Corso di Reti Mobili

Reti Ad Hoc e Reti di Sensori

Paolo Santi

Istituto di Informatica e Telematica del CNR, Pisa, ITALY

paolo.santi@iit.cnr.it

Istituto di Informatica e Telematica

Clustering Algorithms

for Wireless Ad Hoc and Sensor Networks

Istituto di Informatica e Telematica

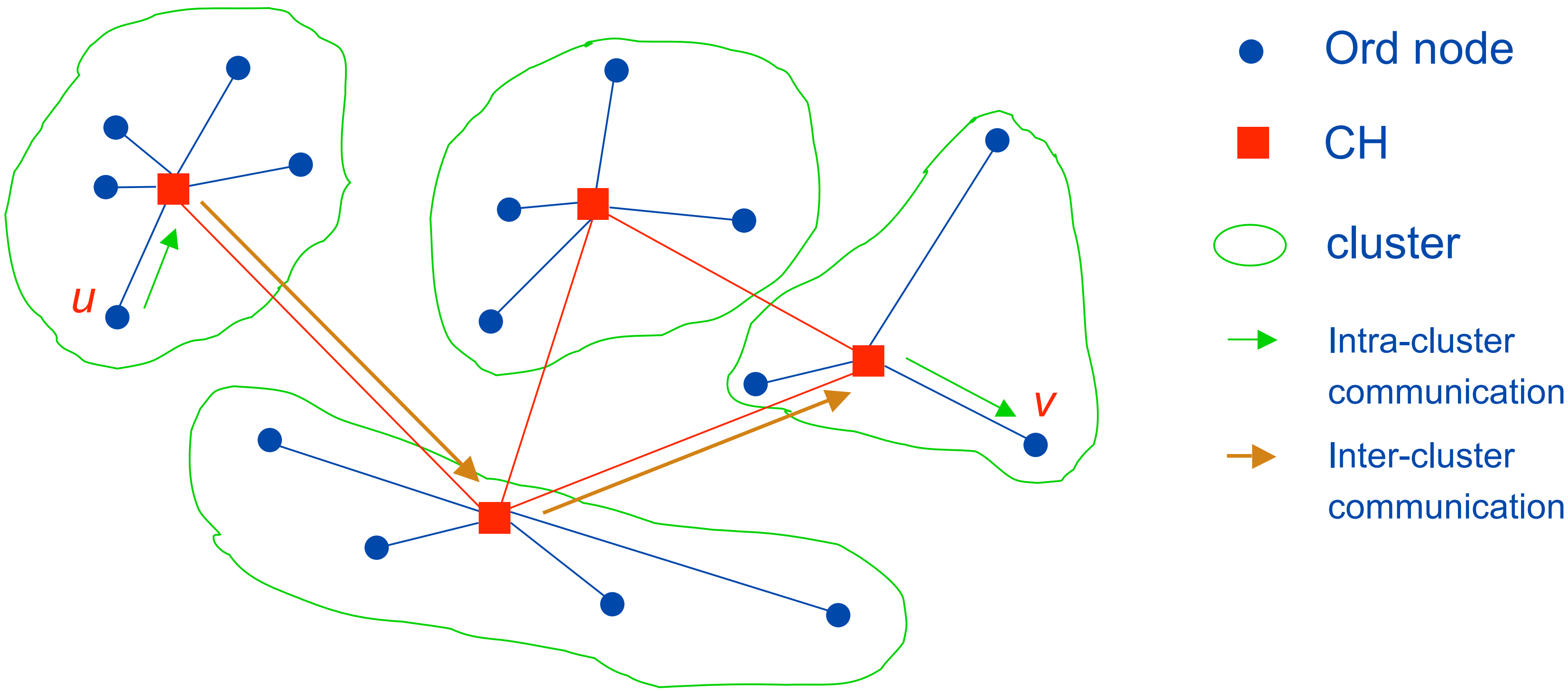
What is clustering?

- Clustering algorithms are used in ad hoc and sensor networks to impose a hierarchy on top of an otherwise flat network organization
- Typically, the hierarchy is obtained by assigning roles to nodes, where the typical roles are Cluster Head (CH) and Ordinary Node (Ord)
- A clustering algorithm is a protocol for dynamically assigning roles to nodes in the network
- Clustering algorithms differ on the rules that are used to assign the roles to the nodes, and to maintain the cluster organization in presence of dynamic network conditions

Motivations for clustering

- What are the advantages of using clustering algorithms?
  - By imposing a hierarchy on the network, certain network-level functionalities such as routing can be simplified.
  - Also, CH can be used to coordinate operations in the cluster, such as:
    - ✓ Coordinate communications between cluster members
    - ✓ Channel access control
    - ✓ Collecting measurements from Ord nodes (for sensor networks)
    - ✓ Coordinate node sleeping times (for sensor networks)
- Summarizing, clustering algorithms can be used to improve the scalability of several ad hoc and sensor network protocols

Clustering and routing



Clustering and routing (2)

- Using clustering algorithms, the routing task is significantly simplified
- Essentially, routing can be seen as an up to three-phases process
  1. Route the packet to the CH (not necessary if the sender is a CH)
  2. Route the packet to the CH to which the destination node belongs
  3. Route the packet to the destination node (not necessary if the destination is a CH)
- Phases 1 and 3 involve intra-cluster communications, while Phase 2 involves inter-cluster communications

