Courage
Inspiration
Trust
Youth
Uniqueness

Welcome to City University of Hong Kong!

SDSC 3019 Lecture 9:
P2P Architecture and Social Learning Network

Minghua Chen

https://www.mhchen.com

Announcement

- Project poster presentation in two weeks (Nov. 21) (20pts)
  - Submit a 2-minute video via Canvas by Nov 19, 8:59 am
  - Submit a poster via Canvas by Nov 19, 8:59 am
  - Arrangement to be announced on Canvas

- Project final report due on Dec. 2 (20 pts)
  - Up to 6 pages, IEEE Trans format

Outline

- Skype and BitTorrent: Peer-to-Peer Architectures
- Social Learning Networks: Optimizing interactions
- **Acknowledgement**: course materials are based on those shared by Prof. Brinton and Prof. Chiang at Purdue University

Skype and BitTorrent: Peer-to-Peer Architectures

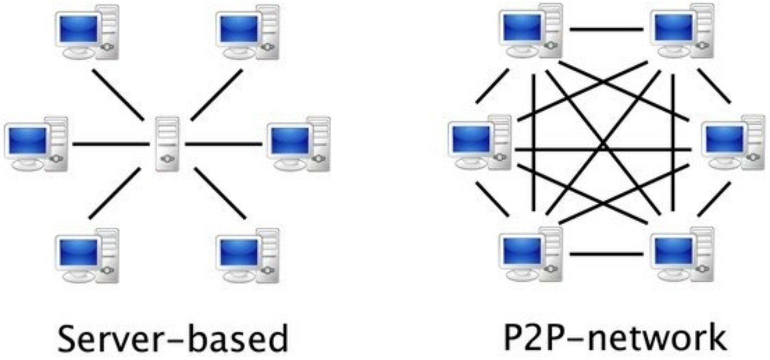
Reading: Ch. 15

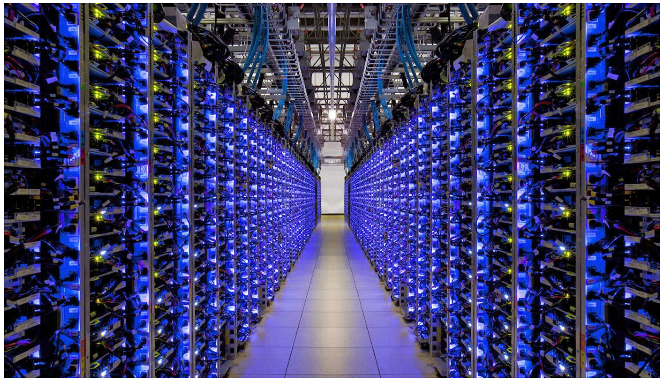
Outline

- □ The need for P2P applications
- □ P2P Skype and BitTorrent basics
- □ Overlay networks and tree graphs
- □ More on P2P Skype and BitTorrent
- □ Back-of-the-envelope download time
- □ Constructing multi-trees

Meeting challenges of scale

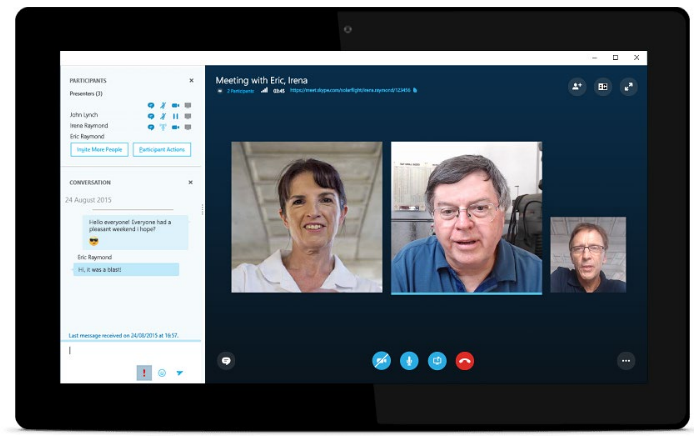
- Massive content distribution
  - Video and multimedia in particular
- Prevalent adoption of mobile wireless technologies
- How to scale up? Two approaches we will look at:
  - Help of peers: P2P (this lecture)
  - Using large data centers: Cloud (last lecture)
- Illustrates a key Internet principle: The power of overlay
  - Under-specify protocols governing the operation of a network
  - Overlay network can be readily built on top of it for future applications





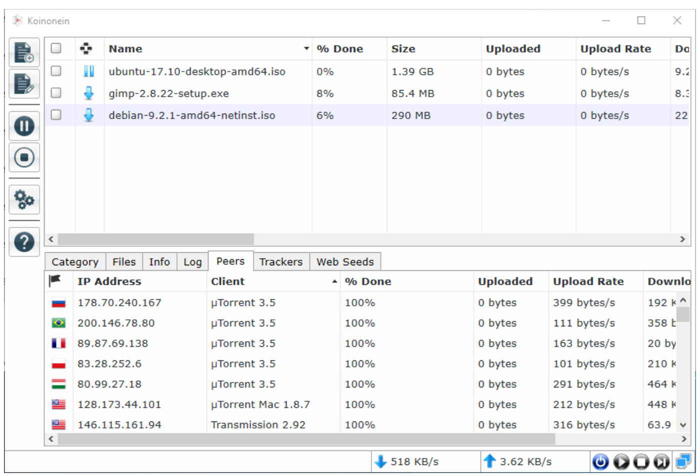
Peer-to-peer (P2P) applications

- Started becoming popular around 1999
  - Kazaa and Gnutella: Widely used P2P file- and music-sharing systems
  - Free riders: Consume much more than contribute



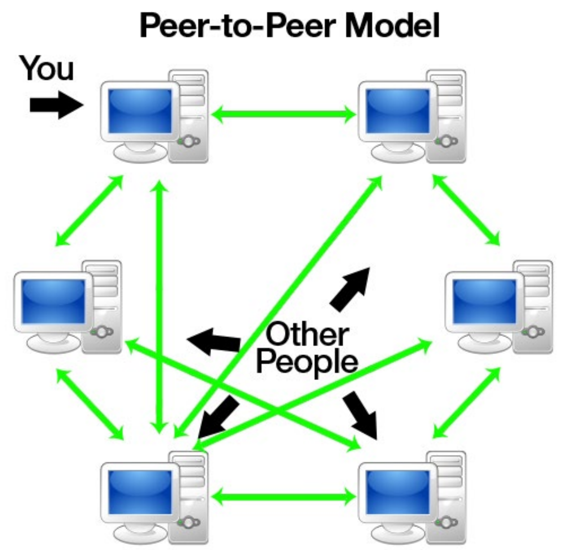
- Skype emerged in 2001
  - Bought by eBay in 2006 for $2.6B, then Microsoft in 2011 for $8B
  - 100M users worldwide as of 2020

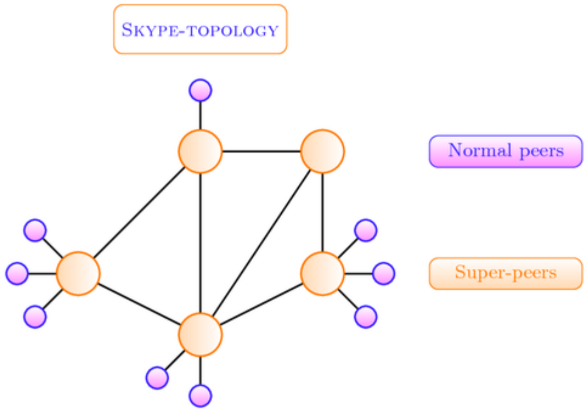
- BitTorrent also started in 2001
  - 45M daily active users as of 2016
  - 20-40% of Internet traffic in mid 2010s



