When Clouds Meet Online Social Networks

Advanced Computer Networks

Summer Semester 2013

Instructor: *Prof. Dr. Xiaoming Fu*

Teaching Assistant of This Session: *Lei Jiao*

Recap: Cloud Computing

- What is "Cloud Computing"?
- What are its typical characteristics?
- What are its service models?
- What are some typical production systems?

Recap: Online Social Networks

- OSN structures
    - Graph theory concepts, power law, small world, *etc.*
- Network formation
    - Random graph, Watts-Strogatz, Rich get richer, *etc.*
- Information cascades
- Social influence maximization

Today's Session

- We narrow down our focus on a specific issue:
  - Online Social Networks (OSNs) and socially aware Internet services in cloud datacenters

- We aim to answer the following questions:
  - What might be some problems "when cloud meets OSN"?
  - How could these problems be modeled and solved?

- We cover the following topics:
  - Scalable OSN data placement in server clusters
  - Cost-minimizing OSN deployment over multiple clouds
  - Social data placement in a datacenter environment

Scalable OSN Data Placement in Server Clusters

Reference:

J. Pujol *et al*, “The Little Engine(s) That Could: Scaling Online Social Networks”, *SIGCOMM 2010*

Introduction to OSN Scaling

- Background
  - Online Social Networks (OSNs) extremely popular
  - OSN grows fast: Twitter 1382% between 2009/2 to 2009/5
  - OSN data placement *across servers* must be scalable

- Conventional scaling approaches
  - Vertically: Upgrade existing hardware
    - Expensive; Sometimes technically infeasible
  - Horizontally: Deploy more servers and partitioning load
    - Suitable only for stateless front-end servers
    - If used for back-end storage servers, data must be partitioned into disjoint components.

Introduction to OSN Scaling (Cont.)

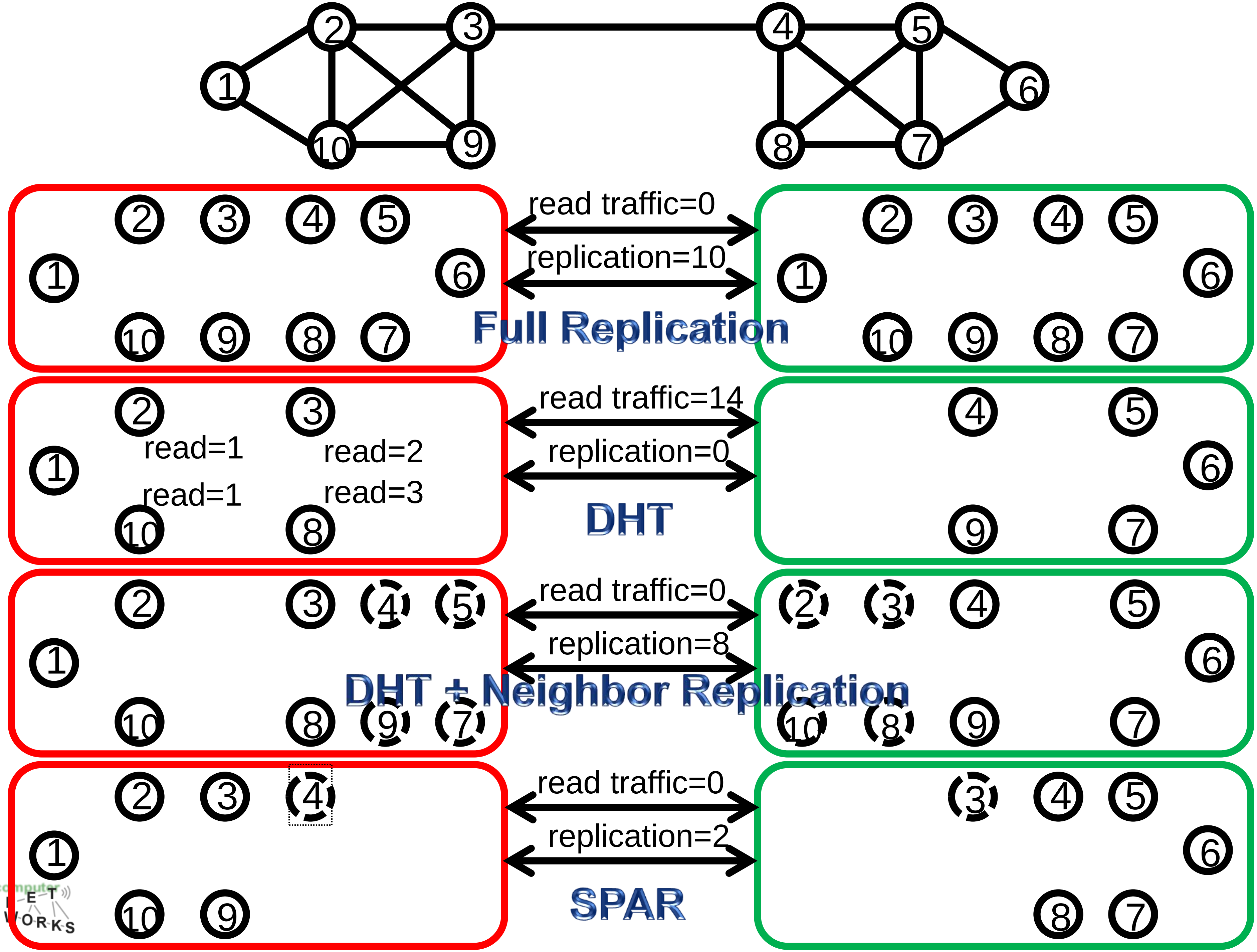
- Conventional approaches inapplicable to OSN
  - Data extremely huge: Makes vertical scaling inapplicable
  - Data inter-connected: Makes horizontal scaling inapplicable

- Problems of using horizontal scaling to OSN
  - Most OSN operations are between a user and her neighbors
  - Neighbors' data are placed on multiple servers
  - The "multi-get" inter-server operations can:
    - Incur a lot of inter-server traffic
    - Incur unpredictable response time

A Novel Solution

- SPAR (Social Partitioning And Replication)
  - "One-hop Replication": Replicating all a user's neighbors' data to the server that hosts the user's own data
  - "Social Locality"

