1. [5 points] make yourself an account on your Dev Kit. To do this, use the reset button if the system is locked, use your password to login, and then use “sudo adduser”, enter a password, and enter user information as you see fit. Add your new user account as a “sudoer” using “visudo” right below root with the same privileges (if you need help with “vi”, here’s a quick reference or reference card– use arrows to position cursor, below root hit Esc, “i” for insert, type username and privileges as above, and when done, Esc, “:”, “wq”). The old unix vi editor was one of the first full-screen visual editors – it still has the advantage of being found on virtually any Unix system in existence, but is otherwise cryptic – along with Emacs it is still widely used in IT, by developers and systems engineers, so it’s good to know the basics. If you really don’t like vi or Emacs, your next best bet is “nano” for Unix systems. Do a quick “sudo whoami” to demonstrate success. Logout of Linux and test your login, then logout. Use Alt+Print-Screen to capture your desktop and save as proof you set up your account. Note that you can always get a terminal with Ctrl+Alt+t key combination. If you don’t like the desktop, you can try “GNOME Flashback” and please play around with customizing your account as you wish.
2. [10 points] Read the paper "Architecture of the Space Shuttle Primary Avionics Software System" [also available on Canvas, “shuttle\_paper.pdf”], by Gene Carlow and provide an explanation and critique of the frequency executive architecture. What advantages and disadvantages does the frequency executive have compared to the real-time threading and tasking implementation methods for real-time software systems? Please be specific about the advantages and disadvantages and provide at least 3 advantages as well as 3 disadvantages.

Solution:

1. Technical Paper Summary and Speculation

* This technical paper primarily discusses about the architecture and working principle of Space Shuttle Primary Avionics Software System (PASS) and allows us to delve into real-time concepts associated with it.
* Its intricate system architecture supports a congregation of objectives to assist disparate shuttle operations such as computer redundancy modules to handle the system during errors and faults.
* Because of the massive software configuration size, the system needs mass memory along with (AP-101 with 106K 32-bit words) it is divided into 8 operational sequences representing 8 operational flight stages.
* In addition, Flight Computer Operating System (FCOS) handles resource allocation as a response to the OPS sections’ requests and manages I/O operations and redundancy.

1. PASS Architectural Drivers

* PASS is the cardinal module to a majority of space shuttle orbiter system functions.
* The factors which influence the architecture of the PASS are:
* Data processing System(DBS) / IBM AP-101 General Purpose Computer(GPC)
* Multicomputer redundancy management and synchronization.
* Operation sequencing mode/control and man/ machine interface requirements.
* Multiple sources of requirements definition, verification and validation and support of the PASS.

1. Operational Structure

* The operational structure main memory has a size of AP-101 (106K 32-bit words). Which is insufficient for the software (500K) containing the PASS requirements.
* The software can be said to be reliable when it handles a critical mission phase (such as ascent) be keeping the redundant GPC main memory for the complete phase to ensure that working of the system in case of main system failure.
* The mass memory feeds the software required for each OPS into the GPC main memory from the mass memory by during initiating the operational sequence. While the non-critical situations like loading of the display during on-orbit coast periods are prevented from loading to the GPC’s main memory when an OPS is executing.
* OPS memory load has three parts:

1. Resident/System Software – OPS loads’ common code/data.
2. Major function base – code & data common to major app functions
3. OPS overlay – applications code and unique OPS data

* The current OPS contents decide which of the three parts must be loaded to support new OPS.

1. Man/ Machine Interface :

* The PASS system consists of man/machine interface where the team controls and monitors orbitar avionics system by interacting with the computers.
* The multimodal OPS makes use of the OPS subordinate structures.
* OPS has major modes sub-structured into blocks linked to CRT displays.
* Switching from one mode/block to another one takes place in two ways

1. Manually done by keyboard input
2. Software detects occurance of specific event or condition without any human intervention.

* SPEC which is the OPS sub-structure’s second element is initiated when the crew member provides keyboard input.
* Once initiated, SPEC executes without any dependencies and in parallel manner along with other processes
* The Display Function (DISP) which is the third element of OPS doesn’t require initialization.
* Finally, the control segments develop the OPS structure, major mode, SPEC or DISP using standardized logic blocks series.

1. System Software :

* The flight system software consists of small size of 15-30k & limited set of functions.
* A synchronous design strategy is adopted by the software architectures of these flight systems where a specific errand is dispatched according to a stringent timeline relative to start point of overall system cycle.
* The associated I/O needs to be strictly synchronized to the start and end of different application processes to prevent overruns at system and process levels
* The reason why a synchronous approach was adopted for the architecture of PASS system was because of a disparate number of application functions needed to be performed in the shuttle program
* Sequence and control operations accomplishment 🡪 Major components interfaces
* Management of GPC internal resources, external interfaces 🡪 FCOS

1. Flight Computer Operating System :

* GPC’s internal resources were controlled and managed by FCOS along with external interfaces
* FCOS handles three main responsibilities:

1. Process Management: System requests and as a response, the allocation of resources is controlled by process management. It validates that the CPU resources are allocated to the task which is ready and has highest priority. A standard set of service interfaces (SVC) are made with reference to specifications that define time, or event ,priorities and frequency to handle a request.
2. I/O Management : Controls the management of I/O processor resources (IOP). Each GPC containing IOP has MSC (Master Sequence Controller) and 24 Bus control Elements (BCEs) where asynochronus communication takes place between a processor and its next higher/lower priority processor at the time of I/O operations.
3. DPS Configuration Management : Hardware initialization, status checks and other IOP states as well as GPC controlling is done here. Performs the transfer of program code to main memory of GPC from mass memory. It implements the overlay of program as well as modifies the contents of mass memory.
4. System Control : System control impersonates a wide range of tasks such as Configure the associated avionic data network & control DPS. SPECS also performs functions like:
5. Resetting the MTU
6. Initiating memory dump
7. Changing and examining specific core location in memory.
8. Changing figuration of DPS
9. User Interface : It accelerates external control of systems or applications. Its three major functions are enunciated below:
10. Command input processing
11. Input processing
12. Operations control and output message processing

The user interfaces supported are:

1. Keyboard and CRT
2. Launch Data Bus (LDB) to communicate with launch processing system
3. Network signal Processing (NSP) to process the data and commands received from Mission Control Center at Johnson Space Station.
4. Application Software : The three applications of application software are guidance, navigation and control (GN & C). GN&C software computes the position , velocity and altitude as well as manages sensor redundancy and controls the avionics subsystems and issues the engine and affector commands for mission takeoff to landing.   
    1. Guidance, navigation & control :

* In this cyclic closed loop application, stringent timing and phasing relationship is followed by GN & C.
* There are 1-10 major modes included in six OPS and there are 10 SPECs and some are present in one or more of OPS
* The initiation, termination and sequencing of each OPS, major mode and SPEC are handled by the control segment.
* The principal functions are executed at 25Hz for basic vehicle system and display update at 0.25Hz

1. System Management :

* The system performance and the configuration of orbiter are monitored by System Management.
* The alert signals are generated and the crew is informed about them when the abrupt and anomalous conditions are detected while monitoring payload subsystems.

3. Vehicle Checkout :

* It handles the initialization and checkout of avionics system under the control of ground or flight crews
* It contains a special feature known as TCS (Test Control Supervisor). A maximum of three TCS commands can be executed at the same time.
* The configuration of associated I/O interfaces and primary functions is done into three ground checkout OPS and one in-flight checkout OPS.

**The advantages of Frequency Executive are:**

1. The frequency executive follows a cyclic architecture to ensure the synchronization and task execution in a defined time frame. This ensures that the execution of the tasks doesn’t get halted on one particular task which might be facing runtime error.
2. Due to a fixed order of execution following a stringent time frame, jitter dets dwindled considerably
3. The application processes can maintain synchronization by implementing redundancy management system.
4. The schedule used is predictable like fixed priority scheduling algorithm unlike using preemption. Therefore, lower priority tasks doesn’t get masked by higher priority ones.

**The disadvantages of Frequency Executive are:**

1. Race conditions can occur during the initialization of new process since it might cause overruns for the processes at process and system levels. This happens due to asynchronization between I/O operations and intercommunications between the GPC.
2. Frequency scheduling follows non-preemptive mechanism i.e a low priority task when executing can block a much higher priority task.
3. In frequency executive method, all the possibly currently executing tasks needs analysis at the same time. Each thread is provided with dynamic or fixed scheduling priority for execution in real time scheduling.
4. There are preordained timing blocks in the cycle loop depending on mission requirements known as “Orbit Insertion Burn Stage” which has a window of 5 minutes. However, for many processes it will be difficult to be divided into blocks with 5 minute window. Therefore, creating an ideal frequency executive is difficult.

5. [30 points] Provide 3 constraints that are made on the RM LUB derivation and 3 assumptions as documented in the Liu and Layland paper and in Chapter 3 of the text. Finally, list 3 key derivation steps in the RM LUB derivation that you either do not understand or that you would consider “tricky” math. Attempt to describe the rationale for those steps as best you can do based upon reading in Chapter 3 of the text.

**SOLUTION** :

The 3 assumptions documented in Liu Layland paper are:

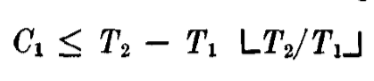
1. The requests for all the tasks with hard deadlines are periodic in nature and there exists a constant rate of request.
2. The tasks are mutually exclusive and independent of each other i.e. the initiation and completion of a particular task do not depend on other tasks.
3. Run-time for a particular task is constant for that task and doesn’t vary with time. Where, run-time is the total time a processor takes to perform a task without any interruption.

