

# Properties of Networks, Random Graph Models

CS224W: Analysis of Networks  
Jure Leskovec, Stanford University  
<http://cs224w.stanford.edu>



# **Network Properties: How to Measure a Network**

# Plan: Key Network Properties

**Degree distribution:**  $P(k)$

**Path length:**  $h$

**Clustering coefficient:**  $C$

**Connected components:**  $s$

Definitions will be presented for undirected graphs, sometimes we will explicitly mention extensions to directed graphs, and sometimes extensions will be obvious

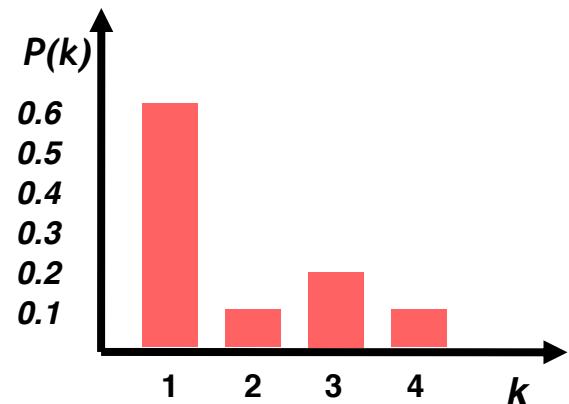
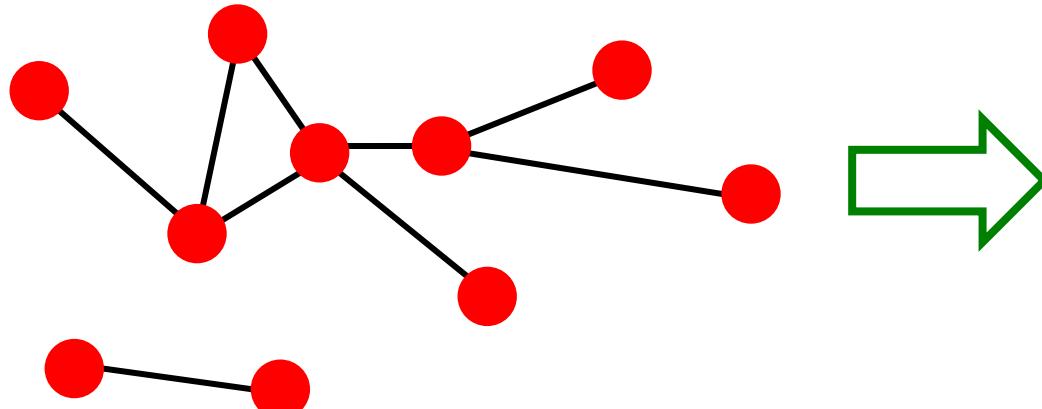
# (1) Degree Distribution

- **Degree distribution  $P(k)$ :** Probability that a randomly chosen node has degree  $k$

$$N_k = \# \text{ nodes with degree } k$$

- Normalized histogram:

$$P(k) = N_k / N \rightarrow \text{plot}$$



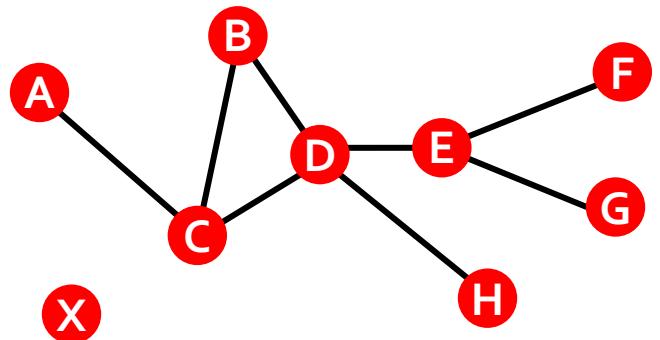
For directed graphs we have separate in- and out-degree distributions.

# (2) Paths in a Graph

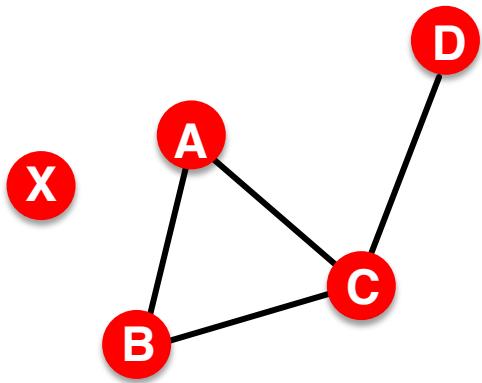
- A **path** is a sequence of nodes in which each node is linked to the next one

$$P_n = \{i_0, i_1, i_2, \dots, i_n\} \quad P_n = \{(i_0, i_1), (i_1, i_2), (i_2, i_3), \dots, (i_{n-1}, i_n)\}$$

- A path can intersect itself and pass through the same edge multiple times
  - E.g.: ACBDCDEG

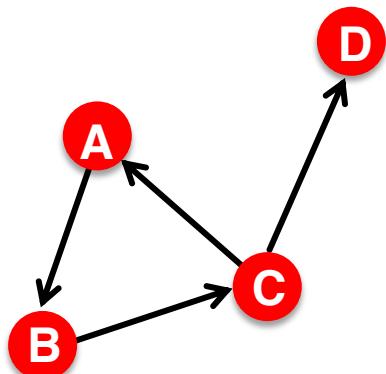


# Distance in a Graph



$$h_{B,D} = 2$$

$$h_{A,X} = \infty$$



$$h_{B,C} = 1, h_{C,B} = 2$$

- **Distance (shortest path, geodesic)** between a pair of nodes is defined as the number of edges along the shortest path connecting the nodes

- \*If the two nodes are not connected, the distance is usually defined as infinite (or zero)

- In **directed graphs**, paths need to follow the direction of the arrows

- Consequence: Distance is **not symmetric**:  $h_{B,C} \neq h_{C,B}$

# Network Diameter

- **Diameter:** The maximum (shortest path) distance between any pair of nodes in a graph
- **Average path length** for a connected graph or a strongly connected directed graph

$$\bar{h} = \frac{1}{2E_{\max}} \sum_{i,j \neq i} h_{ij}$$

- $h_{ij}$  is the distance from node  $i$  to node  $j$
- $E_{\max}$  is the max number of edges (total number of node pairs) =  $n(n-1)/2$

- Many times we compute the average only over the connected pairs of nodes (that is, we ignore “infinite” length paths)
- Note that this measure also applies to (strongly) connected components of a graph

# (3) Clustering Coefficient

## ■ Clustering coefficient (for undirected graphs):

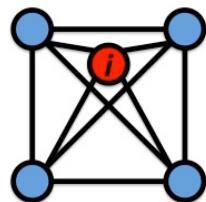
- How connected are  $i$ 's neighbors to each other?

- Node  $i$  with degree  $k_i$

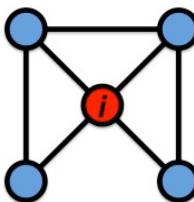
- $C_i \in [0, 1]$

- $C_i = \frac{2e_i}{k_i(k_i - 1)}$  where  $e_i$  is the number of edges between the neighbors of node  $i$

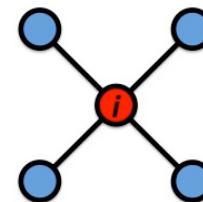
Note  $k_i(k_i - 1)$  is max number of edges between the  $k_i$  neighbors



$$C_i = 1$$



$$C_i = 1/2$$



$$C_i = 0$$

Clustering coefficient is undefined (or defined to be 0) for nodes with degree 0 or 1

## ■ Average clustering coefficient:

$$C = \frac{1}{N} \sum_i C_i$$

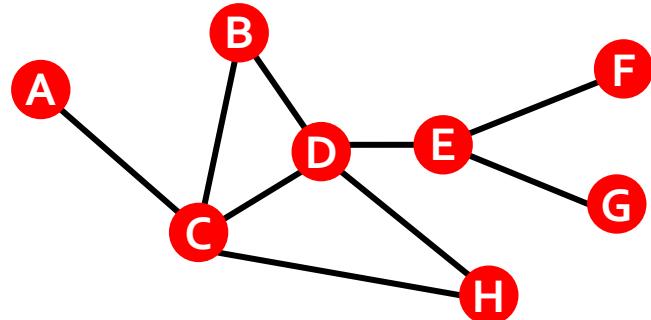
# Clustering Coefficient

## ■ Clustering coefficient (for undirected graphs):

- How connected are  $i$ 's neighbors to each other?
- Node  $i$  with degree  $k_i$

$$\text{■ } C_i = \frac{2e_i}{k_i(k_i - 1)}$$

where  $e_i$  is the number of edges between the neighbors of node  $i$



$$k_B=2, \ e_B=1, \ C_B=2/2 = 1$$

$$k_D=4, \ e_D=2, \ C_D=4/12 = 1/3$$

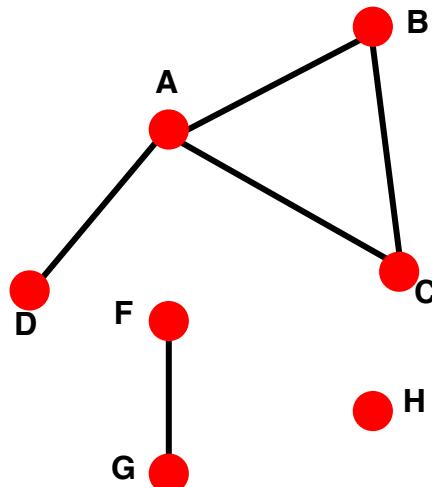
Avg. clustering:  $C=0.33$

# (4) Connectivity

## ■ Size of the largest connected component

- Largest set where any two vertices can be joined by a path

## ■ Largest component = Giant component



### How to find connected components:

- Start from random node and perform Breadth First Search (BFS)
- Label the nodes that BFS visits
- If all nodes are visited, the network is connected
- Otherwise find an unvisited node and repeat BFS

