Virtualization of Network Functions for Bandwidth-Adaptive Video Content Delivery

[ONAP perspective]

Project managed by:

Kaliappa Ravindran and **Arun Adiththan** **

Dept. of Computer Science

City University of New York (CUNY – City College & Graduate Center)

Student participants at CUNY – City College & GC

Joseph Shaker, Ashik Acchu, Tian Chen, Rbecca Batat

Government Lab Collaborator: **US Air Force Research Lab**, Rome, NY
[Amjad Soomro, Matthew Anderson]

Potential Industry Collaborators AT&T Research, Verizon, . . .

[** Research Scientist (General Motors, June'18 - _)]

Organization of talk

1. Past works at CUNY on management of adaptive network services

   Granularity of management:

   service-level, protocol-level, parametric-level

2. Programmable adaptation processes in video transport for ONAP

   Modular structuring of rate controllers

   --- model-predictive versus model-oblivious control

   --- Rate control on aggregated flows

   --- User-participatory control (e.g., QoS dashboard)

3. Deployment scenario in a CDN (content delivery network)

Motivation for development of adaptive network systems with ONAP-style structuring

1. Enhanced performance in a variety of traffic scenarios

2. Resilience against harsh environment conditions

dynamically varying external conditions
(hard-to-observe or predict)

ONAP-style system structures enable a ***decision-engine*** to orchestrate adjustment of system objects at desired granularity

PAST WORKS AT CUNY ON
NETWORK ALGORITHMS AS SOFTWARE OBJECTS

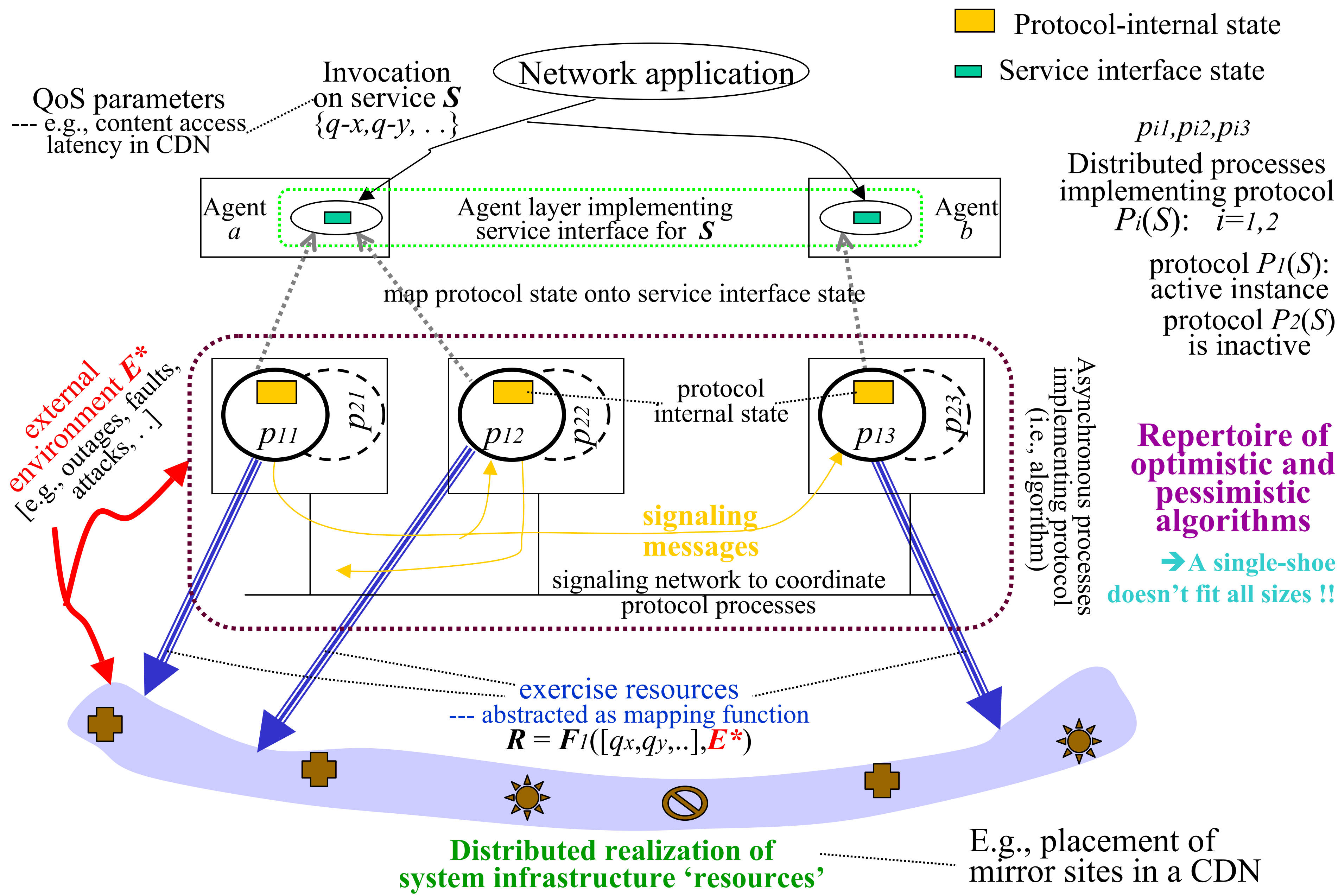
[for enhanced network performance and resilience]

