EE5132 / EE5023

Routing Protocols

Chapter 3

Routing

- Routing involves determining a path from a source node to a destination node through the network based on criteria such as
  - minimum no. of hops
  - least cost, etc.

- There are two main families of routing algorithms:
  - **Link state routing**, based on Dijkstra's algorithm
  - **Distance-vector routing**, based on the Bellman-Ford algorithm

- **Overview of Lecture**
- Routing Concepts
- Dijkstra's Algorithm
- Bellman-Ford Algorithm
- Infrastructure-free or Ad Hoc Networks
- Routing in Mobile Ad Hoc Networks (MANETs)
  - Table-driven Routing Protocols
    - Destination Sequenced Distance-Vector (DSDV)
  - Source Initiated On-demand Routing Protocols
    - Dynamic Source Routing Protocol (DSR)
    - Ad hoc On-demand Distance Vector Routing (AODV)
  - Hybrid Routing Protocol
    - Zone Routing Protocol (ZRP)

Routing Concepts

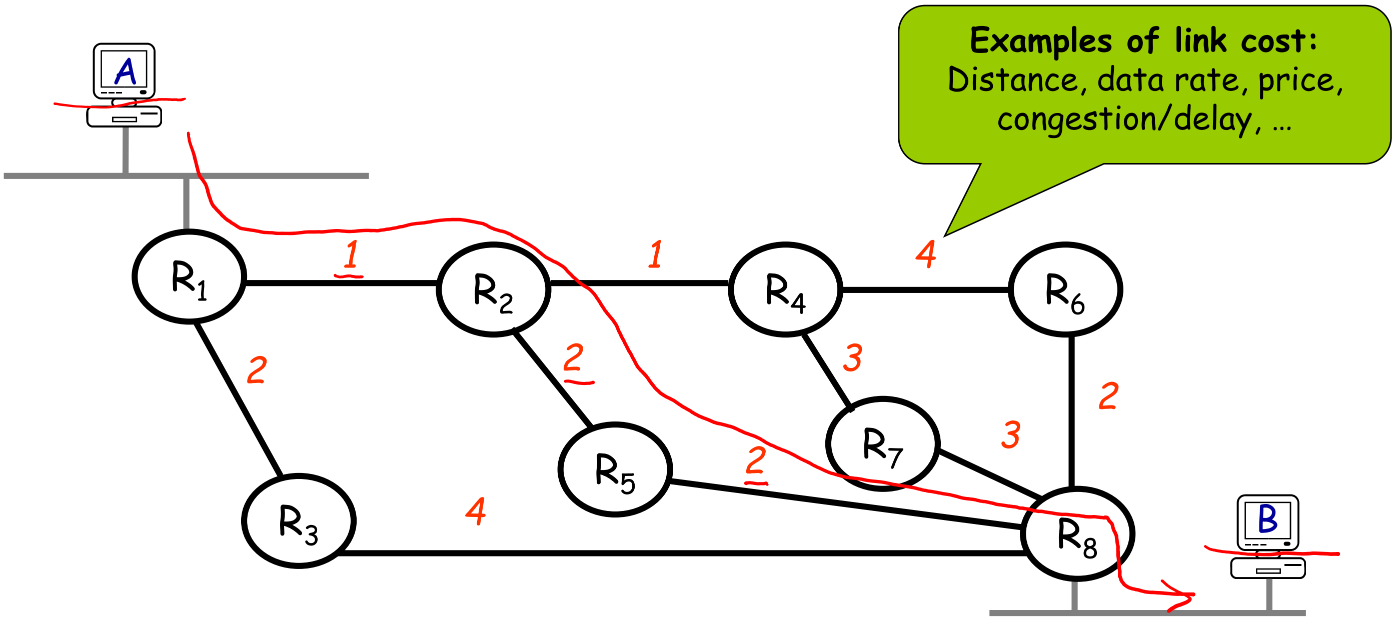
- Performance criteria used for selection of route:
  - Minimum hop
  - Least cost
- To select a path involves 2 issues:
  - The path selection algorithm itself
    - The cost of a path is a function of: *hop count* and *available bandwidth*. Each interface has associated a metric which indicates the amount of remaining available bandwidth.
    - This metric is combined with the hop count to provide a cost value, whose goal is to *pick the path with the minimum number of hops supporting the requested bandwidth*.
    - When several paths are available, the path whose availability is maximal is selected. This way the balance load is maximized. Observe that this algorithm has double objective optimization.
  - When the algorithm is actually invoked. 2 options:
    - On-demand, i.e. computation is triggered for each new request. Could be computationally expensive
    - Using some form of pre-computation. This option amortizes computational cost over multiple requests, but each computation instance is usually more expensive because paths to all destination should be recomputed. Also the final accuracy of the selected path may be lower. Another important issue is when pre-computation should take place. There are two options:
      - Periodic re-computation
      - Pre-computation after 'N' number of updates have been received

Least Cost Algorithms

- Basis for routing decisions
  - Can minimize hop with each link cost
  - Can have link cost inversely proportional to capacity
- Given network of nodes connected by bi-directional links
- Each link has a cost in each direction
- Define cost of path between two nodes as sum of costs of links traversed
- For each pair of nodes, find a path with the least cost
- Link costs in different directions may be different

Example Network

<u>Objective:</u> Determine the route from A to B that minimizes the path cost.



