Online Collection and Monitoring of DVB-T Engineering Channel data stream

Udoyon Bhaduri

MSc Distributed Multimedia Systems
2002/03

The candidate confirms that the work submitted is his own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) ______________________
Summary

This project was done for the Interactive Operations group of BBC Broadcast. It’s main objective was to create a cost-effective solution for monitoring of their Engineering Channel stream which is used for firmware downloads for Set-Top Boxes. These downloads are in the form of private section files.

The project’s main objectives were to understand the DVB-T standard, which is used for the broadcasting of Digital Terrestrial Television signals, and develop a software system to monitor the above-mentioned data stream. The software was to confirm to a minimum of three out of five implementation levels specified by the client. Also, the requirements for the implementation of a real-time monitoring system were to be identified.

The analysis mainly consisted of investigating the DVB-T stream structure and the private section file that is to be broadcasted over the air. These led to the discovery that the stream did not confirm to the specifications, and neither did the private section file.

The final monitoring system implemented completely satisfies the first three levels which were the minimum requirements, and also the modified fourth level. The final level for logging in XML format was not implemented fully as the format was not available in time. Yet, the flexibility to implement this was included in the design and implementation. Future work to be done on the system has also been identified and documented. The system was successfully demonstrated on the 29th of August at the client’s site, and formally handed over.
ACKNOWLEDGEMENTS

First and foremost I must thank my supervisor, Mr W S Whyte, for his invaluable help and guidance throughout the project. He was always there to give me advice and support as well as thinking one step ahead for the complete project process. His emphasis on making me communicate with the client as well as the support staff provided me with a well-rounded project experience.

I must also thank Dr William Cooper of the BBC for providing me with this opportunity, and the support and guidance given by Dr Sue Farrell and Mr Matt Biddulph. I would also like the support staff at the School of Computing, especially Mr Graham Hardman, who made every effort to resolve the hardware problems faced at the beginning of the project, and Mr Simon Saunders who came to my rescue when I needed the TV card to be removed at very short notice for the demo to the client. I must also thank my assessor, Mr Karim Djemame for his expert comments after the demonstration, regarding the approach to be taken for the writing of the project report.

And last but not least, I would like to thank my classmates, who have always given me advice, suggestions and most importantly, support when things were not going well at all.
CONTENTS

SUMMARY ................................................................................................................................. I

ACKNOWLEDGEMENTS ............................................................................................................... II

CONTENTS .................................................................................................................................... III

1. INTRODUCTION .................................................................................................................. 1

2. LITERATURE AND TECHNOLOGY REVIEW ..................................................................... 3

   2.1 BACKGROUND ON DIGITAL VIDEO BROADCASTING ............................................... 3

   2.2 ARCHITECTURE OF DVB STREAMS ................................................................. 4

       The Transport Stream ..................................................................................................... 4

       Carousels – Cycling Repetitive Data ......................................................................... 5

       Firmware Download (Engineering Channel) .............................................................. 6

       DSI and Carousel in terms of file extraction .............................................................. 7

   2.3 ERROR DETECTION ................................................................................................. 8

       Error Checking in DVB ............................................................................................... 8

       Data Stream Monitoring for the Engineering Channel ........................................... 9

   2.4 CURRENTLY AVAILABLE APIs FOR DVB ................................................................. 9

       2.4.1 Java TV API ....................................................................................................... 9

       2.4.2 MHP Specification .......................................................................................... 9

3. ANALYSIS AND DESIGN ................................................................................................. 11

   3.1 METHODOLOGY ...................................................................................................... 11

   3.2 ANALYSIS: INVESTIGATIONS INTO THE STREAM STRUCTURE ..................... 11

       3.2.1 Anticipated structure of the source file and the stream .................................... 11

           Source file ............................................................................................................... 11

           Engineering Channel stream .............................................................................. 12

       3.2.2 Comparing Structures ..................................................................................... 13

       3.2.3 Description Of Investigations .......................................................................... 14

       3.2.4 Effects of Findings .......................................................................................... 15

   3.3 SYSTEM DESIGN ...................................................................................................... 16

       3.3.1 Platform and language used ............................................................................. 16

       3.3.2 Overall architecture ........................................................................................ 16

           The OAD Monitoring System ............................................................................. 16

           The packaged classes ........................................................................................ 18

           The driver class - OADMonitorDriver ................................................................. 18

       3.3.4 Activity Flow .................................................................................................... 19

           Flow of the Driver class ...................................................................................... 19

           Flow of the StreamProcessor class ................................................................... 20

           Log file format design ....................................................................................... 23

4. IMPLEMENTATION .......................................................................................................... 24

   4.1 DIFFERENCES FROM THE DESIGN .......................................................................... 24

   4.2 IMPLEMENTATION SPECIFIC DETAILS .................................................................... 24

           OADMonitorDriver ............................................................................................... 24
5. TESTING AND EVALUATION .................................................................30
  5.1 SYSTEM TESTING ............................................................................30
    5.1.1 Testing against various stream characteristics .........................30
    5.1.2 Testing the real-time nature of the system .....................................32
    5.1.3 Incorrect parameters/options and I/O permissions .......................34
      Input/Output errors .................................................................34
      Incorrect entries in the properties file .........................................35
      Command-line switches ...........................................................36
  5.2 EVALUATION ...................................................................................38
    5.2.1 Based on Objectives .................................................................38
    5.2.2 Existing Products .......................................................................39
      DTV Interactive – ITV Stream Analyzer ........................................40
      Tektronix Products ........................................................................40
    5.2.3 Deficiencies and Possible Improvements ......................................41
  6. PROJECT MANAGEMENT ....................................................................42
    6.1 INITIAL PROJECT PLAN ...............................................................42
    6.2 REVISED PROJECT PLAN .............................................................43
      6.2.1 Reasons for revision ..............................................................43
      6.2.2 Final Plan ...............................................................................44
  7. CONCLUSIONS AND FUTURE WORK ..............................................45
    7.1 CONCLUSION ................................................................................45
    7.2 POSSIBILITIES FOR FUTURE WORK .............................................45

REFERENCES .........................................................................................47

APPENDIX A – PERSONAL REFLECTIONS ...........................................48
APPENDIX B – PROJECT OBJECTIVES AND DELIVERABLES FORM ....50
APPENDIX C – MARKING SCHEME AND INTERIM REPORT COMMENTS ....52
APPENDIX D – PROJECT REQUIREMENTS DOCUMENT ........................54
APPENDIX E – DESIGN DOCUMENT ...................................................58
APPENDIX F – JAVADOC FOR THE CODE .............................................68
APPENDIX G – INSTALLATION AND USAGE DOCUMENT .....................76
APPENDIX H – STREAM AND PRIVATE SECTION FILE COMPARISON RESULTS...81
APPENDIX I – EVALUATION BY THE CLIENT .......................................85
APPENDIX J – LIST OF FIGURES AND TABLES .....................................88
1. Introduction

This project was done for the Interactive Operations group in BBC Broadcast Limited. The team provides data playout services for the BBC and other clients, such as data for interactive services and firmware upgrades for Set Top Boxes or digital television receivers. The firmware upgrades are available from the DVB-T transport stream used for terrestrial broadcast of the digital television signal within a data stream called the Engineering Channel. Since manufacturers have to purchase the slots for such broadcasts, the BBC required a way to verify the reception of this data. Hence, this project looks at a cost-effective solution to the problem.

The formal objectives of the project were as under:

1) Understand the DVB-T standard
   Extensive background research was conducted to find out as much information as possible, on the standard and its implementation. This research included various technical papers, articles and books on the topic, as well as proprietary documents supplied by the client.

2) Design and develop a demonstrator software for online collection and monitoring of the data stream.
   The development was to be in accordance with the five levels specified by the client, as shown in Appendix D. The minimum requirements were set to satisfying level three. They are as follows:
   i. To identify in a (binary) private section file a specific identity code on a Linux filesystem.
   ii. To identify a private data stream in a DVB-T transport stream and store on a linux filesystem, and then identify the specific identity code.
   iii. To monitor a private data stream in a DVB-T transport stream and identify the specific identity code in the file and log the time of detection for each re-occurrence of the identity code.
   iv. To use the above to verify that a file on the filesystem has the same identity and contents (checksum) as the one available in a private data stream, and log to a file any exceptions against time.
   v. To generate error messages for each exception in a defined XML format that can be http posted to another operational system.
3) Identify additional requirements for a possible real-time system.

The requirements for the demonstrator do not enforce a real-time processing constraint. Yet, the final objective of the project is to determine if a real-time system is possible, and identify requirements for the same.

This report starts with a description of the DVB standard, focusing on the particulars concerning the detection and extraction of the data in the Engineering Channel. Also, proprietary Java libraries used for accessing the DVB stream were investigated, but as the report shows, they were not suitable for PC development. During the analysis, investigations were made into the actual structure of the stream and the files to be extracted, during which it was realised that the streams were not strictly according to the structure suggested by earlier readings and information. This resulted in a change within the objectives, whereby extraction and CRC-checking of the file was replaced by the detection of a message in the received stream within a specified time interval. The report describes the investigations and the algorithms used for the same. This is followed by a description of the design phase where the process flow for the system is described, as is the final log format. The implementation section deals with specific issues faced during the implementation.

The final software produced after the investigations confirms to the revised specifications and meets the minimum requirements as well as satisfying additional extra requirements. The project intended to investigate the requirements for a real-time monitoring system, but has resulted in the actual development of such a system. Evaluation of the system has been done in terms of components, whether the requirements have been met, as well as the real-time performance of the system. The code and design for the solution has also been made flexible enough to accommodate future inclusion of a CRC check, once the actual structure of the stream is known.

The report goes on to describe the project management process during the project, comparing the initial plan versus the actual plan that was followed. The final section gives the conclusion as well as future work on this project.
2. Literature and Technology Review

2.1 Background on Digital Video Broadcasting

DVB - The European Standard for digital television

Till the late 90’s television transmission was done using analog signals. Digital television has now taken over, and is available via satellite and cable networks all over the world. It provides a much richer experience, in both audio and video quality, as also interactivity. But there was no international standard for digital transmission. DVB was created to solve the problem of standardisation of television services in Europe, and further adapted to suit the services in America as well as Japan, encompassing all the current service types – NTSC, PAL, SECAM and MAC [1; 2]. The resulting standard uses MPEG-2 compression for both audio and video [1, 14; 2, p.1]. Today, systems which are partly digital and partly analogue, like High Definition TV, are already using this technology [3 p.1].

DVB is a large-scale digital broadband network, providing additional information and interactive services that are not offered by traditional television broadcasting technologies, where set-top boxes at the user end receive the data from the network [3, p.1]. It benefits both the broadcasting service operators as well as the viewers. The availability of more channels as well as better picture quality is a boon to the viewer, while the high efficiency of digital transmission is useful for the service operators because “digital compression packs five or more times as many channels in a given distribution-network bandwidth” [3, p.1]. But along with these benefits, there are some characteristics which some users might not find appealing. Since the signals received are compressed digital signals using forward error correction techniques, a DVB system would either give perfect picture quality, or below a certain threshold of signal quality it would give no picture at all, unlike the analogue systems [4, p.1].

DVB-T

Broadcasting can be split into various transmission mediums such as satellite, cable, terrestrial, SMATV and so on. The specific standards for each of these are listed in Appendix A. This project is concerned only with the terrestrial standard for DVB, known as DVB-T.
Terrestrial DVB has been given importance not only because of technological issues, but also due to financial and logistical factors seen by certain countries, “to make more efficient use of the UHF spectrum, compressing existing services into less spectrum, opening the way to license further broadcasting services, and yet liberating a large part of the very significant amount of spectrum currently used for analogue television broadcasting for auction to new communication service operators”[5, p.6]. DVB-T is growing steadily in acceptance all over Europe, since its launch in the UK and Germany in late 1998 [6, p.224]. The small spectrum available for new digital services has meant that more effort be put in to it as compared to other media. Due to this, it has lagged behind as compared to satellite and cable, however the BBC had planned trials as early as 1998 [7 p. 85].

DVB-T services are provided in the Ultra High Frequency (UHF) band [1 p.188]. It uses Orthogonal Frequency Division Modulation (OFDM) multicarrier technique as a result of a need to support single-frequency networks [6, p.232]. The individual carrier signals within OFDM can be Quadrature Phase Shift Keying (QPSK) or Quadrature Amplitude Modulation (QAM). Satellite transmission uses QPSK for modulation. The BBC uses 16 QAM for its terrestrial transmission.

### 2.2 Architecture of DVB streams

#### The Transport Stream

The DVB-T standard-prescribes a video resolution of 720x576 pixels at 25 frames per second. This results in a very large amount of data, hence the need for compression. Hence, MPEG-2 is used both for audio as well as video compression [1 p.152]. The MPEG-2 standard also allows multiplexing the audio, video and data streams into a single bit stream called the MPEG-2 System [6 p.89]. Additional information about programmes as well as that required for technical servicing is also included in this stream. The following diagram describes multiplexing [6 p.89-90]:

![Multiplexing in the MPEG2 System](image)
It begins with packetising the audio, video and data streams into large chunks called Packetised Elementary Streams (PES). The PESs are further divided into smaller packets and combined to form either a Program Stream or a Transport Stream (TS). DVB uses the TS only. The TS allows transmission of several different time bases meaning many programs in one channel. It is suitable for use in error-prone channels, and has a fixed packet length of 188 bytes. The structure of the TS packets is shown below:

![Fig 2.2: Structure of the Transport Stream](6 p.95)

The sync byte is used to signal the start of the packet. The Packet Identifier or PID is used to determine what kind of data is present in the packet. The optional adaptation field contains signalling information. Some of these PIDs are associated with certain tables, which provide program-specific information (PSI) or service information (SI). The PSI contains information to be used by the Set Top Box (STB), ranging from PIDs of other tables, programme specific information, to private data for conditional access and transmission attributes. The SI contains information that the STB provides to the user, such as programme type, description, classification etc.

