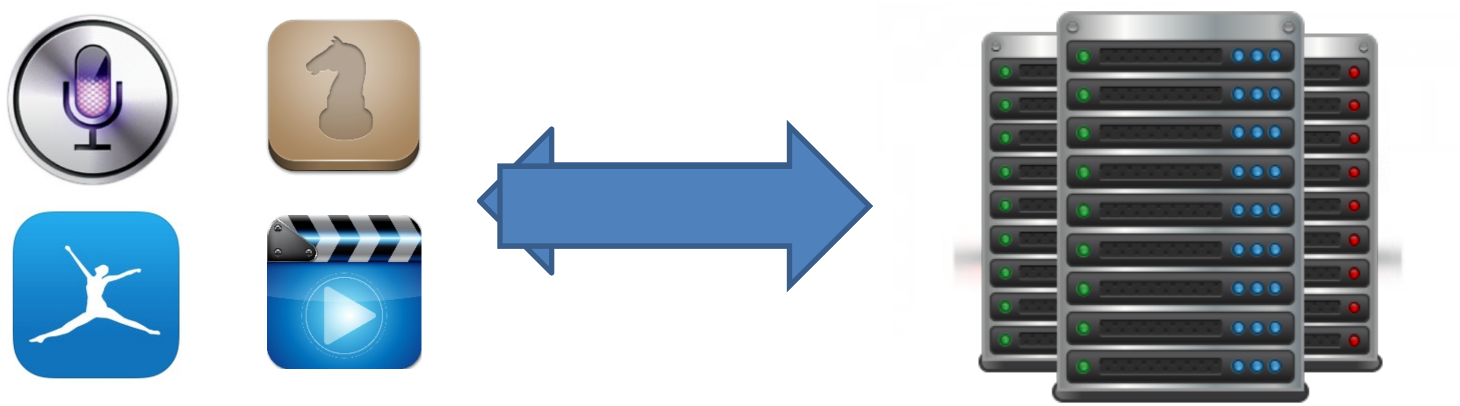
Application-Aware Traffic Scheduling for Workload Offloading in Mobile Clouds

Liang Tong, Wei Gao

University of Tennessee – Knoxville

Cloud Computing for mobile devices

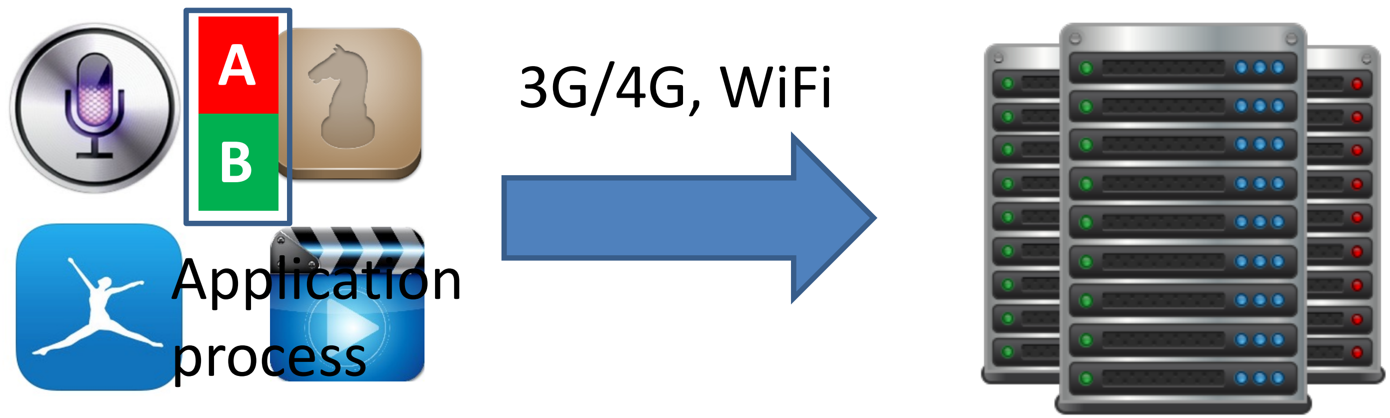
- Contradiction between limited battery and complex mobile applications



- Mobile Cloud Computing (MCC)
  - Offloading local computations to remote execution via wireless communication

Cloud Computing for mobile devices

- Wireless communication is expensive!