Examples of link cost:
Distance, data rate, price, congestion/delay, …

I. LINK STATE

Dijkstra's Algorithm Definitions

- Find shortest paths from given source node to all other nodes, by developing paths in order of increasing path length
- *N* = set of nodes in the network
- *s* = source node
- *T* = set of nodes so far incorporated by the algorithm
- *w*(*i*, *j*) = link cost from node *i* to node *j*
  - *w*(*i*, *i*) = 0
  - *w*(*i*, *j*) = ∞ if the two nodes are not directly connected
  - *w*(*i*, *j*) ≥ 0 if the two nodes are directly connected

- *L*(*n*) = cost of least-cost path from node *s* to node *n* currently known
  - At termination, *L*(*n*) is cost of least-cost path from *s* to *n*

Edsger W. Dijkstra

Dijkstra's Algorithm Method

- Step 1 [Initialization]
  - *T* = {*s*} Set of nodes so far incorporated consists of only source node
  - *L*(*n*) = *w*(*s*, *n*) for *n* ≠ *s*
  - Initial path costs to neighboring nodes are simply link costs
- Step 2 [Get Next Node]
  - Find neighboring node *x* not in *T* with least-cost path from *s*
  - Incorporate node *x* into *T*
  - Also incorporate the edge that is incident on node *x* and a node in *T* that contributes to the path
- Step 3 [Update Least-Cost Paths]
  - *L*(*n*) = min[*L*(*n*), *L*(*x*) + *w*(*x*, *n*)] for all *n* ∉ *T*
  - If latter term is minimum, path from *s* to *n* is path from *s* to *x* concatenated with edge from *x* to *n*
- Algorithm terminates when all nodes have been added to *T*

x? 0

0

02

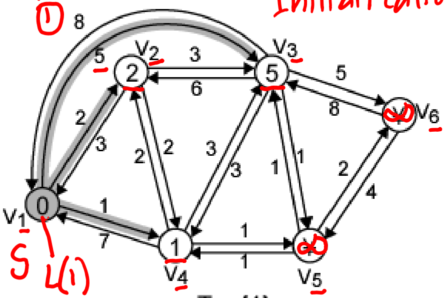
Dijkstra's Algorithm Notes

- At termination, value *L*(*x*) associated with each node *x* is cost (length) of least-cost path from *s* to *x*.
- In addition, *T* defines least-cost path from *s* to each other node
- One iteration of steps 2 and 3 adds one new node to *T*
  - Defines least cost path from *s* to that node

