Reducing Training Overhead of Large Time-Scale Transfer Scheduling for Mobile Devices

Author: **Ting-An Lin**

Collaborator : Yichuan Wang, Xin Liu (UC Davis)

Some slides were created by Y. Wang

New Content Life Cycle

Content

Problems in Mobile Network

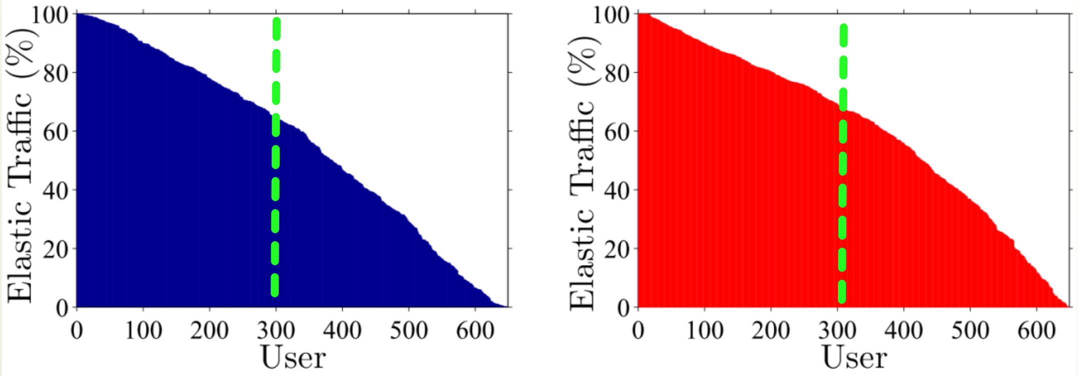
Network overload

Sporadic connectivity

Battery constraints

Mobile users can tolerate some delay

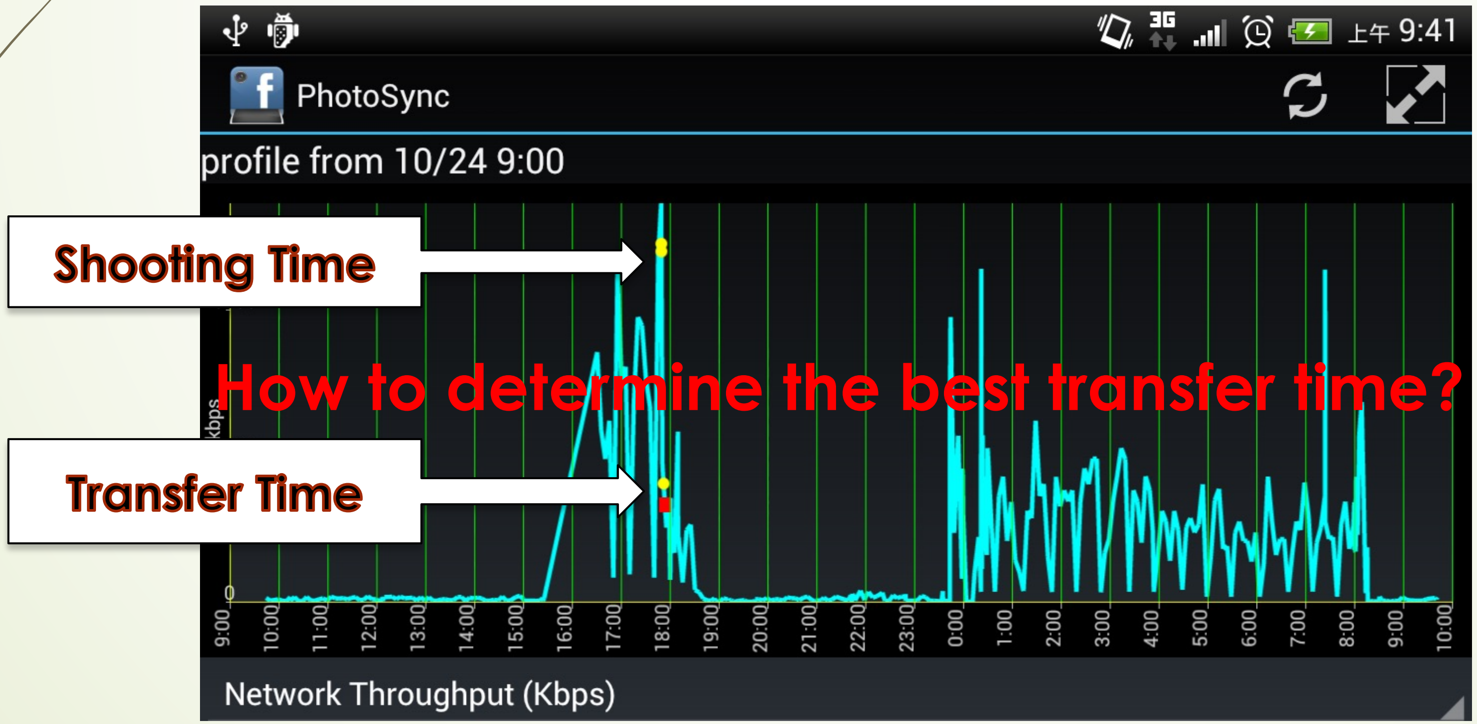
- 55% of mobile multimedia contents are uploaded after 1+ day [Trestian INFOCOM'11]
- We collect traces from1400+ users for 5 months and categorize the application traffic into:
  - Elastic
  - Real-time



More than 50% users generate at least 65% and 70% elastic traffic in downlink and uplink

Dynamics of Network Condition

- Network condition varies over time
- We are all different ← user profiles



Problem Statement

- Scheduling delay-tolerant transfer on mobile devices
- The scheduling should guarantee user experience
- The scheduling supports different optimization objectives
  - Minimizing network resource consumption
  - Minimizing energy consumption
  - Minimizing access cost

Proposed Solution

- Propose large time-scale transfer scheduling
- There has a deadline constraint in the scheduling to guarantee user experience
- Design and implement a framework to schedule the data transmission on mobile devices

Outline

- Large Time-scale Transfer Scheduling
- UPDATE Framework
- Trace-Driven Simulations
- Model Derivation Overhead
- User Clustering for Reducing Model Derivation Overhead
- Conclusion and Future Work