# Summary: Key Network Properties

Degree distribution:  $P(k)$

Path length:  $h$

Clustering coefficient:  $C$

Connected components:  $s$

**Let's measure these properties on  
real-world networks!**

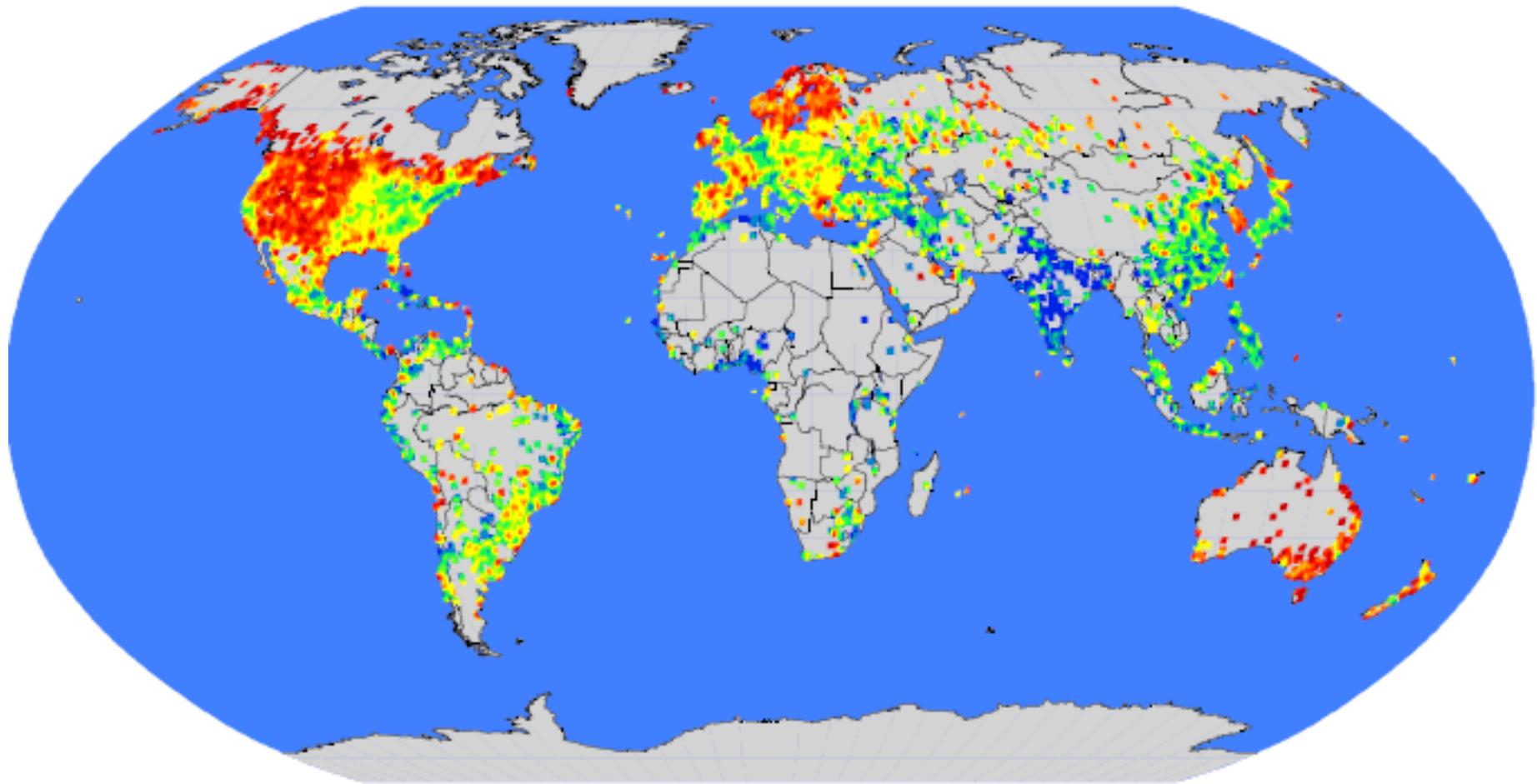
# MSN Messenger



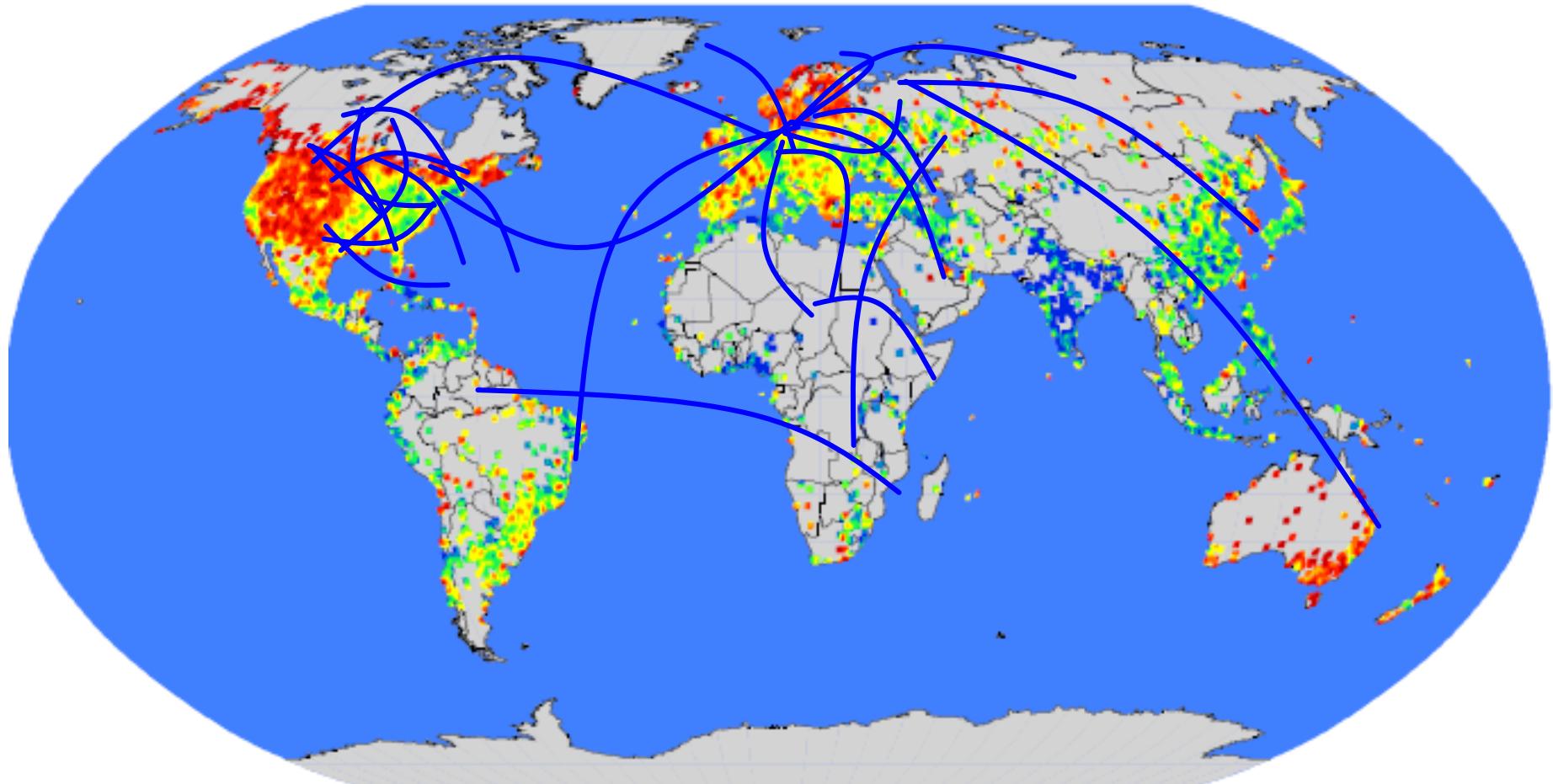
## MSN Messenger: ■ 1 month of activity

- 245 million users logged in
- 180 million users engaged in conversations
- More than 30 billion conversations
- More than 255 billion exchanged messages

