OLSR: Optimized Link State Routing

Leonardo Maccari

leonardo.maccari@unitn.it

25/5/2015

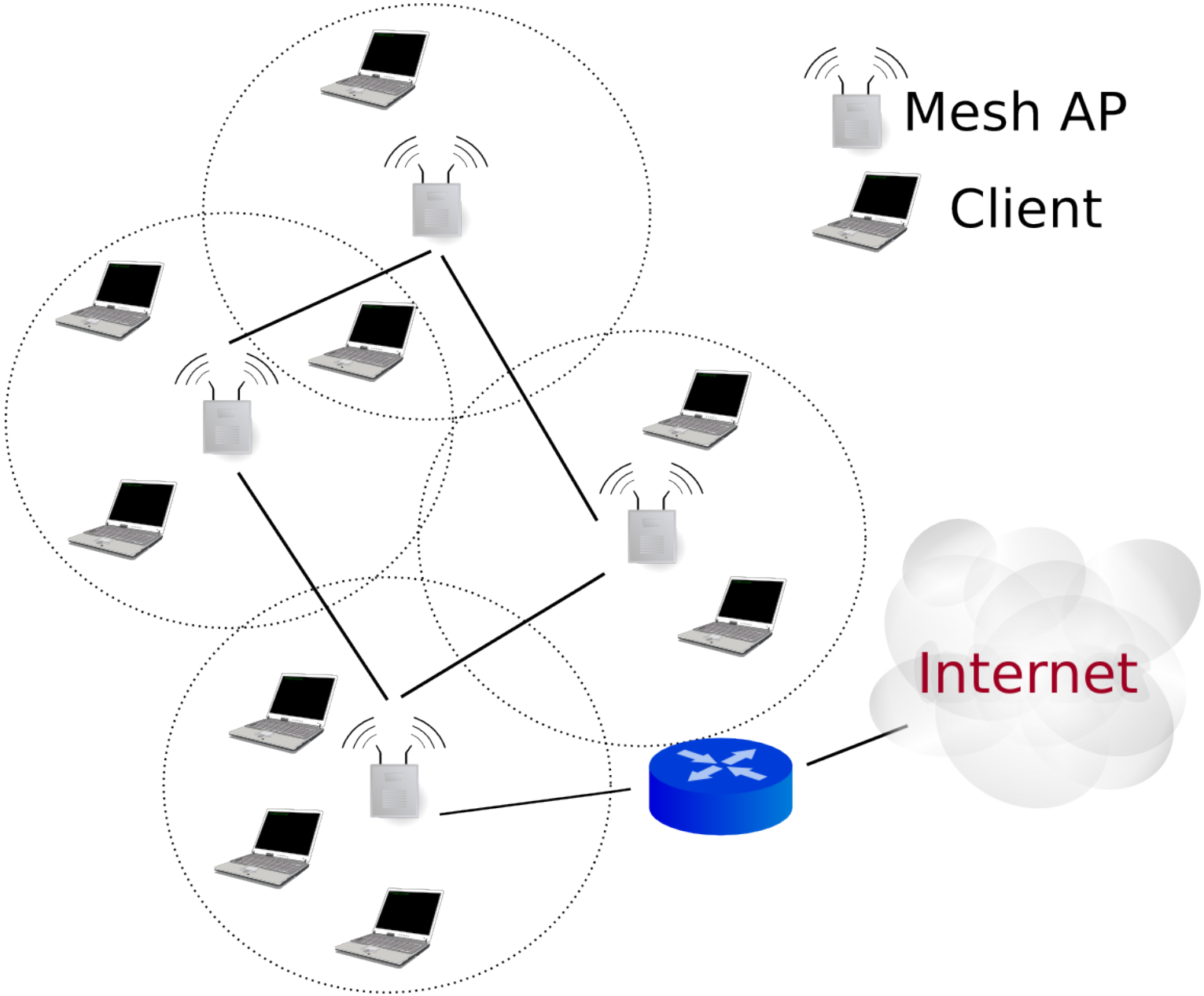
Contents

Wireless multi-hop distributed networks

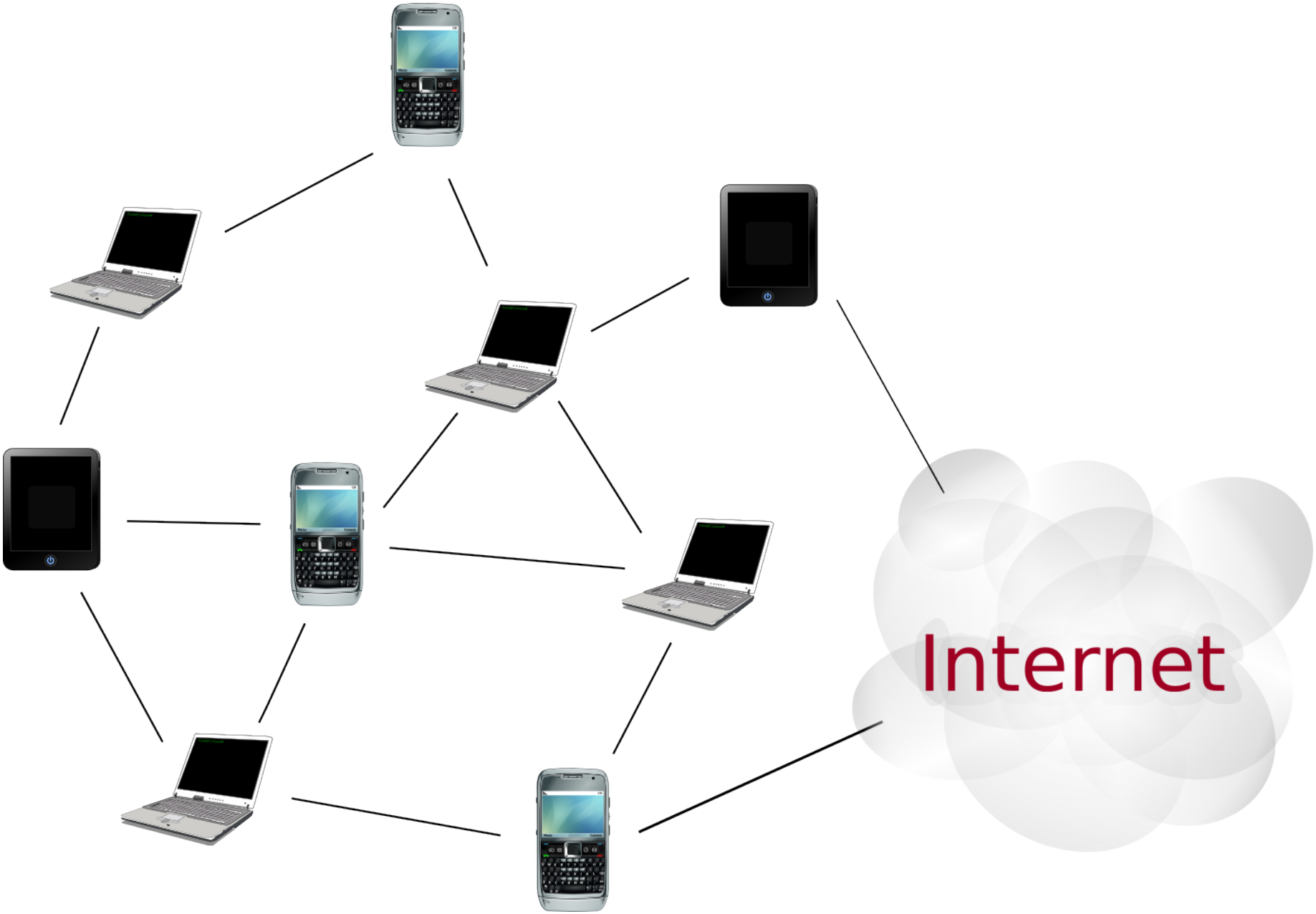
Generally two classes of networks are considered

- Wireless Mesh networks
- Wireless Ad-hoc networks

Wifi Mesh network



Wireless Ad-hoc networks



Features

Relevant features of Wireless Distributed Networks

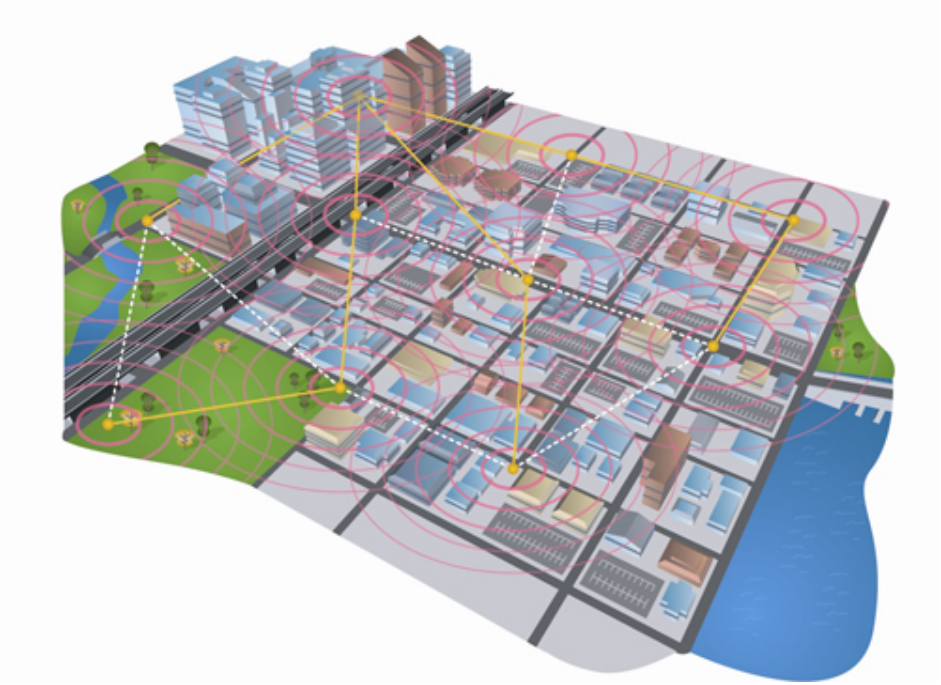
- Multi-hop: cooperative routing to reach the destinations
- Automatic organization: no need for pre-deployment work
- Scalability: networks can be easily extended without planning
- Automatic reconfiguration: recovery from loss of links or failure of nodes

When is a Wireless Distributed Networks useful?

- when there is nothing better to use . . . that happens quite often
- when you want to build an *alternative network*

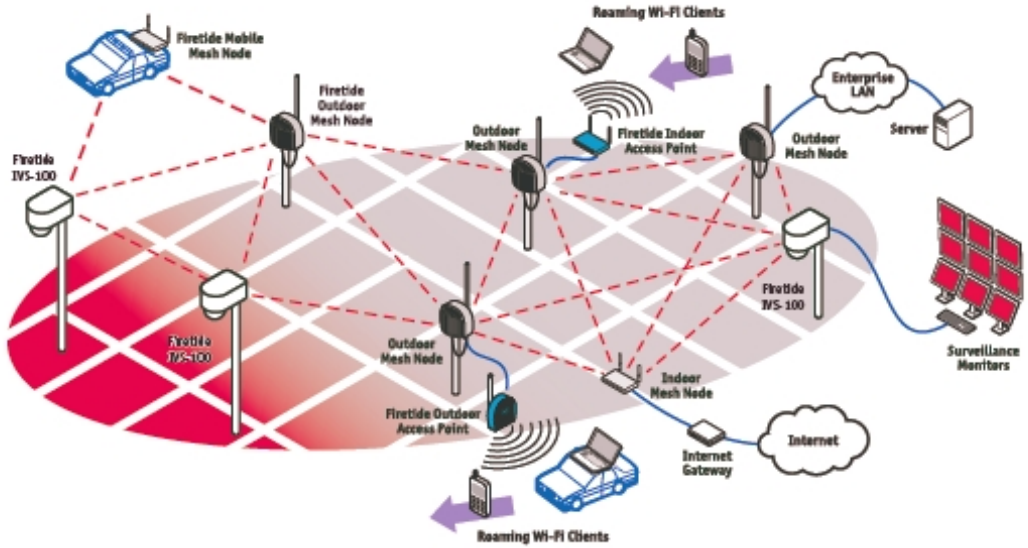
1) Metropolitan Internet connectivity

Bring connectivity when there is no other technology (Cisco does it).



2) Specific applications

surveillance networks (Firetide does it).



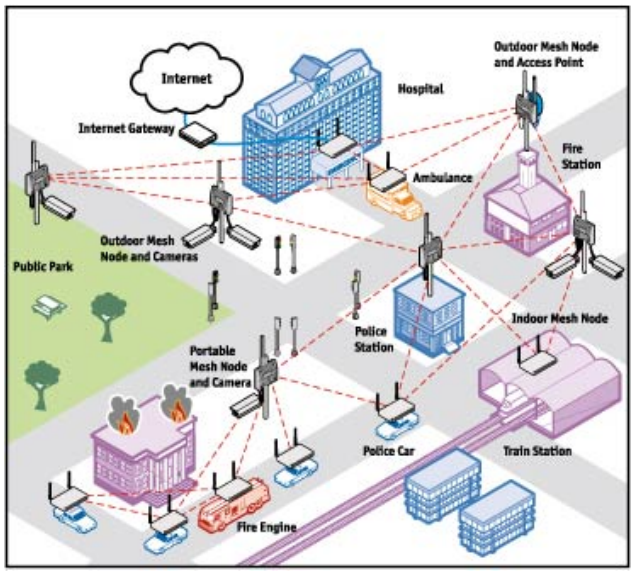
Firetide IVS-100 wireless video surveillance network

2a) Hostile scenarios

- Police/Safety networks (Selex Comms does this):

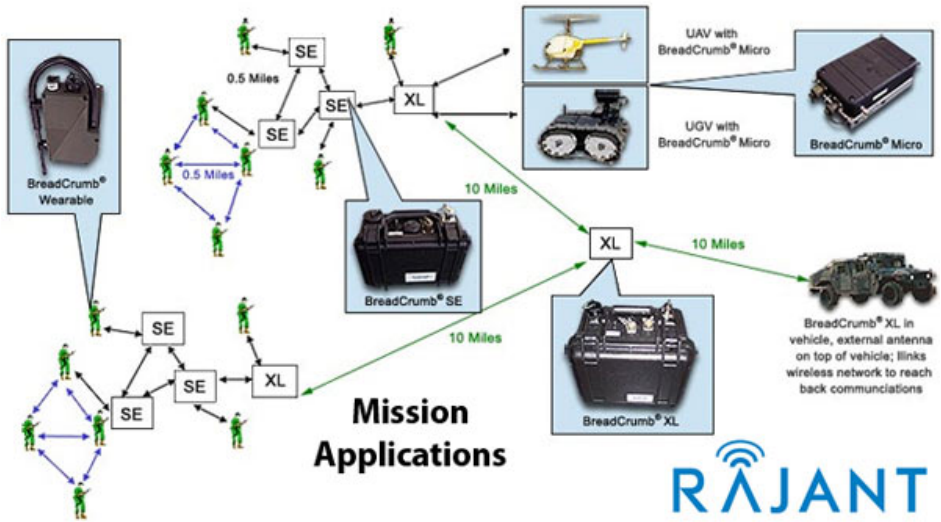
3) There's simply *nothing else*: hostile scenarios

- An emergency network (a lot of people does this):



3a) Hostile scenarios

Military networks (Rajant does it):



Community networks

Community networks are bottom-up networks organized by a local community of people for two reasons:

- Using local services
- Accessing the Internet (when necessary)