Example of Dijkstra’s Algorithm

End of ①

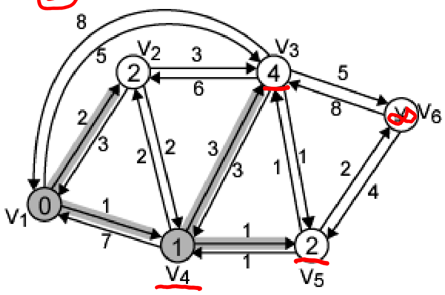
Initialization



T = {1}

At the end of Iteration 1

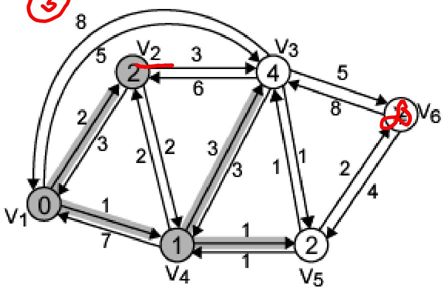
End of ②



T = {1, 4}

$x=4$

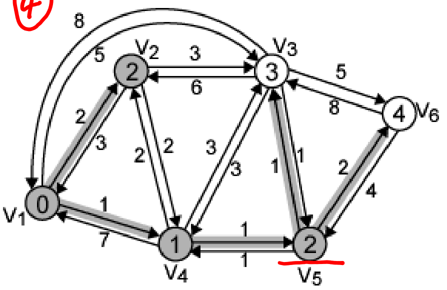
③



T = {1, 2, 4}

$x=2$

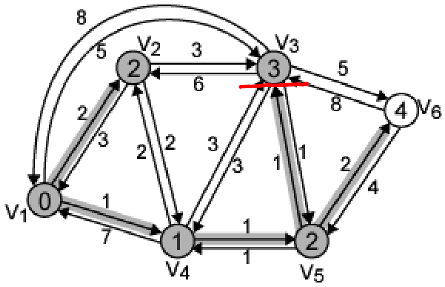
④



T = {1, 2, 4, 5}

$x=5$

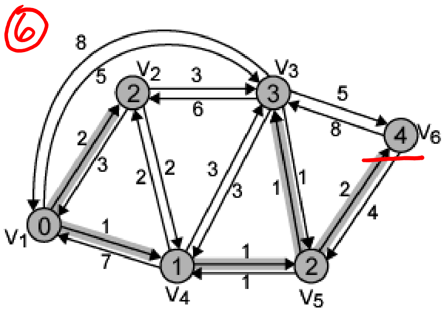
⑤



T = {1, 2, 3, 4, 5}

$x=3$

⑥



T = {1, 2, 3, 4, 5, 6}

$x=6$

Results of Example Dijkstra’s Algorithm

Iteration	*T*	*L*(2)	Path	*L*(3)	Path	*L*(4)	Path	*L*(5)	Path	*L*(6)	Path
1	{1}	2	1–2	5	1-3	1	1–4	∞	-	∞	-
2 $x=4$	{1,4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-
3 $x=2$	{1, 2, 4}	2	1–2	4	1-4-3	1	1–4	2	1-4–5	∞	-
4 $x=5$	{1, 2, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6
5 $x=3$	{1, 2, 3, 4, 5}	2	1–2	3	1-4-5–3	1	1–4	2	1-4–5	4	1-4-5–6
6 $x=6$	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4–5	4	1-4-5-6

See also:
http://weierstrass.is.tokushima-u.ac.jp/ikeda/suuri/dijkstra/Dijkstra.shtml

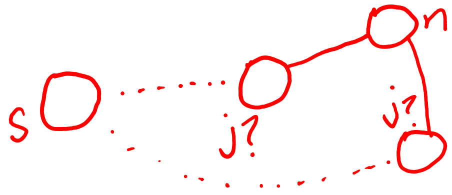
II. DISTANCE VECTOR

Bellman-Ford Algorithm Definitions

- Find shortest paths from given node, subject to constraint that paths contain at most one link
- Find the shortest paths with a constraint of paths of at most two links
- And so on
- *s* = source node
- *w*(*i*, *j*) = link cost from node *i* to node *j*
  - $w(i, i) = 0$
  - $w(i, j) = \infty$ if the two nodes are not directly connected
  - $w(i, j) \geq 0$ if the two nodes are directly connected
- *h* = maximum number of links in path at current stage of the algorithm
- $L_h(n)$ = cost of least-cost path from *s* to *n* under constraint of no more than *h* links

Bellman-Ford Algorithm

- Step 1 [Initialization]
  - $L_0(n) = \infty$, for all $n \neq s$
  - $L_h(s) = 0$, for all *h*
- Step 2 [Update]
- For each successive $h > 0$
  - For each $n \neq s$, compute
  - $L_h(n)=\min_j[L_{h-1}(j)+w(j,n)]$



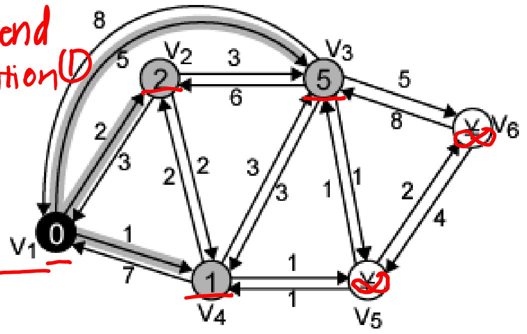
checks various directly connected nodes to n

- Connect *n* with predecessor node *j* that achieves minimum path cost
- Eliminate any connection of *n* with different predecessor node formed during an earlier iteration
- Path from *s* to *n* terminates with link from *j* to *n*

Example of Bellman-Ford Algorithm

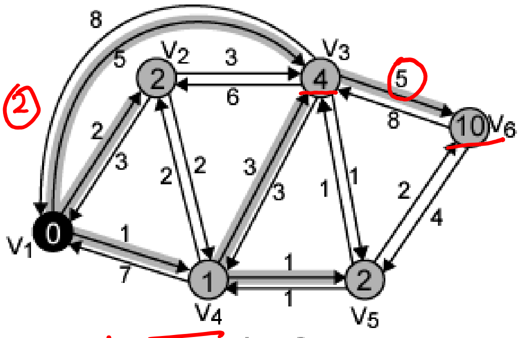
At the end of Iteration(1)

S



h = 1

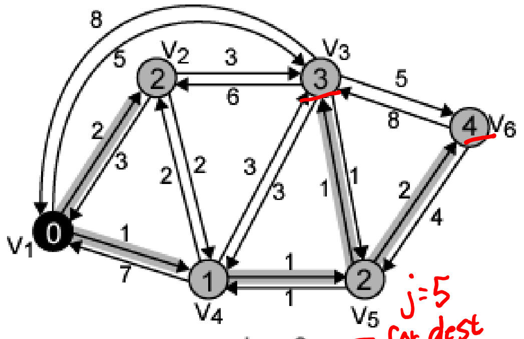
...(2)



h = 2

j=4 for dest node 3

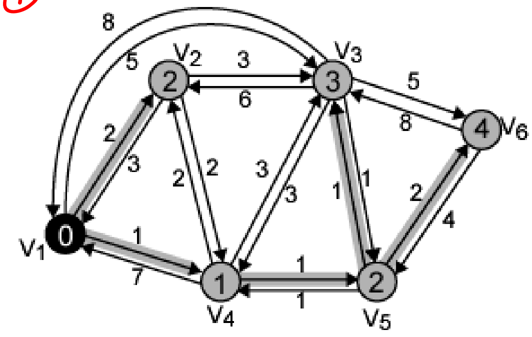
...(3)



h = 3

j=5 — for dest node 3&6

...(4)



h = 4

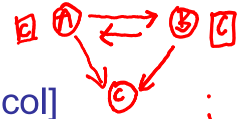
Results of Bellman-Ford Example

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

j

Comparison between B-F and Dijkstra's algorithms

Looping X!