# Geography of Communication

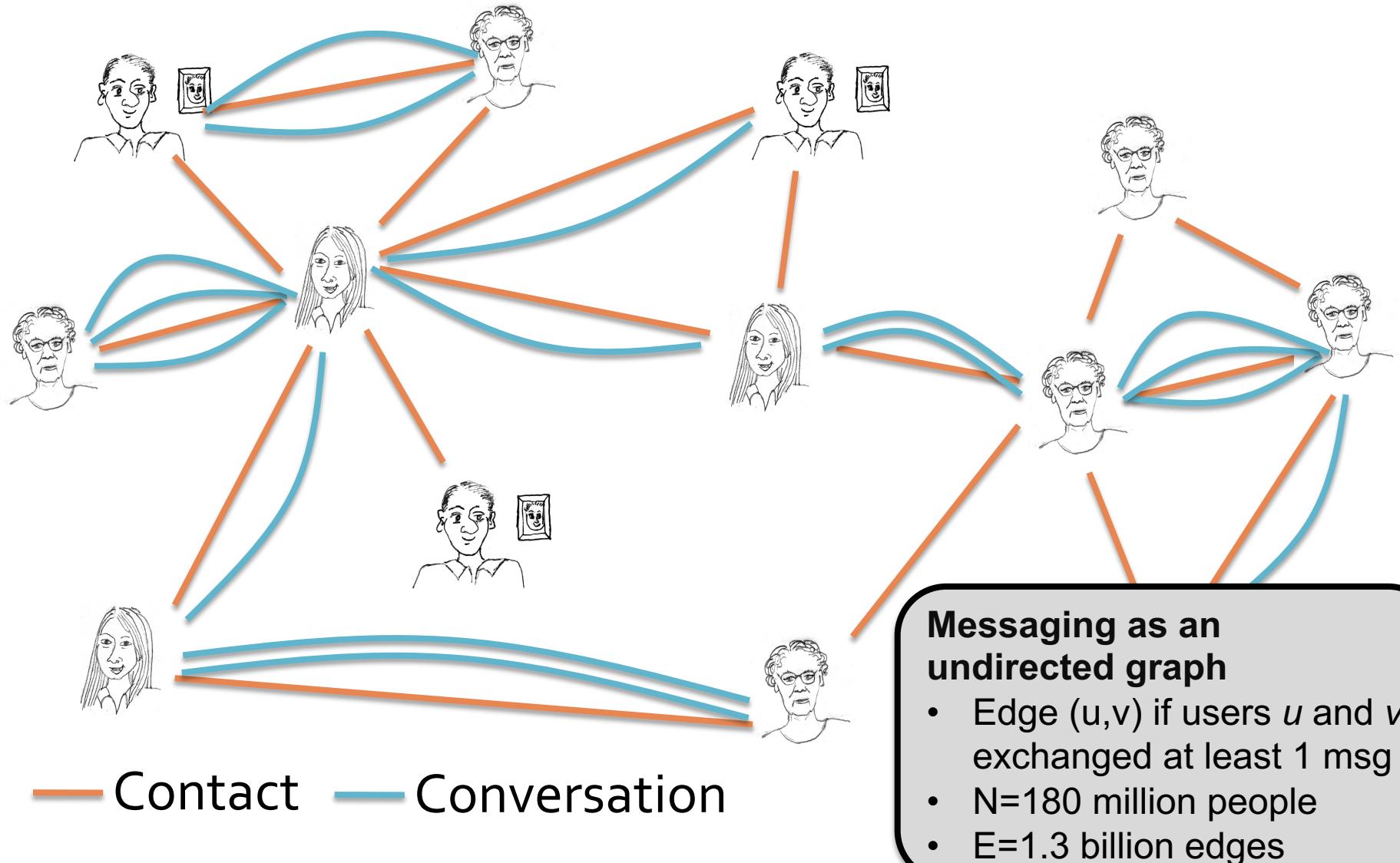


# Communication Network

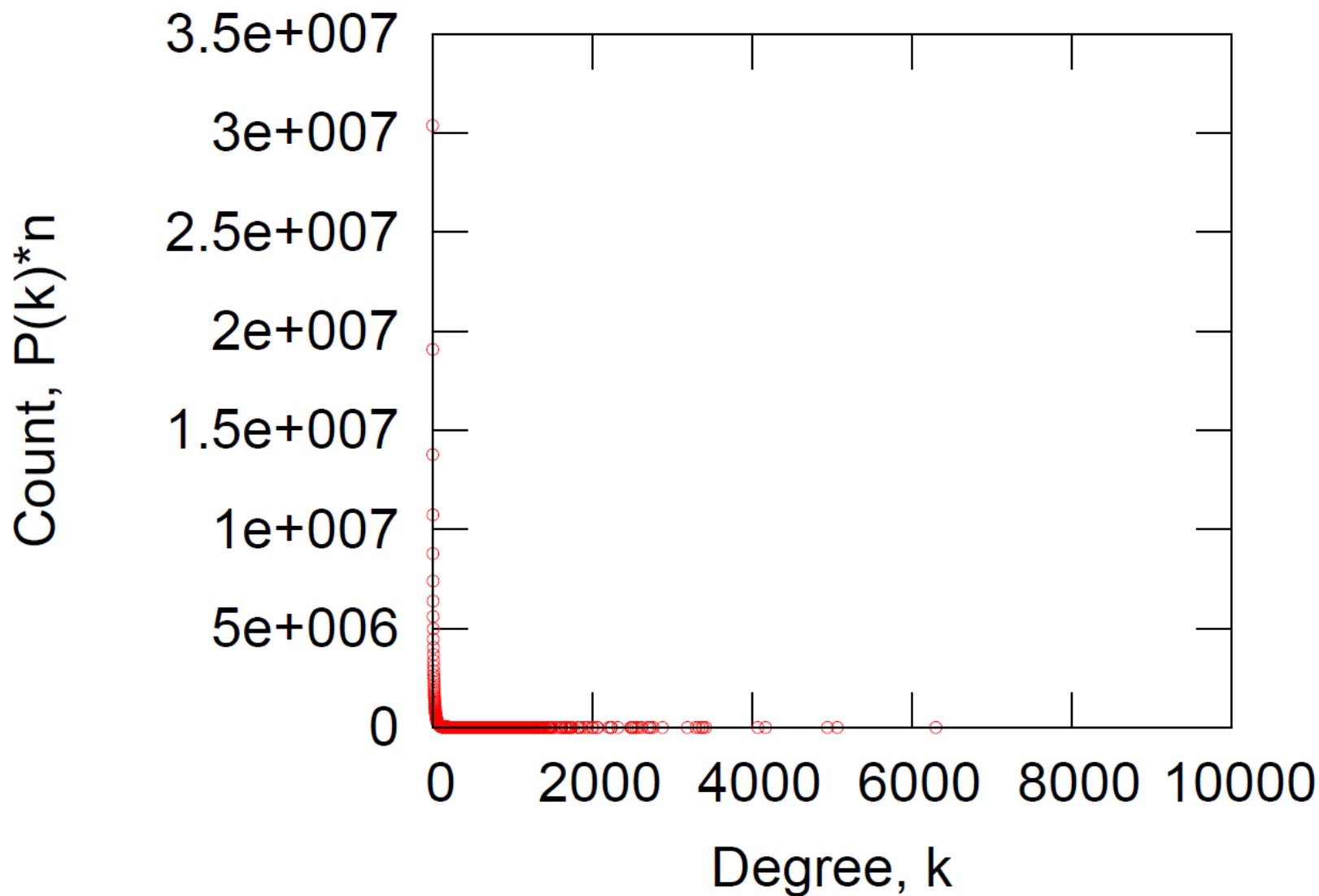


**Network:** 180M people, 1.3B edges

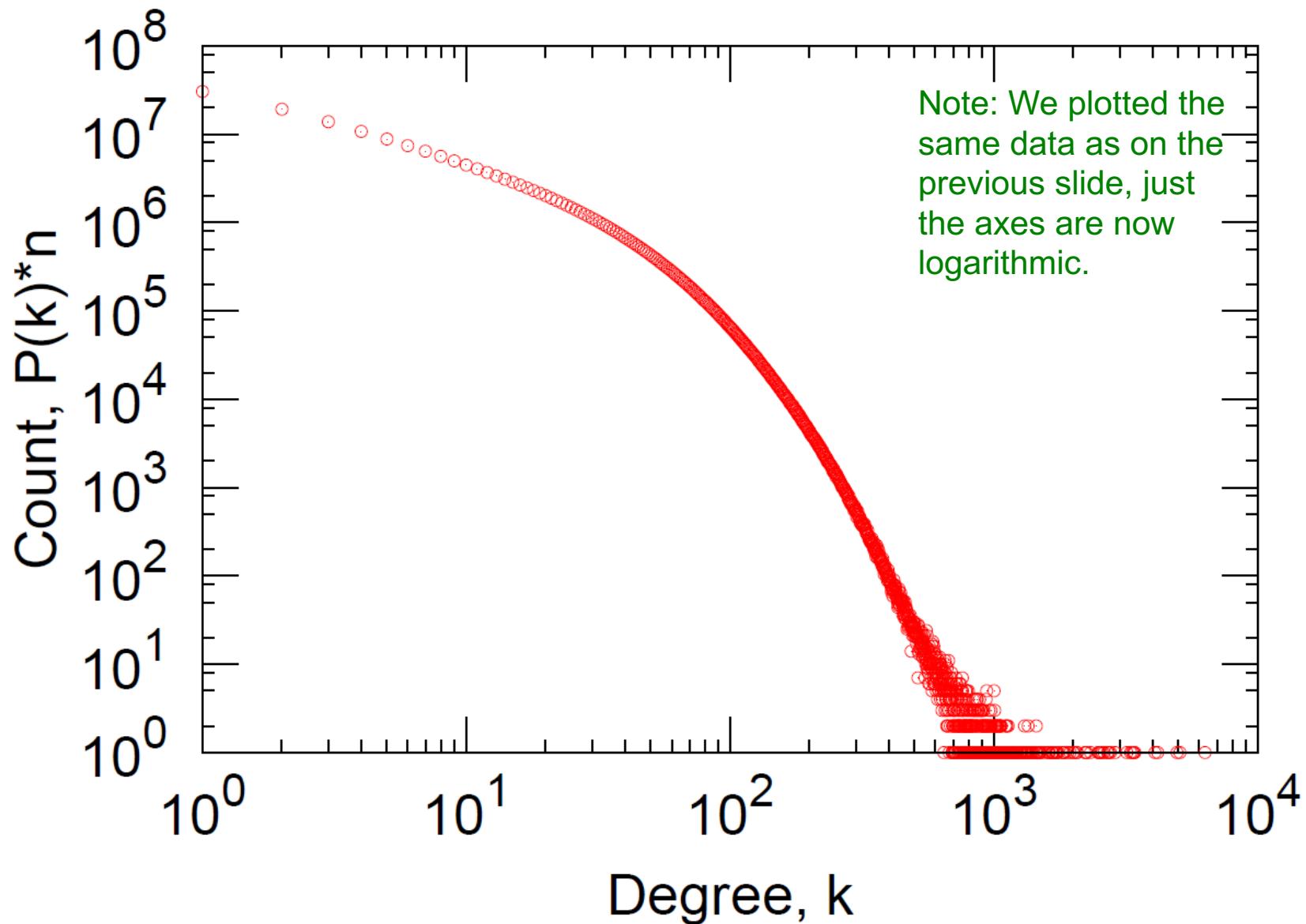
# Messaging as a Multigraph



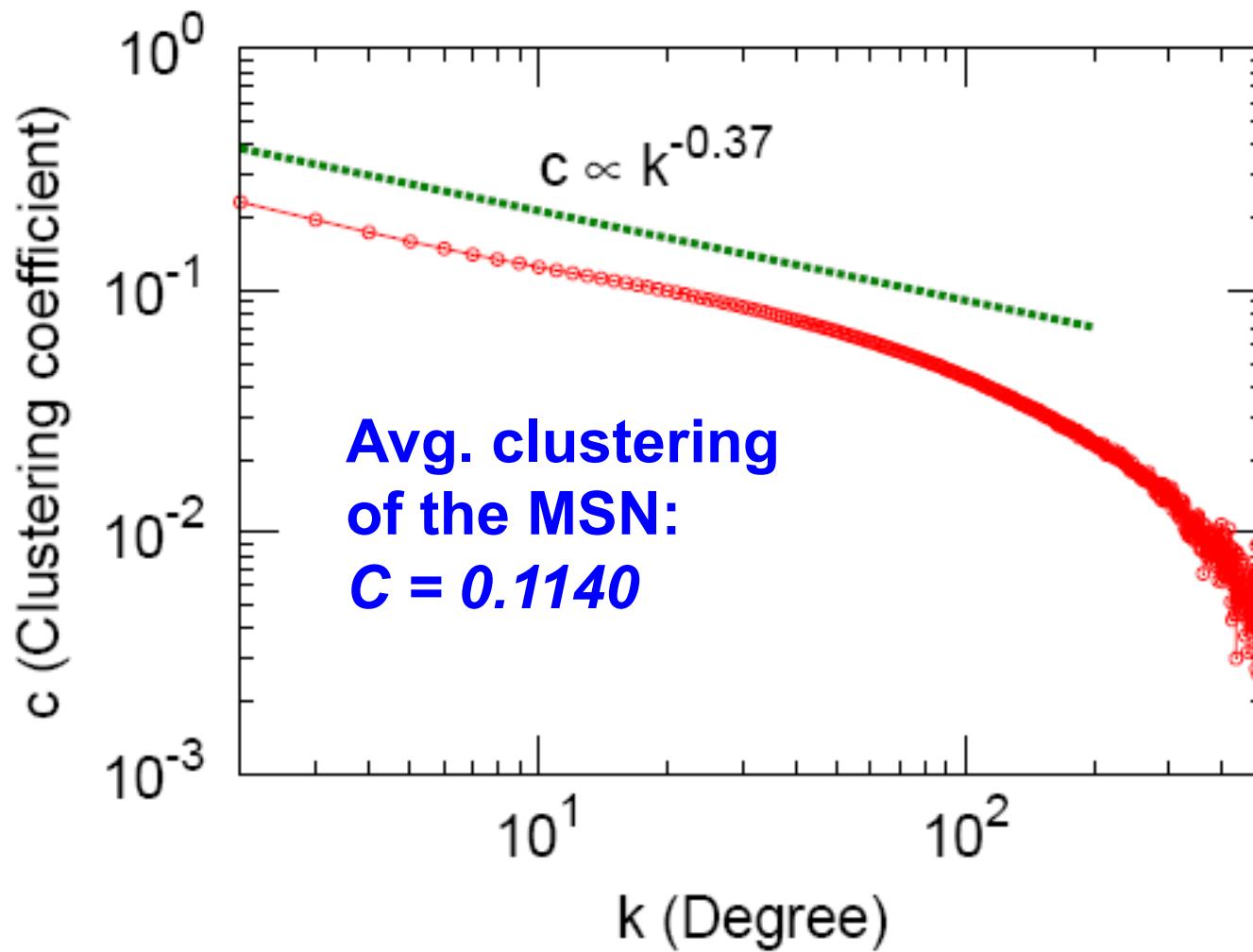
# MSN: (1) Degree Distribution



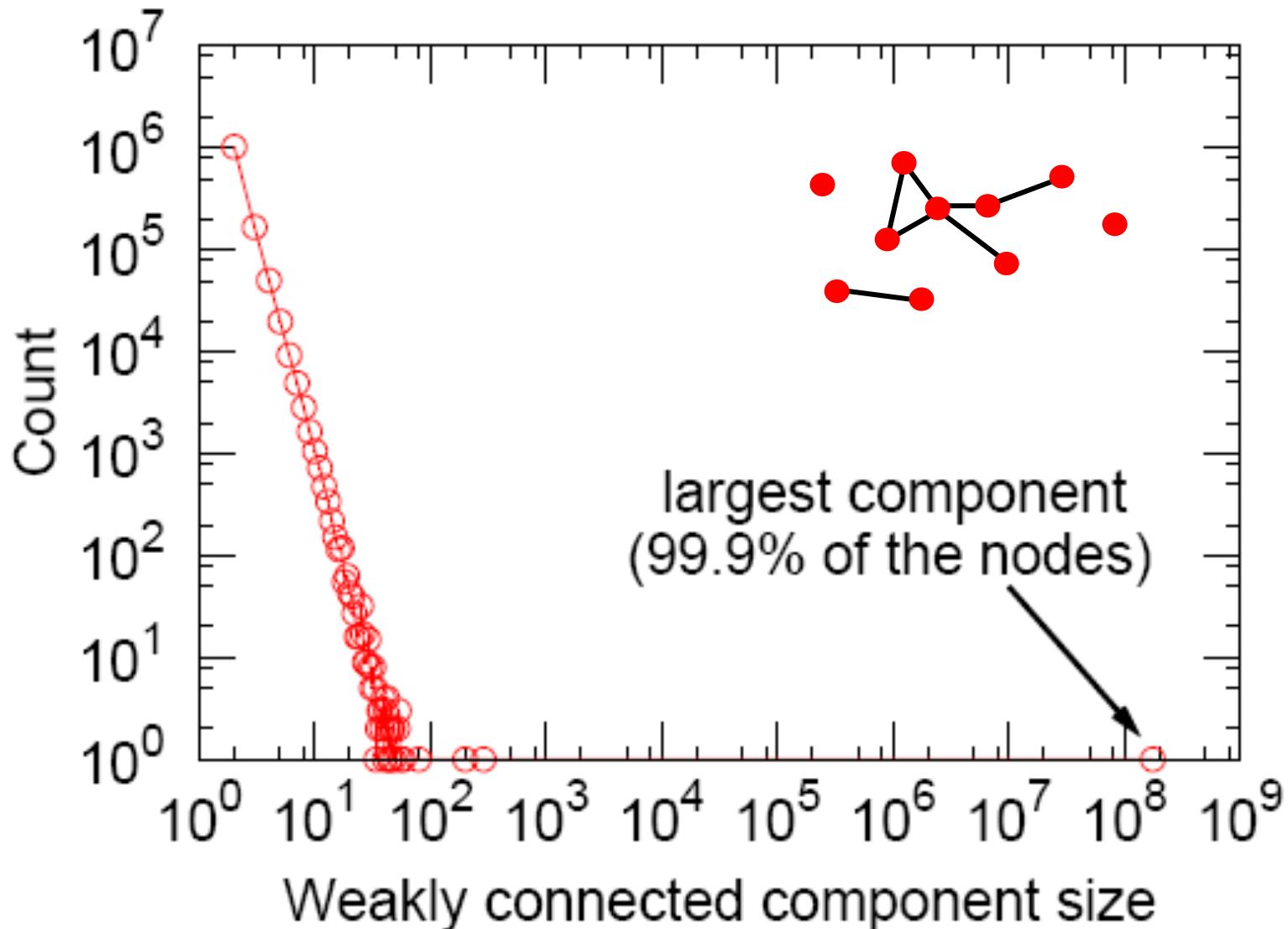
# MSN: Log-Log Degree Distribution



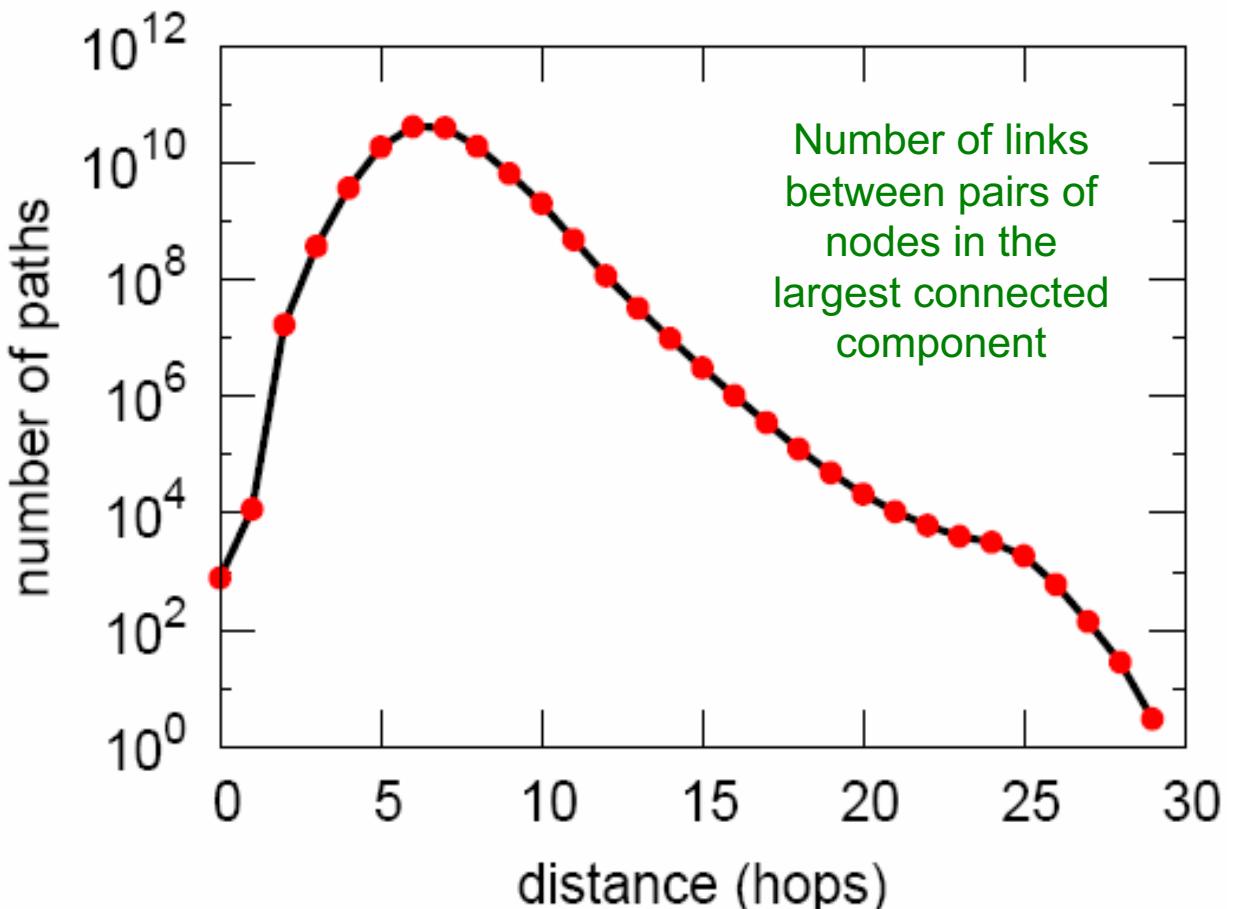
# MSN: (2) Clustering



# MSN: (3) Connected Components



# MSN: (4) Diameter of WCC



Avg. path length 6.6  
90% of the nodes can be reached in < 8 hops

Steps	#Nodes
0	1
1	10
2	78
3	3,96
4	8,648
5	3,299,252
6	28,395,849
7	79,059,497
8	52,995,778
9	10,321,008
10	1,955,007
11	518,410
12	149,945
13	44,616
14	13,740
15	4,476
16	1,542
17	536
18	167
19	71
20	29
21	16
22	10
23	3
24	2
25	3

# MSN: Key Network Properties

Degree distribution:

*Heavily skewed;  
avg. degree = 14.4*

Path length:

6.6

Clustering coefficient:

0.11

Connectivity:

*giant component*

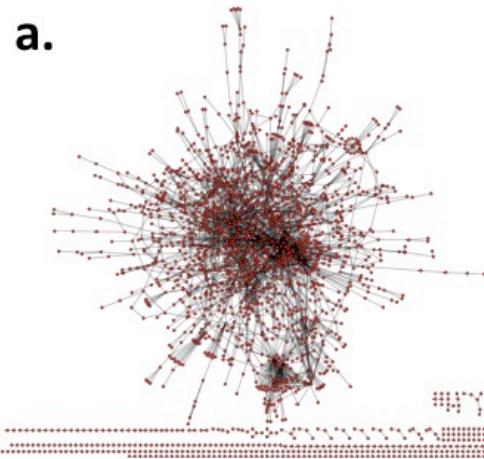
Are these values “expected”?

Are they “surprising”?

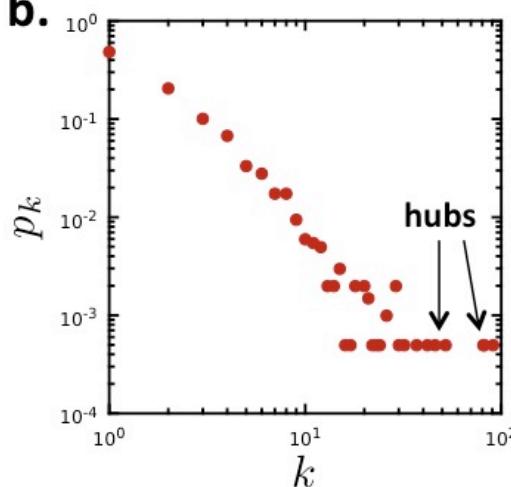
To answer this we need a model!

# Another example: PPI Network

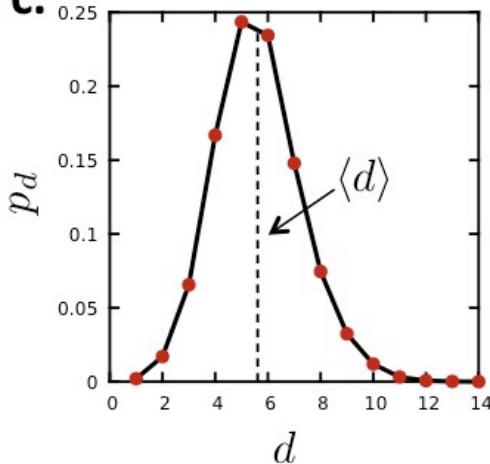
a.



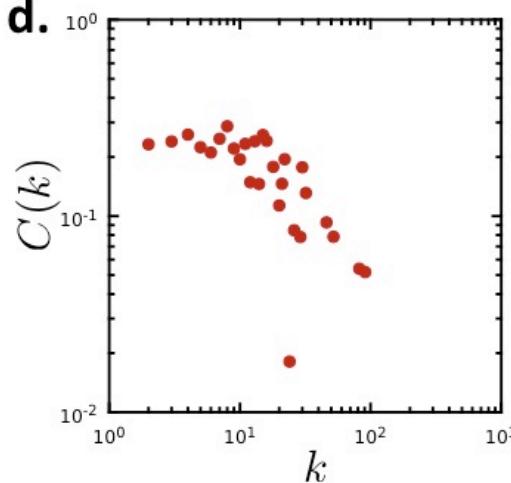
b.



c.



d.



a. Undirected network

$N=2,018$  proteins as nodes

$E=2,930$  binding interactions as links.

b. Degree distribution:

Skewed. Average degree  $\langle k \rangle = 2.90$

c. Diameter:

Avg. path length = 5.8

d. Clustering:

Avg. clustering = 0.12

Connectivity: 185 components

the largest component has  
1,647 nodes (81% of nodes)

# Erdös-Rényi Random Graph Model

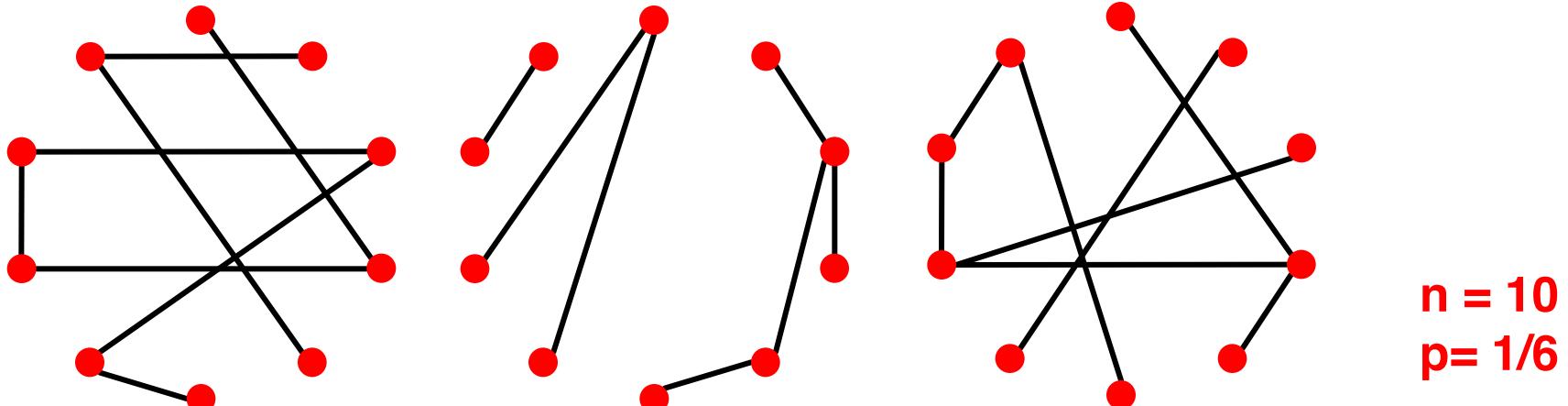
# Simplest Model of Graphs

- Erdös-Renyi Random Graphs [Erdös-Renyi, '60]
- Two variants:
  - $G_{np}$ : undirected graph on  $n$  nodes where each edge  $(u,v)$  appears i.i.d. with probability  $p$
  - $G_{nm}$ : undirected graph with  $n$  nodes, and  $m$  edges picked uniformly at random

