Cloud Computing – Part 3 of 3

Advanced Computer Networks

Summer Semester 2014

Some Administrativa

- Examination: written
- Date: July 24th, 10am-12am
- Room: MN14

- ITIS students: please use the ITIS online form to register
- All others: use FlexNow, take care of the deadline

Research Advances in Cloud Computing

Today's Session

- We look at some issues of cloud computing from a research perspective.
  - Topic: Datacenter Networking

- Determining the future of the cloud
  - It usually takes some time for research results to be implemented

Publishing in Computer Networking

- After working on a project, usually submit the work to a *conference*
  - Peer review
  - Top papers on conferences are fast-forwarded to journals
  - Some conferences with high reputation:
    - SIGCOMM, INFOCOM, ICNP, NSDI, IMC, ATC, CoNEXT, …
    - Acceptance rates of ~10-15%

- After publishing at a conference, submit an extended version (~+30%) to a journal
  - Another round of peer review

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
  - OSNs grow fast:
    - Twitter 1382% between 2009/2 to 2009/5
    - Facebook: > 1 billion users
  - 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.)

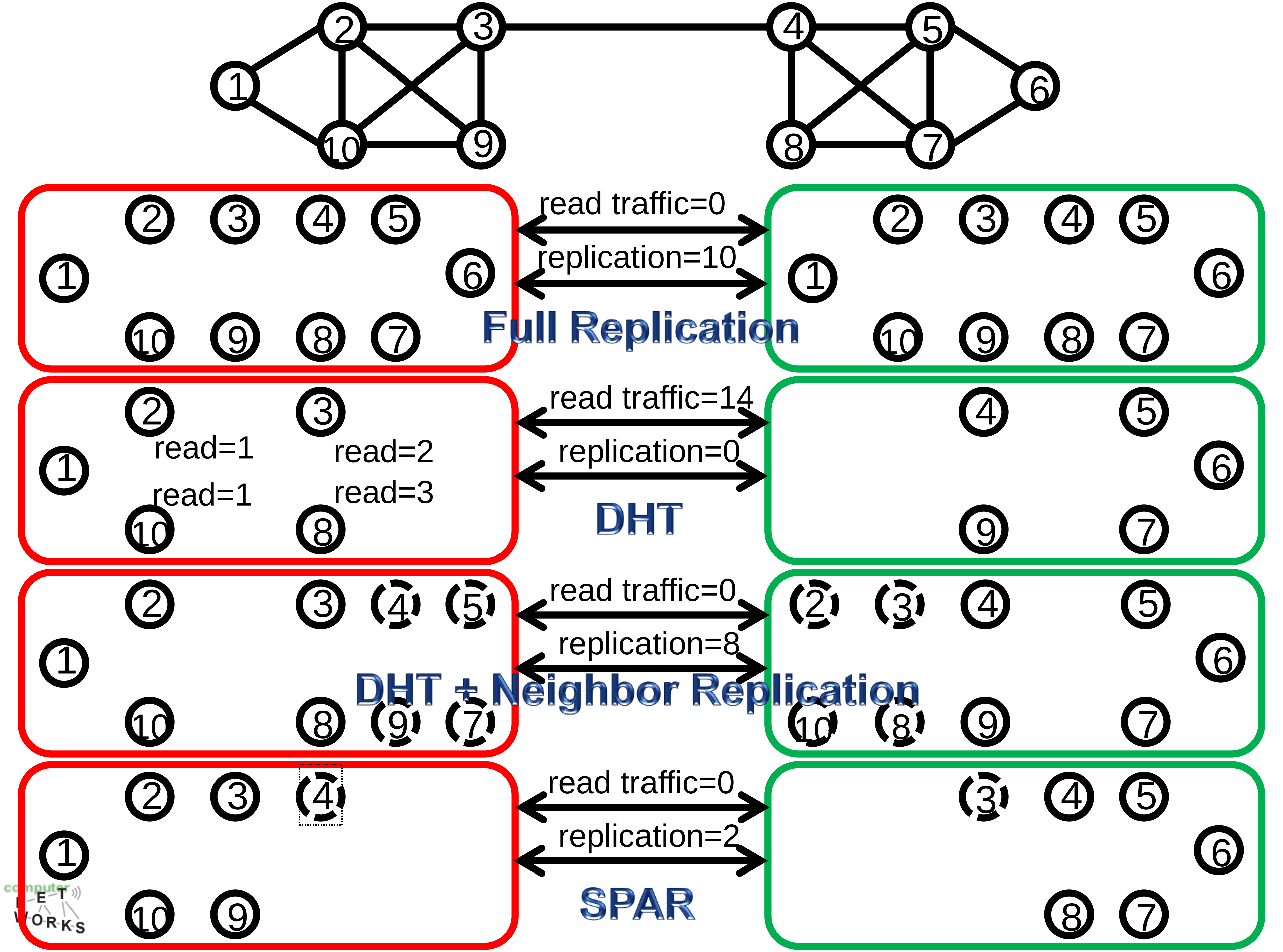
- Conventional approaches inapplicable to OSN
  - Data extremely huge: Makes vertical scaling inapplicable
  - Data inter-connected: Makes horizontal scaling inapplicable

