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

Zhuoqun Cheng, Richard West, *Senior Member, IEEE,* and Craig Einstein

Presented by: Greg Kahl

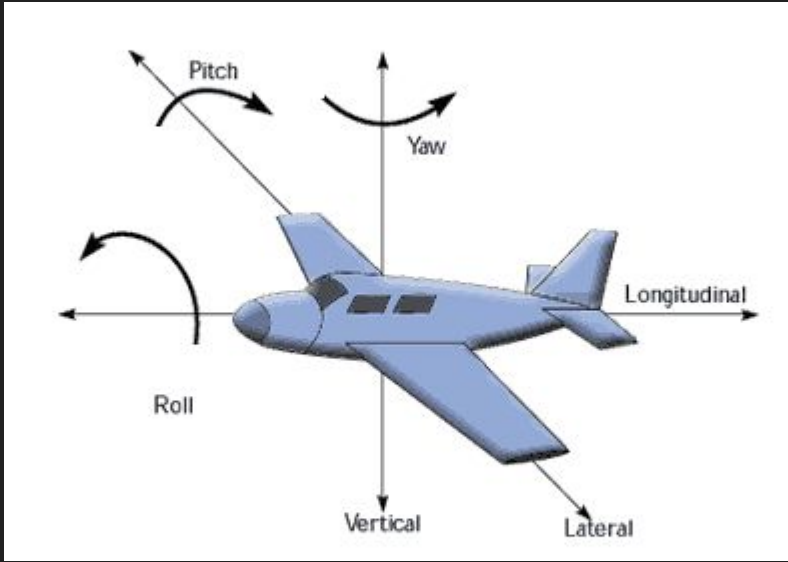
Introduction

- Drones use flight controllers to take in data from their environment and adjust their actuators (motors) appropriately
- In time critical systems, such as drone flight controllers, end to end delay between sensing, processing, and actuation are very important.
- If processing of the sensor data, or actuation of the motors, is too slow the drone will fail to stay in flight.

- Are there any other time critical systems you can think of that may face similar problems?

Attitude of Drone

- The attitude is the Roll, Pitch, and Yaw
- Offset relative to a reference frame, most commonly the earth or pilot on the ground



What is meant by End to End?

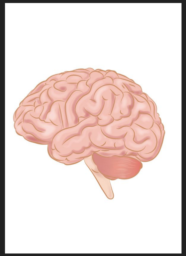
- In Real Time systems there are two aspects to the end to end time.
  - Reaction Time
  - Freshness
- Consider these two constraints:
  - A change in motor speed must be within 2ms of the gyro sensor reading that caused the changed
  - An update to a gyro sensor value must be within 2ms of the corresponding update in motor speed
- Which constraint do you think relates to which of the aspects of the end to end time?

Scheduling Model

- **Periodic** over Aperiodic Tasks: simplifies timing analysis
- **Register based** communication over FIFO Based communication:
  - This allows of the use of the freshest data, not just the data was the next in the queue
- **Implicit** vs Explicit Communication:
  - Explicit Communication: allows shared data access at any point during the task
  - Implicit Communication: Must make a local copy of data the task wishes to use.
- => Data freshness and Consistency

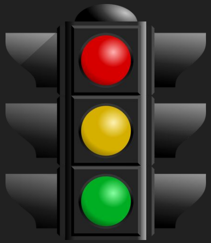
Latency Contributors

- Processing Latency
  - Time it takes to process the data



- Communication Latency
  - Time it takes to send the data to the next stage/pipe

- Scheduling Latency
  - Amount of time between the data arriving and the next pipe being scheduled to work on the data



Pipe Model

- They model a pipe as having one pipe terminal and two pipe ends (an input and output end)
- Pipes and tasks have a one-to-one relationship
- Pipe terminal is represented by a Virtual CPU guaranteed at least C units of execution time every T time units.