Clustering and broadcast

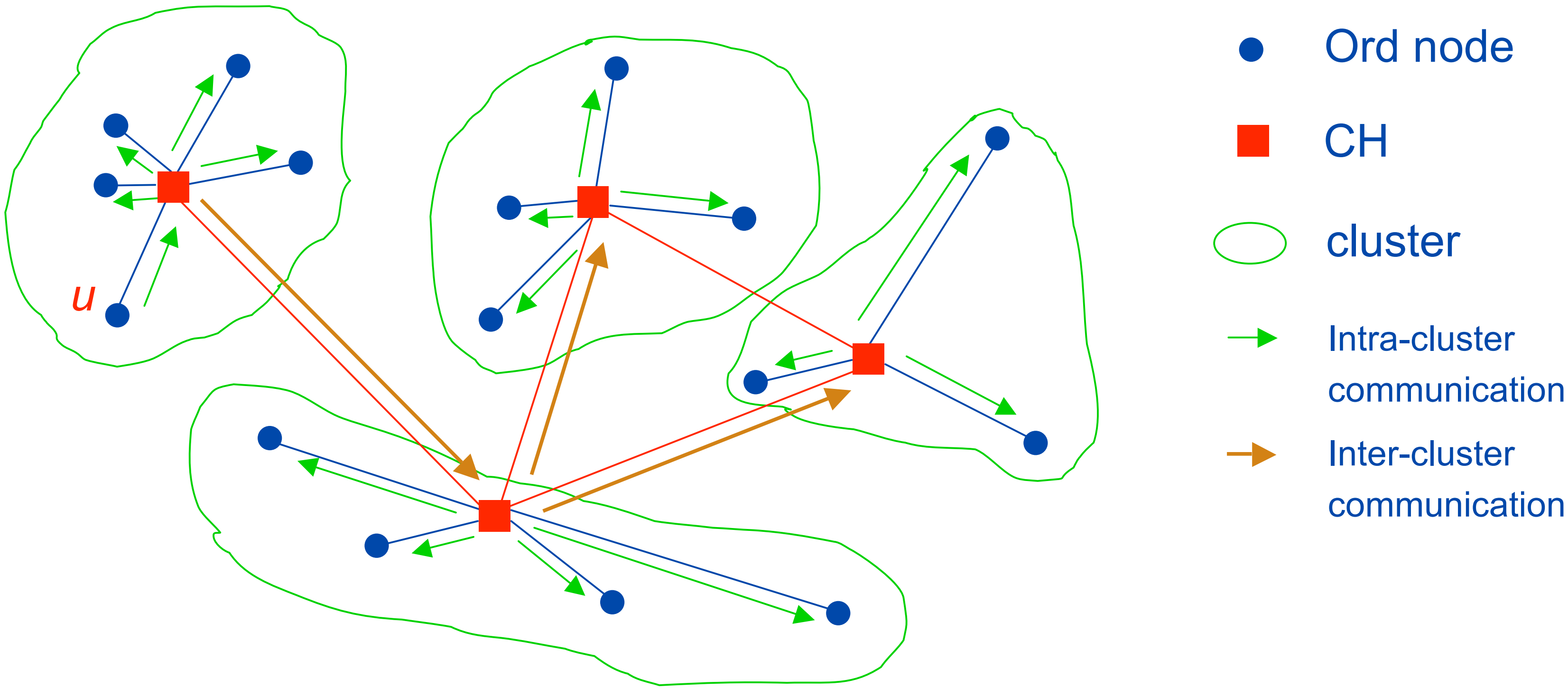
- In other words, the set of CH in the network can be seen as a virtual backbone, which can be used to simplify the task of sending unicast and broadcast messages in the network

- Example of broadcast in a clustered network:

  1. Send the packet the packet to the CH (not necessary if the broadcast originator is a CH)
  2. Broadcast the packet to all the CHs
  3. Each CH broadcasts the received packet to all its Ord nodes

- Similarly to the case of unicast, Phases 1 and 3 involve intra-cluster communications, while Phase 2 involves inter-cluster communications

Clustering and broadcast (2)



Design choices

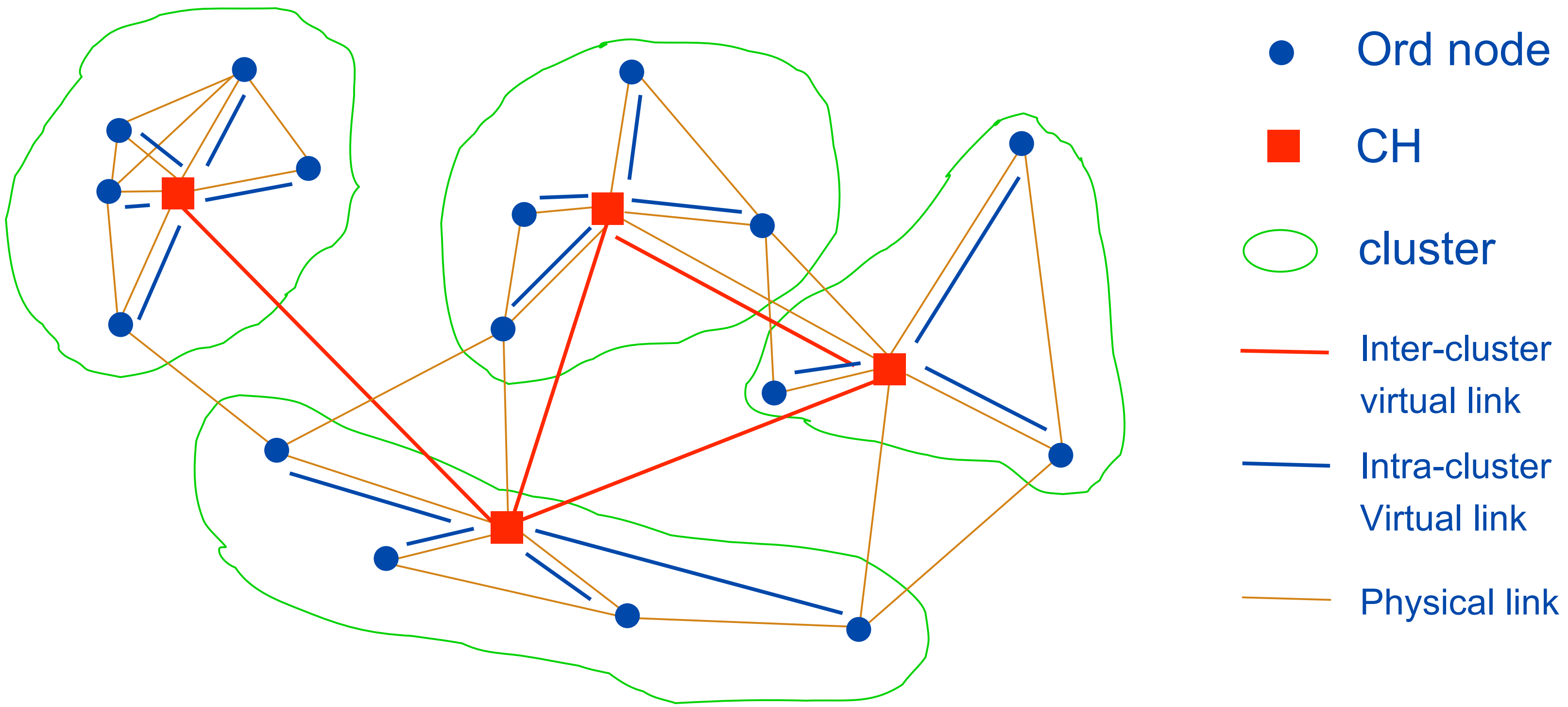
There are several design choices that must be faced when designing a clustering algorithm:

- How to deal with intracluster communication: suppose two nodes within the same cluster want to communicate with each other; should the communication be routed through the CH, or is direct communication between Ord nodes allowed?

