Reliable Data Streaming over Delay Tolerant Networks

Sotirios-Angelos Lenas [1], Scott C. Burleigh [2], and Vassilis Tsaoussidis[1]

[1] Space Internetworking Center (SPICE),
Democritus University of Thrace, Xanthi, Greece
{slenas,vtsaousi}@ee.duth.gr
[2] NASA / Jet Propulsion Laboratory (JPL),
California Institute of Technology, Pasadena, California, USA
scott.c.burleigh@jpl.nasa.gov

Abstract. Data streaming over Delay-Tolerant Networks (DTN) is a challenging task considering jointly the specific characterisics of DTN environments, the demanding nature of streaming applications and their wide applicability. Presently, there are not any advanced mechanisms available to support this functionality and typical configurations fail to efficiently transfer data streams. In this paper, we present our ongoing work in data streaming over DTNs and propose the Bundle Streaming Service (BSS) as a framework to improve the reception and storage of data streams. Our proposed framework exploits the characteristics of Delay Tolerant Networks to allow for reliable delay-tolerant streaming. Here, we present a simple usage scenario along with the proposed framework and evaluate it experimentally at a preliminary stage which, however, suffices to demonstrate its potential suitability for both terrestrial and Space environemnts.

Keywords: Data streaming, Delay Tolerant Networks, Interplanetary Internet.

1 Introduction

After several years of systematic research in various aspects of Delay/Disruptive Tolerant Networking (DTN) such as routing, transport protocols and convergence layers, DTN technology has reached a higher level of maturity. The development of a reliable set of working solutions and associated standards under the auspices of the Consultative Committee for Space Data Systems (CCSDS) and the Internet Research Task Force's (IRTF's) DTN research group [1] has boosted the applicability of DTN architectures, which now present themselves as prominent solutions for global internetworking. Based on that progress, several studies [2, 3, 4] promote the benefits of DTN architectures [5] and highly suggest their use in disruptive environments through the Bundle protocol [6], which encodes most functionalities that an overlay network requires.

Our work here deals with a relevant topic that has not yet seen much progress, despite its potential applicability: data streaming over DTNs. Data (and especially live) streaming in delay/disruptive tolerant environments becomes a particularly challenging task since the presence of high delays, frequent disruptions and variable

Y. Koucheryavy et al. (Eds.): WWIC 2012, LNCS 7277, pp. 358–365, 2012.

bandwidth acts inevitably against the basic application principles of data streaming that call for mechanisms that guarantee smooth viewing experience of end-users.

In this paper, we present the results of our ongoing effort to provide a framework that enables the efficient management of real-time traffic in delay/disruptive tolerant environments in a manner that avoids the overexploitation of available network resources. In this context, we propose the Bundle Streaming Service (BSS) as a practical approach that addresses most of the networking challenges related to streaming over DTNs. BSS is a framework that enables "streaming" data to be conveyed via DTN "bundles" in a manner that supports in-order stream processing with minimal latency while still ensuring reliable delivery of all data to enable ad-hoc "playback" review of recently received information. Potential examples of real-time applications that could exploit the capabilities provided by this framework are one-way voice, video or continuous telemetry streaming.

BSS was designed with Interplanetary Internet (IPN) [7], and its associated issues, in mind. Despite its initial target though, our proposal could also fit in terrestrial delay/disruptive tolerant environments in which network nodes either follow a fixed predetermined route or move freely within a dynamic topology and hence experience occasional disruptions each time they move beyond the communication range; such networks exhibit properties similar to those of space internetworks, in terms of bandwidth capacity and connection availability, and hence may be serviced by a common solution framework.

The rest of the paper is organized as follows. In section 2, we present the related work and highlight the contribution of our proposal. In section 3, we analyze the operation of BSS and present its capabilities. In section 4, we describe a usage scenario that demonstrates how BSS can be exploited in a real world environment. In section 5, we detail the implementation of BSS and present some preliminary experimental results, both for terrestrial and Space environments. Finally, in section 6 we conclude this paper and discuss the framework for future work.

