SAPIENZA
UNIVERSITÀ DI ROMA

Routing in reti ad hoc

Sistemi Wireless, a.a 2009/2010

Un. of Rome "La Sapienza"

Chiara Petrioli[†]

[†] *Department of Computer Science – University of Rome "Sapienza" – Italy*

What is an ad hoc network

- A wireless multi-hop infrastructure-less network whose devices act as source/ destination of messages & as relay for packets generated by a node s and addressed to a node z (iff they are on a s-z route)
- Pros: No need for infrastructure → low cost, enables communication where it is usually not needed or is not viable
- Must be: Self-organizing, self-configuring, self-maintaining

SAPIENZA
UNIVERSITÀ DI ROMA

Application scenarios

- Collaboration between users in office environments
- Disaster recovery applications
- Military networks
- Personal Area Networks
- Home Networking
- Wireless Sensor Networks (WSNs)
- Inter-vehicular communication

SAPIENZA
UNIVERSITÀ DI ROMA

Features of ad hoc networks

- Highly dynamic networks → device mobility, energy saving sleep/awake modes
- Need for low energy/resource-consuming, simple protocols
- Bandwidth and resource constrained environment
- Traffic:
  - All-pairs in general ad hoc networks, from sensors to sink(s) in sensor networks
  - In many case not high
- Scale: Application dependent
  - 10-100 nodes in traditional ad hoc networks
  - 1000-10000 in sensor networks

SAPIENZA
UNIVERSITÀ DI ROMA

Features of highly resource constrained ad hoc networks (WSNs)

- Highly dynamic networks → due to device mobility (only in some specific applications), to the fact the active node set changes in time for sake of energy saving (always to be considered)
- Need to design low energy/resource-consuming, simple protocols → very critical, energy consumption a real bottleneck
- Traffic from sensors to sink(s)
- Scalability is a major issue
- Code must be simple (small storage capability, very simple, inexpensive, resource constrained devices)
- First solutions we will see for traditional ad hoc networks do not scale to high numbers and are not energy-saving

Routing-Traditional approach (from Reti 1)

- Intra-AS routing in the Internet
  - Link State Approaches

  (info on the topology graph gathered at nodes which run shortest path algorithms-Dijkstra- to decide the routes to the different destinations –e.g. OSPF routing protocol)
  - Distance Vector approaches (e.g. RIP)

Bellman-Ford

Given a graph G=(N,A) and a node s find the shortest path from s to every node in N.

A shortest walk from *s* to i subject to the constraint that the walk contains at most h arcs and goes through node *s* only once, is denoted **shortest(<=h) walk and its length is D^h_i.**

Bellman-Ford rule:

Initiatilization $D^h_s=0$, for all h; $c_{i,k}$ = infinity if (i,k) NOT in *A*; $c_{k,k}=0$;
D^0_i=infinity for all i!=s

Iteration:

$$D^{h+1}_i=\min_k [c_{i,k} + D^h_k]$$

Assumption: non negative cycles **(this is the case in a network!!)**

The Bellman-Ford algorithm first finds the one-arc shortest walk lengths, then the two-arc shortest walk length, then the three-arc…etc.
→distributed version used for routing

Bellman-Ford

$$D^{h+1}_i = \min_k [c_{i,k} + D^h_k]$$

Can be computed locally.
What do I need?

For each neighbor *k*, I need to know
-the cost of the link to it (known info)
-The cost of the best route from the neighbor *k* to the destination (←this is an info that each of my neighbor has to send to me via messages)
In the real world: I need to know the best routes among each pair of nodes → we apply distributed Bellman Ford to get the best route for each of the possible destinations

SAPIENZA
UNIVERSITÀ DI ROMA

Distance Vector Routing Algorithm -Distributed Bellman Ford

iterative:

- continues until no nodes exchange info.
- *self-terminating*: no "signal" to stop

asynchronous:

- nodes need *not* exchange info/ iterate in lock step!

Distributed, based on local info:

- each node communicates *only* with directly-attached neighbors

Distance Table data structure

each node has its own

- row for each possible destination
- column for each directly-attached neighbor to node
- example: in node X, for dest. Y via neighbor Z:

