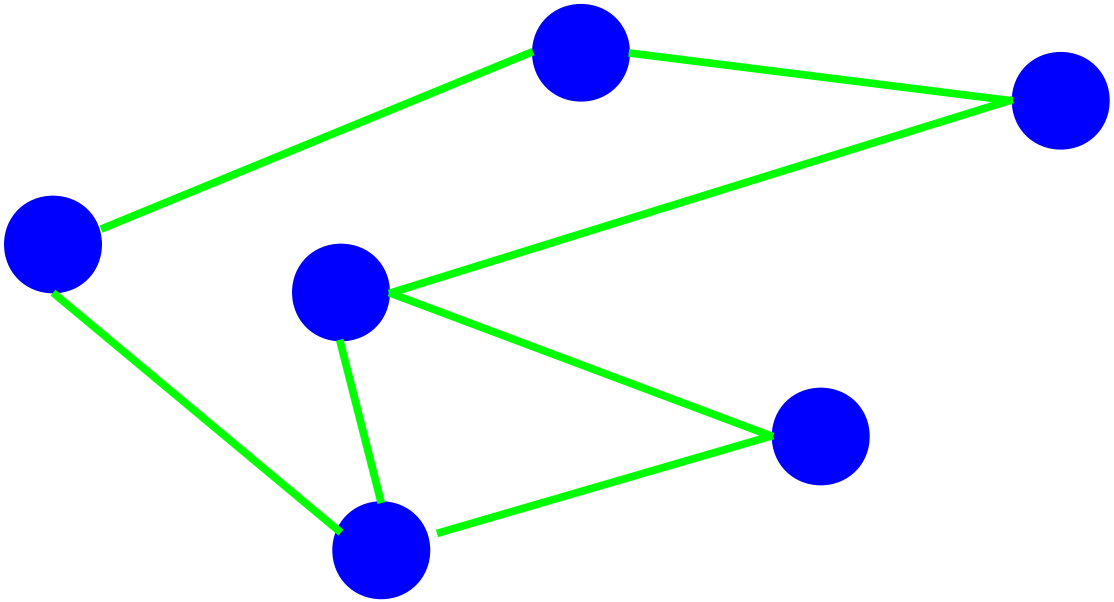
Complexity and Advanced Algorithms
Monsoon 2011