2 Related Work

Mobile Ad-hoc Networks (MANETs) are closely related to DTNs since they share several common characteristics such as network disruptions, high error rates and variable capacity links. A substantial amount of prior works that address several data streaming issues have already been proposed for MANETs. In general, the majority of the efforts are moving in two main directions; i) efficiency improvement and ii) redundancy. Among the most popular approaches suggested so far for improving efficiency are: i) the dynamic optimization of data coding, throughout the streaming session, so that the encoding bitrate does not surpass the available bandwidth of the network [8], ii) routing through multiple paths in order to increase delivery probability [9], iii) packet prioritization to minimize queuing delay and iv) specially adapted transport layer mechanisms that aim in reducing recovery delay of lost data. Redundancy on the other hand, is achieved through the use of FEC codes or by applying content summarization and error spreading techniques in order to provide error resilience. A few cross-layer approaches have also been proposed that combine several of the aforementioned techniques in order to enhance viewers' experience [10, 11].

Yet, to the best of our knowledge, the issue of data streaming in the context of DTN hasn't been studied extensively. Few initial works adopt similar approaches with the ones used in MANETs. In [12], T. Liu and S. Nelakuditi use erasure coding techniques in order to construct a disruption-tolerant video sequence so that in the event of disruption, helpful video content is still provided to clients by injecting additional "summary frames" to the original stream, while in [13], P. U. Tournoux et al. introduce Tetrys, a transport level mechanism based on an on-the-fly coding scheme which provides full reliability under the assumption that the encoding ratio used by Tetrys is higher than the average loss rate.

Due to the fact that the characteristics of each type of DTN may vary and the objective each time may be different, most of the aforementioned approaches cannot be applied in the context of delay/disruptive tolerant networking. In most cases, network functions such as routing, error recovery and congestion are usually located in the source or destination, a fact that mandates end-to-end connectivity. It's clear that any end-to-end approach cannot be considered due to the disruptive nature of DTNs. Therefore, each DTN node should be fully aware of how to handle a bundle that carries frames of a stream without the need for the application to run on every node. The use of FEC codes in the bundle layer also presents drawbacks since it might create conflicts with lower network layers, e.g. in cases that FEC codes are also used in the MAC layer. These conflicts usually lead to increased demand for network resources, mainly bandwidth, without guaranteeing any reliable delivery of the frames in cases where the coding rate is not sufficient to replace the losses imposed by the error rate of the communication channel. Finally, reliability should also be taken into consideration, especially for critical applications, which handle time-sensitive data.

A key observation that also inspired this work is that none of the aforementioned approaches exploit the most appropriate property of the Bundle protocol which naturally allows every DTN-node to temporarily store bundles. In that case, and based on the fact that disruptions are usually localized and experienced only by a few among many receivers, the retransmission effort could be minimal since it is limited to a certain area of the network. Considering this fact, the key concept behind BSS is to employ in the forwarding process of each DTN node both a best effort along with a reliable transfer protocol, in order to achieve minimal latency but also ensure reliable delivery of the whole stream. An additional advantage of our approach is that it does not confine future deployments of other sophisticated mechanisms on top of BSS, but instead, it grafts flexibility that further enhances synergistic application mechanisms.

Finally, as a design choice, we confine all network functionality of BSS within the bundle layer while preserving the closely related to the application functionalities, such as the re-ordering of packets, at the application level, in order to ease the burden of application developers.

3 Bundle Streaming Service Analysis

BSS consists of two basic components: a forwarder daemon and a library for building streaming-oriented applications. Figure 1, depicts the architecture of BSS.

