Outline of Wireless Block

- Game theory and its applications
  - Game theory basics and concepts
  - Distributed Spectrum Sharing Application
- Social Group Maximization Framework
  - Introduction to the framework
  - Wireless Network Applications
- Mobile Data Offloading
  - Mobile cloud computing
  - Mobile media prefetching

Distributed Computation Offloading for Mobile Cloud Computing

Growing Mobile Devices

- **Growing mobile devices**
  - 526 million mobile devices and connections were added in 2013 (Cisco, Feb 2014)
  - Global mobile data traffic reached $1.5x10^{18}$ Bytes/Month at the end of 2013 (Cisco, Feb 2014)

- **Limited Mobile Resource**
  - Due to physical size constraint
  - Limited computation capability & limited battery capacity
  - Difficult to support resource-hungry apps, e.g., face recognition, augmented reality

Mobile Cloud Computing

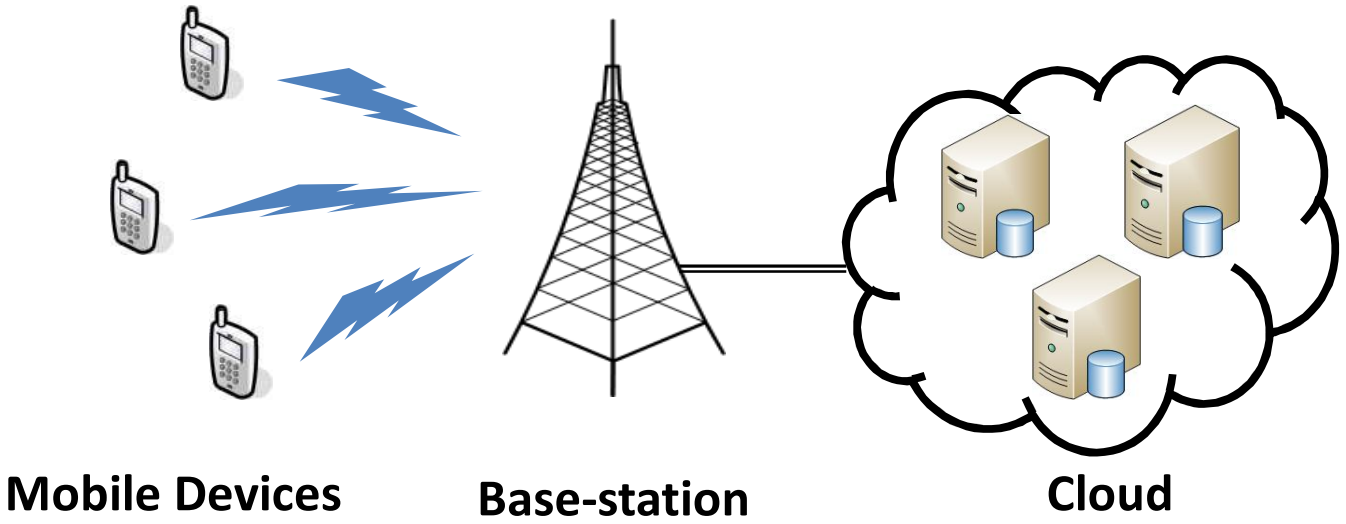
- **Mobile Cloud Computing (MCC)**
  - Augment capability of mobile devices by offloading computation to cloud via wireless access
  - Leverage resource-rich cloud infrastructure
  - Reduce computing time & save energy

- **Wireless Access Efficiency**
  - A critical factor for MCC
  - Mobile users are coupling
  - Simultaneous computation offloading → low wireless access efficiency

Game theoretic approach for interactive computation offloading decision making

System Model

- **Mobile Cloud Computing System**
  - A set of mobile device users N with computation intensive tasks
  - Computation offloading to cloud via a wireless access base-station, e.g., in a macrocell/ femtocell in cellular networks
  - Both communication and computation aspects play a key role



Communication Model

- **Mobile user *n* has computation offloading decision** $a_n \in \{0,1\}$
  i.e., to offload or not to offload

- **Uplink data rate for computation offloading of user *n*:**

$$R_n(\boldsymbol{a}) = W \log_2\left(1 + \frac{P_n H_{n,s}}{\omega_n + \sum_{m \neq n: a_n = 1} P_m H_{m,s}}\right)$$

  - $\boldsymbol{a} = (a_1, \ldots, a_N)$ → computation offloading decision profile of all users
  - W → channel bandwidth
  - P_n → transmission power of user *n*
  - $H_{n,s}$ → channel gain between user *n* and base-station *s*
  - ω_n → background interference

Computation Model

- **Mobile user *n* has a computation task** $I_n=(B_n,D_n)$
  - B_n → computation input data size, e.g., program codes and input states
  - D_n → CPU cycles to accomplish the computation task
- **Local computing approach (a_n=0):**

$$Time: T_n^l = \frac{D_n}{F_n^l}$$
$$Energy: E_n^l = v_n D_n$$

  - F_n^l → CPU cycles per second of user *n's* mobile device
  - v_n → energy consumption per CPU cycle
- **Cloud computing approach (a_n=1):**

$$Time: T_n^c = \frac{B_n}{R_n(\boldsymbol{a})} + \frac{D_n}{F_n^c}$$
$$Energy: E_n^c = P_n \frac{B_n}{R_n(\boldsymbol{a})}$$

  - F_n^c → CPU cycles per second allocated to user *n* by cloud
  - Time overhead = offloading + computing

Computation Offloading

- **Wireless access efficiency is bottleneck**
  - Too many mobile users choose computation offloading simultaneously
    → severe interference occurs
    → lead to low data rate for computation offloading
    → long transmission time and high energy consumption

- **Computation offloading decision makings for mobile users**
  - Users are coupling, "to offload or not" depends on other users' decisions
  - Centralized optimization is challenging
    → Combinatorial optimization and often NP-hard
    → Need complete information, leading to high overhead for information exchange

Distributed Computation Offloading Game

- **Motivations**
  - Mobile devices are owned by different individuals and they may pursue different interests
  - Leverage the power of crowds for devising distributed algorithm with low complexity
- **Game formulation**
  - Player set: mobile user set
  - Strategy space: computation offloading decision $a_n \in \{0,1\}$
  - Cost function: $V_n(\boldsymbol{a}) = \gamma_n^t Time + \gamma_n^e Energy$

