A Robust Command, Communications and Data Acquisition System For Autonomous Sensor Platforms Using The Data Transport Network

Todd Valentic

Center for GeoSpace Studies
SRI International
Menlo Park CA 94025

todd.valentic@sri.com

Need

A reliable, low-power, computer system for collecting and transmitting data from multiple sensors at remote sites that are operated autonomously for long periods of time, supporting the National Science Foundation's Arctic Research Support and Logistics Services program.

Approach

A low-power, temperature rated single board computer (SBC) is selected, upon which we run a version of the Linux operating system tuned for embedded systems. The application layer uses the Data Transport Network as a framework to organize the collection of programs used to collect data from the instruments. A set of software services provide for resource management and data transmission over the Iridium satellite constellation. A number of fail safes are implemented to allow the system to recover from unexpected situations.

Data Transport Network

The Data Transport Network is a system for designing robust field instrumentation that integrates the collection of scientific data, system health monitoring, data processing and distribution of real-time results over unstable and bandwidth limited networks. The system is built around a set of message servers that provide a store and forward mechanism for buffering data, a publish and subscribe interface for accessing the data feeds and a software framework for coordinating the programs in the system. It has been in operation at field sites throughout the Arctic since 1999.

More Information

datatransport.org
transport.sri.com/projects/tincan
transport.sri.com/rudics
busybox.net
buildroot.org

Acknowledgments

National Science Foundation's Arctic Research Support and Logistics Services program.

Fall AGU 2012 - C13E-0669

1 Hardware Stack

We use the TS-7260 single board computer (SBC) from Technologic Systems. It is temperature rated to -40C and is capable of running a standard Linux kernel.

- 200MHz ARM9 CPU
- PC104 expansion bus
- 128MB SDRAM
- 128MB NAND Flash
- 2 USB ports
- 30 DIO lines
- 2 12-bit ADC
- Watchdog timer
- Ethernet
- 3 serial ports

The total power consumption is 1W, which can be reduced to 0.25W when idle. A custom power control board is used to turn instruments off to conserve power and includes signal conditioning to monitor power usage.

2 Tincan Linux

We roll our own embedded Linux distribution. At the core is the 2.6.34 kernel, patched to support the TS-7260 SBC. Most of the common Unix commands are provided by Busybox and the lightweight uClibc C library. A typical system, including the Python programming language, has a footprint of 40MB.

The entire build process is controlled by Buildroot:

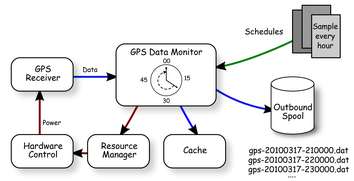
- Makefile based
- Builds the cross-compiler tool chain
- Package selection
- Custom local packages
- Creates root file system

The entire system runs out of the on board flash memory, which uses YAFFS2 to prevent data loss during power outages and provides wear-leveling.

3 Application Structure

The Data Transport framework is used to organize a large collection of programs that make up the application layer. In the Unix tradition, there is one program for each task. The programs fall into two major categories: data monitors and network services.

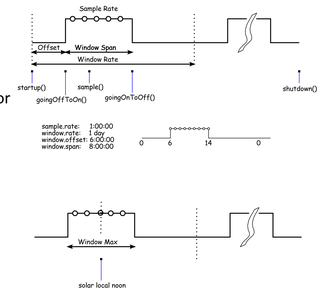
The data monitors are programs that interface to the sensors and collect data. Adding a new sensor into the system only requires writing a small data collection program that slots into the framework. The scheduling of when the instrument turns on, how often the data are sampled and control of the hardware resources it needs are all handled automatically. This approach provides a means of splitting up a problem into small, easily implemented pieces that still function together in a consistent and integrated manner. More over, if one of the programs crashes, the rest of the system still functions.

