Robust Streaming in Delay Tolerant Networks

Pierre Ugo Tournoux[1,2], Emmanuel Lochin[1,2], Jérémie Leguay[3] and Jérôme Lacan[2]
[1] CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
[3] Thales Communications

***Abstract*—Delay Tolerant Networks (DTN) do not provide any end to end connectivity guarantee. Thus, transporting data over such networks is a tough challenge as most of Internet applications assume a form of persistent end to end connection. While research in DTN has mainly addressed the problem of routing in various mobility contexts with the aim to improve bundle delay delivery and data delivery ratio, little attention has been paid to applications. This paper investigates the support of streaming-like applications over DTN. We identify how DTN characteristics impact on the overall performances of these applications and present Tetrys, a transport layer mechanism, which enables robust streaming over DTN. Tetrys is based on an on the fly coding mechanism able to ensure full reliability without retransmission and fast in-order bundle delivery in comparison to classical erasure coding schemes. We evaluate our Tetrys prototype on real DTN connectivity traces captured from the Rollerblading tour in Paris. Simulations show that on average, Tetrys clearly outperforms all other reliability schemes in terms of bundles delivery service.**

I. Introduction

This paper addresses the problem of transporting data streams in a reliable and delay-efficient manner over a Delay Tolerant Networks (DTN) [1]. We introduce Tetrys, a transport level mechanism based on an on-the-fly coding scheme, easy to configure and which provides both full reliability and fast in-order delivery services.

DTN are composed by mobile wireless nodes able to exchange data when they are within the same transmission range. As a DTN topology is intermittently and partially connected, mobiles nodes observe frequent connectivity disruptions. Due to these disruptions, common ad-hoc routing and transport approaches cannot be used and as a result, new solutions must be proposed. Therefore, two classes of transport protocols are commonly envisioned in DTNs. The first ones do not use feedback messages. They intend to improve the bundle delivery ratio and/or delivery delay with mechanisms such as replication, erasure codes and multipath as in [2], [3], [4]. The second ones consider that feedback can be available. Harras et al. for instance investigate various acknowledgement strategies in [5]. In the context of deep-space networking (DSN), transport protocols such as LTP proposed by Farrell et al. [6] for large delay links with connectivity disruptions, use ARQ and Unequal Error Protection to reduce the amount of non-mandatory retransmissions.

Compared to Tetrys, these algorithms and protocols[1] are designed to carry bundles unit with a reliable or unreliable service. Furthermore, *the whole piece of data to transfer must be available at the beginning of the transmission*. This simplifies and allows, for instance, the generation of erasure code blocks which can be spread or routed amongst the next set of contact opportunities as in [2], [3]. However and as shown in [2], these algorithms require a challenging configuration closely linked to the network characteristics without providing any guarantee in terms of delay and reliability. At last and in this context, if we seek to maintain a loss rate below the maximum threshold tolerated by the application, the complexity of this configuration increases significantly nay, might become impossible.

On the opposite, we address in this paper the so called *streaming-like*[2] applications which produce *continuous data* over time and require *to be consumed in-sequence* (data must be ordered) at the receiver side. In this context, this paper proposes Tetrys as a possible solution to support such traffic in DTNs. Tetrys consists in a robust streaming transport mechanism achieving low latency and the reliability thresholds required by applications. Tetrys overruns streaming support challenges in such networks without delay compromise while remaining independent of any mobility or routing schemes.

In the next Section, we firstly present which makes the problem challenging over DTN and then detail our proposal Section III. We evaluate in Section IV the performance obtained by Tetrys compared to other schemes. Finally, we discuss in Section V the complexity, engineering issues and possible improvements.

II. Challenges for streaming support in DTNs

In a DTN, nodes mobility, routing strategies and links and buffers capacities impact on the bundles delivery performances in terms of delay, in-order delivery and losses. As end-to-end paths used for bundle delivery depends on the mobility pattern of each peer, each bundle sent between a given source and destination do not necessarily use the same path. As a result, despite of a relatively stable average transmission delay, the standard deviation of each bundle's delay of a given flow can be very large. In our experiments, this leads to a reordering ratio (following [7]) often higher than 50% with some peaks to 90% while in classical networks, this value is usually below 5%. The network structure in terms of connexity, contacts density and nodes betweenness also evolves as a function of

[1]Except LTP/LTP-T which has been designed for a DSN context.