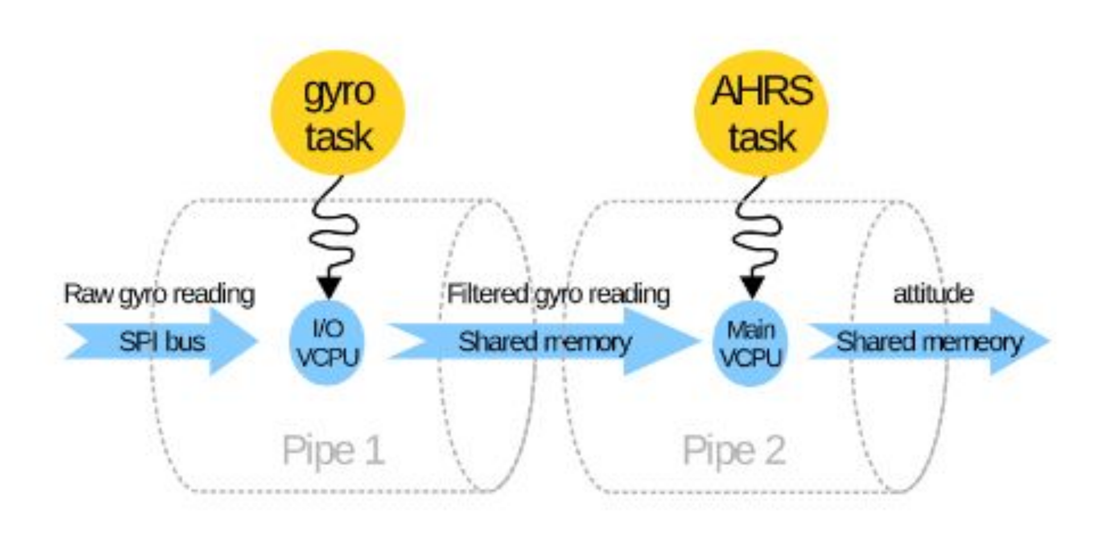


Illustration of two communicating pipes

Pipe Characteristics

$$\pi = ((W_i, \delta_i), (C, T), (W_o, \delta_o)).$$

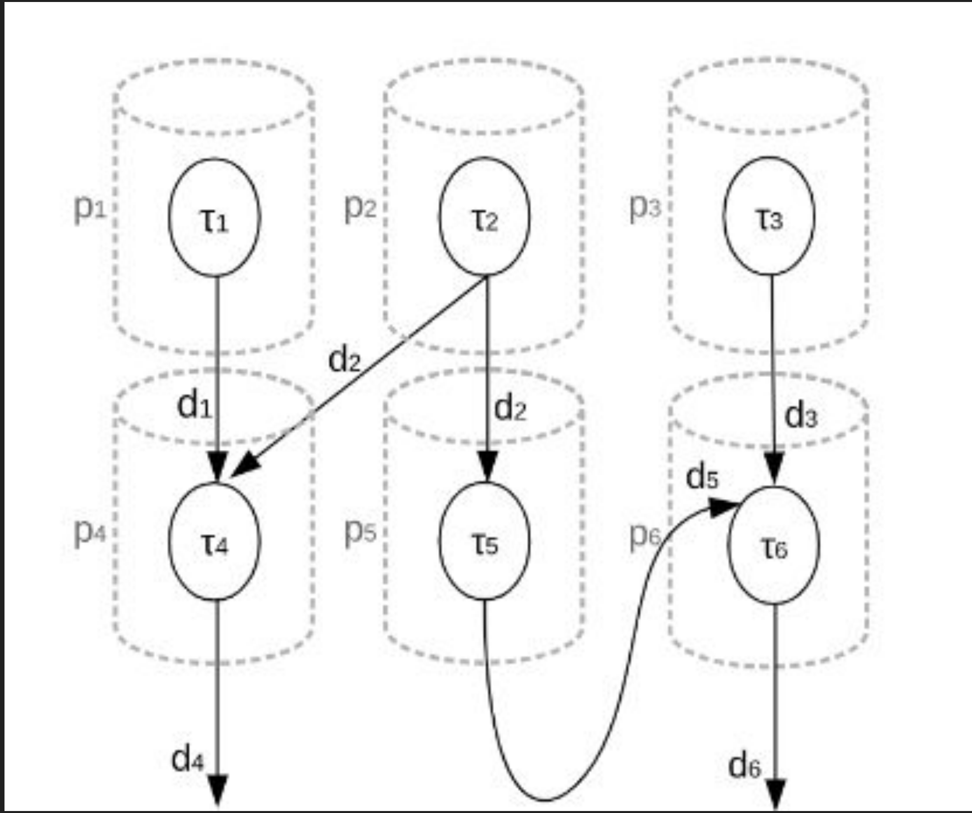
- (W,δ) denotes the bandwidth and software overheads of the input and output terminals
- T denotes the period of the pipe terminal
- C denotes the budget of the pipe terminal, how many cycles the terminal is guaranteed during the period

Tasks

- Each task consists of three stages, the read stage, process stage, and write stage.
- When determining the budget of the pipe terminal a task is assigned to, you should ensure that all three stages can finish in one period.