    $\gamma_n^t = 1, \gamma_n^e = 0$ → user *n* wants to reduce delay

    $\gamma_n^t = 0, \gamma_n^e = 1$ → user *n* wants to save energy
- **Nash equilibrium** $\boldsymbol{a}^* = (a_{\boldsymbol{n}}^*, a_{-\boldsymbol{n}}^*)$ satisfies that

$$a_{\boldsymbol{n}}^* = argmin_{a_n \in \{0,1\}} V_n(a_{\boldsymbol{n}}^*, a_{-\boldsymbol{n}}^*), \forall n \in N$$

  - Stable system state, no user has incentive to deviate

Distributed Computation Offloading Game

- **Existence of Nash equilibrium**

> **THEOREM:** The distributed computation offloading game for mobile cloud computing admits a Nash equilibrium.

- **Structural property**

> **THEOREM:** The distributed computation offloading game for mobile cloud computing possesses the finite improvement property.

  - Any asynchronous better response update process (i.e., no more than one player updates the strategy at any given time) must be finite and leads to a Nash equilibrium

Distributed Computation Offloading Algorithm

- Distributed algorithm for computation offloading decision making
  - Motivation: enable mobile users to achieve a mutually satisfactory decision making, prior to the computation task execution
  - Key idea: based on the finite improvement property, i.e., let one mobile user improve its computation offloading decision at a time

Distributed Computation Offloading Algorithm

- Slotted structure for decision update → using clock signal from base-station for synchronization

Distributed Computation Offloading Algorithm

- The following operations are carried out during a decision slot:
  - Wireless interference measurement: each user locally measures the received interference generated by other users
    - User choosing computation offloading will transmit some pilot signal

Distributed Computation Offloading Algorithm

- The following operations are carried out during a decision slot:
  - Wireless interference measurement: each user locally measures the received interference generated by other users
    - User choosing computation offloading will transmit some pilot signal
  - Decision update contention: based on the interference measurement, each user can check whether it can improve by changing its decision:
    - If yes, content for decision update by informing the cloud with RTU (request-to-update) message
    - If no, do not content and keep the same decision in next slot

Distributed Computation Offloading Algorithm

- The following operations are carried out during a decision slot:
  - Wireless interference measurement: each user locally measures the received interference generated by other users
    - User choosing computation offloading will transmit some pilot signal
  - Decision update contention: based on the interference measurement, each user can check whether it can improve by changing its decision:
    - If yes, content for decision update by informing the cloud with RTU (request-to-update) message
    - If no, do not content and keep the same decision in next slot
  - Decision update scheduling: the cloud randomly selects one update requesting user and send the update-permission (UP) message to the user for updating its decision for the next slot

Distributed Computation Offloading Algorithm

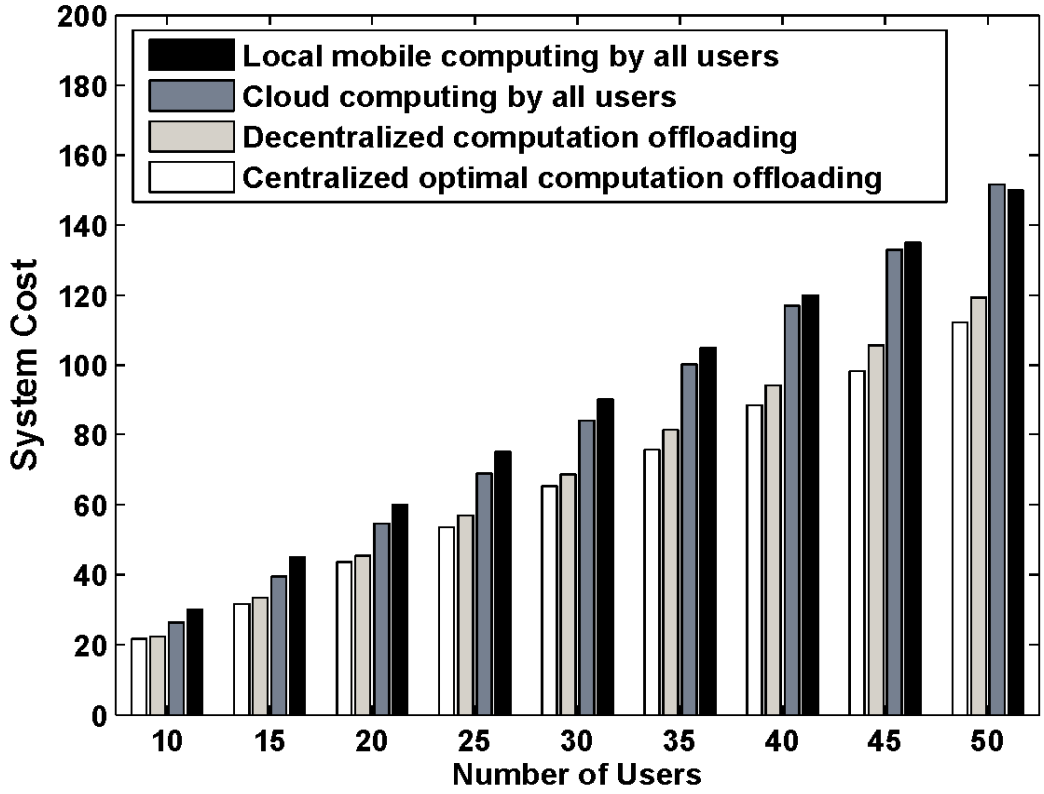
- The following operations are carried out during a decision slot:
  - Wireless interference measurement: each user locally measures the received interference generated by other users
    - User choosing computation offloading will transmit some pilot signal
  - Decision update contention: based on the interference measurement, each user can check whether it can improve by changing its decision:
    - If yes, content for decision update by informing the cloud with RTU (request-to-update) message
    - If no, do not content and keep the same decision in next slot
  - Decision update scheduling: the cloud randomly selects one update requesting user and send the update-permission (UP) message to the user for updating its decision for the next slot
- When no RTU messages are received, the cloud will broadcast the END message
- User will execute local/cloud computing according to achieved decisions at last slot (i.e., Nash equilibrium)