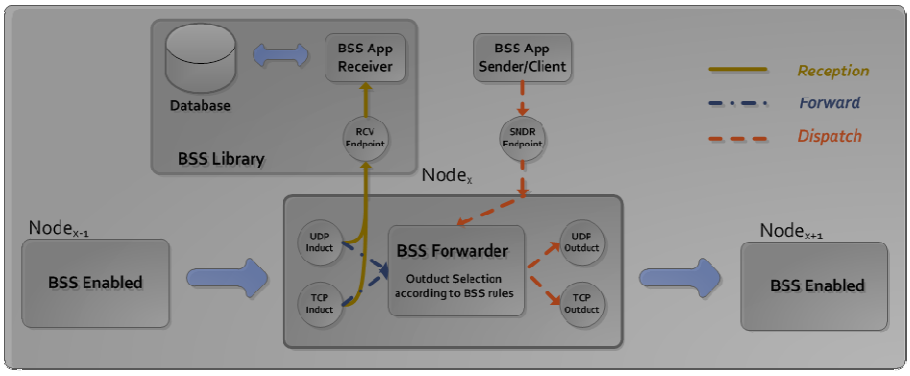

Fig. 1. BSS Architecture

Bundle creation time is the decisive criterion for all forwarding actions. BSS forwarder keeps track of the creation times of the bundles flowing from node X to node Y. Each of the forwarder's neighbors must have two inducts, one for a "best-efforts" convergence-layer protocol such as UDP (or "green" LTP) and one for a reliable convergence-layer protocol such as TCP (or "red" LTP). BSS selects the appropriate *outduct* for forwarding the bundle pending dispatch by applying the following rule: "Each bundle whose creation time is greater than that of any other bundle seen on this stream so far is forwarded to the "best-effort" *outduct*".

A prerequisite of BSS is that every bundle sent by a BSS-enabled node has to be custodially acknowledged. If a bundle's custody-accepted signal does not arrive prior to the timeout, the bundle is re-forwarded; in that case its creation time is not greater than that of any other bundle seen on this stream so far, so it is forwarded to the reliable induct of the next neighbor. Note that non-BSS forwarders will forward streaming traffic, but streaming display performance will be somewhat degraded by every node on the end-to-end path that is not running the BSS forwarder. Whenever the BSS forwarder does not identify the flow of bundles as BSS traffic, it treats them as normal traffic and forwards them accordingly as typical bundles.

The design of BSS ensures that eventually all bundles in the stream are delivered to their final destination, and beyond that, they are delivered in order and synchronized. The flow of streaming data never waits for retransmissions to succeed to avoid degrading the viewing experience of participants in a data streaming session. In the event that a bundle sent over the non-reliable convergence-layer protocol does not arrive at its next-hop node, that bundle simply will not be included in the flow of streaming data displayed at the destination. From this point on, it is identified as out-of-stream because its creation time is out of order. It eventually ends up at the destination, but because it's out-of-stream it does not get included in the displayed flow of streaming data; it just goes into the database at the destination, along with all of the successfully streamed data that has already been displayed. This enables the user to employ the "replay" features of the streaming service library to rewind back through the database and replay the data that were missed – merged with the data that originally were successfully received in transmission sequence; at the same time the current stream continues to be processed, e.g., displayed in another window.

The receiver's application is built using the BSS library, which initiates a background thread that receives all the bundles. Whenever that thread receives a bundle, it inserts the bundle into the BSS database (in creation-time order, for replay on demand) and it also checks bundle's creation time in order to decide, based on the above-described rule, if it will pass the bundle to an application-provided callback function for real-time display or to other stream processing. Meanwhile, the main thread of the application can be responding to user commands by calling BSS library functions that retrieve data from the time-ordered database for replay, with support for running *forward* or *backward*, *fast-forward*, *freeze*, etc.

The result of the above-described process is unimpeded real-time streaming of all the data that don't get dropped, together with comprehensive replay and review of all the data in the stream. And, because BSS does not operate by modifying bundle priorities, it can handle multiple concurrent streams of bundles at different priorities over the same links with notable flexibility: the high-priority streams will be closest to real-time, while the low-priority data will be available a little later.

4 Usage Scenario

In a real-world scenario BSS could be used for streaming video from the Moon to Earth. For simplicity, let's suppose the network topology is comprised of two nodes, the stream's source node (the camera) and the destination node (the user).

The user is sitting at a terminal on which three windows are presented. Window A is the real-time view from the camera. Window B is a GUI comprising VCR-like control widgets for replaying the video stream. Window C is the replay video view, controlled from window B.

