End-to-end Analysis and Design of a Drone Flight Controller

Zhuoqun Cheng, Richard West, Craig Einstein
Boston University

Emerging Drone Applications

Current State of the Art

Most drone apps controlled by humans

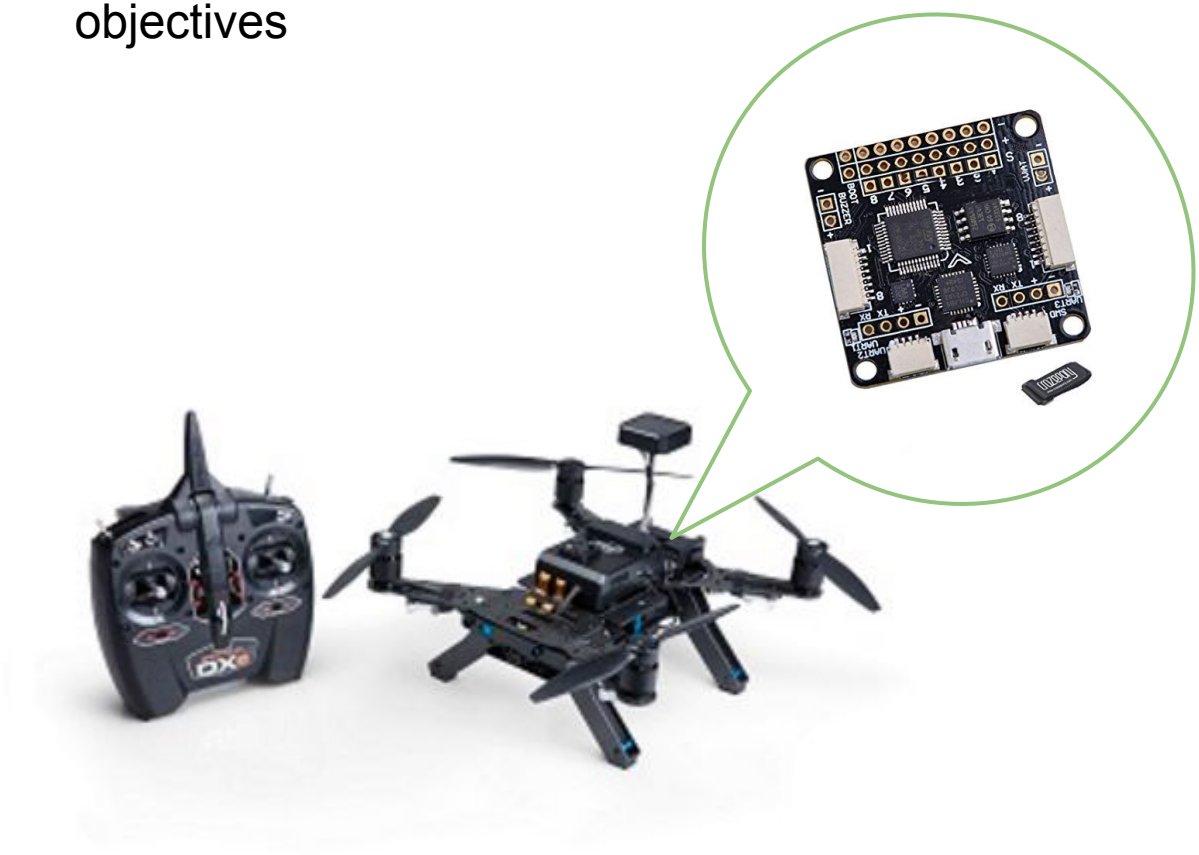
- Use SBCs based predominantly on STM32 ARM Cortex M3/M4 single-core platforms
- Firmwares include Cleanflight, Ardupilot, PX4, etc
- Lack support for complete autonomous control with adaptable mission objectives

Emerging trend towards autonomous drones

- e.g., give examples such as Skydio for object tracking
- Still not flexible enough to support reconfigurable missions
- Use separate flight control and mission processing boards
  - e.g., PX4 + Aero board, DJI example
  - Our AIM to combine flight control + mission objectives onto SBC with sufficient processing power, while meeting SWaP constraints

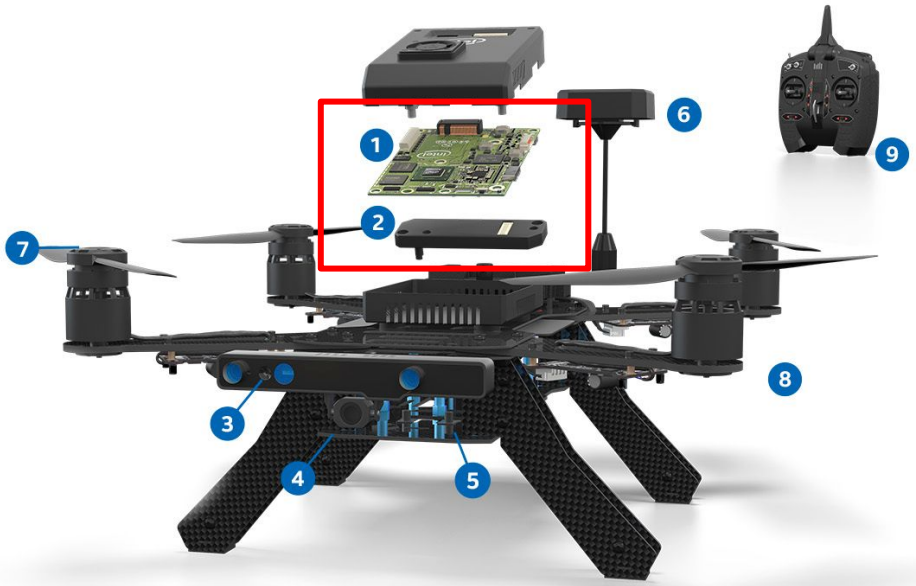
State of the Art

- Most drones are controlled by humans
  - STM32 ARM Cortex M3/M4 single-core SBCs
  - Popular firmwares include Cleanflight, Ardupilot, PX4
  - Lack support for autonomous control w/ adaptable mission objectives



State of the Art

- Emerging trend towards autonomous drones
  - Object-tracking drone, e.g., Skydio
  - Shortcomings:
    - Not flexible enough to support reconfigurable missions
    - Dual board architecture: Microcontroller w/ FC firmware + powerful SBC w/ GPOS
      - DJI Matrice 100: N1 flight controller + DJI Manifold
      - Intel Ready-to-Fly drone: Intel Aero board + PX4



1. Intel® Aero Compute Board
2. Intel® Aero Flight Controller, preprogrammed with Dronecode* PX4* autopilot
3. Intel® RealSense R200 Camera for 3D depth sensing
4. 8 MP RGB camera (front-facing)
5. VGA camera, global shutter, monochrome (down-facing) (not visible in photo)
6. GPS and Compass
7. Four ESCs, Motors, Propellers
8. Carbon Fiber Chassis (Fully Assembled)
9. Radio Control Transmitter and Reciever

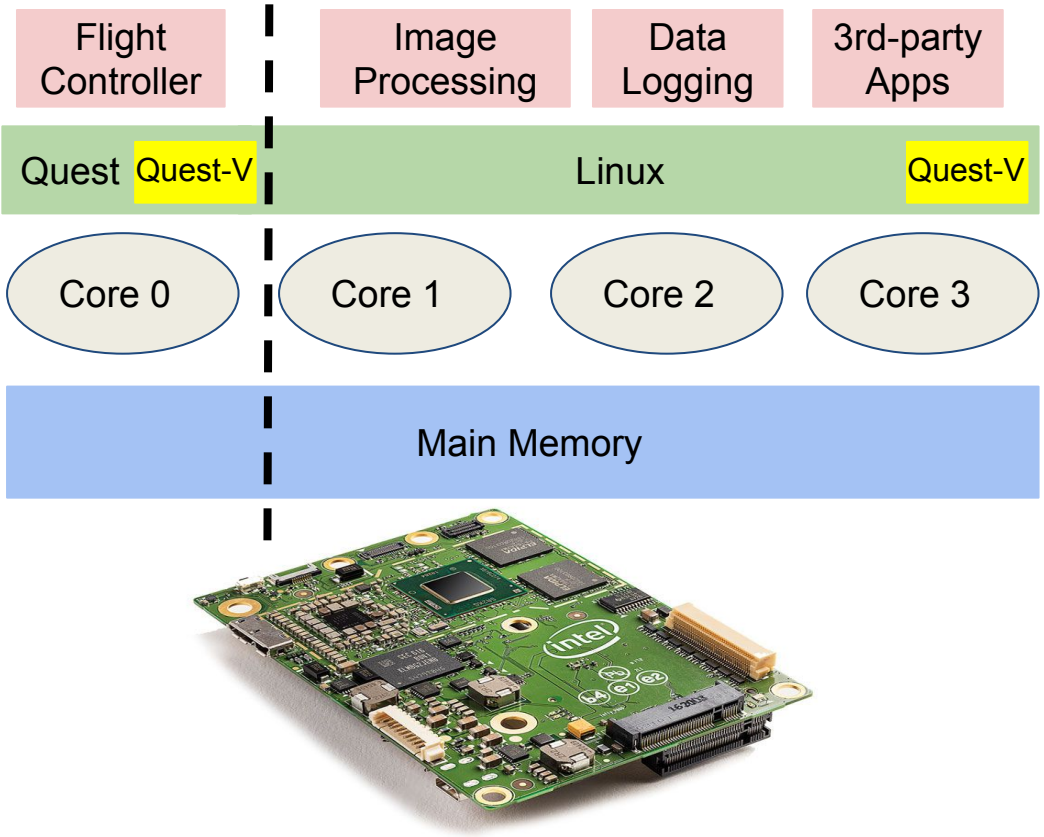
Autonomous Drone example

- Traditional approach
  - Dual board
    - Microcontroller w/ FC firmware + powerful SBC w/ GPOS
      - DJI Matrice 100: N1 flight controller + DJI Manifold
      - Intel Ready-to-Fly drone: Intel Aero board + PX4