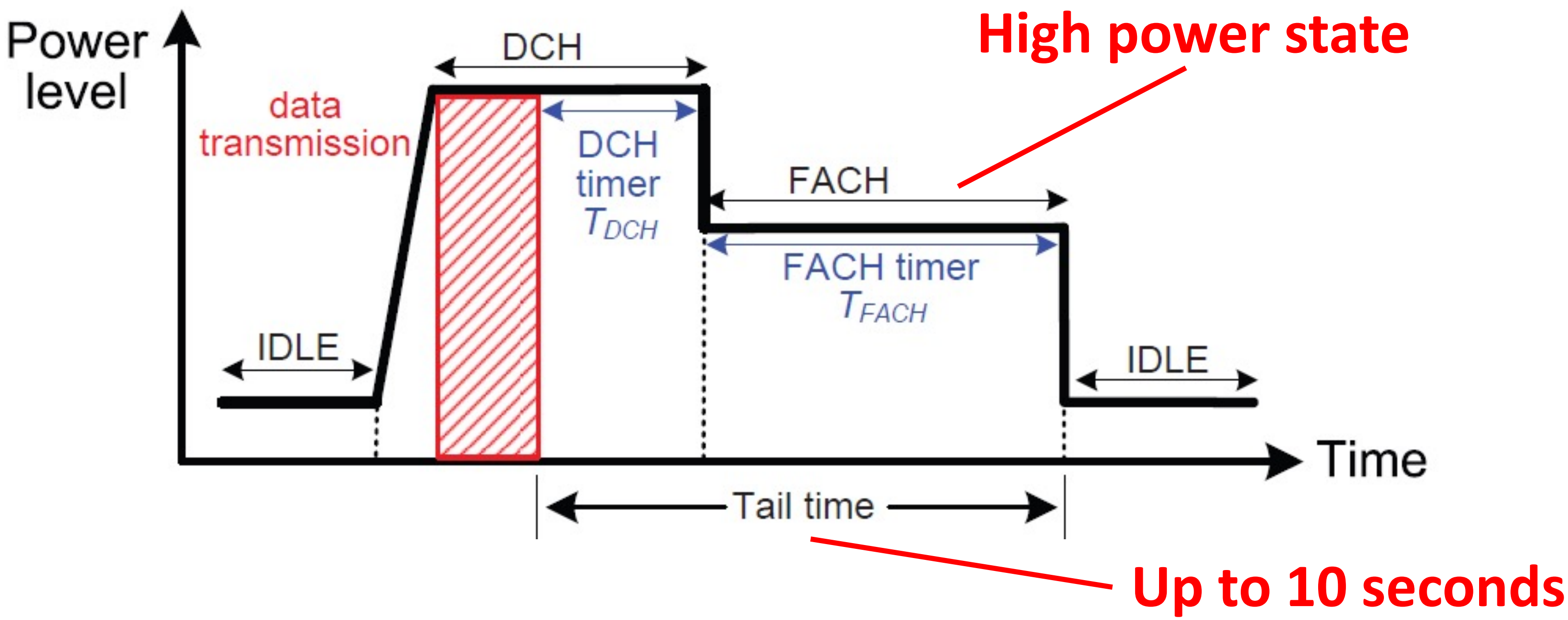
- Partitioning workloads at the method level

B Local execution > wireless data transmission

How to measure the cost?

Cost of wireless transmission

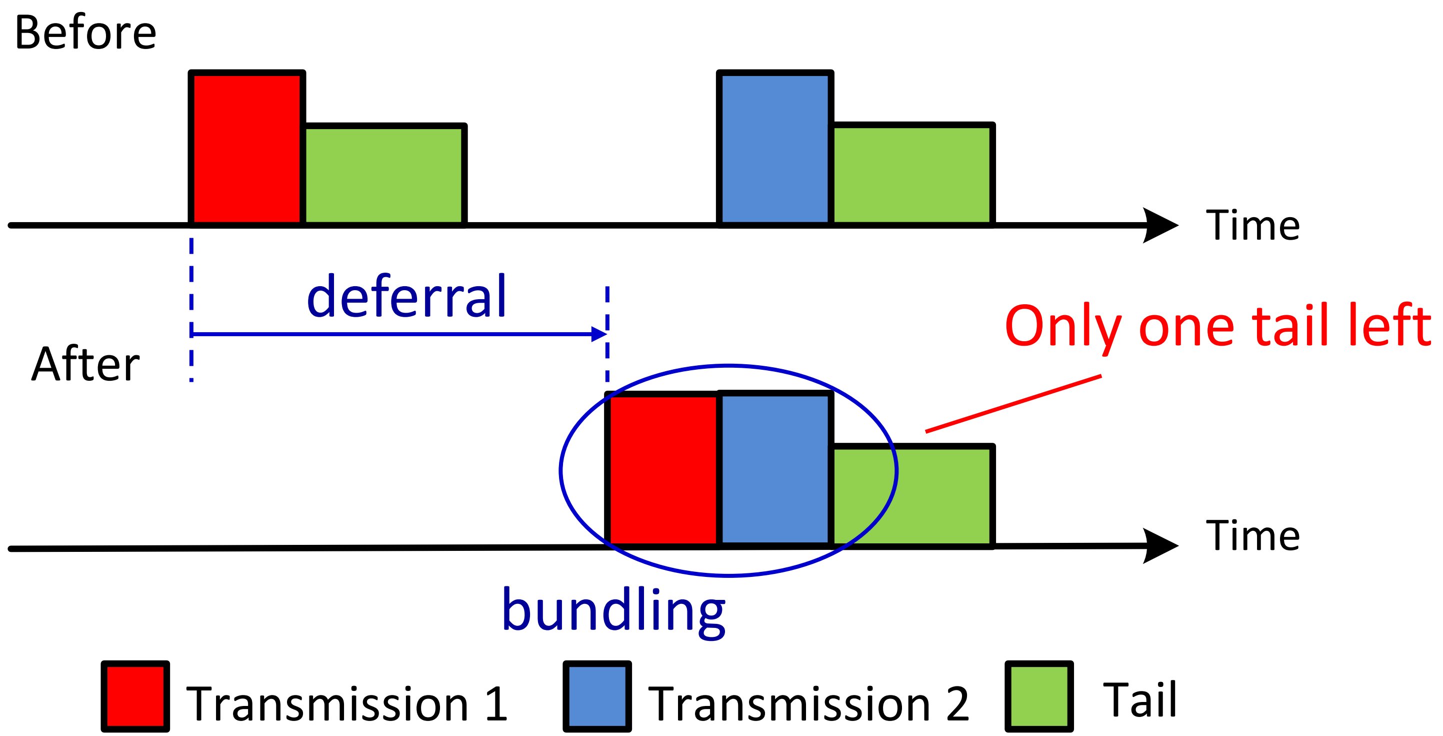
- Energy consumption during wireless transmission
  - Energy model of the UMTS cellular radio interface



A large portion of wireless energy consumption happens during tail times!