Pipe Model

- This a task graph showing each task contained in a pipe terminal where the outputs of some pipes are used as inputs for the next.
- Note: Data from Pipe 2 is not necessarily duplicated to be fed to Pipe 4 and 5 due to register based sharing



Task Graph Using Pipes

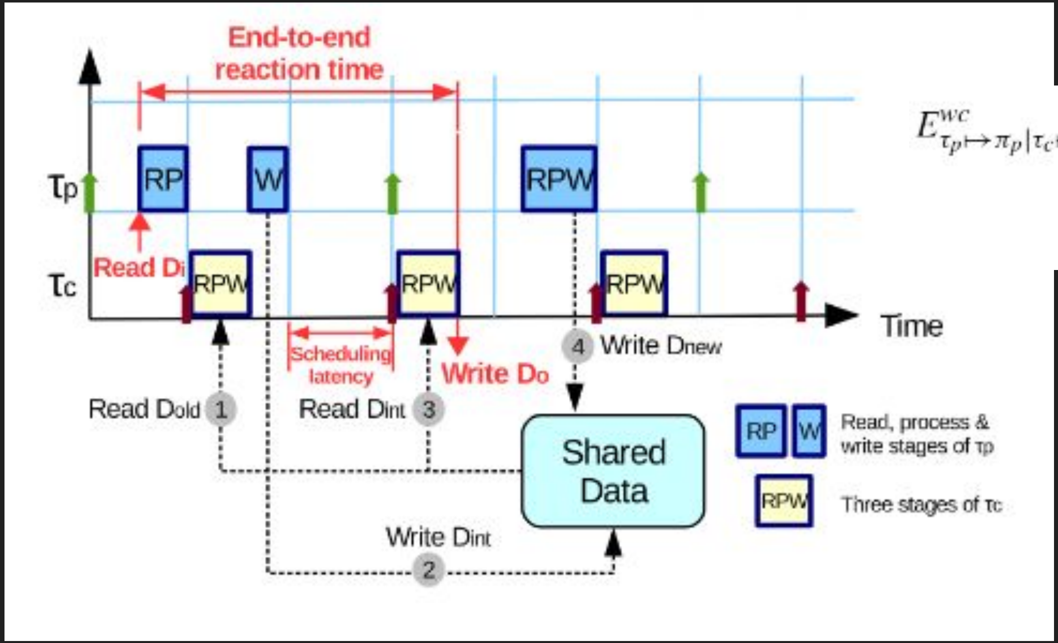
Timing Analysis

$$L^{wc}_{\tau \mapsto \pi} = \left\lfloor \frac{\Delta_{\text{in}} + p + \Delta_{\text{out}}}{C} \right\rfloor \cdot T + (\Delta_{\text{in}} + p + \Delta_{\text{out}}) \bmod C \tag{1}$$

- They used this equation to represent the worst case latency of the system
- Δin and Δout are the communication latency + the software overhead of inputs and outputs

Worst Case Reaction Time of Pipe Chain: Case 1

Case 1 is when the consumer pipe/task has a shorter period than the producer, causing it to have a higher priority according to rate-monotonic scheduling.



$$
\begin{aligned}
E^{wc}_{\tau_p\mapsto\pi_p|\tau_c\mapsto\pi_c} &= L^{wc}_{\tau_p\mapsto\pi_p} + S^{wc}_{\tau_p\mapsto\pi_p|\tau_c\mapsto\pi_c} + L^{wc}_{\tau_c\mapsto\pi_c}\\
&= L^{wc}_{\tau_p\mapsto\pi_p} + L^{wc}_{\tau_c\mapsto\pi_c} + T^c - C^c - \left(\frac{d}{W^p_o} + \delta^p_o\right).
\end{aligned}
$$

