  - Já com o mapa, calcula-se o menor caminho entre todos os nós
    - Geralmente usando alguma variação do algoritmo de Dijkstra
  - Obtém-se assim a tabela de roteamento
  - Exemplos:
    - IS-IS
    - OSPF
    - OLSR

Link State Routing: Example



B(A,1) means B was reached by A, cost 1

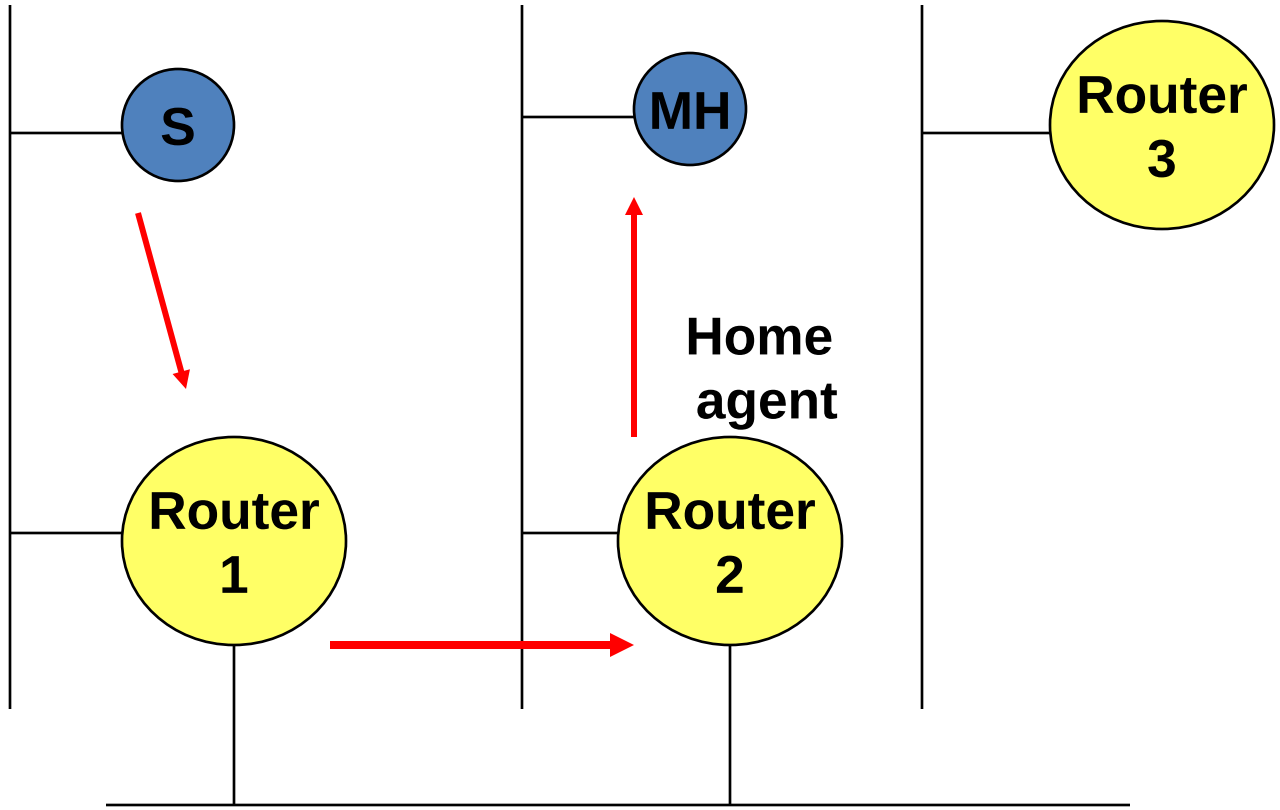
PERMANENT	TEMPORARY	COMMENTS
A	B(A,1), D(A,2)	ROOT AND ITS NEIGHBORS
A, B(A,1)	D(A,2), C(B,2)	ADD C(B,2)
A, B(A,1) D(A,2)	E(D,4), C(B,2)	C(D,3) DIDN'T MAKE IT
A, B(A,1) D(A,2), C(B,2)	E(C,3)	E(D,4) TOO LONG
A, B(A,1) D(A,2), C(B,2) E(C,3)	F(E,6)	
A, B(A,1) C(B,2), D(A,2) E(C,3), F(E,6)	NULL	STOP



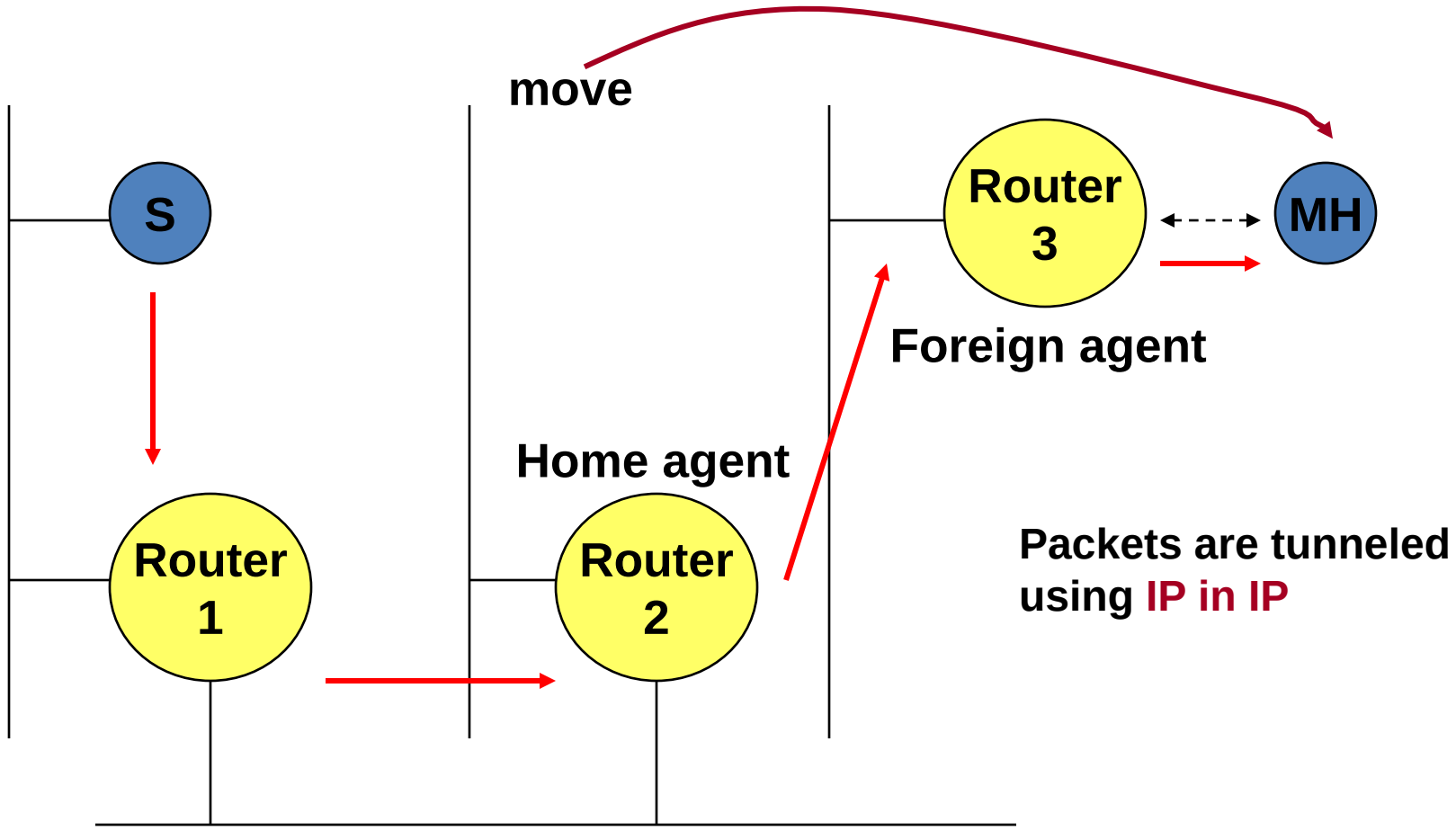
Routing and Mobility

- Finding a path from a source to a destination
- Issues
  - Frequent route changes
    - amount of data transferred between route changes may be much smaller than traditional networks
  - Route changes may be related to host movement
  - Low bandwidth links
- Goal of routing protocols
  - decrease routing-related overhead
  - find short routes
  - find “stable” routes (despite mobility)

Mobile IP



Mobile IP


move
S
Router 3
MH
Foreign agent
Home agent
Router 1
Router 2
Packets are tunneled using IP in IP

Routing in MANET

Unicast Routing Protocols

- Many protocols have been proposed

- Some specifically invented for MANET
- Others adapted from protocols for wired networks

- No single protocol works well in all environments
  - some attempts made to develop adaptive/hybrid protocols

- Standardization efforts in IETF
  - MANET, MobileIP working groups
  - **http://www.ietf.org**

Routing Protocols

- Proactive protocols
  - Traditional distributed shortest-path protocols
  - Maintain routes between every host pair at all times
  - Based on periodic updates; High routing overhead
  - Example: DSDV (destination sequenced distance vector)
- Reactive protocols
  - Determine route if and when needed
  - Source initiates route discovery
  - Example: DSR (dynamic source routing)
- Hybrid protocols
  - Adaptive; Combination of proactive and reactive
  - Example : ZRP (zone routing protocol)

Protocol Trade-offs

- Proactive protocols
  - Always maintain routes
  - Little or no delay for route determination
  - Consume bandwidth to keep routes up-to-date
  - Maintain routes which may never be used

- Reactive protocols
  - Lower overhead since routes are determined on demand
  - Significant delay in route determination
  - Employ flooding (global search)
  - Control traffic may be bursty

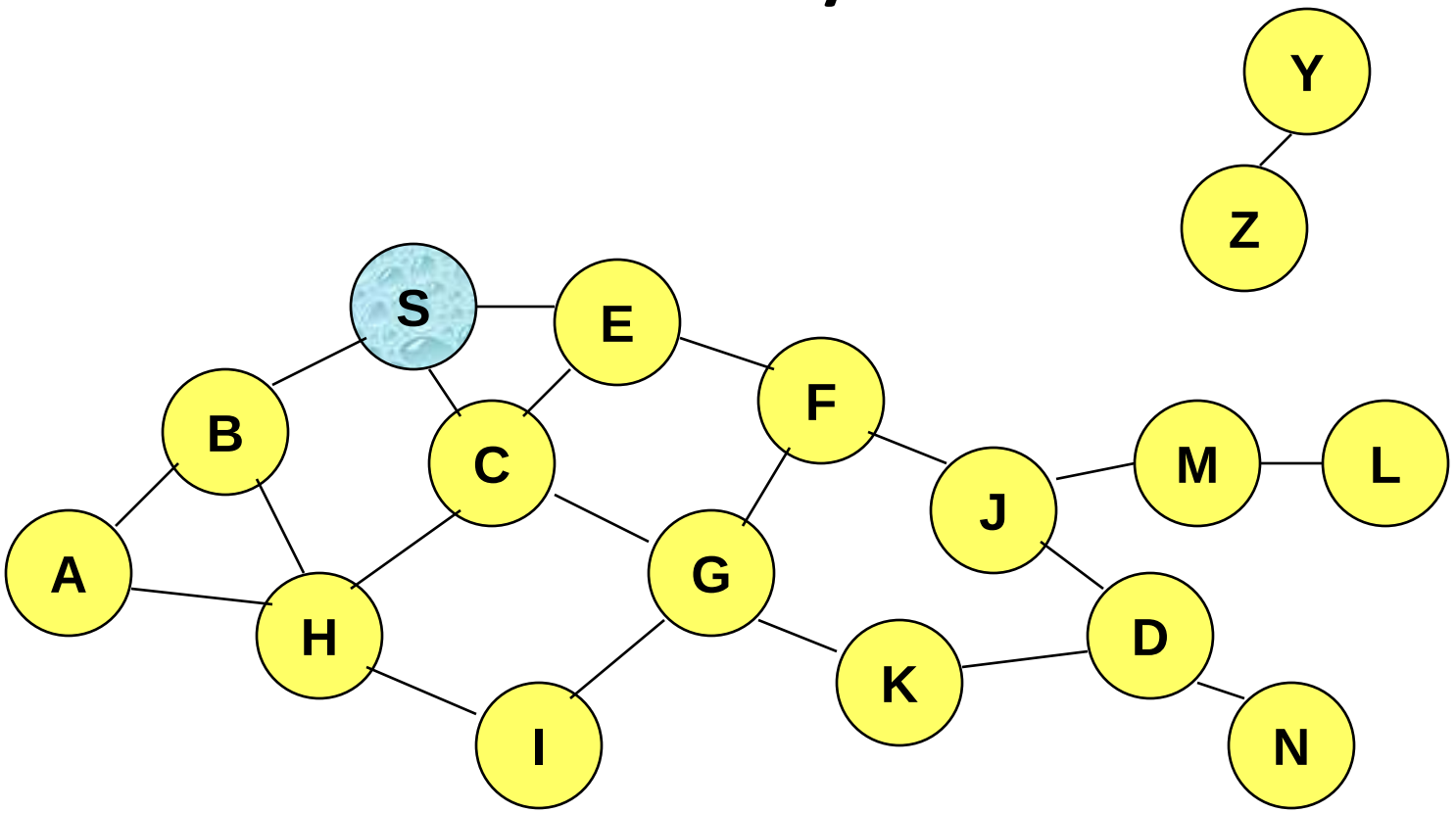
- Which approach achieves a better trade-off depends on the traffic and mobility patterns

Reactive Routing Protocols

Dynamic Source Routing (DSR) [Johnson96]

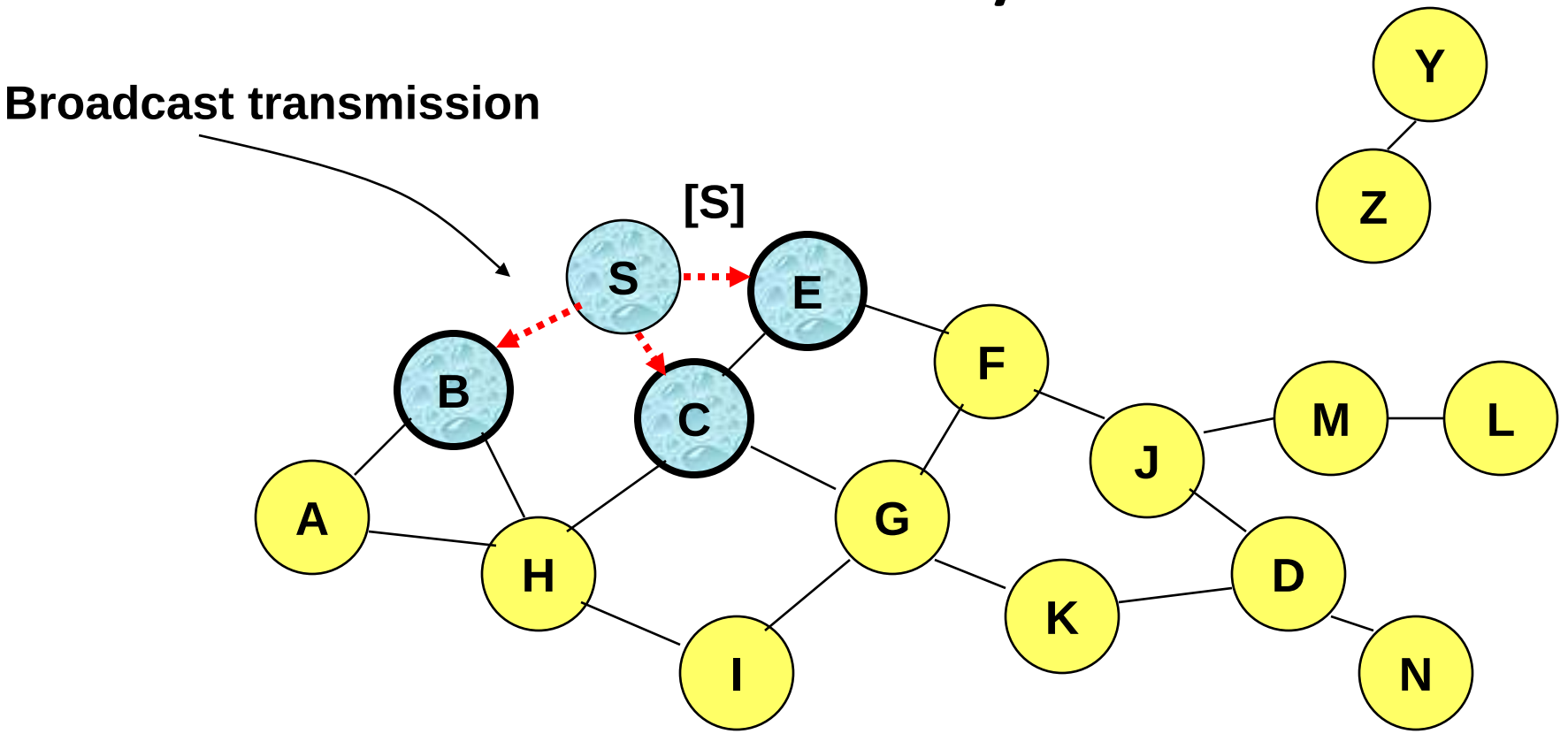
- When node S wants to send a packet to node D, but does not know a route to D, node S initiates a route discovery
- Source node S floods Route Request (RREQ)
- Each node *appends own identifier* when forwarding RREQ