- 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

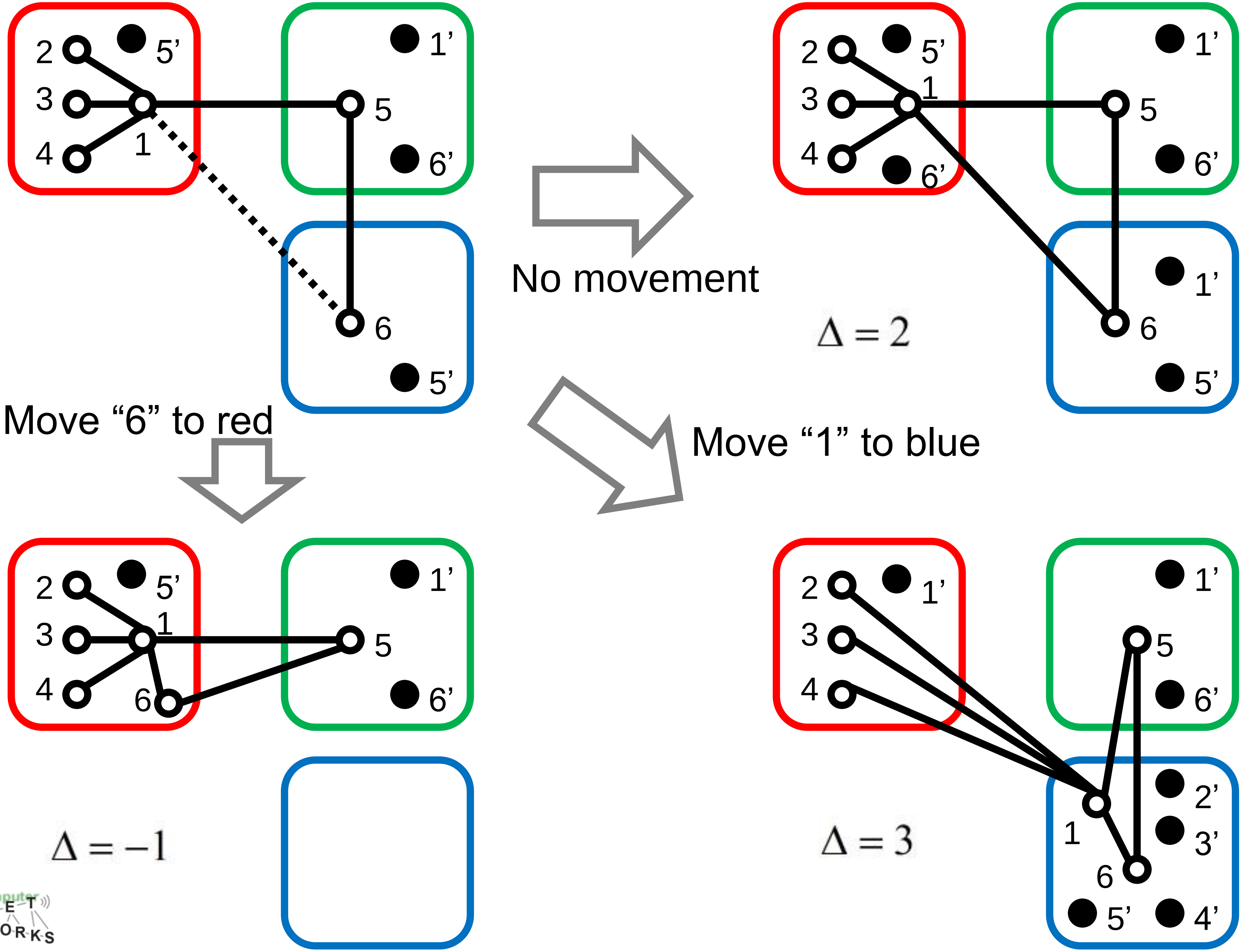


read traffic=0
replication=10
Full Replication
read traffic=14
replication=0
read=1
read=1
read=2
read=3
DHT
read traffic=0
replication=8
DHT + Neighbor Replication
read traffic=0
replication=2
SPAR

