EAI Endorsed Transactions
on Ubiquitous Environments

Research Article

A packet error recovery scheme for vertical handovers mobility management protocols★

Pierre-Ugo Tournoux[1,*], Emmanuel Lochin[2], Henrik Petander[1], Jérôme Lacan[2]

[1]NICTA, Australian Technology Park, Eveleigh, NSW, Australia
[2]Université de Toulouse - ISAE ; 10 av. Edouard Belin - BP 54032 - 31055 Toulouse Cedex 4, France

Abstract

Mobile devices are connecting to the Internet through an increasingly heterogeneous network environment. This connectivity via multiple types of wireless networks allows the mobile devices to take advantage of the high speed and the low cost of wireless local area networks and the large coverage of wireless wide area networks. In this context, we propose a new handoff framework for switching seamlessly between the different network technologies by taking advantage of the temporary availability of both the old and the new network technology through the use of an "on the fly" erasure coding method. The goal is to demonstrate that our framework, based on a real implementation of such coding scheme, 1) allows the application to achieve higher goodput rate compared to existing bicasting proposals and other erasure coding schemes; 2) is easy to configure and as a result 3) is a perfect candidate to ensure the reliability of vertical handovers mobility management protocols. In this paper, we present the implementation of such framework and show that our proposal allows to maintain the TCP goodput (with a negligible transmission overhead) while providing in a timely manner a full reliability in challenged conditions.

Keywords: Mobility management protocol, Vertical handover, On-the-fly coding

Received on 18 August 2011; accepted on 2 July 2012; published on 24 October 2013

doi:10.4108/trans.ue.1.2.e3

1. Introduction

With the proliferation of new wireless access network technologies, mobile users can now access the Internet using multiple types of access network technologies. The characteristics of these access networks vary greatly; Wireless Local Area Networks (WLANs) provide high speed access with a network latency of tens of milliseconds, often at the price of fixed Internet access but with a very limited coverage. Wireless Wide Area Networks (WWANs) on the other hand provide wide coverage but have a significantly lower data rate, higher latencies up to several hundreds of milliseconds and a cost which may be several magnitudes larger than that of WLAN networks. For obvious cost and performance reasons, smartphone users frequently switch to WLAN when a hotspot is available although the cellular connection is almost always enabled. Therefore the ability to switch seamlessly between these different technologies allows a user to maximize his data rates and an operator to free resources in more expensive WWAN networks by maximizing the utilization of lower cost WLAN networks.

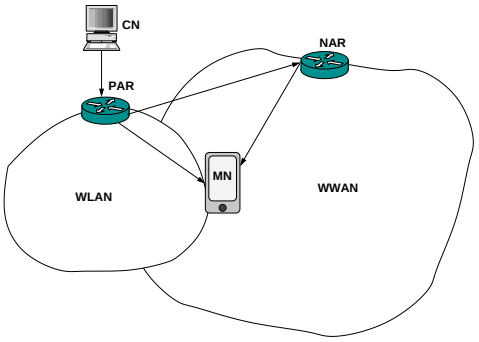
Seamless switching between heterogeneous access networks requires carefully managed vertical (inter-technology) handovers. Protocols, such as Mobile IP (see RFC 3344), can be used to ensure the handover does not break the on-going connections of a mobile node and that the mobile node remains reachable in spite of the handover. In a Mobile IP vertical handoff, on-going traffic is often disrupted due to protocol deficiencies [1]. Although, more advanced handoff protocols such the Safetynet architecture [2] (and Safetynet v.2 [3], where the use of FEC codes are suggested to mitigate the number of lost packets during the bicasting) can be used to reduce these packet losses, the challenging wireless link conditions triggering the handoff may cause unavoidable packet losses. This is especially the case for upward vertical handovers (i.e. handovers

★Part of these results have been presented in ICST Mobiquitous 2010
*Corresponding author. Email: pierre-ugo.tournoux@nicta.com.au

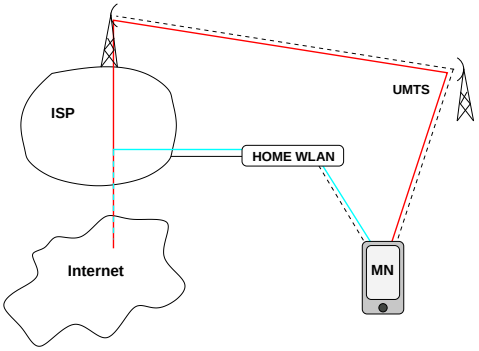
from WLAN to WWAN networks) which are typically performed only when the signal to noise ratio of the WLAN becomes too weak to provide a correct connectivity. This decrease of the signal strength may result in packet losses due to wireless errors or even a complete loss of connectivity with the Previous Access Router (PAR) during the time it takes to prepare the WWAN interface and link layer connection to the Next Access Router (NAR).

There are currently two main classes of solutions that address the problem from a transport layer point of view. The first one aims to improve TCP tolerance to handover [4, 5] while the second one uses multipath SCTP [6] version to benefit from this multiple connectivity capability [7, 8]. In the present contribution, we show that the proposed coding scheme shares both advantages and allows the use of any kind of transport protocol without modification. In other words, this proposal is completely independent of the end-to-end transport solution deployed.

