Live-Streaming in Delay Tolerant Networks

Johannes Morgenroth, Tobias Pögel and Lars Wolf
Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig
Braunschweig, Germany
[morgenroth, poegel, wolf]@ibr.cs.tu-bs.de

ABSTRACT

Streaming of audiovisual media contents such as video or music using mobile devices and wireless networks leads to new challenges because these network connections are typically not very reliable. In this demo, we show streaming of live video using a Delay Tolerant Network (DTN) approach. This allows the integration of multiple connections which leads to a higher availability and enables a video transmission without loss in the data stream.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Store and forward networks*

General Terms

Experimentation, Reliability

Keywords

Streaming, DTN, Delay Tolerant Communication, IBR-DTN

1. INTRODUCTION

Audiovisual media are very popular and drivers for many innovations in the Internet. Several of such systems are based on a download model, but there exist also a huge number of various (live) video and music streaming offerings with a high variety of topics such as sports, news, trailers and radio. All these can be used on different mobile devices, terminals or the car entertainment system. This also provides the chance for new functionalities like a personalized radio streaming on a mobile device. Especially in wireless networks like IEEE 802.11x or cellular networks, the mobility of the users or vehicles leads to communication interruptions and, thus, streaming aborts due to insufficient connectivity. For such environments, DTNs are particularly well suited because they operate by using a store-and-forward approach and need no direct end-to-end connection in the communication path. In case of a disconnection, the data-bundles are stored and forwarded as soon as a new link is available again. Furthermore, multiple DTN-nodes can be involved to forward the video stream which can improve the stability further. In case that the network connection is interrupted, the video will be suspended. Later it can be continued at the last break point without any missing parts once any link comes up again.

In our demo we show a scenario where a video is captured by a camera and streamed over multiple nodes in a DTN. To increases the reliability in this setup, two network links are provided. At the destination node, the stream is received and displayed with a delay which depends on the local video buffer size. Only if all the links are unavailable, the video is interrupted for the time without network connections. The video stream is resumed without any gaps as soon as a connection is available again.

2. RELATED WORK

The DTN architecture was first presented in [2] and can be used in many different scenarios with communication disruptions and high delays. A particularly suitable example for the usage of DTNs is the vehicular communication realm. [3] describes a V2V and V2I communication by using the DTN approach and realizes a multi-hop system to increase the coverage. In [1], the usage of DTNs in the public transport systems area is discussed; there, a communication between the vehicles of the operator and multimedia panels mounted on the stops is enabled. The set-up can also be used for the transmission of multimedia content, e.g. advertisements or movie trailers, to the vehicles and stops.

Guo et al. present in [4] another architecture which allows a video live-streaming between vehicles or between vehicles and a stationary video server over some infrastructure components. They also use a DTN-like store-and-forward mechanism. The focus of their work is on the distribution and transmission of video streams. In our demo, we take the advantages of DTNs and can also resume video streams after a long disruption.

3. DEMONSTRATION

In the demonstration set-up we present an uncommon use-case of a DTN. The delayed and disrupted forwarding of bundles in a permanent partitioned network is the worst enemy of any continuous data streaming. However, we want to demonstrate that streaming a large amount of video data can work in a DTN, too.

3.1 Showcase

The set-up consists of four nodes. One is the sender of video data and encapsulates grabbed video data into streaming chunks using the application described in section 3.3. As shown in figure 1, the sender is connected with two other DTN nodes and these are connected with the receiving node.

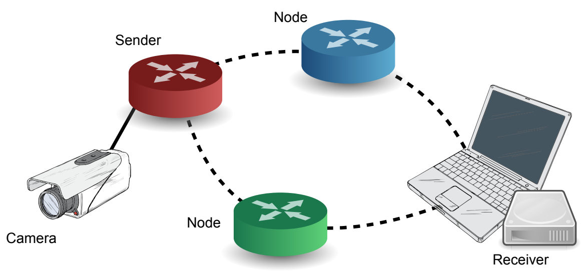

Figure 1: Set-up of our live-streaming demo

By using the epidemic routing protocol, the data chunks are forwarded from the sender to both neighboring nodes and in the next iteration to the receiver. Thus, we have a redundant path between sender and receiver. If one path fails, the streaming data can still be delivered to the receiver using the other path. If both paths are interrupted, the streaming will stuck until the connection is available again.