<u>Published papers in:</u>

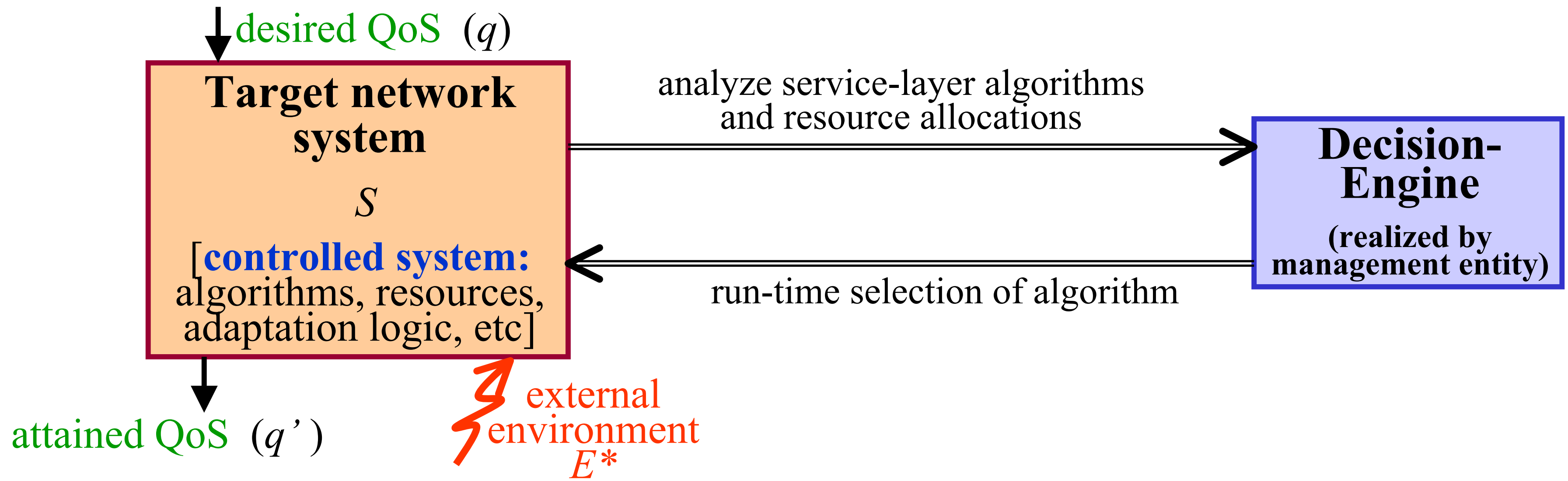
NOMS, IM, CNSM, IPDPS, DRCN, ISSE, COMSNETS, . . . (2006 – 17)