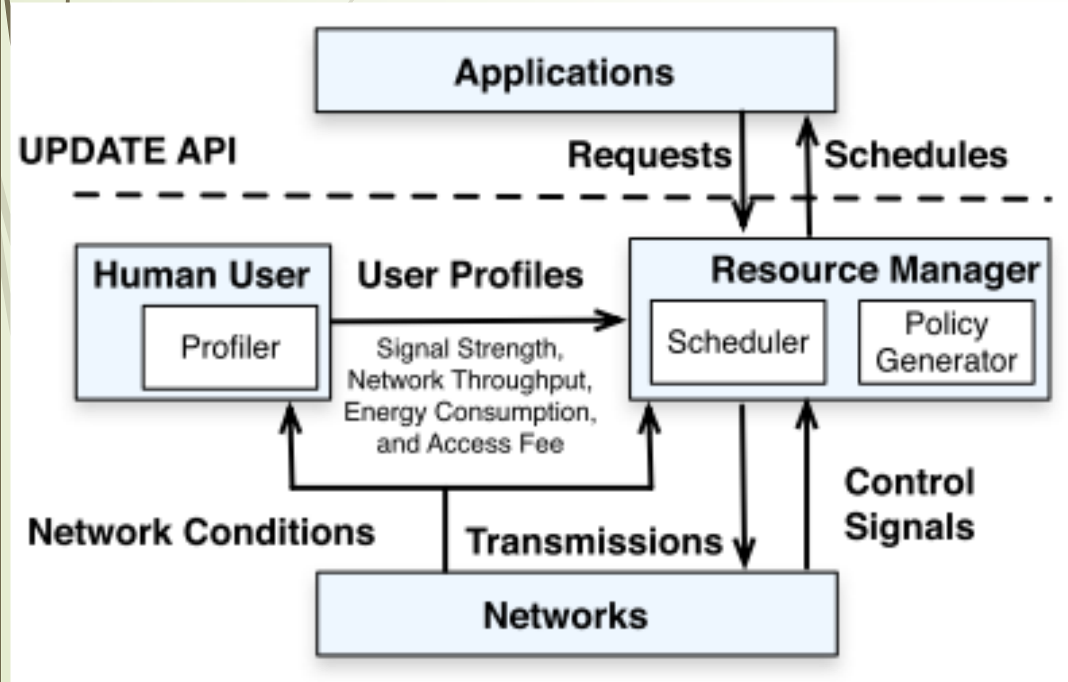
Large Time-scale Transfer Scheduling

Large Time-scale Scheduling

- Most of the existing work consider small time-scale, from seconds to minutes
- We consider a large time-scale scheduling, from minutes to hours
- There is an application-specified deadline for guaranteeing good user experience
- Can optimized for different optimization criteria, such as throughput, energy and network load

UPDATE Framework

UPDATE Framework



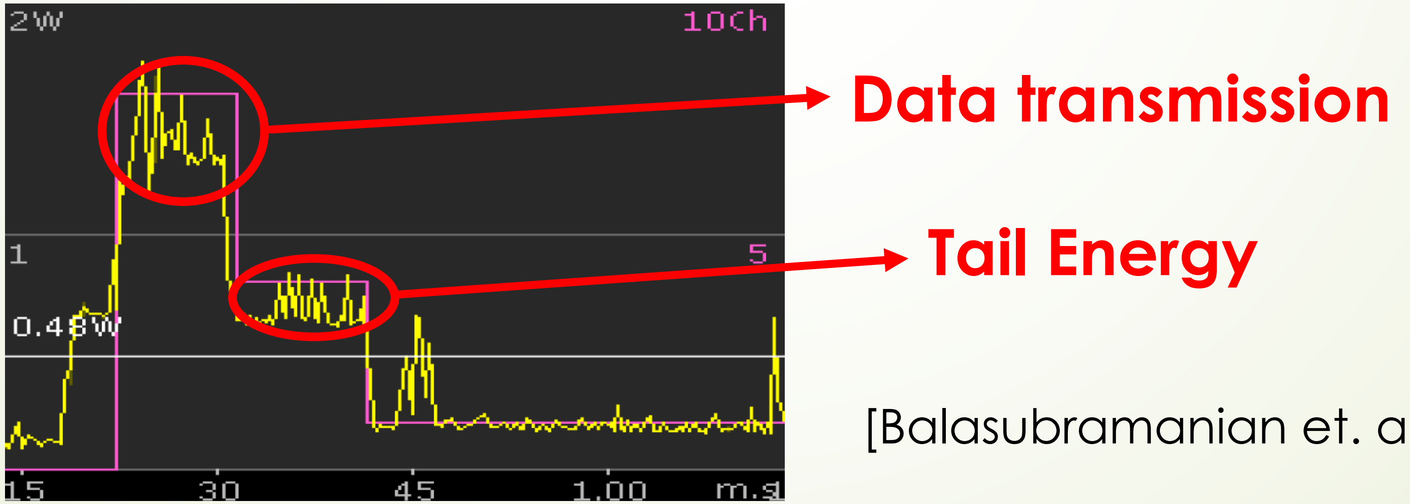
- **Profiler**: collect user profiles
- **Resource Manager**: manage the transfer requests
  - **Policy Generator**: generate scheduling policies based on user profiles
  - **Scheduler**: make scheduling decision according to the policies
  - **Application API**: the interface of mobile application and UPDATE framework

Optimal Stopping Scheduling (**OSS**)and Lightweight Optimal Stopping Scheduling (**OSS_L**) (NOSSDAV'12)

- The algorithms are based on Markov decision process (optimal stopping problem)
- The algorithms decide the best transfer time without exceeding the deadline ***N***
- The algorithms compare the current transfer cost X_t and the expected optimal cost between ***t*** and the deadline ***N*** in each timeslot
- In *OSS_L*, the transfer cost only depends on time

Batched Optimal Stopping Scheduling (***BOSS***) and Lightweight Batched Optimal Stopping Scheduling (***BOSS$_L$***) (MobiCom'13 under review)

- In UMTS networks, waste in tail time is significant when serve small network transfers
- We consider batching in scheduling: combine multiple small data into a single transfer

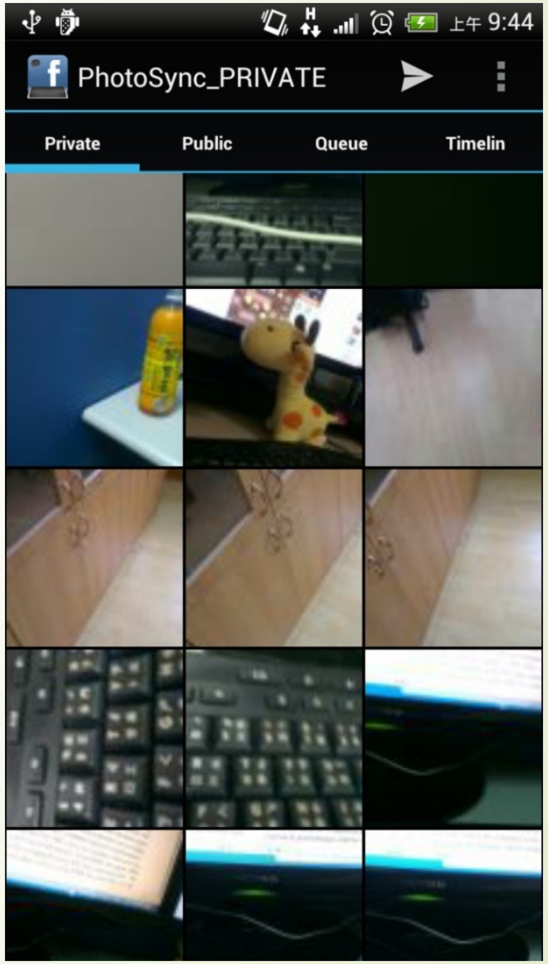


[Balasubramanian et. al IMC'09]