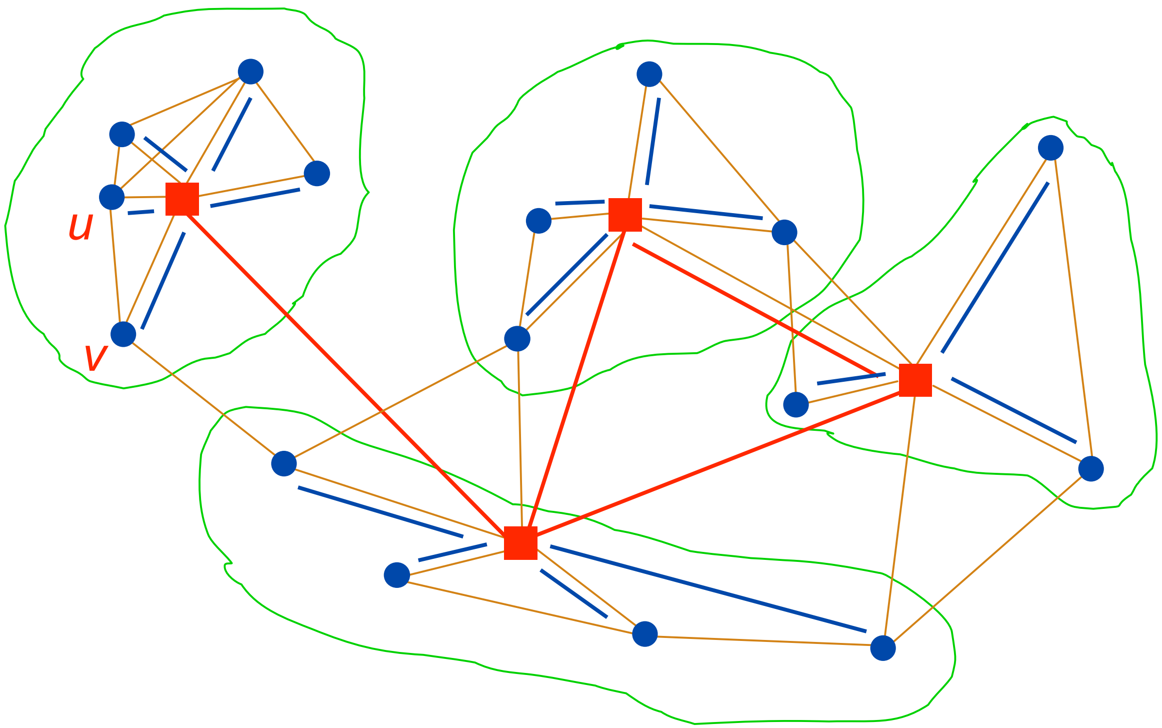
- Mapping of the virtual backbone on the physical network: what described so far is a logical organization of the network topology: in other words, those represented in the previous examples are logical, and not necessarily physical, links. Which rules should govern the mapping between the logical and the physical links?

- Role assignment and maintenace: What are the rules used to assign and dynamically maintain the roles of CH? Which are the desired properties of the virtual backbone?

Physical and logical links



Intra-cluster communications



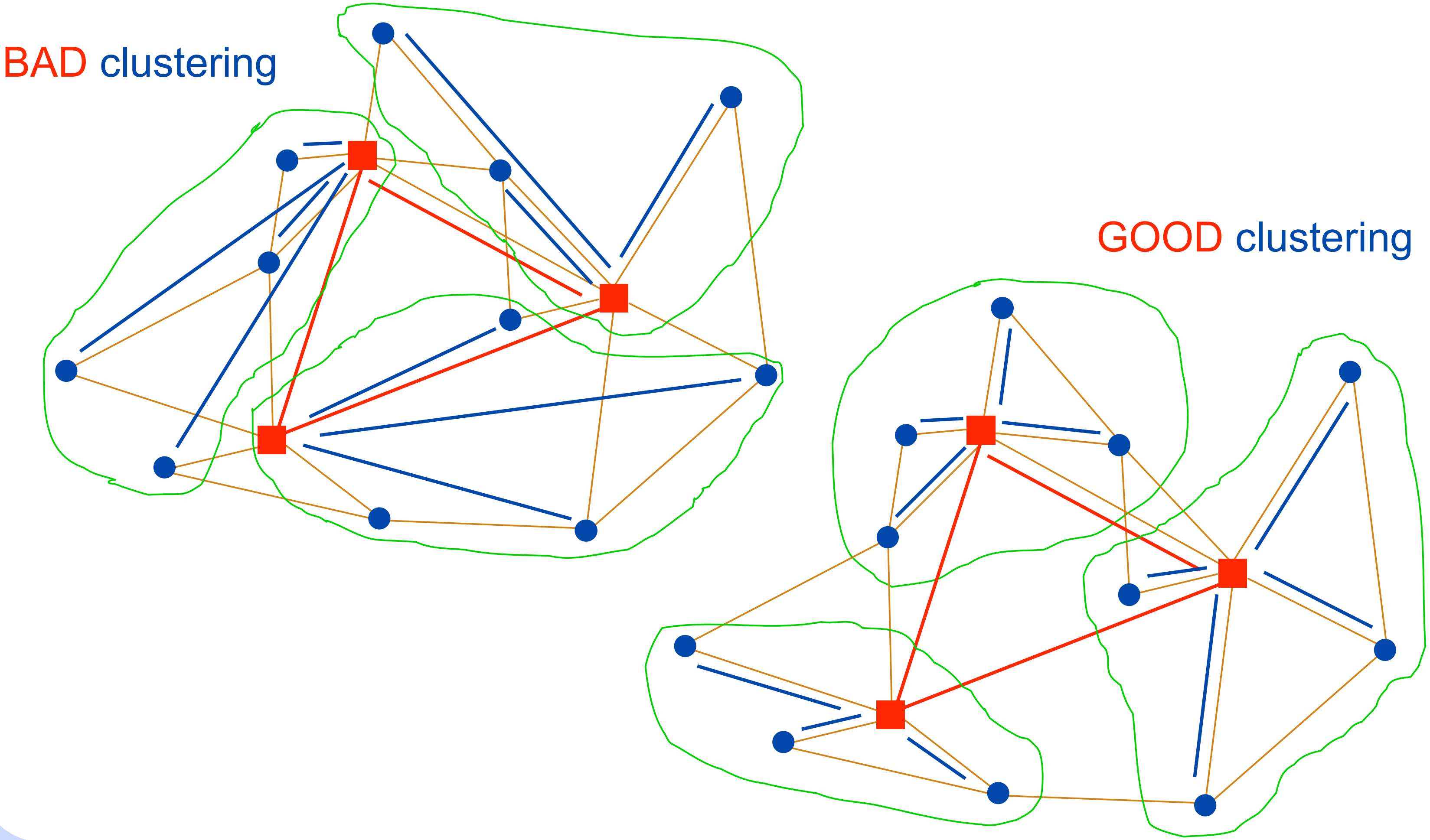
Node *u* wants to send a packet to node *v*. There is no logical links between them, but there is a physical link. Should the nodes use the physical link?