1. Intel® Aero Compute Board
2. Intel® Aero Flight Controller, preprogrammed with Dronecode* PX4* autopilot
3. Intel® RealSense R200 Camera for 3D depth sensing
4. 8 MP RGB camera (front-facing)
5. VGA camera, global shutter, monochrome (down-facing) (not visible in photo)
6. GPS and Compass
7. Four ESCs, Motors, Propellers
8. Carbon Fiber Chassis (Fully Assembled)
9. Radio Control Transmitter and Reciever

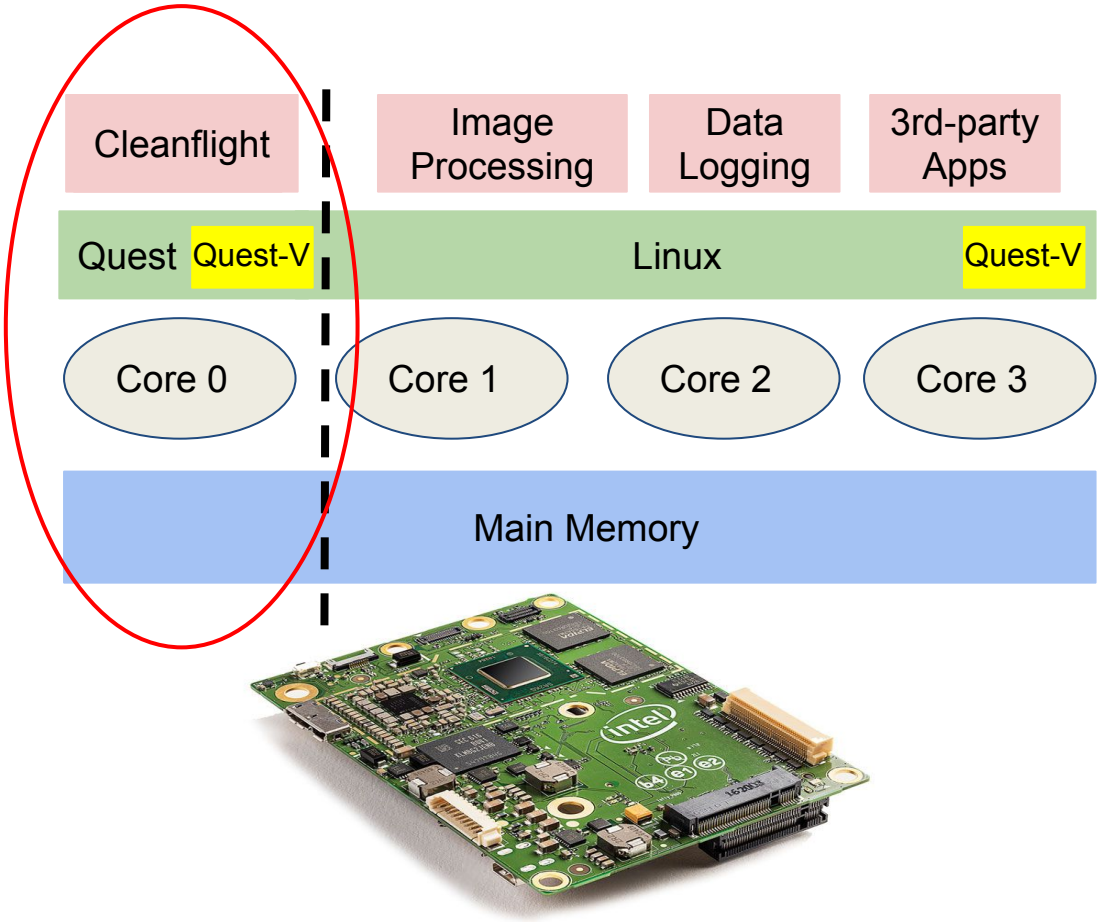
Our Objective

- Combine flight control & mission objectives onto one powerful SBC
  - Aim to meet SWaP (Size Weight and Power) constraints
  - Use Quest-V virtualized separation kernel
    - Virtualization-based CPU, memory and I/O partitioning
    - Quest RTOS and Linux in two sandboxes



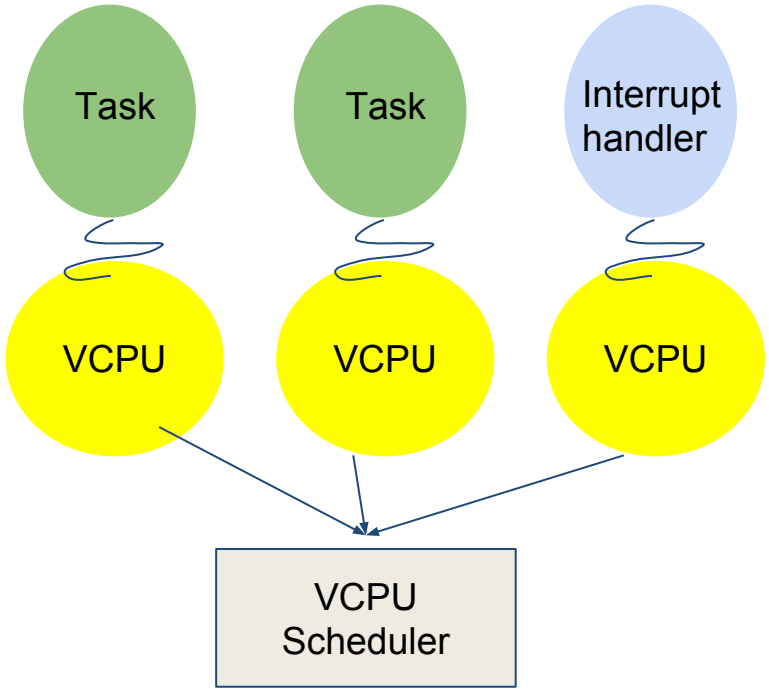
Scope of This Work

- Refactoring Cleanflight
  - Popular racing drone flight controller
  - Firmware on ARM Cortex M3/M4 STM32 SoC
  - Multithreaded application on Quest RTOS



Quest RTOS

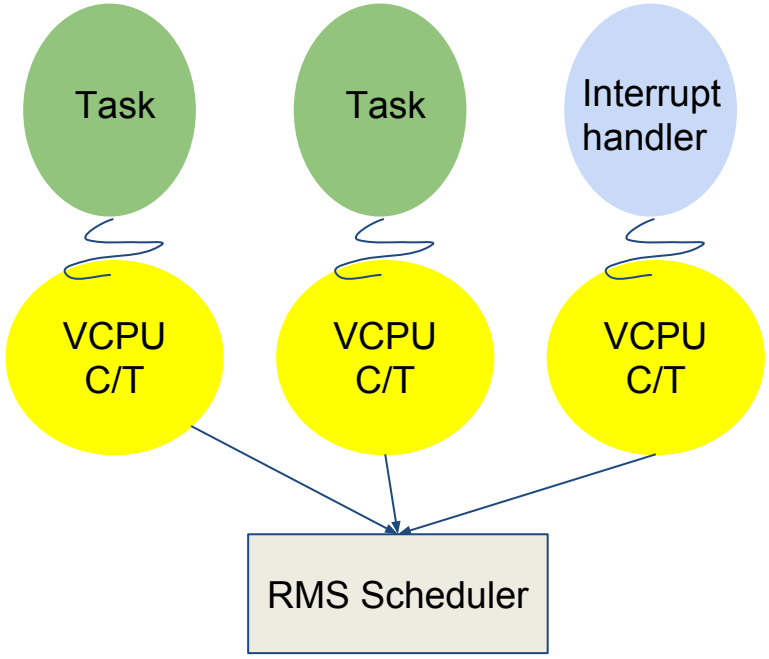
- Supports a series of x86-based SoC
  - Aero, UP2, Edison, MinnowMAX, etc.
- Supports user & kernel threads
  - periodic task ~ user thread; driver INT handler ~ kernel thread
- Each thread mapped to a VCPU



