ROBUST IP AND UDP-LITE HEADER RECOVERY FOR PACKETIZED MULTIMEDIA TRANSMISSION

François Mériaux1,2, Michel Kieffer2

1Département EEA Ecole Normale Supérieure de Cachan
2L2S - CNRS - SUPELEC - Univ Paris-Sud

ABSTRACT

Recently, Joint Source-Channel Decoding (JSCD) techniques have been proposed to improve the reception of multimedia contents transmitted over error-prone channels. These techniques take advantage of the redundancy left by the source coder and of bit reliability measures (soft information) provided by channel decoders to correct transmission errors. To be put at work, protocol stacks have to be made permeable to transmission errors in order to allow soft information to reach the upper protocol layers. For that purpose, headers have to be reliably estimated at each protocol layer. First results have been obtained for lower protocol layers (PHY and MAC) protected by CRCs. The aim of this paper is to extend these results to upper protocol layers (IP and UDP-lite) protected by checksums. As for CRCs, trellis-based decoding techniques may be employed for data protected by checksums. Nevertheless, specific tools have been proposed in this paper to reach a complexity-efficiency trade-off1.

Index Terms— Codes, Communication systems, Transport protocols, Redundancy, MAP estimation.

1. INTRODUCTION

The delivery of multimedia contents over a bandlimited network requires the use of compression schemes to reduce the amount of data to transmit. Nevertheless, compressed data are highly sensitive to transmission errors: a single bit corrupted during its transmission may have heavy consequences on the full encoded content. To avoid such issue, contents which have to be transmitted through a network are packetized and protected with various error-correction and detection mechanisms. The basic protection is channel coding [1], which adds redundancy to the information as it is sent over the physical channel. Another protection mechanism is included in the network architecture and its different layers protocols. For some of those layers, an error-detection code is added to the message to protect the header of the packet. These error-detection codes can be CRCs or checksums [1, 4]. At the receiver, these codes are used to decide whether the message is corrupted and if it is, the sender is asked to send again the packet, if possible.

Nevertheless, there are many applications where packets cannot be sent again: broadcasting, voice over IP, etc. For the last few years, Joint Source-Channel Decoding techniques [3] have been proposed to deal with corrupted packets by correcting them using various sources of redundancy present in the whole coding and transmission chain. These techniques improve the decoding but as they treat the corrupted packets at the application layer, they require the network to be permeable to such erroneous packets. The current network architecture does not allow packets with corrupted headers to reach the application layer because of the error-detection codes.

In [5], a mechanism is proposed to allow the lower protocol layers (PHY and MAC) to be permeable to corrupted packets by using the CRCs in conjunction with inter- and intra-layer redundancy to correct the erroneous headers.

The main contribution of this paper is to extend the ideas of [5] to the IP and UDP-lite protocol layers. The headers of these layers are protected by checksums. As for CRCs, trellis-based decoding techniques [7] may be employed for data protected by checksums. Nevertheless, to reach a complexity-efficiency trade-off, specific techniques have been proposed here.

The paper is organized as follow. Section 2 explains the header correction mechanism, then Section 3 introduces an efficient way to reduce its complexity. Finally, Section 4 presents the simulation context and the results of our method.

2. ROBUST HEADER RECOVERY USING CHECKSUMS

2.1. MAP estimator for robust header recovery

In this section, the formalism of [5] is used. Consider a packet which has to be forwarded from a layer \( L \) to layer \( L+1 \). This packet is composed of a payload and a header. In this header, some fields, gathered in a vector denoted by \( r \) are protected by a checksum. Many fields in \( r \) are a priori unknown but using the various sources of inter- and intra-layer redundancy, some of them may be known or predictable. Then \( r \) may be partitioned as follows \( r = [k, p, u] \), where \( u, k, \) and \( p \) are

\[ r = [k, p, u] \]

This work has been partly supported by the ANR DITEMOI project and the European NoE NEWCOM++.