Cost associated to the (X,Z) link

$$D^{X}(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop}$$

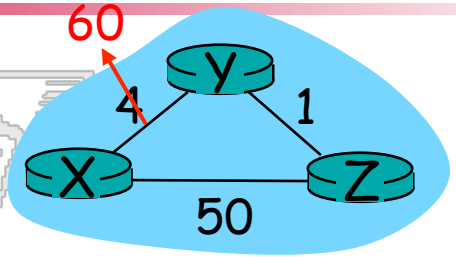
$$= c(X,Z) + \min_{w}\{D^{Z}(Y,w)\}$$

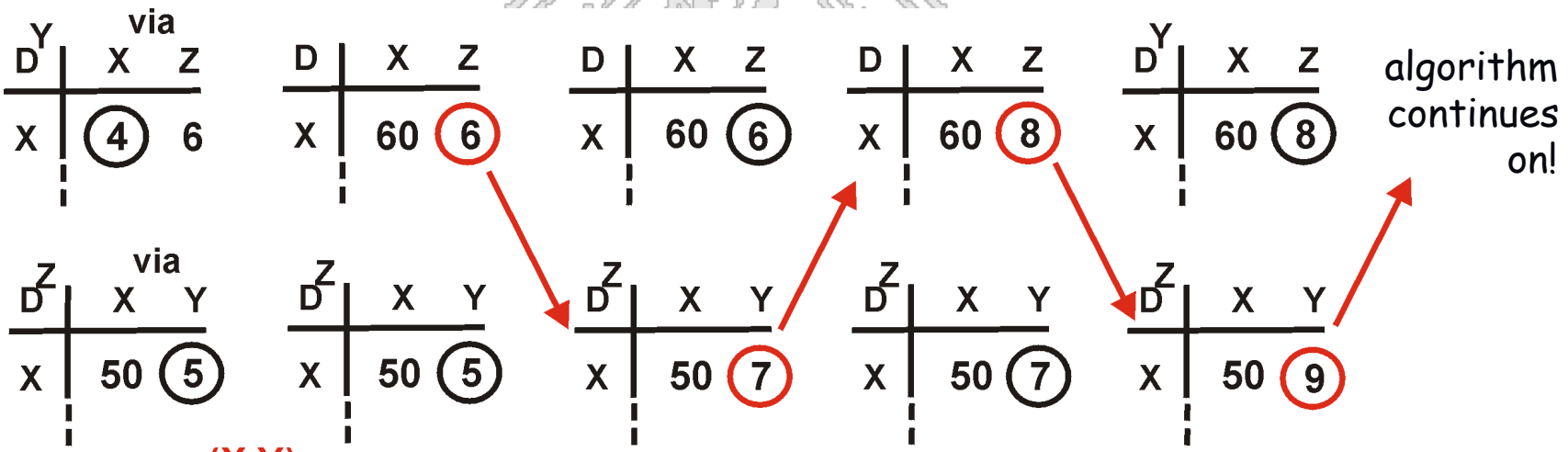
Info maintained at Z. Min must be communicated

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- ***bad news travels slow*** - "count to infinity" problem!





Y detects link cost Increase but think can Reach X through Z at a total cost of 6 (wrong!!)

→ **The path is Y-Z-Y-X**

Count-to-infinity – an everyday life example

Which is the problem here?

the info exchanged by the protocol!! 'the best route to X I have has the following cost…' (no additional info on the route)

A Roman example…

-assumption: there is only one route going from Colosseo to Altare della Patria: Via dei Fori Imperiali. Let us now consider a network, whose nodes are Colosseo., Altare della Patria, Piazza del Popolo

Count-to-infinity – everyday life example (2/2)

The Colosseo. and Alt. Patria nodes exchange the following info

- Colosseo says 'the shortest route from me to P. Popolo is 2 Km'
- Alt. Patria says 'the shortest path from me to P. Popolo is 1Km'

Based on this exchange from Colosseo you go to Al. Patria, and from there to Piazza del Popolo OK **Now due to the big dig they close Via del Corso (Al. Patria—P.Popolo)**

- Al. Patria thinks 'I have to find another route from me to P.Popolo. Look there is a route from Colosseo to P.Popolo that takes 2Km, I can be at Colosseo in 1Km → I have found a 3Km route from me to P.Popolo!!' Communicates the new cost to Colosseo that updates 'OK I can go to P.Popolo via Al. Patria in 4Km'

VERY WRONG!! Why is it so? I didn't know that the route from Colosseo to P.Popolo was going through Via del Corso from Al.Patria to P.Popolo (which is closed)!!

Count to infinity-solutions

- Numero massimo di hop count
  - L'impatto di loop può essere limitato tramite un campo del pacchetto che tenga conto degli hop attraversati. I pacchetti che sono stati in rete per un numero di hop superiore ad una soglia prefissata vengono scartati.
  - se la dimensione della rete è limitata si può limitare (come si fa in RIP) il numero di hop num_max_hops al diametro della rete (pari ad es. al valore 15). Se, come in RIP, il costo dei link è pari a 1 ponendo infinito=16 il protocollo opera correttamente e converge rapidamente nel caso in cui si verifichino problemi di tipo count to infinity.
- Split horizon con poison reverse
  - Si limita l'informazione trasmessa. Se A usa informazioni ricevute da B per scegliere la rotta verso la destinazione D (A passa tramite B per raggiungere D), A non comunicherà un costo per raggiungere D a B oppure (variante Poison Reverse) comunicherà un costo infinito.
    - ✓ La tecnica assume un mezzo non broadcast ed un vicinato noto
    - ✓ Non risolve tutti i loop
- Trigger Updates (velocizzazione della convergenza)
  - Anziché trasmettere periodicamente aggiornamenti tali aggiornamenti possono essere immediatamente inviati nel caso in cui si individuino cambiamenti.

SAPIENZA
UNIVERSITÀ DI ROMA

Routing in ad hoc networks- Goals

- Multi-hop path routing capability
- Dynamic topology maintenance
- "No loops"
- Minimal control overhead
- Low processing overhead
- Self-starting

SAPIENZA
UNIVERSITÀ DI ROMA

Primary Approaches

- Proactive
  - Based on traditional distance-vector and link-state protocols
  - Each node maintains route to each other network node
  - Periodic and/or event triggered routing update exchange
  - Higher overhead in most scenarios
  - Longer route convergence time
  - Examples: DSDV, OLSR

SAPIENZA
UNIVERSITÀ DI ROMA

Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing

- Proactive, distance vector approach (uses distributed asynchronous Bellman Ford). Updates on route costs transmitted periodically or when significant new information is available.
- Difference wrt Bellman Ford: in ad hoc networks there are frequent changes in the topology, solutions must try to avoid loops (approaches such as Poison reverse non effective in broadcast channels, we seek solutions which are simple and fully distributed)
- Metrics: fresh routes better than stale routes, number of hops used to select among the fresh routes
- How to identify fresh routes? By means of sequence numbers identifying the freshness of the communicated information. When changes occur, the sequence number increase.