Window A shows the view from the camera on the Moon exactly as it was 1.28 seconds ago (plus a few milliseconds of queuing, transmission, and processing latency), except that the view may "freeze" once in a while because one or more video frames were lost or corrupted somewhere along the end-to-end DTN path from the camera to the display. When there is such an outage, the missing frames never show up in this window; the displayed image simply remains unchanged until the next frame received in real time arrives. The view in this window is never delayed by any more than the one-way light time (plus processing, etc. latency), and it never regresses.

The user controls the replay display from window B, commanding the replay view to start N seconds ago and then roll forward or perform other available playback features such as pause, rewind, roll backward, etc.

Window C shows the replay view. This may be no more than the frames that originally were displayed in window A just a few seconds or minutes ago. But in the event that there were some outages in the real-time view in window A, the replay may show more than what was originally displayed. That's because the replay view includes frames that arrived out of ascending bundle creation time order at the user's terminal, due to retransmission of lost/corrupt frames or to arrival on different length paths. So, the replay view will always be at least as complete as the real-time view and it may be more complete; moreover, replaying a second time a bit later one may even reveal a more complete view as late-arriving lost bundles, which perhaps were lost again, finally arrive.

5 Implementation Details and Preliminary Experimental Results

BSS forwarder was implemented as a modified forwarder daemon, adapted from the standard "ipn" forwarder daemon included in ION [14]. Following implementation, our first goal was to establish the baseline performance characteristics of BSS over a simple streaming session under various network conditions, including variable propagation delays (PD) and high packet error rates (PER). Up till now, we have conducted some initial experiments, using a simple single-hop scenario (Fig. 2). The time needed for the complete reception of a stream consisting of 5000 frames was evaluated by using two basic sets of transport protocols in combination with BSS in order to evaluate its performance under terrestrial and Space environments.

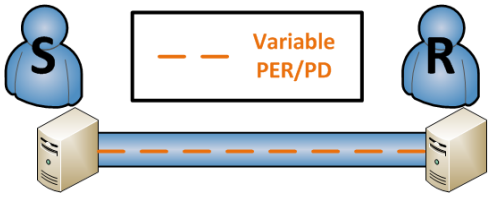

Fig. 2. String topology

The network stack was emulated on the DTN testbed of SPICE center [15] while the streaming process was simulated by developing *bssStreamingApp* and *bssRecv* applications. *bssStreamingApp* simulates the functionality of a media device that produces 30fps, 20866 bytes each. It also wraps each of these frames in bundles and hands them to the bundle layer for transmission. The transmission ratio achieved by *bssStreamingApp* is about 5000kbps which is considered more than enough to simulate the transmission of a high definition H.264 video quality stream [16]. At the other end, *bssRecv* is developed based on the API functions provided by BSS library. It presents two basic functionalities; firstly it immediately displays any in-order frames arriving at the destination; secondly, it saves in a specially-designed database the received stream, both in-order and out-of-order frames, in the appropriate order. The results of our initial experiments are shown in Figure 3.

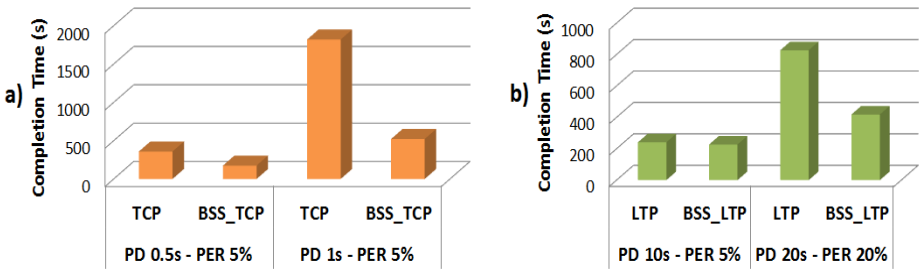

Fig. 3. Preliminary experimental results

Due to the poor performance that TCP exhibits in high error rate and long propagation delay environments, BSS manages to reduce the total requested time of receiving 5000 frames by almost 80% in the worst case. In Space environments, where LTP "red" transmission is used in place of TCP, BSS achieves better results only in cases where the error rate of the channel is above 10%. Furthermore, based on a different set of experiments that due to lack of space we cannot present here, we note another interesting property of BSS: it manages to reduce the total number of out-of-order received packets in comparison with the normal ION configuration using LTP alone.