1
respectively the unknown, known, and predictable fields of $r$. The checksum vector $c_n = C_n(r)$, with $C_n$ is obtained using an $n$-bit checksum evaluation function and we denote by $R$, the implicit information at the receiver which allows to limit the implicit set of unknown fields. The soft information coming from Layer $L$ are grouped in the vector $y = \{y_k, y_p, y_u, y_c, n\}$, $y_k, y_p, y_u$, and $y_c$ being respectively the noisy observations of the known, predictable, unknown, and checksum fields.

Since the fields $k$ and $p$ are known or exactly predictable, the only remaining field to be estimated is $u$. This problem can be solved with a MAP estimator

$$\hat{u}_{\text{MAP}} = \arg \max_u P(u \mid k, p, R, y_u, y_c)$$

After some derivations detailed in [5], one gets

$$\hat{u}_{\text{MAP}} = \arg \max_u P(y_u \mid u)P(y_c \mid C_n(k, p, u))P(u \mid R) \quad (1)$$

2.2. Practical evaluation of the MAP estimator

In (2), the first term $P(y_u \mid u)$ represents the likelihood of the sequence $u$ when having the soft information $y_u$. The second term $P(y_c \mid C_n(k, p, u))$ represents the likelihood of the checksum of the sequence $r = [k, p, u]$. The last term $P(u \mid R)$ limits the range of possible values for $u$.

If we consider $\ell(u)$, the length in bits of the unknown vector $u$, $2^{\ell(u)}$ possible combinations for $u$ have to be considered. Since $\ell(u)$ can be quite long, evaluating $P(y_u \mid u)$ may be prohibitively complex. To reduce the computational load, a trellis structure similar to the one described in [7] is used to compute $P(y_u \mid u)$, see [5] for more details. In our case, the states of the trellis correspond to the different possible values of the checksum and each column of the trellis corresponds to one bit of the unknown fields. Each bit of the unknown sequence leads to two possible values of the checksum, and thus leads to two different branches in the trellis. Upon reaching the end of the trellis, the likelihood of $2^{\ell(c)}$ different possible sequences and of their associated checksums is obtained.

To compute $P(y_c \mid C_n(k, p, u))$, it suffices to compare these checksums with $y_c$ by the mean of a likelihood metric.

3. COMPLEXITY REDUCTION

Both IP and UDP checksums are coded over 16 bits, leading to trellises with $2^{16}$ states, requiring a huge amount of computation. The complexity has to be significantly reduced to get an implementable decoding algorithm. The solution proposed in [5] cannot be applied in our case, since a checksum cannot be divided into several independent pieces, as done for a CRC.

3.1. From a $2n$-bit checksum to a $n$-bit checksum

As detailed in [2], for a given sequence of bits, it is possible to obtain its $n$-bit checksum from its $2n$-bit checksum. Let us consider $K$ words $W_k$, $k \in [1, K]$ of $2n$ bits. One can divide each $W_k$ into two $n$-bit words such as the first one $W^l_k$ is made of the $n$ most significant bits from $W_k$ and the second one $W^r_k$ is made of the $n$ least significant bits from $W_k$. One has then $W_k = W^l_k \times 2^n + W^r_k$ and

$$C_n(C_{2n}(W_1, ..., W_K)) = C_n(W^l_1, W^l_1, ..., W^l_K, W^r_K)$$

(3)

Practically, once we have the $2n$-bit checksum $c_{2n}$ of a given sequence, we just do the 1’s complement sum of of the $n$ most significant bits of $c_{2n}$ with the $n$ least significant bits of $c_{2n}$ to obtain the $n$-bit checksum $c_n$ of the same sequence. This property enables us to work with a $n$-bit checksum rather than a $2n$-bit one so the complexity of the sequence estimation is highly reduced. But the counterpart is that a $n$-bit checksum is a worse protection than a $2n$-bit one. If only error correction was involved, the results would not be as good as with the $2n$-bits checksum, however, the checksum is now used in conjunction with other properties of the header to be protected, and it is shown in Section 4.2 that overall, there is a large improvement on performance.

3.2. Computing the bit-a posteriori probabilities of the $n$-bit checksum

Assuming that a posteriori probability of each bit of a $2n$-bit checksum computed over a given sequence is known, one has to determine the a posteriori probability of each bit of the $n$-bit checksum computed over the same sequence.