CN: history

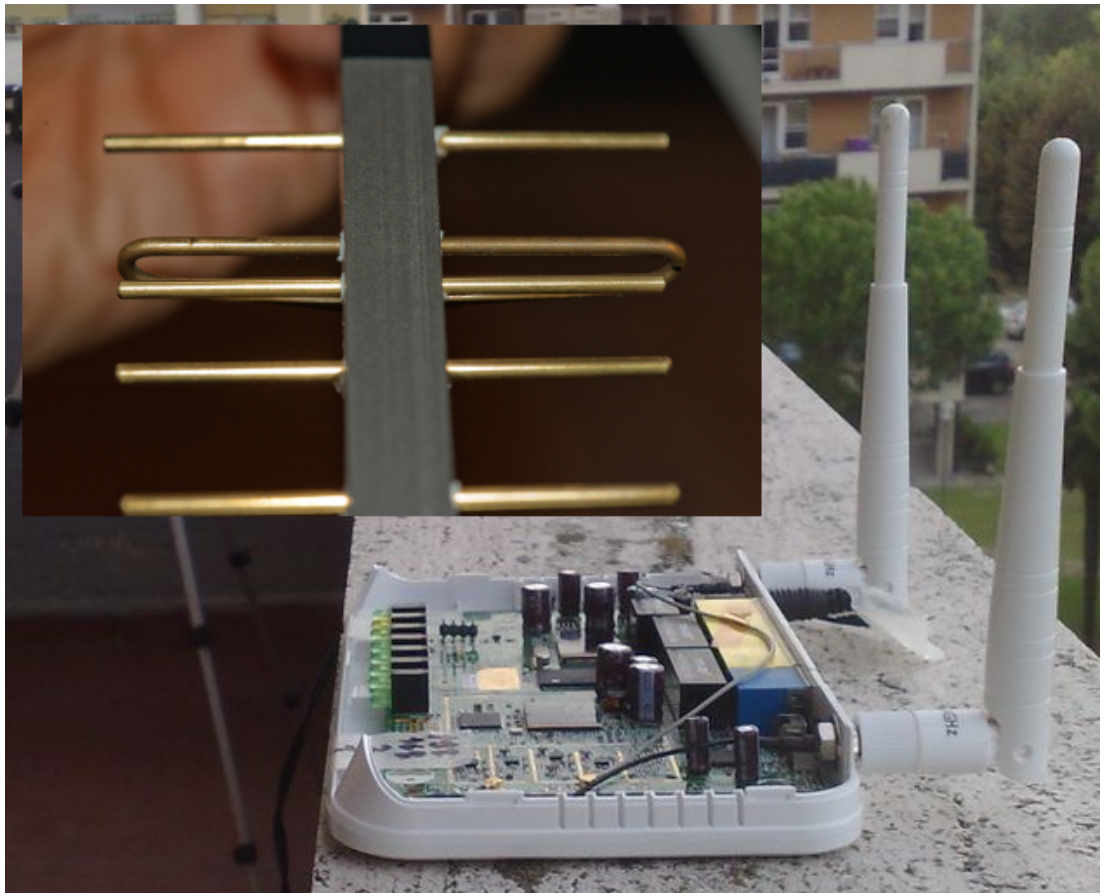
- Most of the community networks were born to solve situations of *digital divide*
- With time, ADSL connections have become common and their cost has lowered
- Some CN then died, others changed
- Today, the spirit that run each community is not to just access the Internet, but to build a community-based network infrastructure

CN: today

- CNs organize and manage their own services:
  - VoIP, Web servers and cloud services, social networks, email . . .
- People are encouraged to use the services offered by the network, instead of the same services offered on the Internet.
- Why?
- Mainly for social reasons:
  - People do not like to be spied (see the *Datagate* scandal)
  - People do not like to be disconnected (see the *three-strikes* policies in Europe and Internet shut-down in many countries)
  - People do not like their traffic to be filtered/shaped by the service providers (see the discussion about the *network neutrality* in EU and USA)
- A CN is not simply a mean to access the Internet, it is part of the Internet.

So, how are CNs made?

Most of the networks start with low-cost refurbished nodes.



So, how are CNs made?

Then, they start using outdoor omni-directional equipment

So, how are CNs made?

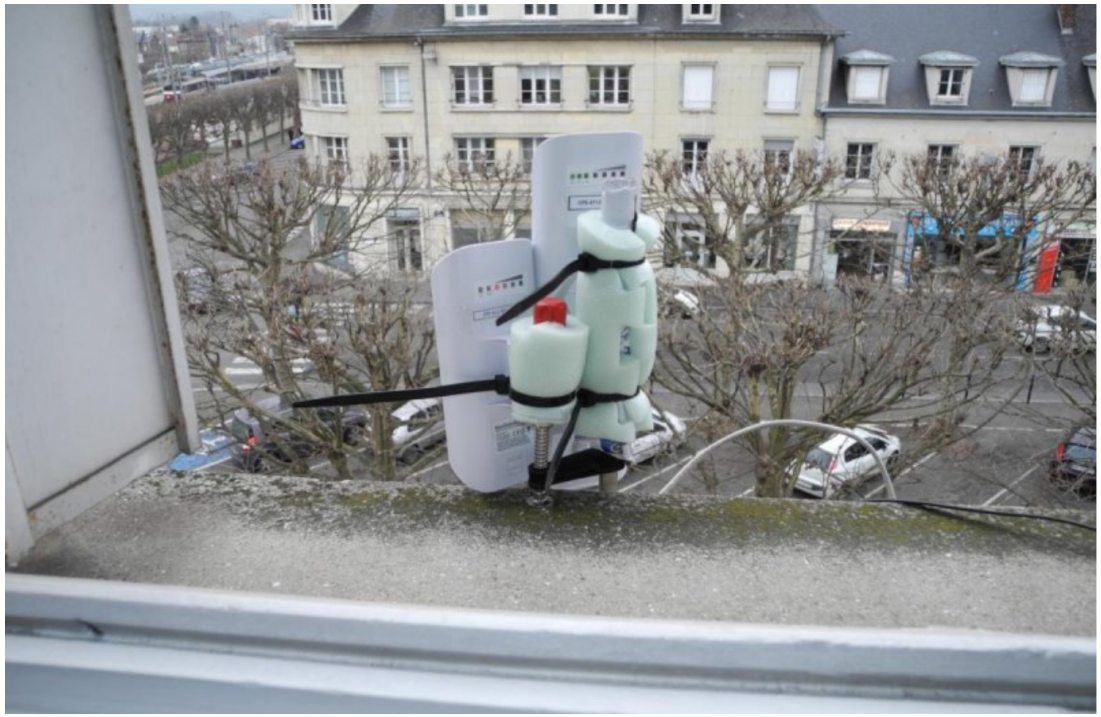
Finally, they end-up with outdoor directional gears

So, how are CNs made?

Finally, they end-up with outdoor directional gears

Where are the nodes placed?

On your windows



Where are the nodes placed?

On your terrace

Where are the nodes placed?

On your terrace

Where are the nodes placed?

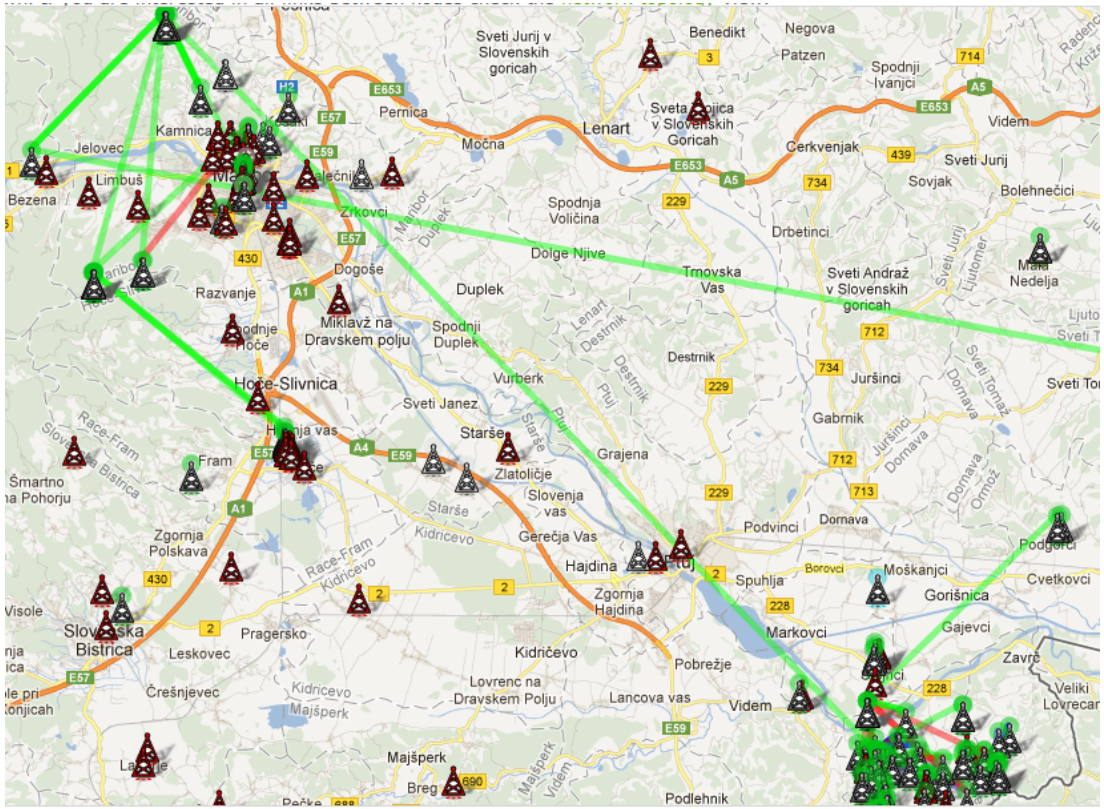
On your roof

Do these network work?

Yes they do. Let's see some examples. . .

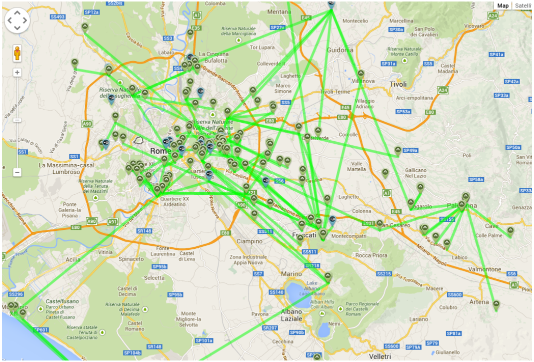
“*Small*” networks

Wlan Slovenia



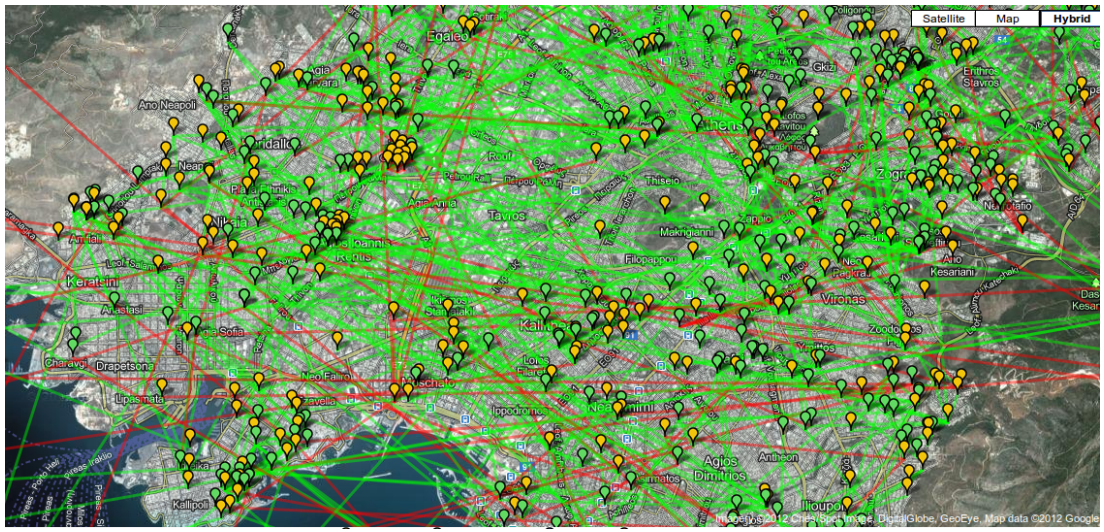
“*Small*” networks

Ninux (Italy)



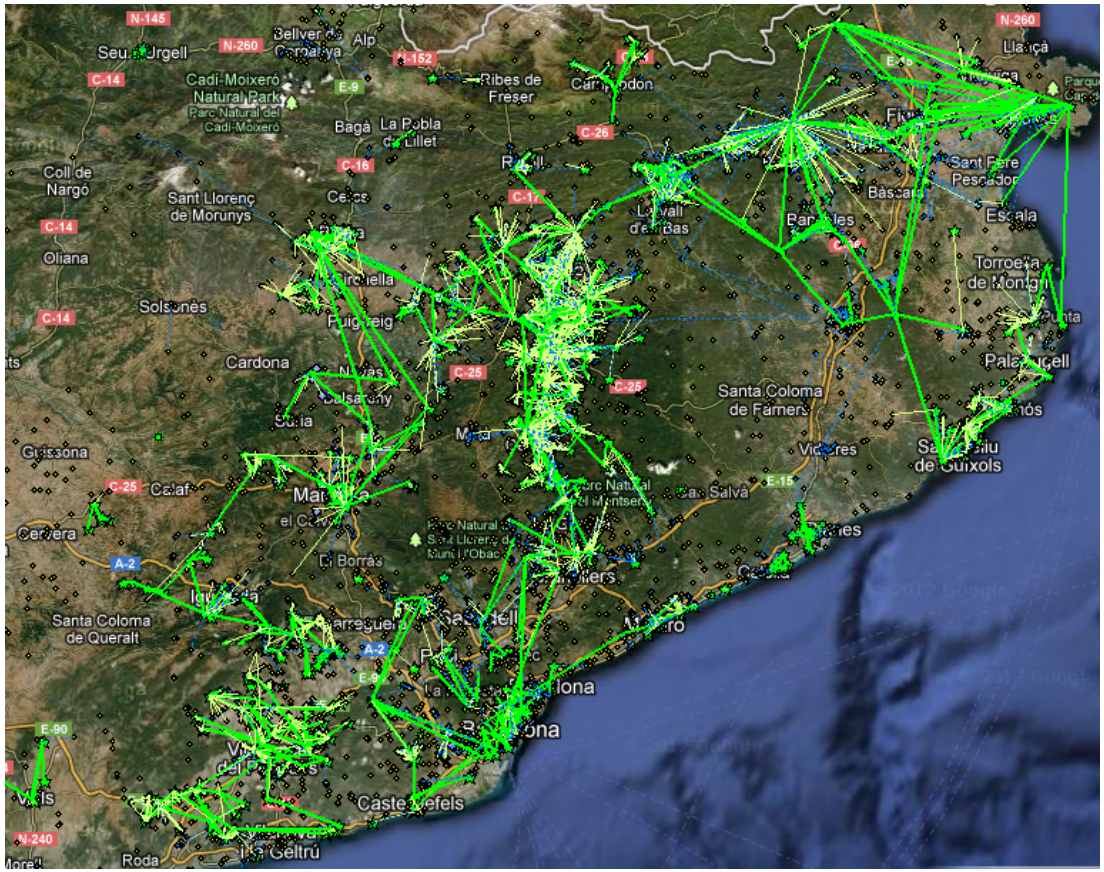
Medium networks

AWMN (Greece)



Huge networks

Guifi (Spain)



