Jerker Björkqvist | Janne Kempe | Kristian Nybom | Michael Stormbom| Raoul Sundsten

A system for measuring application level quality of service in a DVB-H network

Turku Centre *for* Computer Science

TUCS Technical Report
No 807, December 2006

A system for measuring application level quality of service in a DVB-H network

Jerker Björkqvist
Åbo Akademi University, Department of Information Technologies

Janne Kempe
Åbo Akademi University, Department of Information Technologies

Kristian Nybom
Åbo Akademi University, Department of Information Technologies

Michael Stormbom
Åbo Akademi University, Department of Information Technologies

Raoul Sundsten
Åbo Akademi University, Department of Information Technologies

TUCS Technical Report
No 807, December 2006

Abstract

In this report, a measurement system for measuring DVB-H signal performance in field conditions is presented. The system gives performance values for both physical layer DVB-T conditions, such as transport stream error rate, but also error rates after applying application layer coding (ALC). The report presents the fundamentals of the used error correction coding systems used, and the stacks used for delivery of test data objects.

Keywords: DVB-H, Forward Error Correction Coding, Field Measurements, Measurement Systems, Tornado Coding.

TUCS Laboratory
Embedded Systems Laboratory

1 Introduction

The DVB-H technology was standardized in late 2005 for enabling data broadcasting to mobile, handheld terminals. The main objective of DVB-H is to provide video streaming to mobile terminals, but as the transport stream is data, any data can be broadcasted using the system.
DVB-H is based on the DVB-T standard, which main objective was to provide rooftop antenna (fixed-point) reception of MPEG2 encoded video streams. DVB-H adds an additional error correction layer to the system and a time slicing system. The error correcting is needed for improving mobile capabilities of the system, whereas the time slicing is used for energy minimization in the terminals.

Performance of newly standardized telecommunication systems can quite well be estimated using simulation methods (using simulation software) or lab tests in laboratory conditions. However, any results must also be validated using filed tests. In a field test, actual receiver equipment is used in conditions corresponding to typical system use cases, and during the usage, performance measures are recorded. These performance records can for instance be current signal to noise level (S/N), bit error rate, Transport Stream (TS) Packet error rate, physical location and speed vectors of receiver device. Using these measurements, the performance of the system can be estimated and validated against simulated and lab data.

The main difficulty measuring DVB-H systems is the vast amount of parameters that can be used for specifying modulations and error corrections used in different layers of the overall system. However, the physical layer is very similar to DVB-T, and at that level, the number of combinations of parameters is smaller. Both DVB-T and DVB-H use a common packet for transporting data, the TS packet. By recording TS packet error rates, upper layers may be simulated using different parameters. Hence, it becomes important to get relevant TS packet error traces, corresponding to actual use cases.

In order to easily acquire these error traces, a system for field measurements where built. The objective of the system was to build it using standardized, non-expensive equipment. As the DVB-T and DVB-H systems are almost the same on physical layer, the system was build around a DVB-T receiver device targeted for PC (laptop) usage. These receiver devices usually contains the same (or very similar) silicon as that being used in future DVB-H devices, hence performance can be assumed to resemble real future DVB-H receivers.

The target was to build a system by which TS error traces can be easily acquired, but also a system that can be used to show the capabilities of DVB-H in real time, in terms of video streaming, but also in terms of file object download.

2 Equipment

The hardware setup consisted of a laptop, a DVB-T device with an antenna and a GPS.

Figure 1. HP Compaq nx6110

The laptop was an HP Compaq nx6110, equipped with a Celeron M 360 1400 MHz processor and 256 MB memory. It was chosen because of its low price, small weight, relatively long battery time, and compatibility with Linux. The processor was enough for the kind of processing we would do, and the memory requirements were not that high either, so the performance of it fulfilled our needs. When doing measurements in a car a car battery charger was used.

Figure 2. Hauppauge Nova-T USB2.

The DVB-T receiver was Hauppauge Nova-T USB2, a device that is able to deliver the whole transport stream, and it works well with Linux. It supported the front-end parameters that were in use by the DVB-H transponder. The antenna was an active one, also powered by USB.

3 Receiving DVB-H using PC equipment

Since DVB-H is based on DVB-T, it is possible to use a DVB-T receiver to get the DVB-H data, as long as the device supports the transmission parameters in use. Since the Hauppauge device we used supports all parameters except 4K mode, which was not used anyway, it was suitable for our needs.

Because we needed to do a lot of programming, the obvious choice of OS was Linux, more specifically Ubuntu 5.04, which was successfully installed on the laptop. To get the DVB-T device to work, drivers were needed. They were found at www.linuxtv.org that is an open source community developing drivers and utilities for various digital TV cards and devices for Linux.

The driver sources were downloaded from their CVS repository. In order to compile the modules the source for a recent Linux kernel (2.6.x) is also needed, which can be obtained from www.kernel.org, or by using Ubuntu's built in packaging system (apt). After compiling and loading the drivers some devices are created in /dev/dvb/adapterX/, where X is the index of the DVB adapter, which in our case was zero, since we only had one. There is for example a device for the front-end, demux, and DVR (Digital Video Recorder). To do the actual programming the Linuxtv DVB API v3 was used which can be found on their pages.

A typical scenario for receiving data, along with the transport stream headers, would be the following:

1) Open a frontend device, using the open system call.

```
fefd = open("/dev/dvb/adapter0/frontend0", O_RDWR);
```

2) Set the front-end parameters with an ioctl system call, which needs a pointer to a structure specifying the parameters, such as the frequency to tune to, guard interval, code rate and so on. See the DVB API documentation for details.

```
ioctl(fefd, FE_SET_FRONTEND, &parameters);
```

3) Open a demux device, which filters out the PIDs that are interesting.

```
dmxfd = open("/dev/dvb/adapter0/demux0", O_RDWR);
```

4) Setup a PID filter using an ioctl call on the demux device. A pointer to a structure with filter parameters has to be provided; including the PID, we want to filter out (the special value of 8192 means that all available PIDs are taken out). Another parameter is whether we want to read the data

directly from the demux device, or if we want to redirect it to a DVR, in which case we also get the TS headers.

```
ioctl(dmxfd, DMX_SET_PES_FILTER, &filter);
```

5) Open a DVR device.

```
dvrfd = open("/dev/dvb/adapter0/dvr0", O_RDONLY);
```

6) Read the data from the DVR device, which, if PID 8192 is used, is the complete transport stream with headers included.

```
read(dvrfd, buf, sizeof(buf));
```

The data can now be used for the purposes intended, which in our case was to pick out the transport_error_indicator bit in the transport stream header, to get an overview of the quality of the data received.

