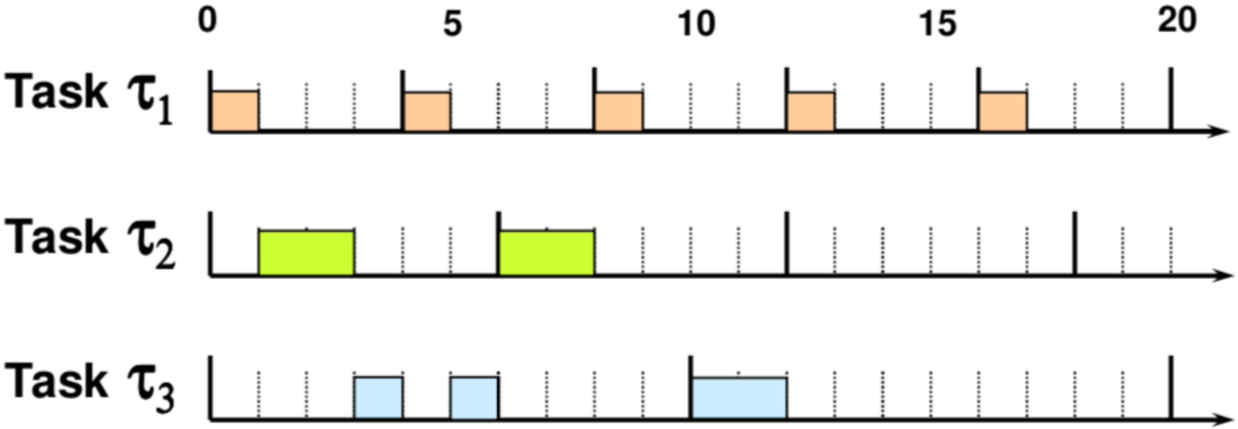
Regular Operation “Runtime”

- RTOS executes a number of Periodic/Aperiodic tasks
- Tasks can be divided into **Data Sampling**, **System Maintenance**, and **Command Response**
- **Data Sampling:** Collecting data from various subsystems/sensors at rates (minimize downsampling to acceptable margins).
- **System Maintenance:** Regular, recurring tasks. (Toggle heat, Send Heartbeat, Telemeter)
- **Command Response**: Aperiodic response to Teleop or Autonomy Commands
- These Runtime Tasks will be guided by the Software Requirements Document

RTOS Periodicity & Schedulability

- RTOS will execute Isolated software functions as “Tasks” with defining values: {C = Worst Case Execution Time, D = Deadline, T = Period}
- Utility = C/T, And the Scheduling algorithm sets the bounds on maximum Utility for a set of tasks to remain schedulable



{1,5,5}

{2,7,7}

{2,12,12}

Dealing with Aperiodic Commands

- Aperiodics, depending on response time requirements, can be serviced in a number of models. These will come implemented in the RTOS we select
- **Polling Server:** Poll the Wifi Module or GPU Message Queue periodically and act on the command if there is anything pending. This is the simplest option, but quantizes the response time
- **Sporadic Server:** Service aperiodic requests immediately, but only allow them a budget that replenishes periodically. Schedulability can be proved via RMA techniques
- **Bandwidth Servers …** Many different models exist, of varying Complexity and ranging RTOS support. Requirements should clear up MoonRanger needs.