At least, they should inform the CH of their intention to communicate!

Mapping of logical links

- Which are the rules to follow for mapping logical into physical links?
- It is desirable that logical links are as close as possible to physical links
- First rule: nodes composing a cluster should be "close" to each other in the physical network!

Mapping of logical links (2)

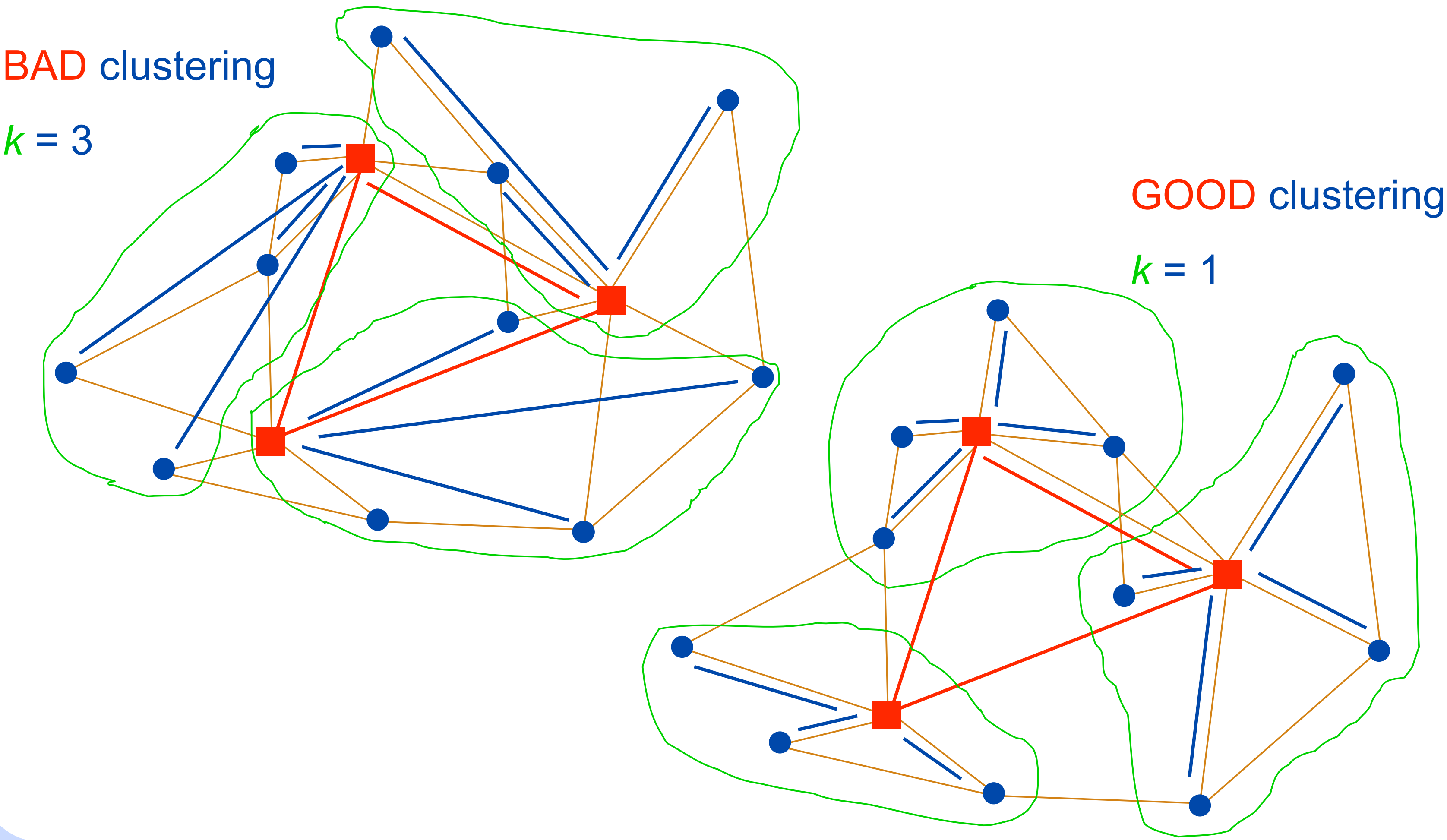


K-Neighbor clustering

- Typically, it is required that ordinary nodes are at most *k* hops away from their CH in the physical network, where *k* is a very small constant (2-3 at most)

- The most common choice is to set *k* = 1, i.e. every Ord node must be adjacent to its CH

K-Neighbor clustering (2)

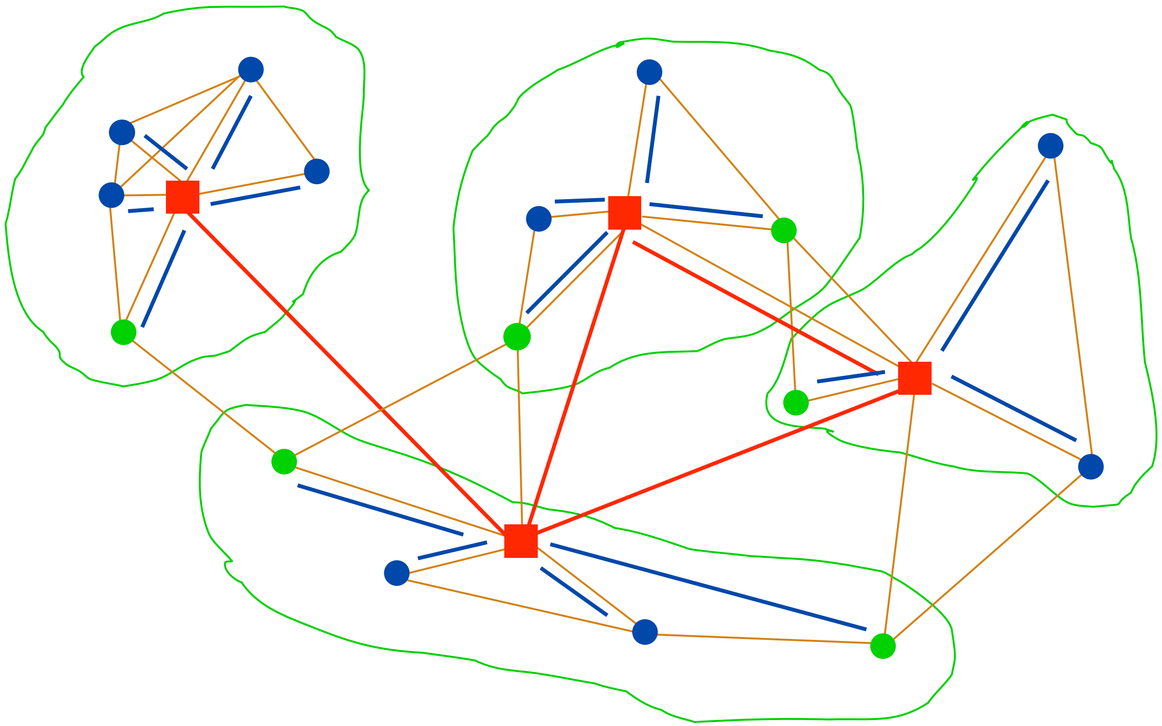


Istituto di Informatica e Telematica

Mapping inter-cluster logical links

- Similar rules apply when defining the mapping of logical links between CH, i.e. when defining the virtual backbone
- In general, it is assumed that a logical inter-cluster link corresponds to a path of length at most *h* in the physical network, where *h* is a small constant
- Typically $h \geq k$: CHs are further away from each other than ordinary nodes from their CH