Skype vs. BitTorrent

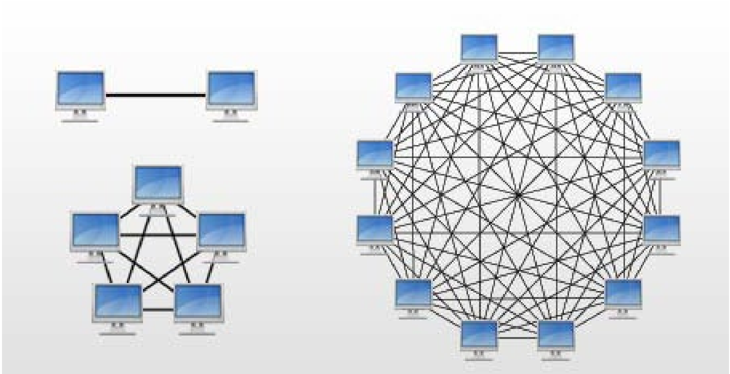
- Skype *used* P2P for signaling
  - Locate each other, establish connections
- BitTorrent uses P2P for content sharing
  - Share chunks of files, leverage peer uplink capacities
- But both are free and scalable
- Both illustrate a positive network effect
  - Each additional node contributes to many other nodes
  - Can add more nodes without creating a bottleneck



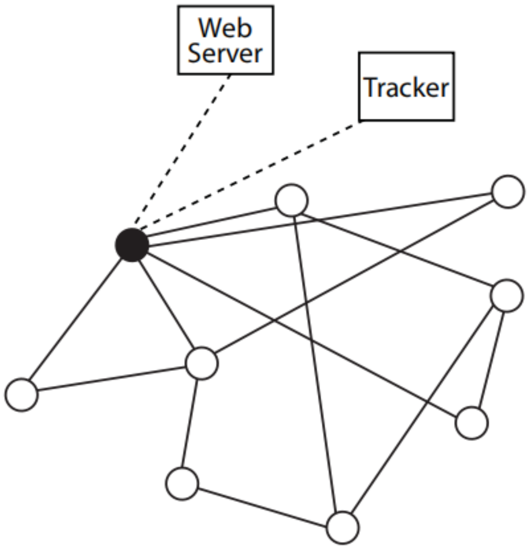


Positive network effect

- Assumes nodes can effectively contribute
  - Requires some smart engineering designs
- **Metcalfe's law**
  - Benefit grows as the square of the number of nodes
  - Assumes all connections are equally important
  - Assumes each node is basically connected to all other nodes
- **P2P systems**: practical realization of this intuition
  - Achieve benefits by carefully choosing neighbors (even constant #)



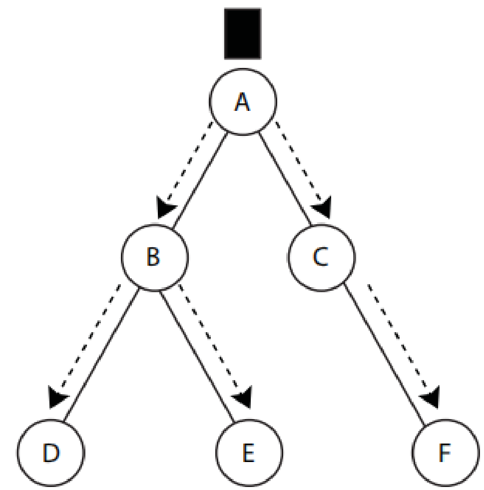
BitTorrent basics



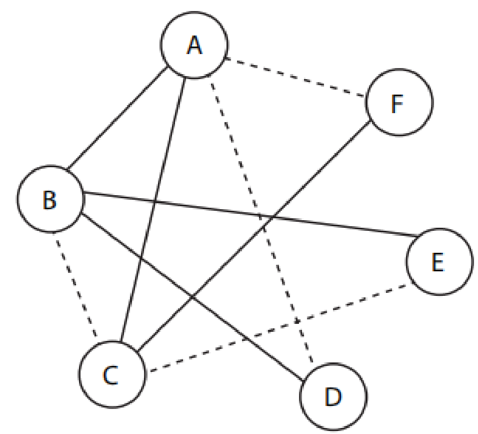
- Uses P2P for resource sharing
- Designed primarily for **multicasting**
  - Many users demanding the same file around the same time
  - Multicast vs. unicast
- Files divided into chunks, typically 256 kB
  - This way pieces can be shared
- Tracker: Centralized directory with peer-chunk database
  - Peer polls tracker for a set of 50 (or so) peers that has chunks it needs
  - Peer picks five of these 50 with which to exchange file chunks
- Peer refreshes its set of five peers at each timeslot
  - Partially based on which peers are helping the most
  - Partially based on randomization

Chunk vs. peering graph, client-server vs. P2P

- Each chunk traverses a tree of peers
  - Chunk is a rectangle, data transmission is dotted lines
  - A **tree** is an undirected graph with only one path from one node to any other node
  - Root node on top, leaf nodes on bottom
- Overall peering relationship is a general graph
  - Solid lines represent current peering relationships
  - Dotted lines represent potentials for the next time period
  - Evolves over time
- Data plane is distributed, P2P
  - In sharp contrast with the traditional **client-server** architecture



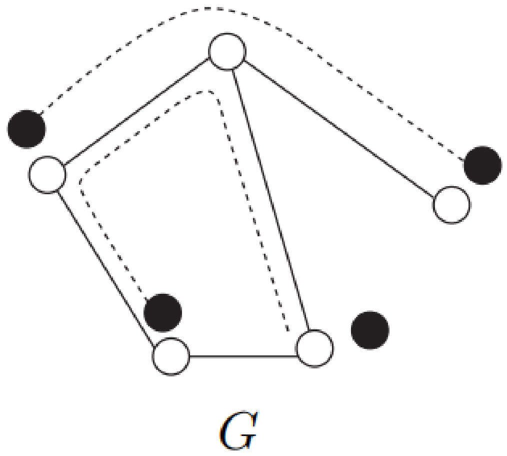
Chunk graph



Peer graph

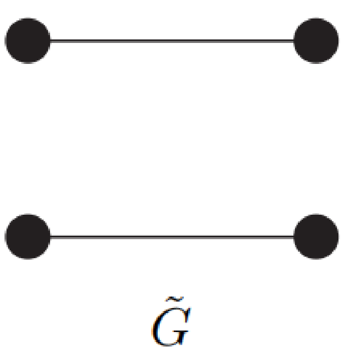
Overlay network

- P2P is an **overlay network** $\tilde{G} = (\tilde{V}, \tilde{E})$
  - Underlay graph $G = (V, E)$
  - $\tilde{V}$ is a subset of V
  - Links in $\tilde{E}$ are some of the path in E



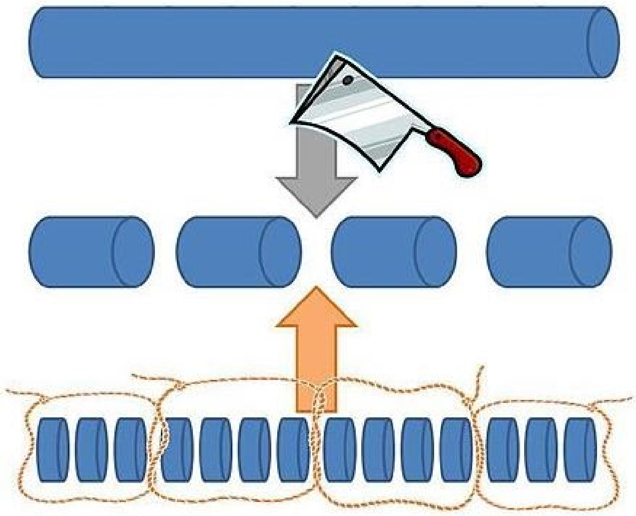
- Other overlays we've seen
  - The Internet is an overlay on top of physical networks
  - Online social networks are an overlay on top of the Internet

