MediaWiN 2009
5 July 2009 - Sousse, Tunisia
2009 IEEE Workshop on multiMedia Applications over Wireless Networks

# *SVEF: an Open-Source Experimental Evaluation Framework for H.264 Scalable Video Streaming*

*Andrea Detti, Giuseppe Bianchi, Claudio Pisa, Francesco Saverio Proto, Pierpaolo Loreti*

University of Rome "Tor Vergata", Italy

*Wolfgang Kellerer, Srisakul Thakolsri, Joerg Widmer*

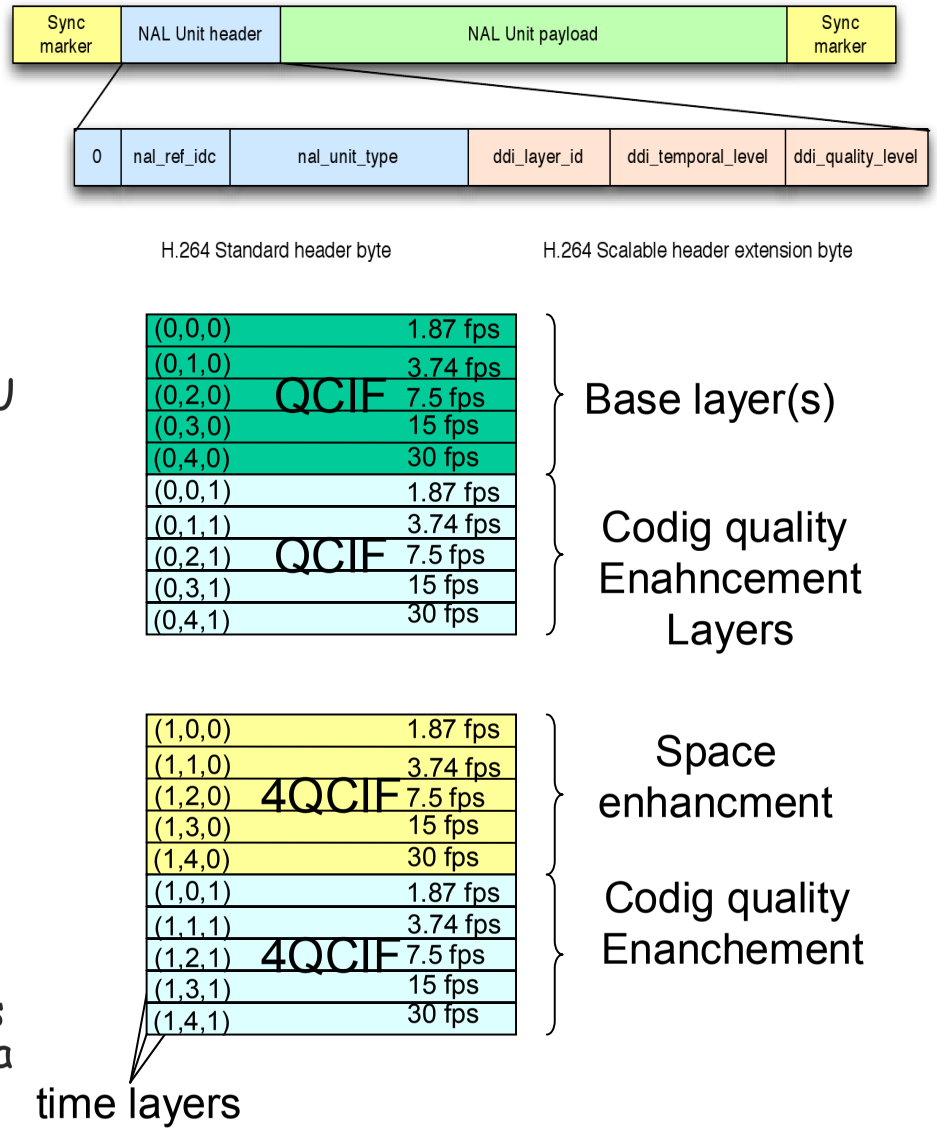
NTT docomo DOCOMO Euro-Labs

# A winning team: WLAN+SVC

- Wireless networks are usually characterized by a network capacity that may quickly vary over time
  - Backlogged stations
  - Auto-rate mechanisms
- Scalable Video Coding (SVC) makes possible to quickly and efficiently control the video bit-rate
  - H.264 video coding standard extending H.264 AVC (Blu-Ray)
  - RTP mature internet draft