Trace-driven Simulations

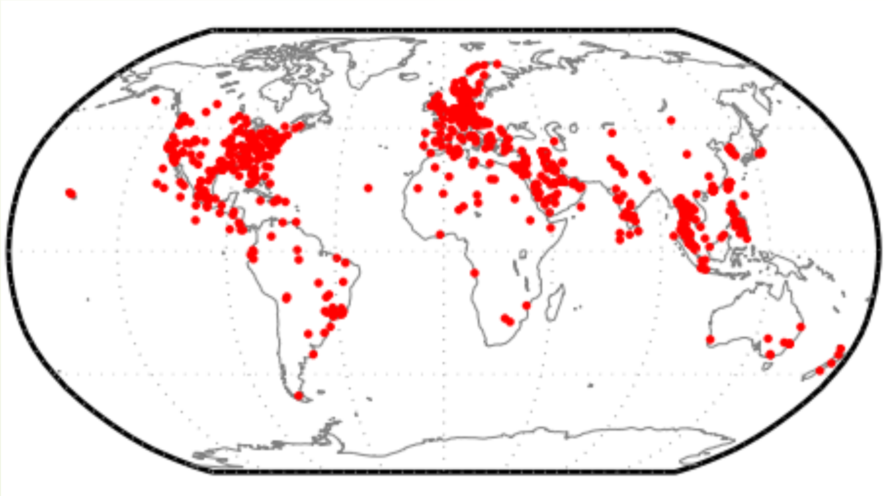
PhotoSync and Profiler

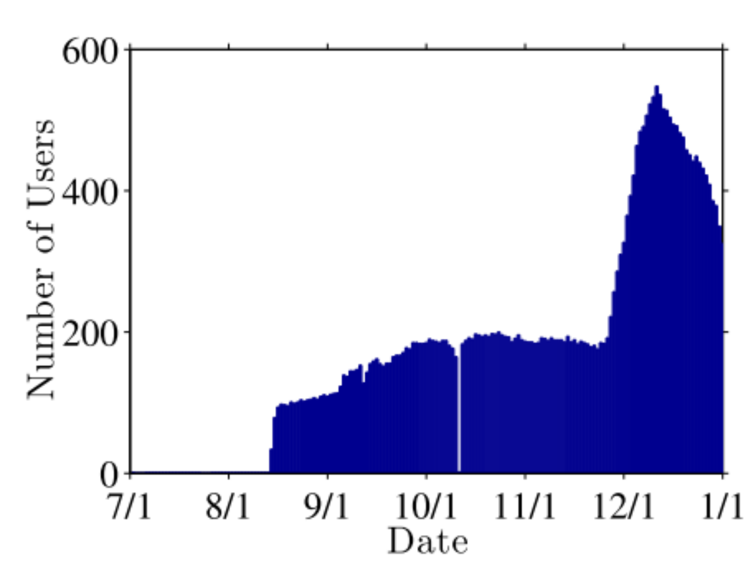
- PhotoSync is an Android application which uploads photos to Facebook automatically
- We also implement the profiler to record the user contexts and periodically upload the profiles to a server
- We publish PhotoSync with profiler to Google Play Store and collect profiles form 1400+ users for 5 months



Profiles Analysis

- The analysis shows that the users are from worldwide
- The longest profile length is 136 days and there are up to 500+ users in some days





APP Traffic Analysis

- We roughly classify the top 10 applications traffic into two groups and the traffic is about 60% of total traffic
  - Delay-tolerant (multimedia content upload, Dropbox, …)
  - Real-time (Browser, Youtube, …)
- There are on average 65% (uplink) and 70% (downlink) delay-tolerant traffic

Trace-driven Simulator

- Driven by traces from real users
- Implemented in Matlab
- Running on a Linux server with 2.6 GHz AMD CPU
- We implemented the four proposed algorithms: OSS, OSS_L, BOSS, and $BOSS_L$
- Also implemented a baseline algorithm called Instant (INS) and an offline optimal algorithm (OPT) and two state-of-art algorithms named BAR (Schulman et. al MobiCom'10) and SALSA (Ra et. al MobiSys'10)

Trace-driven Simulations

- We report the results when optimized for throughput, network load and energy consumption
- We report the results of single-jobs and multi-jobs (with batching) transmission
- We report the complexity and performance of each algorithm
- We empirical choose the system parameters: 5 min timeslots and 40 min deadline

Algorithm Complexity

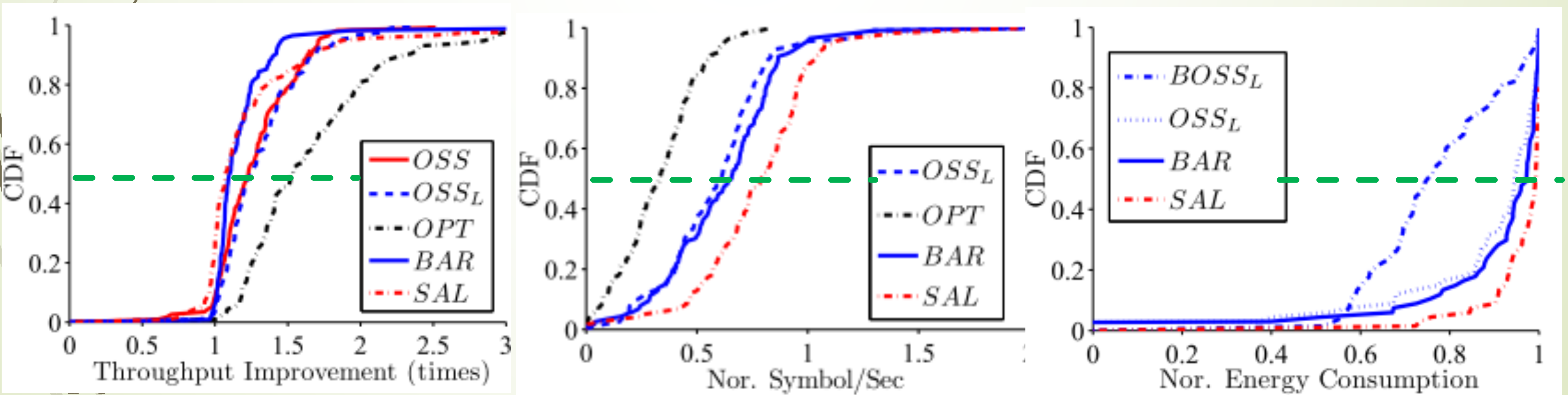
Time Complexity (sec)					
Deadline	2	3	4	8	16
OSS	24.65	63.33	102.56	457.20	3403.38
OSS_L	0.02	0.03	0.05	0.08	0.15
Memory Requirement during Computation					
OSS (GB)	0.25	0.56	0.99	3.96	15.82
OSS_L (KB)	2.25	3.38	4.5	9	18

