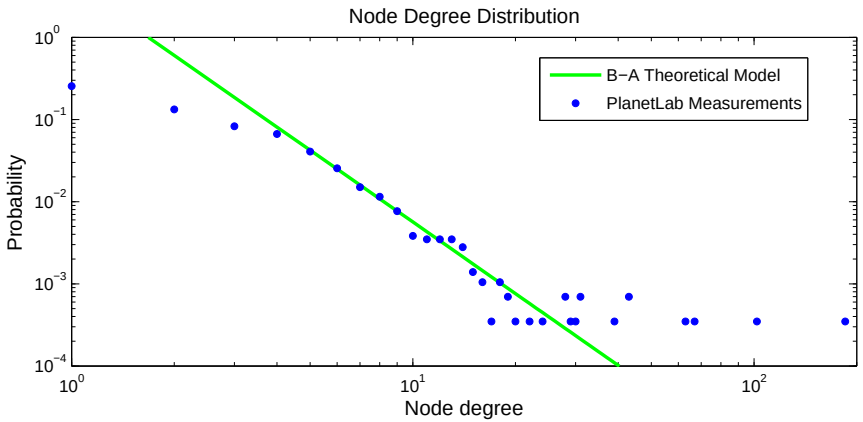
Network Modeling and Barabasi-Albert (BA) Model

- *Scale free* model for simulation of node degree distribution
  - growth
  - preferential attachment

$$P(k) = 2m^{1/\beta} k^{-\gamma} \text{ for } \gamma{=}3,\ \beta{=}0.5$$

- *Clustering* coefficient

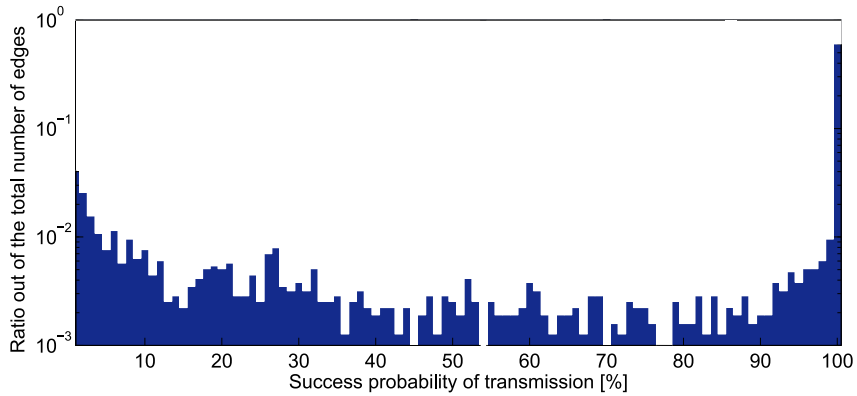
$$C_i = \frac{2E_i}{k_i(k_i - 1)}$$



Node degree distribution - B-A Model vs PlanetLab

Real Scale Free Networks - PlanetLab Measurements

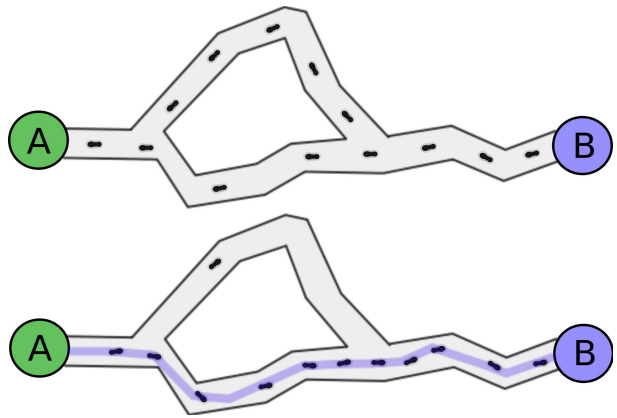
- Internet testbed – 226 end hosts
- Measured topology and packet loss information (*ping*, *traceroute*).
- IP aliasing resolution



Derived link loss *(erasure)* distribution based on path loss measurements, employing network tomography techniques (Netscope).

Ant Colony Optimization for Finding Shortest Path

- Ants walk through the network, looking for food sources. Once a food source is found, ants return to the colony, on the same path, laying down a *pheromone* trail.
- Each ant has a memory of the current path.
- Once food is found, ants return to colony while laying down a pheromone trail.
- At every node, an ant choses one among the adjacent links based on previous pheromone levels of the links.