Further info:
www.questos.org

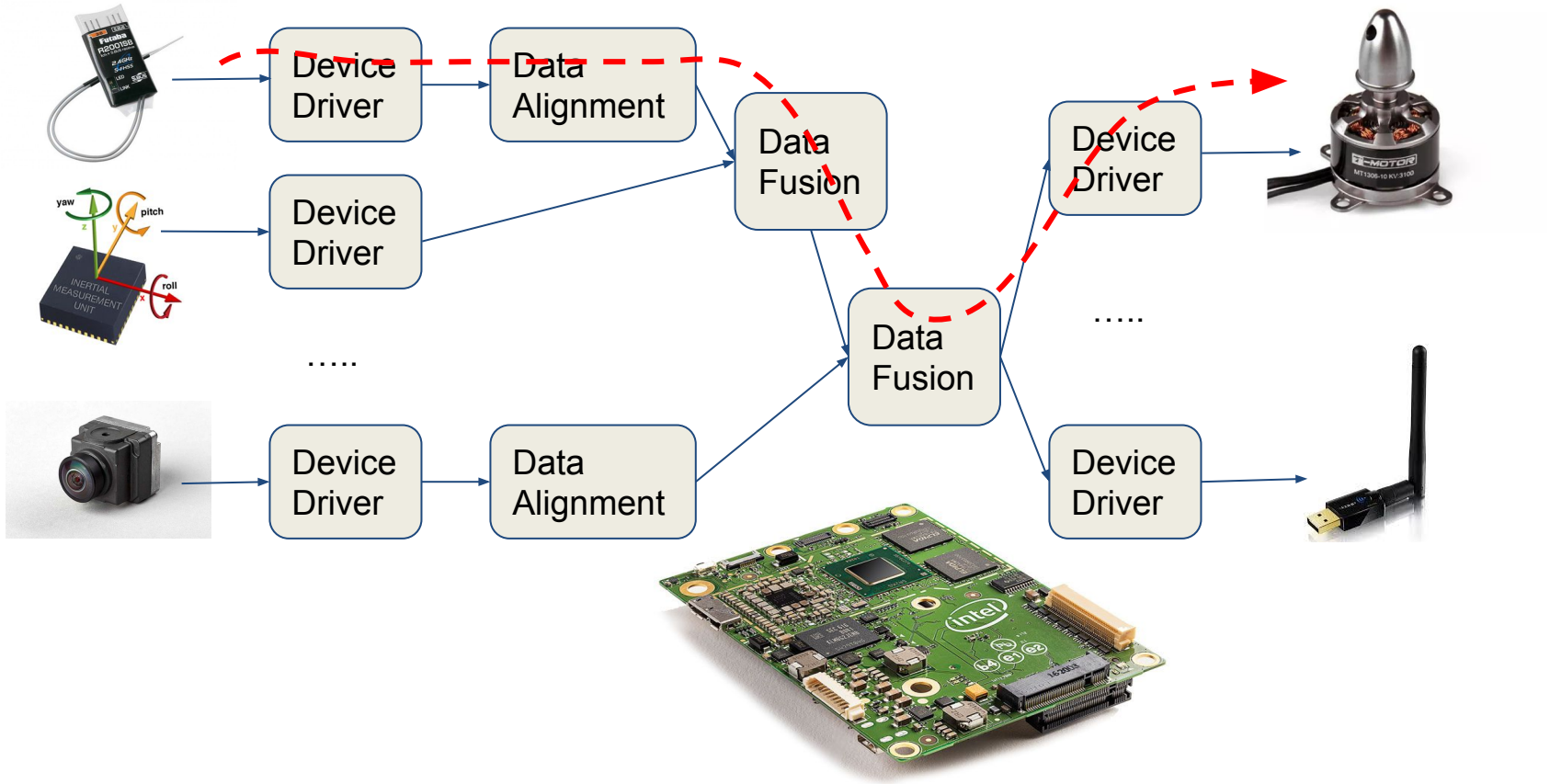
VCPU Scheduling

- Task associated with CPU resource container called VCPU: budget *C* and period *T*
- VCPUs scheduled by RMS
  - guarantees C within T if task is runnable



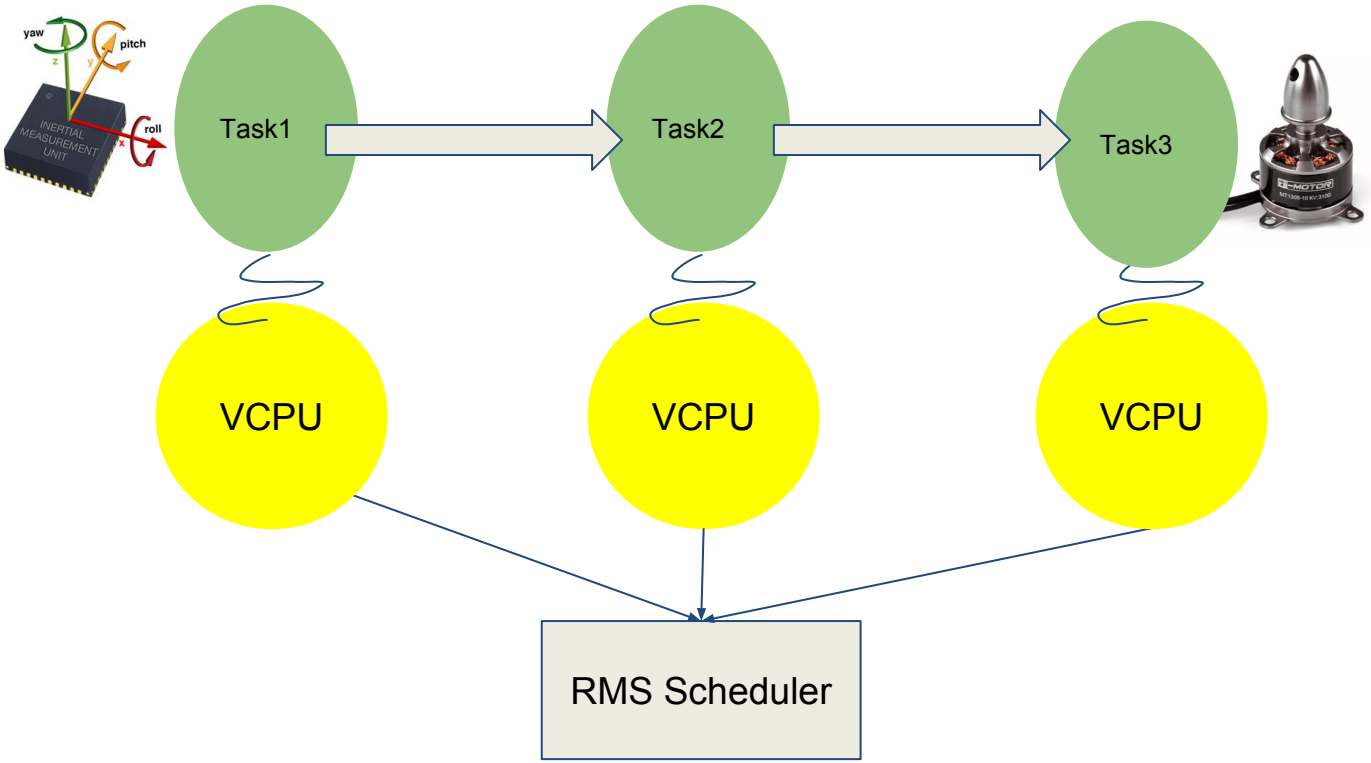
Challenges

- Apart from timing properties of individual tasks, …
- … also crucial to guarantee application-wide end-to-end times



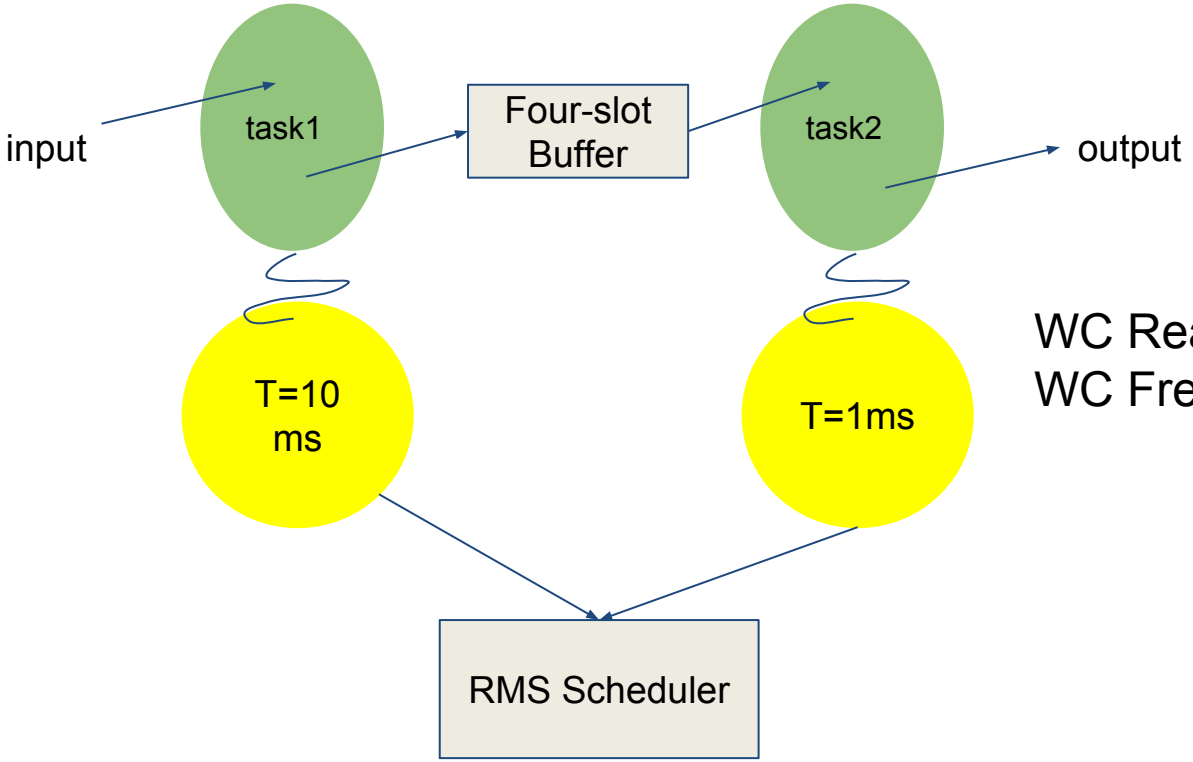
Task Pipeline

- A chain of tasks from sensor to actuator
  - Used to quantify system reaction time, etc.
  - E.g., Delay b/w motor speed reaction to attitude change



