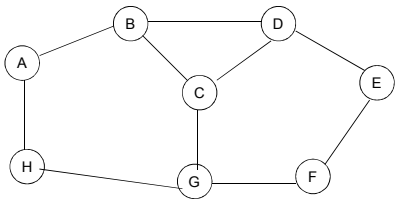
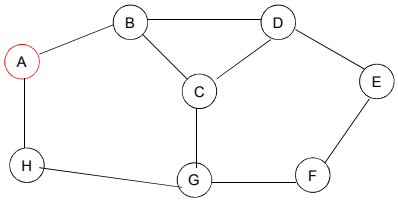
Reliable broadcast algorithms

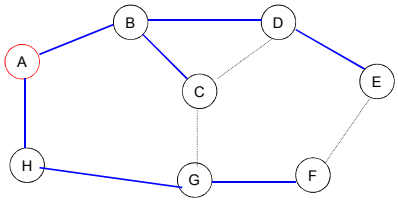
Source dependent broadcast tree



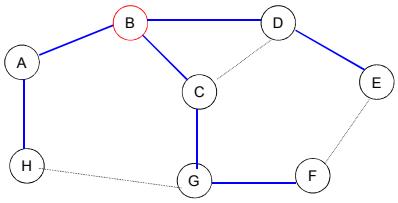
Source dependent broadcast tree (Source A)



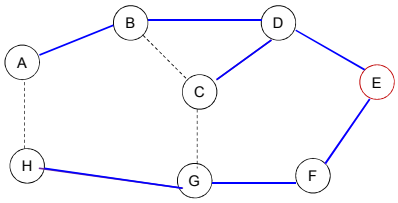
Source dependent broadcast tree



Source dependent broadcast tree (Source B)



Source dependent broadcast tree (Source E)



Nice approach

- Source independent broadcast tree
  - (Tree minimizing number of forwarding nodes)

Optimum broadcast tree problem

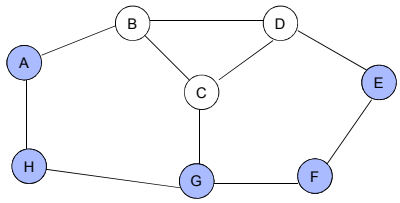
- Very similar to Minimum Connected Dominating set (MCDS) problem

Connected Dominating Set (CDS)

- CDS is defined as follows:
  - Let a graph is G(V,E)
  - CDS is a set of vertices S such that
    - $S \subseteq V$
    - *S* is connected.
    - All elements in V – S is adjacent to at least one element of S

Some example CDS

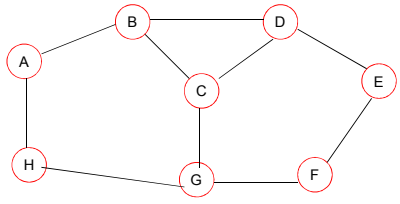
V= { A, B, C, D, E, F, G, H }



S1 = { A, H, G, F, E} S2 = V - S1 = {B, C, D}

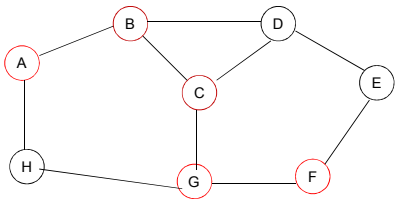
Some example CDS

Trivial CDS



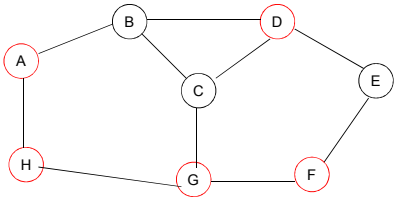
S = { A, B, C, D, H, G, F, E}

Some non-trivial examples



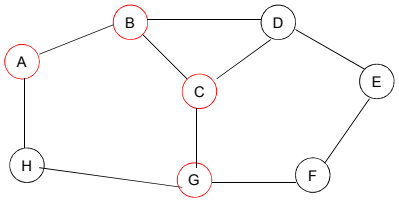
CDS = { A, B, C, G, F}

Some more examples



S = { A, D, H, G, F}

Some more examples

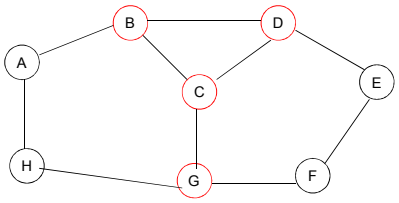


S = { A, B, C, G}

Minimum Connected Dominating Set (MCDS)

