Efficient Algorithms for Public-Private Social Networks

Flavio Chierichetti
Sapienza University

Alessandro Epasto
Brown University

Ravi Kumar
Google

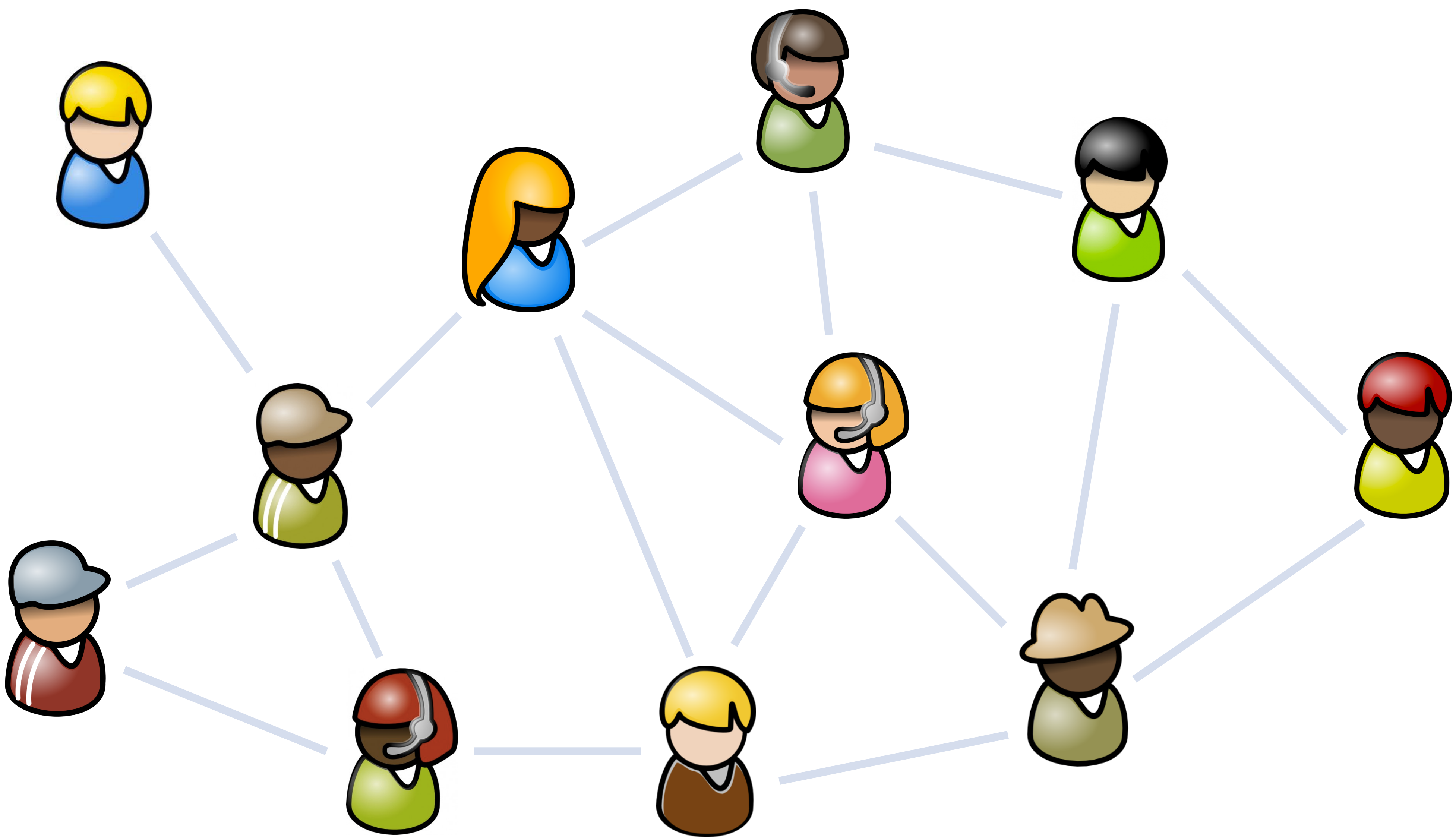
Silvio Lattanzi
Google

Vahab Mirrokni
Google

KDD2015 — Sydney, Australia — August 11, 2015

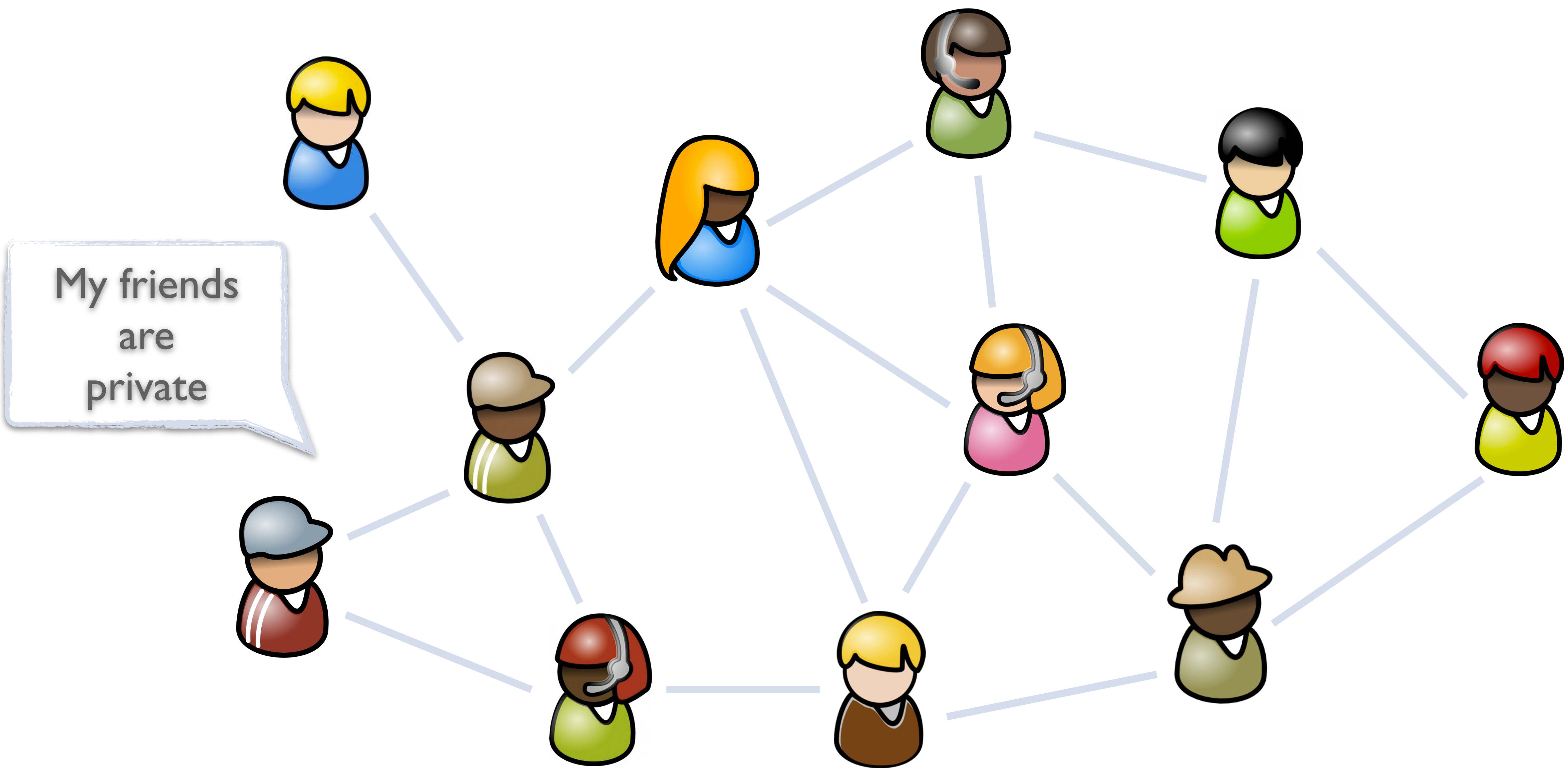
Private-Public networks

Idealized vision



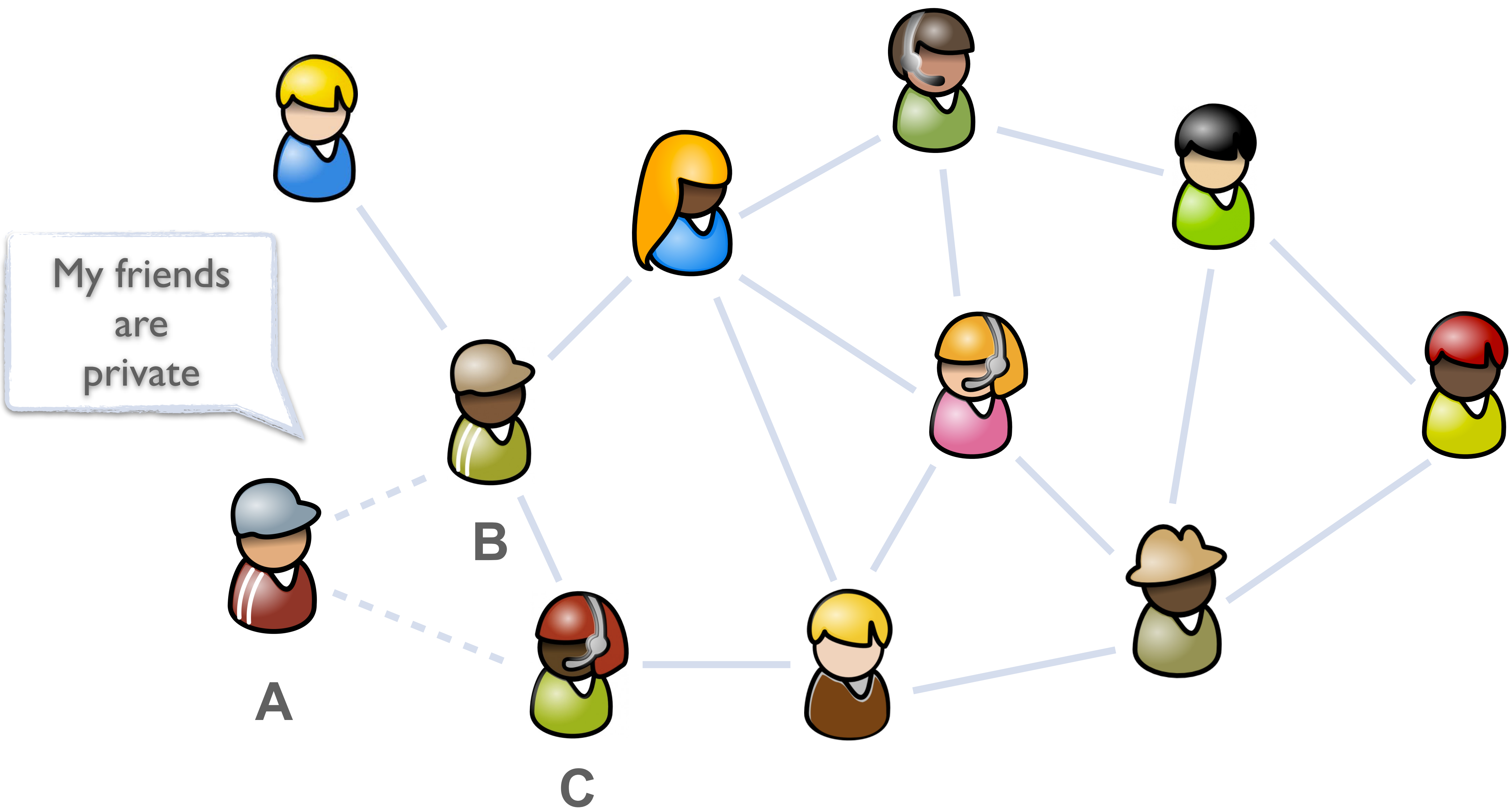
Private-Public networks

Reality



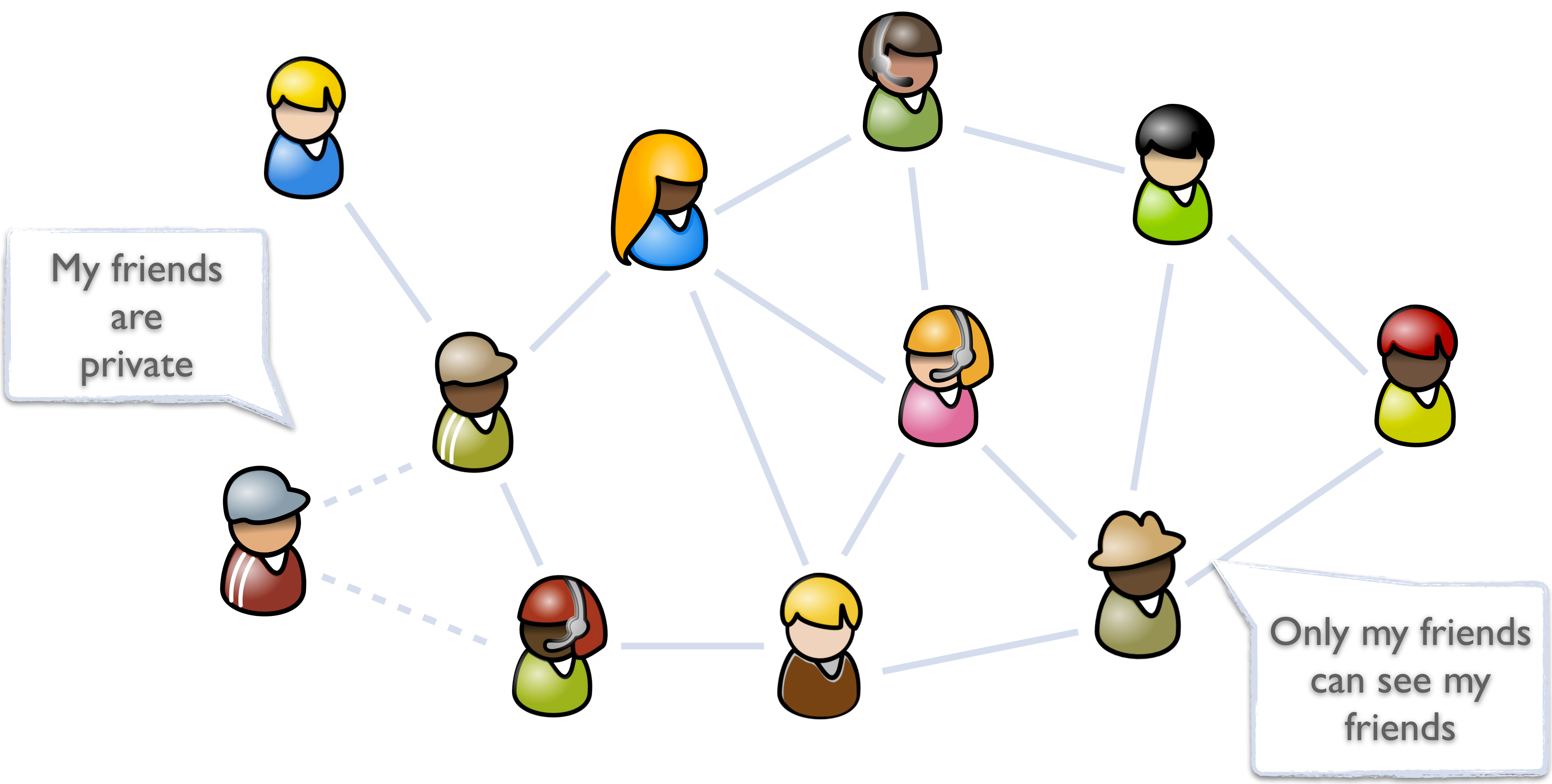
Private-Public networks

Reality



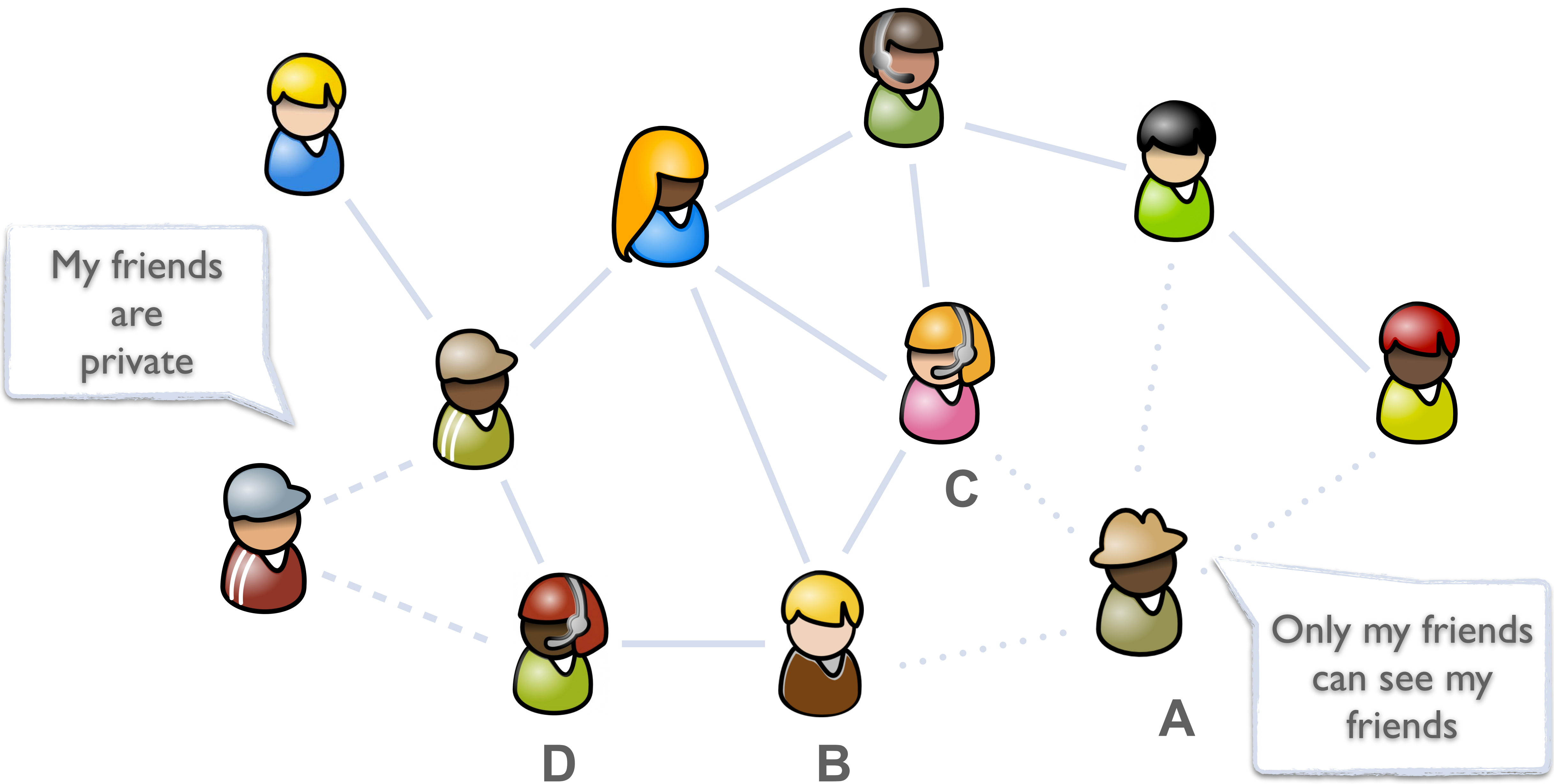
Private-Public networks

Reality



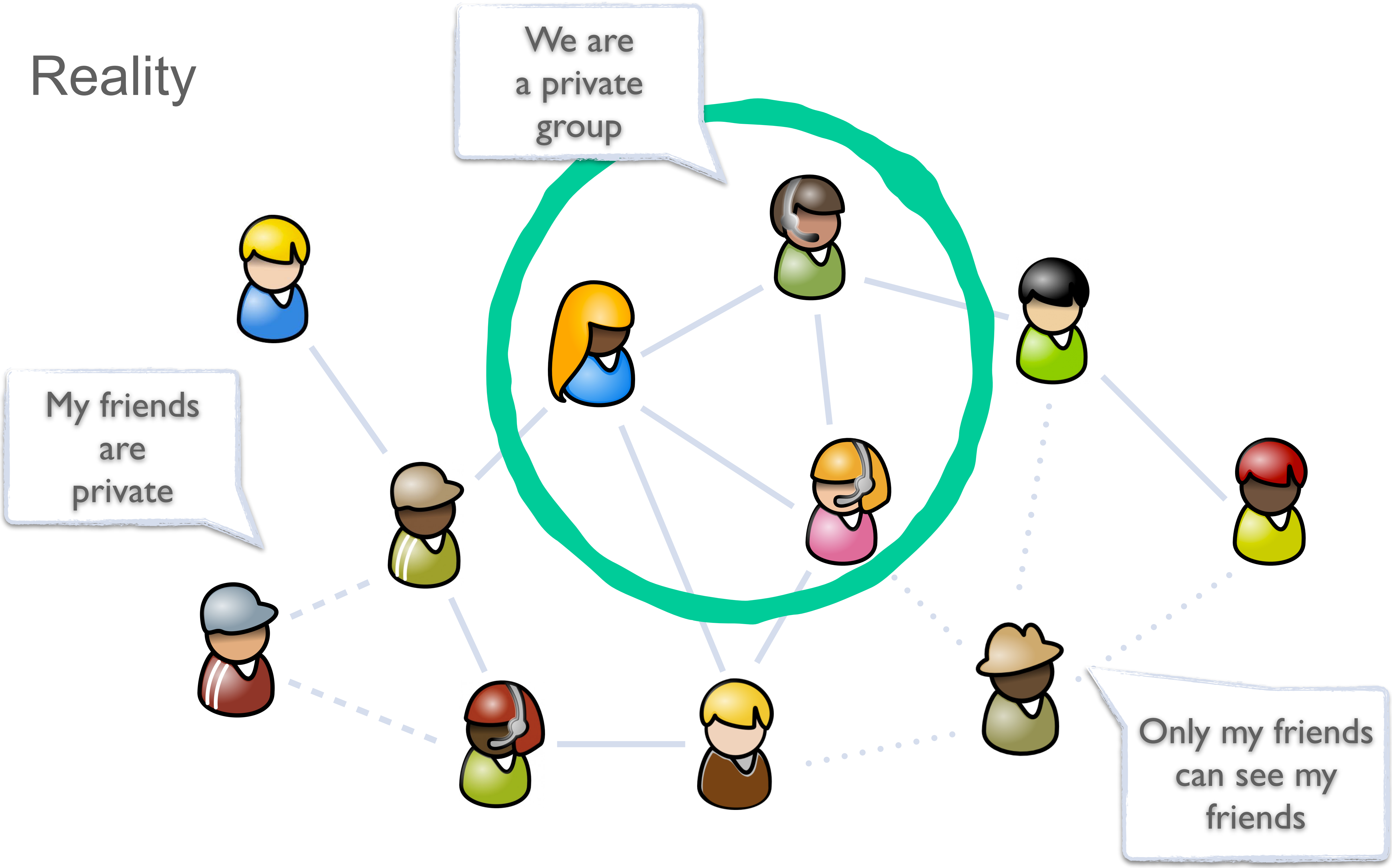
Private-Public networks

Reality



Private-Public networks

