

Mobile Video Streaming in Modern Wireless Networks

Mohamed Hefeeda & Cheng-Hsin Hsu



Outline

- **Introduction**
 - **Importance of mobile multimedia**
 - **Distribution models**
- **Mobile Multimedia Multicast/Broadcast**
 - **Energy saving**
 - **Efficient and extensible mobile TV testbed**
 - **Video streaming over cooperative wireless networks**
 - **Supporting heterogeneous mobile receivers**

Outline (cont.)

- **Mobile Multimedia Unicast**
 - **Video compression and transport**
 - **Video transcoding for timely delivery**
 - **Buffer management in mobile video**
 - **Power-aware video streaming**
 - **Multihomed video streaming**
- **Summary and outlook**

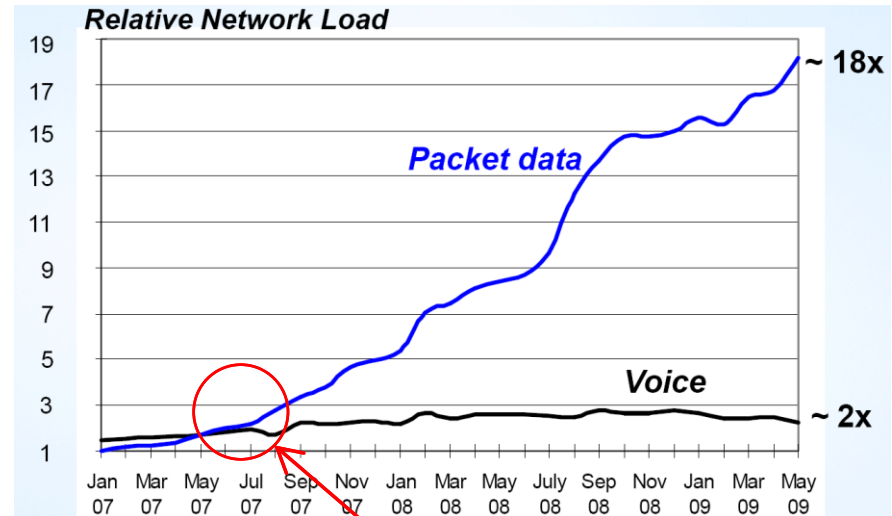
Part I

Introduction

Mobile Video Streaming

- **Market study [Rysavy] indicates that mobile data traffic exceeded voice traffic in mid-2007 in North America**

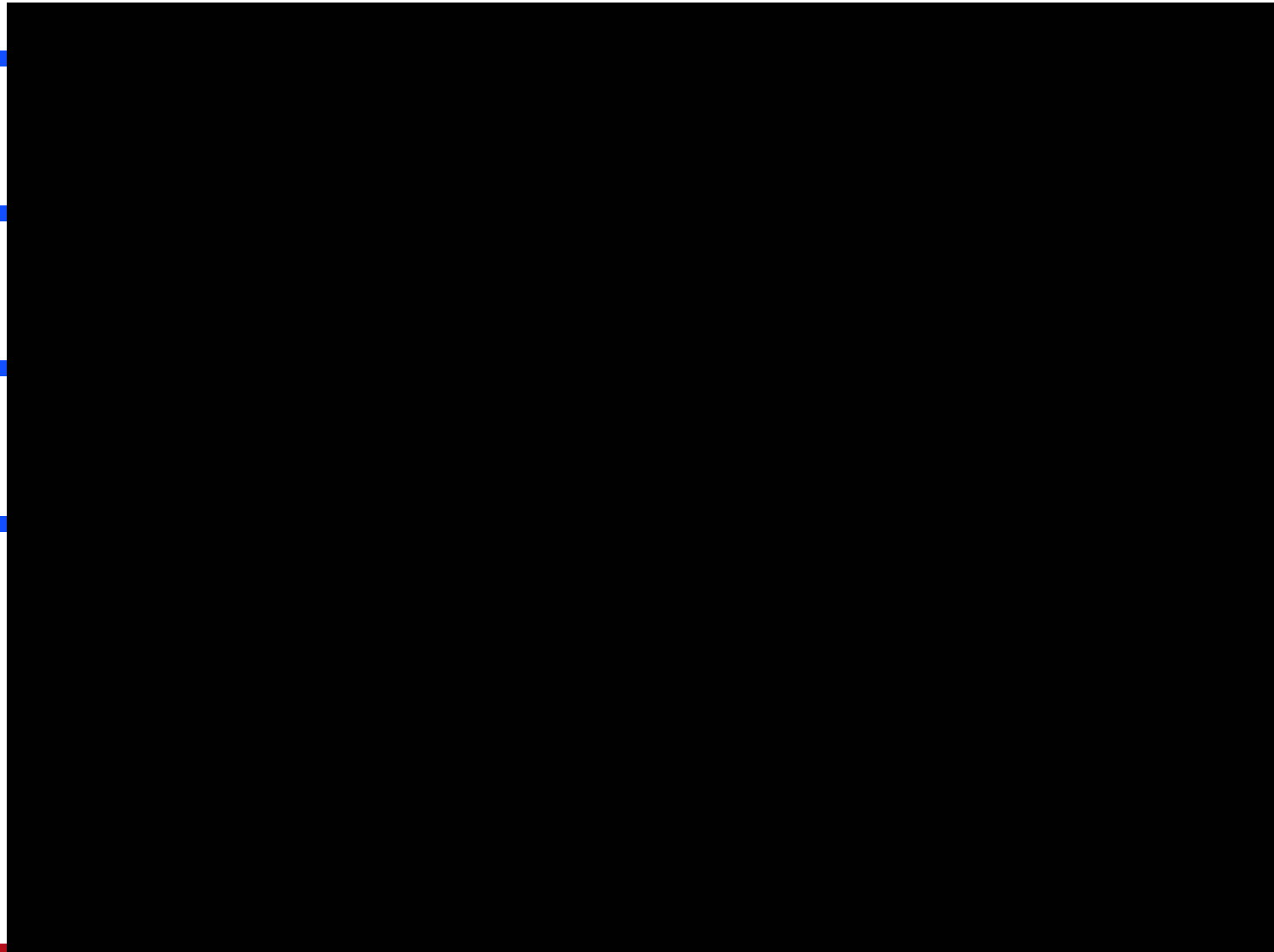
- **18 times increase in two years, and is not expected to slow down**



Release of iPhone

- **Touch screen smartphones allow users to easily access online and multimedia content**
- **Mobile video is the killer application for 3G data networks**
 - **As #1 app, video streaming consumes 35% of all data traffic [Allot]**

Business Opportunity



, e.g.,

g

e

Video Distribution Model

■ Unicast

- Individual connection for each mobile device
- E.g., connect to YouTube using your iPhone
- on-demand service → flexibility for users
- But very high load on the network and servers
- → limited capacity for serving videos
- iPhone users overloaded cell networks with even short video clips

■ Broadcast (Multicast)

- One common stream received by many receivers
- Cost-effective for serving high-quality videos to numerous users
- E.g., live streaming events: political debates, soccer (football), ...
- Not on-demand (receive whatever on the Program Schedule)
- Also known as **Mobile TV**

Mobile Multimedia: Models

- **Broadcast can be done on ...**
- **3G/4G Cellular networks**
 - **Multimedia Broadcast /Multicast Service (MBMS) extension**
 - **Reserve part of the download bandwidth**
- **WiMAX networks**
 - **IEEE 802.16e (mobile WiMAX)**
 - **Reserve part of the download bandwidth**
- **Dedicated networks**
 - **Explicitly set up for multimedia services**
 - **Provides larger bandwidth → more TV channels**
 - **User Interactivity: needs another network (e.g., cell network)**

Mobile Multimedia: Models

- **Examples of dedicated broadcast networks**
 - **DVB-H: Digital Video Broadcast-Handheld:**
 - Europe, International (open standard)
 - Extends DVB-T to support mobile devices
 - **ATSC – M/H: Advanced Television System Committee – Mobile/Handheld**
 - North America
 - Allows TV services to mobiles over portion of the spectrum, which is saved because of moving from analog to digital services
 - **MediaFLO: Forward Link Only**
 - North America (by Qualcomm)
 - **CMMB: China Mobile Multimedia Broadcasting**
 - China

Mobile Multimedia: Problems

- **Two kinds of challenges ...**
- **Research Problems**
 - **Need to be solved to provide efficient utilization of resources and offer good multimedia services**
- **Practical Challenges**
 - **Could slow down deployment**

Mobile Multimedia: Problems

■ Practical Challenges include ...

- **Wireless spectrum licensing and regulations: very complex**
- **Lack of multimedia content customized for mobile devices**
- **Deployment and coverage**
- **Managing user subscriptions and integration with other services**
- **Different standards, formats, etc**
- **.....**
- **not our focus**

Mobile Multimedia: Problems (1/2)

- **Research problems can be classified into two groups: broadcast and unicast video streaming**
- **Broadcast streaming**
 - **Scheduling transmission to save communication power**
 - **Reducing stream switching delay**
 - **Supporting heterogeneous mobile receivers**
 - **Video streaming over cooperative networks**
 - **Designing efficient and extensible mobile broadcasting testbed**

Mobile Multimedia: Problems (2/2)

■ Unicast streaming

- Buffer management in mobile video
- Stream adaptation using video transcoders
- Power-aware video streaming
- Rate control in multihomed video streaming
- Packet scheduling for mobile video

Part II

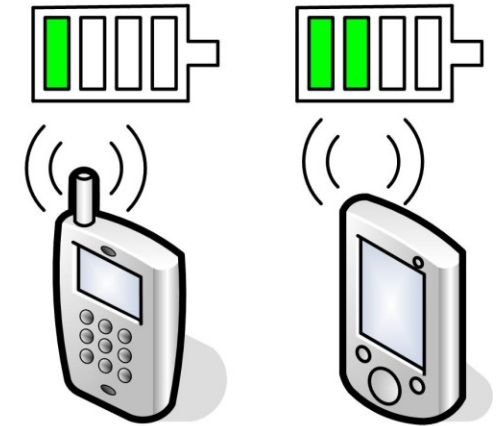
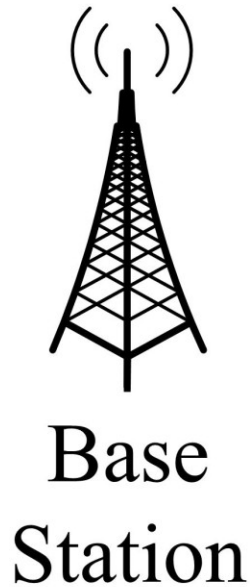
Mobile Multimedia

Multicast/Broadcast



System Model

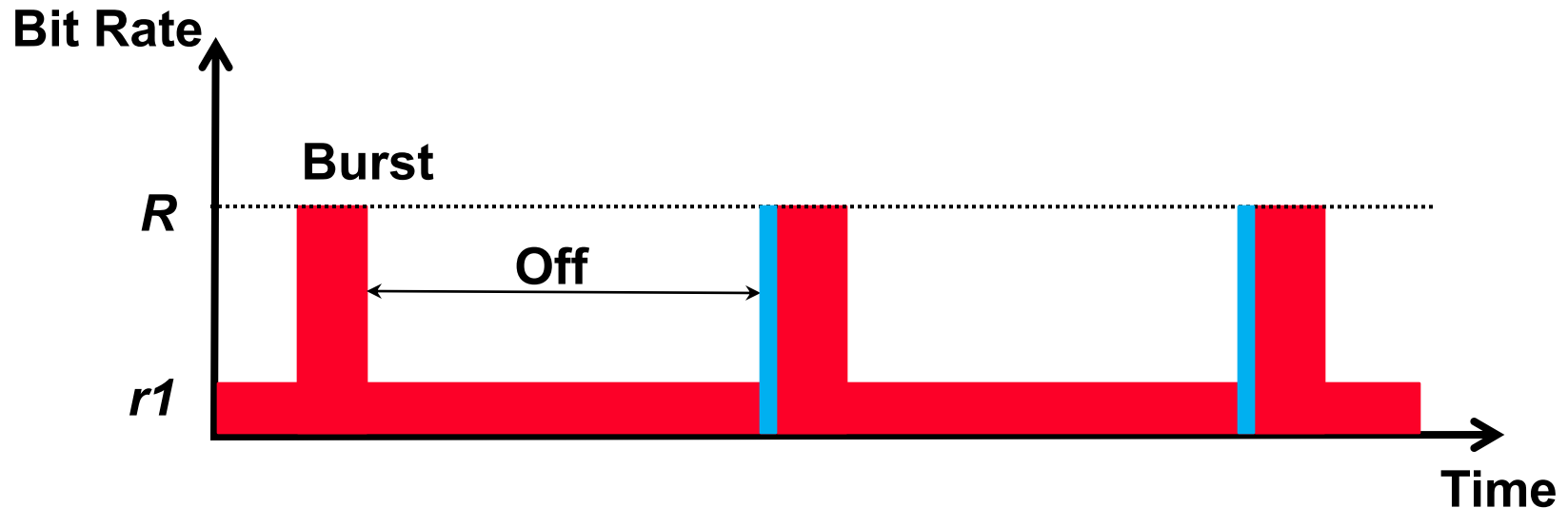
- Base station broadcasting multiple video streams (TV channels) to mobile devices



Mobile Devices

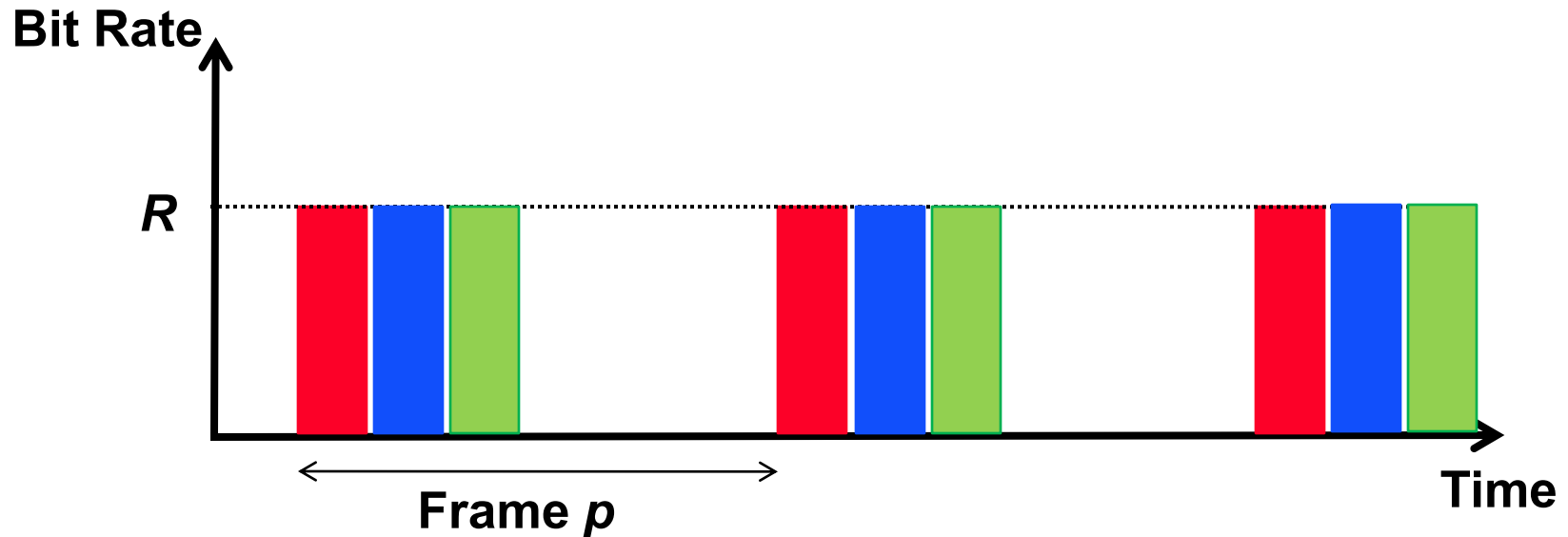
Energy Saving in Mobile Multimedia

Energy Saving for Mobile TV Receivers



- **This is called Time Slicing**
 - Supported (dictated) in DVB-H and MediaFLO
- **Need to construct Burst Transmission Schedule**
 - No receiver buffer under/over flow instances
 - No overlap between bursts

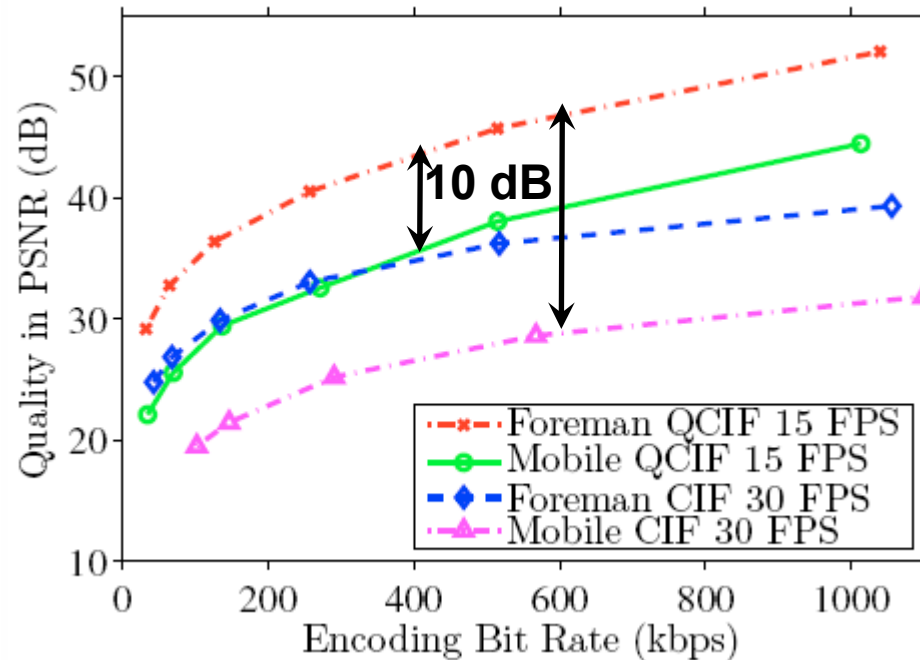
Burst Transmission Schedule Problem



- **Easy IF all TV channels have same bit rate**
 - Currently *assumed* in many deployed networks
 - Simple, but is it efficient (visual quality & bw utilization)?
 - TV channels broadcast different programs (sports, series, talk shows, ...) → different visual complexity/motion

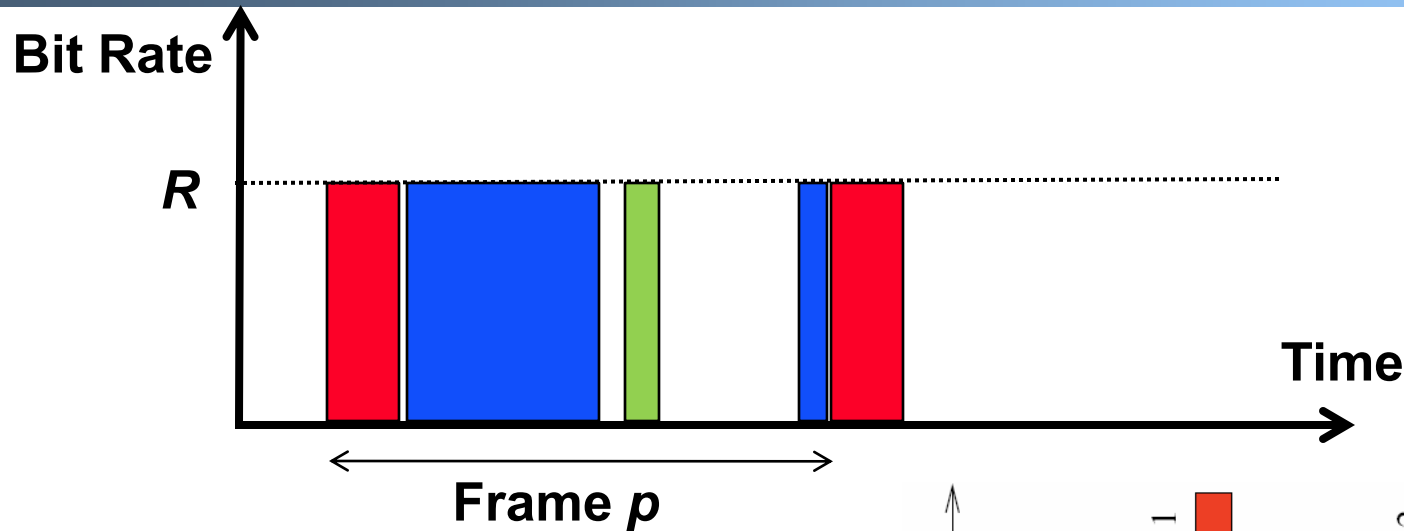
The Need for Different Bit Rates

- Encode multiple video sequences at various bit rates, measure quality

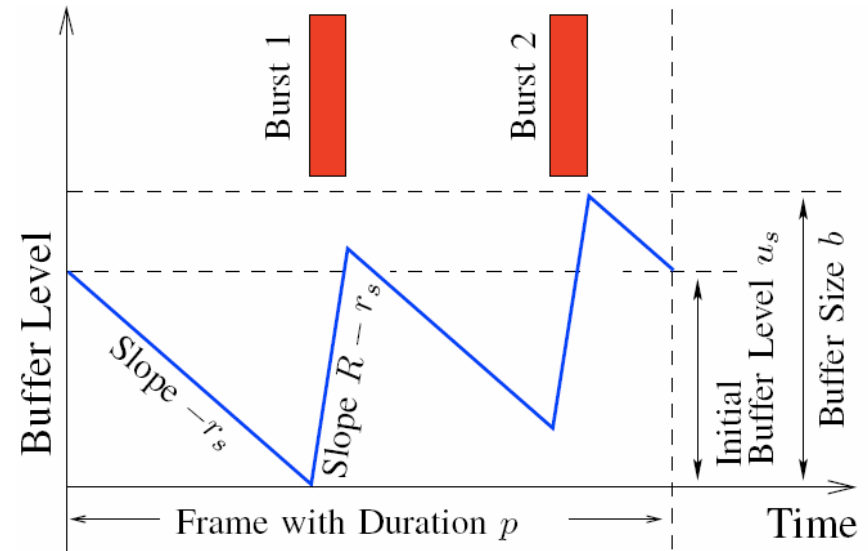


- Wide variations in quality (PSNR), as high as 10—20 dB**
- Bandwidth waste if we encode channels at high rate**

Burst Scheduling with *Different* Bit Rates

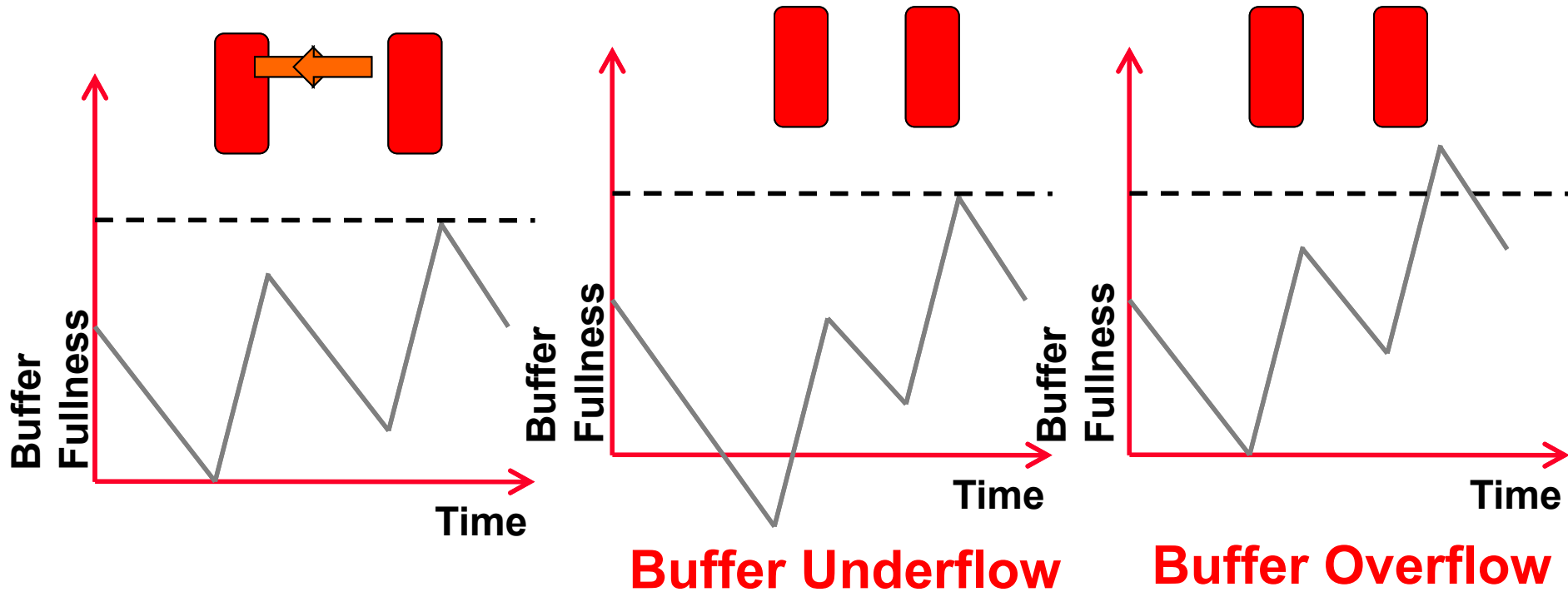


- Ensure no buffer violations for ALL TV channels
- Difficult Problem



Receiver Buffer Dynamics

- Receiver of a specific stream



Burst Scheduling with *Different* Bit Rates

- **Theorem 1: Burst Scheduling to minimize energy consumption for TV channels with arbitrary bit rates is **NP-Complete**** [Hefeeda 10, IEEE ToN]
- **Proof Sketch:**
 - We show that minimizing energy consumption is the same as minimizing number of bursts in each frame
 - Then, we reduce the Task Sequencing with release times and deadlines problem to it
- **We can NOT use exhaustive search in Real Time**

Solution Approach

- **Practical Simplification:**
 - Divide TV channels into classes
 - Channels in class c have bit rate: $r_c = r_1 \times 2^i, i = 0, 1, 2, \dots$
 - E.g., four classes: 150, 300, 600, 1200 kbps for talk shows, episodes, movies, sports
 - Present **optimal** and **efficient** algorithm (**P2OPT**)
- **For the General Problem** [Hsu 09, Infocom]
 - With any bit rate
 - Present a **near-optimal** approximation algorithm (**DBS**)
 - Theoretical (small) bound on the approximation factor
- **All algorithms are validated in a mobile TV testbed**

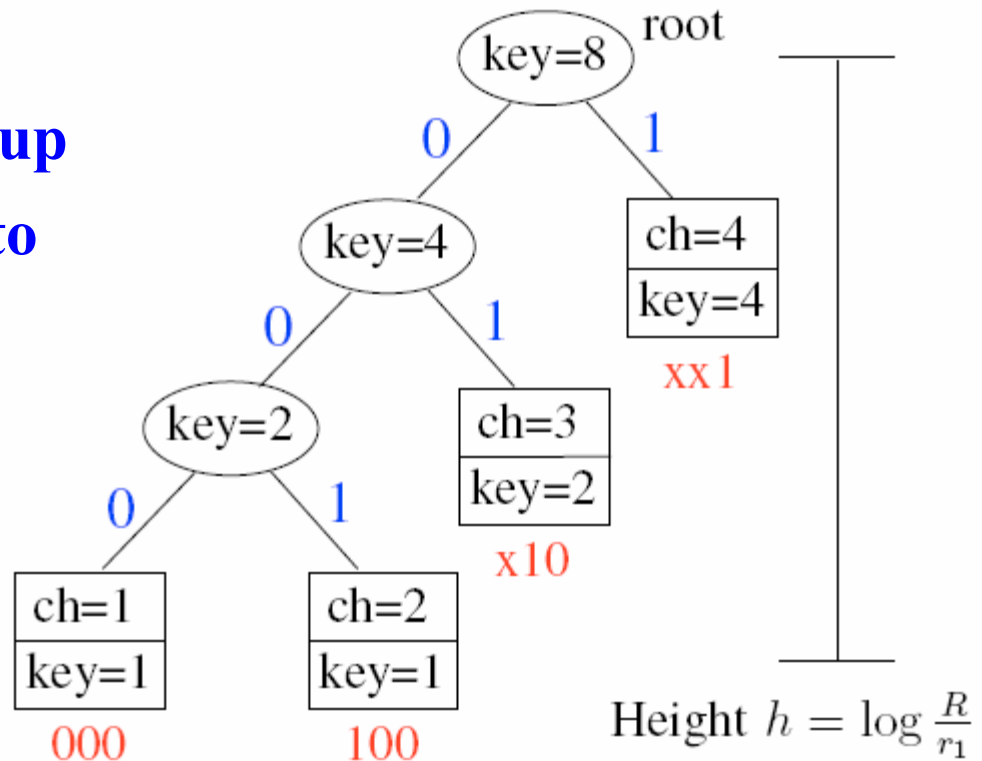
P2OPT Algorithm: Idea

- Assume S channels: $r_1 \leq r_2 \leq \dots$
- Also assume medium bandwidth $R = 2^k \times r_1$
- Compute the optimal frame length p^*
- Divide p^* into R / r_1 bursts, each $p^* r_1$ bits
- Then assign r_s / r_1 bursts to each TV channel s
- Set inter-burst distance as $p^* / (r_s / r_1)$

P2OPT: Example

- Four TV channels: $r_1 = r_2 = 256$, $r_3 = 512$, $r_4 = 1024$ kbps
- Medium bandwidth: $R = 2048$ kbps $= 8r_1$
- p^* is divided into 8 bursts

- Build binary tree, bottom up
- Traverse tree root-down to assign bursts



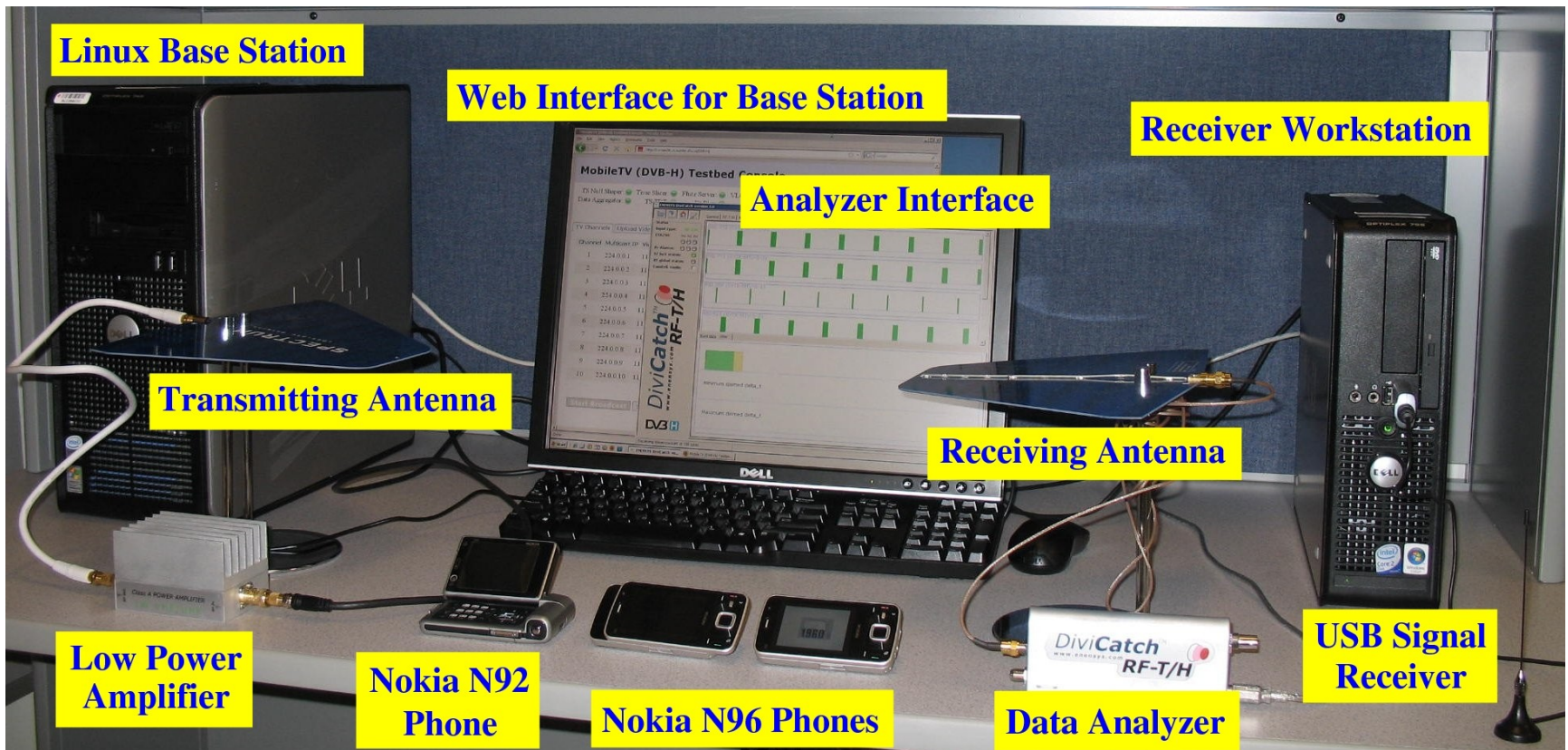
P2OPT: Analysis

- **Theorem 2: P2OPT is correct and runs in $O(S \log S)$**
 - i.e., returns a valid burst schedule **iff** one exists
 - Very efficient, S is typically < 50
- **Theorem 3: P2OPT is optimal when $p^* = b / r_1$**
 - Optimal = minimizes energy consumption for receivers
 - b is the receiver buffer size