Latency of Producer

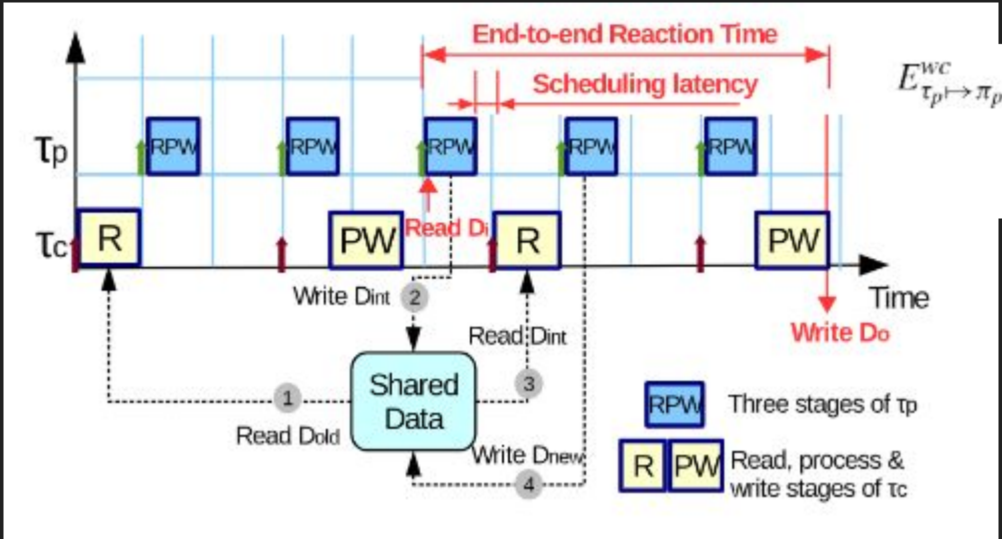
Latency of Consumer

Consumer Period - Consumer Budget

and Time to write new value

Worst Case Reaction Time of Pipe Chain: Case 2

Case 2 is the opposite of Case 1, the producer has a shorter period, and thus higher priority.



$$
\begin{aligned}
E^{wc}_{\tau_p\mapsto\pi_p|\tau_c\mapsto\pi_c} &= L^{wc}_{\tau_p\mapsto\pi_p} + S^{wc}_{\tau_p\mapsto\pi_p|\tau_c\mapsto\pi_c} + L^{wc}_{\tau_c\mapsto\pi_c} \\
&= L^{wc}_{\tau_p\mapsto\pi_p} + L^{wc}_{\tau_c\mapsto\pi_c} + T^p - C^p - \left(\frac{d}{W_i^c} + \delta_i^c\right).
\end{aligned}
$$

Latency of Producer

Latency of Consumer

Producer Period - Producer Budget

and Time to Read value

Simplifications

$$E^{wc}_{\tau_p \mapsto \pi_p | \tau_c \mapsto \pi_c} = \begin{cases} T^c - C^c - \left(\frac{d}{W} + \delta\right) & \\ +L^{wc}_{\tau_p \mapsto \pi_p} + L^{wc}_{\tau_c \mapsto \pi_c}, & \text{if } T^c < T^p \\ T^p - C^p - \left(\frac{d}{W} + \delta\right) & \\ +L^{wc}_{\tau_p \mapsto \pi_p} + L^{wc}_{\tau_c \mapsto \pi_c}, & \text{otherwise.} \end{cases} \quad (6)$$

where $W = W^p_o = W^c_i$ and $\delta = \delta^p_o = \delta^c_i$.

$$E^{wc}_{\tau_p \mapsto \pi_p | \tau_c \mapsto \pi_c} = \begin{cases} T^c + C^p, & \text{if } T^c < T^p \\ T^p + C^c, & \text{otherwise.} \end{cases}$$

Assumption:

$$\begin{aligned} L^{wc}_{\tau \mapsto \pi} &= \lfloor \frac{C - \epsilon}{C} \rfloor \cdot T + [(C - \epsilon) \mod C] \\ &= 0 \cdot T + (C - \epsilon) \approx C. \end{aligned}$$

