**Motivation**

- Many real-time service classes have been defined: HRT, SRT, BE, ...
- Exploiting particularities of a specific service class generates Slack time, e.g. dropping non-mandatory jobs of FRT tasks
- Slack improves the QoS received by other process classes (SRT, BE)
- How to drop tasks such that the benefit for other classes is maximized?

> SRT, BE ?

- But on a resource constrained device?

> Late frames might even damage the user experience

**Preliminary Implementation**

Drop tasks, real application

- Communication channel needed to coordinate frame drops

Drop tasks, simulation

- Simulator (consume CPU)
- Communication through system calls
- Use system call for communication between kernel/user space

**Architecture and Solution**

(HRT, SRT, BE) + FRT in the RBED Model

In a video-clip with 29.97fps:
- Instead of displaying all frames
- Ensure 25/30 frames are on time
- Drop up to 5/30 frames

Drop which tasks? Example: FRT task with (3,5) constraints

- Arbitrary job drop violates (m,k constraints)

- A static drop pattern - deeply red
- Converted to arbitrary drop 1 (of 4) pattern
- Dynamic drop pattern

Slack reclaiming in the extended RBED System

- Hard Real-Time
- Best-Effort
- Firm Real-Time

**Preliminary Results**

- FRT added to RBED scheduler
- Integrated with HRT, SRT, BE
- Can trade off FRT/SRT performance

**Conclusions & Future Work**

- FRT dropping tasks improves SRT/BE performance
- Shown for simple static drop patterns
- Expect further improvements using
t- evenly distributed drop pattern
t- dropping jobs dynamically based on SRT demand

- Use DQM approaches to investigate trade-offs:
  - Reducing QoS using FRT - is it worth it?
  - Would you sacrifice video quality for other tasks?
  - How to quantify QoS cost of dropped jobs for the application?