Centrality & Clustering

Advanced Social Computing

Department of Computer Science
University of Massachusetts, Lowell
Fall 2020

Hadi Amiri
hadi@cs.uml.edu

UMASS LOWELL

Lecture Topics

- Centrality
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality
- Clustering
  - Edge Betweenness
  - Computing Edge Betweenness

Centrality

- What characterizes an important node in a network?
  - Most influential people in social nets,
  - Key infrastructure nodes in the Internet
  - Main spreaders of disease
  - Etc.

- Structural view:
  - Importance of a node is related to its **position in the network**.

Centrality Measures

- Different **centrality measures** capture different **structural characteristics of nodes**!
- There is often a high correlation between these measures!
- Sometimes the most important node might depend on which measure is used!

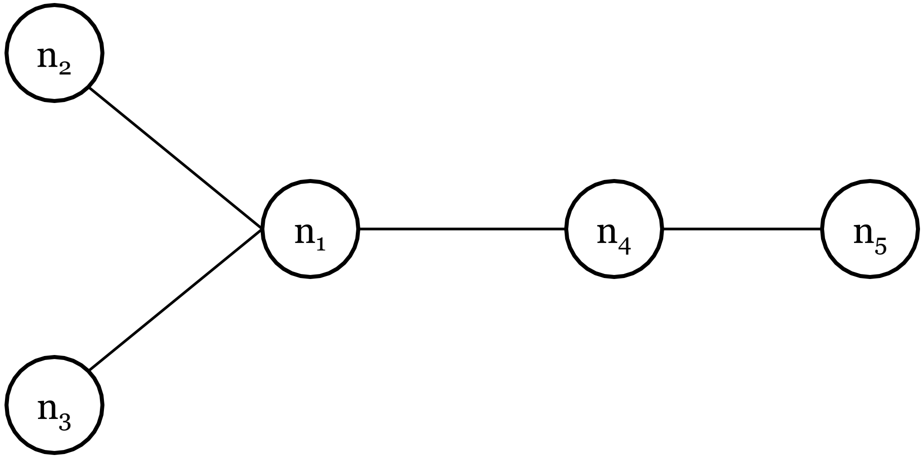
- C : Centrality
  - C (i) : Centrality for node i
  - C(**A**) : Centrality for a group of nodes **A** ϵ **N**

Centrality Measures- Cnt.

- Centrality
  - **Degree Centrality**
  - Closeness Centrality
  - Betweenness Centrality

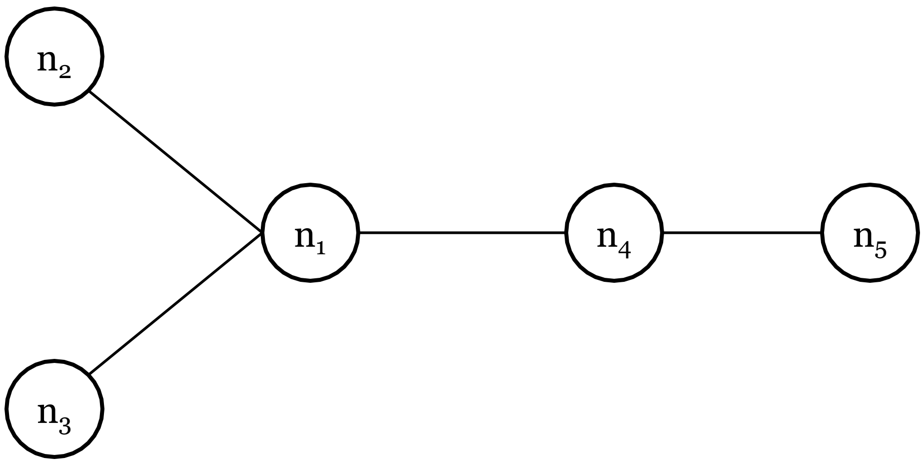
Degree Centrality

- A node is central if it has **links to many nodes**.
  - Look at the node degree



Degree Centrality- Cnt.

- A node is central if it has links to many nodes.
  - Look at the node degree

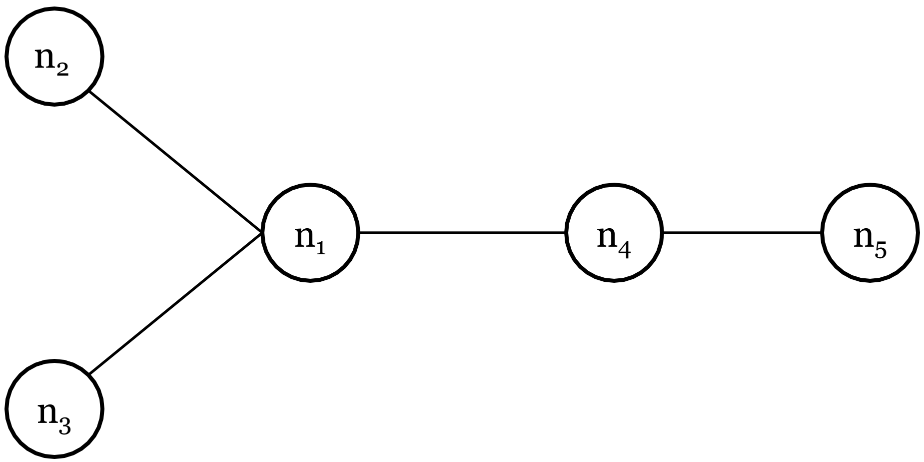


	n1	n_2	n_3	n_4	n_5	
n_1	0	1	1	1	0	**3**
n_2	1	0	0	0	0	**1**
n_3	1	0	0	0	0	**1**
n_4	1	0	0	0	1	**2**
n_5	0	0	0	1	0	**1**
	3	1	1	2	1	

Adjacency Matrix (A)

Degree Centrality- Cnt.

- Standardized Degree Centrality
  - Divide by the maximum possible degree centrality value!
    - N-1



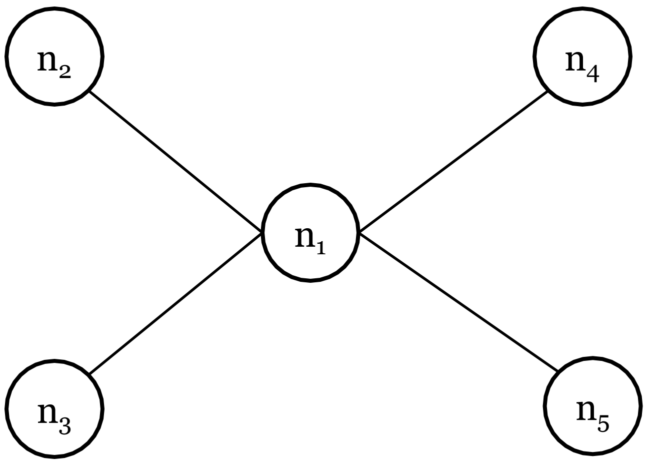
	n1	n_2	n_3	n_4	n_5	
n_1	0	1	1	1	0	¾
n_2	1	0	0	0	0	¼
n_3	1	0	0	0	0	¼
n_4	1	0	0	0	1	½
n_5	0	0	0	1	0	¼

Centrality Measures- Cnt.