Where ϵ is the arbitrarily small surplus to the budget

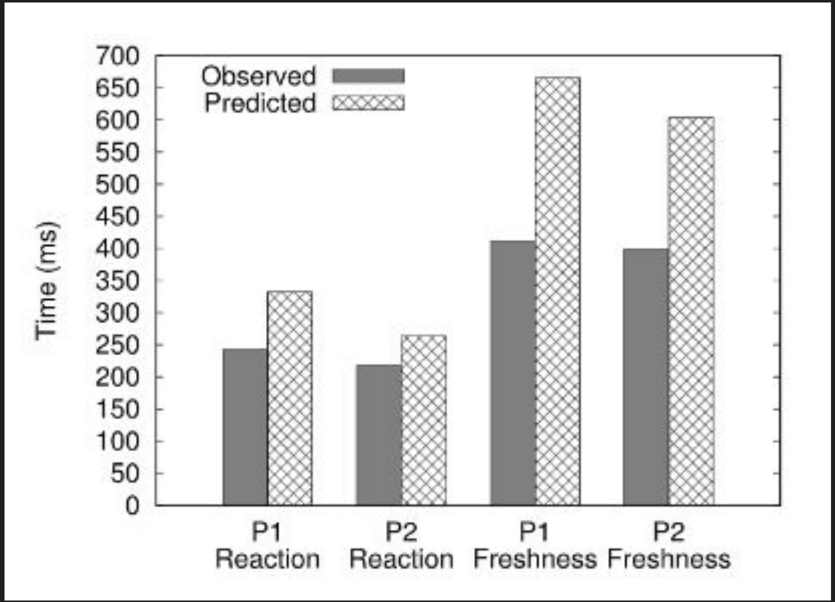
And communication overhead is 0 when communicating data of a small size using shared caches

Solving the Constraints

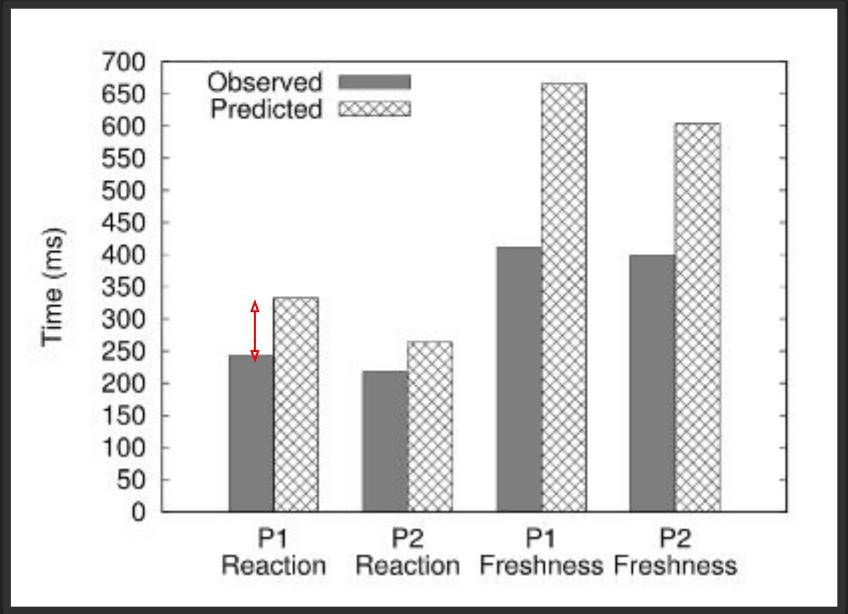
τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
11.5	5.5	3.5	5.5	11.5	11.5	3.5
PT 1	PT 2	PT 3	PT 4	PT 5	PT 6	PT 7
(12,100)	(6,50)	(4,150)	(6,100)	(12,150)	(12,100)	(4,50)

- Using the task model from the previous slides and a constraints table they derive a set of feasible task periods.
- The goal is to find the max value for each period, so that the total CPU Utilization is minimized.
- Optimal solution depends on which constraints are deemed more important, which they claim is out of the scope of this paper.