Example of an Aperiodic Server

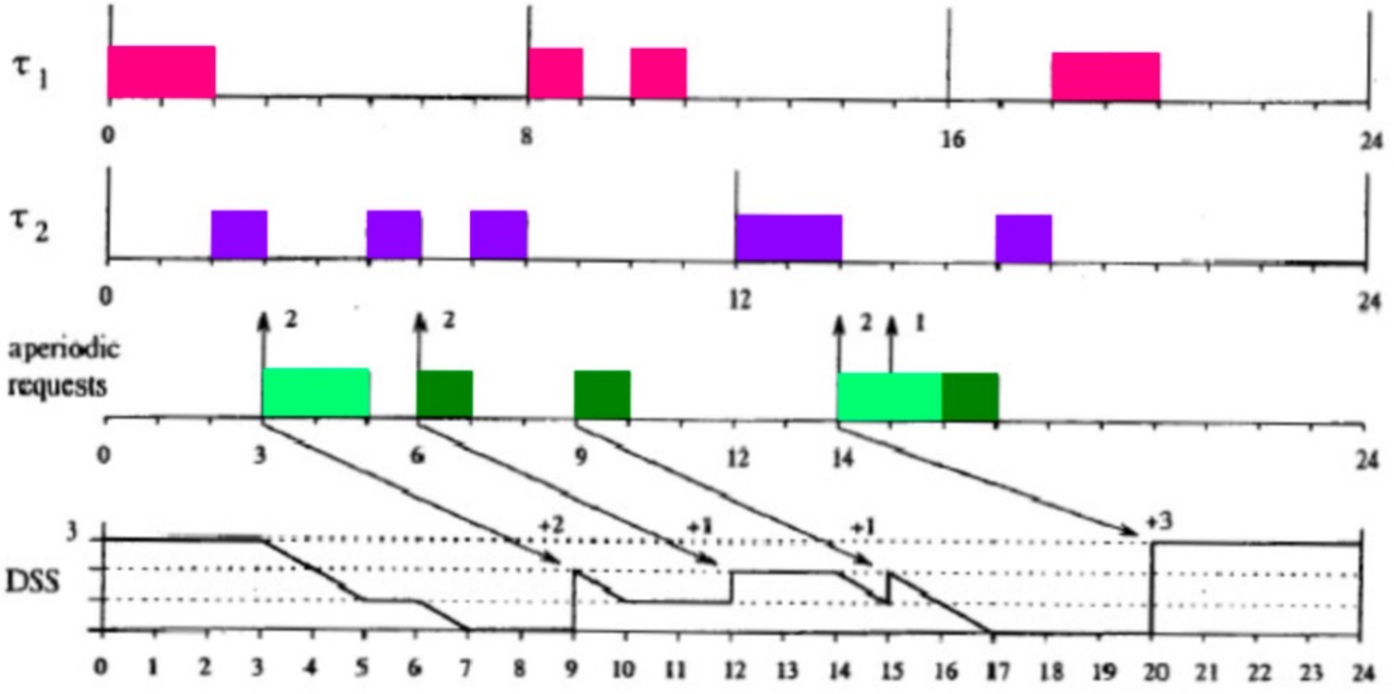
Periodic Tasks,
proven schedulable via Liu-Leyland bound

(*C*=2, *T*=8)

(3,12)

Aperiodic requests
with Budget Refresh

(3,6)



RTOS Algorithm Priority vs. Functional Priority

- RTOS will support declaring Tasks of different priorities (freeRTOS, and several others, may simply schedule these tasks in a preemptive fashion)
- Declaring priorities based on “Functional Priority”, or importance, is not necessarily optimal.
- Developers use scheduling techniques such as Rate Monotonic Scheduling, Earliest Deadline First, etc… to assign priorities in a way that provides a mathematical framework to prove correctness (Liu-Leyland Bound https://www.cs.ru.nl/~hooman/DES/liu-layland.pdf).
- Using standard manipulation techniques, The best designs are where these two methods of assigning priorities yield the same result.

RTOS Execution Models

- **Mode-Switching Static Tasksets** : Partitioning functional capabilities into real-time tasks is defined statically, but switching between multiple modes can toggle presets. **Example:** Devices with Low-Power Modes, Flight mode, etc…
- **Static Tasksets** : Task allocation is defined statically and remains constant throughout operation. **Example:** Keyboard Driver
- **Dynamic Tasksets** : Task allocation done dynamically to service incoming requests. **Example:** Router, Communications Devices
- MoonRanger best suited for a Mode-Switching Static Taskset model

Work to be done

- Isolate System Requirements + Summarize in the Software Requirements Document
- Partition Requirements into a Taskset (Periodics and Aperiodics)
- Mathematically prove schedulability of the taskset with a chosen algorithm, taking into account number of cores and frequency scaling available on the MCB.
- Verify our computation time estimates on the RTOS with Hardware and low-level drivers in the loop.