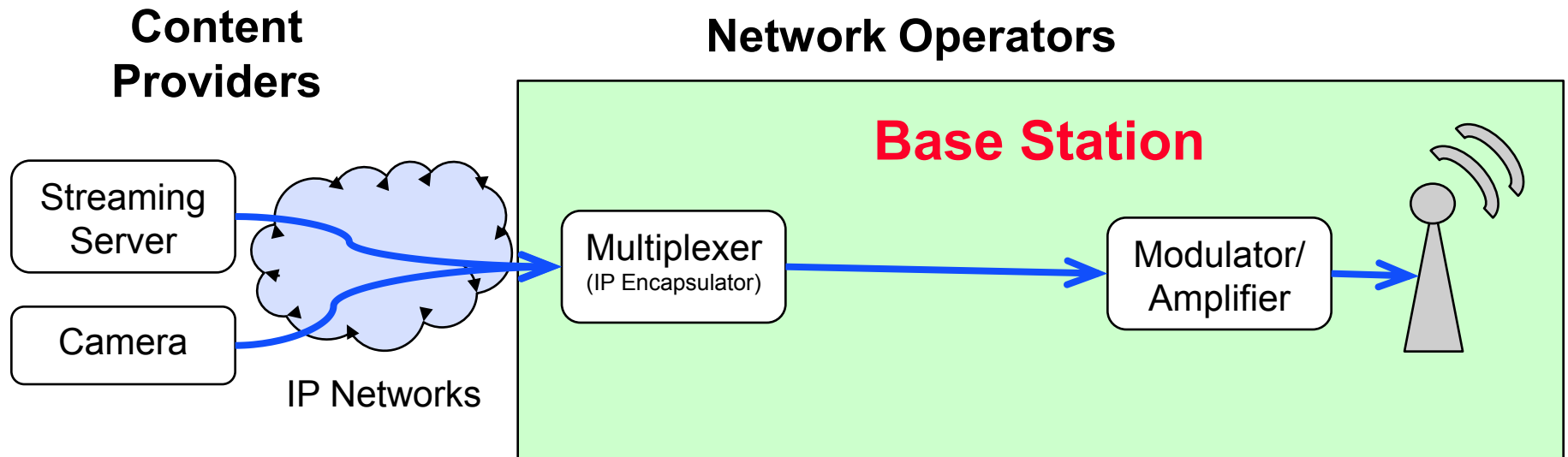
Efficient and Extensible Mobile TV Testbed for Empirical Evaluation

Empirical Validation

- Testbed for DVB-H networks [Hefeeda 10, TOMCCAP]



Mobile TV Networks: Big Picture



- **Program feeds are IP streams from streaming servers or cameras**
- **Multiple TV programs are multiplexed AND time sliced by a multiplexer into a MPEG-2 TS (Traffic Stream)**
- **The MPEG-2 TS stream is modulated, amplified, and broadcast to mobile devices**

Our Goal

- **Design complete mobile TV base station for**
 - **Academic prototyping and research**
 - **cost-efficient small- to medium-size deployments**

- **We use it to analyze: energy consumption, channel switching delay, network capacity, perceived quality, ...**

- **Could broadcast 10-20 TV channels with a commodity PC or low-end server**

Current Solution

- **Commercial Base Stations**
 - expensive, e.g., a single EXPWAY FastESG server costs 75k USD [Sarri09]
 - a complete base station costs even more
- **and sold as Black Box → cannot modify code**

**Need cost-efficient,
open-source, base station!**

Design Goals

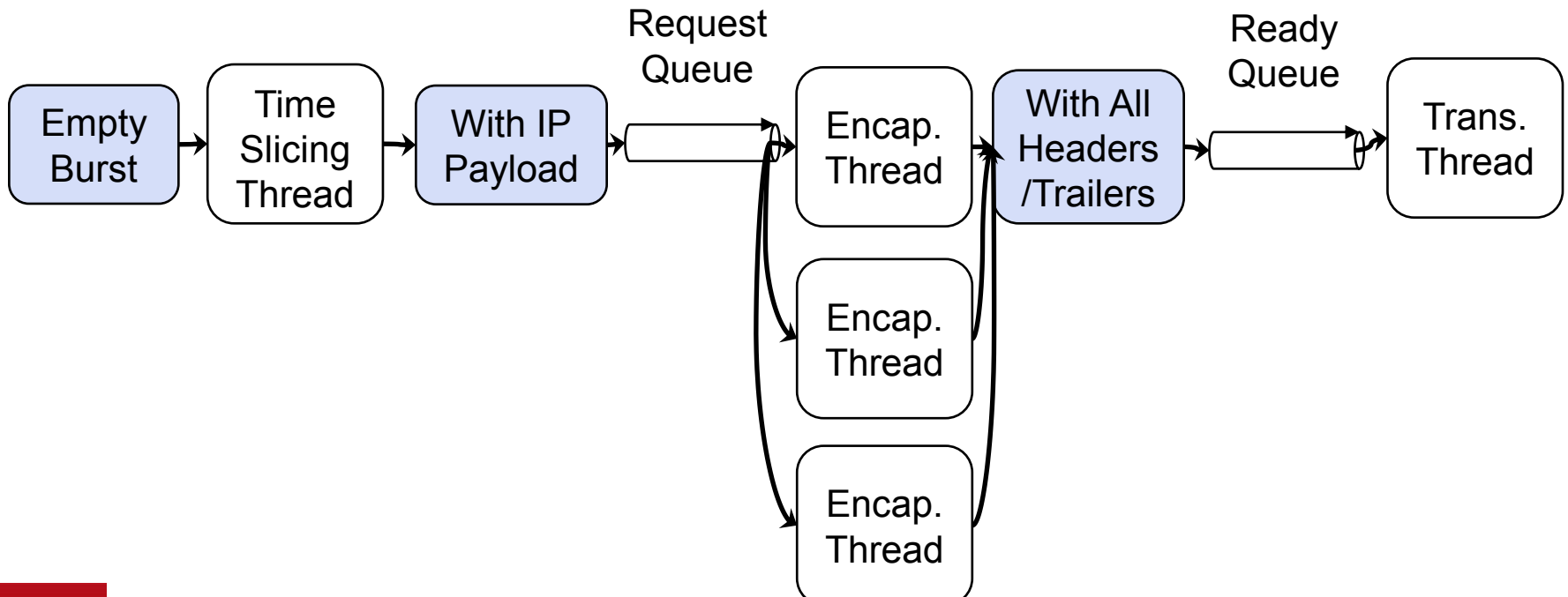
- **[G1] High efficiency and scalability**
 - **avoid disk I/O's and memcpys**
- **[G2] Utilization of multi-core processors**
 - **pipelined structure to allow parallelism**
- **[G3] Integrated software solution**
 - **centralized admin interface**
- **[G4] Extensible**
 - **future supports for other networks such as MediaFLO, WiMAX, and MBMS**

Design Decisions (1/3)

- **[D1] Use **Burst** as the unit of time slicing, encapsulation, and transmission.**
 - **Burst is self-contained with IP payloads and headers/trailers of all protocols**
 - **No disk I/O's for intermediate data**
 - **No memcpy's for IP payloads**

Design Decisions (2/3)

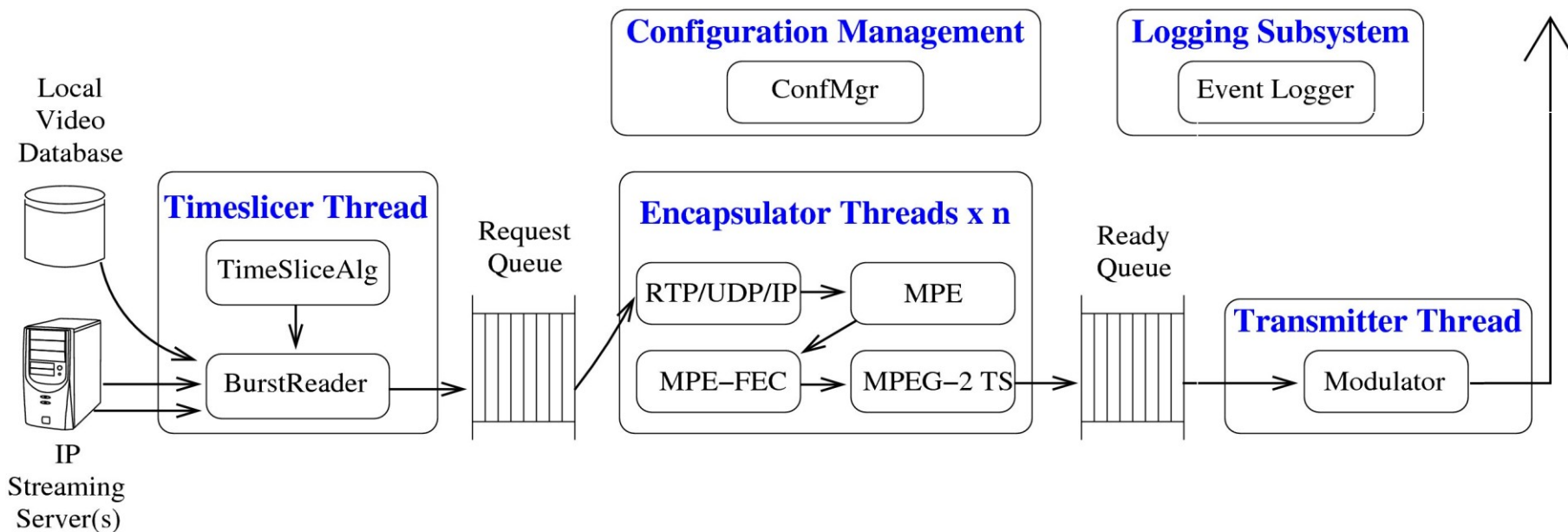
- [D2] Divide the base station into three indep. Phases, which are connected by two priority queue
 - pipelined and parallelism



Design Decisions (3/3)

- **[D3] Implement a centralized Configuration Manager to allow save/restore settings**
 - **interface with Web GUI for management**
- **[D4] Modularized design for future extensions**
 - **For example, MPE-FEC Burst is a subclass of MPE Burst**

Software Architecture



Mobile TV Testbed: Summary

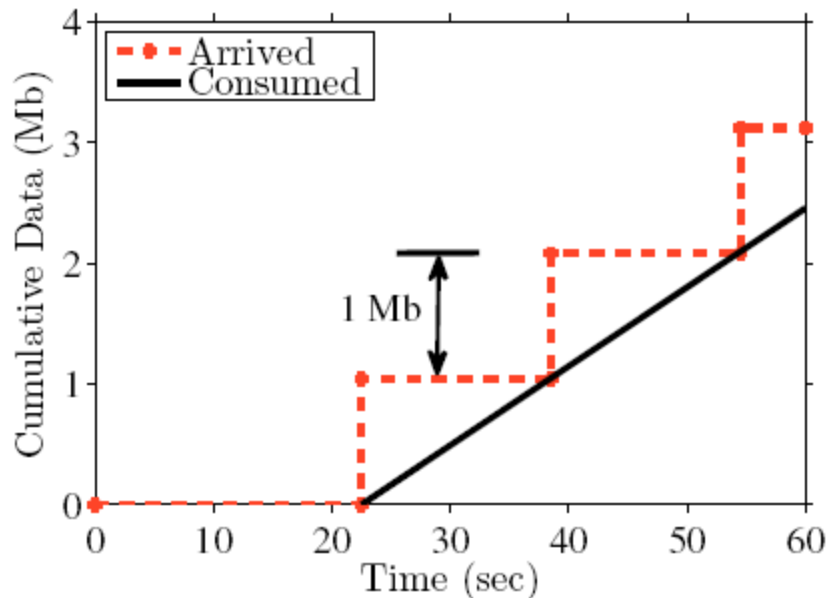
- **Complete Testbed for mobile video streaming**
- **Lessons:**
 - 1. Validation in actual systems is very valuable**
 - **Much more than simulation**
 - 2. Clearly define design goals**
 - 3. Leverage available hardware (multicore processors)**
 - **Carefully consider synchronization issues among threads**
 - 4. Adopt modular design with well-defined interfaces**
 - 5. Share with the research community**

P2OPT: Empirical Evaluation

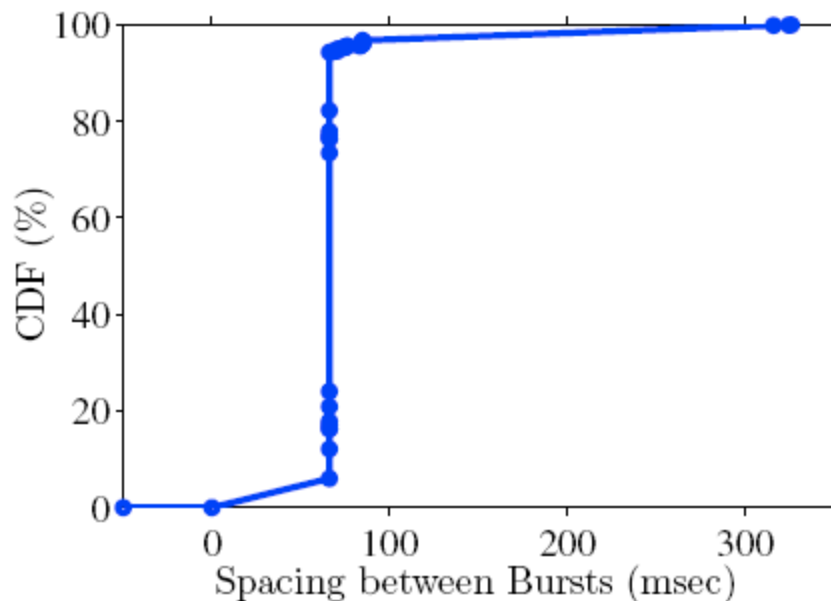
- **P2OPT is implemented in the Time Slicing module**
- **Setup: Broadcast 9 TV channels for 10 minutes**
 - **4 classes: 2 @ 64, 3 @ 256, 2 @ 512, 2 @ 1024 kbps**
 - **Receiver Buffer = 1 Mb**
 - **Collect detailed logs (start/end of each burst in msec)**
 - **Monitor receiver buffer levels with time**
 - **Compute inter-burst intervals for burst conflicts**

P2OPT: Correctness

TV Channel 1



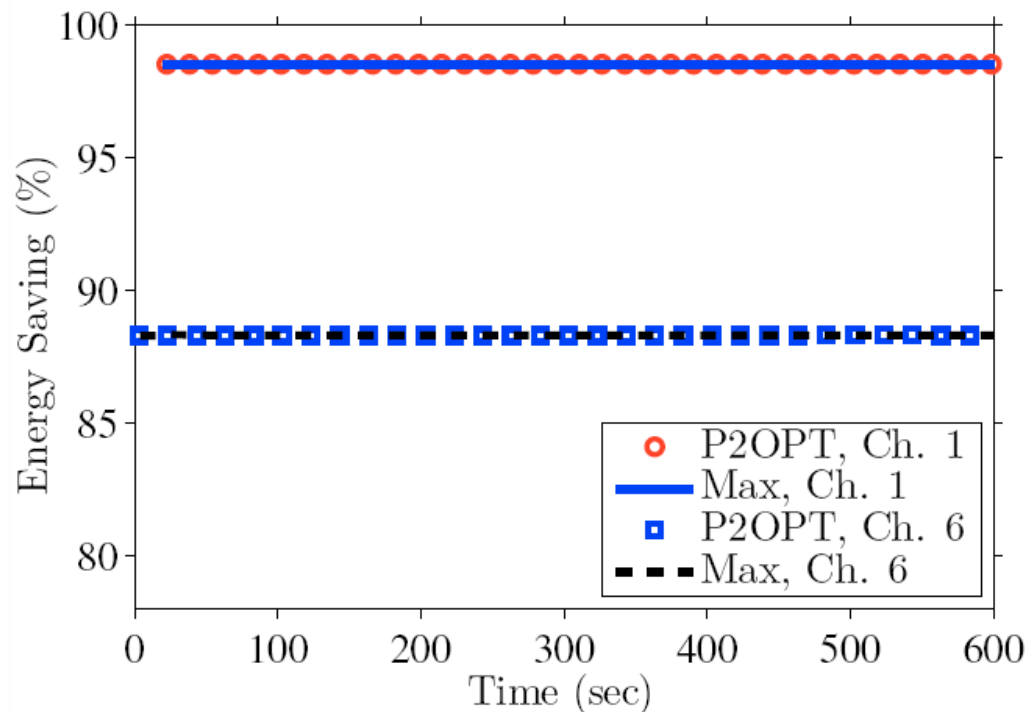
Bursts of all TV Channels



- Never exceeds 1 Mb, nor goes below 0
- No overlap, all positive spacing
- And P2OPT runs in **real time** on a commodity PC

P2OPT: Optimality

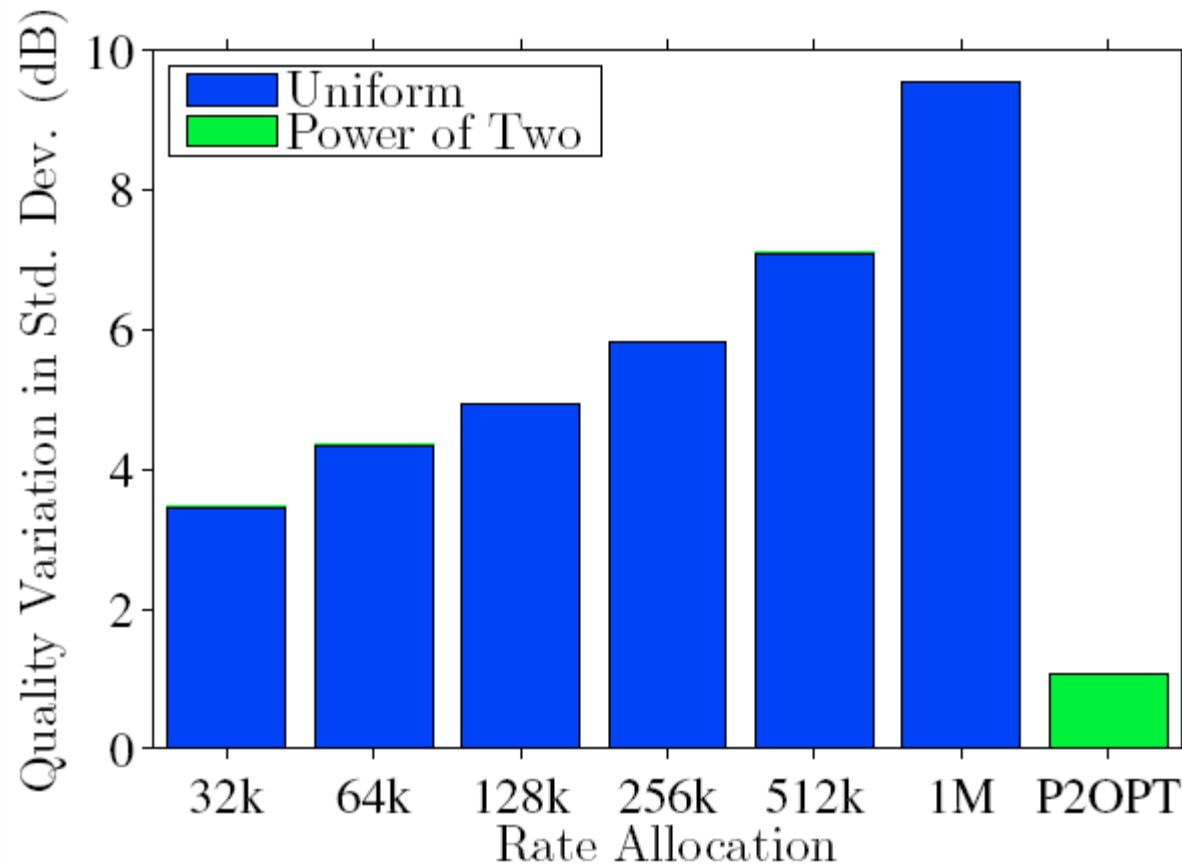
- Compare energy saving against **absolute maximum**
 - Max: broadcast TV channels **one by one**, freely use the largest burst
→ max off time → max energy saving
 - P2OPT: broadcast all TV channels **concurrently**



P2OPT: Quality Variation

- Does encoding channels with power of 2 increments bit rate really help?
- We encode ten (diverse) sequences using H.264:
 - Uniform: all at same rate r (r varies 32 -- 1024 kbps)
 - P2OPT: at 3 different bit rates

P2OPT: Quality Variation



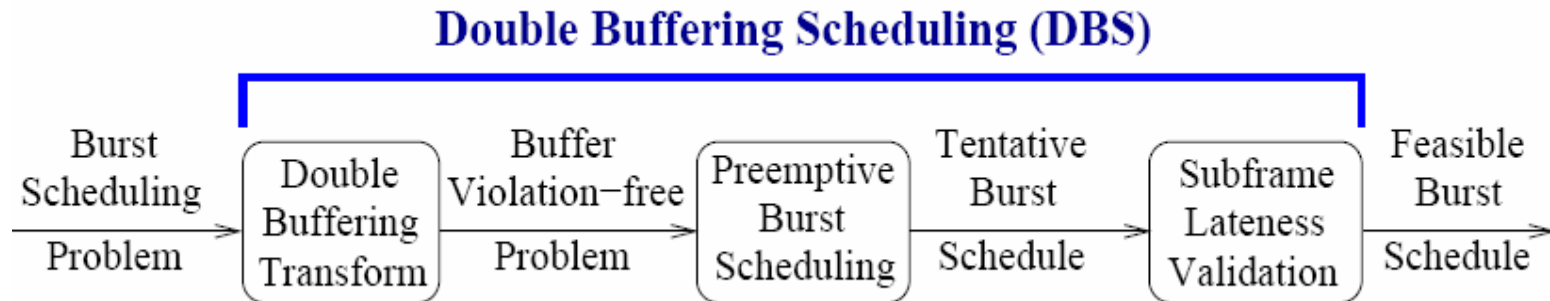
- Quality gap < 1 dB \rightarrow P2OPT is useful in practice

Burst Scheduling: General Problem

- TV channels can take any **arbitrary** bit rates
- **Observation: Hardness is due to tightly-coupled constraints: no burst collision & no buffer violation**
 - **→ could not use previous machine scheduling solutions, because they may produce buffer violations**
- **Our idea: decouple them!**

Burst Scheduling: General Problem

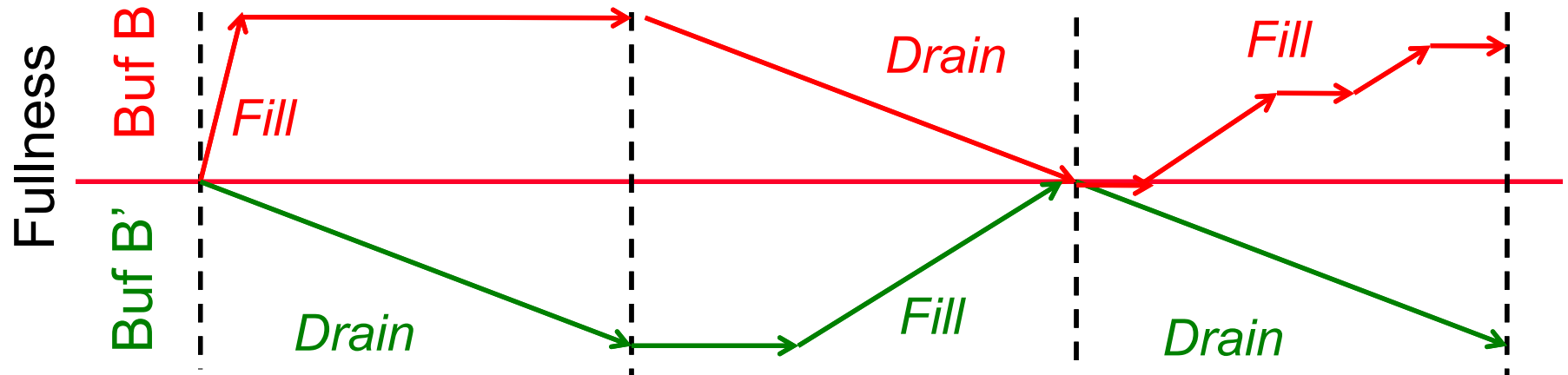
- **Idea of our algorithm:**
 - Transform problem to a **buffer violation-free** one
 - Solve it efficiently
 - Transform the solution back to the original problem
 - **Ensure** correctness and bound optimality gap in all steps



Double Buffering Scheduling (DBS): Overview

■ Transform idea:

- Divide receiver buffer into two: B and B'
- Divide each scheduling frame p into multiple subframes
- Drain B while filling B' and vice versa
- Schedule bursts so that bits consumed in current subframe = bits received in preceding subframe



DBS: Analysis

- **Theorem 4: Any feasible schedule for the buffer violation-free problem can be transformed to a valid schedule for the original problem.**

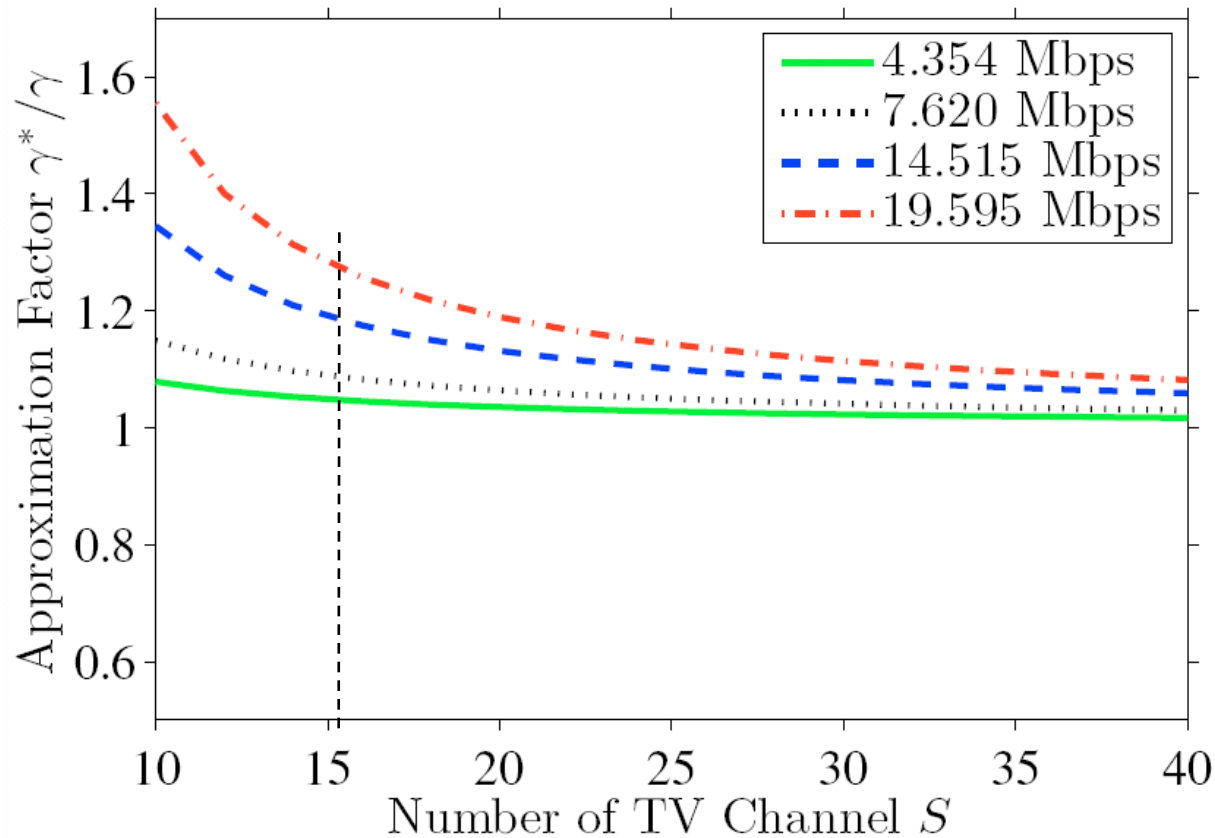
- Also a schedule will be found iff one exists.

- **Theorem 5: The approximation factor is:**

$$\frac{\gamma^*}{\gamma} \leq \frac{1 - T_o R / SQ - 1/S}{1 - 4T_o R / SQ - 1/S}$$

- **How good is this?**

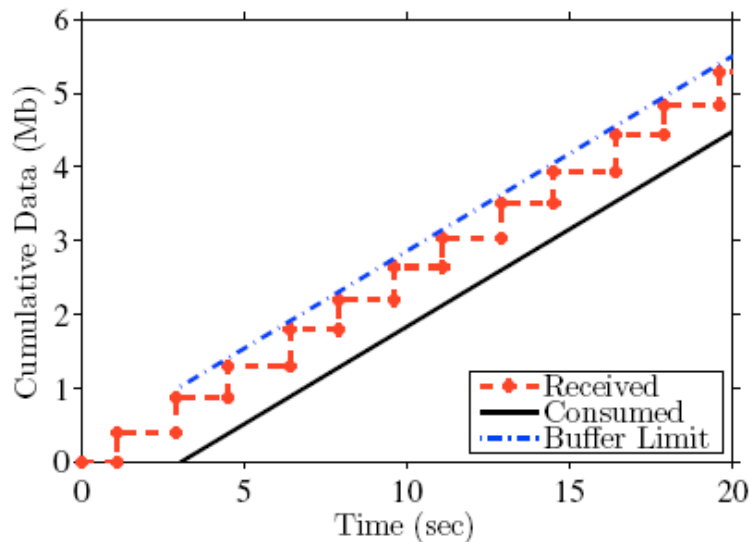
DBS: Analysis



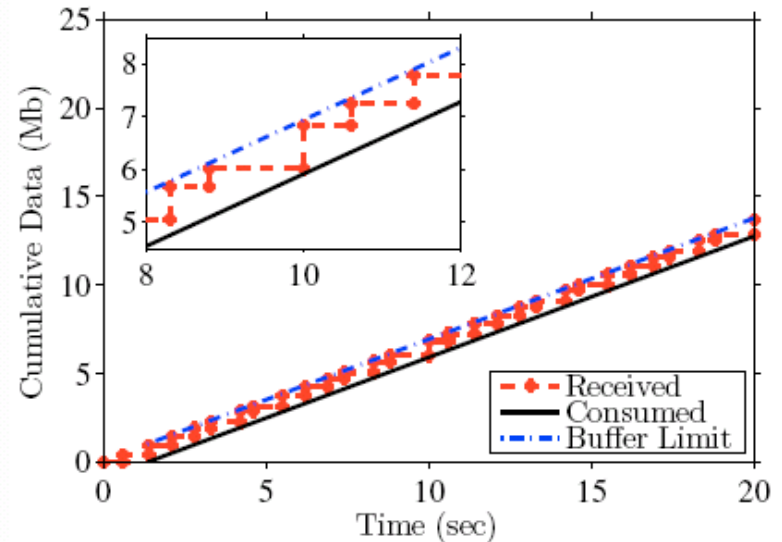
- **20 channels ($R = 7.62$ Mbps), energy saving by DBS is up to 5% less than the optimal**

