



Real-Time Query Scheduling for Wireless Sensor Networks

Chenyang Lu
Department of Computer Science and Engineering

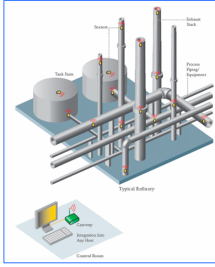
 Washington University in St. Louis

Application challenges


- High data rate
- Low latency
- Prioritization
- Predictability



Structural Health Monitoring




Process Monitoring


 Washington University in St. Louis 2

Outline

- Conflict-free transmission scheduling
 - ❑ Optimized for queries in sensor networks.
- Priority-based real-time scheduling
 - ❑ Trade-off between prioritization and throughput.
- Worst-case delay analysis
 - ❑ Bridging the gap between sensor net and real-time scheduling theory.
- Other projects


 Washington University in St. Louis 3

Query model



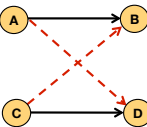
```
SELECT acceleration FROM accelerators
SAMPLE RATE 10Hz
DEADLINE 0.1s
```


- **Query** – periodic data collection from a set of sensors
- **Query instance** – instance of query in a sampling period

 Washington University in St. Louis 4

Network model


- Interference-Communication Graph
 - Communication edge (AB): A's transmission may be received by B
 - Interference edges (AD): D cannot receive when A transmits, even though D cannot decode A's transmission
- Transmissions AB and CD are conflict free if:
 - AD and CB are not present in the graph
- Can be constructed using the RID protocol [Zhou 2005]



 Washington University in St. Louis 5

Overview: query scheduling

- Planner: Offline or when a query arrives
 - Construct a schedule for a **single** query instance
 - Reduce query latency based on transmission dependency
- Scheduler: Run time
 - Dynamically schedule **multiple** concurrent query instances
 - Improve throughput
 - Maintain conflict free
 - Priority-based real-time schedulers

 Washington University in St. Louis 6

Planner

- Plan: a sequence of steps
 - Schedule for a single query instance
 - Independent of other query instances
 - A step includes a set of conflict-free transmissions
 - Respect inter-transmission dependencies incurred by routing or aggregation.
- Planning algorithm
 - Construct a reversed plan
 - Assign priorities to nodes based on (depth, numbers of children, IDs).
 - Assign a transmission to the node with the highest priority to the current step, if no conflict with previous transmissions assigned to the same step.
 - Reverse the plan

Washington University in St. Louis 7

Example of a plan

	senders					
steps	0	q				
	1	n	p			
	2	f	k	o	z	s
	3	e	l	h	t	r
	4	c	j	g	w	
	5	b	m			
6	d					

Washington University in St. Louis 8

Concurrent instances → conflicts

Washington University in St. Louis 9

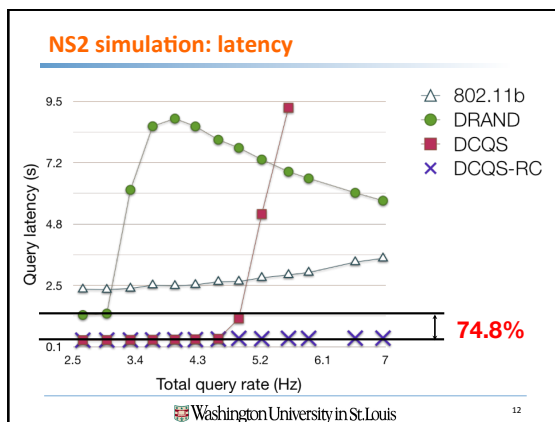
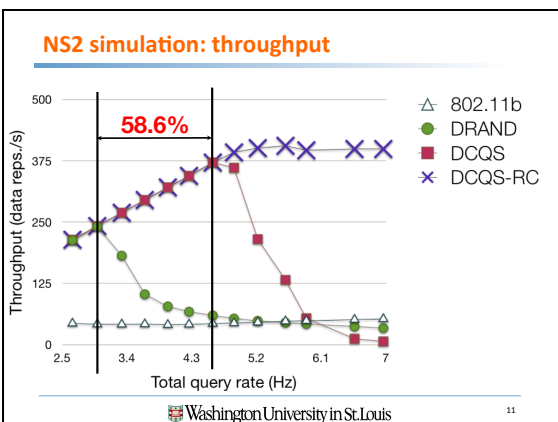
Minimum step distance

- Minimum step distance is the smallest Δ such that: if the distance between **any** two steps $\geq \Delta \rightarrow$ conflict-free
- Scheduler: Enforce a gap of Δ between instances \Rightarrow conflict-free

Min. step distance: $\Delta = 3$

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Washington University in St. Louis 10

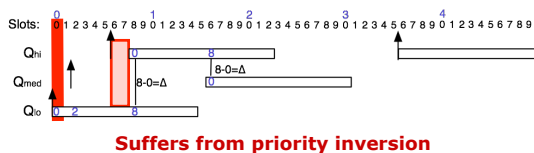


Real-time query scheduling

- Trade-off between prioritization and throughput
- Non-preemptive Query Scheduler (NQS)
 - high throughput, priority inversion
- Preemptive Query Scheduler (PQS)
 - lower throughput, no priority inversion
- Slack-stealing Query Scheduler (SQS)
 - uses preemption only when necessary
 - improves throughput without missing deadlines

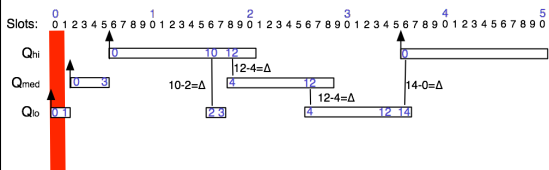
Nonpreemptive Query Scheduler (NQS)