4 Tracking reception conditions using GPS data

Using the coordinates retrieved from the GPS device, it is possible to plot the reception conditions to an image, possibly overlaying some form of map.

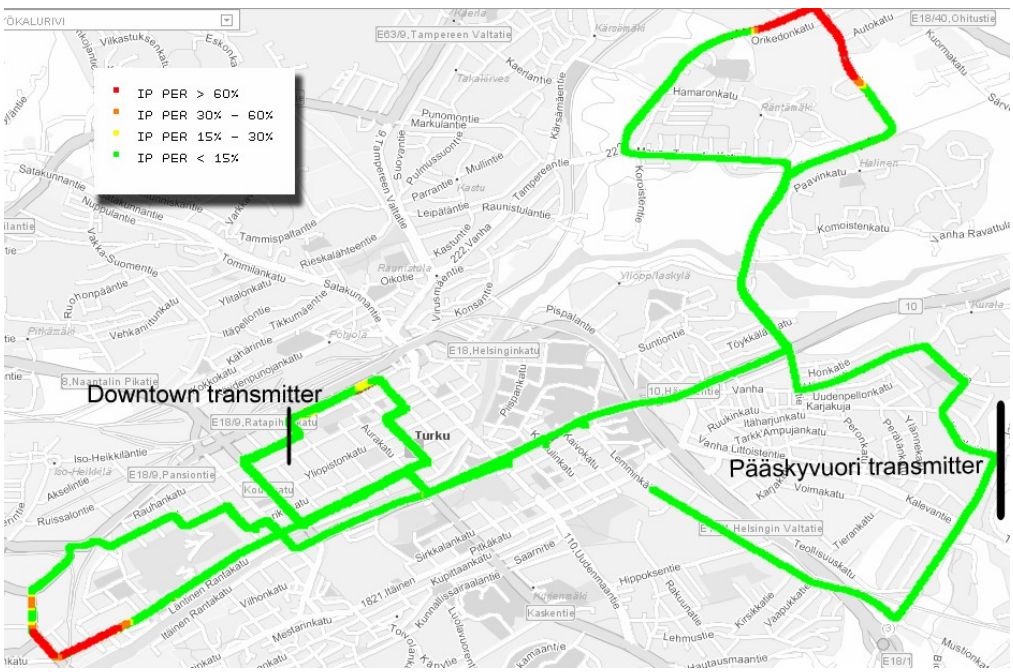


Figure 3. Example of a measurement route plot.

The GPS receiver is a BU-303 USB 1.1 device. Once every second, the GPS receiver generates various ASCII sentences containing the relevant information, such as longitude, latitude, velocity and time, as specified by the NMEA standard.

The GPS device requires signals from four satellites in order to be able to determine longitude and latitude. Based on previous coordinates, the device is also capable of calculating the velocity and direction the device is traveling.

The GPS receiver, though an USB device, acts as a serial device when plugged in, which makes it fairly easy a task to retrieve data from the device.

5 Logging reception conditions

The measurement program operates in two processes. One process polls the DVB device for TS packets and writes an error map, based on the TS_Error flag in the TS packet headers, to a shared memory structure. The process also writes other relevant information to the shared memory, such as device conditions (including signal strength), the total amount of packets received and the total amount of erroneous packets received.
Whenever the GPS device generates an interrupt, the other process stores the new data in a buffer. The second process also contains a timer, which generates an interrupt once every second. When that occurs, the process parses the data from the GPS device, reads the information from the shared memory and displays the current conditions on screen. The process then writes all the relevant information in two rows to a file: one row containing the error map and one row containing the time, longitude, latitude, DVB device status etc. In principle, each error map row should contain error flags for approximately 6750 TS packets for a 16-QAM transmission. However, in practice this is not the case: no error flags are written for TS packets that aren't received, and depending on the status of the buffer, the interval for reading from the buffer may fluctuate somewhat, resulting in a significantly uneven distribution of error flags on the error map rows.

6 Data layers for DVB-H multicasting

The same building blocks as in DVB-T [1], with a few optional modifications, comprise the physical layer of DVB-H [2]. The optional modifications include the new 4k mode, which is believed to offer an alternative solution to the existing 2k and 8k modes, with respect to Doppler tolerance and feasible single frequency network (SFN) cell size. DVB-H also proposes a new in-

depth interleaver mechanism to the DVB-H inner interleaver. The real difference between DVB-T and DVB-H is above the physical layer, also referred to as the link layer. Whereas DVB-T is dedicated to transmission of digital audio and video, DVB-H focuses on IP datacasting. These mechanisms are described in detail in the ETSI standard DVB specification for data broadcasting [3] and the ETSI standard Transmission System for Handheld Terminals (DVB-H) [2]. The link layer in a DVB-H system includes time slicing, multiprotocol encapsulation (MPE) and optional forward error correction for the multiprotocol encapsulated data (MPE-FEC). Time slicing is introduced in order to reduce the average power consumption for the reception terminal and also to enable smooth seamless frequency handovers between adjacent SFNs. The MPE is a mechanism for encapsulation of arbitrary protocols into sections. In case of DVB-H the encapsulated protocol is often IP. The optional MPE-FEC addresses C/N-performance improvement, Doppler tolerance in mobile channels and resistance against impulse interference.

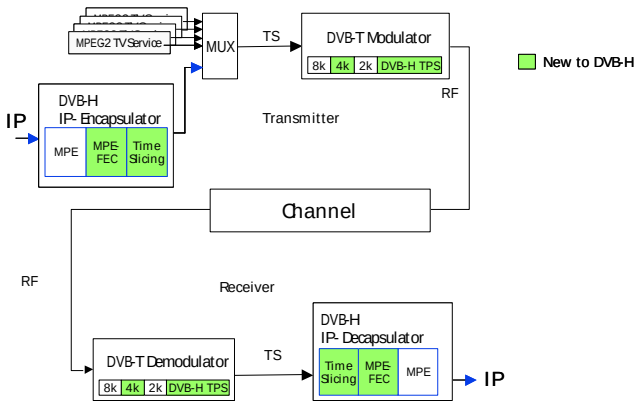


Figure 4. Block diagram for DVB-H transmission

Figure 4 shows an overview of and DVB-H transmission system. The IP Encapsulator is fed with IP packets belonging to a certain service and output consists of standardized mapping of MPE sections into MPEG-2 compliant transport stream. These TS packets may then be multiplexed together with other services and fed into the DVB-T Modulator. After travelling through the channel the OFDM symbols are received and the transmission process is reversed at the terminal side.