What kind of networks do such models produce?

# Random Graph Model

- **$n$  and  $p$  do not uniquely determine the graph!**
  - The graph is a result of a random process
- We can have many different realizations given the same  $n$  and  $p$



# Properties of $G_{np}$

Degree distribution:  $P(k)$

Path length:  $h$

Clustering coefficient:  $C$

What are the values of  
these properties for  $G_{np}$ ?

# Degree Distribution

- **Fact:** Degree distribution of  $G_{np}$  is binomial.
- Let  $P(k)$  denote the fraction of nodes with degree  $k$ :

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

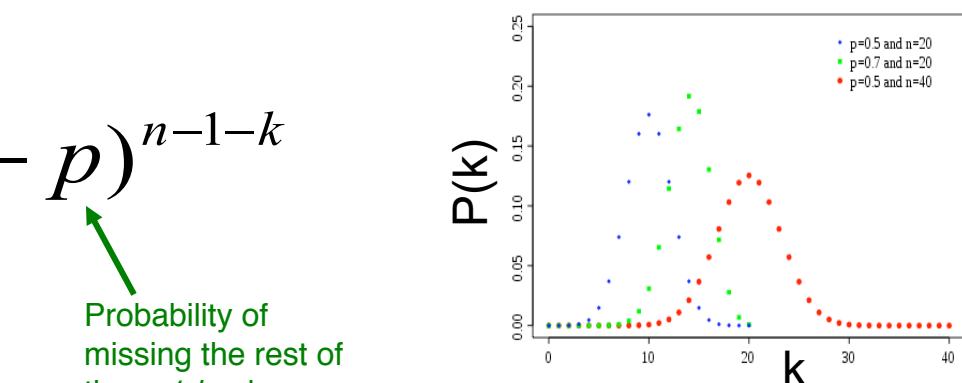
Diagram annotations:

- Select  $k$  nodes out of  $n-1$  (points to the binomial coefficient term)
- Probability of having  $k$  edges (points to  $p^k$ )
- Probability of missing the rest of the  $n-1-k$  edges (points to  $(1-p)^{n-1-k}$ )

Mean, variance of a binomial distribution

$$\bar{k} = p(n-1)$$

$$\sigma^2 = p(1-p)(n-1)$$



$$\frac{\sigma}{\bar{k}} = \left[ \frac{1-p}{p} \frac{1}{(n-1)} \right]^{1/2} \approx \frac{1}{(n-1)^{1/2}}$$

By the law of large numbers, as the network size increases, the distribution becomes increasingly narrow—we are increasingly confident that the degree of a node is in the vicinity of  $k$ .

# Clustering Coefficient of $G_{np}$

- Remember:  $C_i = \frac{2e_i}{k_i(k_i - 1)}$  Where  $e_i$  is the number of edges between i's neighbors
- Edges in  $G_{np}$  appear i.i.d. with prob.  $p$
- So, expected  $E[e_i]$  is:  $= p \frac{k_i(k_i - 1)}{2}$ 
  - Each pair is connected with prob.  $p$
  - Number of distinct pairs of neighbors of node  $i$  of degree  $k_i$
- Then  $E[C_i]$ :  $= \frac{p \cdot k_i(k_i - 1)}{k_i(k_i - 1)} = p = \frac{\bar{k}}{n-1} \approx \frac{\bar{k}}{n}$

Clustering coefficient of a random graph is small.

If we generate bigger and bigger graphs with fixed avg. degree  $k$  (that is we set  $p = k \cdot 1/n$ ), then  $C$  decreases with the graph size  $n$ .