Distributed Algorithms
Lecture 1

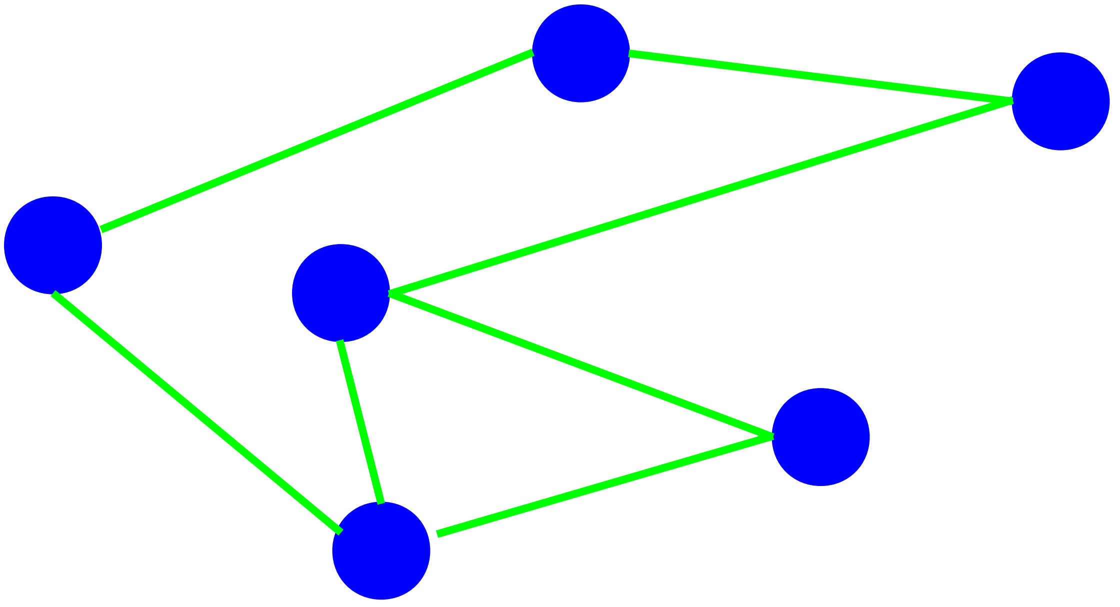
Distributed Computing – Basics



- Distributed computing: Consider a collection of homogeneous processors (computers) linked with an interconnection network.

Distributed Computing – Basics

- Computation is distributed across the processors, possibly exchanging partial results as messages.
- Also called as the message passing model or the network model.



Distributed Computing – Basics

- Several interesting questions arise
- How to indeed distribute the computation?
- How to send/receive messages?
- How to analyze such an algorithm?
  - What are the important parameters?
  - How to quantify each parameter?

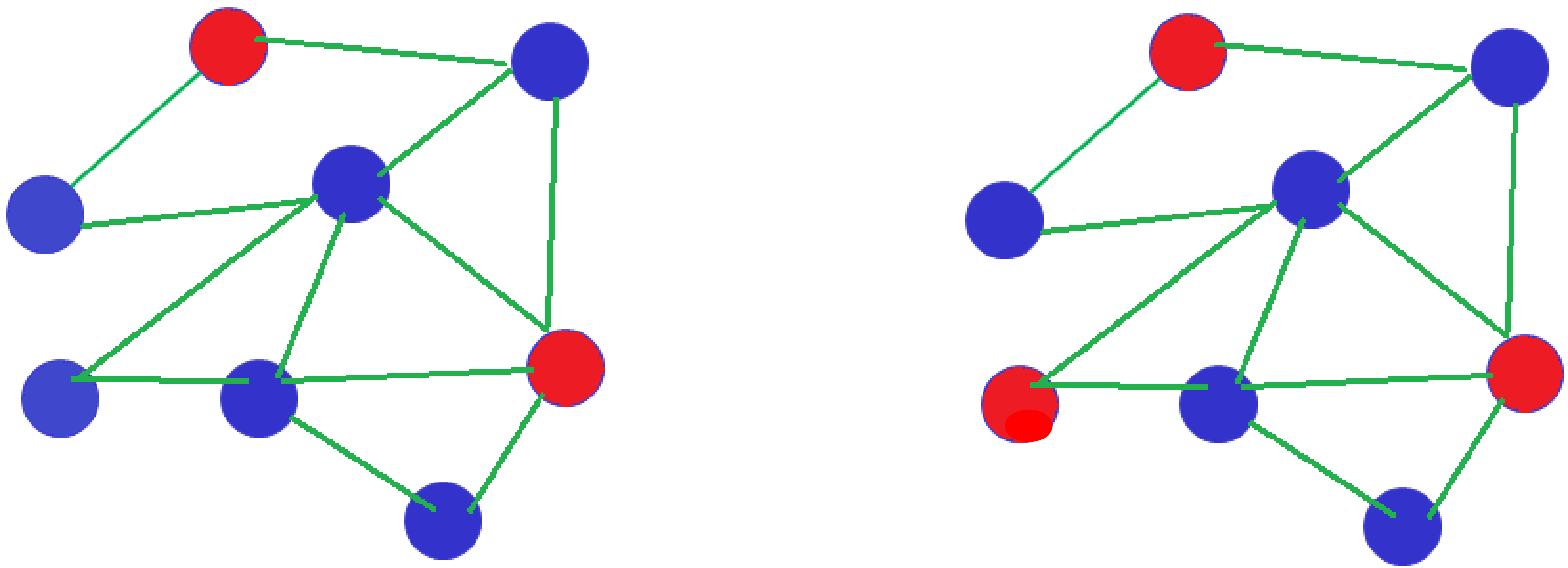
Distributed Computing – Basics

- Typically, computation proceeds in rounds.
- In each round, each node can send/receive messages to all its neighbors.
  - Any limits on the message size?
- In each round, nodes can perform some local computation also.
  - Any limits on the amount of computation?
- Rounds are assumed to be synchronous.