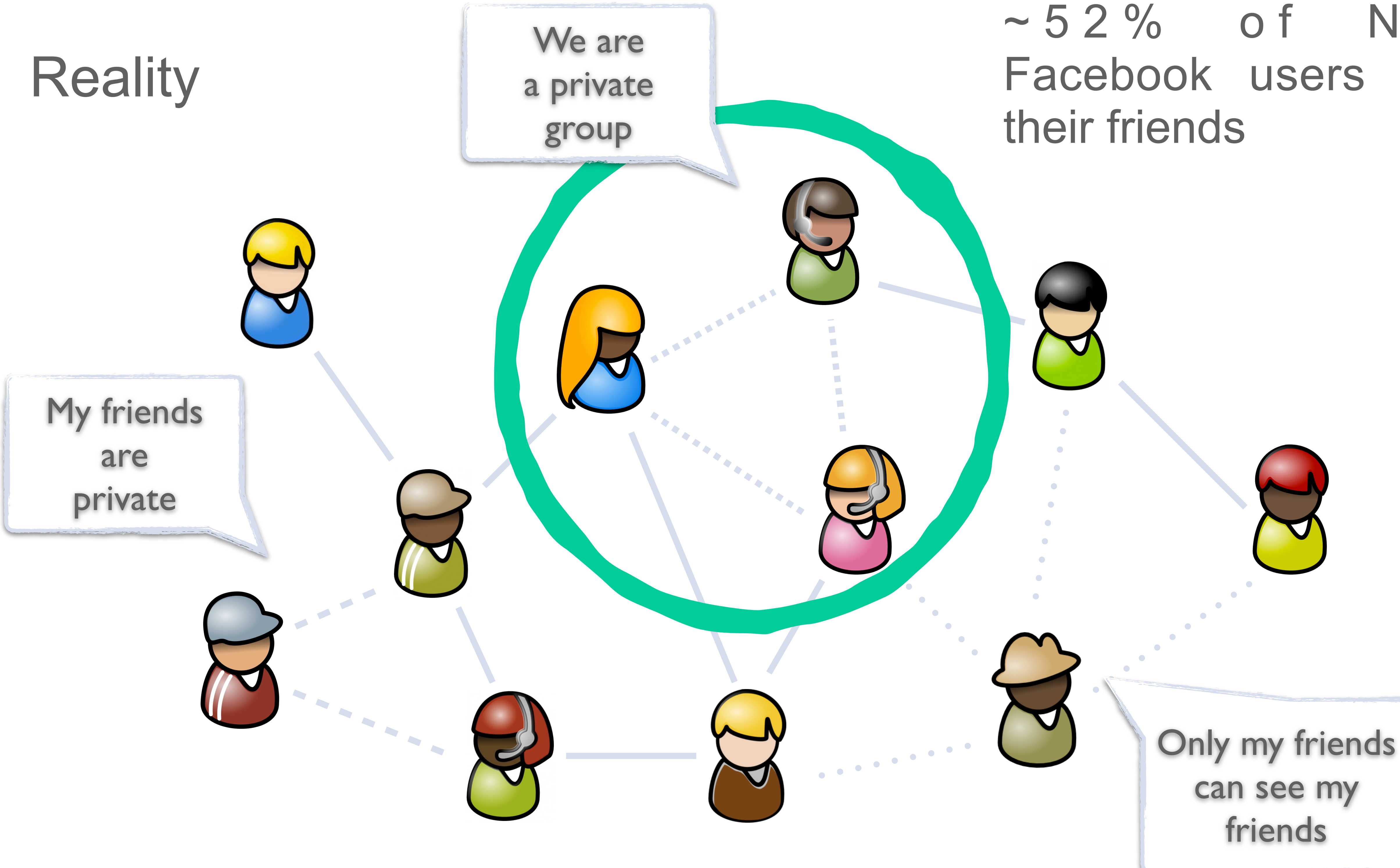
Reality



Private-Public networks

Reality

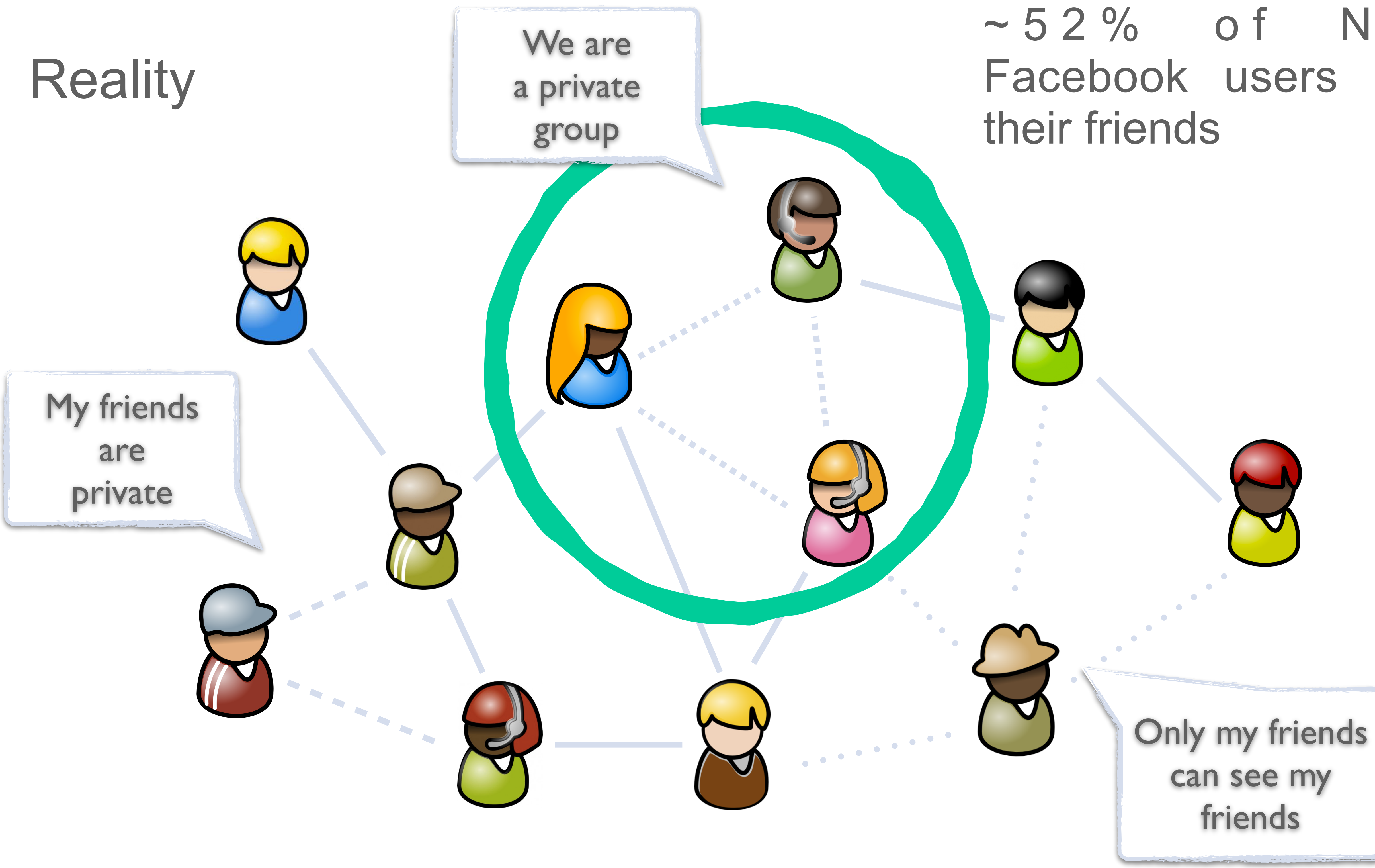
~52% of NYC Facebook users hide their friends



Private-Public networks

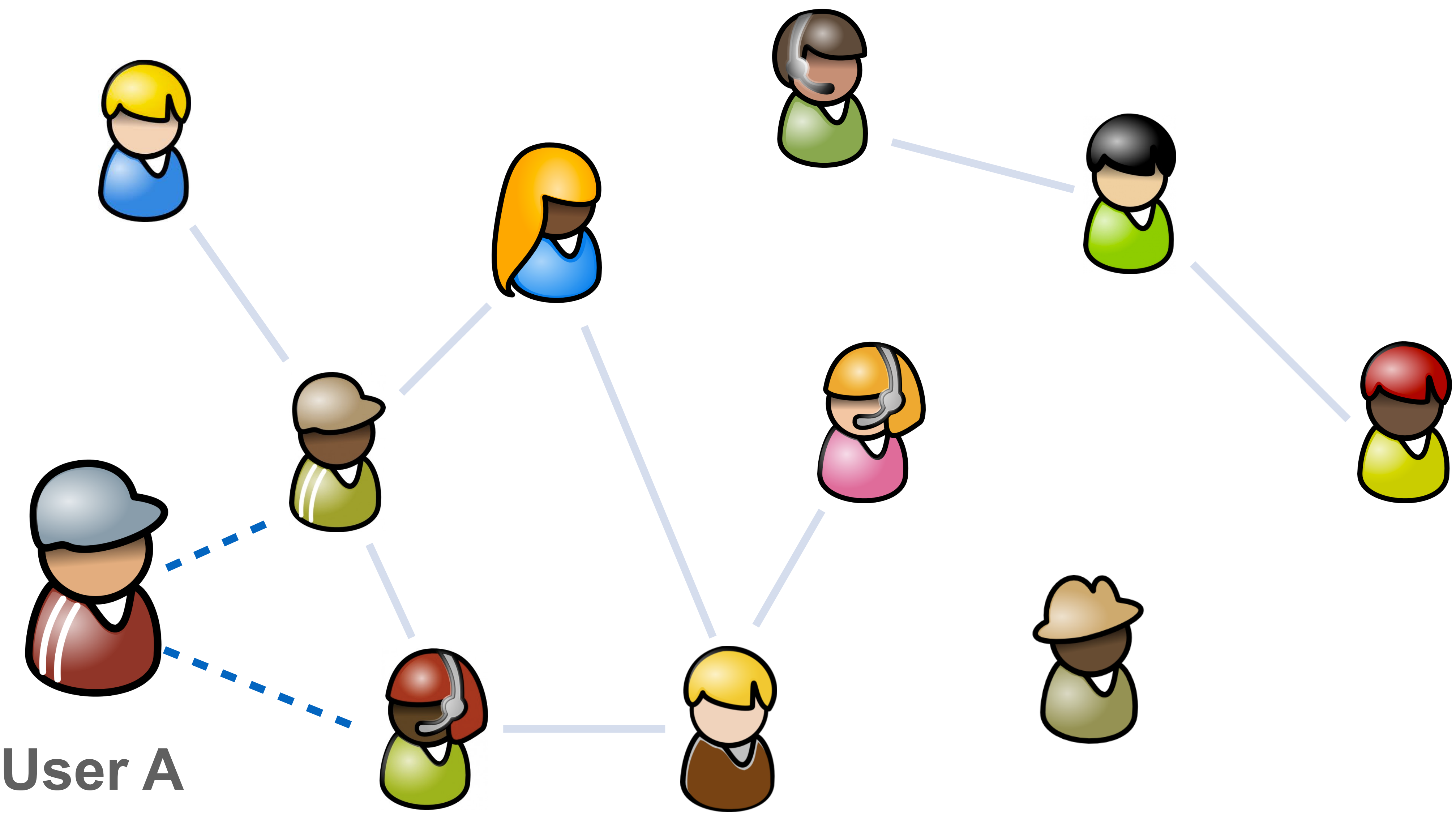
Reality

~52% of NYC Facebook users hide their friends

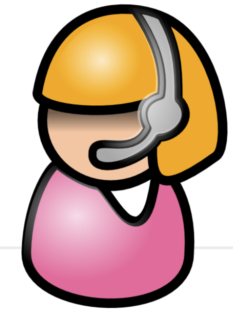


There is **no** such thing as **the Social Network**!

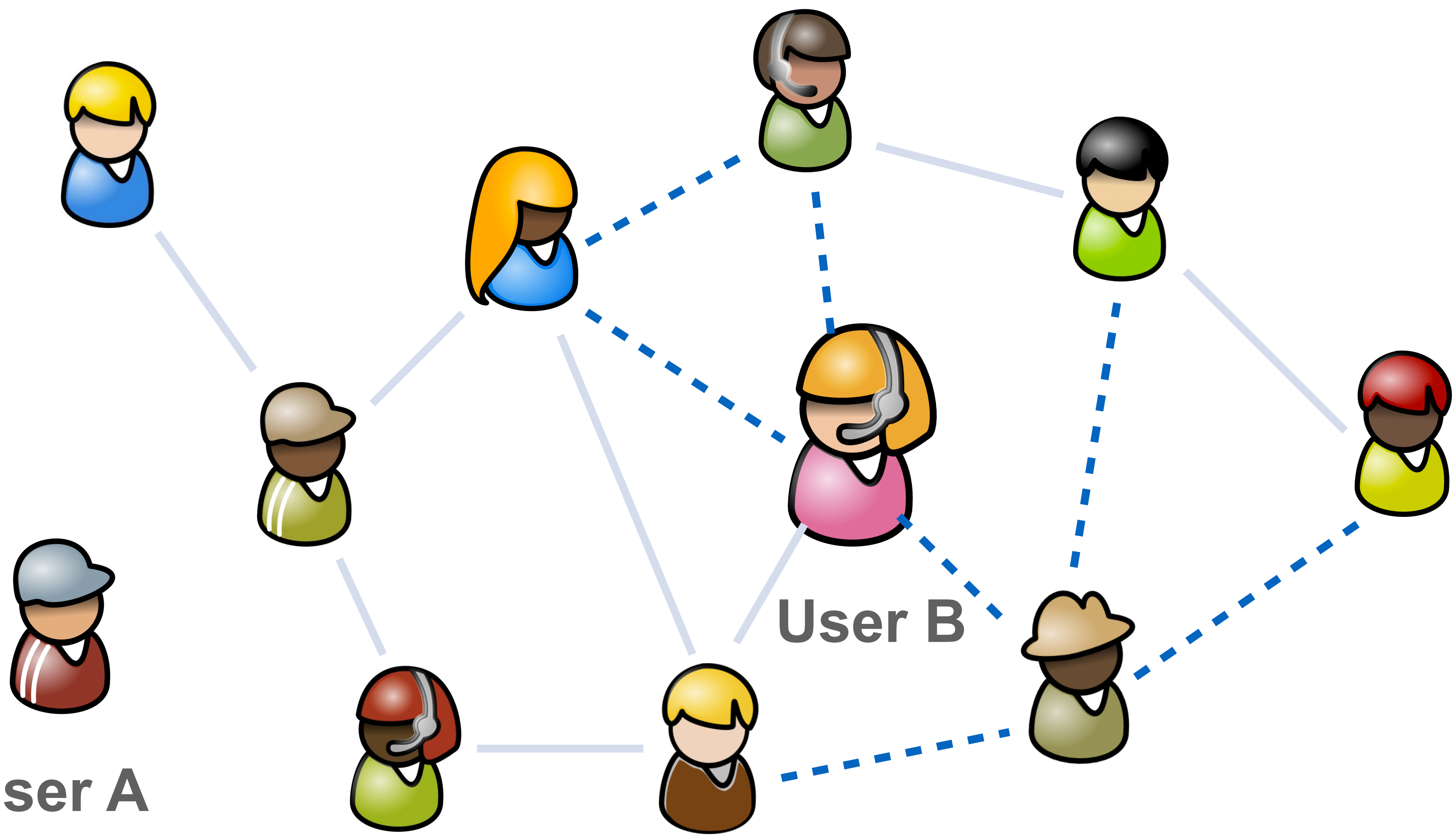
Social network of 



Each user has **his/her own personal** Social Network!

Social network of 

User B



Each user has **his/her <u>own</u> personal** Social Network!

Computational implication

The **algorithms need** to **respect** the privacy of the users.