Gateway nodes



NOTE:

If h > 1, some of the Ord node must act as gateways (GW) between different clusters

● GW nodes

Role assignment

- Another important design choice concerns role assignment. In particular, should Ord nodes belong only to one cluster, or can they belong to multiple clusters?
- The most common choice is to uniquely assign Ord nodes to clusters, i.e. the set of clusters is a partitioning of the network nodes
- Then, in case an Ord node is in close proximity of several CH, we need a rule to uniquely identify the CH (i.e., cluster) to which the node must belong
- In the following, we assume that every Ord node is assigned to a unique CH

Clustering: desired properties

- What are the desirable properties of the cluster structure (output of a clustering algorithm)?
  - Good mapping: the logical structure induced by the cluster organization should resemble the physical structure of the network (*k* and *h* small constants)
  - Load balancing: since a CH in principle experiences a load which is proportional to the number of Ord nodes within its cluster, the number of nodes in the various clusters should approximately be the same
  - Stability: node clustering should be resilient to dynamic network conditions (nodes join/leave) and to node mobility

Clustering and Dominating Sets

Assumption:

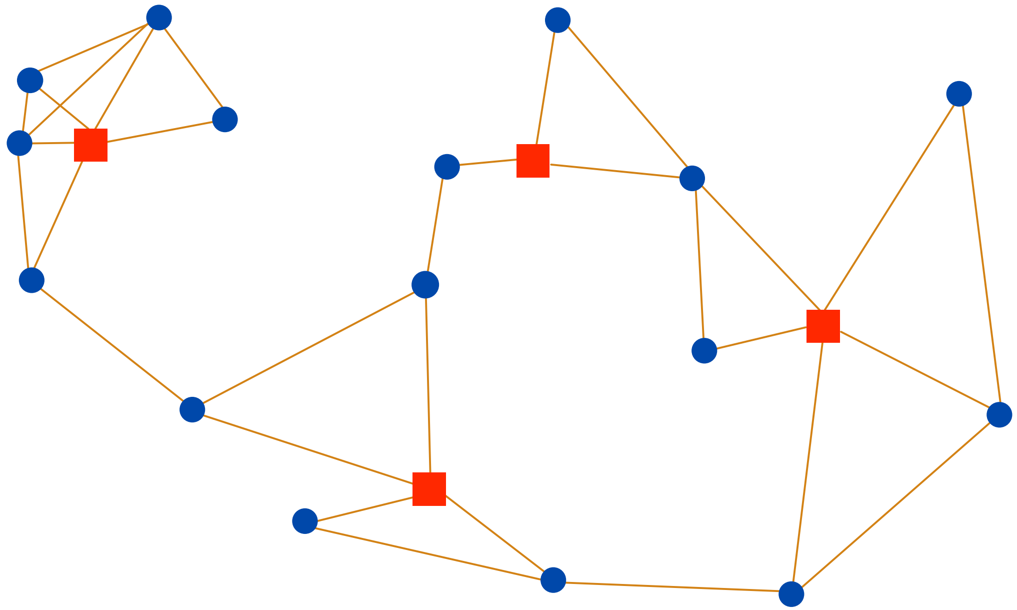
k = 1 (Ord nodes must be adjacent to their CH)

- Denote with G = (N,E) the communication graph. Then a feasible clustering for *G* is a dominating set D for graph G

Dominating Set (DS) for G = (N,E) :

subset $D \subseteq N$ such that for every node u in N - D exists edge $(u,v) \in E$ such that $v \in D$

Dominating set



In this example,
CH nodes are a
dominating set
of the communication
graph

Connected Dominating Sets

Assumption:

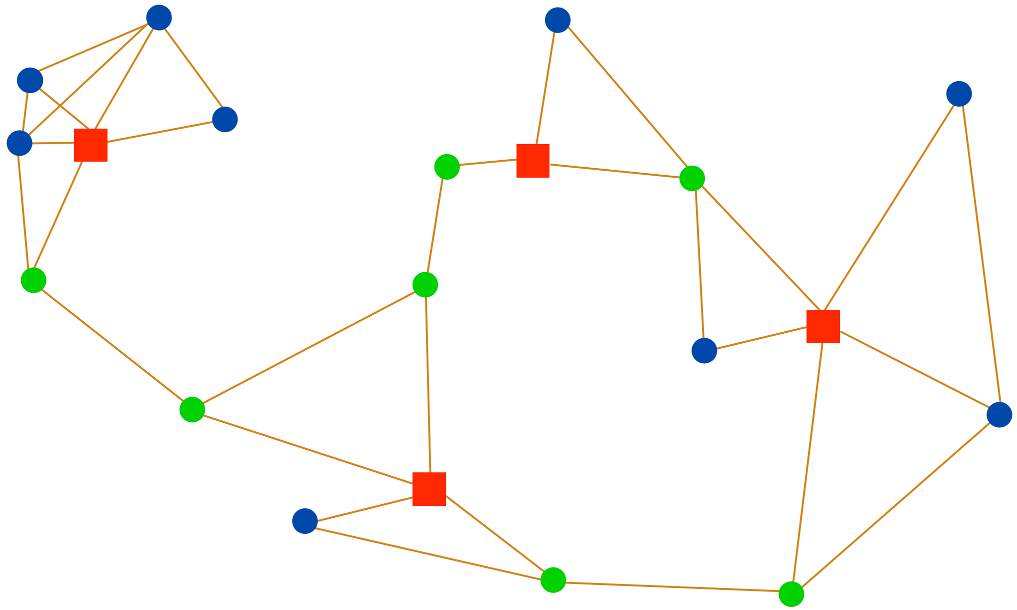
k = 1, and *h* = 1 (also CHs must be adjacent)

- Denote with G = (N,E) the communication graph. Then a feasible clustering for *G* is a connected dominating set C for graph G

Connected Dominating Set (CDS) for G = (N,E) :

subset $C \subseteq N$ such that for every node *u* in N - C exists edge $(u,v) \in E$ such that $v \in C$, and the subgraph of G induced by node set C is connected