Multicasting IP data to endpoints utilize protocols from the standard Internet Protocol Suite (IPS). The IPv6/IPv4 encapsulates UDP datagrams. These protocols enable multicast data to be received by specific endpoints, defined by a combination of an IP address and a UDP port. The choice of using IP datagrams is mainly justified by the vast adoption of the IPS in networking.

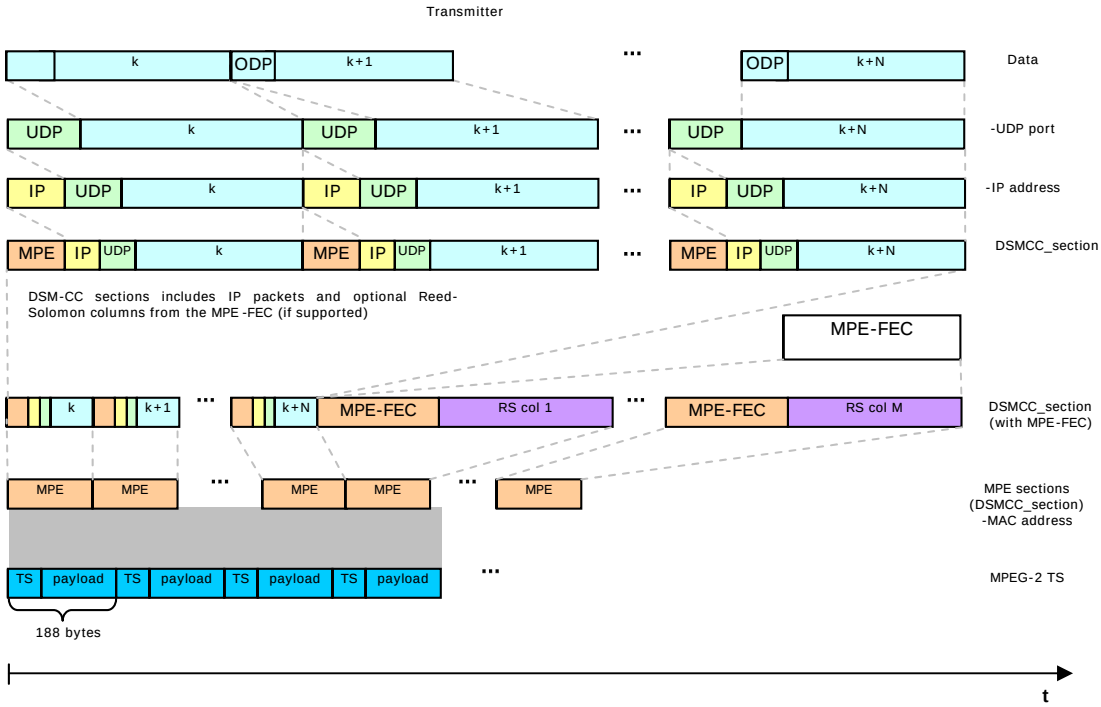


Figure 5. Data layers in DVB-H.

Figure 5 shows the different data layers in a DVB-H transmitter. In the measurement system described in this report a proprietary Object Delivery Protocol (ODP) is included above the UPD layer. The purpose of this protocol is merely to assist in the delivery of larger files over the multicast network. The file to be transmitted is split into fragments, which together with the ODP header fit into a single UDP datagram. The ODP header contains metadata about the payload. In a data carousel application, these object delivery fragments may be transmitted and received in an arbitrary order. The ODP header contains information about which object and which part of the object the payload belongs to, how large the object is and what kind of application layer coding is used. The object fragments are encapsulated into UDP datagrams and further into IP packets. The optional MPE-FEC sections, containing the Reed-Solomon parity bytes, are concatenated to a group of MPE sections, which contain the source IP packets. This ensemble is called an MPE frame. These sections are then mapped into MPEG-2 compliant TS packets, each 188 bytes long. The TS packets are then routed to the DVB-T/H physical layer.

6.1 The application layer

In mobile field measurements, the terminal experiences a great variety of reception conditions. Particularly when measurements occur in an environment where direct line-of-sight to the radio transmitter is not

available (e.g. in urban areas) and the measurement terminal is moving arbitrarily. Even if the actual distance from the transmitter is relatively constant, the quality of reception is constantly affected by building obstacles, multi-path reflections, Doppler phenomena, etc. The affect of these are constantly changing since the terminal is moving, which is not the case for DVB-T designed for fixed rooftop antenna reception. Mobile broadcast systems must be more robust in terms of criteria for antenna placement and size. The encoding of the data should consider the various reception environments that might be encountered. In order to account for all of this with technology designed for fixed rooftop reception the parameters of the transmission should be optimized and possibly the effective data rate decreased.

The measurement system presented in this report focuses on buffered services, i.e. for instance, an object download service. Typically, a binary object is a movie clip, application program, system software patch, document, or a music file. In many of these cases the criterion for the object downloading system is that the file must be delivered to the terminal without any errors. Since requested retransmission is generally not available in broadcast networks, the possibilities of recovering lost or erroneous packets are to incorporate forward error correcting (FEC) codes and/or to simply wait for the recurring transmission from the data carousel.