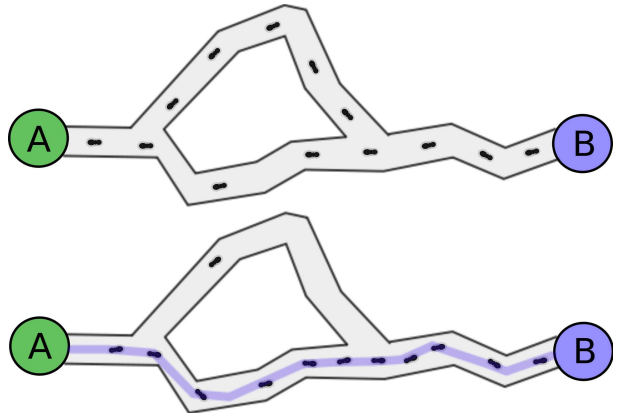
Ant Colony Optimization for Finding Shortest Path

- Transition probability

$$p_{ij}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}}{\sum_{\text{sallowed}} \tau_{is}^{\alpha}(t)\eta_{is}^{\beta}}$$

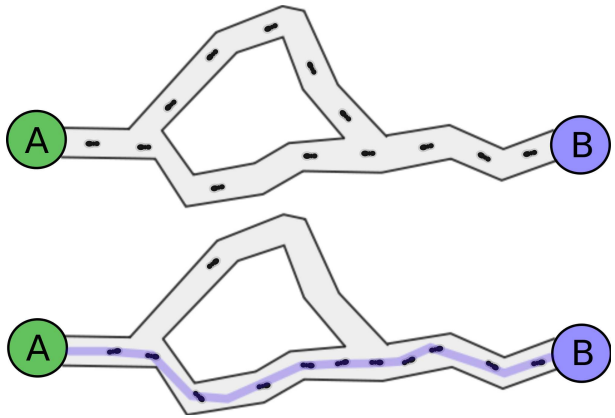
- Pheromone update

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij},$$



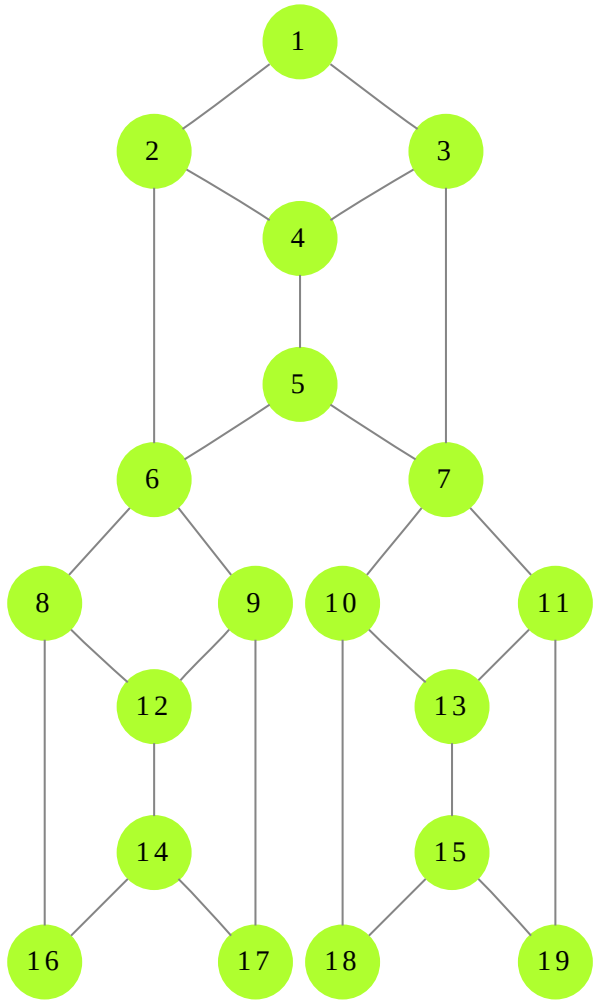
Ant Colony Optimization for Finding Shortest Path

- Since shorter paths will be walked more times in the same interval, a higher pheromone concentration is noticed.
- Ant Colony Optimization (ACO) is preferable among other shortest path algorithms since nodes do not need to be aware of the network topology.