Route Discovery in DSR



Represents a node that has received RREQ for D from S

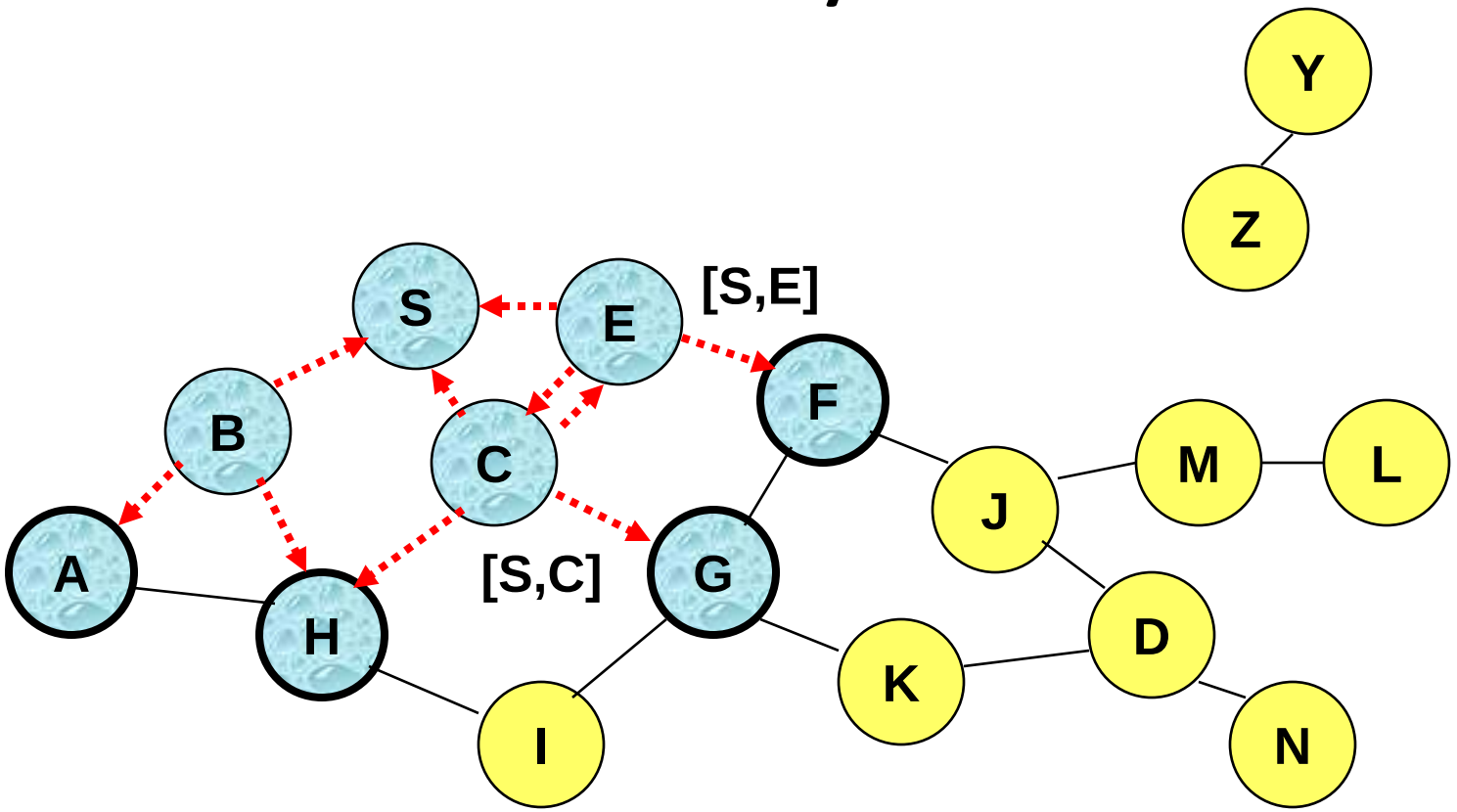
Route Discovery in DSR



Represents transmission of RREQ

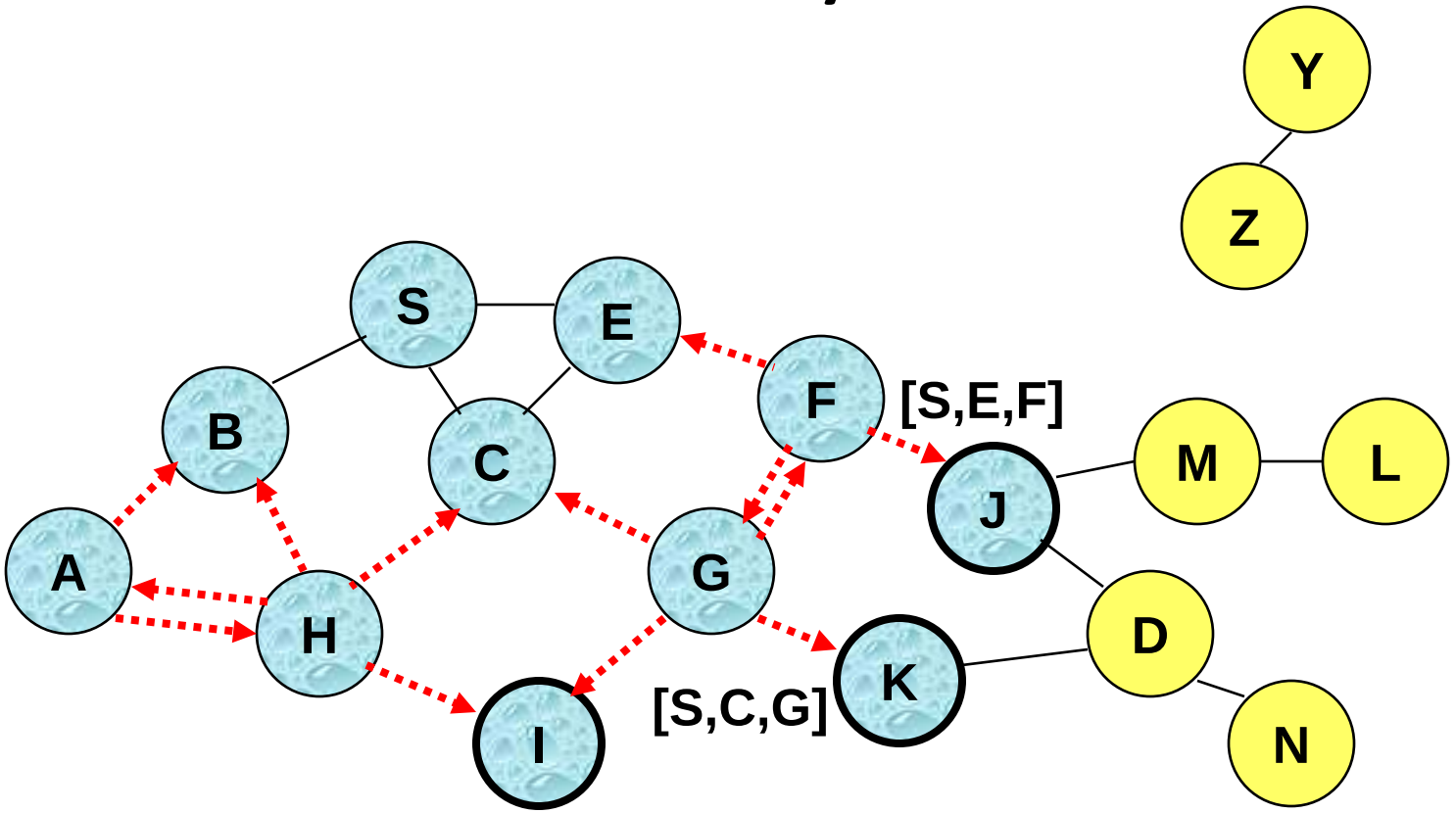
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR



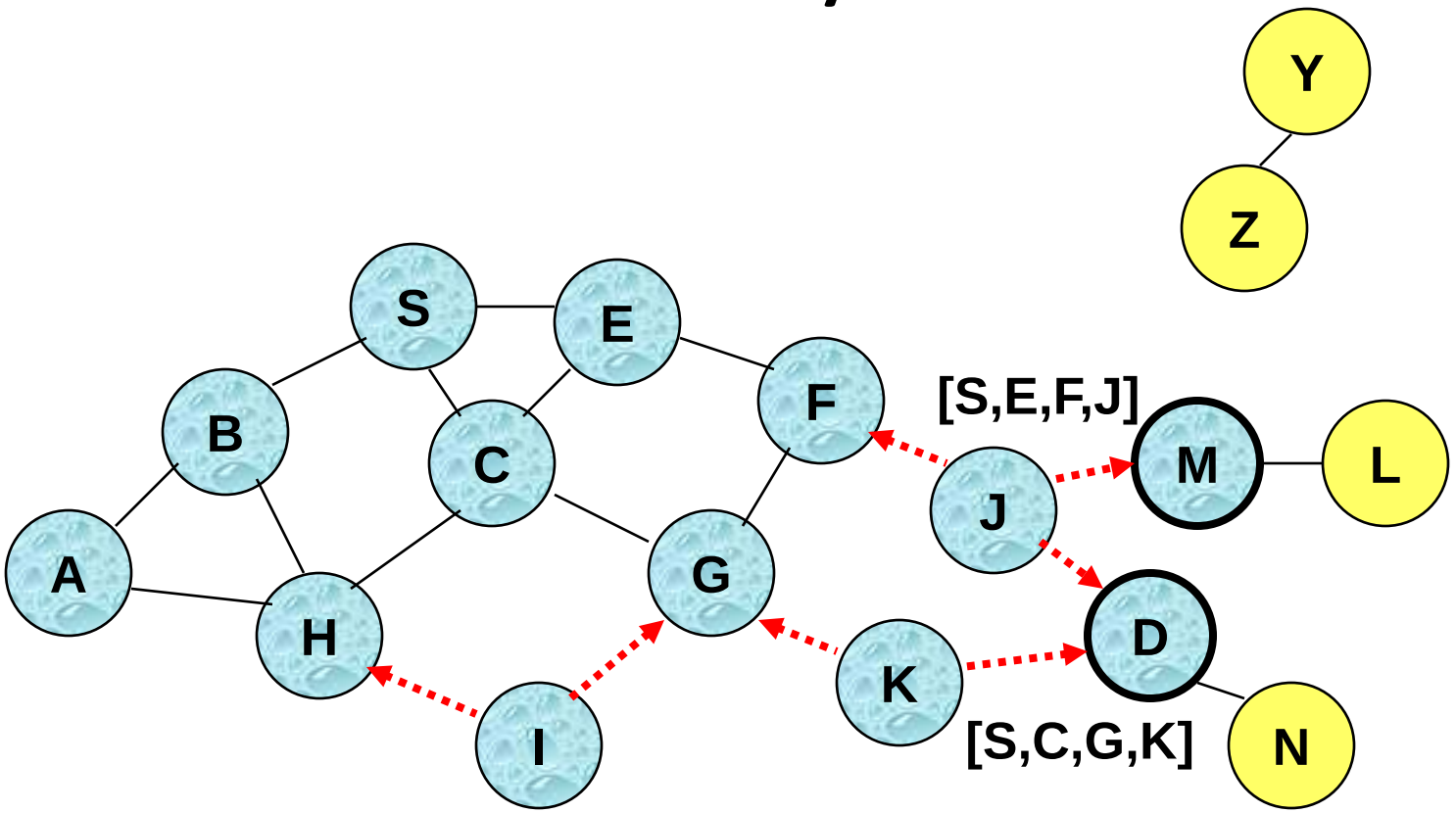
- **Node H receives packet RREQ from two neighbors:**
  potential for collision

Route Discovery in DSR



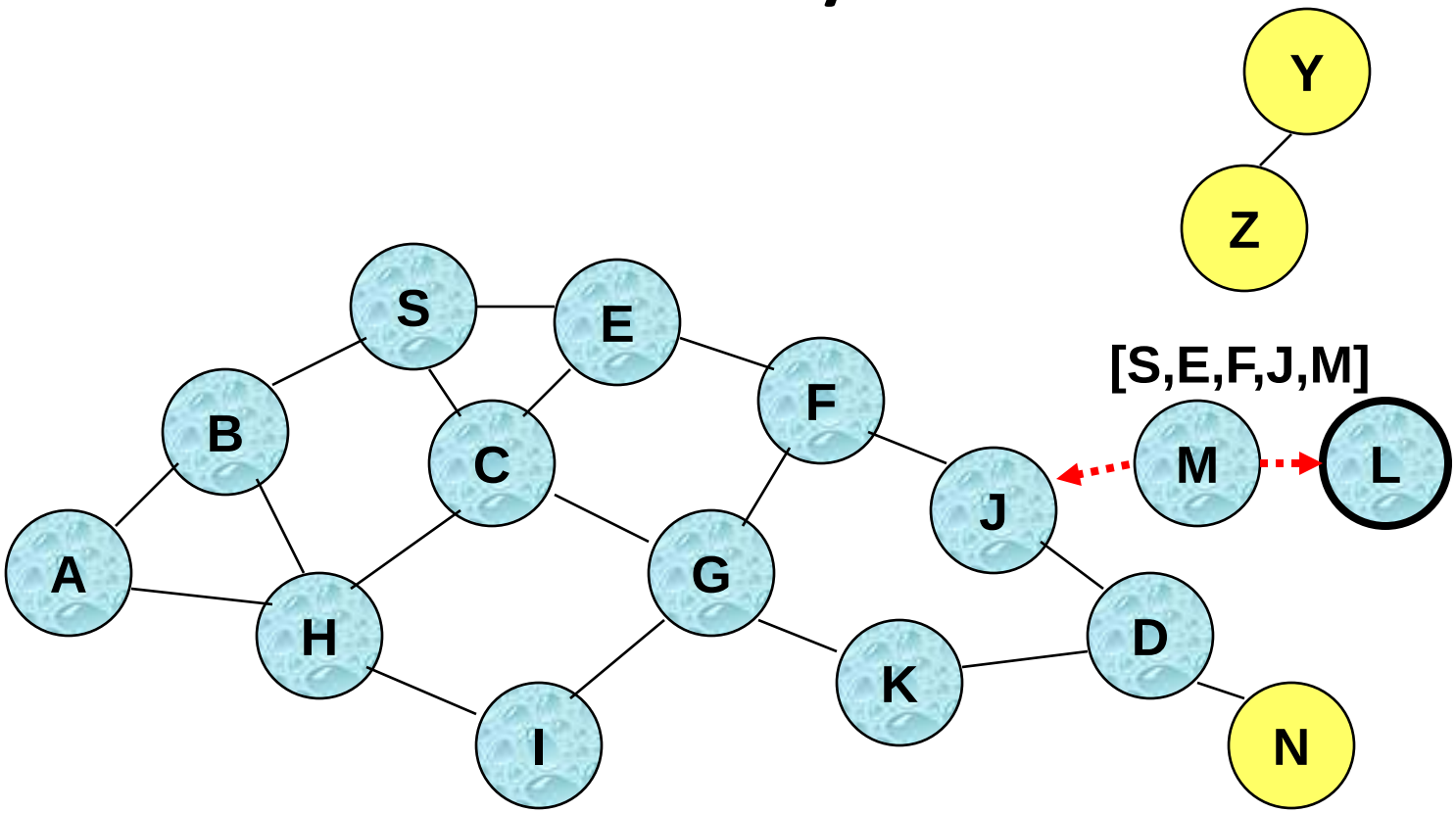
- **Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once**

Route Discovery in DSR



- **Nodes J and K both broadcast RREQ to node D**
- **Since nodes J and K are hidden from each other, their transmissions may collide**