# H.264 SVC Coding background

- SVC video is formed by enhancement substreams
  - Spatial resolution : permits appropriate resolution
  - Temporal resolution permits convenient frame rate
  - Coding refinement permits suitable data rate
- Substream is a sequence of NALUs, NALU header contains the tupla **[Did, Tid, Qid]**, i.e. the role of the substream
  - (Did=0,Tid=0,Qid=0) smaller spatial resolution (Did=0), smaller frame rate (Tid=0) of 1.87 fps and smaller coding quality (Qid=0)
  - (Did=0,Tid=1,Qid=0) increases the frame rate of (Did=0,Tid=0, Qid=0) toward 3.74 fps
- Each SVC video is at least formed by five substreams (Did=0,Tid=0,1,2,3,4,Qid=0) that are named Base Layer(s) and that is an H.264 AVC video
- The substreams removing smooth reduces quality and bit-rate and it is as simple as a packet inspection and a dropping – NO COMPLEX TRANSCODING - embedded

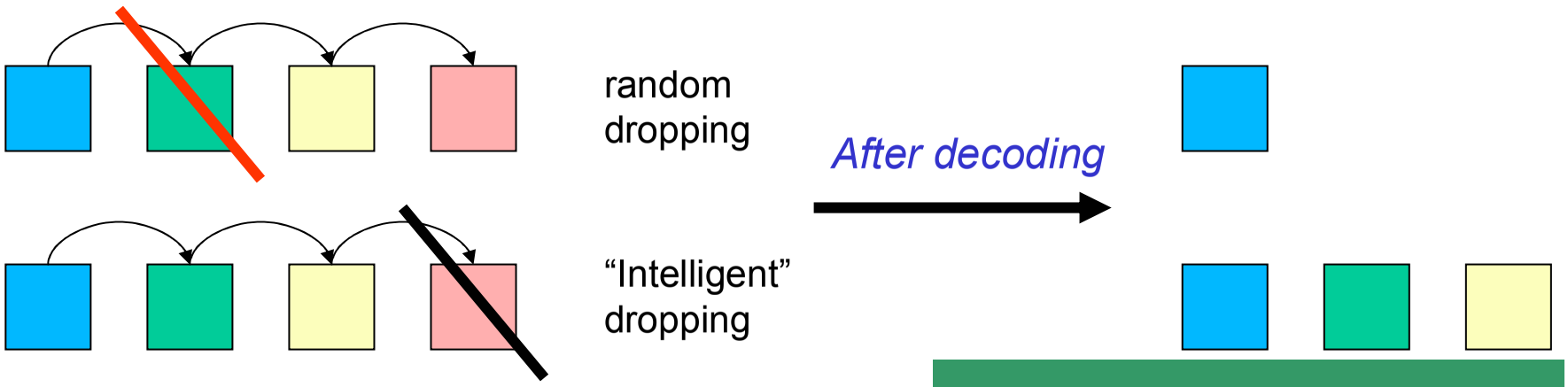


# H.264 SVC NALU dependency

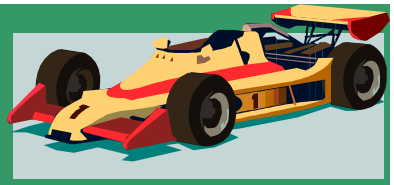
SVC codes layers by prediction, this involves decoding dependency

- Spatial and Temporal NALUs decoding need **two** previous NALUs
  - [did>0,tid>0,0] depends on [did-1,tid,0] (if exists) and [did,tid-1,0]
- Medium grain NALU: decoding need **one** previous NALU
  - [did,tid,qid>0] depends on [did,tid,qid-1]