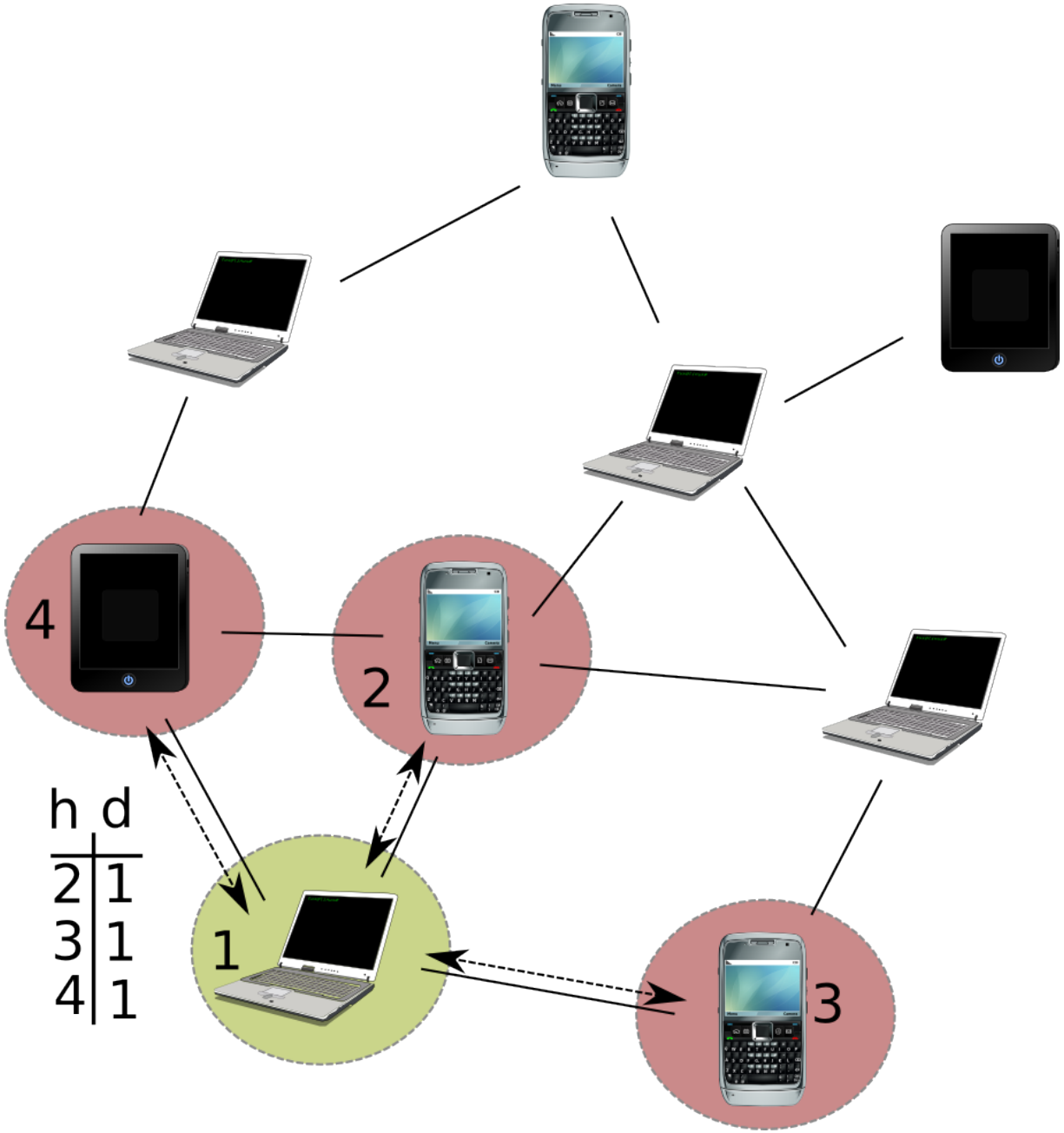
Proactive routing

Proactive routing protocols are protocols that distributes the knowledge necessary to build a RT before it needs to transmit any traffic.

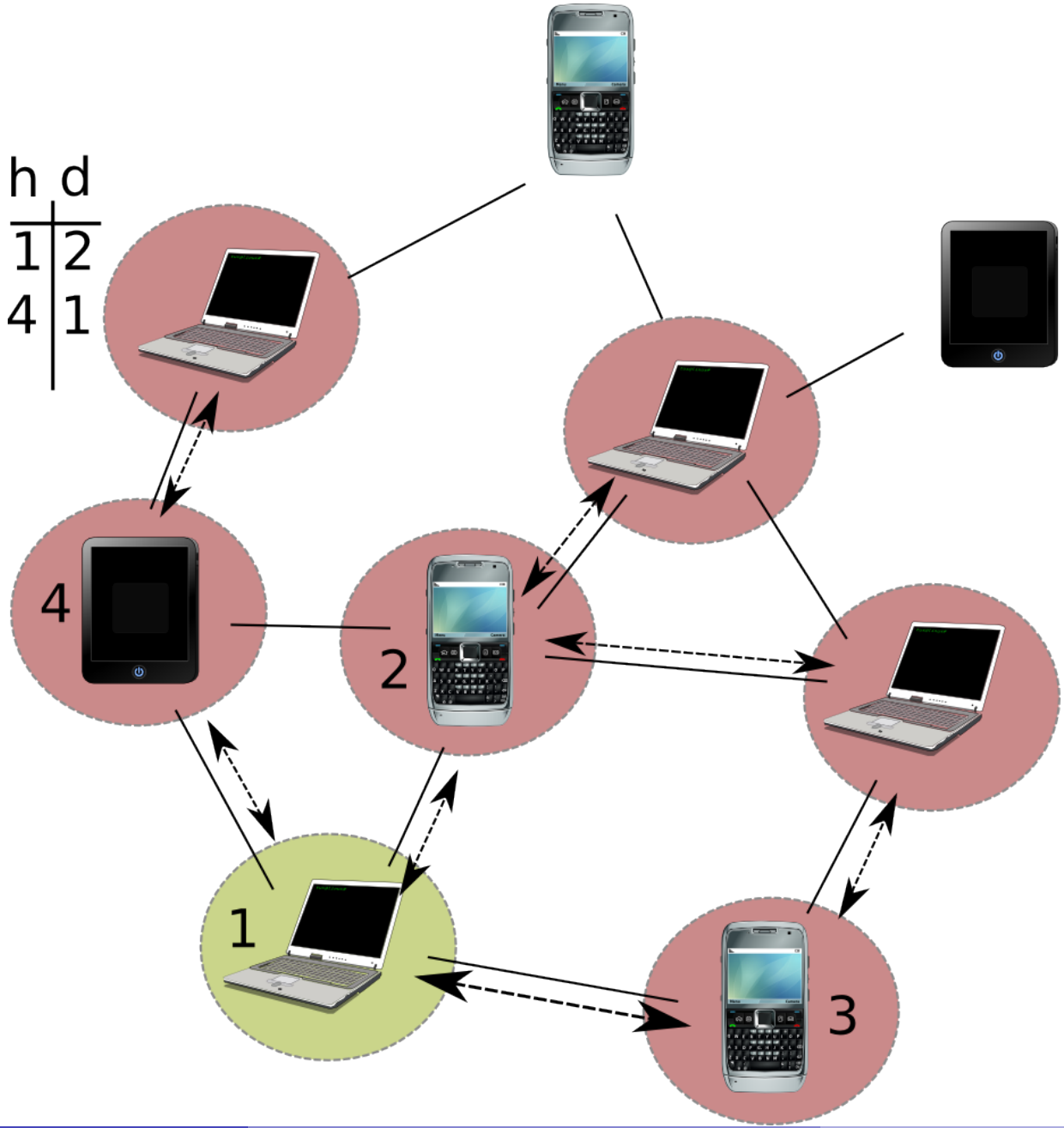
Proactive routing - Naive approach

- Each node at short intervals broadcasts its presence and address with TTL=1 (HELLO messages) $\rightarrow$ local neighborhood is known to every node
- Each node at larger intervals broadcast the address of its neihbors (Topology Control (TC) messages) with large TTL, each topology packet is rebroadcasted by all the nodes in the network (with duplicate detection) $\rightarrow$ every node in the network knows all the other nodes and the links between them
- Every node can build an Adjacency Matrix and calculate a full RT to any destination

Ad-hoc proactive Routing: HELLO messages



Ad-hoc proactive Routing: TC messages



Critical Issues

Criticalities

- The generated control traffic is high
- Wireless links are not symmetric

Issues: traffic generation

Each node of a network with N nodes with an average n neighbors periodically generates:

- Hello messages: few bytes containing only the interface address (size s) with interval t (defaults to 2 sec).
- TC messages: contains all the set of all the neighbors (size $s * n$) at interval T (defaults to 5 sec), reforwarded by every node in the network (multiply by N).

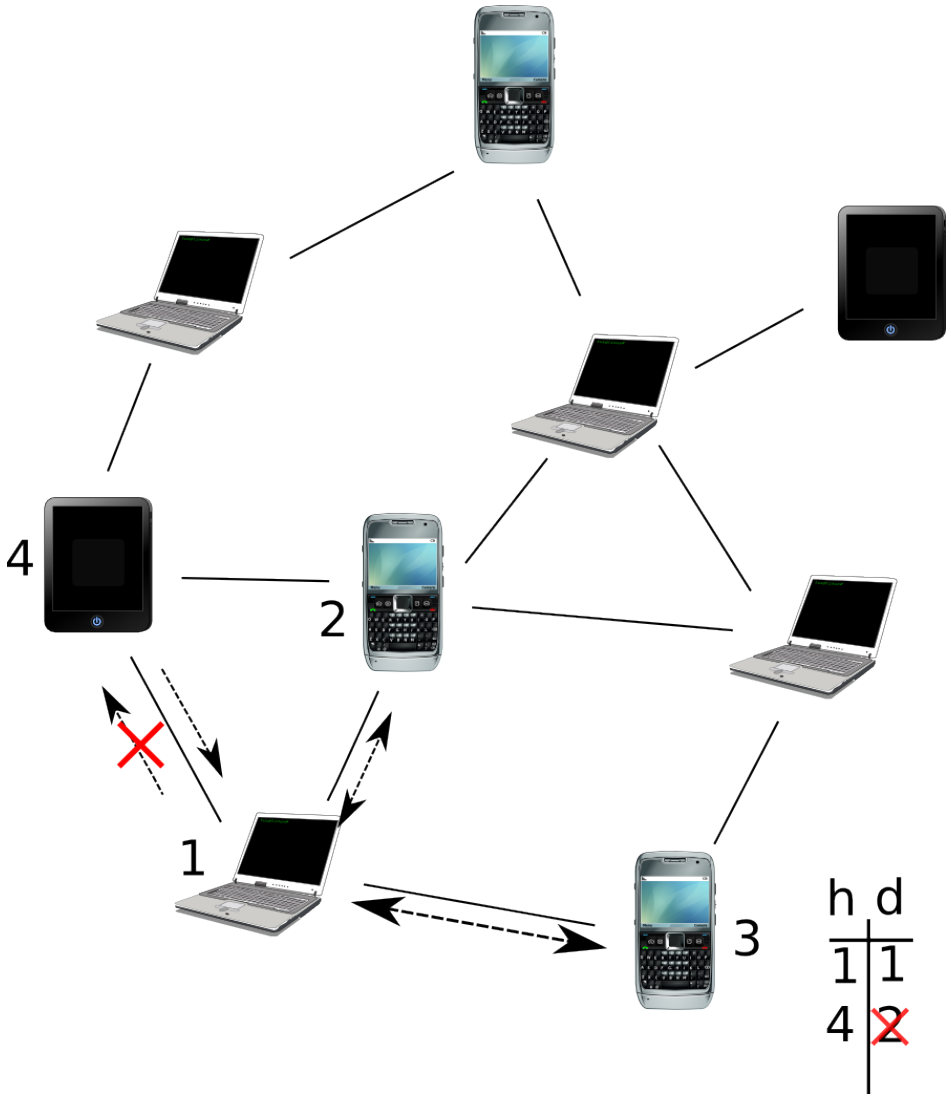
A total of

$$\frac{s * N}{2} + \frac{(s * n) * N^2}{5}$$

bytes per second generated on the whole network (plus protocol overhead). Since the number of links instead grows linearly with N, control traffic per link does not scale well.

Issues: Asymmetric links

Broadcast packets are not acknowledged, this can generate wrong RT



Optimized Link State Routing: RFC 3626, October 2003

OLSR is a routing protocol that tries to solve the previous issues identified in the naive routing protocol described so far. Its two main features are:

- Link state detection using HELLO messages to avoid asymmetric links
- Proactive control message diffusion using MPR nodes to reduce control messages

Some definitions:

N: number of nodes in the network

n_i: set of 1-hop neighbors of node i

n_i^2: set of 2-hop neighbors of node i

m_i: set of MPR nodes chosen by node i

Link Sensing

- To avoid broken links each node will include in HELLO messages its full 1-hop neighbor set.
- For each neighbor in the HELLO a status is added that may be SYM/ASYM
- When node *i* receives an HELLO by neighbor node *j*, *i* will include *j* in its HELLO messages as an ASYM neigh.
- If *j* is able to receive HELLOs from *i* it will do the same
- When *i* receives HELLO messages from *j* containing address of *i* in the neighbors, *i* will sponsor *j* as a symmetric neighbor. *j* will do the same.
- There are hysteresis mechanisms in order to avoid wireless fluctuations.

Link Sensing

OLSR Step 1

Every node must sponsor in HELLO messages its complete neighborhood specifying the kind of neighbor (SYM/ASYM). From now on we will simply call neighbors only the SYM neighbors.