[2]We point out that our definition is not associated to the common one, often related to real-time and multimedia applications. Those are clearly out of scope of the present study.

time. In this case, the average delay might also drastically vary.

Each bundle remains in the network until its Time To Live reaches zero, this implies that the evolution of the network can lead to high loss rate due to TTL expiration. Furthermore, the amount of data that can be transmitted during a given contact is limited. In case of congestion, this can slow down the dissemination of the bundles in the network and lead to TTL expiration. Congestion can also lead to buffer overflow with a high probability to drop all the copies of a given bundle and as a resulst, increase the loss rate.

All these intrinsic DTN characteristics limit the use of applications that need either in-order delivery or full reliability services. Indeed, when network re-ordering ratio is very low or when jitter is quite stable, ARQ mechanisms might be a possible solution. In the context of Deep Space Networks (DSN), this is currently the choice enabled by LTP-T protocol [6]. In this case, the number of false loss detections is small and it is possible to wait a fixed and reasonable amount of time to take a retransmission decision. However, in the case of DTN, both reordering ratio and jitter prevent the use of ARQ mechanisms that would lead to spurious retransmissions. Concerning in-order delivery, the receiver needs to set a timer allowing to wait for missing bundles before transmission to the application. This implies both higher delay and higher loss rate as bundles which arrive after this waiting period might be considered as lost. The design of a mechanism that would determine if a packet is lost or disordered within a reasonable delay bound is thus difficult and possible solutions, if any, would be context dependent (from both mobility and application point of view).

Our proposal prevents the use of ARQ mechanisms while enabling a full reliability without data retransmissions in certain condition. Lost or delayed bundles are recovered in-order, allowing to accelerate data transmission to the upper layer and to solve the technical problem which arises when we have to decide if a packet is lost or delayed.

III. Robust DTN streaming with Tetrys

This section briefly presents how Tetrys [8] can provide a robust streaming protocol when applied to DTNs.

The original goal of Tetrys was to propose an implicit acknowledgment strategy for long-delay networks [9] where recovery of missing packets is thus not possible with classical retransmissions mechanisms (i.e. ARQ). Tetrys is based on an elastic encoding window updated dynamically according to the feedback information returned by the receiver. However one important property of Tetrys highlighted in the present contribution is its capability to reduce recovery delay of lost data at the receiver side. In its most general form, this approach is compatible with any kind of traffic and full reliability can be achieved as soon as the encoding ratio is higher than the average loss rate.

The main principle of this erasure coding scheme is to generate *repair bundles* every k source bundles, where k is an integer determined according to the expected bundle loss rate. The resulting coding rate used is then equal to $1/(k+1)$. These repair bundles are a random linear combination of the bundles included in the variable size encoding window which contains the bundles sent but not yet acknowledged by the receiver. They are computed as follows: $R_{(i..j)} = \sum_{u=i}^{j} \alpha_u^{(i,j)}.B_u$ with B_i to B_j the bundles that belong to the encoding window and $\alpha_u^{(i,j)}$, the coefficient randomly chosen in the finite field $\mathbb{F}_q$ used to encode the u^{th} bundle in the repair bundle $R_{(i..j)}$. Only the seed of the random coefficient generator which is specific to each repair bundle has to be embedded in the latters. The source bundles are sent unmodified (i.e. the code is systematic: meaning that input data are embedded in the encoded output) which leads to reduce coding overhead.

For each repair bundles, the receiver subtracts all the available source bundles encoded. The resulting repair bundles, then, consist in linear combinations of the lost source bundles. The rest of the decoding simply solves this linear system. One should note that decoding can only be done as soon as the number of received repair bundles (the equations) is at least equal to the number of lost source bundles (the unknowns).

Source bundles are removed from the encoding window (at the sender side) with the reception of *acknowledgment bundles* which assess their proper reception or decoding at the receiver side. Acknowledgements include SACK vectors which follows the TCP RFC SACK specifications. The purpose of this acknowledgment scheme is to reduce the number of source bundles involved in the encoding/decoding processes. Their frequency, availability or losses do not impact the performance in terms of reliability or decoding delay. They can be carried by one of the various transmission schemes investigated by Harras et al. [5]. In particular, the authors propose the use of active receipt which are disseminated as new messages; passive receipt which evolves through the nodes previously infected by the bundle to acknowledge; finally network-bridged receipt that transits through a cellular network. As the acknowledgement delay varies with these different methods, their use depends on the affordable encoding/decoding complexity for the nodes.

