**İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY**

# ITU-CSCRS GROUND RECEIVING STATION AUTOMATION & RENOVATION

**Master Thesis by**

**Göker Burak ÇETİN, Eng.**

**Department : Geodesy & Photogrammetry Engineering**

**Programme: Geomatics Engineering**

**SEPTEMBER 2007**

**ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY**

# ITU-CSCRS GROUND RECEIVING STATION AUTOMATION & RENOVATION

**Master Thesis by**

**Göker Burak ÇETİN, Eng.**

**501031614**

**Date of submission : 4 September 2007**

**Date of defence examination: 3 September 2007**

**Supervisor (Chairman) : Prof. Dr. Filiz SUNAR**

**Members of the Examining Committee : Prof. Dr. Derya MAKTAV (ITU)**

**Assist. Prof. Dr. Berk ÜSTÜNDAĞ (ITU)**

**SEPTEMBER 2007**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# İTÜ-UHUZAM UYDU YER İSTASYONU OTOMASYON VE RENOVASYONU

**Master Tezi**

**Müh. Göker Burak ÇETİN**

**501031614**

**Tezin Enstitüye Verildiği Tarih : 4 Eylül 2007**

**Tezi Savunulduğu Tarih : 3 Eylül 2007**

**Tez Danışmanı : Prof. Dr. Filiz SUNAR**

**Diğer Jüri Üyeleri : Prof. Dr. Derya MAKTAV (İ.T.Ü.)**

**Yard. Doç. Dr. Berk ÜSTÜNDAĞ (İ.T.Ü.)**

**EYLÜL 2007**

## FOREWORD

I have tried to solidate and provide this documentation related to satellite ground receiving station in a more methodical way after four years of labor and experience gained as a system administrator at ITU-CSCRS Ground Station. From the day I begin working at the ground station I observed some of the obstacles in the system. Now after 4 years with the aging hardware and software systems, it is now more obvious that the ground station needs a renovation project to carry on its operation. Therefore without constant renovation a ground station, could have never achieve enough service quality, because of the constant increase of demands of the remote sensing data market at the end user side.

I believe, with that document, I'm providing a start point to renovate the ground receiving station that should be undertaken in the future. I'm very grateful to Prof. Filiz SUNAR for her support and encouragements for this thesis since this was a part of my job being responsible.

I hope in the future the role of such Ground Station gains ground in Turkey and notified so it deserves its rightful place among the first 10 ground stations around the world, attracting many valuable scientists either from Turkey and abroad.

I'm thankful to all my colleagues working and in a qualified sense bearing with me from the beginning.

06, 2007 Göker Burak ÇETİN

**CONTENTS**

**ABBREVIATIONS** ..... **vi**
**FIGURE LIST** ..... **viii**
**ÖZET** ..... **ix**
**SUMMARY** ..... **x**
**1. INTRODUCTION** ..... **1**
**2. WHAT IS GROUND STATION?** ..... **1**
**3. ITU - CSCRS & GROUND RECEIVING STATION (SAGRES)** ..... **2**
**4. NECESSITY FOR THE RENOVATION OF CSCRS GROUND STATION** ..... **3**
**5. STATION HARDWARE** ..... **4**
5.1. Control Systems ..... 4
5.1.1. Current Status ..... 4
5.1.2. Proposed Implementation ..... 5
5.1.2.1. Control Servers ..... 6
5.1.2.2. Other Control Systems ..... 7
5.2. Recording Systems ..... 7
5.2.1. Current Status ..... 7
5.2.2. Proposed Implementation ..... 8
5.3. Processing Systems ..... 12
5.3.1. Current Status ..... 12
5.3.2. Proposed Implementation ..... 14
5.4. Data Storage & Archiving Systems ..... 17
5.4.1. Current Status ..... 18
5.4.2. Proposed Implementation ..... 18
5.5. Demodulator Systems ..... 20
5.5.1. Current Status ..... 20
5.5.2. Proposed Implementation ..... 21

**6. STATION SOFTWARE** ..... **22**
6.1. Database Systems ..... 22
6.1.1. Current Status ..... 22
6.1.2. Proposed Implementation ..... 22
6.2. System Softwares ..... 24
6.3. Application Softwares ..... 25
6.3.1. Programming Languages ..... 25
6.3.2. CSCRS Management Middleware ..... 26
6.3.3. Interface Definition Languages ..... 29
**7. RESULTS & RECOMMENDATIONS** ..... **32**
**REFERENCES** ..... **33**
**CURRICULUM VITAE** ..... **35**

**ABBREVIATIONS**

**4GL** : Fourth-Generation Programming Language
**ACU** : Antenna Control Unit
**AMD** : American Microelectronic Devices
**API** : Application Programming Interface
**CGI** : Common Gateway Interface
**CPU** : Central Processing Unit
**CSS** : Cascading Stylesheets
**DART** : Datron Archive Reversible Tape Format
**DDB** : Distribution Data Buffer
**DHTML** : Dynamic Hypertext Markup Language
**DIS** : Direct Ingestion System
**DLT** : Digital Linear Tape
**DOM** : Document Object Model
**ECL** : Emitter Coupled Logic
**FIFO** : First In First Out
**GIS** : Geographic Information System
**GML** : Geography Markup Language
**GPS** : Global Positioning System
**GSC** : Ground Station Controller
**GUI** : Graphical User Interface
**HDD** : Hard Drive Disk
**HPC** : High Performance Computing
**ISCSI** : Internet SCSI
**LTO** : Linear Tape-Open
**MWD** : Moving Window Display
**NCI** : Network Command Interface
**OGC** : Open Geospatial Consortium
**OLAP** : On Line Analytical Processing
**PDB** : Processed Data Buffer
**RAID** : Redundant Array of Inexpensive Disks
**RISC** : Reduced Instruction Set Computer
**SAN** : Storage Area Network
**SATA** : Serial Advanced Technology Attachment
**SCSI** : Small Computer System Interface
**SFS** : Simple Features Specification
**SGI** : Silicon Graphics International
**SQL** : Structured Query Language
**TLE** : Two Line Element
**UDB** : Unprocessed Data Buffer
**UDP** : User Datagram Protocol
**URL** : Uniform Resource Locator
**WMS** : Web Map Server

**WFS** **:** Web Feature Server
**XML** **:** Extensible Markup Language
**XSL** **:** XML Styling Language
**XUL** : XML User Interface Markup Language

## FIGURE LIST


**Figure 1** : Principle Segments of Remote Sensing ..... 1
**Figure 2** : ITU-CSCRS ground station coverage ..... 2
**Figure 3** : Ground Station Resource Distribution ..... 4
**Figure 4** : Ground Station Controller Software Interface ..... 5
**Figure 5** : Node Status Diagram ..... 6
**Figure 6** : Single Capture Topology ..... 10
**Figure 7** : Failsafe Capture Topology ..... 11
**Figure 8** : Dual Capture Topology ..... 11
**Figure 9** : Current Processing System ..... 13
**Figure 10** : SGI CCNuma Architecture ..... 14
**Figure 11** : AMD x86_64 Architecture ..... 15
**Figure 12** : New Four Node Processing Server Setup ..... 16
**Figure 13** : Product Levels Planned in ITU-CSCRS Ground Station ..... 17
**Figure 14** : ITU-CSCRS Unique Volume Management And Partitions ..... 18
**Figure 15** : ISCSI Disk Arrays Employed in ITU-CSCRS Ground Station ..... 19
**Figure 16** : New Tape Unit Employed on ITU-CSCRS Ground Station ..... 20
**Figure 17** : One of the demodulator systems of ITU-CSCRS Ground Station 21
**Figure 18** : In-Snec Software Based Demodulator System ..... 21
**Figure 19** : Proposed Ground Station Topology ..... 27
**Figure 20** : CSCRS Object Based Management Console ..... 28
**Figure 21** : Web Based Pass Schedule Interface ..... 28
**Figure 22** : Some Application Interfaces ..... 30

**ITU-UHUZAM UYDU YER İSTASYONU RENOVASYON VE OTOMASYONU**

## ÖZET

Bu çalışmada, İTÜ bünyesinde bulunan uzaktan algılama merkezine ait uydu yer istasyonunun, sistem altyapısında ve gündelik operasyonlarında karşılaşılan yada karşılaşılması muhtemel olan sorunlarına değinilmiş, bu sorunların çözümlenebilmesi için uygulanan yada uygulanacak olan çözüm yöntemlerinden bahsedilmiştir. Sistem, donanım ve yazılım açısından derinlemesine incelenerek, problem görülen noktalarda olası çözümlerin geliştirilmesine çalışılmıştır. Çalışma, istasyon bünyesinde acil olarak ihtiyaç duyulan noktalarda sistem otomasyonunun optimal derecede uygulanabilmesi için gerekli yazılım altyapısı tasarımını içermektedir. Çalışma, uydu yer istasyonuna ilişkin geliştirilmeye çalışılan donanım ve yazılım politikalarının bir sonucu olup, istasyona ilişkin teknik problemlerin kurumsal anlamda ele alınması konusunda atılan bir adımdır. Çalışma aynı zamanda, yazılımsal ve donanımsal olarak istasyon bileşenlerinin güncel teknolojik standartlara getirilmesi amacı ile başlatılan renovasyon projesinin bir belgelemesi niteliğini de taşımaktadır. Çalışma kapsamında geliştirilen yazılımların, gelecekte uydu yer istasyonunda oluşturulmak istenen tam otomasyon sistemine bir altlık olması hedeflenmiştir. Çalışma içerisinde bahsi geçen tüm yazılım bileşenleri bu bağlamda ele alınmış ve tasarlanmıştır. Tasarlanan sistemde, tam otomasyon işlevselliği yanında artı bir değer olarak istasyonun veri, donanım, yazılım ve insan kaynakları hakkında otomatik olarak istatistik veri toplanması da hedeflenmiştir. Renovasyon projesinin nihai hedeflerinden biri olarak, toplanan yüksek doğruluklu bu istatistik verinin ilgili karar merciilerine ulaştırılması hedeflenmektedir. Bu hedefe ulaşmak için tasarlanan yazılım altyapısı da çalışma içerisinde anlatılmıştır.