Two main consequences:

- The HELLO messages are bigger than naive approach
- Using HELLO messages each node knows not only its 1-hop neighborhood but its 2-hop neighborhood

multipoint relays: MPR nodes

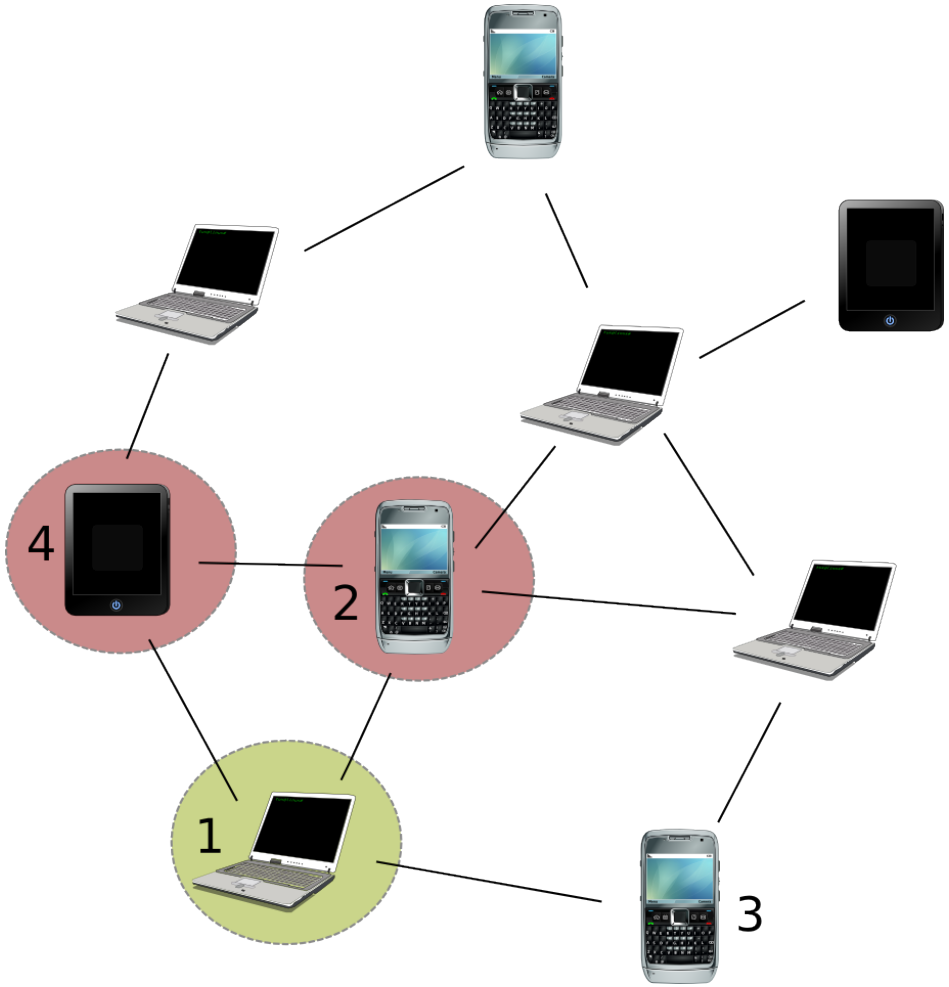
MPR set definition

The MPR set m_i of a node i is an arbitrary subset of its symmetric 1-hop neighborhood n_i which satisfies the following condition: every node in the 2-hop neighborhood n_i^2 of i must have at least a symmetric link towards a node in m_i

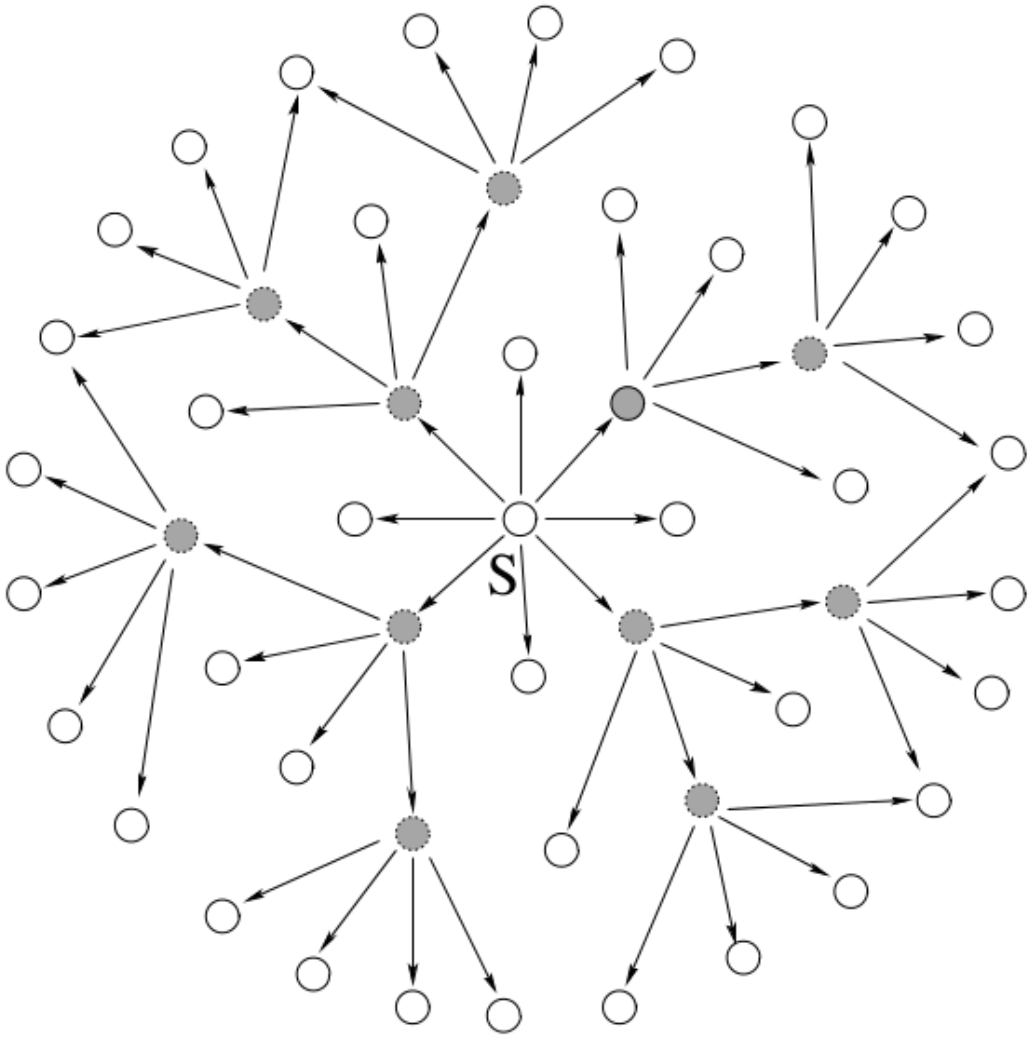
- Basically, m_i is a subset of the neighbors of i that can be used to reach any node in n_i^2 (i.e. they *cover* all the nodes in n_i^2)

MPR

In the example, node 2 and 4 are MPR for node 1 and node 1 is a *selector* for node 2 and 4. Node 3 is not necessary to cover n_1^2.



MPR



MPR properties: 2-hop broadcast

When node i wants to broadcast a message to all the nodes in n_i^2, not all his neighbors need to rebroadcast it. If only nodes in m_i rebroadcast the message the effect is the same with less retransmissions. The smaller m_i compared to n_i^2 the less packets are sent.

OLSR Step 2

Each node must select a subset of N_i, as small as possible. In the HELLO messages, the state of each neighbor contains another bit that is MPR/NOT_MPR so each MPR knows its selector set. Only the MPR nodes participate in flooding messages rebroadcasting the packets from their selectors.

MPR properties: network-wide broadcast

- It is easy to show that if every node follows step 2 then if node i sends a network-wide broadcast control packet M (i.e. a TC message), if there are no losses every node j receives M:
  - By contradiction, if node j did not receive a packet, none of n_j has retransmitted it. This implies that any node k did not select as MPR a node in $n_j \cap n_k$. This, in turn implies that no node is a 2-hop neighbor of j, so j is out of the network or it is a 1-hop neighbor of all the other nodes, so he must have received the packet.

  Step 2 guarantees broadcast communication to all the network using only a subset of nodes, the traffic sent each interval T is reduced (approximately) from $(s * n) * N^2$ to $(s * n) * M * N$ where M is the number of MPR nodes in the network ($M \leq N$).

MPR properties: shortest paths

Shortest paths across MPR nodes.

- Two-hop distance is preserved: Choosing MPRs does not change the connectivity of the network. If j is in n_i^2 its distance from i remains 2 even when routing only on MPRs in m_i
- There exist a shortest path from i to k made of MPR nodes
- Since every MPR generates TCs, k knows every MPR and their neighborhood
- Since every node in the network (excluding trivial topologies) must be selector of at least one MPR node k can compute a shortest path to any other node

TC reduction

- There is no need for a MPR to include in TC messages its full neighborhood, it is sufficient to include only its selector set

Step 3

Only MPR nodes will generate TC messages. Each TC message includes the selector set of the MPR that generated it

OLSR: summing up

OLSR Step 1

Every node must sponsor in HELLO messages its complete neighborhood specifying the kind of neighbor (SYM/ASYM). From now on we will simply call neighbors only the SYM neighbors.

OLSR Step 2

Each node must select a subset of N_i, as small as possible. In the HELLO messages, the state of each neighbor contains another bit that is MPR/NOT_MPR so each MPR knows its selector set. Only the MPR nodes participate in flooding messages rebroadcasting the packets from their selectors.

Step 3

Only MPR nodes will generate TC messages. Each TC message includes the selector set of the MPR that generated it

OLSR: summing up

- If all the nodes apply step 1,2,3 and there is no loss, than all the nodes have enough information to compute the the shortest path to any node in the network.
- Since only the MPR generate TC messages and each TC includes only a fraction of the neighborhood of the MPR that generated it the overall control traffic is reduced to $(s * m) * M^2$ from the initial $s * n * N^2$ with naive mode[1].
- It is very important that each node select a minimal MPR set, since M depends on that.

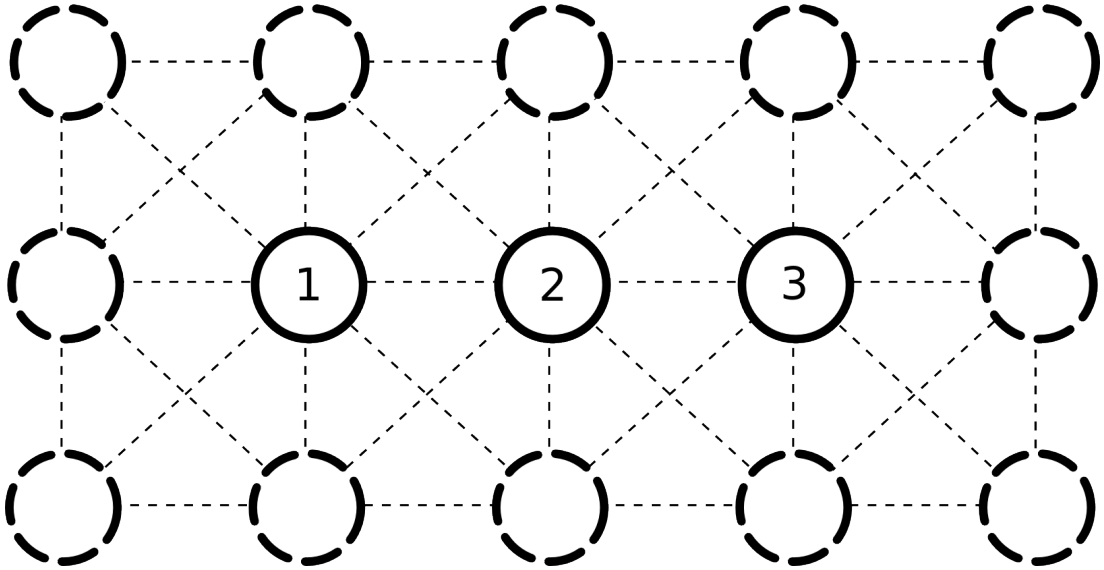
[1] 1 approximations I've done: for each generated TC this will be forwarded by all the MPR.

Examples[2]

- Only 3 nodes over 5 in the network generate TC messages
- Each TC message is forwarded exactly 2 times

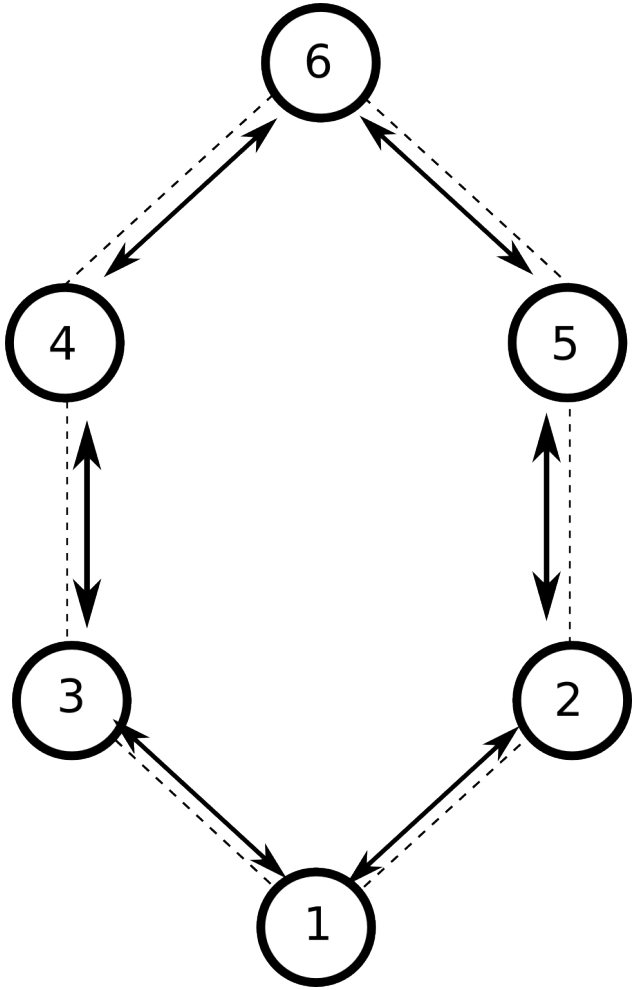
[2] dashed edges are wireless links, arrows go from a selector to the MPR (only the most important ones). Solid vertexes are MPRs.

Examples



- Only 3 nodes over 15 in the network generate TC messages
- Each TC message is forwarded exactly 2 times

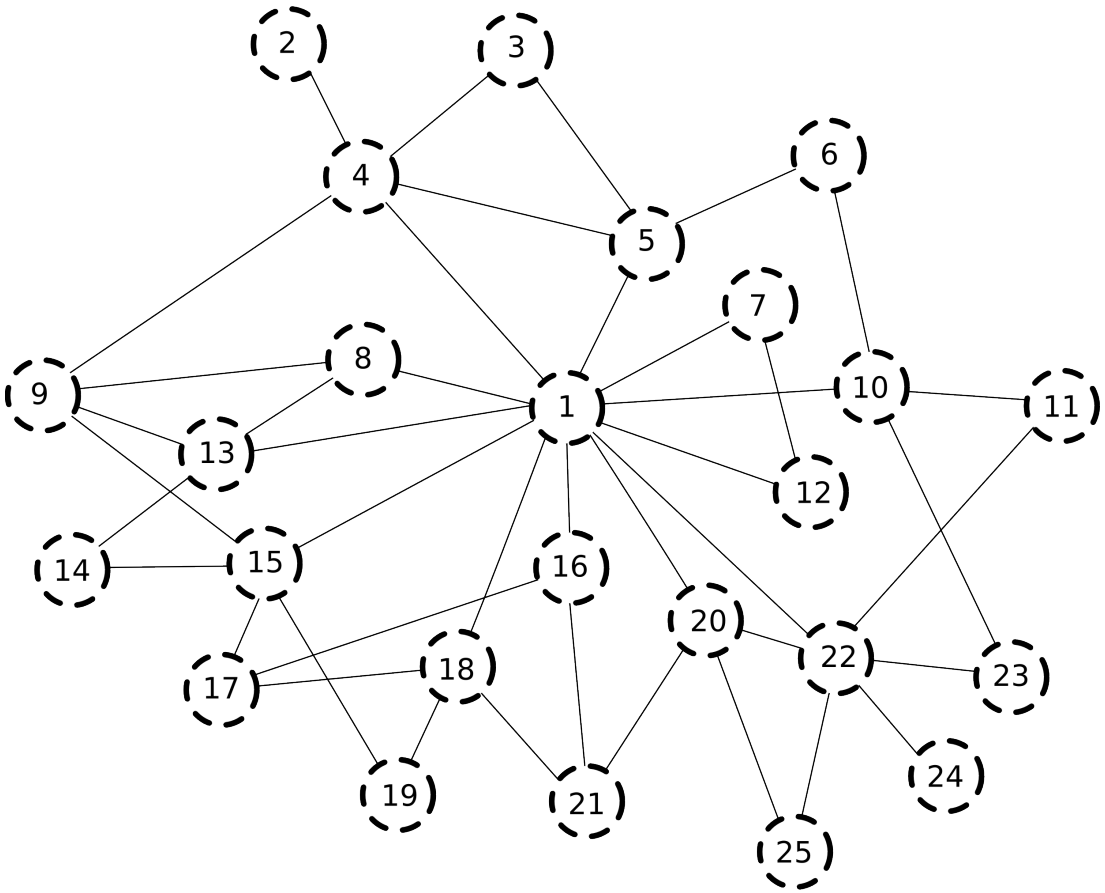
Examples



- All the nodes in the network generate TC messages
- Each TC message is forwarded exactly 5 times