### Carousels – Cycling Repetitive Data

User interactivity data and the Engineering Channel stream are provided in data carousels. A data carousel is simply a structure that contains different sets of data, and keeps cycling through them, and is transmitted on a single PID. This helps in maximising the use of the available transmission bandwidth, and the maximum waiting time for an STB to locate specific programme information is the carousel cycle time [3 p.135]. An object carousel is a similar structure, but carries ‘objects’ such as files and directories and these objects are transported in modules, which represent the data carousel layer [8]. The BBC uses this DSM-CC object carousel to transmit the information for the STB firmware downloads [9].
Firmware Download (Engineering Channel)

According to [9], the data broadcast can be carried as a standalone MPEG program or as a component on one of the existing services. The signalling in the PSI is used to identify the service, and is found in the \textit{data_broadcast_id_descriptor} in the PMT. SI signalling can be used to indicate the presence and content of the download Object Carousel, in the Network Information Table or the Service Description Table. The actual file broadcast is carried in a DSM-CC Object Carousel carried on a single PID. The identification of the receivers the download is meant for is carried in the \textit{UserInfo} field of the \textit{DSMCCServiceGatewayInfo}. This information is called the Manufacturer Information Structure (MIS). The \textit{ServiceGatewayInfo} is carried in the \textit{privateDataBytes} of the Download Server Initiate (DSI) message, which is broadcast every 5 seconds.

The MIS contains information about the manufacturers, receiver models, download times etc, specific to the current download. The structure is as follows [9, p.7-8]:

\begin{table}[h]
\centering
\begin{tabular}{|l|c|c|}
\hline
\textbf{manufacturer_information_structure() } & \textbf{No.of Bits} & \textbf{Format} & \textbf{Value} \\
\hline
manufacturer_information_tag & 8 & bslbf & \textbf{0xE0} \\
manufacturer_data_length & 16 & uimbf & \textbf{N} \\
manufacturer_id & 24 & uimbf & \\
version_id & 32 & uimbf & \\
\textbf{for}(j=0;j<(N-7);i++){} & & & \\
\textbf{ manufacturer specific data } & & & \\
\} & & & \\
\hline
\end{tabular}
\caption{MIS Structure}
\end{table}

\textbf{descriptor_tag}: This 8 bit integer with value \textbf{0xE0} identifies this descriptor.

\textbf{manufacturer_data_length}: This 16-bit field gives the total length in bytes of the following manufacturer specific data. This field may be empty if the broadcast currently contains no active downloads.

\textbf{manufacturer_id}: This 24-bit field carries an IEEE OUI uniquely identifying the origin of the software download. This will identify the receiver manufacturer.

\textbf{version_id}: This 32 bit field identifies the receivers that require this download in a manufacturer specific way.

The receivers use the \textit{manufacturer_id} and \textit{version_id} to determine if the download is for them or not. It was found that the list of IEEE OUI numbers for different corporations is maintained on the IEEE website - \url{http://standards.ieee.org/regauth/oui/index.shtml}. 

Udoyon Bhaduri
MSc Distributed Multimedia Systems
DSI and Carousel in terms of file extraction

For the actual file extraction from the stream, the specifics of the DSI and the object carousel is given in [10 pp.247] and is as follows. The DSI message provides the object reference to the service gateway. A number of “taps” are present. These are [11] data structures defining the elementary stream, the protocol used (whether it is a carousel or a program elementary system) to encode the data element within the elementary stream and how to find the data element within the elementary stream. Further, [10] describes the way to obtain the data carried in the carousel – Using the tap with the BIOP_DELIVERY_PARA_USE, the DownloadInfoIndication (DII) message can be found, which gives information about the broadcasted module and its location. This is contained within the BIOP_OBJECT_USE tap structure. The ServiceGateway object is the root-directory which references the sub-directories for each separate data download – there can be more than one, though the client’s current system has only a single download at a time. Hence, the object carousel contains a number of data carousels, each of which carries a separate download. Links to the modules are provided as object references in the BIOP_DELIVERY_PARA_USE taps within the DII, and the data can now be extracted from the relevant modules.

An example diagrammatic representation of the above follows. We assume that the stream contains more than one download, each in a separated directory. Module 1 holds the directory structure, and module 2 is the desired subdirectory:

![Diagram of DSI and DII structure](image)

Fig 2.3: Example Structure of an Engineering Channel with Object Carousel
2.3 Error Detection

Error Checking in DVB

One of the main aims of the project is to check the validity of the file received. Tests should be undertaken to study the content of the file and decide on that. In his book, “DVB – The International Standard for digital Television”, Reimers specified the following tests for continuous or periodic monitoring [6, p.264-266]

1. Transport_error: One bit of the TS packet header is used as the transport_error_indicator. If the bit is set, the payload of the packet is unusable for decoding purposes.
2. CRC_error: The individual sections that the PSI and SI tables are made up of, are provided with a Cyclic Redundancy Check code. The receiver can carry out the check and determine whether the data have been correctly received.
3. PCR_error: The Program Clock References serve to synchronise the System Time Clocks (STC) of the multiplexer and the demultiplexer. Separate PCRs can be transmitted for each programme. There is an error if the interval between the PCRs of a programme is greater than 40 ms, or if the difference between two subsequent PCRs of a programme is greater than 100 ms, without a temporal discontinuity being indicated.
4. PCR_accuracy_error: This method can only be applied to a TS with a constant data rate. A deviation greater than ±500 ns for the received PCRs indicates an error.
5. PTS_error: The Presentation Time Stamps supply the synchronisation of the video and audio signals of a programme. The maximum time interval should not exceed 700 ms in any elementary stream.
6. CAT_error: The information required for decoding scrambled transport streams is coded in the Conditional Access Table. A CAT_error is indicated if a TS contains packets whose transport_scrambling_control fields signal scrambled packets although there is no CAT, or if a section of another table is coded in a packet with PID 0001HEX, reserved for the CAT.

The DSM-CC object carousel provides an option to use either a CRC32 or a checksum for detecting bit errors. The section_syntax_indicator set to 1 indicates the use of CRC32 and checksum otherwise [10].
Data Stream Monitoring for the Engineering Channel

To monitor the data stream, it has to be first extracted using its PID. Once the stream is located, the manufacturer_id, version_id and manufacture_data_length of the MIS can be used to determine if the current download is the one that has to be logged, and a timestamp can be written to a log file whenever this descriptor is found. The stream has to be reconstructed packet by packet. According to the specification given by the BBC, the CRC or checksum can be used to determine if there is an error in the received file. The alternative approach is to check the file bit by bit. If an error is detected, it should be written to a log file, and can be written in a specified XML format. This kind of error checking requires a major amount of processing, and hence it might not be possible to do it in real-time.

2.4 Currently available APIs for DVB

As the task of decoding the object carousel structure is complex, a search was made to find existing APIs which could be used for this purpose. Neither of the two found were completely usable.

2.4.1 Java TV API

This API is available from Sun Microsystems. It seemed to be a good candidate for solving the problem, but upon further investigation and trials, it was found that it serves only as an emulator.

First of all, the reference implementation available is only for the Windows environment [12]. After experimental installation of the environment and API, it was found that an existing complete transport stream file is to be specified and an object file created out of it. This object file could now be processed by the methods and classes of the API. There was no method outlined by which a live stream could be processed.

Also, it is not possible to create a standalone application. The applications which are created, termed Xlets, can only run within the emulation environment [13]. As the ultimate aim was to create a software that processes a live DVB stream and can be run as an independent cron job on a Linux system, the Java TV API could not be considered further as part of the solution.

2.4.2 MHP Specification

MHP stands for Multimedia Home Platform. This API seemed more promising, and might have been useful on a dump file, but it only works on the audio, video and interactive data within the
transport stream. Initial research suggested that it does not access the Carousel within which the Engineering Channel data can be found.

But later research has found that the new MHP specification does support access to the DSM-CC Object carousel within the transport stream [14]. As the implementation had already been completed by this time, there was no possibility of reviewing if it could be used. Issues such as dependence on the Java TV API and hence the use of Xlets would have to be looked into.

Hence, neither of the APIs could be used for the process, and it was decided that an analysis of the stream structure and the data in the private stream was needed to be able to implement a method of accessing the file within the stream.
3. Analysis and Design

3.1 Methodology

This project has resulted to be an investigative research project rather than being solely a software development project. Consequently, a major part of the time was spent in investigating the stream structures, which resulted in less time being available for the actual design process.

Hence, the software was developed using a modular development methodology. Individual modules were developed during the investigations, and were integrated during the design phase. Modules were developed for individual investigations, and were tested before determining the results. Once all the investigations were completed, the appropriate modules were merged together in the final design, along with additional modules required to satisfy the different levels defined in the requirements.

3.2 Analysis: Investigations into the stream structure

The background reading suggested that the problem of extraction, without using an API for decoding the carousel structure, is a very steep task. However, during further communication with the clients, it was suggested that the stream structure is not as complex as the reading suggests – the only decoding to be done would be the MPEG de-packetisation. Hence, the next step was to try and analyse the private section file as well as the stream structure, to determine if it was indeed possible to extract the file from the stream without the complexities of the carousel structure.

3.2.1 Anticipated structure of the source file and the stream

Source file

The first level of the requirements required the detection of the manufacturer id within the private section file itself. Hence, the task was to read the bytes from the file and match the MIS descriptor as well as the manufacturer ID.
The following diagram shows the private section file structure as it was anticipated to be:

![Diagram of private section file structure](image)

As can be seen, the file is made up of a number of sections, which are separated by the MIS/DSI information carrying the manufacturer ID. Trapping of the ID would involve matching the descriptor 0xE0 and the manufacturer ID, separated by 2 bytes.

**Engineering Channel stream**

The initial study and information from the client implied that the Engineering Channel stream should only have MPEG-packetised chunks of the private section files. This is similar to the description of the DVB transport stream mentioned in the background review chapter.

The structure of the anticipated Engineering Channel stream is given below:

![Anticipated structure of the Engineering Channel Stream](image)

According to this, the extraction of the file involved taking the stream and stripping off the header, and joining the resulting payload. The only issue left would be to determine the beginning of the file in the resulting data.
The readings however had suggested otherwise. According to [10], the payload would contain an object carousel structure out of which the firmware data would have to be extracted. This inconsistency led to the investigations into the actual structure of the stream and file, to determine which of the opinions was actually correct.

### 3.2.2 Comparing Structures

A study of [10] gave the carousel descriptors and other information. To confirm if there were any differences in the structure of the private section file as well, a program was written. It matched the ‘magic number', carousel descriptors and file descriptors within the private section file. It was proven that these descriptors existed within the file, implying that it already had the carousel structure within it, as shown in fig 3.2 a). This was later confirmed by the client.

![Fig 3.2 a) Actual Private Section File.](image)

But upon investigation, it was found that the structure is more complex, and has some additional header information structure surrounding the private section chunks, as shown in fig 3.2 b). It was not possible to determine the structure of this information due to paucity of time, but it was determined that it was definitely of variable length. Later communication with the client revealed that this was additionally part of the carousel structure, and that the private section file needs to be checked for errors section-by-section, and is not to be treated as a simple file.

![Fig 3.2 b) Actual Structure of the Engineering Channel Stream](image)
3.2.3 Description Of Investigations

1) Direct Comparison of the Transport Stream to the Private Section File

The first investigation involved comparing the bytes of the private section file directly to the bytes of the transport stream, after the transport stream’s header was stripped off. This should have resulted in a perfect continuous match till the end of the private section file, if it was according to the client’s information.

But the result was that after a certain number of bytes, varying between 150 and 180, there would be no more matches. This implied that the transport stream payload contained more than just the private section file. This was carried out on three different transport stream and private section files, with similar results.

2) Attempt to find Object/Data Carousel Descriptors in both file as well as stream

A number of descriptors and magic numbers were identified from [10], signifying the presence of a carousel. A program was written to find these descriptors within a file. At this point, the stream was being dumped to a file and not being processed directly off-air.

The first descriptor used was the object carousel “Magic Number” – 0x42494F50, to signify a BIOP (Broadcast Inter-Operable Protocol). Both the private section file and the DVB stream dump file showed several and equal number of occurrences of this descriptor in exactly the same pattern. The difference was in the spacing between occurrences of this and the DSI, which was greater in the case of the DVB stream dump. This had two implications, the first being that the carousel structure was already present within the private section file, at least partially. Secondly, it meant that there was a regular “padding” between portions of this file as it was being broadcast on the stream. Sample output of these is given in Appendix H.

The other descriptors used were the file descriptor – 0x66696C00 – signifying a file broadcast, and a stream descriptor – 0x73747200 – signifying a stream broadcast. There were no matches found for the stream descriptor in either set of private section and stream dumps. However, the file descriptor was matched in both sets of private section and stream dumps. Similar to the Magic Number, the file descriptor followed the same pattern in the respective private section and stream dump. Again, the difference being that there was a greater distance between the occurrences within the stream.
dump. Another observation in both the cases of comparison was that the “padding” between descriptors was not of a standard length.

This further proved the ideas that were implied by the matches of the Magic Number.

3) Automatic detection of the manufacturer ID

Another investigation that was performed was more usability oriented, though it stemmed from the fact that the hexadecimal OUI numbers were not known for all manufacturers. Hence, an attempt was made to parse the private section file and determine the manufacturer ID without knowing what it actually was. The way this was attempted was, each time the MIS descriptor was found, the bytes that should have been the manufacturer ID were checked, and every unique pattern found was assigned a separate list. At the end, the total number of occurrences for each unique pattern was calculated. The one with the maximum number of occurrences should have been the manufacturer ID. However, when this algorithm was implemented and run on the available private section files, the pattern it gave was not the manufacturer ID. This would imply that the MIS descriptor alone is not enough to distinguish the manufacturer ID for automatic detection purposes.