End-to-end Times

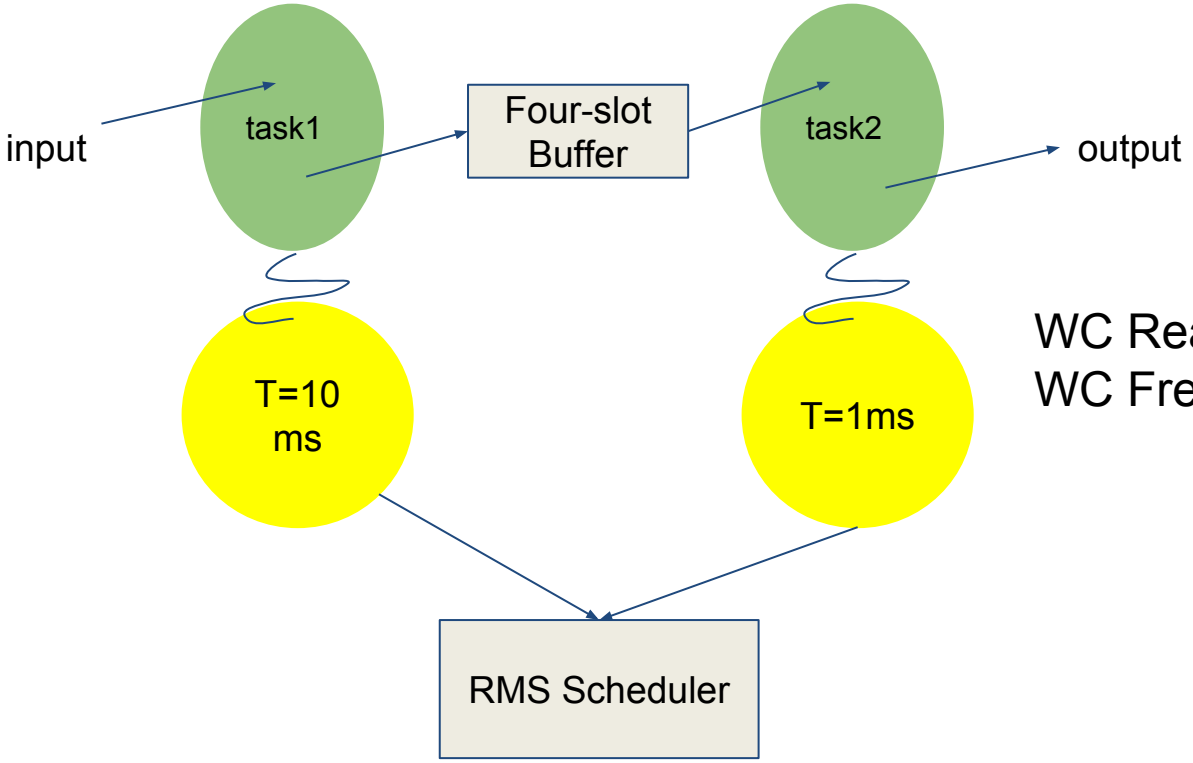
- Two semantics
  - **End-to-end reaction time:** the interval between a sampled input and its first corresponding output
  - **End-to-end freshness time:** the interval between a sampled input and its last corresponding output



WC Reaction: 1 ms
WC Freshness: 10 ms

End-to-end Times

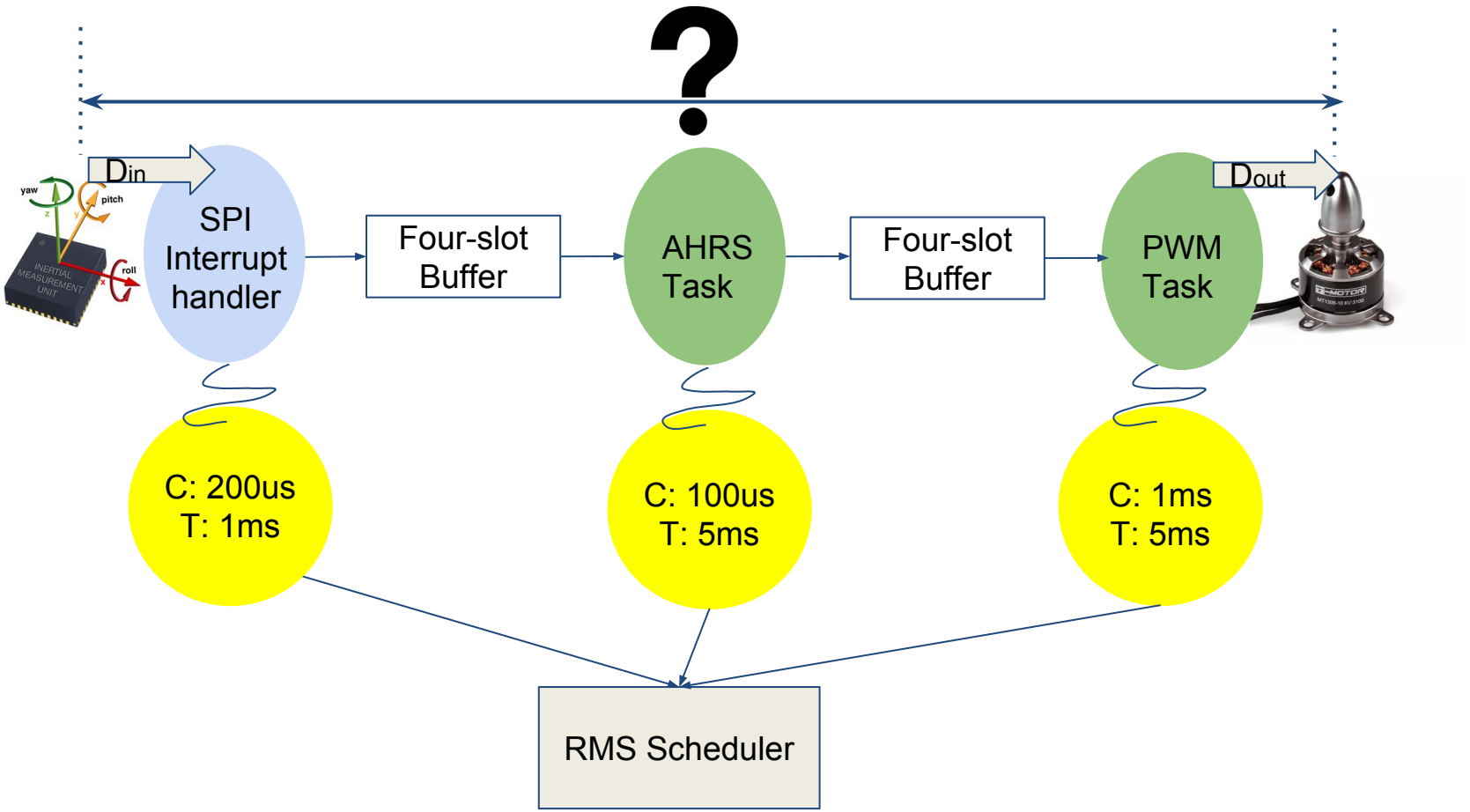
- Two semantics
  - **End-to-end reaction time:** affected by the consumer
  - **End-to-end freshness time:** affected by the producer
  - Use: a combination of reaction and freshness times can bound the periods of tasks



WC Reaction: 1 ms
WC Freshness: 10 ms

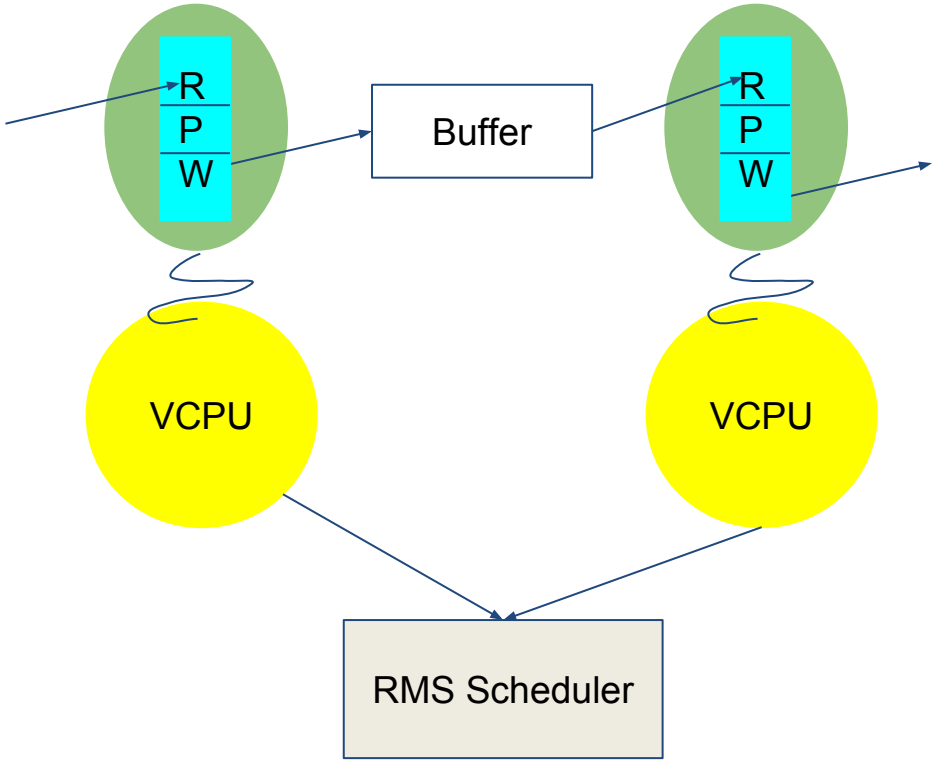
Problem Definition

- Given a task pipeline and VCPU parameters of each task within the pipeline, determine the pipeline's worst case end-to-end reaction and freshness times