- Must be a CDS
- Having minimum number of nodes

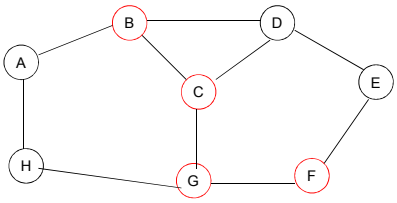
Minimum Connected Dominating Set (MCDS)



There is another MCDS, can you find it?

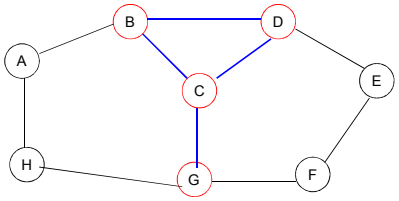
S = { B, C, D, G}

Another MCDS



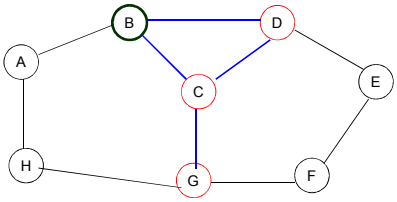
S = { B, C, G, F}

MCDS can be used for broadcast!



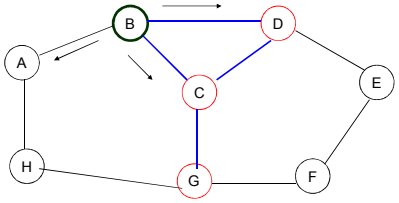
S = { B, C, D, G}

Source is an element of MCDS



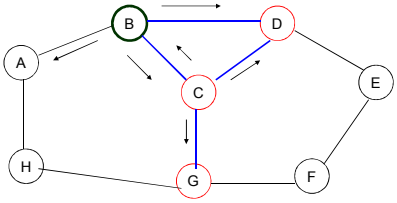
S = { B, C, D, G}

Source is an element of MCDS



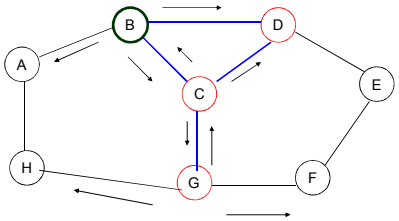
S = { B, C, D, G}

Source is an element of MCDS



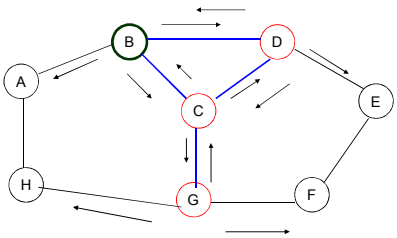
S = { B, C, D, G}

Source is an element of MCDS



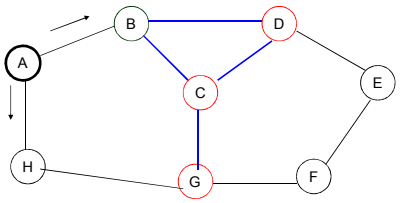
S = { B, C, D, G}

Source is an element of MCDS



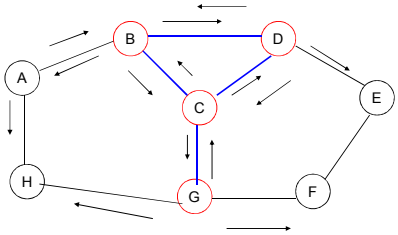
S = { B, C, D, G}

Source is not an element of MCDS (Say A)



S = { B, C, D, G}

Source is not an element of MCDS



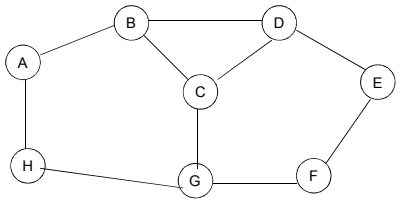
S = { B, C, D, G}

MCDS for broadcast

- Number of forwarding:
  - Best case: |MCDS|
  - Worst case: 1+|MCDS|
- Unfortunately finding MCDS is an NP-complete problem.
- Approximation algorithms are available

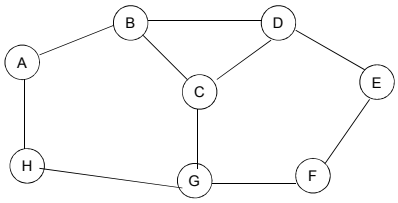
Can you think of an exact algorithm for MCDS construction?
It doesn't have to be in polynomial time….