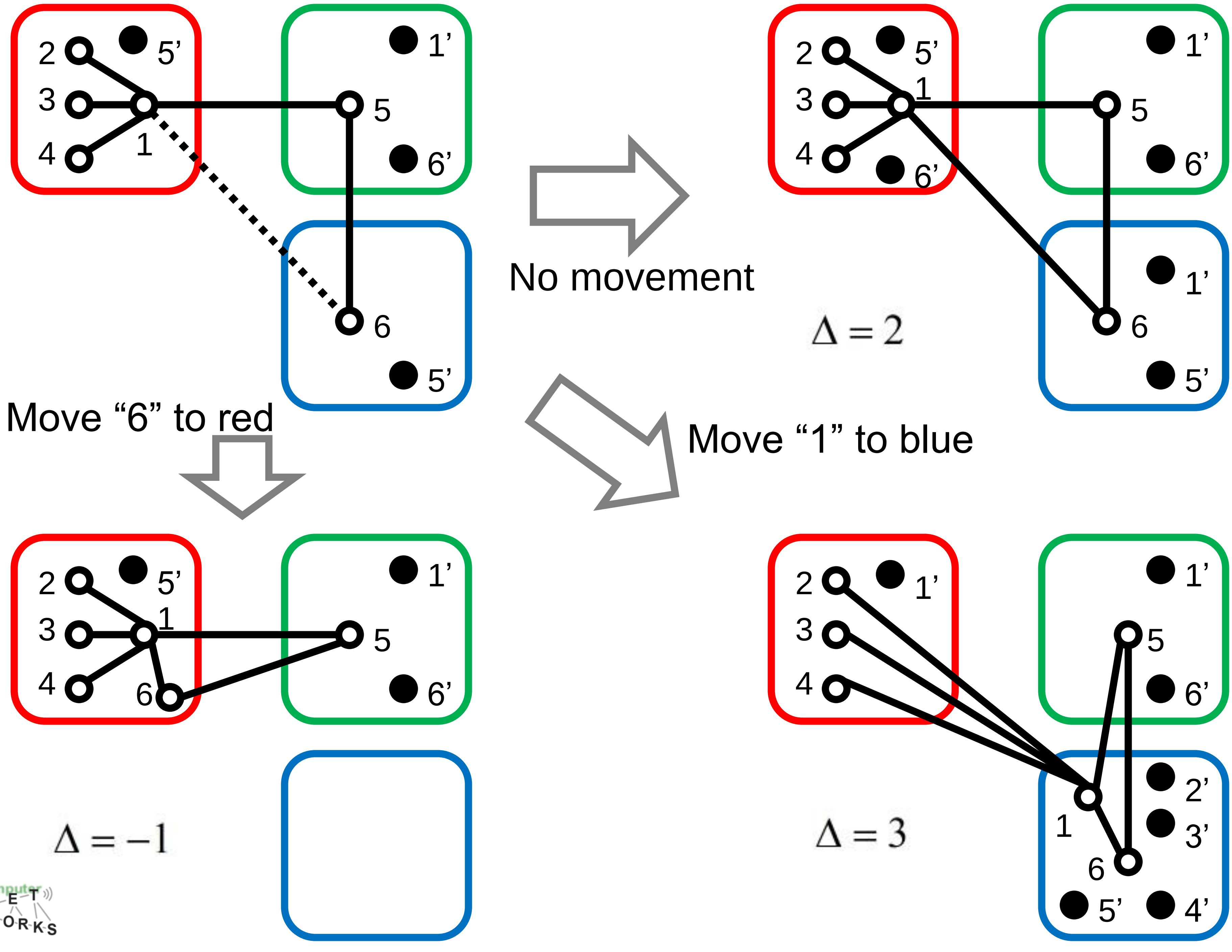
- Requirements for SPAR
  - Maintain local semantics
  - Balance loads
  - Be resilient to machine failures
  - Be amenable to online operations
  - Be stable
  - Minimize the replication overhead


read traffic=0
replication=10
Full Replication
read traffic=14
replication=0
read=1
read=2
read=1
read=3
DHT
read traffic=0
replication=8
DHT + Neighbor Replication
read traffic=0
replication=2
SPAR

The SPAR Algorithm

- SPAR: Dynamically respond to 6 events
  - Node (*i.e.*, User) / Edge (*i.e.*, Social relation) / Server
  - Addition / Removal

- Event case 1: Node addition
  - Create the master on the server with fewest masters
  - Create *k* slaves and place randomly
- Event case 2: Node removal
  - Remove the master and all slaves of this node
  - Remove neighbors' slaves that exist only for social locality of this node, if not violating redundancy requirements
- Event case 3: Edge addition


No movement
$\Delta = 2$
Move “6” to red
$\Delta = -1$
Move “1” to blue
$\Delta = 3$

The SPAR Algorithm (Cont.)

- Event case 4: Edge (between *u* and *v*) removal
  - Remove *u*'s slave on *v*'s master server, if not violating the redundancy requirement
  - Vice versa for *v*'s slave

- Event case 5: Server addition
  - Approach 1: Do nothing since "Event case 1" will place new nodes on the new server automatically.
  - Approach 2: Select and move existing masters to the new server while maintaining one-hop replication for every user.

- Event case 6: Server removal
  - Promote slaves on the remaining servers to be masters

Cost-Minimizing OSN Deployment over Multiple Clouds

Reference:

L. Jiao *et al*, “Cost Optimization for Online Social Networks on Geo-Distributed Clouds”, *ICNP 2012*

Introduction: OSN on Clouds

- OSN often needs to be deployed at diverse geographic locations.
  - Proximity to users, data availability, fault tolerance, *etc.*
- Clouds seamlessly matches this requirement.
  - Geographic distribution
  - “Infinite” on-demand resources
  - “Pay-as-use” flexible charge schemes
  - No need to build/operate one’s own datacenters
- OSN on clouds case studies

Introduction: OSN on Clouds (Cont.)