In this paper, we present the use an “on the fly” coding scheme, Tetrys [9], which can be applied to transport and layer-3 mobility management protocols to achieve a so called "soft handover" and significantly reduce the impact of the handover on the application traffic. A soft handover allows a mobile device to connect to multiple networks at the same time and receive coded streams of traffic from multiple routers or base stations at the same time and to combine those coded, partially redundant streams to a single, complete data stream. This allows the handover process to be very smooth since the ratio of data and the level of coding of the different streams can be dynamically adjusted to handle changing packet loss rates. So far soft handovers have been successfully used only in tightly controlled horizontal handovers in CDMA networks in which the traffic is synchronized between the different base stations and the mobile device. This paper explores whether a similar soft handover can be achieved in handovers between IP based WLAN and WWAN networks with the more challenging asymmetric and non-synchronized network conditions. The purpose of this study is not to propose yet another exhaustive mobility management architecture as this generic coding scheme could be used inside any mobility management protocol (such as Fast Handovers Mobile IPv6 for instance). We rather seek to demonstrate that our adaptive coding scheme can significantly reduce the impact of the challenged network conditions in a vertical handover by using a soft handover like approach. Thus, we evaluate our proposal for vertical soft handovers and the results obtained show that TCP remains close to its pre-handoff bitrate.



(a) Standard handover with the use of two interfaces



(b) Opportunistic use of two interfaces without HO

Figure 1. Two illustrations of multipath.

2. Background and related work

A soft vertical handover differs from a hard vertical handover as no disconnection occurs during the soft handover process. Although both are challenging, they impact the transport and application layers differently and also the methods used to minimize the handover effect at both layers are different.

The main problem in hard handovers, and also failed soft handovers, is the handover delay during which packets are lost. These lost packets cause a number of issues, discussed below. Solutions for reducing this delay, i.e. optimizations for hard handovers, consist of reducing the network detection period, the address configuration interval and the network registration time which reduce (see [1]). The solutions are also valid for reducing the chances of a soft handover turning into a hard handover.

The first and the most pressing issue that a transport protocol, such as TCP, must handle is the loss of its in-flight packets. This handover delay might trigger RTO in TCP and the resulting backoff procedure could lead to a connection stall. This problem can be mitigated by freezing the sending of TCP packets during the handover process [4]. Another challenge in hard and soft handovers is the possible differences in network characteristics between the new and the previous network in terms of propagation delay, bandwidth and

packet loss rate (PLR). This problem is often addressed by adapting the TCP congestion windows length or to quickly update the TCP RTO value on the new link (involving active probing on the new link) in order to prevent RTO buffer overflow and/or RTO expiration [4, 5].

As the multiple interfaces present on a mobile host can be enabled at the same time, many proposals provide soft handovers between heterogeneous networks. This can be achieved with IP-level mobility solutions such as MIH, Multihoming MIP (see RFC 4908 [10]) at the cost of a devoted network infrastructure. This provides (see RFC 4980 [11]) improved reliability, load sharing between the different links and bandwidth aggregation.

Several proposals use multihoming to improve the quality of the communication by bicasting the flows via multiple available interfaces. When a handover becomes highly probable, packets are sent both from the PAR and the NAR. This allows a more robust service as the signal to noise ratio can significantly decrease during the HO leading to high packet loss rate (PLR) as shown in Fig. 1(a). Instead of copying the same packets on both paths, the authors of [12] choose to send data packets on one path and redundant FEC packets on the other, thus reducing drastically the impact of losses on a video stream compared to the standard bicasting procedure. In [13], the authors obtain a similar result using staggered FEC.

Several modifications to the Stream Control Transmission Protocol (SCTP) have been proposed, enabling support for both real-time and non real-time traffic [7, 8] taking benefit of the multihoming capability without the cost of a large network architecture deployment. For instance, the authors in [8] demonstrate that the use of SCTP allowed them to aggregate the bandwidth of different networks, thus providing a video with a better quality and a more robust service even in high mobility scenario.

To illustrate the generic character of our solution, another application where our proposal is of interest is illustrated in Fig. 1(b). In this figure, we represent a mobile node which has subscribed to both ADSL and 3G offer within the same operator and opportunistically uses its home wireless access (or any accessible local wireless spot) conjointly with a 3G access. The benefit of our proposal in this context will be highlighted in Section 4. Our scheme can be deployed both in a end-to-end fashion or from the edge router of the ISP to the terminal host while remaining transparent to the application layer. On the contrary, SCTP needs an end-to-end deployment and the use of specific applications built on top of SCTP socket.

Which issues have still not been addressed?

Adapting TCP to the context of handover would require changing every transport protocol stack already deployed to support these changes. In addition, defining the parameters of these TCP modifications (new values such as RTO and congestion window: *cwnd*, adapted to the new link) require a probing delay or a certain level of a-priori knowledge of the links characteristics which may not always be available. Finally they do not take advantage of the diversity of the links. The main drawback related to the solutions based on multipath SCTP is that they require modifying the interface between the application and the transport layer. Additionally, to date, the deployment of SCTP has been limited to Unix-like hosts. This requirement would make the deployment of these proposals harder. The bi-casting proposals, even when they involve FEC coding, result in halving the available bandwidth which might already be limited in the case of WWAN interfaces. Further, with the exception of the bi-casting procedure, none of these proposals seem to perform well in challenged conditions with high PLR common before and after handover due to poor signal quality.

To the best of our knowledge, there currently exists no solution allowing to benefit from the robustness and bandwidth aggregation provided by the multihoming capability of new devices with an application on top of the standard TCP/IP protocol suite (i.e. without any modification of the suite).

All these facts motivate our proposal described in the following section 3.

3. Our proposal

We present in this section the architecture and internal mechanisms that define our framework proposal.

3.1. Architecture, coding and handover

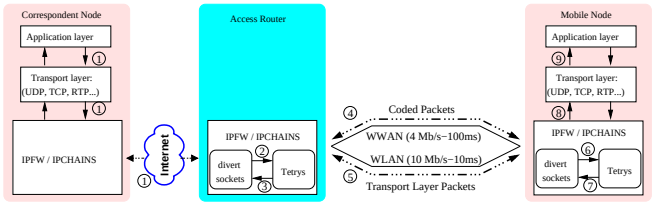


Figure 2. How to plug our coding scheme in the network protocol stack.