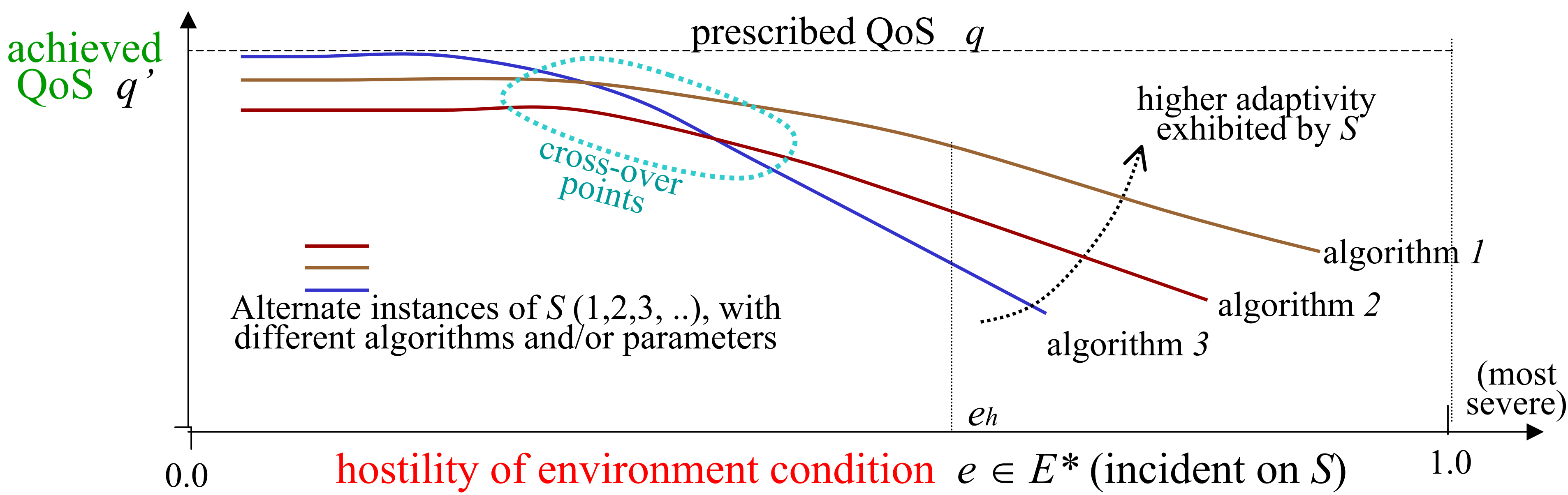
"Distributed Protocols" (algorithms) as virtualized network functions

[enables on-the-fly switching of algorithms by a decision-engine]



Management control of quality of QoS support mechanisms





System instance 1 is more resilient than instance 2, in dealing with environment condition e_h;
Instance 2 is more resilient than instance 3. ➔**A single-shoe doesn't fit all sizes !!**

Why ONAP-style system structures ??

Verifiable **guarantees of system performance and resilience**

(needed for mission-critical applications: such as DOD, NASA)

A system that is good
but is not *verifiably good* is
not good enough !!

[e.g., quality rating of restaurants, hotels, taxi services, . . .]