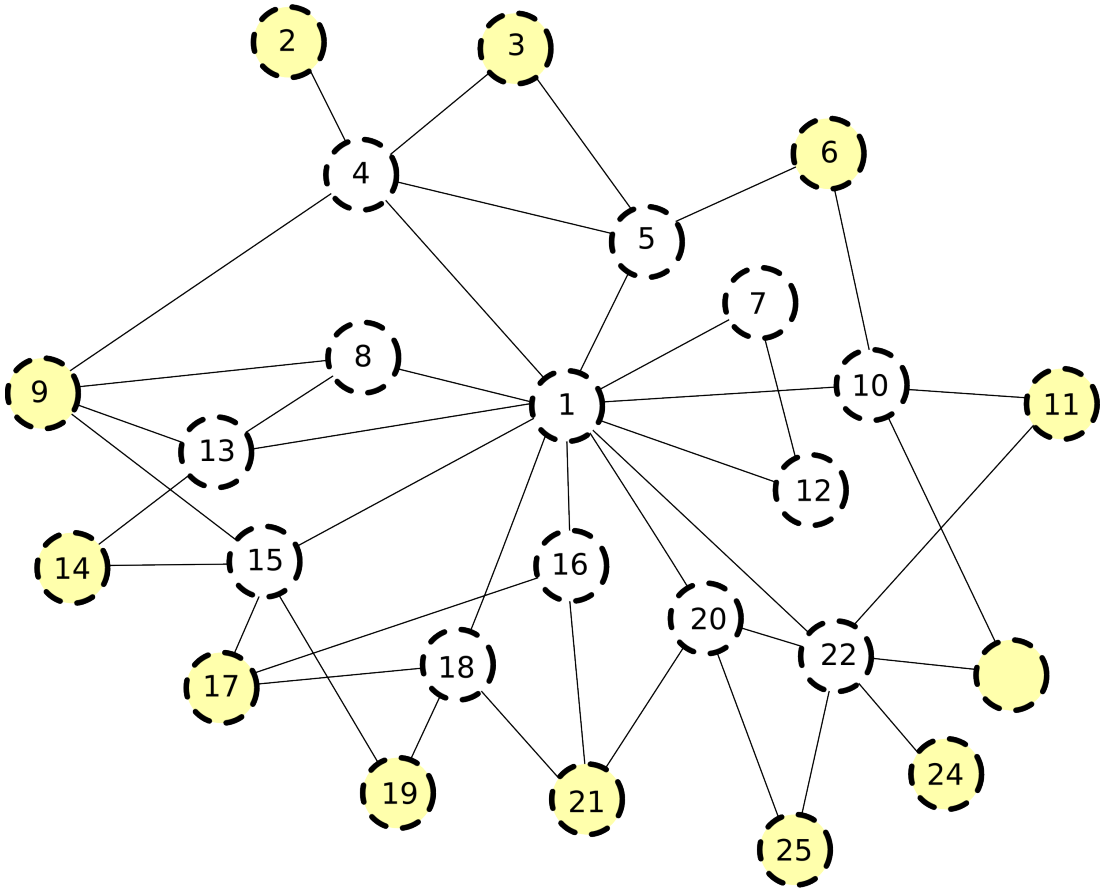
Exercise:

We have seen that the number of MPR in the network has a critical impact on control traffic, each node must try to find the smallest MPR set. Consider this network and try to compute the minimal MPR set for node 1



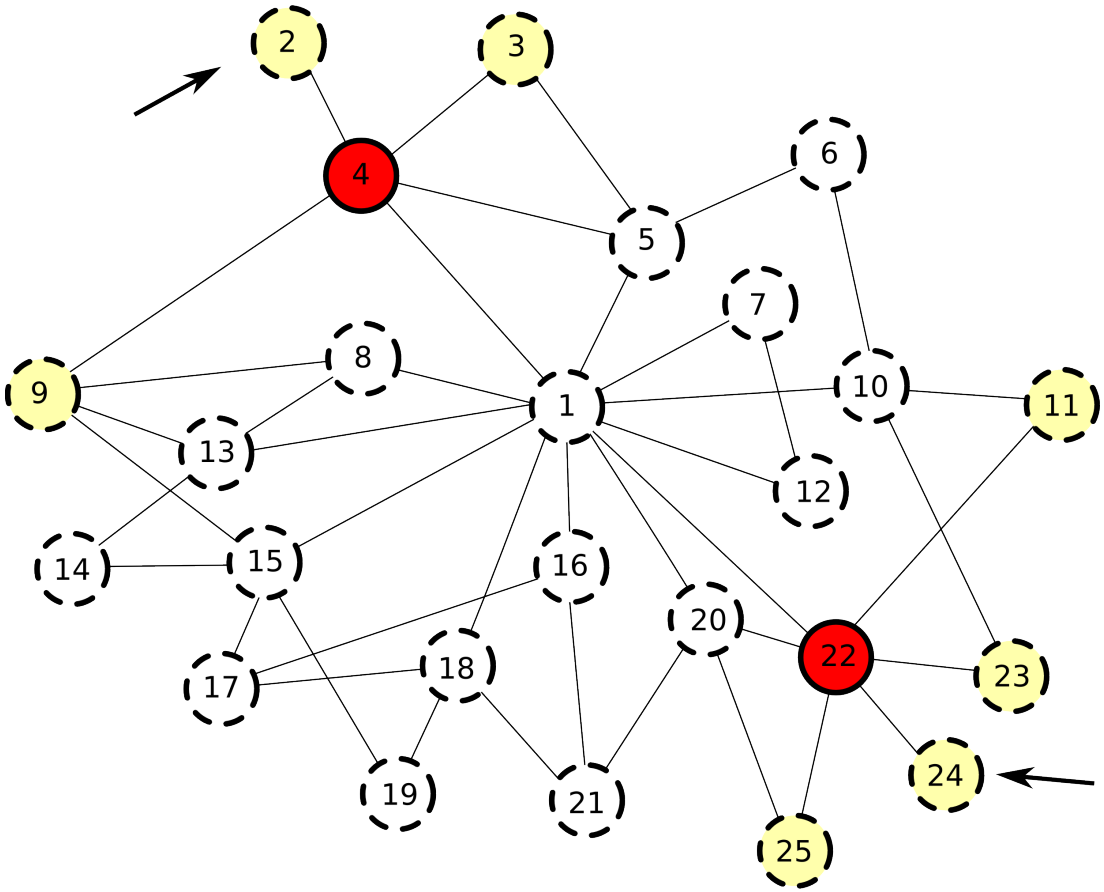
One possible solution: step 1

Identify the 2-hop neighbors



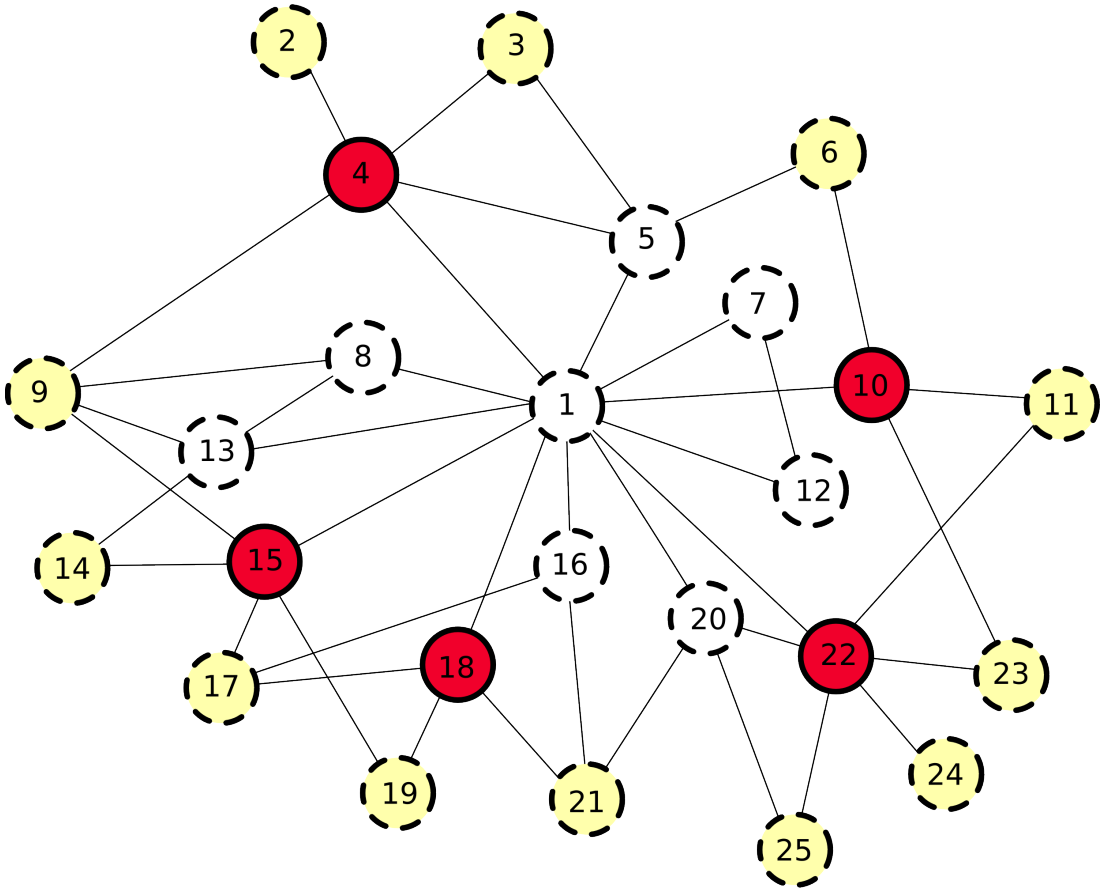
One possible solution: step 2

Identify the forced MPR nodes

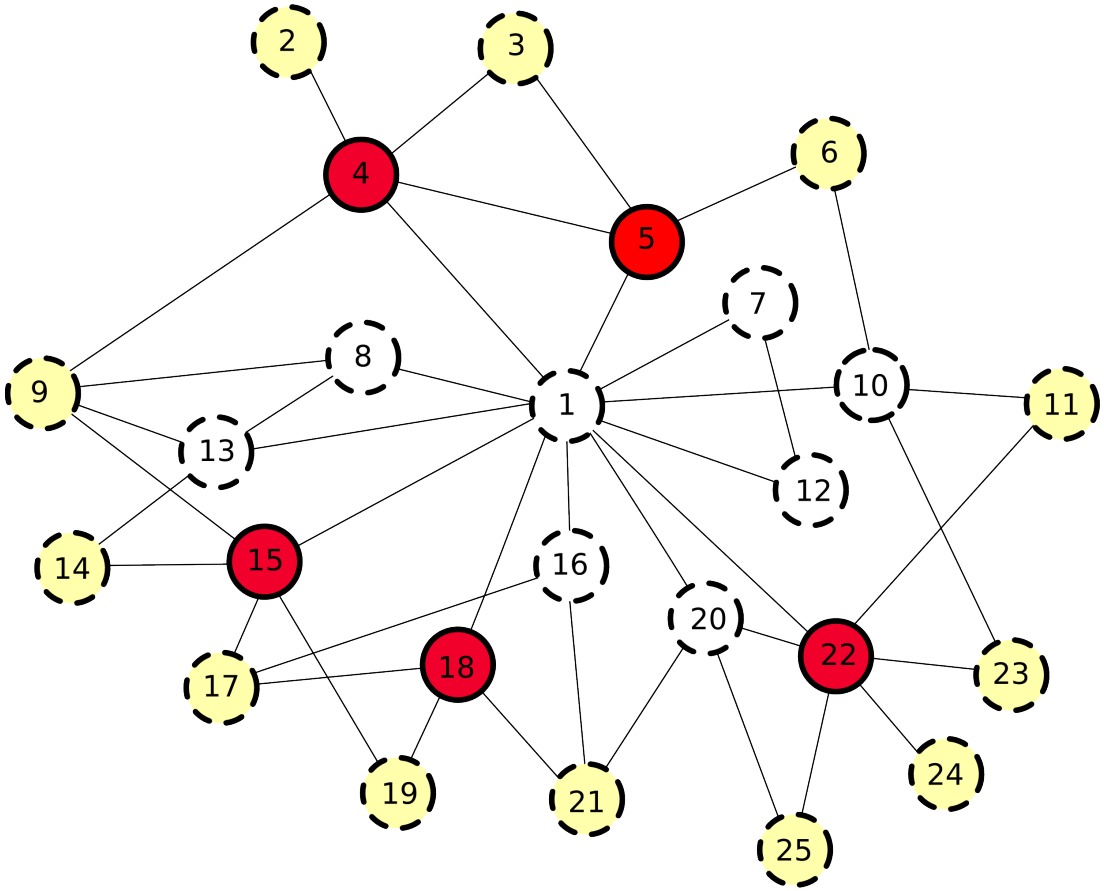


One possible solution: step 3

Try to select the smallest MPR set among the remaining 1-hop neighbors



Another possible solution: step 3



MPR Selection

Two issues emerge from this example

- In some topologies there are nodes that must be included in the MPR set
- There can be more than one minimal MPR set

MPR Selection: complexity

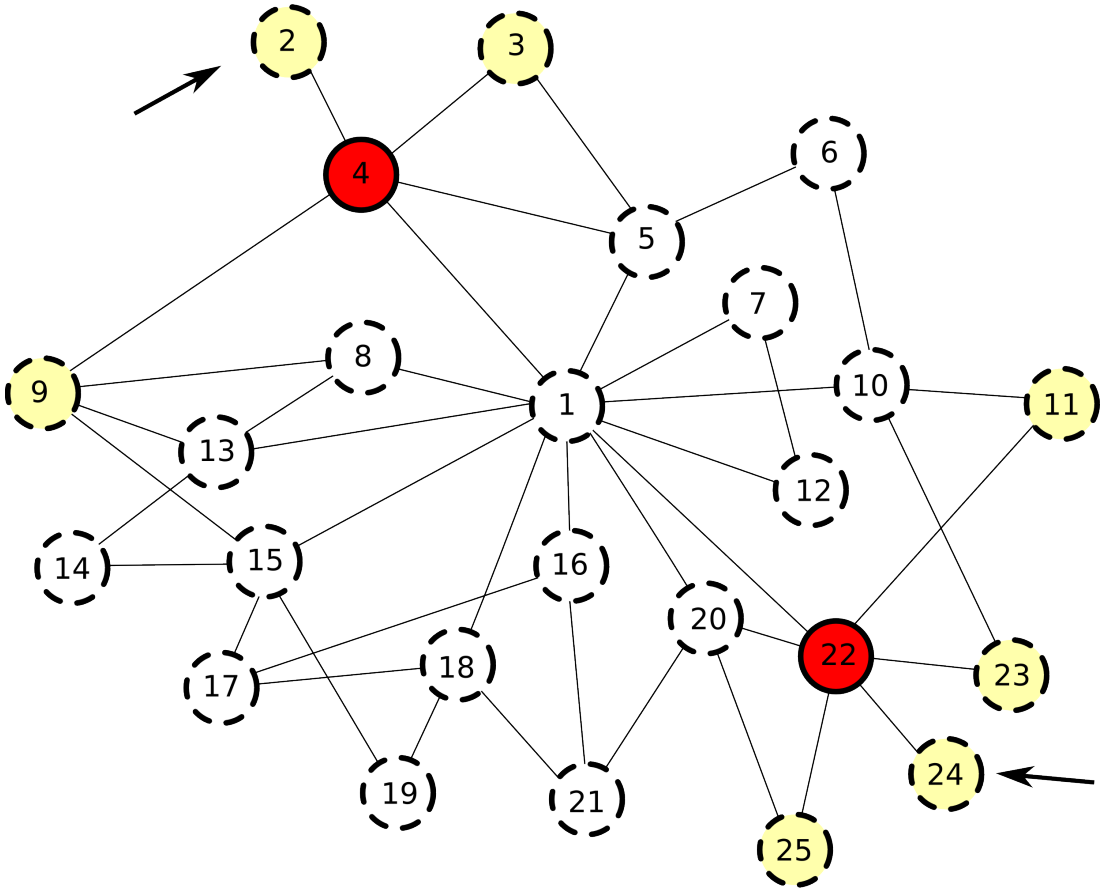
- It can be shown that finding the optimal MPR set is NP-complete.
  - It takes polynomial time to verify a given solution
  - No polynomial time algorithm is known to find a solution
- For large neighborhood it is inconvenient to try to find the optimal solution