- Centrality
  - Degree Centrality
  - **Closeness Centrality**
  - Betweenness Centrality

Closeness Centrality

- A node is central if it is **close to other nodes**.
  - Look at distance btw nodes
  - Closeness: 1 / Sum of distance to other nodes



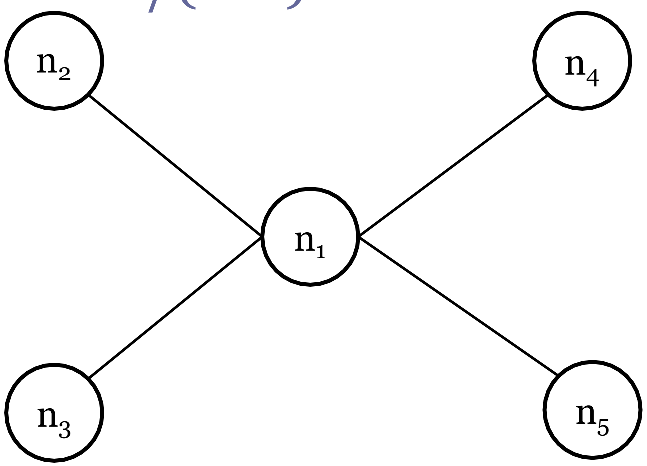
$$C(n_1) = 1/(\sum_{j=1}^{n} D_{1j}) = 1/(\sum_{i=1}^{n} D_{i1}) = 1/4$$

	n1	n_2	n_3	n_4	n_5	
n_1	0	1	1	1	1	1/4
n_2	1	0	2	2	2	1/7
n_3	1	2	0	2	2	1/7
n_4	1	2	2	0	2	1/7
n_5	1	2	2	2	0	1/7

Distance Matrix (D)

Closeness Centrality- Cnt.

- Standardized Closeness Centrality
  - Divide by the maximum possible closeness centrality value!
    - 1/(N-1)



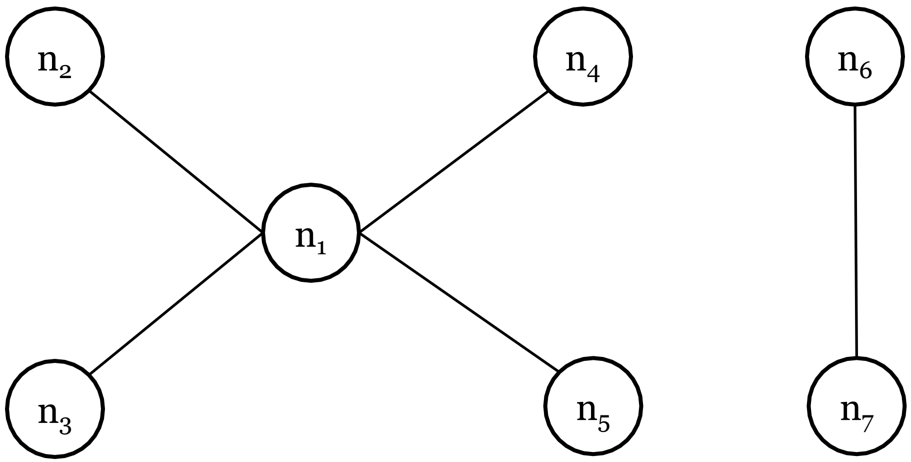
$$C(n_1) = (N-1)/(\sum_{j=1}^{n} D_{1j}) = (N-1)/(\sum_{i=1}^{n} D_{i1}) = 4/4$$

	n1	n_2	n_3	n_4	n_5	
n_1	0	1	1	1	1	4/4
n_2	1	0	2	2	2	4/7
n_3	1	2	0	2	2	4/7
n_4	1	2	2	0	2	4/7
n_5	1	2	2	2	0	4/7

Distance Matrix (D)

Closeness Centrality- Cnt.

- How to compute Closeness Centrality in networks with disconnected components?



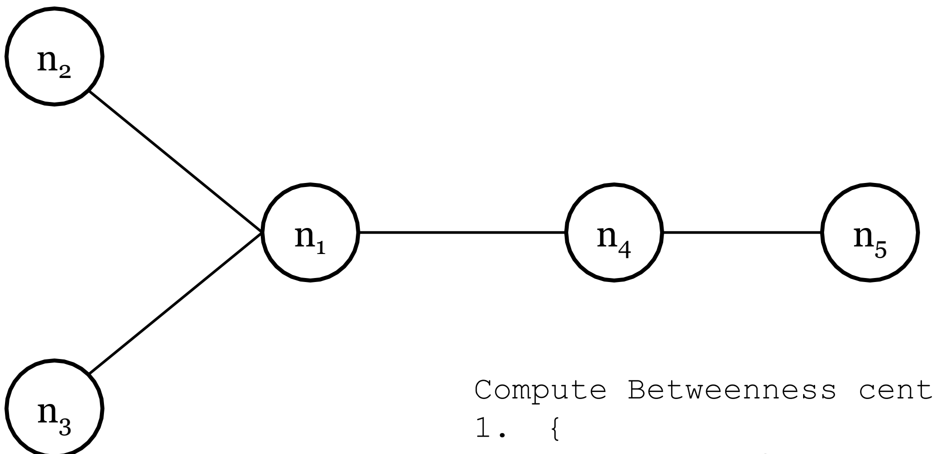
- Only consider the giant component or do graph sampling?
- Only consider nodes that are reachable in paths of length 1, 2, …
  This is called k-Step Reach!

Centrality Measures- Cnt.