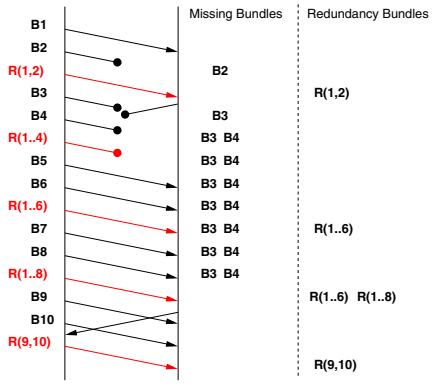

Fig. 1. A simple data exchange

Fig. 1 illustrates the overall mechanism with a simple bundle exchange. For the sake of simplicity, small delay and jitter

are considered. After a correct reception of bundle B_1, B_2 are lost. However, using the received repair bundle $R_{(1,2)}$, the receiver rebuilds B_2. We assume that the receiver sends back an acknowledgment to inform the sender that bundles older than B_3 should be removed from the encoding window. Then, this acknowledgment is lost. However this loss does not compromise the following transmissions and the sender simply continues to compute repair bundles from B_1. After this, we see that B_3, B_4 and $R_{(1..4)}$ bundles are lost. None of these bundles need to be retransmitted as they are rebuilt thanks to the bundles received from B_5 to $R_{(1..8)}$. Indeed, the receiver can rebuild B_3, B_4 by firstly "subtracting" all the received source bundles from the repair bundles in order to obtain $(R'_{(1..6)}, R'_{(1..8)})$ which are linear combinations of B_3 and B_4. By solving this linear system, B_3 and B_4 can be recovered. After receiving the acknowledgment, the encoding in the repair bundle $R_{(9,10)}$ starts from 9 and not from 1. The receiver was also able to acknowledge B_5 and B_6 with a SACK block reducing more significantly the size of the window. If no feedback is received, the sender simply sends $R_{(1..10)}$). The loss of an acknowledgment impacts "only" on the coding window size increase and the encoding relative[3] complexity.

In summary, this scheme allows a full-reliability even if some source data bundles, repair bundles or acknowledgments are lost. More interestingly in the context of DTN, the decoding does not depend on the feedbacks received and thus the loss recovery delay is utterly independent from the RTT. As the decoding process is realized in order, we do not need to decide whether a given bundle is lost or not before we deliver the others to the application. The problem of retransmission with ARQ is also avoided since Tetrys does not need to request for retransmission.

IV. Evaluation

In this section we evaluate the performance of Tetrys in terms of delivery ratio and application delay against classical reliability schemes.

A. Relevant schemes

We compared Tetrys to the following schemes:

a) Uncoded: In this scheme, messages are sent *as is* and are carried out by the routing scheme used.

b) Erasure coding scheme (FEC(n, k)): We consider here a classical block erasure code (also called Forward Error Correction) as the one described in [10]. The parameters n and k impact on the coding as follows: after the emission of k source bundles B_1,..,B_k, FEC(n, k) adds $(n - k)$ repair bundles (F_1..F_{n-k}) sent interleaved with the source bundles of the next block. This results in the following emitted sequence: B_k, F_1, B_{k+1}, F_2.. B_n,F_{n-k},B_{n+1},B_{n+2},(...). We assume a Maximum Distance Separable (MDS) code, this means that the decoding of the lost bundles from a given block is possible as soon as k bundles (B or F) are received. Note that concerning the parameter of correction capability, MDS codes are optimal. They are thus better than Fountain codes studied in [3] which require $(1 + \epsilon) \cdot k$ bundles (with $\epsilon > 0$) on average. Indeed, the main advantage of Fountain codes are the large number of potential redundant bundles and the decoding complexity. These two parameters are not critical in our context. For both FEC(n, k) and Tetrys, we denote R: the redundancy ratio such as $R = \frac{k}{n-k}$.

c) *Automatic Repeat ReQuest (ARQ)?*: As FEC may not provide any full reliability guarantee, it would have been interesting to consider an ARQ-based protocol. Following section II, it is clear that this class of mechanism is not applicable. Even if the technical problems related to loss detection are solved, the recovery delay of the lost bundles would be increased by at least one RTT. The same remark also applies to Hybrid-ARQ where the basic principle is to code data with FEC and retransmit lost packets (ARQ) to rebuild the blocks that got more than $n-k$ losses. Due to space limitation and as we expect that the results would not be pertinent, we do not present experiments for ARQ and HARQ-based reliability schemes.