Connected dominating set



Now, CH nodes form a dominating set, but they are not connected

Need to include also green nodes to form a connected dominating set

Finding MCDS

Assumption:

k = 1, and *h* = 1

- In this context, one possible way of optimizing the clustering is to reduce as much as possible the number of CHs
- Intuitively, for a certain number *n* of nodes in the network, the more the CHs, the less efficient is the clustering (think about the broadcast example)

MCDS Problem:

Given a communication graph G, find a CDS of minimum cardinality

Another application of MCDS

- Other important application of MCDS: extend network lifetime in wireless sensor networks:
  - It is well known that considerable energy can be saved if the radio is shut down in sensor nodes
  - Idea: alternately shut down node radios in order to improve lifetime
  - Clearly, some of the nodes must have their radio on, otherwise the network functionality is impaired
  - Awake nodes should form a CDS. This way, every sleeping node, in case it detects an event, can find a nearby active node to which communicate the detected event. Furthermore, awake nodes form a connected component, i.e. detected events can be communicated network-wide at any time
  - Small size of the CDS is desirable: in principle, the less the nodes in the CDS, the more energy savings can be achieved

Unfortunately, solving the MCDS problem is NP-hard!!

Approximation algorithms for MCDS

- Although optimally solving MCDS is NP-hard, approximated solutions can be computed efficiently
- Several algorithms for computing approximations of MCDS in the literature
- One of the best algorithm presented in the literature can compute a 8-approximation of the optimal solution in a distributed and localized way (up to three-neighbors information is needed), in O(n) time, and exchanging at most O(n) messages (n is the number of nodes in the network).

The DMAC algorithm

- DMAC (Distributed and Mobility Adaptive Clustering) protocol is a simple distributed clustering algorithm for ad hoc networks
- In DMAC, every node is assigned with a weight, which represents, in a sense, its “willingness” to become CH
- Weights can be assigned to nodes depending on several parameters, such as:
  - Node ID
  - Node degree in the physical topology
  - Energy level
  - Combinations of the above parameters

The DMAC algorithm (2)

- Nodes exchange “hello” messages with neighbors, where a hello message contains node ID, weight, and status
- The status can be CH, Ord, or undecided (Und)
- Initially, all the nodes are in Und state
- Also, when a new node joins the network, it sets the status to Und

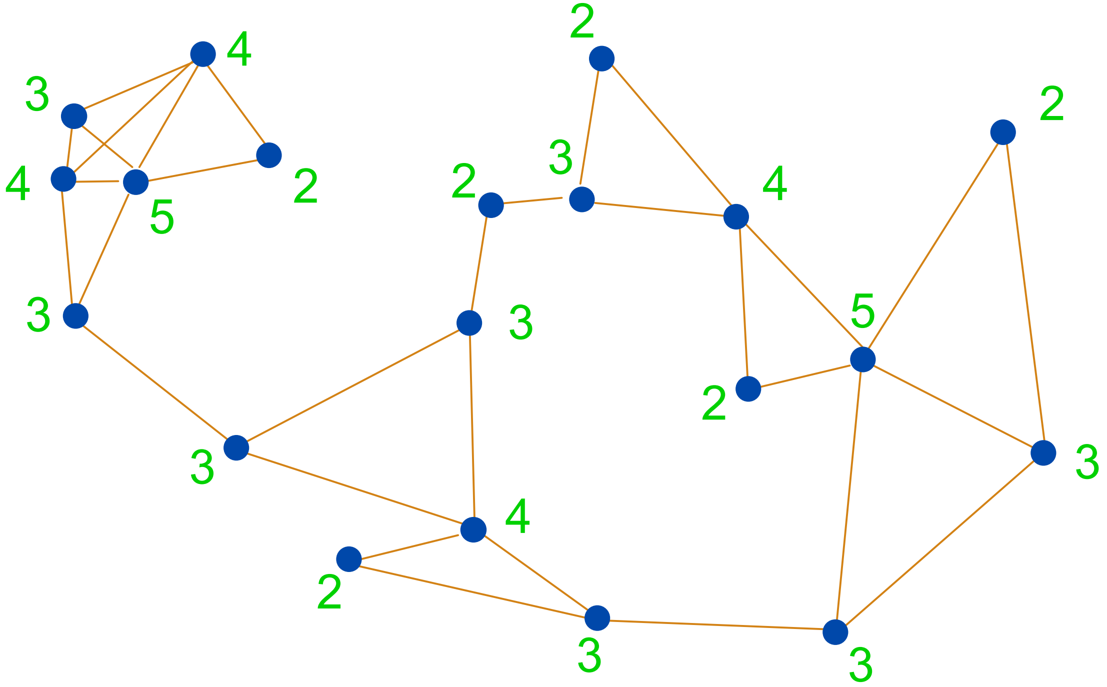
DMAC: Role assignment

- The decision rule for assigning roles to nodes is the following:
  - A node becomes CH if and only if it has the highest weight in its neighborhood (only nodes in CH and Und status are considered)
  - When a node elects itself as CH, all its neighbors become Ord nodes
- The following association rule is used to decide which cluster an Ord node must join:
  - Any ordinary node join the CH with highest weight in its neighborhood

DMAC: Properties

- It is immediate to see that, given DMAC decision rule, every node in the network is either a CH, or it is adjacent to at least one CH (i.e., $k = 1$)
- In other words, the set of CHs computed by DMAC is a dominating set for the communication graph (not of minimum cardinality, though)
- Also, it is immediate to see that any two CH nodes in the network cannot be adjacent to each other (i.e., $h \geq 2$)
- The latter property guarantees a well-spread formation of clusters