Time Complexity (sec)					
Deadline	2	3	4	8	16
BOSS	2200.23	8403.63	-	-	-
$BOSS_L$	0.65	1.15	2.5	40	10229
Memory Requirement during Computation					
BOSS (GB)	3.95	15.82	-	-	-
$BOSS_L$ (KB)	9	18	36	576	147456

- OSS consume more time and memory compared with OSS_L
- The complexity of BOSS is too high can't work in real system

Simulation Results

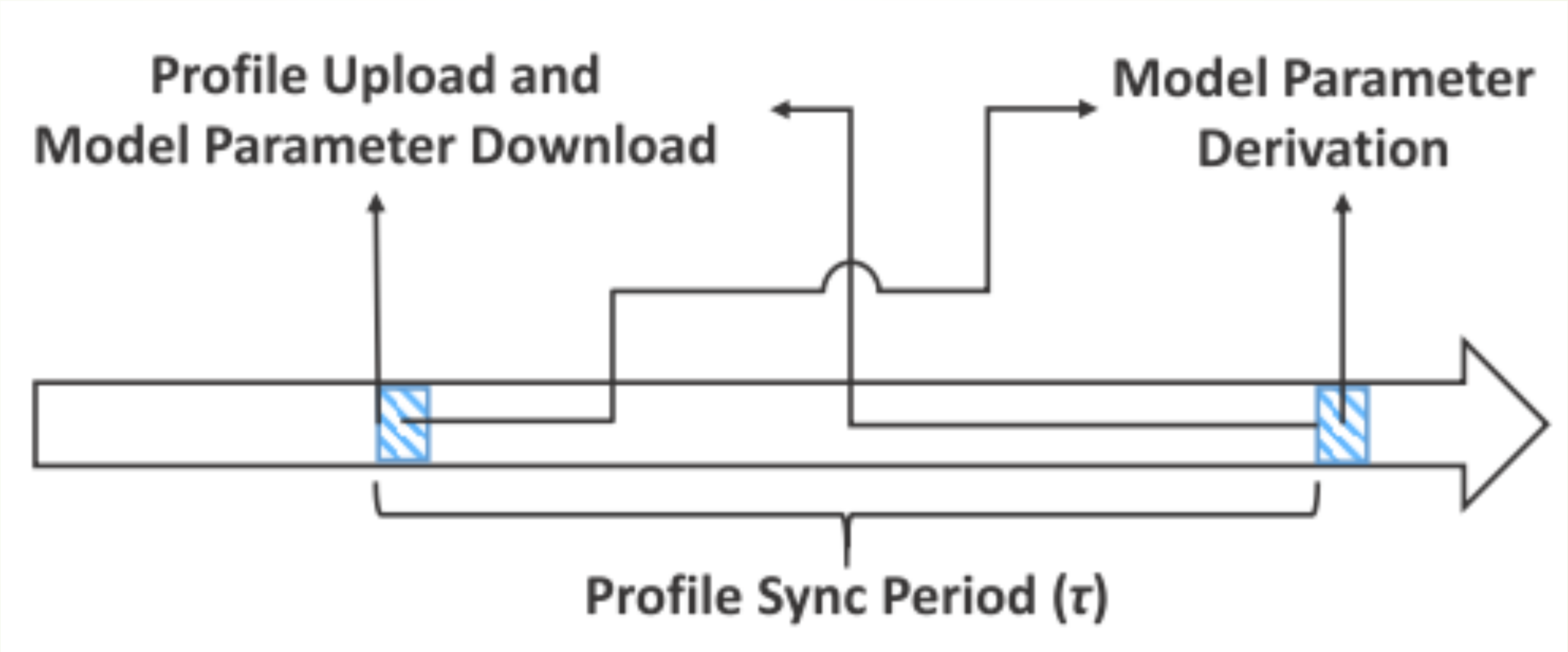
- We report the results with different optimized criteria
- Our algorithms outperform other algorithms
- $BOSS_L$ algorithm has better performance because of batching



Model Derivation Overhead

Model Derivation in Dynamic System

- Profiles upload to the server every *τ* days
- Training windows size ***L***: consider profiles in last ***L*** days when training model parameters