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
Δ = 2
Move “6” to red
Δ = −1
Move “1” to blue
Δ = 3
2
3
4
1
5’
1’
5
6’
6
5’
2
3
4
1
5’
6’
1’
5
6’
1’
6
5’
2
3
4
1
6
5’
1’
5
6’
2
3
4
1’
1’
5
6’
2’
3’
1
6
5’
4’

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

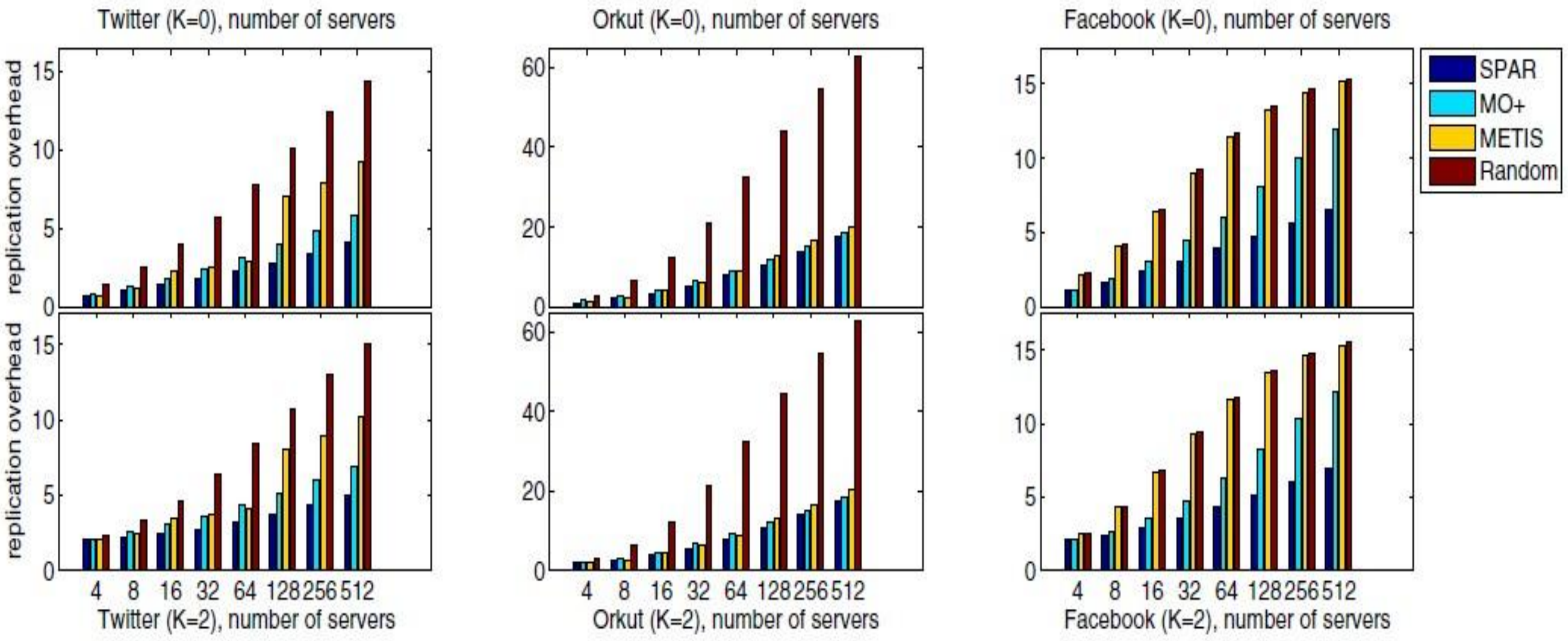
- Event case 5: Server addition
  - Approach 1: Do nothing since "Event case 1" will place new nodes on the new server automatically.
  - Approach 2: Select and move existing masters to the new server while maintaining one-hop replication for every user.