# Network Properties of $G_{np}$

**Degree distribution:**

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

**Clustering coefficient:**  $C = p = \bar{k}/n$

**Path length:**

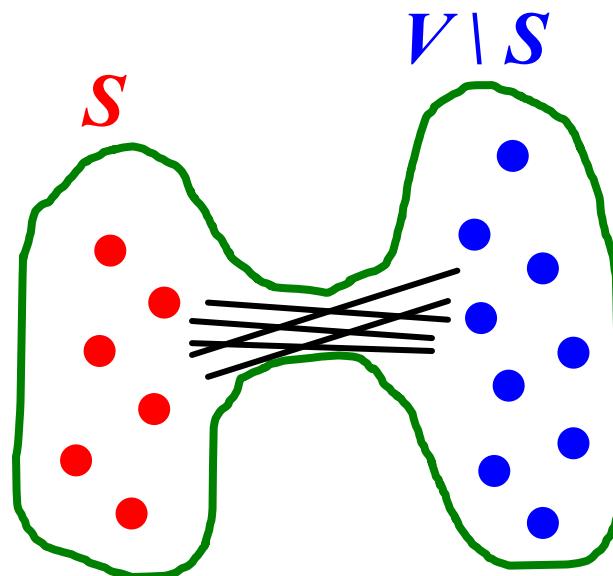
*next!*

**Connectivity:**

# Def: Expansion

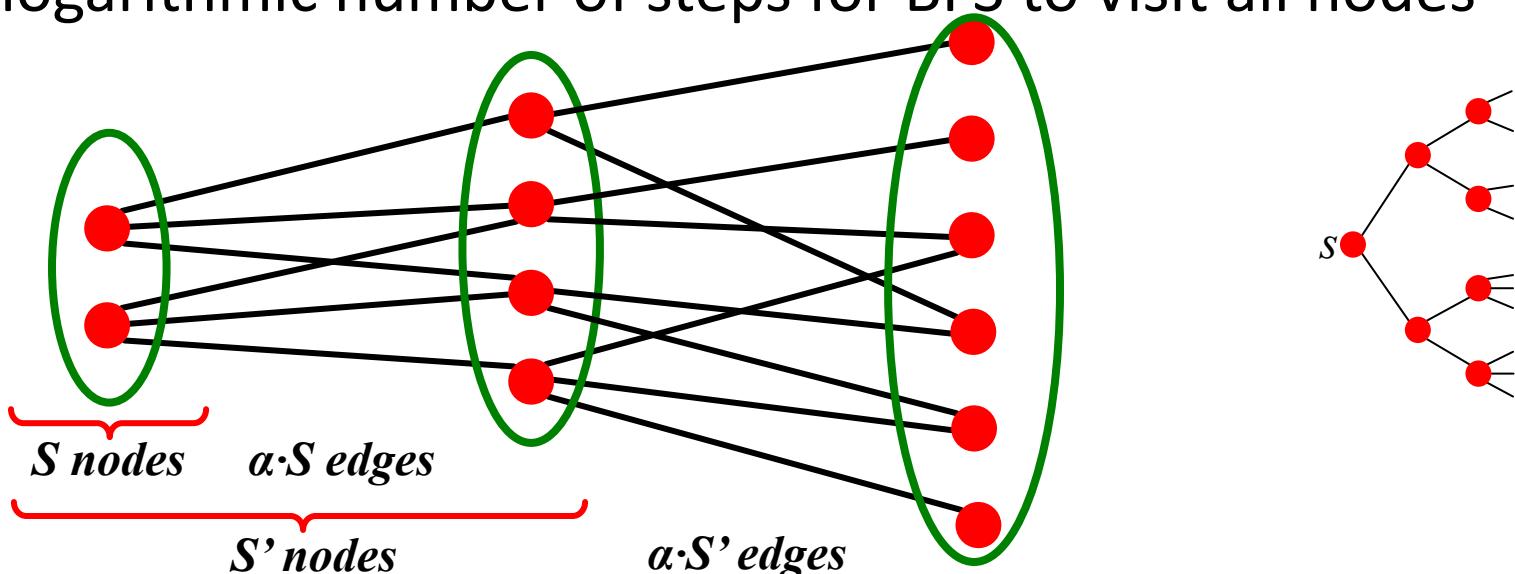
- Graph  $G(V, E)$  has **expansion  $\alpha$** : if  $\forall S \subseteq V$ :  
# of edges leaving  $S \geq \alpha \cdot \min(|S|, |V \setminus S|)$
- **Or equivalently:**

$$\alpha = \min_{S \subseteq V} \frac{\#\text{edges leaving } S}{\min(|S|, |V \setminus S|)}$$



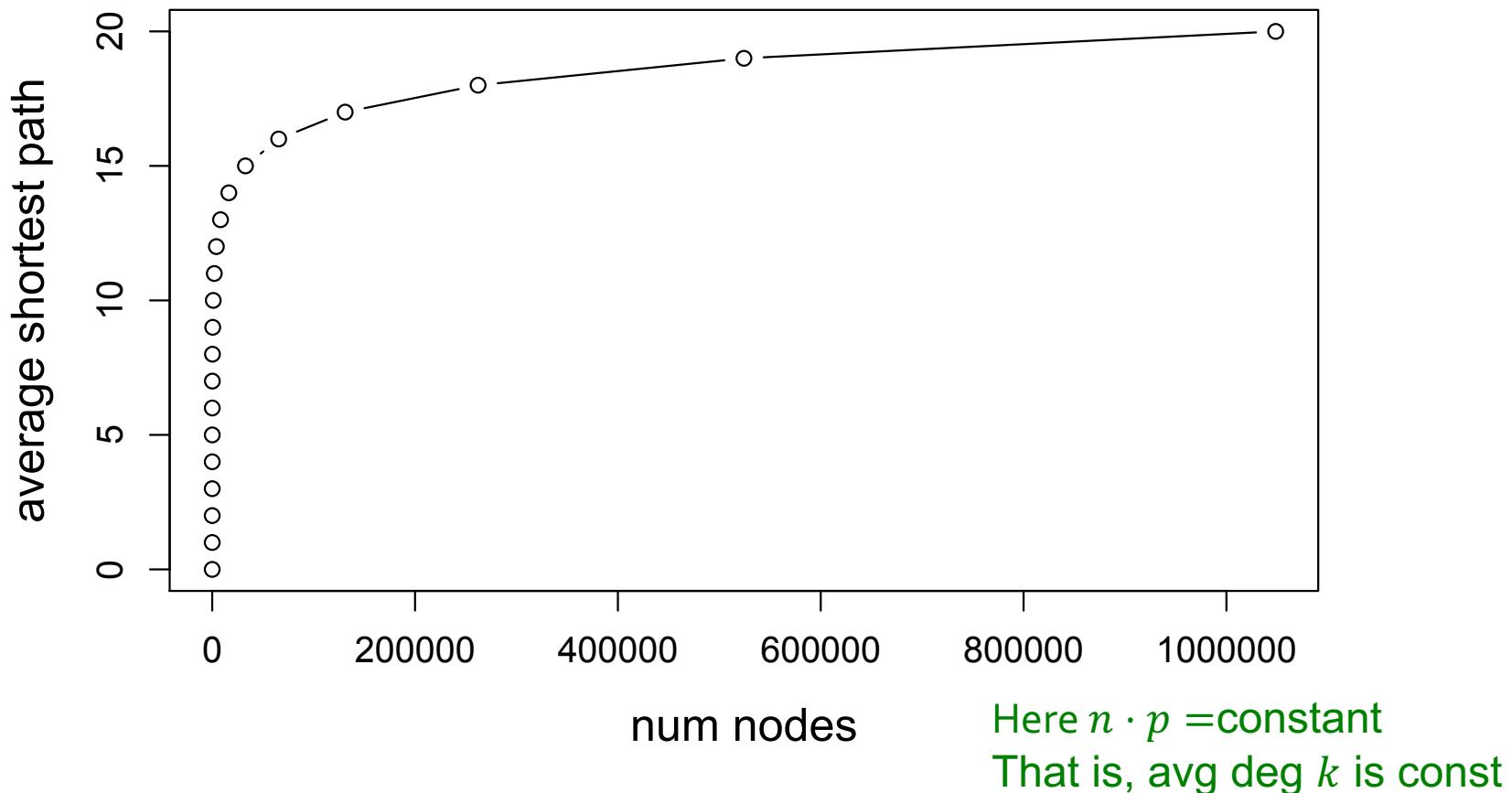
# Expansion: Random Graphs

- **Fact:** In a graph on  $n$  nodes with expansion  $\alpha$ , for all pairs of nodes, there is a path of length  $O((\log n)/\alpha)$ .
- **Random graph  $G_{np}$ :**  
For  $\log n > np > c$ ,  $\text{diam}(G_{np}) = O(\log n / \log(np))$ 
  - Random graphs have good expansion so it takes a logarithmic number of steps for BFS to visit all nodes



# Erdös-Renyi avg. shortest path

Erdös-Renyi Random Graph can grow very large but nodes will be just a few hops apart



# Network Properties of $G_{np}$

**Degree distribution:**

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

**Path length:**

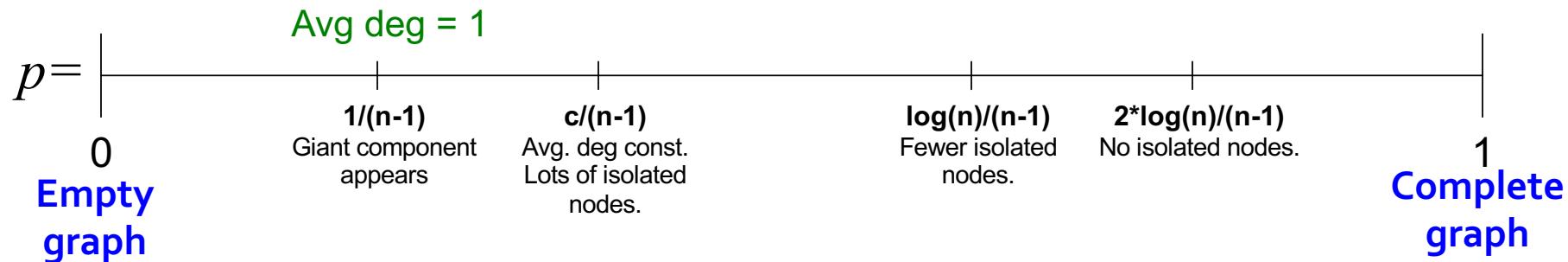
$$O(\log n)$$

**Clustering coefficient:**  $C = p = \bar{k} / n$

**Connected components:** *next!*

# “Evolution” of a Random Graph

- Graph structure of  $G_{np}$  as  $p$  changes:

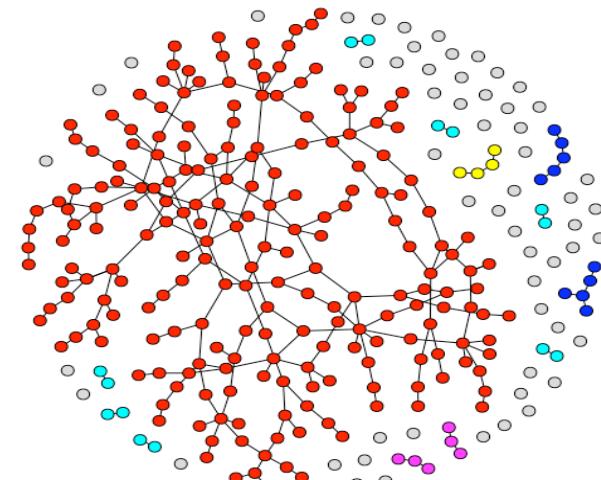
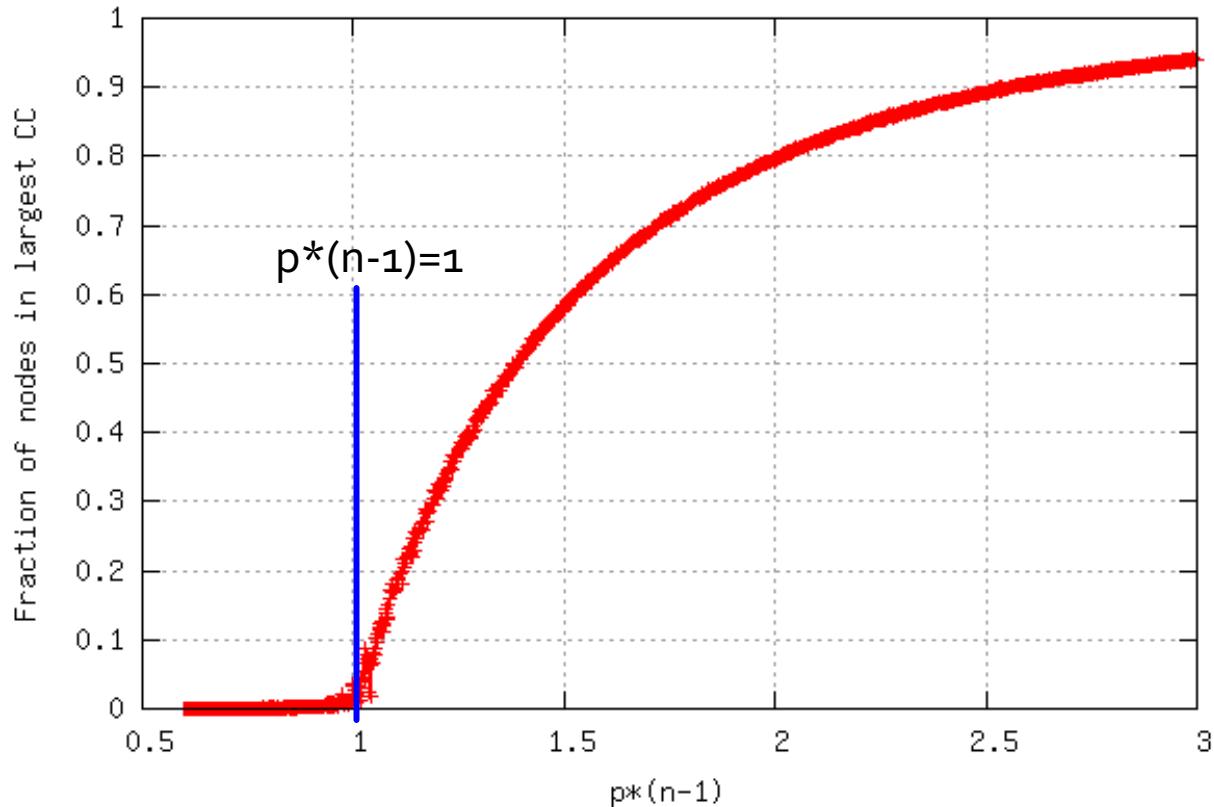


