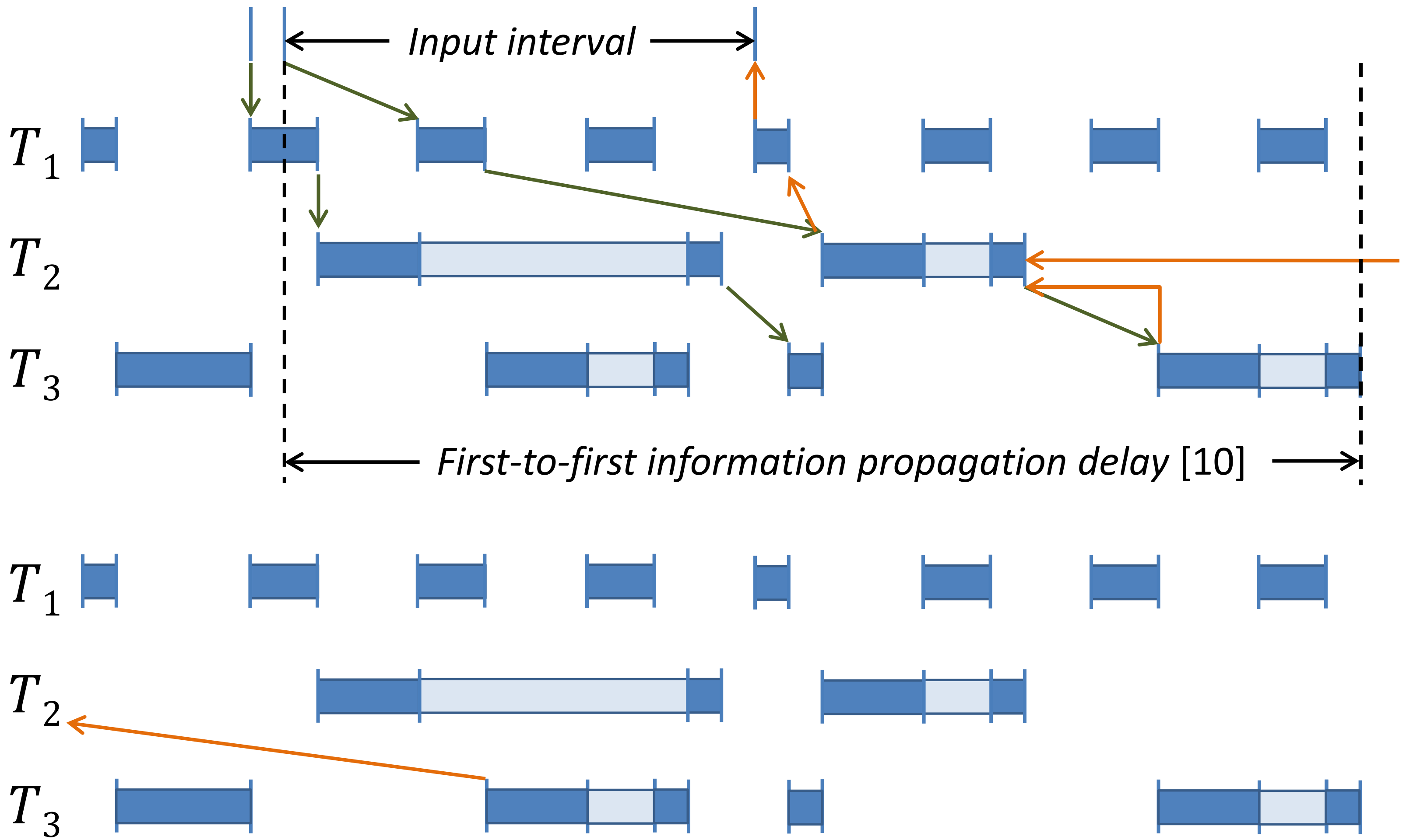
Computing First-to-First Propagation Delays Through Sequences of Fixed-Priority Periodic Tasks

Rodney R. Howell
Kansas State University

Motivation

- In many real-time control systems, tasks use information computed by other tasks
- The responsiveness of the system may depend on the propagation delay of information flowing through a sequence of tasks
- We would like to compute the worst-case propagation delay through a given sequence of tasks

Example



[10] N. Feiertag, et al., RTSS 2008

Task System Model

- Tasks are periodic
- Periods are harmonic
- Task arrivals are synchronous: all tasks initially arrive at time 0
- Each task has a minimum and a maximum execution time
  - Both are integers no greater than the period
- Each task has a distinct fixed priority