**ITU-CSCRS GROUND RECEIVING STATION AUTOMATION & RENOVATION**

## SUMMARY

In this study, applied or possible solutions for daily encountered operational or infrastructural problems in ITU-CSCRS Ground Station was investigated. The system hardware and software was thoroughly investigated, and possible solutions were proposed in the problematic areas. A software infrastructure design for an optimal operation of ground station automation in emergent points was discussed. This study is a part of the result of a going on process of designating institutional policy for station hardware and software system and therefore this should be seen as a step through an institutional approach for handling the technical problems of the station. At the same time, this study can be seen as a documentation effort for the renovation project which had been started to catch up-to-date technological standards for the station hardware and software systems. From the very beginning, the softwares developed were engineered to provide a robust foundation for the full automated ground station operations. With the system designed, it was aimed to collect statistical information about the data, software, hardware, and human resources automatically as a surplus value. As a definitive goal of the renovation project, providing this accurate information to the decision makers was aimed. The software system designed for that purpose was also mentioned in this study.

## 1. Introduction

Turkey's one of the biggest satellite ground stations has been established in ITU Maslak campus in 1997. As the system administrator of that facility since 2003 I've been involved exclusively in the ground station automation process needed for future operations, and all the possible solutions for daily encountered problems in a typical ground station. Therefore in this study, four years of gained experience in the fields of station hardware, software, and human resources were explained in detail.

## 2. What is Ground Station?

An earth station or ground station is the surface-based (terrestrial) end of a communications link to an object in outer space [1]. Where the communications link is used mainly to carry telemetry or must follow a satellite not in geostationary orbit, the earth station is often referred to as a **tracking station**. In Remote sensing it is one of the three essential segment of the system to receive, archive, and process the telemetry data from the remote sensing satellites [fig.1].

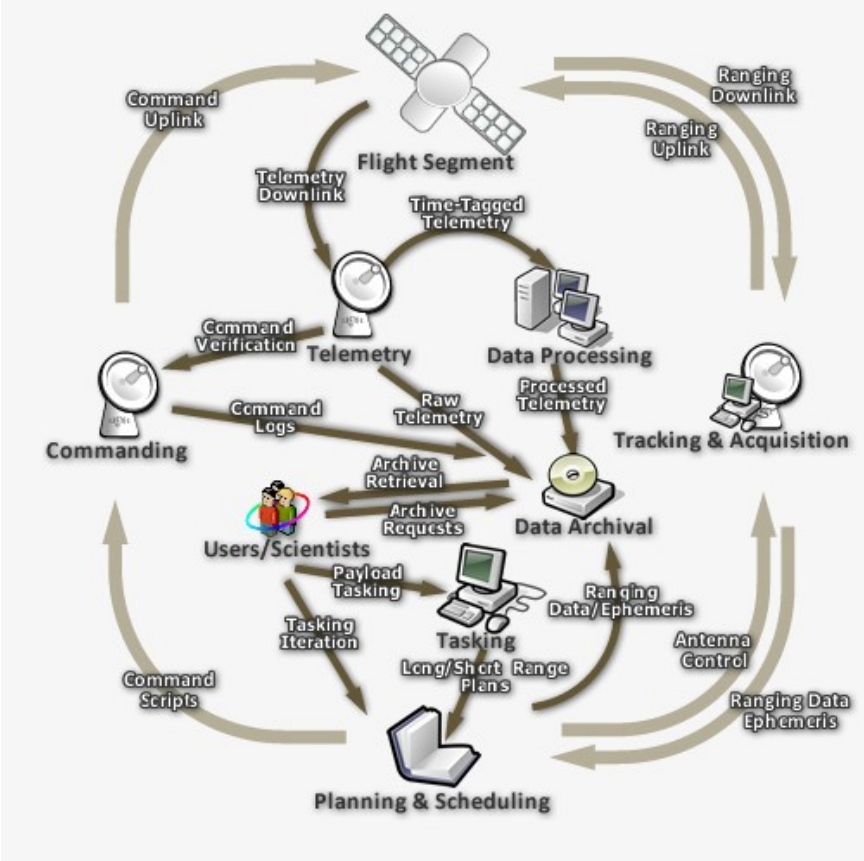


Figure 1. Principle segments of remote sensing.

## 3. ITU - CSCRS & Ground Receiving Station (SAGRES)

Istanbul Technical University - Center for Satellite Communications and Remote Sensing (ITU-CSCRS) is one of the forecoming institutions around the world with a highly capable ground receiving station unit. It is the first center established in Turkey to conduct application oriented projects in remote sensing and satellite communications technologies and serve national/international civil/military companies in their research, development, and educational activities. After successful design, assemble and test stages of the receiving station through the years 1996-2000, ITU-CSCRS was established for operational working under the name ITU-SAGRES (SAtellite Ground REceiving Station) in 2000 as a wide range communications and remote sensing integrated system. In the second half of the year 2003, it was restructured into ITU-CSCRS. ITU-CSCRS has the capabilities of acquiring images from remote sensing satellites, processing data, and sending the products via satellite links to national and international users. The station can receive images of the Earth's surface within a radius of 3000 km., which covers from Sweden to Sudan, and England to Kazakhstan [fig.2].

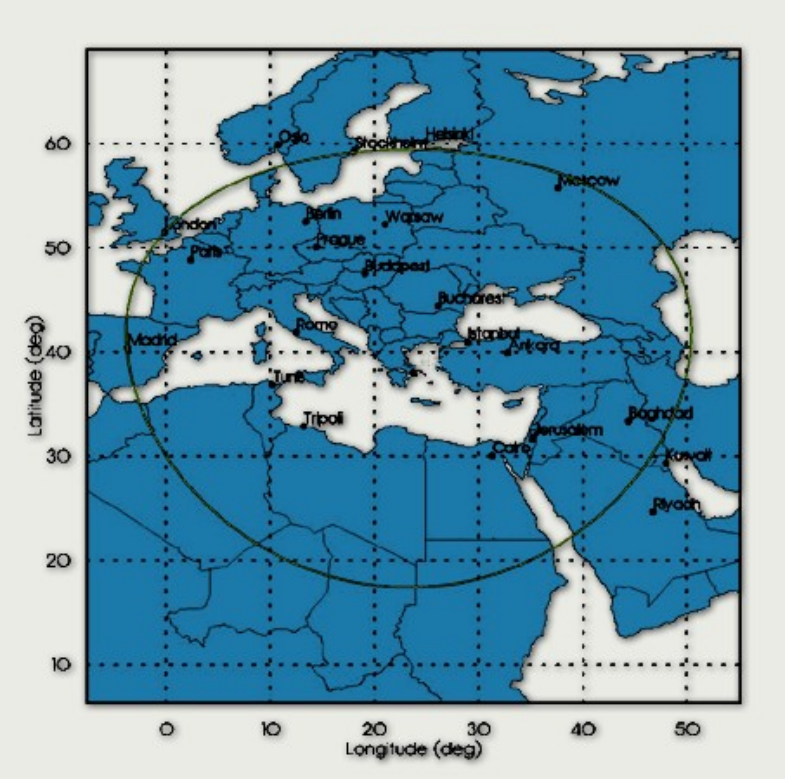


Figure 2. ITU-CSCRS ground station coverage

In the ground station, the data acquired from SPOT-2, SPOT-4, RADARSAT-1, ERS-2, NOAA, and METEOSAT satellites are archived, formatted and processed with the high technology. These successful studies were certificated with Operational and Product Certificate by the Radarsat Inc., Canada in November 2002.

**4. Necessity for the Renovation of CSCRS Ground Station**

Although ITU-CSCRS is one of the 30 ground stations around the world successfully established and operated for many years, it is evitable that some developments need to be done to respond increasing end user demands in the satellite market either in the academical community or in the private sector. Within the years, because of that increase in demands from the market and the aging hardware and software systems used in the ground station, it was taken into account that the renovation work has to be done in a 18 months of period in the software and hardware fields. Within this project, it was determined some basic principles without tempering current operation and modular design of the ground station which will be able to support current and future satellite missions flawlessly and create an automation system can provide a complete API for the whole ground station operations. This was also a necessity, for being able to flexibly adapt the station operations to the changes in satellite operations in years or project based setups.

Those principles can be numbered such as supporting up to 1 Gbps transfer and recording speeds either in hardware and software. Supporting full automation from the downlinking of telemetry data to data distribution to end users, contains fully automated routines for scheduling reception, archiving telemetry data, product processing, data quality assurance, and semi-automated routines. Such automation can reduce the operator load as much as possible for daily ground station operations such as reporting, cloud coverage assessment, etc. For those reasons it was investigated current station hardware and software systems thoroughly and with the help of know-how gained, some sub systems mainly on the software side were developed and/or being developed.

It can be said, this was the key issue in ground station automation, balancing the work load between software and hardware and minimizing the human efforts [fig.3].


Figure 3. Ground station hardware, software, and human resource distribution

## 5. Station Hardware

One of the main bottlenecks for reaching such a goal is the some units of the current hardware setup which strictly tied to the integrator company determined basic operational structure of the hardware that are not allowing to much intervention to its inner workings. The current hardware system can be subgrouped as processing units, telemetry recording units, control systems, Demodulators/Decryptors/Receivers, and data storage systems.