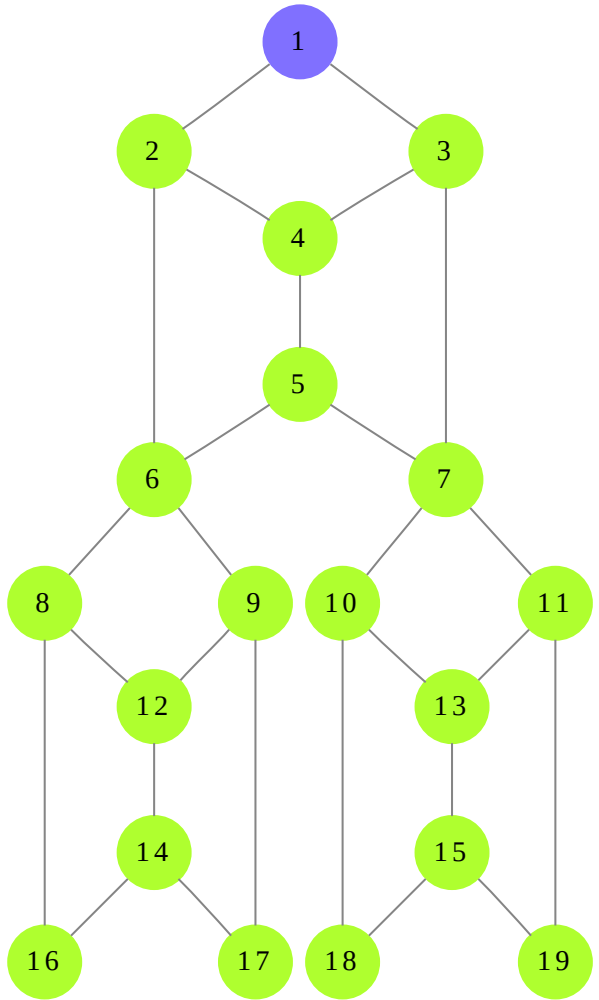
Finding Disjoint Paths

- Select a source node and sink nodes.



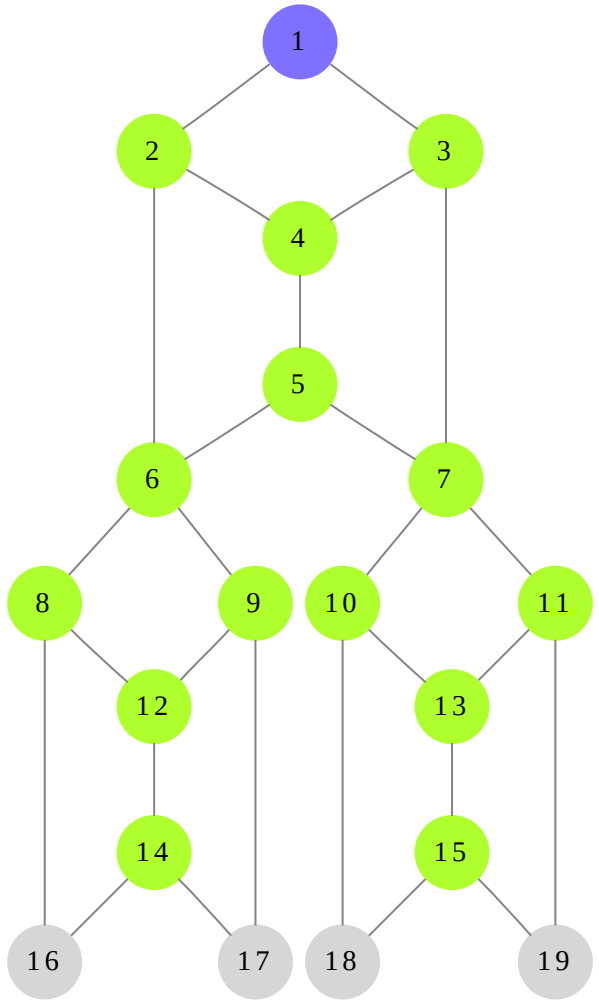
Finding Disjoint Paths

- Select a source node and sink nodes.



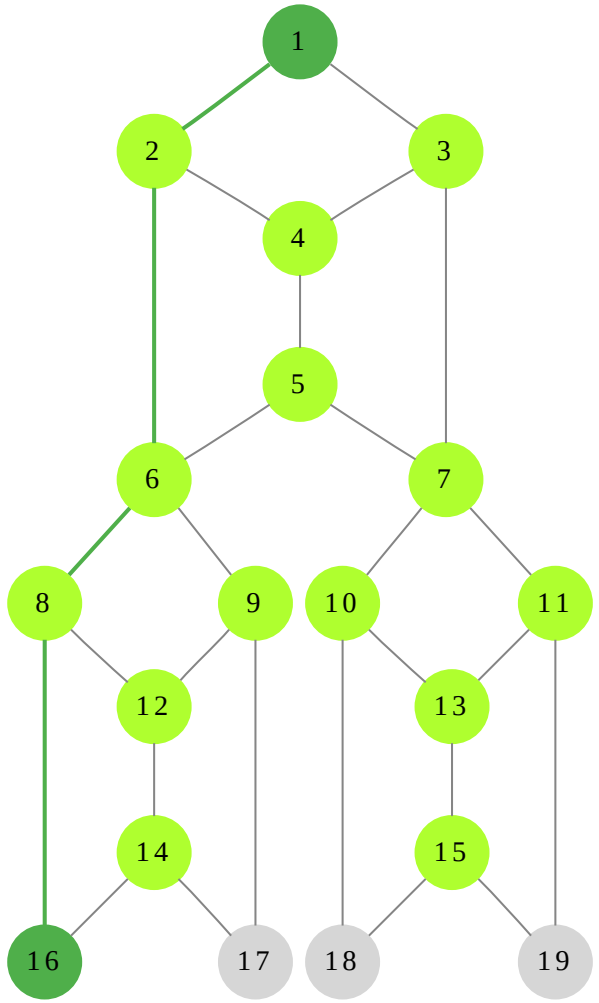
Finding Disjoint Paths

- Select a source node and sink nodes.