- Emergence of a giant component:

avg. degree  $k=2E/n$  or  $p=k/(n-1)$

- $k=1-\varepsilon$ : all components are of size  $\Omega(\log n)$
- $k=1+\varepsilon$ : 1 component of size  $\Omega(n)$ , others have size  $\Omega(\log n)$ 
  - Each node has at least one edge in expectation

# $G_{np}$ Simulation Experiment

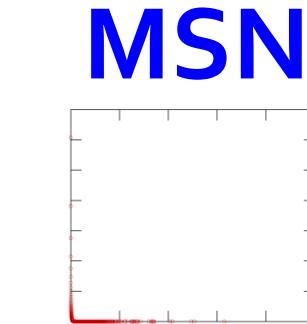


Fraction of nodes in the largest component

- $G_{np}, n=100,000, k=p(n-1) = 0.5 \dots 3$

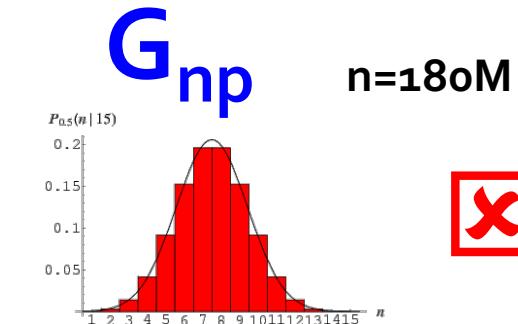
# Back to MSN vs. $G_{np}$

Degree distribution:



Avg. path length:

6.6



Avg. clustering coef.: 0.11

$\bar{k} / n$

$C \approx 8 \cdot 10^{-8}$

Largest Conn. Comp.: 99%

GCC exists  
when  $\bar{k} > 1$ .

$\bar{k} \approx 14$ .

# Real Networks vs. $G_{np}$

- **Are real networks like random graphs?**
  - Giant connected component: 😊
  - Average path length: 😊
  - Clustering Coefficient: 😕
  - Degree Distribution: 😕
- **Problems with the random networks model:**
  - Degree distribution differs from that of real networks
  - Giant component in most real networks does NOT emerge through a phase transition
  - No local structure – clustering coefficient is too low
- **Most important: Are real networks random?**
  - The answer is simply: **NO!**

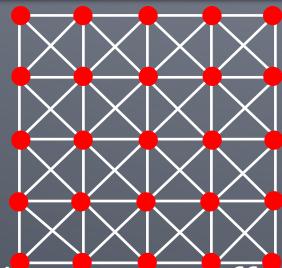
# Real Networks vs. $G_{np}$

- If  $G_{np}$  is wrong, why did we spend time on it?
  - It is the reference model for the rest of the class
  - It will help us calculate many quantities, that can then be compared to the real data
  - It will help us understand to what degree a particular property is the result of some random process

So, while  $G_{np}$  is WRONG, it will turn out to be extremely USEFUL!

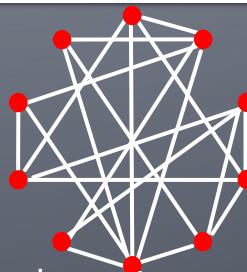
# The Small-World Model

Can we have high clustering while also having short paths?



High clustering coefficient,  
High diameter

Vs.



Low clustering coefficient  
Low diameter

# Clustering Implies Edge Locality

- MSN network has 7 orders of magnitude larger clustering than the corresponding  $G_{np}$ !
- Other examples:

Actor Collaborations (IMDB):  $N = 225,226$  nodes, avg. degree  $\bar{k} = 61$

Electrical power grid:  $N = 4,941$  nodes,  $\bar{k} = 2.67$

Network of neurons:  $N = 282$  nodes,  $\bar{k} = 14$

Network	$h_{\text{actual}}$	$h_{\text{random}}$	$C_{\text{actual}}$	$C_{\text{random}}$
Film actors	3.65	2.99	0.79	0.00027
Power Grid	18.70	12.40	0.080	0.005
C. elegans	2.65	2.25	0.28	0.05

$h$  ... Average shortest path length

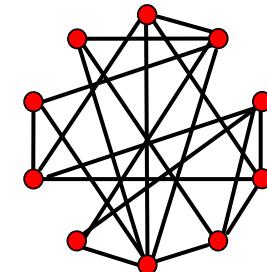
$C$  ... Average clustering coefficient

“actual” ... real network

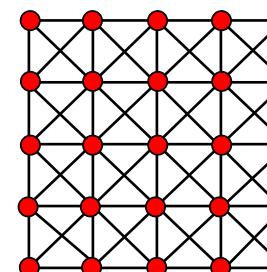
“random” ... random graph with same avg. degree

# The “Controversy”

- **Consequence of expansion:**
  - **Short paths:  $O(\log n)$** 
    - This is the smallest diameter we can get if we keep the degree constant.
  - But clustering is low!
- **But networks have “local” structure:**
  - **Triadic closure:**  
Friend of a friend is my friend
  - High clustering but diameter is also high
- **How can we have both?**



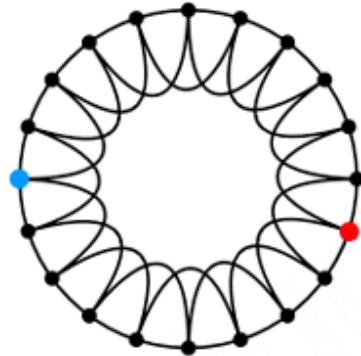
Low diameter  
Low clustering coefficient



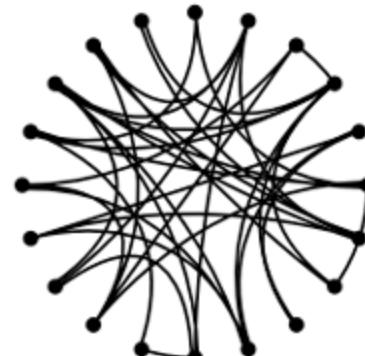
High clustering coefficient  
High diameter

# Small-World: How?

- Could a network with high clustering also be small world (have  $\log n$  diameter)?
  - How can we at the same time have high clustering and small diameter?



High clustering  
High diameter



Low clustering  
Low diameter

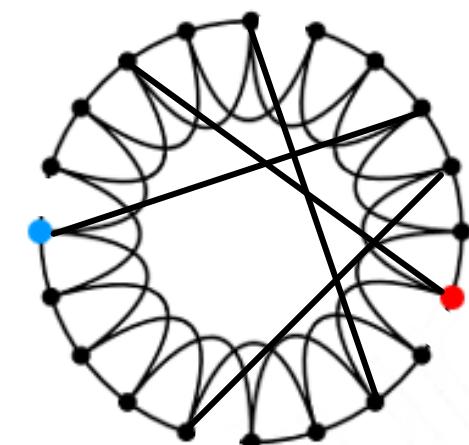
- Clustering implies edge “locality”
- Randomness enables “shortcuts”

# Solution: The Small-World Model

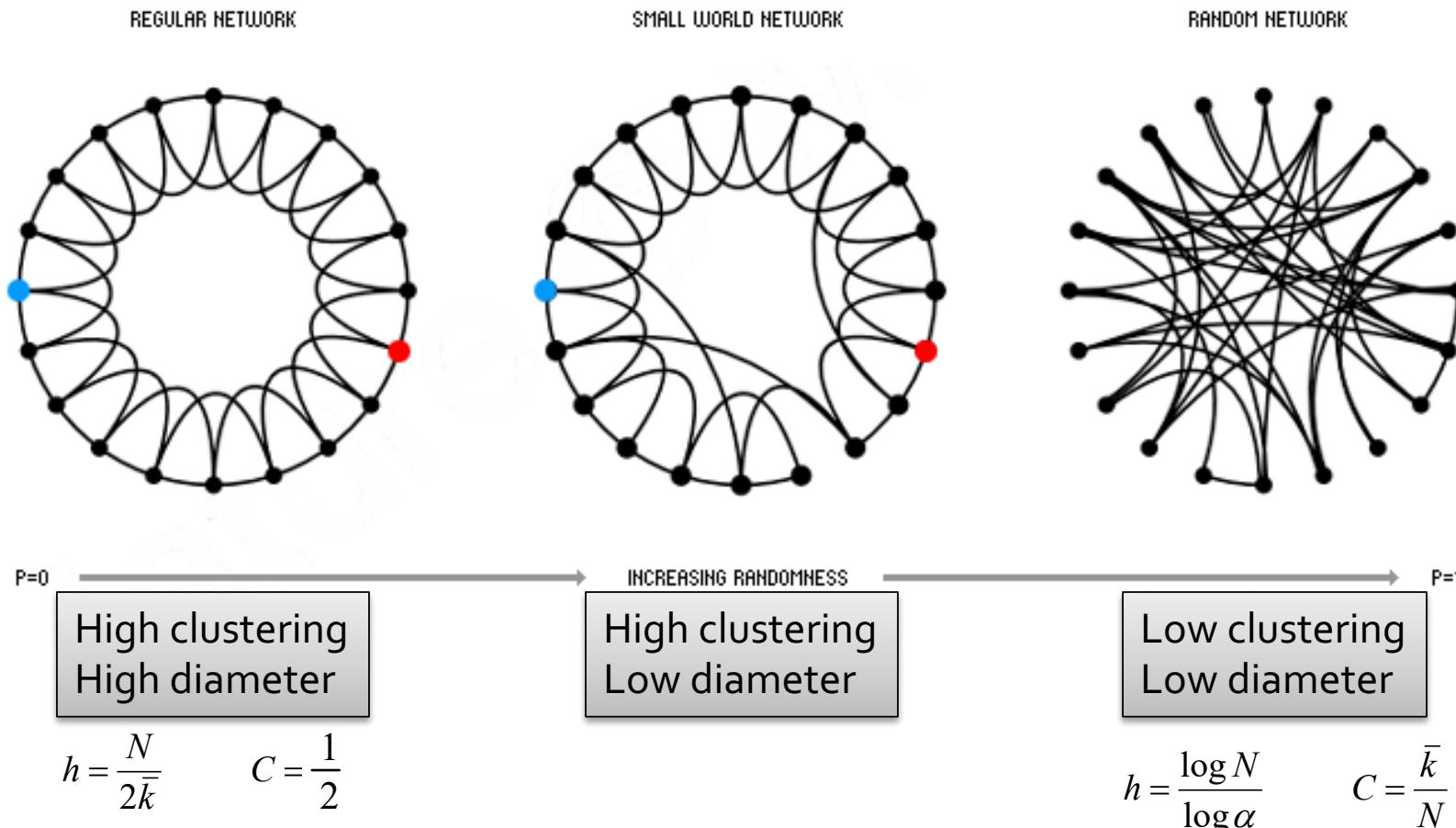
## Small-World Model [Watts-Strogatz '98]

Two components to the model:

- **(1) Start with a low-dimensional regular lattice**
  - (In our case we are using a ring as a lattice)
  - Has high clustering coefficient
- **(2) Rewire: Introduce randomness (“shortcuts”)**
  - Add/remove edges to create shortcuts to join remote parts of the lattice
  - For each edge, with prob.  $p$ , move the other endpoint to a random node