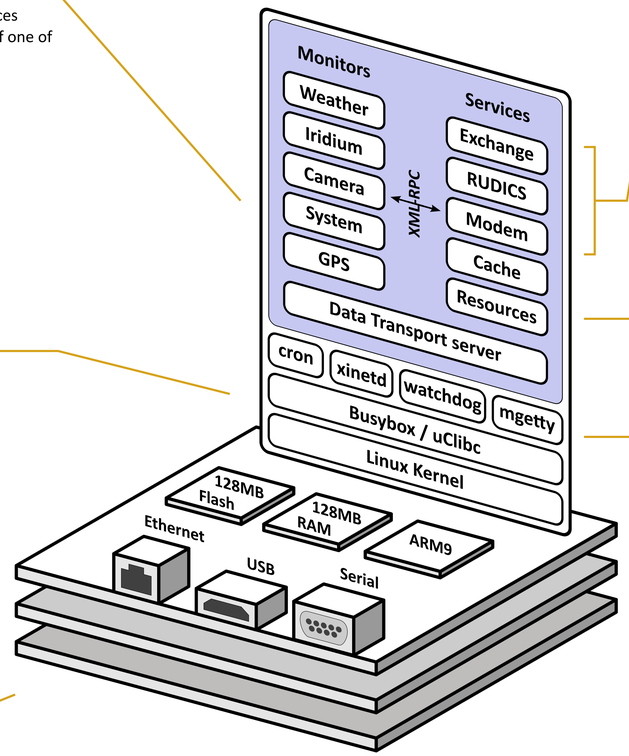


4 Scheduling

Each monitor has a set of schedules that determine when it runs. A simple schedule has a sample window within which data are periodically collected. Multiple schedules are supported, ordered by a priority value, allowing for seasonal or campaign overrides.

More sophisticated schedules are supported, such as sampling around solar local noon. Instruments, such as cameras, only collect data during the day. The solar angle is computed from the GPS position and dynamically fed into the schedules.

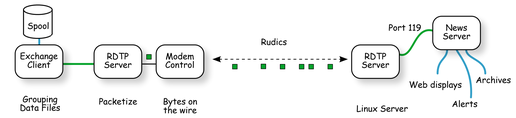




5 Iridium RUDICS File Transfers

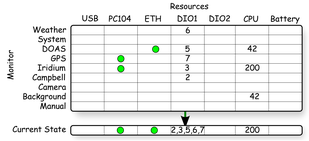
Using a store-and-forward approach, the data files from the instruments are queued in the SBC's internal flash memory and offloaded over an Iridium satellite link using the RUDICS data service. The data transfer is managed by a set of programs in the framework, packetizing the data files and posting them into the Data Transport Network for delivery to the server. The communications programs handle the idiosyncrasies of the Iridium link, resuming the data transmissions when the link drops and prioritizing traffic. Messages can also be sent to the site from the server, allowing for code and schedule changes.

The data files are posted into a central server at the other end of the Iridium RUDICS link. Incoming data files are posted into message queues, to which programs can subscribe. These programs handle the data processing, web displays and file archiving on server. In a manner similar to the on-board software, the Data Transport Network provides the framework for organizing these various programs and message queues.



6 Resource Management

The resource manager service is the key to optimizing the low power performance of the system. It maintains a scoreboard of the global system state, tracking which devices are powered on. When ever a program needs a resource, a request is placed to the resource manager. The resource (an IO pin, the Ethernet port, etc.) will be enabled. When the program is finished, it releases the resource. If another program is still using the resource, it will remain allocated until no other programs are using it. The benefit of this approach is that each client program remains independent of each other, yet the resource usage is fully integrated.



7 Failsafes

Recovering from unexpected problems is necessary for long-term unattended operations. We use a layered approach:

- Kernel parameters set to reboot on panics and oopses
- Hardware watchdog reboots if not reset in 8 seconds
- Software watchdog (via cron) reboots if data transfers stop
- Periodically power on modem for listen only
- PPP dial in server

A flag can be sent to the system instructing it to keep the Iridium modem powered on, at which point you can manually log in to the system.

8 Deployments

- OBuoy
- IceLander
- IceGoat
- IceKid
- Weather stations
- Remote cameras
- Radar controllers
- Power monitoring
- Flux towers