Streaming H.264 Scalable Video over Data Distribution Service in a Wireless Environment

Andrea Detti, Pierpaolo Loreti, Nicola Blefari-Melazzi
Electronic Engineering Dept.
University of Rome ”Tor Vergata”
{andrea.detti,pierpaolo.loreti,nicola.blefari}@uniroma2.it

Francesco Fedi
R&D Department
Space Software Italia
francesco.fedi@ssi.it

***Abstract*—The Data Distribution Service (DDS) middleware is enjoying a rapid adoption in high-performance, mission-critical networks. At the same time, the H.264 Scalable Video Coding (SVC) has been recently standardized and it is deemed to be an effective solution for video streaming over a channel with time-varying bandwidth, like the wireless one. In these conditions, it is critical to adapt the video bit-rate to the actual wireless capacity, and bit-rate adaptation is extremely simple for a H.264 SVC video. In this paper we devise, evaluate and demonstrate a technique for streaming H.264 SVC video over a DDS middleware. The contribution is threefold: i) we design a structure of the DDS data-unit able to carry H.264 SVC video-units; ii) we devise a receiver-driven rate-control mechanism based on our DDS data-unit and exploiting specific DDS functionality; iii) we implement and show the effectiveness of our mechanism in an 802.11 wireless scenario, comparing our proposal with other solutions.**

I. Introduction

Nowadays there is an increasing demand for publish-subscribe, data-centric functionality. A user exploiting a data-centric interface is directly connected with the desired data; data discovery functions are executed below the programming interface, thus simplifying application development. The publish-subscribe service model allows an application (subscriber) to ask for data, without taking the trouble to check if the data source (publisher) is available or not: if the publisher is available at the subscription time, the data is promptly transferred; otherwise the actual transfer of data is deferred until the publisher becomes available. The publish-subscribe model fits naturally well in self-configuring, wireless networks scenarios (e.g., mobile ad-hoc, mesh, delay/disruption tolerant, sensor networks), where data sources and sinks can be temporarily disconnected.

The Data Distribution Service (DDS) is a middleware designed by the Object Management Group (OMG) for the development of data-centric publish-subscribe applications upon a legacy IP network [1]. The DDS is enjoying a rapid adoption in high-performance, mission-critical networks, such as military combat systems and air traffic management; indeed, it is a mandated standard for publish-subscribe messaging by the U.S. Department of Defense (DoD).

In this paper we devise, evaluate and demonstrate a technique for streaming H.264 SVC video over DDS. To the best of our knowledge there is not any other published paper dealing with this specific topic.

The first issue that we face is the specification of a tunnelling technique to transport H.264 data-units (so called NALUs) within DDS data-units (so called data-samples). We observe that DDS allows the user to freely define the structure of data-samples, thus a critical aspect of the tunnelling technique is the definition of the data-sample structure.

The second and most challenging issue that we cope with is the design of a receiver-driven mechanism to control the bit-rate of the video stream. The run-time control of the video bit-rate is a fundamental functionality in communication channels characterized by a transfer capacity that may significantly vary over time, as the wireless ones. Indeed, when the transfer capacity gets lower than the current video bit-rate, it is critical to promptly reduce the video bit-rate, otherwise the decoder experiences ”random” losses of NALUs and the quality of video experience abruptly decreases. The rate-control mechanism is based on the structure of our proposed data-sample and on built-in DDS functionality.

As regards the encoding technique, we choose the H.264 Scalable Video Coding (H.264 SVC [2]) since a SVC source can quickly and flexibly reduce (i.e. scale) the video bit-rate simply by not transmitting a subset of video packets [3][4]. The bit-rate adaptation of SVC has a low complexity, which is very useful in publish-subscribe wireless systems where more subscribers, with different available capacity, are interested in the same video. As a matter of fact, the publisher can simply filter out different subsets of video packets requested by different subscribers. Without a scalable encoding, the publisher would have to perform multiple run-time encoding (or transcoding), to produce video streams with different bit-rates; parallel encodings would be not possible for the typical low-power devices of a self-configuring wireless network. As regards the wireless network we assume that it operates in best effort mode.

II. DDS overview

The DDS is a global data-space whose data-structures and properties are defined by meta-information named *Topic*. Each Topic identifies a set of related data-samples with the same data-structure and data-property. For instance, a Topic named ”Temperature” can be used to store samples of temperature monitored by a distributed set of sensors.

The entities that write (read) data-samples in the data-space are the *publishers* (*subscribers*). A publisher disposes of a set of *Data Writers* modules, each of which is used to write information on a specific Topic. A subscriber may read data-samples of Topics by using its Data Readers modules.

A Topic is characterized by a wide set of Quality of Service parameters that control some aspects of the distribution of related data-samples: for instance the QoS named "LIFESPAN" defines the maximum time a data-sample can remain in the system since its writing time; the HISTORY QoS defines the maximum number of data-samples that can be stored in the data-space, if such maximum number is reached then the latest data-sample replaces the oldest one.