SAPIENZA
UNIVERSITÀ DI ROMA

Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing

- Periodically destination nodes transmit updates with a new sequence number (and such updates are propagated by the other nodes).
- Updates periodically sent by nodes contain information on the costs to achieve the different destinations and the freshness of the route
- Data broadcast include multiple entries each with:
  - Destination address
  - Number of hops required to reach the destination
  - Sequence number of the information received regarding that destination as originally stamped by the destination
- In the header the data broadcast also include:
  - Address (HW address/Net address) of the sender of the message
  - Sequence number created by the transmitter
- Two types of updates (full dump or incremental-only changes- to decrease bandwidth consumption.

Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing

- How can the costs be modified? Cost=number of hops, target: using fresh routes as short as possible → a link cost changes from 1 to inf and from inf to 1
- How do we detect that a link is 'broken'? At layer 2 (no hello messages received for some time, or attempts to retransmit a frame exceeds the MAC protocol threshold) or at layer 3 (do not receive periodic updates by a neighbor)
- Link cost increase (1→ inf):
  - The nodes incident to that link (A,B) discover it (see above)
  - Routes going through that link get assigned an inf cost in nodes A and B routing tables
  - A new sequence number is generated by the mobile node. Mobile nodes different from the destination use odd SN, the destination even SN.
  - Updates with routes with infinite cost are immediately transmitted by nodes
- Link cost decrease (inf→1):
  - Immediately transmits updates

Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing

- When a node receives updates it sees if costs to reach the different destinations can be improved:
  - routes with more recent sequence numbers to a given destination are used
  - if more routes available with the same SN the shortest is used
- Newly recorded routes are scheduled for immediate advertisement (inf→ finite value)
- Routes with improved metric are scheduled for advertisement at a time which depends on the estimated average settling time for routes to that particular destination (based on previous history) → delayed advertisements to decrease the overall overhead
- As soon as a route cost changes the node may delay informing its neighbors but immediately starts using the new information for its forwarding

Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) Routing-Correctness

- Assuming routing tables are stable and a change occurs
  - let G(x) denotes the routes graph from the sources to x BEFORE the change (assume no loop)
  - change occurs at *i* when 1) the link from *i* to its parent p(*i*) in G(x) breaks → *i* sets to inf that route (no loop can occur) 2) node *i* receives from one of its neighbors k a route to x with sequence number SN^x_k and metric m which is selected to replace the current metric *i* has to reach x (this occurs only if SN^x_k greater than the previous SN I had stored Sn^x_i or if the two SN are equal but the new route has a lower hop cost → in the first case if selecting k leads to a loop then $SN^x_k <= Sn^x_i$ which is a contradiction, in the second case comes from the observation reduction in the costs do not bring to loops).

OLSR

- Optimized Link State Routing (OLSR) is a link state protocol for MANETs
  - suited for large and dense ad hoc networks
- The key concept is to decrease the overhead of flooding by means of identifying a subset of nodes (multipoint relays) in charge of forwarding the information during the flooding process
  - **Multipoint relay Y**: a node selected by at least one of its 1-hop neighbors (say node X) to relay all valid broadcast information it receives from X (the broadcast information is valid if it is not expired and not duplicate).
    - ✓ MPR(X)=set of multipoint relays of node X
    - ✓ neighbors of node X which are not in MPR(X) receive and store the broadcast messages transmitted by X but DO NOT retransmit them
  - A node X which has selected a neighbor Y as multi-point relay is called a **multipoint relay selector** of node Y
- Requires only partial link state to be flooded
  - links from MPR to their selectors must be declared
    - ✓ enough to ensure routes to each destination can be found
  - additional link state information MAYBE advertised

OLSR- other features

- The protocol is fully distributed
- Proactive approach: routes are always available when needed
- Other features:
  - Time between updates can be tuned to increase reactivity to topological changes
  - does not require reliable transmission
    - ✓ some losses are tolerated ← needed info are periodically transmitted
      - OLSR control packets are embedded in UDP datagrams
    - ✓ sequenced delivery of the messages is not needed ← proper recostruction of the sequence is possible due to use of sequence numbers
  - support to other MANET related issues
    - ✓ Sleep mode operation
    - ✓ Multicasting