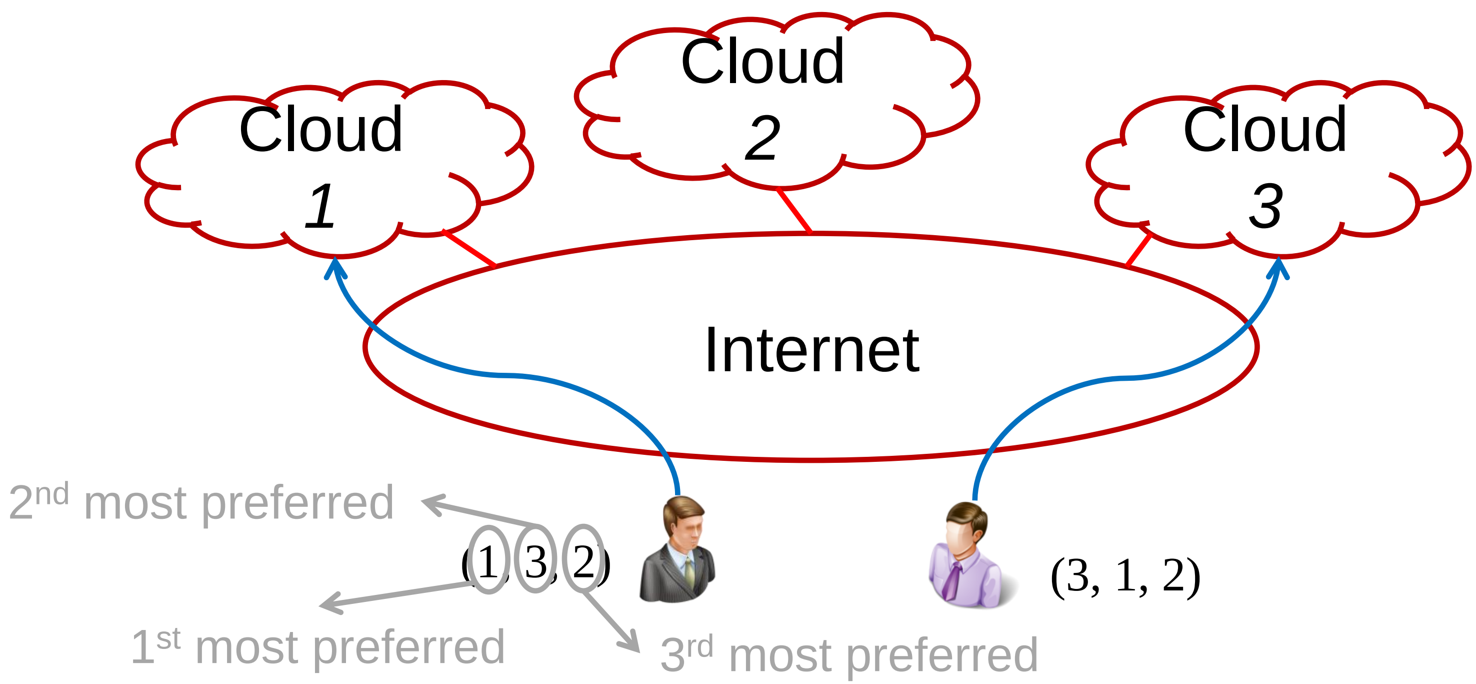
- OSN providers' concerns
  - Cost: The money spent in using cloud resources
  - QoS: The service quality perceived by end users
    - Access latency, *etc.*

- Such "cost-QoS" issue is complicated by OSN dynamics
  - New users join, old users leave, social relations vary, *etc.*

- Let's investigate this problem: Minimizing the cost of an OSN while providing satisfactory QoS to users, over multiple geographically distributed clouds

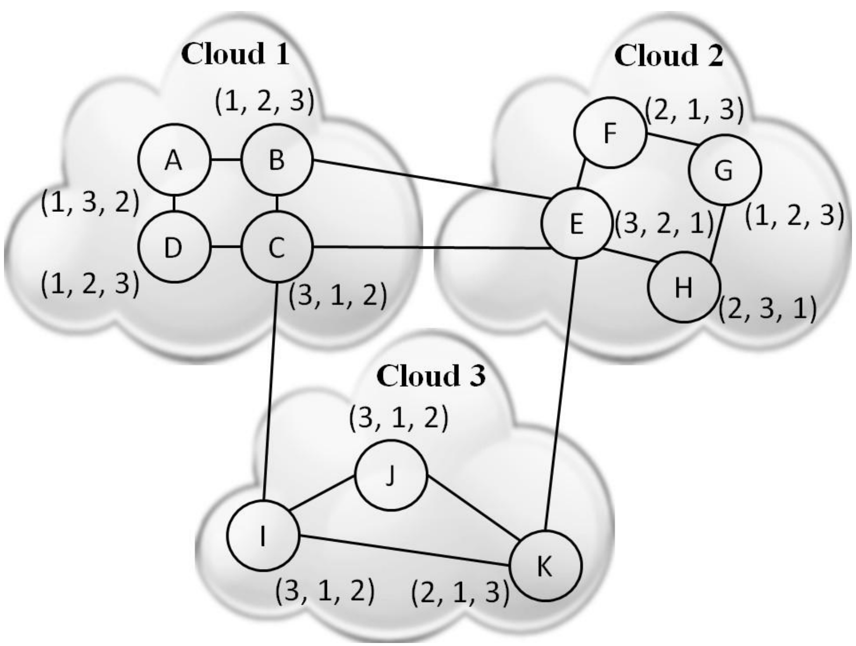
How to define OSN QoS?

- In the multi-cloud scenario, for each user:
  - One cloud is selected to host the user's data, and serve this user.
  - All clouds can be *sorted* or *ranked* in terms of a given metric (*e.g.*, access latency perceived by the user).
    - Each user has her *1st most preferred* cloud, *2nd most preferred*, *etc.*



How to define OSN QoS? (Cont.)

- A vector approach: Define the QoS of an OSN service as $\vec{q} = (q_1, q_2, ..., q_k, ..., q_N)$, where
  - q_k: The percentage of users whose data are placed on any of their *most preferred* k cloud(s)
  - N: The total number of clouds



- In the left example:
  - $N = 3$
  - $q_1 = 7 / 11 = 0.64$
    - A, B, D; F, H; I, J
  - $q_2 = q_1 + 3 / 11 = 0.91$
    - C; E, G
  - $q_3 = q_1 + q_2 + 1 / 11 = 1$
    - K
  - Thus, $\vec{q} = (0.64, 0.91, 1)$

How to define OSN QoS? (Cont.)