Our coding scheme, detailed in the next section, allows recovering in a timely manner from all the losses that occurred on a path, regardless of their distribution. The sole requirement is that in average the amount of redundant packet sent must be greater than the amount of the losses. The key idea is to use our proposal to

enhance the part of a path that can be affected by PLR losses during a handover. The whole path can be protected by coding and decoding the packets at both ends of the connection if the Correspondent Node (CN) is aware of the different addresses of the Mobile Node (MN). Otherwise, as Fig. 2 suggests, in the case where multihoming is provided by an IP-level mobility solution, the coding/decoding can also be done between the Access Router (AR) and the MN. We propose to use our coding scheme at layer-3, thus hiding the losses to the transport layer. Fig. 2, describes a possible way to plug our coding scheme through the use of Divert Socket. In our case, we used the BSD implementation of divert socket which is also provided under GNU/Linux with `ipchains` API.

The sending of a data segment from the CN to the MN works as follows: as a first step, packets travels normally through the TCP/IP protocol stack as the CN is not involved in the coding process. Packets that reach the Access router cross the IP forwarding rules (IPFW) which diverts (second step) the packets destined to the MN to the related Tetrys instance (there might be one instance by MN supported by the AR). Tetrys adds the packets to its encoding window and re-injects them (step 3 and 4) adding a packet sequence number (three bytes might be more than necessary) plus a bit to distinguish redundancy from source data packets inside the IP option field. Redundancy (i.e. coded) packets are injected with such an IP destination address that they go through the WWAN (step 4) or WLAN (step 5) links depending on whether it is an upward or downward vertical handover. The size of these coded packets is equal to the maximum size of the data packets currently in the Tetrys encoding window. Packets reach the MN through the different interfaces and are diverted to Tetrys (step 6) which decodes and rebuilds any lost packets. The whole packets received by the Tetrys encoder at step 2 are re-injected (step 7) without losses and ordered in-sequence. Finally at step 8, packets are transmitted to the transport and application layers in a transparent manner.

When our coding scheme is used to improve the link quality (with two interfaces) the source data packets are sent on the fastest (which is also the more lossy one in our experimental scenario) interface while the redundancy (coded) packets are sent through the WWAN.

During upward handover, the PLR is monitored and when it exceeds a given threshold (70% in the experiment), a coded version is sent over the WWAN for each data packet received from the source. When the WLAN is definitely out of range, packets are sent uncoded over the WWAN. During a downward handover, all the source data packets are sent coded over the WWAN and uncoded over the WLAN. When the WLAN PLR decreases below a threshold, the coded packets will be sent according to the redundancy ratio through the WWAN only until the PLR becomes negligible for TCP.

3.2. The Tetrys on-the-fly coding scheme

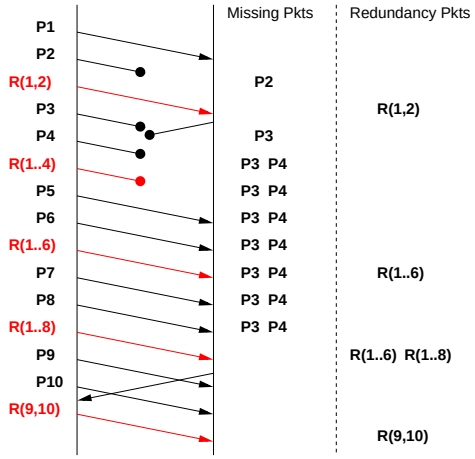


Figure 3. Tetrys principle.

The Tetrys sender uses an *elastic encoding window* (denoted W_{sender}) which includes all the source packets sent and not yet acknowledged. Let P_i be the source packet with sequence number i. Every k source packets, the sender sends a (single) repair packet $R_{(i..j)}$, which is built as a linear combination (with random coefficients) of all the packets currently in W_{sender}. The receiver is expected to periodically acknowledge the received or decoded packets, and each time the sender receives an acknowledgment, the acknowledged packets are removed from W_{sender}. A receiver can decode lost packets as soon as the number of available repair packets is higher or equal to the number of lost packets (the lost packets are detected by the gaps they introduce in the sequence number of the received packets). Fig. 3 illustrates this principle. In the figure $k = 2$, which means that a repair packet is sent each time two source packets have been sent. The right side of this figure shows the list of packets that are lost and not yet rebuilt, as well as the repair packets kept by the receiver in order to recover them. During this data exchange, packet P_2 is lost. However, the repair packet $R_{(1,2)}$ successfully arrives and allows to rebuild P_2. The receiver sends an acknowledgement for packets P_1 and P_2, in order to inform the sender that it can compute the next repair packets from packet P_3. Unfortunately this acknowledgement is lost. However, this loss does not compromise the following transmissions and the sender simply continues to compute repair packets from P_1. After this, we see that P_3, P_4 and $R_{(1..4)}$ packets are also lost. These packets can be rebuilt using $R_{(1..6)}$ and $R_{(1..8)}$ since the number of repair packets becomes higher or equal to the number of lost packets.

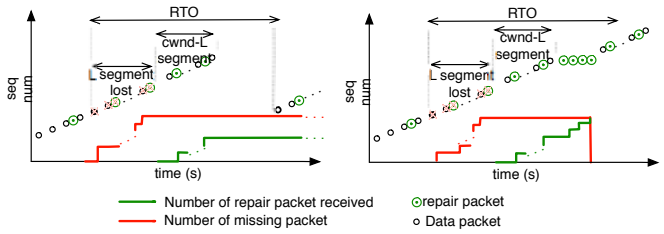


Figure 4. Mechanism to prevent a blocking TCP window.

The acknowledgement path is only used to optimize the encoding process and is not mandatory during a handover of a few seconds. However, these acknowledgements contain information about the PLR experienced by the MN that might be used to update or tune the redundancy ratio when the handover takes a longer time (we consider higher than 30sec). If the role of the coding scheme is to make the transport layer more robust, the redundancy ratio does not require to be frequently adjusted while coding for real-time application would need more accurate adaptation.