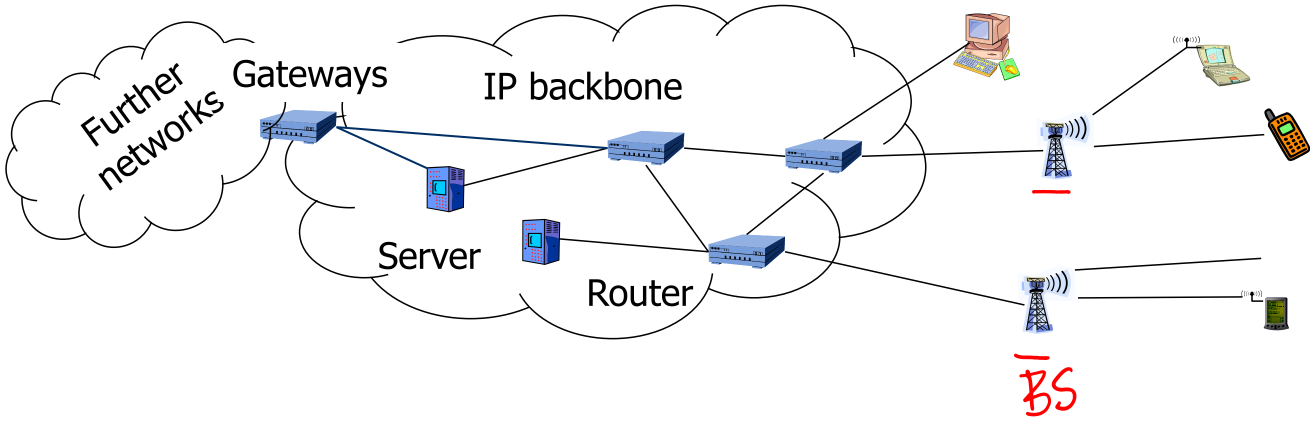
- Solutions from the two algorithms agree
- **Bellman-Ford algorithm**: [distance-vector protocol]
  - Each node bases its path calculation on knowledge of the cost to all directly connected neighbors plus the advertised costs for routes heard from these neighbors.
  - Easily susceptible to loops because the routers depend on information from neighbors, which might no longer be useful after a failure/stale Info
  - Has a distributed version that is fairly effective.

used in: DSDV AODV

- **Dijkstra's algorithm**: [link-state protocol] fails when link cost is -ve
  - Requires each node to have complete information about the entire topology.
  - Each router requires more memory to hold the link-state database and more processing capacity to run the algorithm.
  - Used in routing algorithms such as Open Shortest Path First (OSPF) (for wired networks) and Optimized Link State Routing (OLSR) (for mobile ad hoc networks).

Infrastructure-based Wireless Networks

- Typical wireless network: Based on infrastructure
  - e.g., cellular mobile network
  - Base stations connected to a wired backbone network
  - Mobile nodes communicate wirelessly to these base stations
  - Traffic between different mobile entities is relayed by base stations and wired backbone
  - Mobility is supported by switching from one base station to another
  - Backbone infrastructure required for administrative tasks

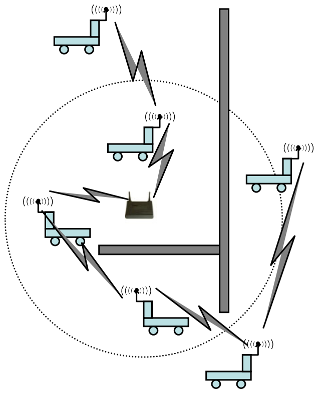


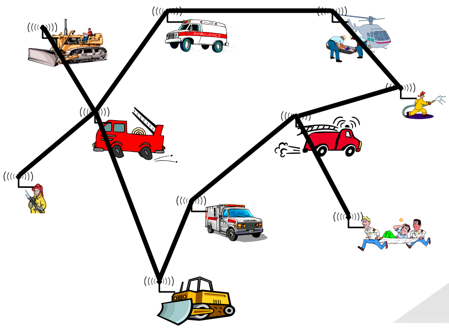
Infrastructure-based Wireless Networks – Limits?