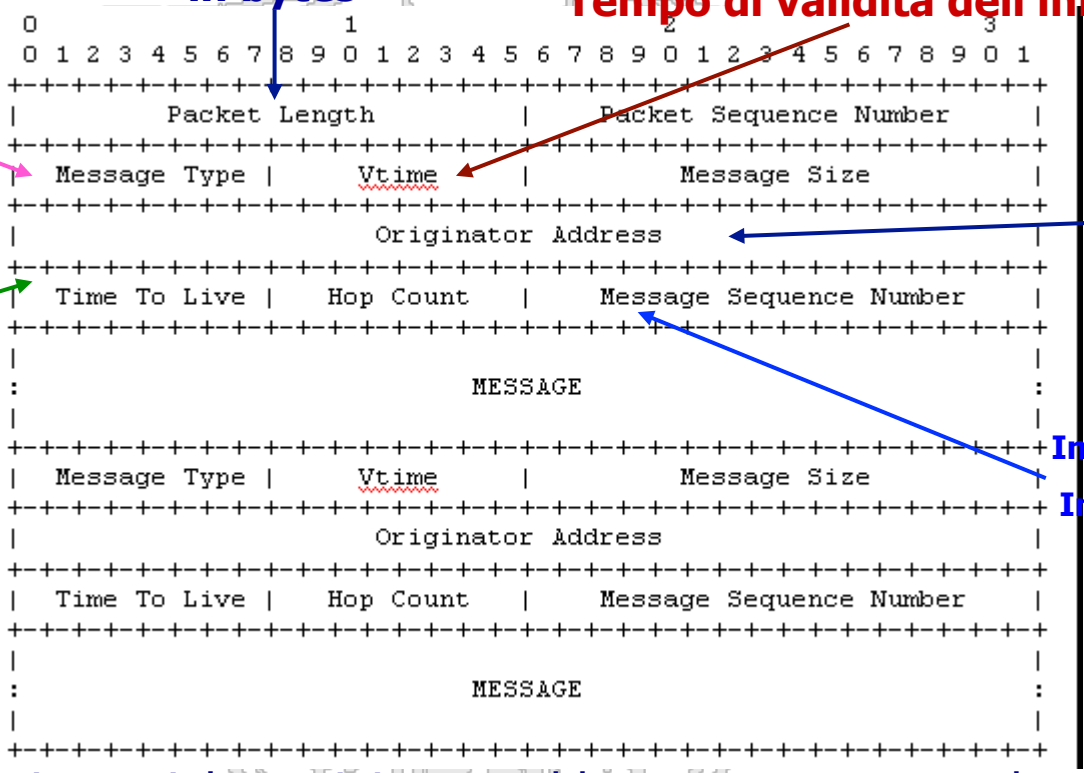
Packet format

in bytes

Tempo di validità dell'informazione

Es: Hello messages
Topology declaration
messages

Num. Max di hop
che un messaggio
può attraversare



Sorgente del
messaggio non
chi lo sta
trasmettendo

Inserito dall'originator

Incrementato ad ogni
messaggio inviato

Ogni nodo mantiene triple <originator address, sequence number, se il messaggio e' stato già inviato> relative a messaggi ricevuti recentemente. In questo modo i duplicati vengono scartati

Sono scartati anche pacchetti con TTL a zero o non coerenti con le specifiche

Hello messages

- Gli hello messages servono a
  - verificare se i link sono ancora su (link sensing)
    - ✓ se non si riceve un hello dal vicino entro un timeout si assume che il link non sia più attivo
    - ✓ scambiare con i vicini il proprio elenco di vicini
      - consente ai nodi di aggiornare le in formazioni relative al loro vicinato a due hop
        - » Serve per poter individuare i multipoint relay

How MPRs are selected

- Each node X selects its MPRs among its one hop neighbors
- The set is selected to cover node X 2-hop neighborhood
  - MPR(X) is an arbitrary subset of node X one hop neighbors such that each node z in node X's two hop neighborhood have a neighbor in MPR (X)
    - ✓ can be selected with a greedy protocol
      - MPR(X)=null, C(X)= vicinato a due hop da X
      - per ciascun vicino Y del nodo X si calcola il suo degree D (Y) escludendo il vicinato a 1-hop di X e X stesso
      - si inserisce nell'insieme MPR(X) un nodo Y se è l'unico vicino di X in grado di coprire un nodo a due hop da X
        - » C(X)=C(X) \ {nodi coperti da Y}
      - fino a quando C(X)=null
        - » inserisci in MPR(X) il vicino di X che consente di coprire più nodi rimasti scoperti in C(X) (a parità si seleziona in base al degree D)
        - » C(X)=C(X) \ {nodi coperti dal vicino selezionato}
  - The smaller MPR(X) the less control overhead exchanged

How are control packets disseminated

Upon receiving a message *m* at node Y

- If the received message is not a duplicate, is valid and has a non zero TTL
  - if it is received by an MPR selector of Y
    - ✓ retransmit ***m***
      - ***reduce by one the message TTL***
      - ***increase by one the message hop count***
      - ***broadcast on all node Y interfaces***
    - ✓ update or create the entry for the message in the duplicate set (→upon receiving the same message the node can identify it was already received and retransmitted→ can be discarded)

SAPIENZA
UNIVERSITÀ DI ROMA

How are data packets forwarded

- Information on topology are disseminated to all the network nodes
  - in an efficient way
    - ✓ exploiting the backbone of multipoint relays
    - ✓ limiting as much as possible topology information
- Each node then locally runs a shortest path algorithm to determine paths to the different destination
  - fills a routing table
  - upon reception of a data packet forwarding is performed according to the routing table

Why traditional approaches have limits?

- Proactive protocols are costly in terms of overhead (the bandwidth and energy are critical resources)
- The cost of maintaining routes updated may not make sense in an environment in which
  - Medium-high mobility
  - Medium-high dynamicity (awake/asleep states)

  Motivate frequent changes in the the optimum route (requiring updates) while
  - Traffic is generally low (so the cost of maintaining always updated routes is not balanced by their use)

  If this is the scenario what can we do?