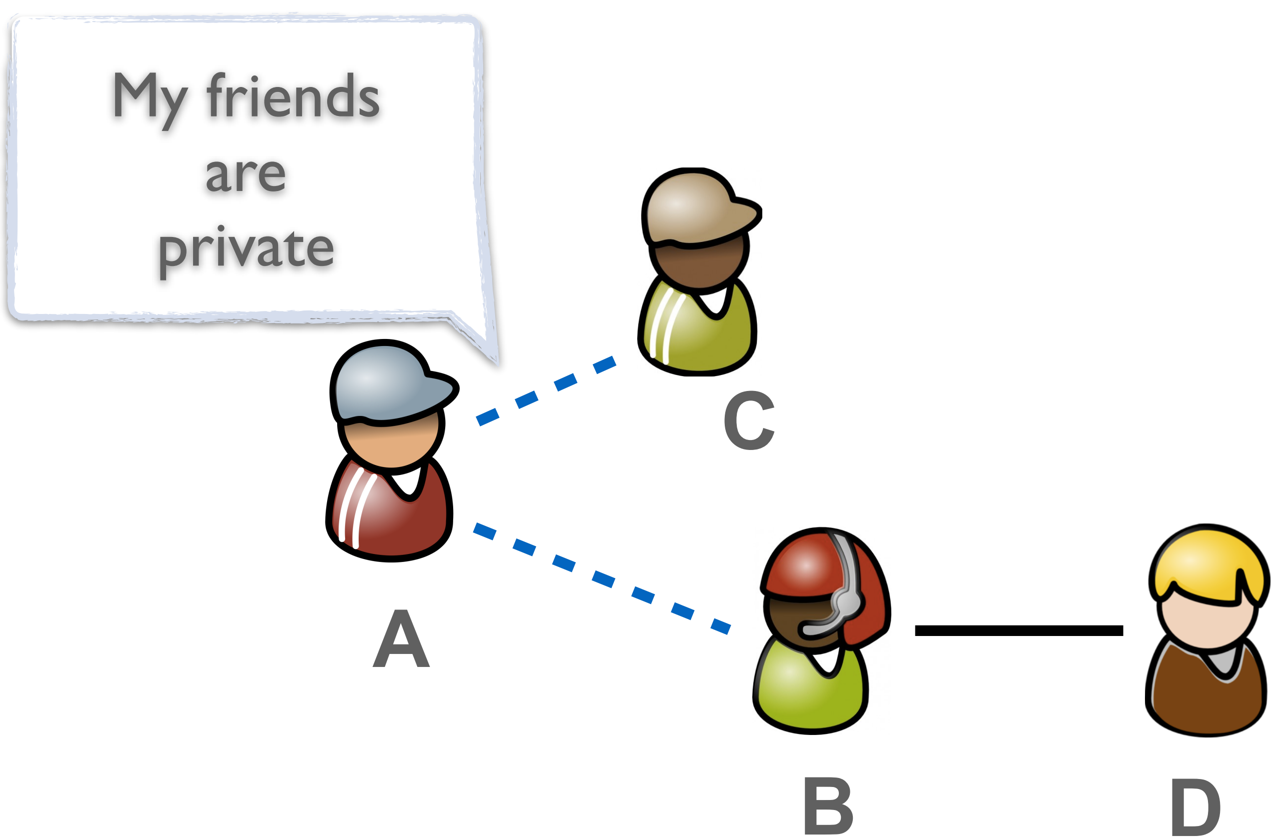
We can only use the data that the user can access.

Naively, we need to **run** the algorithms **once** for **each** user on a **different** (*and huge*) graph!

Application: Friend suggestion

Network signals are very useful

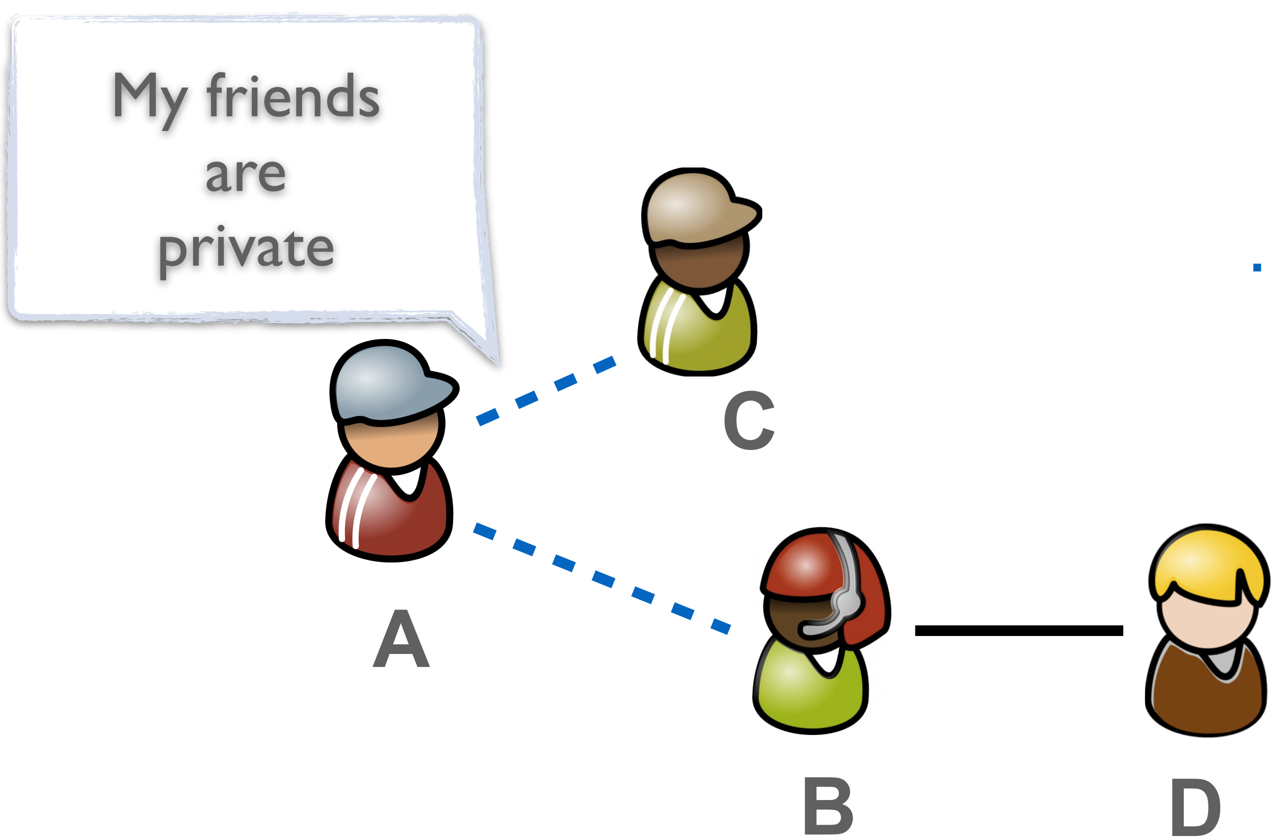
- Number of common neighbors
- Personalized PageRank, etc.



Application: Friend suggestion

Common Neighbors - **Ideal World**

1) Run the algorithm *(in parallel)* on ***the graph G***
2) For each user suggest top k users by common neighbors.



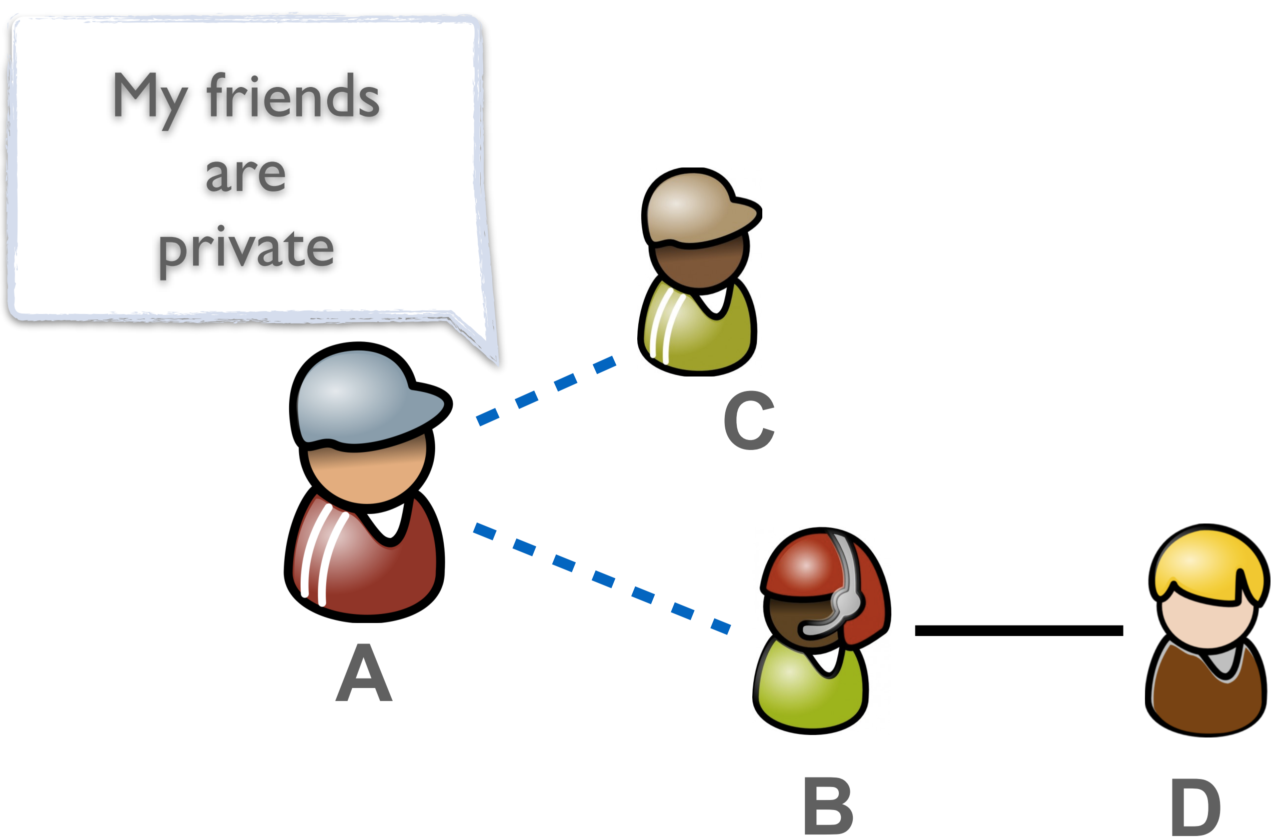
… but there is no such graph G.

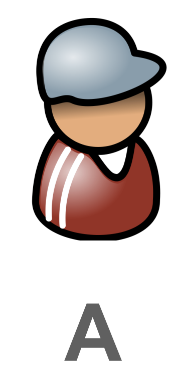
Application: Friend suggestion

Common Neighbors - **Real World**

Multiple graphs = Multiple answers!

How many common neighbors do **B** and **C** have?



Answer for 

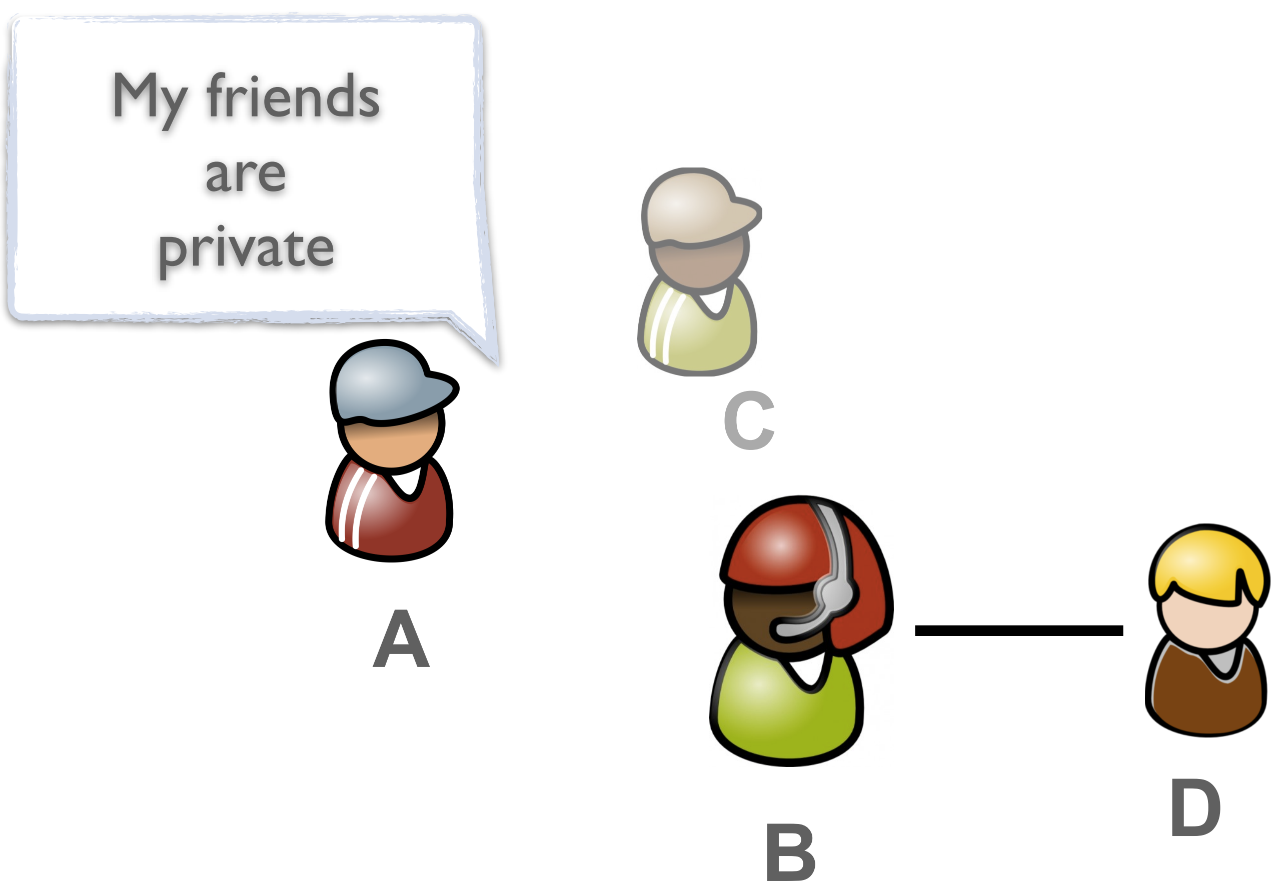
One common
neighbor: **me!**

Application: Friend suggestion

Common Neighbors - **Real World**

Multiple graphs = Multiple answers!

How many common neighbors do **B** and **C** have?



Answer for 

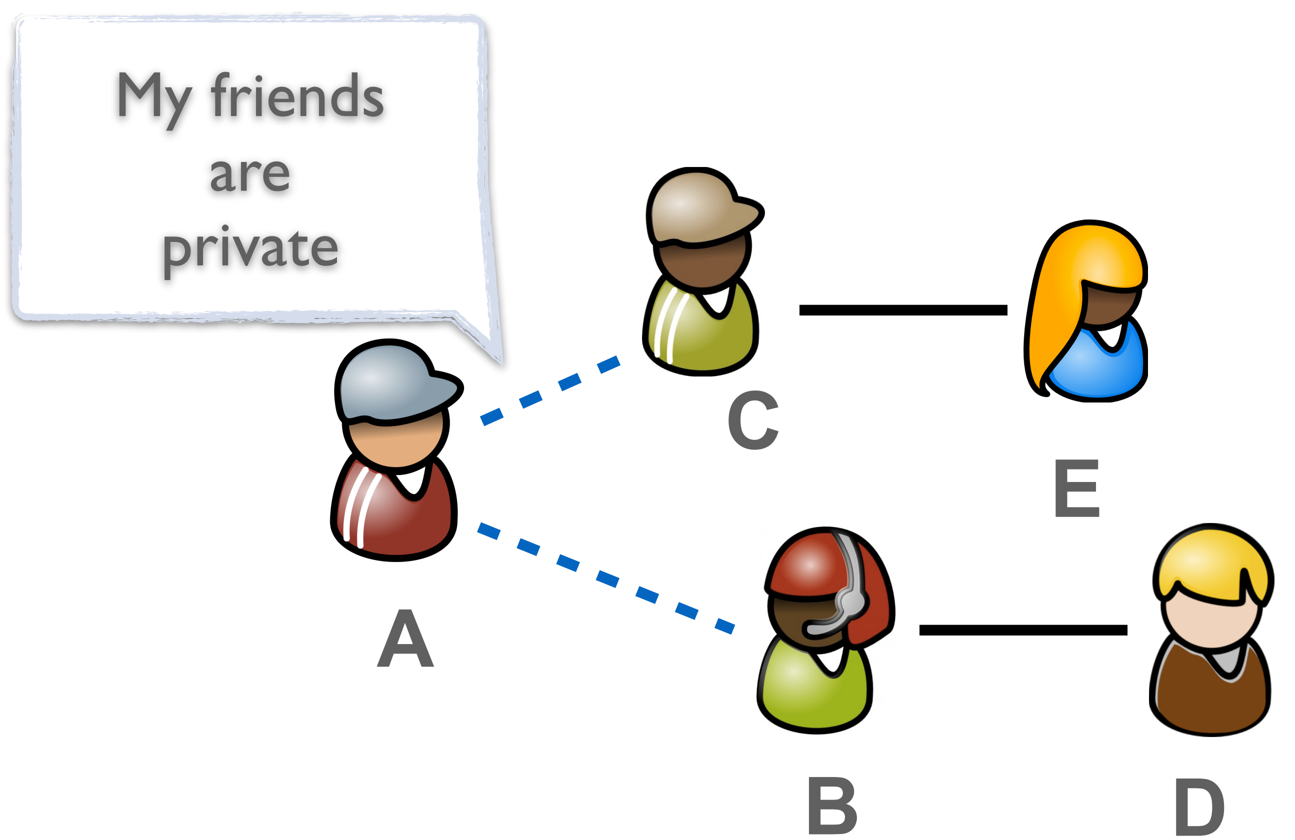
B

Zero common neighbors**!**

We **cannot** suggest C to B as friends based on common neighbors!

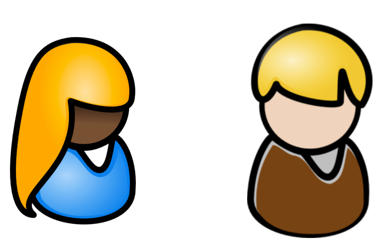
Naive approaches

1) Running the algorithms N times is **infeasible**
2) **Ignoring all private** data is very **ineffective**!



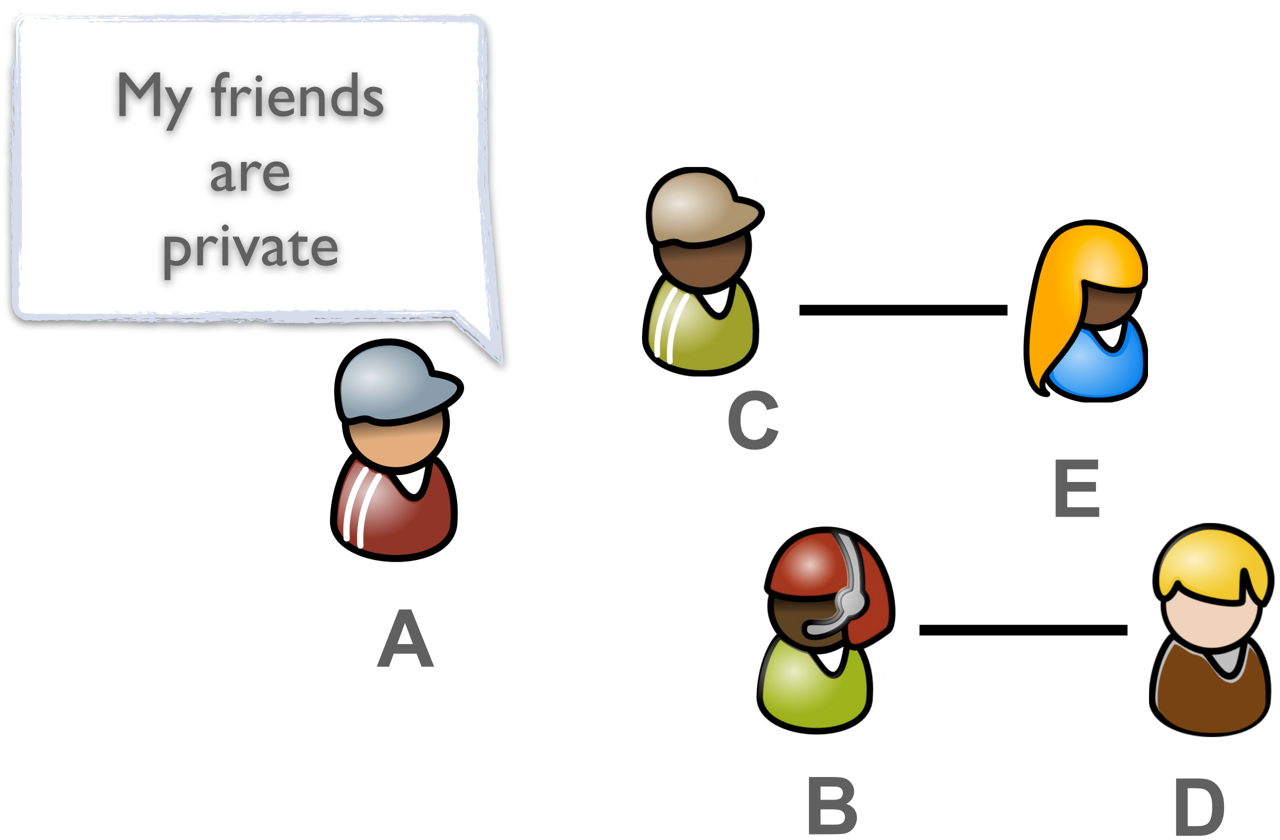
From **user A's prospective** there are interesting signals

E and **D** are good suggestions!