### 5.1 Control Systems

#### 5.1.1 Current Status

One part of current control system is implemented as a standalone software running on a IRIX processing system and supported by a central oracle database backend. That software is responsible for the automation scheduling and data processing tasks that can be managed by simple interface written by integrator company. The other part is is the part of the ground station controller software which is mainly responsible for tracking system automation and scheduling via its own database backend and is also responsible for status of the nodes of the ground station network [fig.4]. That status determination is carried via two different protocols. One of them is conducted via IEEE488 interface which contains demodulators, receivers, signal generators, and serial switch devices. The other one conducted via standard Ethernet interface for GSC (ground station controller), and recording units which all contain a piece of software running as a daemon on Solaris operating system, is responsible for answering specific query codes implemented as plain text protocol works over UDP on port 3069 called NCI interface.

Tracking system control is also on that software, commanding another unit named ACU (Antenna Control Unit) and positioned in the shelter unit of the ground

station. This is a specialized computer system responsible for calculating angle data from the TLE information sent by GSC, and GPS location and time information, needed for antenna operation during satellite tracking.

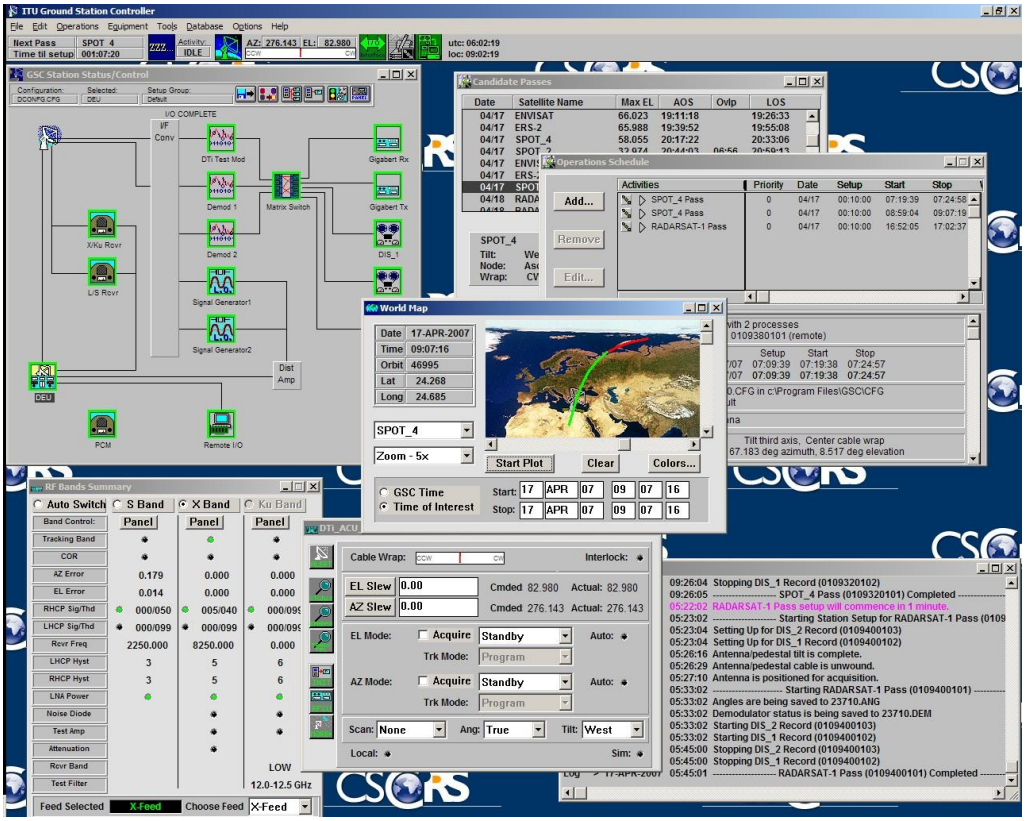


Figure 4. Ground station controller software interface

### 5.1.2 Proposed Implementation

It is proposed that, specifically fragmented health control mechanism should be centralized and made be specialized for easy maintainability, easy deployment of new network nodes, and redundancy requirements needed for its critical role of being responsible for ground station hardware health and ground station operations in general. In the current status it is not very convenient to deploy and replicate all control mechanism on an another computer system. For that reason a control mechanism proposed consists of a standard PC hardware running Linux operating system and a special software developed by ITU-CSCRS R&D department which is capable of understanding both IEEE 488 commands, NCI commands, and a simple control mechanism being developed for the new nodes to be deployed. That program is designed to be able to handle all those protocols. The integration with general station middleware is also planned for the scripting needs.

### 5.1.2.1 Control Servers

The computer systems, which will be responsible for the status of the station network nodes called control servers, will handle node status in 5 different phases as shown in figure 5 [2].

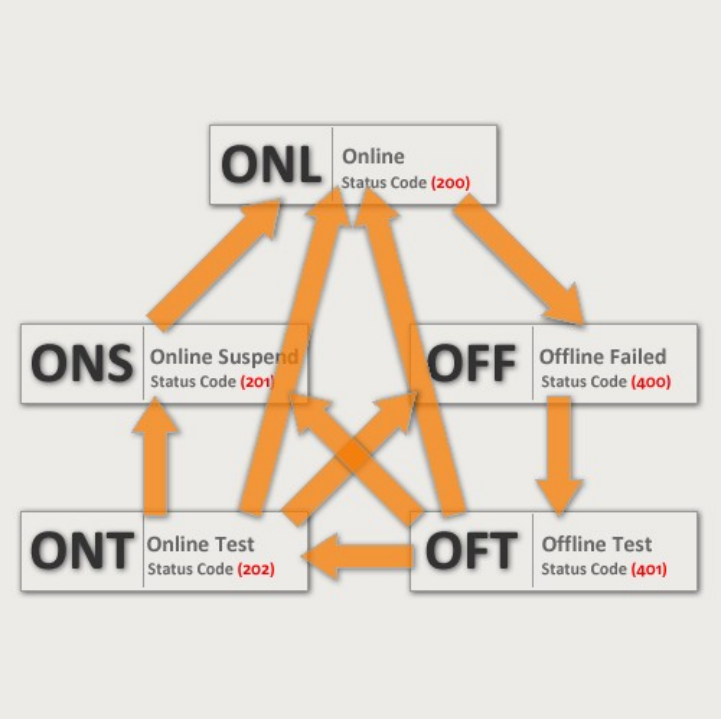


Figure 5. Node status diagram

The “Online” phase which is also called “ONL” means the node is OK and is ready for operation. The status code for the said node in that case is “200”. The “Offline Failed” phase is also called “OFF”. In that case the hardware of the node is not working or control mechanism allocated to the node is not responding. In any case the node is not operational. Status code is determined as “400” for that phase. In the “Offline Test” phase some checks are done by control server and if tests give positive results then status code of the said node is elevated to “200”. In other case it is necessary that the node is rebooted. In that phase the status code for the node is elevated to “401”. After the node is rebooted and, if the post boot checks give positive results, it is checked whether it is critical to elevate the node online for station operation in the current scenerio. Then “ONT” phase is bypassed and its status code is elevated to “200”. In the other case its status code is elevated to “202” and post boot checks are conducted. If the node is not critical in the current

operational scenario for the ground station, its status code elevated to “201” which is called “Online Suspend” or “ONS”. When the node needed in the operation, its status code is elevated to “200” again.

#### 5.1.2.2 Other Control Systems

Even it is in its early development stages, in the future, it is also planned replacing ACU protocol interface, which is normally part of the GSC application, as a separate tool that can be integrated into new automation system for antenna control.

## 5.2 Recording Systems

### 5.2.1 Current Status

In the current status, recording mechanism consists of three independent and differently configured recording units that employ software system which running atop Solaris operating system and specifically designed for that job. The recording units are called DIS (Direct Ingestion System) and consists of 5 pieces of software which runs as daemons on the computer. First and the most important one is called ingest daemon which is responsible for recording captured telemetry data to the HDD arrays found on the units. Those HDD units are working together in the RAID0 mode providing necessary speed for recording. That RAID volume is formatted with a very specialized file system which provides near RAW HDD access performance. That file system is not recognized by OS but the special software written for it. That file system is called SFX file system and has some specially designed utilities for the maintainance of it. The second daemon employed on those machines is responsible for task scheduling and called scheduler. Every task on the DIS machines consists of 6 phases. The status of the task phases and the status of the task together are managed and maintained by this daemon. Task phases can be stopped, paused, or restarted except the recording phase. The third daemon is called “NCI” or Network Command Interface and provides a mechanism for remote control of the other daemons of the DIS systems. Status checks of the DIS systems, and automation tasks are handled through that daemon. The fourth one is called “PLAY” or “REPLAY” daemon. It is used for testing or demonstration purposes. That daemon is responsible for sending recorded data back to the ingest card employed on the DIS systems, and allowing to re-playing it. Fifth daemon is called “MWD” or Moving Window

Display. It is responsible for capturing data passed through the ingest card and sending its spatially reduced form through the Ethernet network. Some clients are written for that daemon that are capable of interpreting the data and used for visualization purposes of the captured data in realtime in a spatially reduced way. In the current situation that system can just handle SPOT data in the system. All that 5 daemons are working through the standard Ethernet network on the top of UDP protocol. DIS systems also employ internal database system which is responsible for storing many details of the acquired and/or archived data. Archive is the 5th phase of every acquisition. In the system, every DIS unit employs its own tape recorder. Those tape recorders are working with DLT4[12] (Digital Linear Tape Generation 4) tapes and every DLT tape can store up to 40 Gbytes data or 10-20 acquisitions depending on its duration. DLT tapes are written in a specialized tape format called DART which is also developed by integrator firm. The internal database system holds the data structure on the DLT tapes. One of the main obstacles in that setup is 3 different forms of data recorded on 3 different recording units. That leads 3 different versions of database information about one acquisition and its data structure. It is seen that the current setup might have leaded some inconsistencies between the database system and DLTs. If some severe exception occurs due to the instability of the software architecture then this kind of inconsistencies can be observed. Since there is no real link between data inside DLTs and the data in the database system, or no method provided for the consistency checking, that kind of exceptions may lead data corruption and even data losses which are unacceptable in a ground station. As of now none of the DIS units is capable of archiving 1Gbps recording speed to support future satellite missions. The fastest one achieves 320 Mbps which is designed for Quickbird data acquisitions. On the other hand, data aging capabilities of the current DIS setup is not flexible enough to allow to manage data aging procedure in an automatic and transparent manner. Shortly, current DIS setup doesn't have that concept at all but just archiving data.