SAPIENZA
UNIVERSITÀ DI ROMA

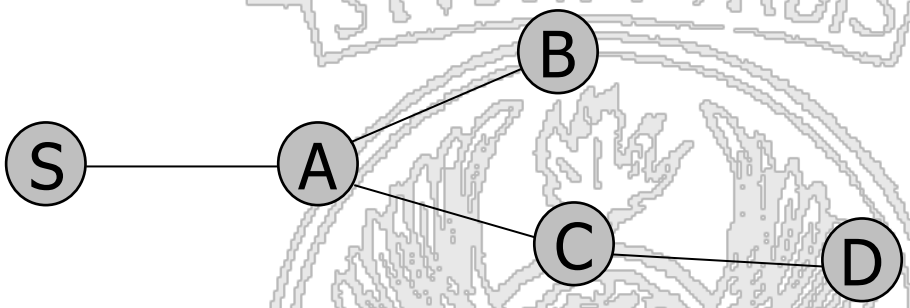
2 Primary Approaches (cont.)

- Reactive (on-demand)
  - Source build routes on-demand by "flooding"
  - Maintain only active routes
  - Route discovery cycle
  - Typically, less control overhead, better scaling properties
  - Drawback: route acquisition latency
  - Example: AODV, DSR

Ad hoc On-Demand Distance Vector (AODV) Routing

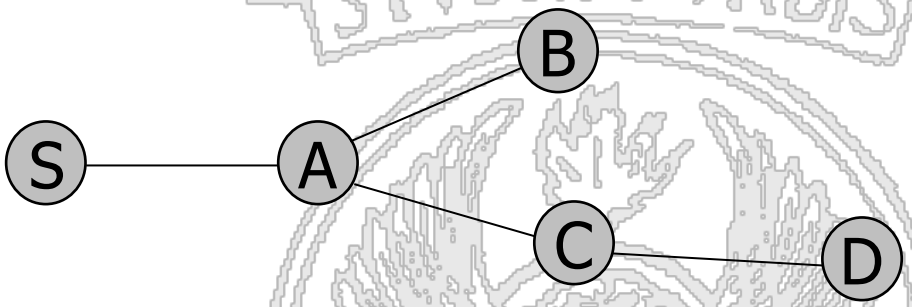
- Reactive (nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchange; a node does not have to discover/maintain a route to a destination till it is on a path to it or has to send messages to it)
- *Route discovery cycle* used for route finding
- Maintenance of *active* routes
- Sequence numbers used for loop prevention and as route freshness criteria
- Descendant of DSDV (standard distance vector approach mapped to ad hoc networks), in AODV no periodic updates but pure on-demand operation.
- Provides unicast and multicast communication

AODV: Route Discovery



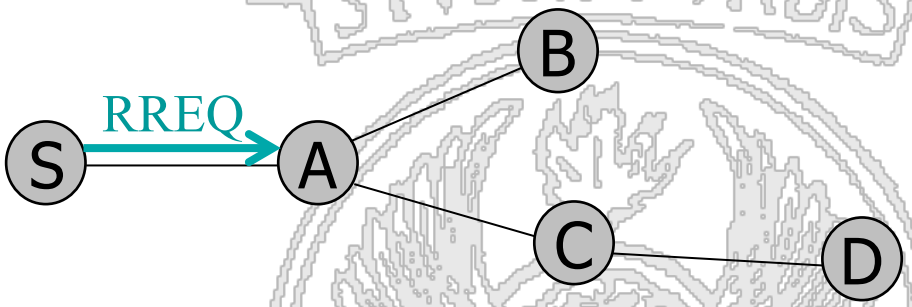
1. Node *S* needs a route to *D AND does not have routing info for it in its table*

AODV: Route Discovery



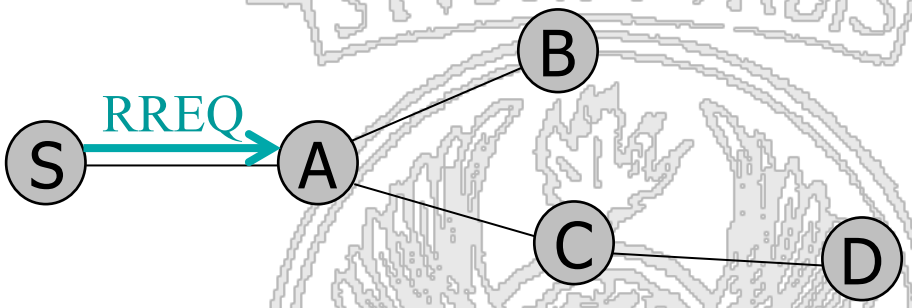
1. Node *S* needs a route to *D*
2. Creates a Route Request (RREQ)
   - Enters *D*'s IP addr, seq#,
   - *S*'s IP addr, seq#
   - hopcount (=0), broadcast ID

AODV: Route Discovery



1. Node *S* needs a route to *D*
2. Creates a Route Request (RREQ)
   - Enters *D*'s IP addr, seq#,
   - *S*'s IP addr, seq#
   - hopcount (=0), broadcast ID
3. Node *S* broadcasts RREQ to neighbors