Existing solution

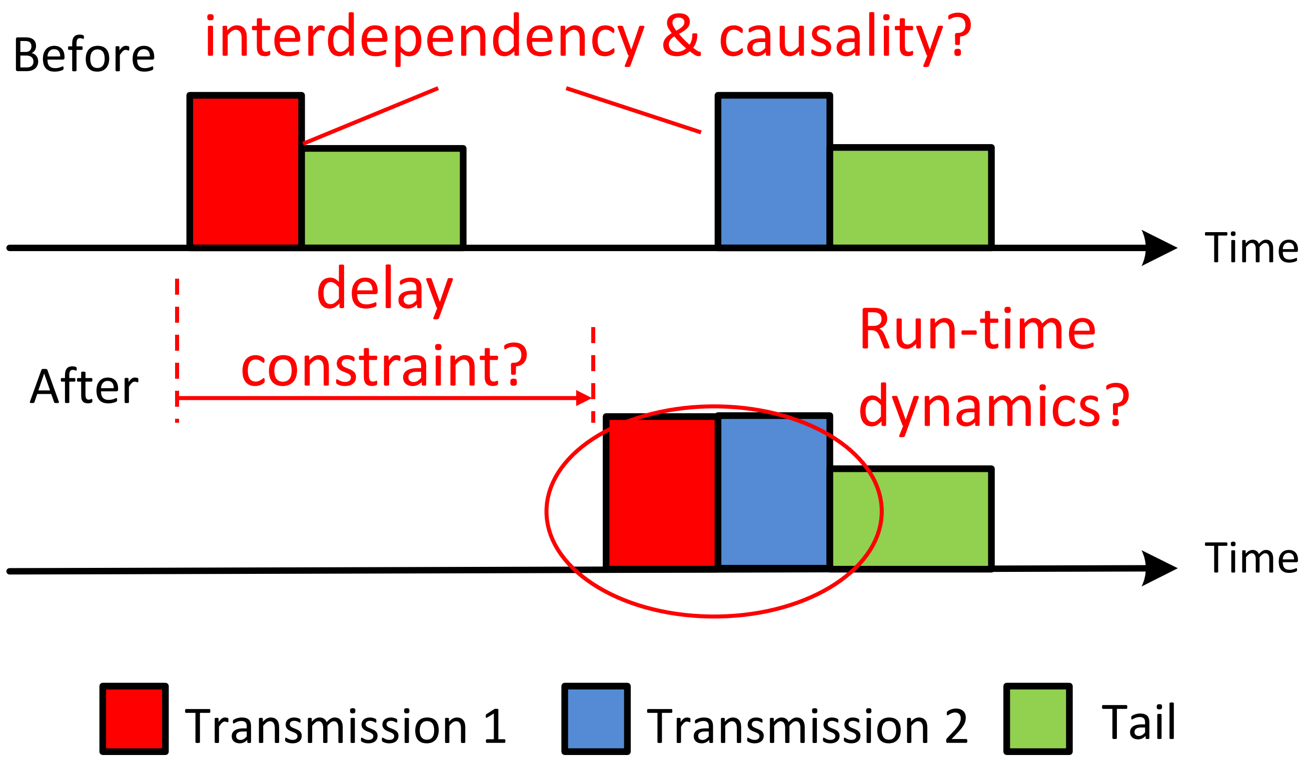
- Deferral and bundling



  - The tail time phenomenon can be alleviated
  - Good enough? No!

Why?

- Ignorance of mobile app characteristics



  - Application performance could be seriously degraded!