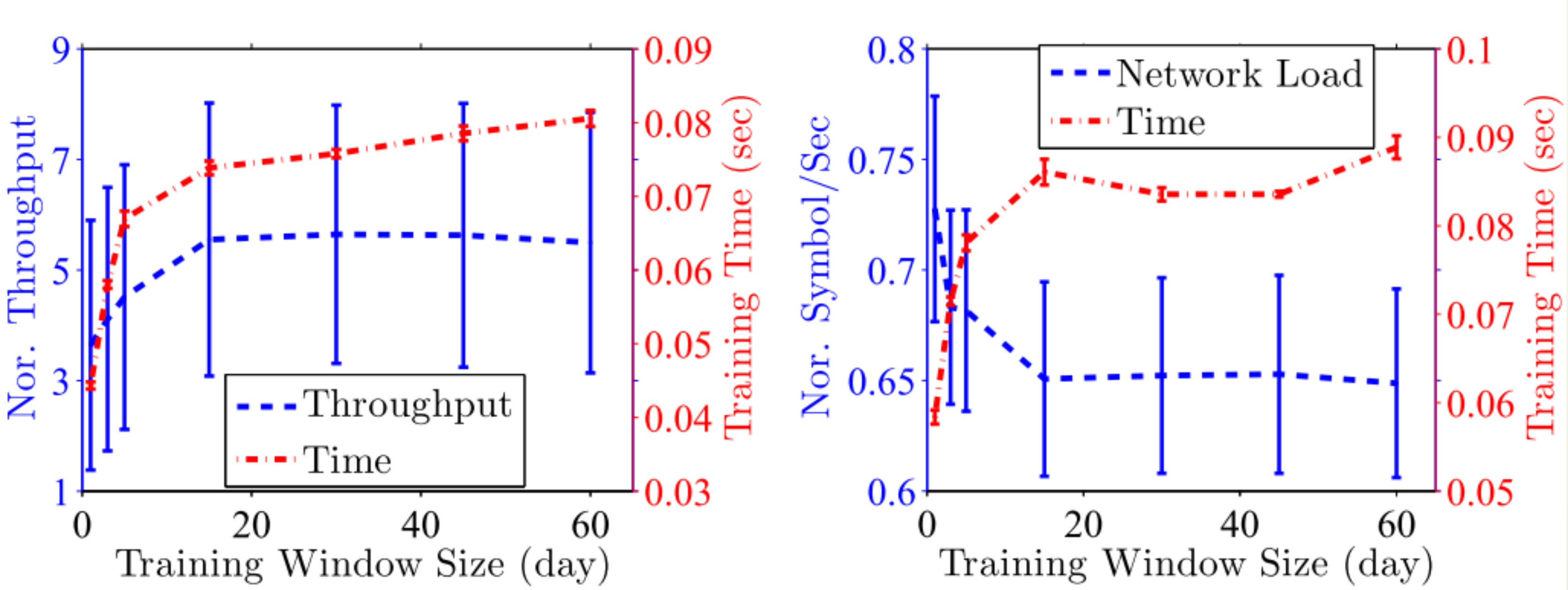
Limitations of OSS

- Even the OSS algorithm with the best training window size, OSS does not outperform OSS_L and consumes more resources

Opt. for Throughput with $L = 15$ days					
	Nor. Performance			Training Cost	
Algo.	Min	Mean	Max	Time (sec)	Memory
OSS	0.01	3.56	5.55	238.67	3.96 GB
OSS_L	0.01	5.55	14.16	0.06	9 KB
Opt. for Network Load with $L = 60$ days					
OSS	0.001	0.56	2.73	231.46	3.96 GB
OSS_L	0.001	0.65	22.98	0.07	9 KB
Opt. for Energy with $L = 30$ days					
OSS	0.003	0.93	4.5	278.82	3.96 GB
OSS_L	0.003	0.90	3.08	0.08	9 KB
$BOSS_L$	0.001	0.47	3.28	38.96	576 KB

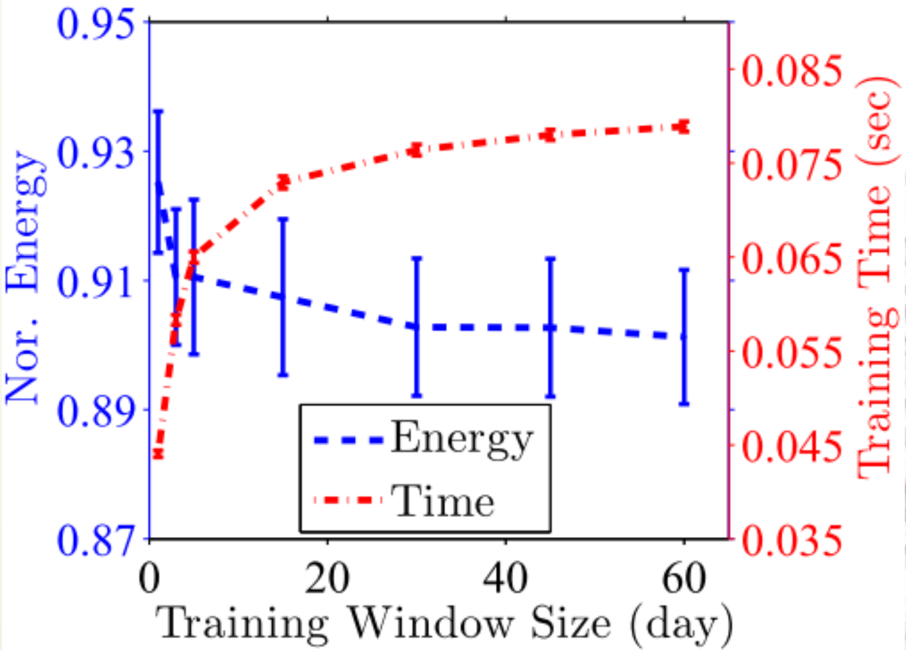
Implication of Training Window Size (single-job)

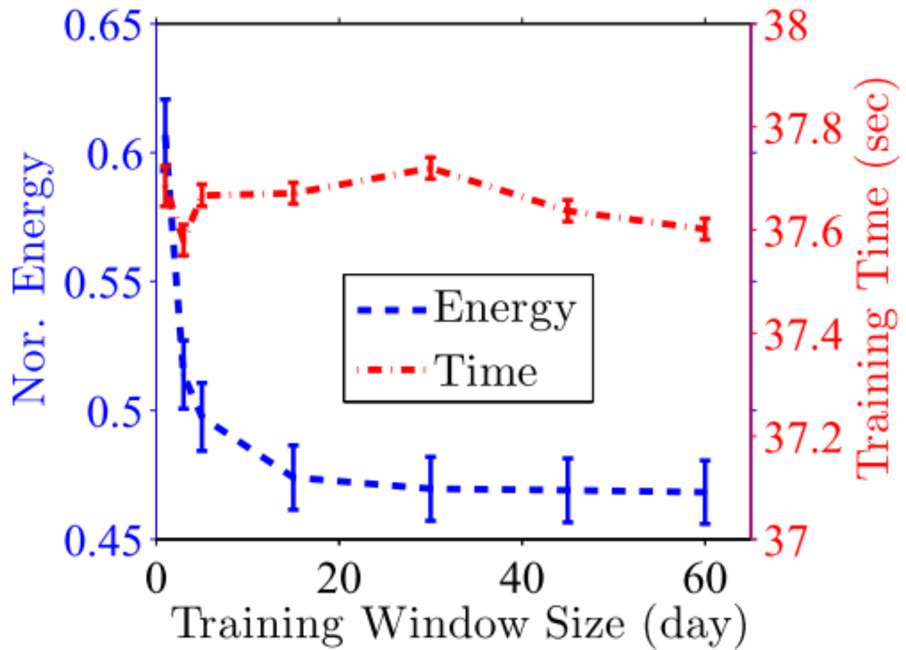
- Larger training window size causes longer training time
- We recommend $L \in [15, 30]$ and $L \in [30, 45]$ days when optimized for throughput and network load



Implication of Training Window Size (multi-job)

- The training time of BOSS$_L$ does not impact by training window size
- We recommend *$L \in [30, 60]$ when optimized for energy*