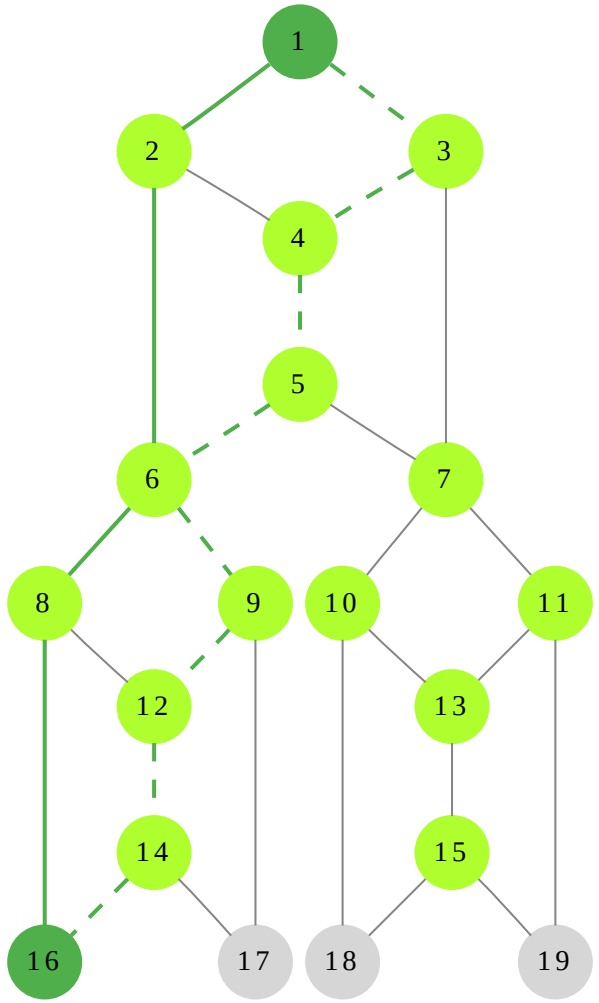
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



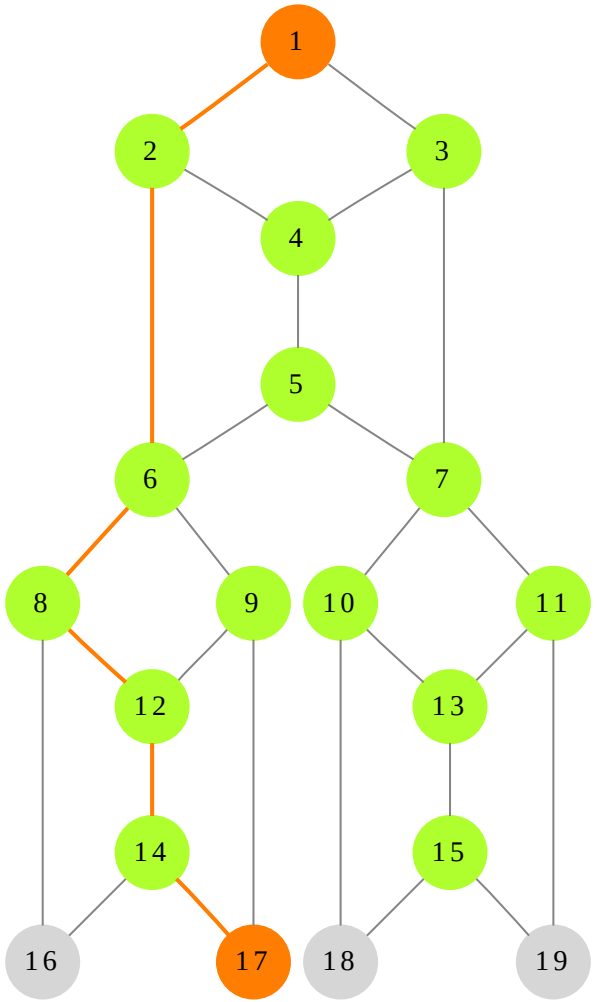
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