To solve this problem, we have to take a closer look at the relations between the bits of $c_n$ and the bits of $c_{2n}$ and for this, we introduce the following notations

- $P_i$ is the probability for the $i$-th bit of $c_{2n}$ to be a 1, $i \in [0, 2n - 1]$. Reciprocally, we note $P_i^c$, the probability for the $i$-th bit of $c_{2n}$ to be a 0.
- $p_i$, $i \in [0, n - 1]$, the probability for the $i$-th bit of $c_n$ to be a 1 and $\overline{p}_i$, the probability for the same bit to be a 0.
- $q_i$, $i \in [0, n - 1]$, the probability to have a carry at the $i$-th position of the 1’s complement sum between the two mid-parts of $c_{2n}$. Reciprocally, $\overline{q}_i$ is the probability not to have a carry at the same position.

Considering the $i$-th bit of $c_n$, we have two possibilities depending on whether there is a carry. If there is a carry, the bit is a 1 if both $i$-th and $i+n$-th bits of $c_{2n}$ are 1 or both bits are 0. In the other case, if there is no carry from the previous step, the $i$-th bit of $c_n$ is a 1 if one of the $i$-th and $i+n$-th bits of $c_{2n}$ is a 1 and the other is a 0. This gives us the following equations

$$\begin{cases}
p_0 = q_{n-1}(P_0 \overline{P}_n + P_0 P_n) + \overline{q}_{n-1}(P_0 P_n + \overline{P}_0 P_n) \\
p_i = q_{i-1}(P_i \overline{P}_{i+n} + P_i P_{i+n}) + \overline{q}_{i-1}(P_i P_{i+n} + \overline{P}_i P_{i+n}), \forall i \in [1, n - 1]
\end{cases}$$

(4)
In (4), the \( P_i \) (and thus the \( \overline{P}_i \)) terms are known since they are contained in the soft information, but \( q_i \) and \( \overline{q}_i \) remain unknown for the moment. To determine the value of the \( i \)-th carry, we have to consider two possibilities: either both \( i \)-th and \( i+n \)-th bits of \( c_{2n} \) are 1, or one of those bits is a 1 and the other a 0. In the first case, there is always a carry. In the second case, the \( i \)-th carry is only a 1 if the \( i+1 \)-th carry is a 1 too. This enables us to write the following system of equations

\[
\begin{align*}
q_0 &= q_{n-1}(P_0\overline{P}_n + \overline{P}_0P_n) + P_0P_n \\
q_j &= q_{j-1}(P_j\overline{P}_{j+n} + \overline{P}_jP_{j+n}) + P_jP_{j+n}, \forall j \in [1, n-1]
\end{align*}
\]

The system (5) is made of \( n \) equations and has \( n \) unknown parameters \( q_i, i \in [0, n-1] \). It can be written as

\[
Aq = b
\]

with

\[
\begin{align*}
q &= (q_0, q_1, ... , q_{n-1})^t \\
b &= (P_0, P_{n}, P_1, P_{n+1}, ... , P_{n-1}, P_{2n-1})^t \\
A &= \begin{pmatrix}
1 & 0 & \cdots & 0 & -k_0 \\
& -k_1 & 1 & \cdots & 0 \\
& 0 & -k_2 & \cdots & \vdots \\
& \vdots & \vdots & \ddots & \vdots \\
& 0 & \cdots & 0 & -k_{n-1} & 1
\end{pmatrix}
\end{align*}
\]

and

\[
k_j = P_j\overline{P}_{j+n} + \overline{P}_jP_{j+n}, \forall j \in [0, n-1]
\]

from which \( q \) is easily obtained inverting \( A \). Once \( q \) is known, we can use the values \( q_i, i \in [0, n-1] \) in (4) in order to obtain \( p_i, i \in [0, n-1] \).

### 3.3. A simplified trellis

Once the a posteriori probability of the \( n \)-bit checksum is known, we have to estimate the unknown sequence. As in Section 2.2, we use a trellis structure to do that. The only difference is that the trellis now has \( 2^n \) states instead of \( 2^{2n} \). The number of operations to be done with this trellis is almost reduced by a factor \( 2^n \), the complexity is highly reduced. The only calculations we add with this process is the solving of (6) and have a complexity \( O(n^3) \), which remains negligible compared to the factor \( 2^n \).