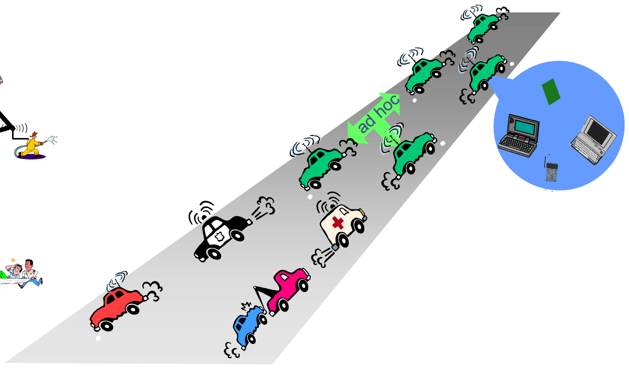
- What if …
  - No infrastructure is available? – e.g., in disaster areas
  - Too expensive/inconvenient to set up? – e.g., in remote, large construction sites
  - There is no time to set it up? – e.g., in military operations

Possible applications for Infrastructure-Free Networks

- Factory floor automation
- Disaster recovery
- Car-to-car communication



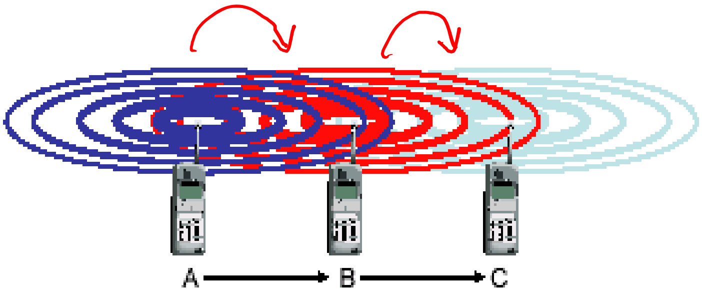




- Military networking: Tanks, soldiers, …
- Finding out empty parking lots in a city, without asking a server
- Search-and-rescue in an avalanche
- Personal area networking (watch, glasses, smartphone, medical appliance, …)
- …

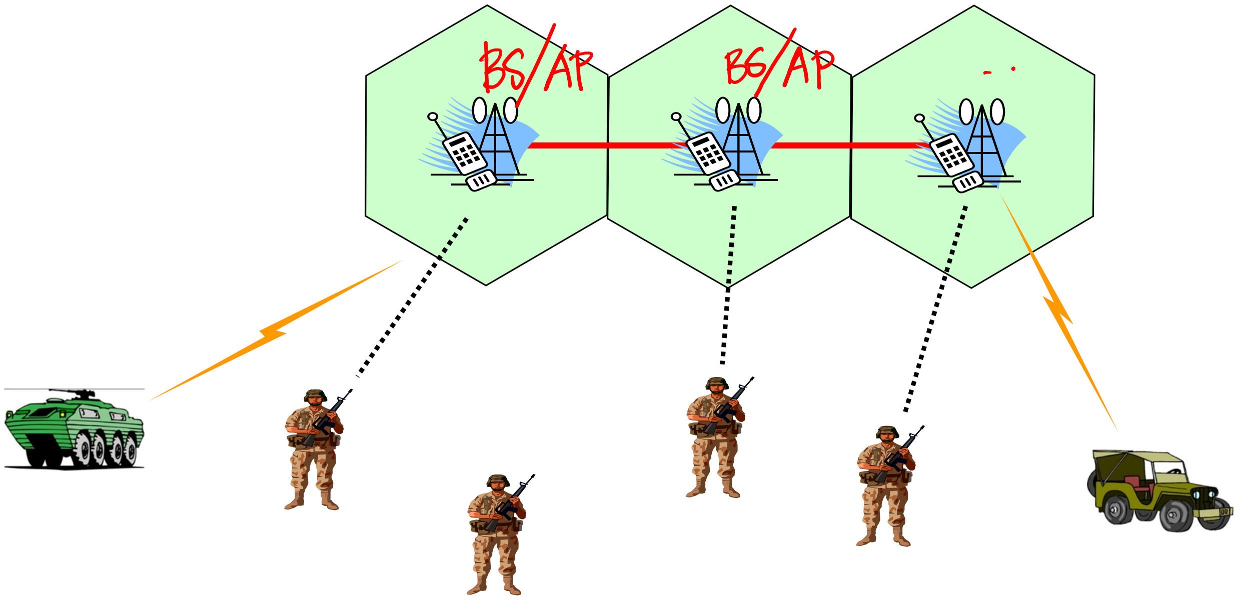
What are Ad Hoc Networks?

- Sometimes there is no infrastructure
  - Automated battlefield, disaster recovery
- Sometimes not every station can hear every other station
  - Data needs to be forwarded in a “multihop” manner