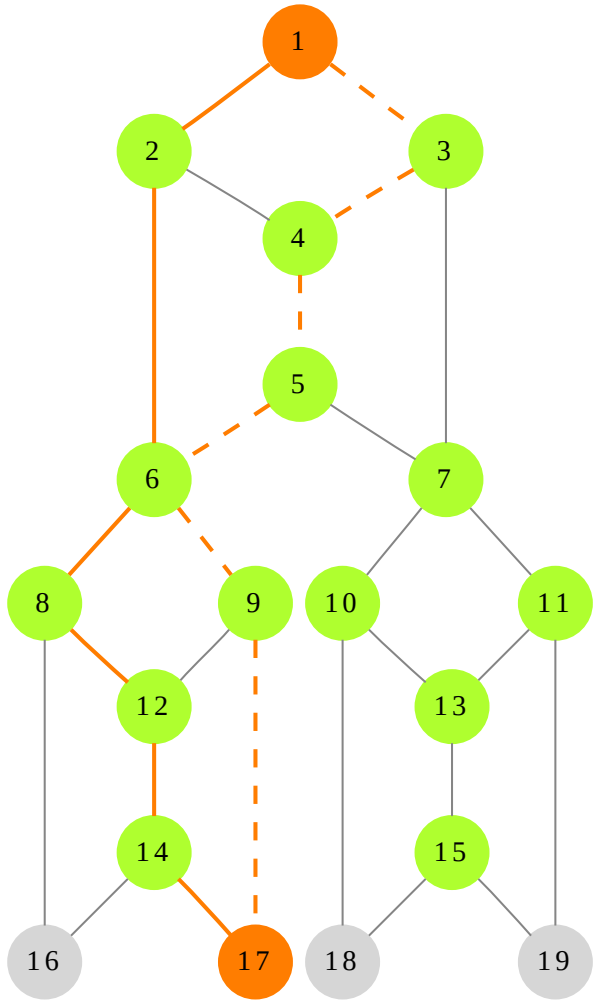
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



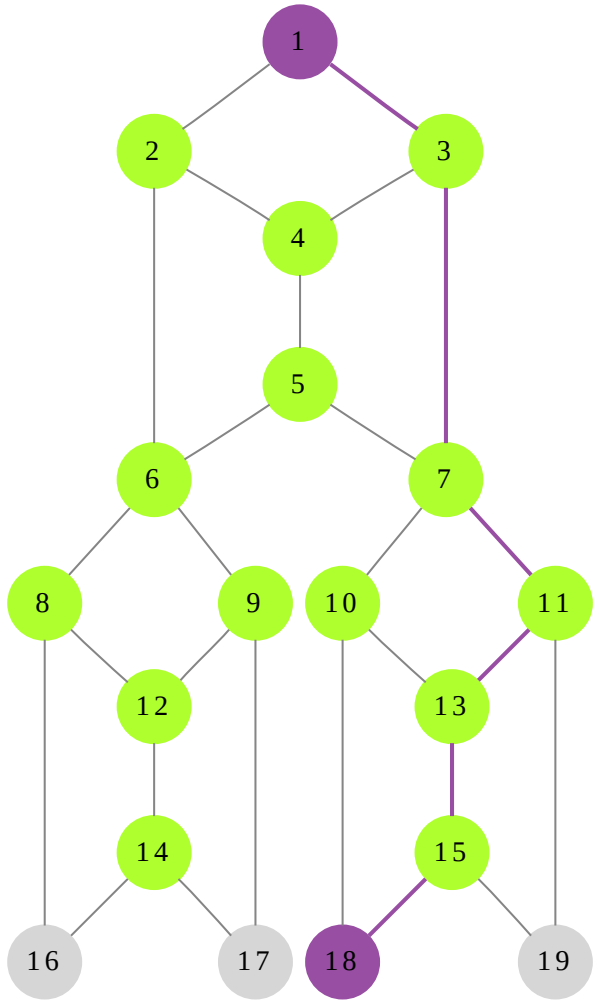
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



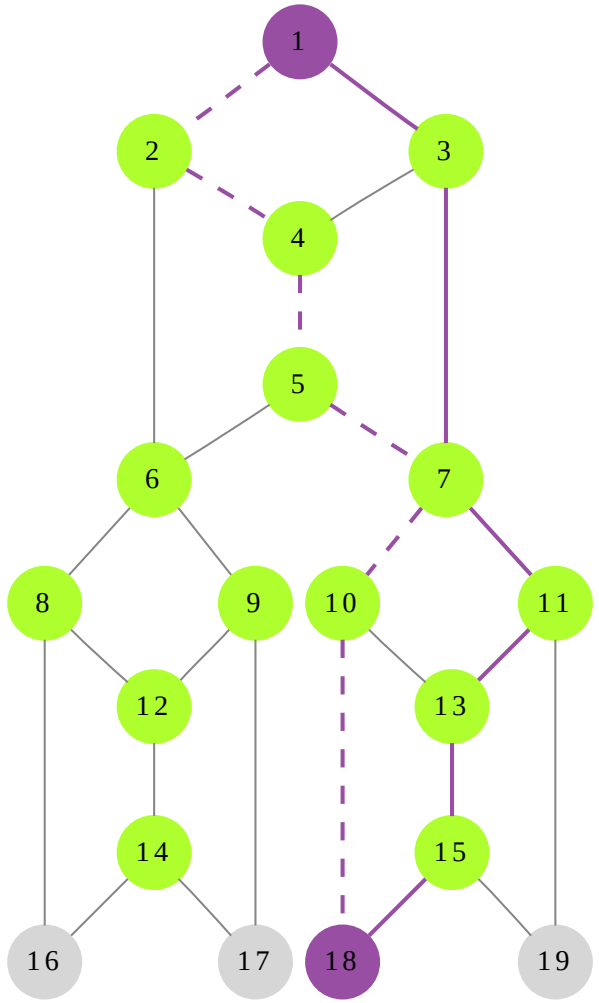
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