- Centrality
  - Degree Centrality
  - Closeness Centrality
  - **Betweenness Centrality**

Betweenness Centrality

- A node is central if **other nodes have to go through it** to reach each other.
  - Look at shortest paths between nodes




```
Compute Betweenness centrality for a target node:
1.  {
2.  For each pair of nodes, compute the shortest paths
    between them.
3.  For each pair of nodes, determine the fraction of
    shortest paths that pass through the target node.
4.Sum the fractions over all pairs of nodes
5.  }
```

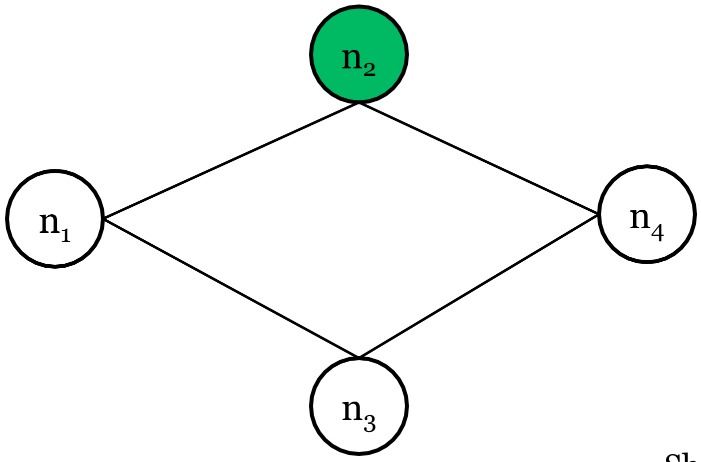
Betweenness Centrality- Cnt.

s_{jk} Number of shortest paths btw nodes n_j and n_k

$s_{jk}(n_i)$ Number of shortest paths btw nodes n_j and n_k that include node n_i

$\frac{s_{jk}(n_i)}{s_{jk}}$ Proportion of shortest paths btw nodes n_j and n_k that include node n_i

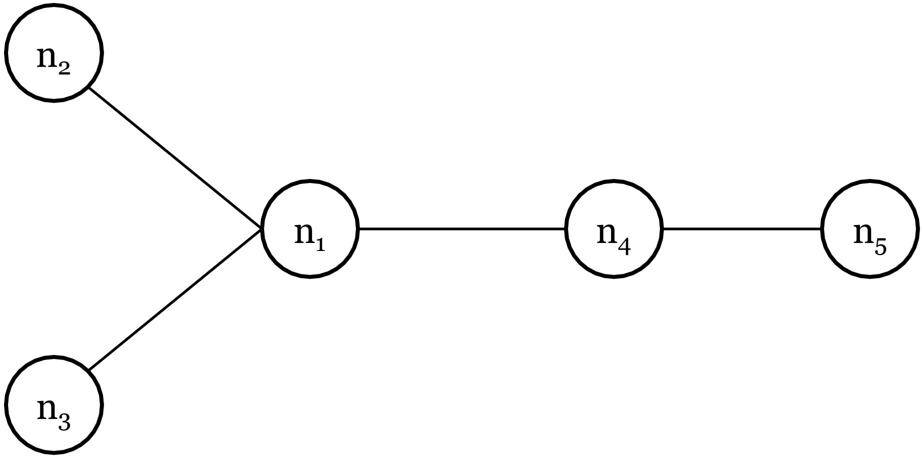
$\mathrm{Sum}(_{j,k!=i} \frac{s_{jk}(n_i)}{s_{jk}})$ Proportion of shortest paths btw all nodes that include node n_i



Shortest paths n_1-n_4	n_1-n_2-n_4, n_1-n_3-n_4
s_{14}	2
$s_{14}(n_2)$	1
$s_{14}(n_2) / s_{14}$	½
$C(n_2)$	½

Shortest paths btw n_1-n_3 and n_3-n_4 don't include n_2! Their corresponding proportions are 0.

Betweenness Centrality- Cnt.

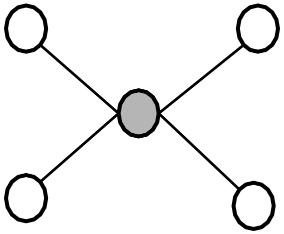


Pair	Shortest path
n1 n2	n1-n2
n1 n3	n1-n3
n1 n4	n1-n4
n1 n5	n1-n4-n5
n2 n3	n2-n1-n3
n2 n4	n2-n1-n4
n2 n5	n2-n1-n4-n5
n3 n4	n3-n1-n4
n3 n5	n3-n1-n4-n5

Betweenness	
n1	5
n2	0
n3	0
n4	3
n5	0

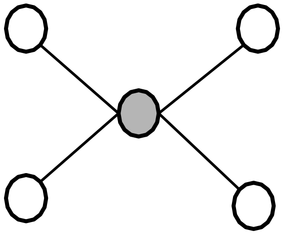
Betweenness Centrality- Cnt.

- Standardized Betweenness Centrality
  - Divide by the maximum possible betweenness centrality value!
    - ?



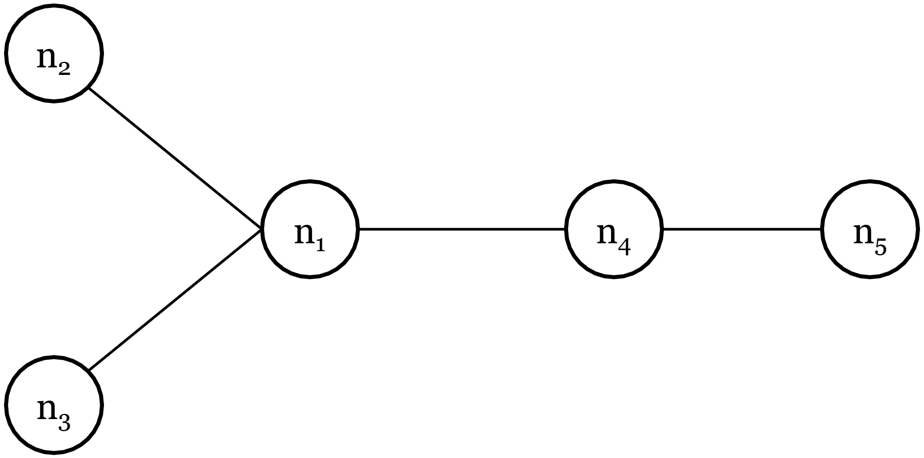
Betweenness Centrality- Cnt.

- Standardized Betweenness Centrality
  - Divide by the maximum possible betweenness centrality value!
    - (N-1)(N-2)/2 : the number of other pairs of nodes (exclude the node itself)



Betweenness Centrality- Cnt.