- Random drop may involves a lot of non-decodable NALU



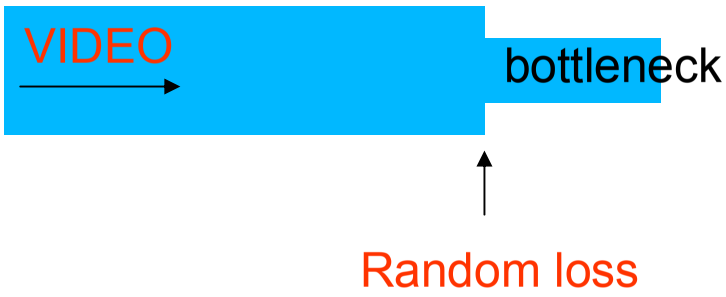
# The scientific challenge

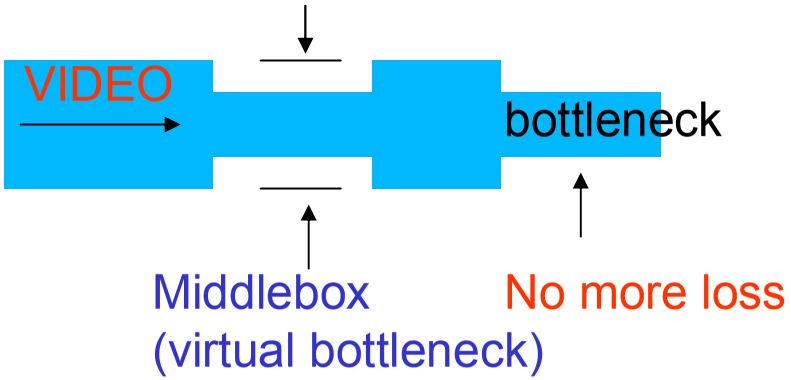


- Design an intelligent scheduling policy adapting the video bit-rate to the actual bottlenecks capacity while maximizing video quality (PSNR)
- Scheduler can be placed IN the bottleneck device or OUT of it (MIDDLEBOX) in the middle between source and bottleneck

# Middlebox

- Avoid random packet loss on the actual bottleneck devices
- Device throttling the video bit-rate at the bandwidth (C) available on the next bottleneck (e.g., WLAN)
- SVC bit-rate throttling through intelligent NALU scheduling that removes or grant enhancement substreams



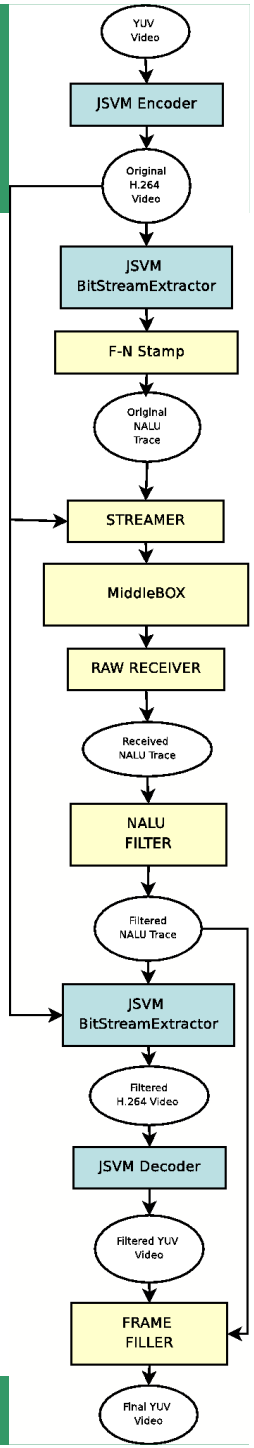



# Network Experimet with SVC: why an issue?

- Public domain library (JSVM) for SVC coding/decoding
- But many elements non public available for an experimental trial
  - SVC Streaming server
    - And related hinting module
  - Robust decoding
    - JSVM crashes when feeded with streams affected by NALU with missing dependency (e.g., random losl)
  - Basic concealment
    - JSVM does not reconstruct missing video frames
  - SVC Player

# SVEF

- YUV video encoded with JSVM Encoder
- NALU trace-file obtained by JSVM BitStreamExtractor ( we add Frame number)
- Encoded video and NALU Trace are passed to the Streamer that, parsing the trace file NALU by NALU at the right time, gets the right bytes block from the encoded video file, add an RTP "pseudo" header and transmits the NALU in a SINGLE IP PACKET that will be probably segmented by lower layer.
- Middlebox (later ….)
- NALUs cross the network and are received by a RAW receiver that parsing the received RTP headers reconstruct the trace-file of the received NALUs
- Trace-file is cleaned from undecodable NALUs because of missing dependency (or too delayed)
- Filtered trace-file is used build an H.264 file without missing dependency that is decoded by JSVM, producing a YUV
- Frame filler conceals the YUV file repeating last frame where a frame in the Filtered video is missed for network impairments, thus we have a YUV file with the same number of frame of the original one




| mem-offset | NALU-size | DID | TID | QID | Frame-Number |
|---|---|---|---|---|---|