### 5.2.2 Proposed Implementation

After determining current system drawbacks of recording units, and the problems with the tape sub-system, it is thought to be logical to migrate archiving subsystem to a standalone and independent architecture. It was planned to achieve

1Gbps recording speed to support future satellite missions. Recording units are one of the most important key components with the demodulator systems, determining recording speed of the ground stations. Since upgrading that speed to 3 fold of the current setup is not an easy task, it is needed to modify the DIS units. First current disk array subsystem must be modified to be able to countervail the speed of the data stream needed to record data without any significant loss. To sustain such recording speed it is needed to support 1.6Gbps recording speed on the HDD side [2]. With a simple arithmetic, this is achievable with 3 SATA2 (Serial ATA generation 2) HDD units bundling them in a RAID0 setup. Since that RAID setup is known as insecure, it is proposed to mirror that setup on to other RAID0 array resulting RAID01 setup that consists of 6 disk units. Considering current SATA disk prices, the overall cost of operating DIS systems are even decreasing regarding to current SCSI setup. That disk arrays should also have an appropriate filesystem which should allow performant write access. Considering the current disk sizes, nowadays the size of the file system would be nearly 1.5 TB, which is not a big problem for modern filesystems but the read/write speed is more important. It is also important to achieve data aging in an automatic manner which is not possible with current setup. For that reason filesystem should be recognized by OS natively, in a transparent manner without any need of extra utilities. It was decided to use Reiser4 FS for DIS setup because of its capabilities [3]. One of the main problems with this filesystem was having no formal support on the Solaris OS directly. So it was decided to use external disk arrays with ISCSI interface which enables to be reached from many systems at the same time if a shared filesystem stack add-on for reiser4 is in setup. Since the Solaris OS has also very fine ISCSI initiator, reaching the ISCSI disk units wouldn't be a problem for that setup. However, stacking all those filesystems onto each other may lead performance bottlenecks, and besides, due to ISCSI being a slow protocol with its nature, the read/write performance may be decreased. But it is easy to balance that bottleneck with the current disk speeds and multiplying disk count in the RAID array.

Another important problem with the DIS units is the ingestion card used inside them. This card is designed and made manufactured by integrator firm and currently it is not in production. Since the software parts of the DIS units are strictly

bonded to that card and written specifically for it, it is not easy task to use any other software or ingest card. So it was decided to keep that setup on DIS units which is not causing too much problem in achieving 1Gbps recording speed. But it is needed to multiply card number on each unit by three since every ingest card can handle up to 200 Mbps transfer speed.

By considering the recording system topology, three different types can be mentioned. Namely single-capture, failsafe-capture, and dual capture systems[2].

The single-capture approach illustrated in figure 6 is the simplest, least expensive, but the least reliable one to implement. With this approach, there is a single system to receive the data and write it to storage. If this system fails, the input data is lost and continues to be lost until the system is repaired.

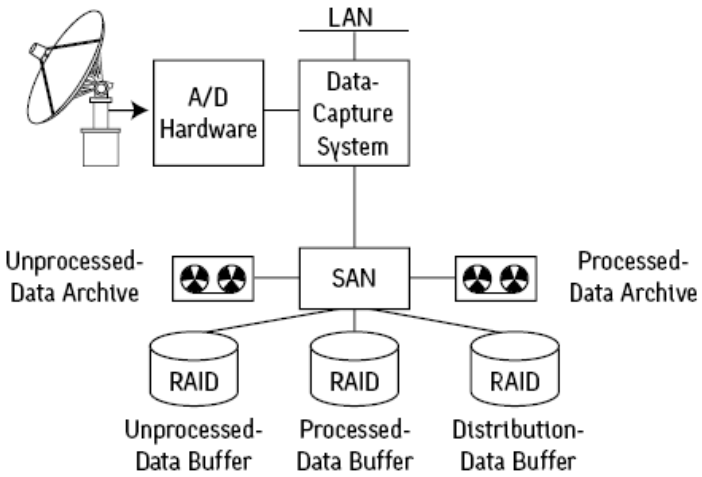


Figure 6. Single capture topology

The failsafe capture approach is a more reliable (but more expensive) approach. With this approach, there are two data-capture systems; one system is online and the other system is in standby mode. There are three degrees of standby: cold, warm, and hot. Cold standby means that the system is not involved in the data-capture function at all, but simply has the necessary capabilities and interfaces to perform the data capture function in case of the online system fail. This approach is illustrated in figure 7. If the online system fails, the processes running in the standby system are terminated and the data-capture process is started (either manually or automatically by the control server). However, during the time required to load the

data-capture function and make it operational on the standby system, all input data is lost. The advantage of this approach is that the system is back online prior to repair of the first system. Another disadvantage, however, is the cost of the second system. This cost is not totally allocated to data capture, since the system can be used for other processing functions when both systems are operational.

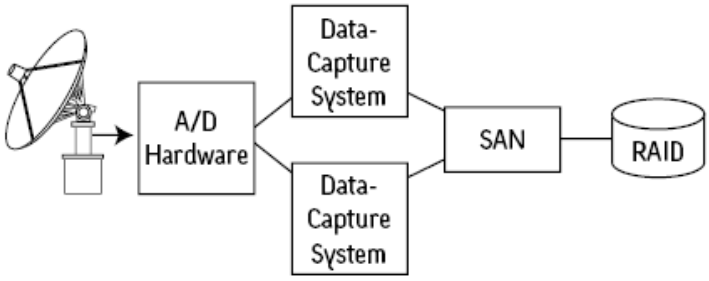


Figure 7. Failsafe capture topology

The dual-capture approach eliminates all single points of failure. Satellite sensor data is received over dual analog to digital hardware, sent to separate capture systems, and then written to separate RAIDs via separate SAN switches using different RAID controllers. Both data-capture systems receive, process, and store the data using completely independent hardware. One of the redundant data files created in the UDB is deleted by the control server upon completion of status processing from each of the data-capture systems. Undoubtedly this is the most reliable data-capture approach, but it is also the most costly one [fig.8].

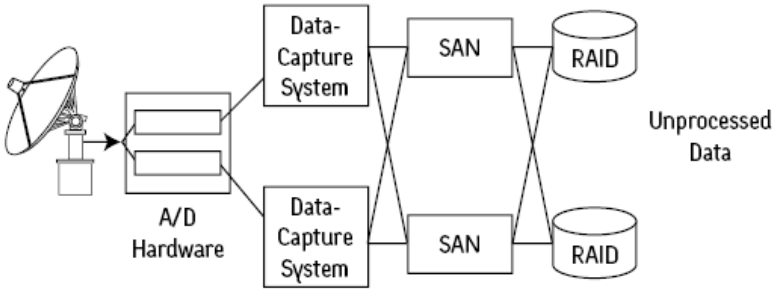


Figure 8. Dual capture topology

With all the consideration respect to given information above, it was decided to use failsafe topology in ITU-CSCRS ground station. With the proposed setup, now data recording speed may reach up to 600Mbps which is far from desired results, however gaining automatic data aging capabilities, transparent filesystem access, data snapshotting and mirroring, and volume management flexibility are positive features to have with that setup making other things a bit easier to achieve.

### 5.3 Processing Systems

In general, processing systems are taking their role after data has been recorded and archived or exposed to data aging routines. ITU-CSCRS ground station has two different processing sub systems, namely one for SPOT-2 and SPOT-4 satellites, and the other one for RADARSAT-1 satellite. Both systems have some automation issues inherently.

#### 5.3.1 Current Status

Current processing system consists of 2 node 3 CPU SGI Origin 200 system which can process a RADARSAT-1 wide beam mode product in a 45 minutes period. Considering near realtime demands of the end users, it can be thought that it is not feasible nowadays. The system consists of 3 MIPS R12000 CPU which has RISC architecture and very suitable for scientific processing in spite of their relatively lower execution speed at 240MHz, they are still achieving relatively good results because of their 64 bit native architecture. The nodes are connected to each other with a high speed interconnect called “craylink” or “numalink” generation 2 operates at 800Mbps [fig.9].

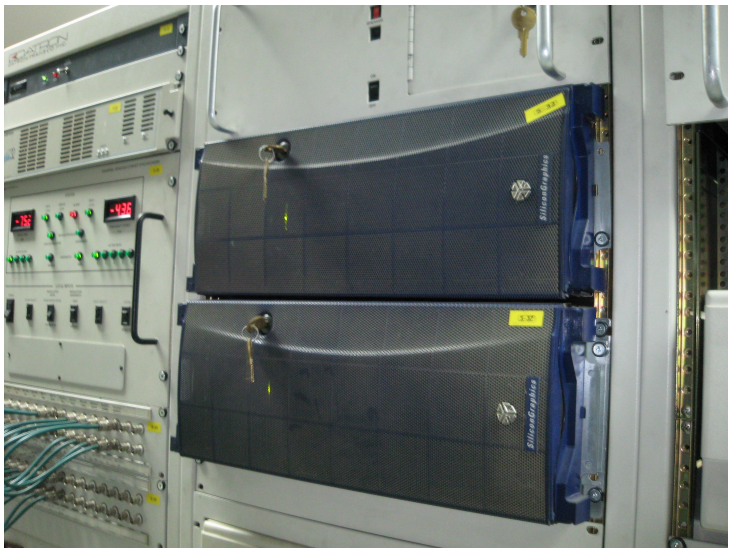

Figure 9. Current processing system

System runs IRIX operating system version 6.5 which has the roots in UNIX System 5 family of operating systems. Although, it was a solid foundation before, that OS is not in active development anymore.