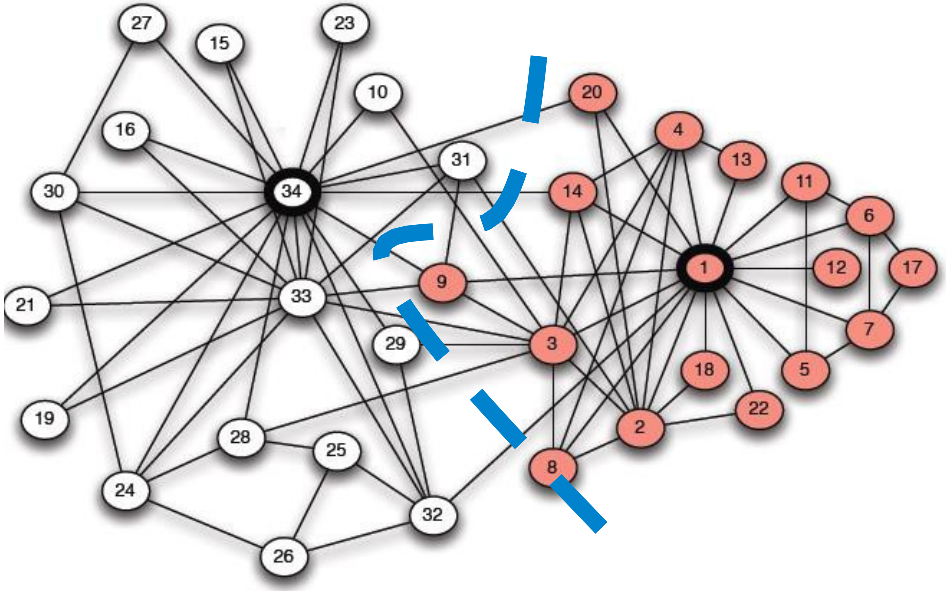
- Standardized Betweenness Centrality
  - Divide by the maximum possible betweenness centrality value!
    - (N-1)(N-2)/2 : the number of other pairs of nodes (exclude the node itself)



	Betweenness	Stnd. Betweenness
n1	5	5/6 = 0.83
n2	0	0/6 = 0.00
n3	0	0/6 = 0.00
n4	3	3/6 = 0.50
n5	0	0/6 = 0.00

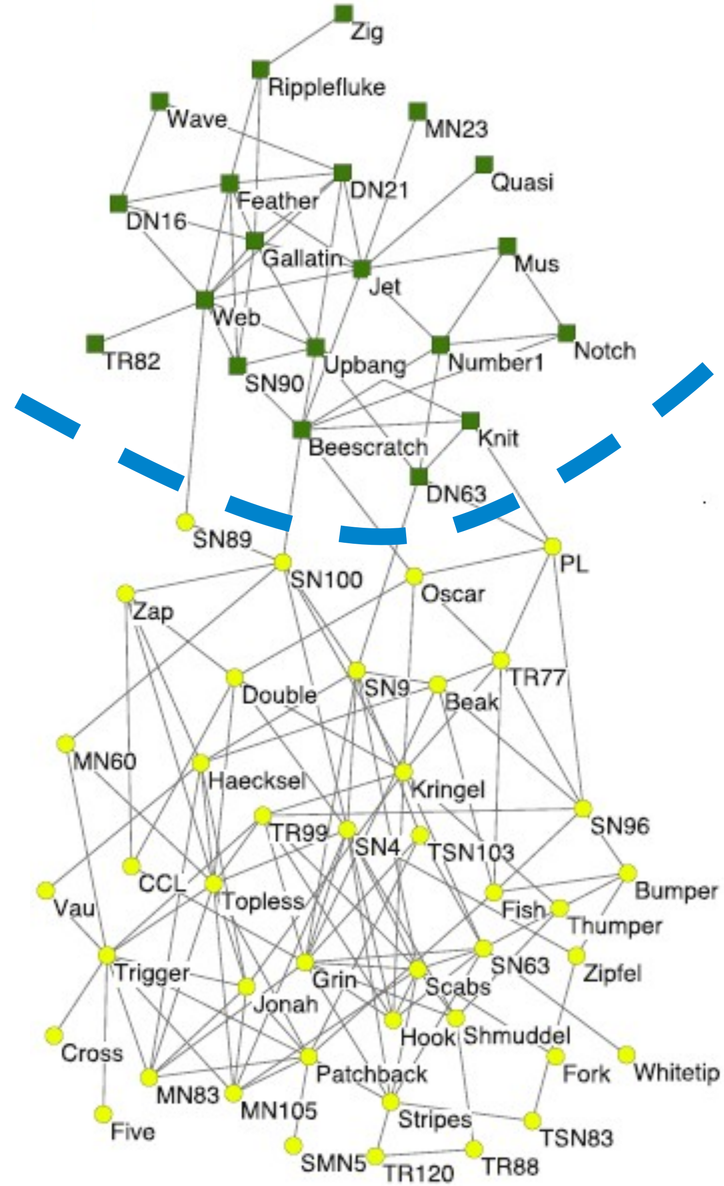
Clustering

- We aim to develop techniques to **identify densely connected regions**
  - breaking a network into a set of densely connected nodes
  - with sparse connections between groups
- Graph Partitioning



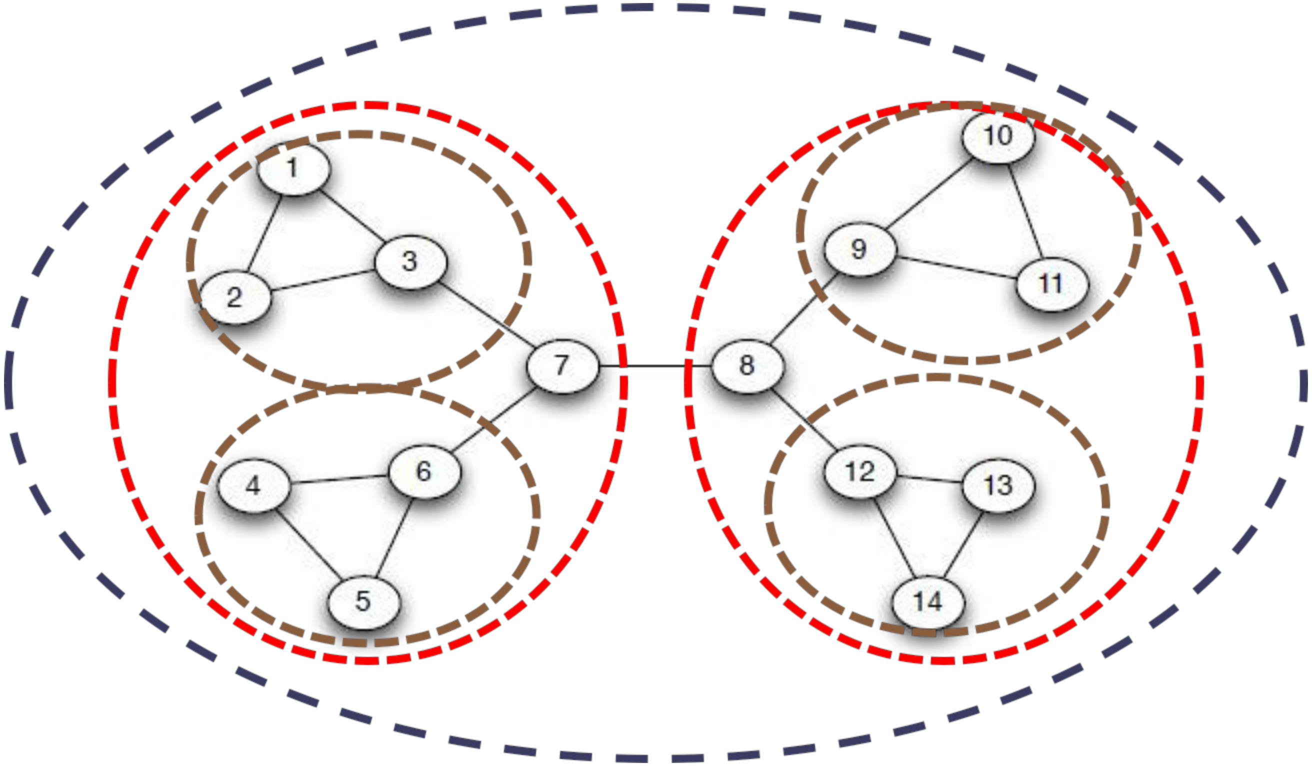
Clustering- Cnt.