This approach could still work if additional information on the structure of the complete DSI was available, not just the MIS portion. At least another unique descriptor at a known distance from the manufacturer ID would be sufficient to successfully implement this approach.

3.2.4 Effects of Findings

The results of these investigations meant that the file could not be extracted from the stream, especially since it would not be obtainable as a private section file, rather it would be extracted as the sections themselves. As the structure of the carousels is not completely known, it would not be possible to design a system which would extract the file without using an existing API which can access it.

Hence, after discussions with the BBC the fourth level of the requirements, regarding the file extraction and CRC check, was changed. Instead, it was decided that the DSI should be monitored alone, and that a check should be included to track if it was being broadcasted every 5 seconds as specified in [11].
3.3 System Design

Though a basic system design was being worked on near the end of the analysis phase, the design was changed according to the new requirements. This section describes the resulting system design.

3.3.1 Platform and language used

The implementation of the system was done using the Java programming language. Despite the fact that currently the Digital TV card drivers for Linux alone can provide the Engineering Channel stream, the software has been designed to be platform independent. Also, the client requested that the main processing units of the software be designed in terms of an API/library. Hence, the choice of Java was made.

3.3.2 Overall architecture

The system consists of a personal computer running the Linux operating system, equipped with a WinTV digital TV receiver card. An antenna is used to catch the terrestrial signal, which is then transmitted to the card. The drivers for the card enable the extraction of the transport stream packets. This extraction can be restricted to a single program id, hence allowing the extraction of the engineering channel data in the form of a transport stream. This stream is to be processed by the OAD Monitoring Software which logs the occurrence of the Manufacturer ID. The following figure shows this complete sequence:

![Fig.3.3: General Architecture of the Monitoring System](image)

The OAD Monitoring System

The structure of the OAD Monitoring system is given below:
Fig 3.4. Software Structure For The OAD Monitor

The processing of the stream needs to be done in the following manner:

Find the sync byte from which processing should start – this can be done by processing the first three packets to ensure that the wrong sync byte sequence is not taken.

The software should first strip off the transport stream headers from each packet, and fill a buffer with the payload alone.

The search for the MIS descriptor and the manufacturer ID can be done on the data in this buffer. Whenever the Manufacturer ID is found, the time of detection should be logged. Also, the time difference from the previous occurrence should be written to the log.

According to the revised specifications, the DSI/MIS should be found at maximum intervals of 5 seconds. Hence, the time when the last manufacturer ID was found should be stored, and every 5 seconds thereafter an error should be logged if the manufacturer ID is not found. But, the CRC check has been included in the design, so that it can be implemented easily once the exact structure of the Transport Stream’s payload is known.

This satisfies upto level 4 of the requirement specification. The output log format can be converted to a defined XML format, once available, to satisfy level 5. This can be done within the println() method of the Util class, which is shown in the next section.
3.3.3 Class design

The system consists of 3 classes which form a package, and a driver class. The classes follow the structure given below:

![Class Diagram](image)

**The packaged classes**

1) **StreamProcessor**: This is the main processing class. It contains two methods:
   a. `StartProcessing()`: Contains the complete logic for processing the DVB stream.
   b. `Read()`: A method to read from a given stream while incorporating a timeout for the read operation.

2) **Util**: This class contains a number of static methods. These are
   a. `hexToDec()`: Convert a hexadecimal number to decimal
   b. `getTwosComplement()`: Get the two’s complement of an integer
   c. `hexToDecTwos()`: Combines the above two methods.
   d. `println()`: Facilitate writing to standard output as well as a `Writer` object. It also takes a parameter to disable writing to standard output.

3) **OADMonitorException**: A sub-class of `Exception`, it is used to differentiate any known exceptions arising in accordance with the high-level logic of the program.

**The driver class - OADMonitorDriver**

This class consists of only a main method, and some package level variables. It is responsible for taking the input parameters, instantiating a `StreamProcessor` object, and then passing the parameters to the `StreamProcessor` object’s `startProcessing()` method. It is also responsible for the validation of the properties and command line options specified.
3.3.4 Activity Flow

The starting point of the software is the *OADMonitorDriver* class. After validations and determining the DVB stream, source private section file and log file, it calls the *StreamProcessor* class with these as arguments. The *StreamProcessor* class then examines the DVB stream for the MIS descriptor and Manufacturer ID, and logs occurrences as well as errors.

**Flow of the Driver class**

The flow of the driver class is outlined in the diagram below:

---

It first takes the command line arguments and validates them. If they are invalid it terminates. Otherwise, it goes on to loading the parameters such as manufacturer ID, private section source file, maximum DSI Interval etc., from a properties file. If any of these are invalid, it terminates. Else, it opens the DVB stream and the private section file for reading, and the log file for writing. If it fails to do so, it terminates, otherwise it writes the parameter information to the log file. Next, it calls the *StreamProcessor* class’ *startProcessing()* method, to process the stream. Finally, it closes all streams and exits.
Flow of the StreamProcessor class

The flow of the driver class is outlined in the diagram below:

Fig. 3.7: Activity Diagram for the StreamProcessor class’ startProcessing() method
The StreamProcessor’s startProcessing() method is called by the OADMonitorDriver class. The startProcessing() algorithm can be subdivided into five sequential stages.

1. Check for the DVB stream

It first opens the DVB stream to check if there are any sync bytes present in the first 3 packets of the stream. If there are any, it stores the positions of these bytes and then determines which of them is the first most likely sync byte. It does this by analysing the distance between the sync bytes, and which of them fit the 188-byte gap pattern. If there is no data at all, it exits with a message saying there is no data in the stream. If no sync bytes are found, it exits with a message saying that no DVB Transport stream was found.

2. Prepare the buffers for processing

Once the first sync byte is determined, the occurrence of the DVB stream is written to the log. The method now reads 188 bytes of data, using the read() method of the class into a buffer called the TS buffer which would be of size 188 bytes. It then initialises a buffer of size 5 bytes called the TS End buffer, with zeroes, and sets all time variables to the current time. The TS Join buffer is also initialised at this point. The TS Join buffer shall be used for actual processing, as it takes into account the last 5 bytes of the packet that are usually left unprocessed (These bytes are taken from the TS End buffer).

3. Process the data

Next, it loops while the Waiting Time is less than the Maximum Waiting Period, else it goes to step (5). Waiting time is defined as the time since the last occurrence of the ID was found.

If the last read from the DVB Stream was successful, it proceeds to fill the TS Join buffer with the 5 bytes from the TS End buffer. It fills the remaining part of the TS Join buffer with the 179 bytes of the TS buffer following the first 4 bytes (which comprise the MPEG Header).

Next, it examines each of the bytes in the TS Join buffer, and checks for the MIS descriptor and Manufacturer ID, which should be separated by 3 bytes. If it finds them, it logs the occurrence with a timestamp, and stores the time of occurrence. The CRC check buffer can be filled at this time, with the bytes in the TS buffer. At the end of the loop the last 5 bytes are left unprocessed, hence these are copied into the TS End buffer. Now, if the CRC check buffer is full, the CRC check can be performed (This is provided the CRC frame size is known and does not vary for different files).

4. Get the next packet of data
If there were no bytes available from the DVB Stream, or if the data has been processed, the next read is performed. The Waiting Time is now re-calculated. If it has exceeded the Maximum DSI Interval and the time since the last check is greater than the Maximum DSI Interval, it logs it as an error. Next, it proceeds as according to step (3).

5. Log Termination and Exit

Reaching this step signifies that either no more data is available or the DSI could not be found for the Maximum Waiting Period. Hence, termination is logged and it exits.
**Log file format design**

The log format was designed to give the most relevant information in a usable form. Its structure is given in the following diagram:

![Figure 3.8: Log File Format](image)

- **Starting Time Stamp**
- **Initialisation Parameters**
  - Manufacturer ID
  - Private Section File
  - DVB Stream File
  - Log File
  - DSI Interval Tolerance
  - Maximum Timeout

- **Start of Stream Processing**
  - Found a DVB stream
    - Manufacturer ID Timestamp. Time difference from previous.
    - Manufacture ID not found in the last X seconds.
    - Manufacture ID not found in the last X seconds. Terminating Processing of the stream.
  - DVB Stream not found.

- **Time Taken**
- **Bytes Read**
- **Stream Processing Ended**
- **Cause of Termination**
4. Implementation

4.1 Differences from the design

As the design and implementation were done almost in parallel, there are no major differences in the implementation. However, though the design did not specifically exclude the option of running the drivers from within the program, the implementation does not permit this. This is due to unreliable performance of this, especially on the Windows platform [18].

4.2 Implementation Specific Details

The implementation follows the design specification of creating a package called the OADMonitor which is comprised of the StreamProcessor class and its helper classes. The driver program for this class is kept separate from the package.

Some important implementation methods for algorithms and other issues faced are mentioned here, class-wise.

OADMonitorDriver

This class consists of only a main() method, as it’s primary purpose is to act as a driver for the StreamProcessor class. Hence, it is mainly responsible for parameter validation, and parameters are taken from a Properties file as well as the command line. Regular expressions were used for the validation of the manufacturer ID and timeout intervals. If any of the parameters are invalid, the program terminates immediately.

A major amount of Exception handling was required as many I/O operations are performed either by the class itself or the StreamProcessor class. Especially in the case of an exception related to the logfile, execution terminated immediately, as the OAD Monitor’s primary aim is continuous logging. Hence disk space requirements should be taken into account before installation.

A large number of nested try, catch and finally blocks were used, which seem unnecessary at first, but are essential to ensure proper closing of all the streams as well as the logging/recording of all events/errors that take place.
StreamProcessor

This is the primary class of the system, providing all the functionality required to process a given stream.

The StreamProcessor() method is the most important method of the system. It is responsible for all the actual processing of the stream. It is implemented exactly according to the design. Currently it only checks for the DSI within the stream, and also whether the DSI is being broadcasted at the specified interval or not. The value of this interval can be changed within the properties. Though the CRC check has not been implemented, a way of implementing it has been mentioned within the code itself, as well as where it should be placed. The initial implementation did not include the re-buffering with the TS_join and TS_end buffers nor did it include the marking and resetting of the stream during the search for the sync-byte. These problems were found during individual testing of the classes and resulted in additions to the design, which has been shown in the design section.

The read() method has been implemented such that a timeout can be included for a read operation on a given stream. This is currently enforced by trying to read the buffered input stream over and over again, until the time since the first read has exceeded the timeout or the specified amount of data has been retrieved.

Util

This class currently only has static methods and hence should be an abstract class. However, to provide flexibility for future thread-based implementations of the package, it has been given a constructor and should be instantiable. Thread-based implementations which pass stream objects and perform I/O operations, such as in the println class which prints to a Writer object, would require locking mechanisms and hence static methods might not be thread-safe.

The println() method takes an integer parameter which can be used to indicate different writing options, such as which output to write to. In a future implementation, this can be used for XML formatting and writing the output in different XML tags suggested by the number. The println() method throws an OADMonitorException when an IOException is thrown within the method’s execution. The reason for this was that the method assumes that there are no I/O permission
problems, as this should have been taken care of by the driver program. The only case in which an exception would be thrown would be that the disk is full, and the writing to the log cannot take place.

The `hexToDec()` method converts the hexadecimal numbers in pairs of two, since that is equivalent to a byte. Hence it returns an array of integers each representing a pair of hexadecimal digits.

The `getTwosComplement()` method takes an array of integers and returns an array of twos complements for the same. It does not specifically calculate the twos complement, but instead casts the integer to byte, and stores the resulting value in an integer.

**OADMonitorException**

This class extends the `Exception` class and is only used to differentiate any specific error occurrence in terms of the processing logic for the file, hence it only has constructors which call the `super()` method.

### 4.3 Packages and Java classes used

For the implementation, only standard Java (version 1.4) packages and classes were used. A list of the packages and their respective classes which were used follows. The classes used from `java.lang` have not been mentioned.

<table>
<thead>
<tr>
<th>Package Name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java.util</td>
<td>Vector</td>
</tr>
<tr>
<td></td>
<td>Properties</td>
</tr>
<tr>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>java.io</td>
<td>BufferedInputStream</td>
</tr>
<tr>
<td></td>
<td>FileWriter</td>
</tr>
<tr>
<td></td>
<td>FileInputStream</td>
</tr>
<tr>
<td></td>
<td>InputStream</td>
</tr>
<tr>
<td></td>
<td>Writer</td>
</tr>
<tr>
<td>java.text</td>
<td>DateFormat</td>
</tr>
<tr>
<td>(not imported)</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.1: Packages and Classes Used*
4.4 Final class structure

The final class structure for the implementation follows the one outlined in the design section, but is in more detail. It is shown in the following diagram.

![Class Diagram for the Implementation](image-url)

Fig 4.1: Class Diagram for the Implementation
4.5 Screenshots

Below are some screenshots of the system. The screenshots for the off-air stream show that no data is available, as there was no download being aired when the screenshots were taken.

Normal Mode
This is the mode in which all the information written to the log is also displayed in the console window.

---

Fig 4.2 a) Screenshot: Empty Stream

Fig 4.2 b) Screenshot: Non-engineering channel live DVB Stream
Silent Mode

In this mode, all information is written to the log, and none to the console window. The only exceptions are if there is a parameter validation error, or the program has to terminate unexpectedly. This does not include the case where the DSI transmission stops during processing. As the software would be run as a cron job, it would be run in this mode most of the time.