Numerical Results

- Mobile users randomly scatter over a square area 100mX100m
- Channel gain of a user depends on the distance from base-station
- Users have different computing tasks (e.g., face recognition) and different demands (delay reduction/energy saving)

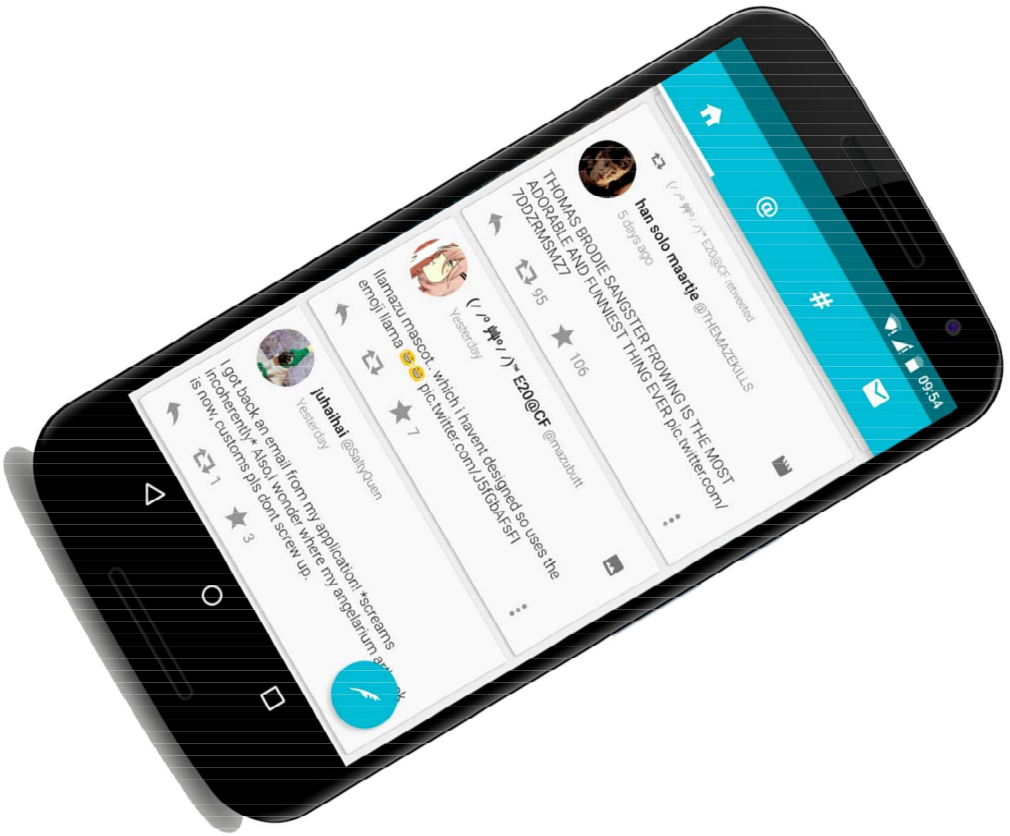
Numerical Results

- System-wide computing cost



- Up-to 33% and 38% cost reduction over local/cloud computing by all users, respectively
- At most 10% performance loss, compared with centralized optimum

Socially-Driven Learning-Based Mobile Media Prefetching

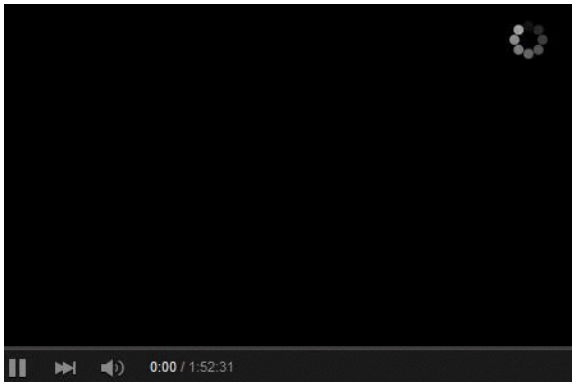


Background and Motivation:

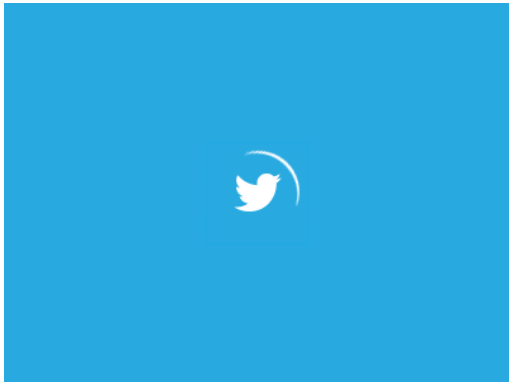
Waiting or Latency in Real-Life Media Usage

We are suffering from media access latency everyday

YouTube- Audios



YouTube-Videos

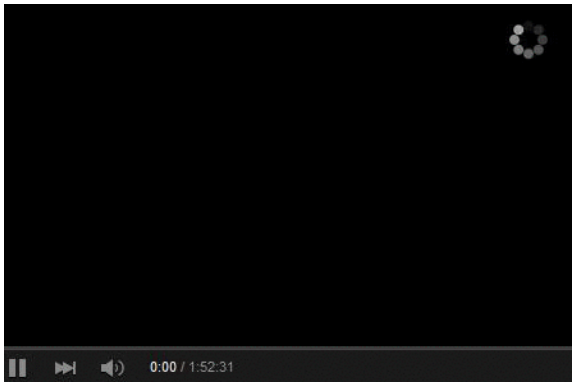


Twitter-Snapshots

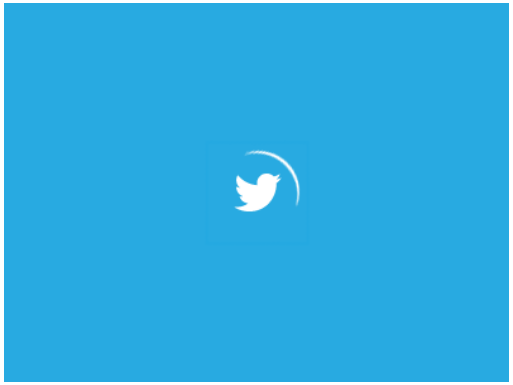
Google Picasa-Pictures

Background and Motivation:

Waiting or Latency in Real-Life Media Usage



YouTube- Audios



Twitter-Snapshots

Let us consider the Twitter media access case.