### 4. SIMULATION AND RESULTS

#### 4.1. Context

The proposed simulation consists in sending packets through an AWGN channel and applying our algorithm to the packets with erroneous headers at the reception. We assume that a soft protocol stack is used at the lower layers, so that soft information reach the IP and UDP-lite layers. The transparent network architecture presented in [6] gives some insights on the way to transmit soft information between protocol layers. Moreover, the UDP-lite layer is tuned to only protect the header and in both IP and UDP-lite headers, some fields may be known or predictable.

Here are the assumptions we made for those fields:

- In UDP-lite, the Length field is considered as known since we know the way we use UDP-lite. The Terminal Port field is unknown but one can assume that a limited number of ports are used at the same time. In our case, it can only take 8 different values.
- In IP, The Version field is known since we use IPv4. The Header Size field is considered as known. The Type of Service field is also assumed to be known. The Length field is considered as predictable as it can be obtained from the MAC layer. The first bit of the Flags field is reserved and set to 0, so it is known. The Protocol field is known as UDP is known to be the upper layer.

The remaining fields are all considered as unknown.

#### 4.2. Results

At the receiver, we compare the HER (Header Error Rate) of the received packets for different values of the SNR. For IP layer, we compare three different decoders: the standard one which decodes the bits of the header sequence according to their a posteriori probabilities, our header-correcter decoder based on the redundancy included in the 16-bit checksum and the same decoder with reduced complexity as explained in Section 3.2.
4.2.1. IP header correction

Figure 1 shows this comparison for the IP header. As expected, both proposed decoders show better results than the standard one. Header correction based on the 16-bit checksum always results in the smaller HER, whereas the reduced complexity one is always between the two other decoders in term of results. We explain this by the fact that when we work with the reduced complexity version, we turn the a posteriori probability of the real 16-bit checksum into a a posteriori probability corresponding 8-bit checksum. But this operation is not reversible and redundancy about the unknown sequence is lost. Compared to classical algorithm, however, the use of headers properties compensates more than this loss.

If we have a closer look at the uncorrected error patterns that remain after we tried to correct them, we clearly notice that there is no single bit error in the corrected sequence. This due to the checksum property of always detecting single bit error. The most common type of error appears when two bits separated by 15 (or 7 for the reduced complexity version) bits are both wrong but when added to each other, they do not change the checksum value. This kind of error is not detectable by our algorithm.

4.2.2. UDP-lite header correction

Figure 2 compares the performance of the standard decoder and the basic header-correcter decoder for the UDP-lite layer. Once again, our decoder has better results than the standard one and if we compare it with the same decoder at the IP layer, we notice that the results are even better. For a HER of $10^{-2}$, our UDP decoder needs 3.33 dB whereas the IP decoder needs 4.75 dB. We explain this observation by two facts. First, the length of the unknown sequence in the UDP-lite header is smaller that the unknown sequence of the IP header. It is 32 bits instead of 39, there are fewer chances for errors to occur. Second, in the UDP-lite header, the Terminal Port field can only take a given number of values which correspond to the opened ports of the Terminal, which highly reduces the possible values for this field.

5. CONCLUSION AND PERSPECTIVE

By using the redundancy due to intra and inter-layer correlation and to the presence of a checksum, we propose a robust IP and UDP-lite header recovery technique. This tool allows the upper protocol layers to be permeable to corrupted packets.

An optimal MAP estimation algorithm and a suboptimal, but computationally more tractable algorithm are proposed in this paper. Both have better performance than the conventional decoder. Some work has still to be done on complexity reduction without introducing too much suboptimality. Instead of replacing the 16-bit checksum by a corresponding 8-bit checksum, it would be interesting to keep the 16-bit checksum and try to estimate the unknown sequence with two trellis of $2^8$ states, one corresponding to the 8 most significant bits of the 16-bit checksum and the other corresponding to 8 least significant bits. Of course, they would not be independent, and have to share information about their respective carries. This way, we would still have a reduced complexity but our sequence estimation should be better.

Acknowledgements

The authors would like to thank Pierre Duhamel for his useful inputs.

6. REFERENCES