Simulation Results

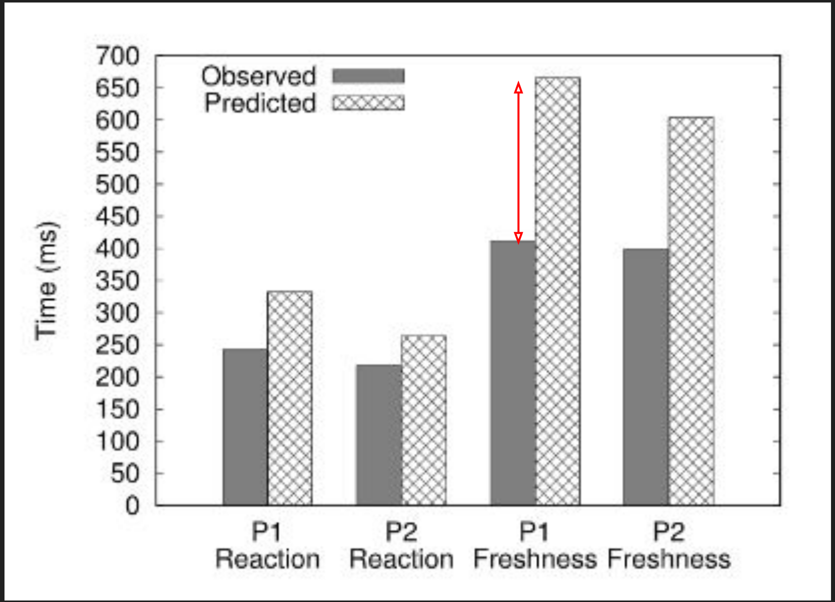


Simulation Results



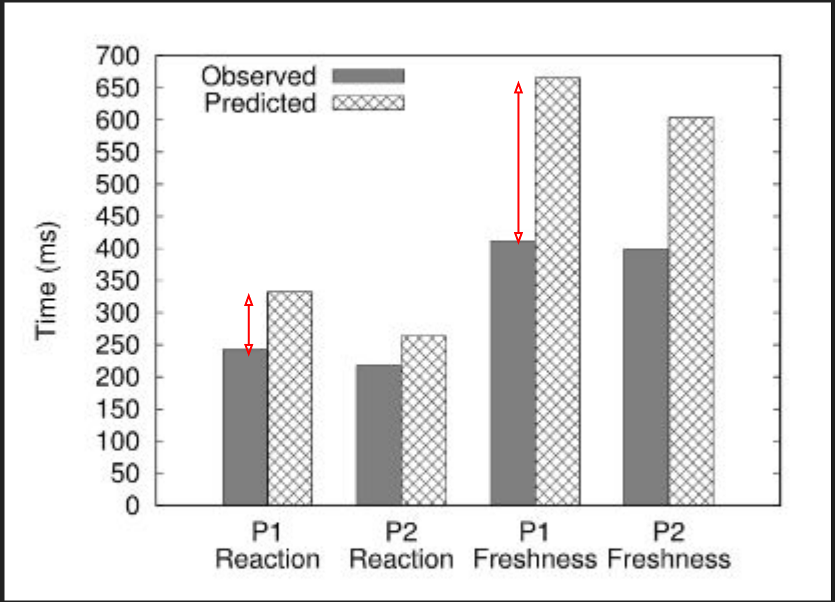
- The observed values should always be within the predicted bounds.
- If not, your system is not working as you predicted, it is performing worse
- You do want it to be fairly close, you want your prediction to be accurate

Simulation Results



- The difference between predicted and observed freshness is larger than the difference for reaction time.