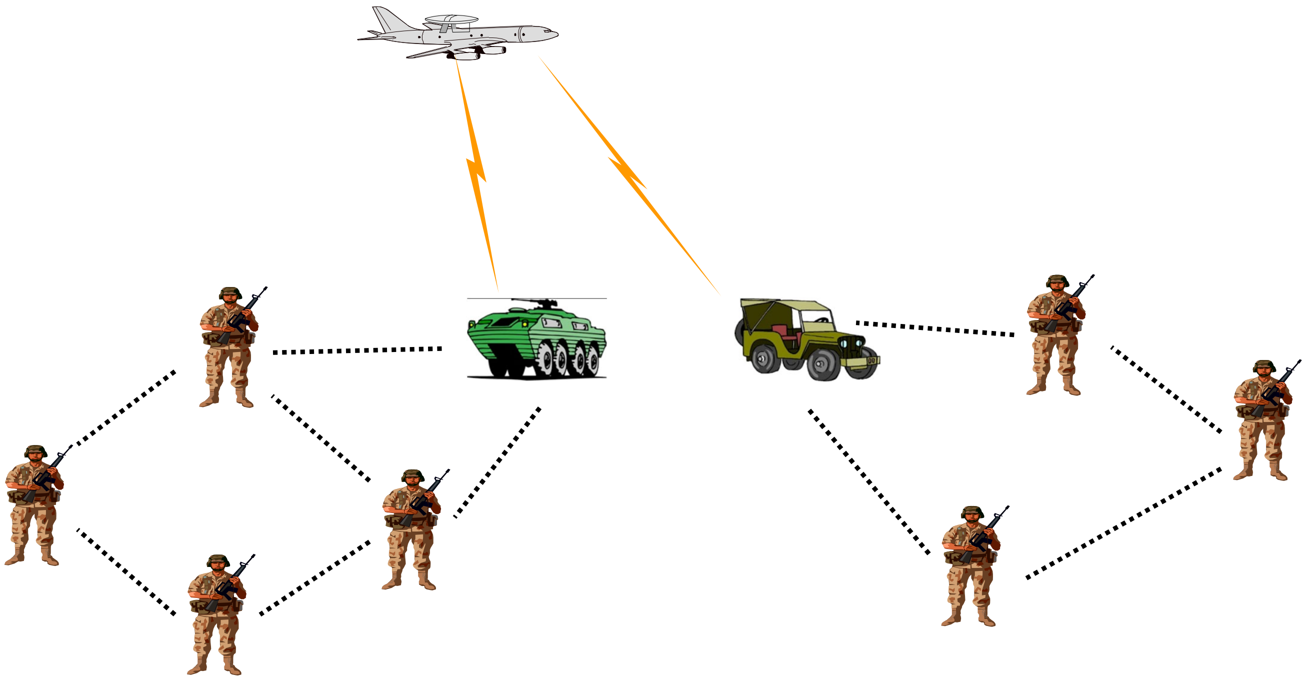
Infrastructure vs. Ad Hoc

Infrastructure Network (cellular or Hot spot)



Infrastructure vs. Ad Hoc

Ad hoc, Multihop Wireless Network



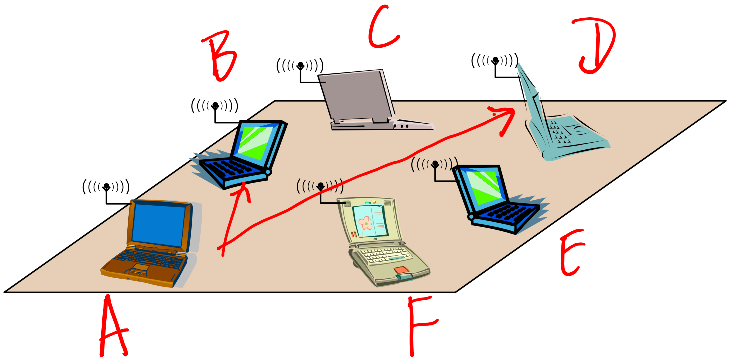
Ad Hoc Networks

- Ad hoc network:
  - A collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration.
- Significant differences to existing wired networks:
  - Wireless
  - Self-starting
  - No administrator
  - Cannot assume that every node is within communication range of every other node
  - Possibly quite dynamic topology of interconnections

(Wireless) Ad Hoc Networks

- Try to construct a network without infrastructure, using networking abilities of the participants
  - This is an ***ad hoc network*** – a network constructed "for a special purpose"

- Simplest example: Laptops in a conference room – a ***single-hop ad hoc network***



Challenges to Routing in MANETs (Mobile Ad Hoc NETworks)

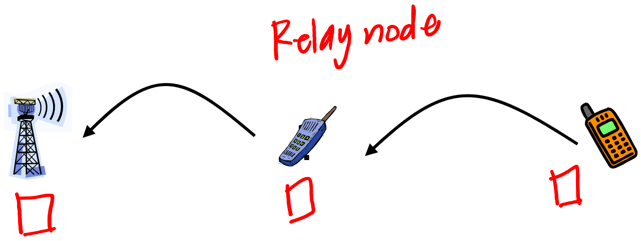
- Lack of a fixed infrastructure
  - Each node in the network must route messages towards their destination
- Nodes operate on battery power
  - Routing of messages may cause faster battery consumption, leading to node going offline
- Dynamic Topology
  - Nodes are constantly moving, leaving, or joining

MANET Protocol Considerations

- Simple, Reliable and Efficient
- Distributed but lightweight in nature
- Quickly adapt to changes in topology and traffic pattern
- Protocol reaction to topology changes should result in minimal control overhead
- Bandwidth efficient
- Mobility Management involving user location management and Hand-off management