3-star, 4-star, etc

Underlying service-layer algorithms should be:

Malleable, ***Programmable***, and ***Quantifiable***

System responsiveness to external environment

QoS specs *q*, algorithm parameters *par*,
system resource allocation *R* are usually controllable inputs

In contrast, environment parameters $e \in E^*$ are often uncontrollable and/or unobservable, but they do impact the system-level performance (e.g., component failures, network traffic fluctuations, attacks, etc)

environment parameter space:

$$E^* = E(\text{yk}) \cup E(\text{nk}) \cup E(\text{ck})$$

- $E(\text{yk})$: parameters that the designer knows about (known "knowns")
- $E(\text{nk})$: parameters that the designer does not currently know about (knowable "unknowns")
- $E(\text{ck})$: parameters that the designer can never know about (known "impossibilities")

What about unknown "unknowns" ?? → Hon. D. Rumsfeld

Algorithm design decisions face this uncertainty --- so, designer makes certain assumptions about the environment (e.g., at most 2 nodes will fail during execution of a data replication algorithm). When assumptions get violated, say, due to attacks, algorithms fall short of what they are designed to achieve

ONAP structure allows evaluating how good an algorithm performs in strenuous conditions

Network Function Virtualization (NFV) for Video Content Delivery to End-users

ONAP perspective

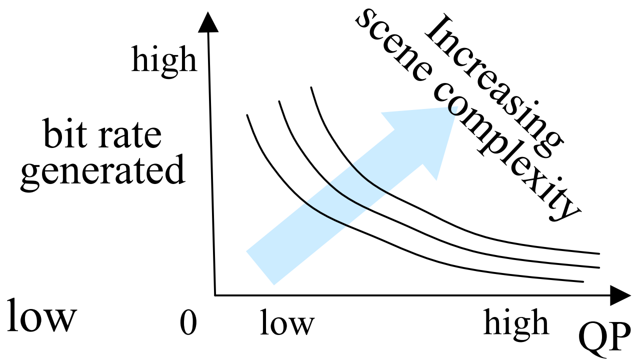
Adjustment of quantization parameter (QP) to control video bit-rate

Take a macro-block and encode it with a certain QP

Low QP ➔ distortion (D) in comparison to the original image will be low, but the bit rate (R) will be high

Choose a high QP

➔ distortion will be high, but the bit-rate will be low



higher QP ➔ (increased distortion & reduced quality)

As quantization QP → 0(+), encoder rate λ → 251 *mbps*

[lossless compression ➔ best visual-quality]

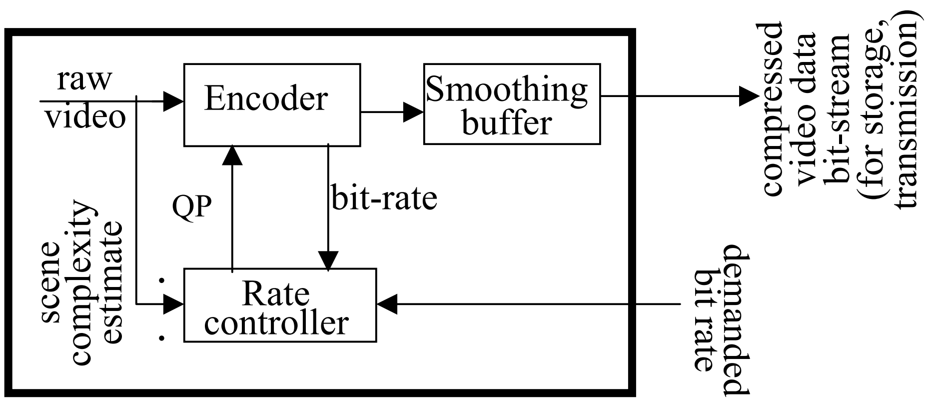
Higher QP ➔ lower bit rate (and hence lower visual-quality)