When measuring the quality of the application layer at least the following aspects may be taken into consideration:

1. The IP error rate (IP PER)
2. The time it takes to download a certain object
3. The overall efficiency of the encoding scheme. This can be calculated by energy per bit, i.e. how much energy is used for error free delivery of an object compared to the size of the actual source file. Naturally, the total amount of redundancy in the system affects this energy.
4. The number of data carousel rounds for successful decoding. How many times an object must be transmitted before an object can be completely received
5. The IP burst error length distribution. This may also include the length distribution of the burst lengths of correct packets

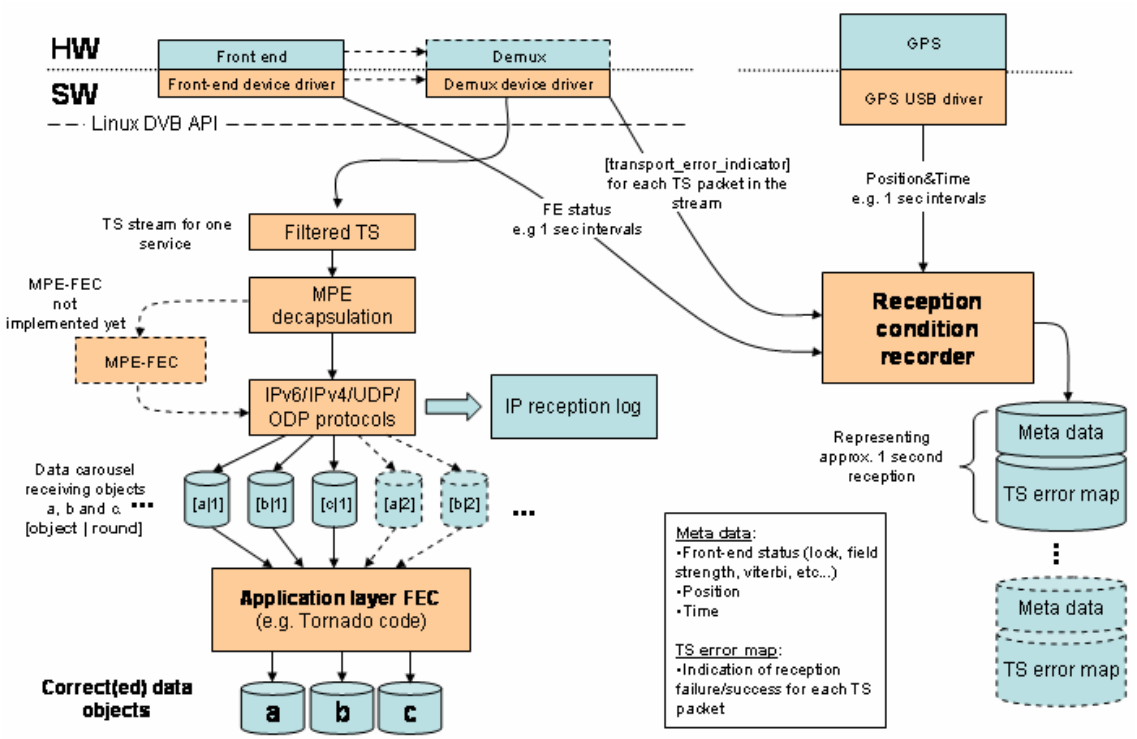


Figure 6. DVB-H field receiver block diagram.

Figure 6 shows a block diagram over the Åbo Akademi University DVB-H measurement receiver. The system consists on two parts: the object download application (to the left) and the reception condition recorder. In addition, this system builds complete IP reception log files. The ODP is designed to contain one extra field, which is used for acquiring precise information about which packets are lost during transmission. A simple two tuple IP packet/carousel round counter at the transmitter side assists the receiver in calculating the actual number of lost IP packets. This information is used for building simple reception success/failure maps on transmitted IP packets. A similar error map is output also from the reception condition receiver but for TS packets. The main difference is that the TS error map is based on the transport_error_indicatior flag embedded into the TS packet header. The flag is computed by the physical layer RS decoder. If the RS decoder is not capable of correcting the errors in a TS packet it will label that packet as erroneous. This implies that when a simple success/failure map is output, the TS packets that are lost during the time that the receiver is out of sync, i.e. outside the radio coverage area, will not show at all in the map. Therefore, the IP reception log is the best source for packet burst length analysis.

7 Application layer coding

Application layer coding (ALC) provides DVB-H networks with additional protection against data corruption. Desirable properties of application layer codes are long code lengths, efficient encoding, and decoding algorithms and large symbol sizes. When the target is file delivery, there is no tolerance for errors in the delivered objects. Therefore, the encoding of objects should include a sufficient amount of redundant data in order for the decoder to be able to reproduce the transmitted objects.

Recently, there have been several breakthroughs in the development of codes suitable for the application layer, such as Tornado codes, LT-codes [4] and Raptor codes [5]. Both the LT-code and the Raptor code are rateless codes that can potentially produce infinite streams of encoding packets, giving each receiver the possibility to wait for enough packets to arrive and then reproduce the original object, regardless of the IP packet error rate (IP PER) in the channel. Because file deliveries require successful decoding, what remains to be analyzed is suitable code rates for fixed-rate codes, guaranteeing this criterion and the average amount of carousel rounds required for error free delivery of objects. Because the reception conditions vary greatly in mobile networks, finding a suitable code rate for fixed-rate codes is a complex issue. Since LT-codes and Raptor codes can produce an infinite number of encoding symbols, these codes are not directly subject to this problem. On the other hand, producing greater numbers of encoding symbols than there are message symbols results in overhead, which is the same as reducing the code rate. The difference is that rateless codes gives the sender the possibility to produce as many encoding symbols as required for error free delivery on the fly while fixed-rate codes requires that the number of redundant symbols is determined in advance. In multicast networks, however, the receiver is likely not able to inform the sender that the entire object has been received and therefore, even when using rateless codes, some upper number of encoding symbols must be determined in advance. In this report, the focus is on the fixed-rate Tornado code, which has been slightly modified so that the probability of successful decoding is improved.

7.1 Tornado Codes

Tornado codes [6-8] are fixed rate, near optimal erasure correcting codes, suitable for multicasting of bulk data. The Tornado code consists of several sparse bipartite graphs in a cascading manner, as illustrated in figure 7. The rightmost graph can be replaced by some other forward error correcting code (FEC). The nodes in the graphs correspond to message symbols or check symbols. The size of the symbols can be chosen arbitrarily. Typically, they are chosen to equal the IP packet payload. A criterion is that all the symbols in a block are of equal length. The error correcting performance of

the code is critically dependent on the dependencies between the nodes. In [7] an analysis of the Tornado code structure is given.

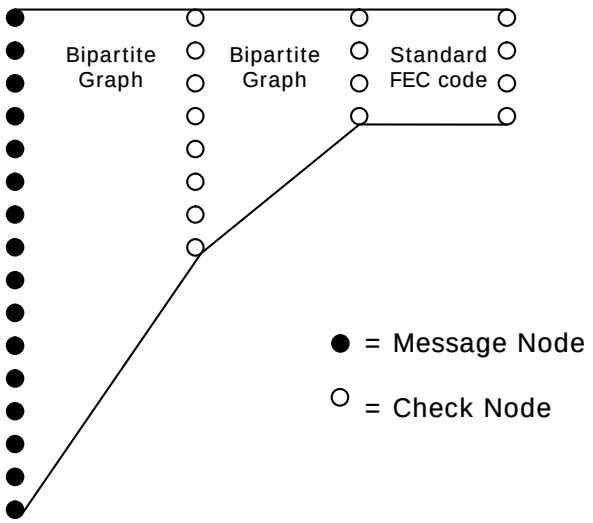


Figure 7. The structure of a Tornado code.