Radarsat processor, used in ITU-CSCRS ground station, runs over IRIX operating system. Although, the vendor called Kongsberg[9] is still providing software update support for the software for very high prices, the software is one of the products on the market that can produce certified RADARSAT-1 end user products up to level 2.

The main part of that software runs as a daemon on the computer and a GUI based client which is very restrictive in the name of automation, is used to control that daemon. On the other hand, SPOT processor which can handle both SPOT-2 and SPOT-4 data has very specific problems. Although it has an commandline interface that allows to handle in automation tasks easily, however some stability problems have occurred within the years, depending mainly on the changes detected within the SPOT-4 data itself. Especially due to the dramatic increase of the bad detectors in the 4$^{th}$ band of the SPOT-4, it became difficult to maintain product quality with the current software. Another main problem with that software is, neither developer nor integrator firms does not support the software anymore. Like other stations do, the detector problems were solved by convenient enhancement algorithms developed by CSCRS R&D unit.

### 5.3.2 Proposed Implementation

The SGI systems have very unique capabilities such as scalability, parallel execution, and shared memory management, which are key foundations of super computing [fig.10], however, that specific hardware is not very easy to reach and maintain in Turkey due to its limited market which leads to technical support problems.

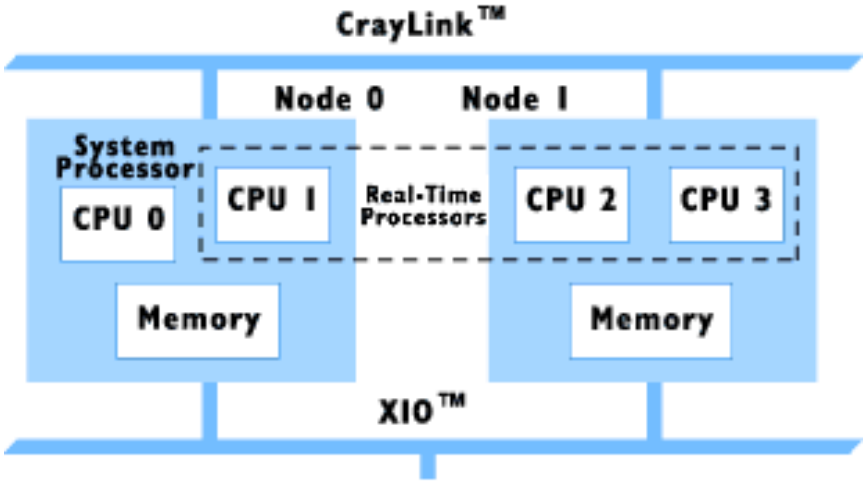


Figure 10. SGI CCNuma architecture

Considering SGI's current financial status on the market, and technical decisions that they taken through the Intel based systems and Linux based OS, it was decided to use X86_64 AMD architecture [fig.11]. Those processors were chosen because of their similarities with the processor architectures which are used in enterprise HPC Cluster setups like ones provided by companies such IBM and SUN. Those processors contain 2 cores in one packet which makes true multitasking is possible. The cores have native 64 bit architecture which is based on x86 architecture from Intel. Being 64 bit, processor enables reaching 40-bit physical address, and 48-bit virtual address spaces which basically means supporting more than 4 Gigabytes of RAM. Cores communicate with each other at CPU's native speed and have a 12.8 Gbps memory bandwidth. The processors have an internal memory controller and special interconnection similar to SGI architecture with each other called Hypertransport™ that can handle data streams up to 8Gbps. The power efficiency of the product should also be taken into consideration.

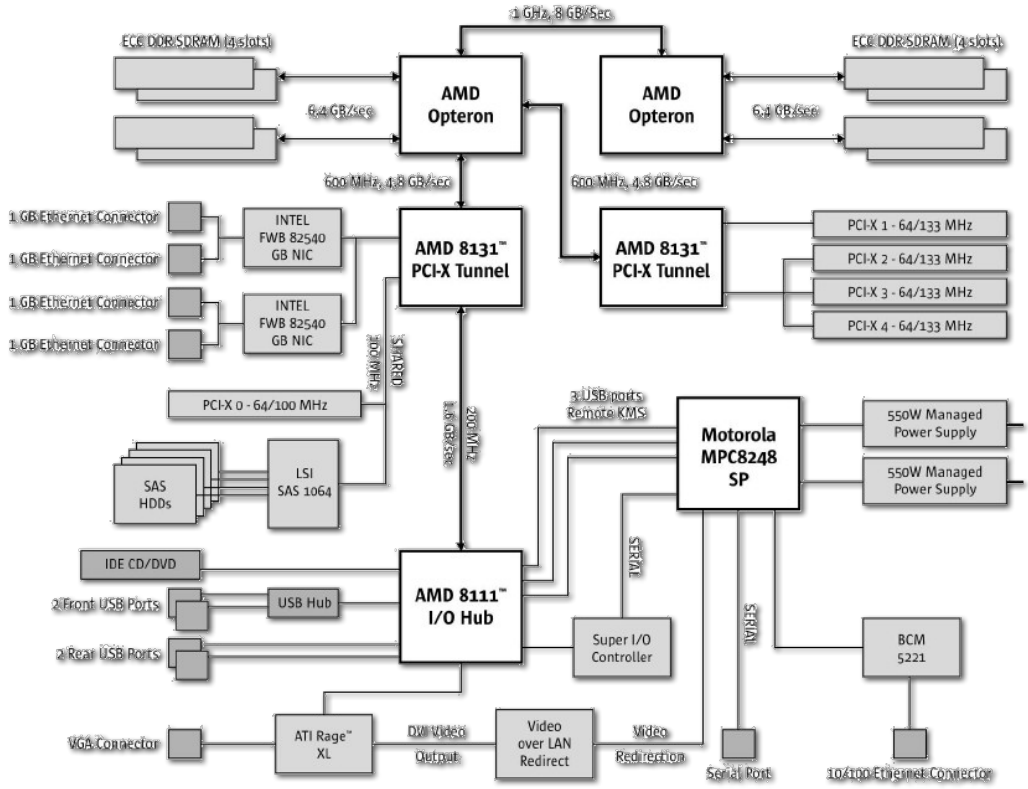


Figure 11. AMD x86_64 architecture

Although, there is no significant problems with the hardware side of the processing system, main problems occur at the software side, not allowing a flexible way for automation. With a detailed analysis and consideration of the market inclination to the Linux based custom OSes, it was decided to change IRIX platform altogether to the Linux based one.

For that purpose it was investigated three main opensource projects namely OpenSSI[4], openMosix[5], and Kerrighed[6] which all claim of providing “Single System Image” cluster software lives atop Linux kernel as a patch.

OpenSSI is the most robust system and covers nearly all SSI features that a user could expect. However the performance exhibited by this system is rather below the one offered by the other systems. The deputation mechanism used by openMosix[20] and the remote resource access mechanism used by OpenSSI lead to dramatic extra overheads for IPC after a process migration. openMosix, which is probably the most popular system, offers a good compromise between performance and covered SSI features. However, the openMosix stability is not as good as the one

offered by OpenSSI. Up to now, Kerrighed is still a research prototype, less robust than other 2 systems. Kerrighed does not support hot node addition and removal. Moreover a node crash often leads to a complete cluster crash. However, Kerrighed offers the best performance, specially regarding IPC and file system. Kerrighed is also the only system offering highly customizable features, efficient cluster wide memory sharing, process checkpointing and able to migrate and schedule threads[7].

Because all of the features provided, it was decided to examine the capabilities of the Kerrighed and OpenSSI in more detail. Since the Kerrighed started to give good results for the stations's needs, it was initiated to built 4 node cluster system with the computers which every one of them employs 2 dualcore AMD X86_64 processors with the help of Kerrighed clusterware [fig.12].

Figure 12. New four node processing server setup

It should be also noted that virtualization software must be also another concern for providing more efficient way to handle processing system consolidation and resource management. This is still in development, however, providing a virtualization software which can be run atop clusterware, seems a necessity for the processing needs of ITU-CSCRS ground station.

Because the current satellite data processor softwares did not work on top of Linux OS, it became a necessity to change the current processor softwares. Therefore, it was decided to purchase a custom RADARSAT processor software that

can run on top of Linux OS, developed by a Russian company named as Scanex [8]. However the integration issue of the Spot processor still remains to be solved, i.e. this is still an ongoing process to achieve the best solution with the current setup.