- Members of the same group are heavily connected, while
- Members of different groups are less connected!



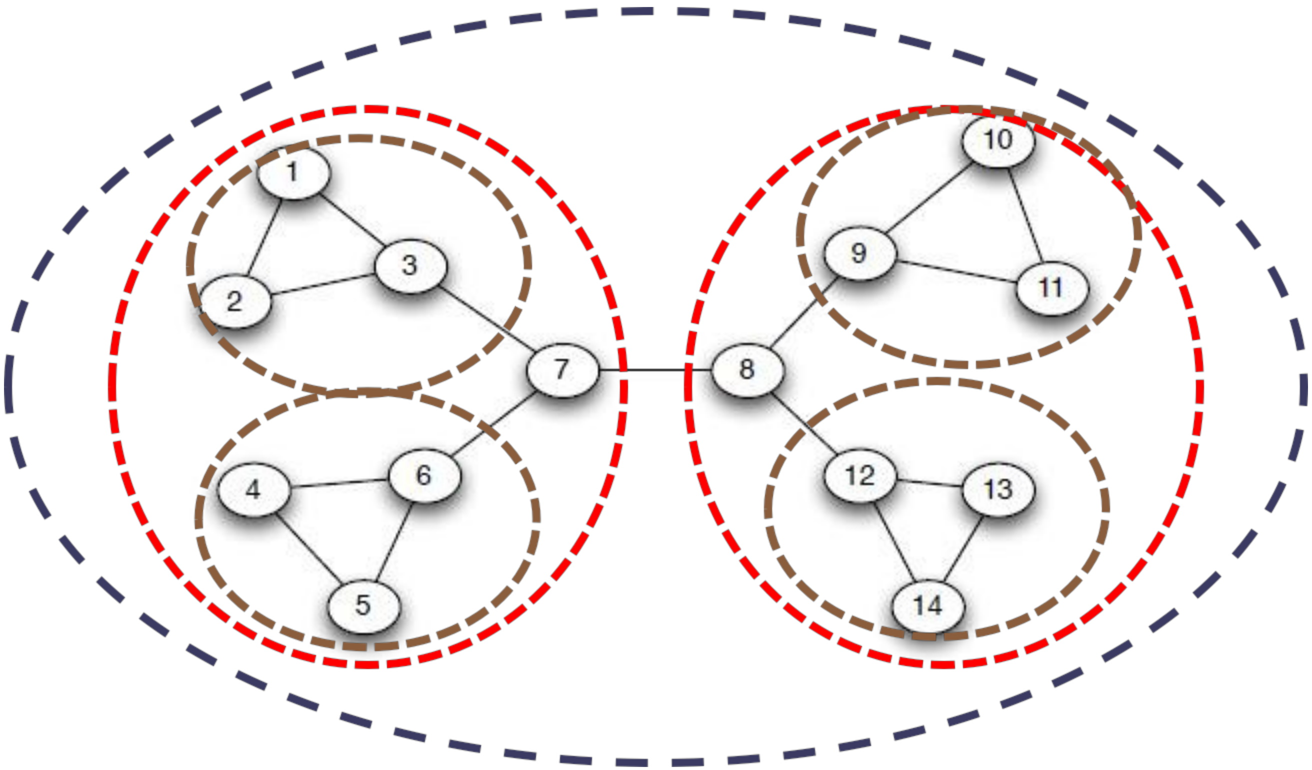
Clustering- Approaches

- Divisive methods
  - breaking first at the 7-8 edge, and then the nodes into nodes 7 and 8



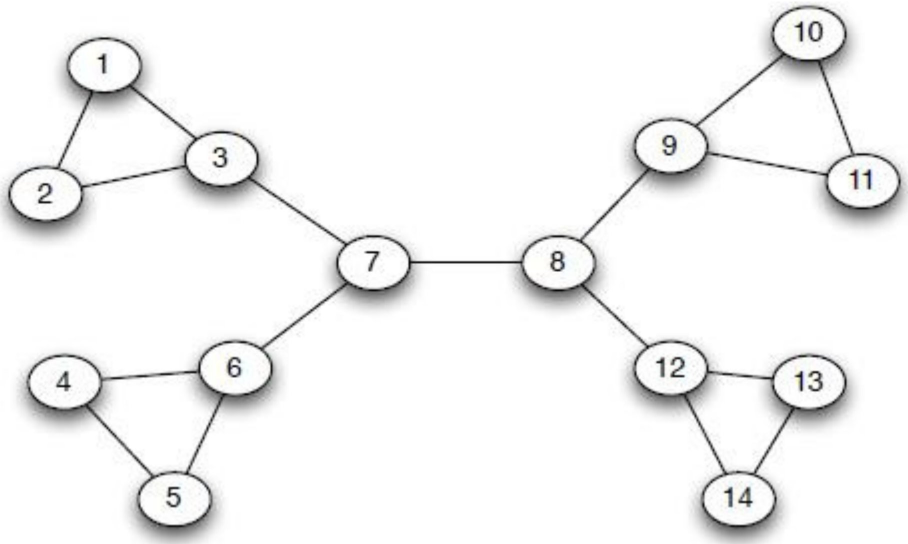
Clustering- Approaches- Cnt.

