**Operating System:**

Operating system (OS) selection is a pivotal aspect of the satellite development and

operational environments. Embedded programming is split in two by the consideration of solution's implementation: with and without Real Time Operating System. When the application becomes very complex, when we have a lot of tasks that need to run very strictly and when we have a lot of events that may come asynchronously from outside, the logic is hard to accomplish. Here comes in scene the second category of embedded applications: using a real time operating system (RTOS).We considered the these operating systems:

**Linux:**

Linux is a flexible Unix–like operating system that can run on a host of platforms from small embedded computers (1–2 MB with no memory management unit) to super computers.

We considered linux for many reasons that any Embedded project benefits from open source code. Linux provided a platform that allowed the operational software to be built and tested on the specific test environment and any generic Linux-based workstation. This supported a distributed development environment that required a minimum overhead to set up and support. It also gave the developers a common set of development tools and a vast working, documented, and tested code base that could be harvested over the Internet for free.

Another advantage was that several key device drivers were already written for AX.25

(packet protocol) .

Linux also provides a host of tools that were utilized in the operational satellite without

burdening the software development team with coding, testing, and maintenance overhead.

These include free utilities that report memory use, disk space utilization, software status, and date/time manipulation. The utilities were easily integrated into a set of diagnostic

scripts that were used to report status to the contact software. Other useful utilities included

compression (bzip2), data integrity checking (md5sum), and shell utilities like cat, grep,

bash, etc.

The major disadvantages of using an OS like Linux are that

(a) The great flexibility requires much more testing (there can be a high number of corner cases), and

(b) There is limited time-tagging accuracy .

(c) The problem is that a kernel like Linux requires the support of many files that work together and must be present in a valid state. With such a small team never had the time or developed the code necessary to validate all of the Linux files necessary to operate the payload software. It should be pointed out that Linux tools exist to solve this problem; one only needs time.

**Salvo:**

Salvo is a proven, powerful, high-performance and royalty-free real-time operating system (RTOS) that requires very little program and data memory, and no task stacks. It is an easy-to-use software tool to help you quickly create powerful, reliable and sophisticated applications (programs) for embedded systems. Salvo was designed from the ground up for use in microprocessors and microcontrollers with severely limited resources, and will typically require from 5 to 100 times less memory than other RTOSes. In fact, Salvo's memory requirements are so minimal that it will run where no other RTOS can. Salvo is ROMable, easily scaleable and extremely portable. It runs

on just about any processor, from a PIC to a Pentium.

Salvo is a purely event-driven cooperative multitasking RTOS, with full support for event and timer services. Multitasking is priority- based, with sixteen separate priority levels supported. Tasks that share the same priority will execute in a round-robin fashion. Salvo provides services for employing semaphores (binary and counting), messages, message queues and event flags for intertask communications and resource management. A full complement of RTOS functions (e.g. context-switch, stop a task, wait on a semaphore, etc.) is supported. Timer functions, including delays, timeouts

and cyclic timers, are also supported. The Salvo RTOS by Pumpkin is a lightweightsystem designed for use on small embedded systems.Because the PIC24H, the considered microarchitecture, has hardware limitations that make it difficult to use an RTOS, Salvo was a perfect fit for the CDH hardware. It allows for a cooperative multitasking environment, which greatly simplifies the software organization. Each major subsystem (thermal, power, commands, sensors, and communications) has one or more tasks to handle specific functionality for the system, which allows the system to be modularized and easily debugged.