Approximation of MCDS



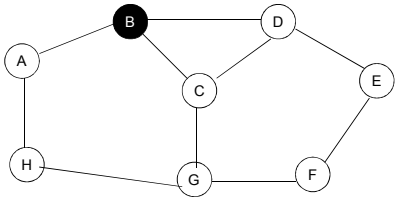
All nodes are colored white

Approximation of MCDS



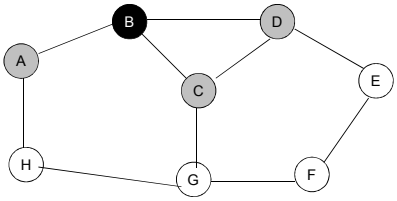
Pick up the node having max white neighbors, and paint it to black

Approximation of MCDS



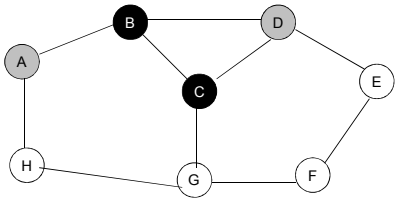
Color all its neighbors to gray

Approximation of MCDS



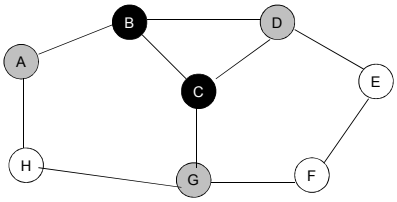
Pick up one of the gray nodes having most white neighbors and color it to black

Approximation of MCDS



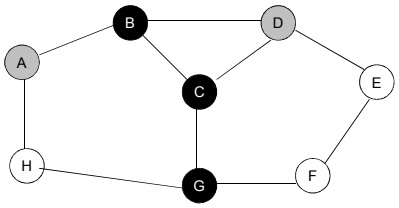
Paint all neighbors of newly picked node to gray

Approximation of MCDS



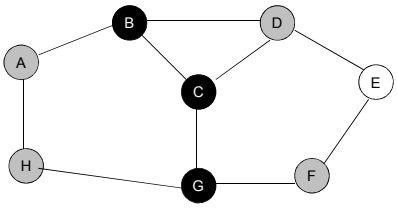
Continue this process…..

Approximation of MCDS



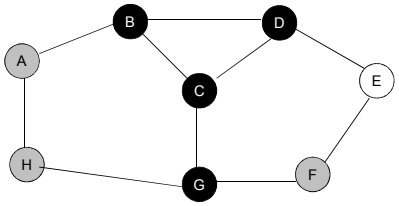
Continue this process…..

Approximation of MCDS



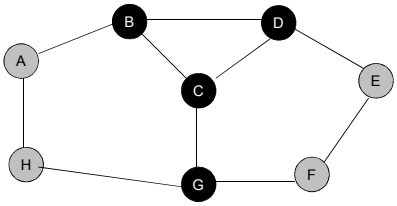
Continue this process…..

Approximation of MCDS



Continue this process…..

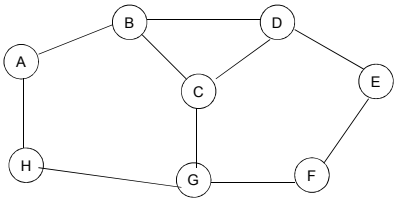
Approximation of MCDS



No white nodes

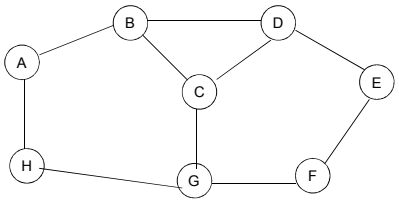
Order of selection is very important!!