![Fig 4.3 a) Screenshot: Emulation of faulty DVB stream](image1)

![Fig 4.3 b) Screenshot: Non-Engineering Channel, live DVB Stream](image2)
5. Testing and Evaluation

5.1 System Testing

As the system is essentially a simple system with very few classes and functions, the testing is limited to checking system behaviour under the various possible circumstances of the stream and also checking input parameters and options. Two tests for checking real-time behaviour were also performed.

5.1.1 Testing against various stream characteristics

A number of cases have been identified with respect to the kinds of stream behaviour that are possible. Some of these cases include the stream being the correct stream, the correct stream with incorrect DSI intervals, and a stream with no data at all. All tests were successful.

1) Perfect DVB Engineering Channel stream:
The system was tested on a currently broadcast stream. The DSI interval was approximately 0.8 seconds. It logged the occurrences successfully and terminated once it could not find the DSI (The stream was terminated).

2) DVB Engineering Channel stream with DSI interval greater than permissible:
The system had been tested earlier with a broadcast stream with a DSI of 20 seconds. As the maximum DSI interval was set to 5 seconds, it logged 3 errors for the 15 seconds that the DSI was not present, and then logged the occurrence of the DSI. This pattern continued until the stream terminated.

3) DVB Engineering Channel stream terminating during processing:
This test was done when a known download was about to terminate according to schedule. The DSI interval for this download was abnormal, and the broadcast terminated at 9:03 a.m. The log showed that 3 DSI errors were logged for every successful detection. This was in accordance with the DSI interval tolerance being set to 5, and the DSI coming every 20 seconds. After the broadcast terminated, 4 more errors were logged as the DSI could not be found. The system terminated after
25 seconds of not finding the DSI, in accordance with the Maximum Waiting Period being set to 25 seconds.

4) Non-engineering channel stream:
The system was given a DVB stream, but it was the video stream for BBC1, and not the Engineering Channel. The system started normally and identified it as a DVB stream. Then it checked at every Maximum DSI Interval time, and upon not finding the DSI, logged it as an error. Once the maximum waiting period was reached, the system terminated saying that the DSI could not be found.

5) Non-DVB stream:
Since the hardware card only provides a DVB stream, it was decided to use a standard file as the stream. The system started normally, and upon not finding any sync bytes in the stream, it displayed a message signifying that this was not a DVB stream. It exited giving the cause of termination as “DVB Transport Stream not found”.

6) Dump file:
This mode is used for testing whether the system is installed properly, even if there is no means to catch the DVB stream directly off air. As reading from a file is much faster than catching a stream coming at 50 Kbps, the values for the DSI interval tolerance and the Maximum Waiting Period have to be set to very low values of approximately 10 and 20 milliseconds respectively. The system successfully logged all occurrences of the DSI, and exited when the end of the file was encountered as no more occurrences could be found.

7) Stream without data:
The hardware card was given a command to give all the packets on a non-existent stream. As the hardware card did not produce any data, the system terminated giving the cause as “No data in the stream”.

5.1.2 Testing the real-time nature of the system

One of the project objectives was to determine whether a real-time system could be designed or not. Hence, it became necessary to test whether the current system can process the stream in real-time. Two methods were devised for this purpose. All tests were successful.

1) Comparing dump size vs. Bytes processed

The first method to test the real-time nature was to compare the size of the dump that was produced in \( n \) seconds from the hardware card, to the number of bytes processed by the system in the same \( n \) seconds. Three trials were conducted with various times.

For \( n = 2 \):
Both the dump size (in bytes) and the number of bytes processed were equal to “11844”.

For \( n = 100 \):
Both the dump size (in bytes) and the number of bytes processed were equal to “623784”.

For \( n = 500 \):
Both the dump size (in bytes) and the number of bytes processed were equal to “3124184”.

2) Bytes available in the stream

The second method for testing the real-time nature was to test the number of bytes available on every read that the system tried to perform. The hardware card delivers packets of 188 bytes at a time. This implies that if the maximum number of bytes available at any time is 188, then there are no packets being queued up. This would further imply that the system is running in real-time.

The system was tested twice in this manner, and each time the number of bytes available was either 0 or 188 bytes. The number of times that it was 0 far exceeded that when it was 188. This would imply that not only is the system running in real-time, but also it would be possible to include some more processing without losing the real-time characteristics of the system.
The graphs below show the behaviour for attempted read operations on the stream, and the number of times it was successful in retrieving data.

**Fig 5.1 Graph for 350,000 read attempts**

This graph is for 350,000 read attempts. As can be seen, it follows a fairly regular pattern. The initial empty reads are less as the time taken to process the most likely sync byte is a considerable amount, since it involves a number of reads as well as a major amount of vector manipulations in Java. After that, it is fairly regular till the end where there is no stream present. The program exits after the maximum timeout period.

**Fig 5.2 Graph for 15,000 read attempts**
This graph shows the first 15,000 read attempts. Here it shows that there are around 2,000 empty reads despite the vector manipulations. The next set of empty reads is about 4,500 and then the regular gap of 8,000 starts appearing. This implies that the CPU is relatively free, and there is scope for further processing before the real-time nature is lost.

5.1.3 Incorrect parameters/options and I/O permissions

Apart from errors which may be present in the main logic of the system, any system which takes parameters has to be tested for its behaviour when incorrect parameters are given to it. The relevant tests are given below. In all three cases, processing depends on the correctness of these parameters. Hence, program execution must stop when such an error is encountered. Yet, the program must be able to identify the problem and inform the user before exiting, so that it can be rectified by the user.

Input/Output errors

As the system deals with files and streams, the main errors that can occur would be I/O errors. The possible errors identified and tested are given below.

1) Source file: In the case of the source file, the file may exist, may not exist, or may exist but not have read permissions.
   a. File exists: This is the normal case.
      The system ran without reporting any error and created the log-file.
   b. File does not exist: A non-existent filename was entered.
      The system gave the message “OADMonitorDriver error - oad65c.sec (No such file or directory)” followed by a termination message. It terminated thereafter.
   c. File does not have read-permissions: The file’s read permissions were removed.
      The system gave the message “OADMonitorDriver error - oad65c.sec (Permission denied)” followed by a termination message. It terminated thereafter.

2) DVB stream file: In the case of the DVB stream, if a file is specified, it may exist, may not exist, or may exist but not have read permissions.
   a. File exists: This is the normal case.
      The system ran without reporting any error and created the log-file.
   b. File does not exist: A non-existent filename was entered.
The system gave the message “OADMonitorDriver error - novapal_03_08.mpg (No such file or directory)” followed by a termination message. It terminated thereafter.

c. **File does not have read-permissions:** The file’s read permissions were removed.
   
   The system gave the message “OADMonitorDriver error - novapal_03_08.mpg (Permission denied)” followed by a termination message. It terminated thereafter.

3) **Log file:** In the case of the source file, the file may exist, may not exist, or may exist but not have write permissions.

a. **File exists:** This is the normal case.
   
   The system ran without reporting any error and created the log-file.

b. **File does not exist:** A non-existent filename was entered.
   
   The system created a new file with the given name and ran successfully.

c. **File does not have write-permissions:** The file’s write permissions were removed.
   
   The system gave the message “OADMonitorDriver error - OAD_24_08_1.log (Permission denied)” followed by a termination message. It terminated thereafter.

**Incorrect entries in the properties file**

Three entries exist in the properties file which have to be of a defined length and consist of limited characters/digits. On encountering any anomaly, the system should identify the anomaly, report it to the user, and exit after giving a termination message. All tests were successful.

1) **Manufacturer ID** – The manufacturer ID is a 6 digit hexadecimal number. The three possible cases of incorrect input are as follows:

a. **More than 6 digits:** A 7 digit number “000AFES” was entered into the properties file.
   
   The system displayed the message “Invalid manufacturer ID - Incorrect length” followed by a termination message. It terminated thereafter.

b. **Less than 6 digits:** A 5 digit number “000AF” was entered.
   
   The system displayed the message “Invalid manufacturer ID - Incorrect length” followed by a termination message. It terminated thereafter.

c. **Non-hex characters:** A string containing a non-hexadecimal character ‘G’ was entered, “000AFG”.
   
   The system displayed the message “Invalid manufacturer ID - Illegal characters.” followed by a termination message. It terminated thereafter.

2) **Maximum DSI Interval** – The maximum DSI interval is taken to be a 2-9 digit positive number.
   
   The three possible cases of incorrect input were:
a. **More than 9 digits**: A 10 digit number “1234567890” was entered. The system displayed the message “Invalid maximum DSI Interval. It has to be from 2-9 digits.” followed by a termination message. It terminated thereafter.

b. **Less than 9 digits**: A single digit number “9” was entered. The system displayed the message “Invalid maximum DSI Interval. It has to be from 2-9 digits.” followed by a termination message. It terminated thereafter.

c. **Non-numeric characters**: A non-numeric string “x1000” was entered. The system displayed the message “Invalid maximum DSI Interval - Illegal characters.” followed by a termination message. It terminated thereafter.

3) **Maximum Waiting Period** – Similar to the previous, this property is also taken to be a 2-9 digit positive number. Hence the incorrect inputs tested were:

a. **More than 9 digits**: A 10 digit number “0123456789” was entered. The system displayed the message “Invalid maximum waiting period. It has to be from 2-9 digits.” followed by a termination message. It terminated thereafter.

b. **Less than 9 digits**: The property value was left blank. The system displayed the message “Invalid maximum waiting period. It has to be from 2-9 digits.” followed by a termination message. It terminated thereafter.

c. **Non-numeric characters**: A non-numeric string “2000!A” was entered. The system displayed the message “Invalid maximum waiting period - Illegal characters.” followed by a termination message. It terminated thereafter.

---

**Command-line switches**

The command line switches are of the format “–character”, for example “-a”. There are only two acceptable switches, “-a” and “-s”. The first tests were to determine if the system behaviour was correct according to the switches. Next, the behaviour when incorrect switches were entered was tested. *All tests were successful.*

**Behaviour with correct switches**

1) **“-a”:** This switch is used to make the program append the log in case a log file exists, instead of re-writing it. Output was tested with and without the switch.

   a. With the switch:
The log-file was deleted before running the system, and then the system was run in append mode. It successfully created the log-file. The system was run again with the switch, and the resulting log-file had both the previous entries as well as the new ones.

b. Without the switch:
The system was run without the switch after the previous test. The log-file thus obtained only contained the new entries.

2) “-s”: This switch is used to force the program into “silent” mode. This means that nothing is written to standard output, all the information is written to the log-file only. Again, output was tested with and without the switch.

a. With the switch:
On running the system with the switch on, only the messages from the drivers of the hardware card appeared, none appeared from the program itself.

b. Without the switch:
When the system was run without the switch, all the log file messages were also displayed on standard output, which is the console terminal.

Behaviour with incorrect switches (3 cases):

In all three cases, the behaviour should be the same. The system should inform the user that an incorrect switch has been used, and the exit.

1) **Missing hyphen**: The append switch was used, but without the hyphen.
The system displayed the correct usage of the switches and then exited.

2) **More than one character**: The word “test” was given as an argument.
Again, the displayed the correct usage of the switches and then exited.

3) **Invalid option**: The switch “-x” was given as an argument.
The system displayed the correct usage of the switches and then exited.
5.2 Evaluation

A majority of the evaluation has been discussed with the client and concurrence has been reached on most of the points raised. On the whole the client was very satisfied with almost every aspect of the project, as can be seen in their evaluation in Appendix I. Points which were not raised by the client are specifically mentioned as the author’s own.

5.2.1 Based on Objectives

As mentioned before, there were three main project objectives:

1) Understanding the DVB-T standard

A large amount of research went into satisfying this objective, and the proof of success is the identification of the difference in structure between the actual implementation of the DVB stream and private section files versus the structure specified in [10] and [11]. Apart from a broad understanding of the standard in terms of the complete broadcast mechanism, an in-depth knowledge was gained into the workings of the Object and Data carousels, as well as the messages such as DSI which are required for the firmware download. Questions that arose based on this research were posed to the client so as to get a better perspective on the problem, as mentioned in the client’s evaluation.

2) Design and development of a demonstrator

This was in executed in five levels of implementation as given in the requirements from the client out of which one was changed. These are listed in the introduction itself. The change to level four was specified in the analysis section under “Effects of Findings”.

The demonstrator completely satisfied the first three objectives, and rigorous testing was done on all DVB-T Engineering Channel downloads which were adhering to the specifications and proved successful.

For the fourth level, the demonstrator was successfully able to detect if the DSI was encountered every 5 seconds or not, with the additional flexibility of being able to change this number. This flexibility was included when trials suggested that the DSI broadcast was not following the rule and
could be at intervals of up to 20 seconds. It also reports the time difference since the last detection, giving a quicker recognition of any problems.

As far as the original requirements for the fourth level were required, the design took them into consideration and includes them. The portion of code and the class and method within which the CRC check should be implemented is outlined in the design itself, and the source code contains comments as well as two possible methods of implementing it. Given the knowledge of the structure of the stream, it should be possible to easily implement the CRC check without major difficulties. Hence, both the new and original requirements are met to a satisfactory level.

For the final level, the design does not include a specific XML processing method as the only requirement is to generate XML. Since the XML is required only for logging, there is a provision of “levels” of logging within the method used for writing to the log. These levels can be interpreted to mean different XML wrappings around the text to be written, and only the case structure within the method needs to be modified. Currently these levels are used to distinguish between silent and verbose logging mode. As the XML format was not provided within time by the client, the XML levels were not implemented in the demonstrator. Hence, this level also has been achieved to a satisfactory degree given the information supplied.