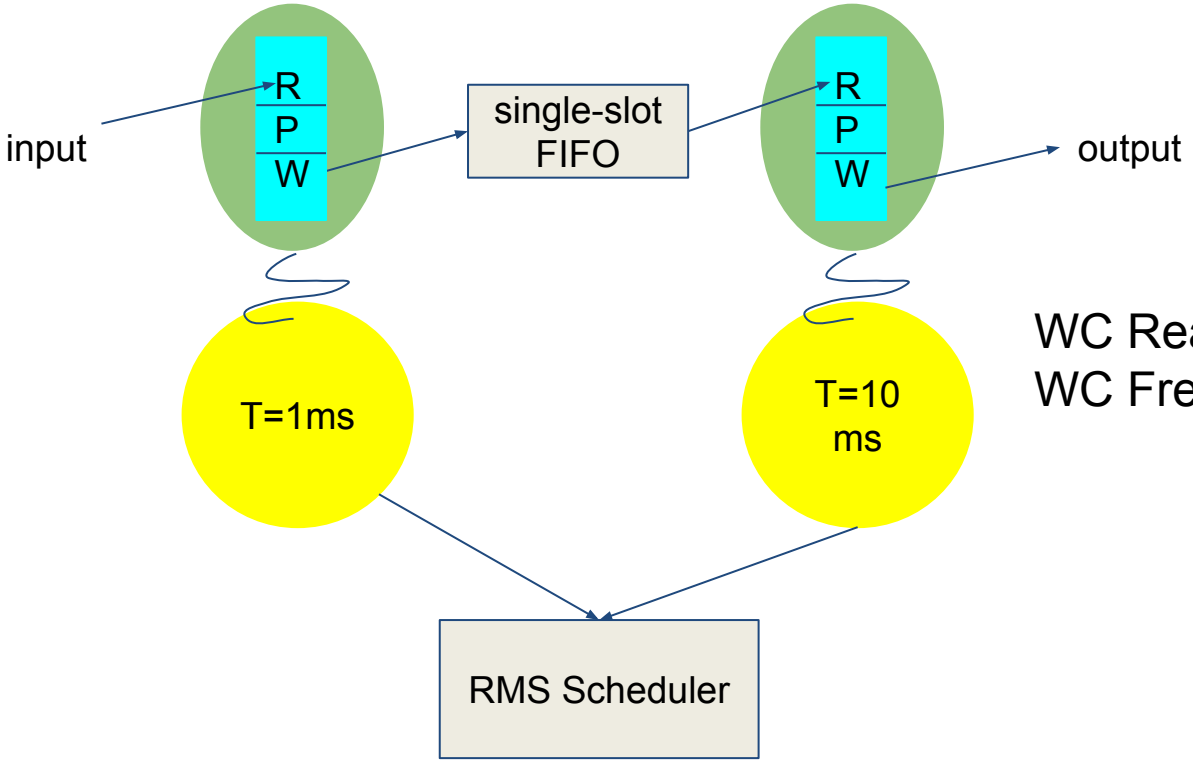
Execution Models

- Task model: periodic tasks
- Scheduling model: VCPU scheduling
- Communication model:
  - three stages: Read, Process, Write
  - Freshness-oriented: Simpson's four-slot buffer
  - Asynchronous



End-to-end Times

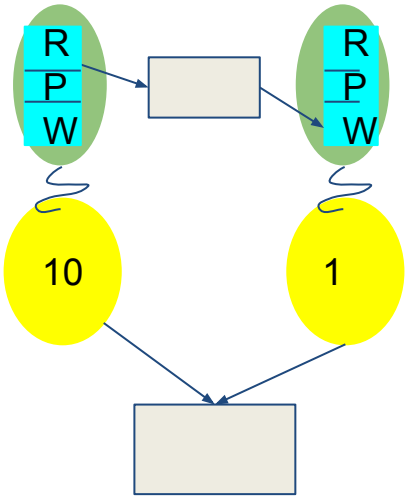
- Two semantics
  - **End-to-end reaction time:** the interval between a sampled input and its first corresponding output
  - **End-to-end freshness time:** the interval between a sampled input and its last corresponding output



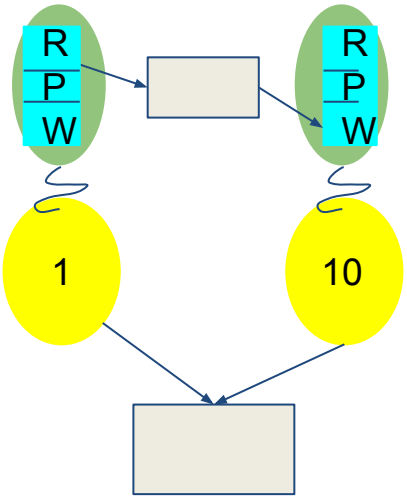
WC Reaction: 10 ms
WC Freshness: 1 ms

End-to-end Times

- Two semantics
  - **End-to-end reaction time:** affected by the consumer
  - **End-to-end freshness time:** affected by the producer
  - Intuition: a combination of reaction and freshness times bounds the periods of tasks



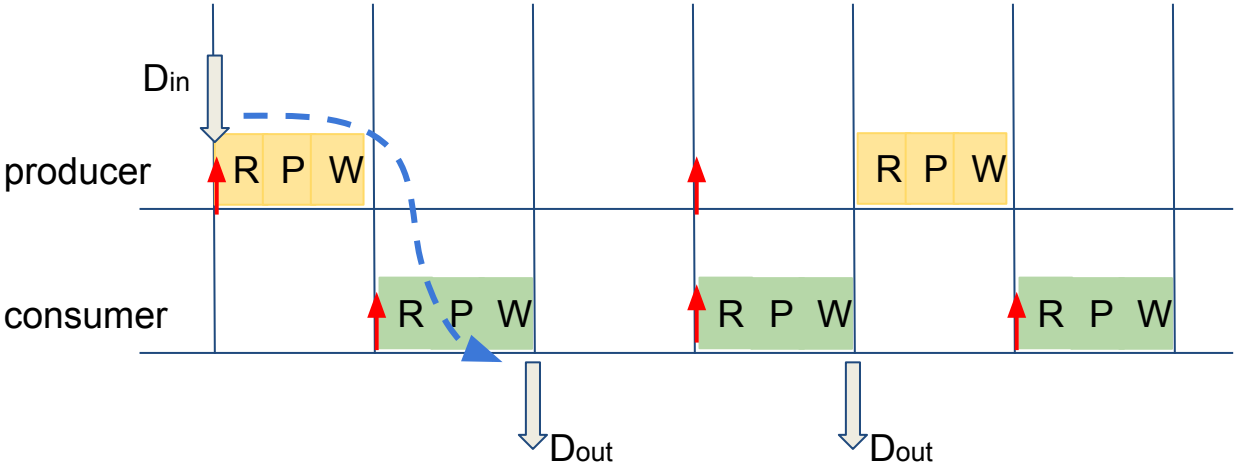
Reaction: 1
Freshness: 10



Reaction: 10
Freshness: 1

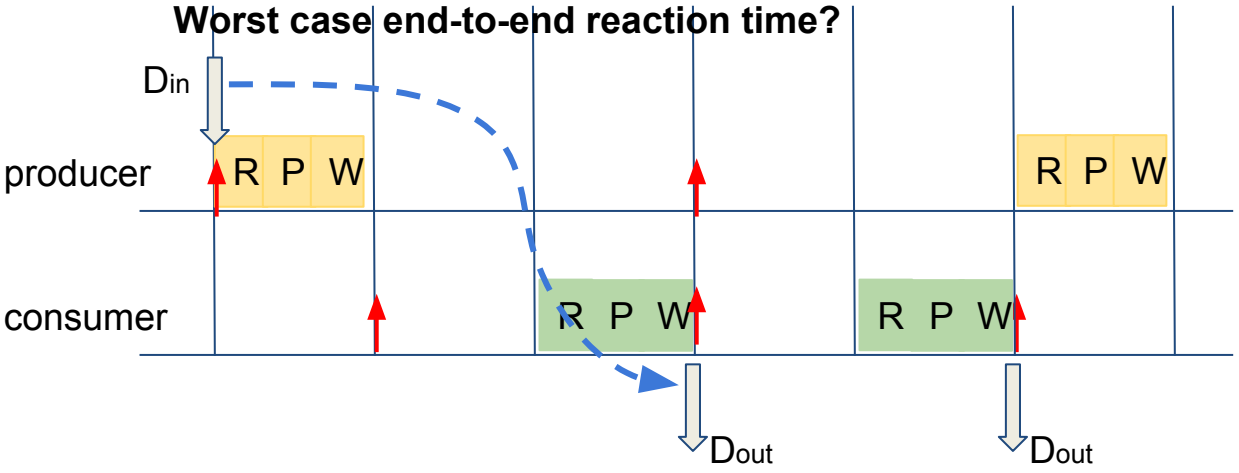
A Base Case

- Worst case reaction time: first corresponding output
- Pipeline of two tasks
- Priority: producer (T=3) < consumer (T=2)