Approximation of MCDS



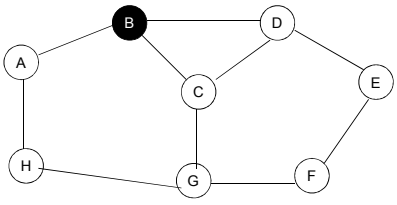
All nodes are colored white

Approximation of MCDS



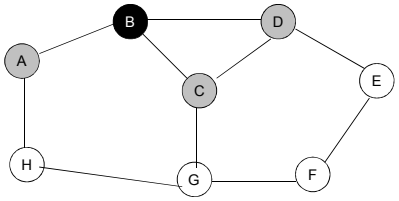
Pick up the node having max white neighbors, and paint it to black

Approximation of MCDS



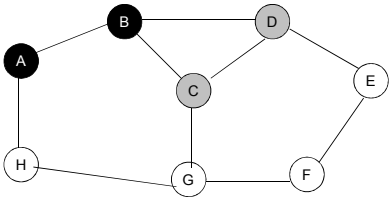
Color all its neighbors to gray

Approximation of MCDS



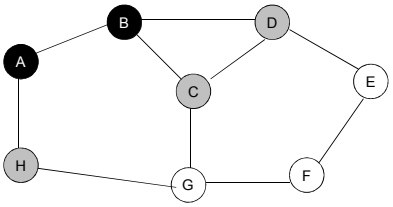
Pick up one of the gray nodes having most white neighbors and color it to black

Approximation of MCDS



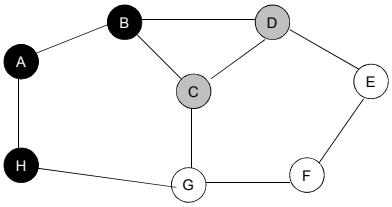
Paint all neighbors of newly picked node to gray

Approximation of MCDS



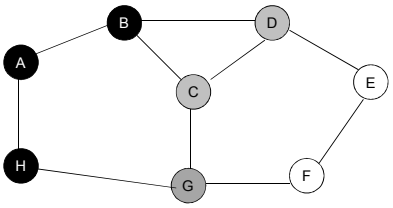
Continue this process…..

Approximation of MCDS



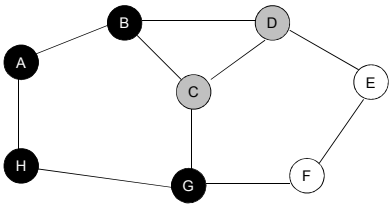
Continue this process…..

Approximation of MCDS



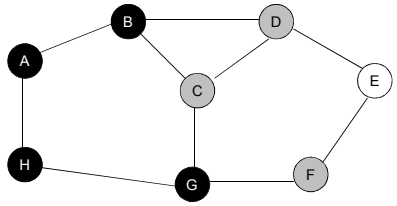
Continue this process…..

Approximation of MCDS



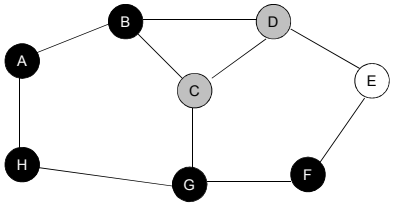
Continue this process…..

Approximation of MCDS



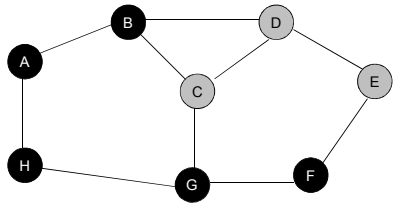
Continue this process…..

Approximation of MCDS



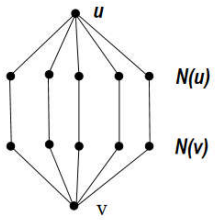
Continue this process…..

Approximation of MCDS



Continue this process…..

Approximation algorithm may not work well!