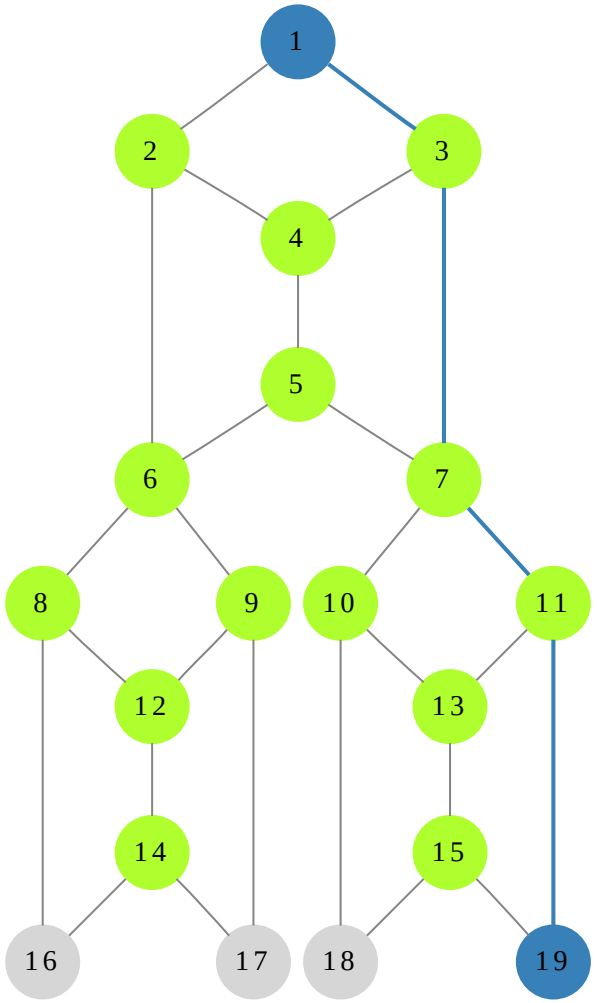
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



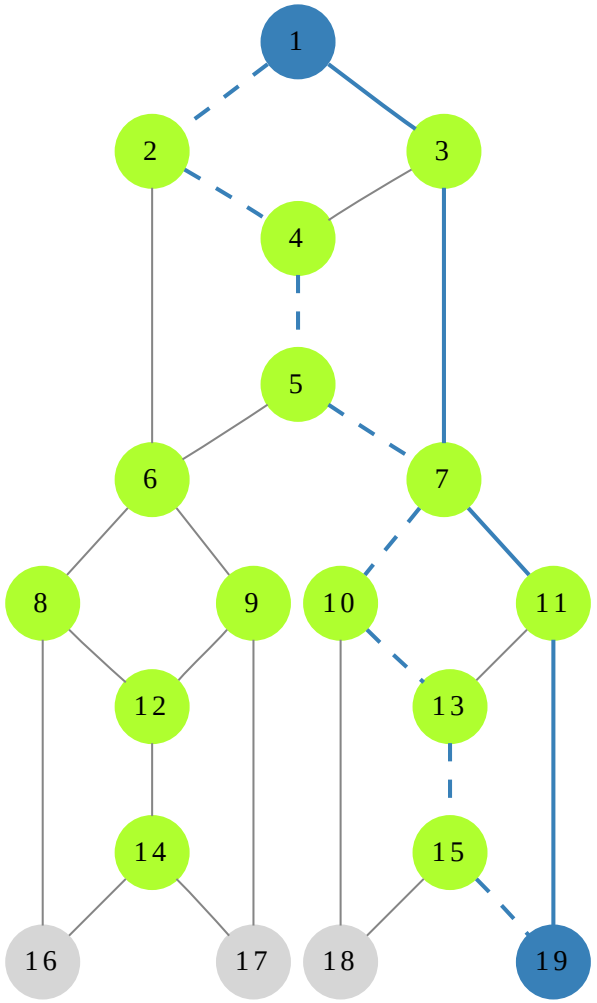
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