- How to use the vector approach to express “80% of all accesses must be satisfied within 200 ms”?
  - Step 1: For any user i, calculate n_i, *i.e.*, placing user i’s data on any of her most preferred n_i clouds can grant her the access latency of less than 200 ms.
  - Step 2: Calculate $n_{min} = \min(n_i)$
  - Step 3: Set $\vec{q}[n_{min}] = 80\%$

How to define OSN QoS? (Cont.)

- Example: User A: (1, 3, 2); B: (1, 2, 3); C: (1, 2, 3)
  - If: A's data must be placed on 1 or 3, *i.e.*, $n_a = 2$
  - B's must be on 1 or 2, *i.e.*, $n_b = 2$
  - C's can be on any cloud, *i.e.*, $n_c = 3$
  - Then set $\vec{q}$ = (*, 0.8, 1)

Always 1

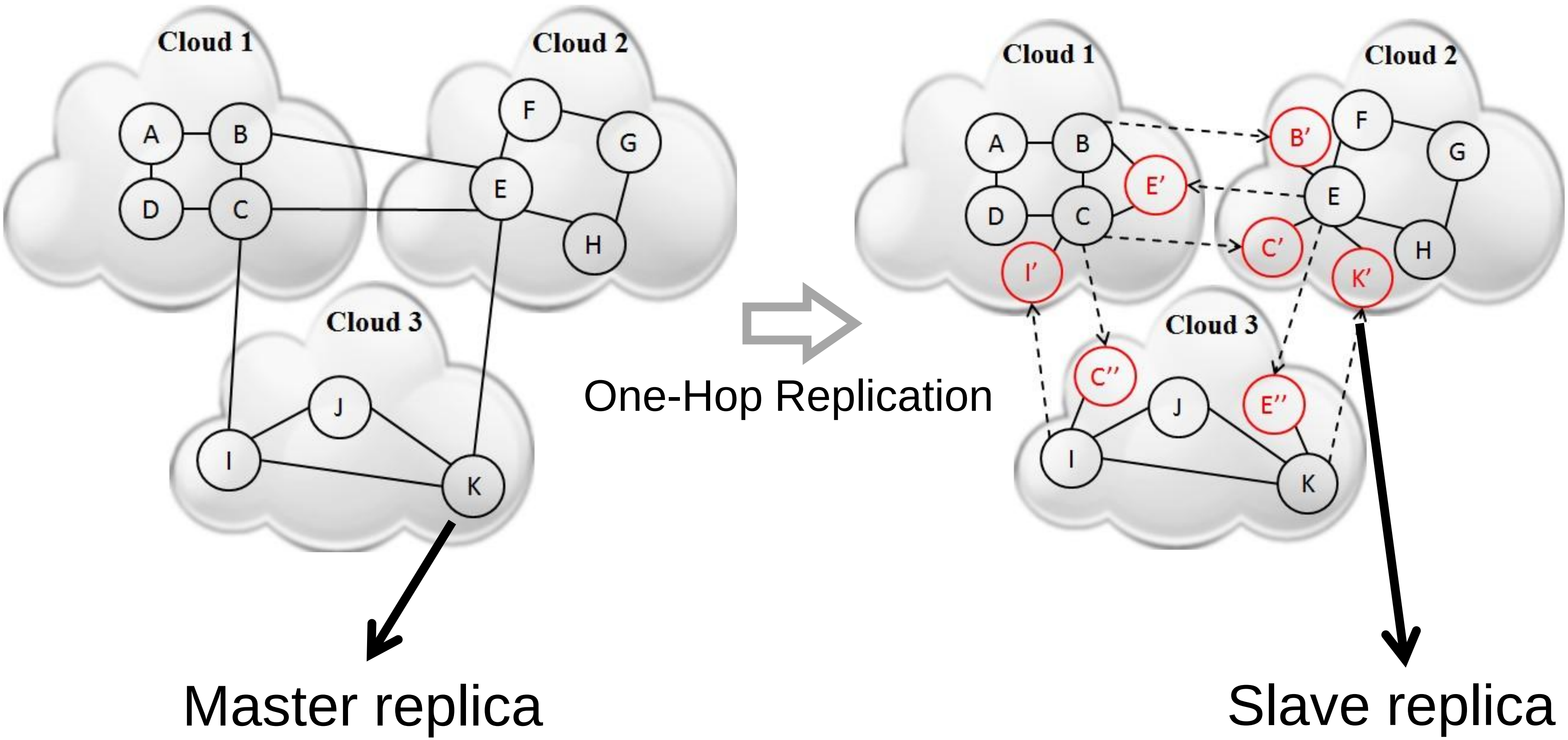
Any value no greater than 0.8 in this case