In practical case, either FEC or Uncoded strategies implement a timer to determine whether bundles are lost or delayed. This obviously introduces a delay increase. In our evaluation, FEC does not get such delay overhead as we use an Oracle to determine if a bundle is lost. However, Tetrys does not benefit from this Oracle as it provides a full reliable service. We use both in-order bundle delivery delay and bundle loss rate metrics (i.e. the ratio between the number of bundles received and number of bundles sent) observed by the application.

B. Network settings

To evaluate these mechanisms, we are replaying real connectivity data from the RollerNet dataset [11]. This dataset plots a high intensity of interactions between mobile nodes where the number of connectivity disruptions prevent the use of classical MANET protocols. We consider data streams which generate $1/2$ bundles per second between both extremities of the tour (*i.e.* specific head and tail nodes are deployed). We use Spray and Wait (SW) routing [12] with 16 copies to disseminate bundles across the 62 iMotes deployed. For the feedback of Tetrys, we use active receipt [5] which are sent using epidemic routing. The packet's lifetime is not limited and bundles are produced during 5000 sec. Bundles can be received until the end of the experiment set at time 7000 sec.

The links capacities are limited according to the Bluetooth bitrate ($\approx$ 700 KB/s) and each buffer is fixed to 300 MB. In order to generate a background traffic, 50 peers are randomly selected to generate $1/2$ bundle per second. All bundles (sources or repairs) have a fixed size of 200 KB. We have estimated a reordering ratio of 95% with an average extent of 348 bundles and a delivery ratio of 82% (see [7] section 4.2).

C. Results

Figure 2 plots the distribution of the in-order delivery delay of Tetrys, FEC$(600, 300)$, FEC$(1000, 500)$ and the Uncoded

[3]Note that only binary XOR operations are done.

	Uncoded	Tetrys	FEC $k = 10$	FEC $k = 50$	FEC $k = 100$	FEC $k = 200$	FEC $k = 300$	FEC $k = 500$
$R = 1/2$	2172 - 0.82	1251 - 1.00	1870 - 0.89	1486 - 0.92	1430 - 0.92	1600 - 0.96	2035 - 1.00	1718 - 1.00
$R = 1/3$	2172 - 0.82	2145 - 1.00	1829 - 0.87	1700 - 0.88	1564 - 0.89	1586 - 0.88	1689 - 0.92	1779 - 0.90
$R = 1/4$	2172 - 0.82	2246 - 1.00	1912 - 0.85	1660 - 0.85	1775 - 0.85	1573 - 0.87	1662 - 0.88	2024 - 0.85

TABLE I
AVERAGE IN-ORDER DELIVERY DELAY (SECONDS) - DELIVERY RATIO AS A FUNCTION OF THE REDUNDANCY RATIO (WE RECALL THAT FECS DELAY BENEFIT FROM AN ORACLE DETAILED IN SEC.IV-A).

strategy. We can see that under the same conditions, the probability for a bundle to be delivered to the application before a given delay is significantly higher with Tetrys than with FEC. As an example, a bundle is delivered before 1000 sec with a probability of 0.48 for Tetrys while we obtain only 0.16 for FEC$(1000, 500)$. It is interesting to observe that in addition to recover from the losses, the coding schemes allow to reduce the delay resulting from network reordering. In particular, Tetrys achieves a delay nearly half the Uncoded scheme.

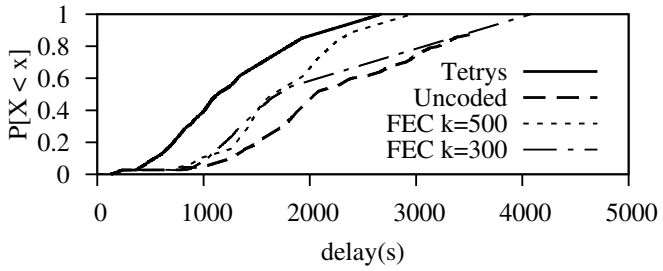

Fig. 2. Distribution of the in-order delivery delay with $R = 0.5$

The first line of table I shows that for a redundancy ratio of 0.5, Tetrys has an average delivery delay of 1251 sec which is significantly smaller than the FEC's minimum average delay of 1430 sec achieved with $k = 100$. We can also notice that FEC reaches full reliability when $k \geqslant 300$. However, compared to Tetrys, FEC based schemes did not achieve the optimum delay and delivery ratio for the same values. This leads to an intricated trade-off between delay and reliability. On the contrary, Tetrys allows the smallest delivery delay on average without reliability compromise.