This would be an emergency QoE issue particularly in the mobile environment (e.g., taking subline, bus, etc).

Potential Solutions:

1) CDN-based accelerate approach

2) Prefetching, e.g.,

1) IMP (MobiSys 2012)
   - High prediction demands
2) EarlyBird (MobiHoc 2015)
   - Content-based solution

Prefetching: Promising but Challenging

Traditional approach:

Combine content-based analysis and machine learning/data mining based solutions.

Challenging:

What to prefetch? *(content selection is too hard!)*
When to prefetch? *(how to schedule the mechanism?)*
How to prefetch? *(in which mobile environment?)*

Nolvety:

It is straightforward to rethink the possibility whether ***what you want to see*** also strongly depends on ***whom published or shared it***?

Question*: We want to use socially-driven prefetching, is that possible?*

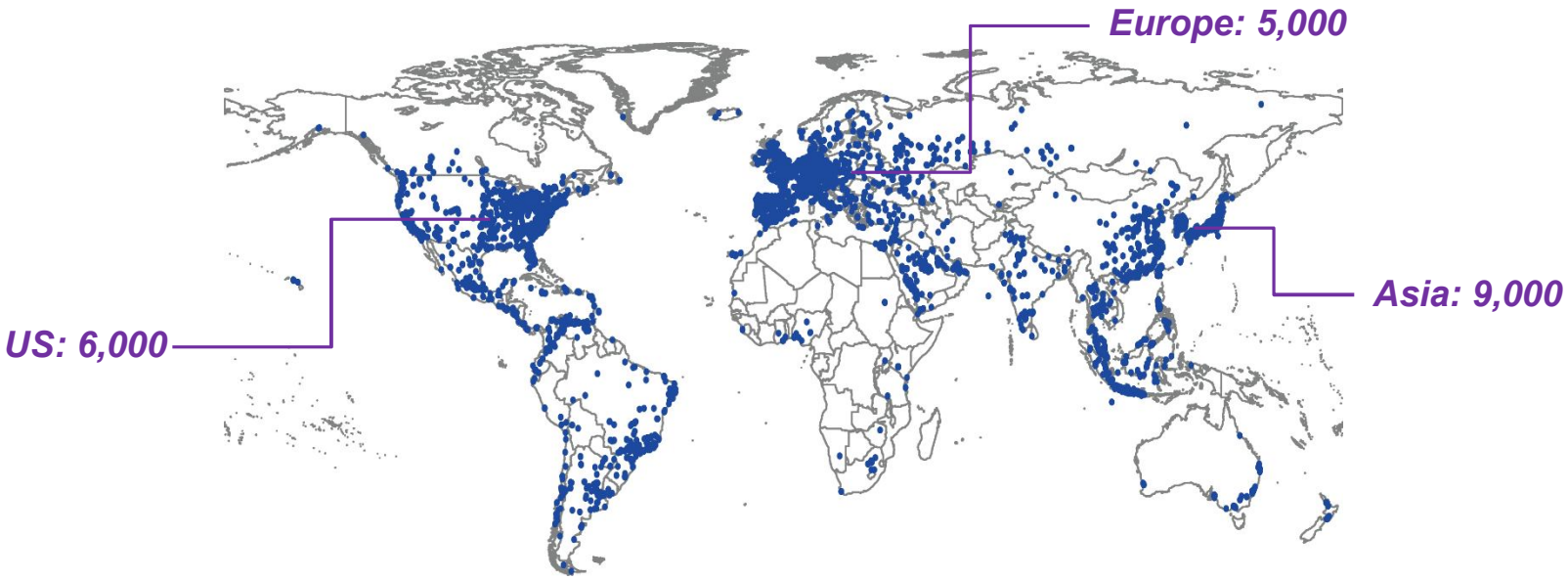
Early Stage Measurements:

Usage Statistics Selection

Table 1. Data collection from Twidere Android app on Google Play

Events	Content
App Launch and Close	Timestamp
Network Availablity	Timestamp, connection pattern
Tweets Click	Timestamp, tweet's attributes, participant
Embedded Media Preview	Timestamp, tweet ID, preview URL
Media Click	Timestamp, tweets ID list, link URL
Coarse Location	Timestamp, coordinates
Others	Tweets' favorite, retweet, and publish

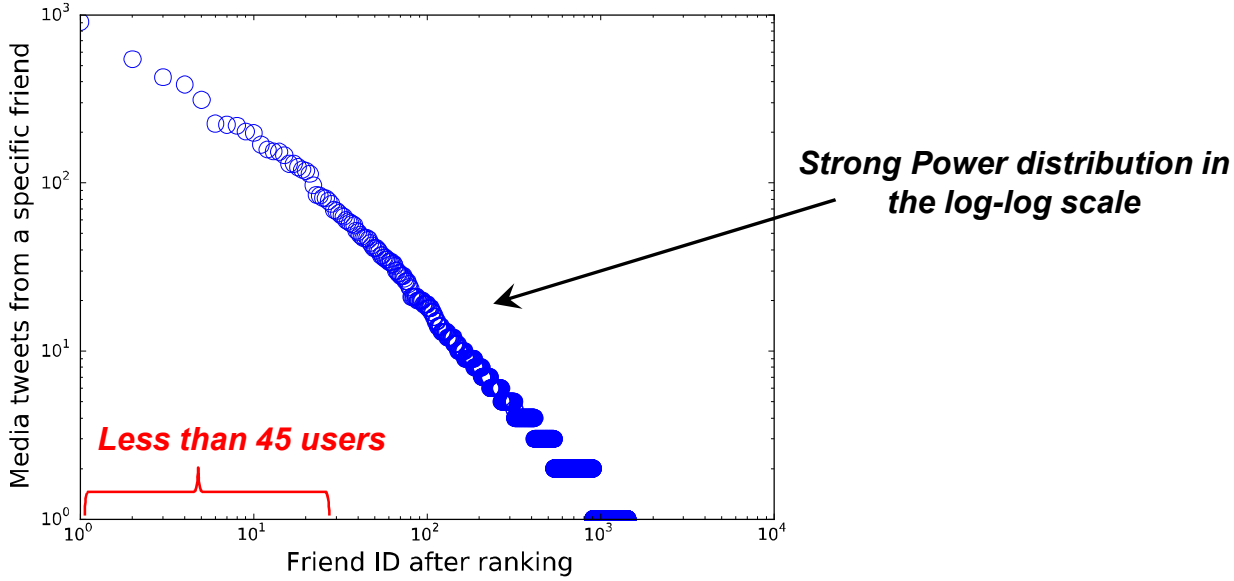
Early Stage Measurements: Dataset Description