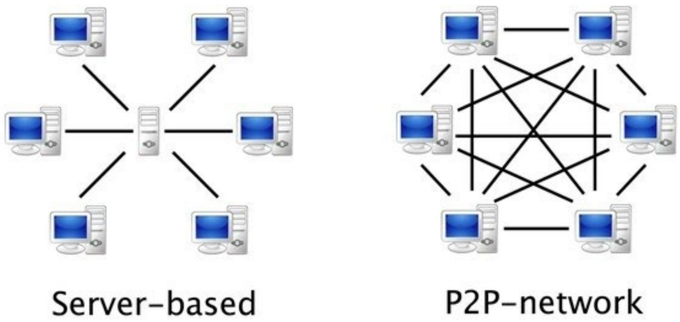
- The concept of an overlay is evolvable
  - Internet provides fundamental services
  - People can build overlay networks on top of existing ones
  - Example: IP multicast is rarely used; let IP take care of addressing, and instead use the P2P alternative with much less overhead



More on BitTorrent: Idea 1

- *Use smaller granularity than full files*
  - File chunks give more flexibility
- Transmission can be *spatially pipelined*
  - Each chunk can traverse a different multicast tree
  - Multi-tree transmission, multi-tree multicasting
- Compare this with packet switching
  - Divide messages into smaller granularity called packets
  - Multipath routing: Packets take possibly different paths
- "Peer": Nodes are both senders and receivers of content



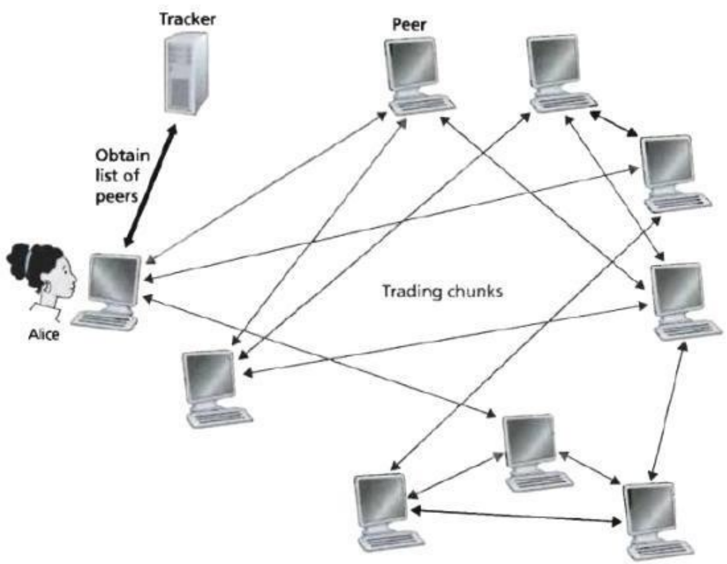


More on BitTorrent: Idea 2

- *Rarest chunk first strategy*
  - A peer should first download the chunks that very few peers have

- This optimizes content placement over different peers

- And equalizes the availability of chunks
  - Mitigates the problem where most of the peers have most of the chunks, but all must wait for the few rare chunks



More on BitTorrent: Idea 3

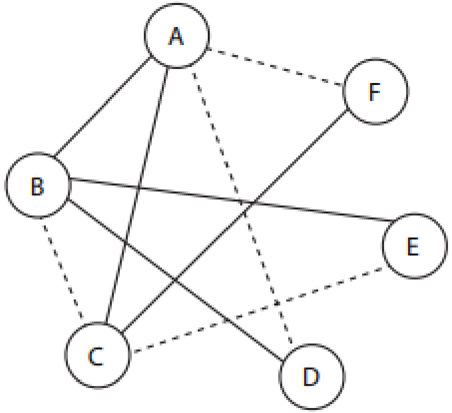
- *Peering construction method*

- First step: Tracker suggests 50-or-so potential peers to a new peer

- Recommendation based on:
  - The content they have
  - Performance-driven factors (e.g., distance to the new peer)
  - Peer churns: Which are still active

- Second step: Let the new peer pick, at each time, her actual "friends"
  - These are the peers to exchange chunks with

- Five peers are selected at each time slot:
  - Four are **tit-for-tat**: The top four peers in terms of the amount of content received
  - One is selected at random from the set of 50



Peer graph

Summary of BitTorrent Operation

① New peer A receives .torrent file from a web server

② Registers with tracker

③ Receives a list of neighbors

④ Selects five peers and establishes connections

⑤ Exchange bitmaps to indicate which chunks of content they have

⑥ With chunks selected, exchange chunks starting with the rarest ones

⑦ Every now and then, each peer updates its peer list

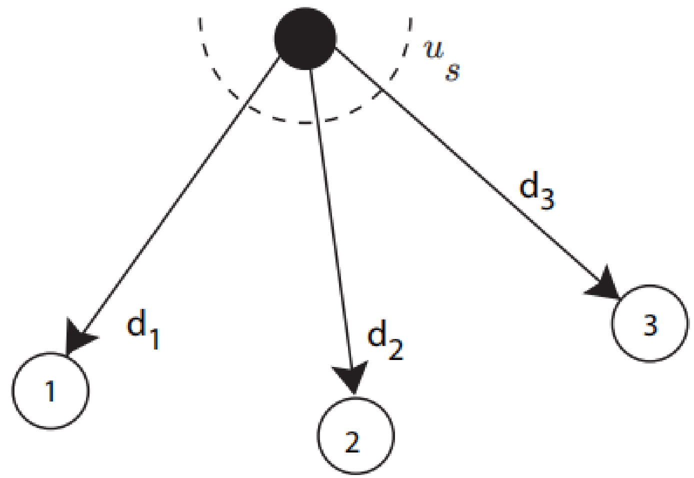
Client-server: Back-of-the-envelope calculation

- Consider N clients requesting a file of size F bits from a server
  - Server upload capacity u_s bps
  - Each of these clients has a download capacity of d_i bps

- How long will this take?
  - The server needs to send out NF bits, so it takes at least NF/us seconds
  - All the clients need to receive the file, including the slowest one with a download capacity of d_{min}, and that takes at least F/d_{min} seconds

  So the total download time is:

$$T = \max\left\{\frac{F}{d_{min}}, \frac{NF}{u_s}\right\}$$



- What happens as N becomes larger?
  - u_s cannot scale with N, especially as N becomes very large

P2P: Back-of-the-envelope calculation

- Suppose each peer i has an upload capacity u_i
  - In addition to a download capacity d_i as before
- Which tend to be larger: u_i or d_i?
- Suppose we construct multicast trees that can fully utilize all u_i
  - For total number of bits to be shared, NF, the total upload bandwidth available to the whole network is $u_s + \sum_{i=1}^{N} u_i$
  - So the time this takes is $NF/(u_s + \sum_{i=1}^{N} u_i)$ seconds
- We also need to wait for ...
  - The server to send out each bit at least once: F/us seconds
  - The slowest peer to receive its bits: F/d_{min} seconds
- Time to distribute through the network:

$$T = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

Back-of-the-envelope comparison

- Client-server:

$$T = \max\left\{\frac{F}{d_{min}}, \frac{NF}{u_s}\right\}$$

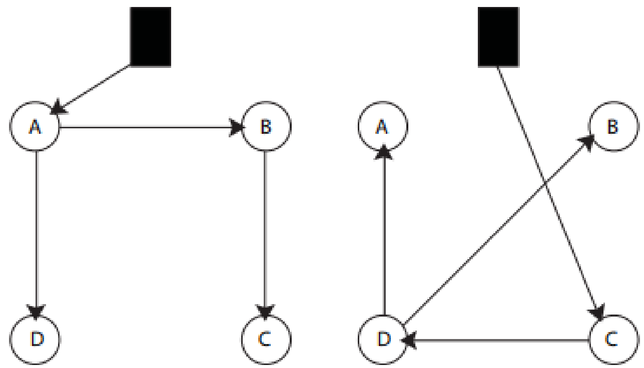
- P2P:

$$T = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

- In both cases, last term has numerator increasing with N
- But for P2P, the denominator also increases with N
  - So, provided that the u_i are sufficiently large, T itself will no longer increase in N
  - The network performance scales itself with the network size