Approximation of MCDS

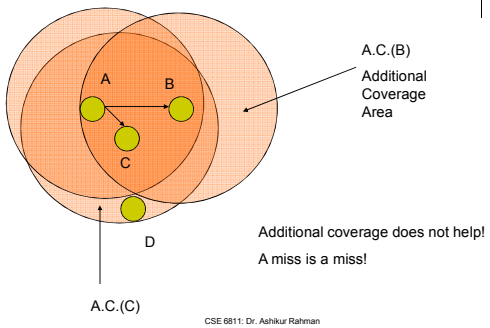
- Approximation is applicable when we have overall topology information
- Obtaining overall topology information in ad hoc wireless network is like “shooting your own foot”.
- We need to develop some heuristics which can work using small topology information

Two heuristics that create CDS in a distributed way

- Self pruning (reactive)
- Dominant pruning (proactive)

- Reactive vs proactive
  - You proposed a topic for your project (reactive approach)
  - I forced you to work on a particular topic (proactive approach).

Self-pruning philosophy



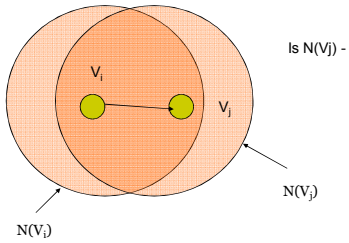
Additional coverage does not help!

A miss is a miss!

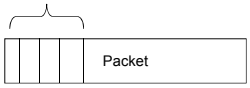
Self Pruning

- Assumptions:
  - A node knows all of its 1-hop neighbors
    - Periodic hello messages
  - When a node generates a broadcast message, it appends its list of neighbors in the header.
  - Neighbor list gets replaced each time a forwarding occurs

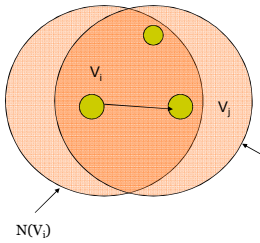
Know before you decide….



Is $N(Vj) - N(Vi)-V_i$ empty?

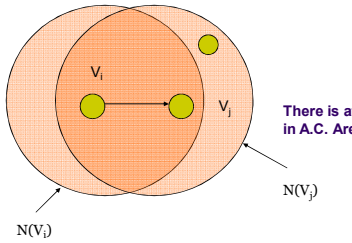


Redundant case



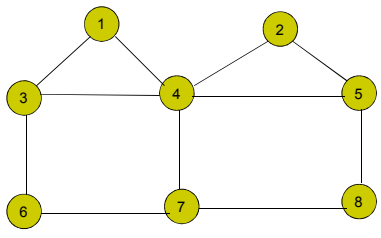
Only common neighbor(s)

Non-redundant case

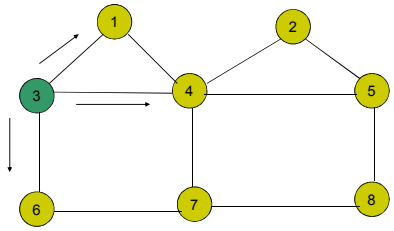


There is at least one neighbor in A.C. Area

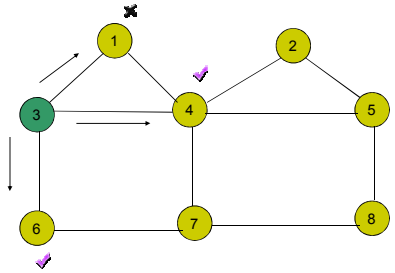
Example (Source 3)