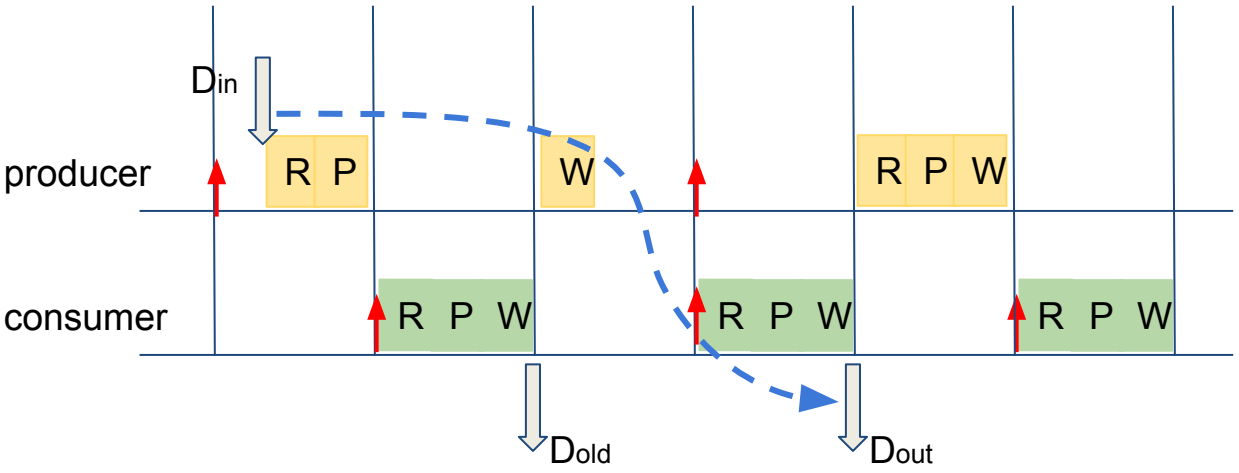
A Base Case

- Worst case reaction time: move apart
- Pipeline of two tasks
- Priority: producer (T=3) < consumer (T=2)



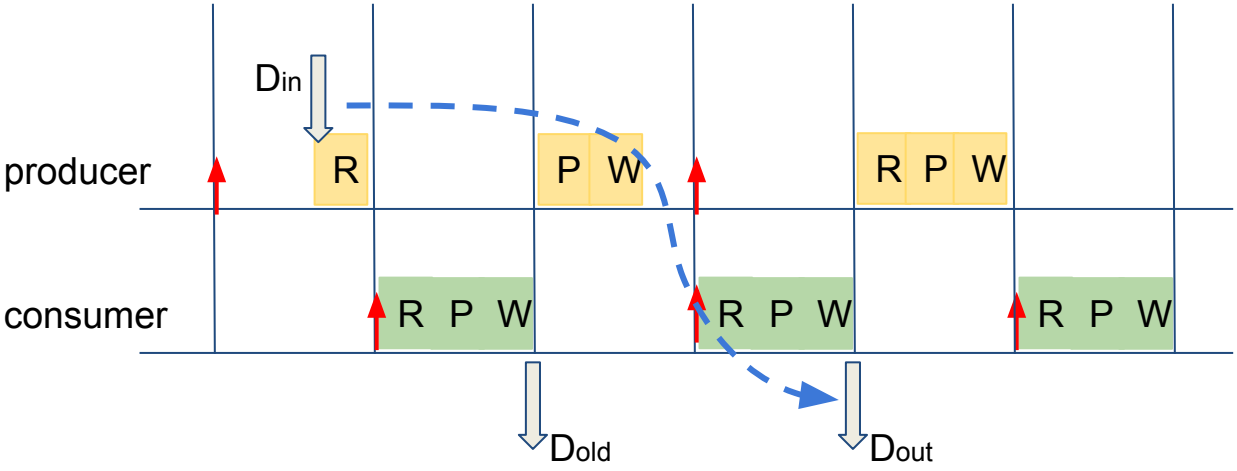
A Base Case

- Worst case reaction time: move closer
- Pipeline of two tasks
- Priority: producer (T=3) < consumer (T=2)



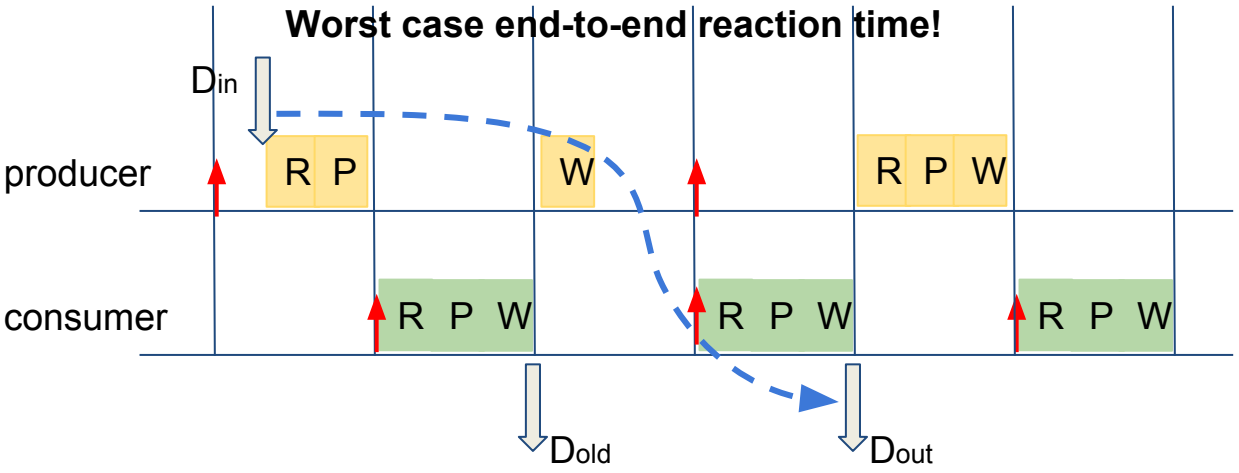
A Base Case

- Worst case reaction time: move even closer
- Pipeline of two tasks
- Priority: producer (T=3) < consumer (T=2)



A Base Case

- Worst case reaction time
- Pipeline of two tasks
- Priority: producer (T=3) < consumer (T=2)