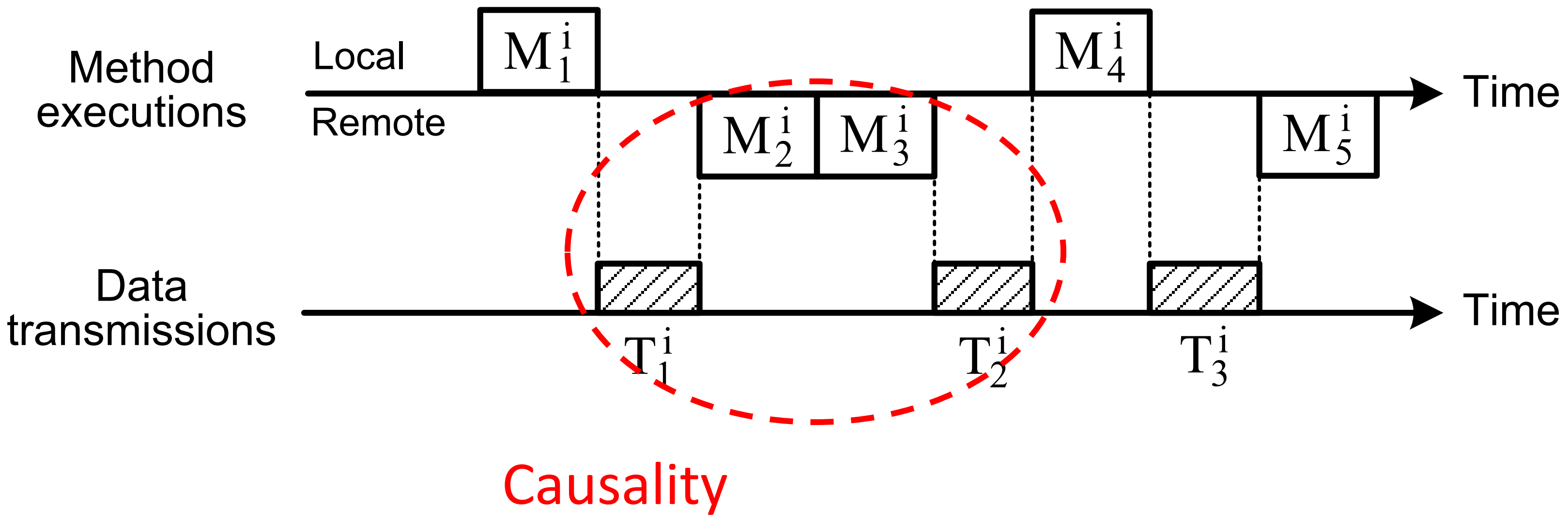
Our solution

- Key idea:
  - Adaptively balancing the energy/delay tradeoff
  - Taking both causality and run-time dynamics of application method executions into account
- Solutions

  - Causality → Offline scheduling
    - Optimal scheduling
    - Heuristic scheduling
  - Run-time dynamics → Online scheduling

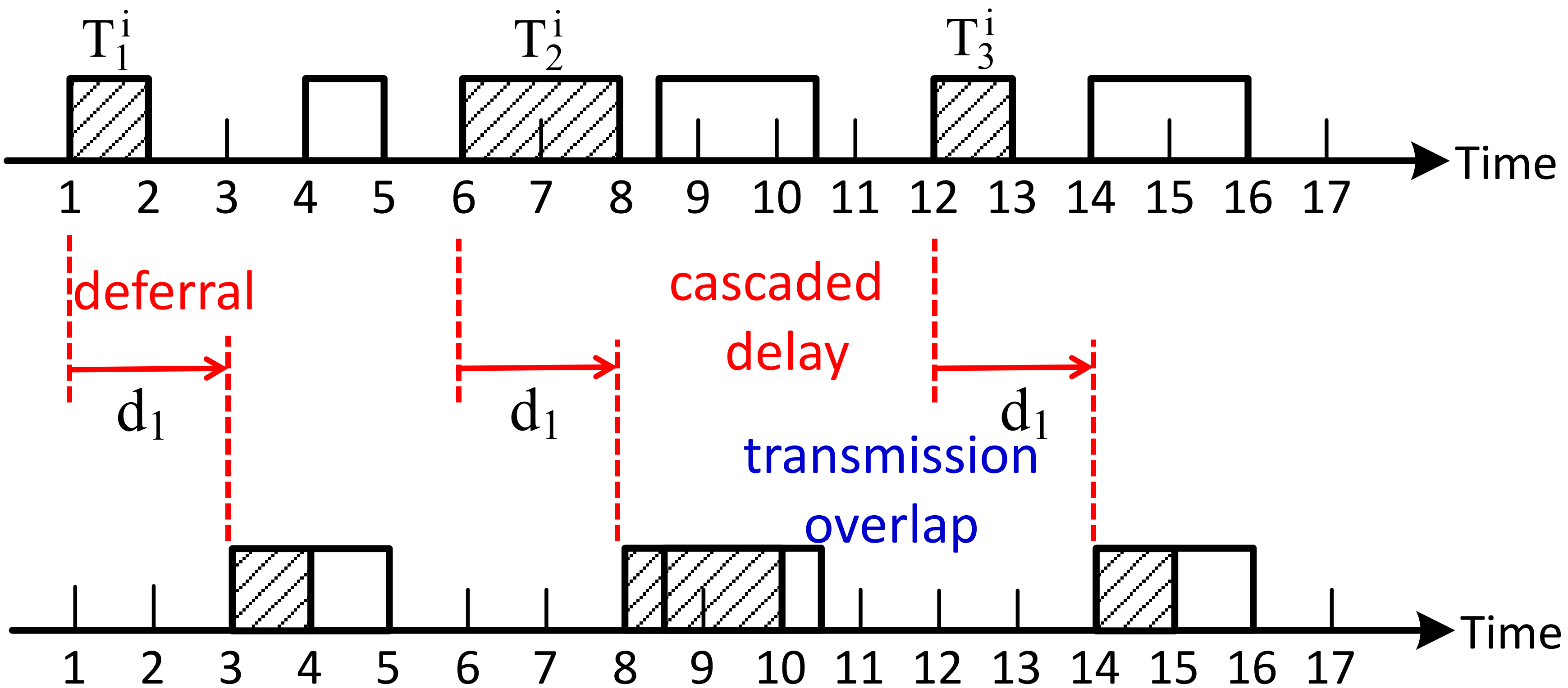
System model

- Multiple applications are running concurrently. For application *i*:
  - Delay constraint: D_i
  - Execution path: $\{M_1^i, M_2^i, \dots, M_{n_i}^i\}$.
  - Offloading decisions: existing work



Challenge

- How to eliminate transmission overlaps?