DBS: Empirical Evaluation

- DBS is implemented in the mobile TV testbed



(a) Channel 1, 264 kbps



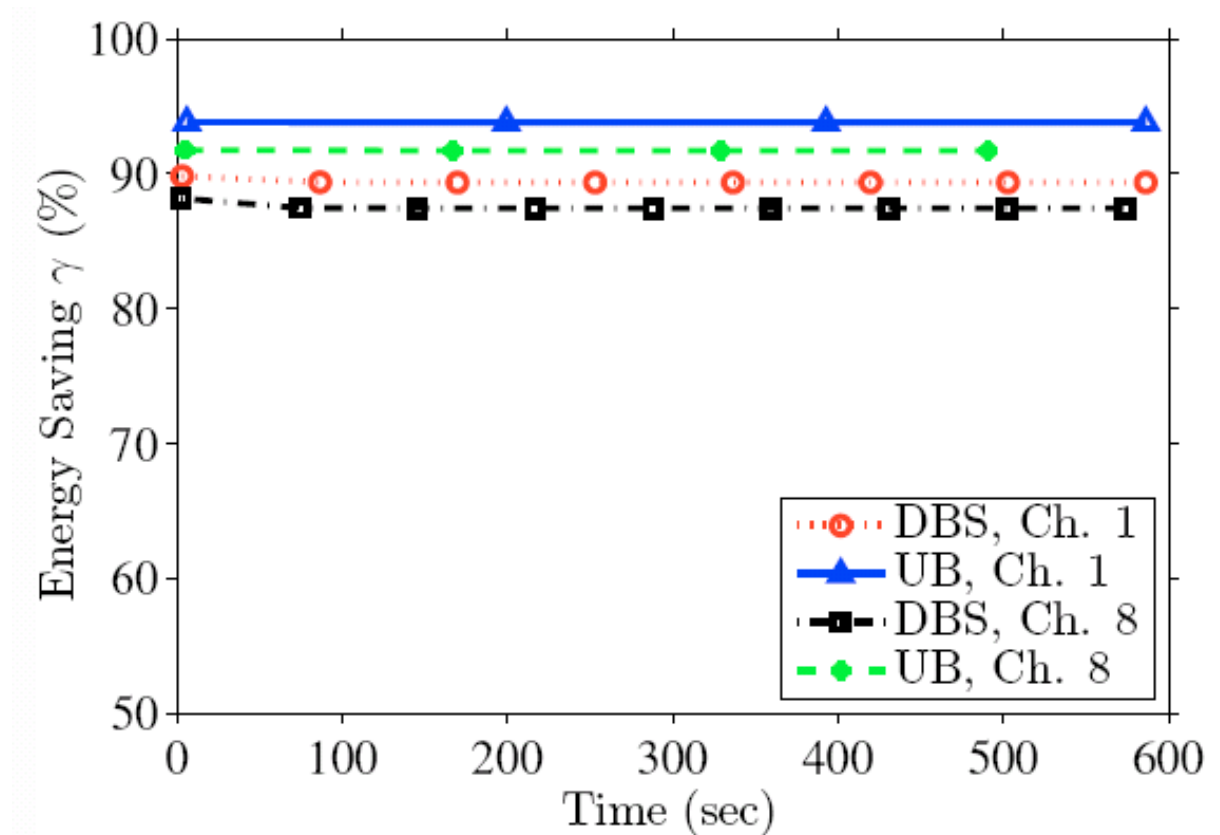
(b) Channel 4, 684 kbps

- No buffer violations
- Notice the buffer dynamics are different

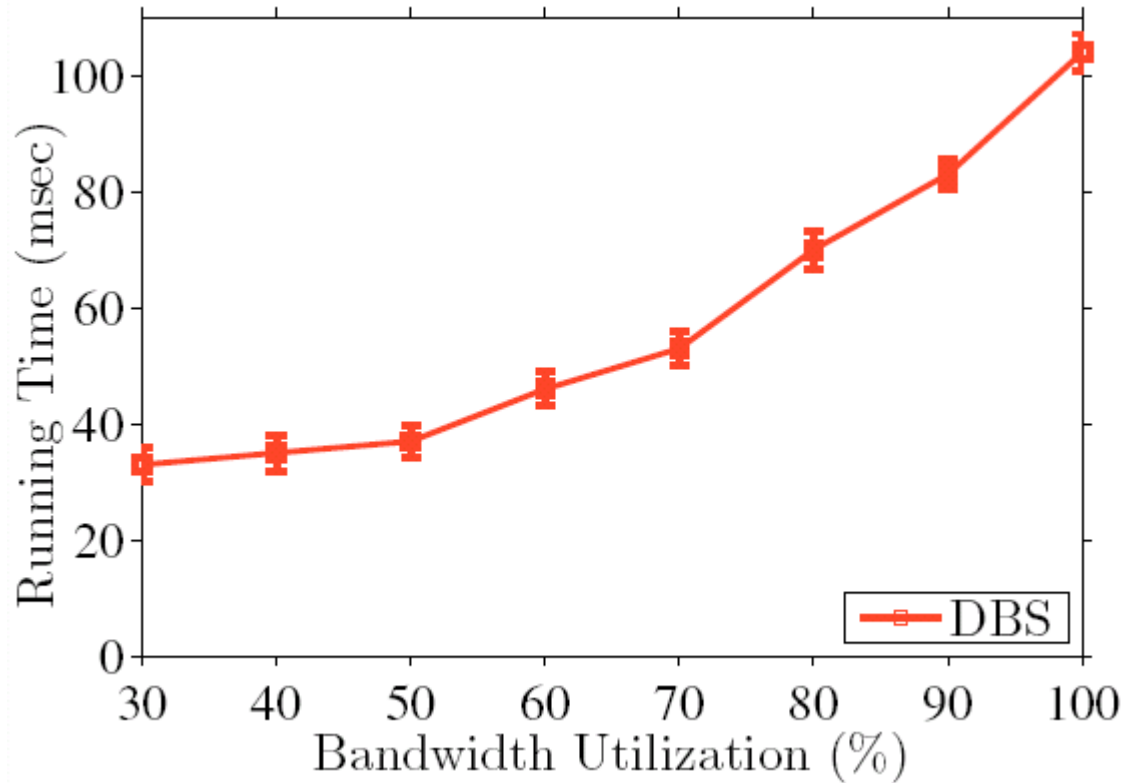
DBS: Near-Optimality

- Compare against a **very conservative upper bound**
 - Broadcast channels one by one

Gap < 7%



DBS: Efficiency



- **Running time <100msec on commodity PC for broadcasting channels saturating the medium**

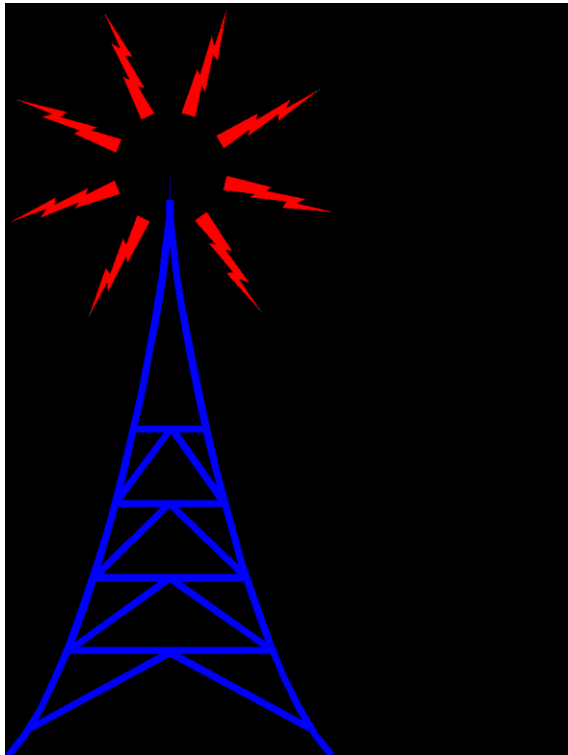
Energy Saving: Summary

- **Energy saving: critical problem for mobile multimedia**
- **Video streams should be encoded at different bit rates**
 - Better visual quality, higher bandwidth utilization
 - BUT make burst transmission scheduling NP-Complete
- **Proposed a practical simplification**
 - Classes of TV channels with power of 2 increments in rate
 - Optimal algorithm (P2OPT) and efficient
- **General Problem**
 - Near-optimal algorithm (DBS): approx factor close to 1 for typical cases
- **Implementation in real mobile TV testbed**

Video Streaming over Cooperative Wireless Networks

Mobile Video Broadcast

- Base station broadcasts multiple video streams to mobile devices over WMAN (Wireless Metropolitan Area Network)



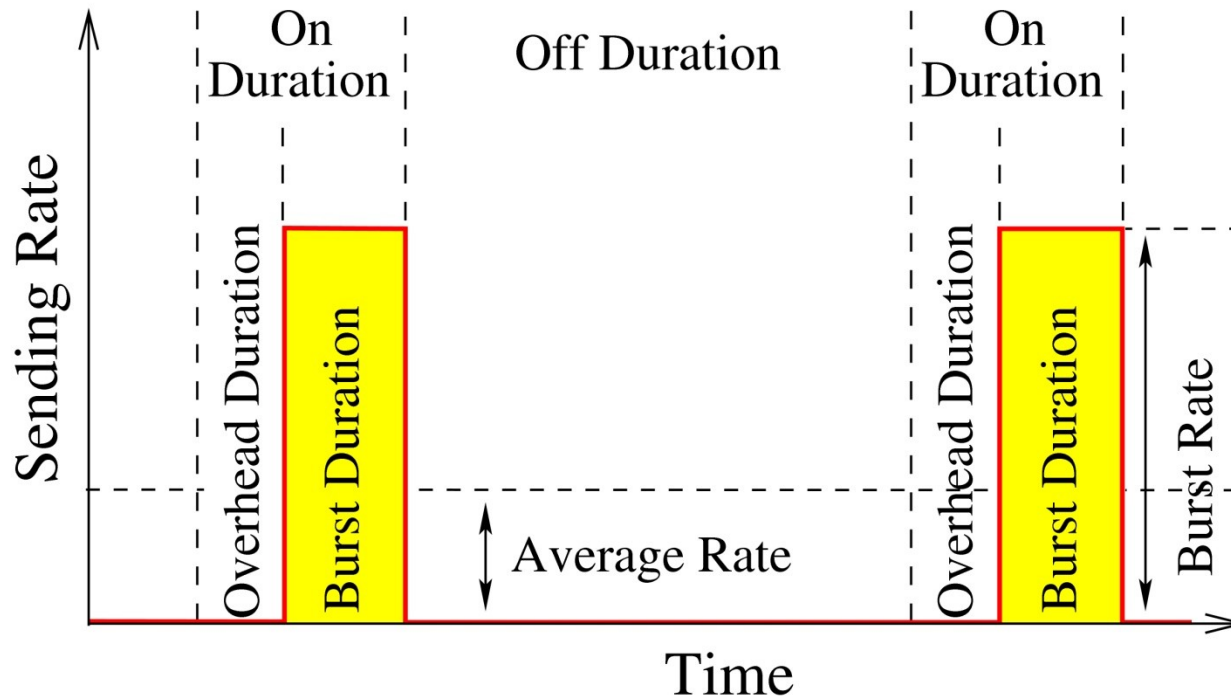
Base Station

WMAN



Energy Saving—Switching Delay Tradeoff

- Base station broadcasts in **bursts** to save energy
- → user has to wait for a burst
- **Tradeoff:** Saving energy introduces delay



Research Problems Considered

- **Energy consumption of mobile devices**
 - Battery powered
 - Video consumes substantial energy → short viewing time
- **Channel switching delay**
 - Delay until user starts viewing the stream
 - Important QoE parameter for users

Our Goal

- Saving **more** energy for mobile devices

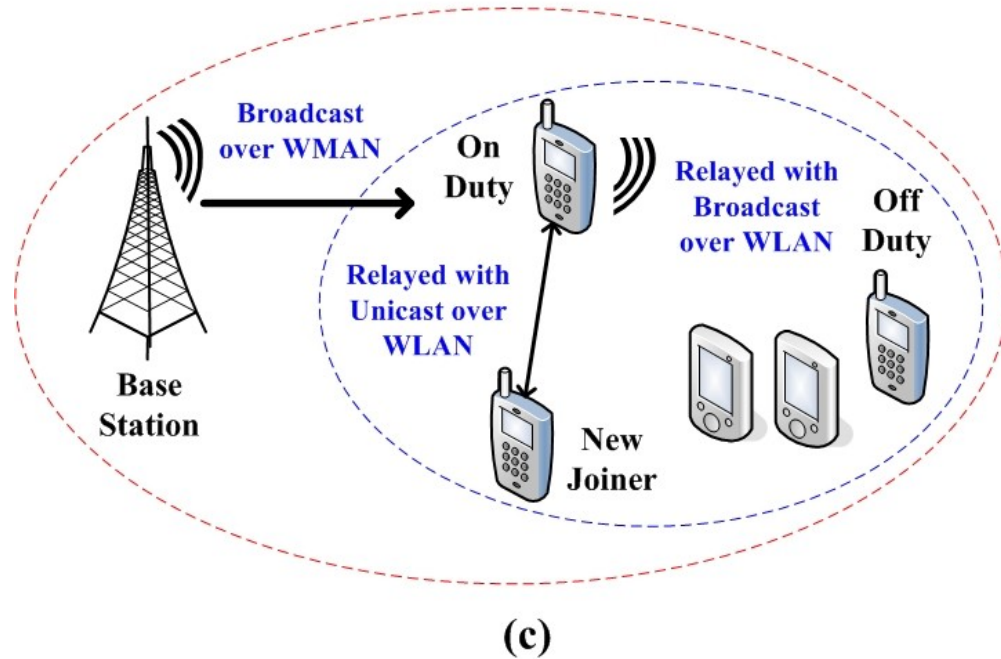
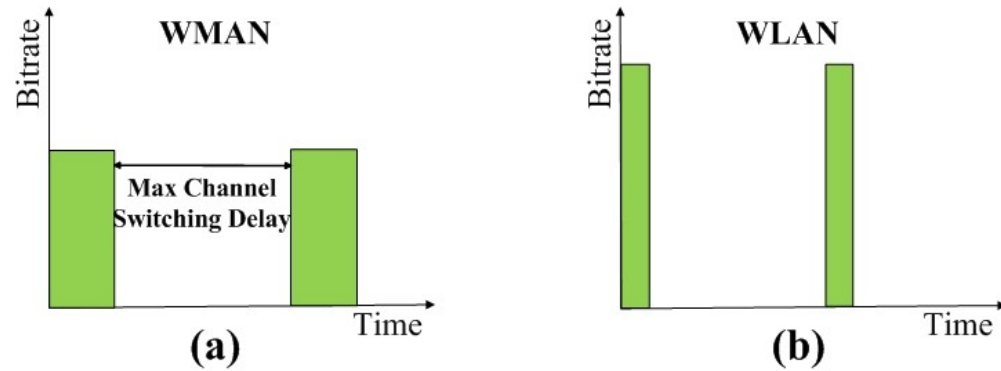
AND

- Reducing channel switching delay
- For more details, see [Liu 10, MMSys]

Our Approach

- Use **cooperation** among mobile devices (peers) to benefit all
 - Cooperation achieved over wireless LAN (WLAN)
- **Why?**
 - Energy per bit in WLAN is lower than in WMAN
 - Faster transmission in WLAN
 - WLANs widely deployed, most phones have them
 - Streams can be obtained quickly over WLAN → very short switching delay

Overview



Our Contributions

- **Distributed algorithm to elect devices and manage data transmission**
 - Simple, efficient, and motivates truthful cooperation
- **Quantitative analysis of the cooperative system**
 - To show the gain with different parameters
- **Implementation in real mobile TV testbed**
 - Proof of concept
- **Empirical results show:**
 - Substantial energy savings (up to 70% gain) AND
 - Switching delay almost eliminated (up to 98% reduction)

System Design: High-Level

- **Mobiles receiving same TV channel form group**
- **1 on-duty node is elected:**
 - **Receives data from base station over WMAN,**
 - **relays it to others in the group over WLAN,**
 - **and serves it immediately to new joiners**
 - **Broadcasts ON-DUTY messages**

 - **On-duty period is one WMAN burst cycle (few seconds)**

System Design: High-Level

- **K backup nodes are elected:**
 - Each has a different timer
 - If on-duty node fails, one will become on-duty
 - Receive data from WMAN, and store it
- **N-k-1 nodes are off-duty**
 - Received data from on-duty node over WLAN
 - WMAN interface is off

System Design: High-Level

■ Election:

- Nodes maintain Contribution list with N entries
- Entry n is total amount of data relayed by node n
- Node computes other nodes' contributions based on *actual* data received
- Node with least contribution becomes on-duty
- Next k nodes become backup
- In case of tie, node with oldest joining time is chosen

Handling Network Dynamics

■ Device Join:

- A join message sent to the on-duty device. On-duty replies with burst data and contribution list

■ Device Leave:

- If backup or off-duty device leave: **LEAVE** message sent to on-duty device
- If on-duty device leave: **LEAVE** message broadcast to the group, one backup device takes over the on-duty role

■ Device Failure:

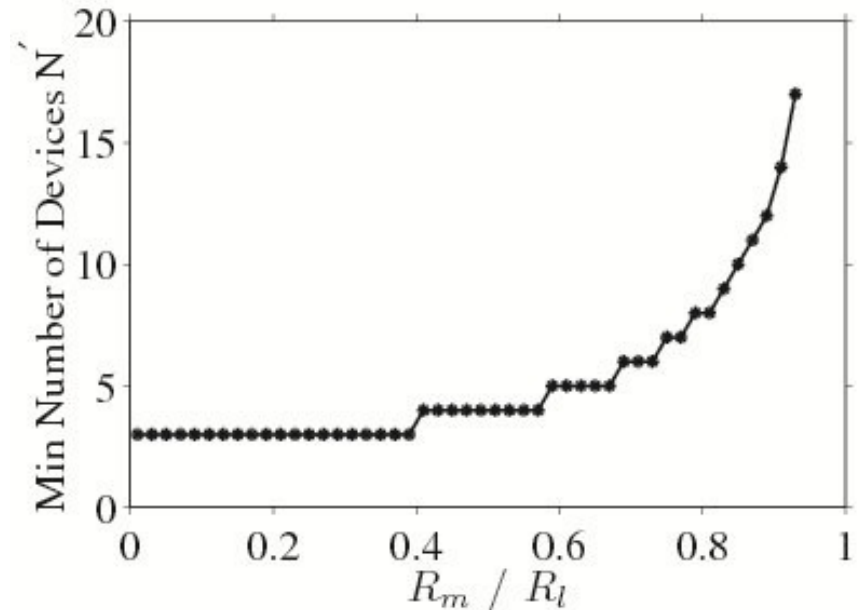
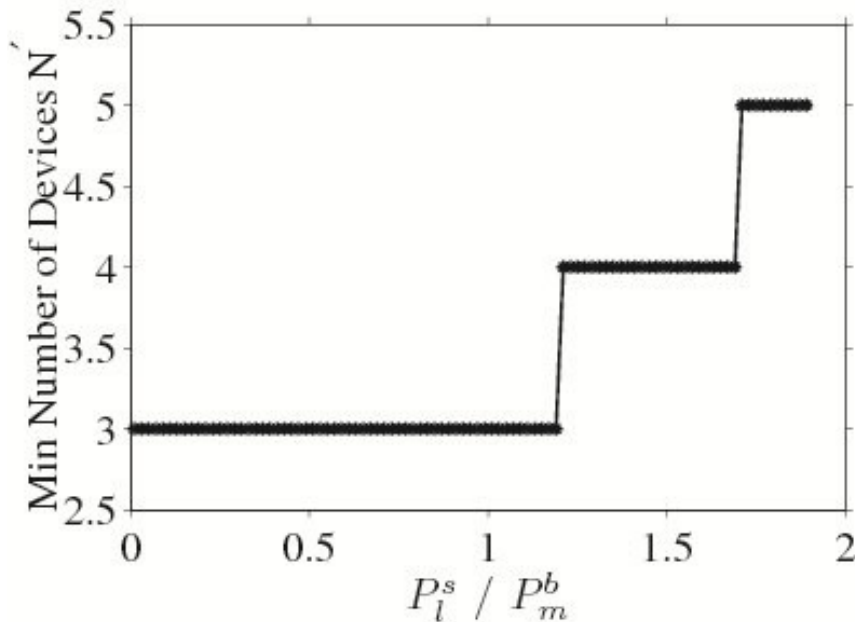
- If backup or off-duty device fail: No harm, will be detected in next cycle
- If on-duty device fail: No more **ON-DUTY** message broadcast, can be detected by backup devices, then one backup device takes the on-duty role

Time Synchronization

- **Time offset contained in the header of burst data packets**
- **No extra clock synchronization algorithm needed**

Analysis Results

- Compute energy saving gain,
- and number of needed nodes for cooperation
- As function of energy consumption values of WMAN and WLAN and their transmission rates



Evaluation in Mobile TV (DVB-H) Testbed

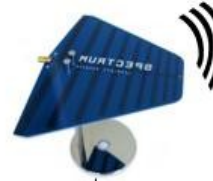
Video/Audio Streaming Application	Electronic Service Guide	File Transfer Application
RTP	FLUTE	
UDP		
IP		
IP Encapsulator		
PSI/SI	MPE-FEC	Time Slicer
DVB-T		

Linux Workstation
Runs Streaming Server
and IP Encapsulator



Base Station

In-door Antenna



Low-power Amplifier



PCI Card
DTA-110T-SP
DVB-H Modulator



Nokia N92

Nokia N96



Wireless Router
(Optional)

Hauppauge WinTV Receiver



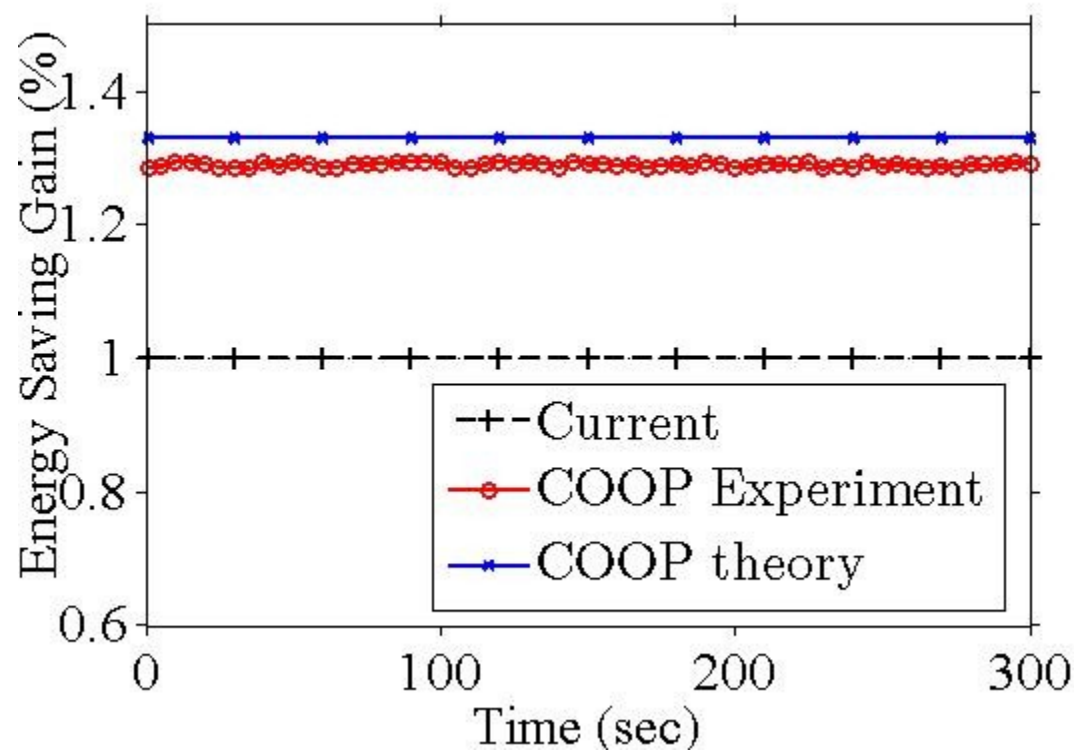
Receivers

Experimental Setup

- We implemented our algorithm in PC with USB DVB-H receivers (4 in total)
- We setup an 8 MHz radio channel to broadcast four 5-minute long TV programs coded at 250 kbps.
- We used the QPSK modulation scheme together with the convolution coding rate at $2/3$ and guard interval at $1/8$.

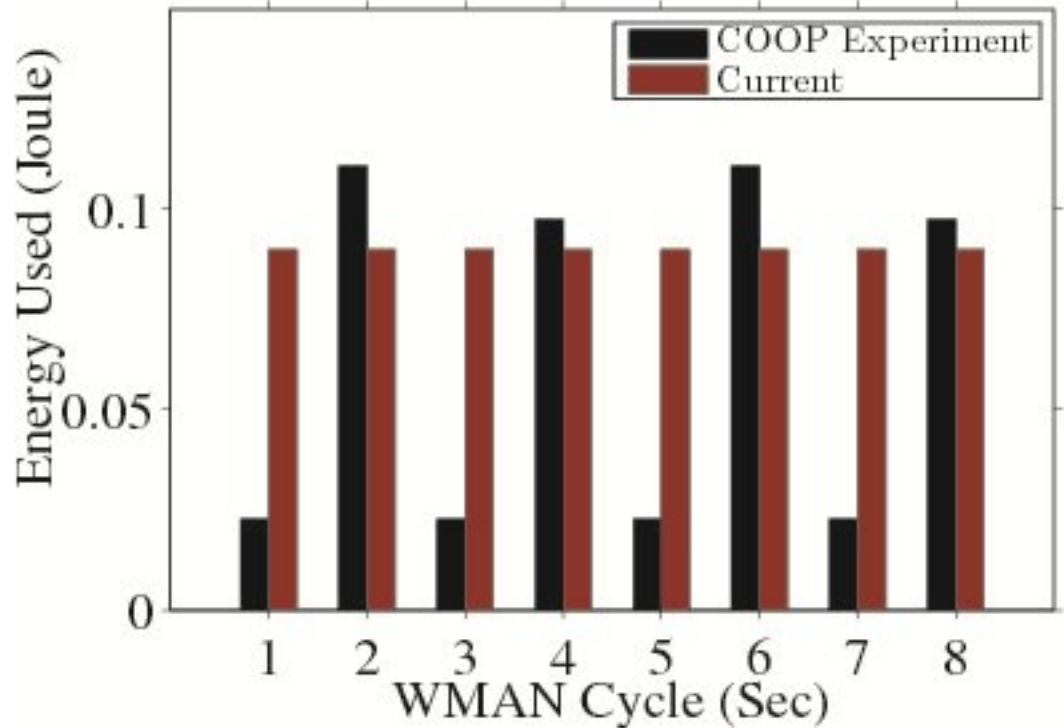
Energy Saving Gain

- In theory, saving about 33%
- In experiment, saving about 29%



Energy consumption of one mobile device

- On-duty mode
spend 23% more
- Backup mode
spend 8% more
- Off-duty mode
save 75%
- One device takes
turns to be in
different mode

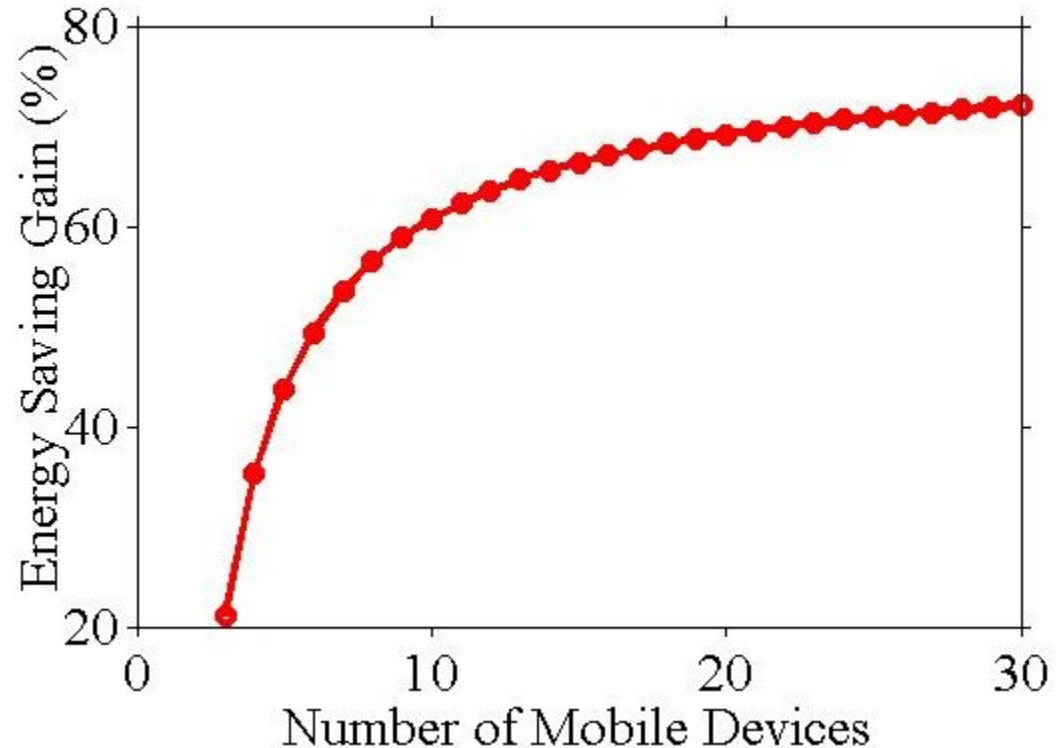


Evaluation in Simulation

- **Trace-driven simulation**
 - **Increase number of nodes, exercise wide range of parameters**
- **Used actual MPEG-TS transport streams (obtained from Nokia)**
- **Used actual power consumption values from chip data sheets**

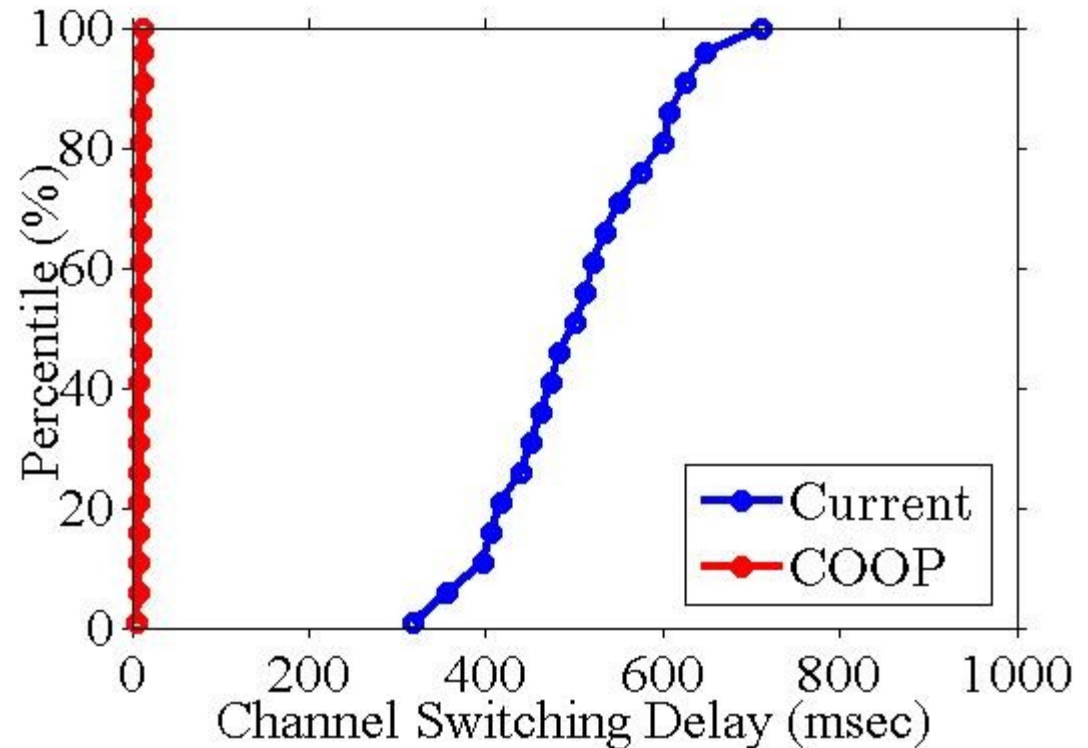
Potential energy saving gain in simulator

- **Only 3 devices needed to outperform current systems**
- **Saving up to 70% with 30 devices**
- **Saving about 21% with 3 devices**



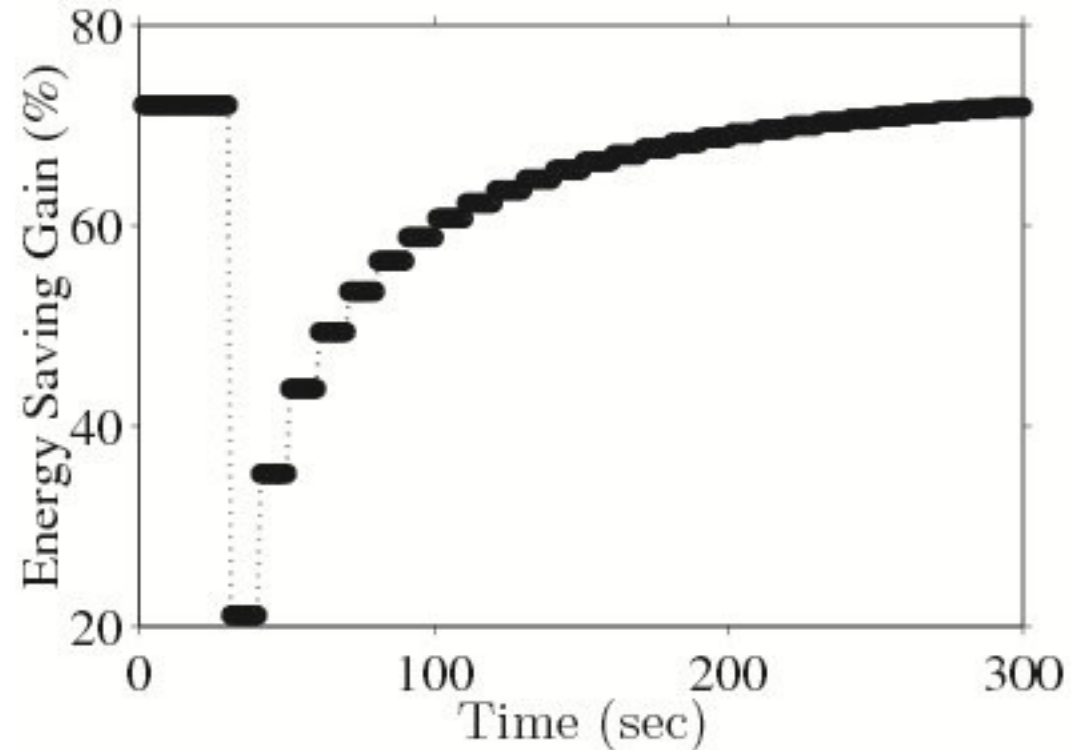
Channel Switching Delay

- Reduce channel switching delay by up to 98%
- From up to 700 msec to at most 13 msec