When an application wants to receive data-samples of a specific Topic, it simply feeds the DDS interface with the name of the Topic; then, the DDS takes care of configuring the underlay networking facility. When the network configuration phase ends, the application receives a bulk of data-samples whose number is equal to the HISTORY QoS, then it will continue to receive novel data-samples, as they are written. It is worth mentioning that the application can define a filtering-condition related to the "content" of data-samples (e.g., temperature value less than 20 degree). In this case, the DDS transfers only data-samples complying with the filtering condition.

III. H.264 OVERVIEW

An H.264 video is formed by a sequence of sub-streams, each of which improves the video in terms of resolution or frame-rate or quality with respect to the set of previous sub-streams in the sequence; the sequence starts with a basic sub-stream that is always present. A sub-stream is a sequence of NALUs that contain both the encoding bits and an header that identifies the sub-stream by means of three parameters: dependency_id (did), temporal_id (tid), quality_id (qid). For instance, the sub-stream $did = x$, $tid = y$, $qid = z$ improves the video performance by adding information regarding resolution x, frame-rate y and quality z. The basic stream has $did = 0$, $tid = 0$, $qid = 0$. Performance and bit-rate scaling (reduction) is achieved by filtering out the NALUs of a specific sub-stream. This operation is as simple as a packet-header inspection.

In H.264 a video can be Coarse-Grain scalable or Medium-Grain scalable, or both [2]. Here we consider only the case of Medium Grain Scalability (MGS), i.e. a video with a single resolution (i.e., $did = 0$), a fixed number of frame rates (e.g., $tid = 0, 1, ..4$) and an arbitrary number of quality layers (e.g., $qid = 0, 1, 2, \cdots$). The MGS is also called *progressive refinement*; the additional quality layers improve the video performance by reducing more and more the coding quantization error. There is also a coding dependency among NALUs belonging to a so-called Group of Picture (GOP). In case of MGS, if we define the sub-stream-id as $ssid = 8 * qid + tid$, then a sufficient condition to decode the NALUs with $ssid = x$ is the availability of all the NALUs with $ssid < x$.

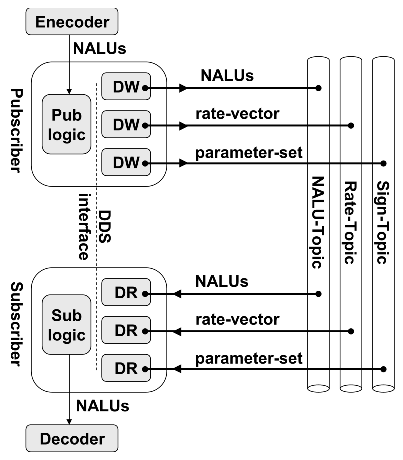

Fig. 1. Streaming architecture

IV. H.264 OVER DDS

A. Overall Architecture

Fig. 1 shows the architecture that we propose to stream H.264 videos over DDS. We have three DDS Topics: a Signalling-Topic, a Rate-Topic and a NALU-Topic. The Signalling-Topic is used to deliver the set of H.264 control NALUs, named "parameter-set", that are necessary to configure a H.264 decoder. This topic enables subscribers to join the video stream at an arbitrary time. The Rate-Topic periodically announces the *rate-vector* R_v, whose i-th element $R_v[i]$ is the average bit-rate of the i-th sub-stream, measured during the last announcement period. This topic is used for rate-control purposes, as explained below. The NALU-Topic is used to deliver NALUs containing video frames. The structure of the data-sample of the NALU-Topic contains: a H.264 NALU, the $ssid$ and a *marker-bit*. Both $ssid$ and marker-bit are used for rate-control purpose. The video-publisher is the sender of the video: it executes the software logic interacting with the DDS facility and hosts the Data Writers. The video-publisher is fed by H.264 NALUs coming from the encoder and, by parsing entering NALUs, builds data-samples and send them to Data Writers (DWs). The video-subscriber is the module used to receive the video, it executes the software logic interacting with the DDS facility and hosts the Data Readers (DRs). When the video-subscriber is started, it gathers the parameter-set from the Signaling-Topic and configures the H.264 decoder. Then the video-subscriber passes to the decoder the NALUs belonging to the NALU-Topic. Moreover, periodically, the video-subscriber performs the rate-control function described in the next subsection.

B. Rate-control mechanism

The rate-control mechanism maximizes the quality of the received video, operating as follows: i) it estimates the available transfer capacity C (as discussed later on); ii) it computes the highest sub-stream k that the subscriber can receive; i.e. $\max_k \sum_{i=1}^{k} R[i] < C$; iii) it issues a filtering command for the NALU-Topic, requiring to transfer only NALU data-samples with $ssid \leq k$, so enforcing the rate-adaptation.