Distributed Computing – Basics

- Parameters to analyze algorithms
  - Number of rounds required – Equivalent to time taken to finish
  - Local computation – Most often ignored, as only simple computations are involved.
  - Message volume – total number of bits/bytes across the overall execution. Also called as the message complexity.
    - Important as communication is a huge consumer of energy.

Basic Problems I – MIS



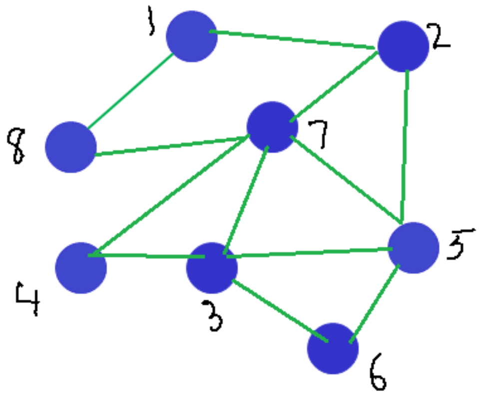
- Given a graph G = (V, E) recall that an independent set of nodes is a subset U $\subseteq$ V s.t. no two elements of U are neighbors in G.
  - U is called a maximal independent set (MIS) if no proper superset of U is also independent.

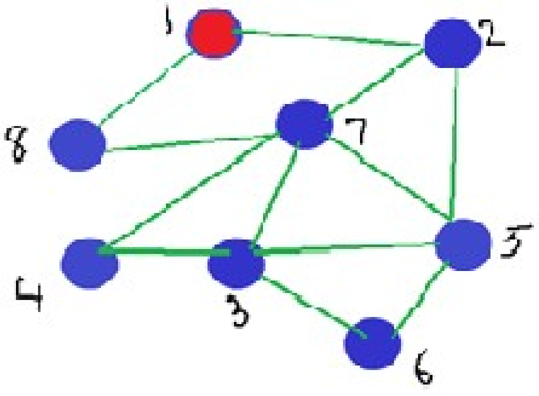
A Sequential Algorithm for MIS

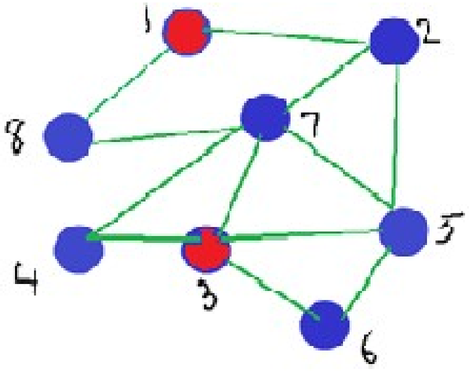
```
Algorithm Greedy-MIS(G)
Begin
   I = {};
   for v = 1 to n do
       if I ∩ N(v) = 
∅ then
          add v to I.
   end-for
End.
```

- Greedy algorithm.
  - Produces lexicographically first MIS.