MPR Selection: heuristic

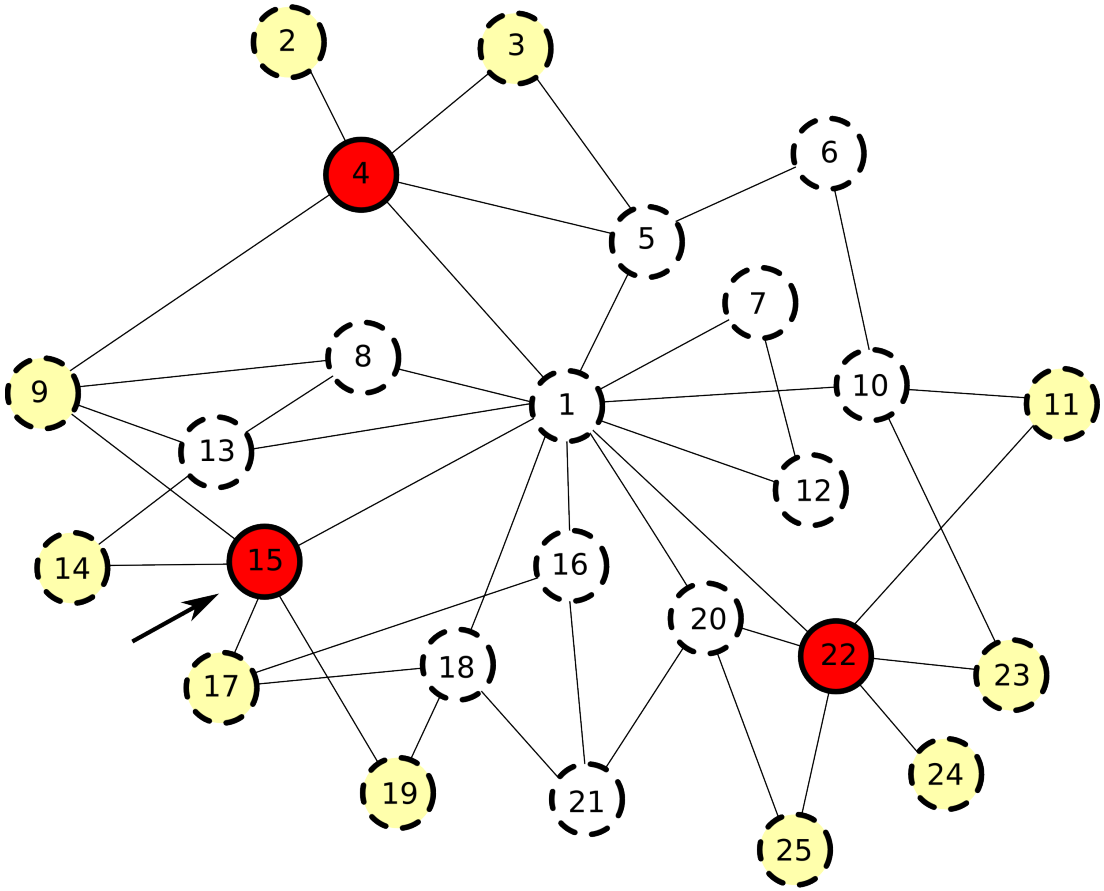
OLSR introduces the following heuristic for a node x:

1. Find all nodes j in n_x^2 that have only a neighbor k in n_x^1, insert k in m_x. Those nodes must be inserted to guarantee full 2-hop connectivity.
2. Repeat the following until any node in n_x^2 is reachable using nodes in m_x:
   1. Order every node u in $n_x \setminus m_x$ for their reachability, i.e. the number of nodes in n_x^2 that are covered by u and are not covered by any other node already in m_x. Insert in m_x the one with highest reachability.
   2. In case of a tie, insert the one with the highest willingness
   3. In case of a further tie compute the degree of u, that is $|n_u^1 \setminus n_x^1|$ and insert the one with higher degree. In practice, get the one with the highest number of 1-hop neighbors that are not shared with x

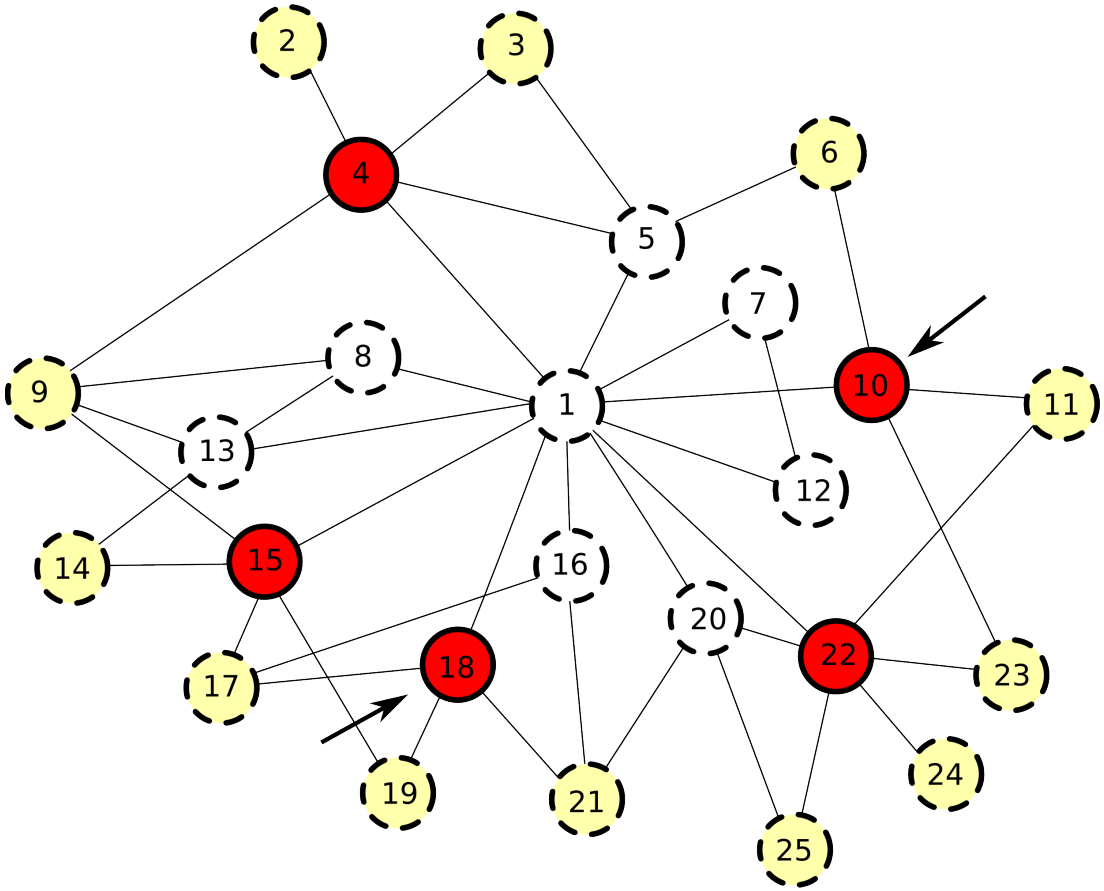
Step 1



Step 2



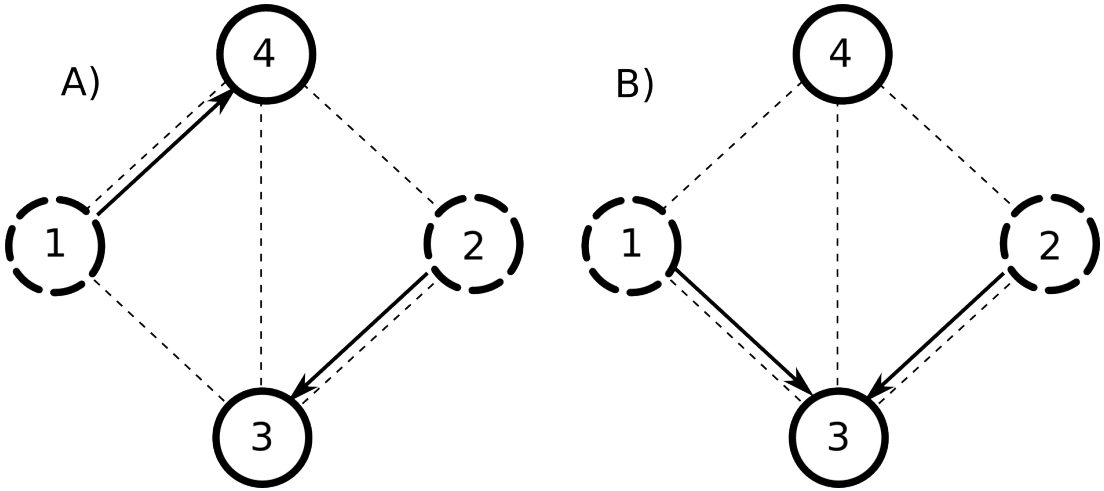
Step 3



MPR Selection: heuristic

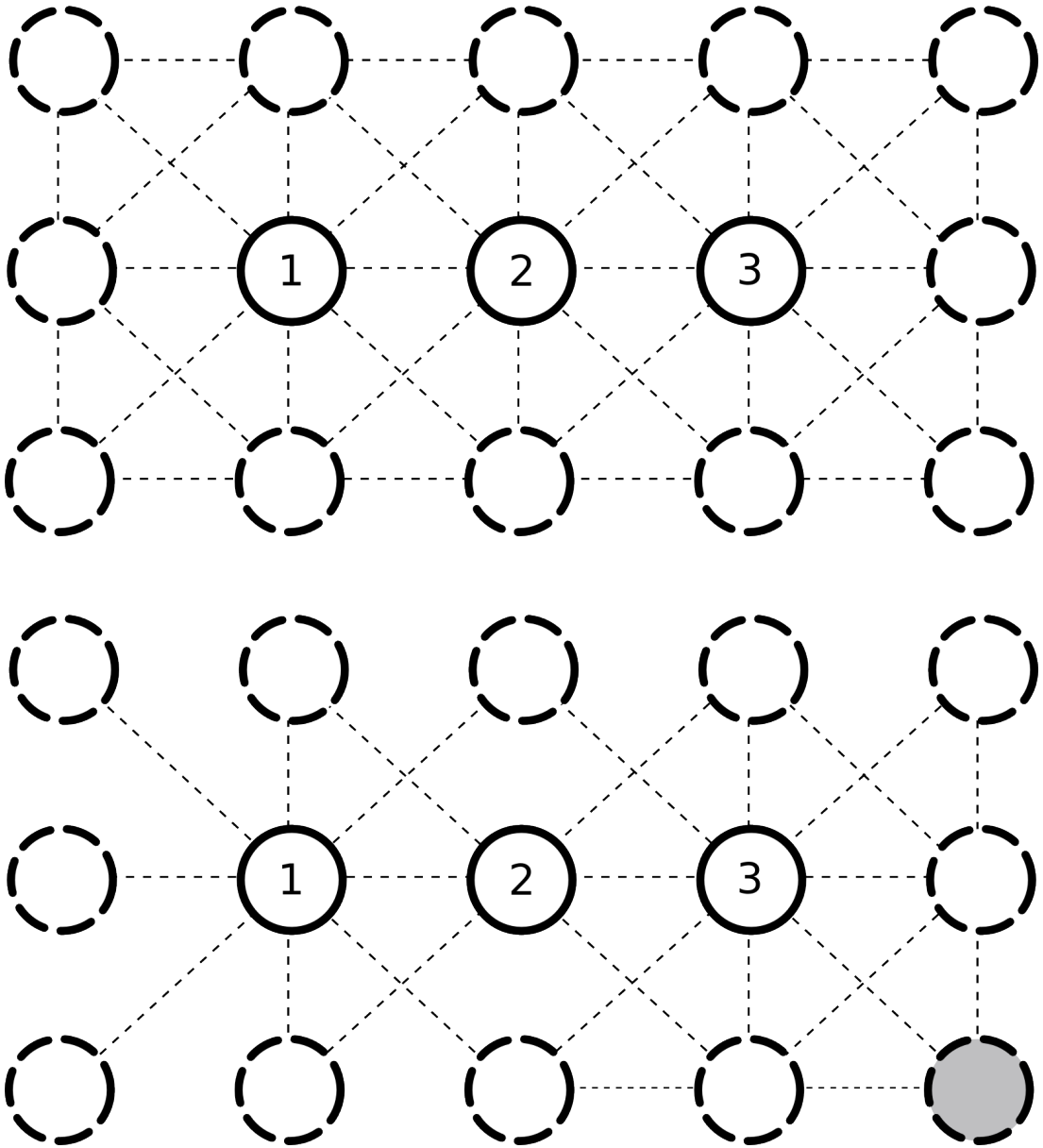
- It can be shown that this heuristic produces an MPR set of size S that compares to the size of the optimal set $S^{\star}$ with the following relation:
  - $S \leq log(\Delta)S^{\star}$
  - Where Δ is the maximum number of nodes in n_x^2 a node in n_x can cover.