3) Identification of requirements for a real-time system

The system already performs in real-time. Between each successful read, it was found that there were a large amount of empty reads, suggesting that the system is idle for reasonable lengths of time. Fig 5.3 in the Testing section illustrates this, and it indicates that the CRC Check should be implementable in real-time depending on the CRC frame size, given that there are 2,500 empty reads even during large vector manipulations. Hence, the third objective has also been met satisfactorily.

5.2.2 Existing Products

Two commercial products were found that deal with the Carousel Structure of the DVB stream. Though not a lot of information is available, neither product seems to fit in completely with the client’s requirements.
DTV Interactive – ITV Stream Analyzer

This product [16] is said to be a real-time monitor for DVB streams that deals with the Object and Data Carousel also. The minimal information available suggests that it is capable of “DVB SI Data and Object Carousel Analysis” and “Dynamic Object Carousel Analysis”, but there is no explanation as to what the vendors mean by these terms. It also mentions “BIOP, DSI, DII, DDB and DSMCC section display and interpretation”, which would imply that it is capable of examining the structure of the carousels and extracting the private sections. Yet, it is inconclusive if it can perform the tasks as required by the client, and there is no mention of customisability. Hence, this product warrants further investigation, but as yet cannot be recommended as an alternative solution.

Tektronix Products

There were two products [17] found which relate to the DVB-T standard.

First, the DVB Carousel Generator, which is able to generate Object and DSMCC carousels. This product is unsuitable for the client however, as it only generates a stream and does not process or analyse it. Secondly, it does not support data carousels.

The second product was the MTM-400 MPEG Transport Stream Monitor. Though it is capable of processing and analysing an incoming DVB stream in real time, there is no mention of the carousel being analysed. It performs error-checking, but it can only be surmised that it does this using the CRC present in the stream. Moreover, there is no evidence suggesting that it can deal with the problem of monitoring or logging file downloads in a carousel.

Hence, this project is still relevant and has potential in terms of what was achieved and what can further be achieved in terms of the monitoring of the Engineering Channel. It is highly focussed on the client’s requirements, and is the first and a very viable step towards a complete monitoring system for the firmware downloads.
5.2.3 Deficiencies and Possible Improvements

Although the project has satisfied the requirements overall, and surpassed the minimum requirements while being a flexible design, there are as always a number of issues that it suffers from. Also, some enhancements can be made, which have not been implemented due to paucity of time. A second cycle of design and implementation may have led to a number of improvements in the product.

In terms of design, in the current implementation the StreamProcessing class’ startProcessing method contained all the logic for the processing. As the design had split the logic into a number of stages, it could be implemented such that different methods are employed for different stages. Though the client has not mentioned any problems with the current design, splitting into different functions would benefit in terms of making the API more flexible in terms of re-usability.

In terms of functionality, the manufacturer ID had to be set manually for each different manufacturer’s file that may be played out. Even though the automatic detection of the ID failed, it is still possible to have a lookup table in a text file, which can be updated. All IDs could be matched, and the system would be able to tell which download is coming. It would make it more convenient to use and also help in easily determining if the wrong file is being broadcast. This was mentioned specifically by the client.

A point raised by the client during the demonstration was the use of a logging library called Log4J. This had already been considered during the design an implementation, but was not used as the error levels defined in the API were more suitable for debugging purposes and not easily customisable for the current system’s requirements.

As mentioned in the client’s evaluation, the only aspect of the project which could have been improved upon would be the project management in terms of following up on requests for information, and putting slightly greater pressure on the party responsible for providing the same. This would have helped in speeding up the communication process, hence providing a little more time for implementation details that may have been left out and faster resolution of issues.

Other future enhancements to the system have been discussed in the section on future work.
6. Project Management

6.1 Initial Project Plan

The project has been divided into separate tasks/stages. These are:

1) Revision and further understanding of the streams and carousel structure:
The first task will involve revising the structure and working of the DVB-T transport stream, the carousels and the data stream.

2) Familiarisation with the software and hardware:
The hardware and software installed should be tested, and extraction of the streams using the software and the parameters associated with it should be well understood.

3) Analysis and design of the solution
Based on an analysis of the information gained so far, a design for the software needs to be created, to ensure its validity and feasibility.

4) Implementation
An implementation based on the above design should be made. It will be in the phases as given in the BBC specification, and development shall be stopped at the end of the allocated time.

5) Testing and Evaluation
A test plan shall be made and the system thoroughly tested and evaluated according to the same. The evaluation would include if the bare minimum requirements or higher, were met, and reasons and comments for the same.

6) Write-up
The creation of the final report would be an on-going process, but actual writing and refinements will be done at the end of the project.

The following chart shows the weeks allotted to each of these stages:

<table>
<thead>
<tr>
<th>S. No</th>
<th>Task/Stage</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Week 8</th>
<th>Week 9</th>
<th>Week 10</th>
<th>Week 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Revision and Further</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Understanding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Familiarisation with H/W &amp;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Analysis and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Testing and Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Write-up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Initial Project Plan
Some part of analysis, design and testing might overlap with implementation, as the implementation is done in stages. The total number of weeks available is twelve, hence one week has been left for contingency purposes.

Milestones Chart

Some milestones were set to ensure that certain tangible output from the project could be obtained from time to time.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Name/Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design specification to customer</td>
<td>30th June, 2003</td>
</tr>
<tr>
<td>2</td>
<td>Mutually approve final design specification</td>
<td>7th July, 2003</td>
</tr>
<tr>
<td>3</td>
<td>Finalise Demonstrator</td>
<td>22nd August, 2003</td>
</tr>
<tr>
<td>4</td>
<td>Arrange demonstration date with the customer</td>
<td>2nd September, 2003</td>
</tr>
<tr>
<td>5</td>
<td>Give the demonstration and submit report</td>
<td>5th September, 2003</td>
</tr>
</tbody>
</table>

Table 6.2: Initial Milestones Chart

A project log was kept to keep track of progress, available at:
http://www.personal.leeds.ac.uk/~scs2ub/project/log.html

6.2 Revised Project Plan

Though the initial plan had less time devoted to analysis and more to implementation, the final outcome has been quite the opposite.

6.2.1 Reasons for revision

The project suffered from a number of delays, the first one being the installation of the Digital TV card. It took three weeks before it was operational on the 1st of July. This used up the first two weeks, which were allocated to it, and also the contingency week. Actual familiarisation with the hardware and software began two weeks later than it should have.

The next major hurdle was that the analysis phase went on for five weeks, as the investigations of the stream structure took up a large portion of the available time, along with the delays in communication between both parties. However, informal implementation and design was begun during the latter half of this phase, as parts of the code written for the investigations were usable for actual implementation. This saved some of the implementation time at the end, and enabled the
project to be completed within time despite the delays. However, time-pressure was present to a great degree and resulted in a few hitches in implementation decisions. Testing and evaluation, however, were given the same amount of time as in the initial plan, to make sure that the system was well validated. Finally, the handover of the project to the client was done well within time, so a good recovery was made overall.

6.2.2 Final Plan

<table>
<thead>
<tr>
<th>S. No</th>
<th>Task/Stage</th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Week 8</th>
<th>Week 9</th>
<th>Week 10</th>
<th>Week 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Revision and Further Understanding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Familiarisation with H/W &amp; S/W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Analysis and Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Testing and Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Write-up</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3: Final Project Plan

Note: The contingency week was used up in the initial three-week delay.

Final Milestones

<table>
<thead>
<tr>
<th>S. No</th>
<th>Name/Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial Design</td>
<td>7th August, 2003</td>
</tr>
<tr>
<td>2</td>
<td>Mutually agreed design specification changes</td>
<td>13th August, 2003</td>
</tr>
<tr>
<td>3</td>
<td>Freeze Implementation</td>
<td>15th August, 2003</td>
</tr>
<tr>
<td>4</td>
<td>Finalise Demonstrator</td>
<td>20th August, 2003</td>
</tr>
<tr>
<td>5</td>
<td>Arrange demonstration date with the customer</td>
<td>22nd August, 2003</td>
</tr>
<tr>
<td>6</td>
<td>Demonstration and hand-over to customer</td>
<td>29th August, 2003</td>
</tr>
<tr>
<td>7</td>
<td>Submit report</td>
<td>5th September, 2003</td>
</tr>
</tbody>
</table>

Table 6.4: Final Milestones
7. Conclusions and Future Work

7.1 Conclusion

The project was originally specified to develop a monitoring and logging system for a data download within a DVB-T stream, including file extraction and CRC checking. In-depth investigations into the stream resulted in the findings that such extraction could not be performed, as the stream specification is too loosely defined. A limited monitoring system was developed, but with scope for including major error checking once the structures are known.

7.2 Possibilities for Future Work

First, identifying the exact structure of the streams. As one of the original objectives of the project could not be met within the time specified, this is probably the most immediate work to be done. For this purpose, further documentation would be required from Research and Development at the BBC. Also, developments in the currently existing APIs, such as Java TV and the MHP specification, should be kept track of.

Once the previous step is accomplished, the error-checking for the extracted file can be implemented. The error levels can be further differentiated according to DSI errors, bit errors or CRC errors and further logging options can be provided as the error differentiation increases.

Another offshoot of this project could be to explore the possibility of monitoring streams of a higher data rate. The present 50 kbps download rate is quite conducive to this, but monitoring of audio and video signals in real-time represents a far greater challenge.

A major enhancement to the system would be the implementation of a threaded model. Separate threads handling the data buffering, DSI monitoring and CRC-checking could increase the efficiency of the system. However, locking and synchronisation methods would be of prime importance in this case. A broad outline for a system incorporating the above-mentioned improvements is described below.
Proposed Outline of an enhanced OAD Monitoring System

The threaded model expands the capabilities of the system. A proposed design would have one main thread for managing the system. Three other threads would be as follows. The first thread, which we shall name the DSI thread, would read from the stream and perform the DSI check. It would also continuously feed a buffer of the same length as the CRC frame size, and notify the second thread once it was full. The second thread (CRC thread) would then put a lock on this buffer and copy the data and empty the buffer. It would then unlock this buffer for use by the DSI thread. Once the CRC thread has the data, it can now perform a CRC check on it, and log any errors.

The third thread, which we shall name the graph thread, shall share two buffers, one with the DSI thread and another with the CRC thread. The DSI-graph buffer would be filled by the DSI thread with the number of DSI errors every minute. Similarly, the CRC-graph buffer would be filled by the CRC thread with the number of CRC errors every minute. Now, the graph thread can use these buffers to give an hourly or daily report on DSI and CRC error rates. Assuming an hourly report, it would do this by putting a lock on the DSI-graph buffer and copying off the data, and then erasing the buffer and unlocking it for re-filling. It would then use the obtained data and plot a graph for the number of DSI errors versus the number of minutes, for that hour. This graph can easily be created by using currently available Java packages. Next the graph could be e-mailed to the responsible personnel as an hourly report. A similar procedure can be followed for the CRC graph.

This kind of reporting would definitely be beneficial for any monitoring system which has such a high priority, and would be much easier to interpret and analyse for statistical purposes.
References

8) Background Information on DSM-CC,
   Last reviewed: 29 Apr. 2003
11) Thomas G. (1998) DTV URI Scheme
    Last reviewed: 30 Jun. 2003
12) The Java TV Technology Home Page
    http://java.sun.com/products/javatv/
    Last reviewed: 30 Jun. 2003
    Last reviewed: 30 Jun. 2003
14) DVB-MHP – Technical Essentials
    http://www.mhp.org/technical_essen/index.html
    Last reviewed: 30 Jun. 2003
    http://www.mhp-interactive.org/tutorial/mhp/filesystems.shtml
    Last reviewed: 2 Sep. 2003
16) ITV Analyzer: MPEG2 DSMCC Object Carousel MHP Application
    http://www.dtvinteractive.com/mhpanalyser.html
    Last reviewed: 3 Sep. 2003
17) Tektronix Video Test
    Last reviewed: 3 Sep. 2003
18) Process (Java 2 Platform SE v1.4.1)
    http://java.sun.com/j2se/1.4.1/docs/api/java/lang/Process.html
    Last reviewed: 4 Sep. 2003
Appendix A – Personal Reflections

Any project undertaken results in not only learning new technologies and methods, but also more personal issues such as communication with people, understanding of time management and ways to tackle new and unknown problems.

This project had its fair share of the same, especially in terms of communication. Given that I have had experience in working with geographically distant teams, I should have been able to better understand the importance of communication, and the issues involved with it. In such situations where the client or information provider is not available on site, frequent communication is very necessary. If for some reason replies are not coming within a reasonable amount of time, it is within the interests of the project to pursue the involved party further. An example would be the issue of the log file format, where I had requested it from Dr Farrell who had passed it on to another person, who by mistake let it slip his mind. I should have tried to follow it up the next day, but did not and finally it was not received within time.

Time management and project planning need some additional thought. Unforeseen events can hinder any stage of the project, especially in the case of a project such as this, where we are dealing with streams whose reliability is not known. A buffer period should be kept which can be used as an add-on to whichever stage takes longer than expected. In this case, I had kept a one week buffer in the plan, which got used up when the hardware card took three weeks to install because of driver compatibility problems. If this buffer had not been in the plan, the project would have suffered even more.

Related to this is another issue. At any stage of the project, if it is possible to start more than one stage or phase, it is definitely best to focus on one stage, but also important to start thinking and working on the other. In my case, this would have been the analysis phase versus the design and implementation. The analysis phase took five weeks instead of the planned two weeks. Fortunately however, the investigations that I undertook during the analysis required me to write code which could be used in the final implementation. Hence, during the analysis itself, I started collecting all these bits and pieces of code, and started determining which ones might be used later, and tested them thoroughly. This saved me an enormous amount of time, which would have been squandered had I waited to finish the analysis phase first, and the demonstrator might not have been realised at all.
Another thing I learned during this project is, if there are processes to be followed, follow them strictly. An example would be the case when I had to get the card removed from the Linux system. I had not filed a ticket with support, but instead approached Mr Hardman directly. Due to a miscommunication, he thought I needed it the next day, whereas I needed it the same evening. So the card was not removed. As no ticket was filed, no one else in support knew about this, so no one took any action. I was lucky enough that Mr Saunders was available, otherwise I might not have been able to give my demonstration at the BBC the next day.