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

Evaluation of SPAR: Settings

- Metric
  - Replication overhead: The total # of slaves of all users
- Datasets
  - Twitter: 2M users, 48M edges
  - Facebook: 60K users, 1M edges
  - Orkut: 3M users, 200M edges
- Other algorithms for comparison
  - METIS: Minimize the # of inter-server edges
  - Random partitioning: Widely used in DBMS
  - MO+: Detect equal-sized communities
- Ensure one-hop replication for all algorithms

Evaluation of SPAR: Results



S. Agarwal *et al*, "Volley: Automated Data Placement for Geo-Distributed Cloud Services", *NSDI 2010*

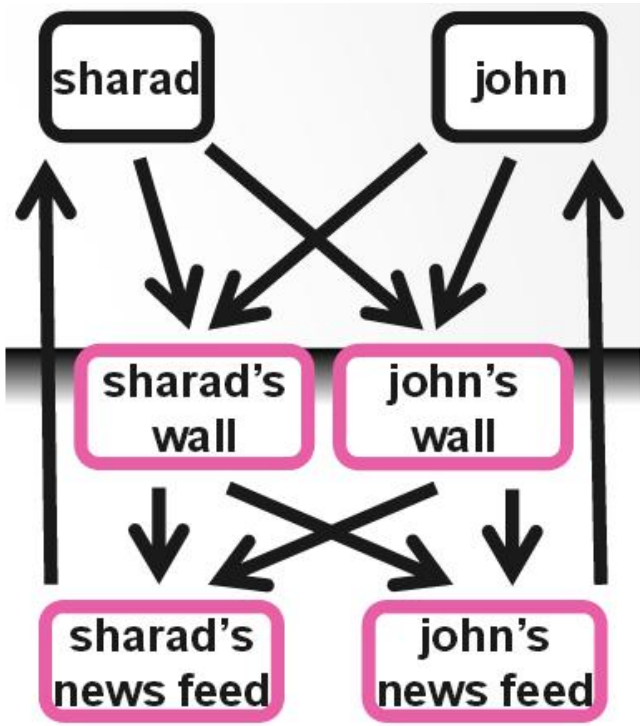
Introduction

- How about multiple geo-distributed DCs?
  - Major cloud providers have tens of DCs at diverse geographic locations.
  - User locations?
  - A user should be served at the best DC for him.

- User/Provider interests:
  - Users want to select shortest latency DC
  - Cloud providers care about cost: inter-DC traffic and DC capacity provisioning.
  - *Placing data for lowest latency and least cost?*

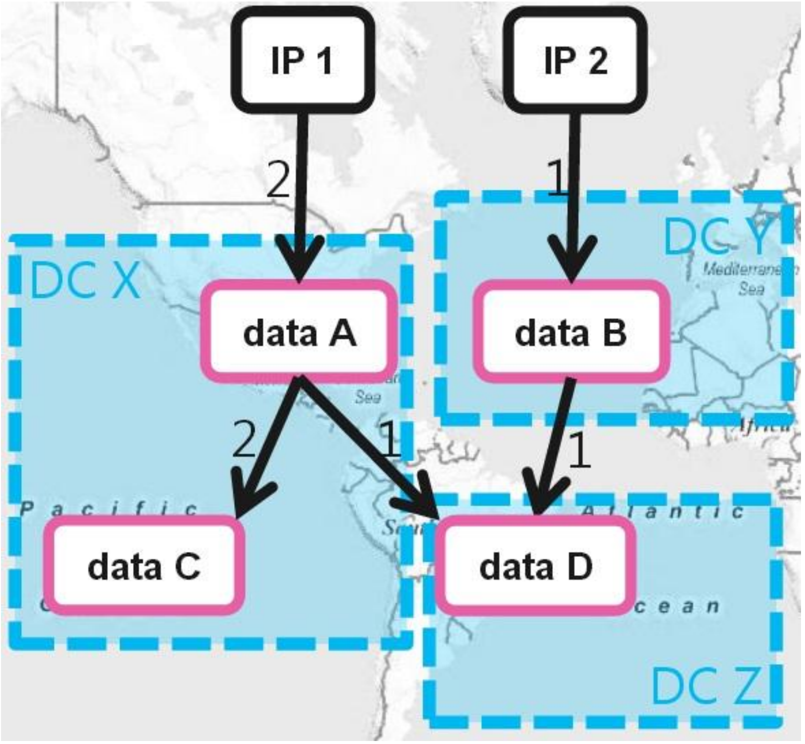
Introduction (Cont.)