  - Additional delay to eliminate overlaps. But **how long?**

Offline transmission scheduling

- Problem formulation

$$\max \sum_{j=1}^{n_i} R_j(d_j) \qquad \text{Total number of bundling}$$

$$\text{s.t.} \quad d_j \le d_{j+1}, \qquad \text{transmission causality}$$

$$I_j(d_j) = 0, \qquad \text{overlap elimination}$$

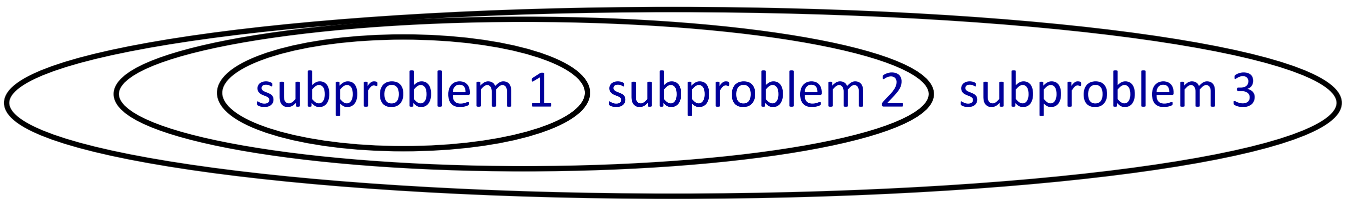
$$0 \le d_k \le D_i, (k \le n_i), \quad \text{delay constraint}$$

$$d_j \in \mathrm{N}$$

- Solution space: $D_i^{n_i}$. Exponential time for exhaustive search!
  - How to find optimal solution with a low complexity?

Optimal transmission scheduling (OTS)

- Basic idea
  - Solve problems by combining the solutions to subproblems

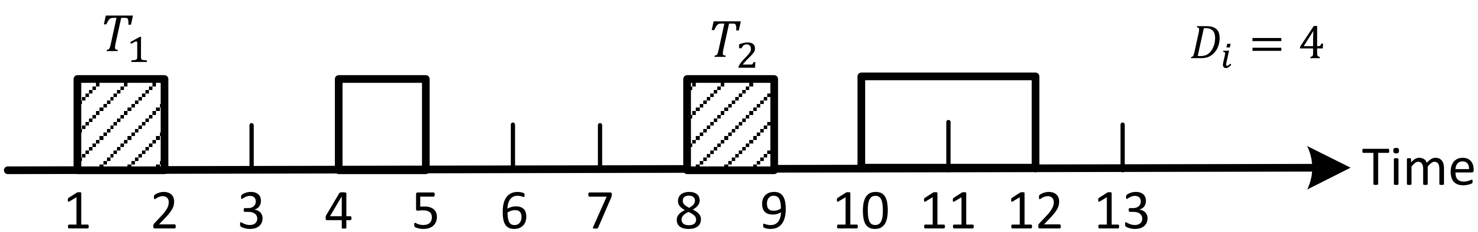


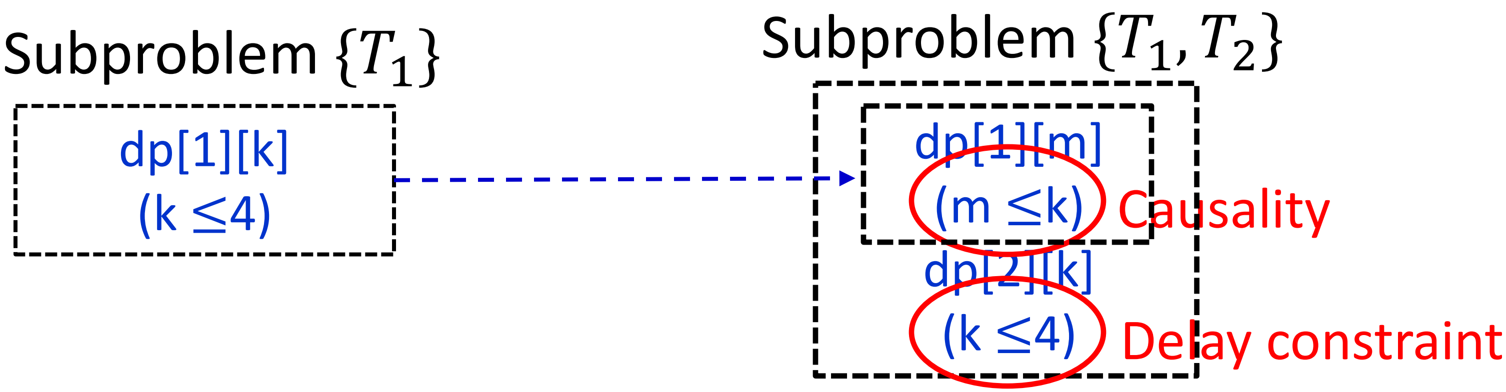
    - Dynamic Programming
    - BUNDLE(j, k): subproblem with j transmissions and delay constraint k
- Guarantee of optimal solution
  - BUNDLE(n_i, D_i) has an optimal substructure
  - What are the optimal solutions for subproblems?