One final thing that happened, which I have experienced before, is that things are sometimes not what they are supposed to be. The DVB stream did not follow the specification, nor was the private section file as simple as it was supposed to be. The client was not aware of the intricacies of the system, as it had not been developed by them. Hence, it is best to question and analyse everything before moving on, and if there are doubts about any issues, they must be cleared at the first possible instance. It is in the interest of all parties to co-operate, hence there are no grounds for hesitation when an issue needs to be resolved or a question needs to be asked.

On the whole it was an enjoyable project experience, during which I got great support and a lot of useful advice and pointers from Mr Whyte. Our counterparts at the BBC, Dr. Sue Farrell and Mr Matt Biddulph were very helpful, but I realised that I could have done a better job of trying to get my questions across to them, which might have resulted in a less stressful project, with possibly even better results than we already got.

Nonetheless, the project was a success, and I am happy that I could think of further directions that the project could be taken into.
Appendix B – Project Objectives and Deliverables Form

School of Computing, University of Leeds

**MSc PROJECT OBJECTIVES AND DELIVERABLES**

This form must be completed by the student, with the agreement of the supervisor of each project, and submitted to the MSc project co-ordinator (Mrs A. Roberts) via CSO by 27th March 2003. A copy should be given to the supervisor and a copy retained by the student. Amendments to the agreed objectives and deliverables may be made by agreement between the student and the supervisor during the project. Any such revision should be noted on this form. At the end of the project, a copy of this form must be included in the Project Report as an Appendix.

Student: Udoyon Bhaduri

Programme of Study: MSc Distributed Multimedia Systems

Supervisor: Mr. Bill Whyte

Title of project: Detect and record changes to specific components of the transport stream

(Online Collection and Monitoring of DVB-T Engineering Channel data stream)

External Organisation*: BBC Broadcast Ltd

* (if applicable)

**AGREED MARKING SCHEME**

<table>
<thead>
<tr>
<th>Understand the Problem</th>
<th>Produce a Solution *</th>
<th>Evaluation</th>
<th>Write-Up</th>
<th>Appendix A</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>100</td>
</tr>
</tbody>
</table>

* This category includes Professionalism

**OVERALL OBJECTIVES:**

The BBC uses the DVB-T standard for terrestrial broadcast of channels. Within these channels is a low data rate channel called the Engineering Channel which is used for downloading upgrades to the firmware of Set Top Boxes. Since broadcast is inherently an unreliable mechanism for transmission of data, a cost-effective solution for monitoring the reception of this data stream is required.

1) Understand the DVB-T standard.

2) Design and develop a demonstrator software for online collection and monitoring of the data stream.

3) Identify additional requirements for a possible real-time system.
MINIMUM REQUIREMENTS:

1) Basic Demonstrator software capable of logging the specific id of the firmware upgrade being transmitted.
2) Recommendations for future work

SOFTWARE AND HARDWARE RESOURCES REQUIRED:

1) Linux Workstation with permission to compile and install software
2) DVB-T PCI card (on loan from the BBC)
3) Rooftop or portable aerial with digital terrestrial reception
4) Software for the PCI card (source provided by the BBC)

DELIVERABLE(s):

1) A project report.
2) The demonstrator
3) Documented code for customer
Appendix C – Marking Scheme and Interim Report Comments

School of Computing, University of Leeds

MSC PROJECT INTERIM REPORT

All MSc students must submit an interim report on their project to the MSc project coordinator (Mrs A. Roberts) via CSO by Thursday 8th May 2003. Note that it may require two or three iterations to agree a suitable report with your supervisor, so you should let him/her have an initial draft well in advance of the deadline. The report should be a maximum of 10 pages long and be attached to this header sheet. It should include:

- the objectives, deliverables and agreed marking scheme
- resources required
- progress report and project schedule
- proposed research methods
- a draft chapter on the literature review and/or an evaluation of tools/techniques
- the WWW document link for the project log to date

The report will be commented upon both by the supervisor and the assessor in order to provide you with feedback on your approach and progress so far.

The submission of this Interim Report is a pre-requisite for proceeding to the main phase of the project.

<table>
<thead>
<tr>
<th>Student:</th>
<th>Udoyon Bhaduri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme of Study:</td>
<td>MSc Distributed Multimedia Systems</td>
</tr>
<tr>
<td>Title of project:</td>
<td>Detect and record changes to specific components of the transport stream (Online Collection and Monitoring of DVB-T Engineering Channel data stream)</td>
</tr>
<tr>
<td>Supervisor:</td>
<td>Mr W S Whyte</td>
</tr>
<tr>
<td>External Company (if appropriate):</td>
<td>BBC Broadcast Ltd.</td>
</tr>
</tbody>
</table>

AGREED MARKING SCHEME

<table>
<thead>
<tr>
<th>Understand the problem</th>
<th>Produce a solution *</th>
<th>Evaluation</th>
<th>Write-up</th>
<th>Appendix A</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>100</td>
</tr>
</tbody>
</table>

* This includes professionalism

Signature of student: [Signature]

Date: 8/5/03

Supervisor’s comments on the Interim Report

Continued over
The student has made a good start on understanding the nature of the DVB-T protocol and how to extract the data. We still have concerns regarding TV reception in the area and also whether support will be able to provide the necessary technical facilities.

Assessor's comments on the Interim Report

Good progress with regard to understanding the problem as well as related issues (e.g., DVB-T technical information) according to the report and the project log. The report is well structured. However, it contains a large number of errors. I look forward to the demonstration of the tool.
Appendix D – Project Requirements Document

BBC Broadcast
Interactive Operations
MSc project definition
Broadcast data services monitoring

COMMERCIAL IN CONFIDENCE
Version 0-1
Contents

INTRODUCTION  55
BRIEF BACKGROUND  56
CONTACTS  56
  BBC Broadcast  56
    Phone  56
    Email  56
  University of Leeds  56
BUSINESS OBJECTIVES  57
TECHNICAL REQUIREMENTS  57
PROJECT DELIVERABLES  57

DOCUMENT HISTORY  67
Introduction

This document offers a general overview of the background to the BBC Broadcast requirement and a set of business objectives for the joint MSc project between BBC Broadcast and the University of Leeds.

Brief background

Interactive Operations within BBC Broadcast provides data playout services for the BBC and other clients on all UK digital television platforms.

Such data services consist of interactive television services and software upgrades for digital receivers.

Data for these services is carried as private data in a DVB transport stream, and is de-coded and presented to the digital receiver as a binary data stream.

The opportunity has been identified to use relatively inexpensive hardware to detect these services in a digital television transport stream and perform some simple analysis to verify and log the successful delivery of these private data services.


Contacts

BBC Broadcast

Phone

General: 020 7557 3700
Fax: 020 7557 3488

Email

Dr William Cooper: (william.cooper@bbc.co.uk), Head of Interactive for BBC Broadcast Ltd and BBC sponsor of this project.

Dr Sue Farrell: (sue.farrell@bbc.co.uk), Technical Manager, Interactive Operations in BBC Broadcast Ltd and technical mentor for this project.

Matt Biddulph: (matt.biddulph@bbc.co.uk), Senior Systems Architect, Interactive Operations in BBC Broadcast Ltd and technical architecture specialist.

University of Leeds

Udoyon Bhaduri: (udoyon@hotmail.com), MSc student specialising in Distributed Multimedia Systems

Bill Whyte: (billw@comp.leeds.ac.uk), Academic mentor for project, from the University of Leeds School of Computing.
Business Objectives

- To identify in a (binary) private section file a specific identity code on a linux filesystem.
- To identify a private data stream in a DVB-T transport stream and store on a linux filesystem, and then identify the specific identity code.
- To monitor a private data stream in a DVB-T transport stream and identify the specific identity code in the file and log the time of detection for each re-occurrence of the identity code.
- To use the above to verify that a file on the filesystem has the same identity and contents (checksum) as the one available in a private data stream, and log to a file any exceptions against time.
- To generate error messages for each exception in a defined XML format that can be http posted to another operational system.

Technical Requirements

- DVB-T PCI card with suitable linux drivers (to be provided on loan from BBC Broadcast)
- Aerial with digital terrestrial reception and distribution
- Linux PC

Project Deliverables

- Software capable of achieving each/any of the above objectives in isolation
- Basic documentation of the software for operational maintenance and configuration
- A copy of the submitted project report

Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>19/03/200</td>
<td>Sue Farrell</td>
<td>Initial draft</td>
</tr>
<tr>
<td></td>
<td>25/03/200</td>
<td>Sue Farrell</td>
<td>Minor revisions</td>
</tr>
</tbody>
</table>

Any comments, questions or change control requests relating to this document should be addressed to:
Head of Interactive, BBC Broadcast
Online Collection and Monitoring of DVB-T Engineering Channel Data Stream

MSc Project for BBC Broadcast

Design Document

Udoyon Bhaduri

University of Leeds
(September 2003)
Contents

Introduction.............................................60

1. General Architecture................................60
The OAD Monitoring System........................16

2. Class Structure.....................................62
The packaged classes.................................18
The driver class – OADMonitorDriver..............18

3. Activity Flow......................................63
3.1 Flow of the Driver class........................19
3.2 Flow of the StreamProcessor class..............20

4. Log file format design.............................23

Document history.....................................80
Introduction

This document describes the general architecture of the system, as well as a proposed class structure for the OAD Monitoring Software. The design of the structure of the log file is included at the end.

1. General Architecture

The system consists of a personal computer running the Linux operating system, equipped with a WinTV digital TV receiver card. An antenna is used to catch the terrestrial signal, which is then transmitted to the card. The drivers for the card enable the extraction of the transport stream packets. This extraction can be restricted to a single program id, hence allowing the extraction of the engineering channel data in the form of a transport stream. This stream is to be processed by the OAD Monitoring Software which logs the occurrence of the Manufacturer ID. The following figure shows this complete sequence:

![Diagram](image)

**Fig 1.1: General Architecture**

**The OAD Monitoring System**

The structure of the OAD Monitoring system is given below:

![Diagram](image)

**Fig 1.2. Software Structure**
The processing of the stream needs to be done in the following manner:

1) Find the sync byte from which processing should start – this can be done by processing the first three packets to ensure that the wrong sync byte sequence is not taken
2) The software should first strip off the transport stream headers from each packet, and fill a buffer with the payload alone.
3) The search for the MIS descriptor and the manufacturer ID can be done on the data in this buffer.
4) Whenever the Manufacturer ID is found, the time of detection should be logged. Also, the time difference from the previous occurrence should be written to the log.

According to the revised specifications, the DSI/MIS should be found at maximum intervals of 5 seconds. Hence, the time when the last manufacturer ID was found should be stored, and every 5 seconds thereafter an error should be logged if the manufacturer ID is not found. But, the CRC check has been included in the design, so that it can be implemented easily once the exact structure of the Transport Stream’s payload is known.

This satisfies upto level 4 of the requirement specification. The output log format can be converted to a defined XML format, once available, to satisfy level 5. This can be done within the println() method of the Util class, which is shown in the next section.

The next section describes the structure of each of the classes designed to develop such a system.
2. Class Structure

The system consists of 3 classes which form a package, and a driver class. The classes follow the structure given below:

Fig 2: Class Diagram for the OAD Monitor

The packaged classes

4) StreamProcessor: This is the main processing class. It contains two methods:
   a. StartProcessing(): Contains the complete logic for processing the DVB stream.
   b. Read(): A method to read from a given stream while incorporating a timeout for the read operation.

5) Util: This class contains a number of static methods. These are
   a. hexToDec(): Convert a hexadecimal number to decimal
   b. getTwoComplement(): Get the two’s complement of an integer
   c. hexToDecTwoComplement(): Combines the above two methods.
   d. println(): Facilitate writing to standard output as well as a Writer object. It also takes a parameter to disable writing to standard output.

6) OADMonitorException: A sub-class of Exception, it is used to differentiate any known exceptions arising in accordance with the high-level logic of the program.

The driver class - OADMonitorDriver

This class consists of only a main method, and some package level variables. It is responsible for taking the input parameters, instantiating a StreamProcessor object, and then passing the parameters to the StreamProcessor object’s startProcessing() method. It is also responsible for the validation of the properties and command line options specified.
3. Activity Flow

The starting point of the software is the OADMonitorDriver class. After validations and determining the DVB stream, source private section file and log file, it calls the StreamProcessor class with these as arguments. The StreamProcessor class then examines the DVB stream for the MIS descriptor and Manufacturer ID, and logs occurrences as well as errors.

3.1 Flow of the Driver class

The flow of the driver class is outlined in the diagram below:

![Activity Diagram for the OADMonitorDriver class](image)

It first takes the command line arguments and validates them. If they are invalid it terminates. Otherwise, it goes on to loading the parameters such as manufacturer ID, private section source file, maximum DSI Interval etc., from a properties file. If any of these are invalid, it terminates. Else, it opens the DVB stream and the private section file for reading, and the log file for writing. If it fails to do so, it terminates, otherwise it writes the parameter information to the log file. Next, it calls the StreamProcessor class’ startProcessing() method, to process the stream. Finally, it closes all streams and exits.
3.2 Flow of the StreamProcessor class

The flow of the driver class is outlined in the diagram below:

Figure 3.2: Activity Diagram for the StreamProcessor class’ startProcessing() method
The StreamProcessor’s startProcessing() method is called by the OADMonitorDriver class. The startProcessing() algorithm can be subdivided into five sequential stages.