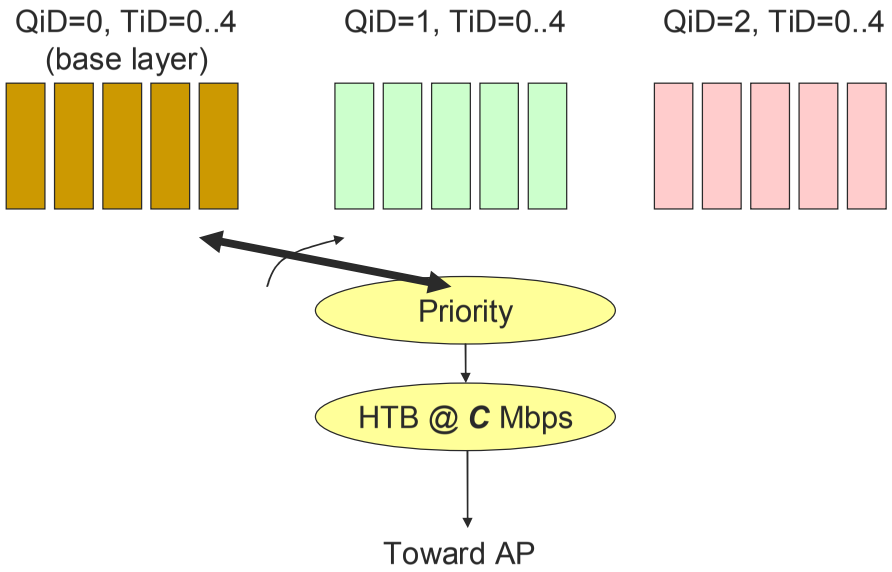
Fig. 2. NALU-trace entry

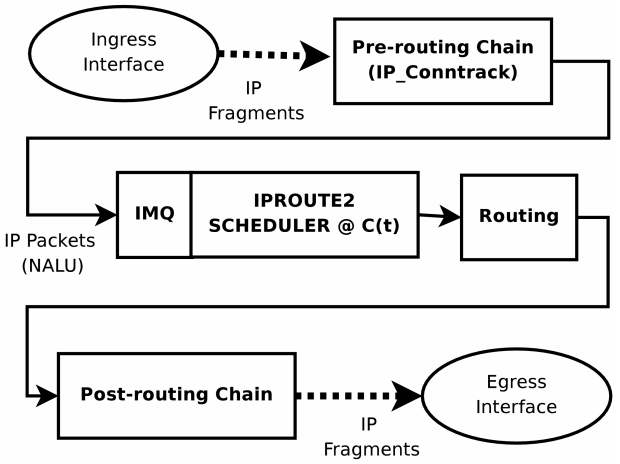
| 0 | | | 31 |
|---|---|---|---|
| DID | TID | QID | flags |
| mem-offset | | | |
| NALU-size | Frame-num | | |