Route Discovery in DSR

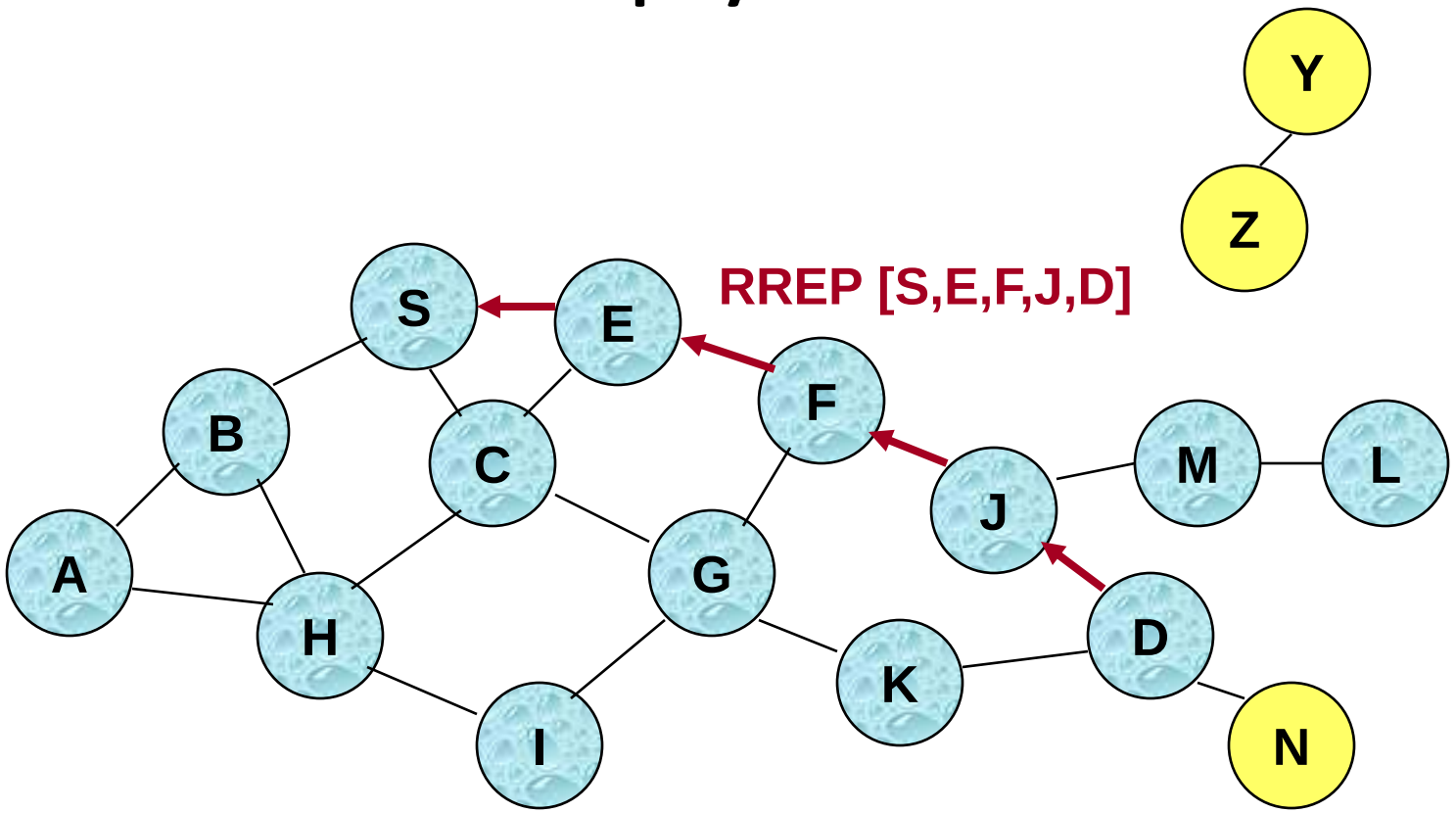


- **Node D does not forward RREQ, because node D is the intended target of the route discovery**

Route Discovery in DSR

- Destination D on receiving the first RREQ, sends a Route Reply (RREP)
- RREP is sent on a route obtained by reversing the route appended to received RREQ
- RREP includes the route from S to D on which RREQ was received by node D

Route Reply in DSR

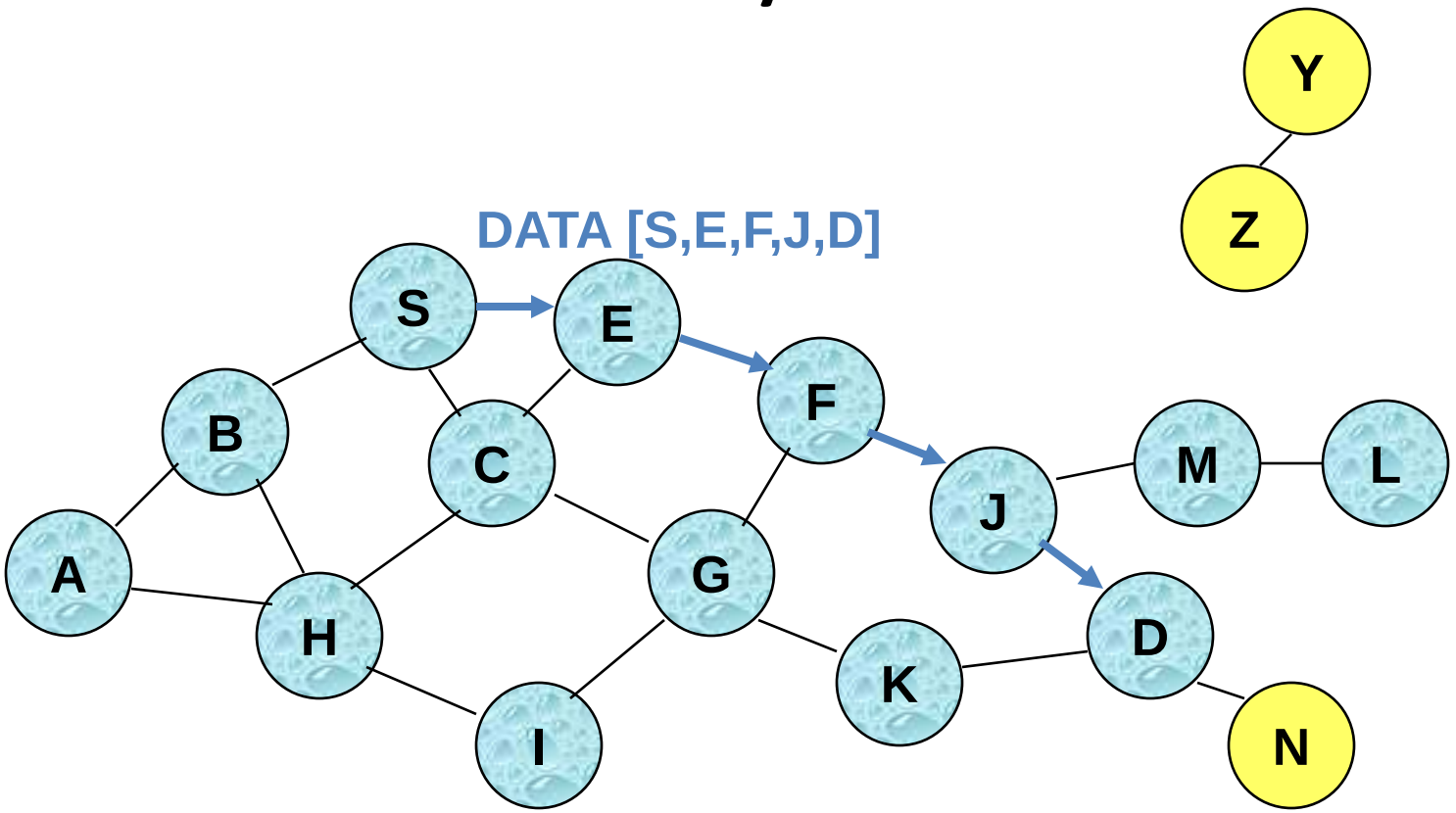


Represents RREP control message

Dynamic Source Routing (DSR)

- Node S on receiving RREP, caches the route included in the RREP
- When node S sends a data packet to D, the entire route is included in the packet header
  - hence the name source routing
- Intermediate nodes use the source route included in a packet to determine to whom a packet should be forwarded

Data Delivery in DSR



Packet header size grows with route length

DSR Optimization: Route Caching

- Each node caches a new route it learns by *any means*
- When node S finds route [S,E,F,J,D] to node D, node S also learns route [S,E,F] to node F
- When node K receives Route Request [S,C,G] destined for node, node K learns route [K,G,C,S] to node S
- When node F forwards Route Reply RREP [S,E,F,J,D], node F learns route [F,J,D] to node D
- When node E forwards Data [S,E,F,J,D] it learns route [E,F,J,D] to node D
- A node may also learn a route when it overhears Data
- Problem: Stale caches may increase overheads

Dynamic Source Routing: Advantages

- Routes maintained only between nodes who need to communicate
  - reduces overhead of route maintenance
- Route caching can further reduce route discovery overhead
- A single route discovery may yield many routes to the destination, due to intermediate nodes replying from local caches

Dynamic Source Routing: Disadvantages

- Packet header size grows with route length due to source routing
- Flood of route requests may potentially reach all nodes in the network
- Potential collisions between route requests propagated by neighboring nodes
  - insertion of random delays before forwarding RREQ
- Increased contention if too many route replies come back due to nodes replying using their local cache
  - Route Reply *Storm* problem
- Stale caches will lead to increased overhead

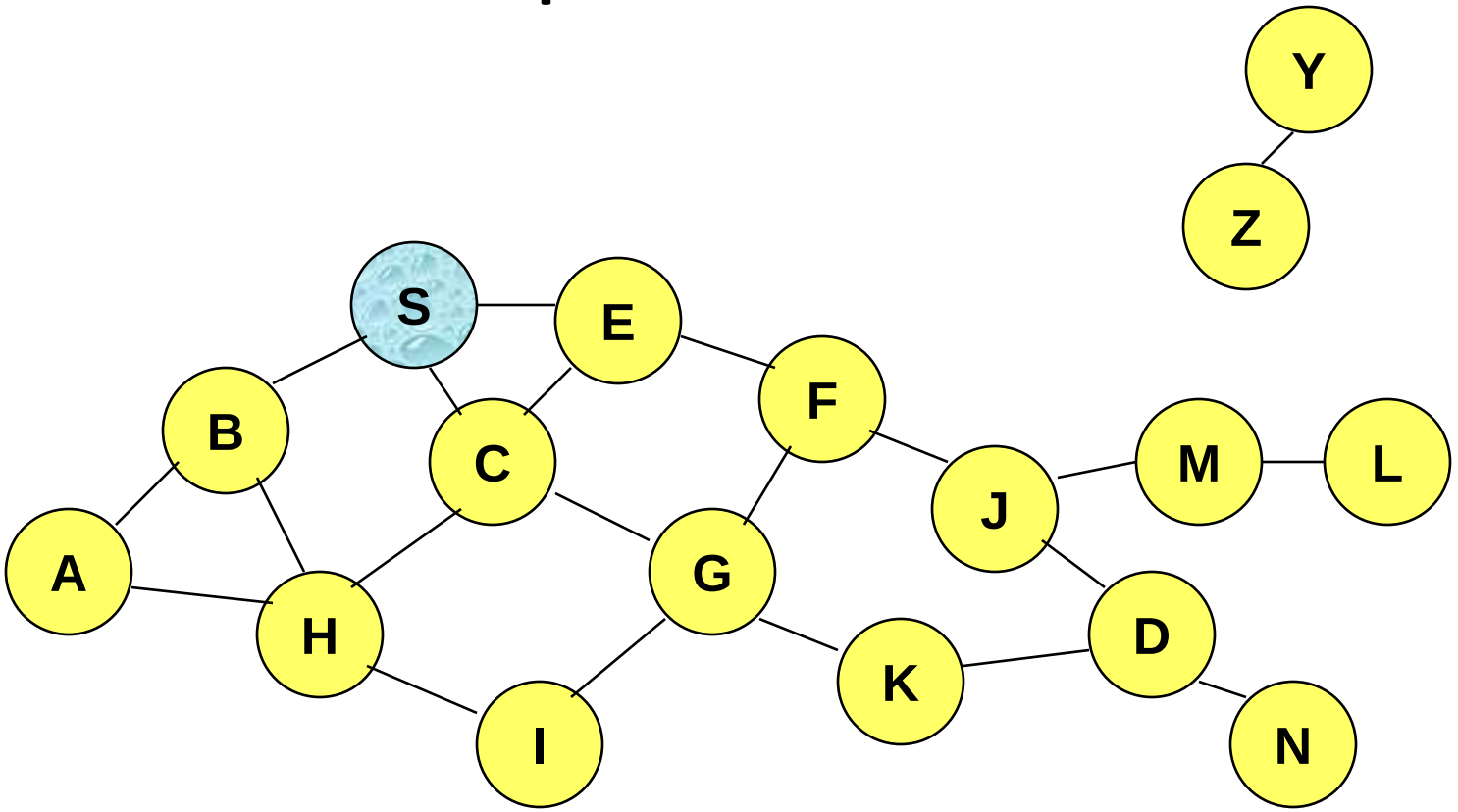
Ad Hoc On-Demand Distance Vector Routing (AODV) [Perkins99Wmcsa]

- DSR includes source routes in packet headers
- Resulting large headers can sometimes degrade performance
  - particularly when data contents of a packet are small
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes
- AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate

AODV

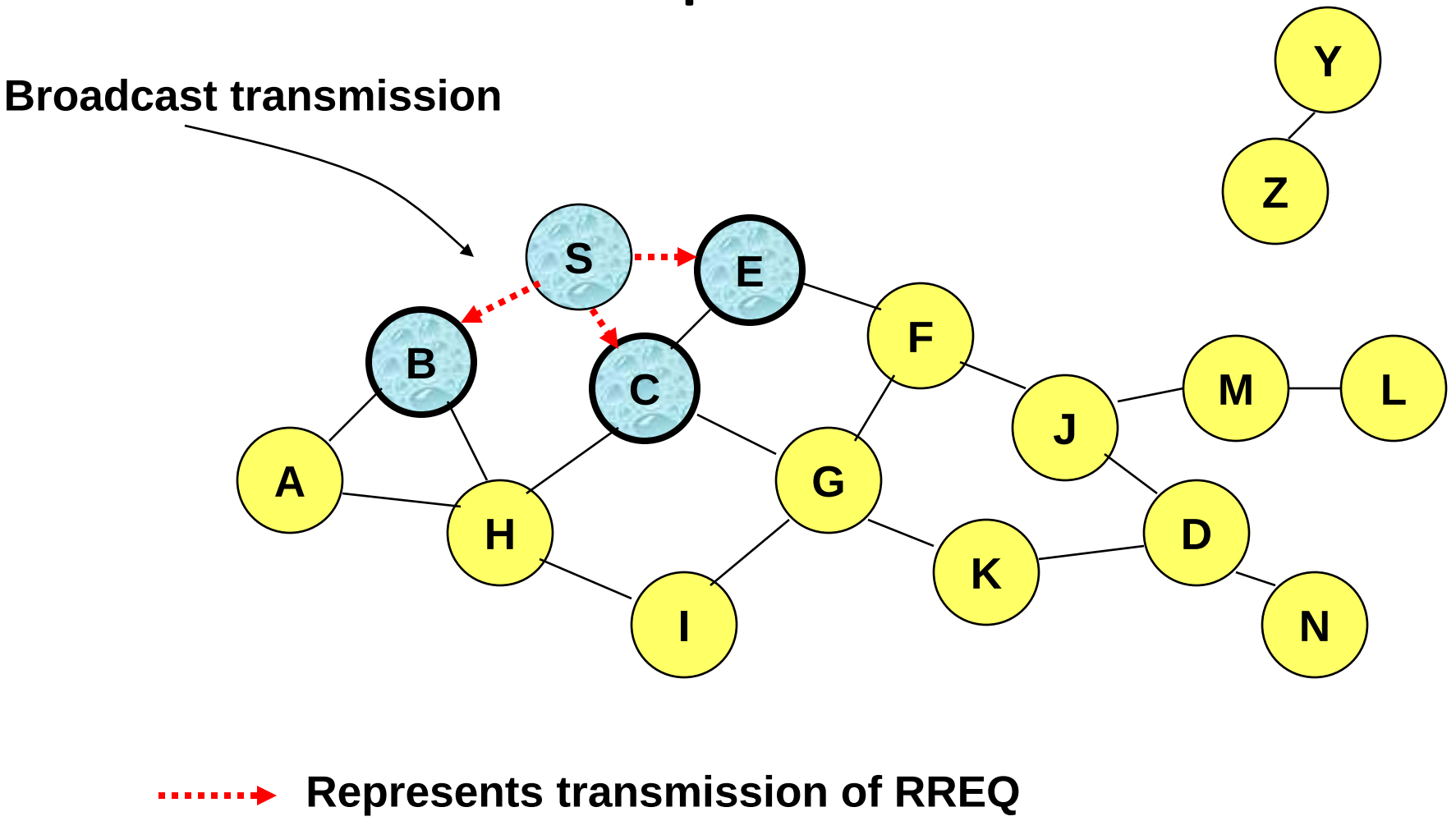
- Route Requests (RREQ) are forwarded in a manner similar to DSR
- When a node re-broadcasts a Route Request, it sets up a reverse path pointing towards the source
  - AODV assumes symmetric (bi-directional) links
- When the intended destination receives a Route Request, it replies by sending a Route Reply (RREP)
- Route Reply travels along the reverse path set-up when Route Request is forwarded