Additionally, the in-network storage which is characteristic for a DTN system comes into play. This allows to buffer the streaming data as near as possible to the receiver. The nodes within the network are equipped with a mass-storage device and store the streaming data until the connection to the receiver is available.

3.2 Components

The base component for this demonstration is our IBR-DTN [5] system. This software provides a daemon and tools to set-up a DTN on embedded hardware or standard PCs. Besides the standard bundle protocol [6], other features like the TCP convergence layer, IP neighbor discovery and epidemic routing support are important for this demo to work.

As shown in figure 1, this demonstration consists of four DTN nodes plus one camera. The camera is a network camera (LevelOne WCS-0030) and provides a constant real-time transport protocol (RTP) video stream with high quality video to the sender. The sender is a standard PC running Linux and re-encapsulate the video stream into bundles using gstreamer and the streaming application introduced in section 3.3. The nodes between sender and receiver are embedded devices (Ubiquiti RouterStation Pro) with 128 MB RAM, 680 MHz CPU clock, 16 MB Flash Memory and a 2 GB SD card which is attached to store bundles temporarily. The device runs a standard OpenWRT Linux and the IBR-DTN daemon. The receiver is a standard PC running Linux and displays the received video using the VLC media player.

3.3 Streaming Application

To stream data from one bundle endpoint to another, we have to gather the data for a short time, encapsulate it into bundles and forward them to the receiver. To compensate for the possible re-ordering of bundles, we need a mechanism at the receiver to arrange the bundles into the correct order. Furthermore, the receiver needs to decide which is the next bundle to playout or whether it has to wait for data.

The application implemented for this demo uses the API of IBR-DTN and is now part of the IBR-DTN tools. The program reads data from the standard input pipe and puts a predefined amount of bytes into a single bundle. Each bundle [6] consists of the standard primary header and a payload block with a data chunk of the stream. Additionally, a stream-block is attached to the bundle and carries the linear raising sequence number of the data chunk. The structure of the stream-block is depicted in figure 3.3. For this experiment, it is marked with the experimental block type number 242 and should be replicated in every fragment to allow partial content delivery, thus the corresponding bit is always set.

```
+------------+-----------------+-----------------+
| Block type | Proc. Flags (*) | Block length(*) |
+------------+-----------------+-----------------+
|            Stream Sequence Number (*)          |
+------------------------------------------------+
```

Figure 2: Block format of the stream-block. All fields marked with (*) are encoded as SDNV.

The receiver re-orders the received bundles using the sequence number of the stream-block and writes the payload data as continuous stream to the standard output pipe. As configurable option, a time-out could be specified to skip lost bundles, if they do not arrive within a defined amount of time. Since, there is no retransmission of lost bundles, this is the only way to deal with this issue.

4. CONCLUSIONS

The demonstration illustrates that video streaming works well in DTNs. Exploiting the DTN mechanisms allows streaming without any losses and increases the reliability. In the future, we will adapt this approach for dynamic stream subscription and extend it to broadcast the video stream to multiple receivers.

5. REFERENCES

[1] M. Doering, T. Pögel, and L. Wolf. Delay tolerant communication in intelligent transportation systems. In *8th International Workshop on Intelligent Transportation (WIT)*, Hamburg, Germany, 3 2011.

[2] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, USA, 2003.

[3] L. Franck and F. Gil-Castineira. Using delay tolerant networks for car2car communications. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 2573 –2578, june 2007.

[4] M. Guo, M. Ammar, and E. Zegura. V3: A vehicle-to-vehicle live video streaming architecture. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 171 – 180, march 2005.

[5] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf. IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation. *Electronic Communications of the EASST*, 37:1–11, Jan 2011.

[6] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.