- Duration: March 2015 to June 2015.
- Demographical Composition: More than 20,000 users around the world.
- Data Size: more than 233 million tweets, among which 9.2 millions are clicked and 72% of the clicked tweets contain media file.

Observations and Insights from Measurements

Mobile media object ranking

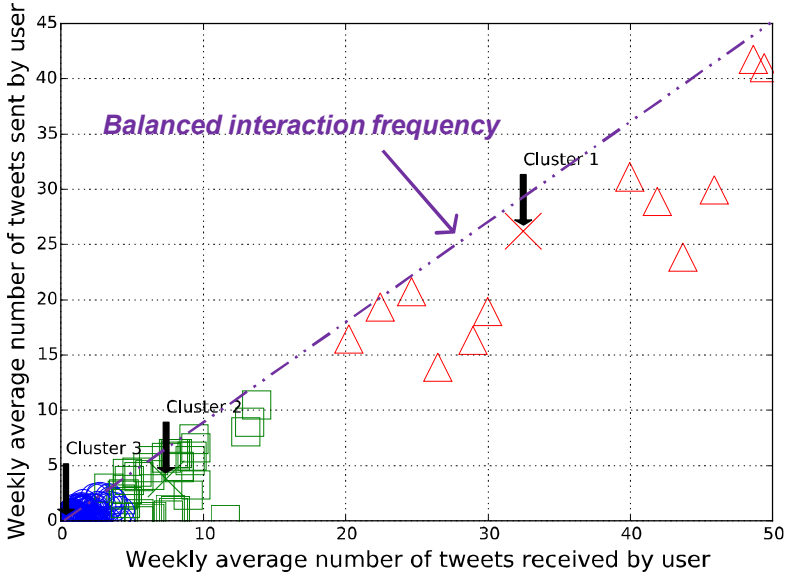


Observation 1: We observe a strong power law phenomenon, i.e., almost 85% of the tweets are from only a few friends (less than 5%), and most other friends have little contribution. This demonstrates that friendship (or social interaction strength) plays a critical role on shaping her usage behavior on Twitter.

Insight 1: It motivates us to design the socially-driven prefetching mechanism by leveraging the **social friendship closeness** among users.

Observations and Insights from Measurements

Social friendship clustering



Observation 2: we utilize the number of tweets received from a specific friend and the number of tweets sent by the user to that friend as the clustering features, and cluster the set of her friends into three types: ***close friends (i.e., Cluster 1)***, ***familiar friends (i.e., Cluster 2)*** and ***unfamiliar friends*** with infrequent contacts ***(i.e., Cluster 3)***.

Insight 2: It motivates us to design the learning algorithm in a perspective of ***social closeness clustering***.

Observations and Insights from Measurements

How social clustering impact user behavior?

Table 2. A user's click probability distribution in different cases.

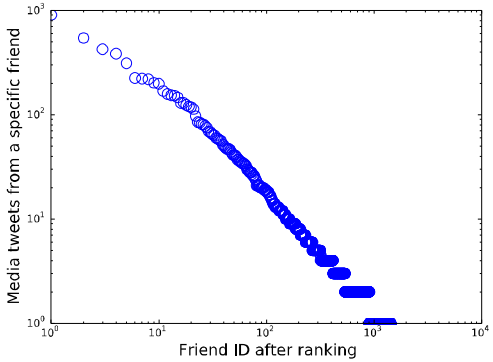
		close (%)	familiar (%)	unfamiliar (%)
Total Statistics	*Media click*	64.36	28.49	13.22
Network Features	*On WiFi*	71.26	19.69	1.97
	On cellular	49.43	13.66	1.47
Interaction Features	*Published by*	33.96	13.46	32.44
	Mentioning	33.96	13.46	9.01
	Favored by	98.99	97.55	0.52
	Retweeted by	97.66	9.57	0.06
	Replied by	50.00	28.49	23.40

Observation 3: The user will click the media file with a probability of ***0.64***, **0.28**, *0.13*, when the media tweet is sent by a close, familiar, and unfamiliar friend, respectively. .

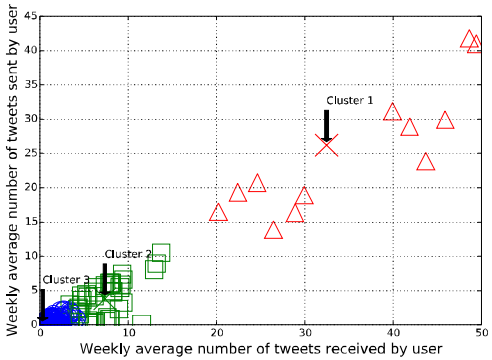
Insight 3: It comforts our assumption that social closeness really matters.