DMAC example



Node weight = degree

- Und node
- Ord node
- CH node

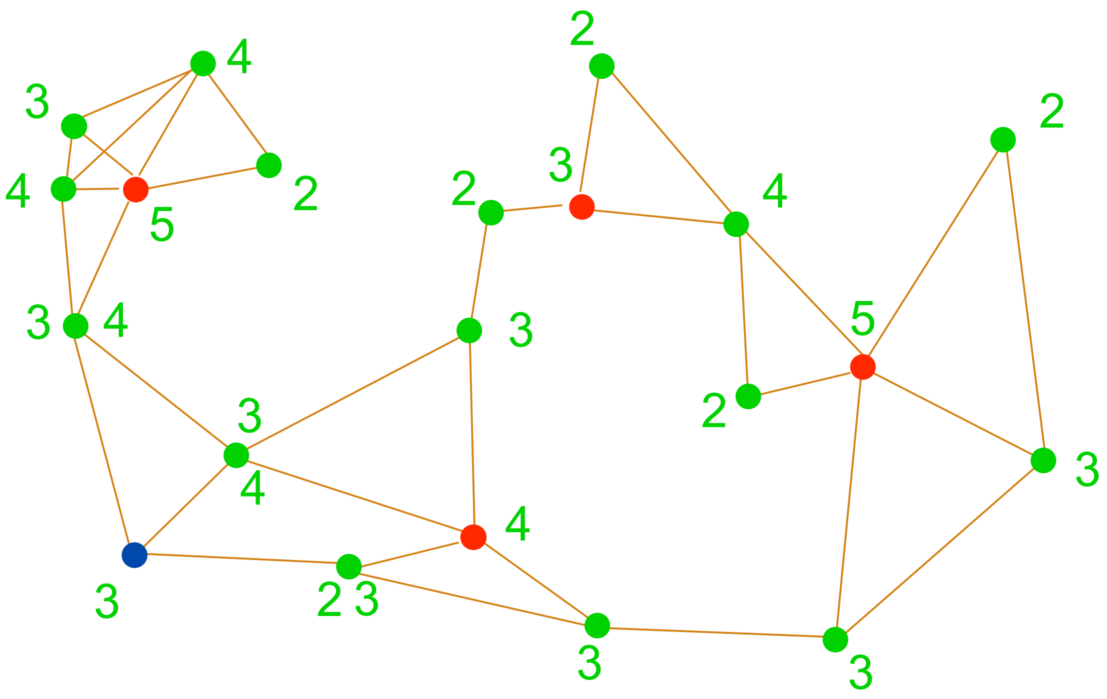
DMAC: Communication overhead

- DMAC is a very simple algorithm, which requires the exchange of only few messages between nodes: number of messages exchanged is O(n), where n is the number of network nodes
- Hence, DMAC can be successfully applied in dynamic scenarios, where nodes can join/leave the network at different time, and/or are mobile
- Let us see how DMAC reacts in presence of new nodes joining the network

DMAC reconfiguration

- When a new node joins an already clustered network, it must determine its role
- First, it exchanges “hello” messages with neighbors, setting its own status to Und
- After it has collected the information on the status and weights of its neighbors, the new node sets its role according to the following rule:
  1. it joins an existing cluster if there exists a neighbor CH with higher weight (status is Ord), or
  2. It forms a new cluster otherwise (status is CH)
- In case 1., the new node joins the CH with highest weight in its neighborhood, sending to this node a JOIN message
- In case 2., the new node informs all its neighbors that it has become a CH sending a CLUSTERHEAD message, possibly causing some of the nodes in the neighborhood to change their associated cluster and/or status

Reconfiguration example



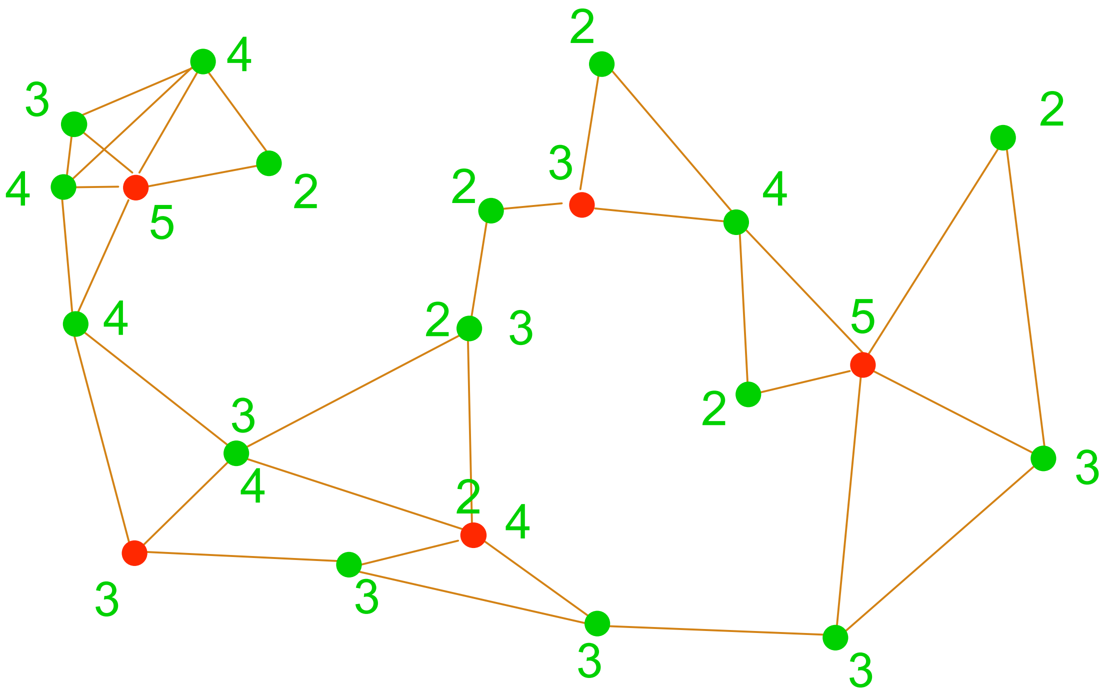
Node weight = degree

- Und node
- Ord node
- CH node

DMAC reconfiguration (2)

- DMAC reconfiguration is also needed to account for other types of dynamic conditions, i.e.:
  - Change in node weights: since node weights are time variant, nodes have to periodically check whether their current status remains valid or not
  - Change in the network topology due to mobility
- Thus, nodes must periodically re-execute the DMAC role assignment algorithm, in order to account for these situations
- The following events might be triggered by DMAC re-execution:
  - Re-association of Ord nodes to other CH nodes
  - Status changes, from Ord to CH, or vice versa

Reconfiguration example (2)



Node weight = degree

- Und node
- Ord node
- CH node

Chain reaction

- As we have seen in the previous example, changes in the role assignments can propagate in the network, causing other nodes to toggle their status
- In general, having frequent status changes in the network is not desirable, since there is always a message overhead entailed by a status change (sending JOIN or CLUSTERHEAD messages)
- To which extent can these status changes propagate in the network?
- Unfortunately, examples can be found in which changing the status of a node initiates a chain reaction, forcing several nodes in the network to toggle their status