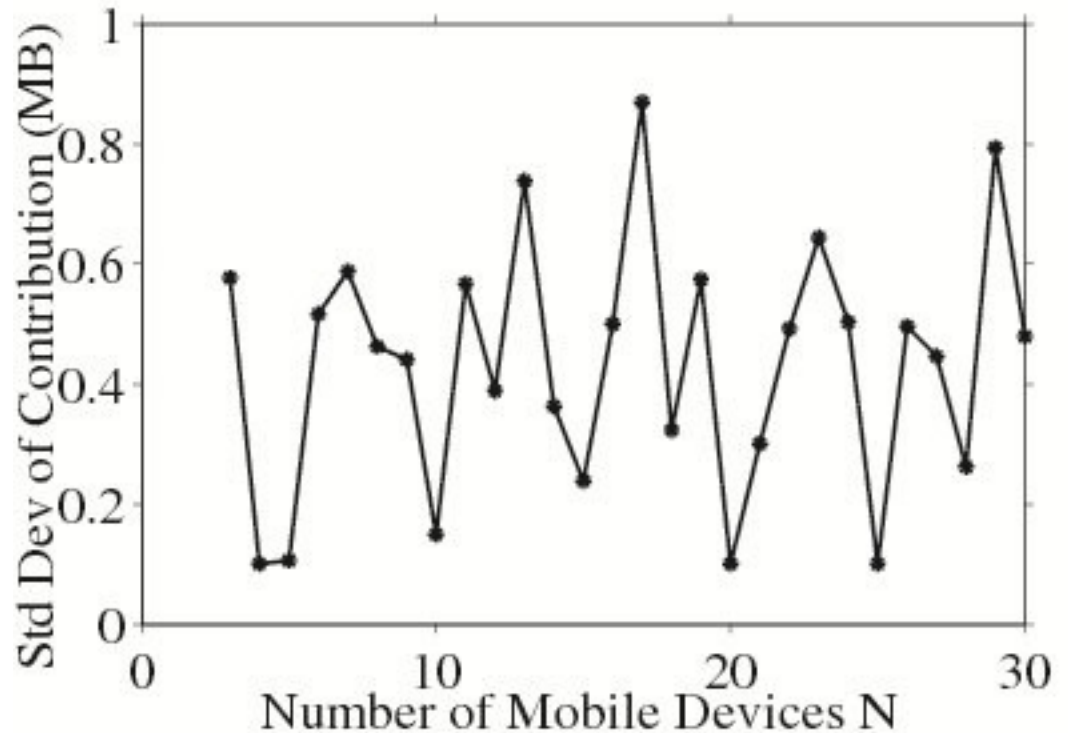
Energy Saving Gain under Network Dynamics

- Survives a sudden loss of 90% devices
- Energy hit of 51% under 90% failure
- Quickly adapts to network dynamics



Load Distribution

- Standard deviation of contribution is less than 0.6 MB
- Total contribution value in the order of hundred MBs

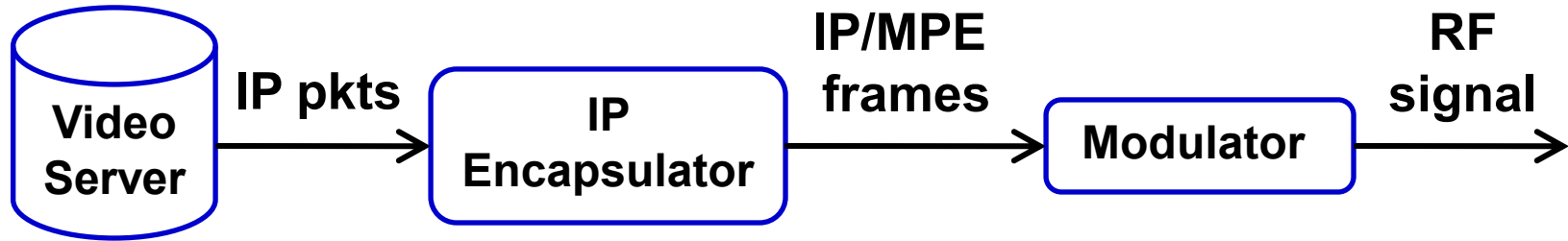


Cooperative Streaming: Summary

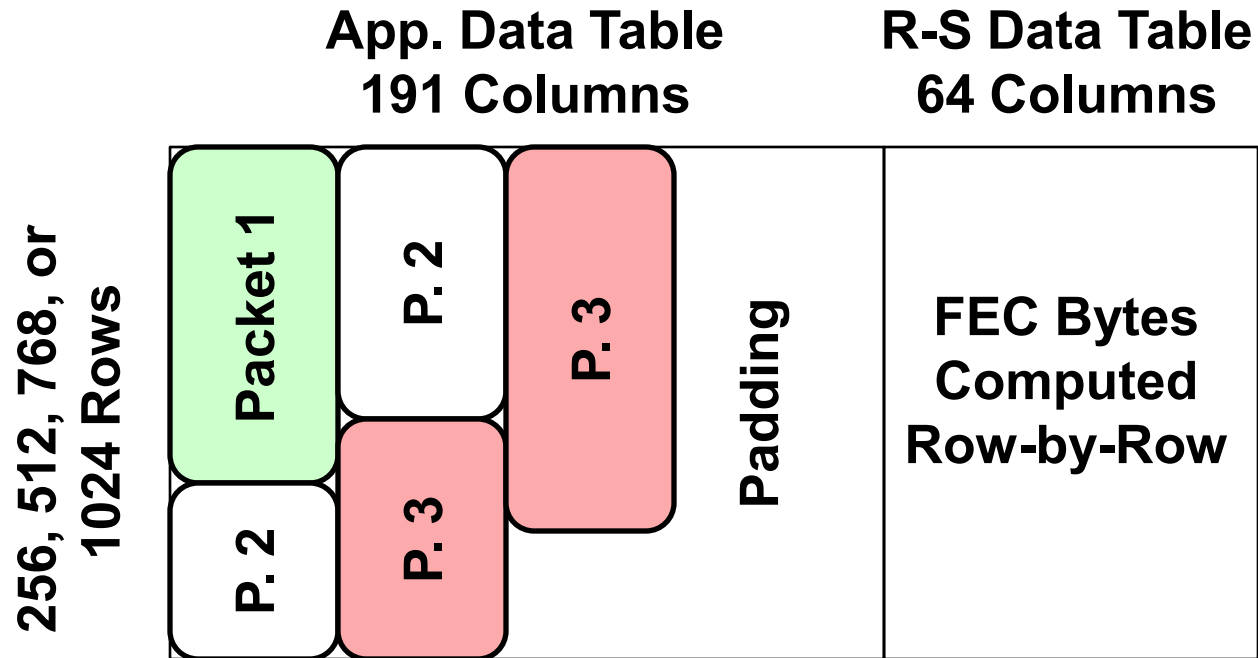
- **Proposed video streaming over cooperative WMAN and WLAN networks**
- **Real implementation, simulation, and analytic analysis show that the proposed system improves energy saving and switching delay **concurrently****

Supporting Heterogeneous Mobile Receivers

Mobile TV Base Station and Frame Format



MPE Frame



Problem Statement

- Design a burst transmission scheme to support **heterogeneous** mobile devices, such that
 - Each mobile device has an energy consumption proportional to its received video quality
- But, how to support heterogeneous devices?

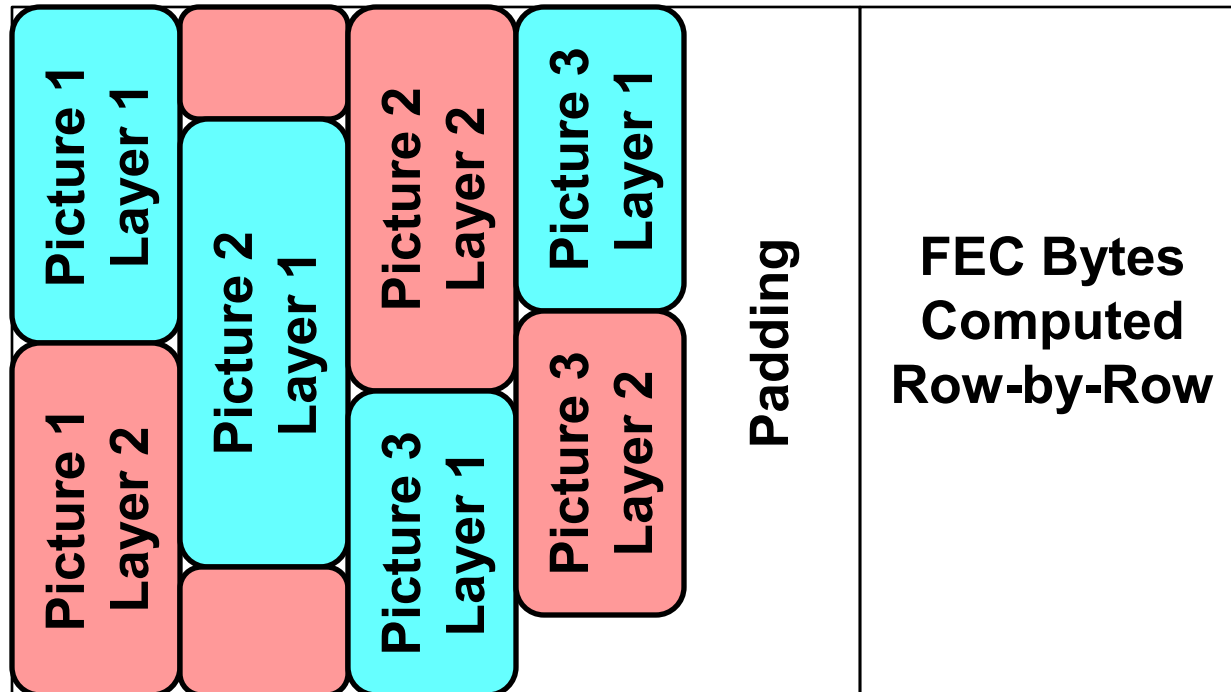


Support Heterogeneous Devices

- **Multi-version: encode each video into several versions, and concurrently broadcast them**
 - Simple, but not efficient (in terms of bw)
 - Spectrum is expensive

- **Multi-layer: encode each video into a scalable stream consists of several layers, and broadcast each layer only once**
 - Bandwidth efficient, **but challenging** in mobile TV broadcast networks

An Illustrative Example: Challenge



OK for devices with high-resolution displays

BAD for devices with low-resolution displays → Higher energy consumption

Our Solution

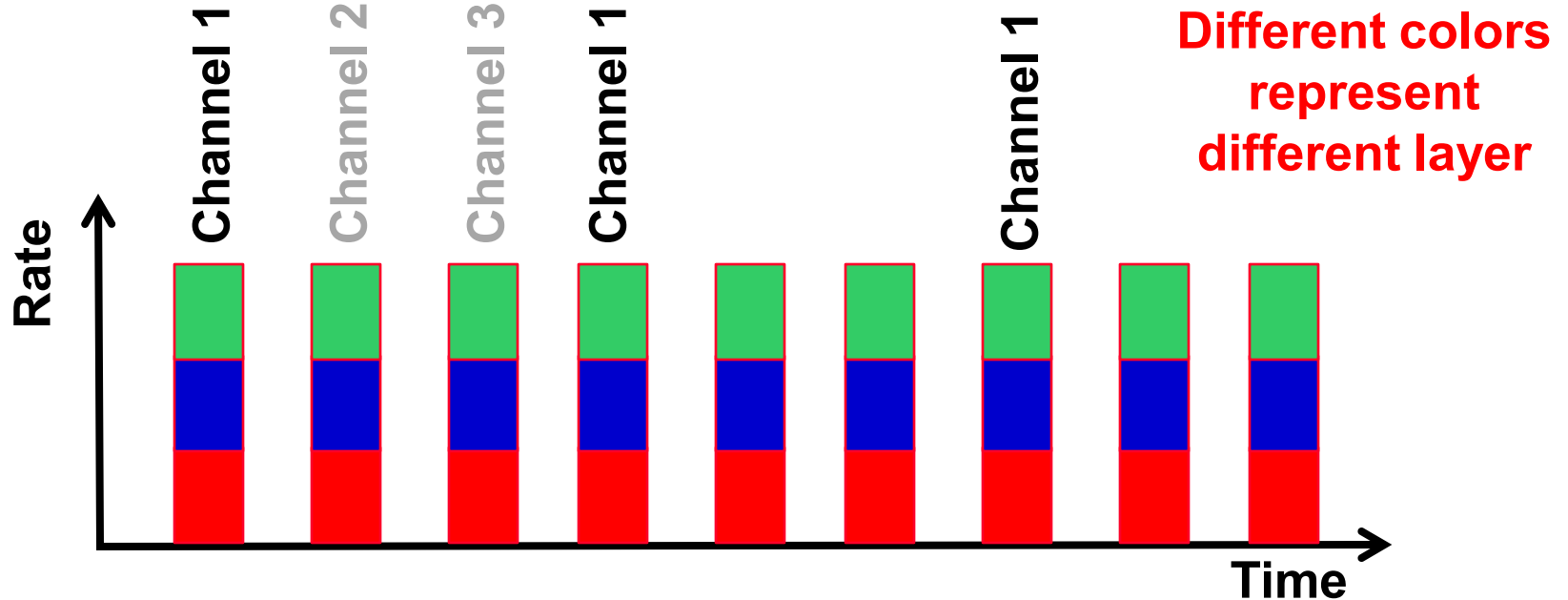
□ Layer-aware burst transmission

- Base station properly **organizes** the video data while broadcasting
- Mobile devices with low resolution displays can **skip** irrelevant data

□ We consider three approaches

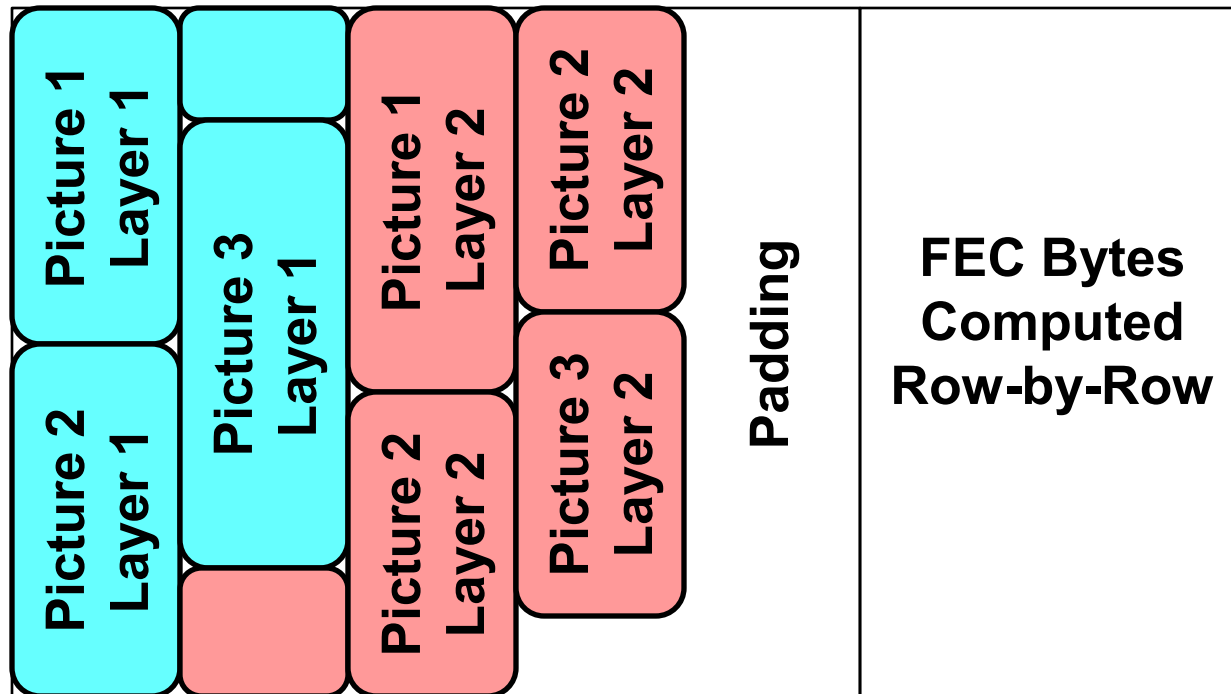
- Parallel service
- Layer-aware FEC
- Layer-aware time slicing

Parallel Service: PS



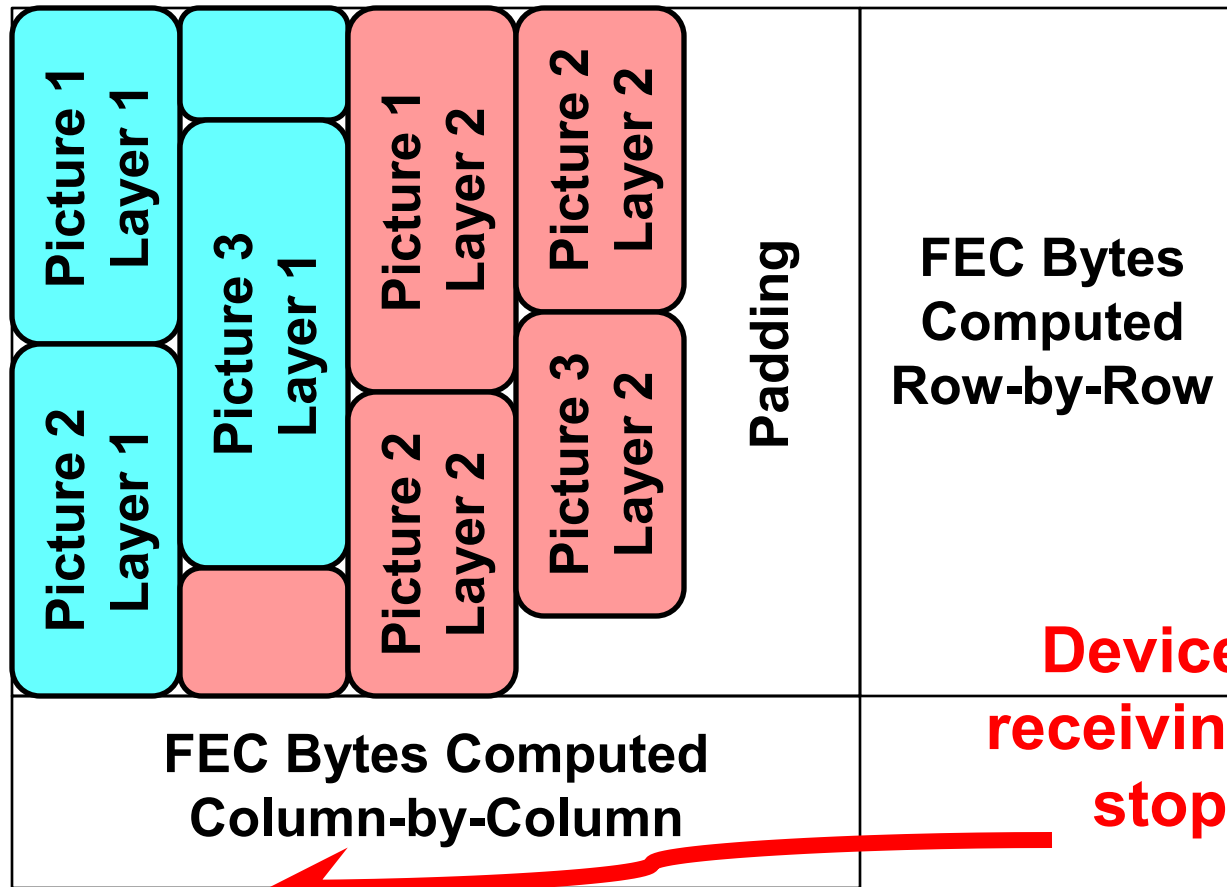
- Use Dest. IP addr. to help demultiplexing at receiver ← lower processing overhead
- **But**, all devices receive **complete** bursts
 - Energy is wasted

Layer-Aware FEC: LAF



- Mobile devices can turn off RF circuits earlier if we **rearrange** packets in MPE frames
- **But**, devices still need to wait for FEC bytes!

Layer-Aware FEC: LAF (cont.)



- Compute the FEC bytes column-by-column
 - Layer-Aware FEC frame

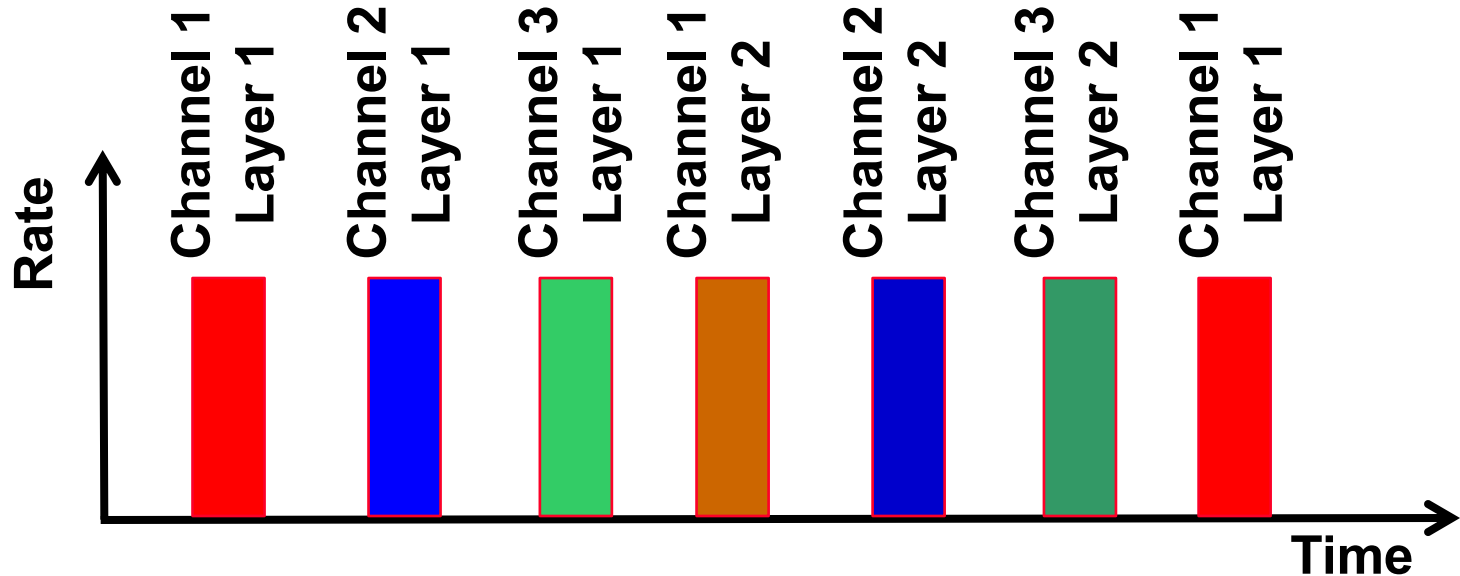
Layer-Aware FEC: LAF (cont.)

□ Limitations of LAF

- Requires modifications on MPE frame format (not standard compliant)
- May be sensitive to bursty errors in broadcast networks, because LAF does not perform time interleaving

□ Can we do better?

Layer Aware Time Slicing Scheme (LATS)



- Burst time for layer c channel s is:

$$(b/R)[(c-1)S + (s-1)],$$

where b is burst size, R is network bandwidth

Performance of LATS

Lemma 1: Devices that receive c layers achieve energy saving

- Smaller c values result in **higher** energy saving

$$1 - \frac{c}{CS} - \frac{RT_o c}{bCS}$$

Lemma 2: LATS scheme achieves higher energy saving than LAF scheme if $c \neq C$. These two schemes lead to the same energy saving if $c = C$.

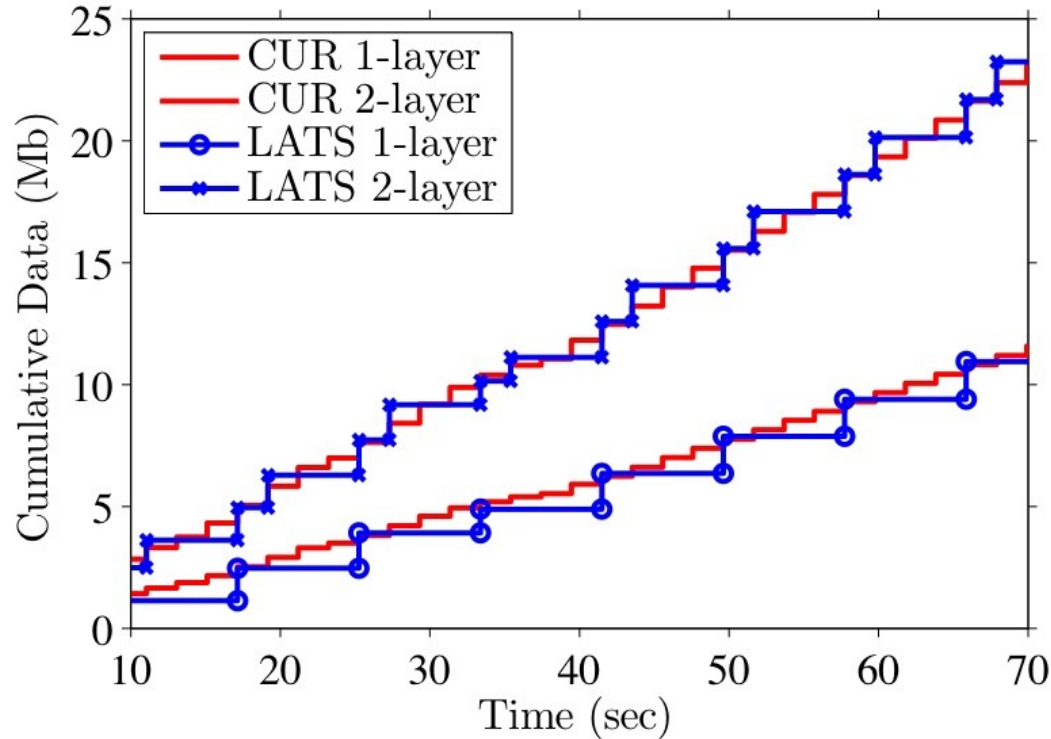
Comparison: the Three Approaches

- **Parallel Service does not achieve diverse energy saving**
- **Layer-Aware FEC achieves diverse energy saving, but has some limitations**
 - **modification of standards (frame format)**
 - **could be sensitive to bursty channel errors (no interleaving)**
 - **lower energy saving than Layer-Aware Time Slicing scheme**
- **We consider LATs in the rest of this paper**

Empirical Evaluation

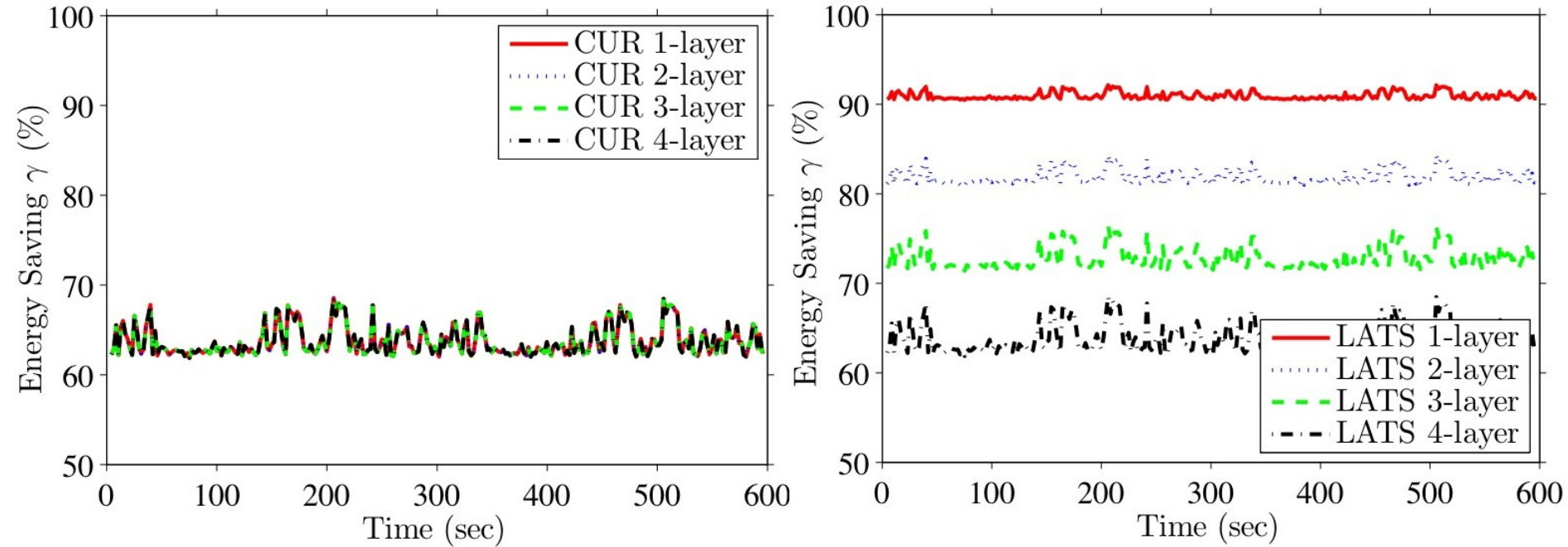
- **Implement LATS in a real base station**
- **Encode videos into SVC streams with 4 layers, where each layer has a bit rate of 192 kbps**
- **Broadcast 8 channels for 10 minutes over a network with 8.289 Mbps bandwidth**
- **Collect detailed burst logs that specify**
 - **Start time and burst size of every burst**
- **Report energy saving based on the burst logs**

Cumulative Received Data



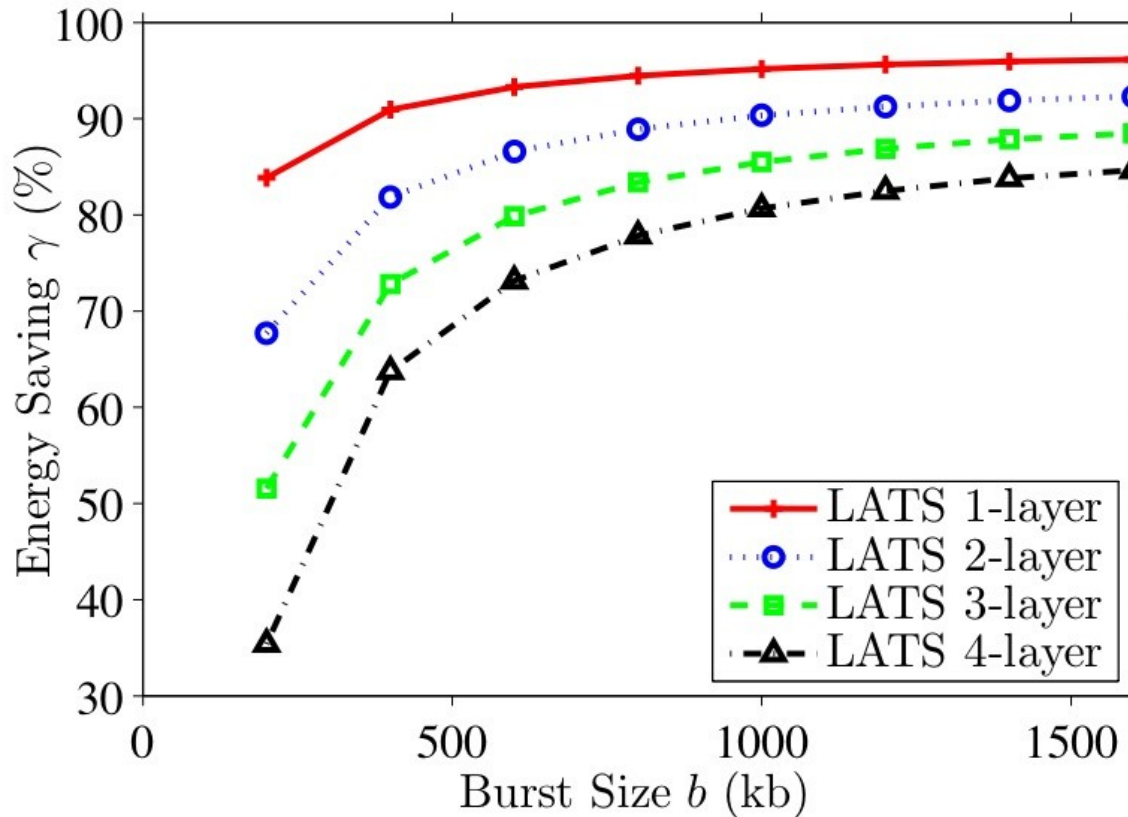
- Both current and LATS receive the **same amount of useful (un-dropped) data**
- LATS allows devices to turn on RF circuits less often

Energy Saving



□ **LATS: diverse energy saving: up to 92%**

Implication of Burst Size b



□ LATS enables a wide range of energy saving

Heterogeneous Receivers: Summary

- **Discussed transmission schemes to efficiently support heterogeneous receivers**
- **Analytically compared different schemes: LATS scheme outperforms others in energy saving**
- **Actual implementation confirms that LATS scheme achieves energy saving (and thus quality) differentiation**