Tornado codes have the property that they require only a small fraction of the check symbols in order to successfully reconstruct the message symbols. The code is, however, quite vulnerable to burst errors and without interleaving mechanisms, the decoding process is prone to fail for higher error rates. When designed as a hyper code [9], the Tornado code has several dimensions, where each dimension is a high code rate Tornado code with the check symbols calculated on permutations of the message symbols. This design methodology results in a better resistance against burst errors, compared to standard Tornado codes. Tornado codes have the property that they can detect when enough data has arrived in order to reconstruct the entire object.

Encoding Tornado codes consists of calculating the check symbols in every bipartite graph as the exclusive-OR of all the other symbols with which the check symbol shares edges. Decoding Tornado codes consists of finding check symbols that know all their neighbors but one, where a neighbor is a symbol that is connected to the given check symbol, and then calculating that missing symbol as the exclusive-OR of the check symbol and all of the check symbols known neighbors.

From the algorithms for encoding and decoding Tornado codes, it can be seen that the time required for encoding and decoding is proportional to the symbol size, the code length and to the number of edges in the bipartite graphs. Because the Tornado codes are built up by sparse bipartite graphs, both the encoding and decoding speeds are very fast.

8 Analyzing the application layer test data

For field testing a 4,2 MB file was encoded with the Tornado code utilizing a code rate of 3/4 (4000,3000) and then encapsulated into IP packets. It should be noted that in this report the notion of a *file* is associated with the actual source data file and that the notion of an *object* is associated with the encoded source file including source data fragments, check data fragments, and ODP fragment headers. One fragment contains exactly one code symbol. The final IP packet size is in this case fixed to 1500 bytes. The encoded 6 MB object was transmitted using two transmitters, also shown in figure 3. The transmission was performed using QPSK modulation, 8 MHz bandwidth, 8k-mode, convolutional code rate 1/2, and guard interval 1/8.

An overview of the application layer system is given in the block diagram in figure 8. Since the ODP fragments can be received in arbitrary order, a block interleaver was used for reorganizing the IP packets belonging to a single object before transmission. The interleaving protects an object against longer burst errors. At the receiver terminal, the reception software extracts object fragments from IP packets destined to a specific object. Simultaneously, the application layer Tornado code corrects and re-constructs missing fragments. If errors occur to such an extent that the Tornado code is not able to correct all the errors which occurred during the reception of the object, the only alternative is to wait for the next transmission of that particular object from the data carousel. During empiric tests, the (4000,3000) Tornado code showed to correct up to approximately 19 % of randomly distributed packet erasures, and up to approximately 16 % of non-randomly distributed erasures (i.e. contains bursts of errors).

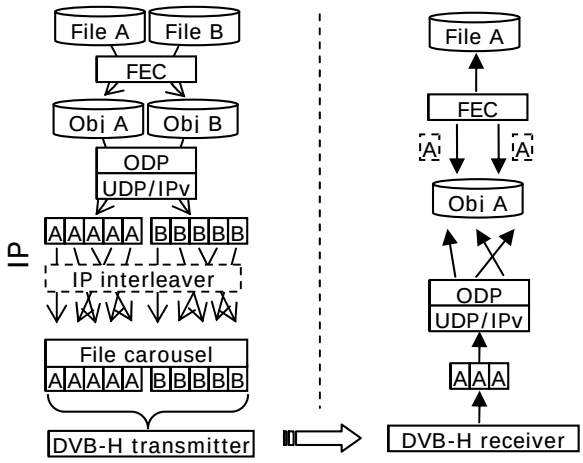


Figure 8. Block diagram of the file delivery system.

The route displayed in figure 3 was subject to measurement during the field tests. The measurement was performed by the receiver equipment in a car.

In figure 9, the measurement is represented as a sequence of completely received objects along the route. For each object, the number of carousel rounds required to receive the object completely is shown. One carousel round is here equal to the transmission of the entire object. Certainly, during the following carousel round the receiver must wait for exactly the missing fragments, which might be found at an arbitrary position in the file. This appears in the results as a fraction of complete carousel rounds. For instance, object number 31 required approximately 1,4 carousel rounds of data to be transmitted before the object was completely received.

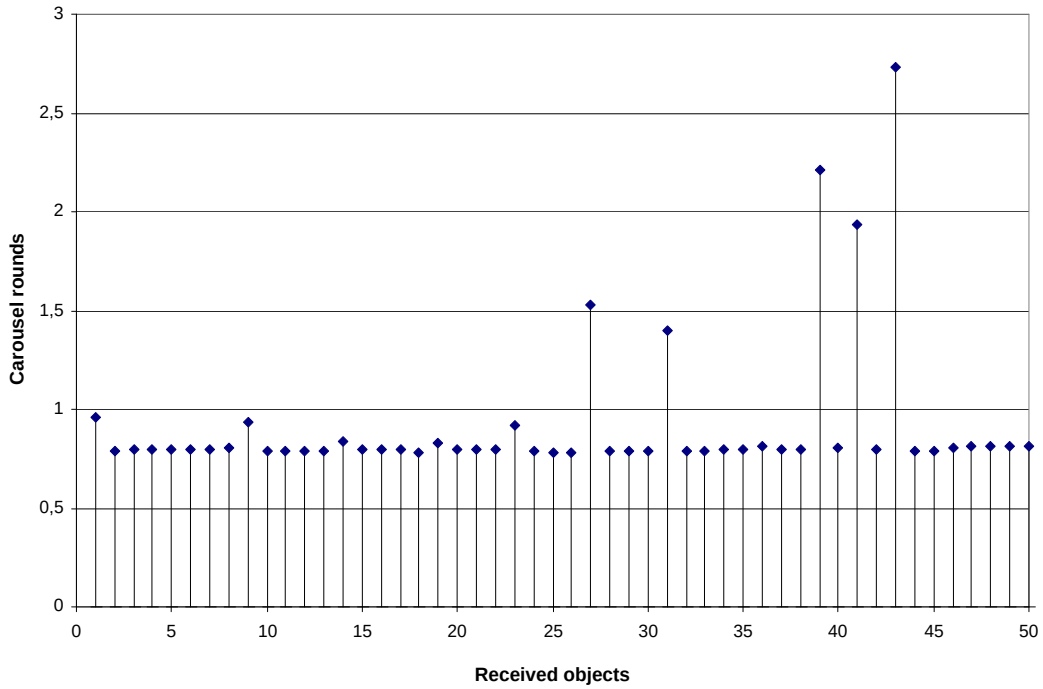


Figure 9. Number of carousel rounds needed for successful object delivery of a 4,2 MB source file, utilizing a Tornado 3/4.