computer NETWORKS

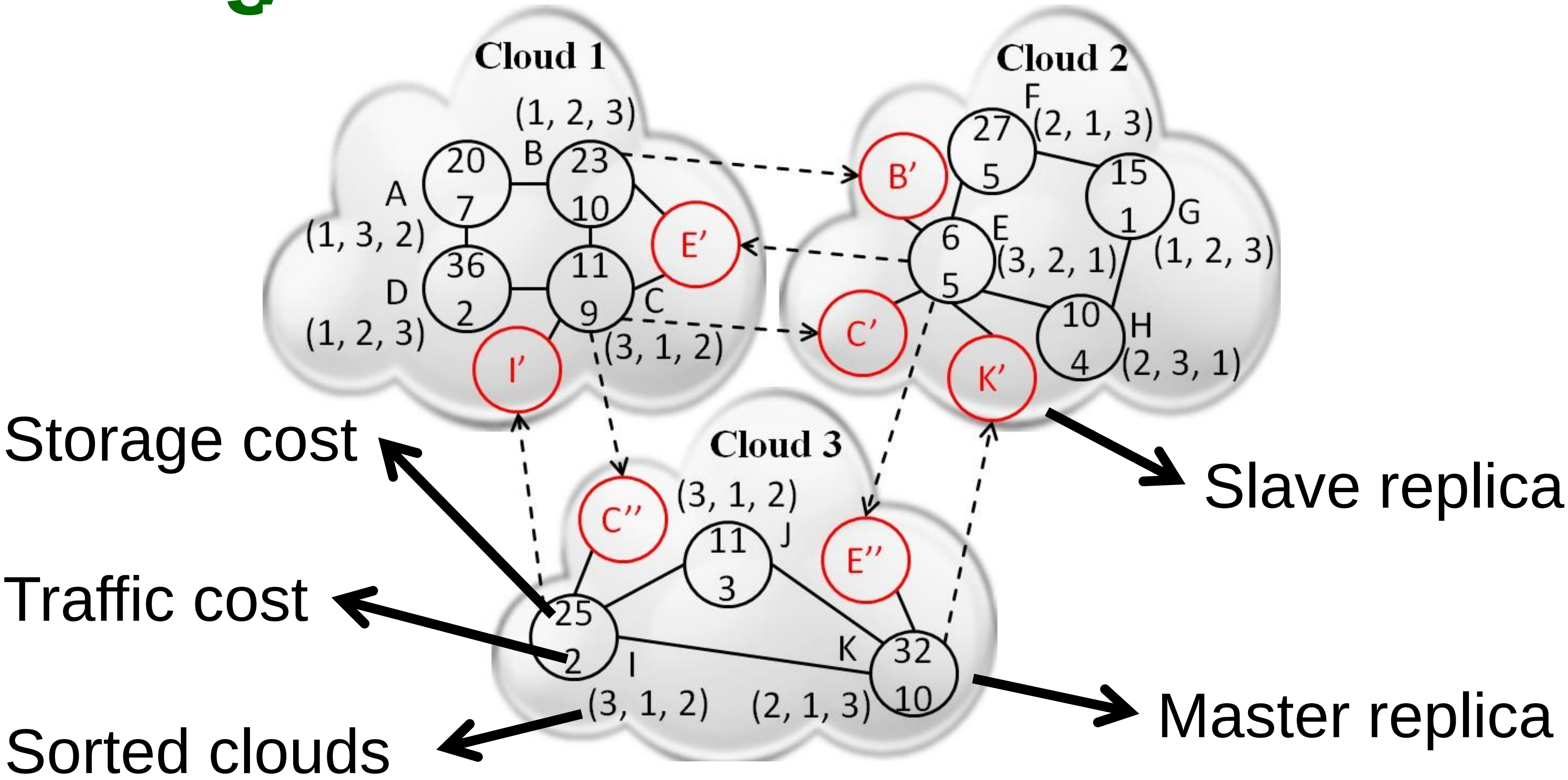
How to define OSN Cost?

- The monetary cost of an OSN service on multi-clouds
  - Front-end cost: VM, traffic between OSN service and users
  - Back-end cost: Storage, inter-cloud traffic, *etc.*
    - Let’s focus on this.
- Different types of cost
  - Storage cost
  - Inter-cloud traffic cost
  - Maintenance cost (for *social locality*)

Storage and Inter-Cloud Traffic Cost



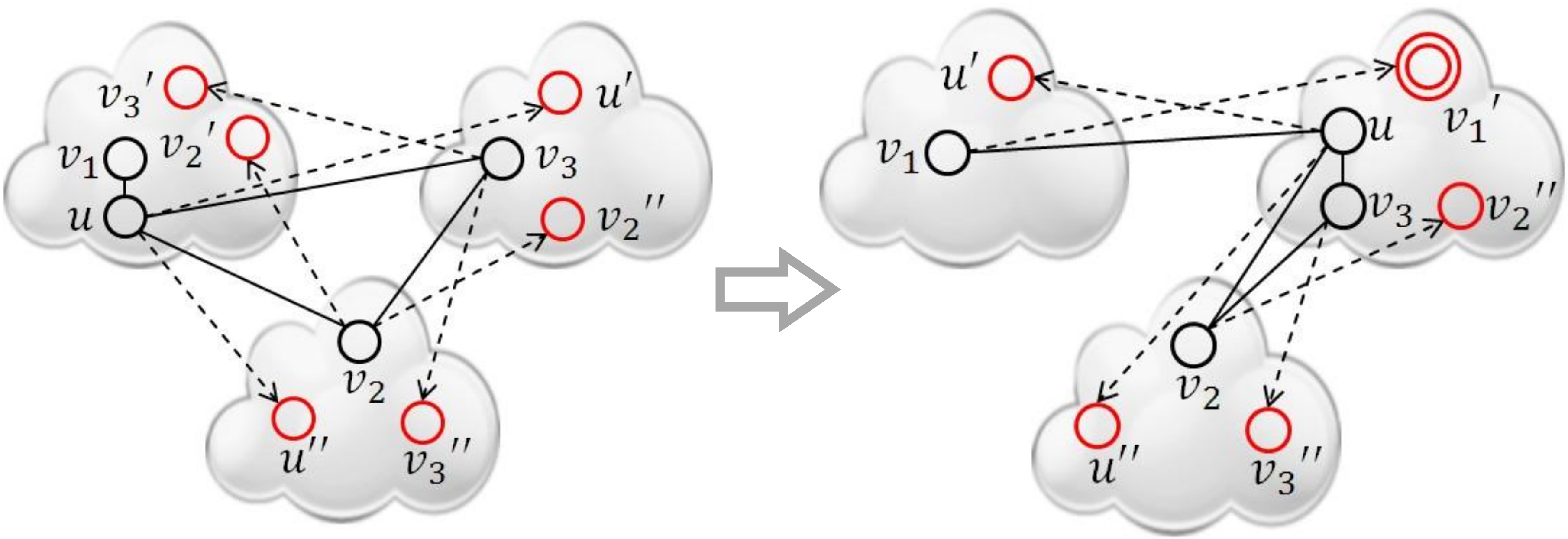
Storage and Inter-Cloud Traffic Cost



- Total storage cost = 330
- Total inter-cloud traffic cost = 50

Algorithm: Cosplay

- Basic idea: Swapping the roles of a user’s master and her slave (if *feasible*) may lead to cost reduction.