Naive approaches

1) Running the algorithms N times is **infeasible**
2) **Ignoring all private** data is very **ineffective**!



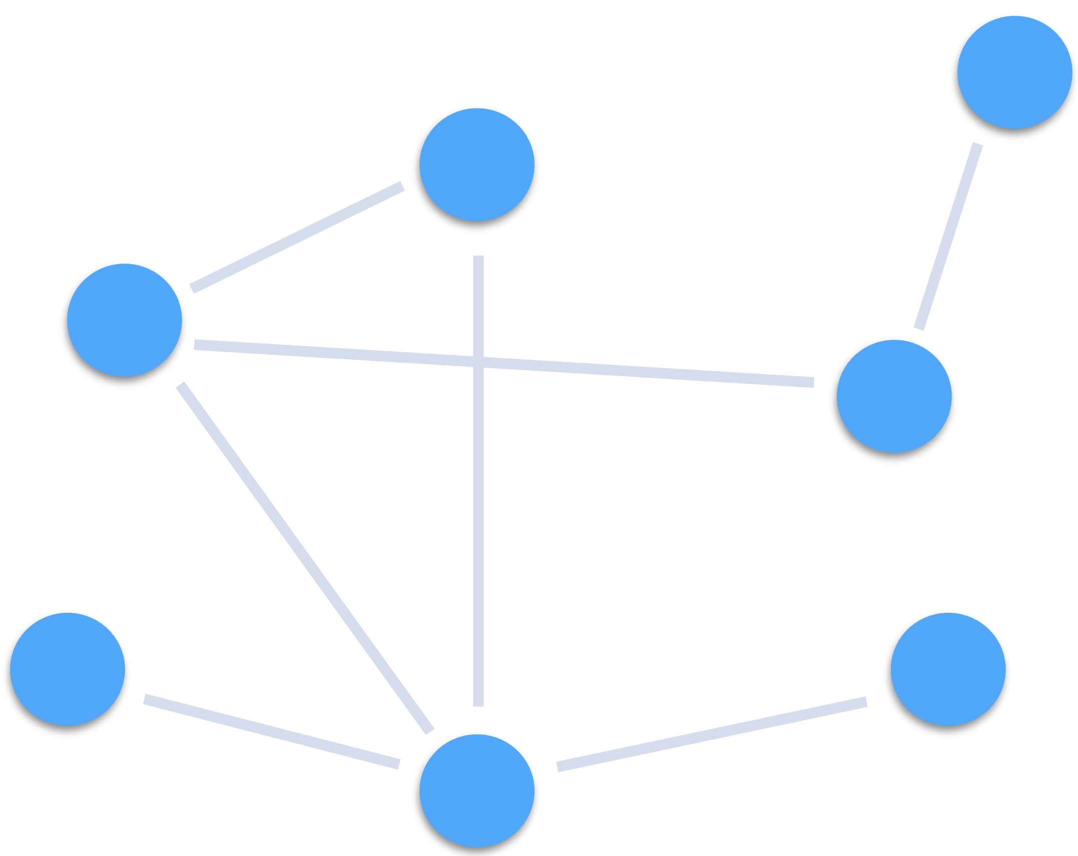
From **public data prospective** there are **no** signals!

No suggestions for the user!

Public-Private Graph Model

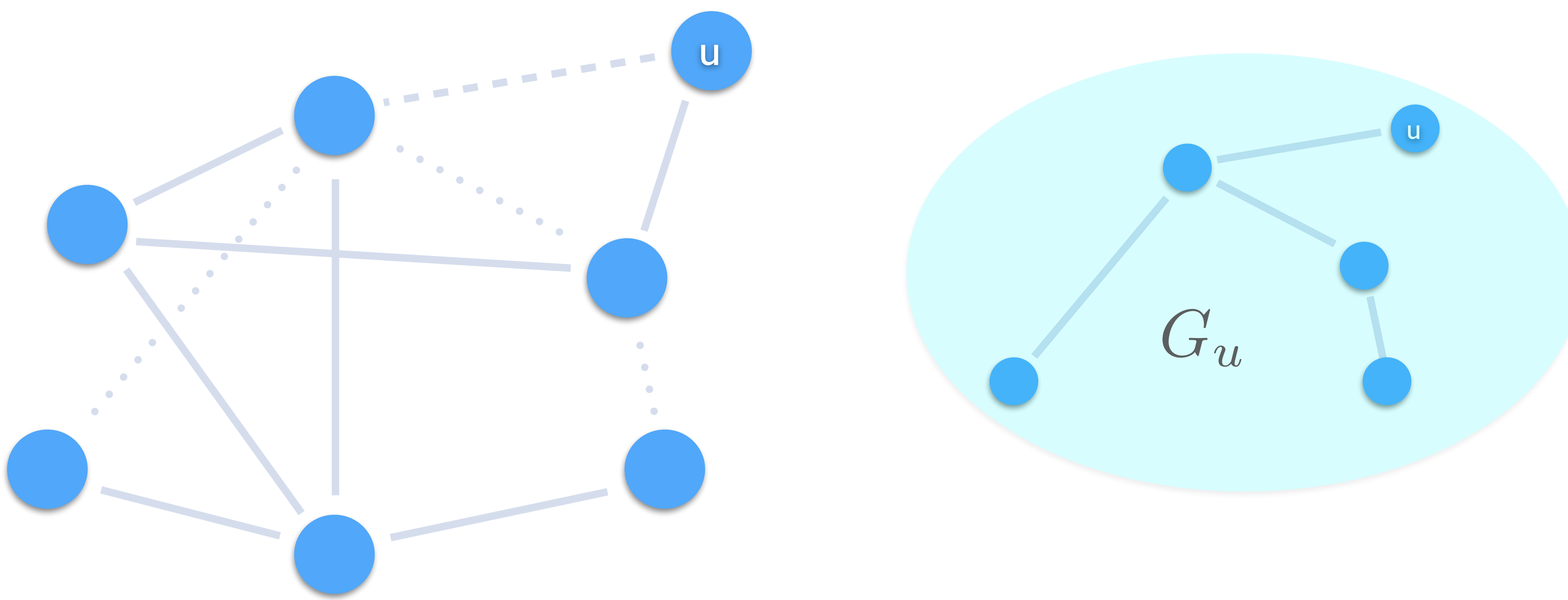
Private-Public model

There is a public graph G



Private-Public model

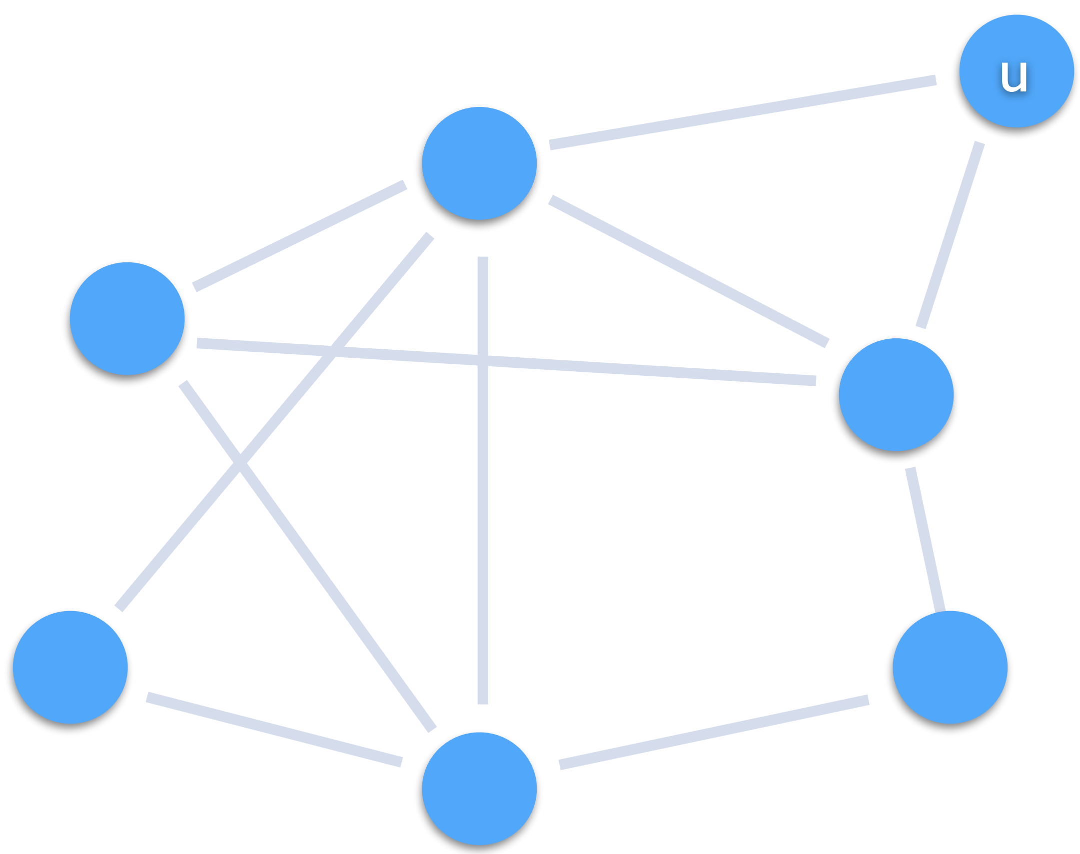
There is a public graph G in addition every node u has access to a private graph G_u



We assume the private graph to be at <= 2 hops from u.

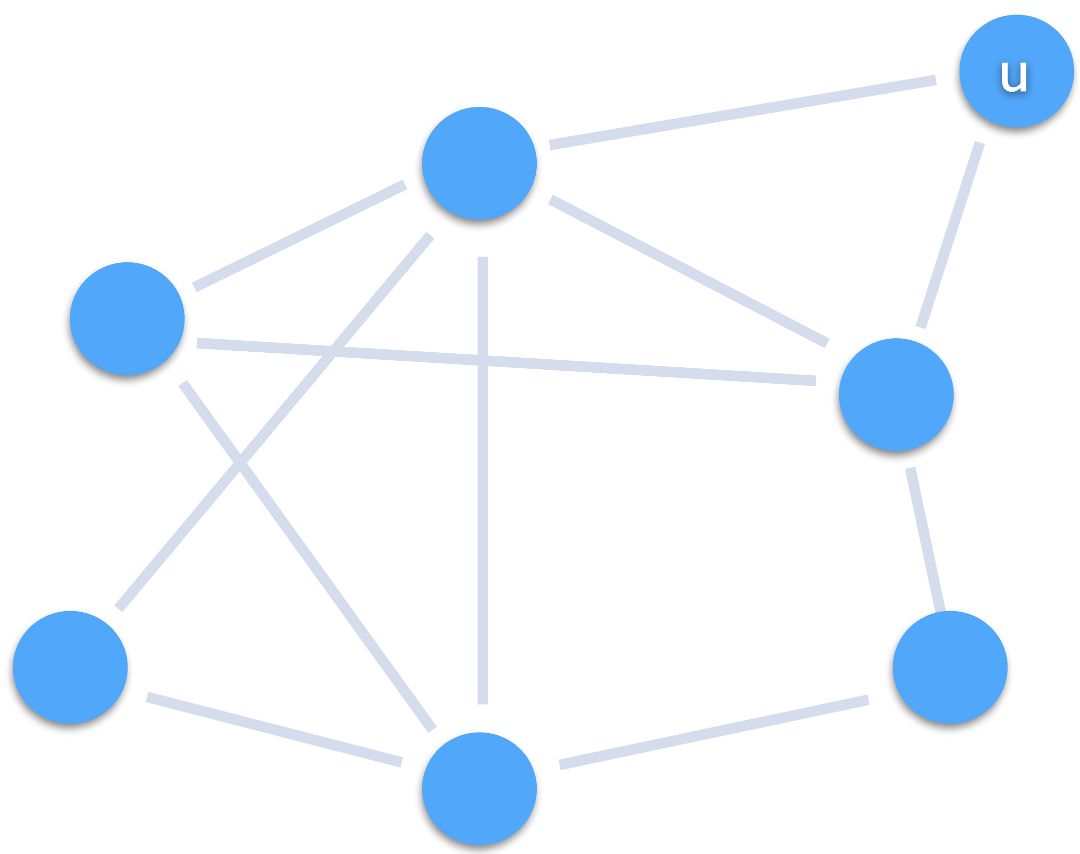
Private-Public model

For each u we would like to execute computation on $G \cup G_u$



Private-Public model

For each u we would like to execute computation on $G \cup G_u$

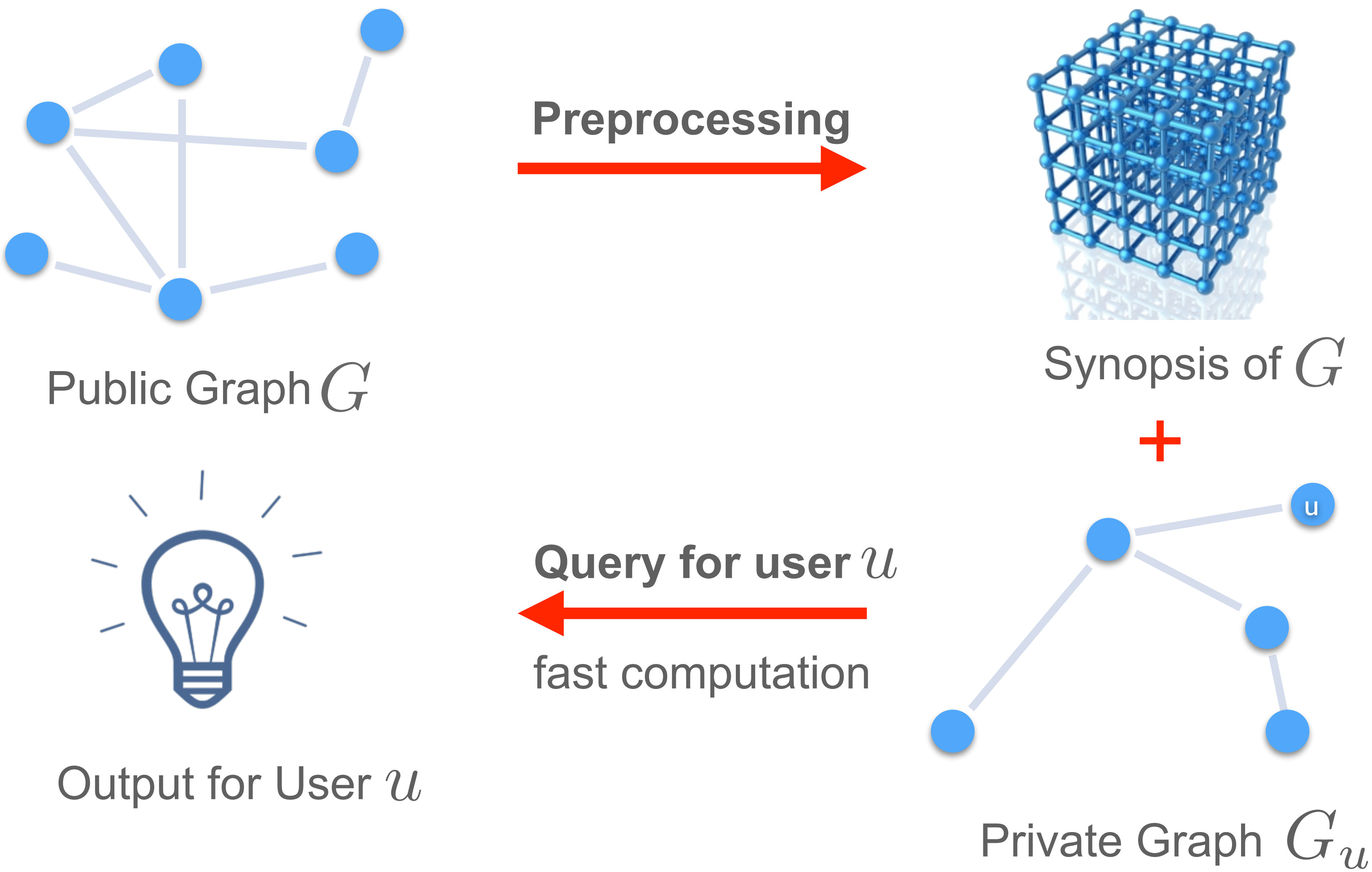


This respects the privacy of each user.

We want the computation to be efficient.

Two-Steps Approach

Precompute data structure for G so that we can solve problems in $G \cup G_u$ efficiently.



Private-Public problem

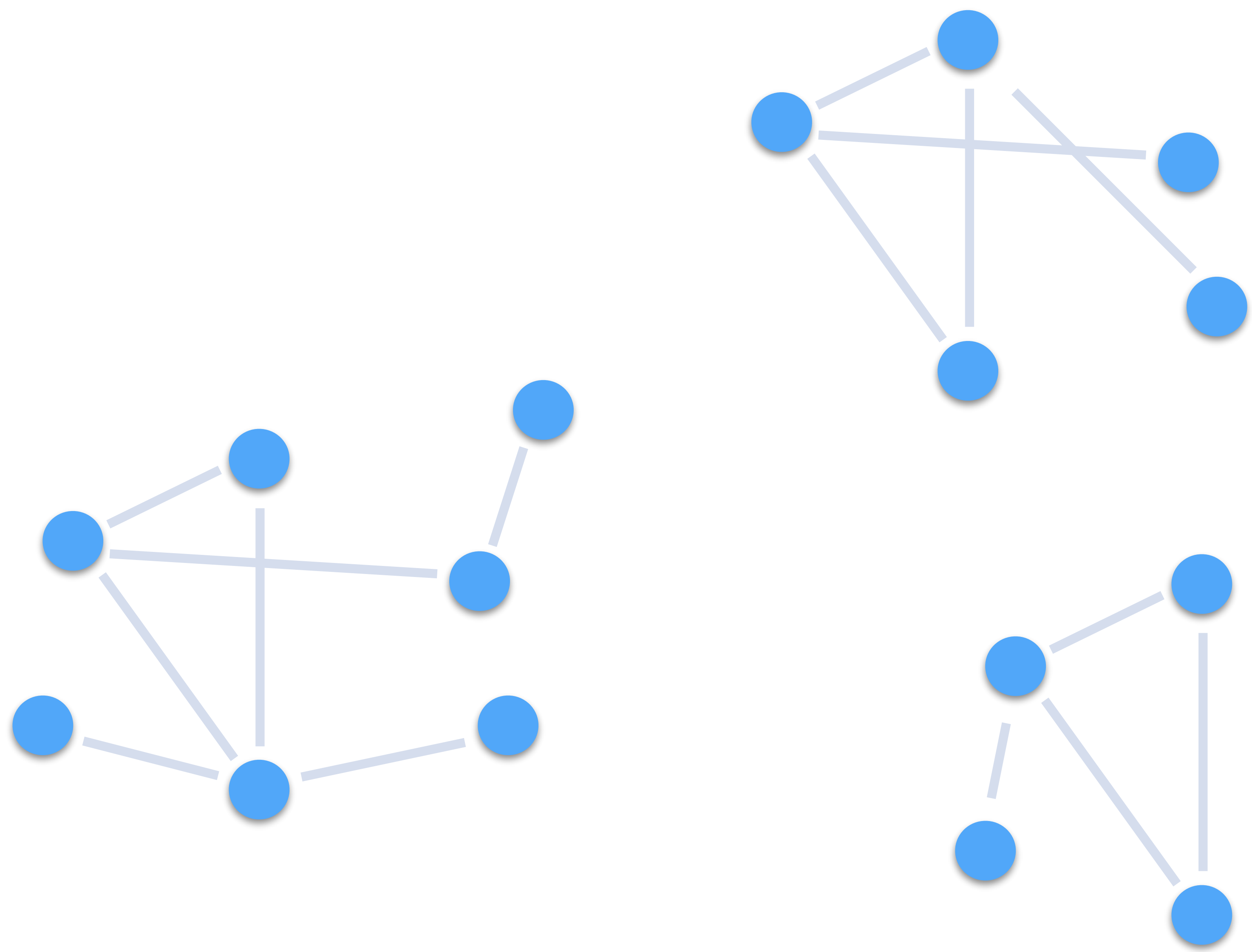
Ideally.