It is also important from the end user point of view to provide product levels higher than the product levels produced by data processor natively. For that reason some custom made processor softwares are in production for the automated product generation and quality validation. For this purpose, it is planned to provide products in seven main levels at ITU-CSCRS ground station as shown in figure 13.

| | | |
|---|---|---|
| LEVEL 7 | Mosaics—digitally assembled images to create large contiguous areas; applied to levels 2, 3, 4, and 6 | Partial |
| LEVEL 6 | Pan-sharpened—multispectral data sharpened with black-and-white data of same area | Available |
| LEVEL 5 | Digital terrain data—precision terrain information and stereoscopic imagery pairs | Not Active |
| LEVEL 4 | Orthorectified—constant elevation and corrected for terrain relief | Partial |
| LEVEL 3 | Precision geometrically corrected—groundcontrol points improve product accuracy; rectified to a constant elevation | Available |
| LEVEL 2 | Standard geometrically corrected—corrected for systematic distortions, no ground control points or terrain elevation required | Available |
| LEVEL 1 | Radiometrically corrected—oriented to sensor patch, corrected for transmission errors, adjusted for brightness/contrast | Available |
| LEVEL 0 | | Available |

Figure 13. Output product levels (available/planned) in ITU-CSCRS ground station

### 5.4 Data Storage & Archiving Systems

As mentioned previously, the current setup hasn't flexibility to age data in any manner, i.e. transparent access to data either by automation system and operator staff. However, with the proposed system changes, it will become possible to maintain, reach, and query data, for either data quality issues or at higher levels for end user needs.

### 5.4.1 Current Status

In the current setup all the storage mechanism focused on DIS systems. Since, they have many drawbacks, there is no easy way to handle data in a transparent way. To record and archive data, current DIS setups employ their internal recording storage volume and their own tape recording sub system. Storage volume consists of a 3 SCSI disk DAS array. After any data record has been done, it is directly recorded to the tapes and after a month of period it has to be fetched from tape units since there will be not enough place on those disk units to hold one month satellite telemetry data.

Tape recording units employ maximum of 7 tape units at a time. However, this is not so feasible for large scale projects and batch processing needs. Tape units are storing data in a format called DART provided by integrator firm. Although it is a comprehensive, reversible, extensible, and media independent[10], this tape format is hardly possible to read without custom made software. DART, employs an approach storing data in a multi-partitioned manner, together with their metadata between data segments. Such approach has been proven to be hard to maintain and not easy to integrate with “on the shelf” tape automation softwares.

### 5.4.2 Proposed Implementation

The data aging concept is a key foundation to provide easy and transparent access to both stored raw telemetry and processed data. To provide this foundation, to increase the storage volume provided by the system is necessary. After increasing “scratch area” volume on DIS units, it is also necessary to provide a space for telemetry data which they can reside for a proper period of time called “UDB” or Unprocessed Data Buffer as the 2nd phase of the process [fig.14]. After a proper period of time passed or as a requirement of the FIFO algorithm (whichever comes first), the data is transferred to tape subsystem as the 3rd phase.


Figure 14. ITU-CSCRS unique volume management and partitions

With the current implementation it is easy to add more storage when it is needed [fig.15]. All storage area seems as one big unique space and its partitions can be dynamically resized when needed. It is also possible to store processed, and distribution data in their respective partitions namely “PDB” or Processed Data Buffer, and “DDB” or Distribution Data Buffer[fig.13]. In those volumes all data is indexed, by providing quick access to the data stored for a couple of months of period.

Figure 15. ISCSI disk arrays employed in the ITU-CSCRS ground station

In the ITU-CSCRS ground station, it is also important handling many small files, especially needed for RADARSAT-1 data processing. For this reason, PDB is formatted with a filesystem called Reiser4, for its unique capabilities for such operation.

Reaching the goal proposed, separating the tape subsystem from the recording units were thought logical. For this purpose was purchased a new tape archiving system employs LTO3[11] generation 3 tapes. That unit employs 50 tape cartridges at the same time and engineered to scale up to 480 cartridges. It provides an infrared barcode reading mechanism for tape determination which was a deficiency in the current tape systems. With this new system, it is much easier to batch ingestion of telemetry data in both direction through the tapes [fig.16].


Figure 16. New tape unit employed on ITU-CSCRS ground station

### 5.5 Demodulator Systems

Demodulator systems are another key elements to achieve proposed recording speed along with the recording systems. In the ITU-CSCRS ground station, demodulators are responsible for bitsyncing, decrypting, and demodulating data altogether.

#### 5.5.1 Current Status

ITU-CSCRS ground station has 2 different demodulator systems which are differently configured for different satellite missions. One of them is capable of sustaining 120 Mbps transfer rates and is used for RADARSAT1, SPOT2, and SPOT4 acquisitions. The other one is configured for the Quickbird acquisitions and can operate at a speed 320 Mbps. Since the Quickbird data downlinking was prohibited by the US government, this demodulator system was started to be used as a backup for the other demodulator system. That modulator systems can support almost any modulation types namely BPSK, QPSK, AQPSK, UQPSK.. etc with a simple card addon upgrade on the hardware [fig.17].

Figure 17. One of the demodulator systems used at the ITU-CSCRS ground station

### 5.5.2 Proposed Implementation

As there was no easy alternative to achieve 1 Gbps transfer speed with the current setup of demodulator systems, it was proposed to purchase new demodulator systems [fig.18] such as ones produced by the a French company called In-snec [13] to support upcoming satellite missions. Those systems are very flexible to be configured for different satellite missions because of their software based nature and capable of sustaining such data stream speeds.

Figure 18. In-Snec software based demodulator system

This type of high data rate receiver systems provide support for BPSK, QPSK, O/S QPSK. A/U QPSK, 8PSK, GMSK modulation types. It is also tunable for recording speeds from 500 Kbps up to 2 Gbps. Providing flexible interconnection options such as ECL, makes it a good candidate for integration into ITU-CSCRS ground station [14].

## 6. Station Software

ITU-CSCRS ground station employs, 3 different types of software. First group is named as the system software which provides services for basic execution of system hardware tasks found in the station. The second group is application software used to provide automation in all ground stations and to command the hardware or other software systems. Third group is the database systems used in the different parts of the system.

### 6.1 Database Systems

#### 6.1.1 Current Status

In the current situation, ITU-CSCRS ground station employs 3 different database systems used in the different parts of the system. A Oracle database system which resides in the processing servers is used for automation tasks and SPOT catalog, and processing information. That database is also responsible for SPOT data query for SPOT data inventory. There is a utility called “world” is used for that purpose. The second database system, slightly reduced form of the database mentioned above, is found on the web server for web based access to SPOT catalog queries. The third database system is found on DIS units as mentioned in the DIS section above. In general, all those three database systems have some drawbacks for the current demands.

#### 6.1.2 Proposed Implementation

Since, none of the current database systems are allowing spatial database queries or supporting image data mining features in any manner, it was a necessity to setup a database system supporting that features for map generation, spatial and image query purposes severely needed in the ground station. For this purpose, 2 open source alternatives, PostgreSQL and MySQL, were evaluated and PostgreSQL

database management and its spatial extension (PostGIS) were chosen due to its large database handling capabilities which was also a big necessity for the ground station. The PostgreSQL & PostGIS are both opensource softwares that enable to modify source code and extend for the specific needs of the ground station. The current version of the PostGIS extension, has basic topology support, data validation, coordinate transformation, and related programming APIs for those purposes. For future versions it is planned to have full topology support, raster support, networks and routing, three dimensional surfaces, curves and splines. With the help of GEOS and Proj4 library, it can be easily used for map generation and map based queries. The PostGIS implementation is based on "light-weight" geometries and indexes optimized to reduce disk and memory footprint. The use of light-weight geometries helps servers to increase the amount of data migrated up from the physical disk storage into RAM, improving query performance substantially. PostGIS conforms OGC (Open Geospatial Consortium)[15] standards such as GML, WMS, WFS, SFS and many others.

Some of the features supported by PostGIS are;

- Geometry types for points, linestrings, polygons, multipoints, multilinestrings, multipolygons and geometry collections,
- Spatial predicates for determining the interactions of geometries using the 3x3 Egenhofer matrix,
- Spatial operators for determining geospatial measurements like area, distance, length and perimeter.
- Spatial operators for determining geospatial set operations, like union, difference, symmetric difference and buffers,
- R-tree spatial indexes for high speed spatial querying,
- Index selectivity support, providing high performance query plans for mixed spatial/non-spatial queries.

Another opensource database system planned to use for mainly XML data queries in preparation, is called MonetDB. The MonetDB is an extensible database system with its own algebraic language. The MonetDB can support different types of

backends in the same server architecture and can provide different query mechanisms. It was developed in the CWI (Centrum voor Wiskunde en Informatica) Holland and designed to provide high performance on complex queries against large databases, e.g. combining tables with hundreds of columns and multi-million rows. Therefore, the MonetDB can be used in application areas since performance issues are no-go areas using traditional database technology in a real-time manner. The MonetDB has been successfully applied in high-performance applications for data mining, OLAP, GIS, XML Query, text and multimedia retrieval. MonetDB internal data representation is memory-based, relying on the huge memory addressing ranges of contemporary CPUs, and thus differs from the traditional DBMS designs involving complex management of large data stores in limited memory. MonetDB introduced innovations at all layers of a DBMS. A storage model based on the vertical fragmentation, and a modern CPU-tuned vectorized query execution architecture often gives MonetDB a more than 10 fold raw speed advantage on the same algorithm over a typical interpreter-based RDBMS. MonetDB is thought one of the first database systems focused its query optimization effort on exploiting CPU caches. MonetDB also features automatic and self-tuning indexes, run-time query optimizations, and a modular software architecture. MonetDB conforms SQL-99 standards.

As of now, MonetDB supports 2 different types of frontend for data query.

- MonetDB/SQL: the relational database solution.
- MonetDB/XQuery: the XML database solution.

There are some language bindings and standard interfaces are also provided along that database system as such JDBC, ODBC, PHP, Python, Perl and C. MonetDB can be compiled and run over many different OSes ranging from UNIX, Linux and Mac OS X to Windows.

### 6.2 System Softwares

Satellite data processors, recording unit software, and clustering softwares are fall into that category and they are all explained in detail in their respective sections.