Fig. 3. Layer-5 header

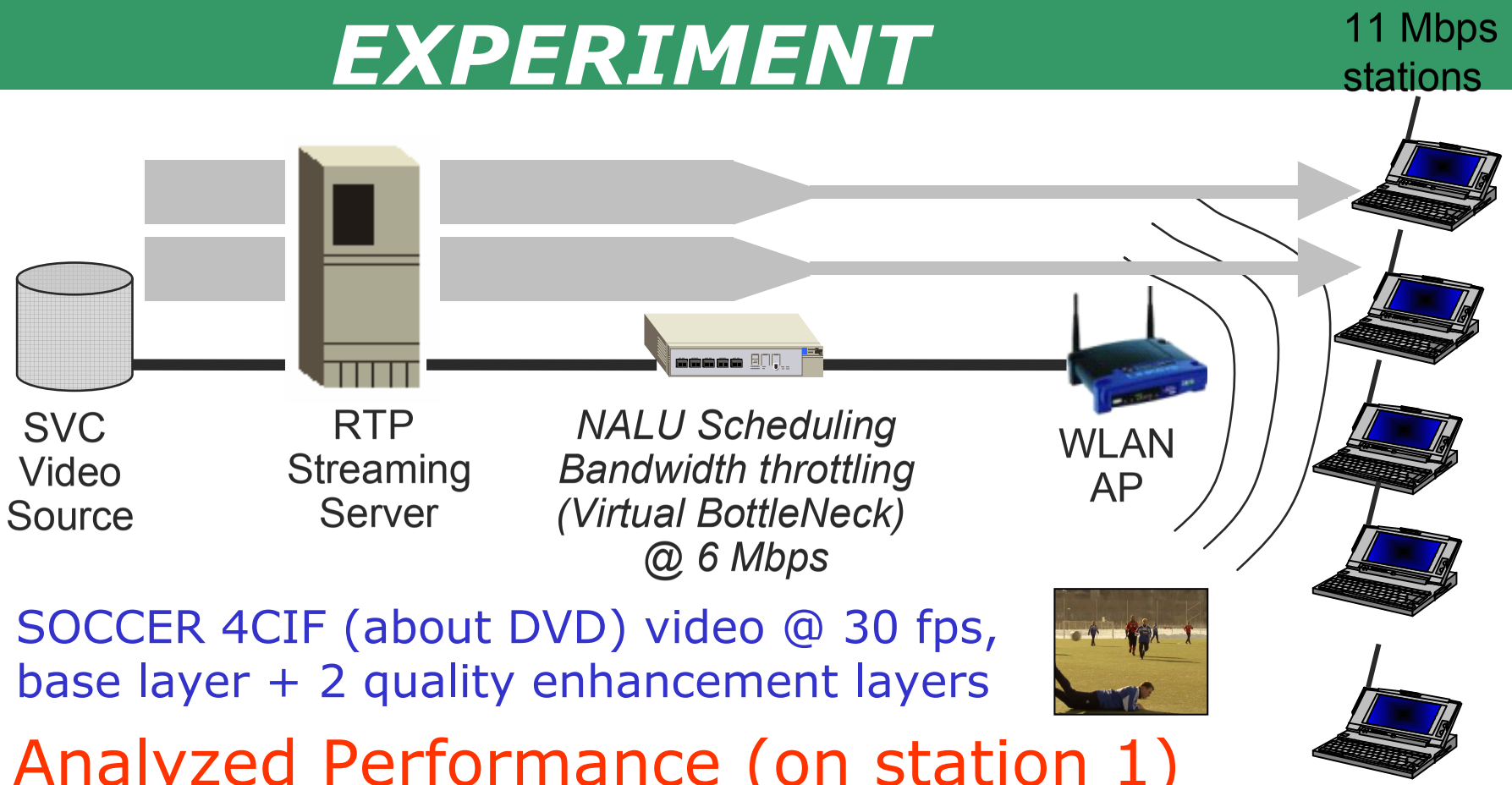
# SVEF Middlebox and a scheduler example

- One NALU per IP packet allow simple NALU reassembling through IP_CONTRACT
- Reassembled NALU are passed through a virtual interface (IMQ) on which Linux Trafic control toold perform scheduling
- Simple scheduler for a SVC video formed by base layer and two additional quality layers (QiD=1, QiD=2):
  - one queue for substream
  - substream priority ordered accounting of quality and dependecncy