Key Insight: Social Closeness Plays A Critical Role on User's Mobile Media Usage

Observation 1



Observation 2



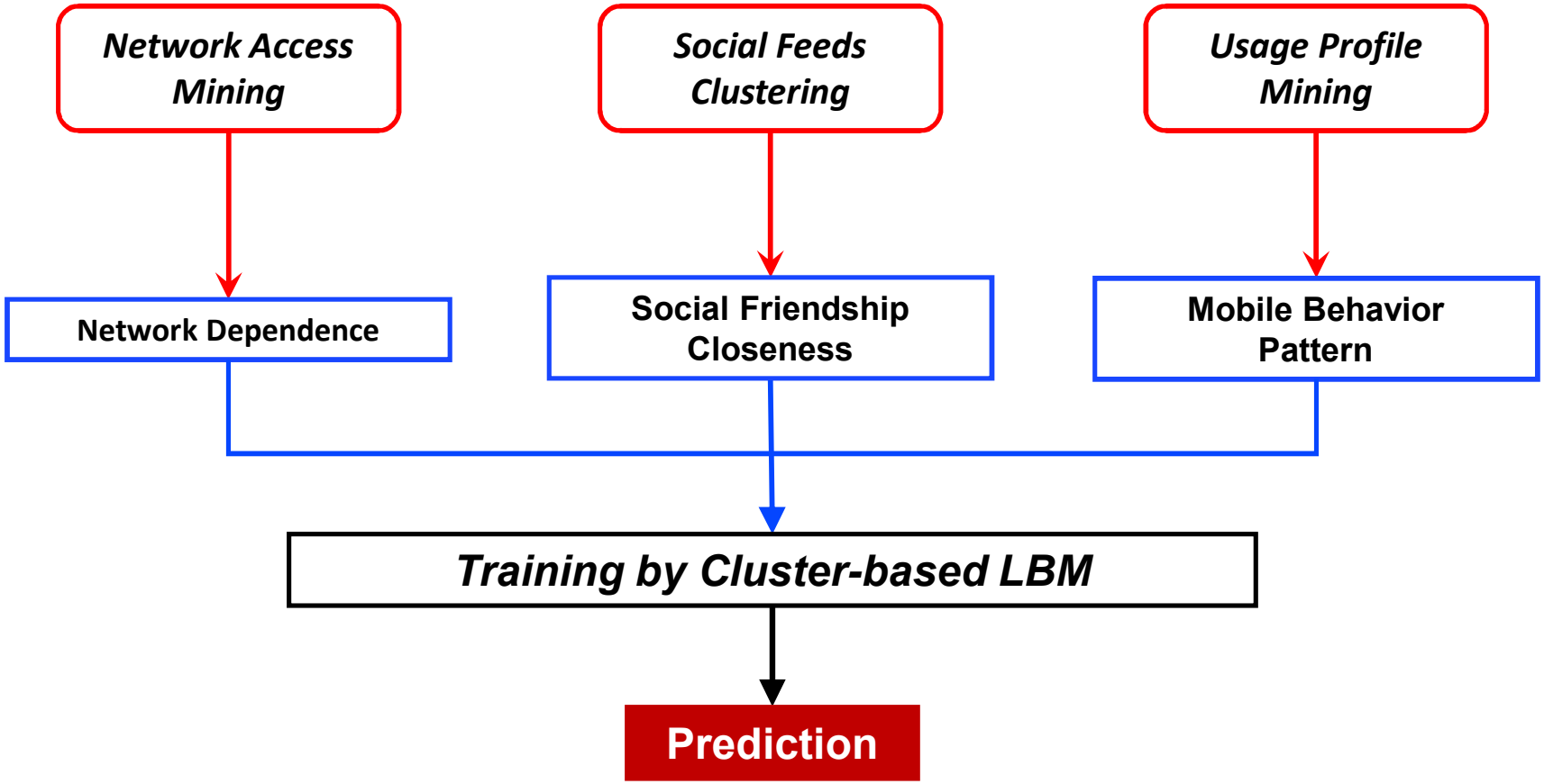
Observation 3

		close (%)	familiar (%)	unfamiliar (%)
Total Statistics	*Media click*	64.36	28.49	13.22
Network Features	*On WiFi*	71.26	19.69	1.97
	On cellular	49.43	13.66	1.47
Interaction Features	*Published by*	33.96	13.46	32.44
	Mentioning	33.96	13.46	9.01
	Favored by	98.99	97.55	0.52
	Retweeted by	97.66	9.57	0.06
	Replied by	50.00	28.49	23.40

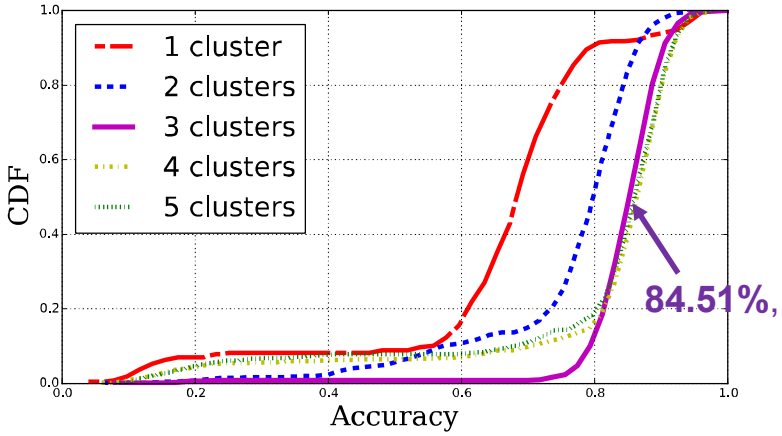
Based on the above observations and insights, we propose

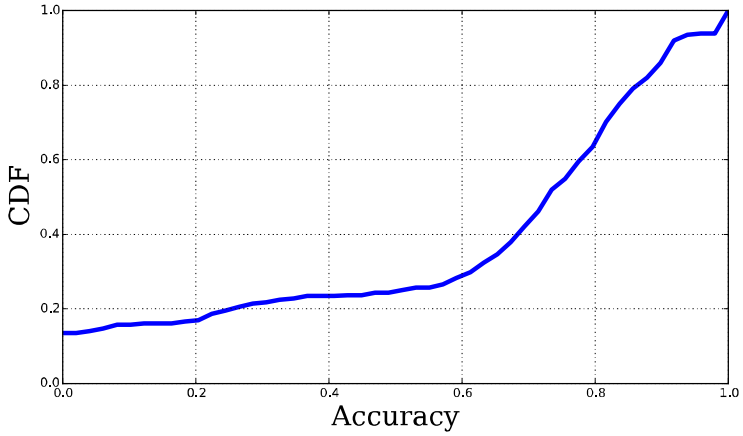
Spice: Socially-Driven Learning-Based Mobile Media Prefetching