Route Requests in AODV

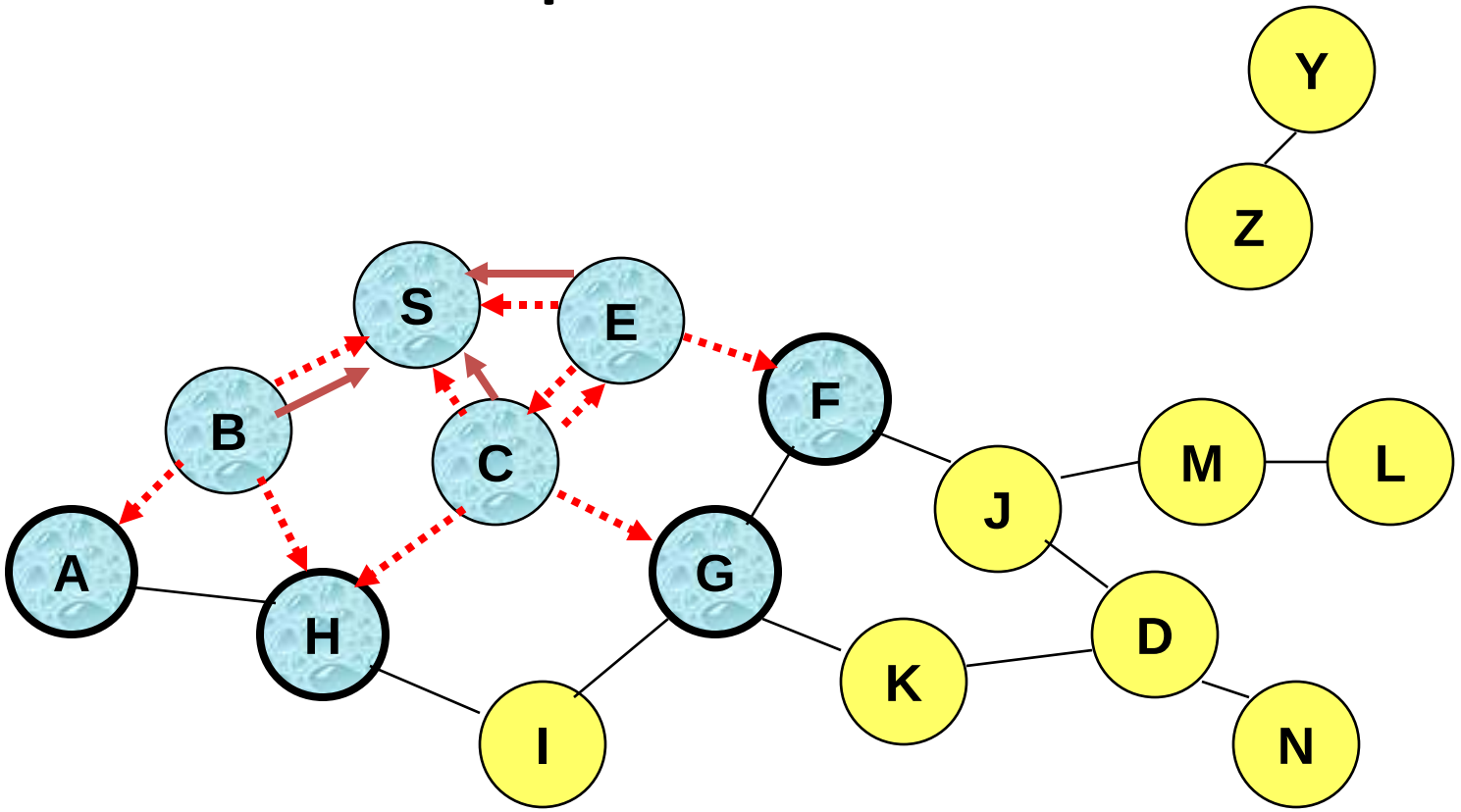


Represents a node that has received RREQ for D from S

Route Requests in AODV

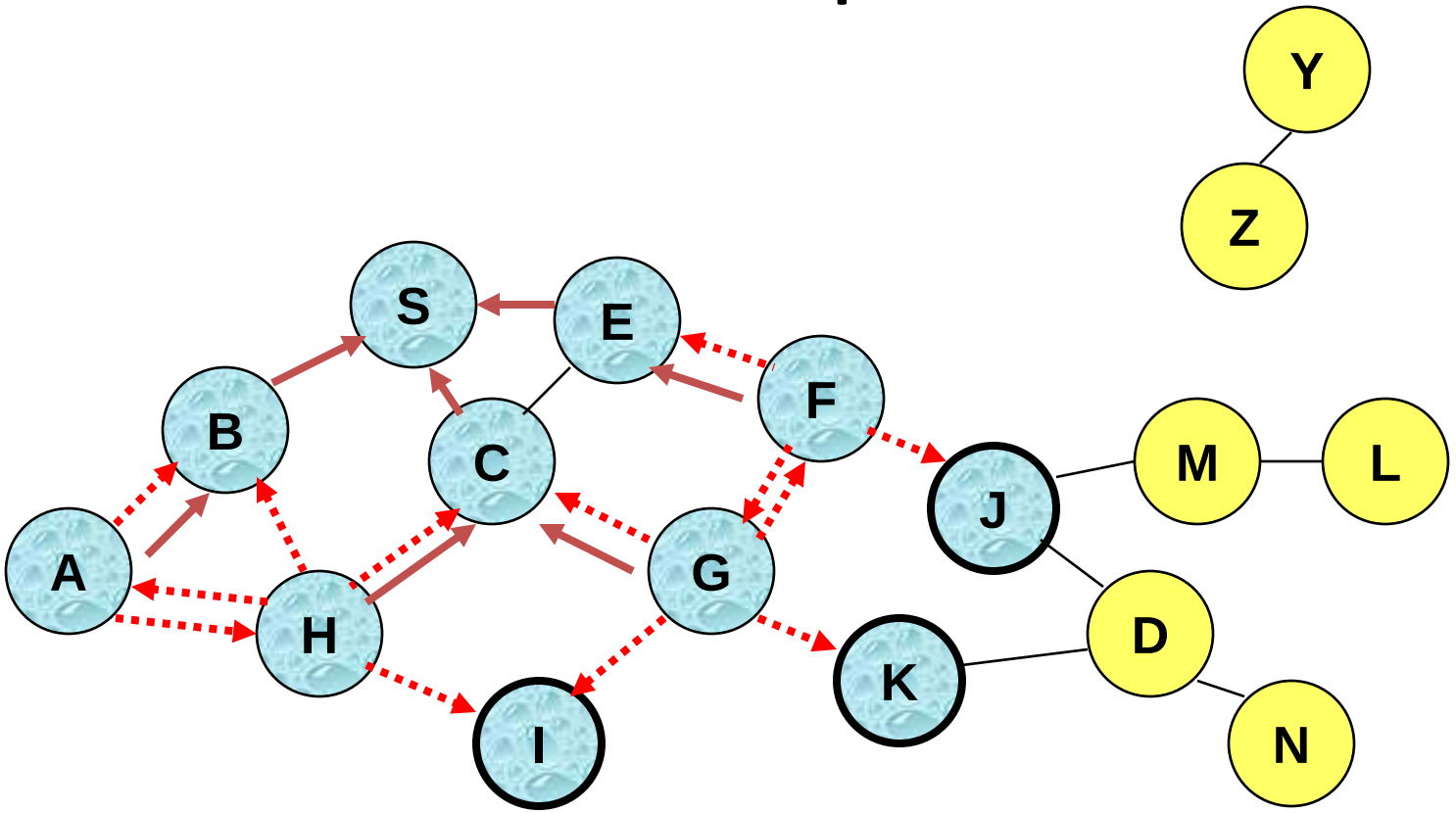


Route Requests in AODV



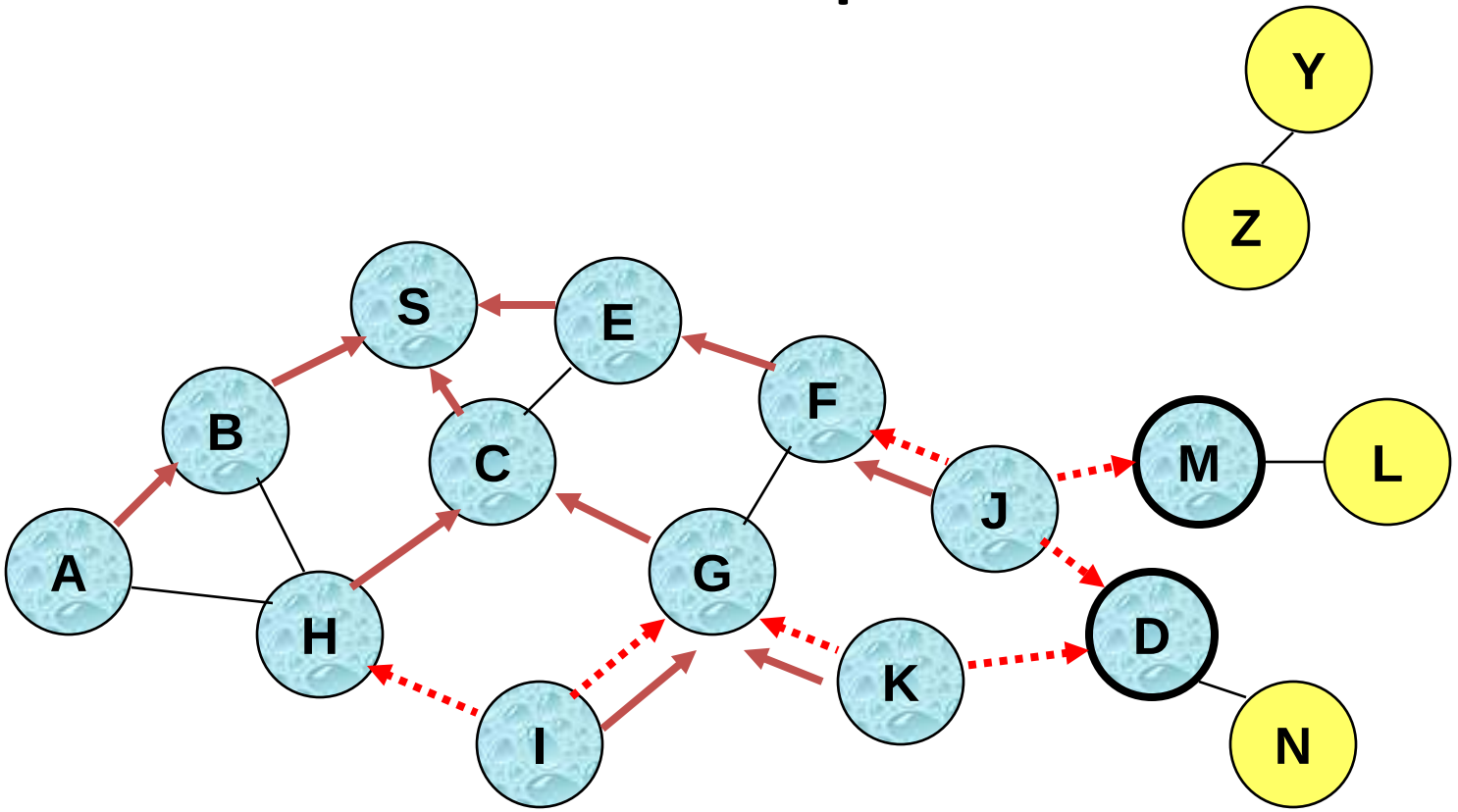
Represents links on Reverse Path

Reverse Path Setup in AODV

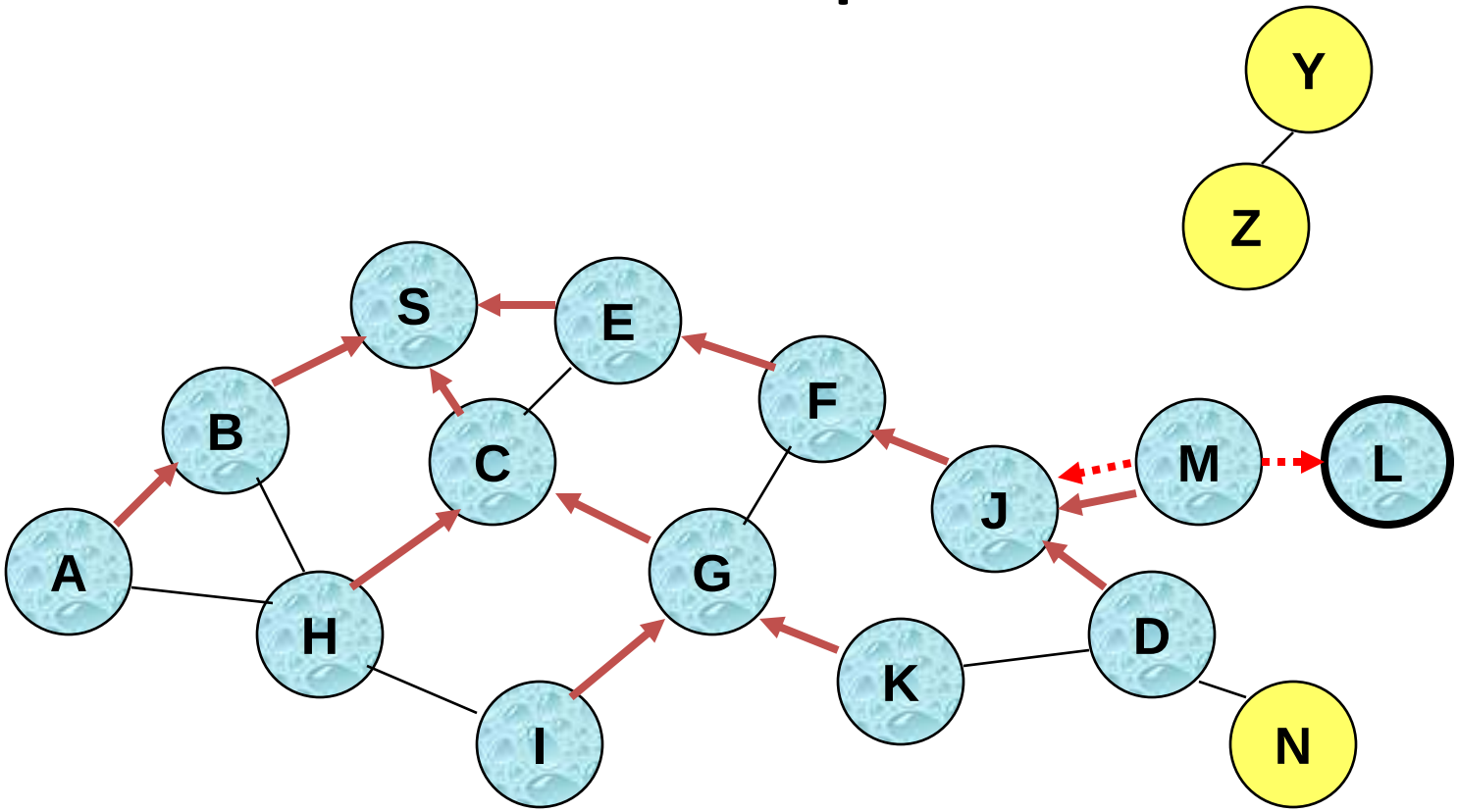


- **Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once**

Reverse Path Setup in AODV

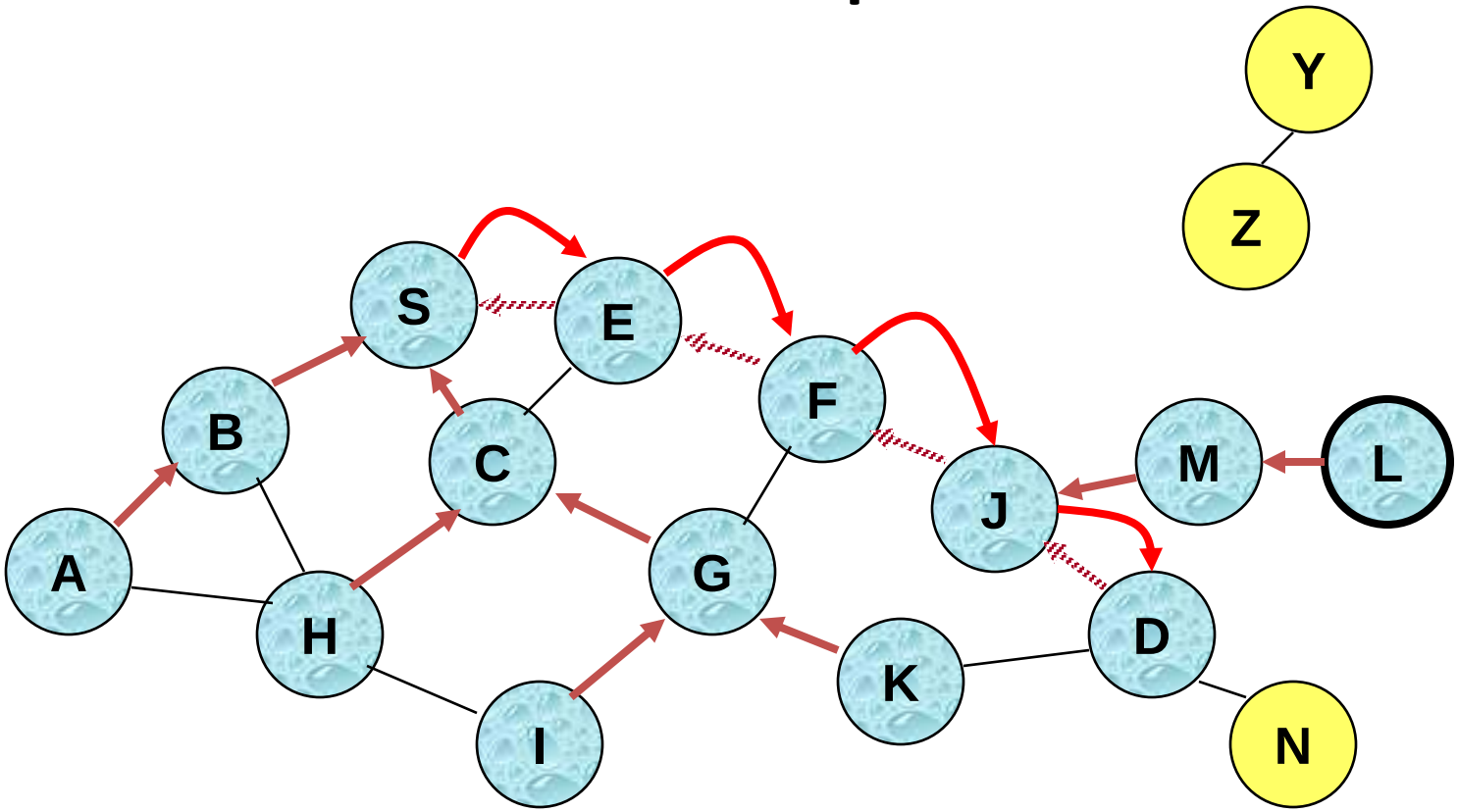


Reverse Path Setup in AODV



- **Node D does not forward RREQ, because node D is the intended target of the RREQ**