Constructing trees

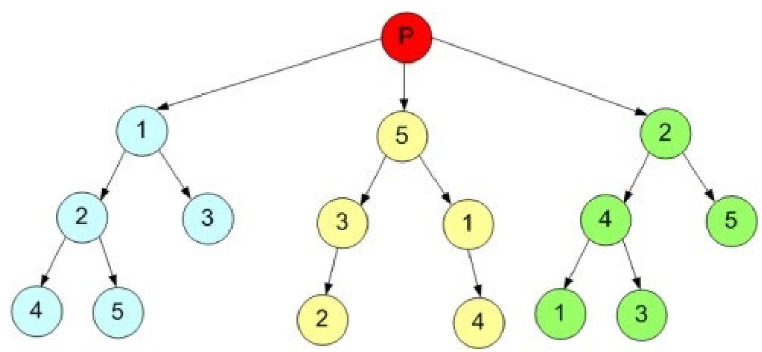
- Back-of-the-envelope makes a huge assumption
  - All upload capacities can be fully utilized
  - But this can be hard to do, especially with only one tree
  - Consider each tree on the right: Some capacity not used in each case



C and D wasted

A and B wasted

- **Multi-tree** construction of peering relationships
  - Use more than one distribution tree
  - Helps, but the best way to construct all the trees is still not clear
  - Basic idea: Peers with more leftover upload capacities should be placed higher up in the next constructed trees
  - Still a very hard combinatorial optimization problem



Special case: No download bottleneck

- Consider the P2P case again, and assume the d_i are large enough

- We want to show that our back-of-the-envelope result can be obtained:

$$T = \max\left\{\frac{F}{u_s}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

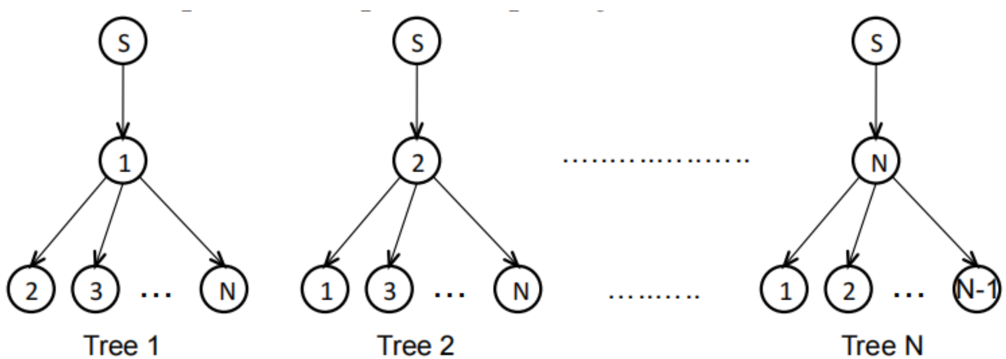
- To do this, we can construct a multi-tree that collectively achieve the corresponding desired rates:

$$r_{max} = \min\left\{u_s, \frac{u_s + \sum_{i=1}^{N} u_i}{N}\right\}$$

- We break this into two cases:
  ① $u_s \leq (us + \sum_{i=1}^{N} u_i)/N$
  ② $u_s > (us + \sum_{i=1}^{N} u_i)/N$

Case I: $u_s \leq (u_s + \sum_{i=1}^{N} u_i)/N$

- The maximum broadcast rate $r_{max} = u_s$ should be supported
  - Server upload capacity is too small
- Consider the following multi-tree of N trees, each two hops
  - Tree i has s take i as its child, and i take the other $N-1$ peers as its children
  - Should deplete all upload capacity u_s



- Let each tree carry an upload capacity proportional to u_i:

$$r_i = \left(\frac{u_i}{\sum_{j=1}^{N} u_j} \right) u_s, \quad i = 1, \ldots, N$$

Case I: $u_s \leq (u_s + \sum_{i=1}^{N} u_i)/N$

- This is possible because the upload speeds required are within capacity constraint. For the server,

$$\sum_{i=1}^{N} r_i = u_s$$

and for peer i,

$$\begin{aligned}(N-1)r_i &= (N-1)\frac{u_i}{\sum_{j=1}^{N} u_j} u_s = \frac{u_i}{\sum_{j=1}^{N} u_j}(Nu_s - u_s) \\ &\leq \frac{u_i}{\sum_{j=1}^{N} u_j}\left(u_s + \sum_{j=1}^{N} u_j - u_s\right) = u_i\end{aligned}$$

- The aggregate broadcast at which any peer i receives is then

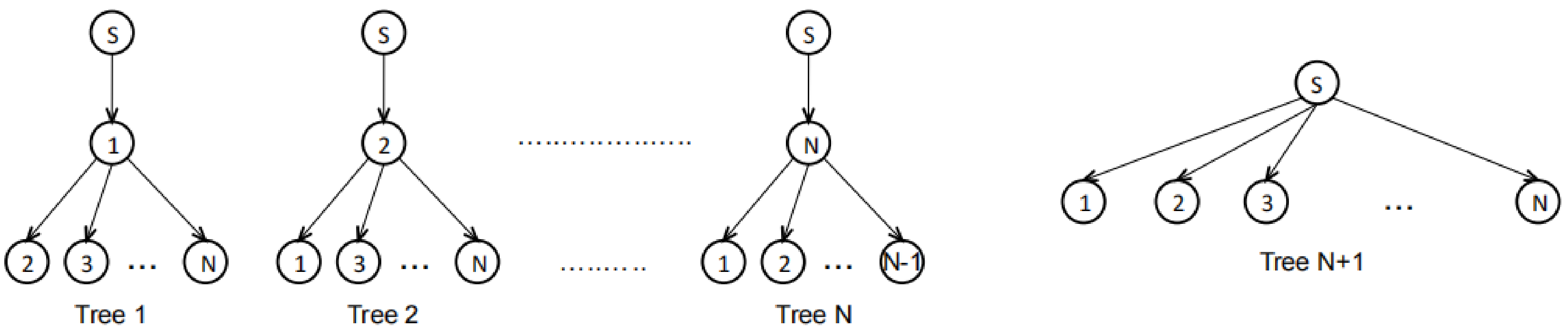
$$r_{max} = r_i + \sum_{j \neq i} r_j = \sum_i r_i = u_s$$

- Hence, it takes F/us seconds to transfer the whole file

Case II: $u_s > (u_s + \sum_{i=1}^{N} u_i)/N$

- In this case, the maximum broadcast rate $r_{max} = (us + \sum_{i=1}^{N} u_i)/N$ should be supported
  - Server upload capacity is large enough for a different set of trees, including a server-client tree

- Consider the following multi-tree of $N+1$ trees, N two hops and 1 one hop:



- Assume the trees carry the following rates:

$$r_i = \frac{u_i}{N-1} \quad i = 1, \ldots, N \qquad r_{N+1} = \frac{u_s - \frac{\sum_{i=1}^{N} u_i}{N-1}}{N}$$

meaning the last tree gets the leftover uplink capacity from the server

Case II: $u_s > (u_s + \sum_{i=1}^{N} u_i)/N$

- These upload speeds required are within constraint. For peer i,

$$(N-1)r_i = u_i$$

and for the sever,

$$\begin{aligned}\sum_{i=1}^{N} r_i + N \cdot r_{N+1} &= \sum_{i=1}^{N} \frac{u_i}{N-1} + N \cdot \frac{u_s - \frac{\sum_{i=1}^{N} u_i}{N-1}}{N} \\ &= \sum_{i=1}^{N} \frac{u_i}{N-1} + u_s - \frac{\sum_{i=1}^{N} u_i}{N-1} = u_s\end{aligned}$$

- The aggregate broadcast at which any peer i receives is then

$$\begin{aligned}r_{max} &= r_i + r_{N+1} + \sum_{j \neq i} r_j = \frac{u_i}{N-1} + \frac{u_s - \frac{\sum_{i=1}^{N} u_i}{N-1}}{N} + \sum_{j \neq i} \frac{u_j}{N-1} \\ &= \frac{N \sum_i u_i}{N(N-1)} + \frac{u_s}{N} - \frac{\sum_{i=1}^{N} u_i}{N(N-1)} = \frac{u_s + \sum_i u_i}{N}\end{aligned}$$

Case II: $u_s > (u_s + \sum_{i=1}^{N} u_i)/N$

- Hence, it takes $NF/(us\ + \sum_{i=1}^{N} u_i)$ seconds to transfer the whole file

- Combining the two cases above produces our desired results

Outline

- □ Online learning
- □ Social Learning Networks (SLN)
- □ Learner benefit in an SLN
- □ SLN utility maximization
- □ Numerical example
- □ Summary

Distance learning

- Mid-1800s: Degree programs operating by postal mail

- Early-to-mid-1900s: Course lecture broadcasts over radio and television

- 1990s: Online programs, degrees, and universities
  - Rise of the web
  - By 2003: 80% of colleges had at least one class using eLearning technology
  - By 2014: More than 95% of public colleges offered some form of online program

- **Learning modes** offered by different technologies:

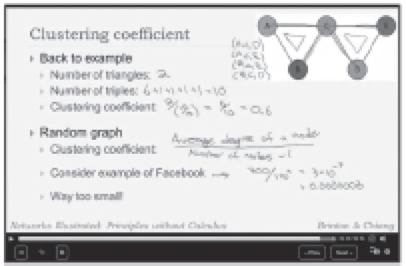
	Auditory	Visual	Textual	Social	Synchronous
In Class	✔	✔	✔	✔	✔
Mail		✔	✔		
Radio	✔				
Television	✔	✔			
Internet	✔	✔	✔	✔	

Massive Open Online Courses (MOOCs)

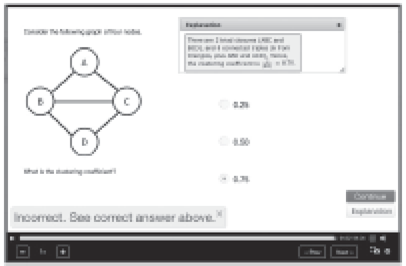
- 2002: MIT creates an online resource of free course materials
  - Anyone can access
  - Why would MIT do this?

- Became a pathfinder for **MOOC**
  - Transform classroom-sized online courses into MOOCs
  - Coursera, edX, Udacity, Udemy, ...
  - Partnership with top universities

- MOOCs are ...
  - Massive: Up to hundreds of thousands per session
  - Open: Typically free or very cheap
  - Online
  - Course: Typically full course delivery



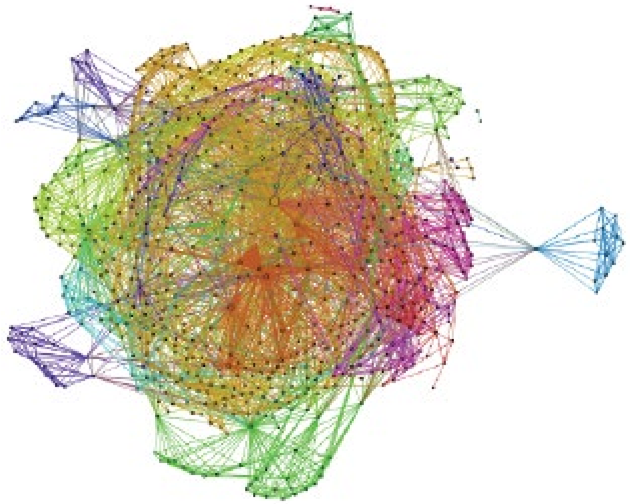
Lecture video



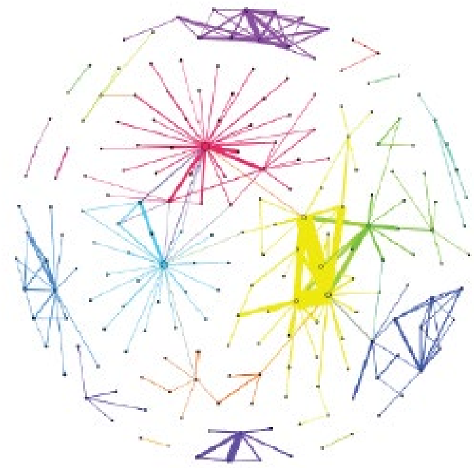
In-video quiz

Social Learning Networks (SLN)

- Networks of ...
  - Students
  - Instructors
  - Modules of learning



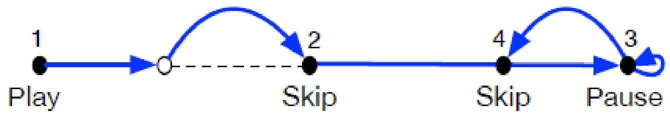
- One of the scalable aspect of MOOCs
  - Address diverse sets of learning needs
  - Interesting spin on how to achieve individualization
- Three aspects
  - *Social* : Hinges on interaction between peers
  - *Network*: The topology (student-to-student)
  - *Learning*: The functionality



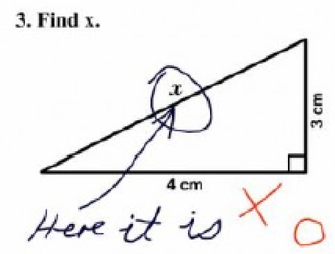
- How to study and optimize SLN?
  - Massive amounts of data collected

Data Sources in eLearning

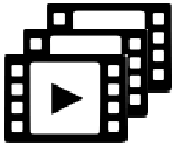
- At least four groups of data:



- Clickstream measurements
  - Learners watching video lectures



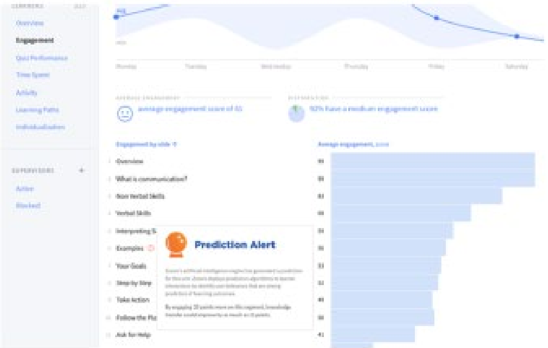
- Social interactions
  - Learners on discussion forums