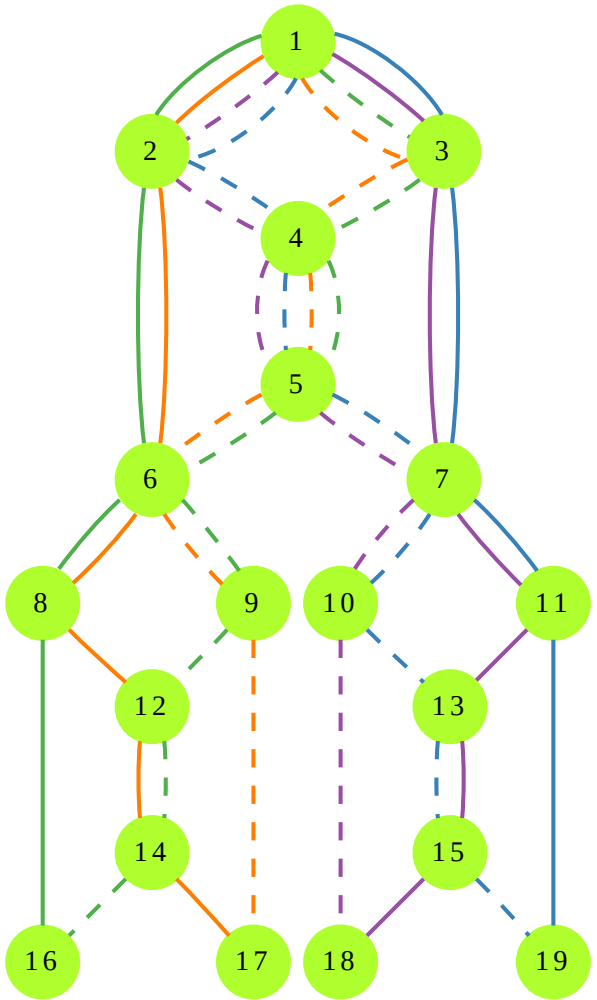
Finding Disjoint Paths

- Get multicast rate m_i for every pair $(source, sink_i)$. Overall multicast rate is $min(m_i)$.
- For all sinks, find $min(m_i)$ disjoint paths.



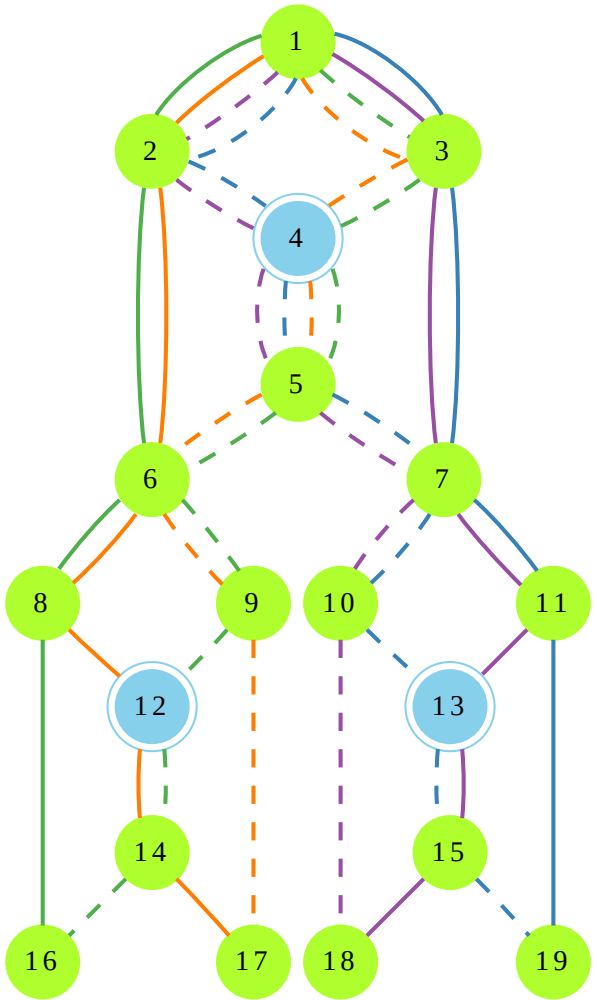
Finding Coding Nodes

- Obtain coding nodes from the intersection of paths corresponding to different sinks.
- Ensure that at all times each edge passes either coded or not coded messages, not both in the same time, otherwise flip disjoint paths.