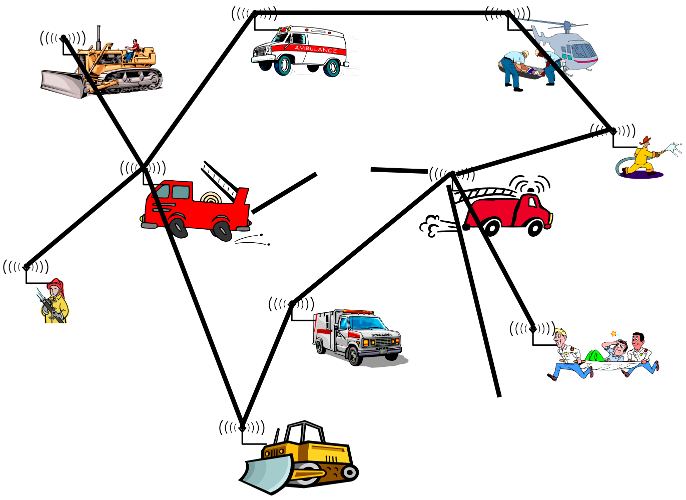
Limited range! Multi-hopping

- For many scenarios, communication with peers outside immediate communication range is required
  - Direct communication limited because of distance, obstacles, …
  - Solution: ***multi-hop network***



Mobility! Suitable, Adaptive Protocols

- In many (not all!) ad-hoc network applications, participants move around
  - In cellular network: simply hand over to another base station
- In ***mobile ad hoc networks*** **(*MANET*)**:
  - Mobility changes neighborhood relationships
  - Must be compensated for
  - e.g., routes in the network have to be changed
- Complicated by scale
  - Large number of such nodes difficult to support



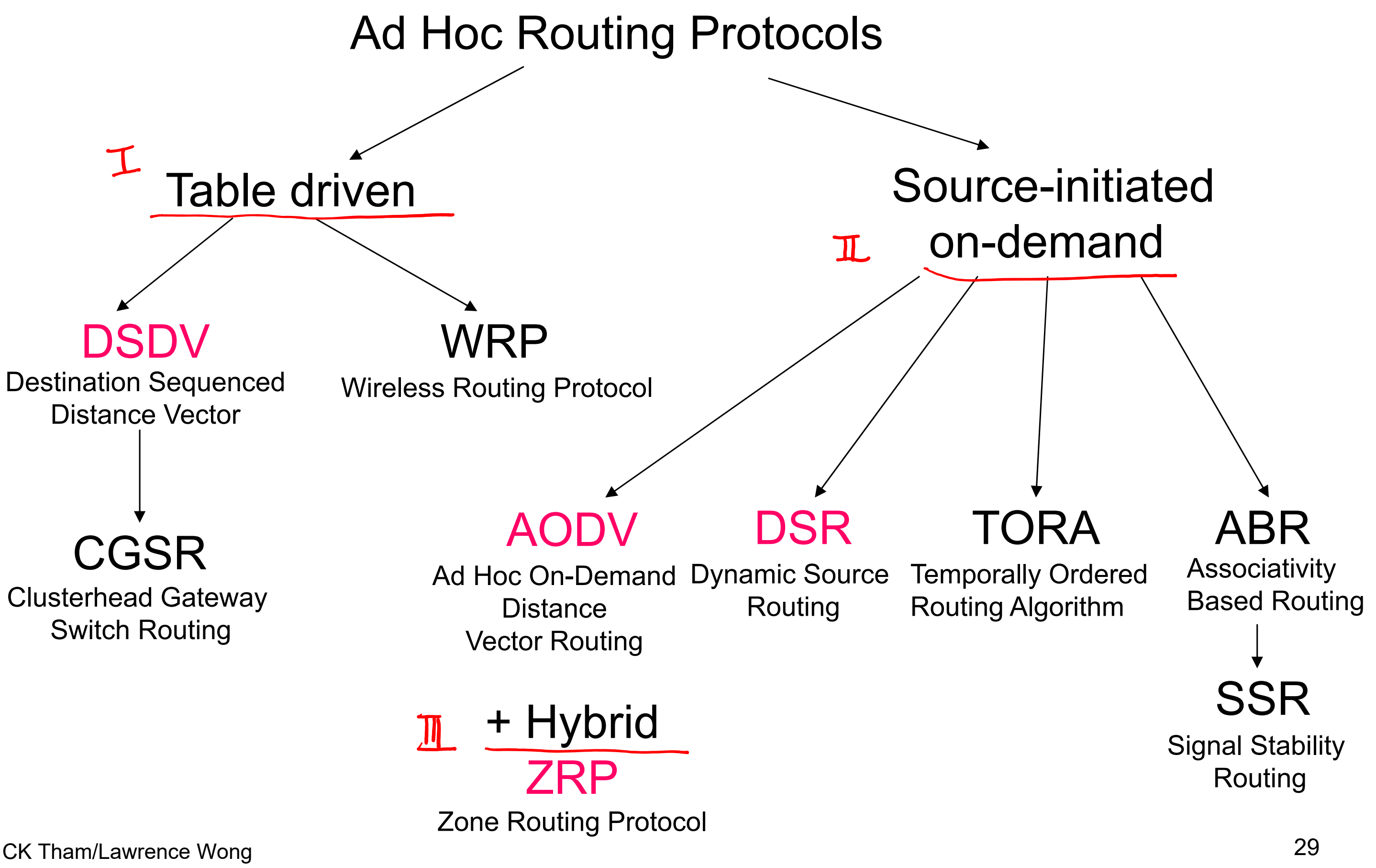
Three Categories of Protocols

II
- **Source Initiated On-Demand Routing Protocols**
  - Reactive
  - Create routes only when desired by the source node