Table I also shows that if the redundancy ratio is reduced to $R = 1/3$ or $R = 1/4$, FEC does not rebuild all the lost bundles, even with large blocks. In this case, we cannot accurately compare both in-order delivery delay of Tetrys and FEC as the latter does not rebuild all the bundles and takes advantage of the Oracle. This issue is highlighted when considering only 85% of the Tetrys bundles as the delay decreases to 1786 sec. This corresponds to the same couple (delay, reliability) achieved by FEC with k =$10, 50, 100, 500$.

Although we focused on the in-order delivery, we must notice that Tetrys remains a good candidate when we relax the in-order delivery constraint (e.g. without the de-jitter part of the delay). With $R = 1/2$, the average delay is 1125 sec for Tetrys, 1295 sec for FEC with $k = 300$ and 1243 sec for FEC with $k = 500$. Eventually, we can notice that as long as FEC rebuilds all lost bundles, Tetrys delay will be lower or equal to FEC delay.

V. DISCUSSION

A. Choice of parameters

Table. II shows the variation of the average loss recovery delay as a function of the loss rate[4] and redundancy ratio. For each recovered bundles, it corresponds to the delay between the detection that a bundle is lost using the sequence numbers and its reconstruction. The table contains $\emptyset$ in the case where Tetrys did not decode all the lost bundles when the loss rate becomes higher than the redundancy ratio. As shown in this table, a redundancy slightly higher than the current loss rate estimated is mandatory to perform with Tetrys.

Concerning the decoding delay, it can be observed for a given loss rate, the more the redundancy ratio, the less the decoding delay. As an example, for a bundle loss rate of 18%, Tetrys achieves an average decoding delay of 252 sec for a redundancy ratio of 0.5 which increases to, respectively, 928 sec and 1395 sec for redundancy ratio of 0.33 and 0.25. Obviously, for a fixed redundancy ratio, the decoding delay increases with the loss rate as observed for $R = 1/2$.

According to the results presented, we observe that the estimation of the average loss rate is necessary to assess the performance of the system. As it depends of the routing and the mobility context, this information can be provided by the lower layer thanks to a cross-layer metric. These information can also be provided by the receivers' feedbacks in order to dynamically refine the redundancy ratio.

In a future work, we will focus on the analytical derivation of the expectation of the decoding delay according to synthetic network metrics such as average loss rate and reordering.

B. Complexity

We discuss here the complexity of Tetrys. *a*) Computation complexity: The per bundle computation complexity at sender and receiver sides is mainly[5] $O(\frac{RTT \cdot \lambda}{k})$ with (n, k) the coding parameter, λ average sending rate of the application in bundles per second, and RTT the average delay between the emission of a bundle and the reception of the acknowledgement feedback. For comparison purpose, the per bundle complexity of the FEC mechanism considered in

[4]We changed the size of the forwarding node's buffer (from 140MB to 310MB) to obtain different loss rate.

[5]See our technical report [8] for details on the mechanism complexity.

Bundle loss rate	$R = 1/2$	$R = 1/3$	$R = 1/4$
15.0 %	217 sec	792 sec	1209 sec
18.0 %	252 sec	928 sec	1395 sec
19.3 %	446 sec	1064 sec	$\emptyset$
22.2 %	419 sec	1117 sec	$\emptyset$
24.6 %	625 sec	1376 sec	$\emptyset$
35.8 %	854 sec	$\emptyset$	$\emptyset$
40.0 %	937 sec	$\emptyset$	$\emptyset$
47.5 %	1380 sec	$\emptyset$	$\emptyset$

TABLE II
AVERAGE LOSS RECOVERY DELAY AS A FUNCTION OF THE BUNDLE LOSS RATE AND THE REDUNDANCY RATIO

the previous section is $O(k)$. If we look at the example Fig. 2, the average size of the Tetrys encoding window is 521.6 bundles. Following [13] and the encoding window size, we would have been able to achieve a maximum bit rate of $210KB/s$ on a PIII 933 Mhz, twice the bit rate assumed in our settings. Depending on the trade-off between the complexity and the cost of the feedback mechanism, we can adapt the latter (active/passive/bridged) [5] since the smaller the feedback delay, the smaller the complexity.

b) Buffer complexity: the complexity of Tetrys in space is also mainly $O(RTT * \lambda)$ as the source bundles remain in the source node buffer until acknowledged. At the receiver side, the source bundles remain in the buffer as long as they are encoded in the repair bundles received. FEC needs a fixed number of k bundles both at the sender and receiver sides.