THE UNIVERSITY of TENNESSEE KNOXVILLE

Optimal transmission scheduling (OTS)

- Optimal solutions of subproblems
  - dp$[j][k](k \leq D)$: the maximum number of bundles for a subproblem of $\{T_1, T_2 \ldots, T_j\}$ when T_j is delayed for k





Time complexity: $O(n_i D_i^2)$

Improvement of Computational Efficiency

- Problem: large computational overhead of OTS
  - Time complexity of OTS: $O\left(n_i {D_i}^2\right)$
  - D_i could be very large
- Solution: 2-stage transmission scheduling
  - Eliminating transmission overlaps heuristically

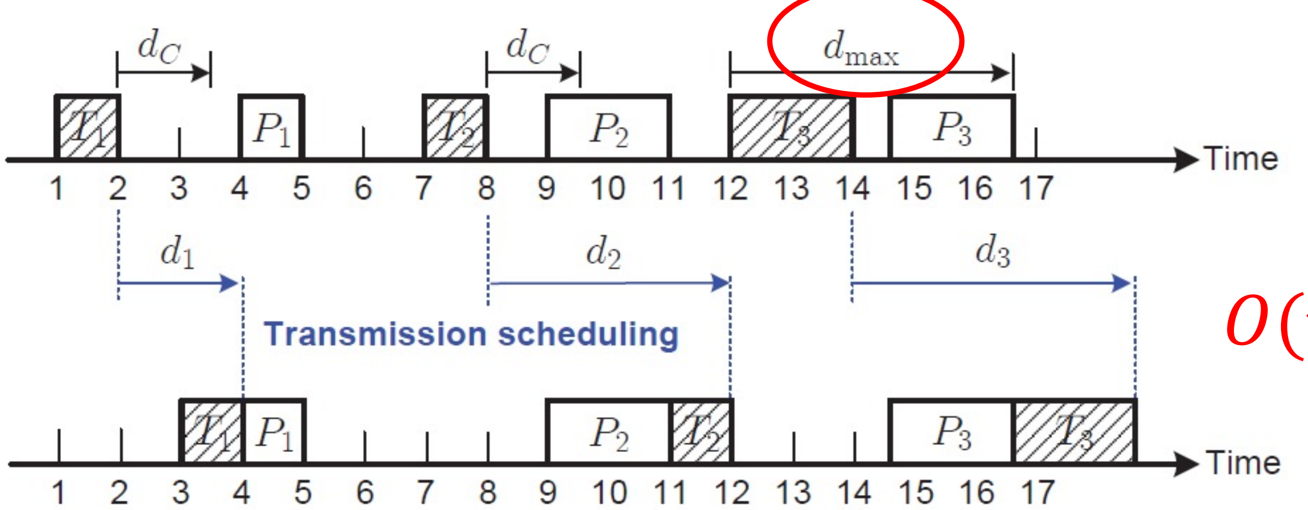
Stage 1: Posterior overlap elimination

↓

Stage 2: Prior overlap avoidance

Improvement of Computational Efficiency

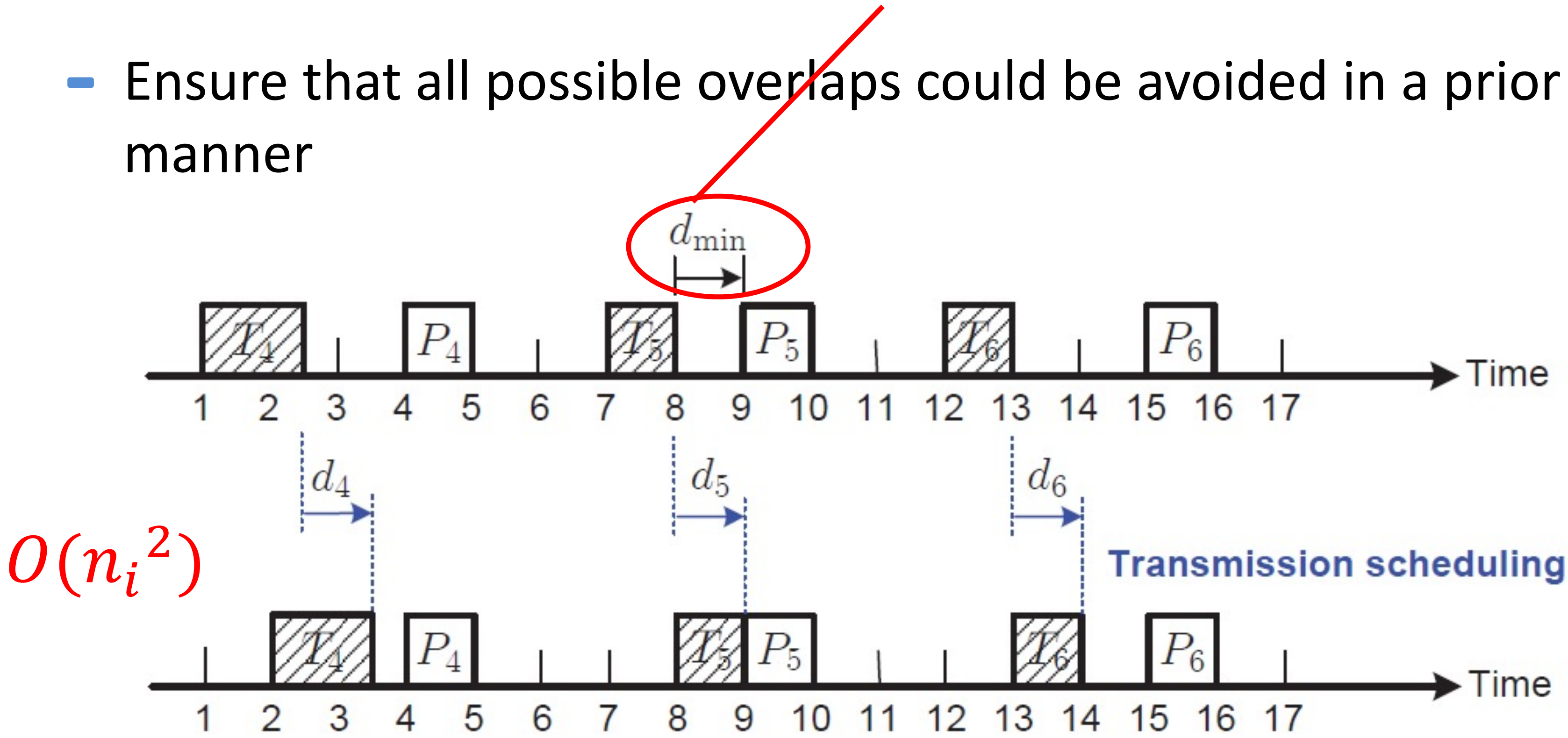
- Posterior overlap elimination
  - Iteratively looking for the maximally allowed transmission delay within the application delay constraint
  - Eliminating the transmission overlap due to such delay in a posterior manner



$O(n_i^2)$

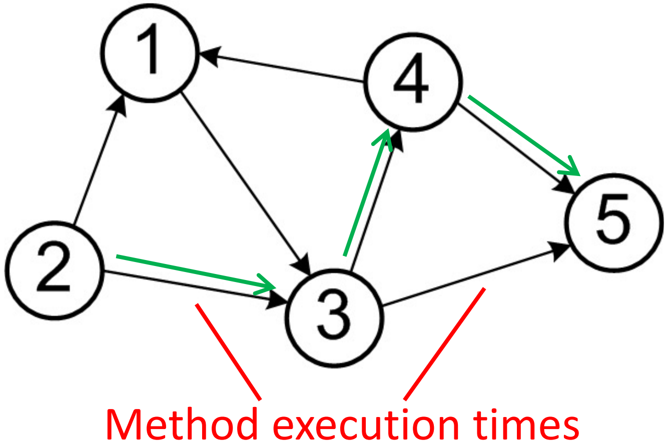
Improvement of Computational Efficiency

- Prior overlap avoidance
  - Iteratively looking for the minimum transmission delay
  - Ensure that all possible overlaps could be avoided in a prior manner



Online transmission scheduling

- Prediction of application execution path