The constraints made on RM LUB derivation are:

1. The time period of the task is equal to the deadline of the task. Therefore, it is mandatory that the task should be provided response before its next occurrence. This will simplify the mathematical calculations. There can be placed some buffering hardware for each peripheral function to complete the task before next occurrence.
2. Scheduling policies such as preemptive, fixed-priority and run-to-completion scheduling policy must be used. This is because rate monotonic scheduling uses pre-emptive or fixed priority tasks only. Therefore, non-preemptive and dynamic tasks are not permitted.
3. Context switching refers to the time taken by the CPU to switch between lower priority task to higher priority task on its arrival. The derivation of RM LUB can become complicated if this switching time is taken into consideration for its derivation. Therefore, to prevent complexity, this time is overlooked.
4. The CPU utilization factor is U = m (2^(1/m) -1). This equation is valid only when Ti/Tj<=2 i.e the ratio of any 2 of the tasks should be less than 2.

The three key derivation steps in RM LUB derivation which I found tricky were:

1. In Liu Layland paper, the Theorem 3 of Achievable Processor Utilization states that for a set of two tasks with fixed priority assignment, the least upper bound to processor utilization factor is U = 2 x (2^(1/2) -1). While explaining the case of run-time C1 short enough condition, I didn’t understand how did we get the equation:



However, speculating through the below two-service diagram taken from RTECS, I was able to visualize the concept. From, that, I could infer the further calculations done to evaluate the value of C2.

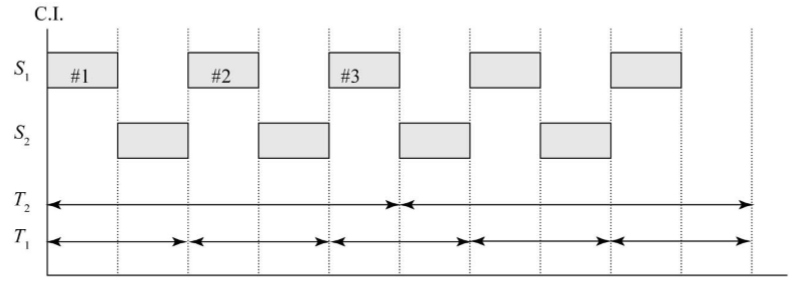
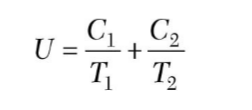
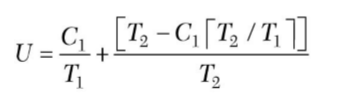


Figure : Two service example to derive RM LUB

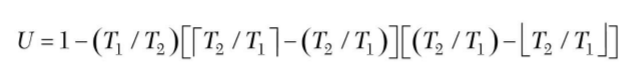
1. The generalized case of the utility can be obtained from the following equation



On substituting the value of C2 in the above equation, we get,



Further simplifying the equation and evaluating it in terms of T2 and T1 we get,



The fractional interference f and the Interference I is given as





The least upper bound for utility can be calculated when its value is reduced. However, its obscure to understand that to reduce the utility, when I needs to be decreased and at the same it, we weren’t able to figure out the reason why I was substituted to 1.

1. Consider the evaluated utility equation:



Here when T2 > T1, U increases monotonically with C1

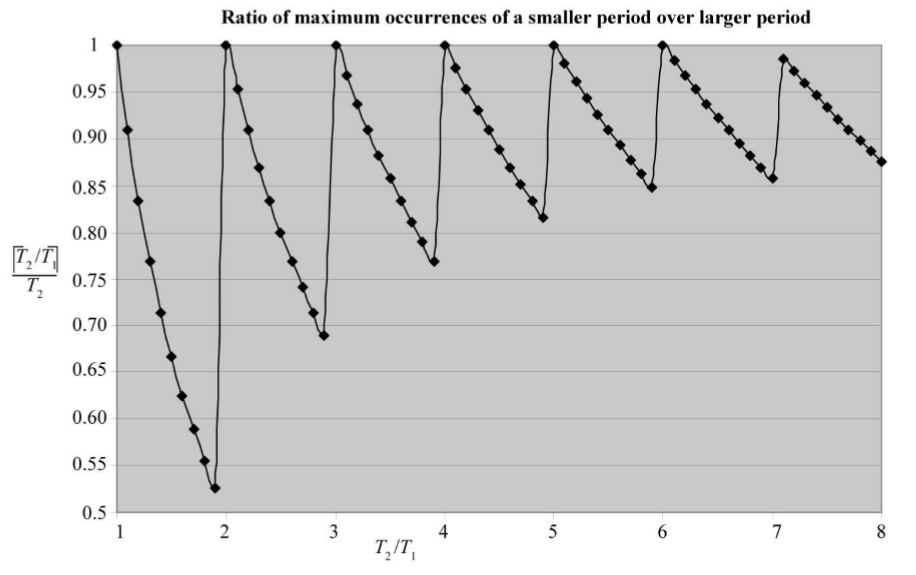


Figure : Relationship for T2 and T1 for Case 2.

The term (1/T2) [T2 / T1] will be always smaller than (1/T2) since T1 will be always smaller than T2. To simplify the calculation, there is an assumption made that T1=1 and T2 is given value which is greater than 1 till infinity. However, I couldn’t decipher why T1 is considered as 1 and not some other value. If that is understood, then we can comprehend the flow of derivation and it won’t be obscure.