Forward Path Setup in AODV



Forward links are setup when RREP travels along the reverse path

Represents a link on the forward path

Route Request and Route Reply

- Route Request (RREQ) includes the last known sequence number for the destination

- An intermediate node may also send a Route Reply (RREP) provided that it knows a more recent path than the one previously known to sender
- Intermediate nodes that forward the RREP, also record the next hop to destination

- A routing table entry maintaining a reverse path is purged after a timeout interval
- A routing table entry maintaining a forward path is purged if *not used* for a *active_route_timeout* interval

Link Failure

- A neighbor of node X is considered active for a routing table entry if the neighbor sent a packet within *active_route_timeout* interval which was forwarded using that entry
- Neighboring nodes periodically exchange hello message
- When the next hop link in a routing table entry breaks, all active neighbors are informed
- Link failures are propagated by means of Route Error (RERR) messages, which also update destination sequence numbers

Route Error

- When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message
- Node X increments the destination sequence number for D cached at node X
- The incremented sequence number *N* is included in the RERR
- When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*
- When node D receives the route request with destination sequence number N, node D will set its sequence number to N, unless it is already larger than N

AODV: Summary

- Routes need not be included in packet headers

- Nodes maintain routing tables containing entries only for routes that are in active use
- At most one next-hop per destination maintained at each node
  - DSR may maintain several routes for a single destination

- Sequence numbers are used to avoid old/broken routes
- Sequence numbers prevent formation of routing loops

- Unused routes expire even if topology does not change

Proactive Routing Protocols

Background

- Proactive Protocols
  - Os nós avaliam constantemente as rotas para manter a consistência
  - Quando a topologia da rede muda, todos os nós devem ser atualizados
  - Rotas estão sempre disponíveis
  - Sobrecarga da rede para controle de tráfico
  - Baseado em tabelas

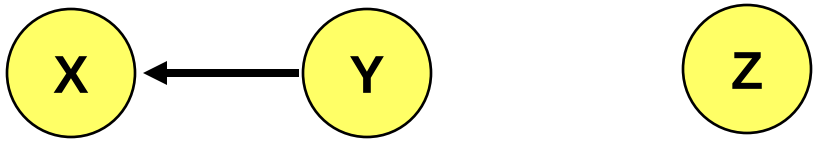
Destination-Sequenced Distance-Vector (DSDV) [Perkins94Sigcomm]

- Each node maintains a routing table which stores
  - next hop, cost metric towards each destination
  - a sequence number that is created by the destination itself
- Each node periodically forwards routing table to neighbors
  - Each node increments and appends its sequence number when sending its local routing table
- Each route is tagged with a sequence number; routes with greater sequence numbers are preferred

- Each node advertises a monotonically increasing even sequence number for itself
- When a node decides that a route is broken, it increments the sequence number of the route and advertises it with infinite metric
- Destination advertises new sequence number

Destination-Sequenced Distance-Vector (DSDV)

- When X receives information from Y about a route to Z
  - Let destination sequence number for Z at X be S(X), S(Y) is sent from Y



  - If S(X) > S(Y), then X ignores the routing information received from Y
  - If S(X) = S(Y), and cost of going through Y is smaller than the route known to X, then X sets Y as the next hop to Z
  - If S(X) < S(Y), then X sets Y as the next hop to Z, and S(X) is updated to equal S(Y)

Características

- Utilizado em MANETs (otimizado para tal)
- Faz uso de LinkState Routing
- Utiliza MultiPoint Relaying pra otimizar o LinkState Routing
- É Pró-Ativo
- É hierárquico
  - Alguns nós são diferenciados com relação aos outros
- Roda sobre UDP na porta 698

Modo de Endereçamento

- Endereços IP’s como identificadores únicos de cada nó
- OLSR suporta múltiplas interfaces de comunicação em cada nó, logo um IP deve ser escolhido como o Principal
- Suporte a IPv4 e IPv6
  - diferença se dá no tamanho mínimo de cada mensagem

Repositórios

- OLSR precisa de diferentes bases de dados para manter as informações dos estados
- As bases são atualizadas sempre que uma mensagem é recebida
- As bases são acessadas sempre que uma mensagem nova é gerada
- Toda informação registrada possui um *timeout*, ou tempo de vida. Esse tempo é obtido na mensagem que forneceu a atualização do repositório

Tipos de Repositórios

- Base de Associação a Múltiplas Interfaces
  - contém os endereços de todas as interfaces de um nó
- Links
  - mantém o estado dos links com seus vizinhos
  - única que opera nos links interface-interface diretamente
- Vizinhos
  - registra os vizinhos “1-hop”
  - dados atualizados dinamicamente com o Repositório de Links

Tipos de Repositórios

- Vizinhos "2-hop”
  - registra os vizinhos que podem ser atingidos através de um vizinho "1-hop”
  - ambos os repositórios podem ter registros em comum
- MPR
  - todos os MPR's do nó local são registrados aqui
- MPR-Selected
  - registra quais vizinhos selecionaram o nó local como MPR

Tipos de Repositórios

- Informações de Topologia
  - contém informações de todos os estados recebidos no domínio
- Replicação
  - registra informações de mensagens processadas e repassadas recentemente

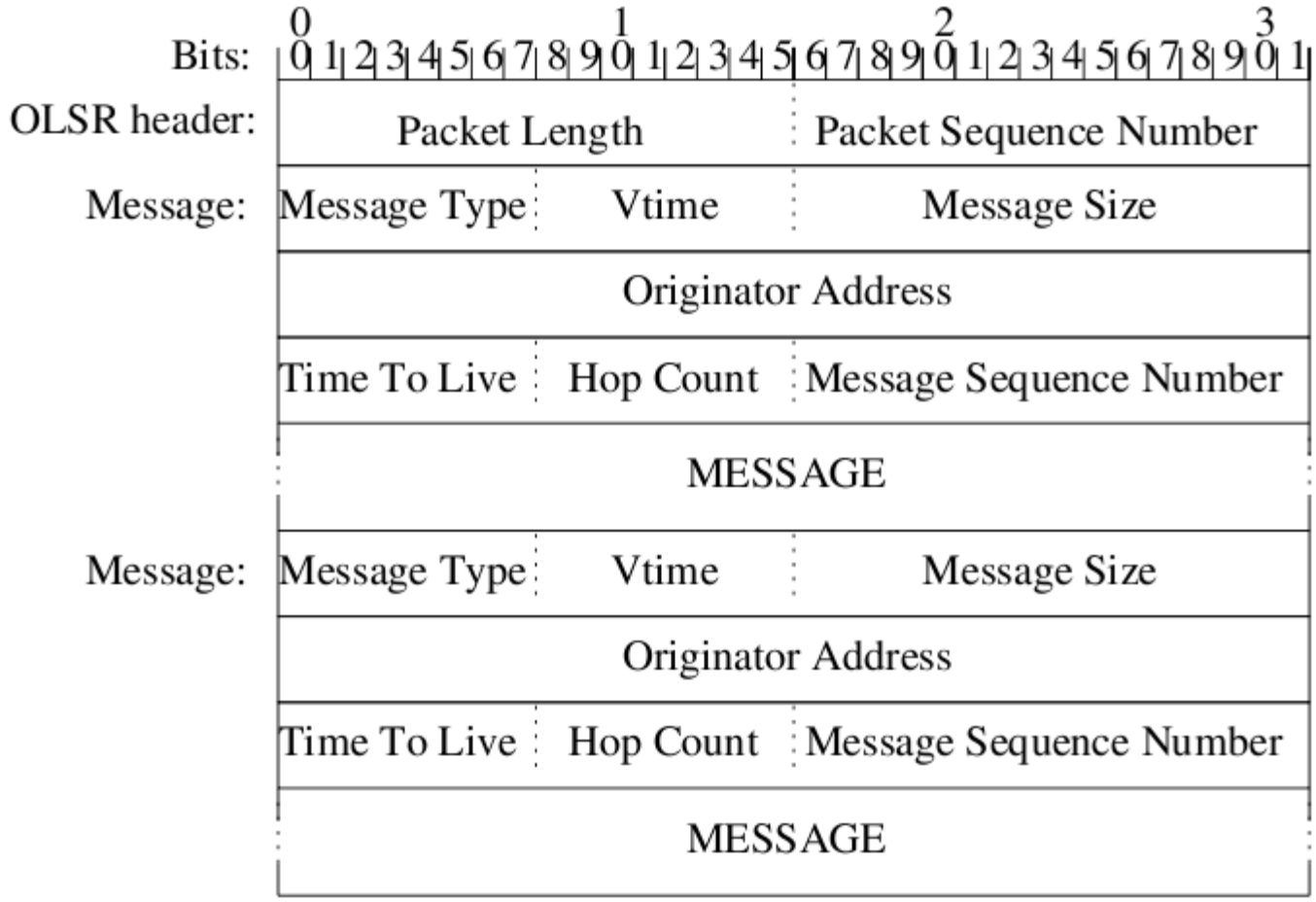
Formato de Pacotes

- OLSR Header
  - Packet Length
  - Packet Sequence Number
- OLSR Message
  - Message Type
  - Vtime
  - Message Size
  - Originator Address

Formato de Pacotes

- OLSR Message
  - Time To Live
  - Hop Count
  - Message Sequence Number
  - Message (conteúdo)

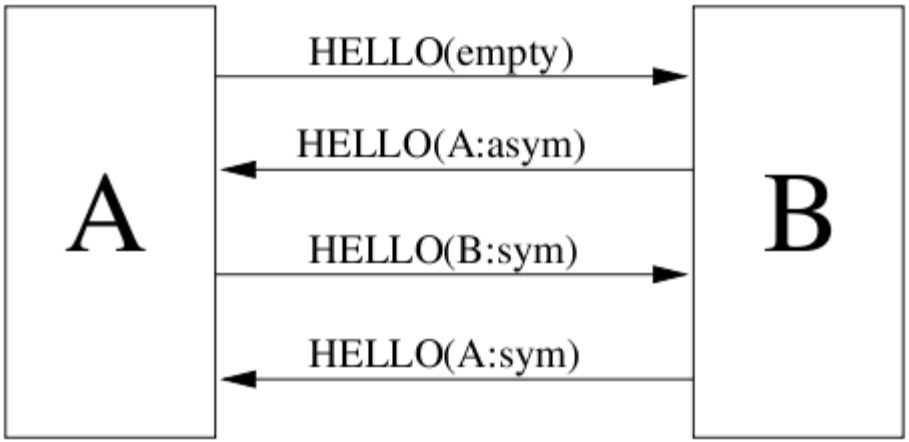
Formato de Pacotes



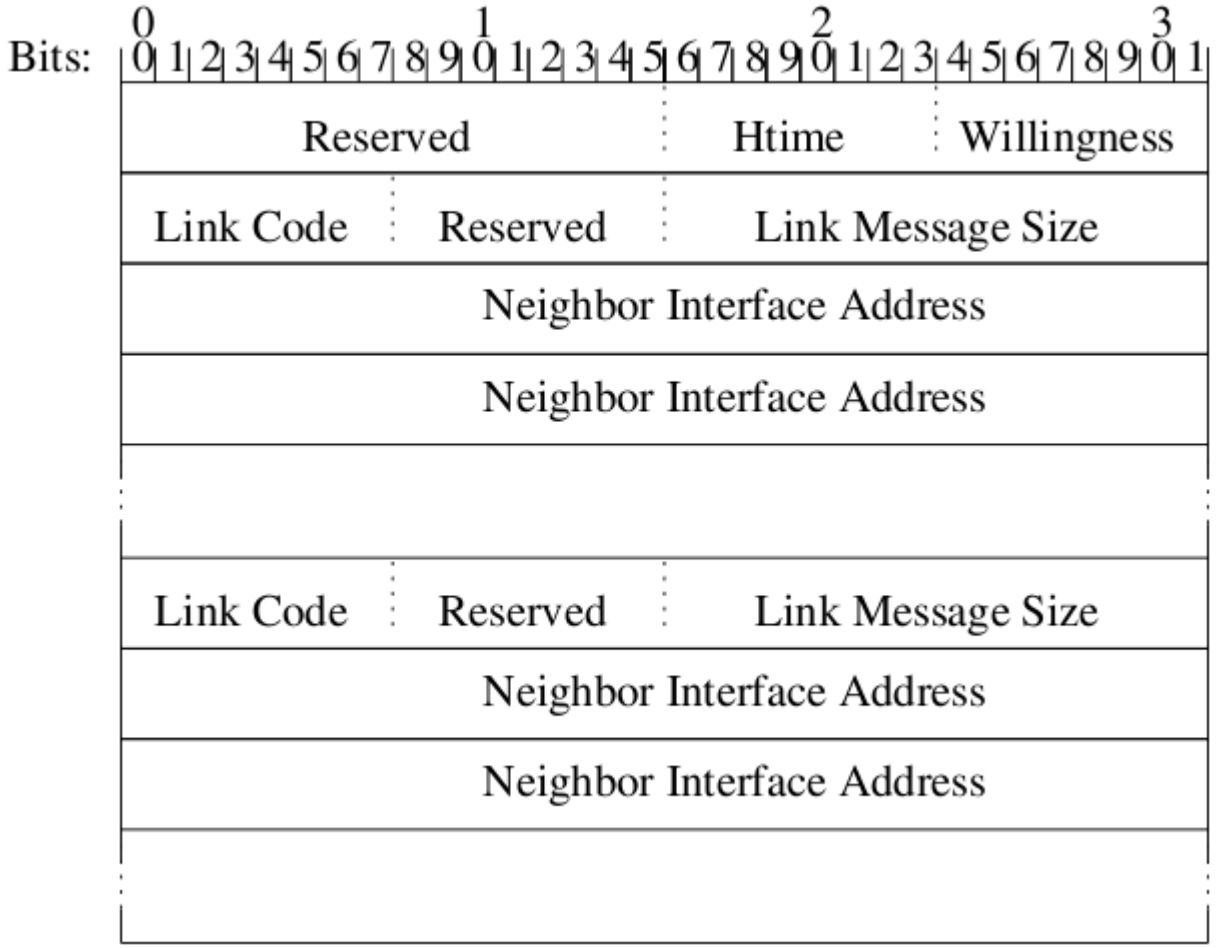
Link and neighbor Sensing

- Mensagens HELLO
  - Intervalos regulares
  - Vizinhaça conhecida
    - Topologia local
  - Broadcasted para vizinhos
  - Nunca passada adiante

Link and neighbor Sensing



Link and neighbor Sensing



Link and neighbor Sensing

- Link Sensing
  - Troca de pacotes entre interfaces
  - Nó verifica link com vizinhos 1-hop
  - Verificação bidirecional
    - Propagação de rádio
  - Par de interfaces: local/remota
  - Link: simétrico/assimétrico

Link and neighbor Sensing

- Neighbor Detection
  - Preocupação com nós e endereços principais
  - Neighbor tuples, baseados em link tuples
  - Um nó é vizinho de outro se e só se existe pelo menos um link entre os nós
  - Vizinho 2-hop: nós que têm um link simétrico para um vizinho simétrico