  - Swap *u* and *u’* (*i.e.*, *u* becomes *u’* and *u’* becomes *u*)
  - Do NOT forget to maintain social locality
  - Cost reduction: (10+6)-(9+5)-1=1

Social Data Placement in a Datacenter Environment

Reference:

L. Jiao *et al*, “Optimizing Data Center Traffic for Online Social Networks”, *LANMAN 2013*

X. Cheng *et al,* “Load-Balanced Migration of Social Media to Content Clouds”, *NOSSDAV 2011*

Data Center Network Performance Goals

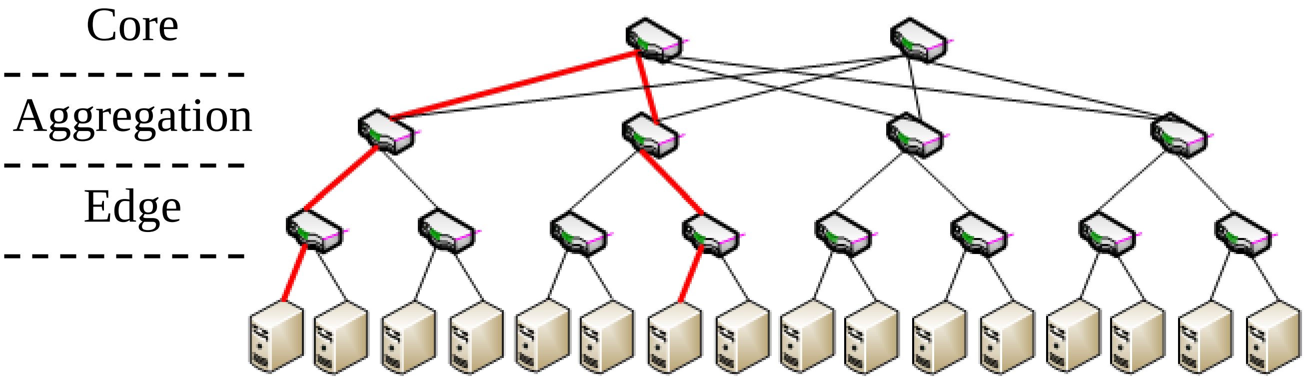


Fig. 1 Tree

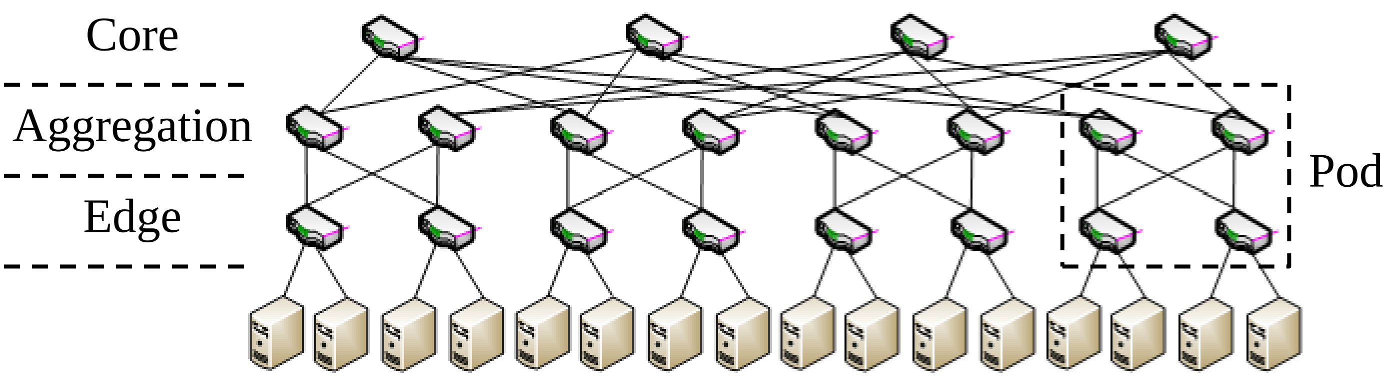


Fig. 2 Fat-tree

- Goal #1: Minimizing the core-layer traffic (Tree)
  - The synchronization traffic traveling through core switches
- Goal #2: Minimizing the total perceived traffic (Tree/Fat-tree)
  - The sum of the synchronization traffic perceived by every switch

Algorithm

- Basic idea: *Swapping the roles* of a master-replica pair can possibly reduce the traffic counted by the control matrix.