# WLAN MIDDLEBOX EXPERIMENT

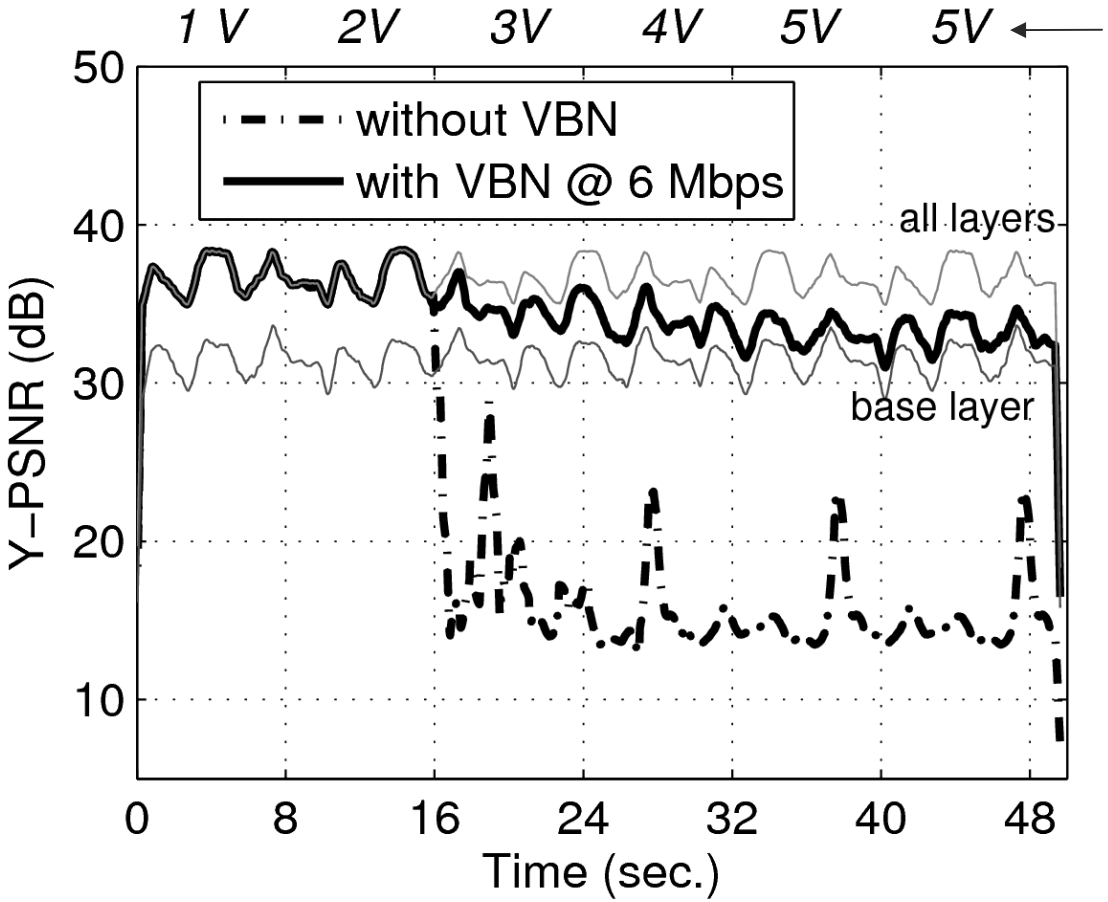



SOCCER 4CIF (about DVD) video @ 30 fps, base layer + 2 quality enhancement layers

## Analyzed Performance (on station 1)

- PSNR: classical distortion measure between two videos, i.e. uncoded original and received coded
  - about 40dB practically equal, 10 dB practically noise
- (novel) Transmission Efficiency: ratio among useful NALU received and all NALUs received. Useful may be less than all because of missing dependency