- Question responses
  - How knowledge transfer is normally assessed
- Content (and user) words
  - Topic-specific learning
- How to use these to analyze and optimize SLN?
  - Methodologies at the intersection of data science, networking, and learning sciences

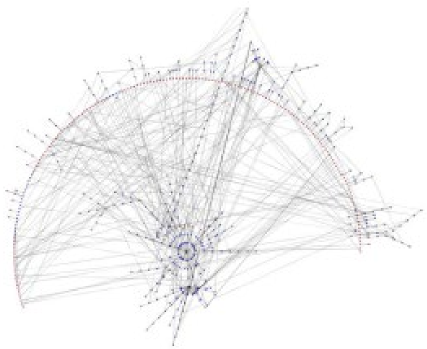
Zoomi: Artificial Intelligence for Learning

- Predictive Learning Analytics (PLA)
  - Fine granular data analysis
  - Behavior-based early detection outcome prediction
  - Behavioral pattern identification



- Network Efficiency Optimization (NEO)
  - Predicting link formation over time
  - Optimizing interaction structures (our focus in this lecture)
  - Comparing SLN and course learning outcomes

- Deep Learning Personalization (DLP)
  - Autonomous creation of remediation content
  - Reinforcement learning-based personalization algorithms

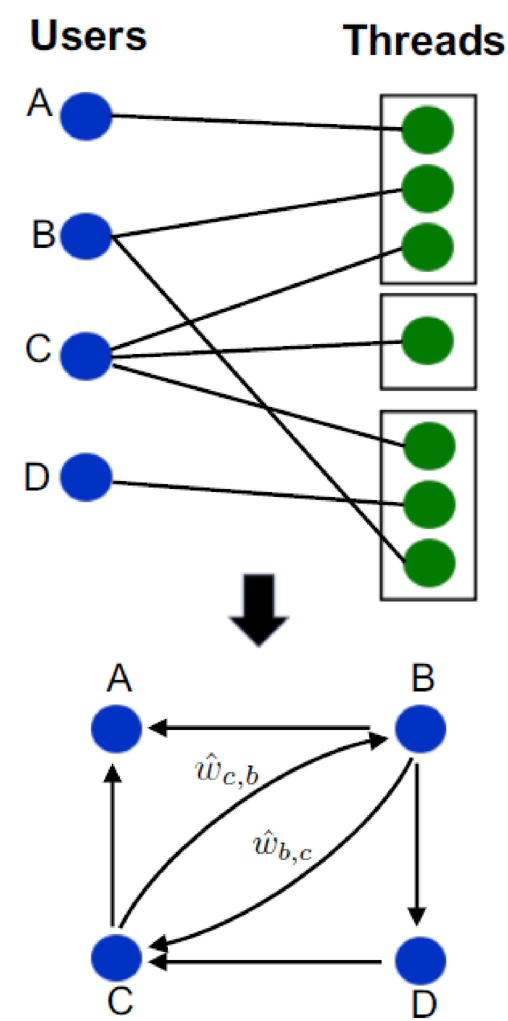


Graph of an SLN

- Nodes: Learners sharing information
  - Fine granular data analysis
  - Index users by u ∈ U, the set of all users

- Links: Interaction structure between users
  - Define $\boldsymbol{W}$ = $[\mathrm{w}_{u,v}]$ as the weighted adjacency matrix
  - $w_{u,v}$ represents spread of information from u to v

- How to dene/interpret $w_{u,v}$?
  - Several possibilities
  - One useful designation: Consider $w_{u,v}$ ∈ [0,1] as the probability that u responds when v posts
  - **W is asymmetric**, i.e., $w_{u,v} \neq w_{v,u}$ in general

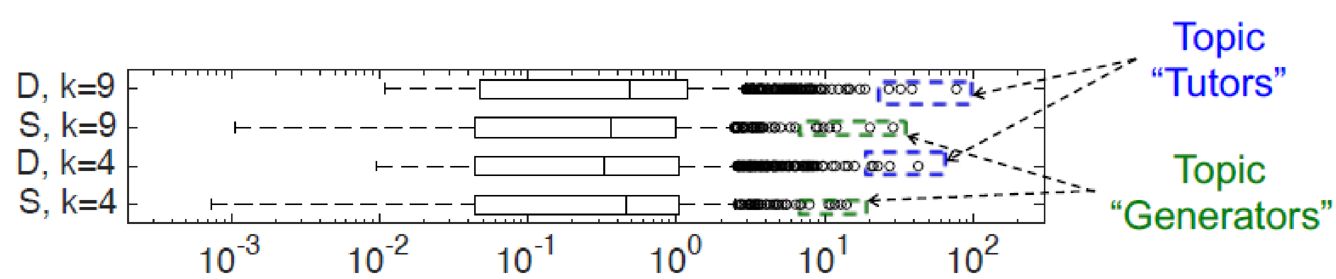


Modeling SLN Discussions

- Topics: Focal points of discussions
  - These can be latent dimensions
  - Index by $k \in K$, the set of discussion topics

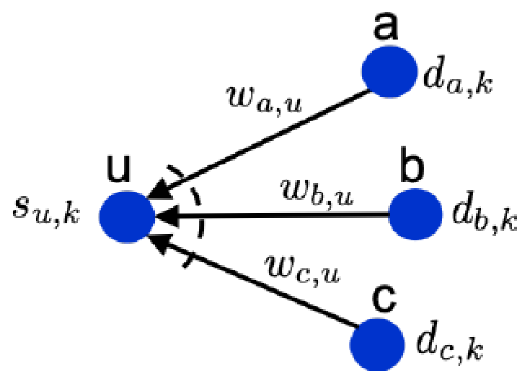
- Seeking: Tendency to ask questions
  - Topic specific Let S = $[s_{u,k}]$ be the matrix of seeking tendencies, with $s_{u,k} \geq 0$ being u 's seeking tendency on topic k

- Disseminating: Tendency to provide answers/facts about the material
  - Similarly topic specific
  - Let D = $[d_{u,k}]$ be the matrix of disseminating tendencies, with $d_{u,k} \geq 0$ being u's disseminating tendency on topic k

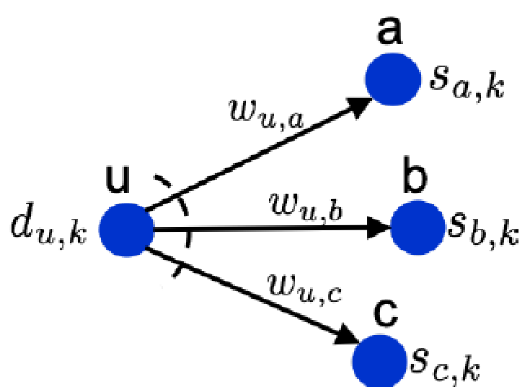


Modeling User Benefits

- Learning benefit: Incoming weights on topics with high seeking tendency
  - We quantify this as: $s_{u,k} \cdot f(\sum_v w_{u,v} d_{v,k})$
  - $w_{u,v} d_{v,k}$ is the expected amount of response from v to u on topic k
  - f is a concave function: Diminishing marginal returns
  - Weighted by seeking tendency $s_{u,k}$



- Teaching benefit: Outgoing weights on topics with high disseminating tendency
  - Similarly, this can be quantified as: $d_{u,k} \cdot f(\sum_v w_{u,v} s_{v,k})$
  - $w_{u,v} s_{v,k}$ is the amount by which u provides information to v sought on topic k
  - f is a concave function
  - Weighted by $d_{u,k}$



Quantifying Utility

- User-topic benefits: $b_{u,k} \geq 0$ is the total utility obtained by u with respect to k, quantified as

$$b_{u,k} = s_{u,k} \log(1 + \sum_v w_{v,u} d_{v,k}) + \alpha_u \cdot d_{u,k} \log(1 + \sum_v w_{u,v} s_{v,k}),$$