I
- **Table-driven Routing Protocols**
  - Pro-active
  - Maintain consistent, up-to-date routing information from each node to every other node

III
- **Hybrid Routing Protocols**
  - Use pro-active protocol in local zone, use reactive protocol between zones

Existing Ad Hoc Routing Protocols



Ad Hoc Routing Protocols

I Table driven

DSDV
Destination Sequenced Distance Vector

CGSR
Clusterhead Gateway Switch Routing

WRP
Wireless Routing Protocol

II Source-initiated on-demand

AODV
Ad Hoc On-Demand Distance Vector Routing

DSR
Dynamic Source Routing

TORA
Temporally Ordered Routing Algorithm

ABR
Associativity Based Routing

SSR
Signal Stability Routing

III + Hybrid

ZRP
Zone Routing Protocol

Trade off

- Latency of route discovery
  - Proactive protocols may have lower latency since routes are maintained at all times
  - Reactive protocols may have higher latency because a route from *X* to *Y* will be found only when *X* attempts to send to *Y*

vs

- Overhead of route discovery/maintenance
  - Reactive protocols may have lower overhead since routes are determined only if needed
  - Proactive protocols can result in higher overhead due to continuous route updating

I.

Table-driven Routing Protocols

*

- **Destination Sequenced Distance-Vector (DSDV)**
  CE Perkins and P Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers", SIGCOMM '94, London, UK, 31 Aug – 2 Sep 1994, pp 234-244.
- **Wireless Routing Protocol (WRP)**
  S Murthy and JJ Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks", Journal of Mobile Networks and Applications, vol. 1, no. 2, Oct 1996, pp 183-197.
- **Clusterhead Gateway Switch Routing (CGSR)**
  CC Chiang, HK Wu, W Liu and M Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel", IEEE SICON, 1997.

Distance Vector (Distributed Bellman-Ford)

- Link costs may change over time
  - changes in traffic conditions
  - link failures
  - mobility

next hop to destination

- Each node maintains its own routing table
  - need to update table regularly to reflect changes in network

- Let D_i be the shortest distance from node i to a destination node
- *note*: there are usually a number of destinations, do separate computation for each destination

note: this is a flipped version of earlier B-F algorithm

For a particular destination: $D_i = \min_j [d_{ij} + D_j]$: update equation

- Each node i regularly updates the values of D_i using the update equation
  - each node maintains the values of d_{ij} to its neighbors, as well as values of D_j received from its neighbors
  - uses those to compute D_i and send new value of D_i to its neighbors
  - if no changes occur in the network, algorithm will converge to shortest paths in no more than N steps, when N is the no. of nodes

Destination Sequenced Distance-Vector (DSDV)

- Based on the Distance Vector (Distributed Bellman-Ford) algorithm
  - modified to work better in MANET
  - produces loop-free fewest hops path
- Each node maintains a routing table containing:
  - next hop towards each destination
  - a cost metric (distance in no. of hops) for the path to each destination
  - a destination sequence number that is created by the destination (sequence numbers help to distinguish new routes from stale ones, thereby avoiding formation of loops)
- Each node periodically forwards its routing table to its neighbors
  - each node increments and appends its own sequence number when sending its local routing table
  - this sequence number will be attached to route entries created for this node

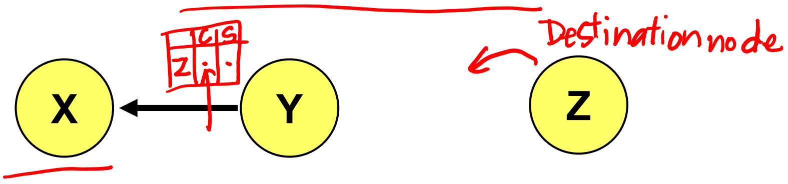
Refer to DSDV paper (e.g. MH_1)

Destination Sequenced Distance-Vector (DSDV)

- To minimize the routing updates:
  - Either **full dump** carrying all available routing information, or
  - Smaller **incremental packets** containing the change in information since last full dump
- A route broadcast contains:
  - Destination address
  - Number of hops required to reach destination
  - Sequence number of information received about the destination
- Updates are made when the received sequence number of a destination node is larger than current table entry, or if same, the metric is better (e.g., smaller hop count)

Destination Sequenced Distance-Vector (DSDV)

- Assume that X receives routing information from Y about a route to Z



  - S(X): destination sequence # for node Z as stored at node X
  - S(Y): destination sequence # for node Z sent by Y with its routing table to node X

- Node X takes the following steps:
  - 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)