If the reception conditions are very good and almost all transmitted packets are received, the object is complete when slightly more than 3/4 of the object fragments have arrived. However, if the fragments would be transmitted without interleaving, i.e. all the source symbols followed by the check symbols, the redundant check symbols would not be used and the object would be complete when exactly 3/4 of the data is received. Most of the objects in this measurement required approximately 0,8 carousel rounds of transmission to be completely received. This means that downloading a 4,2 MB file encoded with a (4000,3000) Tornado code, approximately 3200 (0,8*4000) interleaved object fragments (IP packets) must be received before the file is completely available at the terminal. The source data consists of 3000 fragments, which mean that there is an approximate 200-fragment overhead, in error-free delivery conditions, for the IP interleaver and the structure of the Tornado code. The justification

for including the interleaver is its ability to protect the system against longer burst errors.

Figure 10 shows an excerpt from the outdoor field measurement data. On average, 37,7 IP packets were transmitted per second. The curve shows the received IP packets per second during a period of 1500 seconds. The circles denote the points in time when the file download completed. The transmission of the object is constantly recurring. Figure 10 shows a typical situation when reception is almost impossible for long periods of time during a file download. Unsurprisingly, the downloading time increases when the reception is poor. However, this picture also illustrates that the FEC code recovers missing packets and that a second transmission is not always needed even if portions of the initial transmission were missed. The three first and the three last circles in the figure gives an impression of the error-free object reception pace.

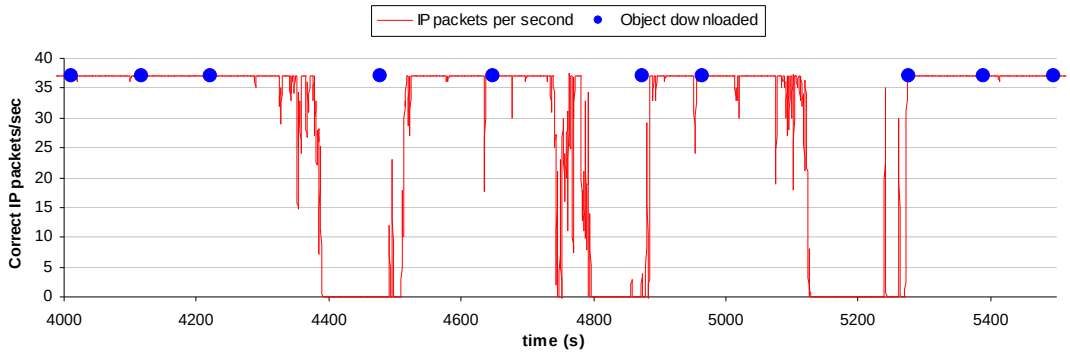


Figure 10. Average received IP packets per second and completed file downloads.

The measurement equipment also logs the C/N values read in one-second intervals from the frontend hardware. In figure 11, average C/N values have been calculated for the same download measurement as displayed in figure 9. The number of data points is too low for further statistical analysis, but it is clear that lower C/N values result in more reception errors increasing the downloading time.

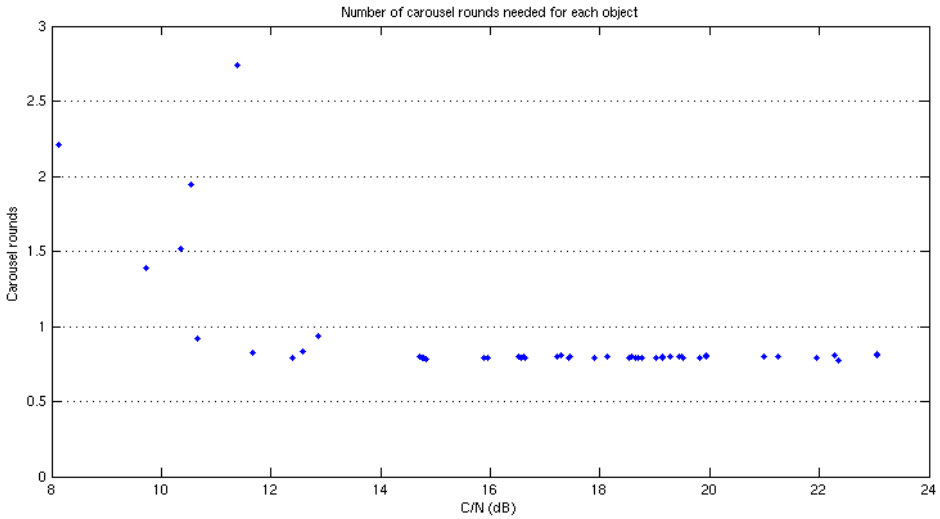


Figure 11. Outdoor measurement plot of the average C/N value and the number of carousel rounds needed for complete reception of the 4,2 MB file.

8.1 Analyzing ALC performance

Figure 12 shows how the application layer codes performed in terms of reconstructing received objects when the receiving terminal was indoors. The ideal code would successfully reconstruct every transmitted object in every carousel round and, hence, produce 18 objects in this time span. Because the Tornado code managed to reconstruct 14 of the 18 transmitted objects, one can see that the code is close to the ideal code. This implies that for good code performance, the curve for the code should be parallel to the ideal code. This was clearly not the case for the MPE-FEC.