Training Window Size

- In summary, our algorithms preform well when L=30
- We consider L=30 in the rest of the experiments

User Clustering for Reducing Model Derivation Overhead

User Clustering

- To mitigate the model training overhead, we propose to cluster users then train a single set of model parameters for each group
- We cluster users according to the optimization criteria
- Two system parameters: timeslot size *T* for partitioning contexts and clustering ratio $\alpha = {}^{K}/_{N}$, where *K* is the number of clusters and *N* is the number of users
- We implement 3 clustering algorithms and 4 distance metrics in our simulator

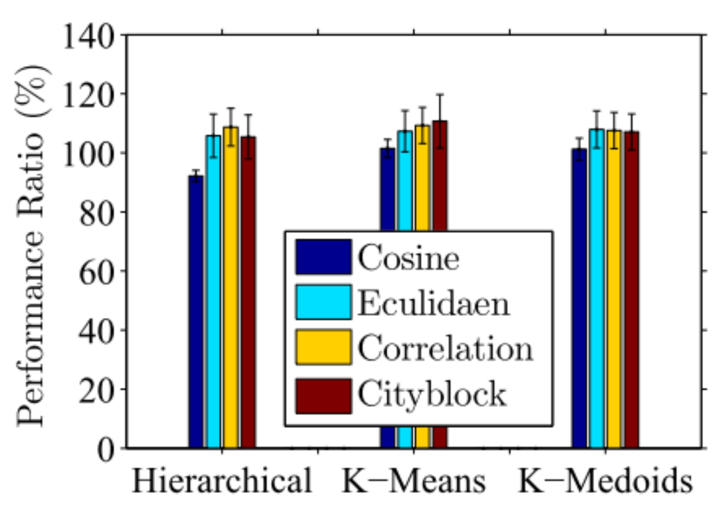
Clustering Algorithm Complexity

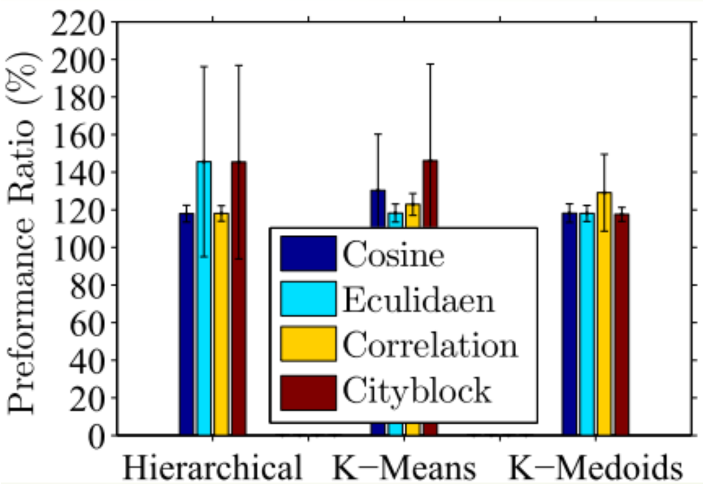
- K-Medoids algorithm consumes the longest time and hierarchical clustering is the fastest algorithm
- Cosine distance consumes the longest time and Cityblock distance consumes the shortest time

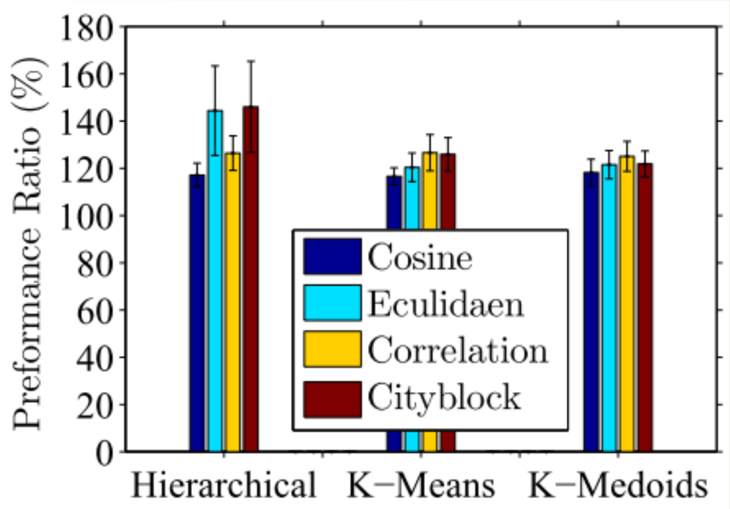
Average Running Time (sec)			
Distance	**Hierarchical**	**K-Means**	**K-Medoids**
Cosine	0.01	1.27	15.80
Euclidean	0.01	0.53	2.05
Cityblock	0.01	0.32	1.97
Correlation	0.01	0.86	2.13

Comparison of Clustering Methods

- We report the performance loss due to clustering
- Use K-Means/cityblock, K-Medoids/cityblock, and K-Means/cosine when optimized for: throughput, network load, and energy







Reducing Time Overhead and Performance Impact

- About 12% throughput improvement, 118% and 117% performance ratio when optimized for throughput, network load, and energy
- Total time savings with our user clustering algorithms are 58.8%, 37.5% and 59.9% when optimized for throughput, network load, and energy

Conclusion and Future Work

Conclusion

- We propose and implement UPDATE, a user-profile-driven framework to schedule data traffic for improved battery performance and network efficiency
- We study the overhead of training the model parameters
- In order to reduce the training overhead, we propose to cluster users
- In our simulation results, our proposed solution saves up to 59.9% on training time with <18% performance degradation

Future Work

- Classify user profiles into different profile types (e.g., weekday and weekend)
- Determine the best context for clustering users
- Propose new clustering approach that incorporates the batched transfers

Contributions

- Collect user profiles form general public
- User profiles analysis
- Quantify model derivation overhead
- Reduce model derivation overhead by clustering users

Publications

- Conference Papers
  - Yichuan Wang, Xin. Liu, Angela Nicoara, **Ting-An Lin**, and Cheng-Hsin Hsu, "Smarttransfer: Transferring your mobile multimedia contents at the "right" time". In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'12)*, Toronto, Canada, June 2012.
  - **Ting-An Lin**, Yichuan Wang, Cheng-Hsin Hsu, and Xin Liu, "Poster: Mobile user clustering in large time-scale data transfer scheduling". In *Proc. of ACM International Conference on Mobile Systems,* Applications, *and Services (MobiSys'13)*, Taipei, Taiwan, June 2013.
  - Shu-Ting Wang, **Ting-An Lin**, Yichuan Wang, Cheng-Hsin Hsu, and Xin Liu, "Poster: Fusing Prefetch and Delay-Tolerant Transfer for Mobile Videos". In *Proc. of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'13)*, Taipei, Taiwan, June 2013.