1. Check for the DVB stream

It first opens the DVB stream to check if there are any sync bytes present in the first 3 packets of the stream. If there are any, it stores the positions of these bytes and then determines which of them is the first most likely sync byte. It does this by analysing the distance between the sync bytes, and which of them fit the 188-byte gap pattern. If there is no data at all, it exits with a message saying there is no data in the stream. If no sync bytes are found, it exits with a message saying that no DVB Transport stream was found.

2. Prepare the buffers for processing

Once the first sync byte is determined, the occurrence of the DVB stream is written to the log. The method now reads 188 bytes of data, using the read() method of the class into a buffer called the TS buffer which would be of size 188 bytes. It then initialises a buffer of size 5 bytes called the TS End buffer, with zeroes, and sets all time variables to the current time. The TS Join buffer is also initialised at this point. The TS Join buffer shall be used for actual processing, as it takes into account the last 5 bytes of the packet that are usually left unprocessed (These bytes are taken from the TS End buffer).

3. Process the data

Next, it loops while the Waiting Time is less than the Maximum Waiting Period, else it goes to step (5). Waiting time is defined as the time since the last occurrence of the ID was found.

If the last read from the DVB Stream was successful, it proceeds to fill the TS Join buffer with the 5 bytes from the TS End buffer. It fills the remaining part of the TS Join buffer with the 179 bytes of the TS buffer following the first 4 bytes (which comprise the MPEG Header).

Next, it examines each of the bytes in the TS Join buffer, and checks for the MIS descriptor and Manufacturer ID, which should be separated by 3 bytes. If it finds them, it logs the occurrence with a timestamp, and stores the time of occurrence. The CRC check buffer can be filled at this time, with the bytes in the TS buffer. At the end of the loop the last 5 bytes are left unprocessed, hence these are copied into the TS End buffer. Now, if the CRC check buffer is full, the CRC check can be performed (This is provided the CRC frame size is known and does not vary for different files).

4. Get the next packet of data

If there were no bytes available from the DVB Stream, or if the data has been processed, the next read is performed. The Waiting Time is now re-calculated. If it has exceeded the Maximum DSI Interval and the time since the last check is greater than the Maximum DSI Interval, it logs it as an error. Next, it proceeds as according to step (3).

5. Log Termination and Exit

Reaching this step signifies that either no more data is available or the DSI could not be found for the Maximum Waiting Period. Hence, termination is logged and it exits.
4. Log file format design

The log format was designed to give the most relevant information in a usable form. Its structure is given in the following diagram:
# Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>13/08/2003</td>
<td>Udoyon Bhaduri</td>
<td>Initial draft</td>
<td>Mr W S Whyte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(University of Leeds)</td>
</tr>
<tr>
<td>0.2</td>
<td>28/08/2003</td>
<td>Udoyon Bhaduri</td>
<td>Revisions – Format and reference to req. spec</td>
<td>Dr. Sue Farrell</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(BBC Broadcast)</td>
</tr>
</tbody>
</table>
Appendix F – Javadoc For The Code

1. Package Summary

Package OADMonitor

<table>
<thead>
<tr>
<th>Class Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StreamProcessor</strong></td>
</tr>
<tr>
<td><strong>Util</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exception Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OADMonitorException</strong></td>
</tr>
</tbody>
</table>

2. StreamProcessor.java

OADMonitor

Class StreamProcessor

```
<table>
<thead>
<tr>
<th>java.lang.Object</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

public class **StreamProcessor**
extends java.lang.Object

The main class of the OAD Monitor, used for processing the dvb stream and writing the results to a log file.

It processes the dvb stream and writes all occurrences as well as errors to the log file. As input, it takes the dvb stream, manufacturer id, the private section stream, the log writer, maximum dsi interval, maximum DSI waiting period, and output preference (log only or standard output also)

Version:

1.0

Author:
Udoyon Bhaduri

**Constructor Summary**

<table>
<thead>
<tr>
<th><strong>StreamProcessor</strong>&lt;br&gt;java.io.InputStream i_stream, int[] id, java.io.InputStream src_stream, java.io.Writer log, int mdi, int mwp, int output_pref)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sole constructor for the class It takes each and every external Object or parameter required and sets its own attributes accordingly</strong></td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th><strong>protected int</strong>&lt;br&gt;read(java.io.BufferedInputStream bis, byte[] buffer, int start, int end, int timeout)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method to read from a BufferedInputStream into a given buffer with a timeout.</strong></td>
</tr>
<tr>
<td><strong>void startProcessing()</strong></td>
</tr>
<tr>
<td><strong>The main stream processing method It opens the dvb stream, searches for sync bytes and determines the first sync byte it can start processing from.</strong></td>
</tr>
</tbody>
</table>

**Constructor Detail**

**StreamProcessor**

<table>
<thead>
<tr>
<th><strong>public StreamProcessor</strong>&lt;br&gt;java.io.InputStream i_stream, int[] id, java.io.InputStream src_stream, java.io.Writer log, int mdi, int mwp, int output_pref)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sole constructor for the class It takes each and every external Object or parameter required and sets its own attributes accordingly</strong></td>
</tr>
</tbody>
</table>

**Parameters:**

- **i_stream** - The DVB transport stream containing the Engineering Channel Packets
- **id** - The manufacturer ID to be detected
- **src_stream** - The source private section file to be compared with. Though it is required, it is not used currently
- **log** - The log file to be written to.
- **mdi** - The maximum DSI interval after which an error is logged
mwp - The maximum waiting period after which processing should terminate
output_pref - Value 0 or 1, 0 specifying write to both stdout and the log, 1 specifying log only

## Method Detail

### startProcessing

```java
public void startProcessing()
    throws OADMonitorException
```

The main stream processing method
It opens the dvb stream, searches for sync bytes and determines the first sync byte it can start processing from.< > Next, it reads the stream in chunks of 188 bytes, and skips the 4 byte TS header.< > It looks for the manufacturer ID, and logs it if found.< > If not found within the Max DSI Interval period, it logs an error If not found within the Max Waiting Period, it terminates.

**Throws:**
OADMonitorException - If the processing should be stopped due to a problem with the stream.

### read

```java
protected int read(java.io.BufferedInputStream bis,
                   byte[] buffer,
                   int start,
                   int end,
                   int timeout)
    throws java.io.EOFException,
           java.io.IOException
```

Method to read from a BufferedInputStream into a given buffer with a timeout. The start and end offsets can be specified, along with a timeout within which the read operation should be successful

**Parameters:**
bis - The BufferedInputStream object to be read from
buffer - The buffer in which the read data should be put
start - The starting offset for the read (Similar to the read method of BufferedInputStream)
date - The number of buffers to be read (Similar to the read method of BufferedInputStream)
timeout - The maximum time interval till which re-tries should be allowed

**Returns:**
The number of bytes read

**Throws:**
java.io.EOFException - If an end of file exception occurred
java.io.IOException - If an input or output exception occurred
3. **Util.java**

OADMonitor

**Class Util**

```java
public class Util
extends java.lang.Object
```

Utility class containing static methods such as hexadecimal to decimal conversion and writing to a stream as well as standard input.

**Version:**

1.0

**Author:**

Udoyon Bhaduri

---

**Constructor Summary**

<table>
<thead>
<tr>
<th><strong>Util()</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Default constructor, not used presently as all methods are static</td>
</tr>
</tbody>
</table>

---

**Method Summary**

<table>
<thead>
<tr>
<th><strong>static int[] getTwosComplement(int[] org)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method to convert decimals into signed bytes (2's complement)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>static int[] hexToDec(java.lang.String hex)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method to convert hexadecimal digits to the decimal form, two at a time (1 byte = 2 hex digits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>static int[] hexToDecTwo(java.lang.String hex)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method to get the decimal conversion of the hexadecimal number in signed byte form which uses 2s complement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>static void println(java.lang.String s, java.io.Writer w, int output_pref)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Method to write to a writer object, as well as standard output.</td>
</tr>
</tbody>
</table>
**Constructor Detail**

**Util**

```java
public Util()
```

Default constructor, not used presently as all methods are static

**Method Detail**

**hexToDec**

```java
public static int[] hexToDec(java.lang.String hex)
```

Method to convert hexadecimal digits to the decimal form, two at a time (1 byte = 2 hex
digits)

**Parameters:**

hex - The hexadecimal string to be converted

**Returns:**

An integer array of the resulting decimal integers

**hexToDecTwos**

```java
public static int[] hexToDecTwos(java.lang.String hex)
```

Method to get the decimal conversion of the hexadecimal number in signed byte form which
uses 2s complement

**Parameters:**

hex - The hexadecimal string to be converted

**Returns:**

An integer array of the resulting decimal integers after the two's complement

**getTwosComplement**

```java
public static int[] getTwosComplement(int[] org)
```

Method to convert decimals into signed bytes (2s complement)

**Parameters:**

org - The original array of integers

**Returns:**

An array of the integers converted to twos complement

**println**

```java
public static void println(java.lang.String s,
    java.io.Writer w,
```


Method to write to a writer object, as well as standard output. Standard output can be
skipped, depending on the value of output_pref

Parameters:
- String to be written
- The Writer object to be written to
- output_pref - Currently 0 or 1, 0 signifying write to standard output also

Throws:
- OADMonitorException - If the write was not successful - Usually due to permissions or disk
space.

4. OADMonitorException.java

OADMonitor

Class OADMonitorException

java.lang.Object
 |
- java.lang.Throwable
  |
  - java.lang.Exception
    |
    - OADMonitor.OADMonitorException

All Implemented Interfaces:
- java.io.Serializable

public class OADMonitorException
    extends java.lang.Exception

An exception class signifying that an exception specific to this package is being thrown.
It's main utility is to easily distinguish between known and unknown errors, and pass a
meaningful and specific message to the user.

Version:
- 1.0

Author:
- Udoyon Bhaduri

See Also:
- Serialized Form
OADMonitorException()
    Default constructor

OADMonitorException(java.lang.String s)
    Constructor using the normal method of creating an exception object with a given message

### Constructor Detail

**OADMonitorException**

```java
public OADMonitorException()
    Default constructor
```

```java
public OADMonitorException(java.lang.String s)
    Constructor using the normal method of creating an exception object with a given message

Parameters:
    s - The string message for the exception
```

### 5. OADMonitorDriver.java

**Class OADMonitorDriver**

```java
java.lang.Object

    +--- OADMonitorDriver
```

```java
public class OADMonitorDriver
    extends java.lang.Object

The driver program for the OAD Monitor.
It takes the DVB stream from a file or from standard input (if it is piped to this from another
program) It calls the startProcessing of the StreamProcessor class, to monitor the DVB stream.
It takes one flag, -s, which forces it to silence mode, in which no output goes to standard
output, only to the log file.