In matrix form, **B** = $[b_{u,k}]$

- α_u quantifies the benefit of teaching relative to learning for user u
- Choose f(x) = log(1 + x) as a standard concave function

- Seeking to Incoming Disseminating Ratio (SIDR):

$$\phi_{u,k} = \frac{s_{u,k}}{\sum_v w_{v,u} d_{v,k}}, \qquad \Phi = [\phi_{u,k}]$$

- Lower makes user's learning benefit in $b_{u,k}$ higher
- More concretely, smaller SIDR $\phi_{u,k}$ implies that u 's seeking tendency on topic k has higher satisfaction from the incoming disseminating tendencies of their neighbors

Quantifying Utility

- Disseminating to Incoming Seeking Ratio (DISR):

$$\psi_{u,k} = \frac{d_{u,k}}{\sum_v w_{u,v} s_{v,k}}, \qquad \Psi = [\psi_{u,k}]$$

  – Lower makes user's teaching benefit in $b_{u,k}$ higher
  – A smaller DISR $\psi_{u,k}$ implies that u 's disseminating tendency on k is being used to satisfy more of their neighbor's seeking tendency

- Local utility (total benefit obtained by u across k):

$$l_u = \sum_k b_{u,k}$$

- Global utility (average local utility across users):

$$g_{\alpha_u} = \frac{1}{|\mathcal{U}|} \sum_{u,k} b_{u,k}$$

- Can we maximize g without sacrificing l?, by tuning

Optimizing the SLN

- Formulate as a Network Utility Maximization (NUM) problem:

$$\begin{array}{ll} \underset{\mathbf{W}}{\text{maximize}} & g(\mathbf{W}) \\ \text{subject to} & \Phi(\mathbf{W}) \leq C_s \hat{\Phi} \\ & \Psi(\mathbf{W}) \geq C_d \hat{\Psi} \\ & \mathbf{0} \leq \mathbf{W} \leq \mathbf{1}, \ \operatorname{diag}(\mathbf{W}) = \mathbf{0} \\ \text{variables} & \mathbf{W} \end{array}$$

First two constraints:

- Preserving incoming information: $\Phi(W) \leq C_s \widehat{\Phi}$
  - $\widehat{\Phi}$ = $[\hat{\phi}_{u,k}]$ is the matrix of observed SIDR from the observed $\widehat{\mathbf{W}}$
  - $C_s > 0$ is a parameter: If C_s <1, we require a better SIDR than before, while if $C_s < 1$, we provide some slack to the feasible set
- Balancing load: $\Psi(W) \geq C_d \ \widehat{\Psi}$
  - $\widehat{\Psi} = \hat{\psi}_{u,k}$ is the matrix of observed SIDR
  - $C_d > 0$: If $C_d > 1$, we require users to participate more, while if $C_d < 1$, we allow any particular DISR to drop if needed

Quantifying the Impact of Optimization

- How to quantify the effect of optimization?
  - Several dimensions
  - Can boil down to a comparison of network **efficiency** and **fairness**

- Efficiency: Fraction of optimal global utility achieved in the observed network, i.e.,

$$\eta_{\alpha_u, C_s, C_d} = g_{\alpha_u}(\hat{W}) / g^{\star}_{\alpha_u, C_s, C_d}$$

- Fairness: Ratio of fairness after vs. before, i.e.,

$$\epsilon_{\alpha_u, C_s, C_d} = F\left(\hat{\mathbf{l}}_{\alpha_u}(\hat{W})\right) / F\left(\mathbf{l}^{\star}_{\alpha_u, C_s, C_d}\right)$$

  - $\boldsymbol{l} = (l_1, l_2, \cdots,)$ is the vector of local utilities, $\hat{\boldsymbol{l}}$ is before and $\boldsymbol{l}^{*}$is after optimization
  - F() is a function quantifying fairness of a distribution, e.g., α-fairness (from data pricing) or Jain's Index

Inference and Optimization Algorithms

- Quantifying the observed network $\widehat{\boldsymbol{W}}$
  - If v posts in a thread, what is the probability u will post after that?

$$\hat{w}_{u,v}(\sigma) = \frac{n_{u,v} + \sigma(\sum_j n_{u,j} / \sum_j N_j) N_{max}}{N_v + \sigma N_{max}}$$

  - $n_{u,v}$ is the number of times that u posts after v, N_v is the total number of times v posted
  - σ is a Bayesian adjustment parameter because different users post different amounts (remember the product rating lecture)

- Inferring seeking and disseminating tendencies
  - Obtain post-topic distributions through e.g., Latent Dirichlet Allocation, a common topic modeling prodecure
  - Label posts as question or answer through e.g., rule-based information retrieval methods

- Solving for the optimal network $\boldsymbol{W}^*$?
  - Convex optimization problem (convex objective, linear constraints)
  - But it is a very big problem: W can have millions of entries

Numerical Example: Dataset

- Give $\widehat{W}$, S, and D matrices for a toy SLN with |U|= 10 users and |K|= 10 topics, we would have to compute the $s_{u,k}$ and $d_{u,k}$ based on e.g., the discussion forum structure.

- Here are some of the matrix values:

$$\hat{\mathbf{W}} = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.1 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.1 & 1.0 & 0.0 & 0.0 & \cdots \\ 0.1 & 1.0 & 0.166 & 0.0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.854 & 1.192 & 0.390 & 0.116 & \cdots \\ 1.542 & 0.042 & 0.860 & 0.042 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 5.538 & 2.821 & 0.691 & 2.429 & \cdots \\ 1.754 & 0.529 & 0.046 & 0.046 & \cdots \\ 0.798 & 0.107 & 0.794 & 0.107 & \cdots \\ 0.322 & 0.041 & 0.322 & 0.041 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Numerical Example: DISR and SIDR

- First, we calculate the DISR ψ and SIDR ϕ values. For example, $\phi_{4,1}$

$$
\begin{aligned}
\hat{\psi}_{4,1} &= \frac{d_{4,1}}{\sum_v w_{4,v} s_{v,1}} = \frac{0.322}{0.1 \times 0.0 + 1.0 \times 0.0 + 0.166 \times 0.854 + \cdots} \\
&= 2.26
\end{aligned}
$$

- So user u = 4 has about twice as much dissemination on topic k = 1 than what is being sought by her neighbors. As another example, $\phi_{3:4}$ is:

$$
\begin{aligned}
\hat{\phi}_{3,4} &= \frac{s_{3,4}}{\sum_v w_{v,3} d_{v,4}} = \frac{0.116}{0.0 \times 2.429 + 0.0 \times 0.046 + 0.0 \times 0.107 + \cdots} \\
&= 0.0097
\end{aligned}
$$

- So user u = 3 has substantially more incoming dissemination on topic k = 4 than what she is asking for.