AODV: Route Discovery

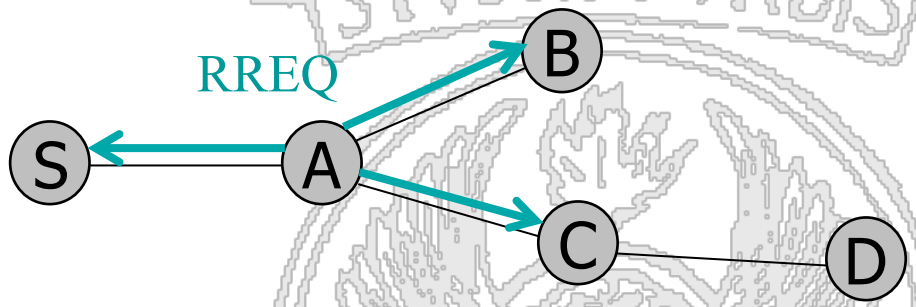


4. Node *A* receives RREQ
   - Makes reverse route entry for *S*

   dest=*S*, nexthop=*S*, hopcnt=1,expiration time for reverse path

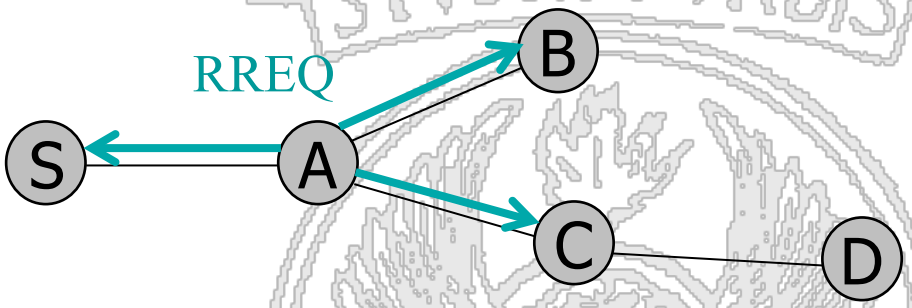
   Source node, Source node SN,D,broadcastID also maintained
   - It has no route to *D*, so it rebroadcasts RREQ (hopcount increased)
   - If it has already received that request (same source and broadcast ID) it discards the RREQ
   - if it knows a valid path to D it will send back a reply to the source

AODV: Route Discovery



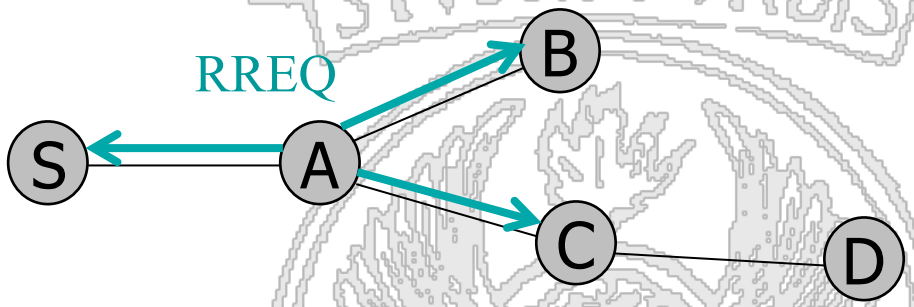
4. Node *A* receives RREQ
   - Makes reverse route entry for *S*
     dest=*S*, nexthop=*S*, hopcnt=1
   - It has no route to *D*, so it rebroadcasts RREQ

AODV: Route Discovery



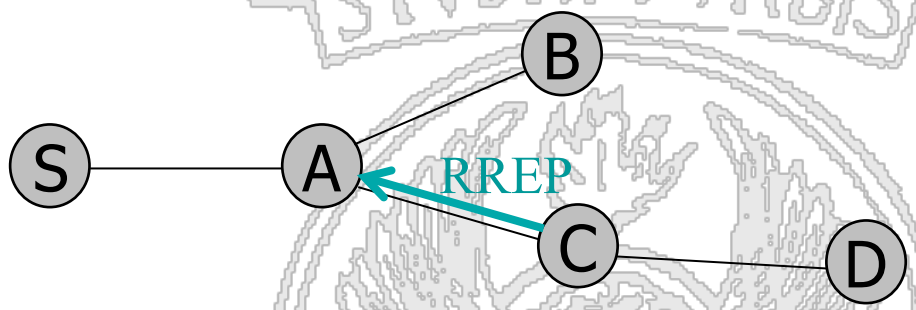
5. Node *C* receives RREQ
   - Makes reverse route entry for *S*
     dest=*S*, nexthop=*A*, hopcnt=2
   - It has a route to *D*, and the seq# for route to *D* is >= *D*'s seq# in RREQ

AODV: Route Discovery



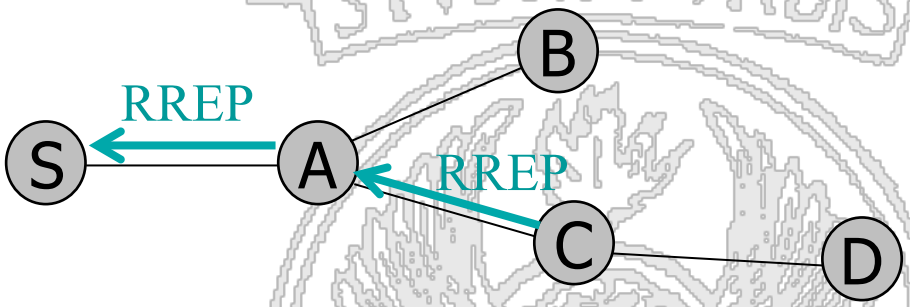
5. Node *C* receives RREQ (cont.)
   - *C* creates a Route Reply (RREP)
     Enters *D*'s IP addr, seq#
     S's IP addr, hopcount to *D* (= 1), lifetime of the forward route
   - Unicasts RREP to *A*