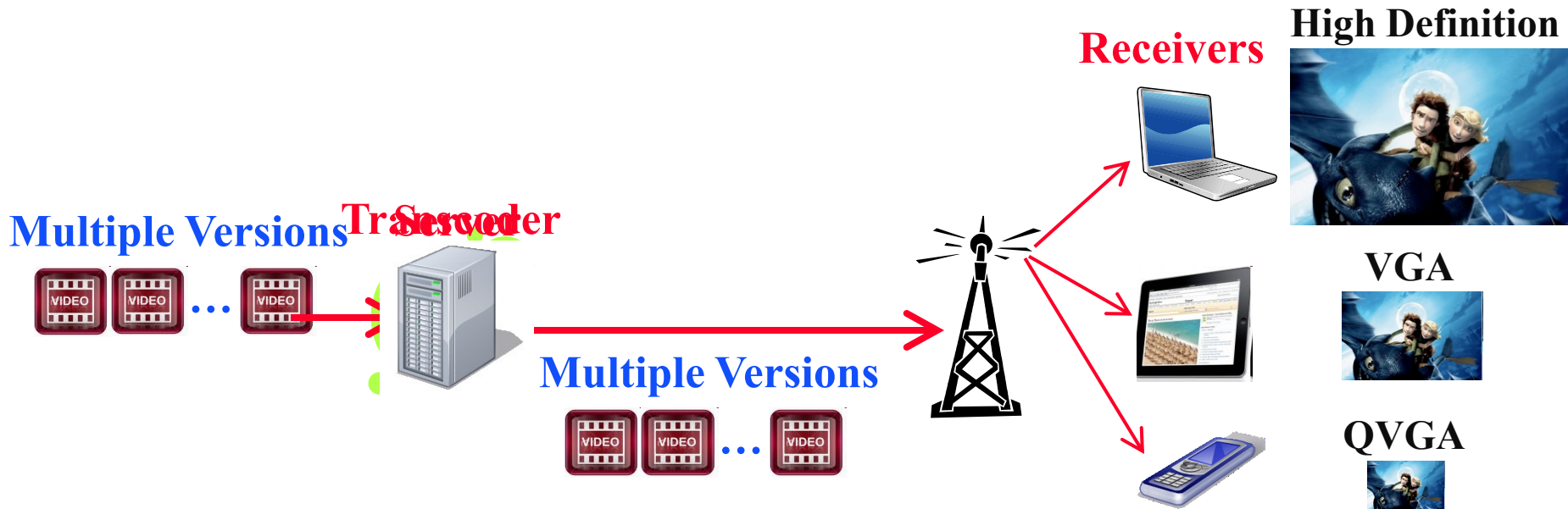


Part III

Mobile Multimedia Unicast

System Model

- Streaming server transmits a customized video stream for each mobile receiver ← different bit rates
- Usually, only a few versions are pre-encoded
- Transcoder (or scalable streams) can be used to dynamically generate the most suitable version of stream



Video Compression and Transport

Compression

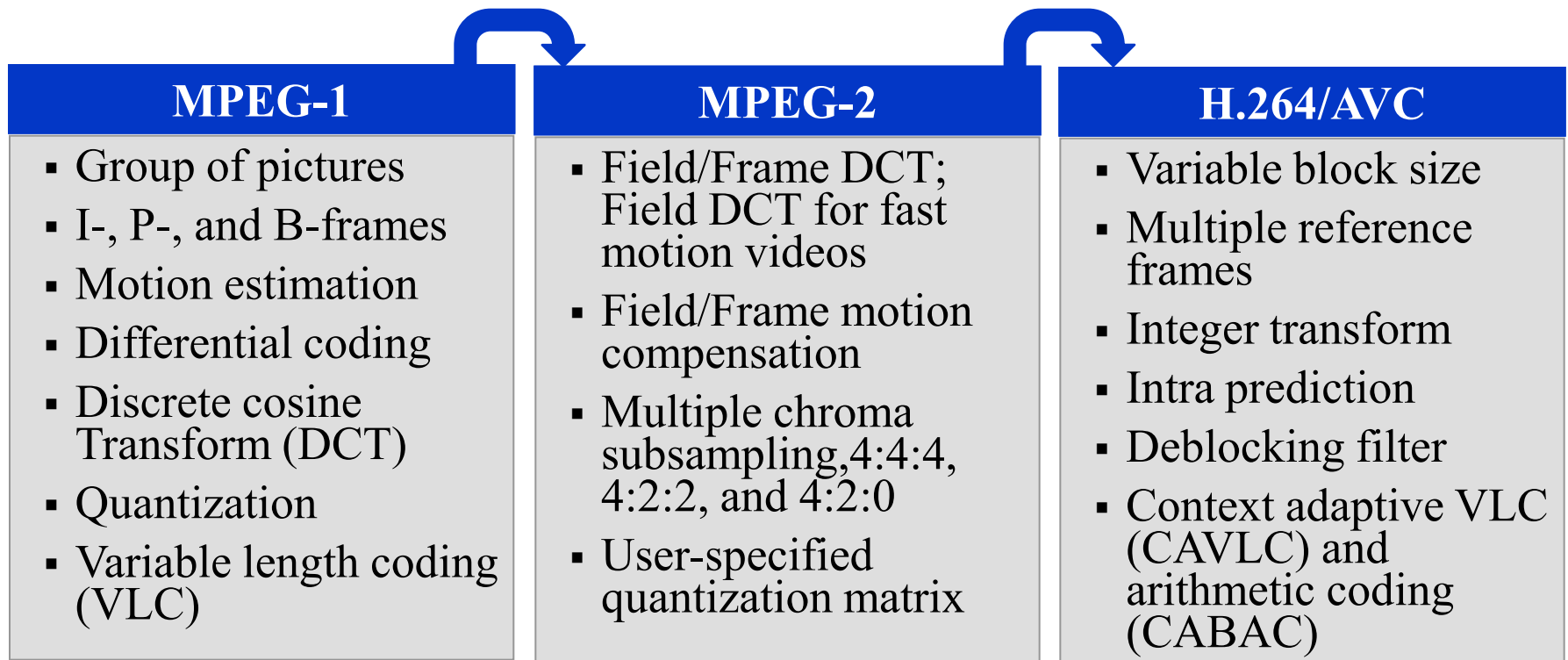
- **Sending raw videos consumes a lot of network bandwidth**
 - In the order of MB/sec
- **Data compression can be categorized into**
 - **Lossless compression: allows perfect data reconstructions, such as zip and rar**
 - **Lossy compression: drops less critical information for higher compression ratio, often used for multimedia data, because **human perception is not linear****



Video Compression

■ Most popular video coders use

- Transform coding to exploit spatial redundancy; similar to image compression
- Motion compensation to exploit temporal redundancy, caused by object or camera movements



Google's Open-source Video Format

- **WebM is based on**
 - **VP8 Video codec from On2; ON2's VP6 is a popular proprietary codec due to the widespread of Adobe flash**
 - **Vorbis audio codec**
 - **Martroska contain format**
- **Why open-source?**
 - **MPEG royalty fee can be as high as a few million dollars for larger commercial deployments**
 - **MPEG is royalty-free for free content until 2015**
- **(Will be) supported by Firefox, Opera, Google Chrome, Internet Explorer 9, Safari, Epiphany, Adobe Flash Player, VLC, Miro, Moovida, Winamp, MPlayer, Ffmpeg, and Android.**

Performance Comparison: VP8 vs. H.264

- Experiments ^[WebM] show that
 - VP8 has a longer encoding/decoding time, due to lack of H/W accelerations on most platforms
 - Object tests using PSNR and SSIM reveal that H.264 outperforms VP8 by 20-30%
 - Subject tests show H.264 is slightly better than VP8 with a small edge (720p video at 800 kbps)



Video Transports

- **Different methods to deliver compressed videos**
 - **Download and play.** Still used for high-quality content that cannot be sent in real-time, e.g., DVD Rips from P2P networks.
 - **Streaming.** Video files are divided into small packets and sent by specialized protocols, typically RTSP and RTP.
 - **Progressive download.** Use bulky data transfer protocol, typically HTTP, and start playing while files are still being transferred. Getting popular recently partly due to YouTube (and flash streaming in general). Main strength is simplicity. **Main weakness is lack of adaptation.**
 - **HTTP adaptive streaming.** Adjust video rate to accommodate network dynamics but remain **simple**. Used in proprietary systems such as MS Smooth Streaming, Adobe, Move Networks, and Swarmcast ^[PPSN10]. Standardization activity is ongoing: **Dynamic Adaptive Streaming over HTTP (DASH).**

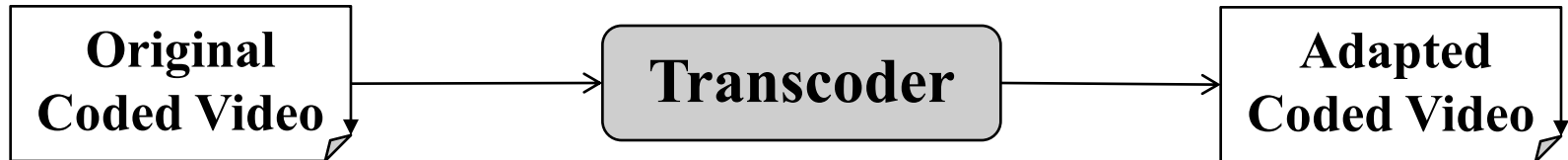
Streaming vs. Progressive Download

	Streaming	Progressive Download
Network Dynamics	Fast rate adaptation, and thus better quality	Restart the playout with another version of the video
Startup Time	Shorter	Longer
Seek and Navigations	Generally faster	Slower unless already downloaded
Content Control	Much harder to copy	Vulnerable to illegal copies
Live Video	Support	Do not support
Complexity	Hard to setup/implement	Easier, merely a Web server
Cost	Additional hosting cost	Web hosting is less expensive

We focus on streaming in the tutorial for better perceived quality, but our techniques can be applied to HTTP adaptive streaming

Video Transcoding for Timely Delivery

Video Transcoding



- **Generate a different video stream with different**
 - bitrates
 - coding standards
 - features, such as supporting error-resilience techniques
 - contents, adding watermarks

- **Focus on rate adaption**
 - allow video streams to be delivered on time over bandwidth-limited channels
 - can be achieved by changing frame rate, resolution, or quality

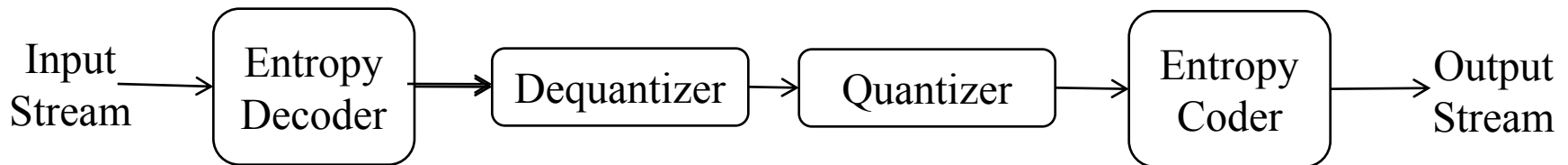
Open-loop Transcoders

□ Transcoders can be classified into

- open-loop
- pixel-domain
- transform-domain

□ Open-loop transcoders

- drop high-frequency coefficients, or
- perform requantization
- drawback: severe drifting due to reference picture mismatch



Pixel-Domain Transcoders

□ (Cascaded) pixel-domain transcoders

- concatenate a decoder and an encoder
- cause no drifting error
- flexible, various encoding parameters can be adjusted at the encoder
- simplified encode can be used, for example, motion vector in the input stream can be reused in the output stream



Transform-Domain Transcoders

- **Cascaded transform-domain transcoders**
 - similar to cascaded pixel-domain transcoders, but the motion compensation is done in transform-domain
 - support spatial/temporal down-sampling

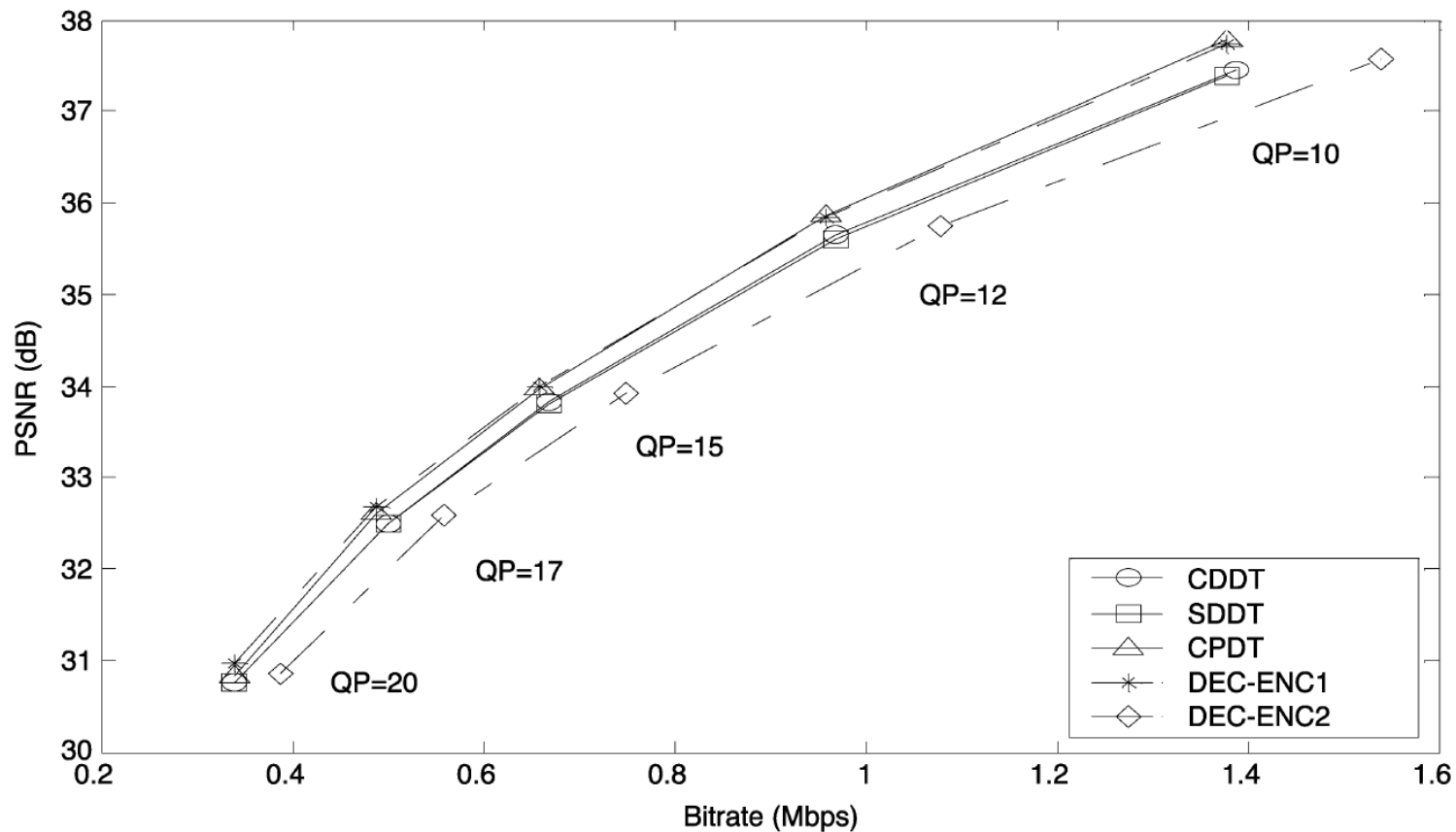
- **Simplified transform-domain transcoders**
 - more efficient because the frame buffers at the encoder are eliminated under the linear assumption
 - doesn't support down-sampling
 - lead to drifting because of the errors caused by the linear assumption

Running Time [Xin et al. '05]

	Foreman (CIF) no B-frames	Foreman (CIF) B-frames	Mobile (CIF) no B-frames	Mobile (CIF) B-frames
Cascaded Pixel Domain	7.0 fps	7.1 fps	6.6 fps	6.7 fps
Cascaded Transform Domain	14.1 fps	9.7 fps	15.7 fps	10.7 fps
Simplified Transform Domain	23.2 fps	17.1 fps	23.0 fps	17.5 fps

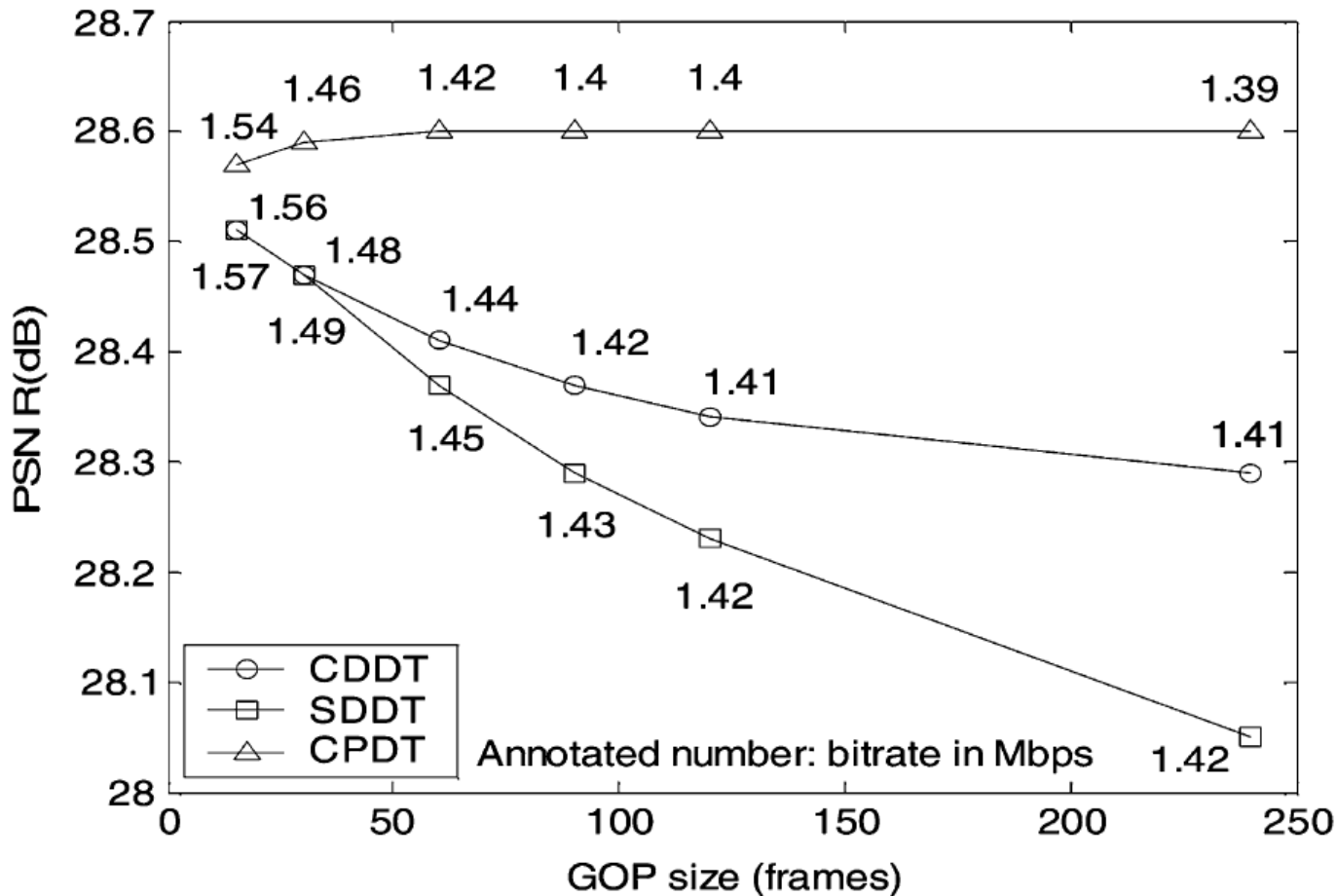
- **Transform domain transcoders are more efficient, 2X at least**
- **Simplified transform domain further double the speed**

Coding Efficiency [xin et al. '05]



- In general, pixel-domain transcoders outperform transform-domain transcoders by up to 0.5 dB
- Foreman sequence

Drifting Error [xin et al. '05]



- **Drifting error of transform-domain transcoders is < 0.1 dB when GoP size is small**
- **Mobile sequence**

Discussion

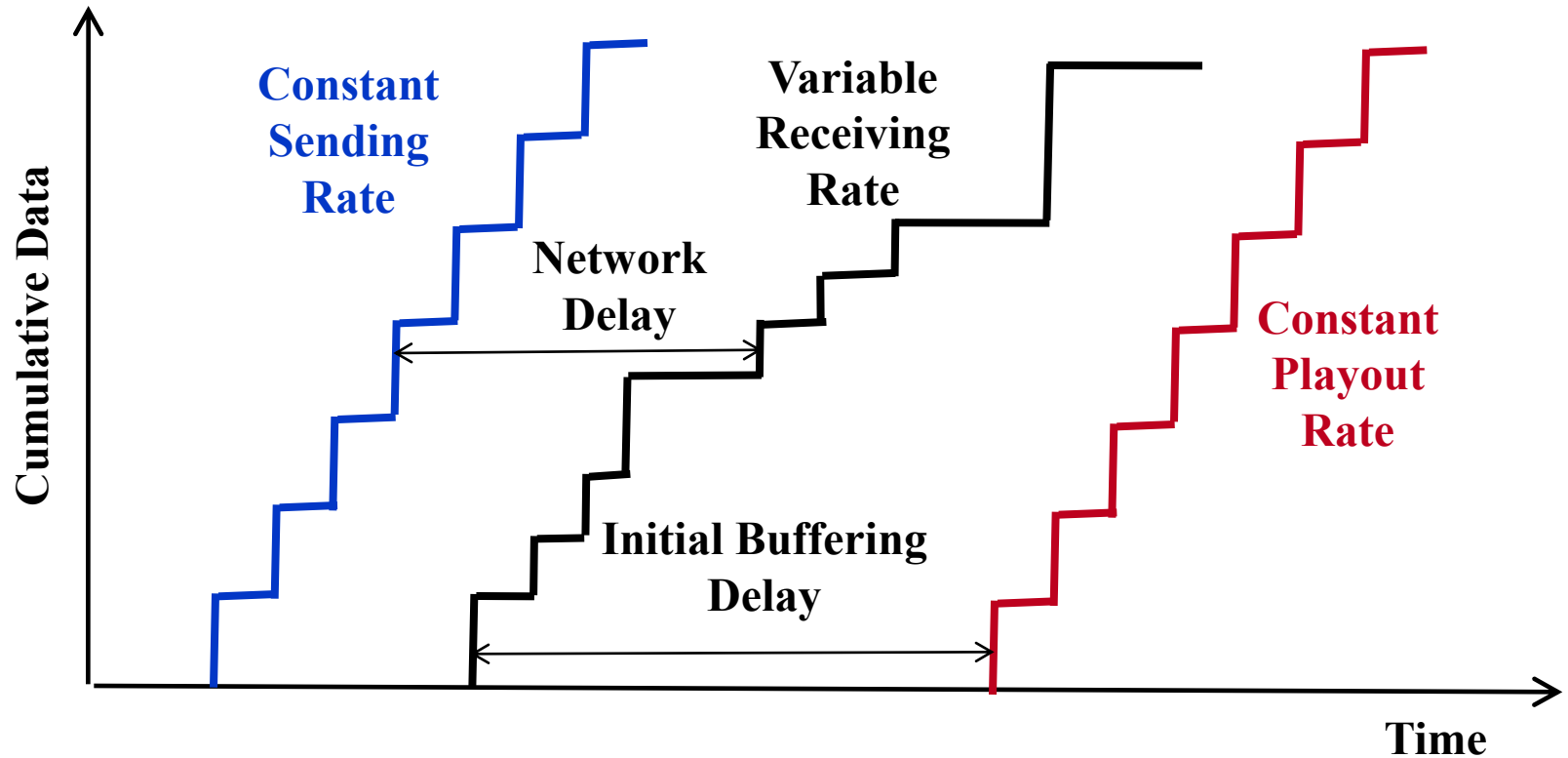
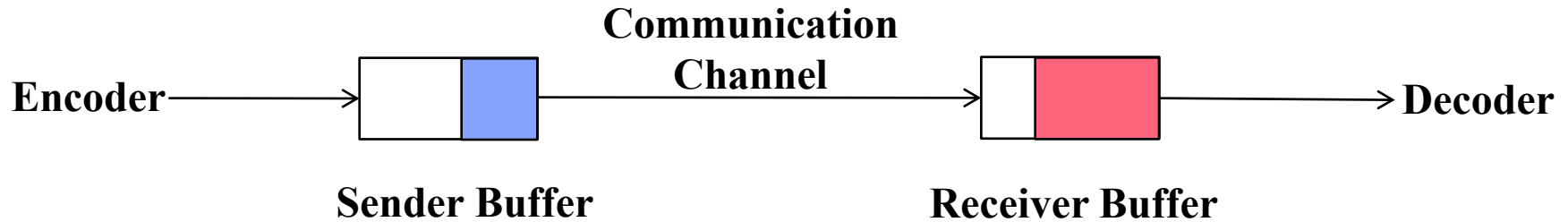
- **Tradeoff between performance and complexity**
 - Pixel-domain transcoders lead to higher coding efficiency at the expense of higher complexity
 - Cascaded transcoders result in higher coding efficiency at the expense of higher complexity
- **However, real-time transcoding for unicast service is challenging**
 - **processing power increases linearly with the number of users**
 - deployment cost of transcoders is high
- **Alternatives?**

Codecs for Efficient Transcoding

- **Through Multiple Description Coding (MDC) or Scalable Video Coding (SVC)**
 - **selectively transmit a subset of packets**
 - **scale well compared to traditional transcoding**
- **MDC: encode each video into multiple independent streams**
 - **any subset of substreams is decodable**
 - **but, high overhead on coding efficiency [Li et al. 03']**
- **SVC: encode each video into multiple layers**
 - **a layer is decodable if all lower layers are received**
 - **only about 10% coding efficiency overhead compared to non-scalable coding**

Buffer Management in Mobile Video

Sending/Receiving Buffers



Functionalities of Buffer

- **Smoothing bandwidth fluctuations**
- **Absorbing jitter**
- **Facilitating retransmission**
- **Allowing interleaving for error resilience**
 - **dealing with bursty errors**

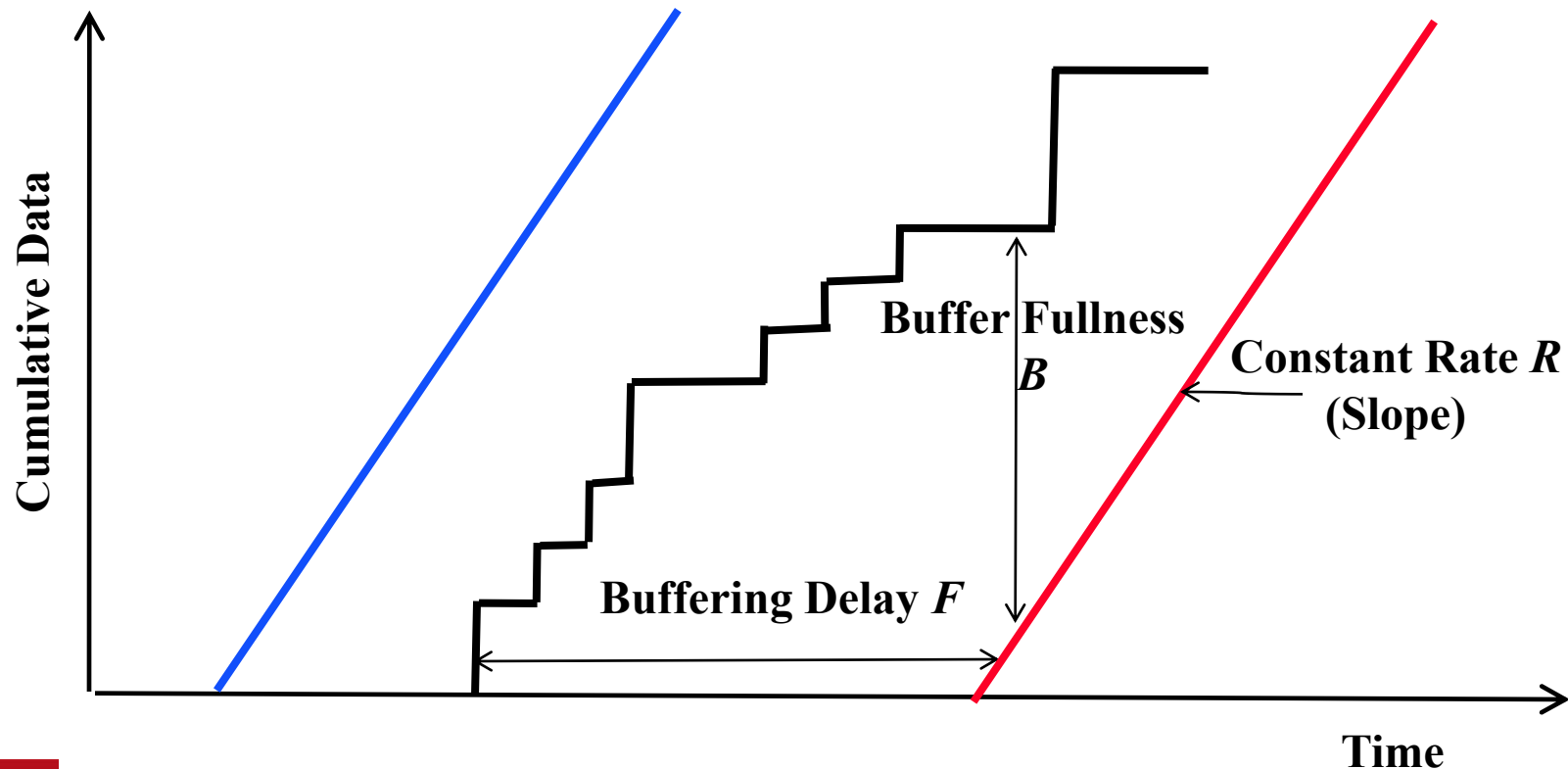
- **Potential issues: buffer violations**
 - **buffer overflow**
 - **buffer underflow**
- **Question: how to prevent buffer violations?**

Hypothetical Reference Decoder Model

- Define constraints on the rate variations of variable-bit-rate streams
- Part of video coding standards, such as MPEG-4 and H.264/AVC
- Video coding standards require encoders to control instantaneous rates following a HRD model to prevent overflow and underflow
- HRD model is essentially leaky buckets, which can be described by
 - R : mean decoding/encoding rate
 - B : buffer size
 - F : initial decoder buffer fullness, or initial buffering delay

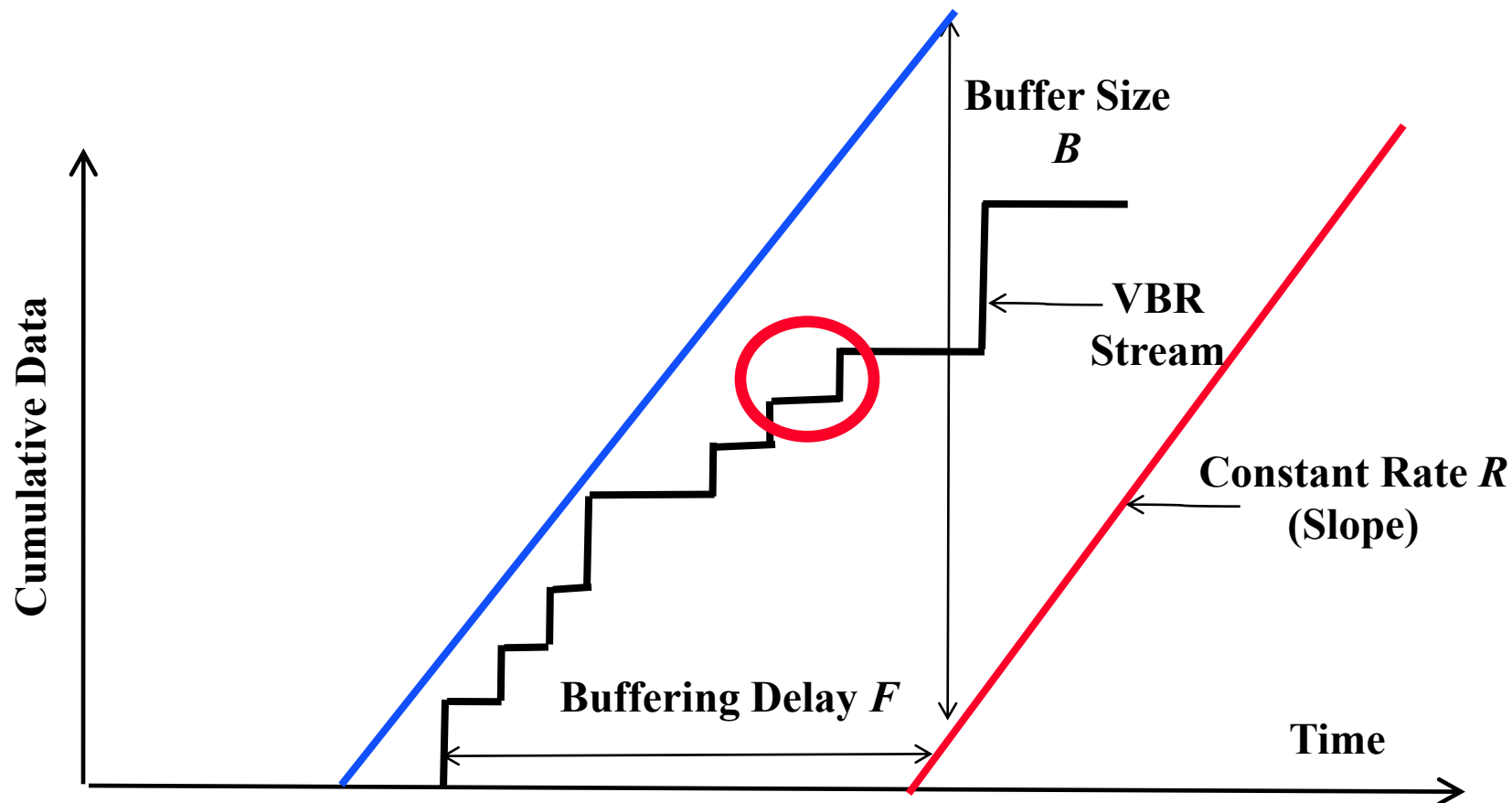
Usage of HRD model

- Encoder determines a valid (R, B, F) ; sender sends it to each receiver as meta-data
- A receiver uses it to determine the initial buffering delay based on buffer size