Unlike Tetrys, most of the forward error codes (FEC) used over packet erasure channels are block codes [14]. This means that at the encoder side, a set of repair packets (R) is built from a given set of source data (SD) packets and at the decoder side, these repair packets can only be used to recover SD packets from their corresponding set. If too many packets (among the SD and repair packets) are lost during the transmission, the recovery of the missing SD packets is then not possible.

As a result and compared to block codes:

- Tetrys is tolerant to any burst of source, repair or acknowledgement losses, as long as the amount of redundancy exceeds the PLR;
- the lost packets are recovered within a delay that does not depend on the RTT;
- the configuration is much easier and more robust to network variation than configuration for a block code. This is a key point in the context of handover;

These properties make Tetrys a perfect candidate to reduce packet loss and recovery delay during a handover process.

3.3. Redundancy emission/allocation and interaction with TCP

TCP is well-known for bad performance over lossy links as every lost packet is considered by TCP as an indication of congestion. A possible solution to mitigate this effect would be to use a FEC mechanism to correct losses due to link errors. However in [15][1], the authors show that the joint use of end-to-end FEC with TCP does not solve the problem in case of significant PLR. This is due to the fact that TCP needs in-order delivery of data packets and is also strongly sensitive to RTT variations which trigger spurious timeouts resulting in a decreased throughput. A spurious timeout occurs when a non lost packet is retransmitted due to a sudden RTT increase (typically when the mobile node moves from a WLAN to WWAN) which implies an expiration of the retransmission timer set with a previous, and thus outdated, RTT value.

In a previous work [9], we have already shown that this code protects efficiently real-time traffic such as Voice over IP and video-conferencing over links with high PLR. Even if TCP behaves and performs better above Tetrys than above FEC, rebuilding a burst of L lost packets requires receiving at least L redundant packets. As explained in Fig. 4, the sending of 1 repair packet every k source data packets (left subfigure), L data losses would require $k*L$ more data packets to be sent by TCP. As TCP cannot send more than $cwnd - L$ data packets, if $k*L > cwnd - L$, RTOs may be triggered and the connection may stall. As the subfigure on the right suggests, this problem can be solved by sending repair packets at a minimal rate f_r when the TCP window is abnormally stalled (e.g. when the throughput drops below $k*f_r$).

The requirement to correct errors timely while minimising extra transmissions is thus to size the frequency so that $\frac{1}{RTT} > f_r \geqslant \frac{L-R \cdot cwnd}{4 \cdot RTT}$, with f_r the minimal frequency for the emission of redundancy packet (assuming the RTO is roughly four times the RTT).

3.4. A model for TCP/Tetrys configuration

To refine the estimation of f_r, we propose to assess the probability that TCP/Tetrys detects a congestion event (*e.g.* $k*L > cwnd - L$) for a given $cwnd$. According to the distribution of the number of packets still missing when TCP sends its $cwnd$, we size f_r and choose the most efficient combination of k and f_r.

As losses are induced by errors on the channel instead of congestion, we assume the packet losses are independant and identically distributed (i.e. follow a Bernoulli law) with a loss probability p. Under this assumption, we introduce a Markov chain: $\{Y_n, n > 0\}$, which represents the difference between the number of lost packets and the number of received repair packets observed after the reception of each repair packet. As in

[1] In their scheme, TCP is modified to ignore losses. In our case, we assume a complete separation between the coding layer and the transport protocols.

section Sec. 3, we assume the receiver can decode when $Y_j = 0$.

The evaluation of $\{Y_n, n > 0\}$ is performed after each Tetrys block. We define a block as a set of $k+1$ consecutive packets that begins at the first source packet sent after a repair packet and ends at the next repair packet. We point out that our definition of block does not correspond to the usual definition in coding theory which is a set of symbols encoded together. In this context, a repair packet can be encoded from a set of source data packets belonging to several blocks.

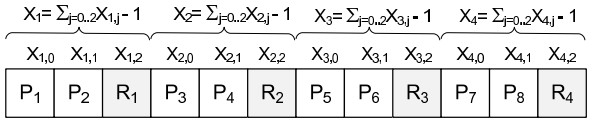


Figure 5. Random variables for Markov chain in the Bernoulli channel

As shown on Fig 5, the reception of each packet is represented by a random variable (r. v.) $X_{i,j}$, where $i > 0$ and $0 \leqslant j \leqslant k$. With this notation, i corresponds to the block and j to the position of the packet in the block.

On the Bernoulli channel, we have $P[X_{i,j} = 1] = p$ (the packet is lost), and $P[X_{i,j} = 0] = 1 - p$ (the packet is received). $X_{i,j}$, where $0 \leqslant j \leqslant k-1$ thus corresponds to source packets and $X_{i,k}$ corresponds to the repair packets. We define the r.v. X_i, where $i > 0$, as follows:

$$X_i = \sum_{j=0}^{k} X_{i,j} - 1 \tag{1}$$

Then, the loss of one of the first k (source) packet increments the value of X_i while the reception of the repair packet decrements the value of X_i. Since X_i is obtained from a sum of Bernoulli variables, we have:

$$P(X_i = u - 1) = \binom{k+1}{u} p^u (1-p)^{k+1-u} \text{ with } u = 0, \ldots, k+1$$

As Y_n does not capture the amount of time step away from the state $Y_n = 0$, we extend it using $Z_n = (Y_n, s_n)$ (see Fig.6), where s_n is the amount of steps spent since the last decoding (*i.e* $Z_n = (0,0)$). $Z_n = (Y_n, s_n)$ is defined as follows:

$$Z_n = (Y_n, s_n) \begin{cases} (Y_{n-1} + X_n, s+1) & \text{if } Y_{n-1} + X_n \geqslant 0 \\ & \text{and } s < \frac{cwnd}{k} : \\ (0,0) & \text{if } Y_{n-1} + X_n < 0 \\ (Y_{n-1}, \frac{cwnd}{k}) & \text{if } s \geqslant \frac{cwnd}{k} \end{cases} \tag{2}$$