In the studied scenarii, both in time and space, the complexity of Tetrys is in the same order of magnitude than other schemes.

VI. CONCLUSION AND FUTURE WORK

In this paper we have studied the possible use of streaming-like applications over a DTN. We have detailed the challenges to solve and proposed the use of Tetrys as a potential solution. Tetrys achieves full-reliability when the redundancy ratio is greater than the average loss rate. Furthermore, the loss recovery delay is independent of the RTT and tolerates bursty losses either on data or feedback paths. These characteristics allow Tetrys to fit both network and application requirements. Indeed, Tetrys accepts any reliability thresholds required by an application while recovering the bundles ordered. The latter prevents the use of a complex (and possibly inaccurate) algorithm to determine whether a bundle is lost or delayed[6]. Compared to other mechanisms that aim to improve the delivery ratio, Tetrys does not depend of any mobility context or routing mechanism and the sole parameter required for its configuration is the average loss rate.

We evaluated both Tetrys and a classical erasure coding scheme in the context of the Paris Roller Tour dataset. The results demonstrate the perfect ability of Tetrys to rebuild all streamed bundles and to achieve the best ratio between delay and reliability compared to various FEC schemes. It also appears in our evaluation that the mechanism requirement is in the same order of magnitude than FEC of large block sizes for both buffer size and computation complexity.

Although this paper introduces the use of Tetrys for unicast robust streaming in DTN, we have to notice that it would also work without modification in case of multicast robust streaming over DTN which is still a challenging problem, even in classical wired networks. In addition to multicast, our future work will focus on the best computation method to determine a minimal redundancy ratio that fulfills application requirements in terms of loss recovery delay according to synthetic network metrics such as the average loss rate and the reordering ratio. Finally, as Tetrys linear combination mechanism is close to Network Coding principle (as for per flow RLC[4]), both schemes could also be combined, in order to improve the resulting performances in terms of delivery ratio and decoding delay. A deeper analysis of this interaction will also be tackled in our future work.

[6]Indeed, implementing such timer is complex and might involve spurious detection.

REFERENCES

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. ACM SIGCOMM*, 2003.

[2] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall, "Using redundancy to cope with failures in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 109–120, 2005.

[3] E. Altman and F. De Pellegrini, "Forward correction and fountain codes in delay tolerant networks," in *INFOCOM 2009, IEEE*, April 2009, pp. 1899–1907.

[4] Xiaolan Zhang, G. Neglia, J. Kurose, and D. Towsley, "On the benefits of random linear coding for unicast applications in disruption tolerant networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, 2006 4th International Symposium on*, April 2006.

[5] Khaled A. Harras and Kevin C. Almeroth, "Transport layer issues in delay tolerant mobile networks," in *IN IFIP NETWORKING*, 2006.

[6] Stephen Farrell and Vinny Cahill, "Evaluating LTP-T: A DTN-Friendly transport protocol," in *IWSSC*, Salzburg, Austria, Sept. 2007.

[7] A. Morton, L. Ciavattone, G. Ramachandran, S. Shalunov, and J. Perser, "Packet Reordering Metrics," RFC 4737 (Proposed Standard), 2006.

[8] Jérôme Lacan, Emmanuel Lochin, Pierre-Ugo Tournoux, Amine Bouabdallah, and Vincent Roca, "On-the-fly coding for time-constrained applications," *CoRR*, vol. abs/0904.4202, 2009.

[9] J.Lacan and E. Lochin, "Rethinking reliability for long-delay networks," in *IEEE International Workshop on Satellite and Space Communications, 2008. IWSSC 2008.*, Oct. 2008, pp. 90–94.

[10] Luigi Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication Review*, Apr. 1997.

[11] P.-U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M.D. de Amorim, and J. Whitbeck, "The accordion phenomenon: Analysis, characterization, and impact on dtn routing," in *INFOCOM 2009, IEEE*, April 2009, pp. 1116–1124.

[12] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. WDTN*, 2005.

[13] Laurent Dairaine, Laurent Lancérica, Jérôme Lacan, and Jérôme Fimes, "Content-access qos in peer-to-peer networks using a fast mds erasure code," *Computer Communications*, vol. vol. 28, no. 15, pp. 1778–1790, September 2005.