



AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

19CSE337 Social Networking Security

Lecture 8



Topics to Discuss

- Betweenness Centrality

The header features a blue background with a grid of circular icons. These icons include a dollar sign, a wrench, a car, a sun, a shopping cart, a briefcase, a smartphone, a family silhouette, a lightbulb, a handshake, a radio tower, and headphones. The title 'Betweenness Centrality' is written in a large, orange, sans-serif font across the top right.

Betweenness Centrality

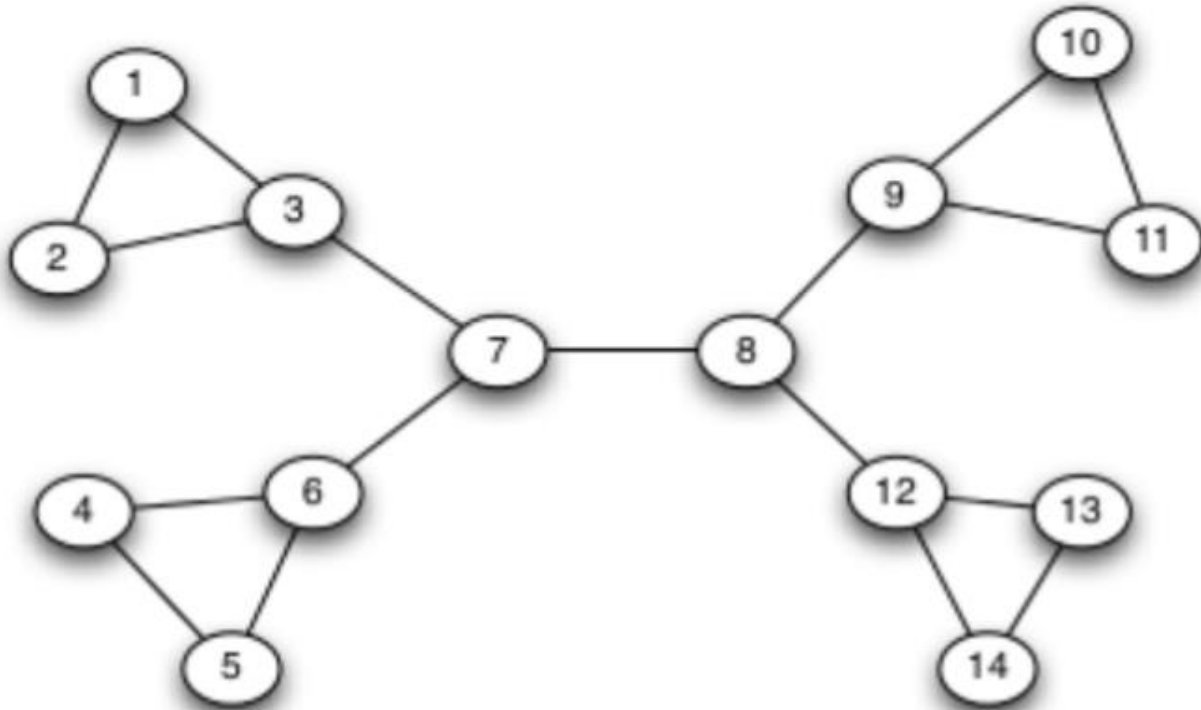
- Betweenness centrality measures the number of times a node lies on the shortest path between other nodes.
- It is a measure of how often a node appears in the shortest path connecting two other nodes.
- Nodes with a high betweenness centrality score are the ones that most frequently act as 'bridges' between other nodes.
- They form the shortest pathways of communication within the network.
- Usually this would indicate important gatekeepers of information between groups.