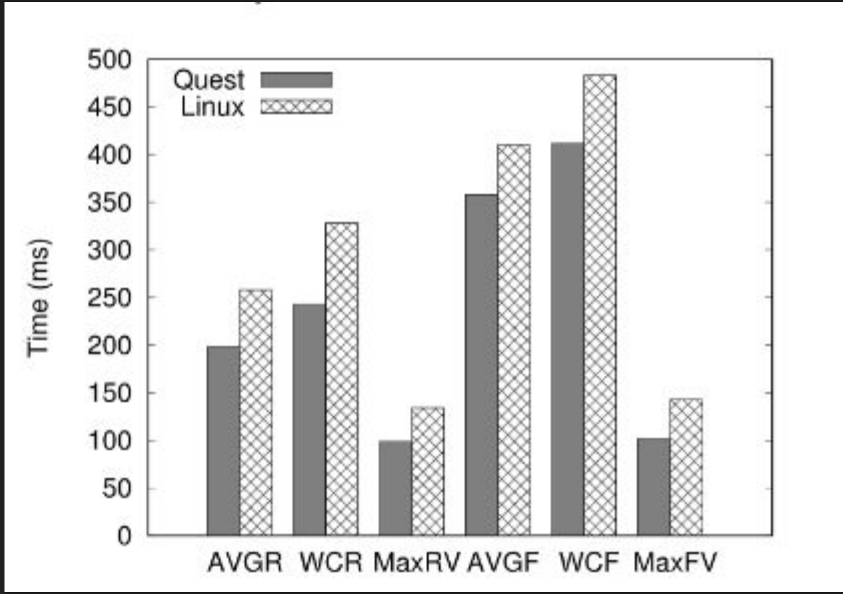
Simulation Results



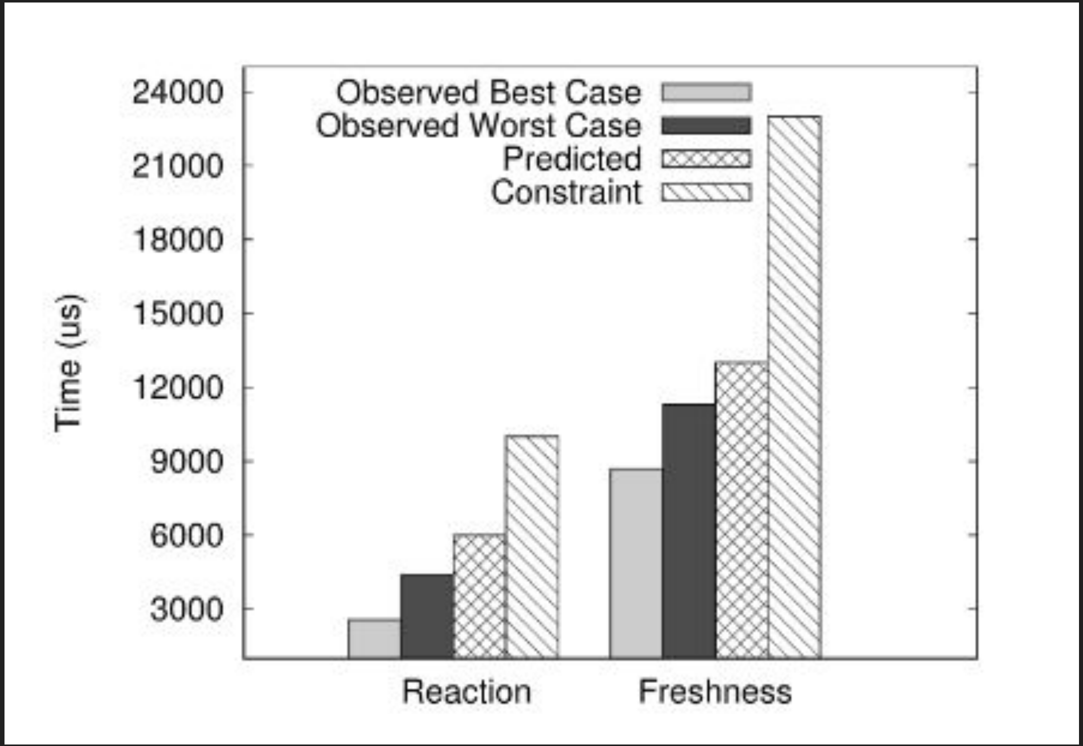
- The prediction isn't as tight for freshness because their implementation gives producers a greater period by default.
  - Freshness depends on the period of the producer; Reaction on the period of the consumer

Simulation Results

- Implementation on Quest vs Linux



Cleanflight Implementation Results



- They implemented it on an instance of Cleanflight.
- The constraints were taken from implementation on Linux, they were not as worried about what the exact constraint was, but that the system successfully met them

Critiques and Questions

- The section of the paper discussing the Timing Analysis was overly complicated and quite confusing
- How do you think security would affect timing requirements?
- The authors said they turned off nontime-critical functions for testing such as the black box, but isn't that still an important feature that could affect actuation?

Conclusions

- The composable pipe model using the abstractions explained allowed them to set task periods and budgets so that the end-to-end freshness and reaction time fits within constraints for time critical systems.
- They ran experiments using Cleanflight ported to their Quest RTOS and achieved end-to-end latencies within the desired bounds.
- Future work will continue the project to use the pipe model to create a fully autonomous flight management system with Cleanflight implemented on Quest.