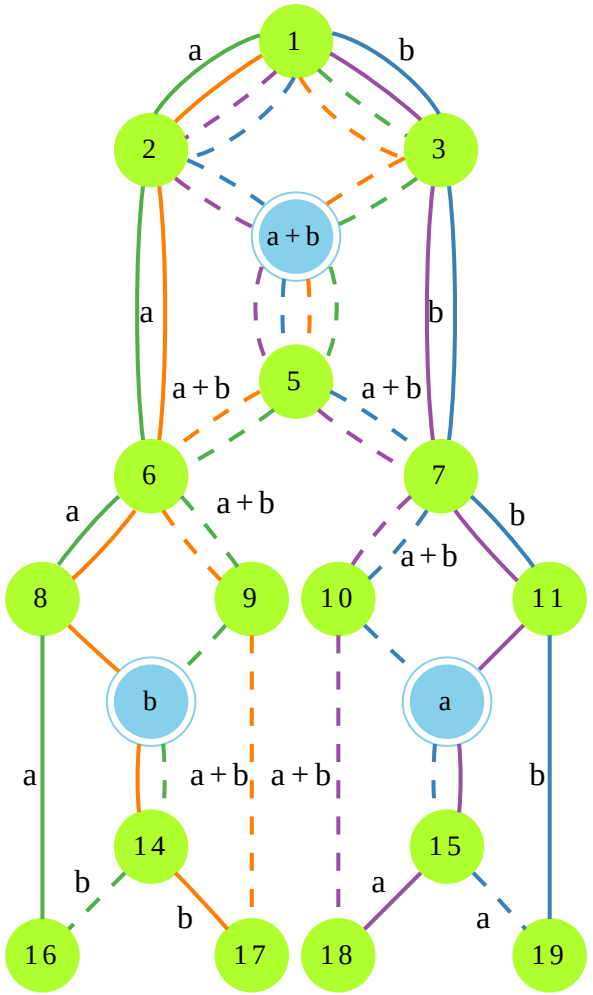
Finding Coding Nodes

- Obtain coding nodes from the intersection of paths corresponding to different sinks.
- Ensure that at all times each edge passes either coded or not coded messages, not both in the same time, otherwise flip disjoint paths.

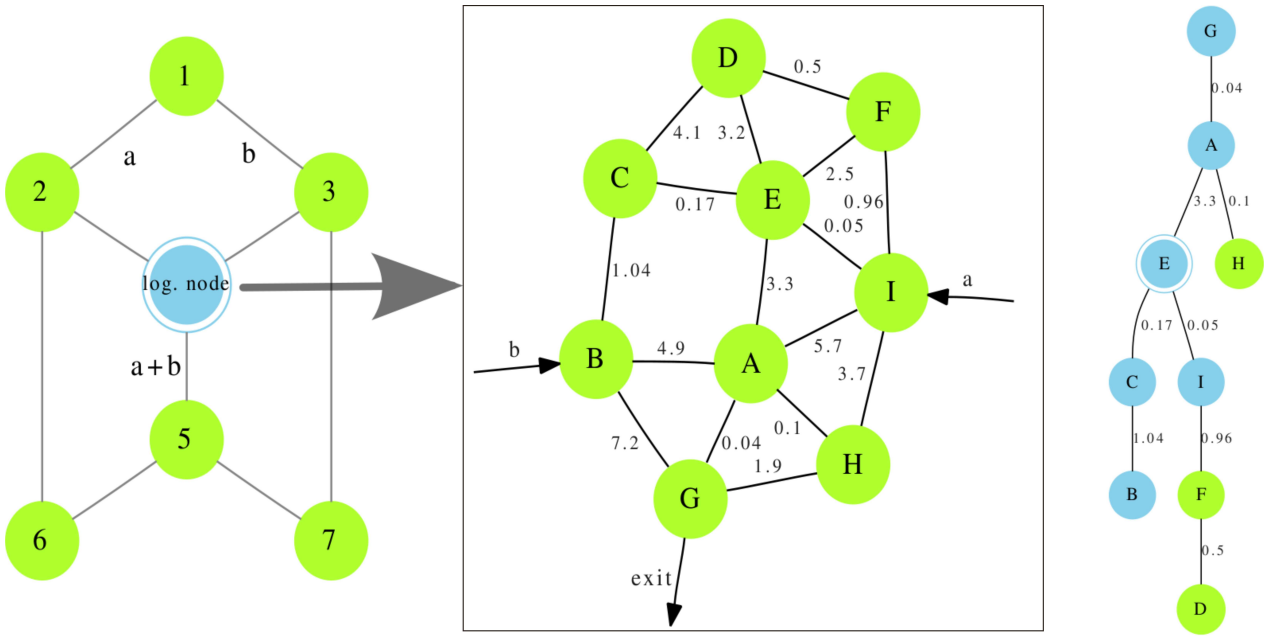


Finding Coding Nodes

- Obtain coding nodes from the intersection of paths corresponding to different sinks.
- Ensure that at all times each edge passes either coded or not coded messages, not both at the same time, otherwise flip disjoint paths.



Hierarchical Network Structure



- *Logical node*
- Minimum Spanning Tree