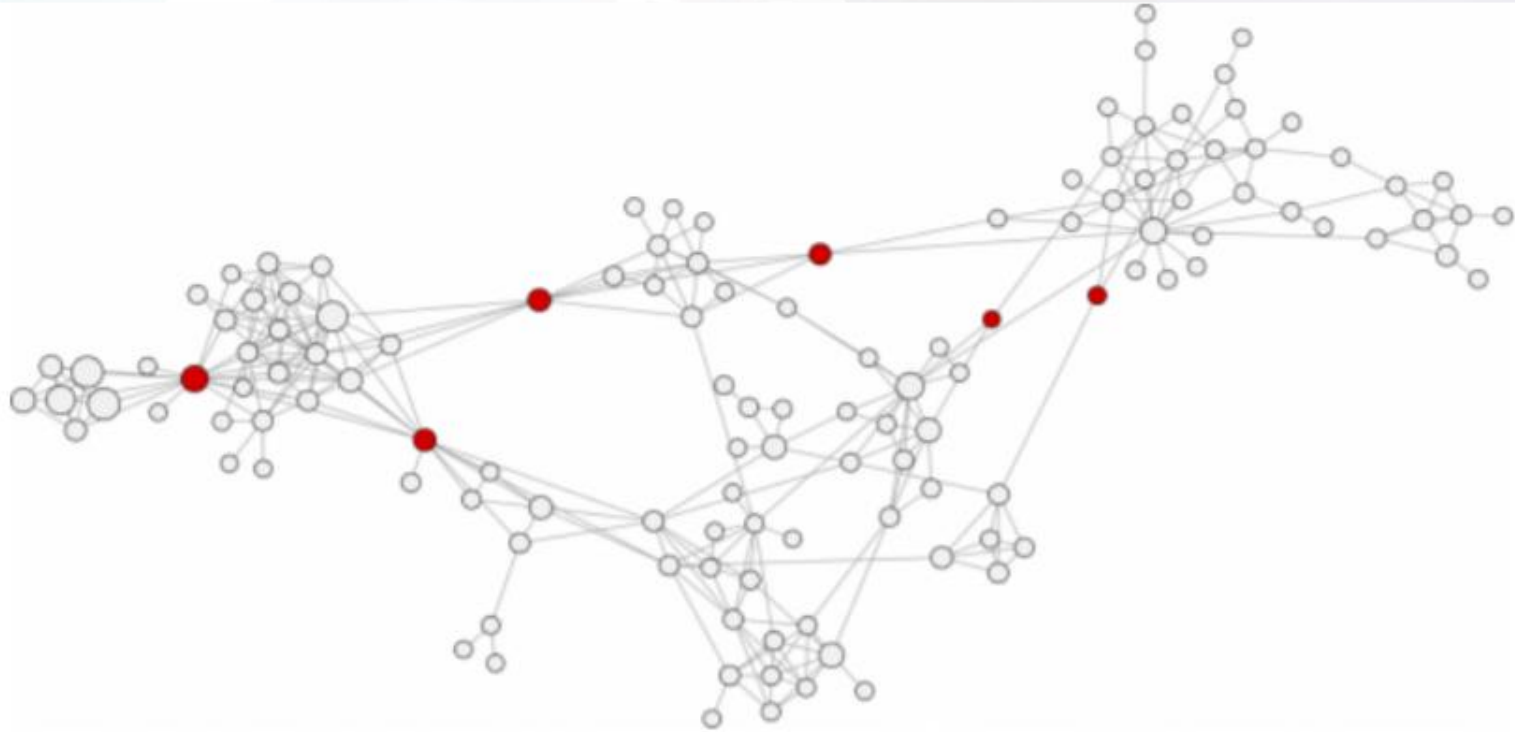
Betweenness Centrality

- Useful in finding the individuals who influence the flow around a system.
- A high betweenness count could indicate someone holds authority over disparate clusters in a network, or just that they are on the periphery of both clusters.

Betweenness Centrality

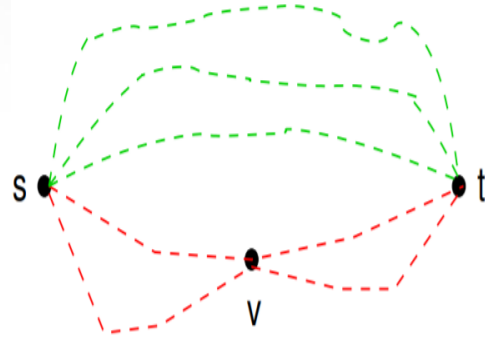


Betweenness Centrality



Betweenness Computation

- The betweenness centrality of a node V is defined as the proportion of shortest paths between all pairs of nodes that go through V .
- Consider a node v and two other nodes s and t .
- Each shortest path between s and t shown in green doesn't pass through node v .
- Each shortest path between s and t shown in red passes through node v .