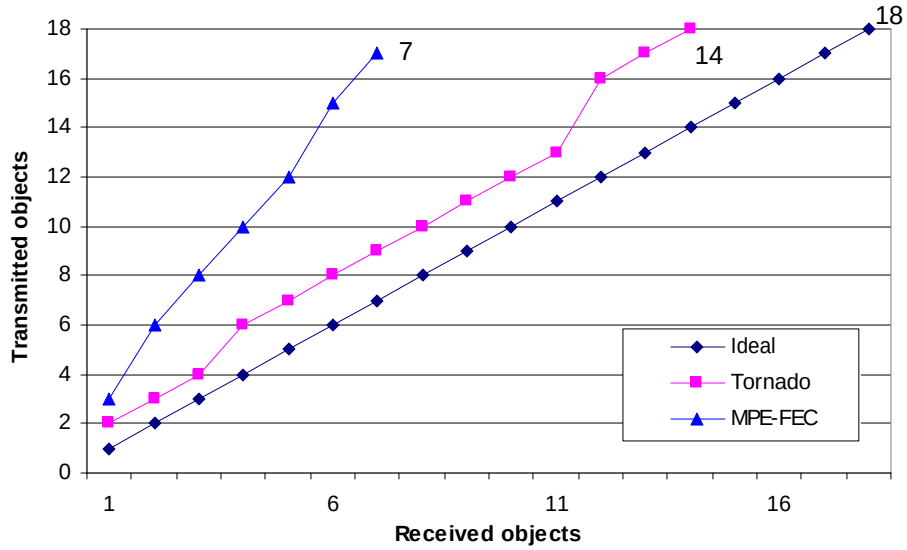


Figure 12. Example of an indoor measurement. Comparison between ideal, Tornado-based, and MPE-FEC reception of a 4,2 MB object. The code rate is 3/4 for both Tornado and MPE-FEC.

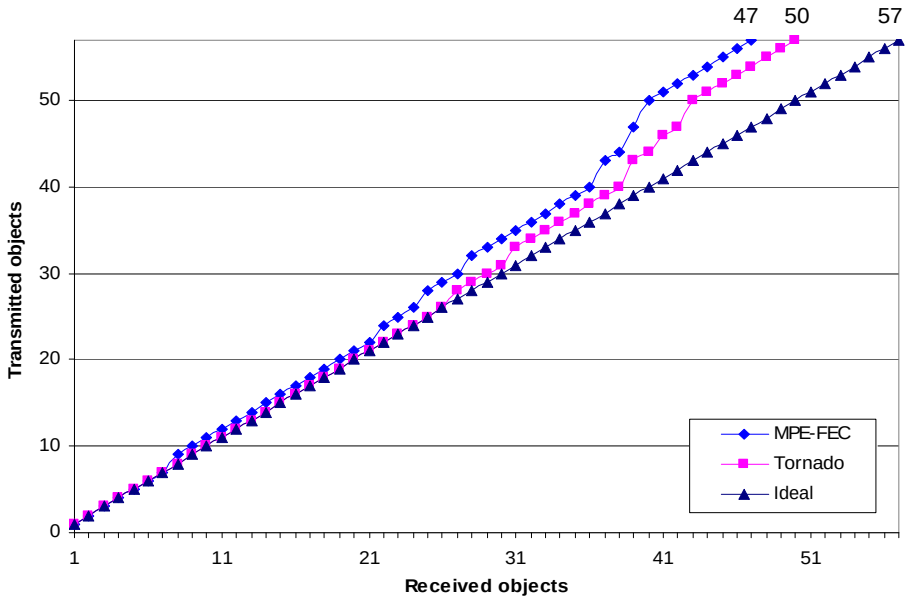


Figure 13. Example of an outdoor measurement. Comparison between ideal, Tornado-based, and MPE-FEC reception of a 4,2 MB object. The code rate is 3/4 for both Tornado and MPE-FEC.

When the receiving terminal was outdoors, the difference between performances of the MPE-FEC and the Tornado code was quite small. The outdoor measurement is shown in figure 13 and the figure reveals that both

of the codes reconstructed close to 90% of the transmitted objects. These results do not give much information about the performances of the codes since the IP PERs were very low, but rather proved that the system works quite well.

Figure 12 and figure 13 show the benefit of using application layer coding instead of the MPE-FEC, used in DVB-H, when delivering files over the network. In this scenario, the Tornado code is clearly more reliable than the MPE-FEC, since the transmitted object was successfully downloaded more often than with the MPE-FEC. This difference in efficiency comes from the longer code length of the Tornado code.

8.2 Burst length analysis

A tool for analyzing the burst length profiles provides "live" graphs such as seen in figure 14. The y-axis on the histogram represents the number of consecutive bad/good packets. The four histograms can be dimensioned individually, but in the case of figure 14 the graph in the upper left corner shows a 43 packet (typically the number of packets in an 256 row MPE-FEC frame) burst profile. The y-axis tick marks are positioned in 20 packet intervals in all histograms but in the lower right corner, where this interval equals 100. The x-axis shows the occurrence count of bad (red) and good (green) packet bursts of certain lengths. The window to the left displays data for a mobile indoor measurement and the window to the right for the outdoor measurement.

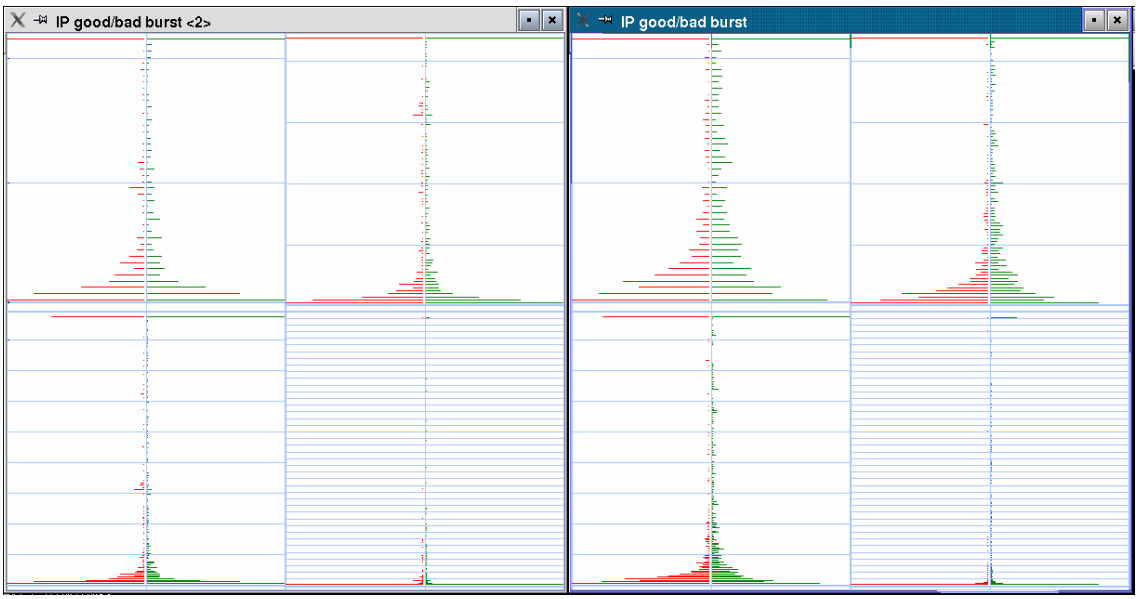


Figure 14. Plot of the IP error map as a burst length histogram. Indoor (left) and outdoor (right).