Example







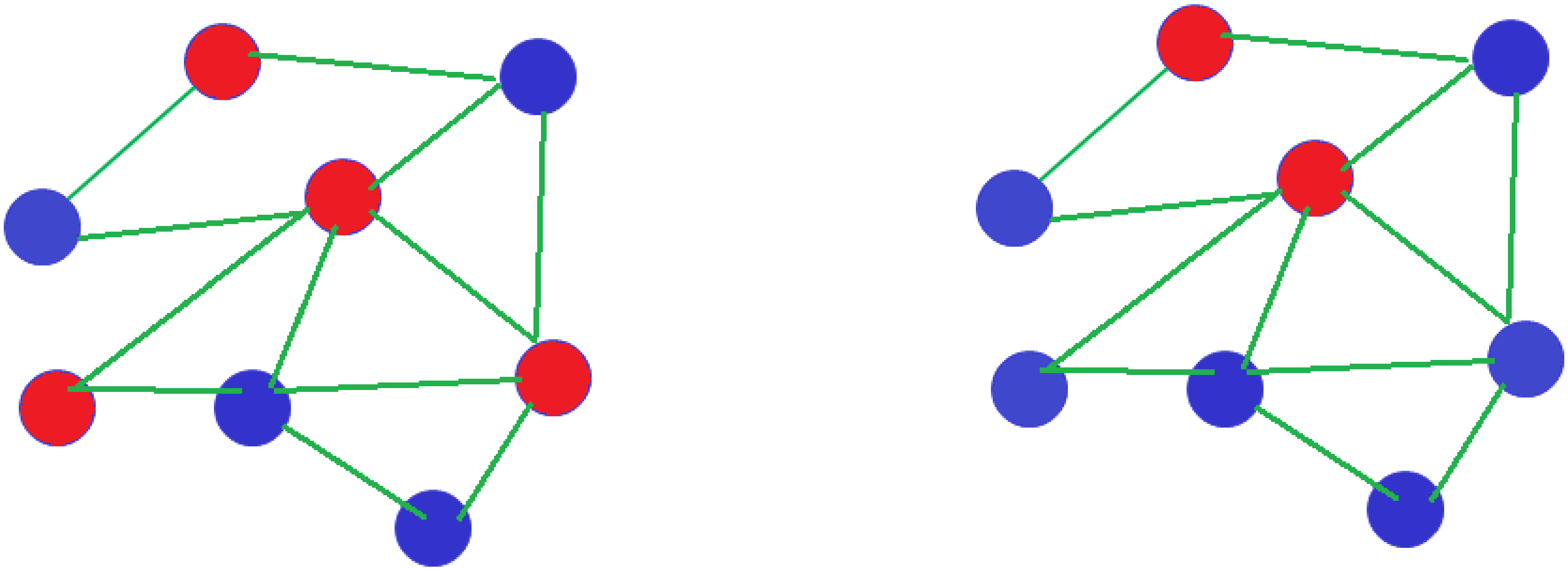
Parallel/Distributed Algorithm for MIS

- The sequential algorithm is not good to parallelize.
- Indeed several complexity theoretic notions surrounding MIS in parallel.
  - Will visit them in due course.

Parallel/Distributed Algorithm for MIS

- Some pointers towards a solution
  - Think of an iterative approach, but
  - Add more vertices in each iteration to I than just a single vertex.
  - Amounts to finding an independent set S in each iteration.
  - Each iteration also should run in parallel efficiently.
- The set S should be large enough in each iteration.
  - So that there are fewer iterations.

A Randomized Algorithm



- To find such a set S, we start with a random set R that is larger than S in size.
- The set R may not be independent, but can trim R suitably.
  - Favor vertices of higher degree!

The Algorithm

```
Algorithm Parallel-MIS(G)
Begin
   I =∅;
   repeat
       1. for all v ∈ V do in parallel
              if d(v) = 0 then add v to I and delete v from V;
              else mark v with probability 1/2d(v);
       2. for all (u,v) ∈ E do in parallel
              if both u and v are marked then
                 unmark the vertex of lower degree
       3. for all v ∈V do in parallel
               If v is marked then add v to S.
       4. I = I ∪ S;
       5. Delete S ∪ N(S) from V, and all incident edges from E
   until V =∅;
End.
```

Analysis

- Wish to show that the algorithm terminates in O(log n) iterations, on expectation.
- For this, we show that in every iteration, at least a constant fraction of the edges are deleted on expectation.