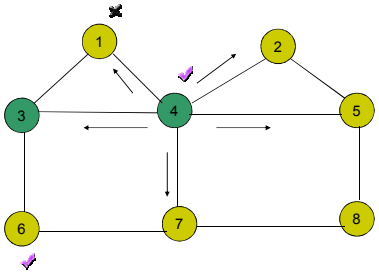
Example



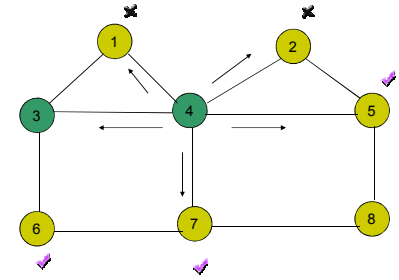
Example



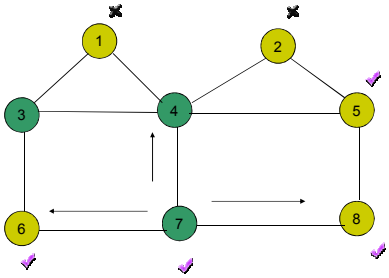
Example



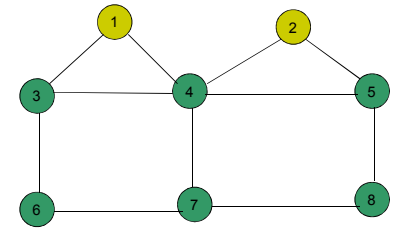
Example



Example

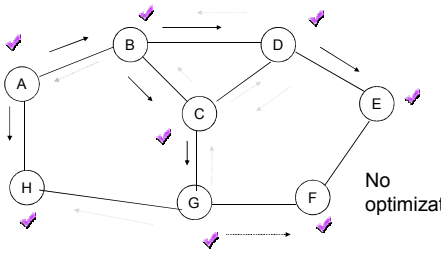


Example



6 forwardings

How good self-pruning is?

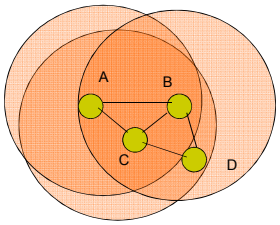


No optimization!

More optimization possible…. (proposed in another work)

N(B)-N(A)-{A} = {D}

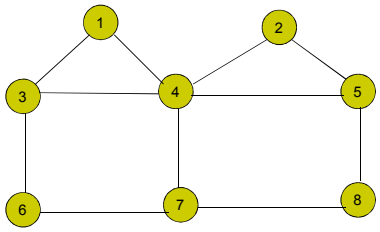
N(C)-N(A)-{A} = {D}



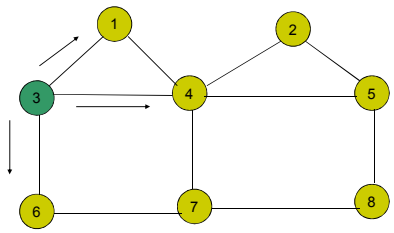
More optimization Algorithm

- Suppose v receives a packet m from u
- Calculate C(v,m) = N(u) U {u}
- Wait for a time T before taking decision
- During wait suppose v receives same broadcast from w
  - Update C(v,m) = C(v,m) U N(w) U {w}
  - Is N(v) a subset of C(v,m)?
    - If yes then drop
    - Otherwise continue waiting
- End of waiting rebroadcast
- T can be set intelligently

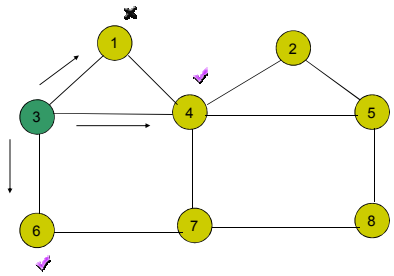
Example



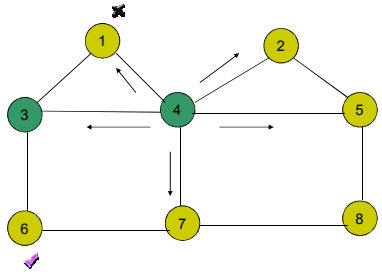
Example



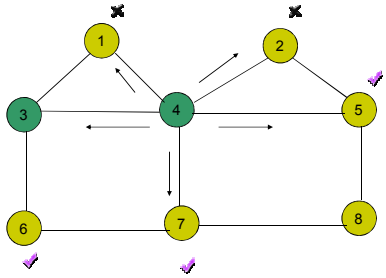
Example



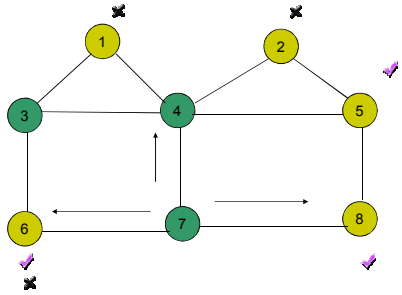
Example



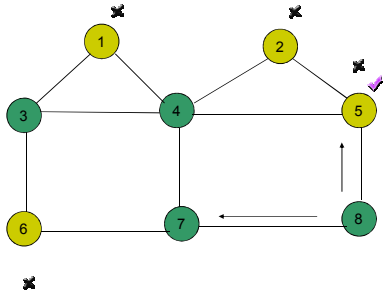
Example



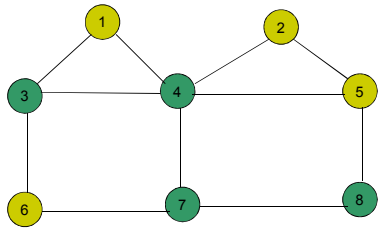
Example



Example



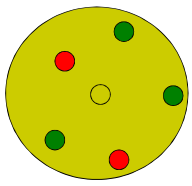
More optimization applied to the previous example