Typical range of QP used during no congestion: 28-35

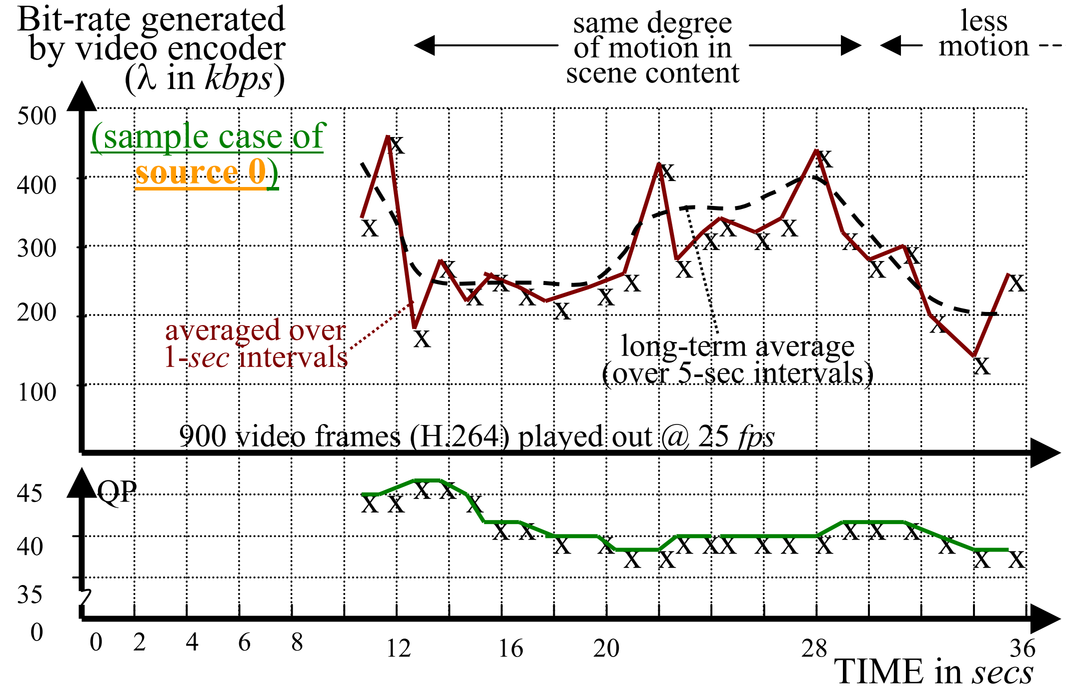
visual-quality (VQ) is a user-oriented subjective parameter

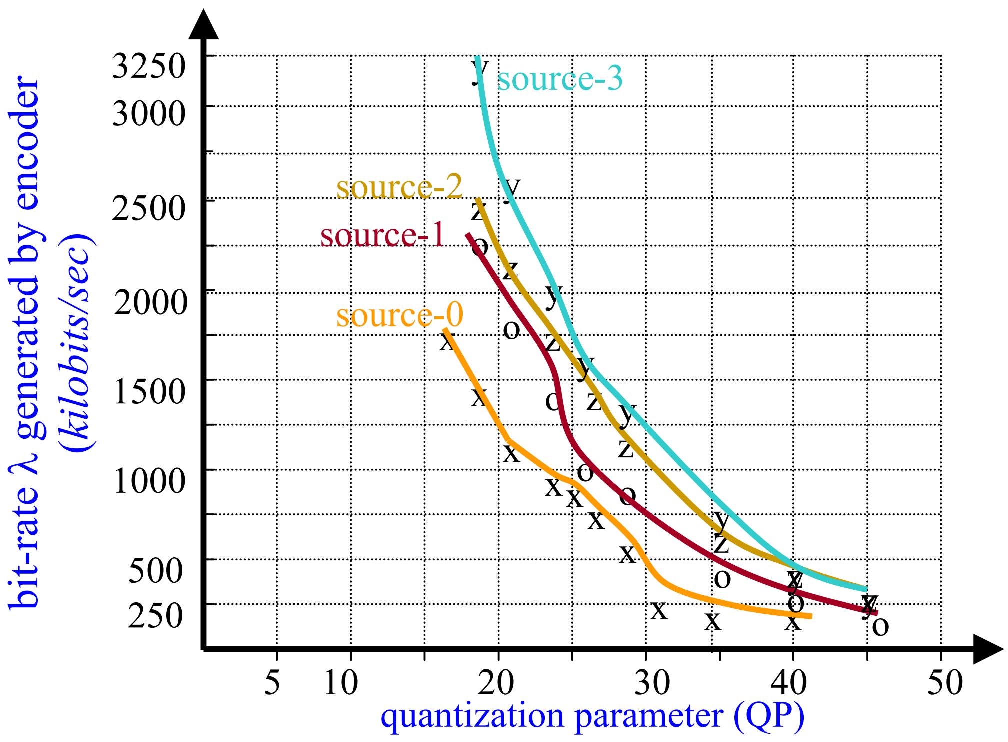
➔ can be categorized in decreasing order, say: [BEST, BEST(-), BETTER, GOOD(+), GOOD, GOOD(-), BAD]