Scheduling Model

- All tasks are scheduled on the same processor
- Each task instance requires an integer-valued execution time no less that its minimum execution time and no greater than its maximum execution time
- At each integer time value, the ready task (if any exist) with highest priority is executed

Feasibility

- A schedule is feasible if each task instance completes no later than the next arrival of that task
- A task set is feasible if each possible schedule is feasible
  - Different schedules are produced by different execution time requirements of task instances
- We will only consider feasible task sets

First-Read Information Flow

- Let $\mathscr{T} = \langle T_1, \ldots, T_n \rangle$ be a sequence of distinct tasks, and let S be a schedule for a task set containing these tasks
- Given a time t_0, the *first-read information flow* from t_0 is the sequence $t_0 < \cdots < t_n$ such that for $1 \leq i \leq n$, t_i is the finish time of the first instance of T_i that begins executing no earlier than t_{i-1}

Last-Write Information Flow

- Let $\mathscr{T} = \langle T_1, \ldots, T_n \rangle$ be a sequence of distinct tasks, and let S be a schedule for a task set containing these tasks
- Given a time t_{n+1}, the *last-write information flow* to t_{n+1} is the sequence $t_1 < \cdots < t_{n+1}$ such that for $1 \leq i \leq n$, t_i is the start time of the last instance of T_i that finishes executing no later than t_{i+1}
- For small t_{n+1}, there may be no last-write information flow

Propagation Delays

- The *first-read propagation delay* $d^{FR}_{\mathscr{T}}(S, t_0)$ from time t_0 in schedule S through task sequence $\mathscr{T}$ is the *length* $t_n - t_0$ of the first-read information flow from t_0
- The *last-write propagation delay* $d^{LW}_{\mathscr{T}}(S, t_{n+1})$ to time t_{n+1} in schedule S through task sequence $\mathscr{T}$ is the *length* $t_{n+1} - t_1$ of the last-write information flow to t_{n+1}
  - $d^{LW}_{\mathscr{T}}(S, t_{n+1})$ is undefined for small t_{n+1}

Worst Case Propagation Delay

- The *worst-case first-to-first propagation delay* $D_{\mathcal{T}}^{FF}$ is the maximum value of $d_{\mathcal{T}}^{FR}(S, t_0) + 1$, taken over all schedules S and all times t_0

A Useful Relationship

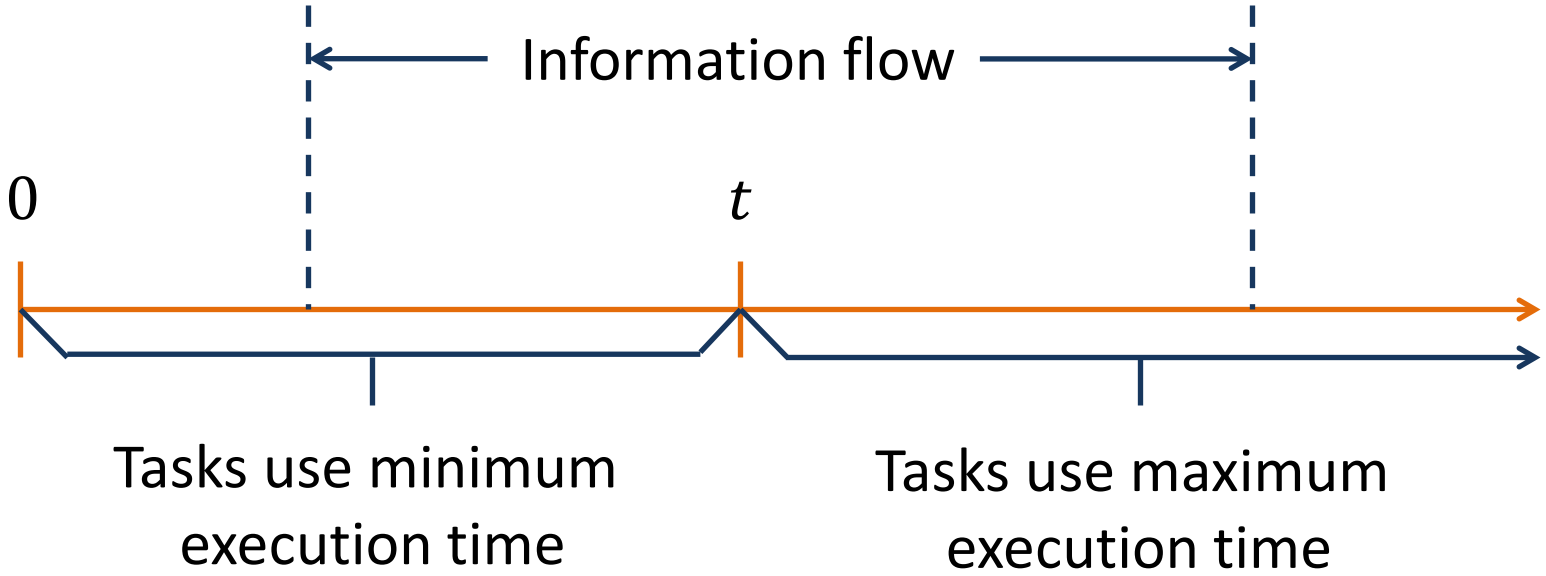
Theorem: For any schedule S and time t:

- $d_{\mathscr{T}}^{FR}(S,t) \ \leq d_{\mathscr{T}}^{FR}(S,t+1) \ = \ d$ iff
- $d_{\mathscr{T}}^{LW}(S,t+d+1) \ \leq d_{\mathscr{T}}^{LW}(S,t \ + \ d) \ = \ d$

Thus:

- a maximum-length first-read information flow occurs in S from time $t+1$ to time t $+ \, d + 1$ iff
- a maximum-length last-write information flow occurs in S from time t to time $t + d$

Pivoting Schedules



Monotonically Decreasing Priorities

- Let $\mathscr{T} = \langle T_1, \ldots, T_n \rangle$ be a task sequence with monotonically decreasing priorities within a feasible task set
- Suppose P is the largest period of any task in the task set
- Let S be any schedule that pivots at time P
- Then this schedule contains a first-read information flow $\langle t_0, \ldots, t_n \rangle$ with maximum length
- Furthermore, t_n can be chosen to be $P + R(T_n)$, where $R(T_n)$ is the *worst-case response time* of T_n

Algorithm

- Build a schedule from $P - p_k$ to $P + R(T_n)$
  - p_k is the largest period in the task sequence
  - This schedule should pivot at P
- Find the last-write information flow to $P + R(T_n) - 1$, and add 1 to its length
- **Running Time:** $O(m + p_k \log m)$, where m is the number of tasks in the task set

Monotonically Increasing Priorities

Let $\mathscr{T} = \langle T_1, \ldots, T_n \rangle$ be a task sequence with monotonically increasing priorities within a feasible task set

Suppose P is the largest period in the task set

Then there is a schedule S with a first-read information flow $\langle t_0, \ldots, t_n \rangle$ such that

- $0 < t_0 \leq P$
- S pivots at t_0
- $t_n - t_0 = D_{\mathscr{T}}^{FF} - 1$

Algorithm

For $0 \leq i < P/p_1$:

- Build a schedule of length $3P$
  - Pivots at t_0, 1 time unit after the instance of T_1 that arrives at time ip_1 begins executing
- Compute an array *Next*$[1..2P]$ such that if T_i finishes at time t, *Next*$[t]$ gives the next finish time of an instance of T_{i+1}
- Compute the first-read information flow from t_0, and add 1 to its length

Running Time

- $O(P^2 \log m \,/\, p_1)$
- If rate monotonic priorities are used, this can be improved to $O(m + p_1 \log m)$

Arbitrary Sequences

- In general, there may be no maximum-length first-read information flow in any schedule that pivots
- However, a task sequence may be partitioned into monotonically increasing and monotonically decreasing sequences
- Summing $D^{FF}_{\mathscr{T}_k} - 1$ over all monotonic subsequences $\mathscr{T}_k$ gives an upper bound on $D^{FF}_{\mathscr{T}} - 1$

Example

- Suppose the sequence of priorities is $\langle 1, 3, 2 \rangle$
- We can partition this sequence into either
  - $\langle 1, 3 \rangle$ and $\langle 2 \rangle$ or
  - $\langle 1 \rangle$ and $\langle 3, 2 \rangle$
- Our goal is to choose the partitioning giving the best upper bound

Performance

- **Running Time:** $O(P^2 \log m)$
- For rate-monotonic priorities, this can be improved to $O((m + s) \log m + n^2)$
  - s is the sum of the periods of the tasks in the sequence
  - n is the number of tasks in the sequence
- When viewed as an approximation algorithm for minimizing the upper bound on $D_{\mathscr{T}}^{FF}$, this algorithm has an approximation ratio of $\lceil n/2 \rceil$

Future Work

- Is there an efficient algorithm for non-monotonic sequences?
- Can these results be extended to other types of delays?
  - The algorithm for monotonically increasing priorities extends to last-to-last delays
- What if the periods are not harmonic or start times are offset?
- Can priorities be adjusted to shorten the propagation delay?

Questions?