# Experimental result: with and without scheduler (VBN)




Base Layer @ 650 kbps

Enhancement Layer 1 @ 900 kbps

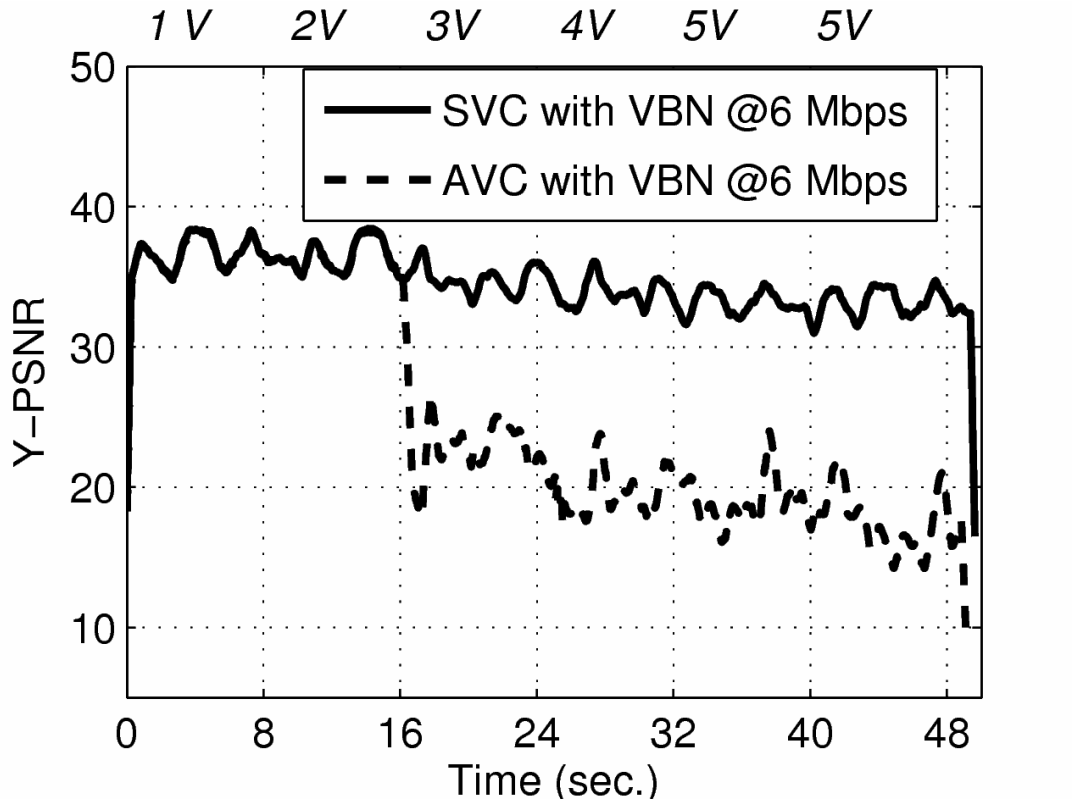
Enhancement Layer 2 @ 1300 kbps

---

ALL @ 2850 kbps

TX Eff. with scheduler 100%, without 54 %

# Experimental result: AVC vs SVC



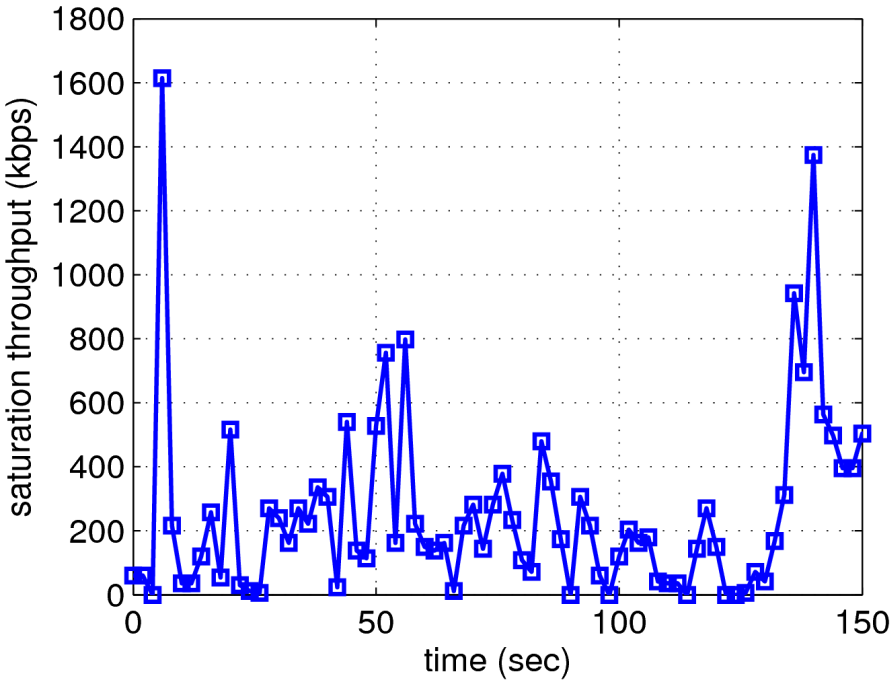

TX Eff. SVC 100%, AVC 99.94 %

- AVC coded at the same bit rate of SVC (i.e., 2.8 Mbps)
- AVC scales only in terms of frame rate -> high distortion
- SVC Quality scaling tackles the issue

# *Ongoing work and findings*

# WLAN MIDDLEBOX: some issues

- We wish to run-time estimate the capacity *C* from the Middlebox (VBN) in an operative WLAN condition (e.g., multirate, uplink traffic, etc.)
- We developed several tracking algorithms that failed because of in real conditions the WLAN capacity quickly change
- Thus we are actually developing the NALU scheduler within a LINUX AP where we modified the mad-wify driver to retrieve capacity information