Publications(cont.)

- Conference Papers
  - Ting-Yi Lin, **Ting-An** Lin, Chung-Ta King, and Cheng-Hsin Hsu, "Context-Aware Decision Engine for Mobile Cloud Offloading". In *Proc. 2013 IEEE WCNC Workshop on Mobile Cloud Computing and Networking (MCC'13)*, Shanghai, China, April 2013.
  - Yu-Sian Li, Chien-Chang Chen, **Ting-An Lin**, Cheng-Hsin Hsu, Yichuan Wang, and Xin Liu, "An End-to-end Testbed for Scalable Video Streaming to Mobile Devices over Http". IEEE International Conference on Multimedia and Expo (ICME'13), San Jose, California, USA.
- Journals Paper
  - Yichuan Wang, **Ting-An Lin**, Cheng-Hsin Hsu and Xin Liu, "Region and action aware virtual world clients". *ACM Transactions on Multimedia Computing, Communications, and Applications*, Volume 9 Issue 1, February 2013.

Q & A

Backup

Scheduling Model

- Each transfer request has a specific deadline: ***N***
- At each time slot, the scheduler makes a decision $D_t \in$ ***{Wait, Transfer}***
- The decision are based on the current transmission cost (X_t, $t \in [1, N]$) and future estimates V_t
- V_t is the optimal cost to transfer data between time slot t and ***N***
- V_t can be calculated using the statistics of X_t

Optimal Stopping Scheduling (**OSS**)

$$D_t = \begin{cases} \text{Transfer}, & X_t \leq E(V_{t+1}|X_t); \\ \text{Wait}, & X_t > E(V_{t+1}|X_t). \end{cases}$$

$$E(V_N|X_{N-1}) = E(X_N|X_{N-1});$$
$$E(V_t|X_{t-1}) =$$
$$\sum_c P(X_t = c|X_{t-1}) \min(c, E(V_{t+1}|X_t = c)).$$

OSS requires longer user profiles to derive model parameters and with higher complexity

Lightweight Optimal Stopping Scheduling (***OSS$_L$***)

$$D_t = \begin{cases} \text{Transfer}, & X_t \leq E(V_{t+1}) \\ \text{Wait}, & X_t > E(V_{t+1}) \end{cases},$$

$$\begin{aligned} E(V_N) =& E(X_N); \\ E(V_t) =& P(X_t > E(V_{t+1}))E(V_{t+1}) \\ &+ P(X_t \leq E(V_{t+1}))E(X_t | X_t \leq E(V_{t+1})). \end{aligned}$$

The transfer cost X_t only depends on time in OSS_L

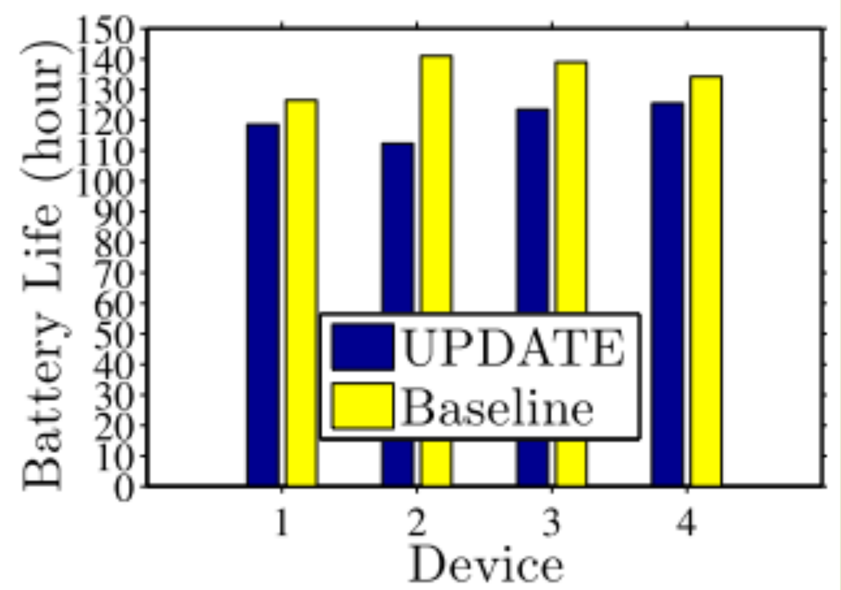
Collected User Profiles

Context	Profiling Type	Period (min)	Profiling Level ($\geq$)
WiFi Connectivity	Event-driven	-	Default
3G Signal Strength	Event-driven	-	Default
Activity Information	Periodical	5	Verbose
Task Information	Periodical	5	Verbose
Battery Level	Periodical	5	Baseline
Network Throughput	Periodical	5	Default
Application Traffic Amount	Periodical	5	Default
GPS Location	Periodical	30	Verbose
Neighboring WiFi AP Information	Periodical	30	Verbose
Neighboring Cell Tower Information	Periodical	5	Verbose

Profiler Overhead

- The average power overhead of our profiler is 2.94 mW, or 6% of the total power consumption
- The battery lifetime is longer than 4.5 days with the profiler running

Setup	Average (mW)	Min (mW)	Max (mW)
Baseline	48.9	47.1	50.5
UPDATE	51.84	46.9	56.3



Batching Scheduling Model

- Q denotes the scheduler queue at the beginning of the time slot; Q^{+} the queue after job arrivals; and Q^{-} the queue after transfer the job with the closest deadline
- We call a timeslot **active** if one or more content transfers are scheduled, otherwise the timeslot is **inactive**.
- A^{Q}_{t} / C^{Q}_{t} :the expected cost when using the optimal policy in active/inactive timeslot

Batched Optimal Stopping Scheduling (***BOSS***)

- If no job is transmitted in the current timeslot *t−1*

$$C_{t-1}^{Q} = C_t^{Q^+}, A_{t-1}^{Q} = C_t^{Q^+}.$$

- If schedules the job with the earliest deadline in the queue, and stays in the current timeslot *t −1*

$$C_{t-1}^{Q} = O + X_{t-1} + A_{t-1}^{Q^-}, A_{t-1}^{Q} = X_{t-1} + A_{t-1}^{Q^-}.$$

Batched Optimal Stopping Scheduling (***BOSS***) (cont.)