Let us denote $a_{(l_i,s_i),(l_j,s_j)} := P(Y_n = j \text{ and } s_n = s_j | Y_{n-1} = l_i \text{ and } s_{n-1} = s_i)$ the transition probabilities between the states (l_i, s_i) and(l_j, s_j). Let us now define A the matrix $(a_{(l_i,s_i),(l_j,s_j)})$ with $l_i, l_j, s_i, s_j \geqslant 0$, $l_i, l_j <= cwnd$ and $s_i, s_j \geqslant \frac{cwnd}{k}$. Given $Z_0 = (1, 0, 0, ..)$ the $1 + \sum_{i=1}^{i=\frac{cwnd}{k}} (i * k)$ dimension probability vector at the initial state, we can obtain $E[inter_{CE}]$ the average number of data packets sent before a congestion event:

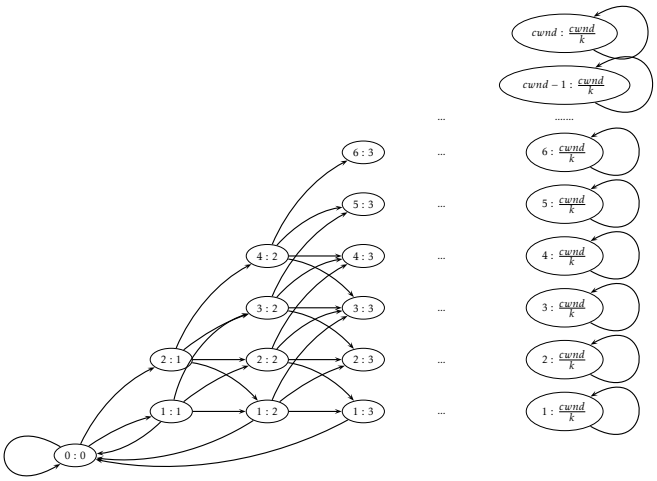


Figure 6. Illustration of the Markov chain $Z_n = (Y_n, s_n)$ with $k = 2$.

$$\sum_{n=1}^{n\to\infty} \sum_{i=1}^{i=cwnd} i \times (P[Z_n = (i, cwnd)] - P[Z_{n-1} = (i, cwnd)]) \tag{3}$$

As the probability $p > 0$, all the states $(i,j)_{j<cwnd}$ are transitives, the probability $P[Missing = i]$ to have i missing packets at the end of the run is :

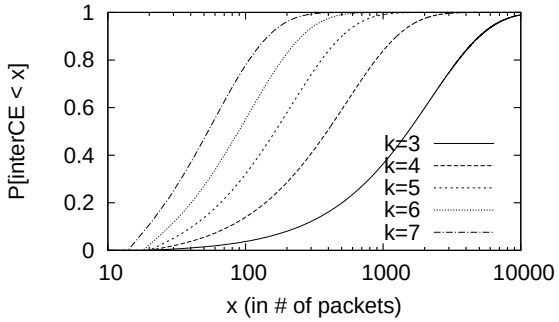
$$P[Missing = i] = \lim_{n\to\infty} P[Z_n = (i, cwnd)] \tag{4}$$

Given $P[Missing = i]$ we can assess the probability to recover a stalled decoding with a f_r (namely : $P_{dec\ with\ f_r}$) as follows:

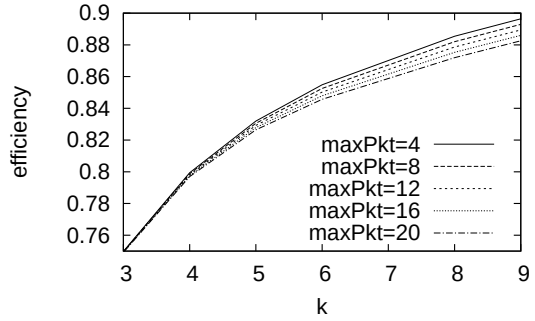
$$P_{dec\ with\ f_r} = \sum_{i=1}^{i=cwnd} P[Missing = i] \times \sum_{u=0}^{u=maxPkt} \binom{k+1}{u} p^u (1-p)^{k+1-u} \tag{5}$$

with $maxPkt$ the maximum amount of packets sent with the rate f_r before an RTO expires *e.g.* $f_r \cdot \frac{3 \cdot RTT}{2}$.

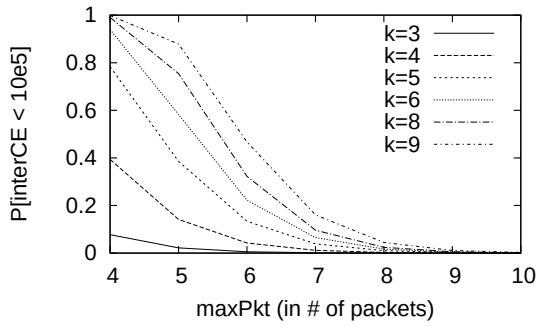
Figures 7(a), 7(b) and 7(c) respectively show $P[interCE < x]$, the efficiency of the overall solution using k and f_r, and the probability for a 10^5 packets TCP transfer to complete without any RTO triggering. It clearly exhibits the gain induced by f_r and shows that it is preferable to keep the redundancy ratio $\frac{1}{k+1}$ close to the loss rate p and to allow a higher value of f_r. As a result, the parameters should be computed as follows taking k as $arg_{max}(k \mid \frac{1}{k+1} > p)$ and the minimum value of f_r such as the results of (5) remains above a threshold *e.g.* 99% of the outage should be decoded.