  - Formulating method transitions as an order-k semi-Markov model
    - *Semi-markov*: arbitrary sojourn times between method transitions
    - *Order-k*: precise prediction of method Invocation (k-step interdependency)



  - Incorporation of run-time dynamics
    - Predicting the number of future method invocations
    - Predicting the execution time of method to be invoked in the future

Online transmission scheduling

- Probabilistic transmission scheduling
  - A probabilistic framework to adaptively schedule each transmission
  - Probabilistically estimate the cumulative transmission delay

Predicted number of future execution

Predicted execution time

$$\mathbb{P}\left(l_j \cdot d_j + \sum_{u=j+1}^{j+l_j} t_e^u \le D_i\right) \ge p$$

Cumulative deferral

Cumulative Execution time

Performance evaluations

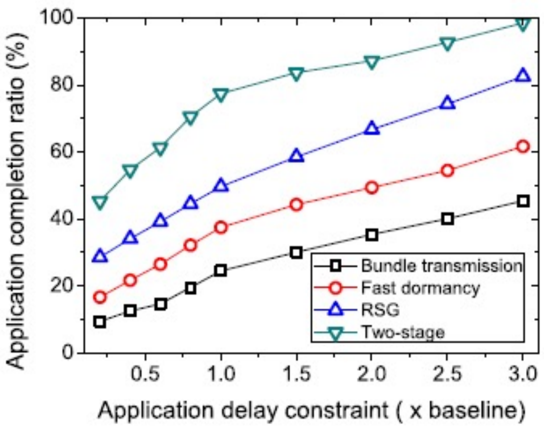
- Comparisons
  - *Bundle transmission*: performance requirements and delay constraint are not considered.
  - *Fast dormancy*: the mobile device switches to IDLE quickly after data transmission
  - *RSG*: run-time causality and dynamics are ignored.
- Evaluation metrics
  - Application completion ratio
  - Amount of energy saved
  - Computational overhead
    - The percentage of the energy consumption of application executions