Learning Feature Selection



Trace-Driven Evaluation of the Learning Algorithm





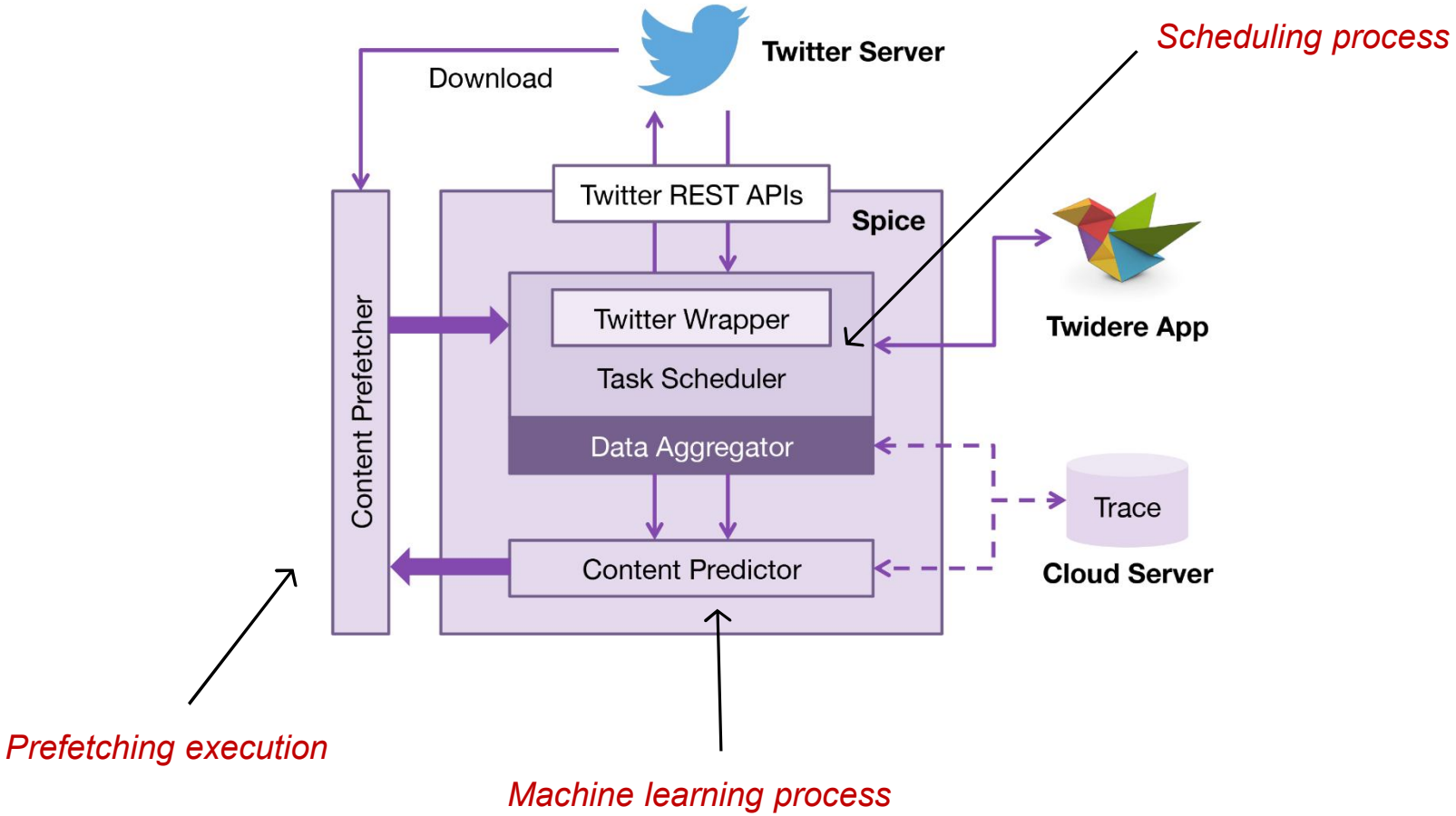
Cluster-based LBM

Linear Regression

With the average prediction accuracy of 65.54%, 77.93%, **84.51%**, 80.38%, 80.03% for 1, 2, ..., 5 clusters, respectively

With the average prediction accuracy of **63.82%**

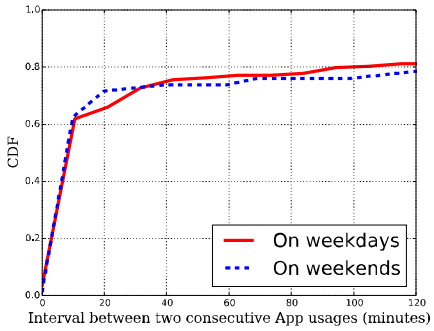
System Overview



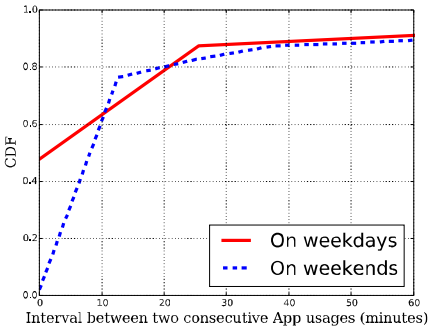
Spice: Usage Adaptive Scheduling

Observations

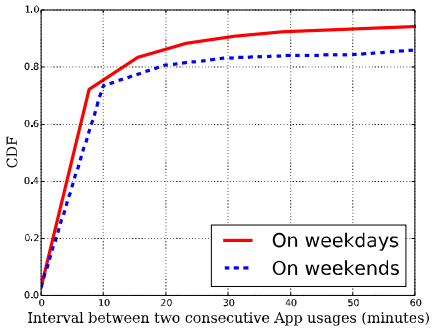
Interval distribution of two consecutive app usage (in 4 time zone):