- Critical challenges
  - Scale: *O*(100 millions) users
  - Addressing both latency and cost
  - Data to be placed are *dynamic*
  - Shared data: *interdependency*
    - An OSN example
  - Applications change
    - Data patterns also change
    - See FB messenger introduction
  - Users can be mobile
    - Quick data migration?



An Example of Data Placement

- How the latency and the inter-DC traffic generated
  - Transaction 1: User at IP 2 updates wall B with subscriber D.
  - Transaction 2: User at IP 1 updates wall A with subscribers C and D.



The Volley Algorithm

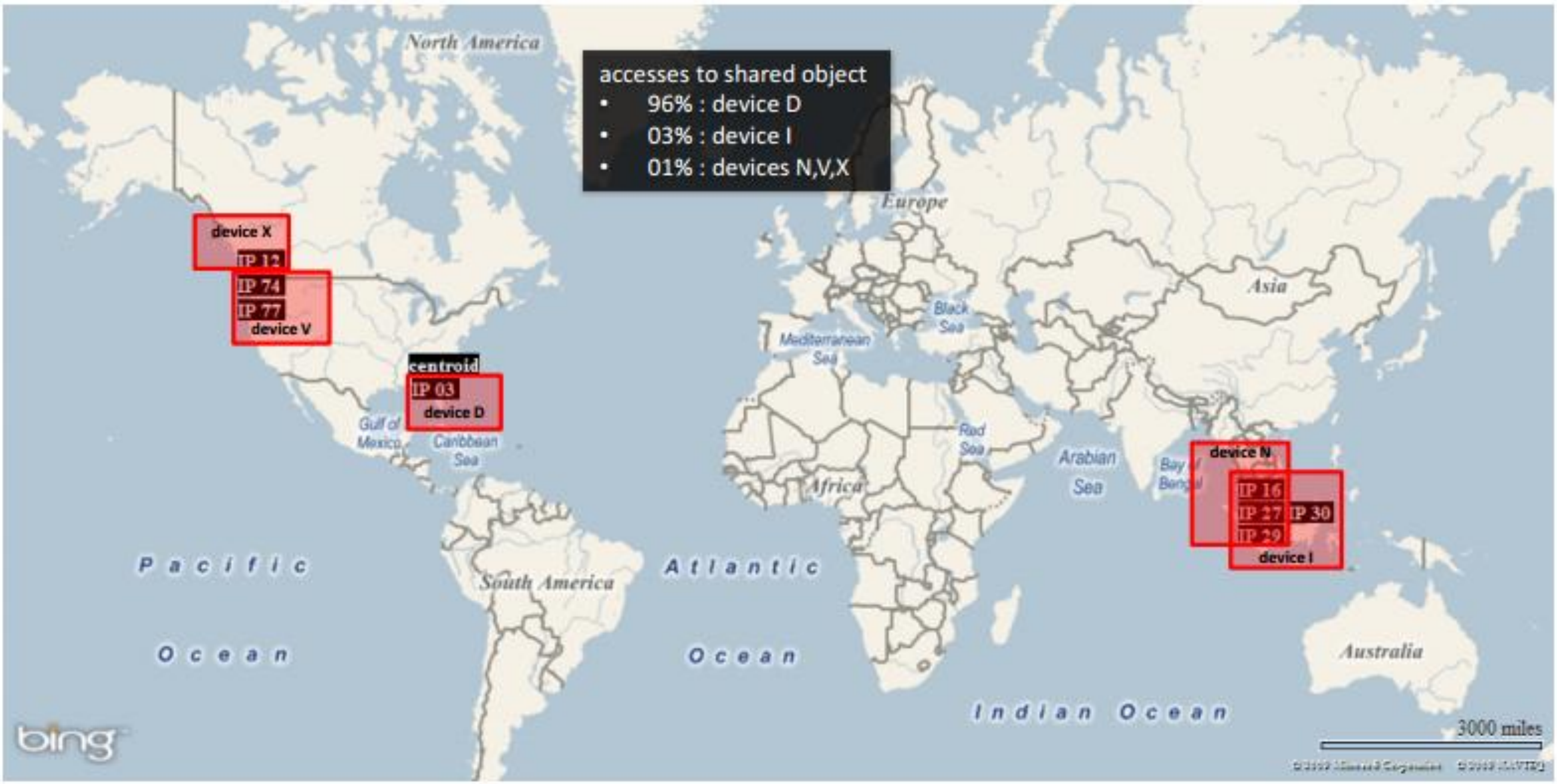
- *Three phases*:
  - Phase 1: Compute initial placement
  - Phase 2: Iteratively move data to reduce latency
  - Phase 3: Iteratively collapse data to datacenters