(a) Probability that the TCP source transmit x packet before it get stalled.



(b) Efficiency *e.g.* $\frac{cumul\ goodput}{cumul\ throughput}$ as a function of the fixed code rate k and the number of packet $maxPkt$ sent through f_r before the RTO expire.



(c) Probability that a TCP transfer of length 10^5 packets completes (*e.g.* **no** RTO expire) as a function of k and $maxPkt$.

Figure 7. Results obtained with the Markov chain model Z_n.

4. Evaluation

The evaluation of our scheme is done using the ns-2 simulator and a real implementation under BSD. The ns-2 simulator allows to set up several flows and allow an easy monitoring of the TCP parameters while the BSD allows a proof of concept implementation over a handover scenario.

4.1. Dynamic of TCP/Tetrys flows

We first evaluate our solution using the ns-2 simulator under the topology described on Fig.8. Figure 9 shows the throughput and the fairness as a function of the link loss rate for TCP/Tetrys. For comparison purpose, fig. 9 also plots the results for regular TCP flows without applying link-layer losses. We can see that as suggested by the results presented in Fig. 7(c) TCP/Tetrys allows a graceful degradation while maintaining a fair sharing of the bandwidth, with a Jain-index close to 1. The fairness remains similar even in case where more flows are involved.

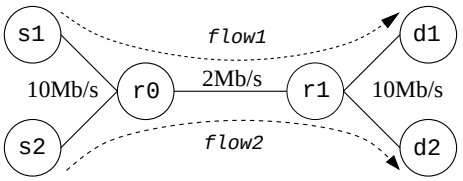


Figure 8. Topology used for the evaluation using *ns-2*.

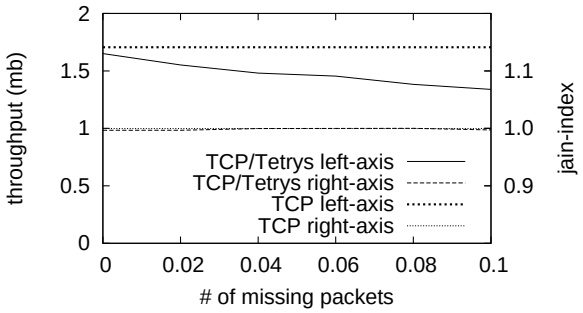
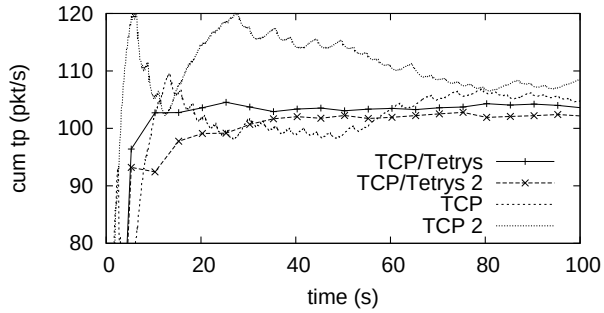
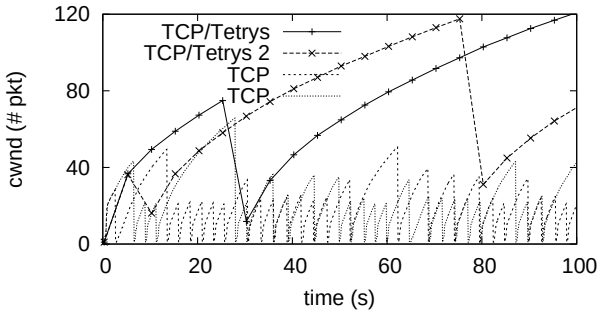


Figure 9. Average throughput and fairness of two TCP/Tetrys (resp. TCP) flows as a function of the packet loss rate.

Figures 10(a) and 10(b) show respectively the dynamic of the congestion window and throughput for the flows $flow1$ and $flow2$ (see the topology in Fig.8) in the case of TCP/Tetrys and TCP. Figure 10(b) corroborates the model of Fig. 7(c) as the occurrence of loss event perceived by TCP is significantly reduced with only two $cwnd$ reduction within the 100 seconds of the simulation. As a result, Fig. 10(a) shows that the throughput is less variable for TCP/Tetrys and with a standard deviation of the instantaneous throughput (not shown in the figure) of 23 packets for TCP/Tetrys and 31 for TCP. Even if the throughput of TCP/Tetrys is more stable, only two congestion event occurs and we might expect a smaller standard variation. The variations are actually due to the fact that when Tetrys can not decode with the code rate, it needs to detect the drop in the TCP's sending rate to start sending the redundancy packets at the rate f_r. During this period, the TCP throughput is frozen thus leading to such variation.



(a) Evolution of the two TCP/Tetrys (resp. TCP) cumulative throughput without link layer losses.



(b) Evolution of the two TCP/Tetrys (resp. TCP) congestion window (*i.e.* cwnd) without link layer losses.

4.2. Comparison with FEC over lossy links

Our testbed architecture is similar to the one presented in Fig. 2, except for the WLAN and WWAN links which are emulated with Netem [16] on top of two Ethernet links. The CN and AR are connected with a 10Mbit/s Ethernet link with a negligible transmission delay. Default settings assume 10Mbit/s and an RTT of 10ms for the WLAN and 4Mbit/s with a RTT of 100ms for WWAN (this is realistic for UMTS HSDPA cellular networks).

As previously stated, Tetrys is not the only code that can be used with TCP. In this section we compare the performance of TCP/Tetrys and TCP/FEC using an ideal block code as FEC. For a given loss rate, the optimal code rate of a TCP/FEC code is not the same as TCP/Tetrys. To allow a fair comparison between those two type of codes, we keep a fixed the code rate of 80% (*i.e.* the value of (n, k) are such that $\frac{k}{n} = 0.8$). As we have seen in sec.3.4, the overhead induced by f_r is negligible and it is not biasing the investigated trade off between efficiency and throughput.