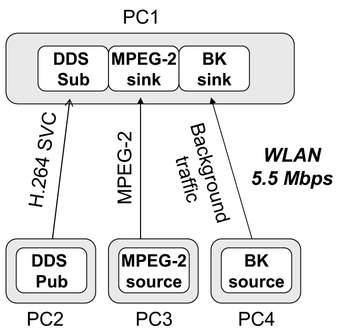

Fig. 2. Test-bed framework

To estimate the available capacity C we devise a technique involving both the video-publisher and video-subscriber modules. This technique is effective also in the case of multiple subscribers and operates as follows: after that the encoder has produced video NALUs, the video-publisher does not immediately send them to the NALU-Topic, but periodically accumulates a fixed number of NALUs (named *aggregation-set*), for instance a whole GOP. At the start of the next aggregation, the NALUs of the previous aggregation-set are sent to the NALU-Topic, all at once. Since they form a continuous block of data, the NALUs of an aggregation-set (complying with the filtering condition) are transferred to the subscriber at the maximum available bit rate of the network (the sender does not introduce any gap between the transmission of a NALU and the transmission of the next NALU). On the receiver side, the video-subscriber identifies the aggregation-set [1] and estimates the available capacity as the ratio between the number of bits of the aggregation-set and its duration.

The aggregation of NALUs allows estimating the available capacity but has two drawbacks: it adds a delay and it increases the traffic burstiness. As regards delay, it can be limited by reducing the size of the aggregation-set, even if its duration has to be quite greater than the time-resolution of the computing device. As regards burstiness, it can be reduced by shaping the bit flow in such a way that the maximum flow bit-rate does not exceed the rate of the full-quality video, B_f, i.e. the video containing all possible sub-streams. When the available capacity is lower than B_f, the rate-control limits the video bit-rate to the available capacity, thus burstiness is limited. When the available capacity is greater then B_f, the shaper limits the burstiness of the emitted bit rate to the burstiness of the full quality video.

V. Demonstration

We show the effectiveness of the architecture in the test-bed framework reported in Fig.2. We have four PCs connected by an ad-hoc WLAN at 5.5 Mbit/s. During the demonstration we stream the same video in two different ways, at the same time: by using H.264 over DDS and by using plain MPEG-2. We can compare real-time the two alternatives. The H.264 video contains 3 quality layers (i.e., $qid = 0, 1, 2$) and five frame-rates (i.e., $tid = 0, 1..4$: from 1.875 to 30 fps); the bit-rate of the full-quality video is about 2 Mbit/s. The MPEG-2 encoding produces a video with a rate of 2 Mbit/sec at 30 fps. In addition to the two videos, we inject a background UDP traffic. During the test we vary the amount of the background traffic, so varying the capacity left available for the video streams. To deploy our solution, we use a Java DDS implementation developed by Space Software Italia (http://www.ssi.it); the rate-control is performed every 2 seconds. We use VLC to stream the MPEG-2 video, MPlayer as receiving client and the IPERF tool to generate the background traffic.

[1]The first data-sample of an aggregation-set has the marker-bit set to 1.

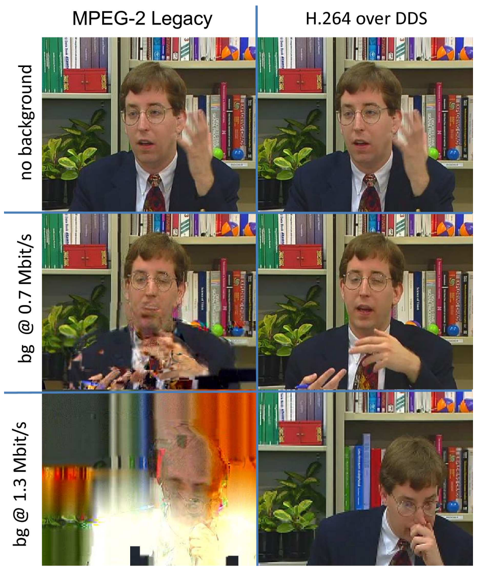

Fig. 3. Snapshots with different background traffic throughput

Fig. 3 shows three couples of snapshots captured during the demonstration, with a background (bg) traffic set to 0, 0.7 and 1.3 Mbit/s, respectively. Without background traffic, the transfer capacity available to the video streams is enough to avoid loss of video units, and both videos look the same. If we increase the background traffic, random losses of video units affect the MPEG-2 stream, while our rate-control succeeds in maintaining a very good perceived quality of the video transmitted with H.264 over DDS. Increasing again the background traffic causes a harsh degradation to the MPEG-2 video, while the quality H.264 over DDS degrades very smoothly, in a way not visible in the picture.

References

[1] Object Management Group, *Data-Distribution Service for Real-Time Systems*, Version 1.2, , January 2007, http://www.omg.org/
[2] ITU-T recommendation H.264: *Advanced video coding for generic audio visual services*, International Telecommunications Union, Nov. 2007.
[3] T. Schierl, T. Stockhammer, T. Wiegand, *Mobile Video Transmission using Scalable Video Coding(SVC)*,IEEE Trans. on Circuits and Systems for Video Technology, June 2007.
[4] G. Bianchi, A. Detti, P. Loreti, C. Pisa,F. S. Proto, W. Kellerer, S. Thakolsri ,J. Widmer, *Application-aware H.264 Scalable Video Coding delivery over WLANs: Experimental Assessment*,IEEE IWCLD,June 2009, Palma de Mallorca.