Evaluation setup

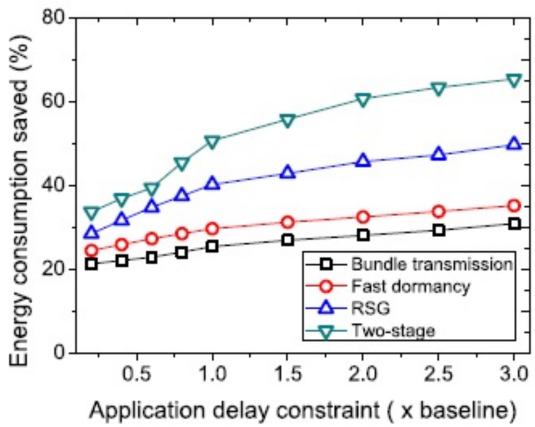
- Evaluation against open-source Android apps
  - *Firefox*, *Chess-Walk*, *Barcode Scanner*.
  - Implementing the transmission scheduling approach into app codes
- Offloading operations
  - Adopt MAUI for workload offloading decisions
  - Adopt *CloneCloud* to maintain a clone VM at the cloud server for each app
- Experiments
  - 100 times with different input data for statistical convergence

Effectiveness of offline scheduling

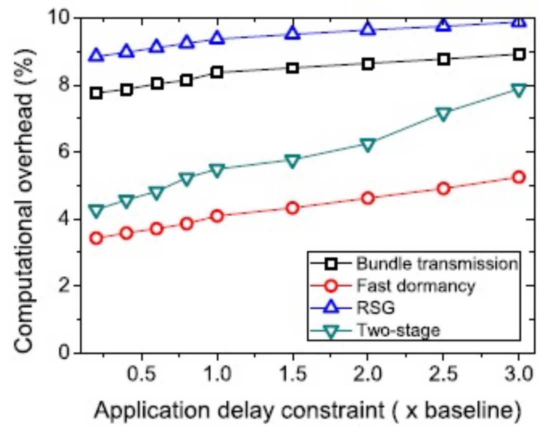
- The optimal scheduling (baseline < 2.0) and 2-stage scheduling (baseline ≥ 2.0) are used
  - Baseline: completion time of local application executions



(a) Application completion ratio



(b) Amount of energy saved



(c) Computational overhead

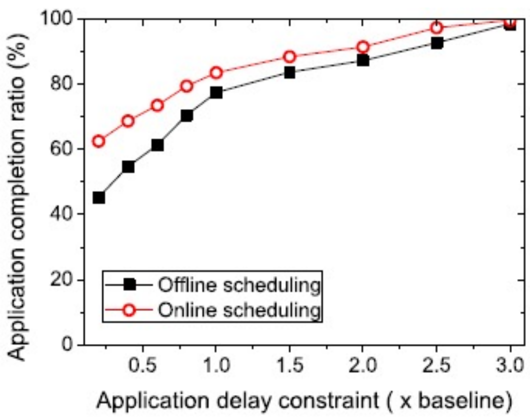
25% more completion ratio

40% more energy saved

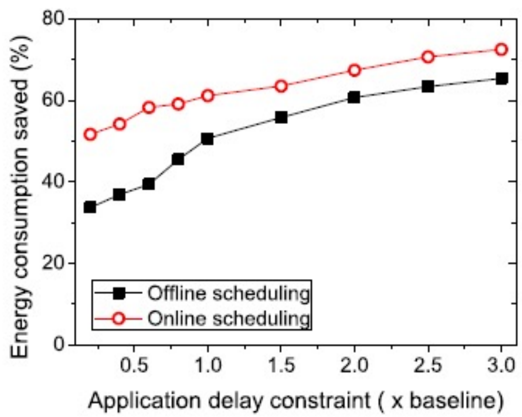
Less than **4%** overhead

Effectiveness of online scheduling

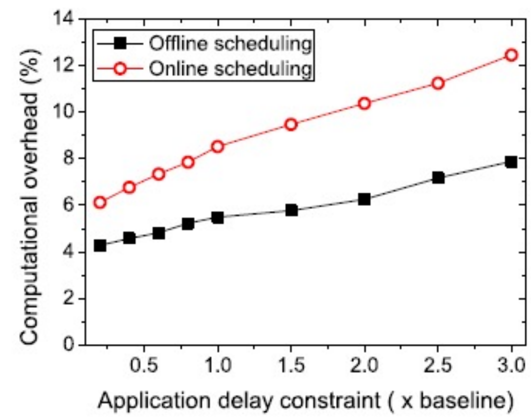
- An order-3 semi-Markov model is used
  - Online transmission algorithm performs better
  - Higher overhead is incurred



(a) Application completion ratio



(b) Amount of energy saved



(c) Computational overhead

20% more completion ratio

25% more energy saved

more than **50%** overhead

Summary

- Mobile Cloud Computing is critical to
  - Augment mobile devices' local capabilities
  - Energy saving
- MCC offloading energy efficiency is determined by wireless transmission scheduling
- *Insight:* exploiting run-time causality and dynamics is the key
  - Offline algorithm with causality being considered
  - Online algorithm incorporating with run-time dynamics

Thank you!

- Questions?

- The paper and slides are also available at:
  http://web.eecs.utk.edu/~weigao/