4 forwardings

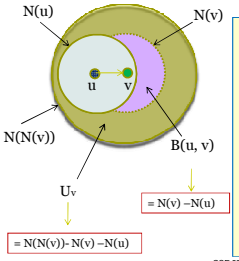
Dominant pruning (DP)

- Proactive
- A node selects some of its neighbors for forwarding. (based on some criteria)
- The selected set of nodes are called forward list and specified in the packet header
- Need 2-hop information
  - N(u), N(N(u))



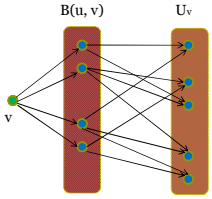
Dominant pruning explained

- Suppose node *v* received a broadcast from *u*.
- Node *v* creates a list of neighbors who should re-forward after receiving from v.



- Here is how *v* creates the forward list
  - Among all its 2-hop neighbors, nodes in *N(u)* already received the broadcast from *u*.
  - All its 1-hop neighbors in *N(v)* will receive when *v* will broadcast.
  - Among its all 1-hop and 2-hop neighbors the following set of nodes U_v haven't received yet: $U_v = N(N(v)) - N(v) - N(u)$
  - The forward list is the minimum subset of B(u,v) such that each node in U_v is neighbor of some node in B(u,v).

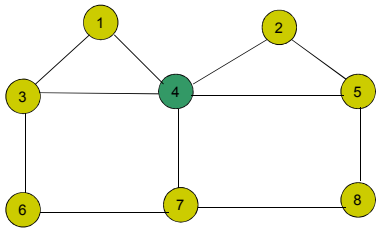
Can be mapped to set cover problem



DP algorithm

1. Let $F = \emptyset$, $K = \{S_1, S_2, \cdots, S_n\}$ where $S_k = N(v_k) \cap U (1 \le k \le n)$, $Z = \emptyset$.
2. Find the set S_k whose size is maximal in a set K.
3. $F = F \cup \{v_k\}$, $Z = Z \cup S_k$, $K = K - \{S_k\}$, $S_l = S_l - S_k$ for all $S_l \in K$.
4. If $Z = U$, complete the algorithm.
5. Otherwise, repeat from 2 again.

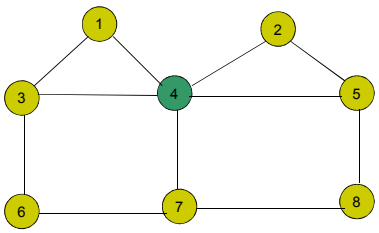
DP Algorithm in action



N(4) = {1,2,3,4,5,7}

N(N(4)) = N(4) U {6,8}
={1,2,3,4,5,6,7,8}

DP Algorithm in action



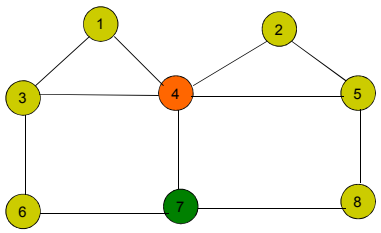
B = {1,2,3,5,7}

K = {{6}, {8}, {6,8}}

U = {6,8}

Choose 7

DP Algorithm in action



N(7) = {4,6,7,8}

N(N(7)) = {1,2,3,4,5,6,7,8}

N(4) = {1,2,3,4,5,7}

U = N(N(7)) – N(4) – N(7) = { }

Comparison Self-pruning Vs. Dominant pruning

- Reactive vs. Proactive
- Overhead –
  - winner Self-pruning
- Optimization –
  - winner Dominant pruning
- Complexity –
  - winner Self-pruning
- Frequent topology changes (high mobility)-
  - winner Self pruning