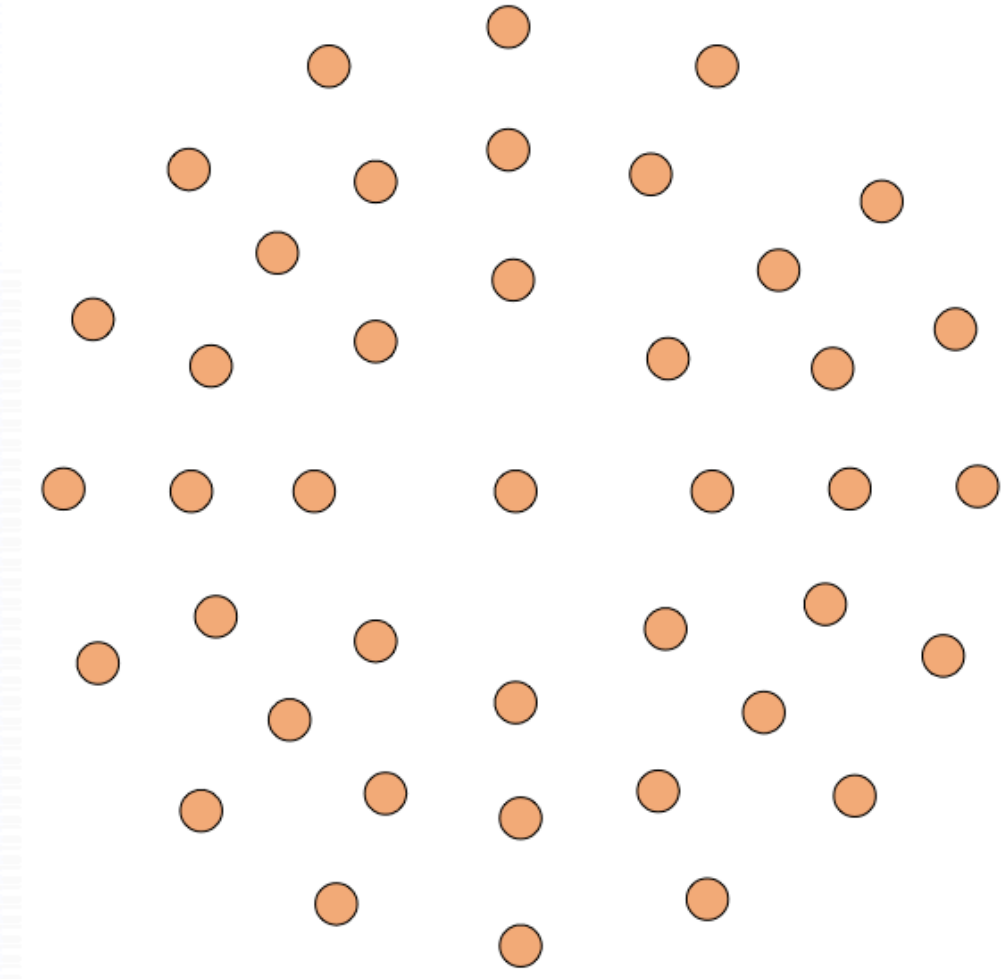
MultiPoint Relaying

- O conceito de multipoing relaying tem por objetivo reduzir of número de retransmissões duplicadas durante o a inundação de um pacote pela rede
- Essa técnica substitui o processo clássico de inundação por um processo onde apenas um conjunto de nós (MultiPoint Relay Nodes – MPR nodes) retransmite os pacotes
- O tamanho do conjunto de MPR selecionados depende da topologia da rede
- Todo nó calcula seu conjunto de MPRs através dos seu conjunto de nós vizinhos simétricos (symmetric neighbor nodes) escolhidos de tal forma que todos os vizinhos de 2 saltos (2 hop neighbors) podem ser atingidos através de um MPR.
- Essa técnica herdou certos aspectos do protocolo de roteamento FishEye.

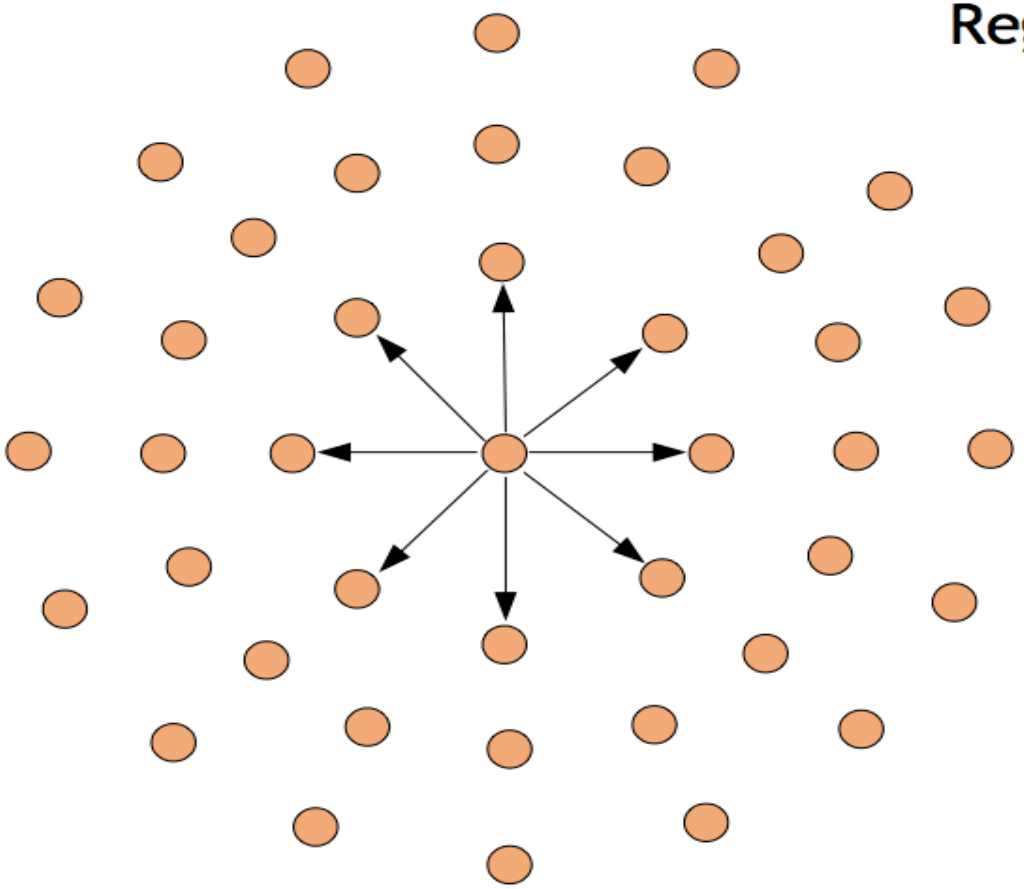
Seleção do MPR

- Obter o melhor conjunto MPR foi provado ser um problema NP-Completo
- Todos os nós selecionam e mantém os seus próprios MPRs
- OLSR provê ao nó meios de anunciar (mensagens de HELLO) a sua voluntariedade (willingness) em agir como MPR e essa informação é um dos fatores utilizados na seleção.
  - Willingness – 8 níveis (0-7)
  - WILL_NEVER(0) - Nunca ser utilizado como MPR
  - WILL_EVER(7) - Sempre ser utilizado como MPT
- RFC 3626 propoe uma heuristica simples pra o processo de seleção do MPR (S-MPR), mas outros processos podem ser utilizados em diferentes implementações como:
  - Classical flooding
  - NS-MPR
  - MPR-CDS
  - E-CDS
  - Cluster

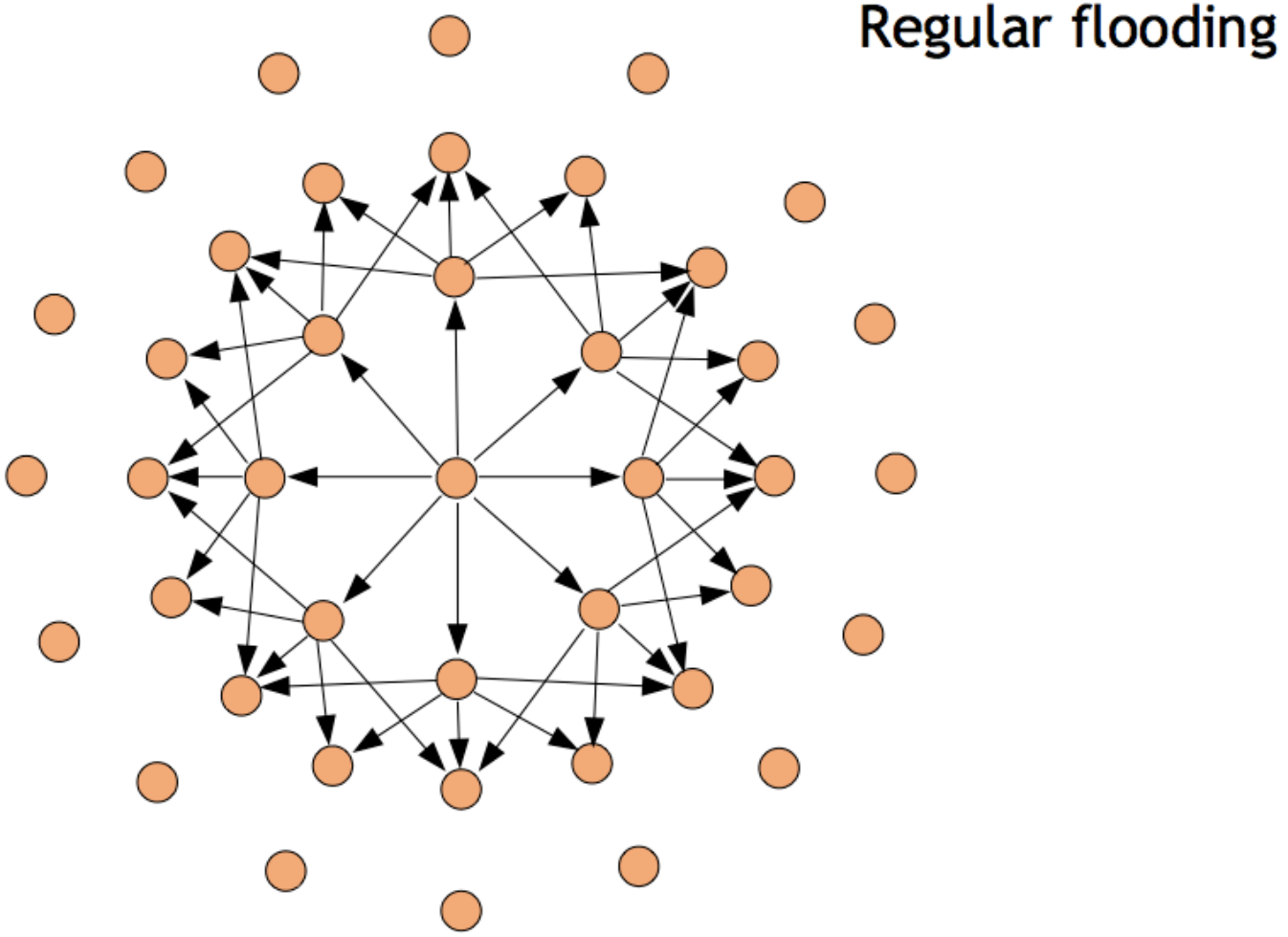
Multipoint Relay (MPR)



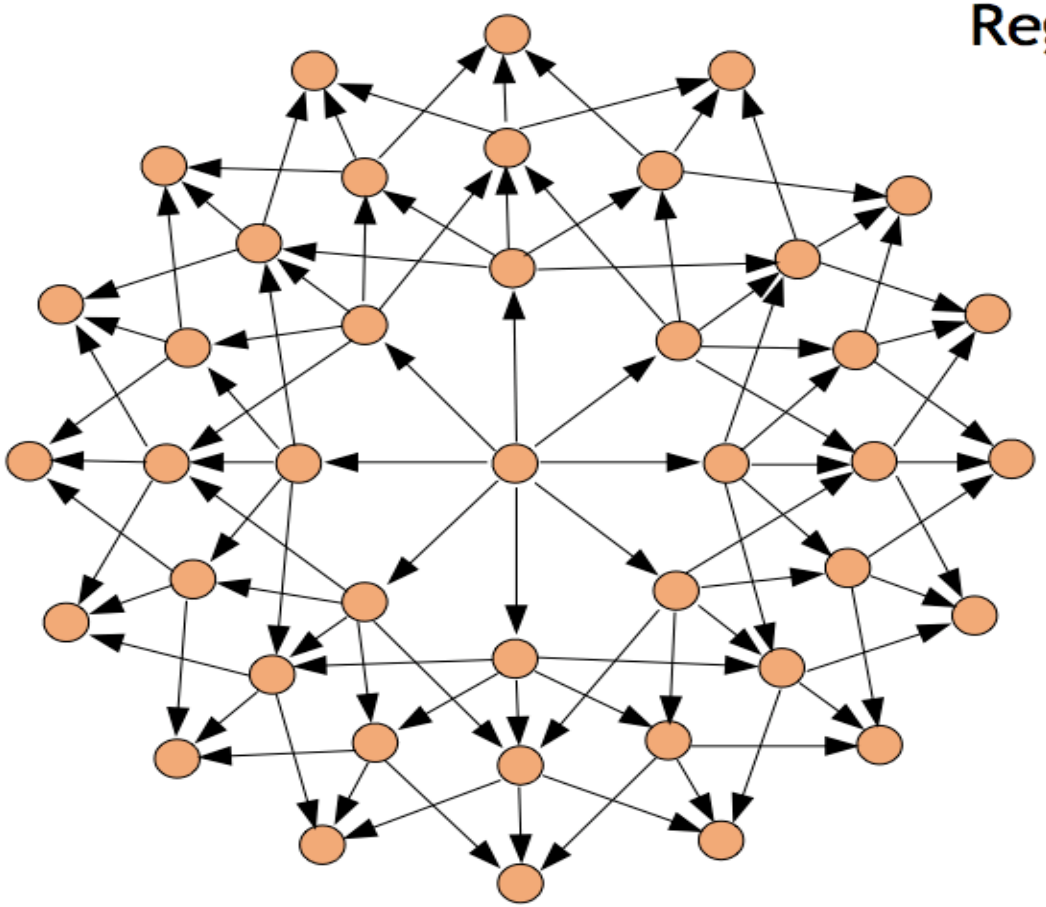
Multipoint Relay (MPR)



Multipoint Relay (MPR)

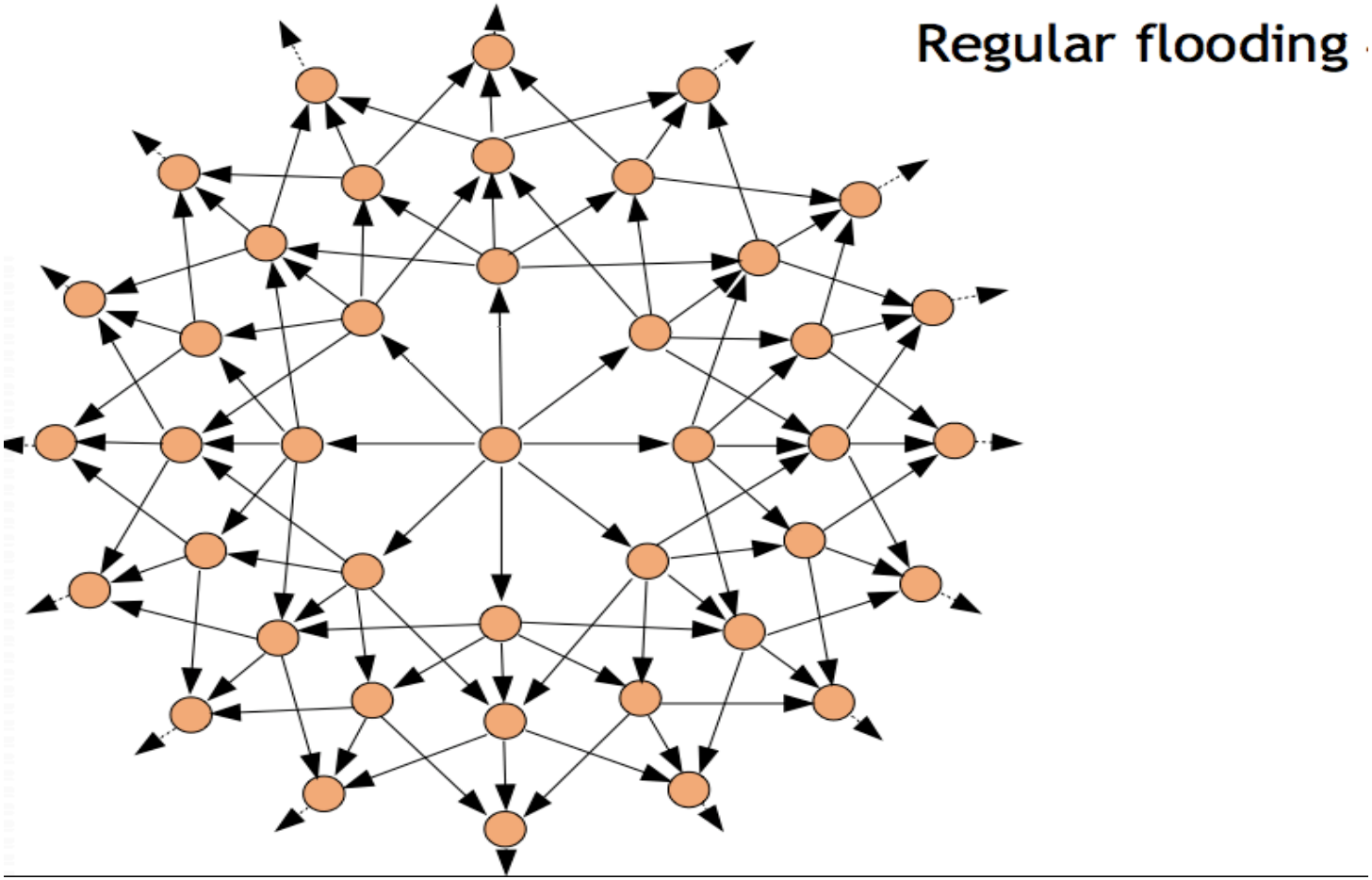


Multipoint Relay (MPR)