Internal structure of H.264 video encoder (typical)

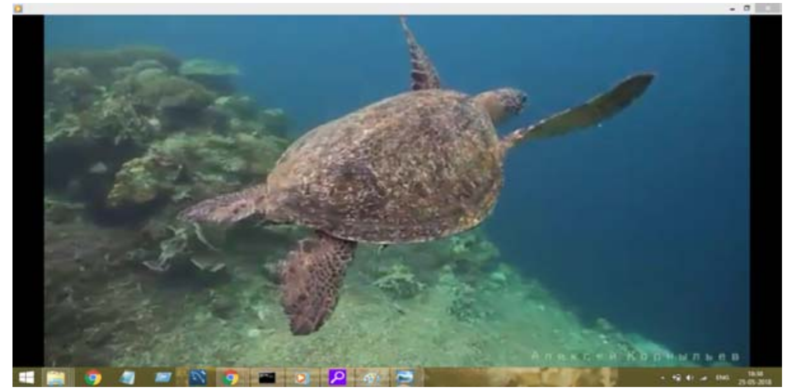


Experimental results collected on sample video sources (with FF-MPEG software) show rate burstiness and variability





Source-1: QP=26, visual_quality = GOOD



Source-2: QP=28, visual_quality = BETTER

Source-3: QP=28, visual_quality = BETTER

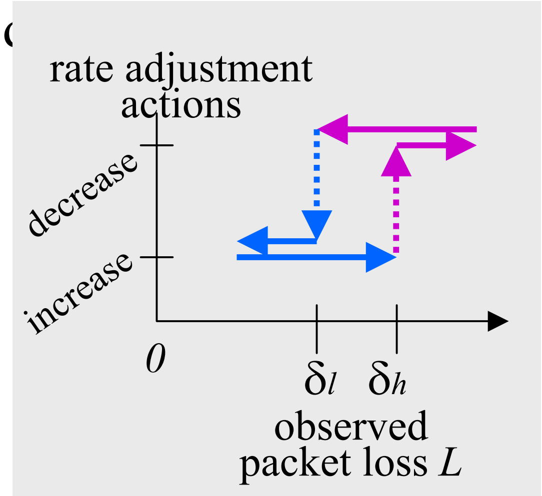
Iterative adjustment of video bit-rate to effect congestion-relief

Additive Increase Multiplicative Decrease (AIMD)

[CDIOT(1994), VBHARGAVAN(2001), JKUROSE(1998), . . .]

“available bandwidth” on a transport path is unknown

In each interval for ‘packet-loss reporting’, adjust send rate of c



$$\lambda_{(\text{new})} = \lambda_{(cur)} - \beta . L \quad \text{when} \quad L > \delta_h,$$

where $\beta > 0$

$$\lambda_{(new)} = \lambda_{(cur)} + \alpha \quad \text{when} \quad L < \delta_l,$$

where $\alpha > 0$

L: observed “packet loss ratio”