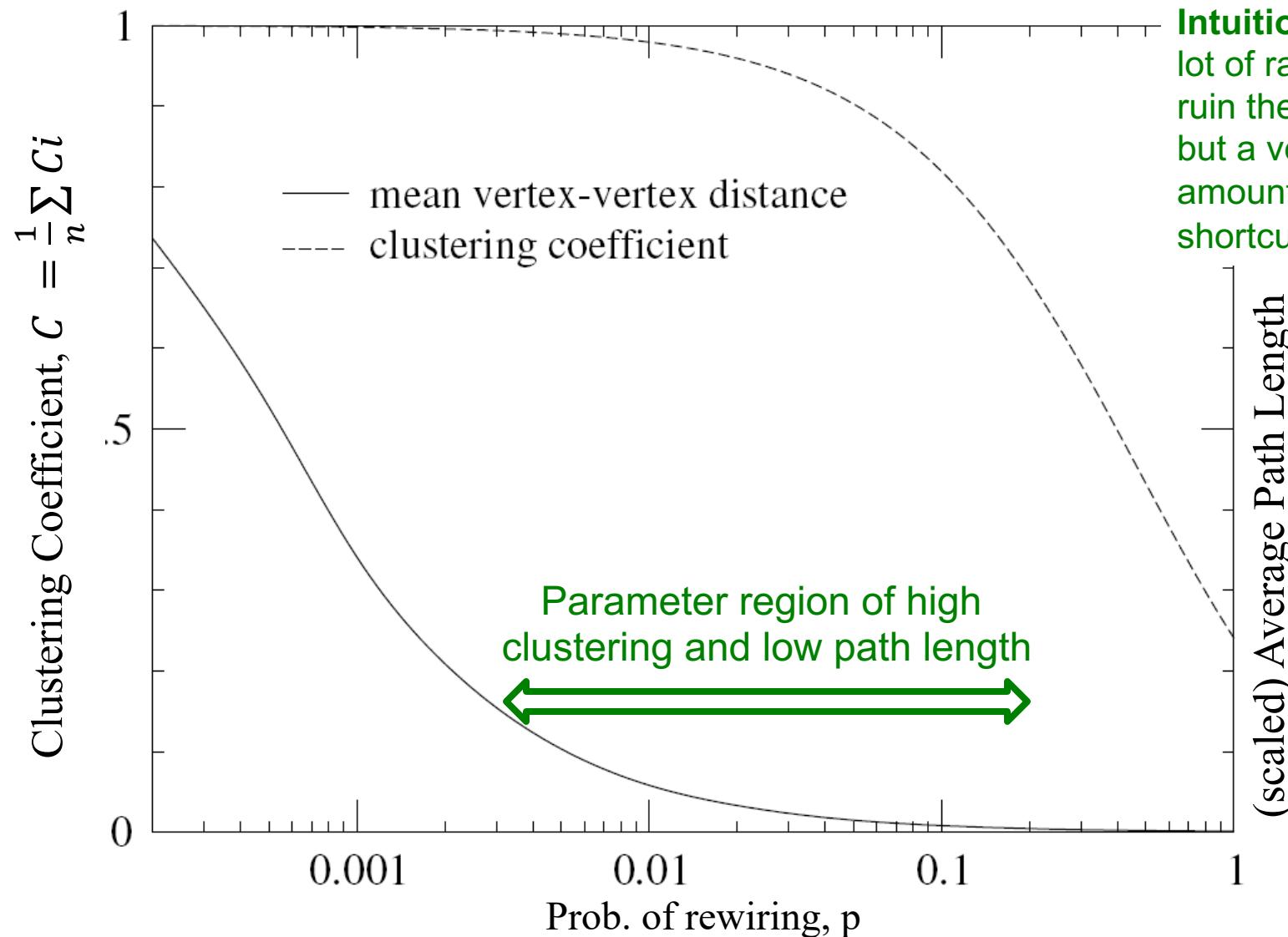


# The Small-World Model



Rewiring allows us to “interpolate” between a regular lattice and a random graph

# The Small-World Model



**Intuition:** It takes a lot of randomness to ruin the clustering, but a very small amount to create shortcuts.

# Small-World: Summary

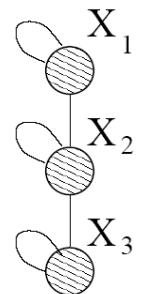
- Could a network with high clustering be at the same time a small world?
  - Yes! You don't need more than a few random links
- The Watts Strogatz Model:
  - Provides insight on the interplay between clustering and the small-world
  - Captures the structure of many realistic networks
  - Accounts for the high clustering of real networks
  - Does not lead to the correct degree distribution

# Kronecker Graph Model

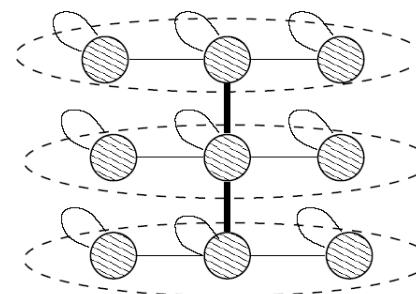
Generating large realistic graphs

# Idea: Recursive Graph Generation

- How can we think of network structure recursively? Intuition: Self-similarity
  - Object is similar to a part of itself: the whole has the same shape as one or more of the parts
- Mimic recursive graph/community growth:



Initial graph



Recursive expansion

- Kronecker product is a way of generating self-similar matrices

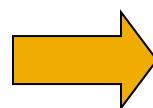
# Kronecker Graph

- Kronecker graphs:
- A recursive model of network structure

1	1	0
1	1	1
0	1	1

$K_1$

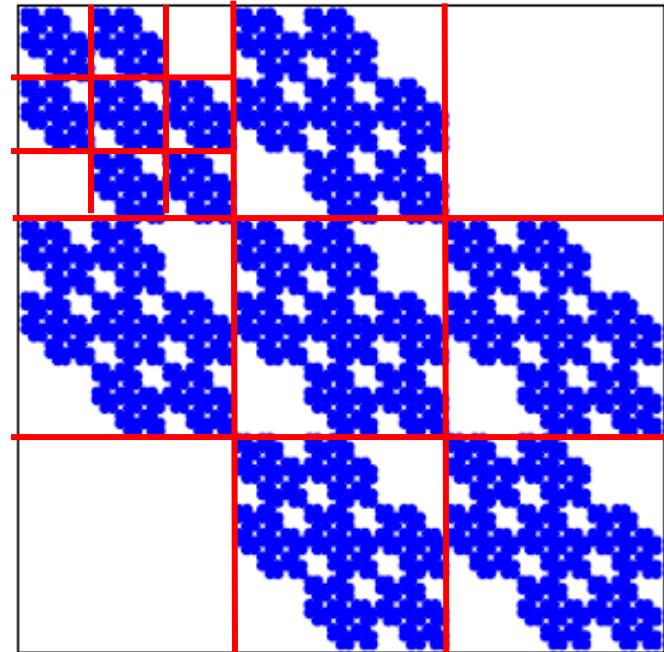
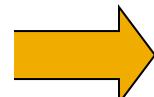
$3 \times 3$



$K_1$	$K_1$	0
$K_1$	$K_1$	$K_1$
0	$K_1$	$K_1$

$K_2 = K_1 \otimes K_1$

$9 \times 9$



$81 \times 81$  adjacency matrix

# Kronecker Product: Definition

- Kronecker product of matrices  $A$  and  $B$  is given by

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \doteq \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \dots & a_{n,m}\mathbf{B} \end{pmatrix}_{N^*K \times M^*L}$$

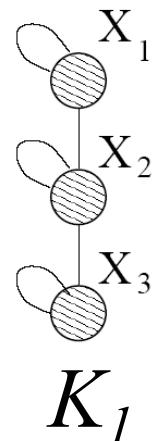
- Define a Kronecker product of two graphs as a Kronecker product of their adjacency matrices

# Kronecker Graphs

- Kronecker graph is obtained by growing sequence of graphs by iterating the Kronecker product over the initiator matrix  $K_1$ :

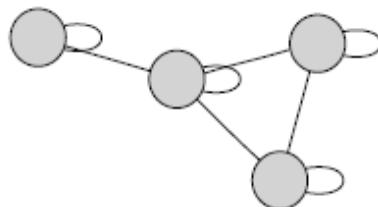
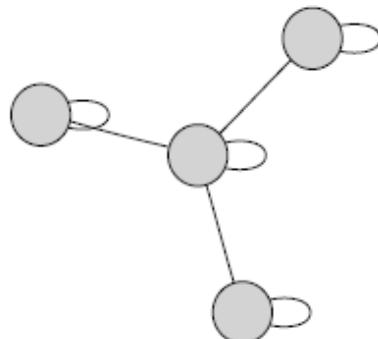
$$K_1^{[m]} = K_m = \underbrace{K_1 \otimes K_1 \otimes \dots K_1}_{m \text{ times}} = K_{m-1} \otimes K_1$$

1	1	0
1	1	1
0	1	1



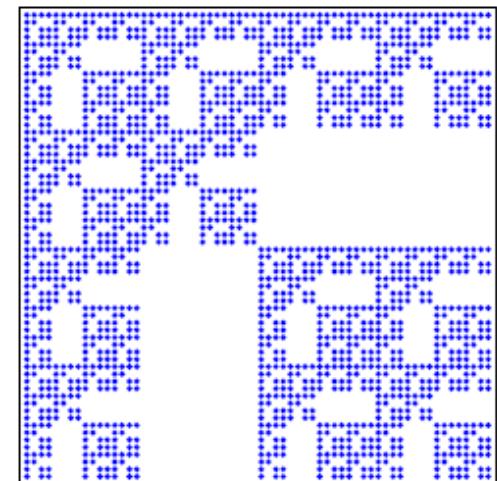
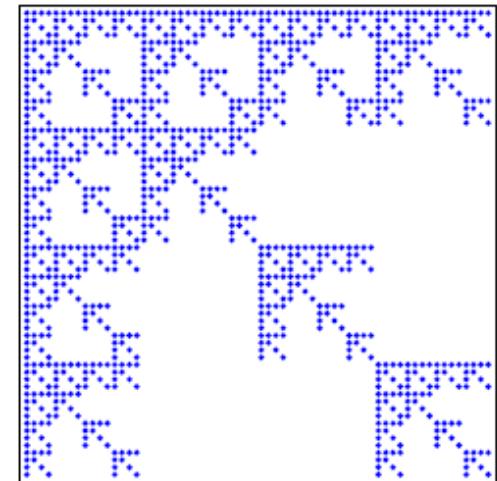
- Note: One can easily use multiple initiator matrices ( $K_1', K_1'', K_1'''$ ) (even of different sizes)

# Kronecker Initiator Matrices

Initiator  $K_1$ 

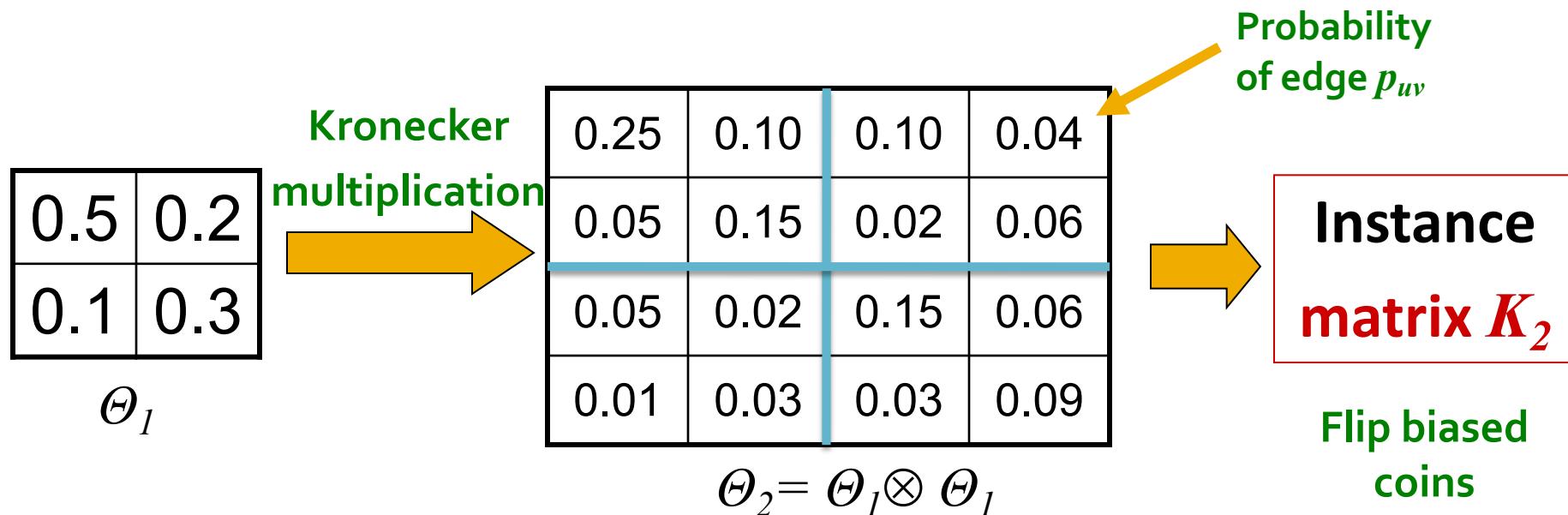
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1

1	1	1	1
1	1	0	0
1	0	1	1
1	0	1	1

 $K_1$  adjacency matrix $K_3$  adjacency matrix

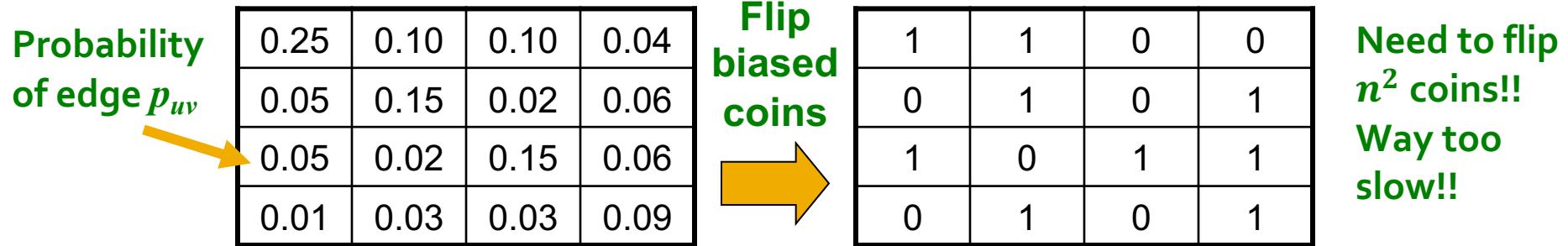
# Stochastic Kronecker Graphs

- Create  $N_1 \times N_1$  **probability matrix**  $\Theta_1$
- Compute the  $k^{\text{th}}$  Kronecker power  $\Theta_k$
- For each entry  $p_{uv}$  of  $\Theta_k$  include an edge  $(u, v)$  in  $K_k$  with probability  $p_{uv}$



# Generation of Kronecker Graphs

- How do we generate an instance of a (Directed) stochastic Kronecker graph?