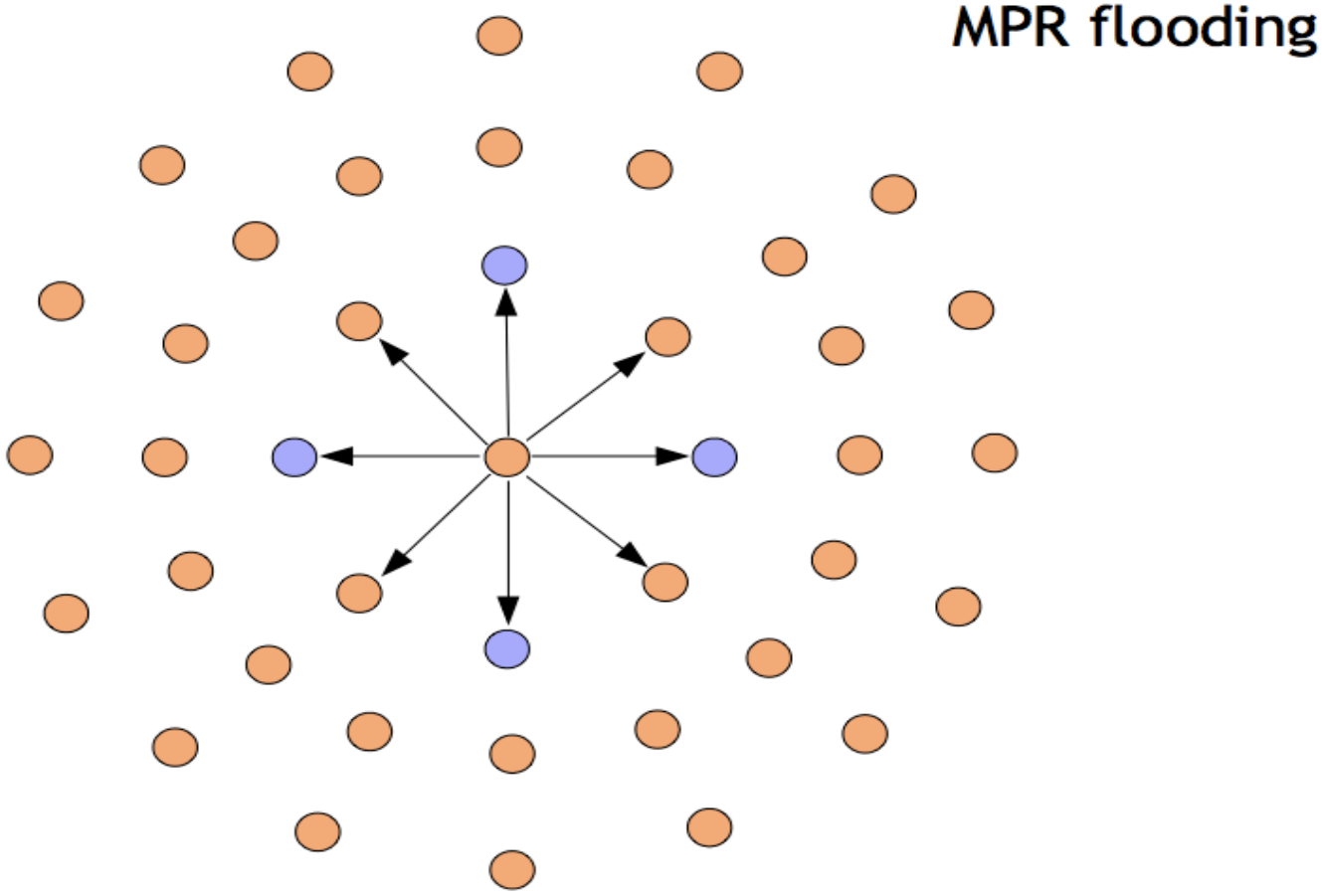


Regular flooding

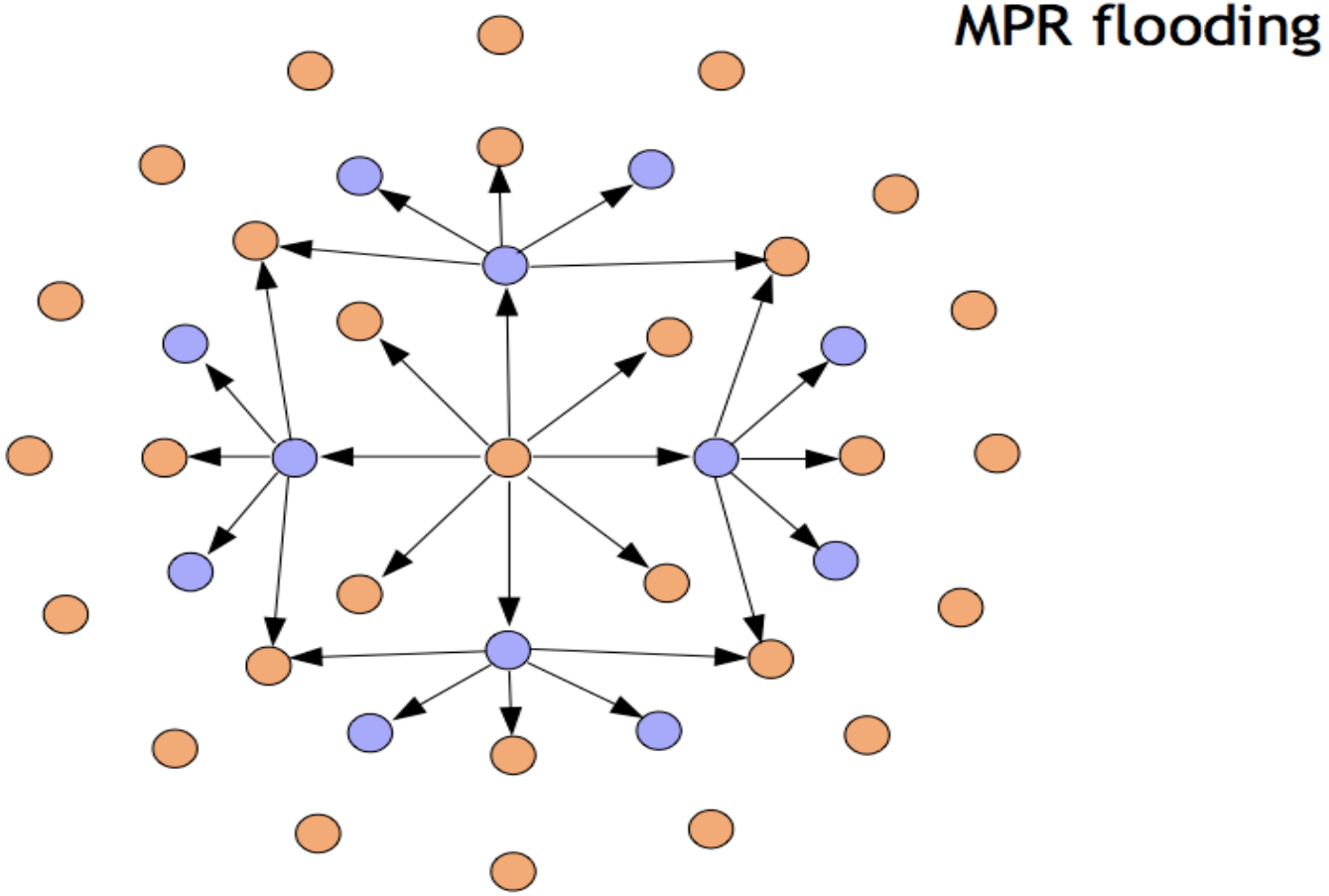
Multipoint Relay (MPR)



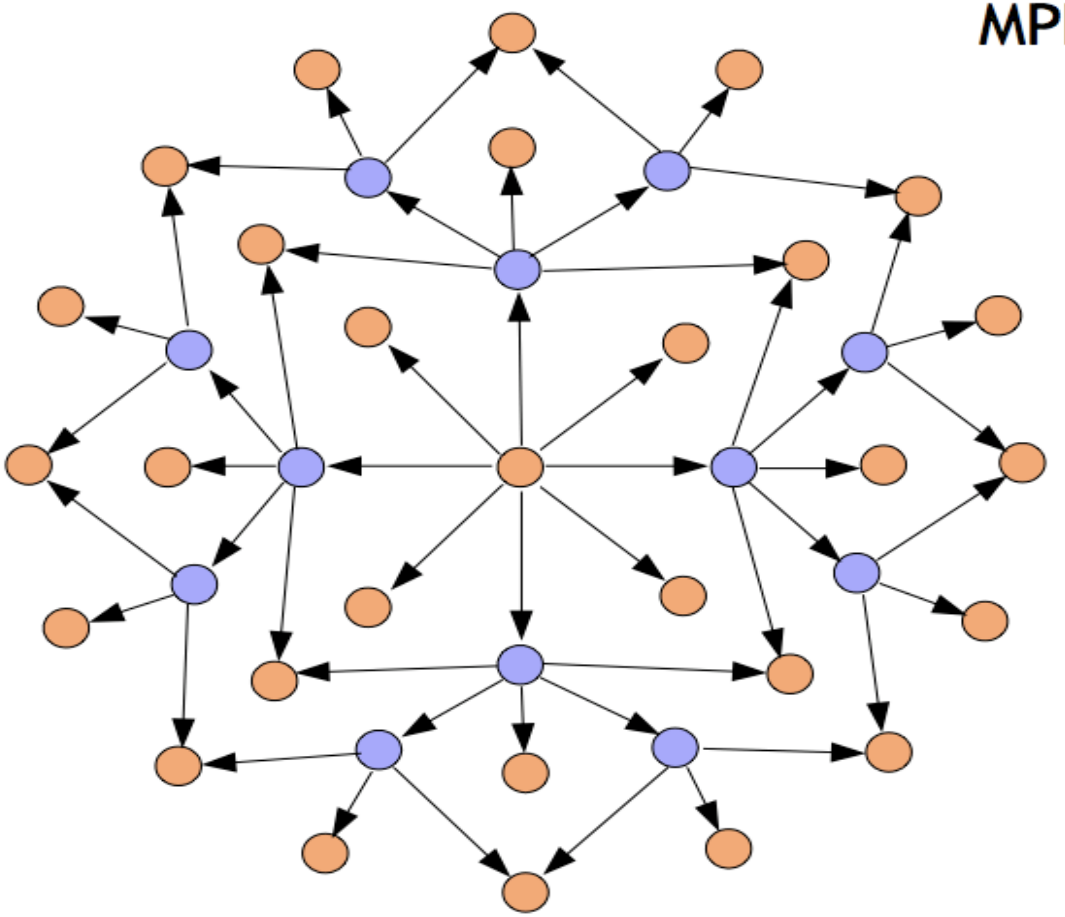
Multipoint Relay (MPR)



Multipoint Relay (MPR)

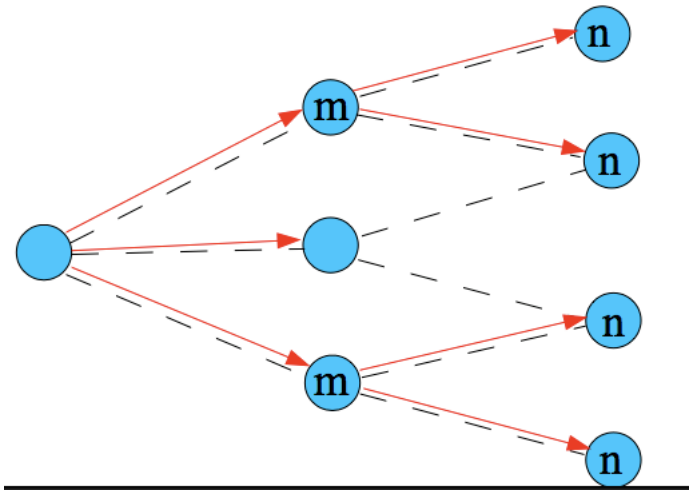


Multipoint Relay (MPR)



Encaminhamento de Tráfico

- Regra
  - Um nó só irá retransmitir um pacote OLSR se tiver sido escolhido como MPR pelo ultimo nó que retransmitiu o pacote e se esse pacote tiver TTL > 0
- Duplicate-Set
- Forward-jitter



Implementação do OLSR

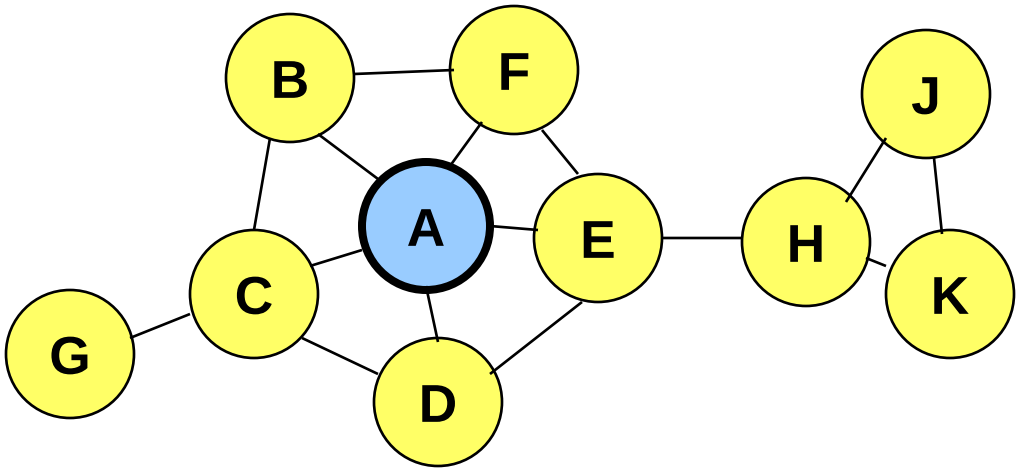
- Olsrd
  - Tem suporte a plugins
  - Existem redes comunitárias sem fio em mesh que o utilizam:
    - 2000 nós (Athens wireless network)
    - ~ 600 nós (berlin FreiFunk.net)
    - ~ 400 nós - Leipzig Freifunk net
  - Plataformas Suportadas:
    - Windows (XP e Vista)
    - Linux (i386, arm, alpha, mips, xscale)
    - OS X (powerpc, intel, xscale, iPhone)
    - VxWorks
    - NetBSD, FreeBSD, OpenBSD
    - WIP (WiFi Phones)
    - $100 laptop ;-)
    - Intel Classmate

Implementação do OLSR

- OOLSR
  - Desenvolvido pela Hipercom (INRIA)
  - Implementado em C++
  - Plataformas Suportadas:
    - Windows (XP/2000 e CE)
    - Linux (i386)
    - Network Simulator 2- NS-2
- NRL OLSR
  - Produzida pela laborátorio de pesquisa naval dos USA (NRL).
  - Plataformas Suportadas:
    - Windows,
    - MacOSX
    - Linux
    - Zaurus (PDA)
    - PocketPC
  - Foi implementada usando a protolib programming toolkit para gerar código para outros ambientes (NS, OPNET)

Optimized Link State Routing (OLSR) [Jacquet00ietf]

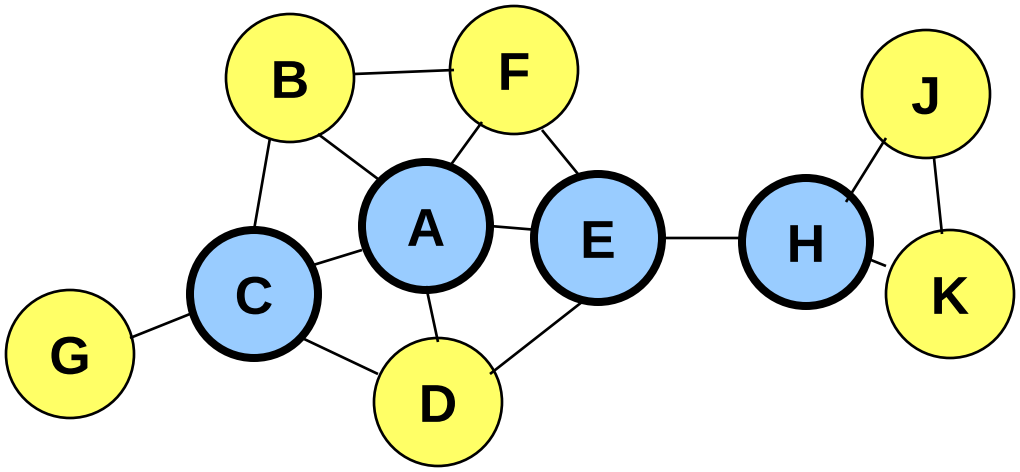
- Nodes C and E are multipoint relays of node A
  - Multipoint relays of A are its neighbors such that each two-hop neighbor of A is a one-hop neighbor of one multipoint relay of A
  - Nodes exchange neighbor lists to know their 2-hop neighbors and choose the multipoint relays