Numerical Example: DISR and SIDR

- Continuing for all (u; k) pairs, the observed DISR and SIDR matrices are ($\phi_{\mathrm{u},k} = \infty$ means $d_{u,k} > 0$ while $\sum_v w_{u,v} s_{v,k} = 0$):

$$\hat{\Psi} = \begin{bmatrix} 146.9 & 73.5 & 7.8 & 49.6 & \cdots \\ \infty & \infty & \infty & \infty & \cdots \\ 13.5 & 0.9 & 13.6 & 1.2 & \ldots \\ 2.2 & 0.2 & 4.9 & 2.1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad \hat{\Phi} = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.04 & 0.10 & 0.010 & 0.009 & \cdots \\ 0.06 & 0.003 & 0.020 & 0.003 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

- We can see, for example:
  - users 1 and 2 are not asking questions ($\hat{\phi}_{1,k}, \hat{\phi}_{2,k} = 0$) but have a substantial amount of dissemination to offer for those they are not connected to ($\hat{\psi}_{1,k}, \hat{\psi}_{2,k} \gg 0$ while $\widehat{w}_{1,v}, \widehat{w}_{2,\mathrm{v}} = 0$ for most v)
  - most users have substantial incoming dissemination ($\hat{\phi}_{\mathrm{u},k} \ll 1$) based on the amount of questions they are asking

Numerical Example: Benefits and Utilities

- We can also calculate the observed benefits $\hat{b}_{u,k}$, local utilities $\hat{l}_u$, and global utility $\hat{g}$(the u for each user are provided with the CSVs as well). For user 3 on topic 1, the benefit is

$$\begin{aligned} b_{3,1} &= s_{3,1}\log(1+\sum_v w_{v,3}d_{v,1}) + \alpha_u \cdot d_{3,1}\log(1+\sum_v w_{3,v}s_{v,1}) \\ &= 0.85431 \cdot \log(1+(0\times 5.53 + 0\times 1.75 + 0\times 0.798 + \cdots)) \\ &\quad + 0.03421 \cdot 0.7983 \cdot \log(1+(0.1\times 0 + 1\times 0 + \cdots)) = 1.1380 \end{aligned}$$

with u = 0:03421. The local utility for this user is

$$\hat{l}_3 = b_{3,1} + b_{3,2} + b_{3,3} + \cdots = 10.3716$$

Repeating this for every user, the local utilities are

$$\hat{\mathrm{l}} = (0.09, 0, 4.27, 10.37, 20.78, 0.06, 0.009, 17.93, 0.010, 0.077)$$

So users 4, 5, and 8 have the most utility. The global utility is

$$\hat{g} = (1/10)\sum \hat{l}_u = 5.3596$$

Numerical Example: Benefits and Utilities

- An optimal solution **W**? for this problem is provided in the CSVs also. A few of the entries are as follows:

$$\mathbf{W}^\star = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.053 & 0.0 & 0.0 & 0.0 & \cdots \\ 0.079 & 0.956 & 0.0 & 0.00061 & \cdots \\ 0.068 & 1.0 & 0.166 & 0.0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

- Compared with $\widehat{\boldsymbol{W}}$, for example, we see that $w^*_{2,1} \approx \widehat{w}_{2,1}$, i.e., user 2 should respond to user 1 with about half the frequency. Also, while, $\widehat{w}_{3,4} = 0, w^*_{3,4} > 0$ i.e., user 3 should respond to user 4's questions (albeit infrequently).

- With this, we can find the optimal local and global utilities to be:

$$\mathbf{l}^\star = (0.12, 0.0003, 4.28, 10.95, 24.16, 1.79, 0.009, 19.66, 0.013, 0.095)$$

and $g^* = 6.11$.

Numerical Example: Comparing Observed and Optimal

- Comparing $\hat{g}$ to g^*, the efficiency is

$$\eta = 5.36/6.11 = 0.88,$$

i.e., there is about 12% improvement under optimization. To compare $\hat{\boldsymbol{l}}$ to $\boldsymbol{l}^*$,we need to choose an efficiency measure. For Jain's Index, the fairness of a vector $x = (x_1, x_2, \cdots, x_n)$is

$$F(\mathbf{x}) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}$$

- With this, F($\hat{\boldsymbol{l}}$) = 0:327 and F($\boldsymbol{l}^*$)= 0:336, which gives a fairness ratio of

$$\epsilon = 0.327/0.336 = 0.97$$

- So, the fairness actually improves (albeit slightly) under optimization.

End-of-Lecture Quiz

and

Weekly Anonymous Survey

closed book, closed notes, and no discussion

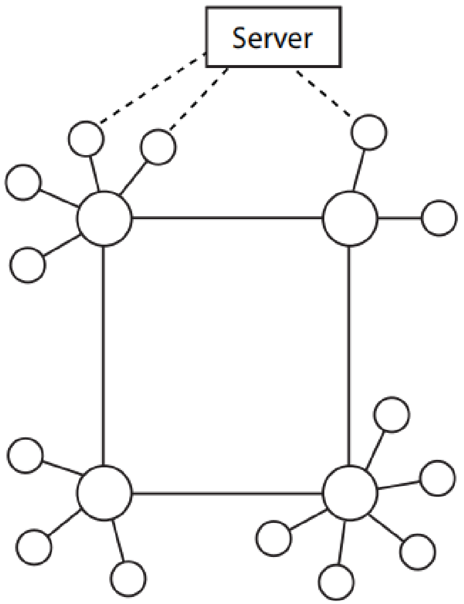
Thank you!

Minghua Chen (minghua.chen@cityu.edu.hk)

http://personal.cityu.edu.hk/~mchen88/

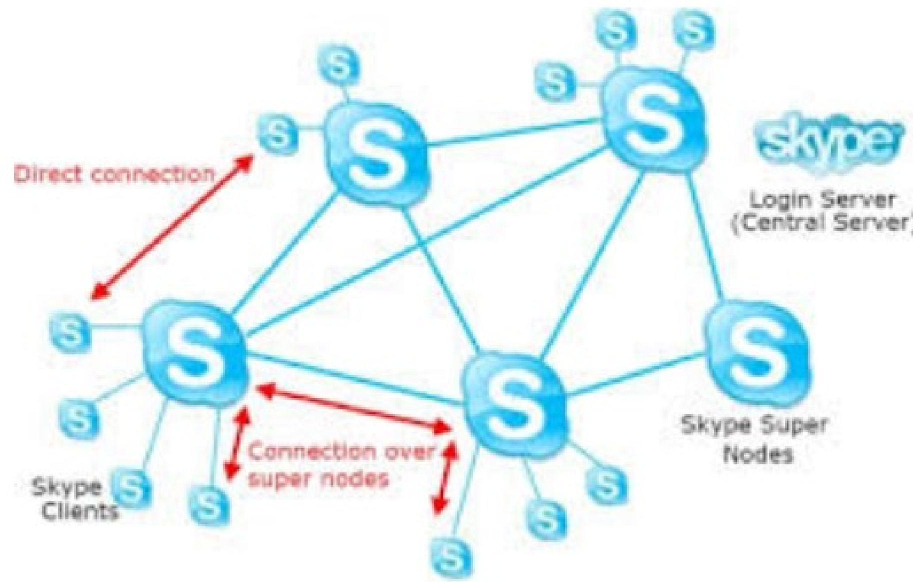
P2P Skype basics

- Two major technologies
  - Voice over IP (VoIP)
  - P2P: Our focus here
- Phone calls are intrinsically P2P, but Skype uses P2P to:
  - Discover peers
  - Traverse **firewalls**: Software and hardware blocking incoming data connections
- **Super Nodes** (SNs): Machines with public IP addresses
  - Contain replicas of Skype's central directory (including IP)
  - Distributed throughout the network, forming a full mesh
  - Act as a network of publicly visible relays
- What happens when caller and callee are behind firewalls?
  - Caller initiates connection with an SN, and callee with another SN
  - Two-way communication established via the SNs
  - Mutually agree on a single SN to shorten the path



More on P2P Skype

- Super nodes (SN)
  - Needs public IP address to traverse NATs and firewalls
  - Should have abundant resources: CPU, memory, ...



- Two tiers of Skype overlay
  - Mesh of SNs
  - Shallow tree of ordinary hosts rooted at each SN

- During login, a host ...
  - Advertises its presence to other hosts
  - Discovers whether it is behind a firewall, which SN to connect to, and which public IP address to use