Another way of representing the results from such an analysis is to pair a consecutive bad and good burst, analyze their lengths, and combine these in a logarithmic scatter plot. In figure 15, the y-axis represents the length of a good burst and the x-axis the length of a bad burst. Data for an indoor measurement (to the left) and an outdoor measurement (to the right) are plotted in the figure. The area of the circle is proportional to the occurrence count of a particular bad/good burst pair. It can be seen that indoors contain pairs of long bad burst and long good bursts. This may be explained by the varying reception conditions, when for instance entering and exiting an elevator, resulting in a long bad burst followed buy a long good burst. The measurement is not in anyway comprehensive. It is included here for demonstration only.

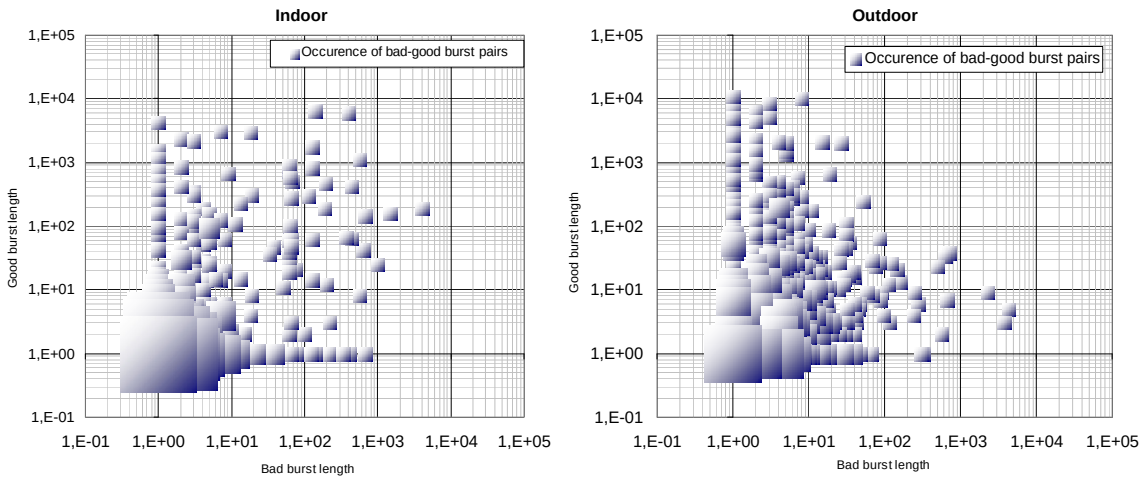


Figure 15. Bad/good burst pair plot.

9 Summary

This report presents a measurement system for DVB-H systems. The system is built using standard consumer equipment, i.e. normal PC hardware, DVB-T receiver using USB interface and GPS receiver. The measurement system was mainly built to collect Transport Stream (TS) error traces in normal field conditions, but can also be used as an online system evaluation tool.

As the system is built from consumer quality equipment, there is a certain uncertainty about how given signal strength indicators correspond to actual value. For this particular equipment, the system was calibrated using laboratory test equipment.

The data acquired using this system was used for several analyses after the actual measurement session. The recorded TS error traces was used as input in DVB-H system simulation models, implementing the higher layers of the DVB-H system. By using TS error trace, the upper layers can be simulating with different parameters, such as coding rates, block sizes etc. Even if the simulations do not exactly correspond to the actual system, the simulations give a good picture of final system performance.
The report shows how the recorded TS error traces can be used. An application layer coding system is evaluated using the system, showing improving the data transmission for file delivery systems, which require error free transmissions. In parallel a system simple implementation of a file delivery protocol is shown, which cooperates with the ALC system.
Using an analysis of the relative lengths of TS packet error bursts versus TS packet correct burst, a classification of the reception conditions can be made. Using this classification, the required strength of the error correction system can be made.
As conclusions, the report shows how standard equipment can be utilized to build a measurement system that can be used for rigorous assessment of DVB-H system performance. As many part of the communication system is digital, i.e. has deterministic properties, these parts can be modeled in software. However, the probabilistic properties given by antenna and receiver chip implementation varies from hardware instance to hardware instance, and should be subject to critical review before stating performance of measured system.

References

[1] "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television". (DVB-T), ETSI EN 300 744.

[2] "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)", ETSI EN 302304 V1.1.1, 2004.

[3] "Digital Video Broadcasting (DVB); DVB specification for data broadcasting". (DVB-DATA), ETSI EN 301192

[4] M. A. Shokrollahi, "Raptor Codes", in Proceedings of ISIT 2004.

[5] M. Luby, "LT Codes", in the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[6] M. Luby, M. Mitzenmacher, M.A. Shokrollahi, D.A. Spielman and V. Stemann, "Practical Loss-Resilient Codes", in Proceedings of the 29th Annual Symposium on Theory of Computing, 1997, pp. 150-159.

[7] M. Luby, M. Mitzenmacher, M.A. Shokrollahi and D.A. Spielman, “Efficient Erasure Correcting Codes”, IEEE Transactions on Information Theory, 2001, 47(2). pp. 569-584.

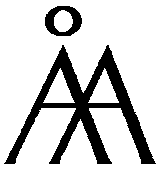
[8] J.W. Byers, M. Luby, M. Mitzenmacher and A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, in Proceedings of ACM SIGCOM’98, 1998, pp. 56-67.

[9] K. Nybom and J. Björkqvist, “Designing Tornado Codes as Hyper Codes for Improved Error Correcting Performance”, in Proceedings of AICT’06, 2006, to appear.

Turku
Centre *for*
Computer
Science

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics



Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 978-952-12-1856-9
ISSN 1239-1891