- Agglomerative methods
  - merge the 4 triangles and then pairs of triangles (via nodes 7 and 8)



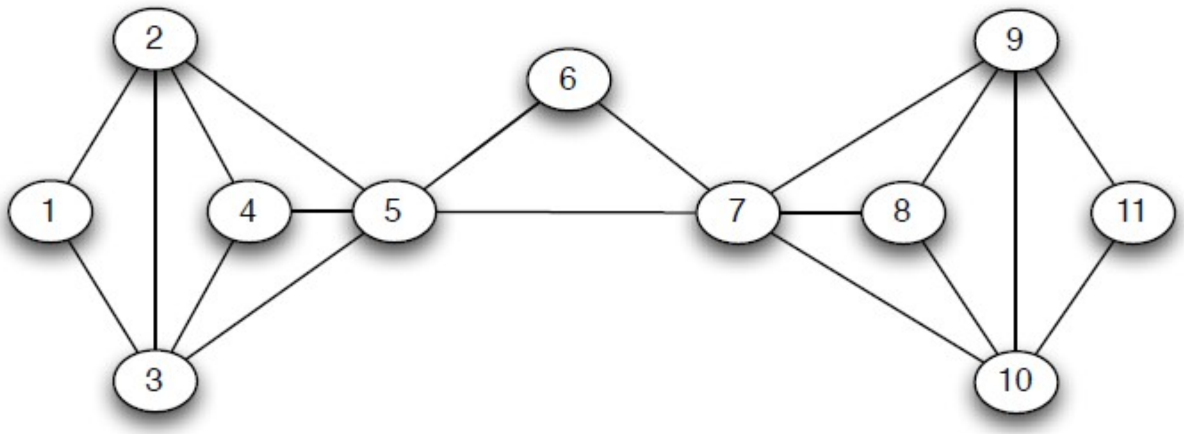
Divisive Approach

- Bridges connect tightly-knit groups in networks!
  - To find clusters, remove bridges and local bridges!
  - **Issue 1**: when there are several bridges, which one to remove?



Divisive Approach- Cnt.

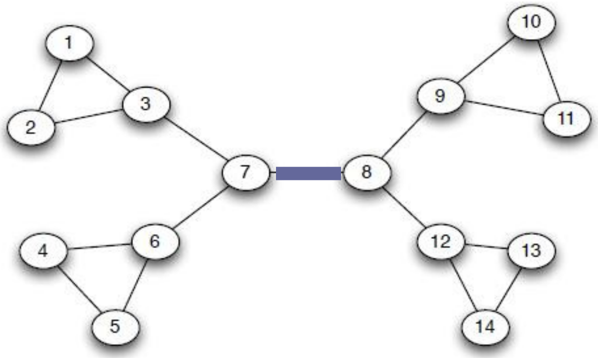
- Bridges connect tightly-knit groups in networks!
  - To find cluster, remove bridges and local bridges!
  - **Issue 2**: What if there is no bridge?

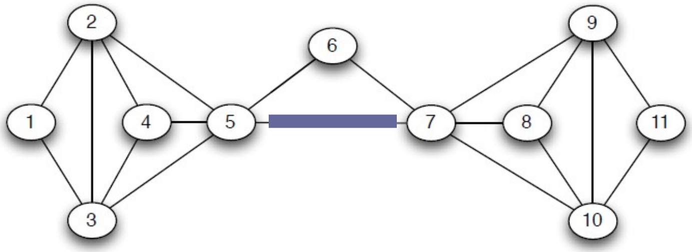


A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.

Divisive Approach- Cnt.

- Bridges form part of the **shortest path between pairs of nodes** in different parts of the network!
  - Find **edges that carry most of "traffic"** in the network and **successively remove** edges of high traffic!

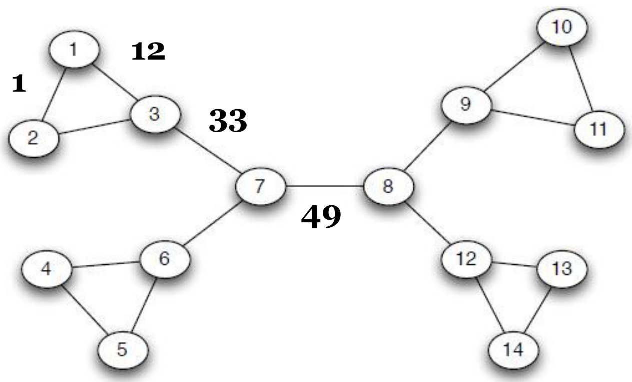


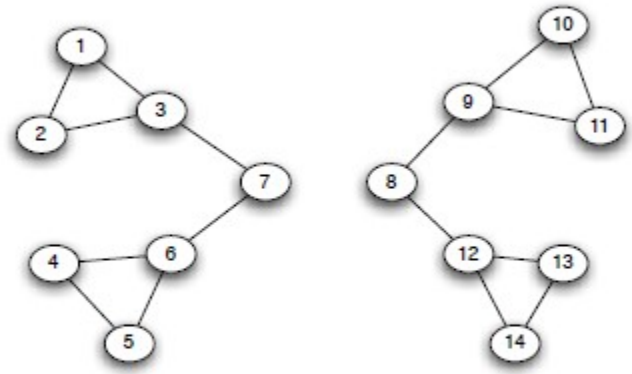


Edge Betweenness

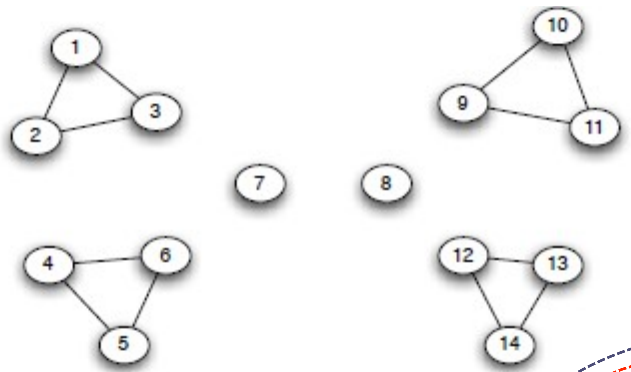
- Edge Betweenness:
  - Let's assume 1 unit of "flow" will pass over all shortest paths btw any pair of nodes A and B.
  - If there are *k* shortest path btw A and B, then 1/k units of flow will go along each shortest path!
  - Betweenness of an edge is the total amount of flow it carries!
- Girvan-Newman Algorithm:
  - Repeat until no edges are left:
    - Calculate betweenness of edges
    - Remove edges with highest betweenness

Edge Betweenness- Cnt.