Betweenness Computation

- Any shortest path between nodes s and t will be called an s - t shortest path.
- Let σ_{st} denote the number of all s - t shortest paths.
- Let $\sigma_{st}(v)$ denote the number of all s - t shortest paths that pass through node v .
- Consider the ratio $(\sigma_{st}(v) / \sigma_{st})$:
 - This gives the fraction of s - t shortest paths passing through v .
 - The larger the ratio, the more important v is with respect to the pair of nodes s and t .
 - To properly measure the importance of a node v , we need to consider all pairs of nodes (not involving v).



Betweenness Computation

- The betweenness centrality of a node v , denoted by $\beta(v)$, is defined by

$$\beta(v) = \sum_{\substack{s, t \\ s \neq v, t \neq v}} \left[\frac{\sigma_{st}(v)}{\sigma_{st}} \right]$$

- Note: For two nodes x and y , if $\beta(x) > \beta(y)$, then x is more central than y .



Betweenness Computation

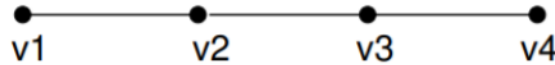
- Suppose we want to compute $\beta(v)$ for some node v . The formula suggests the following steps.
 - Set $\beta(v) = 0$.
 - For each pair of nodes s and t such that $s \neq v$ and $t \neq v$,
 - Compute σ_{st} and $\sigma_{st}(v)$.
 - Set $\beta(v) = \beta(v) + \sigma_{st}(v)/\sigma_{st}$.
 - Output $\beta(v)$.

Betweenness Computation



- Note: Here, there is only one path between any pair of nodes. (So, that path is also the shortest path.)
- Consider the computation of $\beta(v2)$ first.
- The s-t pairs to be considered are: $(v1, v3)$, $(v1, v4)$ and $(v3, v4)$.
- For the pair $(v1, v3)$:
 - The number of shortest paths between $v1$ and $v3$ is 1; thus, $\sigma_{v1,v3} = 1$.
 - The (only) path between $v1$ and $v3$ passes through $v2$; thus, $\sigma_{v1,v3}(v2) = 1$.
 - So, the ratio $\sigma_{v1,v3}(v2)/\sigma_{v1,v3} = 1$.

Betweenness Computation



- In a similar manner, $\sigma_{v_1, v_4} = 1$, and $\sigma_{v_1, v_4}(v_2) = 1$.
- So, for the pair (v_1, v_4) , the ratio $\sigma_{v_1, v_4}(v_2) / \sigma_{v_1, v_4} = 1$.
- For the pair (v_3, v_4) , $\sigma_{v_3, v_4} = 1$, and $\sigma_{v_3, v_4}(v_2) = 0$.
- So, for the pair (v_3, v_4) , the ratio $\sigma_{v_3, v_4}(v_2) / \sigma_{v_3, v_4} = 0$.



Betweenness Computation

- Therefore,
- $\beta(v_2) = 1$ (for the pair (v_1, v_3))
+ 1 (for the pair (v_1, v_4))
+ 0 (for the pair (v_3, v_4))
= 2.
- Repeat the procedure to compute $\beta(v_1)$, $\beta(v_3)$, $\beta(v_4)$.
- The values are $\beta(v_1)=0$, $\beta(v_3)=2$, $\beta(v_4)=0$ respectively.
- In the above example, nodes v_2 and v_3 are most influential nodes.



Betweenness Computation

- Graphs with larger number of nodes will tend to have higher centrality than graphs with smaller number of nodes.
- Because in large graphs, there are more nodes, s and t , to choose from to compute the centrality of the nodes.
- So, if we want to compare betweenness centrality across networks, it's useful to normalize.