Tab. 1 shows the throughput obtained by TCP on top of Tetrys or FEC when packets are sent over a single lossy link (over WLAN only). We can see that both Tetrys and FEC can handle a low loss rate efficiently maintaining a throughput of 8Mbit/s for the TCP flow. Similarly to previous work on TCP/FEC [15], we observe that with a significant loss rate, the TCP/FEC throughput decreases and the connection often stalls. In contrast to this, TCP over Tetrys is not severely impacted by these loss rates, and in fact the TCP throughput starts to decrease only after $PLR = 14\%$. We have to remark that the code rate is fixed during the experiments. As a matter of fact, there would not a decrease of the TCP throughput by dynamically adjusting the code rate as a function of the PLR.

Tab. 2 shows the throughput obtained by TCP on the top of Tetrys or FEC when data packets are sent over the (10Mbit/s, 10ms) WLAN link lossy link and the coded packets over the (4Mbit/s, 100ms) WWAN link.

We can notice that the results of TCP/FEC are even worse than in the one link only experiment (Tab. 1). We can make the same observation for TCP/Tetrys under small PLR (for 0.5% or 2%). This is explained by the delay asymmetry between the two links and the TCP *cwnd* which fits the bandwidth delay product (BDP) corresponding to the WLAN link. When losses occur, their reconstruction requires to wait for the coded packet that arrives 90ms later. During this time, no packet reaches the TCP receiver and thus there is no acknowledgement sent to the TCP sender that would slide the congestion window. The RTT perceived by TCP increases with the PLR and the *cwnd* also increases until it reaches the BDP corresponding to the slowest link. This explains the poor performance[2] of TCP/Tetrys for small PLR and the improvement observed when the PLR is higher. These two facts: 1) the link delay asymmetry, and 2) the few losses not recovered by FEC, impact the TCP throughput over FEC more significantly.

In spite of our testbed not enabling bandwidth aggregation (as in SCTP), these results show that in contrast to previous coding proposals, Tetrys allows transport protocols such as TCP to remain efficient in spite of the deteriorated link conditions during and around a handover.

4.3. Illustration of the mechanism during a handover scenario

Fig. 10 shows the results for various WWAN bandwidths (300kbit/s, 1Mbit/s and 4Mbit/s) It takes 0.5 second to the WLAN link to change from "up" state to "down" state and the same for the opposite transition. We can see that even if these various parameters have an impact on the TCP throughput, they do not significantly impact the amount of time required by TCP to reach its average throughput with Tetrys. Similar results hold for different values of the WWAN RTT.

Fig. 11 shows the impact of the redundancy ratio $(R = 1/5, R = 1/6, R = 1/8, R = 1/9)$ on TCP during a

[2]This can be solved by artificially delaying the packet at the speed of the slowest interface.

PLR	0.0	0.5	2	4	5
TCP/Tetrys (4,5)	7.78/0.01	7.81/0.01	7.81/0.01	7.80/0.01	7.81/0.00
TCP/FEC (4,5)	7.79/0.02	7.83/0.00	6.48/1.2	3.01/0.6	2.68/2.24
TCP/FEC (8,10)	7.81/0.01	7.78/0.05	7.81/0.03	6.3/1.48	Timeout
TCP/FEC (12,15)	7.82/0.02	7.82/0.01	7.79/0.03	7.54/0.10	4.06/5.05
TCP/FEC (16,20)	7.81/0.01	7.81/0.02	7.82/0.01	7.82/0.01	Timeout
PLR	8	10	12	16	20
TCP/Tetrys (4,5)	7.82/0.01	7.81/0.01	7.81/0.01	7.18/0.02	4.61/0.26
TCP/FEC (4,5)	Timeout				
TCP/FEC (8,10)	Timeout				
TCP/FEC (12,15)	Timeout				
TCP/FEC (16,20)	Timeout				

Table 1. Throughput/Std. dev. in Mb/s for $R = 0.2$, $B_D = 10Mbps$, $B_R = 4.0Mbps$ with data and repair packets sent on WLAN only.

PLR	0.0	0.5	2	4	5
TCP/Tetrys (4,5)	9.54/0.00	7.69/0.18	6.5/0.49	8.18/0.34	9.02/0.05
TCP/FEC (4,5)	9.53/0.00	5.96/0.02	3.07/0.08	1.13/0.6	1.25/0.14
TCP/FEC (8,10)	9.54/0.00	7.19/0.08	4.72/0.32	2.71/0.46	1.85/0.40
TCP/FEC (12,15)	9.55/0.00	7.7/0.11	4.79/1.36	3.73/1.13	Timeout
TCP/FEC (16,20)	9.53/0.00	7.37/0.65	5.83/0.68	3.73/1.13	0.84/1.18
PLR	8	10	12	16	20
TCP/Tetrys (4,5)	8.55/0.7	8.66/0.3	8.45/0.57	7.10/0.31	5.12/0.2
TCP/FEC (4,5)	0.17/0.05	Timeout			
TCP/FEC (8,10)	Timeout				
TCP/FEC (12,15)	Timeout				
TCP/FEC (16,20)	Timeout				

Table 2. Throughput/Std. dev. in Mb/s for $R = 0.2$, $B_D = 10Mbps$, $B_R = 4.0Mbps$ with data sent on WLAN and repair sent on WWAN only.

handover. In this case, it takes 10 seconds for the WLAN PLR to switch from 0 to 100% and inversely. The three sub-figures show different runs of the experiment. We can see that the configuration of the Tetrys redundancy ratio does not require to be timely adjusted as there is no impact on the throughput achieved by TCP.