Preprocessing time: $\tilde{O}\left(|E_G|\right)$

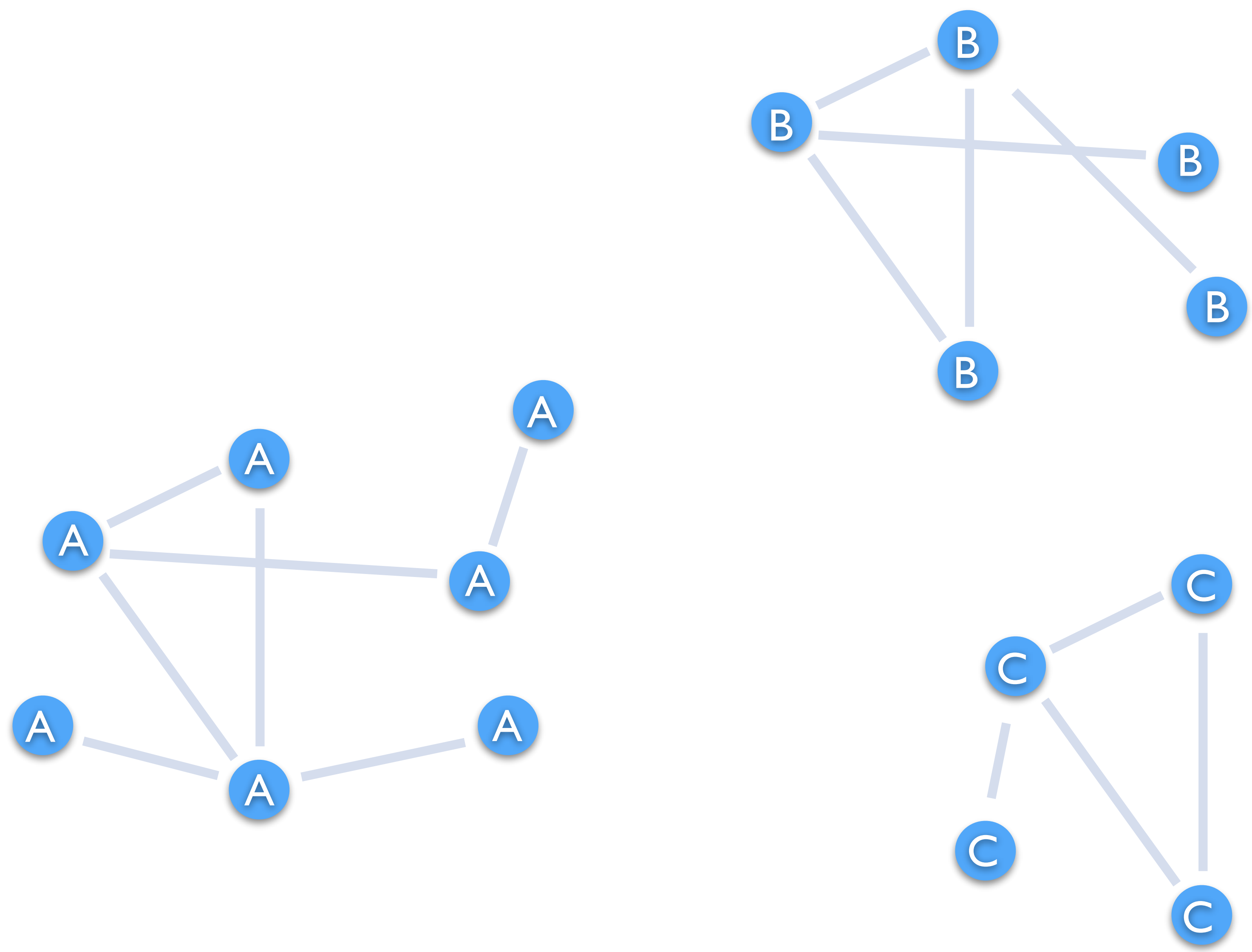
Preprocessing space: $\tilde{O}\left(|V_G|\right)$

Query time: $\tilde{O}\left(|E_{G_u}|\right)$

Warm-up: # connected components

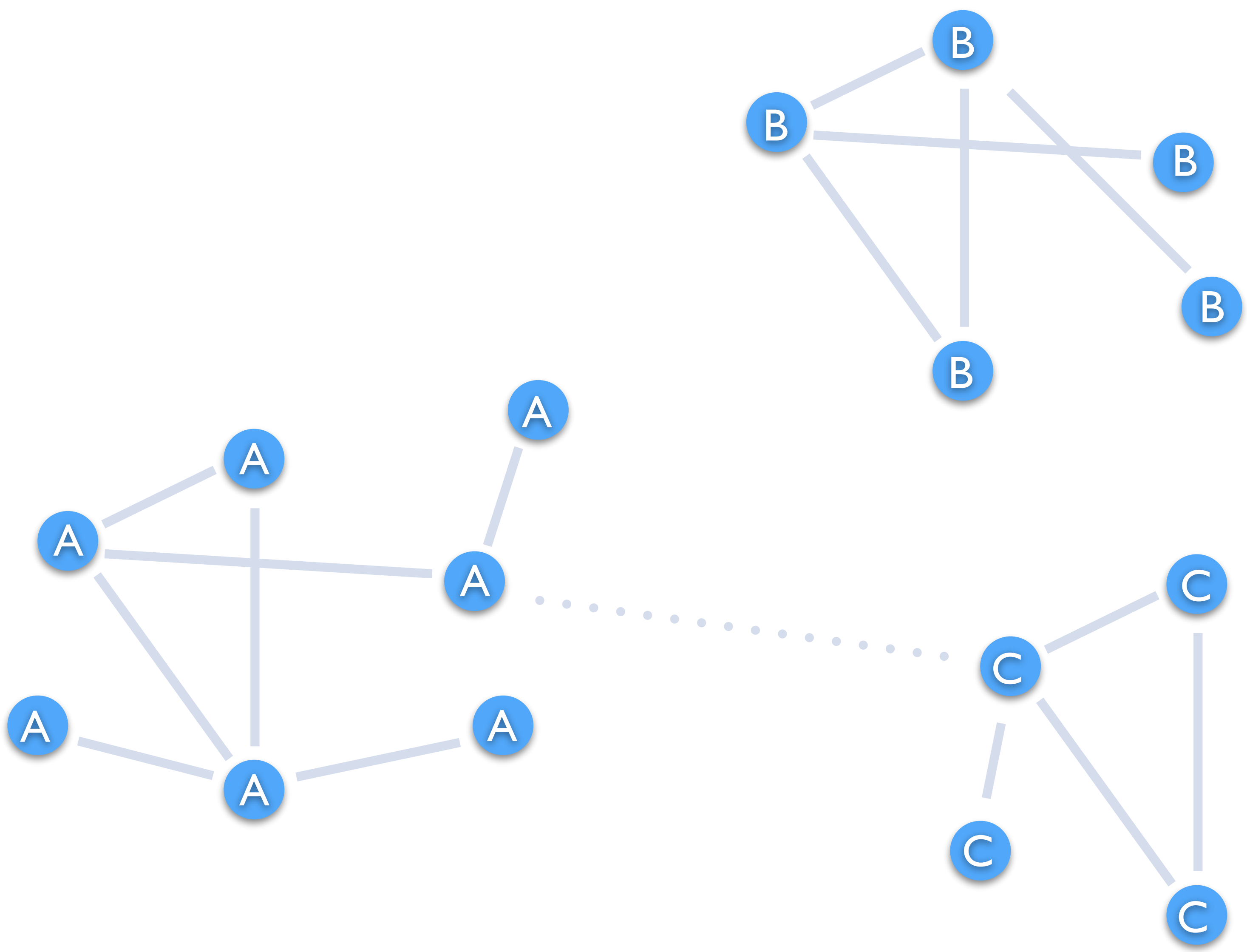


Warm-up: # connected components



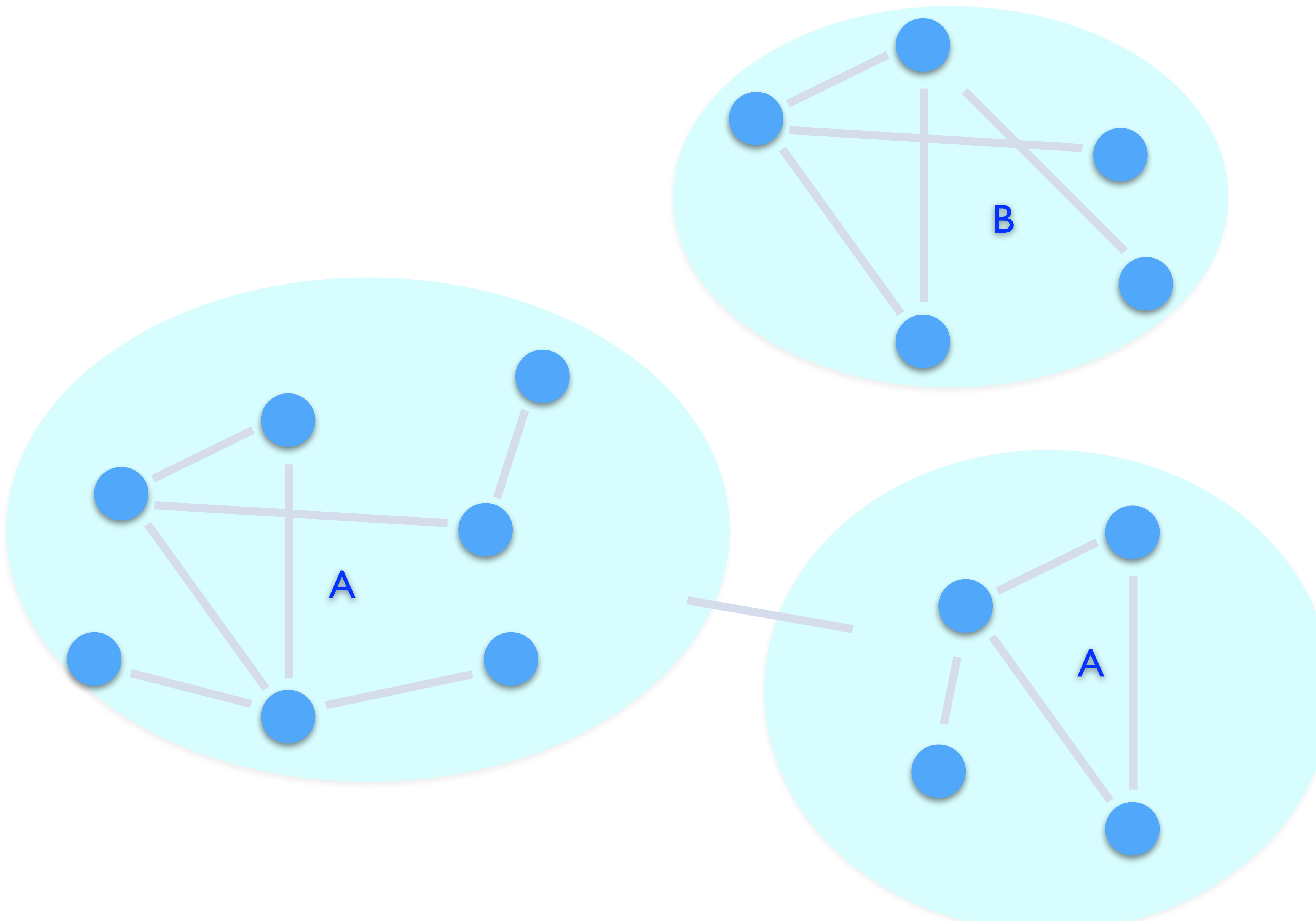
Precompute component IDs in G

Warm-up: # connected components



Add private edges and merge conn. components

Warm-up: # connected components



Add private edges and merge conn. components.

Results

Algorithms

- Reachability
- Approximate All-pairs shortest paths
- Correlation clustering
- Social affinity

Heuristics

- Personalized PageRank
- Centrality measures

Results

Algorithms

Reachability

Approximate All-pairs shortest paths

Correlation clustering

Social affinity

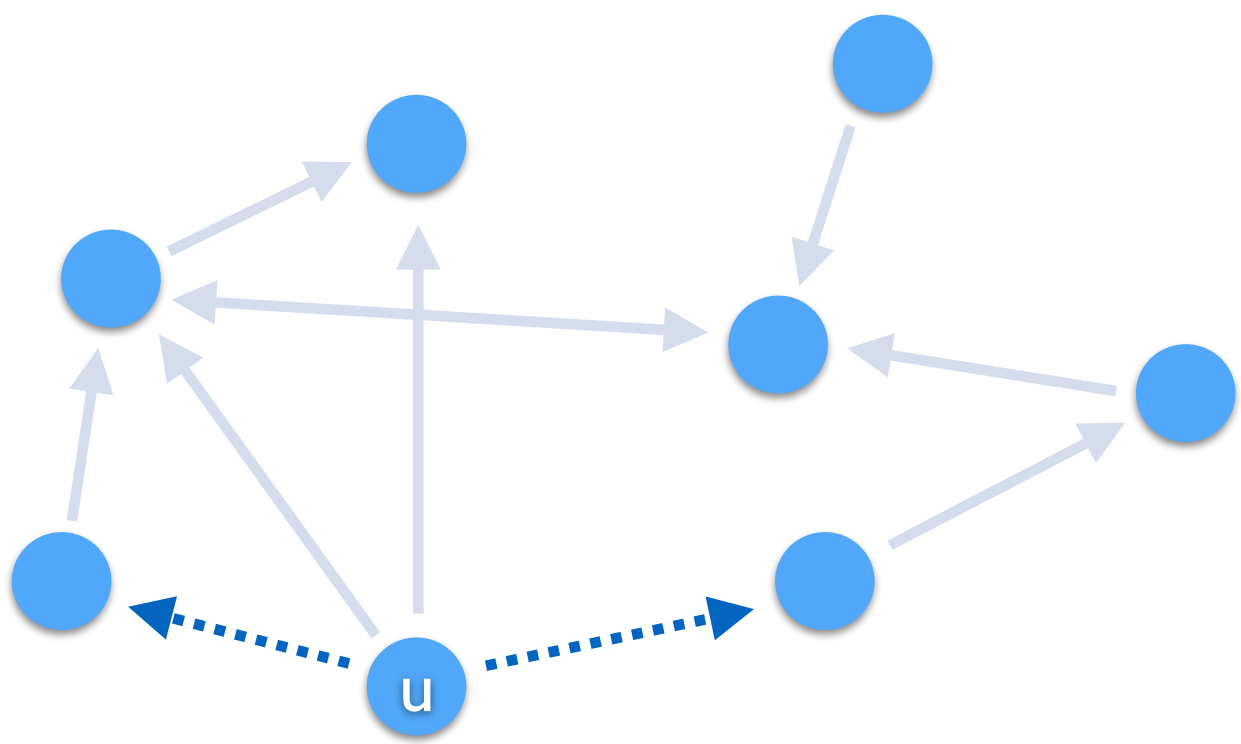
Heuristics

Personalized PageRank

Centrality measures

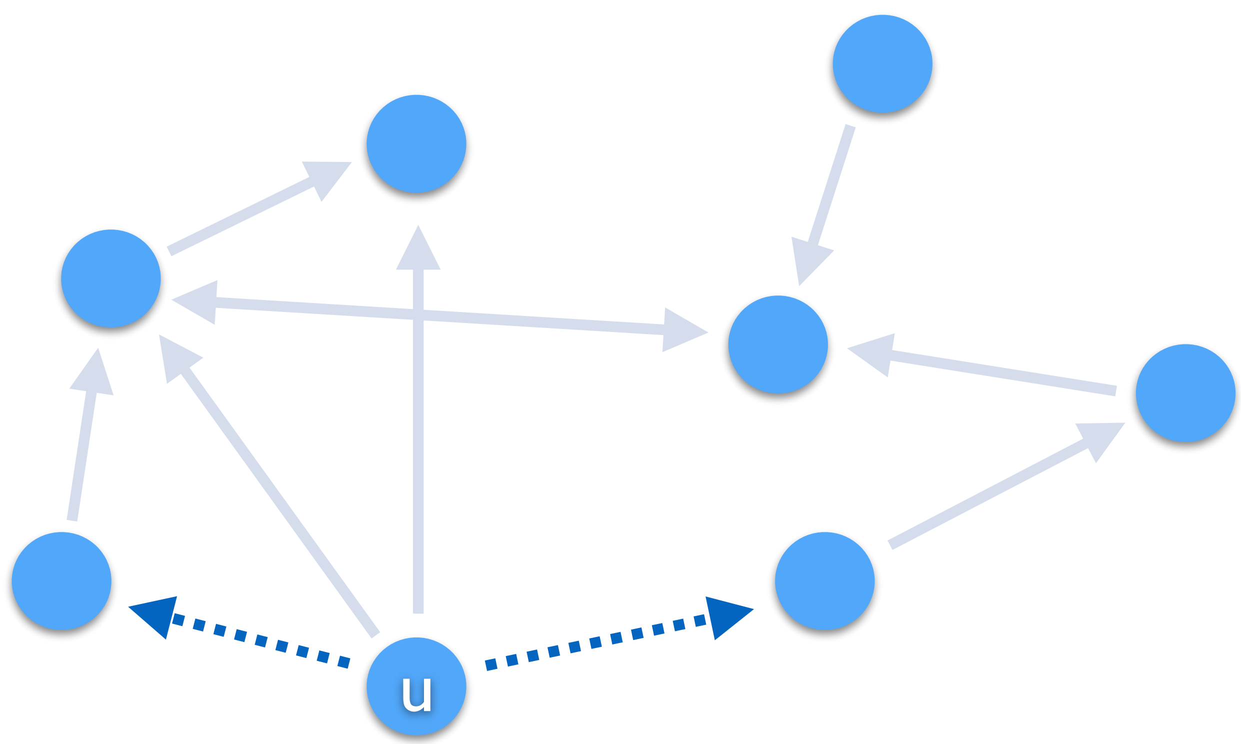
Reachability

How many nodes can I reach from u?



Reachability

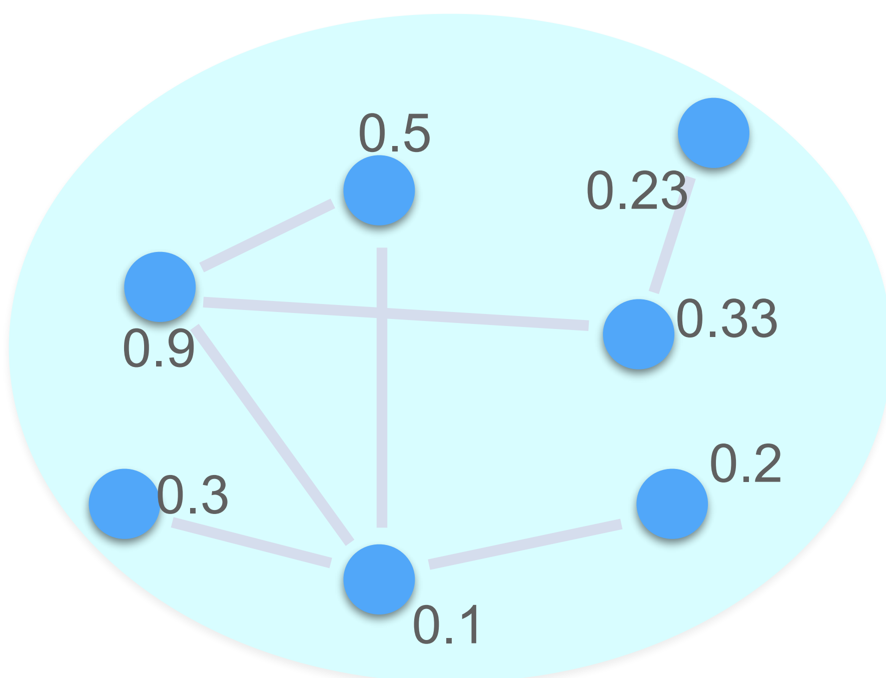
How many nodes can I reach from u?