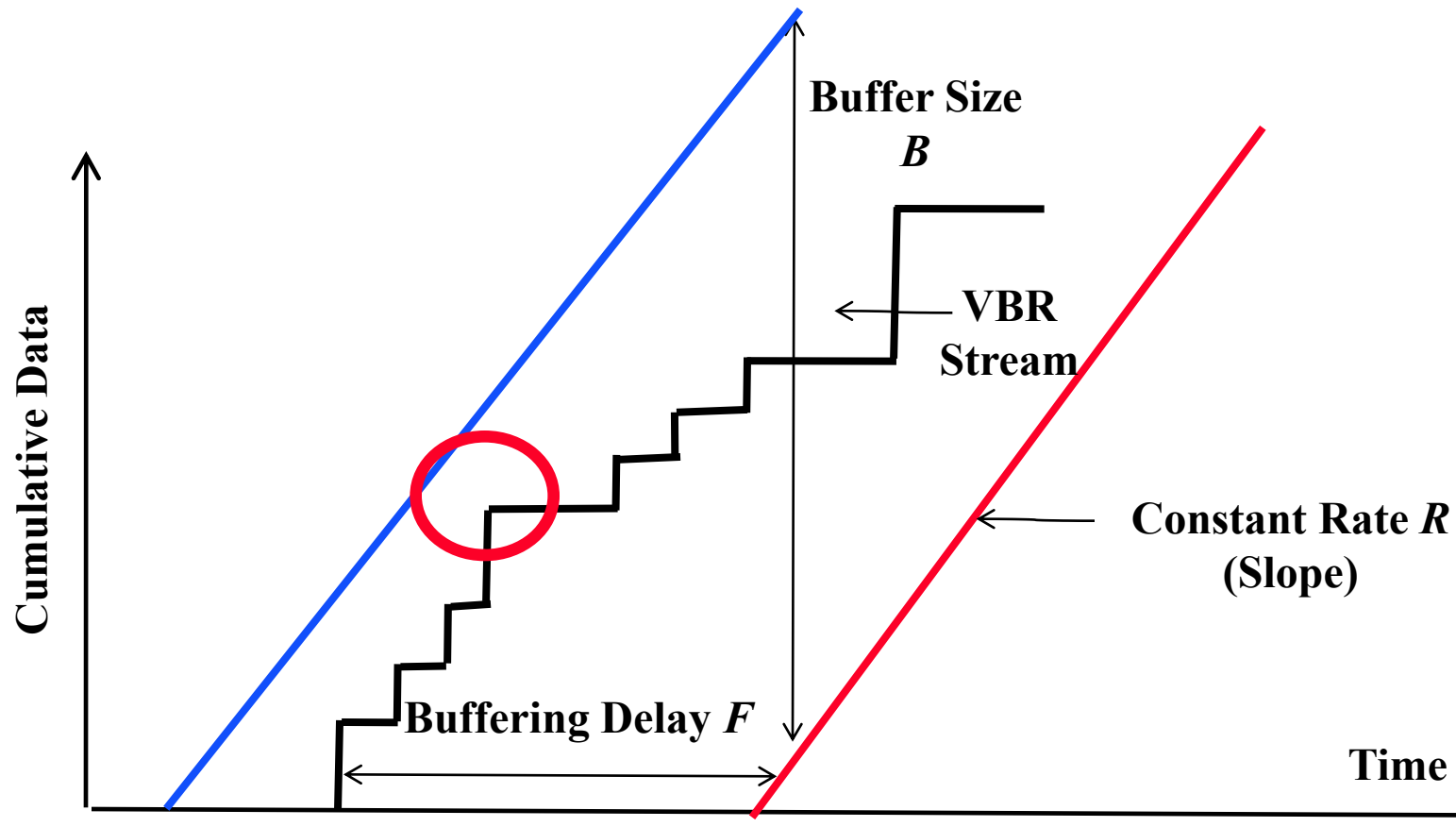
Buffer Underflow

- If receiver selects a too short initial buffering delay
 - decoder has nothing to playout



Buffer Underflow

- If receiver selects a too long initial buffering delay
 - receiver has no space to store received data



H.264/AVC HRD Model [Ribas-Corbera 03']

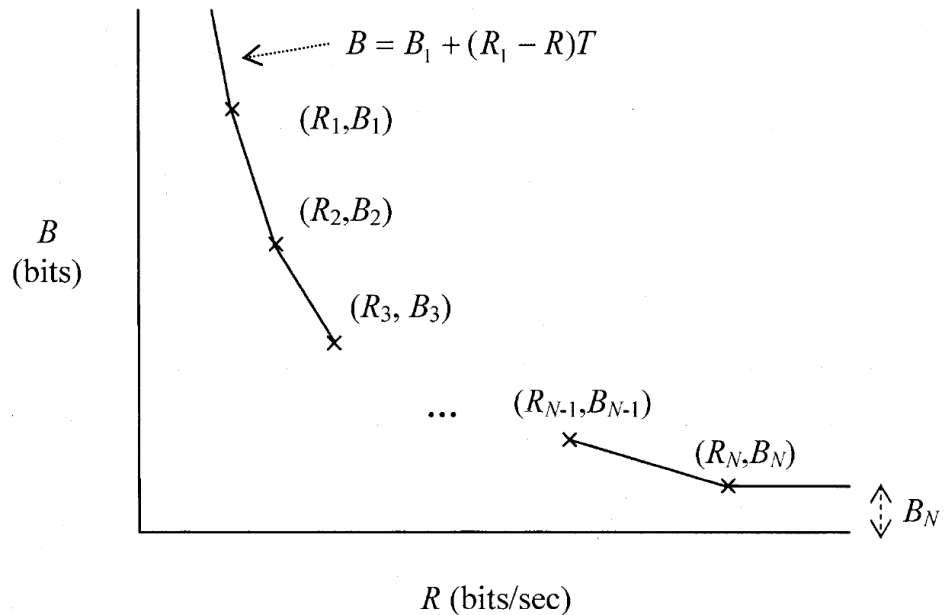
- Each video can be streamed with many leaky buckets
- H.264/AVC encoder provides multiple (R, B, F) values
- A receiver may choose any of the (R, B, F) value
- A receiver may also use interpolation for a more suitable (R, B, F) value

- **Examples:**

R=60 kbps, B=20 kb, F=15 kb

R=30 kbps, B=40 kb, F=40 kb

- **Lower rate →
larger buffer**



Computing HRD Parameters

```
# Consider bit rates from 50 kbps to 3 Mbps.
# buff1 is buffer state before frame removal.
# buff2 is buffer state after frame removal.
for R = 50000 : 50000 : 3000000
    j = j + 1
    B = 20*R
    buff1(1) = B
    minbuff = buff1(1)
    for i = 1:1:N,
        buff2(i) = buff1(i) - bits(i)
        if (buff2(i) < minbuff)
            minbuff = buff2(i)
        end
        buff1(i + 1) = buff2(i) + R*FPS
        if (buff1(i + 1) > B)
            buff1(i + 1) = B
        end
    end
end
# Minimum buffer size in bits
Bmin = B - minbuff
```

**Computes the
minimum buffer
size**

Computing HRD Parameters (cont.)

```
# Simulate leaky bucket to find Fmin.
# Set buffer size to be its minimum value.
# Initially assume Fmin is zero.
# Whenever underflow occurs do (1) and (2).
# (1) increase Fmin by underflow amount.
# (2) reset buffer.
B = Bmin
Fmin = 0
Buff1(1) = Fmin
for i = 1:1:N
    buff2(i) = buff1(i) - bits(i)
    if (buff2(i) < 0)
        Fmin = Fmin + (0 - buff2(i))
        buff2(i) = 0
    end
    buff1(i + 1) = buff2(i) + R = FPS
    if (buff1(i + 1) > B) buff1(i + 1) = B
end
End of loop for R
# Results are in Fmin and Bmin
```

**Computes the
initial fullness**

Power-Aware Video Streaming

Power Consumption of Mobile Video Streaming

- **Mobile devices often have limited battery capacity**
 - e.g., Samsung GalaxyS comes with 5.55 Watt-Hr
- **Power consumption of mobile devices is divided into** [Nokia 6630]
 - **computation: processors, ~0.6 Watt**
 - **communication: wireless modems, ~1.2 Watt**
 - **background: LCD panels, backlights, and speakers, ~1.2 Watt**
- **Power budget is tight, e.g., GalaxyS would have less than 2 hour battery life without any energy saving techniques**
- **Power consumption is critical to user experience**
 - **video streaming should not affect the availability of voice call service**

Energy Saving Techniques

- Several techniques have been proposed, which can be classified into two groups: lossless and lossy

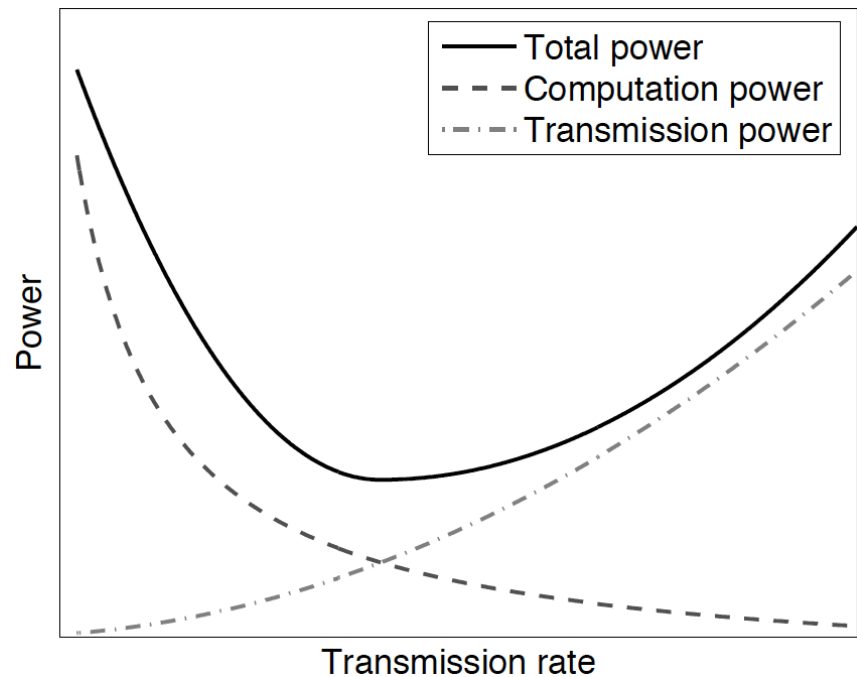
- Lossless techniques exploit various hardware characteristics to save energy **without** sacrificing perceived video quality
 - with limited operational range of energy saving
- Lossy techniques further **trade** video quality for longer battery life
 - longer battery life at slightly lower quality

Lossless Techniques

- **Dynamic voltage and frequency scaling**
 - adjust processing power to meet computational demand
 - finish multimedia tasks **just-in-time** ← reduce idling cycles
- **Battery-aware job scheduling**
 - leverage on nonlinearity between drawn current and battery life
 - scheduling jobs to match the optimal discharge rate
- **Power-aware transmission**
 - wireless interface has different modes with different power consumption levels
 - it also has nontrivial power consumption for mode transitions
 - streaming videos in bursts can prolong sleep time as well as reduce mode transitions, and thus save energy

Lossy Techniques

- **There exists a tradeoff between computation power consumption and communication power consumption**
 - exercise the tradeoff using complexity-scalable video codecs
- **How to reduce complexity?**
- **Skip some coding tools**
 - coding tools have different effectiveness on coding efficiency



Minimizing Power Consumption

- **Change video and channel coding parameters to save energy**
- **Under constraints of minimum quality and maximum delay**

$$\min_{s,c} E(s,c)$$

$$s.t.: D(s,c) \leq D_0, T(s,c) \leq T_0,$$

where s and c are parameters for video coding and channel coding, respectively; $E(\cdot)$, $D(\cdot)$, and $T(\cdot)$ represent energy, distortion, and delivery delay. D_0 and T_0 are the bounds on video quality and transmission delay.

- **Total energy $E(s,c) = E_c(s,c) + E_t(s,c) + E_m(s,c)$, where**
 - $E_c(s,c)$ is computational energy, $E_t(s,c)$ is communication energy, and $E_m(s,c)$ is miscellaneous energy

Maximizing Video Quality

- Under constraints of maximum energy consumption and maximum delay

$$\min_{s,c} D(s,c)$$

$$s.t.: E(s,c) \leq E_0, T(s,c) \leq T_0,$$

where E_0 and T_0 are the bounds of energy consumption and transmission delay.

- Specialized formulations are possible

Open Challenges [Zhang '09]

- **Battery management of mobile video is still difficult**
 - **battery nonlinearity, real-time requirement, network dynamics, and human interactivity**

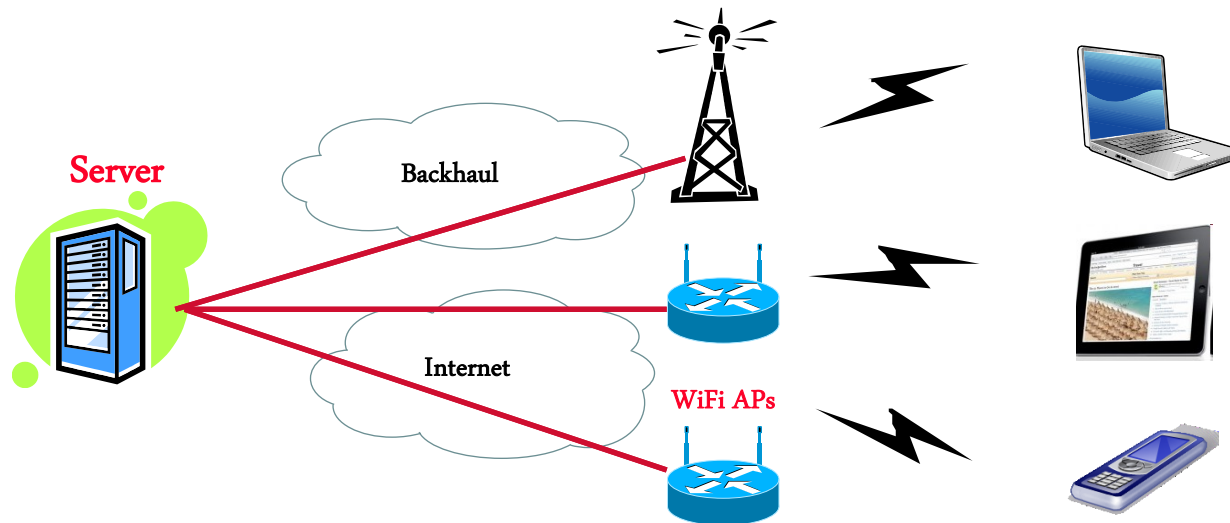
- **Complexity of video codec is hard to model**
 - **too many parameters to choose, and diverse video characteristics**

- **Resulting optimization problems may not be tractable**
 - **both video and channel encoders have many controllable parameters**

Multihomed Video Streaming

Offloading Traffic from Cellular Networks

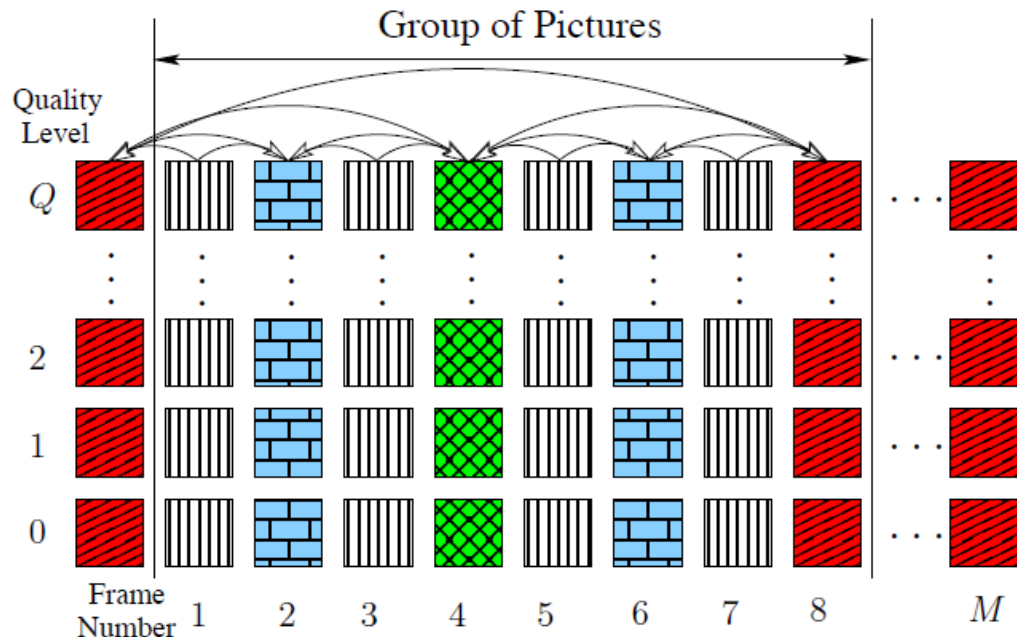
- Video streaming has high bandwidth requirements
- However, T-mobile and AT&T recently reported more than 50 times of data traffic increase [Open Mobile Summit '09]



- This is called **multihoming**, which is attractive to
 - ISPs, such as T-Mobile, for lower transit cost
 - Subscribers for better quality-of-service

Dynamic Network Conditions

- **Problem: access networks are heterogeneous and dynamic**
- **Employ scalable video: frames are coded into multiple layers**
 - **incremental quality improvement**
 - **complicated interdependency due to prediction**



Challenges and Problem Statement

- **Determine streaming rate on each access network is hard**
[Hsu ISM'10]
 - streaming at a rate close to end-to-end network capacity leads to congestion, and late packets
 - streaming at a low rate wastes available resources
 - need a network model to proactively prevent congestion
- **Packets of scalable streams have complex inter-dependency**
 - need a video model to predict expected quality
- **The problem: determine (i) what video packets to send, (ii) over which network interface, and (iii) at what rate, so that the overall streaming quality is maximized**

Notations


□ Scalability

- Client: $u=1,\dots,U$
- Temporal: Different frames with inter-frame prediction $m=1,\dots,M_u$
- Spatial: Quality layers $q=0,\dots,Q_u$
- Multihoming: networks $n=1,\dots,N$
- Network Abstraction Layer Unit (NALU) : $g_{u,m,q}$

□ Scheduling

- Deterministic: $x_{u,m,q,n} \in \{0, \textcircled{1}\}$
- Randomized: $x_{u,m,q,n} \in [0, 1]$

If $g_{u,m,q}$ is sent over network n



Video Quality Model

Truncation distortion: capturing loss of a NALU $g_{u,m,q}$

- A packet is decodable if all packets in lower quality ($q' < q$) layers are received

$$e_{u,m} = \hat{\delta}_{u,m} + \sum_{q=0}^{Q_u} (1 - \prod_{q' \leq q} x_{u,m,q'}) \delta_{u,m,q}$$

Distortion if all packets are received

Additional distortion
If $g_{u,m,q}$ is not decoded

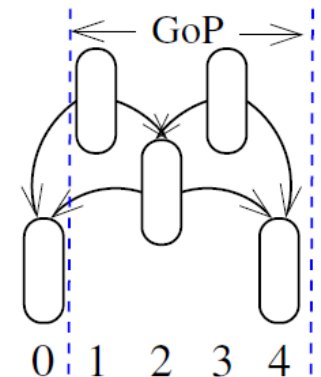
Drifting distortion: capturing error propagation

- Inter-frame predictions based on imperfectly reconstructed parent packets, $P_{u,m}$
- Convex increasing function

$$y_{u,m} = \alpha_{u,m} + \sum_{k \in P_{u,m}} \beta_{u,m,k} e_{u,k}$$

- Parameters: Estimated from actual data

Nonnegative



Network Model

□ Packet loss probability (p_n) depends on

- Rate: (r_n)
- Available bandwidth (c_n)
- Packet decoding deadline (t_0)

□ Model

- M/M/1 model $p_n = e^{-\frac{t_0(c_n - r_n)}{\alpha_n}}$
- Increasing in c_n , decreasing in r_n
- α_n : linear regression parameter
- accurate in streaming video applications [Zhu et. al '05]

□ Assumption : statistical independence of different networks

- Good approximation using a two-timescale approach [Jiang et al. '10]
- Network converges to steady-state in between scheduling events

Problem Formulation

□ Cost minimization problem

- Accounts for service differentiation and fairness among users and frames

$$\begin{aligned}
 \min_x \quad & C(\mathbf{d}) \quad \text{Cost function (increasing, convex)} \\
 \text{s.t.} \quad & r_n = \sum_{u=1}^U \frac{F_u}{M_u} \sum_{m=1}^{M_u} \sum_{q=0}^{Q_u} S_{u,m,q} x_{u,m,q,n}, \quad \text{Rate} \\
 & p_n = e^{-t_0(c_n - r_n)/\alpha_n}, \quad \text{Loss probability} \\
 & x_{u,m,q} = \sum_{n=1}^N (1 - p_n) x_{u,m,q,n}, \\
 & e_{u,m} = \hat{\delta}_{u,m} + \sum_{q=0}^{Q_u} (1 - \prod_{q' \leq q} x_{u,m,q'}) \delta_{u,m,q}, \quad \text{Not convex} \\
 & y_{u,m} = \alpha_{u,m} + \sum_{k \in \mathbf{P}_{u,m}} \beta_{u,m,k} e_{u,k}, \\
 & d_{u,m} = e_{u,m} + y_{u,m}, \\
 & \sum_{n=1}^N x_{u,m,q,n} \leq 1, \\
 & x_{u,m,q,n} \in \{0, 1\}. \quad \text{Randomized scheduling}
 \end{aligned}$$

Heuristic Algorithm 1/2

SRDO

0. INPUT: P_{\max} is the maximum packet loss rate
 1. let $\mathbf{x} = \{x_{u,m,q,n} = 0 \mid \forall u, m, q, n\}$
 2. sort $g_{u,m,q}$ on $\frac{\delta_{u,m,q}}{s_{u,m,q}}$
 3. for $\hat{n} = \operatorname{argmin}_{n=1}^N p_n$
 4. let $g_{\hat{u},\hat{m},\hat{q}}$ be the next unsent NALU
 5. if sending $g_{\hat{u},\hat{m},\hat{q}}$ on \hat{n} causes $p_{\hat{n}} > P_{\max}$ return \mathbf{x}
 6. else update \mathbf{x} with $x_{\hat{u},\hat{m},\hat{q},\hat{n}} = 1$
 7. if no more unsent NALU return \mathbf{x}
-

Heuristic Algorithm 2/2

PRDO

1. **let** $\mathbf{x} = \{x_{u,m,q,n} = 0 \mid \forall u, m, q, n\}$
 2. **forever**
 3. **if** \mathbf{g}_d is empty **return** \mathbf{x}
 4. **let** \mathbf{g}_d be all immediately decodable NALGs
 5. **for** $g_{u,m,q} \in \mathbf{g}_d$
 6. **for** $n = 1$ to N
 7. **compute** $b_{u,m,q,n}$ based on \mathbf{x}
 8. **let** $\frac{b_{\hat{u},\hat{m},\hat{q},\hat{n}}}{s_{\hat{u},\hat{m},\hat{q}}} \geq \frac{b_{u,m,q,n}}{s_{u,m,q}} \quad \forall u, m, q, n$
 9. **if** $b_{\hat{u},\hat{m},\hat{q},\hat{n}} \leq 0$ **return** \mathbf{x}
 10. **update** \mathbf{x} with $x_{\hat{u},\hat{m},\hat{q},\hat{n}} = 1$, **update** \mathbf{g}_d .
-

Term-by-Term Convex Approximation

Goal: Obtain a convex superset of the constraint set

1. Term-by-term convex approximation (TTC)

$$\begin{aligned}x_{u,m,q} &\leq \sum_{n=1}^N \min(1 - p_n, x_{u,m,q,n}), \\e_{u,m} &\geq \hat{\delta}_{u,m} + \sum_{q=0}^Q (1 - \min_{q' \leq q} x_{u,m,q'}) \delta_{u,m,q},\end{aligned}$$

- **Polynomial number of constraints in U,M,Q,N**
- **Weak approximation of the probability of successful packet delivery $x_{u,m,q}$**

Multilinear Convex Approximation

Goal: Obtain a convex superset of the constraint set

2. Multilinear convex approximation (MC)

- **Convex envelope of multilinear functions [Sherali '97]**
 - **Minimum of affine functions**
- **Tightest convex approximation**
- **Exponential number of constraints in Q, N**
- **Constraint on $x_{u,m,q}$ depends exclusively on N , NOT on problem parameters**

Hybrid Convex Approximation

Goal: Obtain a convex superset of the constraint set

3. Hybrid Convex Approximation (HC)

- **Term-by-term approximation for truncation distortion $e_{u,m}$**
- **Multilinear approximation for probability of successful packet delivery $x_{u,m,q}$**

$$\begin{aligned} x_{u,m,q} &\leq \sum_{n=1}^N \min(1 - p_n, x_{u,m,q,n}), \\ e_m &\geq \hat{\delta}_m + \sum_{q=0}^Q \delta_{m,q} - \min \{ L_m^2(\xi, \bar{\mathbf{x}}) := \sum_{q=0}^Q (\sum_{q'=0}^{q-1} \prod_{i \leq q'} \xi(i) \delta_{m,q'} + \\ &\quad \sum_{q'=q}^Q \prod_{i \leq q', i \neq q} \xi(i) x_{m,q} \delta_{m,q'}) - Q \sum_{q=0}^Q \prod_{i \leq q} \xi(i) \delta_{m,q} \\ \text{s.t. } &\xi \in \{0, 1\}^{Q+1}, L_m^2(\xi, \bar{\mathbf{x}}) \leq \sum_{q=0}^Q \bar{x}_{m,q} \delta_{m,q} \quad \forall \bar{\mathbf{x}} \in \{0, 1\}^{Q+1} \}, \end{aligned}$$

- **Polynomial complexity in U, M, Q , exponential in N**
- **Good trade-off of approximation accuracy vs. complexity for low N**

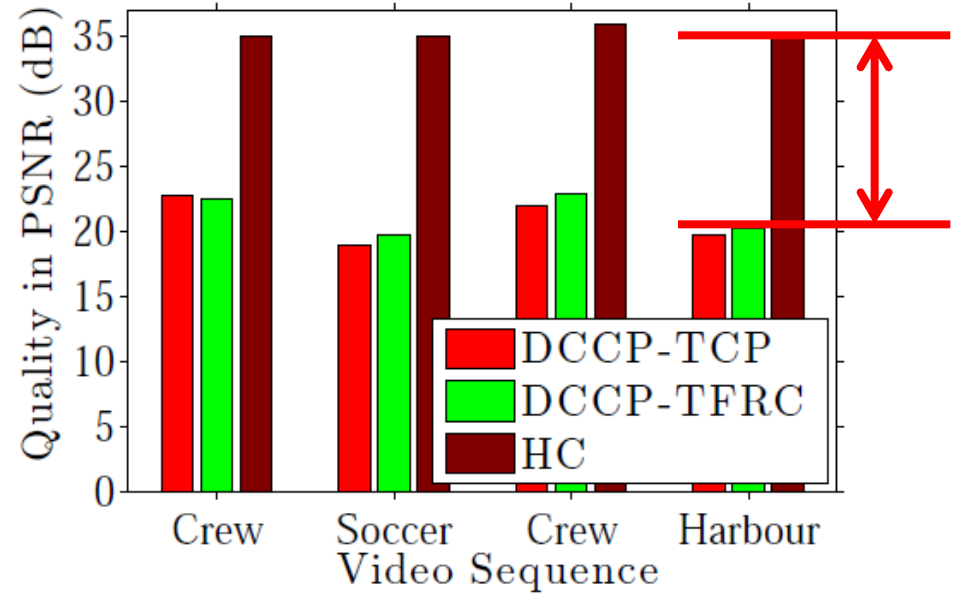
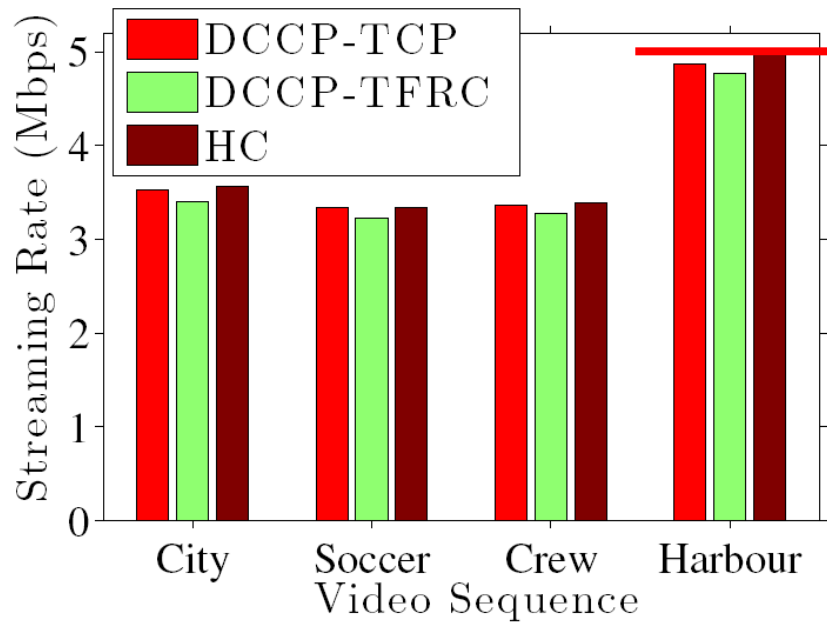
Solving the Convex Approximations

- **Properties of our convex approximations**
 - **Non-empty compact set of solutions**
 - **Strong duality**
 - **Non-empty set of dual optimal solutions**
- **These properties are important for the performance of numerical methods [Boyd et al. 04']**
- **We use CVX to solve our convex programs**
 - **a convex program solvers based on Matlab**
 - **developed at Stanford**

Simulation Setup

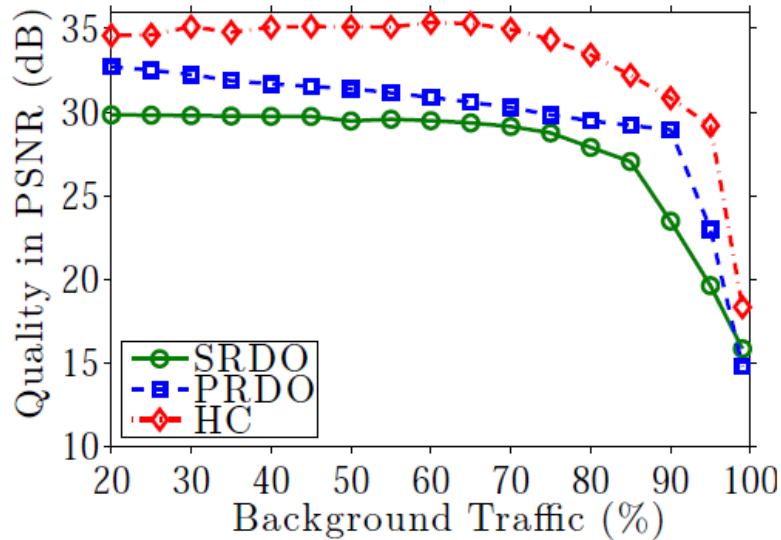
- **Scheduling period : $M = 32$**
- **Number of quality enhancement layers : $Q=7$**
- **Number of access networks : $N=3$**
- **Decoding deadline : $t_0 = 1$ sec**
- **SVC video streams: Crew, Harbour, City, and Soccer**
- **Trace-driven simulations (NS-2)**
 - **Data from subnets at Stanford University and DT Labs Berlin**
 - **Used Abing to measure end-to-end available bandwidth and round-trip time**
 - **Run 300 simulations for each setup**

Comparison against Current Solutions

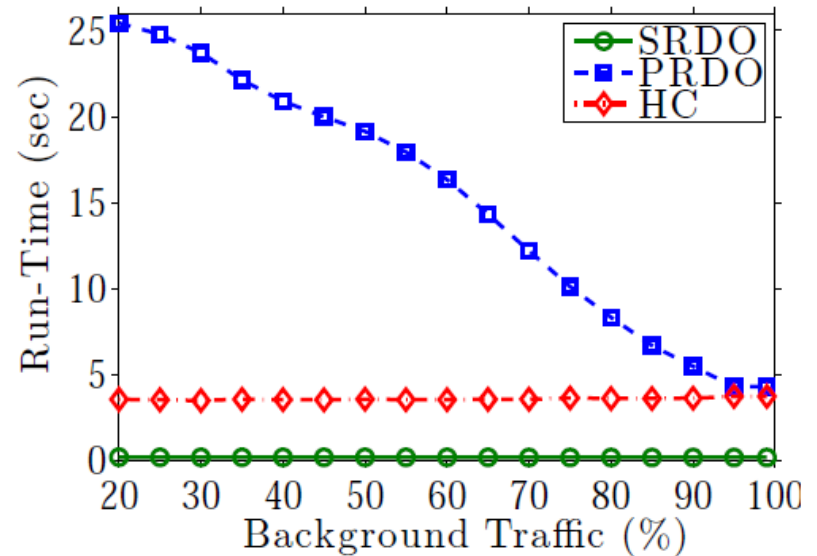


- **Proposed algorithms are TCP-Friendly**
- **Proposed algorithms constantly outperform current ones by more than 10 dB**

Complexity versus Performance



**Convex solution
outperforms heuristics
in performance**



**Convex solution has a
reasonable time complexity**

Part IV

Summary and Outlook

Summary

- **Huge interest in mobile multimedia**
- **Multimedia distribution models**
 - **Multicast/Broadcast**
 - **1-to-many (Mobile TV)**
 - **Unicast**
 - **1-to-1 (on demand, e.g., YouTube)**
- **Research problems in the mobile multicast model**
 - **Burst transmission and energy saving**
 - **Cooperative wireless streaming to save more energy**
 - **Supporting heterogeneous receivers**
 - **Design and implementation of a mobile TV testbed**

Summary

- **Research problems in the mobile unicast model**
 - **Different video compression techniques and various stream transport methods**
 - **Stream adaptation methods: transcoding and scalable video coding**
 - **Avoiding buffer overflow/underflow in wireless networks**
 - **Power-aware video streaming**
 - **Needs and techniques for multihomed video streaming**

Future Work

- **Generalize to 3D Video Streams**
 - 3D video: multiple views merged
 - Quite challenging specially on mobile devices
- **Context-aware adaptation and services**
 - Adapt to current conditions (battery, error rate, ..., even viewing angle in 3D setting) of mobile devices
- **Design models for 3D videos**
 - Quality of experience models
 - Power-Rate-Distortion (P-R-D)
- **Transmitting User Generated Contents**
 - Automatic classification of content
 - Efficient transmission of relevant materials to mobiles

Future Work

- **Video adaptation in heterogeneous access networks**
 - Have diverse network-level QoS characteristics
 - Mapping network QoS levels to human perceived QoE is still challenging in multimedia applications
- **TCP (or HTTP) video streaming**
 - Most smoothing algorithms were designed for UDP
 - How they interact with TCP rate control is not well understood
- **Many new applications are enabled by cloud computing**
 - Offloading computational complexity to remote servers
 - Challenge: dealing with network latency in interactive applications, such as distributed games

Tools for Experiments

■ Video Traces

- Arizona State: <http://trace.eas.asu.edu/>, long video sequences coded in SVC, AVC, MPEG-4, MPEG-2, and MDC coders
- TU Berlin <http://www.tkn.tu-berlin.de/research/trace/ltvt.html>, long video sequences coded in MPEG-4 and H.263

■ Video Sequences

- Xiph Open-source Video Production <http://media.xiph.org/>, pointing to many other links for Raw video sequences

■ Codecs

- AVC Reference Coder <http://iphome.hhi.de/suehring/tml/>
- SVC Reference Coder http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm
- X264 Coder <http://www.videolan.org/developers/x264.htm>
- Nokia's 3D Coder/Decoder <http://research.nokia.com/research/mobile3D>

Tools for Experiments (cont.)