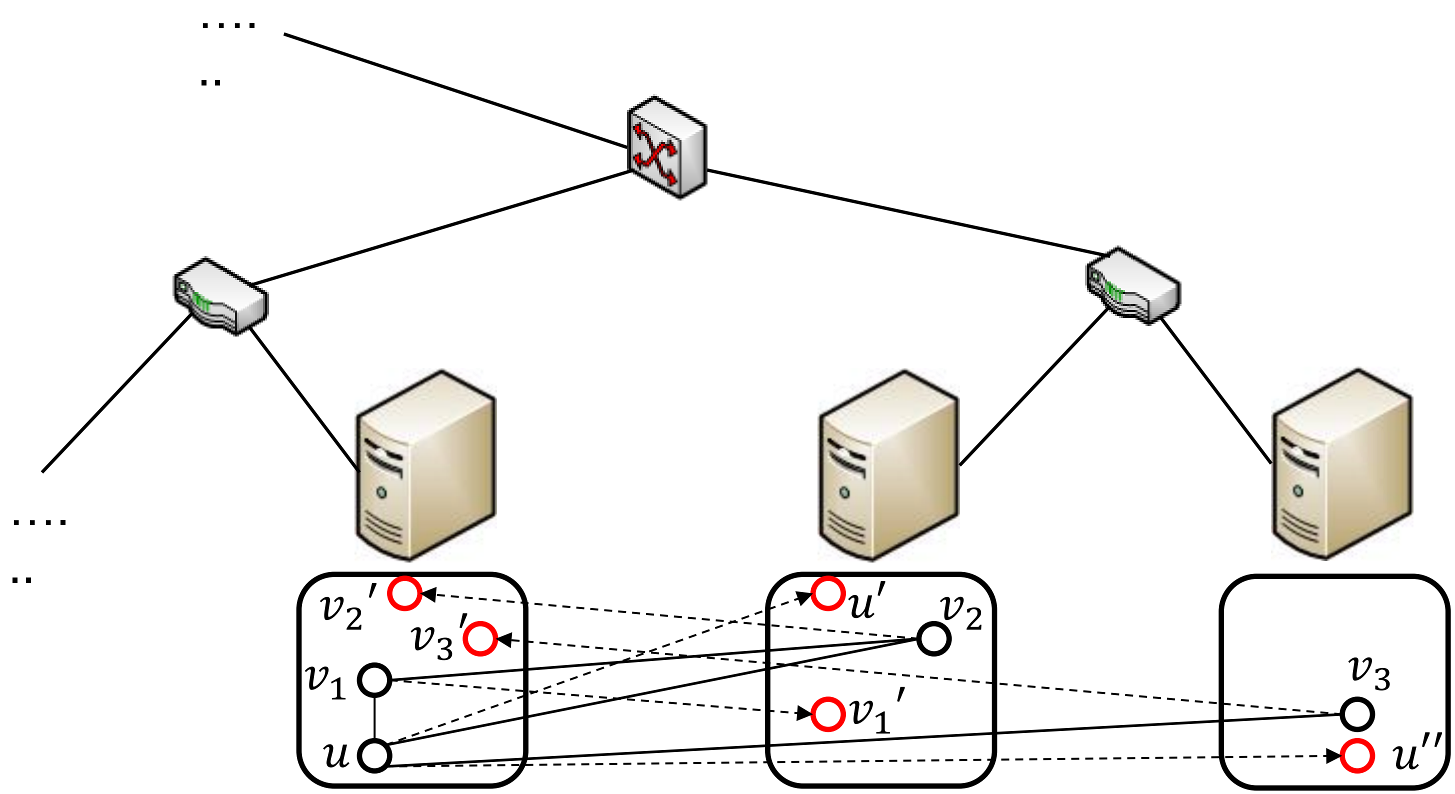


Fig. 1 Before Swapping: Traffic = 15; Load = (2, 1, 1)

Algorithm (Cont.)

- Swapping the roles of u and u', while maintaining social locality.

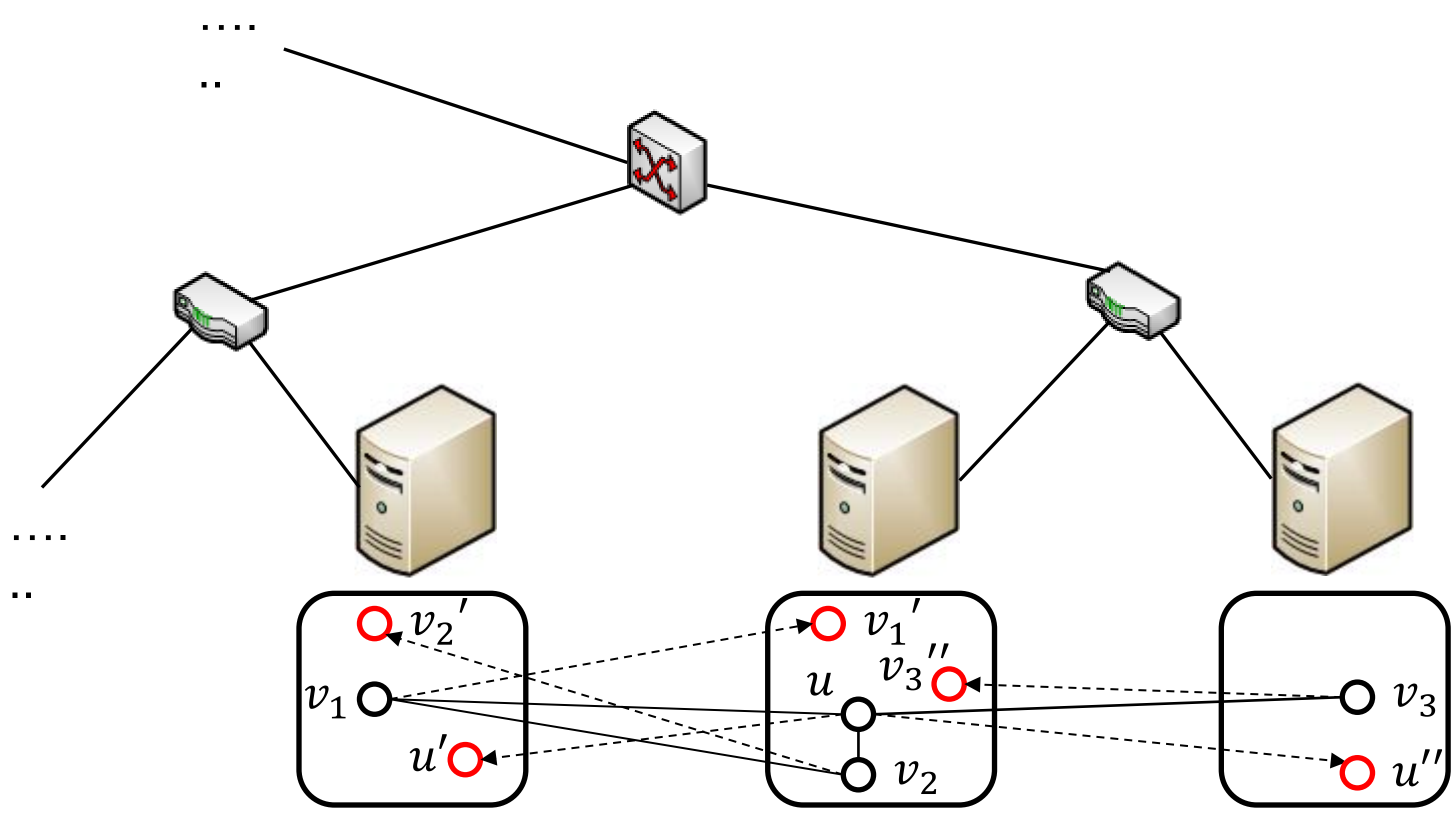


Fig. 2 After Swapping: Traffic = 11; Load = (1, 2, 1)

Algorithm (Cont.)

- Swapping the roles of v_2 and v_2', while maintaining social locality.