Node that has broadcast state information from A

Optimized Link State Routing (OLSR)

- Nodes C and E forward information received from A
- Nodes E and K are multipoint relays for node H
- Node K forwards information received from H



Node that has broadcast state information from A

GPSR: Greedy Perimeter Stateless Routing for Wireless Networks

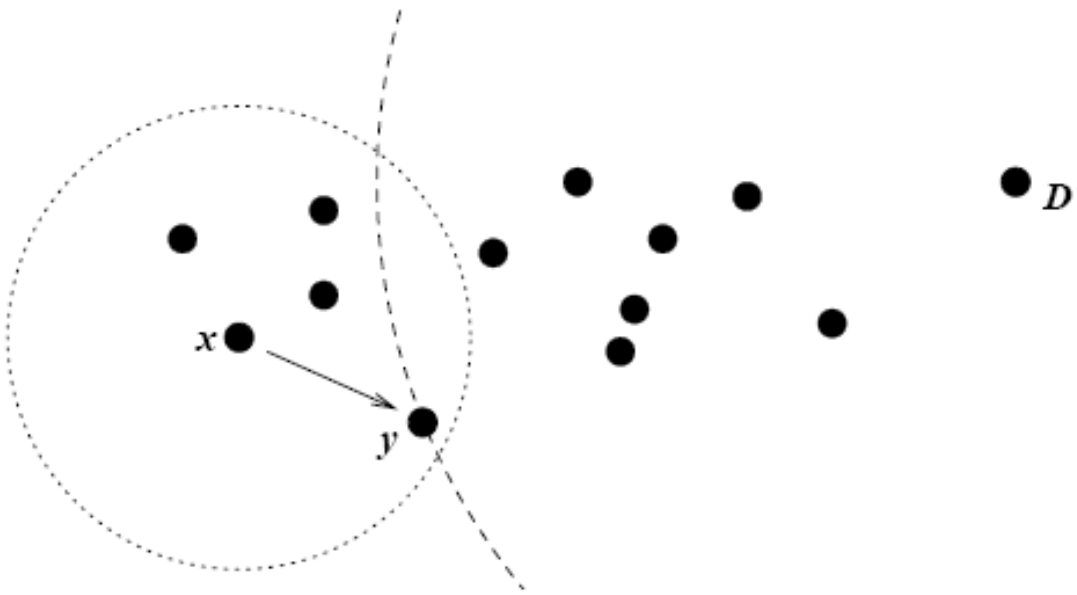
Brad Karp, H. T. Kung
Harvard University

GPSR: Motivation

- Ad-hoc routing algorithms (DSR, AODV)
  - Suffer from out of date state
  - Hard to scale
- Use geographic information for routing
  - Assume every node knows position (x,y)
  - Keep a lot less state in the network
  - Require fewer update messages

GPSR Algorithm : Greedy Forwarding

- Each node knows the geographic location of its neighbors and destination
- Select the neighbor that is geographically closest to the



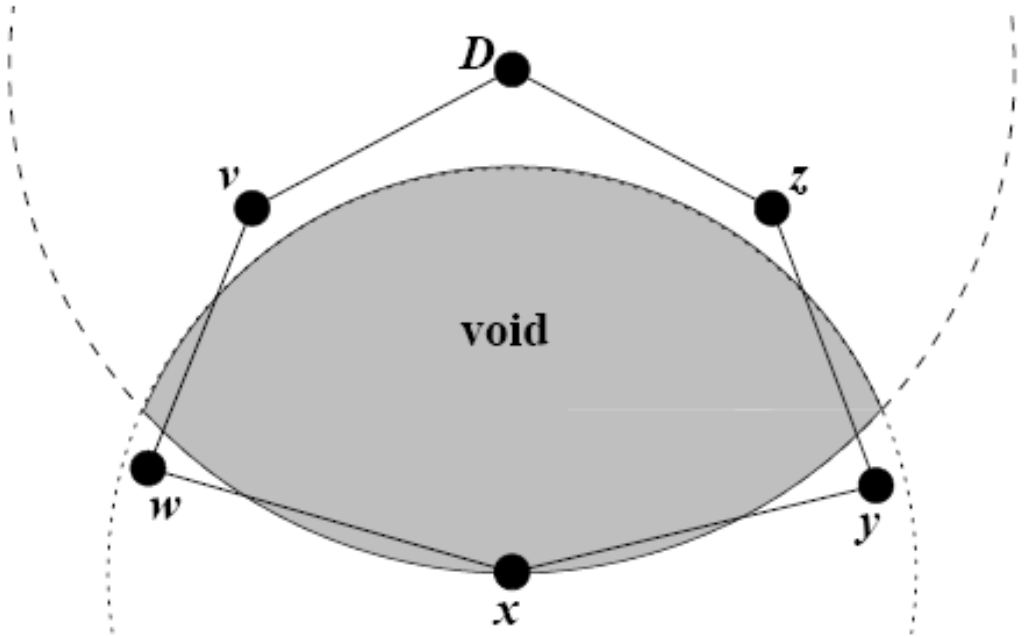
GPSR Algorithm : Greedy Forwarding (Cont.)

- Each node only needs to keep state for its neighbors

- Beaconing mechanism
  - Provides all nodes with neighbors' positions
  - Beacon contains broadcast MAC and position
  - To minimize costs: piggybacking

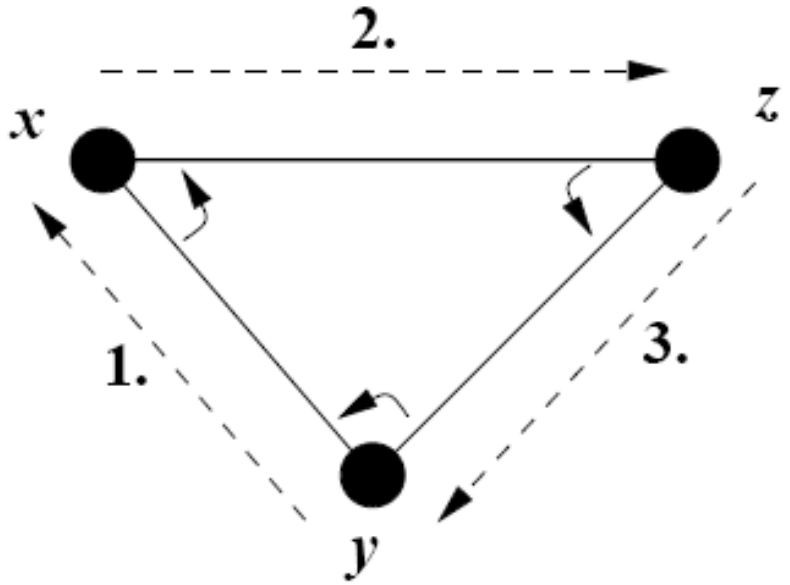
GPSR Algorithm : Greedy Forwarding (Cont.)

- Greedy forwarding does not always work!



Getting Around Void

- The right hand rule
  - When arriving at node x from node y, the next edge traversed is the next one sequentially counterclockwise about x from edge (x,y)
  - Traverse the exterior region in counter-clockwise edge order



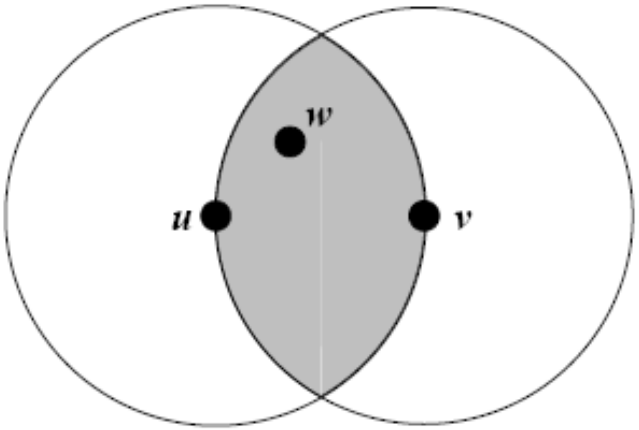
Planarized Graphs

- A graph in which no two edges cross is known as *planar.*
  - Relative Neighborhood Graph (RNG)
  - Gabriel Graph (GG)

Relative Neighborhood Graph

An edge (u, v) exists between vertices u and v if the distance between them, $d(u, v)$, is less than or equal to the distance between every *other* vertex w, and whichever of u and v is farther from w. In equational form:

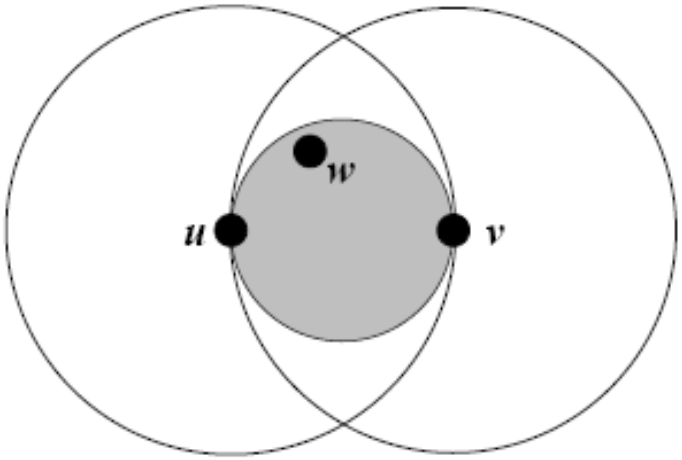
$$\forall w \neq u, v : d(u, v) \leq \max[d(u, w), d(v, w)]$$



Gabriel Graph

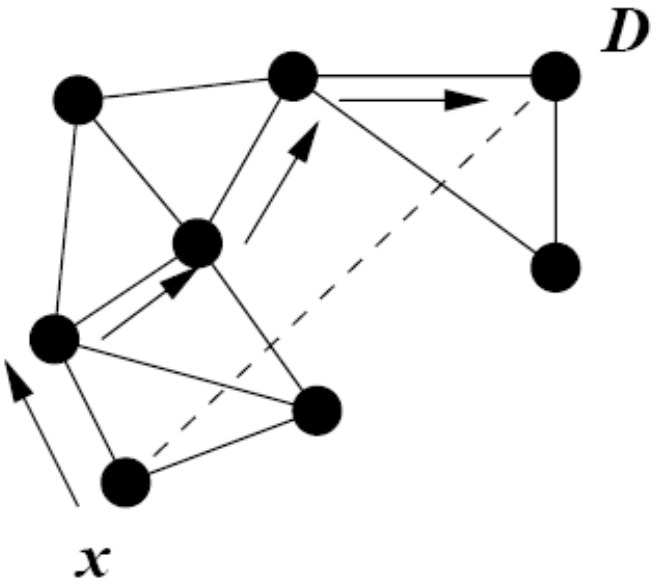
An edge (u, v) exists between vertices u and v if no other vertex w is present within the circle whose diameter is $\overline{uv}$. In equational form:

$$\forall w \neq u, v : d^2(u, v) < [d^2(u, w) + d^2(v, w)]$$



Final Algorithm

- Combine greedy forwarding + perimeter routing
  - Use greedy forwarding whenever possible
  - Resort to perimeter routing when greedy forwarding fails and record current location Lc
  - Resume greedy forwarding when we are closer to destinat



Pros and Cons?

Pros and Cons

- Pros:
  - Low routing state and control traffic → scalable
  - Handles mobility

- Cons:
  - GPS location system might not be available everywhere.
  - Geographic distance does not correlate well with network proximity.
  - Overhead in location registration and lookup
  - Limitations of planarization algorithm
    - works under unit disk model, which doesn't hold in practical network
    - hard to handle mobility
    - planarization reduces network connectivity

Beacon Vector Routing

Scalable Point-to-point Routing in Wireless Sensor Networks

R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, I. Stoica

UC Berkeley

Beacon Vector Routing

- Solution: fake geography
  - Create a routing gradient from connectivity information rather than geography
    - Nodes assigned positions based on connectivity
    - Greedy forwarding on this space

Beacon-Vector: Algorithm

- 3 pieces
  - Deriving positions
  - Forwarding rules
  - Lookup: mapping node IDs → positions

Beacon-Vector: deriving positions

1. r beacon nodes (B_0,B_1,…,B_r) flood the network; a node q's position, P(q), is its distance in hops to each beacon

   $P(q) = \langle B_1(q), B_2(q),\ldots,B_r(q) \rangle$

2. Node p advertises its coordinates using the k closest beacons (we call this set of beacons C(k,p))
3. Nodes know their own neighbors' positions
4. Nodes also know how to get to each beacon

Beacon-Vector: forwarding

1. Define the distance between two nodes P and Q as

$$\mathrm{dist}_k(p,q) = \sum_{i \in C(k,q)} \omega_i \left| B_i(p) - B_i(q) \right|$$

2. To reach destination Q, choose neighbor to reduce $\mathrm{dist}_k(*,Q)$
3. If no neighbor improves, enter Fallback mode: route towards the beacon which is closer to the destination
4. If Fallback fails and you reach the beacon, do a scoped flood

Does the forwarding scheme guarantee reachability?

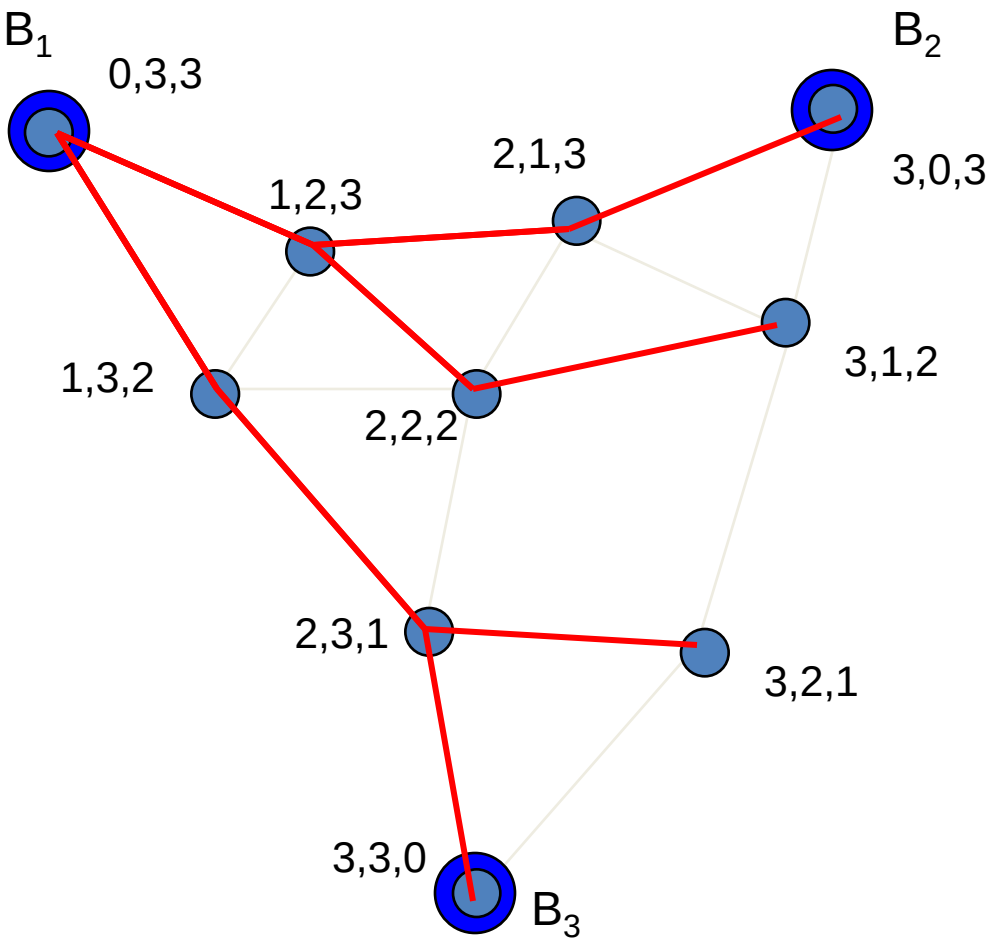
Beacon maintenance

- Route based on the beacons the source and destination have in common
  - Does not require perfect beacon info.
- Each entry in the beacon vector has a sequence number
  - Periodically updated by the corresponding beacon
  - Timeout
- If the #beacons < r, non-beacon nodes nominate themselves as beacons

Location directory

- Store location mapping at beacon nodes
  - Hashing H: nodeid → beaconid [14]
- Each node k that wants to be a destination periodically publishes its coordinates to its corresponding beacon $b_k = H(k)$
- When a node wants to route to node k, it sends a lookup request to b_k
- Cache the coordinates

Simple example



Simple example