■ Streaming Tools

- Darwin Open-source Version of QuickTime Server
<http://dss.macosforge.org/>
- VLS VideoLAN's Streaming Server
<http://www.videolan.org/vlc/streaming.html>
- VLC VideoLAN's Player <http://www.videolan.org/vlc/>
- Live555 Streaming Library <http://www.live555.com/liveMedia/>

■ Video Quality Scripts

- Matlab Central's File Exchange
<http://www.mathworks.com/matlabcentral/>
 - For example, computing PSNR of two YUV files
<http://www.mathworks.com/matlabcentral/fileexchange/12455-psnr-of-yuv-videos>
- SSIM Tool <http://www.ece.uwaterloo.ca/~z70wang/research/ssim/>
- MSU Video Quality Tool
http://compression.ru/video/quality_measure/video_measurement_tool_en.html

References

- [Rysavy]: Rysavy Research, HSPA to LTE-Advanced: 3GPP Broadband Evolution to IMT-Advanced (4G), http://www.3gamericas.org/documents/3G_Americas_RysavyResearch_HSPA-LTE_Advanced_Sept2009.pdf , September 2009.
- [Allot]: Allot Communications, Allot Mobile Trends Report Shows 68% Growth in Global Mobile Data Bandwidth Usage in H1, 2010, <http://www.allot.com/index.aspx?id=3797&itemID=40579> , September 2010.
- [WebM]: Jan Ozer, WebM vs. H.264: A First Look, Streaming Media magazine, <http://www.streamingmedia.com/Articles/News/Featured-News/WebM-vs.-H.264-A-First-Look-69351.aspx>, August/September 2010.
- [Hefeeda 10, ToN]: M. Hefeeda and C. Hsu, [On Burst Transmission Scheduling in Mobile TV Broadcast Networks](#), IEEE/ACM Transactions on Networking, Accepted July 2009.
- [Hsu 09, INFOCOM]: C. Hsu and M. Hefeeda, [Time Slicing in Mobile TV Broadcast Networks with Arbitrary Channel Bit Rates](#), In Proc. of IEEE INFOCOM 2009, pp. 2231--2239 , Rio de Janeiro, Brazil, April 2009.

References

- [Hsu 10, TOMCCAP] : C. Hsu and M. Hefeeda, Using Simulcast to Control Channel Switching Delay in Mobile TV Broadcast Networks, ACM Transactions on Multimedia Computing, Communications, and Applications, Accepted October 2009.
- [Liu 10, MMSys]: Y. Liu and M. Hefeeda, Video Streaming over Cooperative Wireless Networks, In Proc. of ACM Multimedia Systems (MMSys'10), pp. 99--110, Scottsdale, AZ, February 2010.
- [Hefeeda 10, TOMCCAP]: M. Hefeeda and C. Hsu, Design and Evaluation of a Testbed for Mobile TV Networks, ACM Transactions on Multimedia Computing, Communications, and Applications, Accepted January 2010.
- [Hsu 10, ISM]: N. Freris, C. Hsu, X. Zhu, and J. Singh, Resource Allocation for Multihomed Scalable Video Streaming to Multiple Clients, in Proc. of IEEE International Symposium on Multimedia (ISM'10), Taichung, Taiwan, December 2010.
- [Hsu10, MMSys]: C. Hsu and M. Hefeeda, Quality-aware Segment Transmission Scheduling in Peer-to-Peer Streaming Systems, In Proc. of ACM Multimedia Systems (MMSys'10), pp. 169--180, Scottsdale, AZ, February 2010.

References

- [Xin 05]: J. Xin, C. Lin, and M. Sun, Digital Video Transcoding, Proceedings of the IEEE, 93(1):84–97, January 2005.
- [Zhang 09]: J. Zhang, D. Wu, S. Ci, H. Wang, and A. Katsaggelos, Power-aware Mobile Multimedia: a Survey, Journal of Communications, 4(9):600–613, October 2009.
- [Ribas-Corbera 03]: J. Ribas-Corbera, P. Chou, and S. Regunathan, A Generalized Hypothetical Reference Decoder for H.264/AVC, IEEE Transactions on Circuits and Systems for Video Technology, 13(7):674–687, July 2003.

Thank You

Backup Slides

Outline (old)

- **Introduction**
 - **Importance of mobile multimedia**
 - **Distribution models**
- **Mobile Multimedia Multicast/Broadcast**
 - **Energy saving**
 - **Reducing channel switching delay**
 - **Supporting heterogeneous receivers**
- **Mobile Multimedia Unicast**
 - **Buffer management**
 - **Energy saving**
 - **Stream adaptation**
- **Summary and outlook**

Switching Delay in Mobile Multimedia Networks

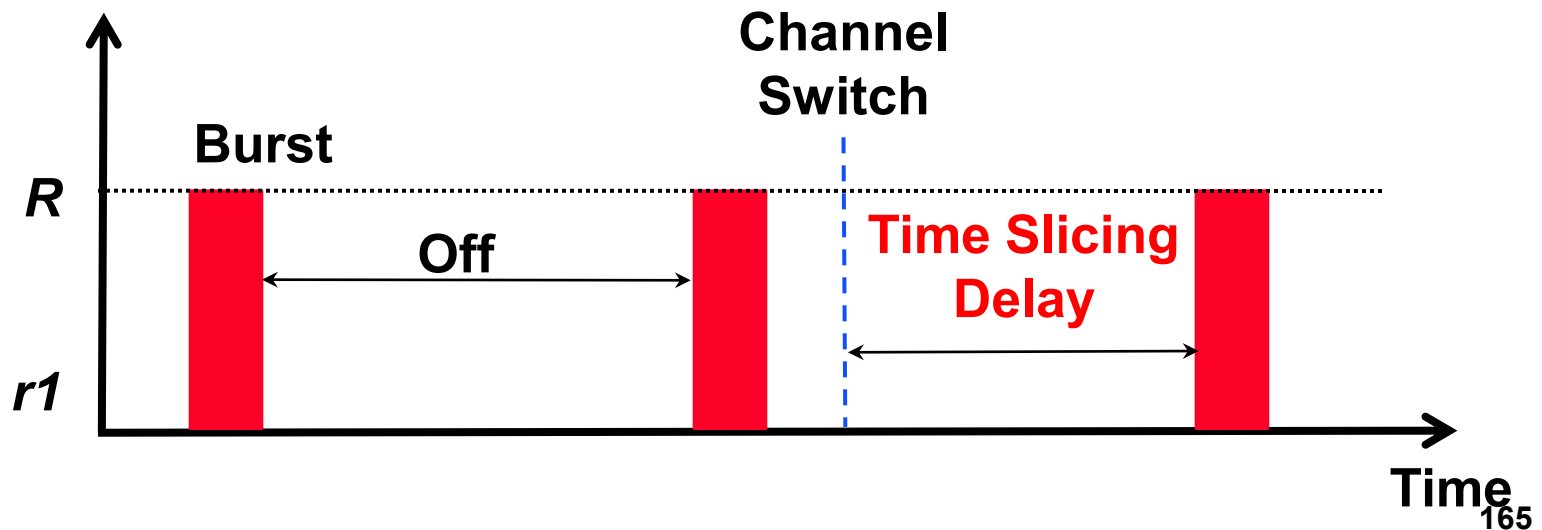
Controlling Channel Switching Delay

- Users usually flip through many channels
- Long/variable delays are annoying
- One of the complaints of DVB-H subscribers
 - Delay could be up to 6 sec
 - Our own measurement on Nokia N92/N96 phones: **delay > 5 secs**
- Goal: bound maximum switching delay **without** sacrificing energy saving for mobile receivers [Hsu 10, TOMCCAP]

Controlling Channel Switching Delay

- **Switching delay has multiple components**

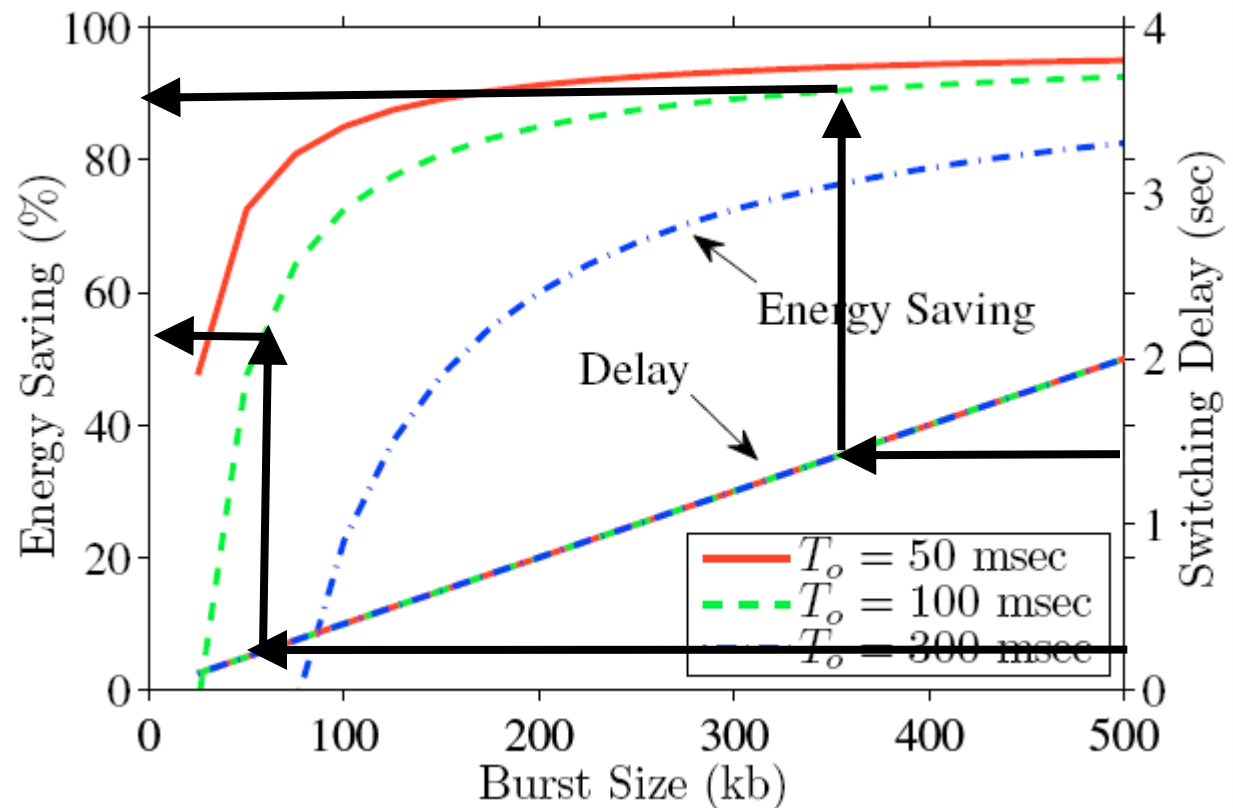
- Time slicing delay (our focus)
- Frame refresh delay (till an I-frame arrives)
 - Add more/redundant I-frames [Vadakital 07]
 - Move I-frames closer to start of burst [Rezaei 07, 08]
- Processing and Decoding delays



Controlling Delay: Current Approach #1

- Reduce inter-burst periods → wastes energy
- Reduce delay from 1.5 to 0.25 sec →

energy saving drops from 90% to 55%



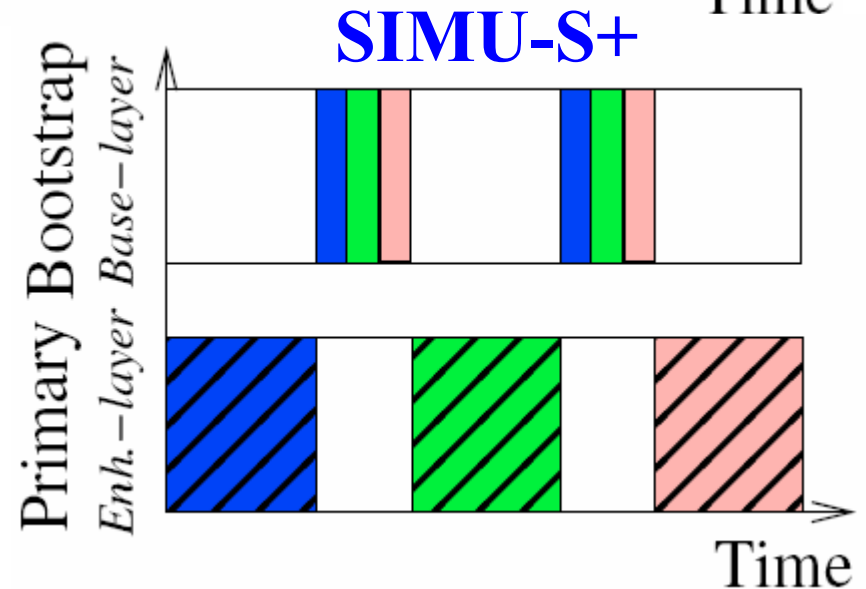
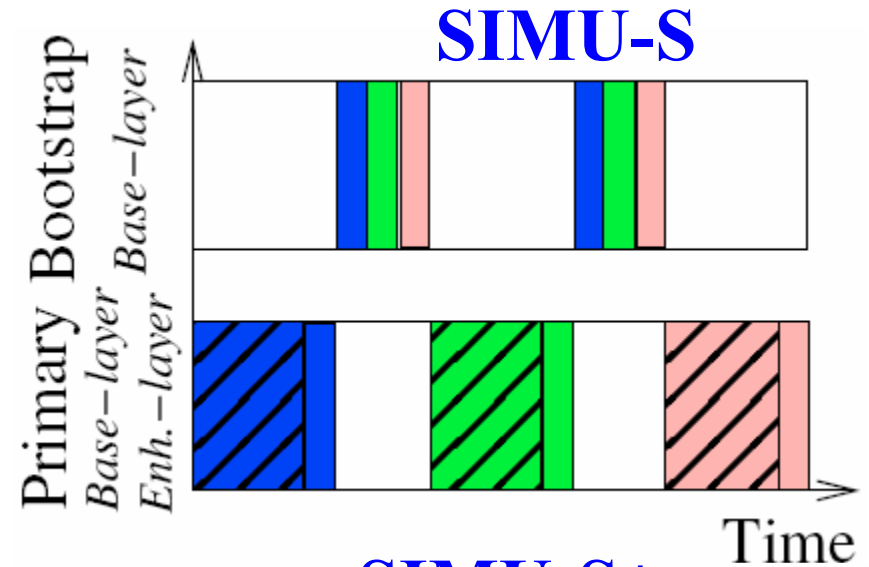
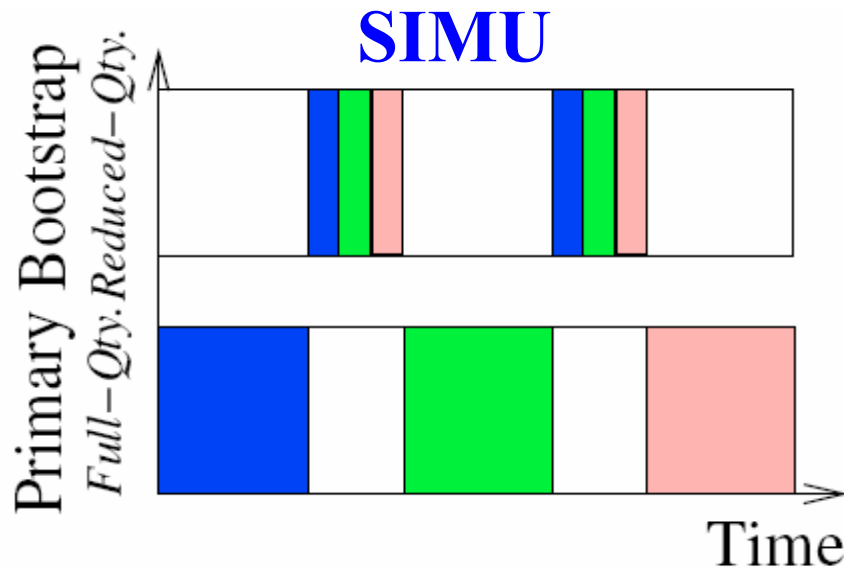
Controlling Delay: Current Approach #2

- **DVB-H standard** [EN 102377, May 2007]
 - Suggests **bundling** multiple channels in one group → virtually zero switching delay within a group
- **But,**
 - Delay across groups can be large
 - Devices receive all data of the bundle → **wastes energy**
 - How do we group channels in the first place (manual)?

Controlling Delay: Our Approach

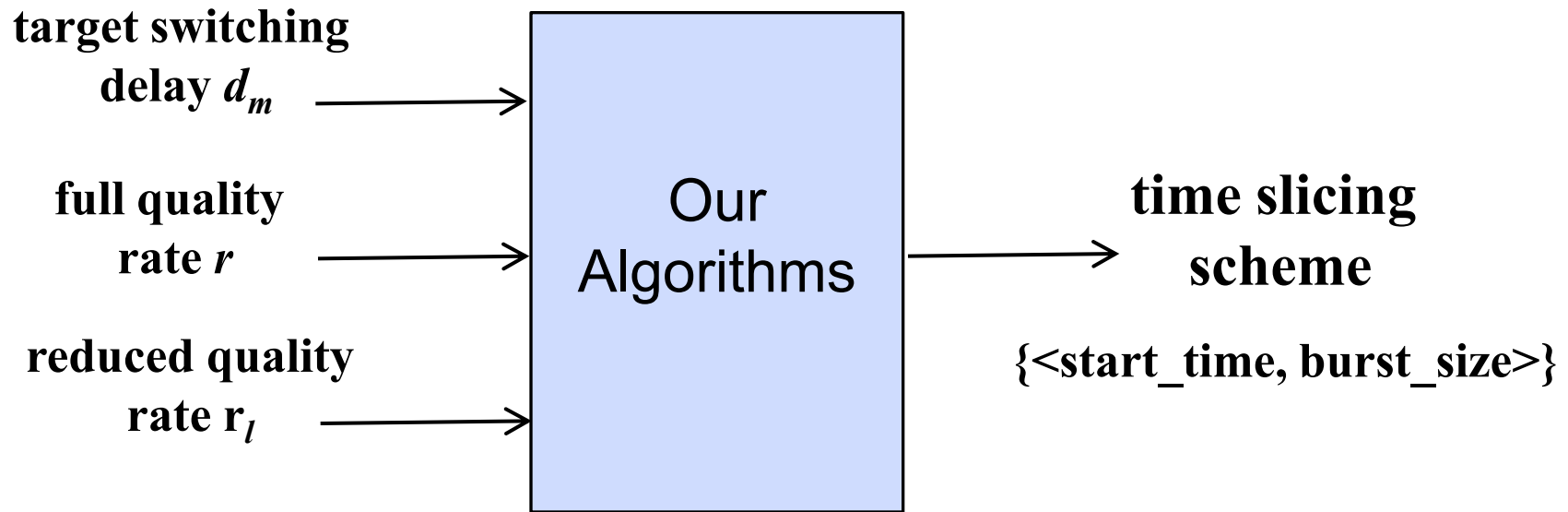
- **Use simulcast**
 - Broadcast each TV channel over two burst trains
 - One optimized for delay (**bootstrap**)
 - The other optimized for energy saving (**primary**)
 - Devices tune to bootstrap bursts for fast playout, then tune to primary bursts for high energy saving
- **Systematically construct optimal time slicing schemes**
- **Three variations**
 - SIMU : traditional video systems (non-scalable codecs)
 - SIMU-S: scalable codecs
 - SIMU-S+: scalable codecs, bandwidth limited networks

Controlling Delay: Our Approach



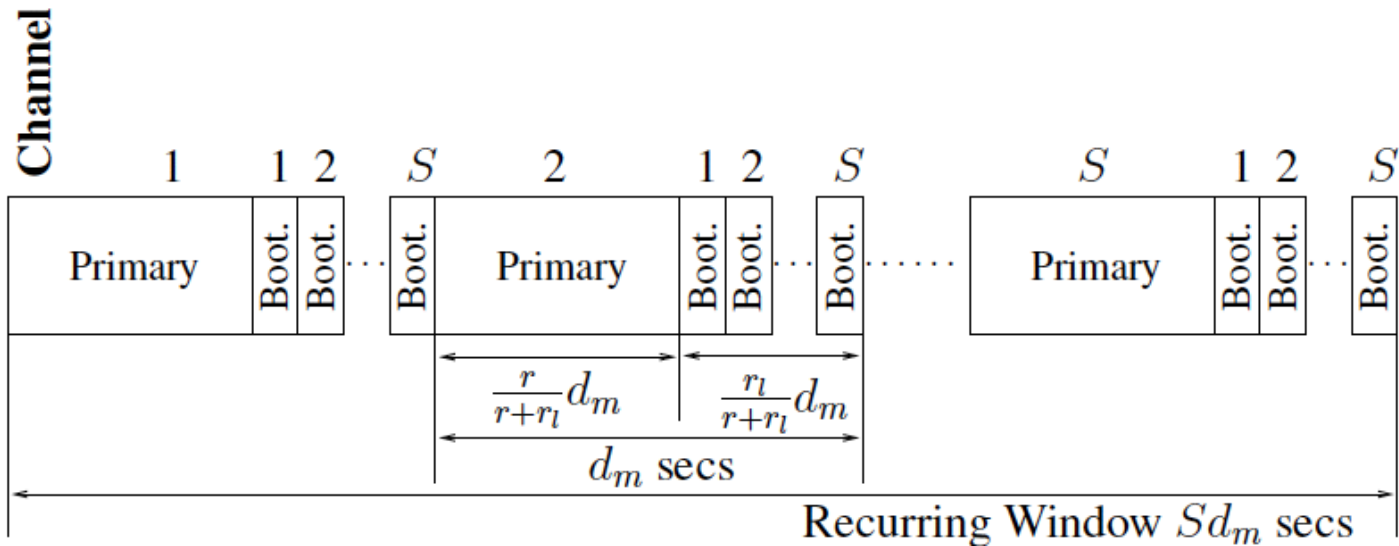
- Low quality not noticed during flipping
- Scalable codecs facilitate stream management
- SIMU-S+ less energy saving than SIMU-S, but better bw utilization

Bounding Switching Delay



- **Run at the base stations to multiplex TV channels into a traffic stream**

Time Slicing Scheme – SIMU/SIMU-S



- Primary bursts:
- Bootstrap bursts:

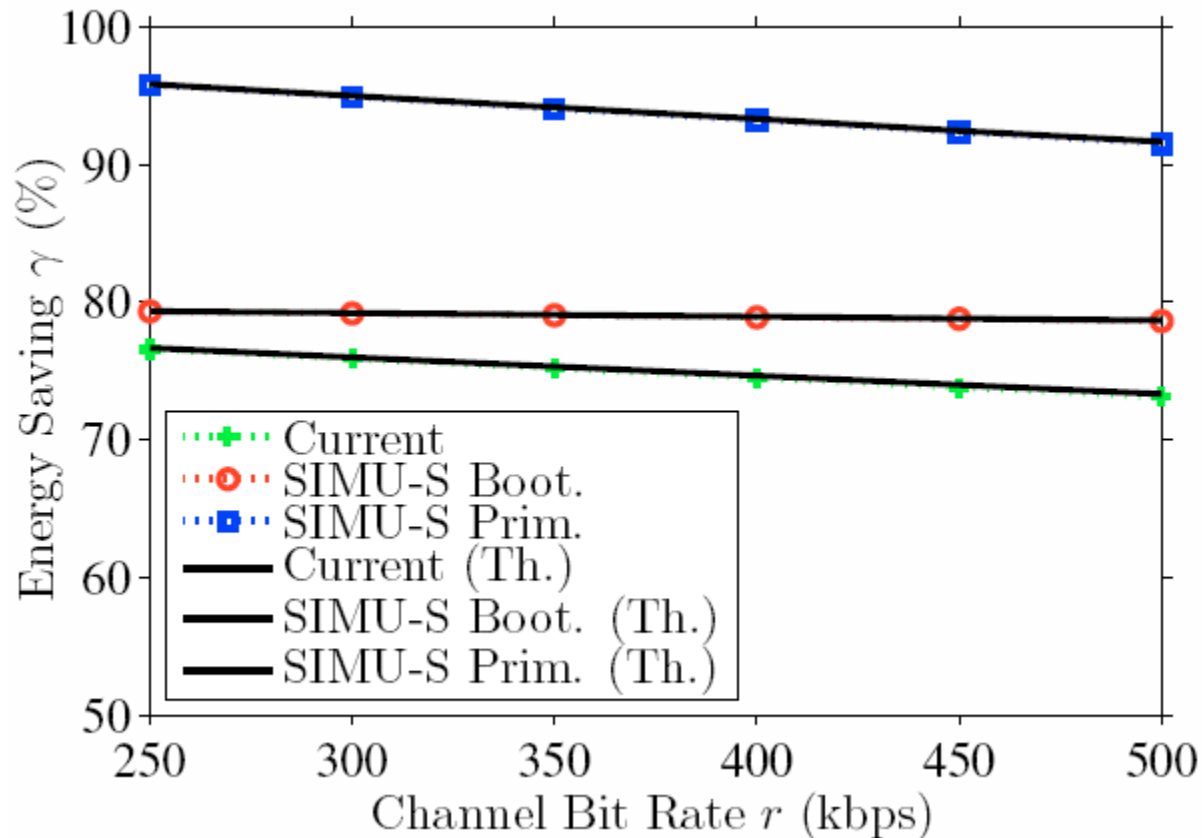
Correctness and Performance – SIMU/SIMU-S

- **Prove the scheme is feasible**
- **Show the scheme maximizes energy saving**
 - **First, show our scheme outperforms any scheme that does not employ simulcast idea**
 - **Then, show our scheme is optimal among all simulcast schemes**
- **Analytically derive energy saving**
 - **for devices receiving bootstrap bursts**
 - **for devices receiving primary bursts**
- **For details, see [Hsu 10, TOMCCAP]**

Simulation and Implementation in Testbed

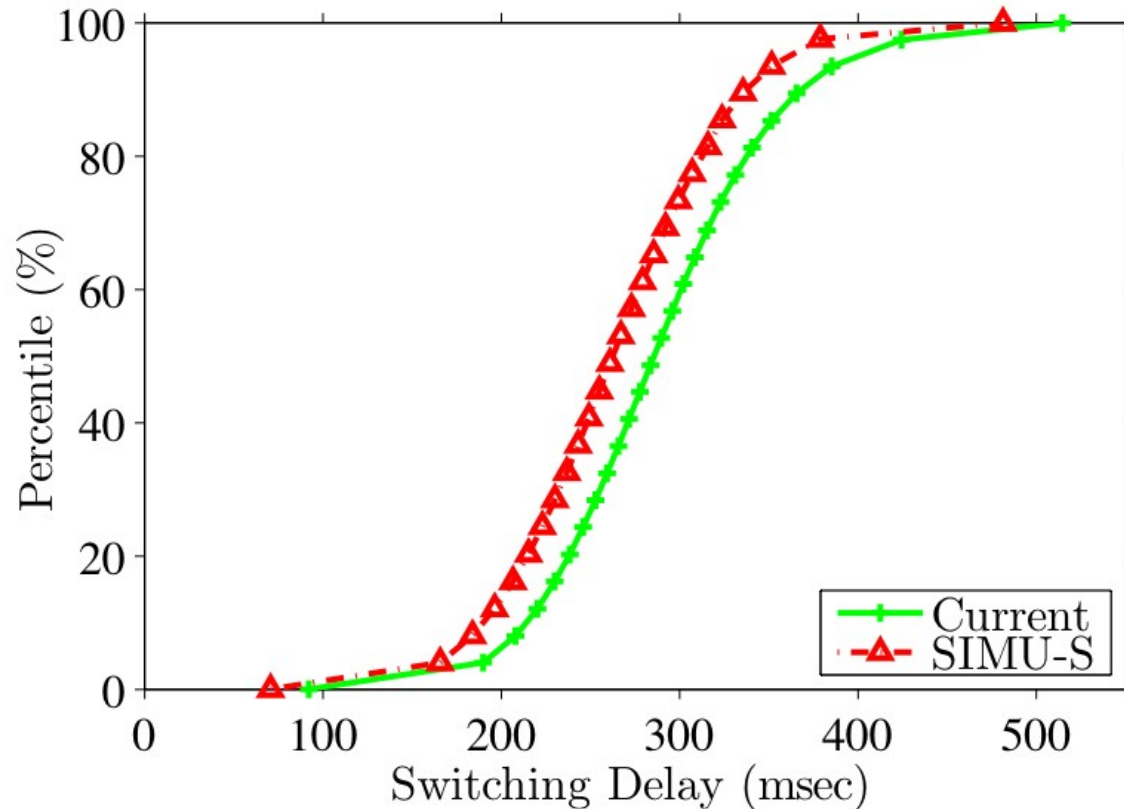
- Implemented SIMU-S scheme in C++ simulator
- Broadcast 8 TV channels for 10 min
- Simulate **1 million** users, randomly switching channels
 - let average watch time for each channel be 100 sec
- Set target switching delay 500 msec
- Compute switching delay and **weighted** energy saving
- Collect detailed logs that contain
 - time and size of each burst