AODV: Route Discovery



5. Node *C* receives RREQ (cont.)
   - *C* creates a Route Reply (RREP)
     Enters *D*'s IP addr, seq#
     S's IP addr, hopcount to *D* (= 1)….
   - Unicasts RREP to *A*

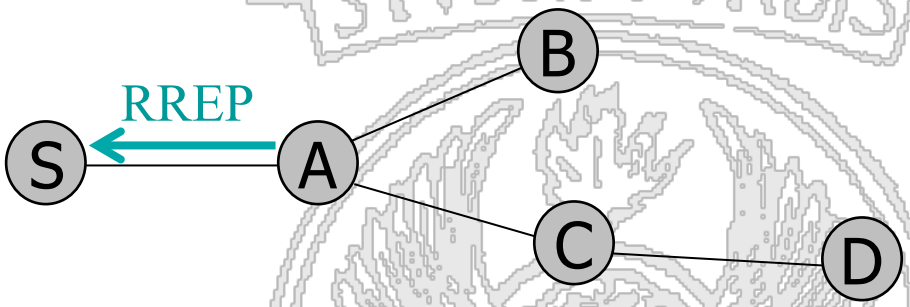
AODV: Route Discovery



6. Node *A* receives RREP
   - Makes forward route entry to *D*

     dest = *D*, nexthop = *C*, hopcount = 2
   - Unicasts RREP to *S*

AODV: Route Discovery



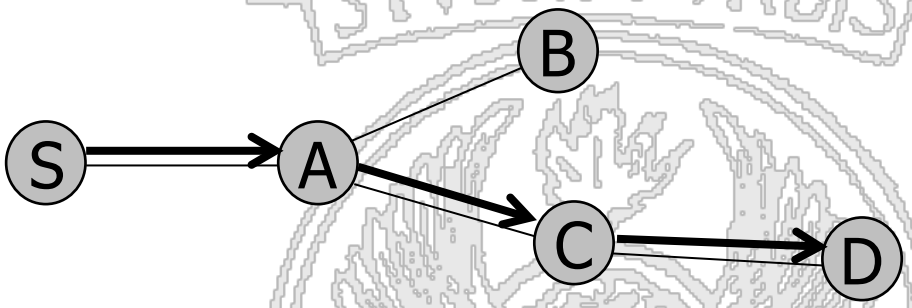
7. Node *S* receives RREP
   - Makes forward route entry to *D*

     dest = *D*, nexthop = *A*, hopcount = 3

Also the latest SN of the destination is updated when receiving the RREP

Nodes not along the path determined by the RREP will timeout after ACTIVE_ROUTE_TIMEOUT (3000ms) and will delete the reverse pointer

AODV: Route Discovery



7. Node *S* receives RREP
   - Makes forward route entry to *D*

     dest = *D*, nexthop = *A*, hopcount = 3
   - Sends data packets on route to *D*

SAPIENZA
UNIVERSITÀ DI ROMA

What if....

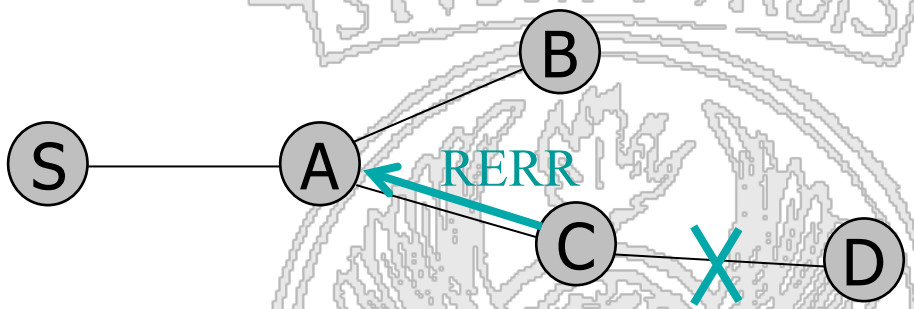
- A node receives further RREPs for the same request? (e.g. more neighbors of a node had paths to the destination in cache)?
  - upon reception of another RREP the node updates its routing information and propagates the RREP only if the RREP contains either a greater destination SN, or the same destination SN with a smaller hopcount

SAPIENZA
UNIVERSITÀ DI ROMA

Other info maintained

- Each node maintains the list of active neighbors, neighbors sending to a given destination through it
  - useful for route maintenance
- Routing table entries: dest,next hop, hopcount, dest SN, active neighbors for this route, expiration time for route table entry (updates each time the route is used for transmitting data → routes entries are maintained if the route is active)

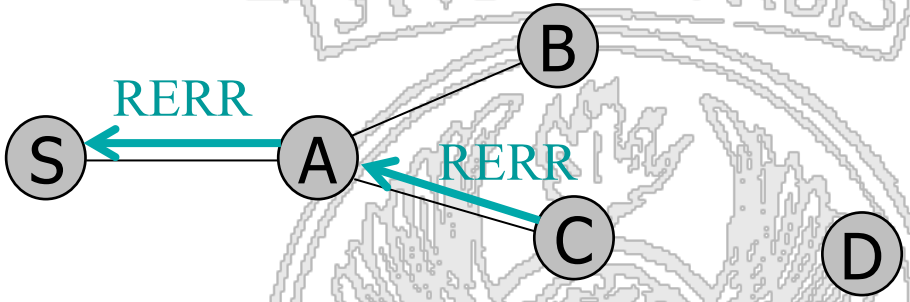
AODV: Route Maintenance



1. Link between *C* and *D* breaks
2. Node *C* invalidates route to *D* in route table
3. Node *C* creates Route Error (RERR) message
   - Lists all destinations which are now unreachable
   - Sends to upstream neighbors
   - Increases of one the SN of the destination