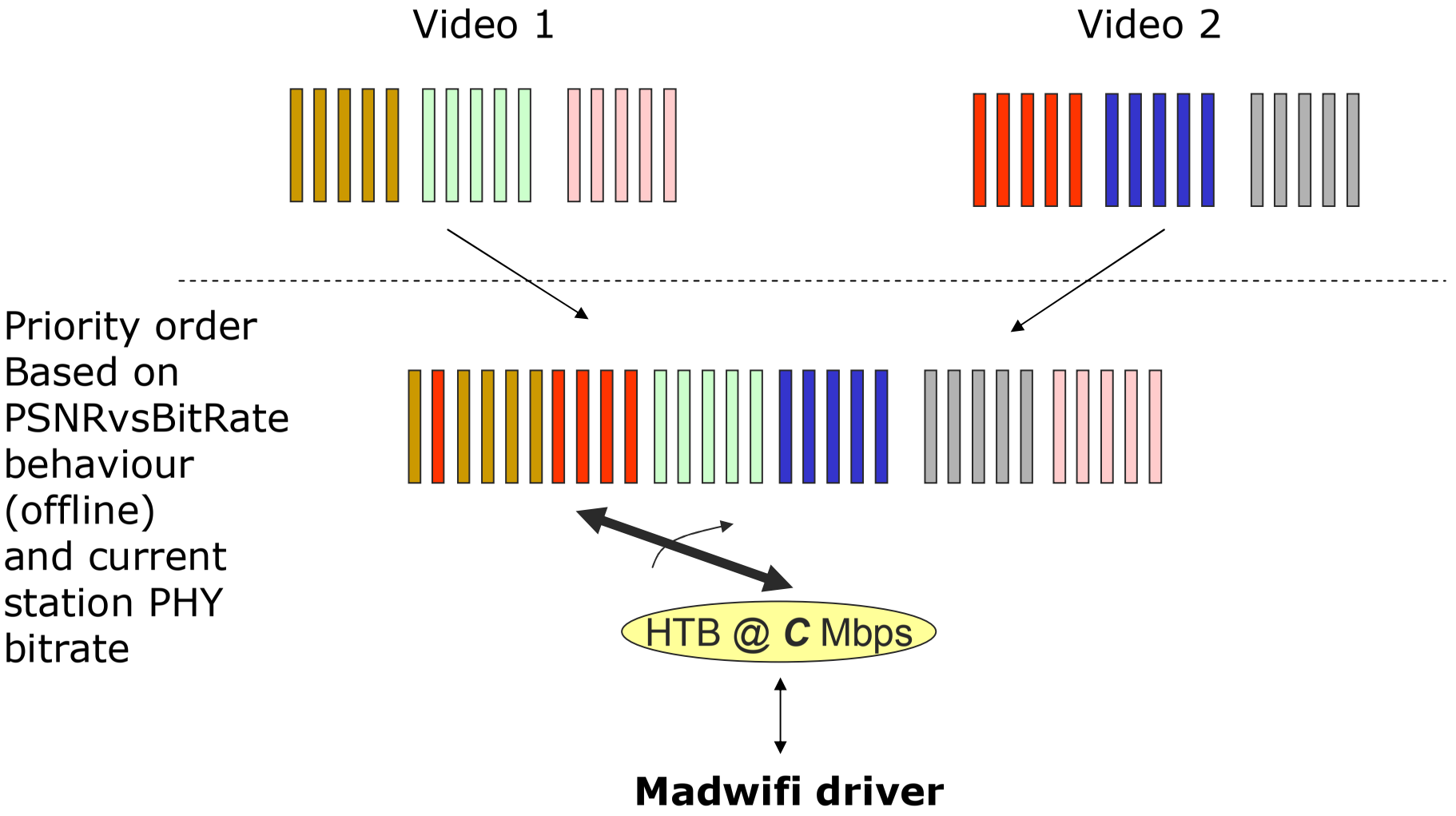



Two stations: one near the AP and one placed away. Downlink saturation throughput available

# Different video, different station PHY rate…what is the best cross-layer scheduler

- Assumptions: current station bit rate is know and PSNRvsBitRate is known
- AP has amount of time to dedicate to the video streams … what is the right distribution of such a time that maximize the cumulative PSNR of video delivery ?
  - The overall bit-rate transferred by the AP depends on the time AP dedicate to each video/station
    - More time dedicated to slow (i.e., 1 Mbps) stations means less overall bit rate, and vise versa
  - Different videos have different bit-rate requirements to obtain same PSNR
- Linear programming formulation
- What is the best (sub-optimal) priority queuing scheduler maximizing PSNR

# The best priority queuing scheduler



# *Thank you*

Andrea Detti

University of Rome
"Tor Vergata", Italy