6 Conclusion

In this initial phase of our study, a preliminary performance evaluation was conducted under various network conditions. The results obtained so far show that the suggested framework has the potential to improve stream reception in both terrestrial and Space environments.

As future work, we plan to extend this preliminary evaluation by conducting more tests and employing several other metrics, such as out-of-order delivered packet ratio, packet loss ratio, frame loss ratio and peak signal-to-noise ratio, in order to accurately assess the efficacy of BSS and evaluate the impact of frame size, hop count and mobility on its performance.

Armed with the knowledge acquired by these evaluations, we plan to proceed by modifying *bssStreamingApp* and *bssRecv* applications to support real video traffic, probably by importing/exporting video frames from/to VLC and deploying BSS in embedded devices in order to confirm our emulation results through tests in real wireless environments.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013_FP7-REGPOT-2010-1, SP4 Capacities, Coordination and Support Actions) under grant agreement n° 264226 (project title: Space Internetworking Center-SPICE). This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein.

References

1. Internet Research Task Force Delay Tolerant Networking Research Group, `http://www.dtnrg.org/`, `http://www.irtf.org/charters/dtnrg.html`
2. Farrell, S., Cahill, V., Geraghty, D., Humphreys, I., McDonald, P.: When TCP Breaks: Delay and Disruption Tolerant Networking. IEEE Internet Computing, 72–78 (2006)
3. Fall, K.: A delay-tolerant network architecture for challenged internets. In: Proceedings of ACM SIGCOMM (2003)
4. Burleigh, S., Hooke, A., Torgerson, L., Fall, K., Cerf, V., Durst, B., Scott, K., Weiss, H.: Delay-tolerant networking: An approach to interplanetary Internet. IEEE Communications Magazine 41(6), 128–136 (2003)

5. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay–Tolerant Networking Architecture. Internet RFC 4838 (April 2007)
6. Scott, K., Burleigh, S.: Bundle Protocol Specification. RFC 5050 (November 2007)
7. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Travis, E., Weiss, H.: Interplanetary Internet (IPN): Architectural Definition, http://www.ipnsig.org/reports/memo-ipnrg-arch-00.pdf
8. Qin, M., Zimmermann, R.: Improving mobile ad-hoc streaming performance through adaptive layer selection with scalable video coding. In: Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA 2007, New York, NY, USA, pp. 717–726 (2007)
9. Calafate, C., Malumbres, M., Manzoni, P.: Mitigating the impact of mobility on H.264 real-time video streams using multiple paths. Journal of Communications and Networks 6(4), 387–396 (2004)
10. Zhao, Z., Long, S., Shu, Y.: Rate Adaptive Live Video Streaming in Manets. In: Communications and Networking in China, ChinaCom 2006, pp. 1–5, 25–27 (2006)
11. Delgado, G., Frias, V., Igartua, M.: Video-streaming transmission with qos over cross-layered ad hoc networks. In: Proceedings of SoftCOM 2006, Dubrovnik, Croatian, pp. 102–106 (2006)
12. Liu, T.,Nelakuditi, S.: Disruption-tolerant content-aware video streaming. In: Proceedings of ACM, Multimedia (2004)
13. Tournoux, P., Lochin, E., Leguay, J., Lacan, J.: Robust streaming in delay tolerant networks. In: Proc. IEEE ICC, Cape Town, South Africa, pp. 1–5 (2010)
14. Burleigh, S.: Interplanetary Overlay Network: Design and Operation V1.13. JPL D-48259. Jet Propulsion Laboratory, California Institute of Technology (2011)
15. Samaras, C., Komnios, I., Diamantopoulos, S., Koutsogiannis, E., Tsaoussidis, V., Papastergiou, G., Peccia, N.: Extending Internet into Space – ESA DTN Testbed Implementation and Evaluation. In: Granelli, F., Skianis, C., Chatzimisios, P., Xiao, Y., Redana, S. (eds.) MOBILIGHT 2009. LNICST, vol. 13, pp. 397–404. Springer, Heidelberg (2009)
16. High-definition video, Wikipedia (2012), http://en.wikipedia.org/wiki/High-definition_video