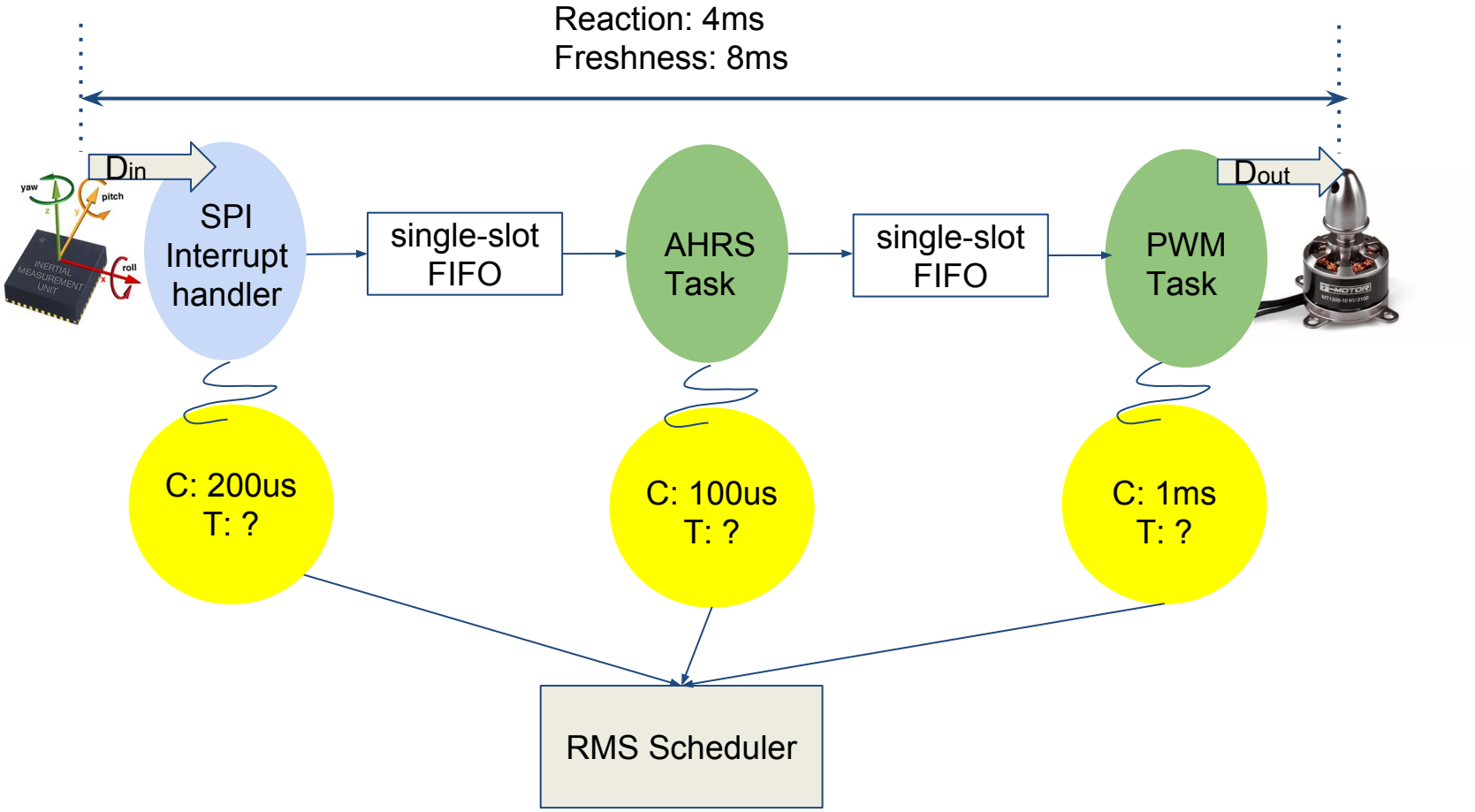


End-to-end Timing Analysis

- For two tasks, the same intuition applies for:
  - reaction time, producer has higher priority
  - freshness time, producer has lower priority
  - freshness time, producer has higher priority
- For longer pipelines:
  - Composability: appending tasks to a pipeline might preempt previous tasks, but will not affect the worst case reaction and freshness times of the prior tasks as long as all the tasks are schedulable
  - End-to-end time of the pipeline is extended by the period of each appended task plus scheduling latency b/w them

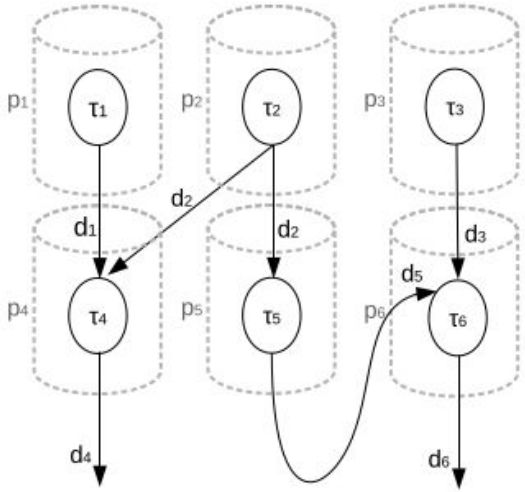
Flipping the Problem

- A combination of reaction and freshness times can bound the periods of tasks
- Given a pipeline's end-to-end timing requirements, determine its tasks' VCPU periods



End-to-end Design

- Worst case end-to-end times should be bounded by the specified requirements
- Use linear programming to find a feasible set of periods that satisfy the inequations
  - To prune the search space, start w/ $T_{prod} > T_{cons}$
  - Also use sensor & actuator hardware frequency

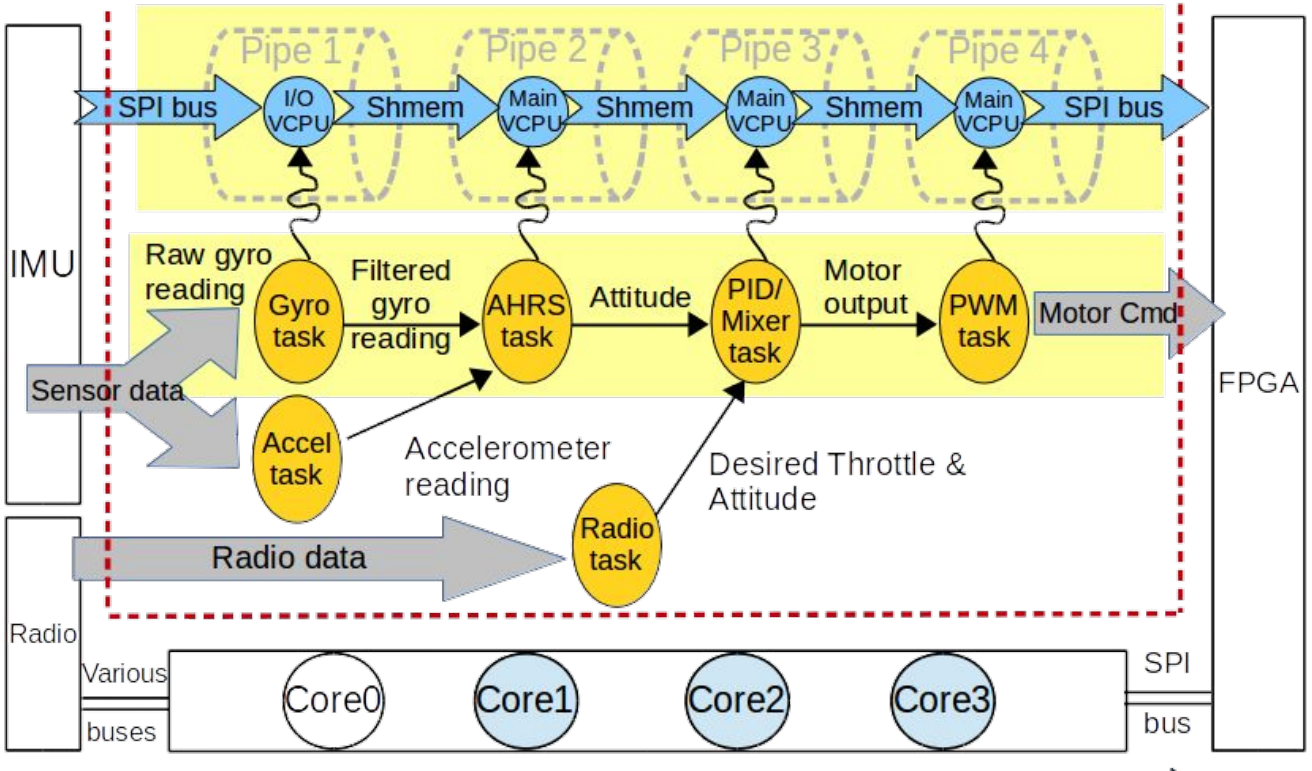


Reaction	Freshness
$E_{\tau_1 \mapsto \pi_1 \mid \tau_4 \mapsto \pi_4} \leq 10,$ $E_{\tau_2 \mapsto \pi_2 \mid \tau_4 \mapsto \pi_4} \leq 15,$ $E_{\tau_2 \mapsto \pi_2 \mid \tau_5 \mapsto \pi_5 \mid \tau_6 \mapsto \pi_6} \leq 25,$ $E_{\tau_3 \mapsto \pi_3 \mid \tau_6 \mapsto \pi_6} \leq 15;$	$F_{\tau_1 \mapsto \pi_1 \mid \tau_4 \mapsto \pi_4} \leq 20,$ $F_{\tau_2 \mapsto \pi_2 \mid \tau_4 \mapsto \pi_4} \leq 30,$ $F_{\tau_2 \mapsto \pi_2 \mid \tau_5 \mapsto \pi_5 \mid \tau_6 \mapsto \pi_6} \leq 50,$ $F_{\tau_3 \mapsto \pi_3 \mid \tau_6 \mapsto \pi_6} \leq 20;$
Schedulability	**Execution Time**
$\sum_{j=1}^{6} \frac{C^j}{T^j} \leq 6(\sqrt[6]{2} - 1)$	$\forall j \in \{1, 2, \cdots, 6\},$ $d_i^j = d_o^j = 3,$ $W_i^j = W_o^j = 20,$ $\delta_i^j = \delta_o^j = 0.1,$ $p^j = 0.5;$

Evaluation

- Intel Aero board:
  - Atom x7-Z8750 4 cores @1.6 GHz (only use core 0 for now)
  - On-board IMU, PWM

- A refactored multithreaded Cleanflight on Quest
  - Port Cleanflight to Linux to measure end-to-end times
  - Profile task execution time