Diamond network



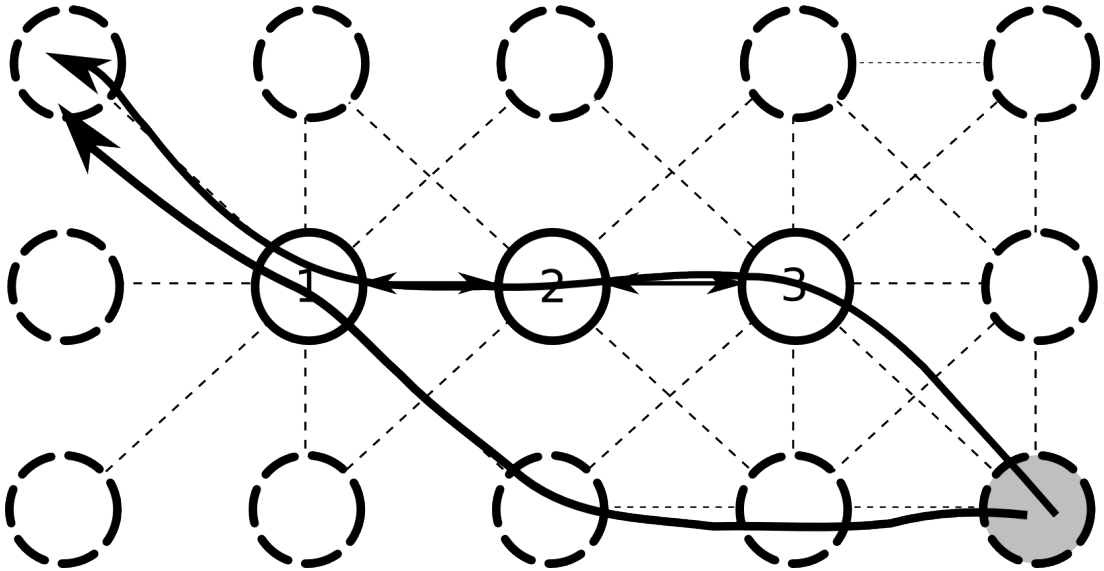
- Both these scenarios are perfectly possible but one generates twice the TC messages of the other
- This is the price to pay to have a completely distributed MPR selection algorithm

Network approximation



Equivalent routes

- Once you know an approximated topology, you can compute shortest-path routes
- Depending on the algorithm you chose you can have various paths



RT building

Note that:

- Note that the routes are decided hop-by-hop so any node can decide only the first hop
- In the following hops the route can be changed
- The closer you get to the destination, the more precise is the knowledge of the topology

OLSR Extensions

- The protocol described in the OLSR RFC has been improved to solve some problems.
  - Too much control traffic generated $\rightarrow$ Fisheye
  - No measure of quality for the links $\rightarrow$ ETX

Fish-eye

- A node needs only the necessary information to decide what is the next hop on the shortest path to the destination.
- We have seen that nodes already have a limited view of the topology outside their 2-hop neighborhood
- This approach can be enhanced with fisheye strategies

Fisheye

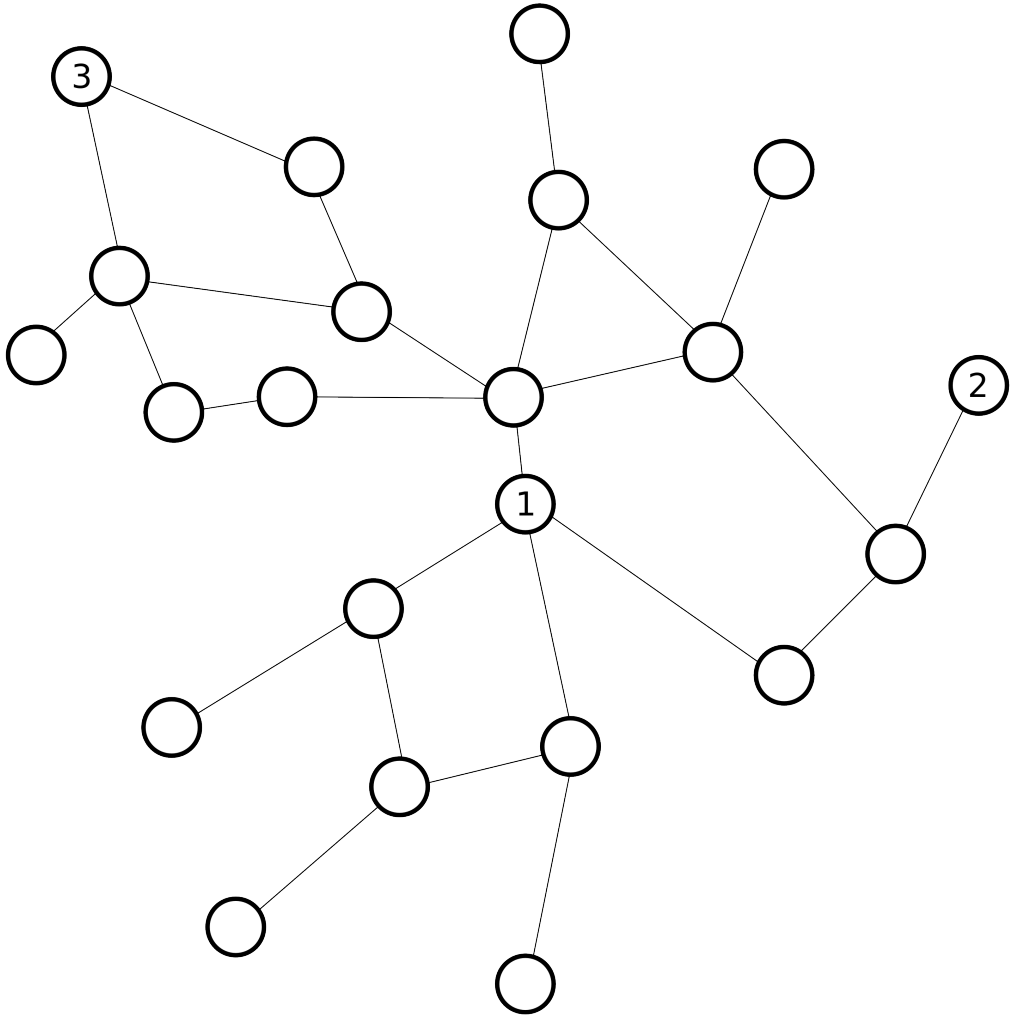
- The word fisheye refers to the view of the world a fish has under the water
- If you look up from below the water you see the whole 180 degrees horizon compressed in a cone with a smaller angle
- Basically, you have better vision on the close thinks and a fading compressed vision on objects far away. The same happens if you use a *fisheye* lens on a camera

Fisheye [3]

[3] http://en.wikipedia.org/wiki/File:Museumfisheye.jpg

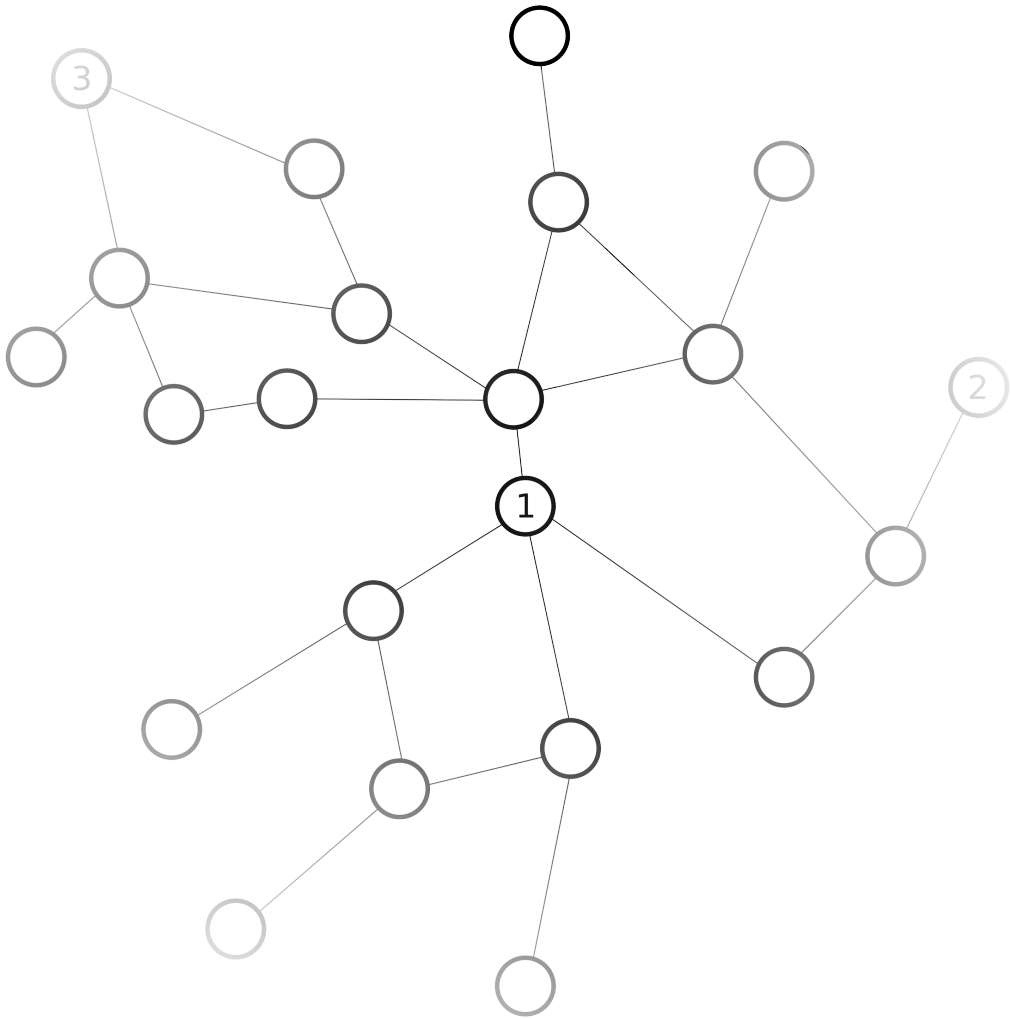
Fisheye OLSR

Fisheye transforms the network from this . . .



Fisheye OLSR

. . . to this

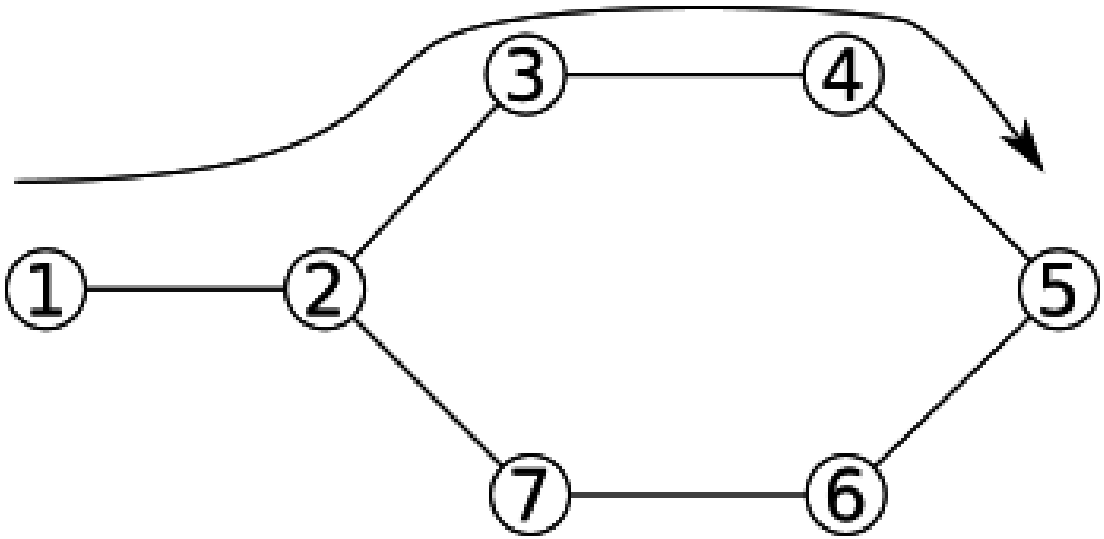


Fisheye

- To obtain this effect, the TTL of the TC messages is not always to the maximum.
- a sequence of TTL can be used to control the diffusion of TC messages, for instance {2,4,16}
- Doing this, the first TC message will be sent only to nodes that are 2-hops away, the second to nodes that are 4-hops away, the third to all the nodes in the network
- Periodically every node receives enough information to build all the routing table, but the information is more accurate (it is fresher) for closer nodes
- With Fisheye, the number of TC messages forwarded is not any more linear with the number of MPR nodes
- How to define the TTL sequence is an open issue

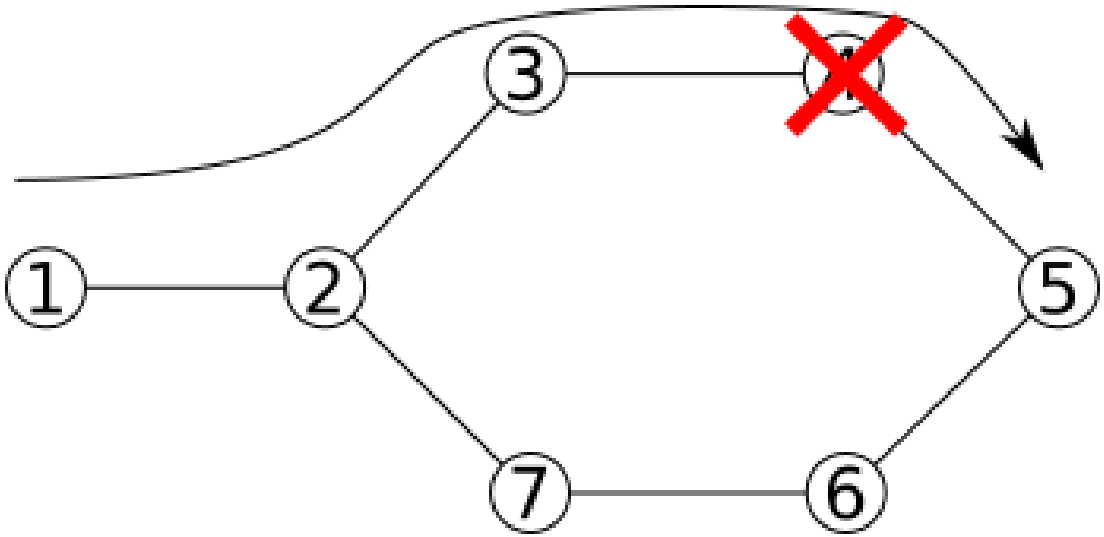
Fisheye and loops

Let's consider this simple network, with the path 1-5 as shown:



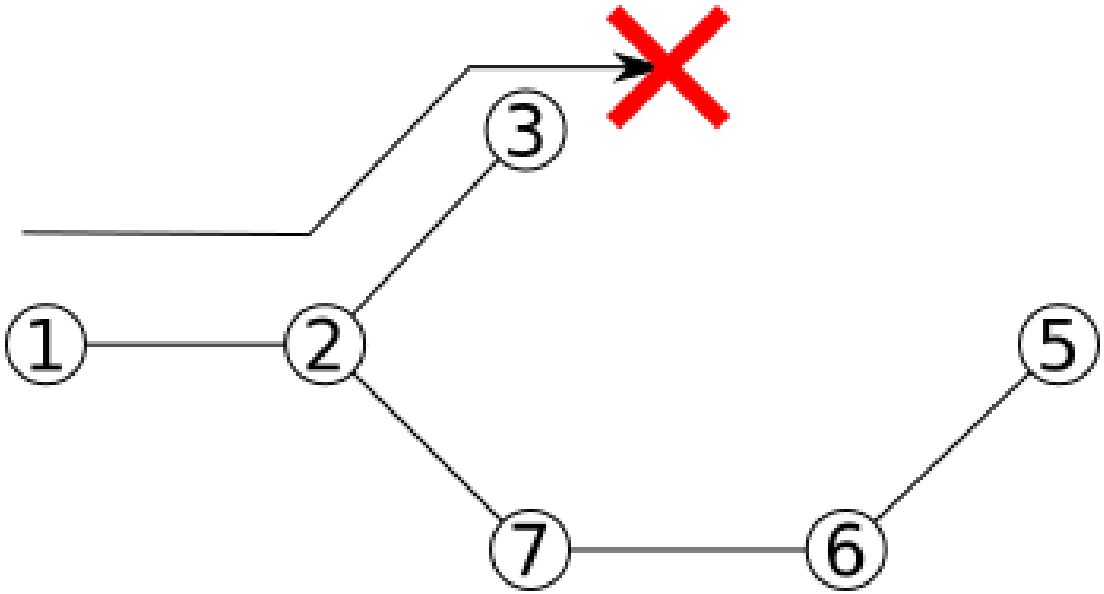
Fisheye and loops

Imagine node 4 breaks



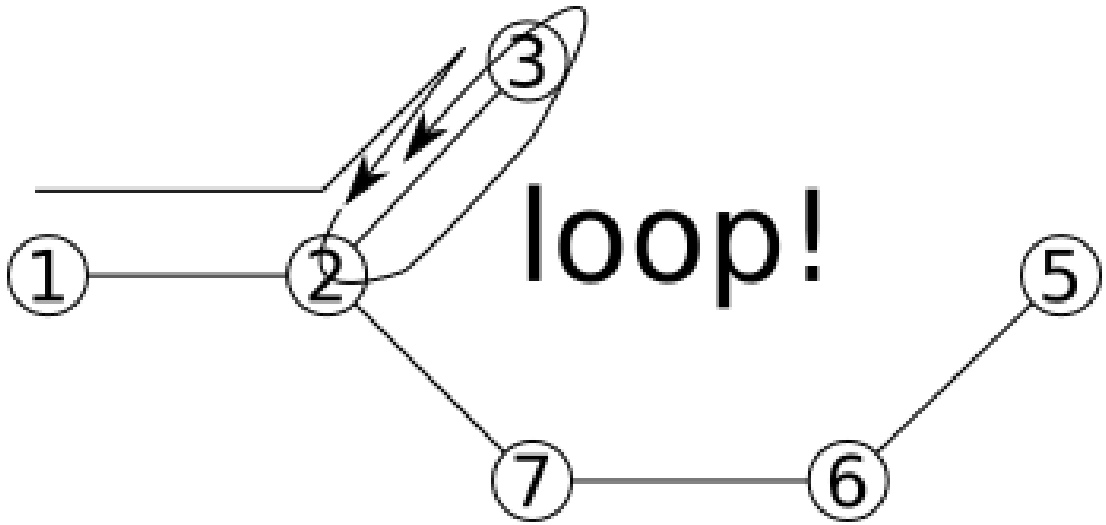
Fisheye and loops

Up to when node 3 does not sense the broken link, packets are lost on the broken link.