- In an inactive timeslot $t-1$, if , then transmit no requests, and go to next timeslot;
- Otherwise, transmit the first request, and stay in the current timeslot
- In an active timeslot $t-1$, if , transmit no more request, and go to the next timeslot
- Otherwise, transmit the request with the closest deadline, and stay in the current timeslot
- Complexity of BOSS is very high because the number of states is large

Lightweight Batched Optimal Stopping Scheduling (***BOSS$_L$***)

$$C_{t-1}^Q = P(X_{t-1} > C_t^Q - A_{t-1}^{Q^-} - O)C_t^{Q^+} \\ +P(X_{t-1} \leq C_t^Q - A_{t-1}^{Q^-} - O)(X_{t-1} + A_{t-1}^{Q^-} + O);$$

$$A_{t-1}^Q = P(X_{t-1} > C_t^Q - A_{t-1}^{Q^-})C_t^{Q^+} \\ +P(X_{t-1} \leq C_t^Q - A_{t-1}^{Q^-})(X_{t-1} + A_{t-1}^{Q^-}).$$

$$C_T^Q = O + |Q|E(X_T), \\ A_T^Q = |Q|E(X_T).$$

Energy Model

- We measure the energy consumption of HTC Sensation XE phone, using an Agilent 66321D power meter
- We place the phone in locations with different RSSI values, and compute the mean current of each location based on 100,000 samples
- We use the same setup to measure the WiFi ramp and cellular tail energy

WiFi Network Interface					
RSSI (dBm)	-81.24	-71.24	-60.94	-46.60	-36.6
Current (A)	0.28	0.26	0.25	0.24	0.23
Cellular Network Interface					
RSSI (dBm)	-91.65	-86.14	-73.16	-67.05	
Current (A)	0.33	0.26	0.22	0.21	

Vector Representation

Timestamp, context

⋮

Timestamped log files

$$\begin{bmatrix} c_{1,1} & \cdots & c_{1,T} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,T} \end{bmatrix}$$

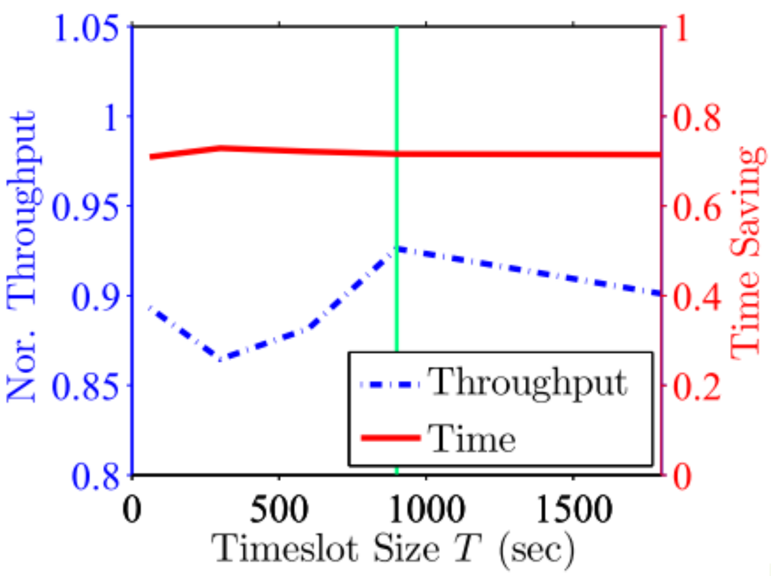
N days profiles and *T* timeslots per day

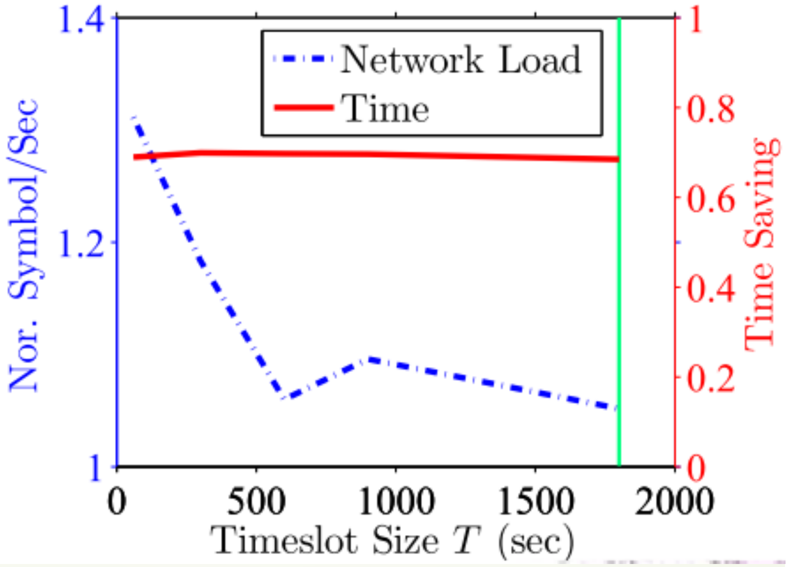
$$[v_1, v_2, \ldots, v_T]$$

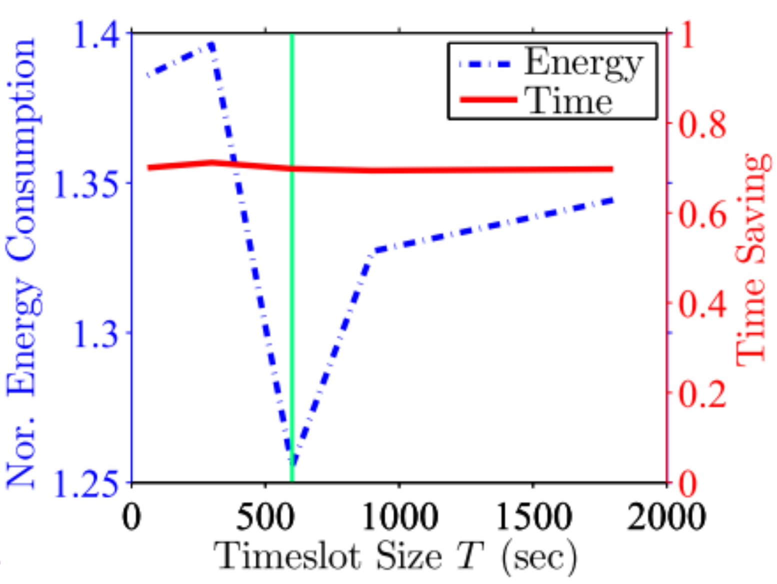
Average context value in each timeslot

Impact of System Parameters in User Clustering (***T***)

- The best *T* when optimized for throughput, network load and energy consumption are 900-sec, 1800-sec and 600-sec







Impact of System Parameters in User Clustering (*α*)

- Consider hierarchical clustering algorithm and cosine distance when clustering users
- OSS_L with clustering can achieves 92.6% of original throughput, 5.2% additional network load and 25.5% additional energy with only 30% of original model parameters training time when $\alpha = 0.3$