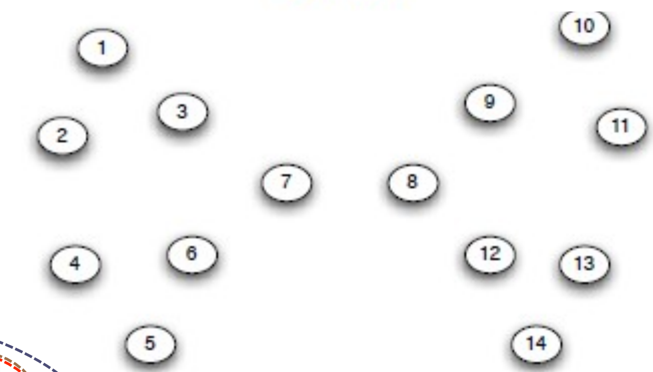




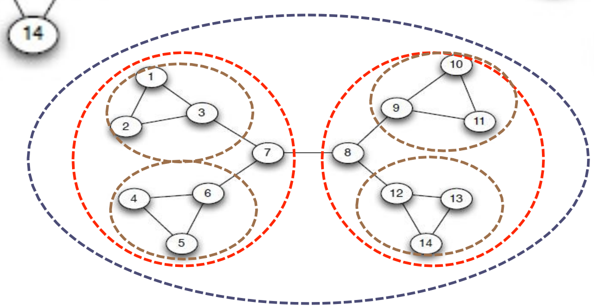
(a) *Step 1*



(b) *Step 2*



(c) *Step 3*



Communities are the resulting connected components!

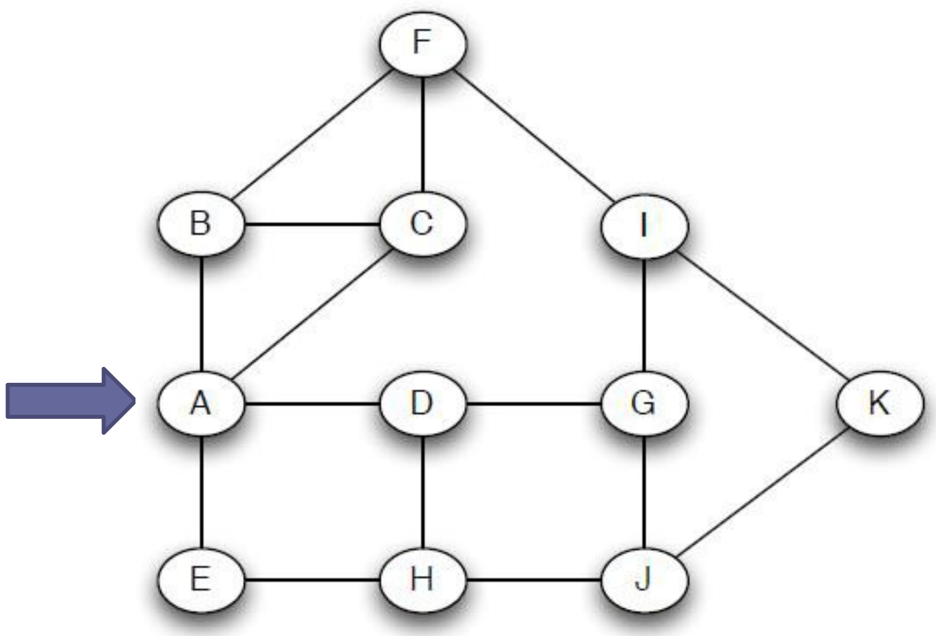
Computing Edge Betweenness

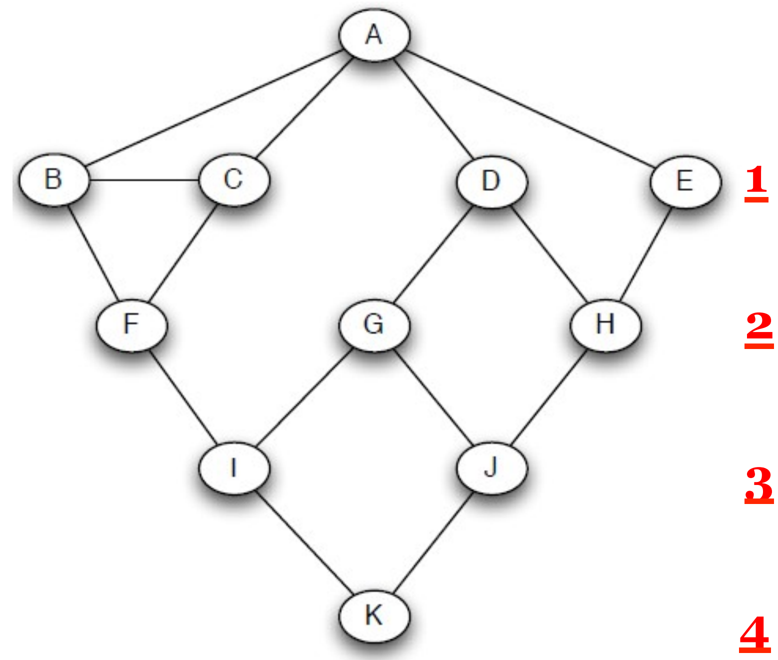
- A clever way to compute betweennesses efficiently
  - Use breadth-first Search

```
For each node A{
     Run BFS on A
     Count the number of shortest paths from A to any other node
     Determine the amount of traffic from A to other nodes
}
Compute betweenness for each edge by summing all the traffic passing over the edge
```

Computing Edge Betweenness

- A clever way to compute betweennesses efficiently
  - Use breadth-first Search
  - Consider the graph from the perspective of one node at a time!