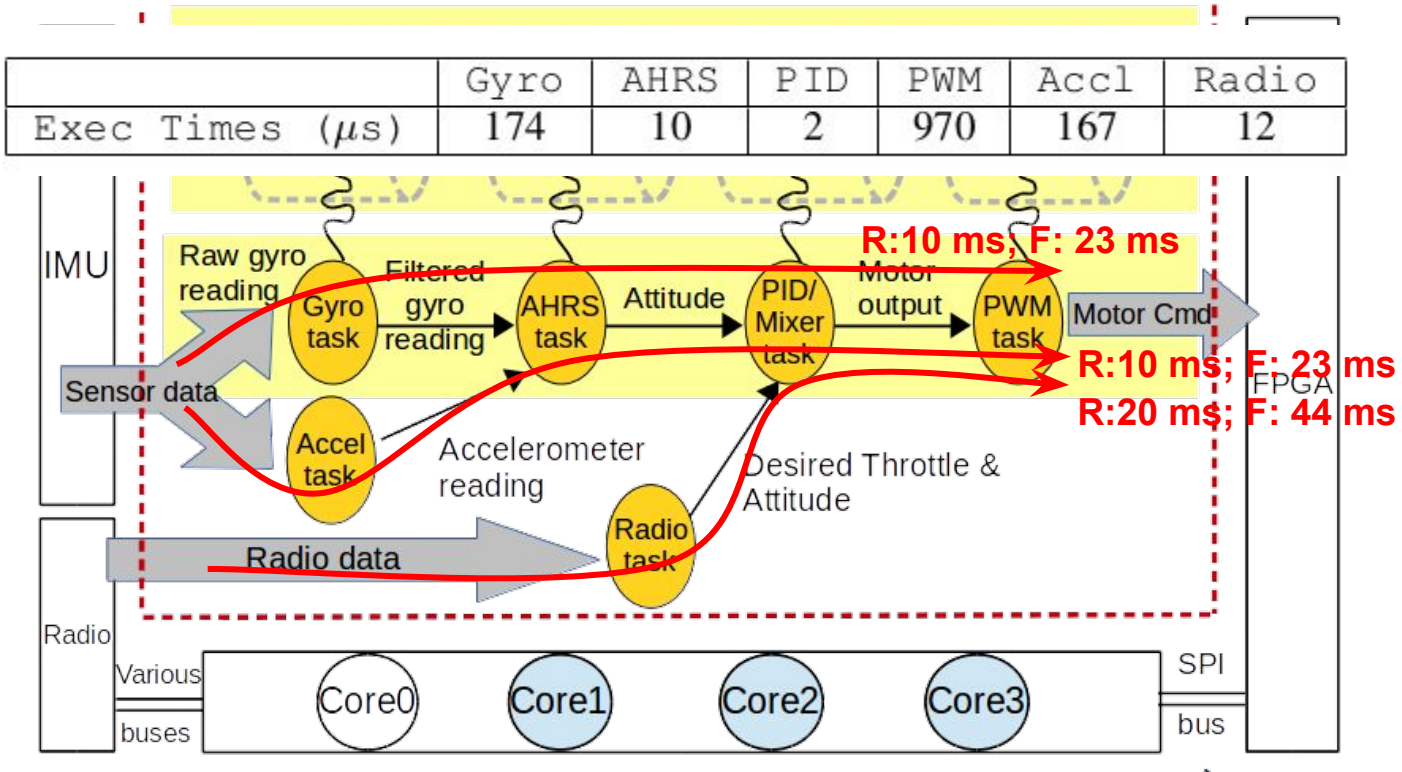


Evaluation

- Intel Aero board:
  - Atom x7-Z8750 4 cores @1.6 GHz (only use core 0 for now)
  - On-board IMU, PWM

- A refactored multithreaded Cleanflight on Quest
  - Port Cleanflight to Linux to measure end-to-end times
  - Profile task execution time

	Gyro	AHRS	PID	PWM	Accl	Radio
Exec Times (μs)	174	10	2	970	167	12

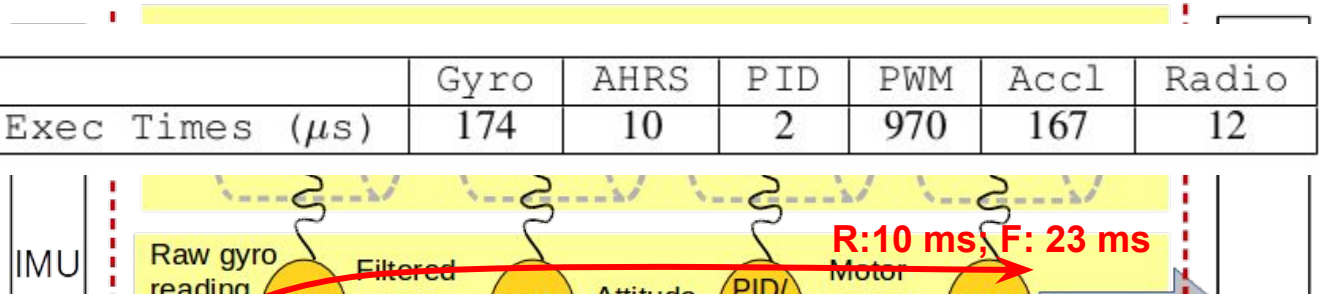


Evaluation

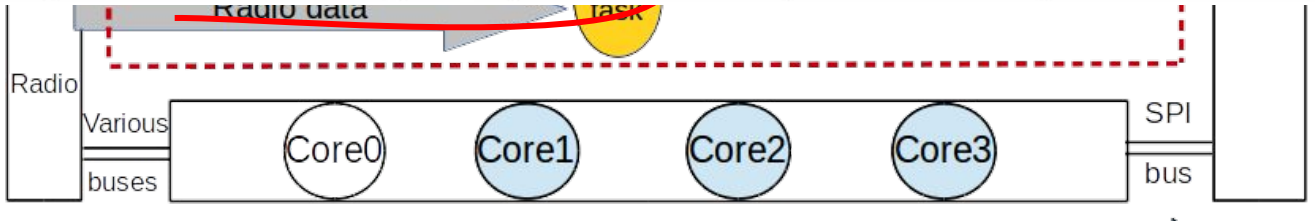
- Intel Aero board:
  - Atom x7-Z8750 4 cores @1.6 GHz (only use core 0 for now)
  - On-board IMU, PWM

- A refactored multithreaded Cleanflight on Quest
  - Use end-to-end design to derive task periods

	Gyro	AHRS	PID	PWM	Accl	Radio
Exec Times (μs)	174	10	2	970	167	12

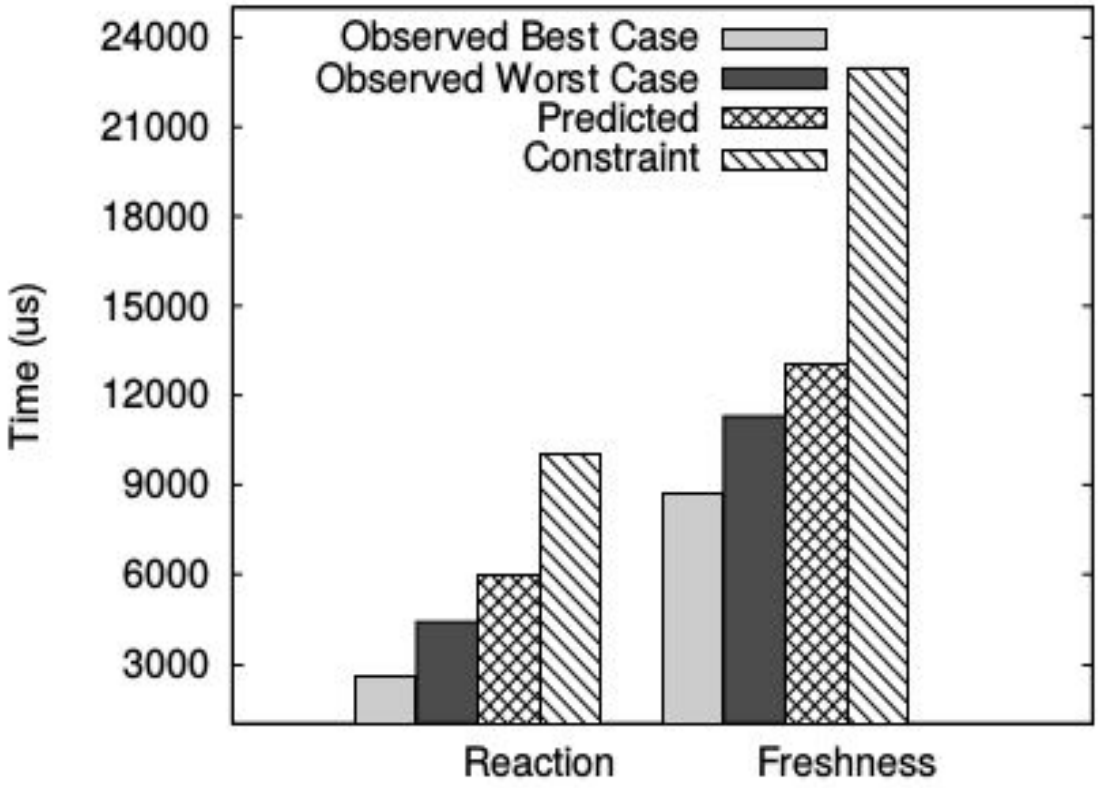


Task	Gyro	AHRS	PID
Budget/Period (μs)	200/1000	100/5000	100/2000
Task	PWM	Accl	Radio
Budget/Period (μs)	1000/5000	200/1000	100/10000



Evaluation

- Timestamp data exchanged along the task pipeline
  - Observed worst reaction and freshness end-to-end times are always less than timing constraints



Conclusion

- Temporal isolation between individual tasks can be used to derive worst-case end-to-end times of task pipelines
- End-to-end timing requirements can be used to derive task periods
- End-to-end timing analysis and design can be used to meet drone flight controllers' end-to-end timing requirements

Thank you

Comments or Questions?

Future Work

- Communication b/w flight controller & 3rd-party apps
- Applications for autonomous drone