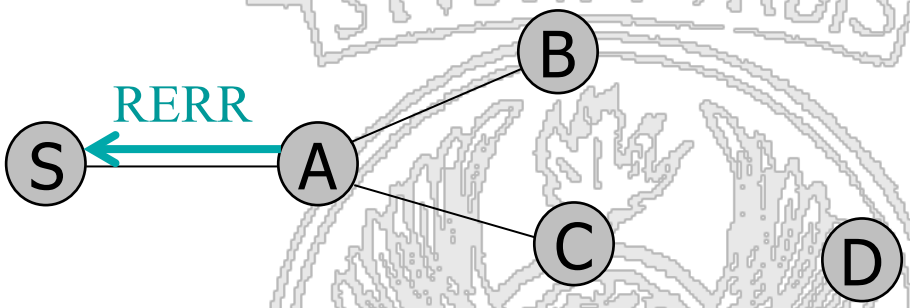
SAPIENZA
UNIVERSITÀ DI ROMA

AODV: Route Maintenance



4. Node *A* receives RERR
   - Checks whether *C* is its next hop on route to *D*
   - Deletes route to *D*
   - Forwards RERR to *S*

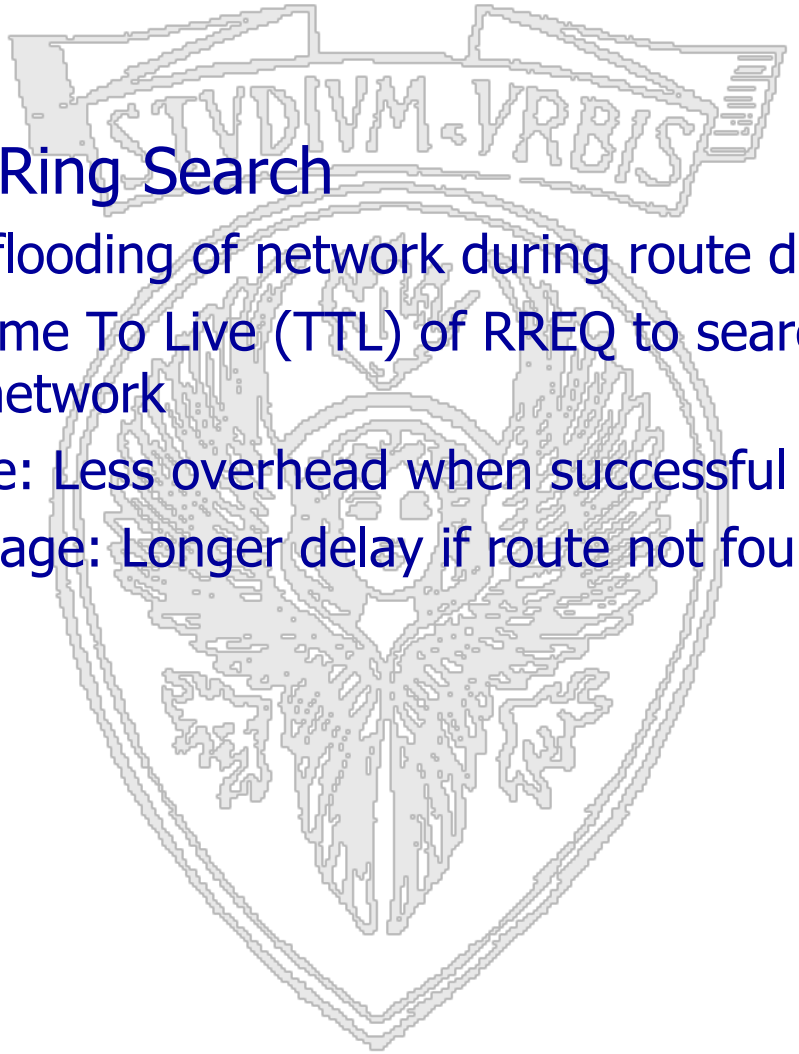
AODV: Route Maintenance



5. Node *S* receives RERR
   - Checks whether *A* is its next hop on route to *D*
   - Deletes route to *D*
   - Rediscovers route if still needed (in that case it sends a RREQ with a SN which is equal to the last known destination Sequence Number +1)

SAPIENZA
UNIVERSITÀ DI ROMA

AODV: Optimizations



- Expanding Ring Search
  - Prevents flooding of network during route discovery
  - Control Time To Live (TTL) of RREQ to search incrementally larger areas of network
  - Advantage: Less overhead when successful
  - Disadvantage: Longer delay if route not found immediately

AODV: Optimizations (cont.)

- Local Repair
  - Repair breaks in active routes locally instead of notifying source
  - Use small TTL because destination probably hasn't moved far
  - If first repair attempt is unsuccessful, send RERR to source
  - Advantage: repair links with less overhead, delay and packet loss
  - Disadvantage: longer delay and greater packet loss when unsuccessful

SAPIENZA
UNIVERSITÀ DI ROMA

AODV: Summary

- Reactive/on-demand
- Sequence numbers used for route freshness and loop prevention
- Route discovery cycle
- Maintain only active routes
- Optimizations can be used to reduce overhead and increase scalability