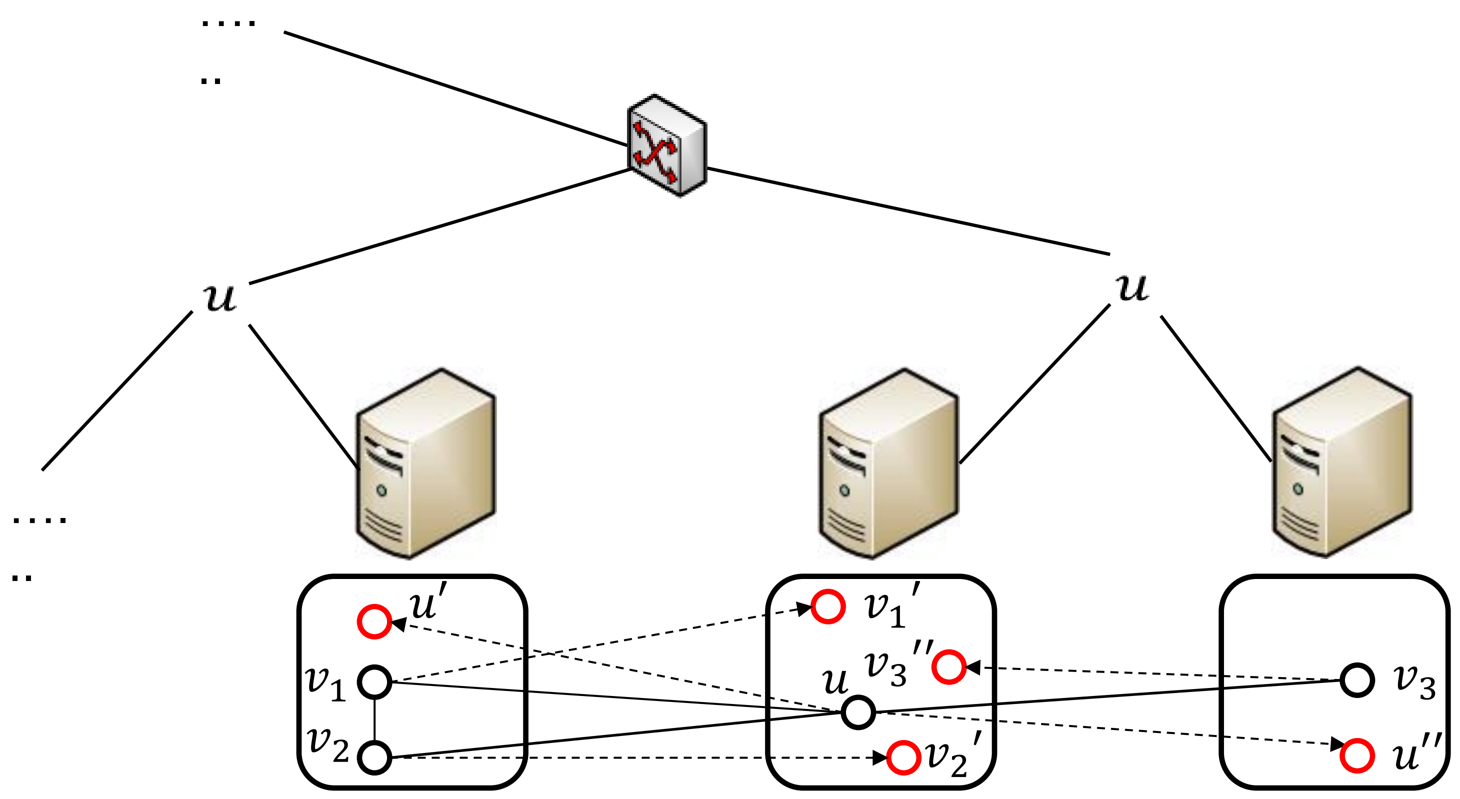


Fig. 3 After Swapping: Traffic = 11; Load = (2, 1, 1)

Load-Balanced Data Placement

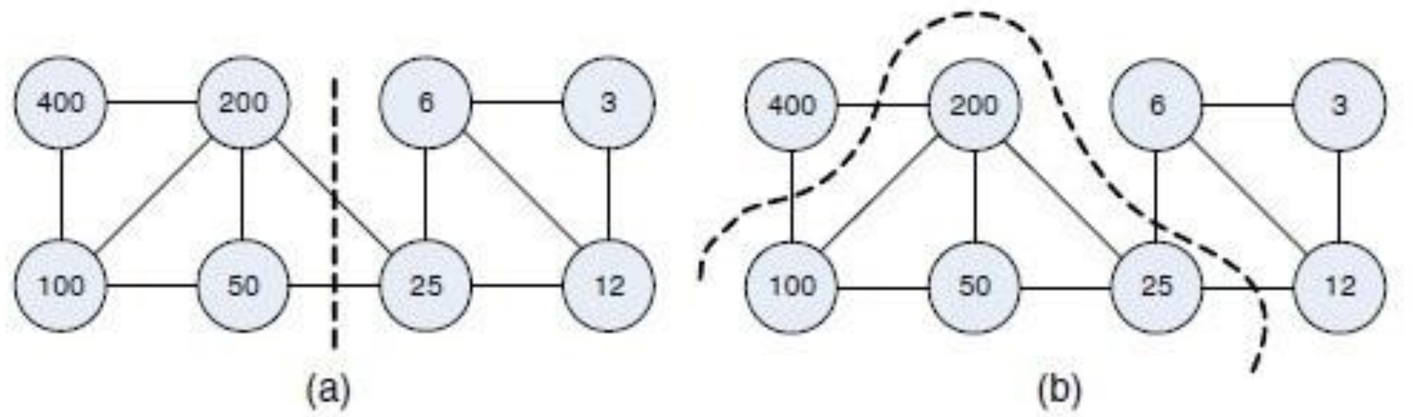


Figure : Example of different partitions based on (a) social relationship only, (b) both social relationship and popularity

Summary of Today's Session

- We investigate a specific issue:
  - Online Social Networks (OSNs) and socially aware Internet services in cloud datacenters

- We introduce the problems, and algorithms on the following topics:
  - Scalable OSN data placement in server clusters
  - Cost-minimizing OSN deployment over multiple clouds
  - Social data placement in a datacenter environment

Thanks!

For any questions or concerns, please feel free to contact:
Lei Jiao, http://user.informatik.uni-goettingen.de/~ljiao/