### 6.3 Application Softwares

Programming languages, definition & modeling languages, data management & mining softwares, CSCRS object based management middleware and its interfaces fall into this category.

#### 6.3.1 Programming Languages

Although, system development in ground station is mainly made by utilizing C, C++ languages, however it was seen that there was a need for a higher level language which can be used for connecting different software parts together for automation purposes and also for configuration purposes of the software subsystems developed mainly by C and C++. For this purpose many candidates were evaluated with a careful examination. Unfortunately, none of them were found conforming the high level needs of the ground station operations. So it was decided to develop a language called CSlang for this purpose. CSlang has been chosen to support object oriented approach from the very beginning and specifically designed for the station's needs. As of now, it can be compiled for different CPU architectures and has a virtual machine for x86, x86_64, and ARM processors. It is designed to be as scalable as possible therefore it is possible to develop software that can run on a scale enabling to change from multi processor servers to PDA systems having limited memories. CSlang supports C like syntax that eases system development and decreases the learning curve for beginners. It also supports byte-compilation for rapid execution and Boehm-Demers-Weiser garbage collector for automatic memory management. As of now, it has consistent and flexible libraries for socket programming, regular expressions based on PCRE (Perl Compatible Regular Expressions), DB access routines for MySQL, PostgreSQL, and MonetDB, an XML processing library based on libxml2, a primitive CGI library, and a console programming library which based on ncurses. It is planned to open source the code of the language for further development under GPL v2.

Sample code for CSlang that uses a system DLL on Windows™ platform is given below:

```
import "gui"
import "system"
user32 = system.dll("user32.dll")
user32.__loadpfunc__("GetWndRect", "GetWindowRect", 'bool, ['int, 'structp])

w = gui.window("Test", [20, 30, 200, 170])
e = gui.note(w, "", [0, 25, w.clientrect[3], w.clientrect[4]])
b = gui.button(w, "Get Main Window Rect", [0, 0, w.clientrect[3], 25])
b.onClick = func(){
        theRect = [:{itype:"i4", dim:4}, nil:]
        if (user32.GetWndRect(w.hwnd, theRect)){
                cr$ = chr(13) + chr(10)
                e.text = "Left = " + str(theRect[1]) + cr$
                e.text = e.text + "Top = " + str(theRect[2]) + cr$
                e.text = e.text + "Right = " + str(theRect[3]) + cr$
                e.text = e.text + "Bottom = " + str(theRect[4]) + cr$
        }else{
                e.text = "Call to GetWindowRect in User32.dll failed!"
        }
}
gui.enter()
```

### 6.3.2 CSCRS Management Middleware

Considering ground station topology given in figure 19, it is an obligation to consolidate all network nodes in a consistent way for automation purposes. For this purpose, a middleware system that can support different types of protocols to manage different aspects of the ground station is in preparation. Such middleware application will also be used for development of GUI based interfaces and web based interfaces in a more consistent way.

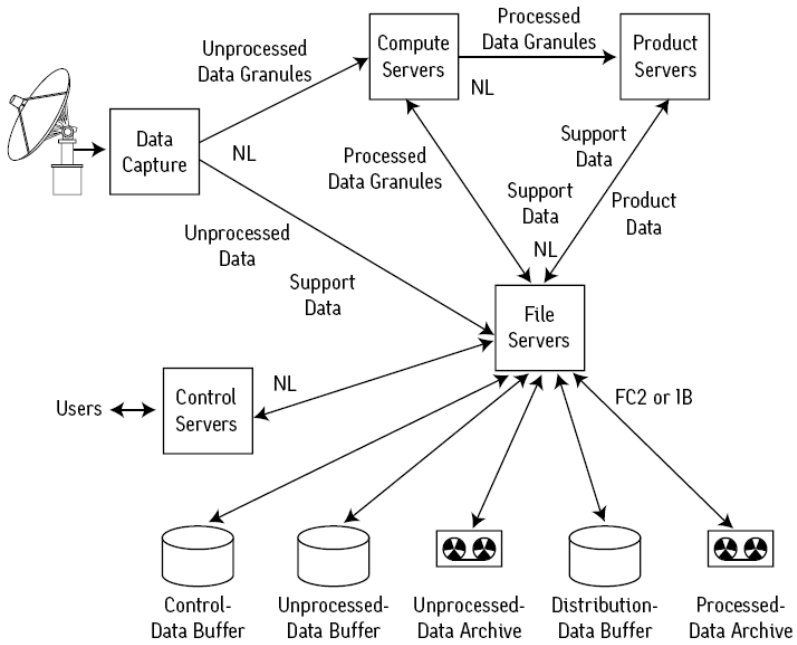


Figure 19. Proposed ground station topology

The proposed middleware will provide many automation routines for controlling IEEE488 interfaces, DIS and GSC automation routines, database query interfaces on a higher level, and data mining routines which are all specific to ground station needs. Internally that middleware will employ an object oriented approach, and objects will be passed through serialization around the station software components which would allow very flexible and inheritable configurations. For example, with a console based client, it is possible to give a Radarsat data ingestion command to system for further processing of raw telemetry data [fig.20].

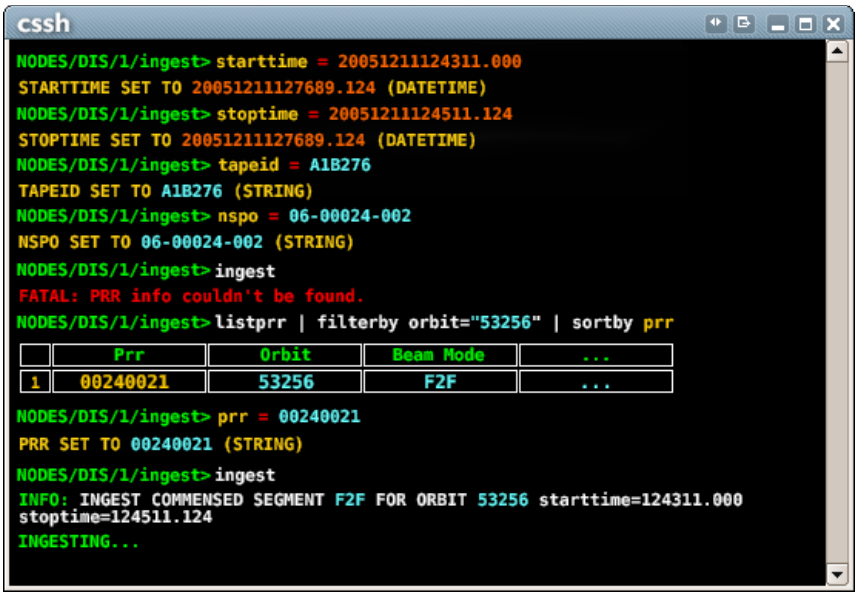


Figure 20. CSCRS object based management console

Same middleware software can be used for feeding web based interfaces with its XML production capabilities. The example query, above, which has been showing the list of PRR files for specific orbit, could also feed a web based interface with a slight change.

*listprr | filterby orbit="53526" | sortby prr | xmlize*

This command produces an XML output which can be fetched and used in a web based client easily. Such a web based interface is under development for Weekly Scheduling of the ground station shown in the figure 21.

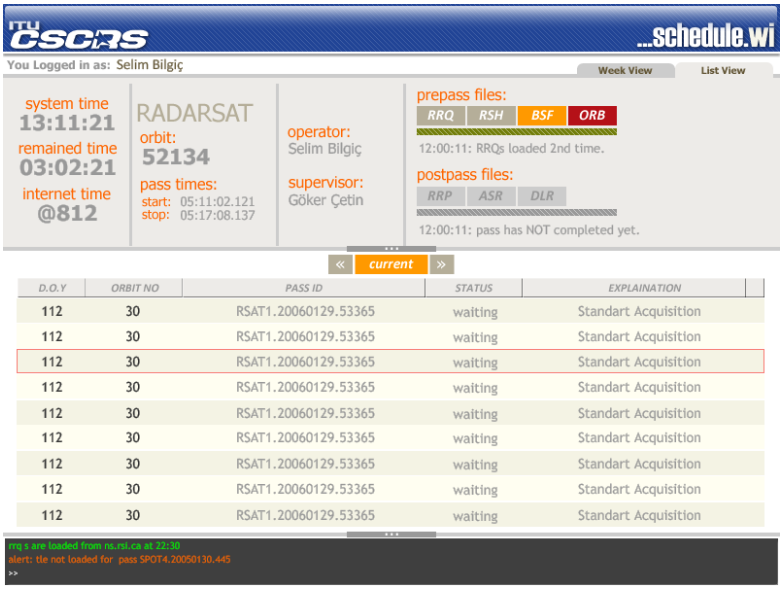


Figure 21. Web based pass schedule interface

Middleware provides a foundation for a commandlet based architecture which its commands are also inheritable objects. So that feature makes the software, conceptually similar to the projects such as “Monad Project” from Microsoft [17] and an opensource project “Osh” Object Oriented Shell [18]. All commandlets conforming base commandlet class have object oriented interfaces for standard input, standard output, and standard error streams that are very similar to traditional shell systems. This feature makes it easy to use in scripting complex tasks required for ground station automation.