Theory vs. Simulation



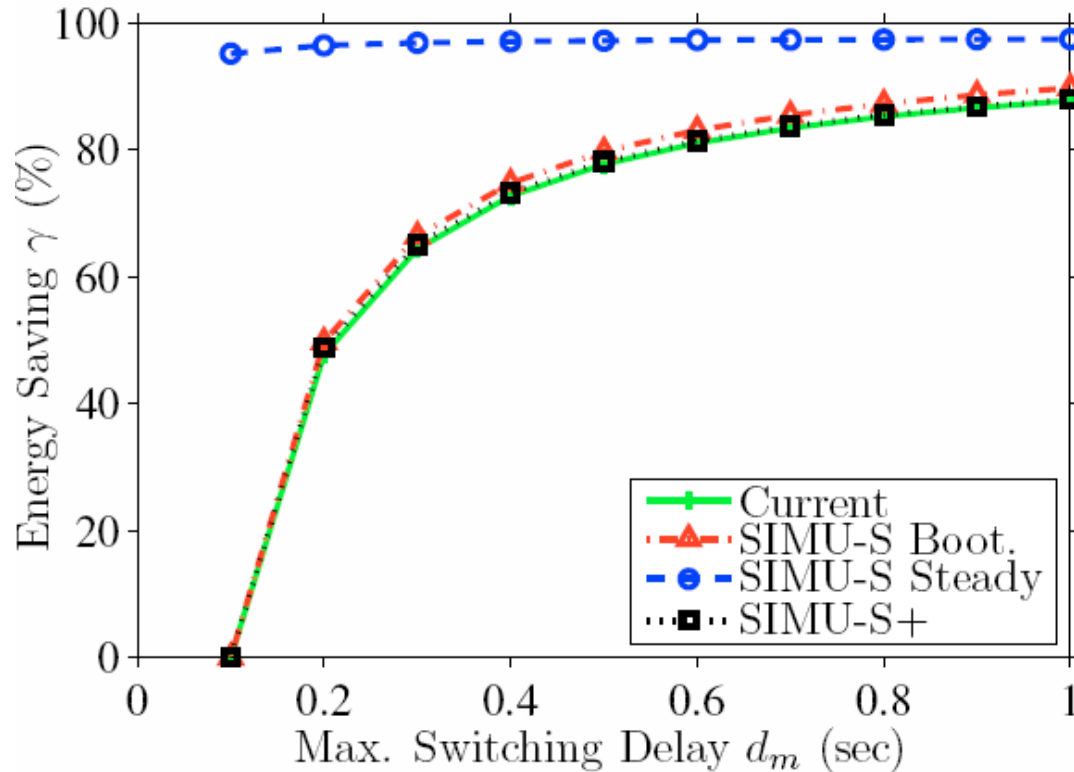
- Theoretical and empirical data match
- SIMU much better than Current

Channel Switching Delay



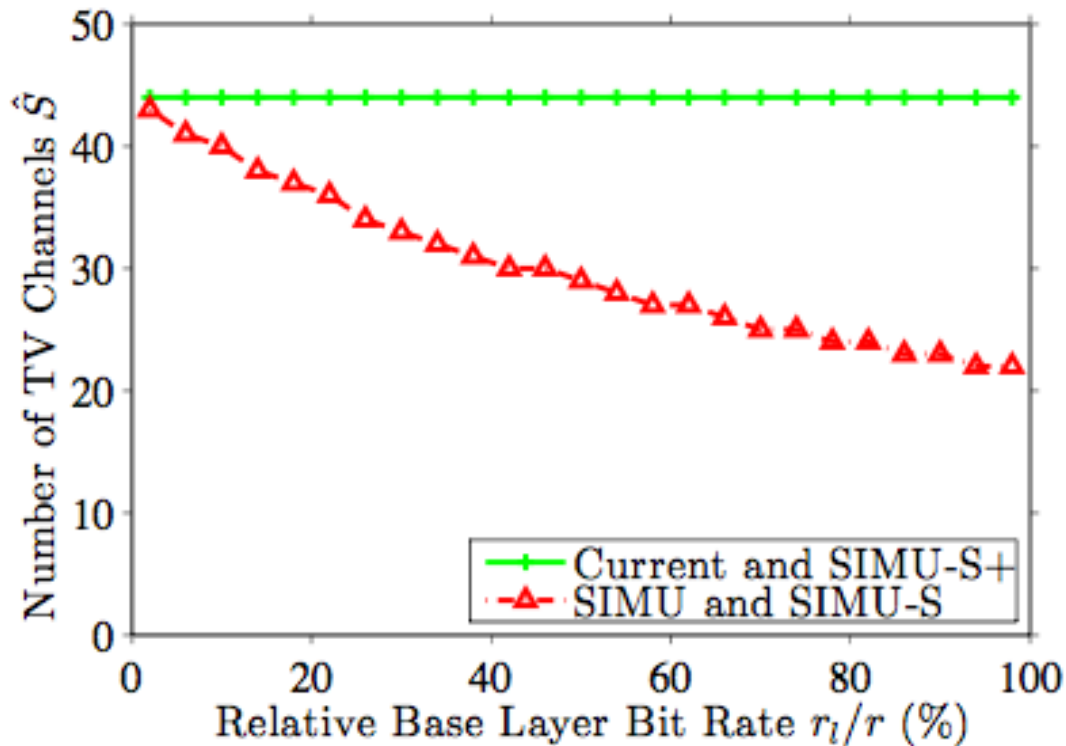
- **SIMU-S achieves the target switching delay bound**

Comparison on Energy Saving



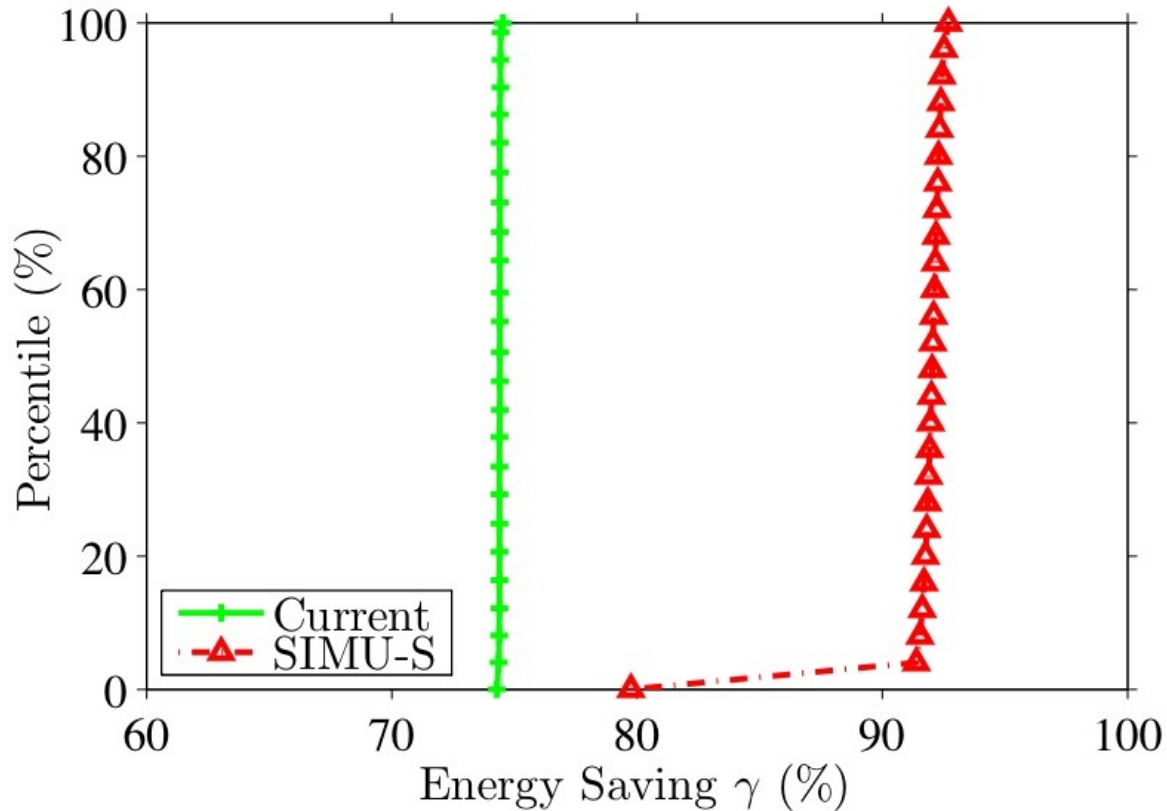
- **SIMU-S Primary: More than 95% energy saving**

Comparison on Network Utilization



- **SIMU/SIMU-S incur (controllable) BW overhead**
- **SIMU+ is BW efficient, but results in lower energy saving than SIMU/SIMU-S**

Energy Saving: From Testbed

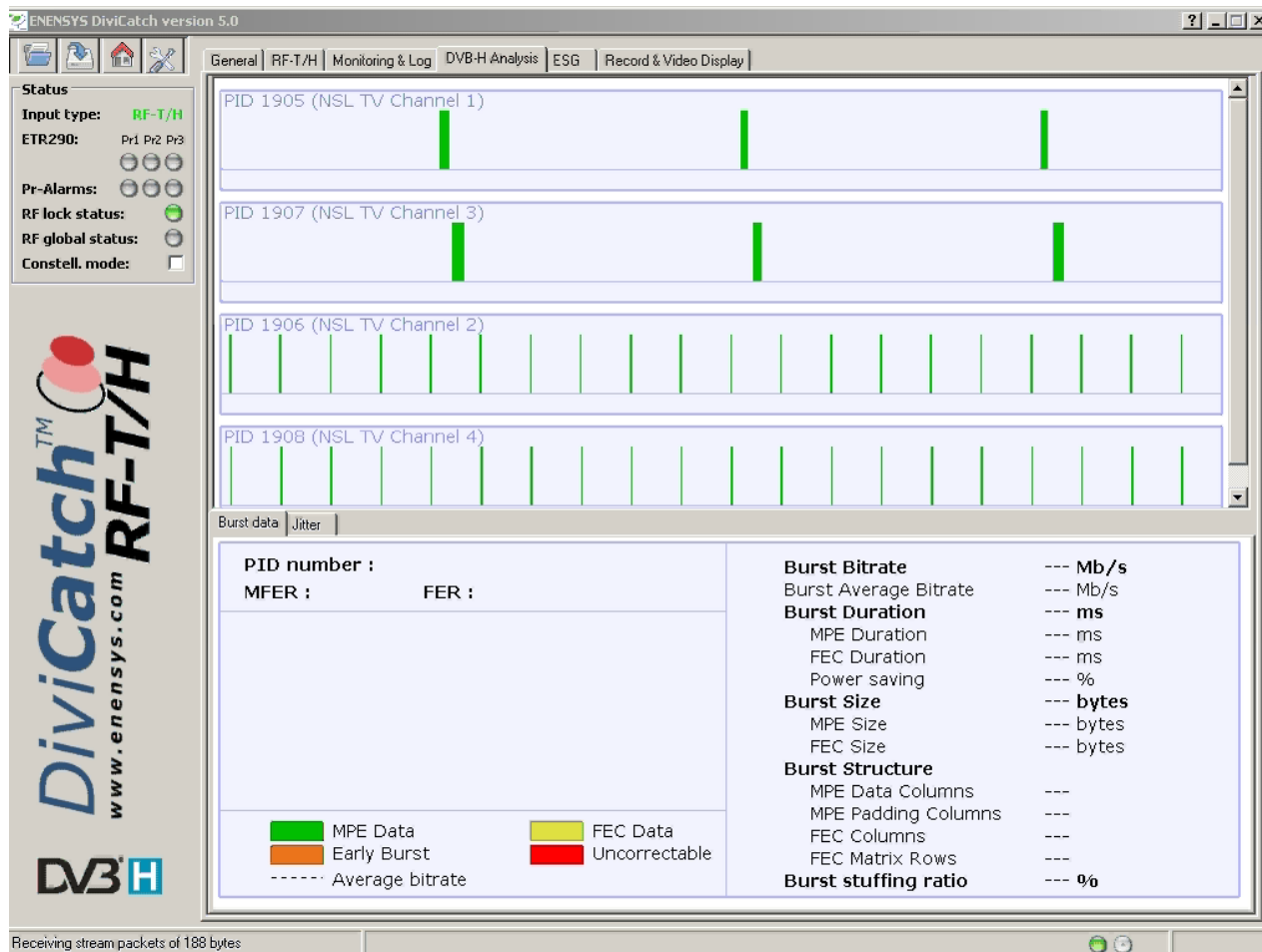


- **SIMU-S increases energy saving from 74% to 93% in real testbed**

Switching Delay: Summary

- **Controlling switching delay is important for users**
- **Proposed and analyzed three optimal (in terms of energy saving) video transmission schemes**
- **Validated in simulation and DVB-H testbed**
- **Demo (screen capture)**

Sample Video Shot from our Testbed



- **Burst analysis for SIMU: 2 primary & 2 bootstrap trains**

Deadline Oriented Packet Scheduling

Packet Scheduling

- Video streaming is real-time, each video packet has a deadline, which is determined by
 - timestamp of the corresponding video frame
 - initial buffering delay

- Video packets missing their deadline are **useless**

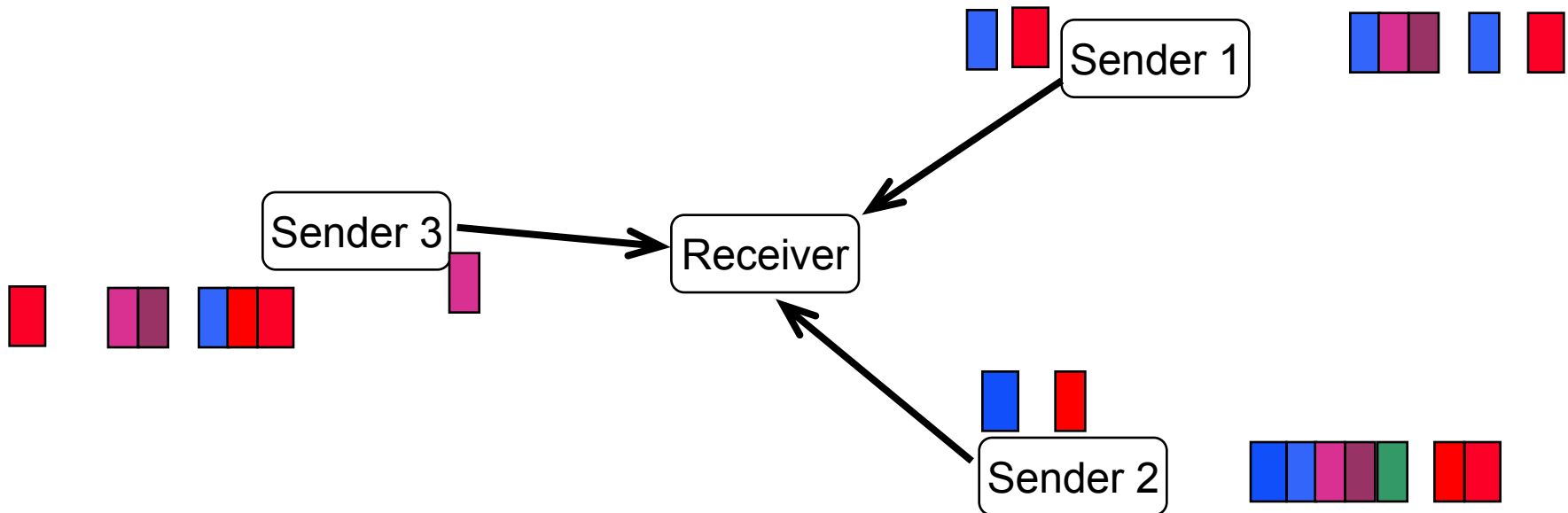
- Optimally scheduling packet transmissions is critical to perceived quality

Scheduling Algorithms

- Different network environments require different scheduling algorithms
- Huang et al. [Huang et al. 08'] consider the TDM (Time Division Multiplexing) scheduling problem in cellular networks
 - they propose earliest-deadline-first based algorithms to schedule packet transmission of multiple unicast streams
 - each video stream is sent from a mobile user to the base station (or vice versa)
 - they show the algorithm is optimal
- What if a video stream is sent from **multiple senders**?
 - for example, P2P video streaming?

Segment Scheduling in P2P Streaming

- Video is divided into segments
- Senders hold different segments
- A receiver runs a scheduling algorithm for a schedule
 - specifying which segments each sender should transmit
 - specifying the transmission time of each segment



Segment Scheduling Algorithm

- **Segment scheduling algorithm is important in both live and on-demand P2P streaming**
 - **only ontime delivered segments can be rendered to users for better video quality**
- **Recent studies, such as [Hei *et al.* ToM 08'], show that users suffer from long startup delays and playout lags, and suggest that better segment scheduling algorithms are required**
- **But, scheduling segments to maximize video quality is a hard problem**

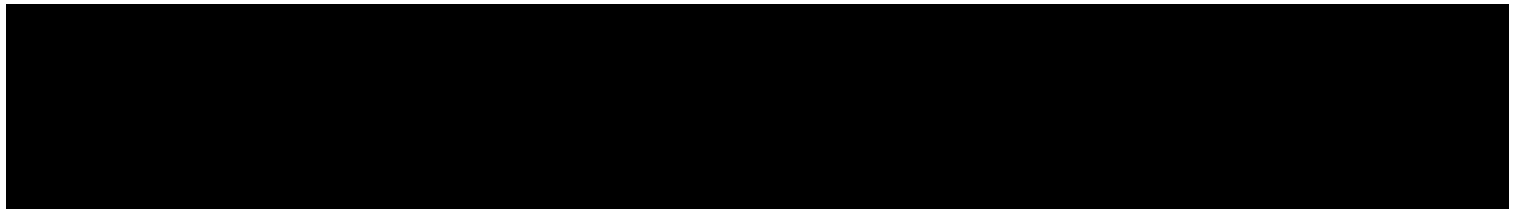
Scheduling Algorithms in Current Systems

- **Heuristic algorithms: random** [Pai *et al.* '05'], **rarest-first** [Zhang *et al.* '05'], and **weighted round-robin** [Agarwal and Rajaie '05]
 - they do not perform well in VoD services, nor do they provide performance guarantee

- **Solving simplified scheduling problem** [Chakareski and Frossard '09] [Zhang *et al.* '09]
 - for example, by defining ad-hoc utility function
 - may be optimally solved, but for a utility different from video quality

Modeling a P2P Streaming Session

- We consider a streaming session with M senders and one receiver [Hsu 10, MMSys]
- The videos are encoded at F frame per second
- Every G frames is aggregated into a segment n with size s_n , and the video consists of N segments
- Segment n has a decoding deadline $d_n = (n-1)G/F$
- The receiver maintains the segment availability info, we let



Modeling a P2P Streaming Session (cont.)

- The upload bandwidth b_m for sender m is periodically measured by a lightweight utility
- We let w_n be the weight/value of segment n , which can be in any quality metric, such as PSNR
- We periodically solve the segment scheduling problem every δ sec, which is a system parameter

Modeling a P2P Streaming Session (cont.)

- **Goal of our algorithm: construct an optimal schedule**
 $\{ \langle m_i, n_i, t_i \rangle, 1 \leq i \leq Q \}$ **for each scheduling window of δ sec,**
which indicates that sender n_i sends segment m_i at time t_i
 - A segment is ontime if
 - The sum of weights of all ontime segments is maximized

Formulation

- We divide the time axis into T time slots and define

$$x_{n,m,t} = \begin{cases} 1, & \text{if segment } n \text{ is sent by sender } m \text{ at time slot } t \\ 0, & \text{otherwise} \end{cases}$$

- We write the formulation

$$z^* = \max \sum_{m=1}^M \sum_{n=1}^N \sum_{t=1}^{d_n - \frac{s_n}{b_m}} w_n x_{n,m,t} \quad (1a)$$

(1a) Maximize sum of weights of ontime segments

(1b) Schedule a segment to a sender holding it

(1c) Schedule up to a segment for each time slot

(1d) Schedule each segment to at most one sender

An Optimal Solution

- We solve this formulation using ILP solvers, such as CPLEX
- **But**, solving ILP problems may take a long time
- Hence, we develop an approx. algorithm in the following

Our Approx. Algorithm -- Approach

- Relax the ILP formulation into an LP (linear programming) formulation
- Solve the LP problem using simplex or interior point methods for fractional schedule [REDACTED]
- Round the fractional solution [REDACTED] for integral solution with **performance bound**

Our Approx. Algorithm -- Rounding

- For each sender $m = 1, 2, \dots, M$, construct multiple integral schedules from the fractional schedule
- Then select the best schedule out of all integral schedules
- We schedule the segments in the best schedule to sender m , and remove these segments from the problem

Next m



Analysis of Our Algorithm

- **[Lemma 1] Our algorithm achieves approx. factor of 2 when there is only one sender**

Proof Idea: the way we create integral schedules guarantees that at least one of them achieves approx. factor of 2

- **[Theorem 2] Our algorithm achieves approx. factor of 3 when there are multiple senders**

Proof Idea: proved from the fact that we sequentially assign segments to senders

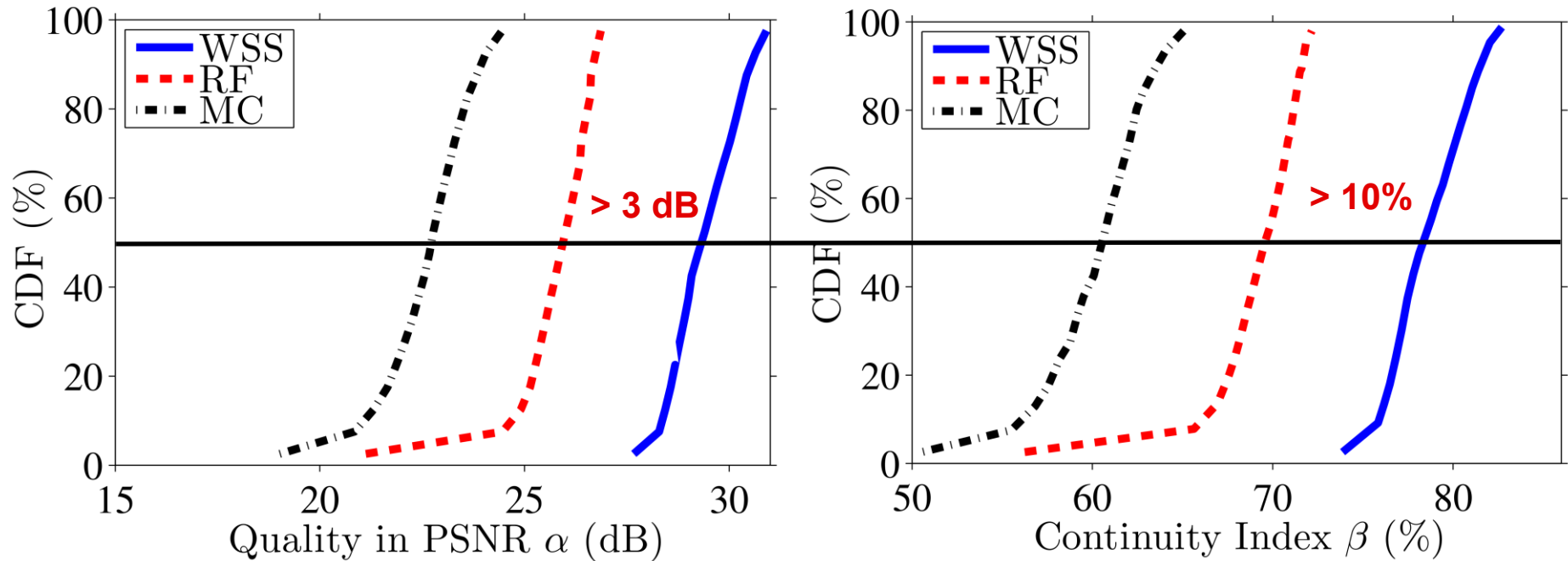
Evaluation

- **We implement a P2P simulator**
- **We implement four scheduling algorithms in it**
 - **OPT: ILP solver**
 - **WSS: our approx. algorithm**
 - **RF: rarest first**
 - **MC: mincost flow based**
- **We encode 10 videos into H.264 streams**
- **We simulate a P2P system with 2000 peers for 24 hours**

Evaluation (cont.)

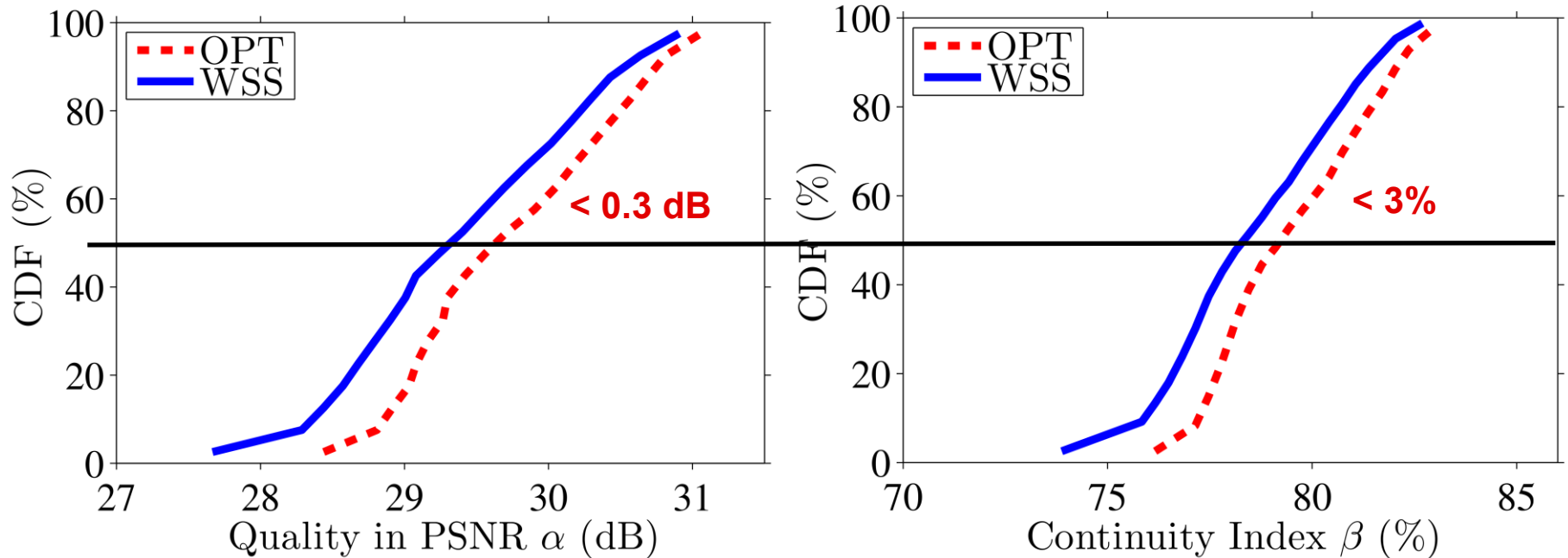
- Each peer connects to 10 senders
- Peers have realistic upload bandwidth [Liu *et al.* '08]
- Joining and leaving times are randomly chosen
- Considered two performance metrics
 - Average video quality in PSNR
 - Continuity index, which is the fraction of video frames arrive ontime

Comparison against Current Solutions



- **Better quality in PSNR: > 3 dB improvement**
- **Higher continuity index: > 10% difference**

Comparison against Optimal Solution



- Close to optimum performance under realistic system parameters

Summary of P2P Segment Scheduling

- **We presented an ILP formulation of this problem, and solved it using ILP solvers**
- **We proposed an approx. algorithm, and proved that it has an approx. factor of 3**
- **We evaluated our approx. algorithm in a P2P simulator**
 - **It outperforms algorithms used in current systems**
 - **It is almost-optimal with typical system parameters**

Cost Functions

- **Additive function** $C(\mathbf{d}) = \sum_u^U C_u(\mathbf{d}_u)$
 - $C_u(\mathbf{d}_u)$ **convex, increasing in each entry**
 - $C_u(\mathbf{d}_u) = \sum_{m=1}^{M_u} w_{u,m} d_{u,m}, w_{u,m} \geq 0$

- **Weighted Min-Max fairness**

$$C(\mathbf{d}) = \max_{u=1, \dots, U} w_u \sum_{m=1}^{M_u} d_{u,m}, w_u \geq 0$$

- **Simplification** : $M_u = M, Q_u = Q$

Multiple Packets

- NALU $g_{u,m,q}$ comprises $P_{u,m,q}$ packets
 - due to MTU (Max. Transmission Unit)
- Define $x_{u,m,q,p,n} = 1$, if the p -th packet of $g_{u,m,q}$ is sent over network n

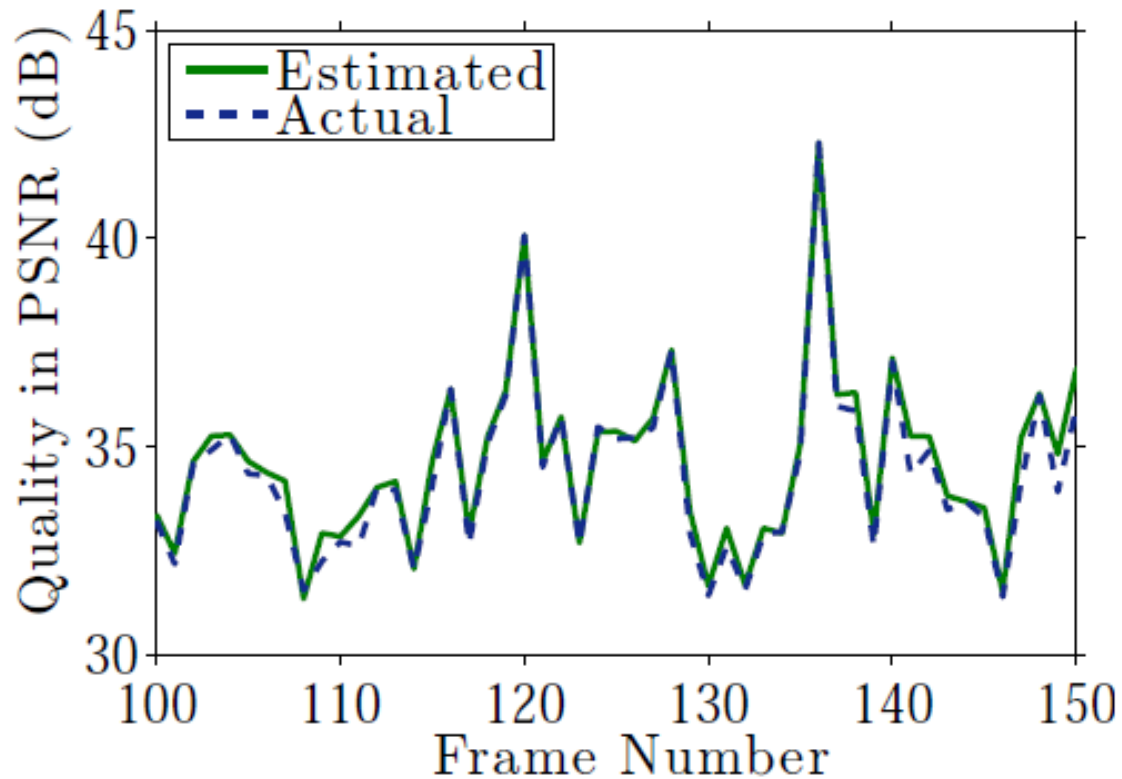
$$x_{u,m,q,p} = \sum_{n=1}^N (1 - p_n) x_{u,m,q,p,n}$$

$$e_{u,m} = \hat{\delta}_{u,m} + \sum_{q=0}^{Q_u} \left(1 - \prod_{q' \leq q} \prod_{p=1}^{P_{u,m,q'}} x_{u,m,q',p} \right) \delta_{u,m,q}$$

New Constraint

Video Model Accuracy

□ Samples from *Soccer*



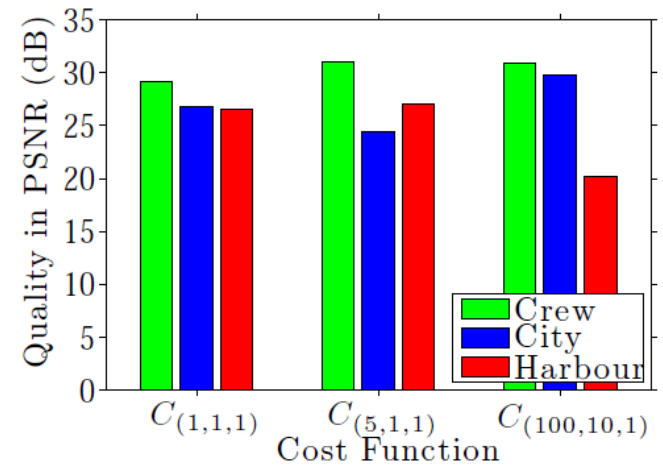
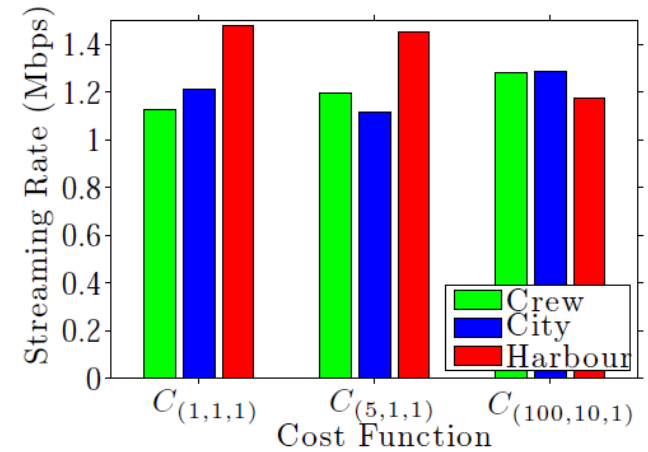
Service Differentiation

- Cost function (3 users) :

$$C_{(w_1, w_2, w_3)} := \sum_{u=1}^3 w_u \sum_{m=1}^{M_u} d_{u,m}$$

- Video quality

- Streaming rate



Summary of Multihomed Streaming

■ Modeling

- Video quality model for H.264/SVC streams
- Network model for proactive rate control

■ Joint rate-control and distortion optimization for multiple clients

- Cost minimization
- Heuristic algorithms
- Convex programming approximations

■ Simulations

- Model accuracy
- 10 dB quality improvement over DCCP
- TCP-friendliness of our algorithms
- Significant delay reduction (~80%)
- HC outperforms the heuristics, and is suitable for real-time implementation
- Service differentiation