The Volley Algorithm

- *Phase 1*: Calculate geographic centroid for each data
  - considering client locations without data interdependencies
  - Centroid: IP-to-location mapping based on a geo database
  - Each data item that is directly accessed by client(s):
    - Map to weigthed-average of the geo coordinates of accessing client(s) IP addresses
  - highly parallel
    - Uses SCOPE (“Microsofts MapReduce”)

The Volley Algorithm

- *Phase 1*: Calculate geographic centroid for each data



The Volley Algorithm

- *Phase 2*: Refine *centroid* (not DC!) for each data iteratively
  - considers client locations, and data interdependencies
    - i.e., moves data items closer to clients and other communicating data items
  - using *weighted spring* model that attracts data items on a spherical coordinate system
    - Principle: latency distance and amount of communication between them increase the spring force that is pulling them together (i.e. lets them be placed close to each other)
    - Simultaneously reduces inter-DC traffic (data items closer to each other) and latency (users closer to data)
    - Results in near ideal placement

The Volley Algorithm

- *Phase 3*: Confine centroids to individual DCs
- Initially map each item to closest to centroid data center
  - Oversubscribed DCs: iteratively roll over least-used data and move it to next closest data center
  - as many iterations as number of DCs is enough in practice

The Volley Algorithm

- If an item moved: Output a *migration proposal* to migration mechanism
  - Volley designed to work on many different cloud services

  - Different mechanisms for different services
    - Marking of data, replication, …
  - Leave actual migration to the service!

Migration Proposal Record Format

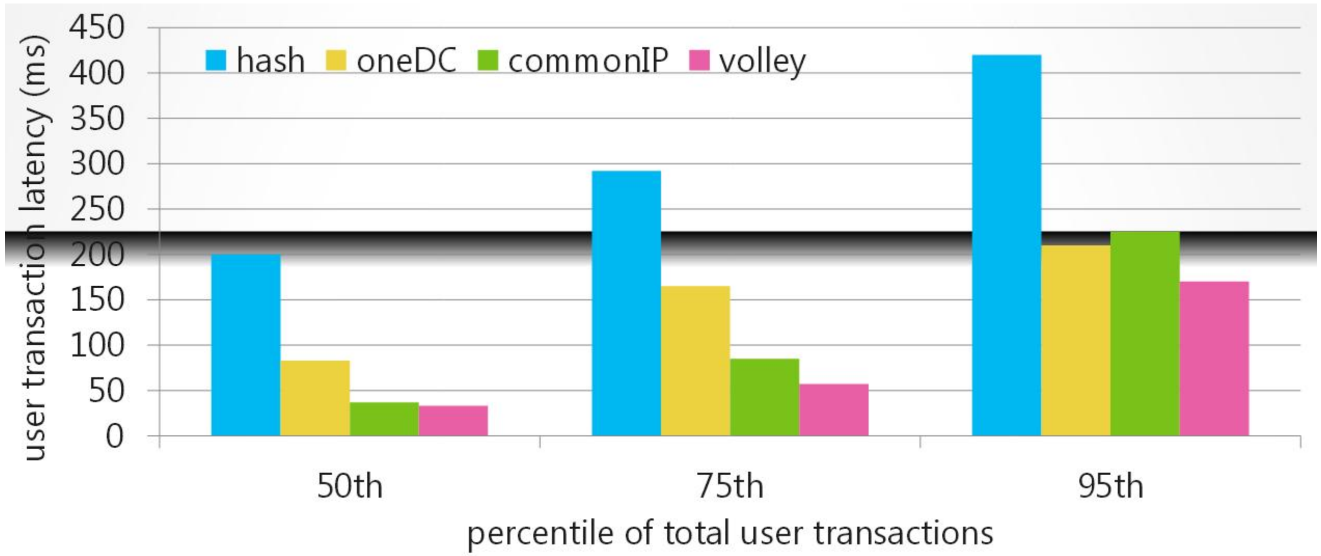
Field	**Meaning**
Entity	The GUID naming the entity (40B)
Datacenter	The GUID naming the new datacenter for this entity (40B)
Latency-Change	The average change in latency per request to this object (4B)
Ongoing-Bandwidth-Change	The change in egress and ingress bandwidth per day (4B)
Migration-Bandwidth	The one-time bandwidth required to migrate (4B)