### 6.3.3 Interface Definition Languages

With the development of middleware software, it became a necessity to use a interface language which can be used on different media with no modification or in a easily transformable way. That language should also provide a language agnostic way of handling user interfaces that will be developed in ground station. For this purpose XML syntax was chosen. With an investigation of a couple possibilities, one project is being conformed the needs. XML User Interface Language or XUL, is an XML user interface markup language developed by the Mozilla project for use in its cross-platform applications, such as Firefox. XUL relies on multiple existing web standards and technologies, including CSS, JavaScript, and DOM, which make it relatively easy to learn for people with a background in web programming and design. The main benefit of XUL is to provide a simple and portable definition of common widgets. This reduces the software development efforts in a way analogous to the savings offered by 4GL tools. Although XUL is usable in a web based development and directly inside Mozilla Project based browsers and Netscape generation 6 and 7, unfortunately it is not known by any other browser on the market. Therefore, it is necessary to develop another Ajax toolkit which is also in preparation by R&D team for other browsers. Holding interface structure with XUL, while developing interfaces for internal usage in station and then utilizing an XSL based transformation to convert it to an DHTML and/or XHTML based widgets for other browsers, seems a logical way in developing web based applications such as online catalog search. Data Distribution Server Interface, Spot Programming Request Interface, and Cloud Coverage Assessment Interface are all examples of such development approach [fig.22].

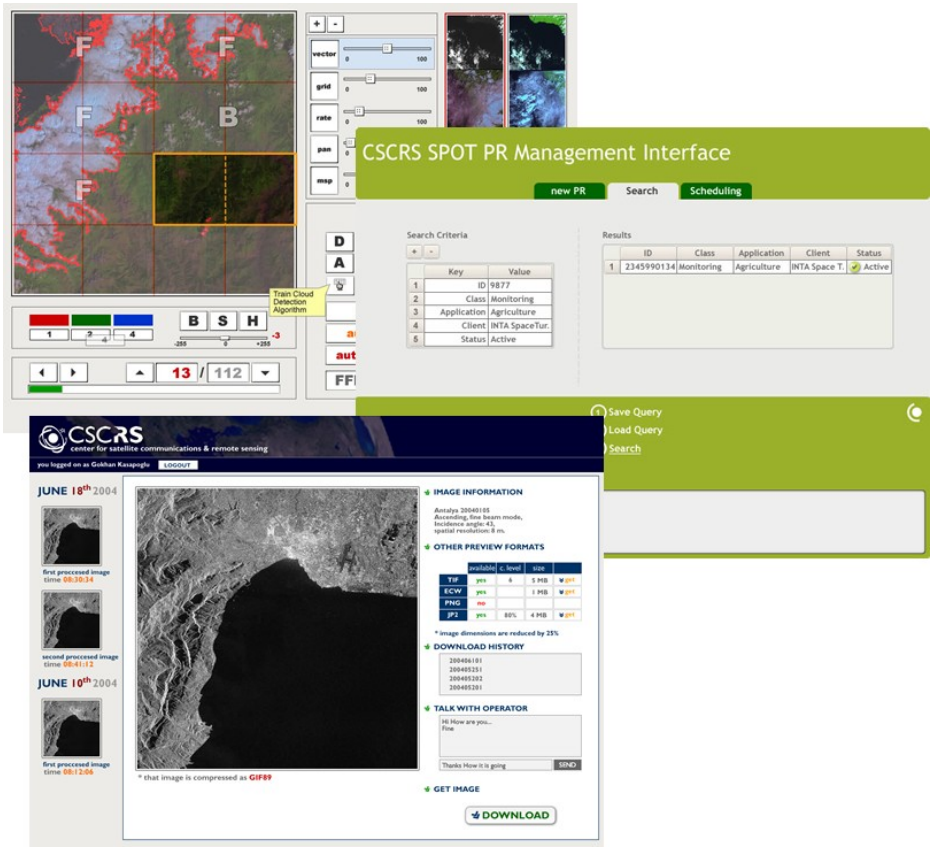


Figure 22. Some application interface examples

A fragment of a XUL file as used in a real world application can also be seen below;

```
<x:tabpanels flex="1">
<x:vbox>
        <x:hbox>
                <x:vbox flex="1">
                        <h:div class="infobox1"><h:span class="title1">system
time:</h:span><h:br/><h:div id="systemtime" class="title2"></h:div><h:span
class="title1">remaining time:</h:span><h:br/><h:div id="remainingtime"
class="title2"></h:div><h:span class="title1">internet time:</h:span><h:br/><h:div id="internettime"
class="title2"></h:div></h:div>
                </x:vbox>
                <x:vbox flex="1">
                        <h:div class="infobox1">?<h:div id="satellite"
class="title2"></h:div><h:span class="title1">orbit:</h:span><h:br/><h:div id="orbitno"
class="title2"></h:div><h:span class="title1">pass times:</h:span><h:br/><h:div id="passstarttime"
class="text"></h:div><h:br/><h:div id="passstoptime" class="text"></h:div></h:div>
                </x:vbox>
                <x:vbox flex="1">
                        <h:div class="infobox1"></h:div>
                </x:vbox>
```

```
<x:vbox flex="3">
<h:div class="infobox2"></h:div>
</x:vbox>
</x:hbox>
<x:hbox style="background:#FFFFFF;border-top:1px #CCCCCC solid;padding:3px">
<h:table border="0" width="95%" align="center">
<h:tr>
<h:td align="center">
<h:button class="button2">«</h:button>

<h:button class="button1" style="width:75px;font-family:Tahoma;font-weight:bold;font-size:13px;"
onclick="csConsole.commandlet.getpasslist(1);">current</h:button>
<h:button class="button2" style="width:25px;font-family:Tahoma;font-weight:bold;font-size:13px;">»</h:button>
</h:td>
</h:tr>
</h:table>
</x:hbox>
<x:hbox id="treeloader" flex="1">
<x:tree id ="passtree" hidecolumnpicker="false" seltype="multiple"
enableColumnDrag="true" flex="1" minheight="100px">
<x:treecols>
<x:treecol id="doy"  flex="1" label="D.O.Y"/>
<x:splitter class="tree-splitter" />
<x:treecol id="orbno"  flex="2" label="Orbit No"/>
<x:splitter class="tree-splitter" />
<x:treecol id="passid"  flex="3" label="Pass ID"  />
<x:splitter class="tree-splitter" />
<x:treecol id="status"  flex="2" label="Status"  />
<x:splitter class="tree-splitter" />

<x:treecol id="exp"  flex="3" label="Explaination"  />
</x:treecols>
<!--<x:treechildren id="passlist"
xmlns:x="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul" />-->
</x:tree>
</x:hbox>
</x:vbox>
</x:tabpanels>
```

## 7. Results & Recommendations

In this study, the necessary steps that should be taken for renovating the ITU-CSCRS ground station and also a guide for hardware and software policy were explained. With four years of experience, a base for software development standards conforming ground station needs, has been provided. It can be thought that, it was easier now to achieve planned goals with the supplied libraries and the development frameworks produced along many years for the related software technologies.

There is still a significant amount of work at the system software side. Provided frameworks and development environments should supply a start point to development team for their further efforts. There is also some polishment needed for the operation and automation routines of GSC, and DIS units. Some commandlet packages should also be developed using supplied middleware foundation libraries for that purpose. There is also a need for development at the internal workings of the middleware for transparent access to its settings which is currently hold in an XML file. A uniform file access mechanism similar to URL scheme, should also be implemented for transparent access to streams of different protocols. Therefore, development team can create more consistent, sharable, conformant, and readable code for frequently used development tasks such as network programming, stream based access to resources etc.

Overall software system in preparation, also makes it possible to develop mobile interfaces for system applications with just applying appropriate styling. Preparation of that styles are also seen as a future job to achieve.

With the proposed software architecture, it is going to be possible to automate, orchestrate station nodes. Collecting, and querying operational data about the nodes, flexible configuration and adaption of the system for a specific project are also going to be surplus values. It is also going to be possible to automatically collect, query and create reports related to operator labor in the station. As the result, designed system, mentioned in this study, will provide extensive information about data, hardware, software, and human resources for the decision makers.

## REFERENCES


[1] http://en.wikipedia.org/wiki/Ground_station, 2007

[2] **Rick Reid**, 2002. http://www.silicongraphics.net/pdfs/3285.pdf, *Whitepaper, SGI*

[3] http://www.namesys.com/benchmarks.html, 2007

[4] http://openssi.org, 2007

[5] http://openmosix.sourceforge.net, 2007

[6] http://www.kerrighed.org, 2007

[7] **Renaud Lottiaux and Benoit Boissinot**. Openmosix, openssi and kerrighed: A comparative study. Technical Report PI-1656, IRISA, Rennes, France, November 2004.

[8] http://www.scanex.ru/en/index.html, 2007

[9] http://www.kongsberg.com, 2007

[10] **DART Specification**, Oct 98. *Whitepaper, DATRON*

[11] http://www.quantum.com/Products/TapeDrives/LTOUltrium/LTO-3/Index.aspx, 2007

[12] http://www.dlttape.com, 2007

[13] http://www.in-snec.com/home/index.htm, 2007

[14] http://www.in-snec.com/products/pdf/ftp99=HDR.pdf, 2007

[15] http://www.opengeospatial.org, 2007

[16] http://monetdb.cwi.nl, 2007

[17] http://www.microsoft.com/windowsserver2003/technologies/management/powershell, 2007

[18] http://geophile.com/osh, 2007

[19] http://www.mozilla.org/projects/xul/, 2007

[20] **Amnon Barak, Shai Guday, and Richard G. Wheeler.** *The MOSIX Distributed Operating System, Load Balancing for UNIX*, volume 672 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

## CURRICULUM VITAE

He was born in Istanbul in 1977. He has started and finished his elemantary school in Istanbul. He started his education in Anatolian Zincirlikuyu Technical High School in 1990 and graduated in 1995. In 1998, he started his under graduate program at ITU Civil Engineering Faculty, Geodesy and Photogrammetry Engineering Department and finished that program in 2003. In that year he has been accepted to master program in the Geomatics Enginnering and became an assistant and started as a system administrator in Istanbul Technical University Center for Satellite Communications & Remote Sensing. He still continues his job as a system administrator at the CSCRS.