- Order pending queries based on priority
 - Enforce Δ between the **start times** of consecutive instances
 - High-priority instance does not preempt low-priority instances that have already started
- Plan length = 15, $\Delta = 8$

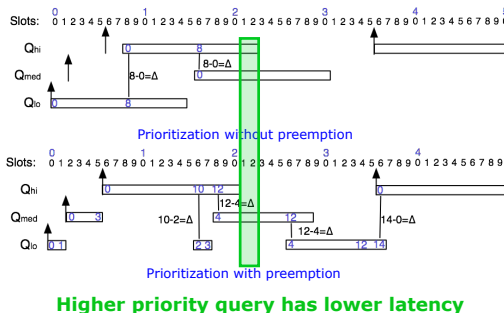


Preemptive Query Scheduler (PQS)

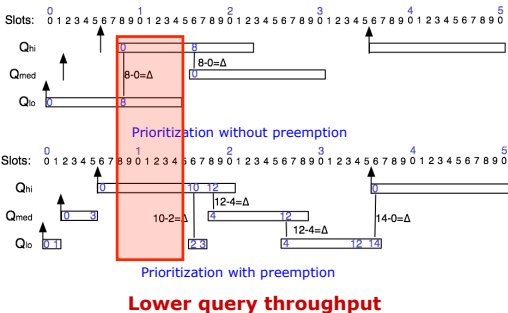
- High-priority instance preempts low-priority instance if they would conflict.



NQS/PQS Comparison

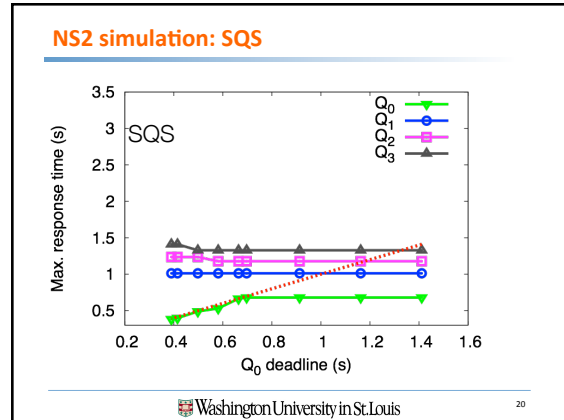
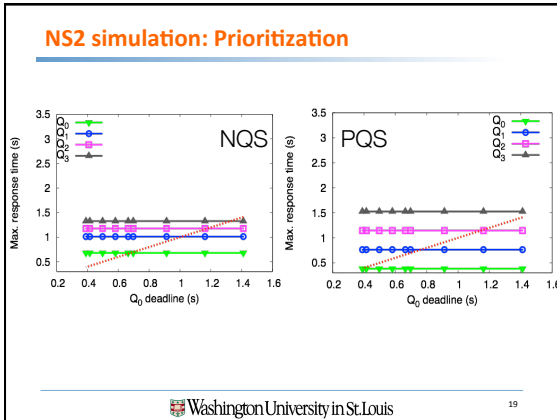


NQS/PQS Comparison



Slack stealing query scheduling (SQS)

- SQS combines benefits of NQS and PQS
 - Use preemption only when necessary to meet deadlines
 - Improve throughput while meeting all deadlines
- Slack - the maximum time a query instance may be delayed without missing its deadline
- SQS scheduling algorithm
 - If it has enough slack, a higher priority instance allows a lower priority instance to complete its first Δ steps
 - Otherwise, the higher priority instance preempts the low priority instance immediately



Worst-case delay analysis

- Assumption: period <= deadline
- Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- Response time:

$$R_l = W_l + L$$

Washington University in St. Louis 21

Worst-case delay analysis

- Assumption: period <= deadline
- Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- Response time:

$$R_l = W_l + L \rightarrow \text{execution time}$$

Washington University in St. Louis 22

Worst-case delay analysis

- Assumption: period <= deadline
- Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- Response time:

$$R_l = W_l + L$$

worst-case delay before instance L starts

Washington University in St. Louis 23

Worst-case delay analysis

- Assumption: period <= deadline
- Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- Response time:

$$R_l = W_l + L$$

$$W_l = (\Delta - 1) + \sum_{h \in hp(l)} \left\lceil \frac{W_l}{P_h} \right\rceil \Delta$$

Washington University in St. Louis 24

Worst-case delay analysis

- > Assumption: period <= deadline
- > Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- > Response time:

$$R_l = W_l + L$$

blocking time

$$W_l = (\Delta - 1) + \sum_{h \in hp(l)} \left\lceil \frac{W_l}{P_h} \right\rceil \Delta$$

Washington University in St. Louis 25

Worst-case delay analysis

- > Assumption: period <= deadline
- > Map to Response Time Analysis in real-time scheduling theory
 - Execution time: length of the plan L
 - Interference from a high-priority instance: Δ
 - Blocking time: $\Delta - 1$
- > Response time:

$$R_l = W_l + L$$

interference

$$W_l = (\Delta - 1) + \sum_{h \in hp(l)} \left\lceil \frac{W_l}{P_h} \right\rceil \Delta$$

Washington University in St. Louis 26

Conclusions

- > Conflict-free transmission scheduling
 - Optimized for queries in sensor networks.
 - Adaptive to workload changes.
- > Priority-based real-time schedulers
 - Trade-off between prioritization and throughput.
- > Worst-case delay analysis
 - Bridging the gap between sensor net and real-time scheduling theory.

O. Chipara, C. Lu, J.A. Stankovic, Dynamic Conflict-free Query Scheduling for Wireless Sensor Networks, ICNP'06.

O. Chipara, C. Lu, G.-C. Roman, Real-time Query Scheduling for Wireless Sensor Networks, RTSS'07.

Washington University in St. Louis 27