Analysis

- To carry out the analysis, define:
  - A vertex v is good if at least 1/3rd of its neighbors have degree less than d(v).
  - A vertex v is bad if at least 2/3rd of its neighbors have degree at least d(v).
  - An edge e is good if at least one endpoint of e is good.
  - An edge e is bad if both its endpoints are bad.

Analysis – Sketch

- Good vertices have enough low degree neighbors.
- So at least one such low degree neighbors is in S with good probability.
  - Helps delete good vertices.
  - This in turn helps delete good edges.
- If we can show that there are enough good edges, then suffices if a fraction of them are deleted.

Analysis – Claim 1

- For every good vertex v with $d(v) > 0$, the probability that some neighbor w of v gets marked is at least $1 - e^{-1/6}$
- Proof: Pr(w is marked) = $\frac{1}{2d(w)}$.
- Since v is good, at least $d(v)/3$ neighbors have degree at most $d(v)$. Let w be such a neighbor.
- Pr(w is marked) = $1/2d(w) \geq \frac{1}{2d(v)}$.
- Pr(no neighbor of v is marked) $\leq$

$$\left(1 - \frac{1}{2d(v)}\right)^{d(v)/3} = e^{-1/6}.$$

Analysis – Claim 2

- If a vertex w is marked, then $\Pr(w \text{ in } S) \geq \frac{1}{2}$.
- Proof: If w is marked, then w is not in S only if some high degree neighbor of w is also marked.
- Each such high degree neighbors of w is marked with probability at most $\frac{1}{2d(w)}$.
- Number of such high-degree neighbors $\leq d(w)$.

$$\Pr(w \text{ in } S) = 1 - \Pr(w \text{ not in } S)$$
$$= 1 - \Pr(\exists u \in N(w),\ d(u) \geq d(w),\ u \text{ marked})$$
$$= 1 - |u \in N(w),\ d(u) \geq d(w)|\ \frac{1}{2d(w)}$$
$$\geq 1 - |u \in N(w)|\ \frac{1}{2d(w)} = 1 - d(w).\ \frac{1}{2d(w)}$$
$$= \frac{1}{2}.$$

Analysis – Claim 3

- Let v be a good vertex. Then,

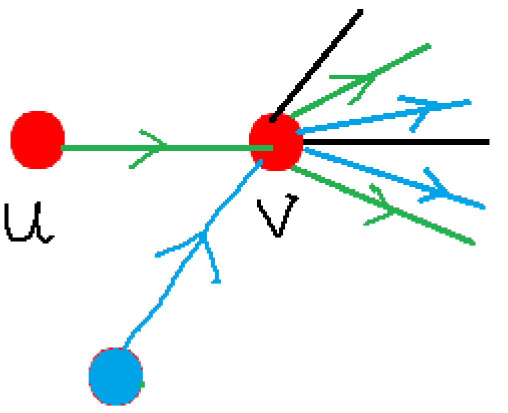
  $$\Pr(v \text{ is deleted}) \geq (1 - e^{-1/6})/2.$$

- Proof: Combine Claims 1 and 2.

Analysis – Claim 4

- At least half the edges are good.
- Proof: For every bad edge e, associate a pair of edges via a function $f{:}E_B \to \binom{E}{2}$ such that for any two distinct bad edges e_1 and e_2, $f(e_1) \cap f(e_2) = \emptyset$.
- Completes the proof since only |E|/2 such pairs exist.
- The function f defined as follows.

Analysis – Claim 4



- For each edge (u, v) ∈ E, orient it towards the vertex of higher degree.
- Consider a bad edge e = (u,v) oriented towards v.
- Since v is bad, the out-degree of v is at least twice its in-degree.
- So, there exists a way to pick a pair of edges for every bad edge.

Putting Everything Together

- In each iteration, it is expected that a constant fraction of edges are deleted.
  - Half the edges are good, and a good edge is deleted with probability at least $(1 - e^{-1/6})/2$.
- So, on expectation, only O(log m) = O(log n) iterations suffice.
- Can also show that with high probability O(log n) iterations suffice.