We have to handle overlaps.

Reachability

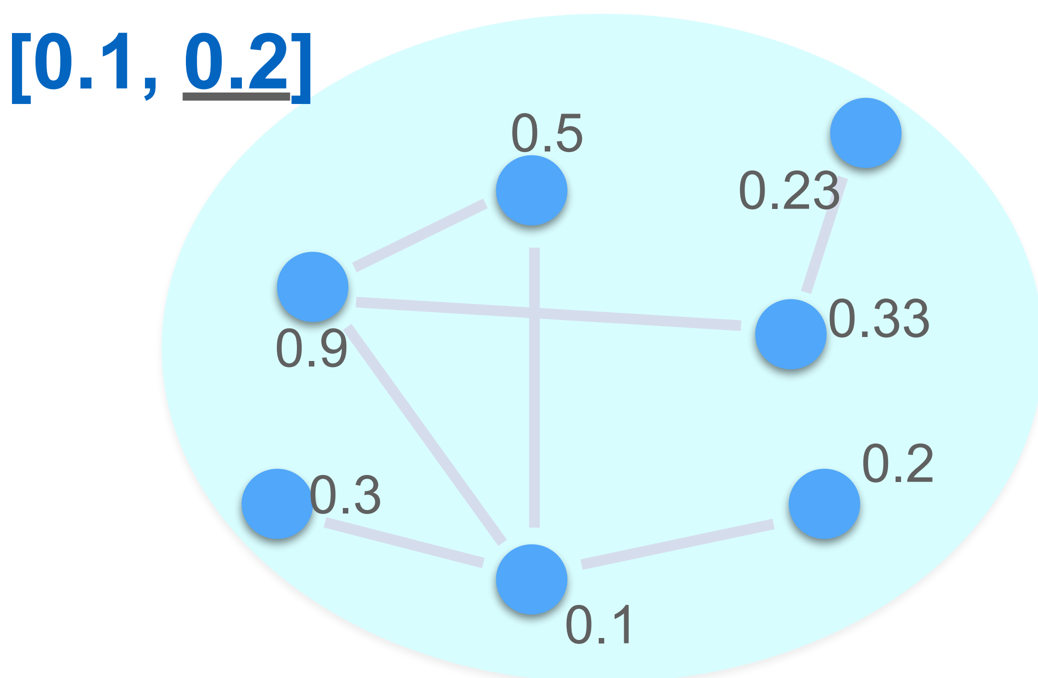
Key idea: use size-estimation sketch [Cohen JCSS97]



Every node samples a random number between [0,1]

Reachability

Key idea: use size-estimation sketch [Cohen JCSS97]

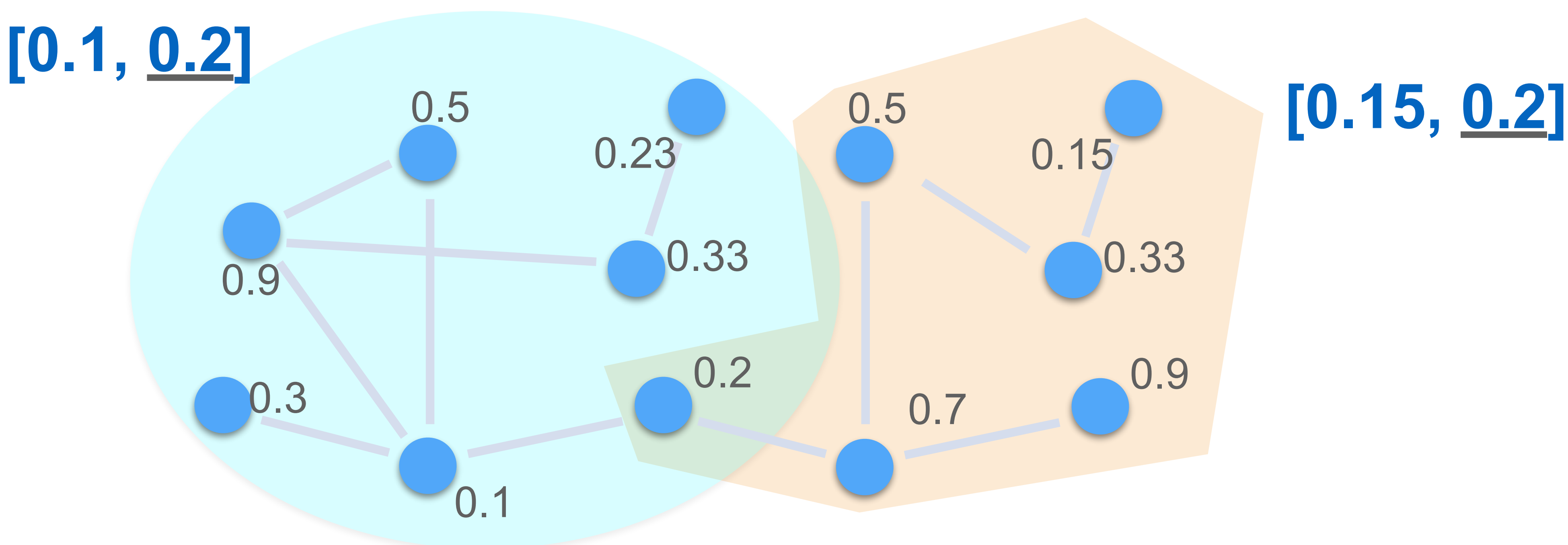


Every node samples a random number between [0,1].

Look at the **k-th smallest value**, use it to estimate the size of the set.

Reachability

Key idea: use size-estimation sketch [Cohen JCSS97]



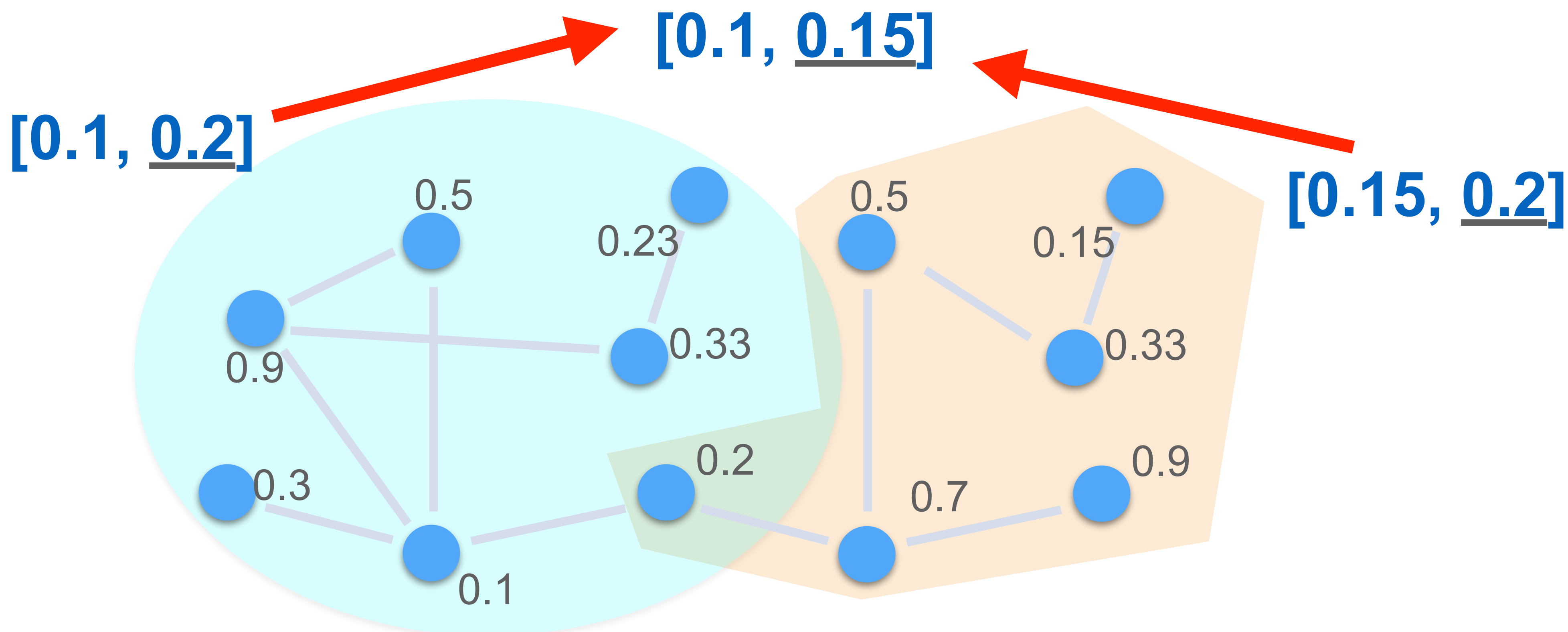
Every node samples a random number between [0,1].

Look at the **k-th smallest value**, use it to estimate the size of the set.

Composable sketch of size **k.**

Reachability

Key idea: use size-estimation sketch [Cohen JCSS97]



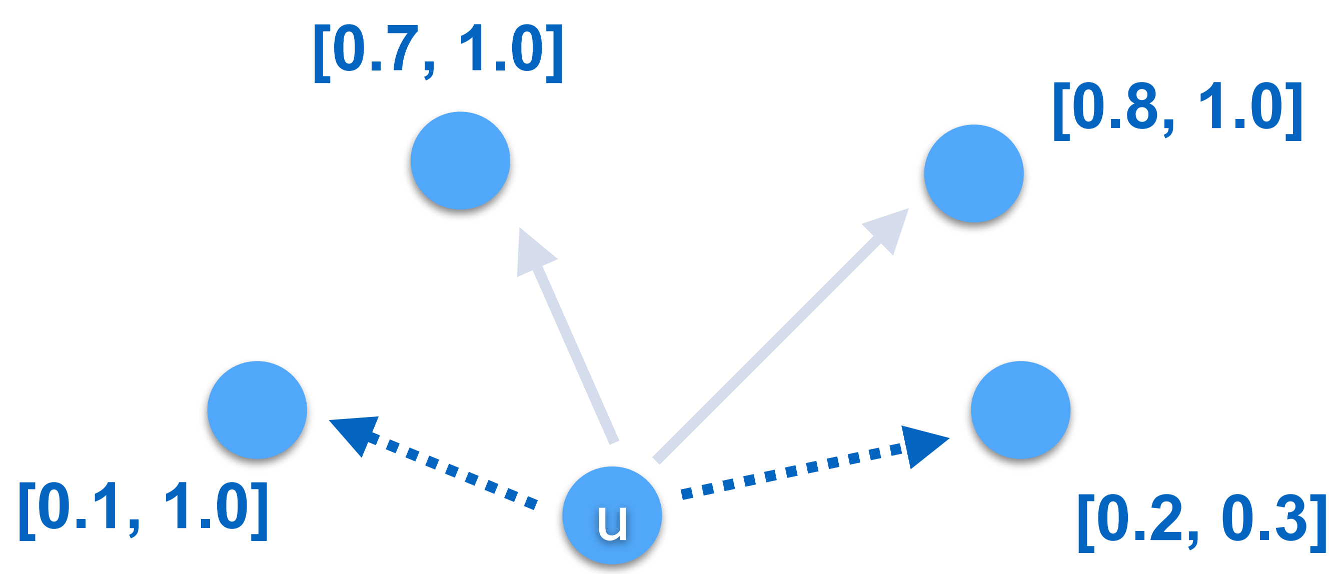
Every node samples a random number between [0,1].

Look at the **k-th smallest value**, use it to estimate the size of the set.

Composable sketch of size **k.**

Reachability

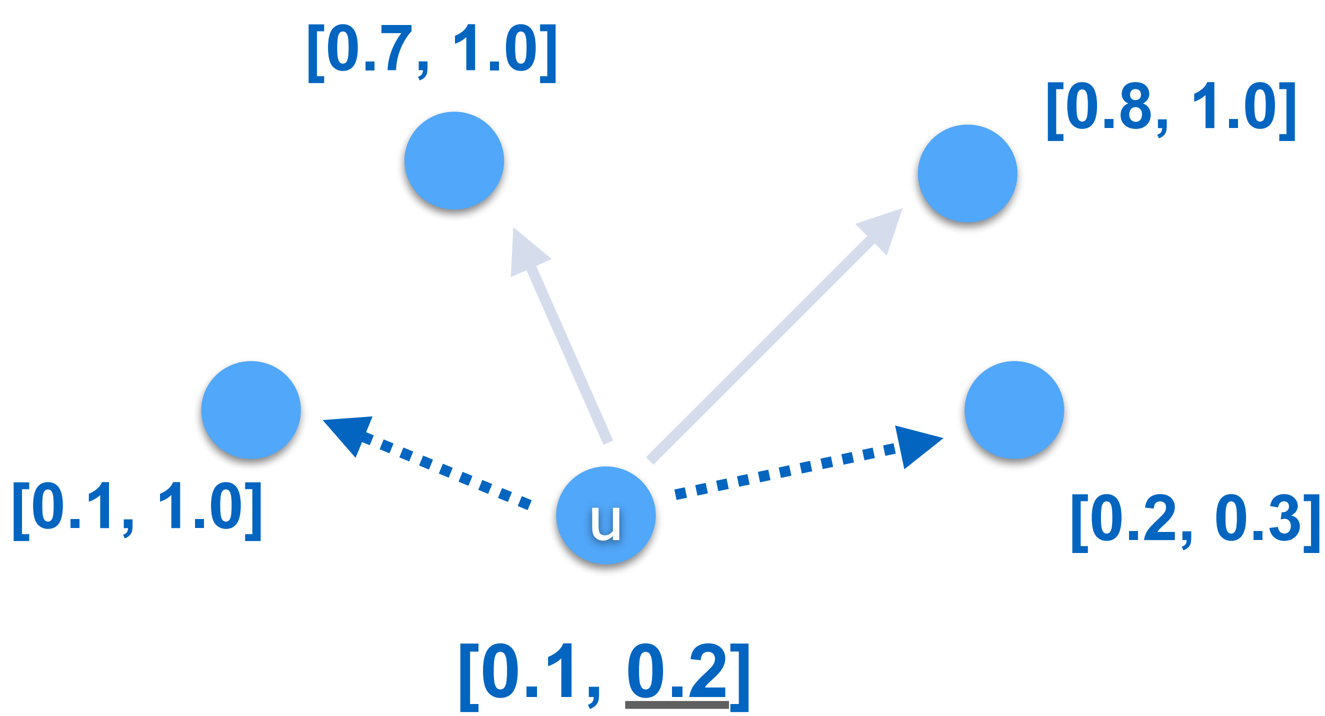
How many nodes can I reach from u?



Precompute sketches for each node in public graph.

Reachability

How many nodes can I reach from u?



Compose sketches of nodes reachable in private graph.

Experiments Personalized PageRank

Approximating the **PPR** stationary distribution.

Graph	A/B	Cosine
DBLP	6.5e-3	99.8%
LiveJournal	3.5e-4	99.1%
Orkut	1.6e-3	99.9%
YouTube	1.7e-2	99.8%

Up to **4 orders** of **magnitudes faster** naive approach.

Conclusions

- New model for practical problems;
- Some algorithms designed using sampling and sketching techniques;
- Large speed-up in practice.

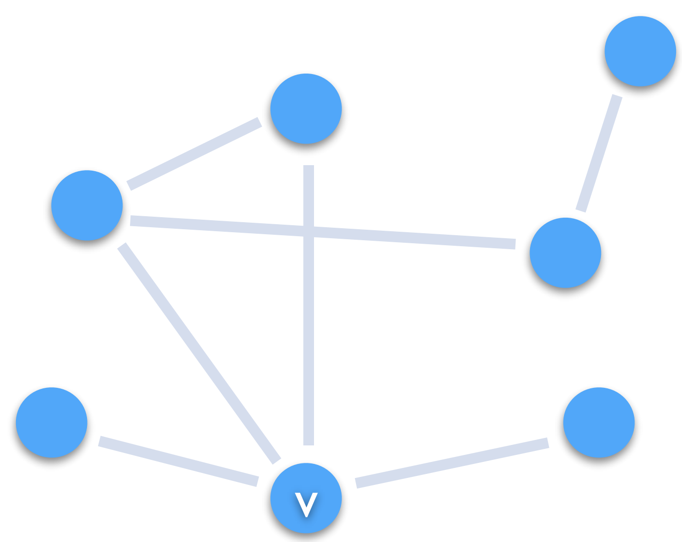
Future works

- New algorithms for other problems;
- Not only graph problems;
- Study limit of the model (lower bounds).

Thanks!