Fisheye and loops

When node 3 looses enough HELLOs from 4, it will recalculate its own path to node 5.



Node 2 still doesn't know that node 4 died, a loop is created!

Fisheye and loops

- The loop is due to the fact that node 3 and node 2 have a different knowledge of the network topology
- Node 3 in fact receives HELLOs from node 4 with a higher frequency than node 2 receives TCs.

Fisheye and loops

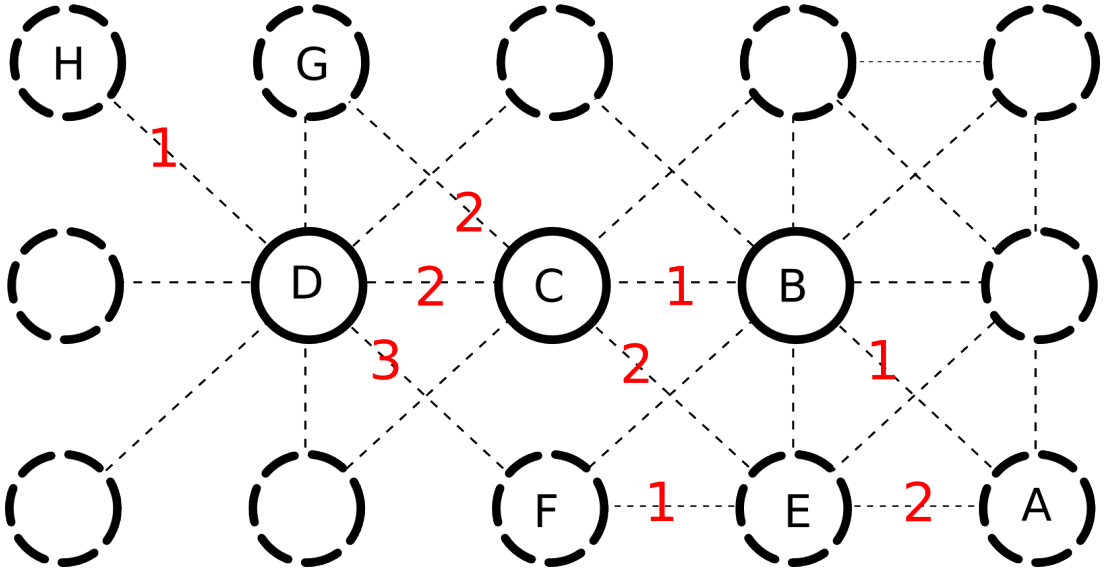
- But this is what Fisheye does all the time!
- Fisheye create regions of nodes with different (more or less updated) knowledge of the topology.
- Nodes in the border between regions can create loops.

QOLSR

Can we add QoS to the routing tables?

- Each node makes an estimation of the link quality for its neighbors (a numeric value).
- Each node includes this value in its HELLO and possibly TC messages.
- The adjacency matrix now has weights on the edges and smarter algorithms can be used for routing (Dijkstra)

QOLSR



- From A to H, route A-B-C-D-H has weight 5 while route A-E-C-D-H has weight 7
- Note however that again, there is no warranty that the real path will be the one calculated by A, even if, with quality it is less likely to have multiple choices
- How to define and measure the metrics, and how to avoid oscillation of routes are open issues.

The ETX metric

ETX (Expected Transmission Count) is the simplest, yet one of the most used metrics to estimate link quality. Take two neighbor nodes, A and B.

- In each HELLO packet A sends it is included the frequency of generation of the HELLOs for A
- B knows the time interval in which it will receive a new HELLO
- B can sense the loss of HELLOs and keep a time-window in which it counts received and missed HELLOs.
- The ratio received/lost is called the Link Quality (LQ) measured by B for neighbor A.

The ETX metric

- In each HELLO, for each neighbor, B includes the LQ, same does A.
- When B receives HELLOs from A he receives the reverse link quality (RLQ) that was measured on the link from A.
- Finally ETX metric is given by:

$$ETX = \frac{1}{LQ * RLQ}$$

The ETX metric

Note:

- LQ: probability to successfully send a packet from A to B
- RLQ: probability to successfully send a packet from B to A
- Since each 802.11 packet to be successfully transmitted requires two packets (data packet + ACK in the other direction) then the probability of successfully transmit a packet is estimated by RLQ*LQ.
- ETX thus is the inverse of this probability, that is, the average number of transmission attempts B needs to successfully send a packet to A
- ETX is symmetric

The ETX metric: issues

- Broadcast packets are sent at the base rate, unicast packets are sent at the negotiated rate.
- Broadcast packets are small packets, so the chance of collision is smaller than traffic packets.
  - **ETX is optimistic**
- ETX does not say anything about the capacity of the link
- ETX does not say anything about the delay on the link

The ETX metric: issues

On the other hand

- ETX does not require any cross-layer interaction, it is based only on layer 3
- It can be safely implemented in any driver/hardware/OS

Path costs

Once there is a link cost metric, given the whole path $P = ETX_1, ETX_2 \ldots ETX_n$ how is the cost $C(P)$ computed?

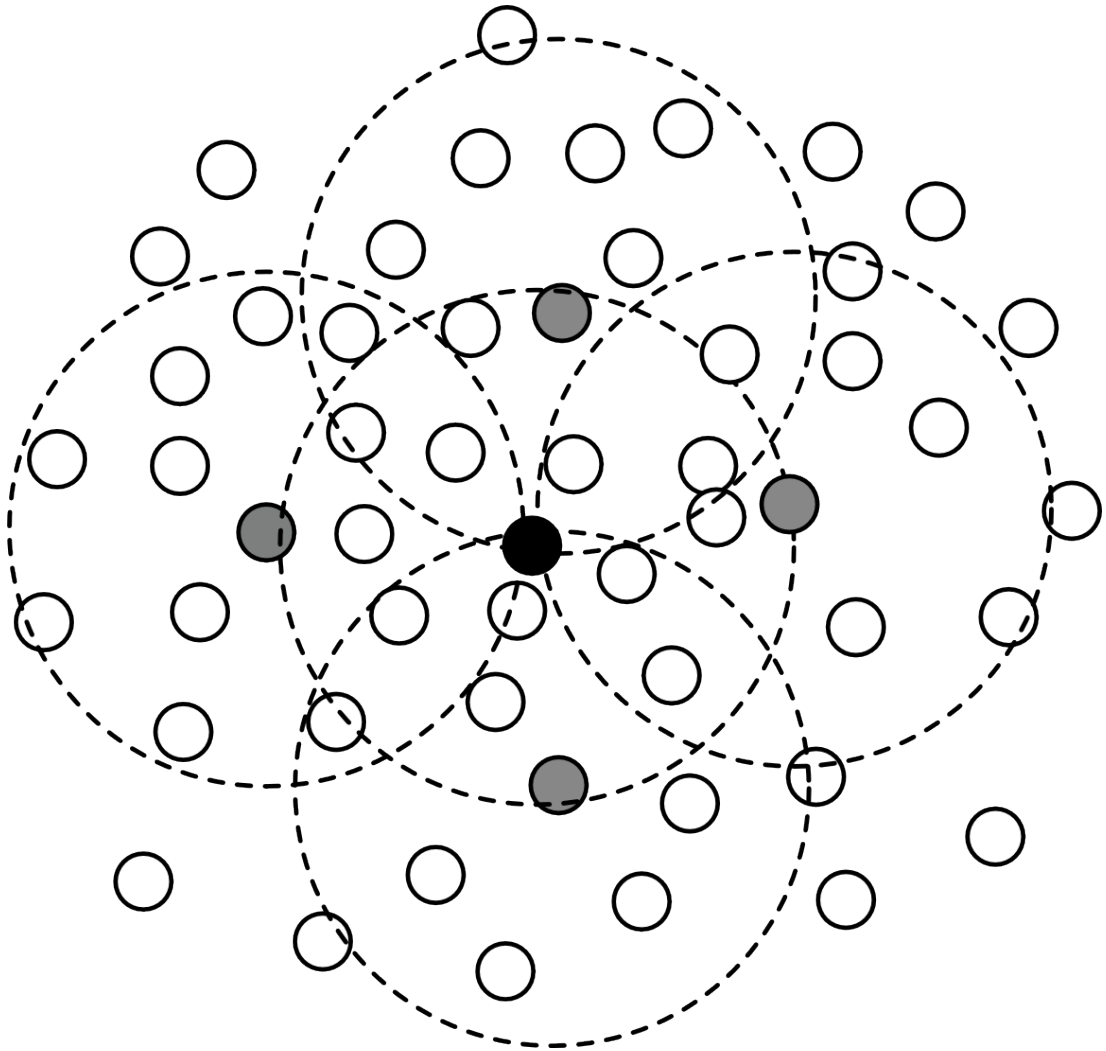
- Sum of the costs. This is the approach OLSR uses.
  - Interpretation: minimising $C(P) = \sum_i ETX_i$ means choosing a route that minimises the total number of packets sent on the network to reach the destination.
  - Tends to choose the shortest route, since every hop adds a fixed quantity.
  - Linear operand does not consider critical badness, sum of acceptable links can equal one single very bad link. Very bad links are generally more problematic (for instance, more instable).
- Product of the costs. This is the approach BATMAN uses.
  - Interpretation: maximising $C(P) = \prod_i \frac{1}{ETX_i}$ means choosing a route that has the highest probability of delivering one packet.
  - It is less sensitive to path lenght, if a route is made of good links it will be chosen compared to short routes with worse links.
- Norm of the costs array. Similar to the approach BMX6 uses.
  - Interpretation: maximising $C(P) = \sum_i ETX_i^2$ means choosing a route that equalises the costs of each link.
  - Avoids routes for a single very bad link.

Metric: example

Path Path 1	1	1	3	1	1	x	x	$\sum$	$\prod$	$\dot{\|\|\|\|}$
Path 2	1	1	2	1	1	1	1	x		
Path 3	1	1	1	1	1	1.3	1.3	1.3		

QOLSR and MPRs

- By design, MPR nodes are chosen on the border of the coverage area of the selector
- This doesn't make them a good candidate to have high quality links



MPRs with QOLSR

Electing MPR with QOLSR:

- MPR nodes in QOLSR are used not only for broadcast messages, but to route unicast messages, so they must be chosen for the quality of their link with the selectors.
- A node may choose a set of MPR only for broadcast flooding trying to minimize the n^2 coverage, and a distinct set of MPR nodes trying to maximize the quality of the links.
- This will increase the control traffic but will also provide better routes. This is what OLSRv2 does.

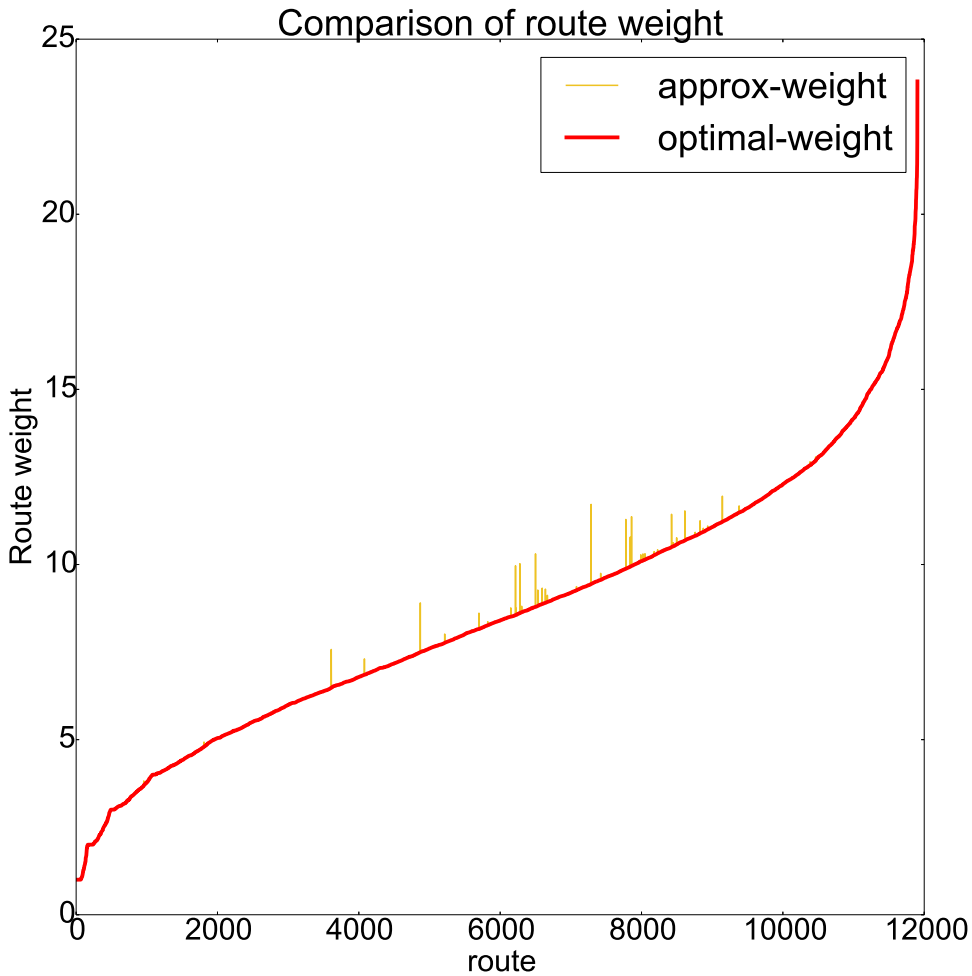
MPRs with QOLSR

Actually, we have shown that this is not needed for real networks:

- The number of routes from node A to B is not so high (real networks are not very dense)
- Routing is done hop-by-hop and the errors due to partial knowledge at sending node is compensated along the path

MPRs with QOLSR

Graz Network. Comparison of weight chosen by QOLSR with MPR choice as per RFC and the best route available.



OLSRv2

- OLSR is under revision to produce a new RFC including:
  - alignment with other MANET RFC
  - QoS features
- see draft-ietf-manet-olsrv2-14

Contact me!

- email: leonardo.maccari@unitn.it
- web: http://disi.unitn.it/maccari

OLSR: Optimized Link State Routing

Leonardo Maccari

leonardo.maccari@unitn.it

25/5/2015