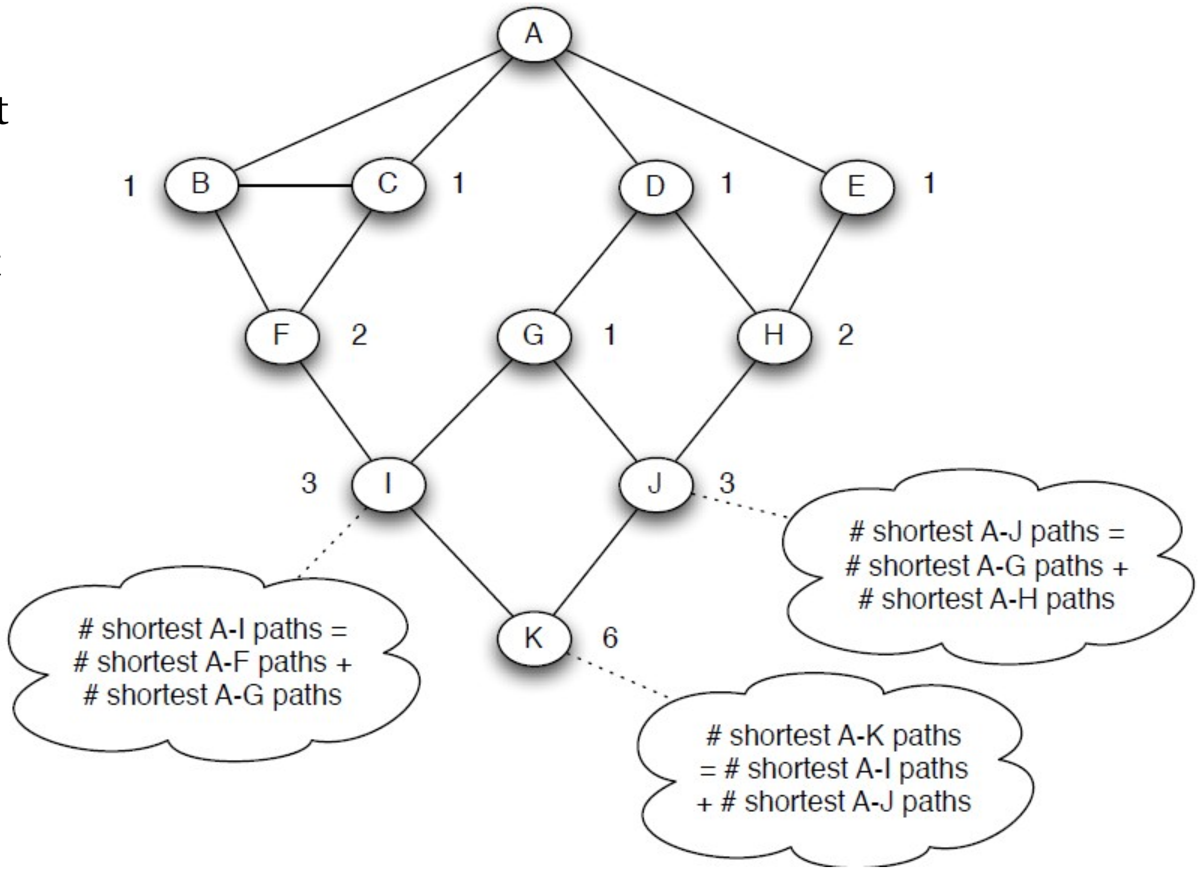
(b) *Breadth-first search starting at node A*

BFS divides a graph into layers starting at a given node, with all the nodes in layer *d* having distance *d* from A.

Computing Edge Betweenness- Cnt.

- A clever way to compute betweennesses efficiently
  - Count the number of shortest paths from A to all other nodes of the network

Number of shortest paths to each node is the sum of the number of shortest paths to all nodes directly above it!

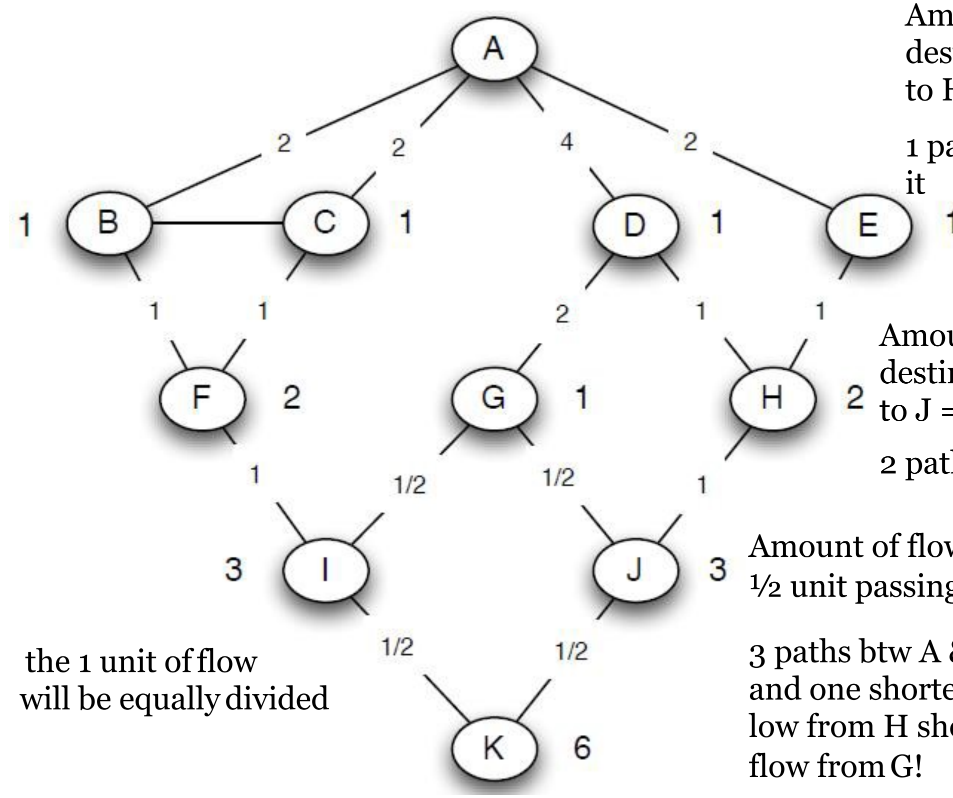


Computing Edge Betweenness- Cnt.

- A clever way to compute betweennesses efficiently
  - Determine the amount of traffic from A to others
    - If there are *k* shortest path btw A and B, then 1/k units of flow will go through each shortest path.
    - Working up from the lowest layers and computing the amount of flow that pass through each edge!

Computing Edge Betweenness- Cnt.

- A clever way to compute betweennesses efficiently
  - Determine the amount of traffic from A to others



Amount of flow from A to E: 1 unit destined for E+ 1 unit passing through to H = 2

1 path btw A & E → all flow go through it

Amount of flow from A to H: 1 unit destined for H+ 1 unit passing through to J = 2

2 paths btw A & H → flow equally divided

Amount of flow from A to J: 1 unit destined for J+ ½ unit passing through to K = 3/2

3 paths btw A & J → 2 shortest paths from H and one shortest path from G → the amount of low from H should be twice than the amount of flow from G!

3 paths from I and 3 from J → the 1 unit of flow will be equally divided

Amount of flow from A to K: 1 unit

Computing Edge Betweenness- Cnt.

- A clever way to compute betweennesses efficiently
  - Use breadth-first Search

```
For each node A{
      Run BFS on A
      Count the number of shortest paths from A to any other node
      Determine the amount of traffic from A to other nodes
}
Compute betweenness for each edge by summing all the traffic passing over the edge and
divide by 2
```

Note that we count the flow between each pair of nodes A and B twice (once when running BFS from A and once when running BFS from B)! So, we need to divide resulting values by 2!

Reading

- Ch.03 Strong and Weak Ties [NCM]
- Why we twitter: understanding microblogging usage and communities. WebKDD'07.
- Community detection in graphs. Fortunato, S. Physics reports 2010
- Searching for superspreaders of information in real-world social media. Pei, S., et al. Scientific reports 2014.