- Is there a faster way? YES!
- Idea: Exploit the recursive structure of Kronecker graphs
  - “Drop” edges onto the graph one by one

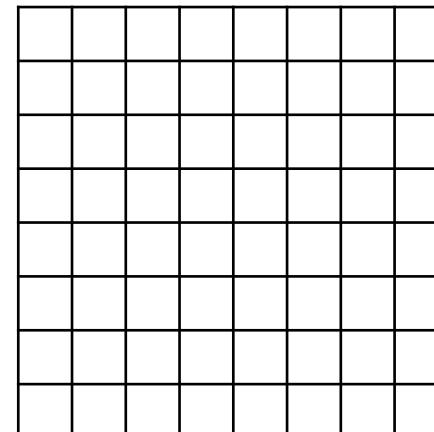
# Generation of Kronecker Graphs

- A faster way to generate Kronecker graphs

$$\Theta = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a \cdot a & a \cdot b & b \cdot a & b \cdot b \\ \hline v_2 & a \cdot c & a \cdot d & b \cdot c & b \cdot d \\ \hline v_3 & c \cdot a & c \cdot b & d \cdot a & d \cdot b \\ \hline v_4 & c \cdot c & c \cdot d & d \cdot c & d \cdot d \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a & b & a & b \\ \hline v_2 & a & c & d & c \\ \hline v_3 & c & b & a & b \\ \hline v_4 & c & d & c & d \\ \hline \end{array}$$

$\Theta \otimes \Theta$

- How to “drop” an edge into a graph  $G$  on  $n = 2^m$  nodes



Adjacency matrix  $G$

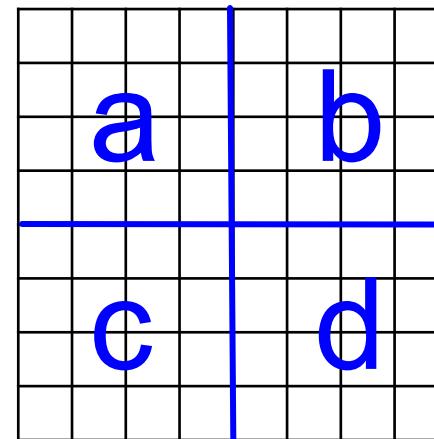
# Generation of Kronecker Graphs

- A faster way to generate Kronecker graphs

$$\Theta = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a \cdot a & a \cdot b & b \cdot a & b \cdot b \\ \hline v_2 & a \cdot c & a \cdot d & b \cdot c & b \cdot d \\ \hline v_3 & c \cdot a & c \cdot b & d \cdot a & d \cdot b \\ \hline v_4 & c \cdot c & c \cdot d & d \cdot c & d \cdot d \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a & b & a & b \\ \hline v_2 & a & c & d & c \\ \hline v_3 & c & b & a & b \\ \hline v_4 & c & d & c & d \\ \hline \end{array}$$

$\Theta \otimes \Theta$

- How to “drop” an edge into a graph  $G$  on  $n = 2^m$  nodes



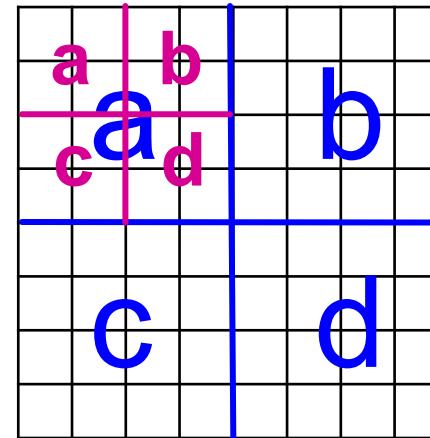
# Generation of Kronecker Graphs

- A faster way to generate Kronecker graphs

$$\Theta = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a \cdot a & a \cdot b & b \cdot a & b \cdot b \\ \hline v_2 & a \cdot c & a \cdot d & b \cdot c & b \cdot d \\ \hline v_3 & c \cdot a & c \cdot b & d \cdot a & d \cdot b \\ \hline v_4 & c \cdot c & c \cdot d & d \cdot c & d \cdot d \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & a & b & a & b \\ \hline v_2 & a & c & d & c \\ \hline v_3 & c & b & a & b \\ \hline v_4 & c & d & c & d \\ \hline \end{array}$$

$\Theta \otimes \Theta$

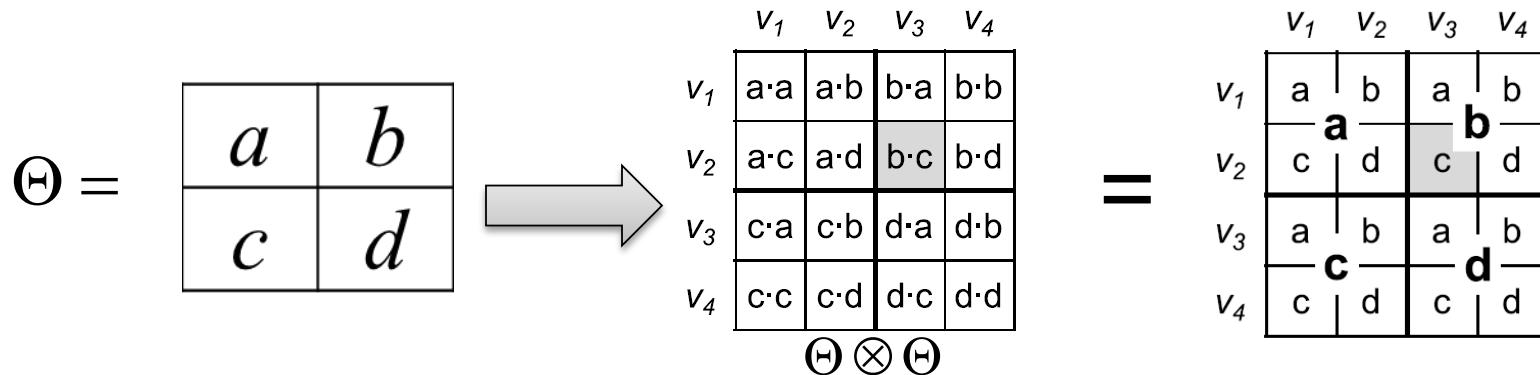
- How to “drop” an edge into a graph  $G$  on  $n = 2^m$  nodes



Adjacency matrix  $G$

# Generation of Kronecker Graphs

- A faster way to generate Kronecker graphs



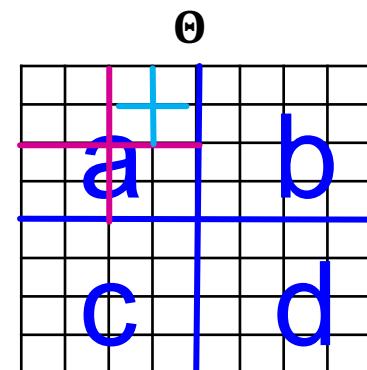
- How to “drop” an edge into a graph  $G$  on  $n = 2^m$  nodes:
  - We may get a few edges colliding. We simply reinsert them.

Adjacency matrix  $G$

# Generation of Kronecker Graphs

## Fast Kronecker generator algorithm:

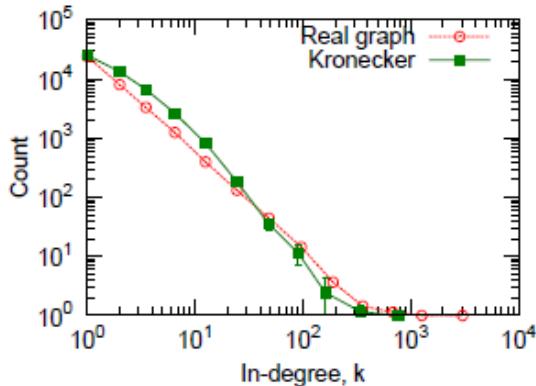
- For generating **directed graphs**
- **Insert 1 edge on graph  $G$  on  $n = 2^m$  nodes:**
  - Create normalized matrix  $L_{uv} = \Theta_{uv}/(\sum_{op} \Theta_{op})$
  - **For  $i = 1 \dots m$** 
    - Start with  $x = 0, y = 0$
    - Pick a row/column  $(u, v)$  with prob.  $L_{uv}$
    - Descend into quadrant  $(u, v)$  at level  $i$  of  $G$ 
      - This means:  $x += u \cdot 2^{m-i}, y += v \cdot 2^{m-i}$
    - Add an edge  $(x, y)$  to  $G$



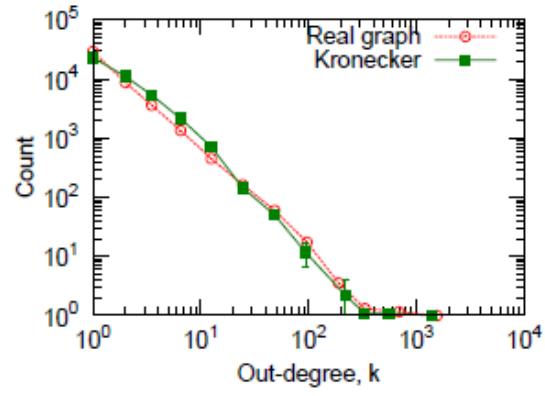
# Estimation: Epinions (n=76k, m=510k)

- Real and Kronecker are very close:

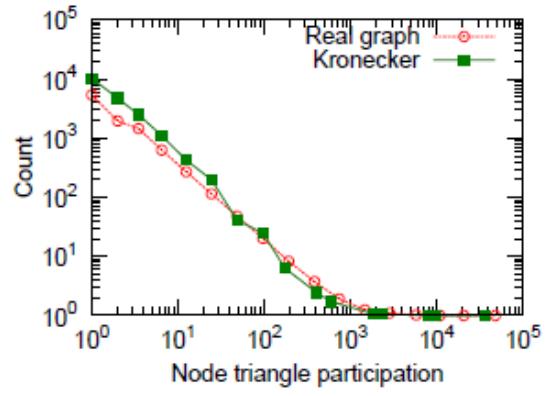
$$\Theta_1 = \begin{bmatrix} 0.99 & 0.54 \\ 0.49 & 0.13 \end{bmatrix}$$



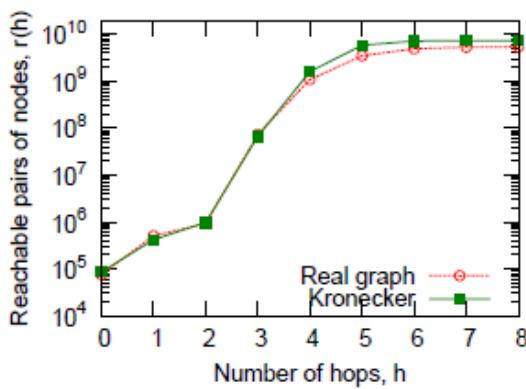
(a) In-Degree



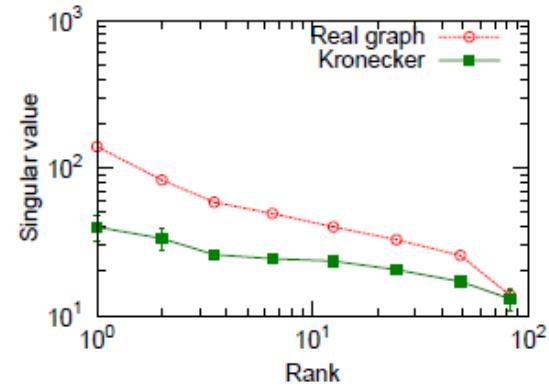
(b) Out-degree



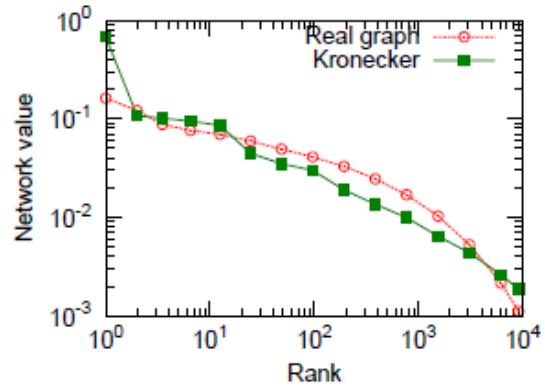
(c) Triangle participation



(d) Hop plot



(e) Scree plot



(f) "Network" value