Personalized PageRank

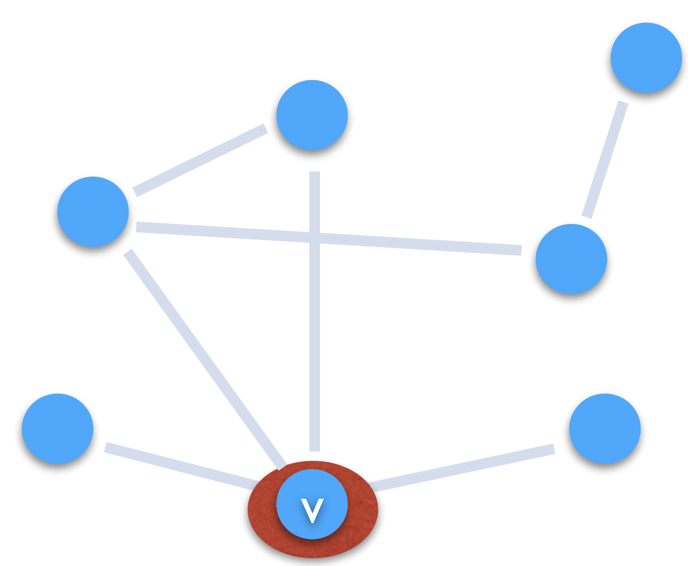
$PPR(v,z)$ is the probability of visiting z in the following lazy random walk:
- with probability α jumps to v
- with probability $1-\alpha$ jumps to a random neighbor



Personalized PageRank

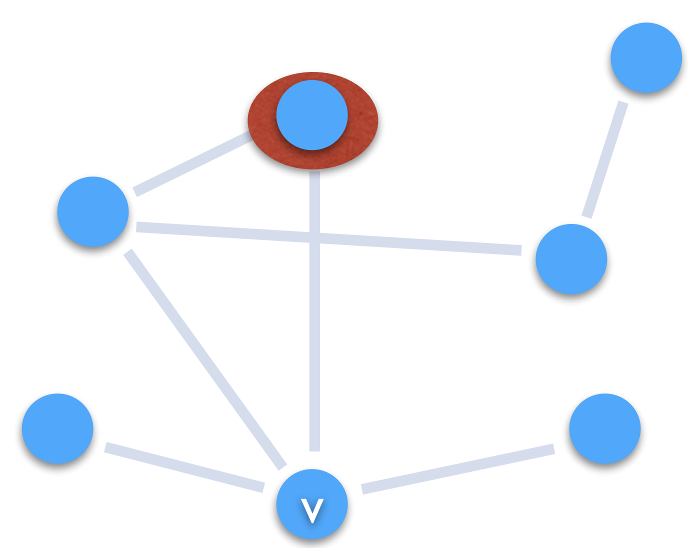
$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:

- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



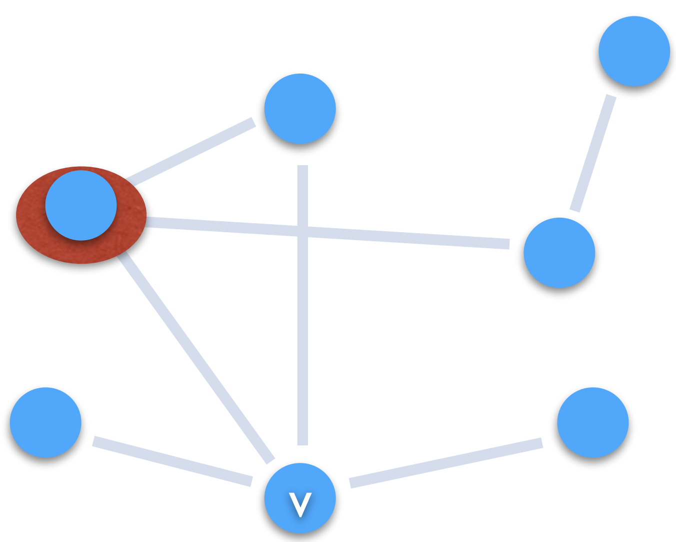
Personalized PageRank

$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:
- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



Personalized PageRank

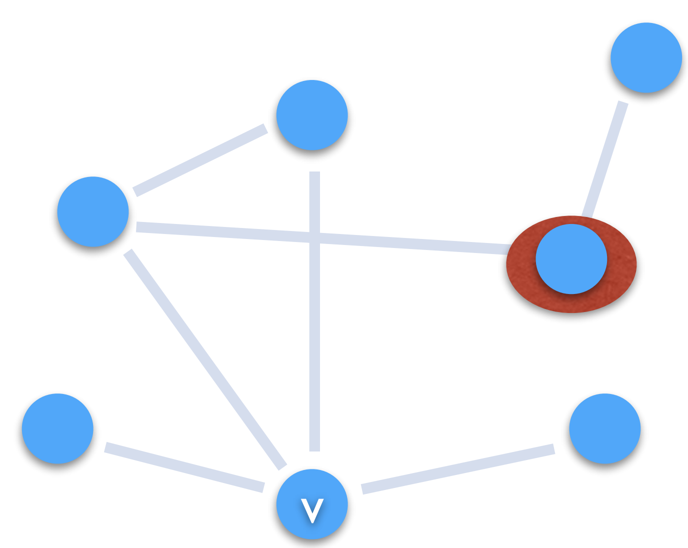
$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:
- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



Personalized PageRank

$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:

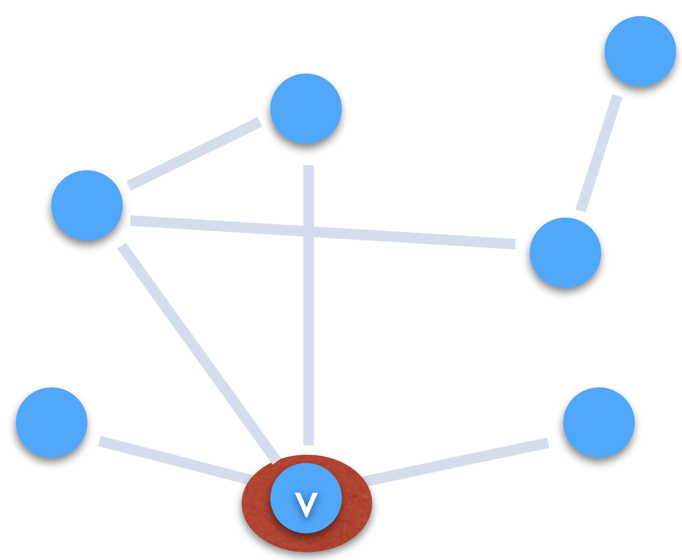
- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



Personalized PageRank

$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:

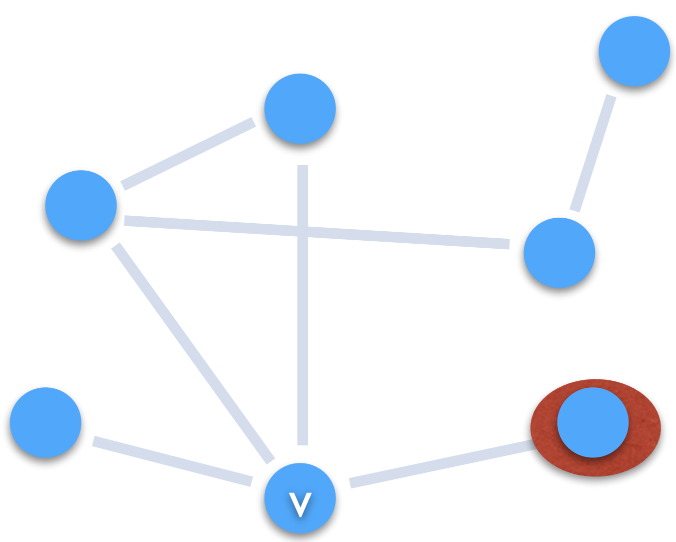
- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



Personalized PageRank

$PPR(v, z)$ is the probability of visiting z in the following lazy random walk:

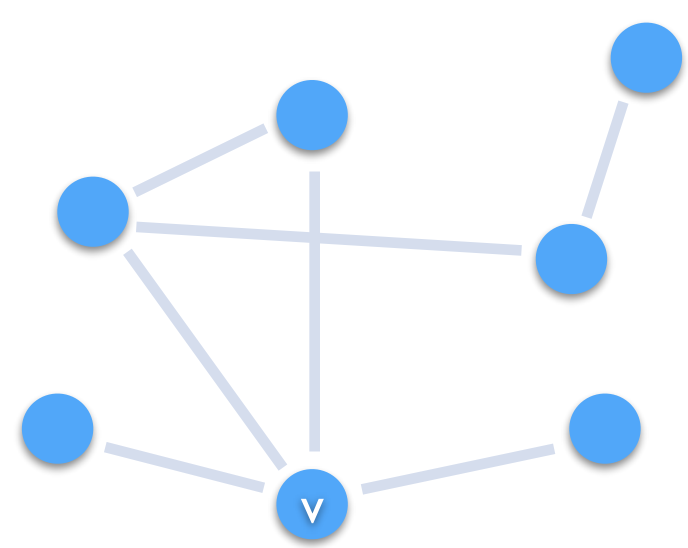
- with probability α jumps to v
- with probability $1 - \alpha$ jumps to a random neighbor



Personalized PageRank

Nice property [Jeh and Widom WWW03]

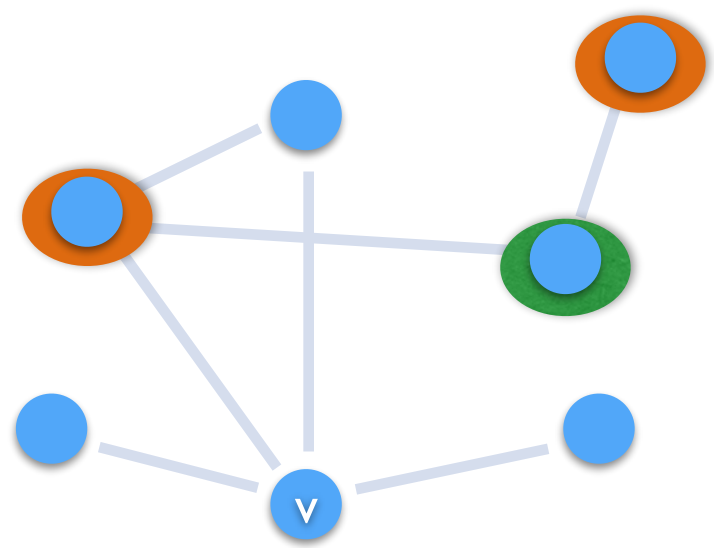
$$PPR_{G \cup G_u}(v,z) = (1-\alpha) d_{G \cup G_u}(y)^{-1} \sum_{y \in N(z)} PPR_{G \cup G_u}(v,y) + \alpha \mathbf{1_v}$$



Personalized PageRank

Nice property [Jeh and Widom WWW03]

$$PPR_{G \cup G_u}(v, z) = (1 - \alpha) d_{G \cup G_u}(y)^{-1} \sum_{y \in N(z)} PPR_{G \cup G_u}(v, y) + \alpha \mathbf{1_v}$$

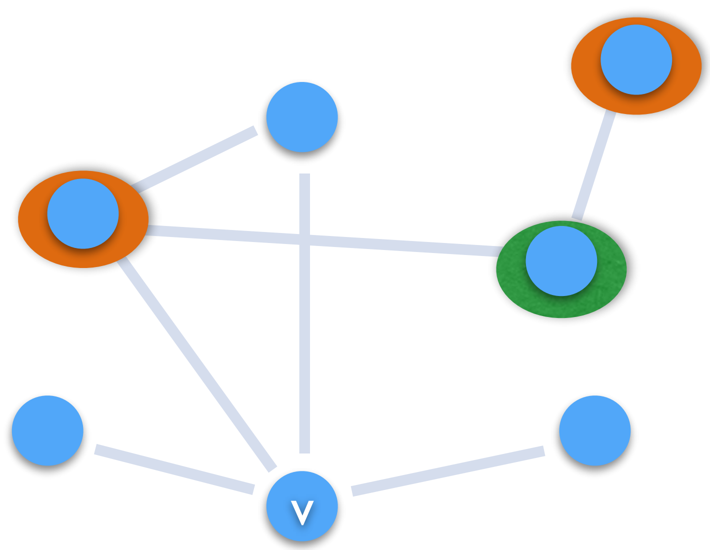


Personalized PageRank

Nice property [Jeh and Widom WWW03]

$$PPR_{G \cup G_u}(v,z) = (1-\alpha) d_{G \cup G_u}(y)^{-1} \sum_{y \in N(z)} \boxed{PPR_{G \cup G_u}(v,y)} + \alpha \mathbf{1_v}$$

We don't have it



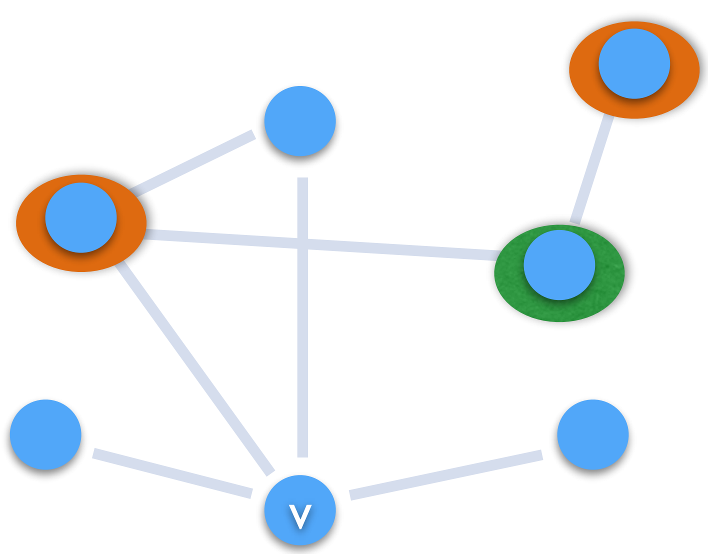
Personalized PageRank

Nice property [Jeh and Widom WWW03]

$$PPR_{G\cup G_u}(v,z) = (1-\alpha)d_{G\cup G_u}(y)^{-1}\sum_{y\in N(z)} PPR_{G\cup G_u}(v,y) + \alpha\mathbf{1_v}$$

Simple heuristic:

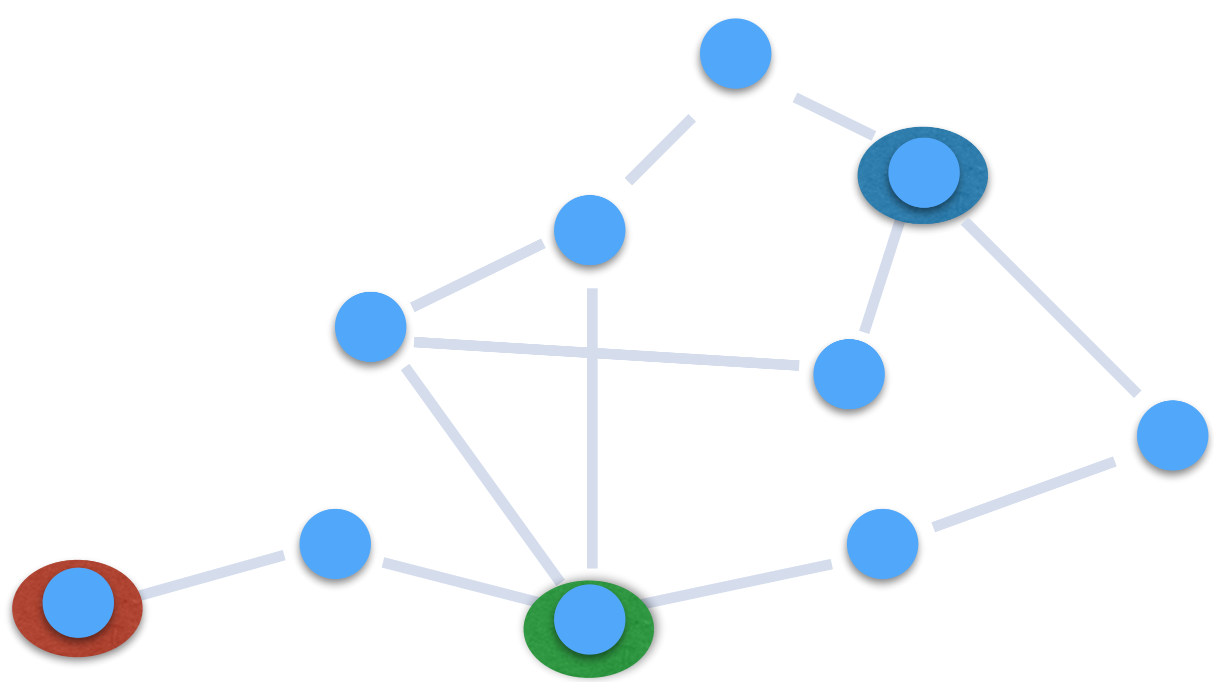
$$PPR_{G\cup G_u}(v,z) \approx (1-\alpha)d_{G\cup G_u}(y)^{-1}\sum_{y\in N(z)} \boxed{PPR_G(v,y)} + \alpha\mathbf{1_v}$$



Using public graph distribution

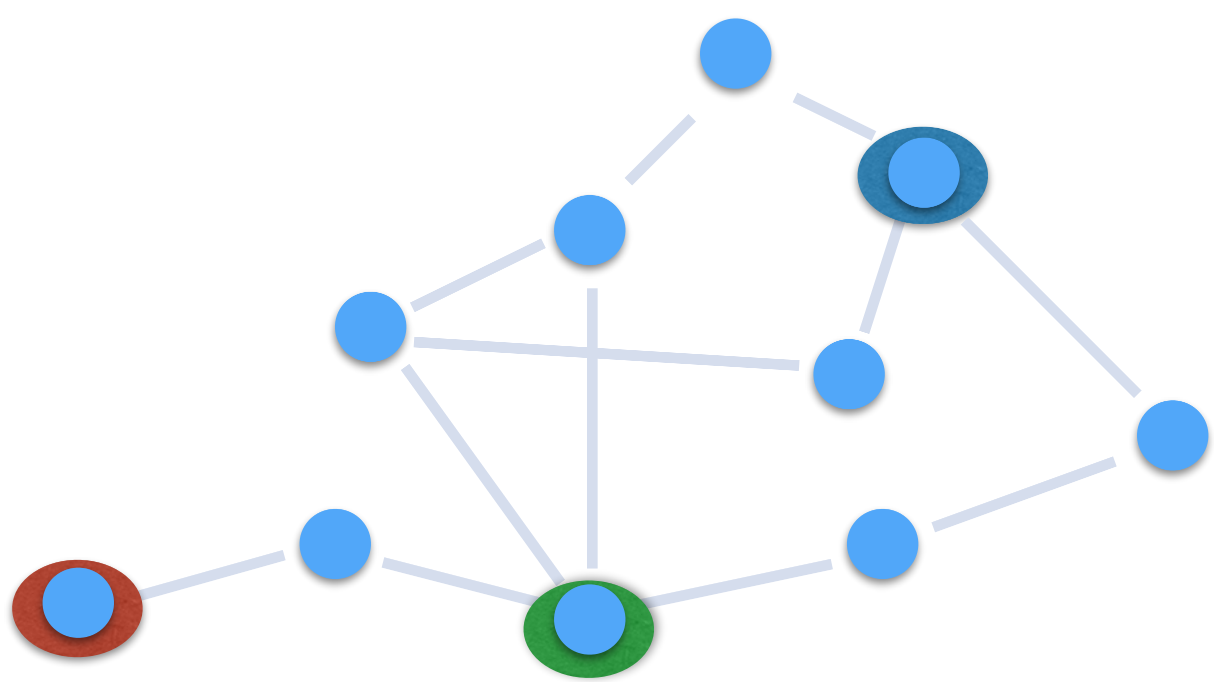
Social affinity

Which connection is stronger?