Betweenness Computation

- Normalization can be done by dividing the betweenness centrality of node v by the number of possible pairs of nodes in the network, excluding node v .
- It is different for directed and undirected graphs.
 - For undirected graphs, $[\beta(v)/((N-1)(N-2)/2)]$
 - For directed graphs, $[\beta(v)/(N-1)(N-2)]$ (Since you have twice the number of pairs because for any pair $s-t$, you could have a path from $t-s$ also).
- So, in our example, the normalized value for v_2 is $[\beta(v_2)/((N-1)(N-2)/2)]=2/3=0.67$

Betweenness Centrality using NetworkX

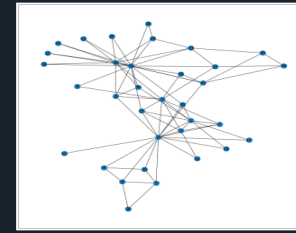
Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\mails\.spyder-py3\untitled0.py

```
1 from pylab import *
2 import networkx as nx
3 # G is the Karate Club Graph, the famous social graph published in 1977
4 G = nx.karate_club_graph()
5 plt.figure(figsize=(15, 15))
6 nx.draw_networkx(G, with_labels=True)
7
8 #calculate and print betweenness centrality
9 bet_centrality = nx.betweenness_centrality(G, normalized=True,
10 endpoints=False)
11 print(bet_centrality)
12
```

C:\Users\mails\.spyder-py3



Help Variable Explorer Plots Files

Console 1/A X

IPython 7.29.0 -- An enhanced Interactive Python.

```
In [1]: runfile('C:/Users/mails/.spyder-py3/untitled0.py', wdir='C:/Users/
mails/.spyder-py3')
{0: 0.43763528138528146, 1: 0.053936688311688304, 2: 0.14365680615680618, 3:
0.011909271284271283, 4: 0.0006313131313131313, 5: 0.02998737373737374, 6:
0.029987373737373736, 7: 0.0, 8: 0.05592682780182781, 9: 0.0008477633477633478, 10:
0.0006313131313131313, 11: 0.0, 12: 0.0, 13: 0.04586339586339586, 14: 0.0, 15: 0.0,
16: 0.0, 17: 0.0, 18: 0.0, 19: 0.03247504810004811, 20: 0.0, 21: 0.0, 22: 0.0, 23:
0.017613636363636363, 24: 0.0022095959595959595, 25: 0.0038404882154882154, 26: 0.0,
27: 0.02233345358345358, 28: 0.0017947330447330447, 29: 0.0029220779220779218, 30:
0.014411976911976909, 31: 0.13827561327561325, 32: 0.145247113997114, 33:
0.30407497594997596}
```

IPython console History

LSP Python: ready conda: base (Python 3.9.7) Line 12, Col 1 UTF-8 CRLF RW Mem 42%

20:15
16-01-2022

Betweenness Centrality using NetworkX

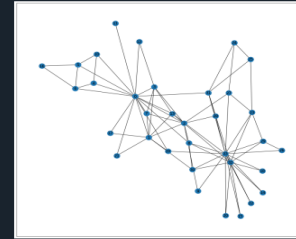
Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\mails\.spyder-py3\untitled0.py

```
untitled0.py x DegCen.py x
1 from pylab import *
2 from operator import itemgetter
3 import networkx as nx
4 # G is the Karate Club Graph, the famous social graph published in 1977
5 G = nx.karate_club_graph()
6 plt.figure(figsize=(15, 15))
7 nx.draw_networkx(G, with_labels = True)
8
9 #calculate and print betweenness centrality
10 bet_centrality = nx.betweenness_centrality(G, normalized = True,
11                                           endpoints = False)
12 #To get top 5 nodes
13 sorted_degree = sorted(bet_centrality.items(), key=itemgetter(1),
14                       reverse=True)
15 for d in sorted_degree[:5]:
16     print(d)
17
```

C:\Users\mails\.spyder-py3



Help Variable Explorer Plots Files

Console 1/A x

```
In [4]: runfile('C:/Users/mails/.spyder-py3/untitled0.py', wdir='C:/Users/
mails/.spyder-py3')
(0, 0.43763528138528146)
(33, 0.30407497594997596)
(32, 0.145247113997114)
(2, 0.14365680615680618)
(31, 0.13827561327561325)
```

In [5]:

IPython console History

LSP Python: ready conda: base (Python 3.9.7) Line 17, Col 1 UTF-8 CRLF RW Mem 45%

ENG IN 20:29 16-01-2022



Thanks.....