Version:
    1.0

Author:
    Udoyon Bhaduri
```
Constructor Summary

OADMonitorDriver()

Method Summary

static void main(java.lang.String[] args)

The single point of entry for the driver program.

Constructor Detail

OADMonitorDriver

public OADMonitorDriver()

Method Detail

main

public static void main(java.lang.String[] args)

The single point of entry for the driver program. The main method performs all the necessary parameter validations and opens all relevant streams before calling the StreamProcessor class.

Parameters:
args - Command line switches -s and -a

Returns:
void
Appendix G – Installation and Usage Document

Online Collection and Monitoring of DVB-T Engineering Channel Data Stream
MSc Project for BBC Broadcast

Installation and Usage Document

Udoyon Bhaduri
University of Leeds
(September 2003)
Contents

INSTALLATION........................................................................................................ 78
RUNNING THE SOFTWARE..................................................................................... 78
EXITING THE SOFTWARE....................................................................................... 79
DOCUMENT HISTORY ............................................................................................ 80


Installation

1. Make sure that the OADMonitorDriver.jar file is present on the system in a directory <LIB>.
2. Make a directory <INSTALL> where the software should be installed.
3. Extract the OADMonitor.properties file from the jar file, to the INSTALL directory.
4. Ensure that the properties file in INSTALL has the correct settings:
   a. The correct Manufacturer ID
   b. The correct private section file path.
      i. Currently the private section file is not used for comparison, hence any dummy file can be used. Nonetheless, it must be ensured that the file exists.
   c. The correct DVB stream file path
      i. This should be set only if the system is to read the stream from a file. This is not generally how it would be used. This facility is only for testing purposes.
      ii. If the stream is NOT taken from a file, comment out this property.
   d. The correct path where the log-file should be created. If the append switch is not used, any existing file with the same name will be over-written. If the append switch is used, any existing file with the same name shall have entries appended to it.
   e. The correct Maximum DSI Interval (in milliseconds): By default, this is set to 5000.
   f. The correct Maximum Waiting Period (in milliseconds): By default, this is set to 10000.

Running the Software

1) To run the software on a test dump, use the command:

   a) On Unix/Linux: java –jar <LIB>/OADMonitorDriver.jar
   b) On Windows: java –jar <LIB>\OADMonitorDriver.jar

2) To run the software on a stream being piped from, use the command:

   a) On Unix/Linux: <DvbStream> | java –jar <LIB>/OADMonitorDriver.jar
      Example:
      /usr/local/DVB/dvbstream/dvbstream –o 653 | lib/OADMonitorDriver.jar –s
   b) On Windows: <DvbStream> java –jar <LIB>\OADMonitorDriver.jar
Exiting the Software

1) When running it on a stream piped from the Nova-T drivers:
   a) If the driver is running for a certain number of seconds, the program will exit
      at the end automatically, after the maximum waiting period.
      i) **Exceptions** to this rule:
         (1) If the system encounters an error (NOT a DSI error), it shall terminate
             execution immediately, and return to the command prompt.
         (2) In the case that there is no data in the stream, the program shall
             terminate execution. But, it does not come back to the command
             prompt until the driver stops running.
   b) If the driver is providing the stream for an unspecified length of time, the
      program will **end automatically if there is any error**, else it shall keep running.
      i) **Exceptions** to this rule:
         (1) In the case that there is no data in the stream, the program shall
             terminate execution. But, it does not come back to the command
             prompt. This may also happen when the stream stops while the
             program is running.
             The only way to terminate it is by pressing Ctrl+C, or killing the
             process.

2) When running it on a dump file
   a) On encountering any configuration errors or reaching the maximum waiting
      period, the program shall exit immediately, returning to the command prompt.
   b) Otherwise, it shall run till the end of file is reached and return to the command
      prompt.
# Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>22/08/2003</td>
<td>Udoyon Bhaduri</td>
<td>Initial draft</td>
<td>Mr W S Whyte (University of Leeds)</td>
</tr>
<tr>
<td>0.2</td>
<td>28/08/2003</td>
<td>Udoyon Bhaduri</td>
<td>Revisions – Format</td>
<td>Dr. Sue Farrell (BBC Broadcast)</td>
</tr>
</tbody>
</table>
Appendix H – Stream and Private Section File Comparison Results

Only one stream and private section file pair has been shown. The diff value is the difference in the number of bytes from the previous match of either a manufacturer ID or a magic number. As can be seen, the pattern of the stream file starts matching with the private section file from the second page of the stream file’s output.

Panasonic Private Section File

File size: 2371388
Manufacturer ID At 112 diff: 112 ver: 96 0 1 14 length: 0 7 dscr: -32 before
dscr: 10 0
Magic Number Found at: 149 diff: 37
Magic Number Found at: 336 diff: 187
Magic Number Found at: 554 diff: 218
Magic Number Found at: 4650 diff: 4096
Manufacturer ID At 41600 diff: 36950 ver: 96 0 1 14 length: 0 7 dscr: -32 before
dscr: 10 0
Magic Number Found at: 123661 diff: 82061
Magic Number Found at: 123848 diff: 187
Manufacturer ID At 165112 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 247173 diff: 82061
Magic Number Found at: 247360 diff: 187
Manufacturer ID At 288624 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 370685 diff: 82061
Magic Number Found at: 370872 diff: 187
Manufacturer ID At 412136 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 494197 diff: 82061
Magic Number Found at: 494384 diff: 187
Manufacturer ID At 535648 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 617709 diff: 82061
Magic Number Found at: 617896 diff: 187
Manufacturer ID At 659160 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 741221 diff: 82061
Magic Number Found at: 741408 diff: 187
Manufacturer ID At 782672 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 864733 diff: 82061
Magic Number Found at: 864920 diff: 187
Manufacturer ID At 906184 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 988245 diff: 82061
Magic Number Found at: 988432 diff: 187
Manufacturer ID At 1029696 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1111757 diff: 82061
Magic Number Found at: 1111944 diff: 187
Manufacturer ID At 1153208 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1235269 diff: 82061
Magic Number Found at: 1235456 diff: 187
Manufacturer ID At 1276720 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Online Collection and Monitoring of DVB-T Engineering Channel Data Stream

Magic Number Found at: 1358781 diff: 82061
Magic Number Found at: 1358968 diff: 187
Manufacturer ID At 1400232 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1482293 diff: 82061
Magic Number Found at: 1482480 diff: 187
Manufacturer ID At 1523744 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1605805 diff: 82061
Magic Number Found at: 1605992 diff: 187
Manufacturer ID At 1647256 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1729317 diff: 82061
Magic Number Found at: 1729504 diff: 187
Manufacturer ID At 1770768 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1852829 diff: 82061
Magic Number Found at: 1853016 diff: 187
Manufacturer ID At 1894280 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 1976341 diff: 82061
Magic Number Found at: 1976528 diff: 187
Manufacturer ID At 2017792 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 2099853 diff: 82061
Magic Number Found at: 2100040 diff: 187
Manufacturer ID At 2141304 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 2223365 diff: 82061
Magic Number Found at: 2223552 diff: 187
Manufacturer ID At 2264816 diff: 41264 ver: 96 0 1 14 length: 0 7 dscr: -32
before dscr: 10 0
Magic Number Found at: 2346877 diff: 82061
Magic Number Found at: 2347064 diff: 187

Panasonic DVB Stream

File size: 3669329
Magic Number Found at: 5547 diff: 5547
Magic Number Found at: 5734 diff: 187
Manufacturer ID At 48505 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 133427 diff: 84922
Magic Number Found at: 133614 diff: 187
Manufacturer ID At 176385 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 261307 diff: 84922
Magic Number Found at: 261494 diff: 187
Manufacturer ID At 304265 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 389187 diff: 84922
Magic Number Found at: 389374 diff: 187
Manufacturer ID At 432145 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 517067 diff: 84922
Magic Number Found at: 517254 diff: 187
Manufacturer ID At 560025 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 644947 diff: 84922
Magic Number Found at: 645134 diff: 187
Manufacturer ID At 687905 diff: 42771 ver: 96 0 1 14 length: 0 7 dscr: -32 before dscr: 10 0
Magic Number Found at: 772827 diff: 84922
Magic Number Found at: 773014 diff: 187
Online Collection and Monitoring of DVB-T Engineering Channel Data Stream

Magic Number Found at: 2844670 diff: 187
Manufacturer ID At 2887441 diff: 42771 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 2972363 diff: 84922
Magic Number Found at: 2972550 diff: 187
Manufacturer ID At 3015321 diff: 42771 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3100243 diff: 84922
Magic Number Found at: 3100430 diff: 187
Manufacturer ID At 3143201 diff: 42771 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3228123 diff: 84922
Magic Number Found at: 3228310 diff: 187
Manufacturer ID At 3253601 diff: 25291 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3253699 diff: 98
Magic Number Found at: 3253886 diff: 187
Magic Number Found at: 3254251 diff: 365
Magic Number Found at: 3258483 diff: 4232
Manufacturer ID At 3296657 diff: 38174 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3381579 diff: 84922
Magic Number Found at: 3381766 diff: 187
Manufacturer ID At 3424537 diff: 42771 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3509459 diff: 84922
Magic Number Found at: 3509646 diff: 187
Manufacturer ID At 3552417 diff: 42771 ver: 96 0 1 14 length: 0 7 descr: -32
before descr: 10 0
Magic Number Found at: 3637339 diff: 84922
Magic Number Found at: 3637526 diff: 187
Appendix I – Evaluation by the Client

Evaluation

Overview

This was an ambitious project that had been separated into a number of implementation options and steps. The technical complexity of the area of investigation proved to be greater than anticipated when the project was originally proposed.

A significant delay was introduced when the DVB-T hardware and software procured for the project proved to be inappropriate as a result in a subtle change in chipset in the hardware. This was resolved by a replacement card but this had an impact on the implementation time available.

Udoyon showed commitment to the project and worked independently without requiring close supervision or management, seeking advice and assistance only when necessary, and has delivered a competent implementation of a sensible system design.

I was generally satisfied with the ongoing communication around the technical solution and the necessary negotiation around the functionality during the course of the project meeting obstacles and unanticipated complexity.

While I have identified improvements that could be made in the project management and technical solution, I was particularly pleased that these had already been identified by Udoyon and discussed during the system handover and review. Udoyon responded well to this feedback in a positive and constructive manner during our review meeting.

I am satisfied that the proof of concept system design and implementation have achieved the core objectives for BBC Broadcast.

Understanding of the requirements

The technical specialisation required to deliver to the BBC Broadcast requirements was substantial, and Udoyon showed a basic understanding of the business objectives and deliverables at the outset of the project. Further research and investigation into the area resulted in intelligent and relevant questions to refine his understanding of the requirements.

The deliverables clearly meet the first three business objectives.

The fourth objective proved to be more complex than originally anticipated by all parties and a solution and the groundwork for achieving this was identified, but there wasn’t sufficient time for implementation.

The fifth objective was not demonstrated but Udoyon identified how this can be implemented. This was not achieved due to delays from BBC Broadcast in providing the format.
I am satisfied that Udoyon understood the requirements, and identified and agreed variation with us through the project.

**Project Management**

Udoyon demonstrated a rare skill for a software developer, in that he set a deadline for freezing the code base and stuck to it! This is a particularly valuable project management skill that will provide a good foundation for his professional future.

Udoyon kept the project to time and was able to demonstrate the system to us at the agreed time and place.

The project would have benefited from a more proactive approach to project management. While consistent progress was maintained, it was not regularly reported and delays waiting for answers to questions could have been reduced by more project management confidence.

**Technical solution**

The technical solution was well-documented and had identified a number of complexities with the technical environment and objectives.

The design was sound and achieved the fundamental objectives.

Udoyon identified a number of improvements that could be made to the technical solution given additional implementation time.

The system currently requires regular operational manipulation to identify the identity of the intended OAD and logs compliance. This would be significantly improved by the ability to determine the identity of the OAD from a lookup table and the DVB descriptors and log the established identity.

This solution proved too ambitious for the duration of this project, and I am satisfied that Udoyon had identified the limitations of the solution delivered and had already identified and communicated a number of advisable improvements, which is important in a project of this type.

**Documentation**

A number of documents were voluntarily delivered throughout the project. All documents were well presented with logical structure and competently communicated the relevant information.

- Interim report, describing the project requirements and a literature review of the relevant technologies.
- Design document, describing the technical design of the solution, including architecture, class structure, activity flow and log format.
- Installation and Usage document, describing the installation and processes required to use the system.
- Presentation, offering an overview of the project and solution. We anticipate receiving the project report as agreed.
Delivery

The software was delivered by Udoyon during a site visit on 29 August.

During the site visit, the DVB-T card and software was installed and Udoyon went through a handover session with Matt Biddulph (Senior Systems Architect, BBC Broadcast).

A presentation of a project overview then preceded a demonstration of the software.

Unfortunately the real-time element of the solution was unable to be demonstrated as there was no data-stream playing out during the presentation.

The demonstration on BBC Broadcast systems satisfied the fundamental objectives of the software deliverables, and proved that in the case of a data stream being unavailable (a formal condition of the system), the system responds correctly and logs the absence.

The handover was completed by a project review between Sue Farrell and Udoyon, discussing verbally this evaluation of the project.
Appendix J – List of Figures and Tables

List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig 2.1</td>
<td>Multiplexing in the MPEG2 System.</td>
<td>04</td>
</tr>
<tr>
<td>Fig 2.2</td>
<td>Structure of the Transport Stream.</td>
<td>05</td>
</tr>
<tr>
<td>Fig 2.3</td>
<td>Example Structure of an Engineering Channel with Object Carousel.</td>
<td>07</td>
</tr>
<tr>
<td>Fig 3.1a</td>
<td>Private Section File according to readings and information.</td>
<td>12</td>
</tr>
<tr>
<td>Fig 3.1b</td>
<td>Anticipated Structure of the Engineering Channel Stream.</td>
<td>12</td>
</tr>
<tr>
<td>Fig 3.2a</td>
<td>Actual Private Section File.</td>
<td>13</td>
</tr>
<tr>
<td>Fig 3.2b</td>
<td>Actual Structure of the Engineering Channel Stream.</td>
<td>13</td>
</tr>
<tr>
<td>Fig 3.3</td>
<td>General Architecture of the Monitoring System.</td>
<td>16</td>
</tr>
<tr>
<td>Fig 3.4</td>
<td>Software Structure For The OAD Monitor</td>
<td>17</td>
</tr>
<tr>
<td>Fig 3.5</td>
<td>Design level Class Diagram.</td>
<td>18</td>
</tr>
<tr>
<td>Fig 3.6</td>
<td>Activity Diagram for the OADMonitorDriver class.</td>
<td>19</td>
</tr>
<tr>
<td>Fig 3.7a</td>
<td>Activity Diagram for the StreamProcessor class’ startProcessing() method</td>
<td>20</td>
</tr>
<tr>
<td>Fig 3.8</td>
<td>Log File Format.</td>
<td>23</td>
</tr>
<tr>
<td>Fig 4.1a</td>
<td>Class Diagram for the Implementation.</td>
<td>27</td>
</tr>
<tr>
<td>Fig 4.2a</td>
<td>Screenshot: Empty Stream.</td>
<td>28</td>
</tr>
<tr>
<td>Fig 4.2b</td>
<td>Screenshot: Non-engineering channel live DVB Stream.</td>
<td>28</td>
</tr>
<tr>
<td>Fig 4.3a</td>
<td>Screenshot: Emulation of faulty DVB stream.</td>
<td>29</td>
</tr>
<tr>
<td>Fig 4.3b</td>
<td>Screenshot: Non-Engineering Channel, live DVB Stream.</td>
<td>29</td>
</tr>
<tr>
<td>Fig 5.1</td>
<td>Graph for 350,000 read attempt.</td>
<td>33</td>
</tr>
<tr>
<td>Fig 5.2</td>
<td>Graph for 15,000 read attempts.</td>
<td>33</td>
</tr>
</tbody>
</table>

List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>MIS Structure.</td>
<td>06</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Packages and Classes Used.</td>
<td>26</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Initial Project Plan.</td>
<td>42</td>
</tr>
<tr>
<td>Table 6.2</td>
<td>Initial Milestones Chart.</td>
<td>43</td>
</tr>
<tr>
<td>Table 6.3</td>
<td>Final Project Plan.</td>
<td>44</td>
</tr>
<tr>
<td>Table 6.4</td>
<td>Final Milestones</td>
<td>44</td>
</tr>
</tbody>
</table>