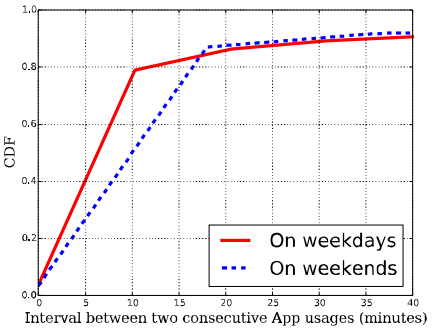
01:00 – 06:59 (midnight)



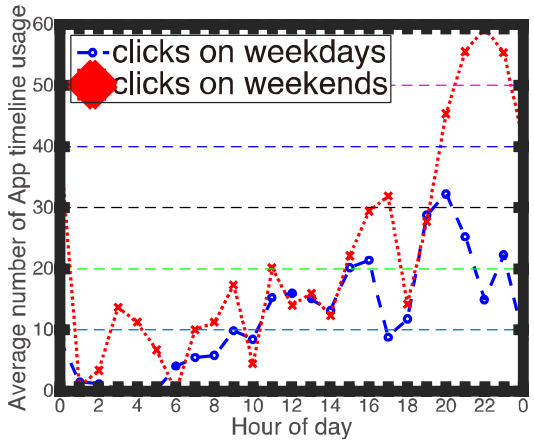
07:00 – 12:59 (morning)



13:00 – 18:59 (afternoon)



19:00 – 00:59 (night)

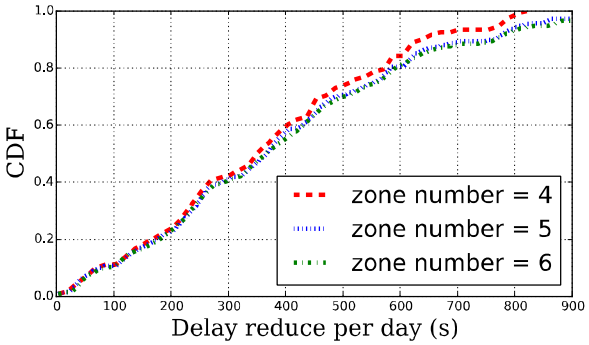


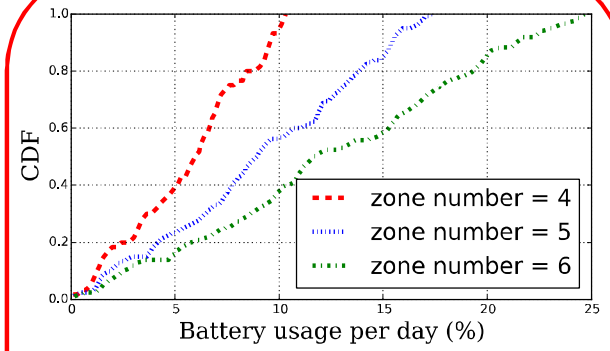
Media tweets consumption in different time slot

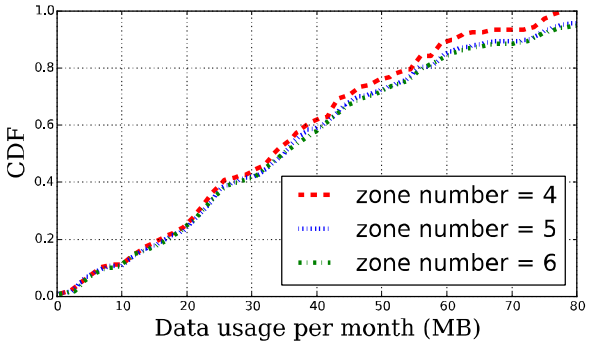
Using mobile user behavior pattern to schedule the prefetching task

Case Study of the Time Zone Number

Trace-driven emulations with *the most active 1,000* users







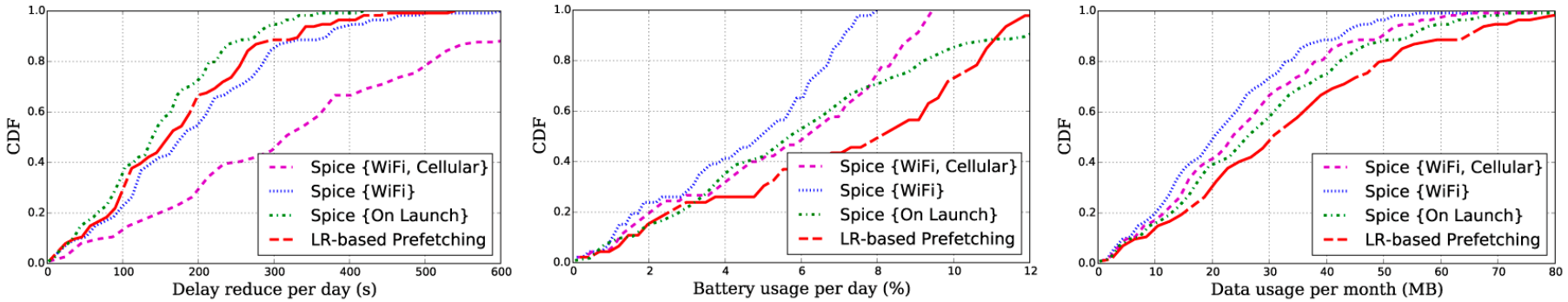
Rarely impact on benefit

Significantly impact on battery overhead

Rarely impact on data consumption

For most users, 4 time zone should enough, while it still changes with different OSNs and users

Trace-Driven Emulation Evaluations



Results:

- ***Spice with WiFi and cellular*** achieves the best performance of 67.2% delay reduction per day. It uses 27.3 MB cellular traffic per month and 5.3% battery usage per day on average.
- ***Spice with WiFi only*** achieves the second best performance of 58.4% delay reduction per day. It achieves the lowest cost among all the prefetching strategies, with 21.4 MB per month and 4.6% battery usage per day on average.
- ***LR-based*** prefetching can only achieve 54.4% delay reduction with 31 MB cellular traffic per month and 6.8% battery usage per day on average.
- Compared with the case without prefetching (on launch), Spice with WiFi only consumes less cellular data traffic with an increase of around 1.4% battery usage per day on average