Evaluation Settings

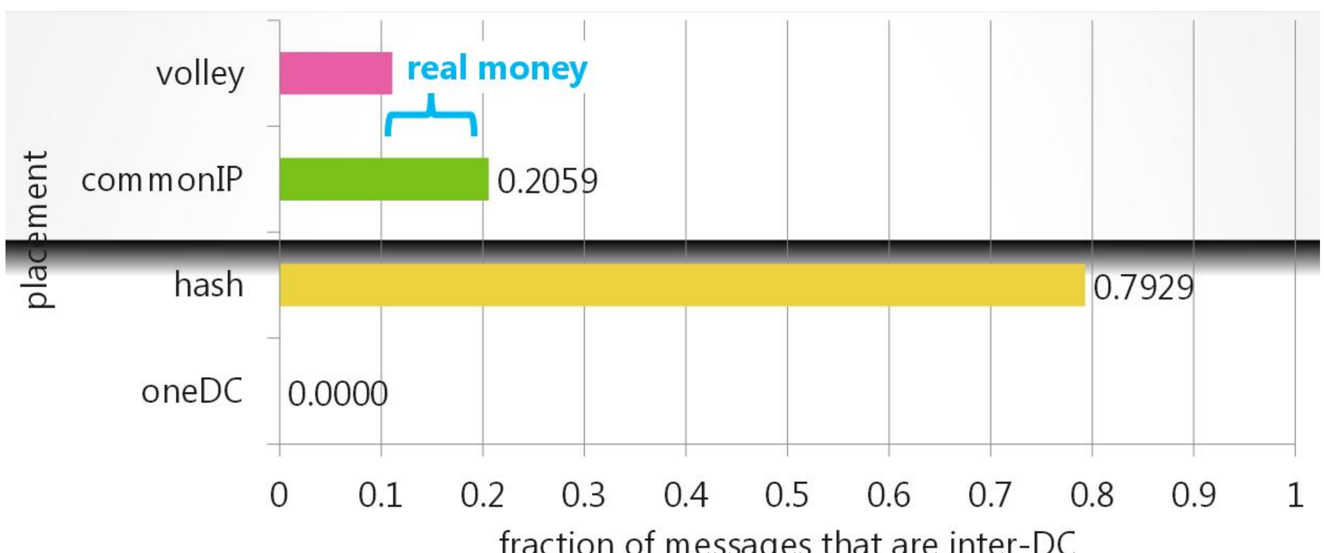
- Inputs
  - Windows Live Mesh traces from June 2009
  - Compute placement on week 1, evaluate it on weeks 2, 3, 4
  - 12 geo-distributed DCs
- Metrics
  - Latency, inter-DC traffic
- Methods for comparison
  - **Hash**: randomly place data over DCs
  - **OneDC**: place all data at one DC
  - **CommonIP**: pick DC closest to IP that most frequently uses data

Evaluation Results

- Latency



- Traffic



Summary & Outlook

- This is just one particular research problem in the cloud
  - Others include: green cloud computing (energy efficiency), specific algorithms for mobile cloud computing, etc.
  - Our group has published research papers on a number of top-level venues
  - We will have a new EU project focusing on the cloud starting in autumn this year

- Cloud continues to be one of the major research directions.
  - SIGCOMM 2013: 15/39 papers cloud computing and/or SDN

Summary & Outlook

- Cloud Sessions finished!
- What you should have learned:
  - What are the main concepts enabling cloud computing and how do these concepts work? (Virtualization, SDN, …)
  - What are the current mechanisms to enable efficient parallel processing of massive amounts of data and how do they work (MapReduce, Mesos, …)
  - An understanding of some research problems and solutions.
- Next week: No lecture!
- June 5th: Information Centric Networking - Introduction