Compared to block codes (characterized by a specific FEC coding configuration) where we would have to dynamically reconfigure the redundancy parameters ((k, n)) as a function of the size of the loss burst, Tetrys is resistant to any kind of loss burst patterns and does not need to be dynamically adjusted (as already highlighted Section 3.2 illustrated and Tab. 1, 2). Furthermore, although the increase of the redundancy parameters allows to correct larger burst of losses, they trigger TCP timeout as the decoding process can be longer than the RTO value.

This last result illustrates that compared to block codes (such as a specific FEC coding configuration), Tetrys is resistant to any kind of loss burst patterns.

5. Conclusion

In this paper, we evaluate the benefits of using an "on the fly" coding scheme to reduce packet losses during a soft vertical handover due to low signal quality. The experimental evaluation suggests that the use of this type of coding scheme may be an interesting complementary strategy to vertical handover management protocols due to its fast configurability and in the context of multipath communications. Our experiments clearly show that this coding scheme allows to maintain the TCP throughput during a handover by taking advantage of the multiple wireless interfaces present in today's smartphones. Particularly, results show that it significantly improves the quality of TCP flows in terms of delivery ratio. As a next step, we are planning to integrate the implementation of this error recovery algorithm called Tetrys as a part of the SafetyNet architecture and evaluate the performance empirically using our SafetyNet implementation.

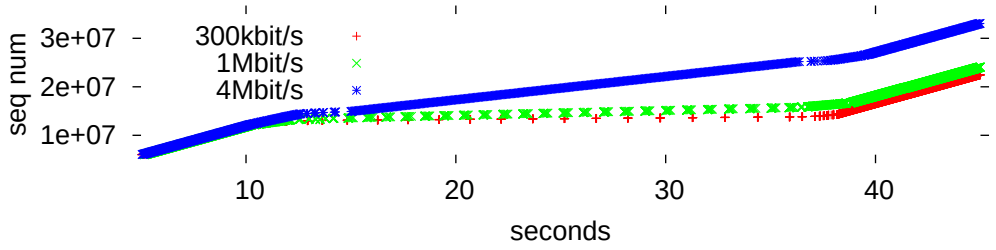


Figure 10. Handover scenario with various WWAN bandwidth and (10Mb/s, 10ms) WLAN.

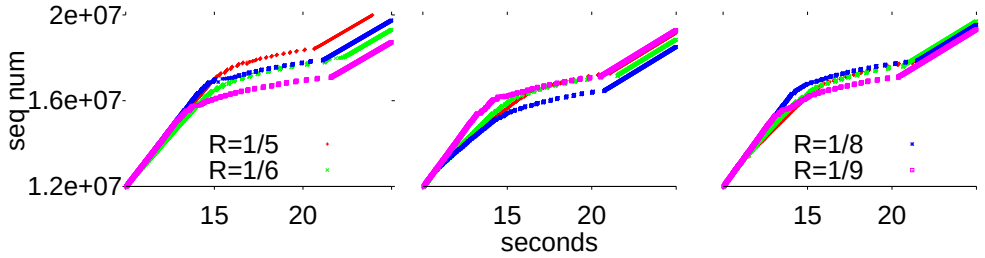


Figure 11. Handover scenario with various redundancy ratio with (10Mb/s, 10ms) WLAN and (4Mb/s, 100ms) WWAN.

References

[1] Chakravorty, R., Vidales, P., Subramanian, K., Pratt, I. and Crowcroft, J. (2004) Performance issues with vertical handovers - experiences from GPRS cellular and WLAN hot-spots integration. In *IEEE PERCOM*.

[2] Petander, H., Perera, E. and Seneviratne, A. (2007) Multicasting with selective delivery: a safetynet for vertical handoffs. *Wirel. Pers. Commun.* **43**(3): 945–958.

[3] Petander, H. and Lochin, E. (2010) Safetynet version 2, a packet error recovery architecture for vertical handoffs. In *ICST MONAMI* (Santander, Spain).

[4] Kim, S.E. and Copeland, J.A. (2003) TCP for seamless vertical handoff in hybrid mobile data networks. In *IEEE Globecom*.

[5] Daniel, L. and Kojo, M. (2009) The performance of multiple TCP flows with vertical handoff. In *ACM MobiWAC*.

[6] Stewart, R. *et al.* (2007), Stream Control Transmission Protocol, RFC 4960.

[7] Fiore, M. and Casetti, C. (2005) An adaptive transport protocol for balanced multihoming of real-time traffic. In *IEEE GLOBECOM*.

[8] Wang, J., J. and Zhu, X. (2008) Latent handover: A flow-oriented progressive handover mechanism. *Computer Communications* **31**(10): 2319–2340.

[9] Tournoux, P.U., Bouabdallah, A., Lacan, J. and Lochin, E. (2009) On-the-fly coding for real-time applications. In *ACM Multimedia*.

[10] Nagami, K. *et al.* (2007), Multi-homing for small scale fixed network using mobile IP and NEMO, RFC 4908.

[11] Ng, C. *et al.* (2007), Analysis of multihoming in network mobility support, RFC 4980.

[12] Matsuoka, H., Yoshimura, T. and Ohya, T. (2003) A robust method for soft IP handover. *IEEE Internet Computing* **7**: 18–24.

[13] et Al., H.L. (2007) A staggered FEC system for seamless handoff in WLANs: Implementation experience and experimental study. In *IEEE ISM*.

[14] Lin, S. and Costello, D. (1983) *Error Control Coding: Fundamentals and Applications* (Prentice-Hall, Englewood Cliffs, NJ).

[15] Anker, T., Cohen, R. and Dolev, D. (2004), Transport layer end-to-end error correcting. Tech. Rep. School of Engineering and Computer Science, Hebrew University.

[16] Hemminger, S. (2005) Network emulation with netem. In *Australia's national Linux conference (LCA)* (Canberra, Australia).