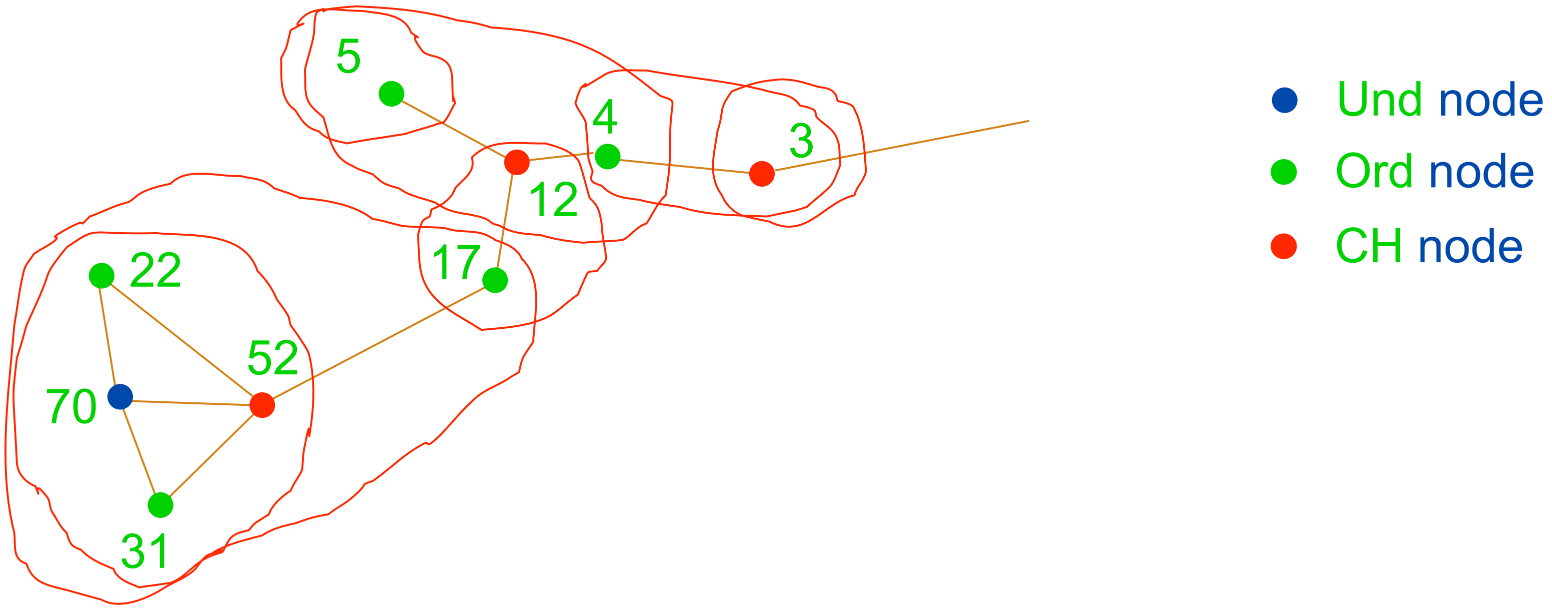
Conditions for chain reaction

- It has been observed that, when a new node appears in the network, the cluster structure can change along a directed path if the following conditions are fulfilled

1. CHs and Ord nodes appear alternately in the path
2. The successor of a node in the path has lower weight than the node itself
3. No Ord node has a CH with higher weight than its own predecessor in the path

The chain reaction is triggered if the new node has higher weight than its neighboring CH nodes

Chain reaction example



Chain reaction (2)

- Observe that the chain reaction can occur only when a new node elects itself as a CH
- No chain reaction occurs when a new node sets its role to Ord: in this case, sending one JOIN message is the only overhead generated by the new node
- In general, is it more probable that a new node becomes CH (risky situation), or that a new node becomes Ord?
- It has been observed through simulation that the event "new node becomes Ord" is much more probable than the event "new node becomes CH"

Improving CH stability

- Since the Und -> CH status change is a critical event for the stability of the cluster structure, the following question is natural:

  Is there any way of improving CH stability in DMAC?

- This leads to the definition of a generalized (and more stable) version of the DMAC protocol, called G-DMAC

G-DMAC

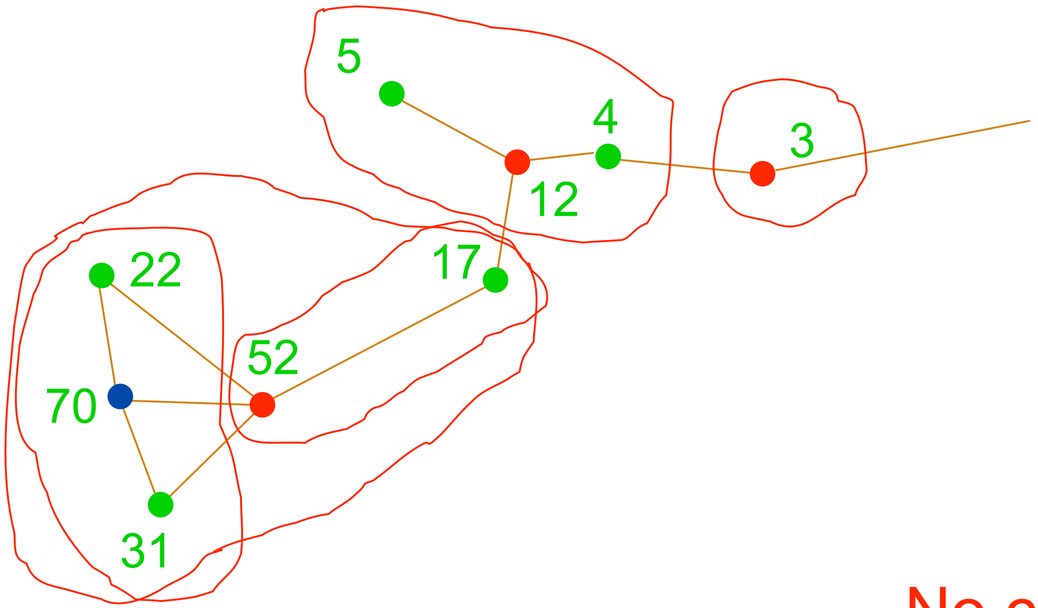
- The basic idea of G-DMAC is to relax the strict DMAC decision rule for assigning roles to nodes

- We recall that DMAC forbids neighboring CH, and forces Ord nodes to join the neighboring CH with higher weight

- In G-DMAC, these rules are relaxed as follows:
  - Up to $S \geq 0$ CHs are allowed to be neighbors
  - An Ord node switches to a newly arrived CH u only when the weight of u exceeds the weight of the current CH by at least T, for some $T \geq 0$

Back to the previous example

G-DMAC with S = 1



- Und node
- Ord node
- CH node

No chain reaction!!!

No chain reaction with G-DMAC?

- In the previous example, using G-DMAC with S = 1 was sufficient to avoid the chain effect
- Is it true that with G-DMAC the chain reaction can always be avoided?
- Unfortunately, the answer to this question is NO: there exist situations in which chain reactions occur also with G-DMAC, for any setting of S and T

Conditions for chain reaction

- A chain reaction along a path occurs with G-DMAC if the following conditions are fulfilled:

1. CHs and Ord nodes appear alternately in the path
2. The successor of a node in the path has lower weight than the node itself and satisfies additional conditions which depend on S and T
3. No Ord node has a CH with higher weight than its own predecessor in the path

Note that condition 2 is more restrictive than in case of DMAC

Hence, chain reactions occur less frequently with G-DMAC than with DMAC