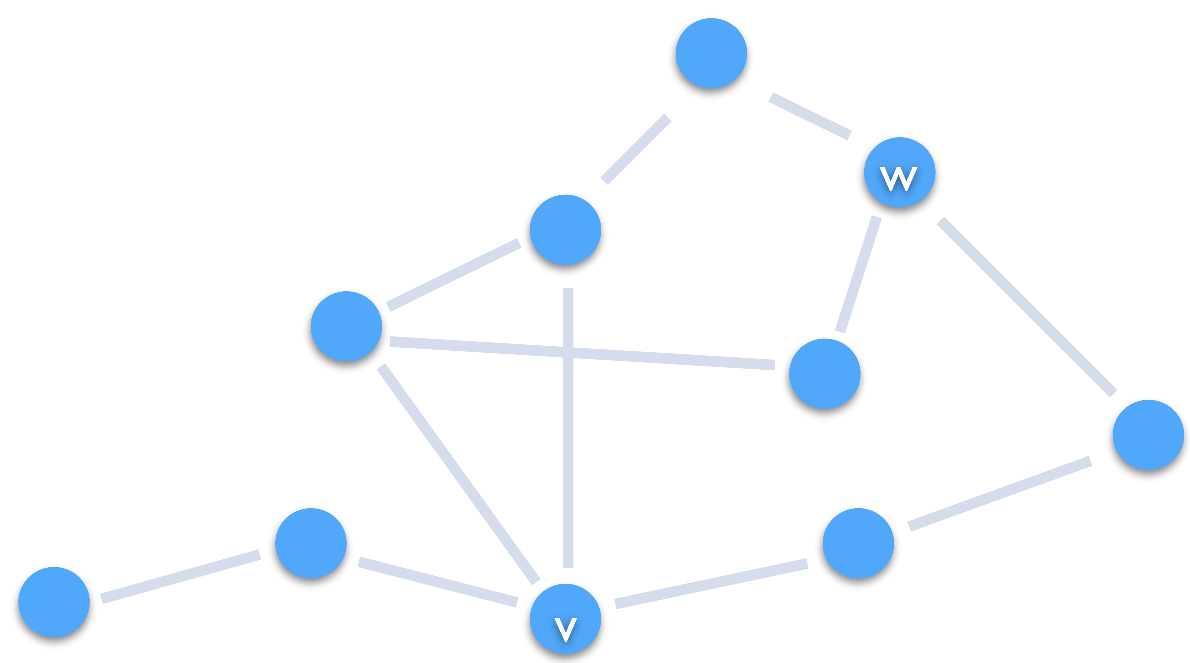
Social affinity

Which connection is stronger?
It is important to consider the number of paths and their lengths



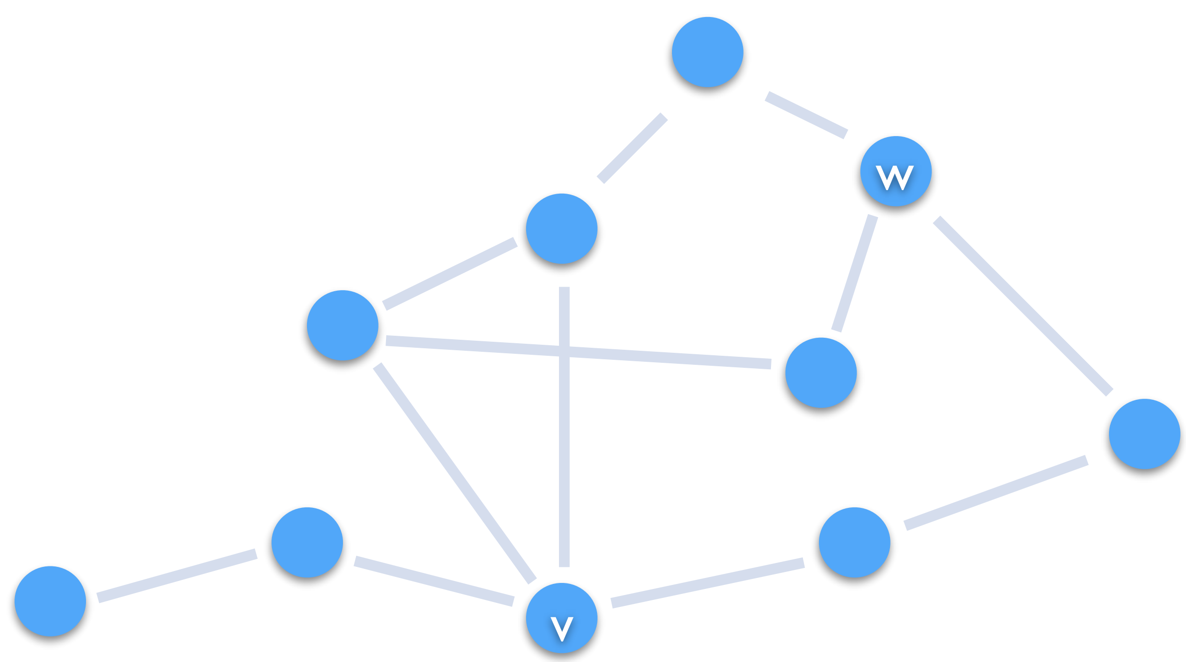
Social affinity

$A_\theta(v, w)$ is the maximum fraction of edges that it is possible to delete and still have v and w connected with probability at least θ



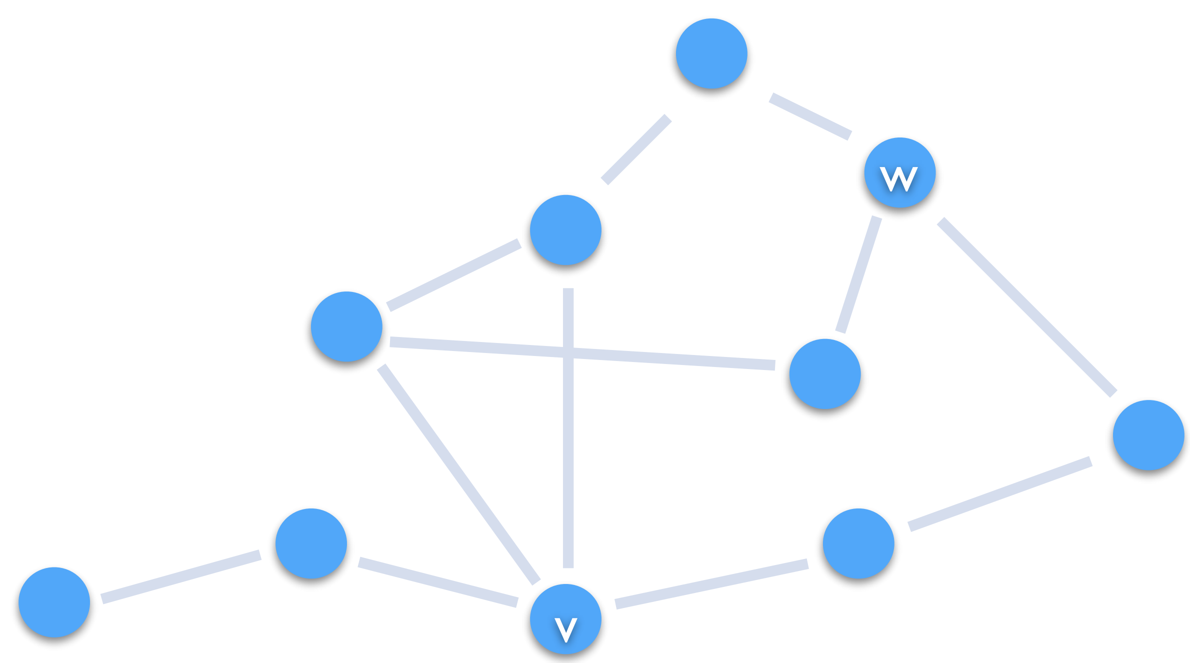
Social affinity

How can we compute it?



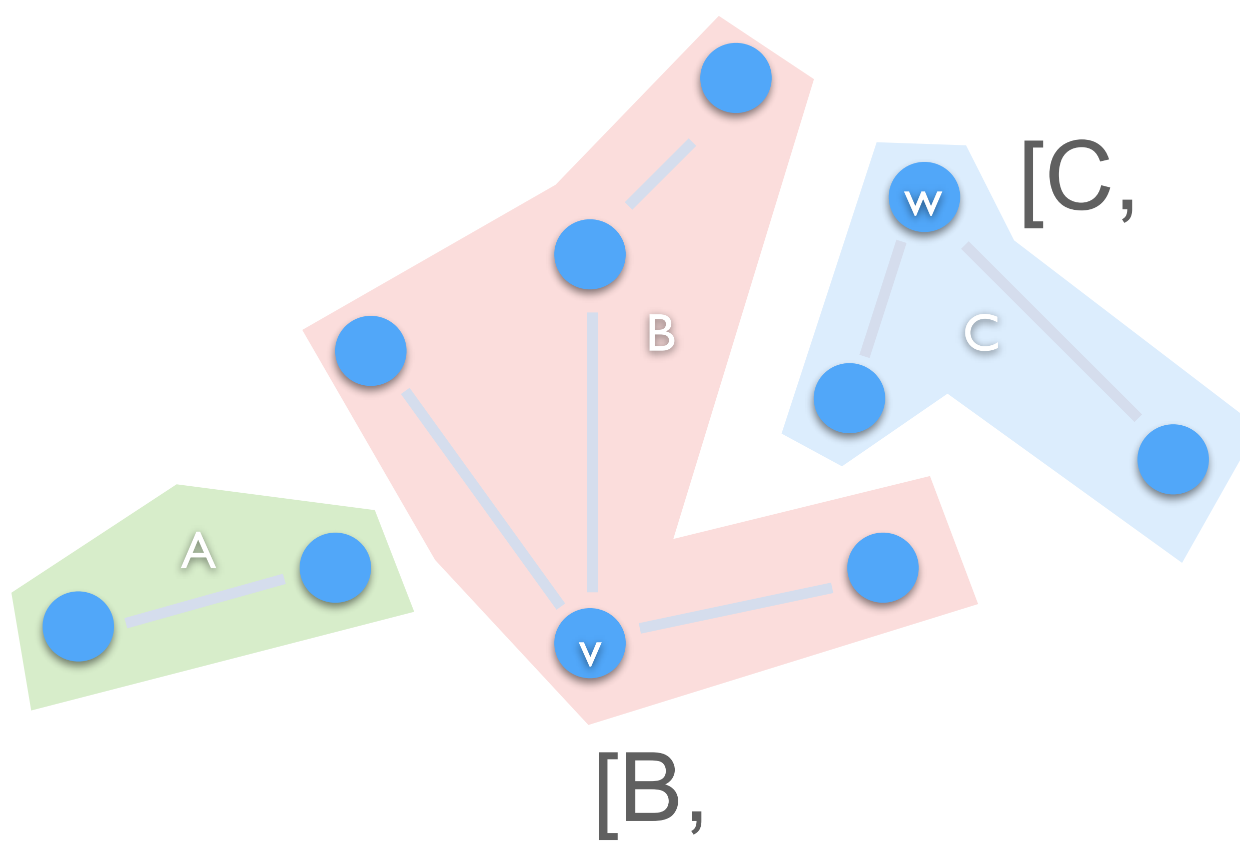
Social affinity

How can we compute it? [Panigrahy et al. WSDM12]
For each $p \in [0, 1+\epsilon, (1+\epsilon)^2, \ldots]$ for $\log n$ delete the edge in the graph with probability p. Store for each node the component ids



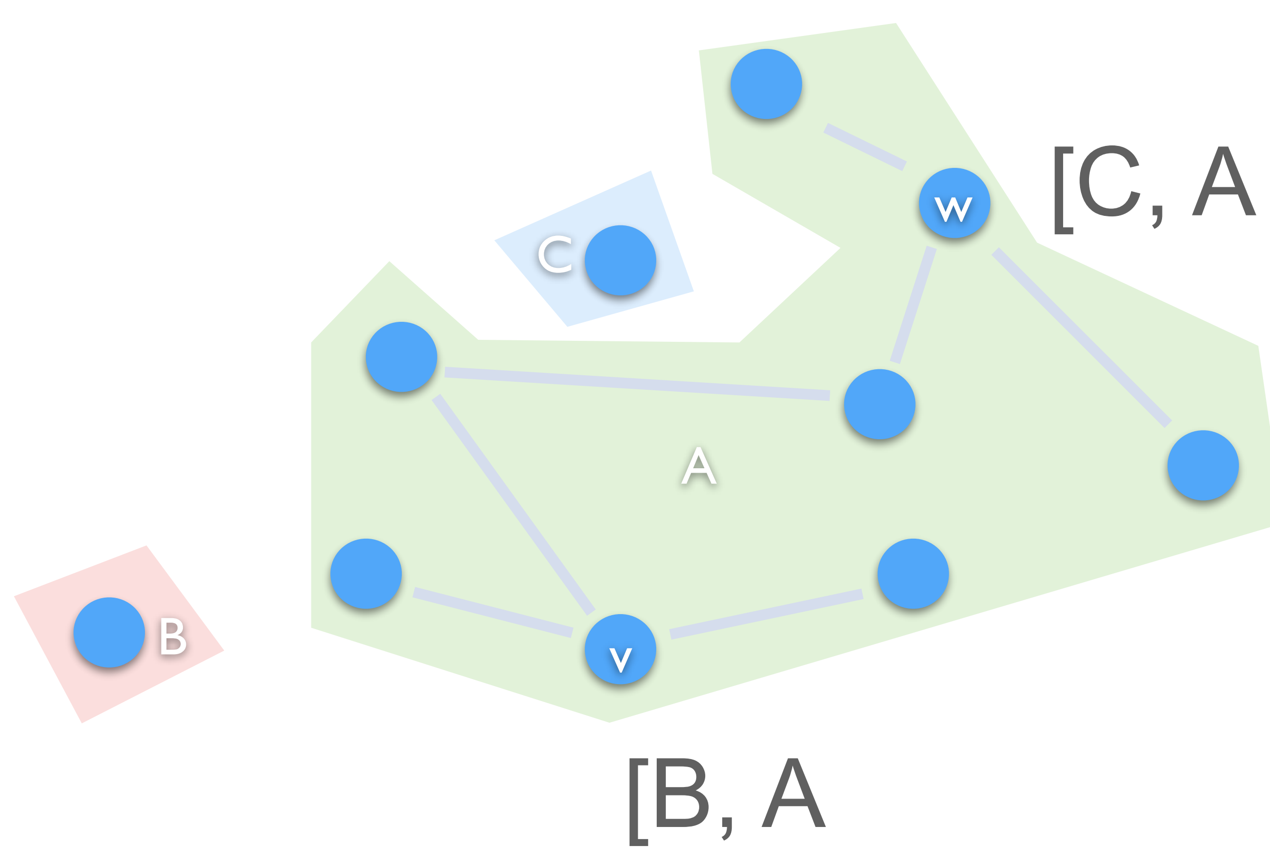
Social affinity

How can we compute it? [Panigrahy et al. WSDM12]
For each $p \in [0, 1+\epsilon, (1+\epsilon)^2, \ldots]$ for $\log n$ delete the edge in the graph with probability p. Store for each node the component ids



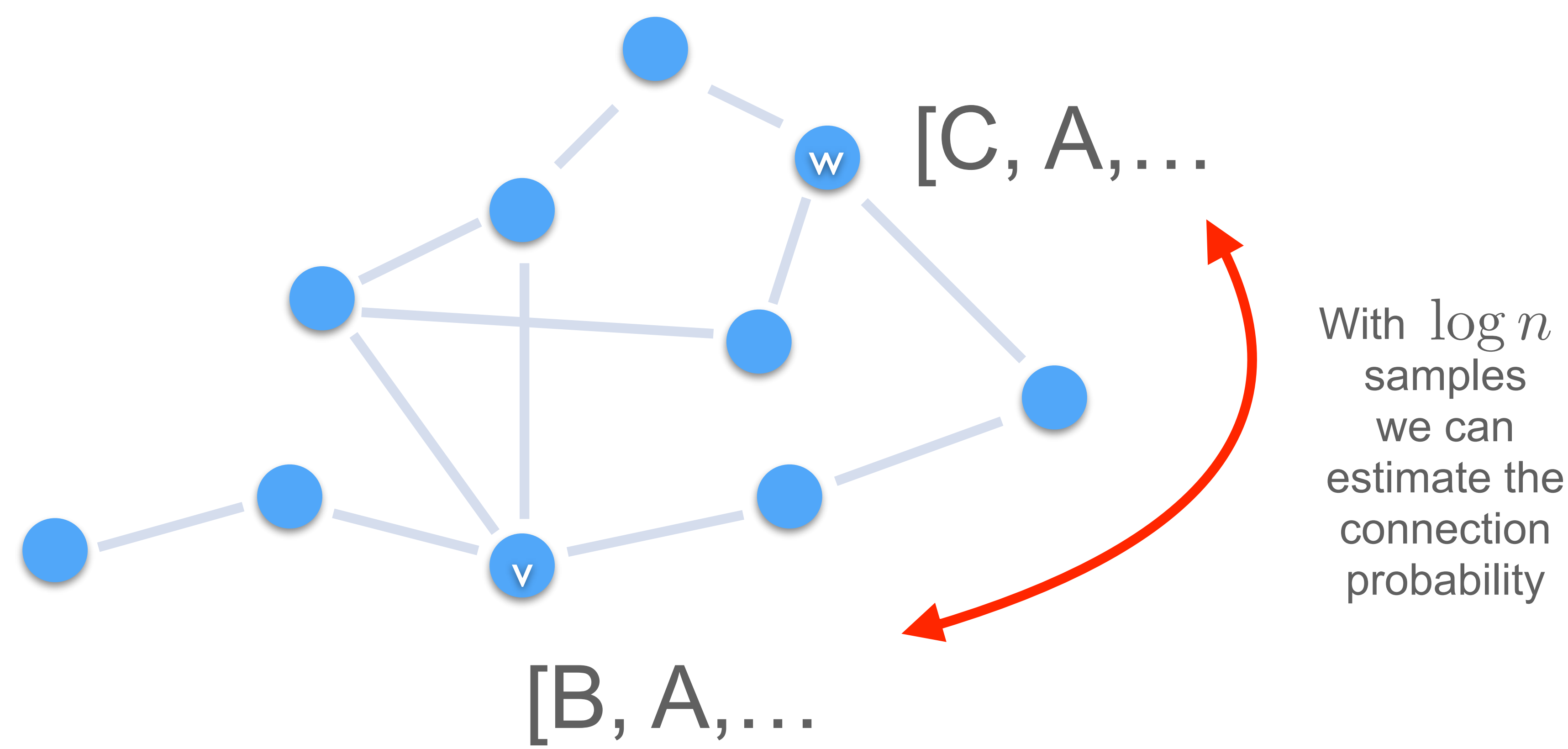
Social affinity

How can we compute it? [Panigrahy et al. WSDM12]
For each $p \in [0, 1+\epsilon, (1+\epsilon)^2, \ldots]$ for $\log n$ delete the edge in the graph with probability p. Store for each node the component ids



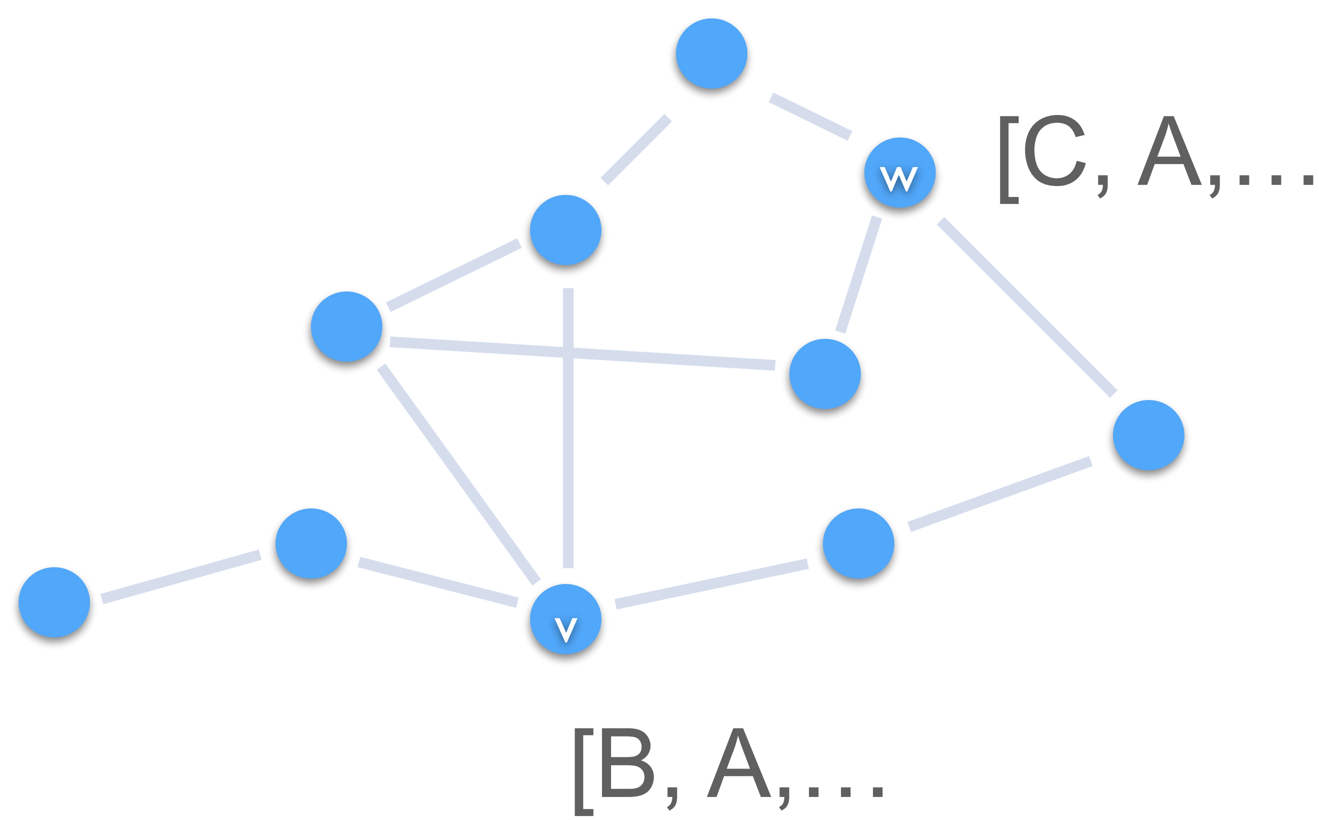
Social affinity

How can we compute it? [Panigrahy et al. WSDM12]
For each $p \in [0, 1+\epsilon, (1+\epsilon)^2, \ldots]$ for $\log n$ delete the edge in the graph with probability p. Store for each node the component ids



Social affinity

Using sketches of size $\log^2 n$ per node we can estimate affinity.



Social affinity

Using sketches of size $\log^2 n$ per node we can estimate social affinity.
When we add G_u we have to update the sketches, it is enough to update the connected components!