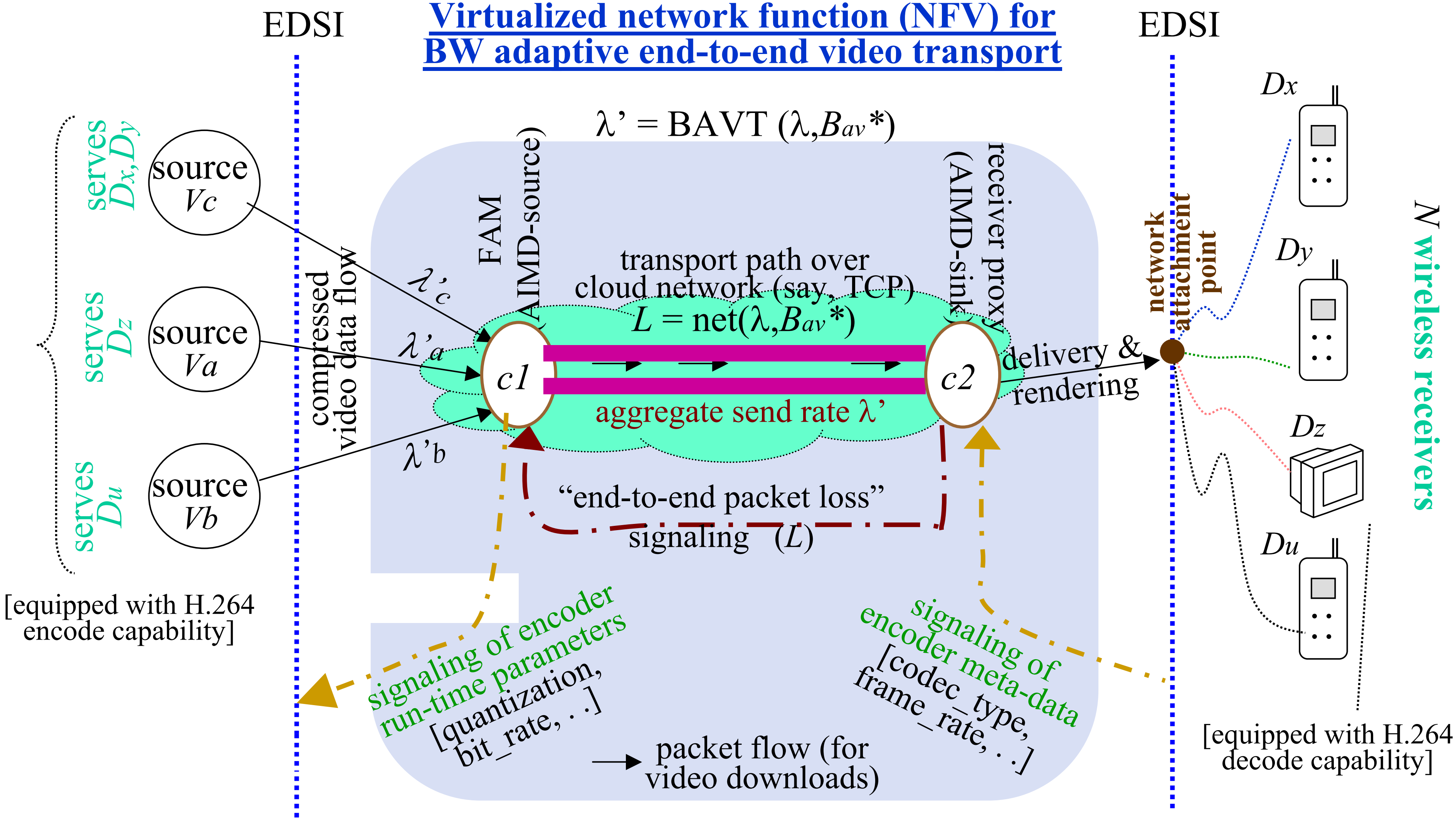
δ_l, δ_h: Acceptable loss thresholds

[$\delta_l < \delta_h$ to avoid ping-pong effect]

^^ Each execution of this procedure constitutes a “**control iteration**”

^^ A sequence iterations that lead to a steady-state in bandwidth usage (when the bit-rate specs change or new video flows are admitted) constitutes a “**control round**”

Virtualized network function (NFV) for BW adaptive end-to-end video transport



reduced signaling overhead: $O(N)$

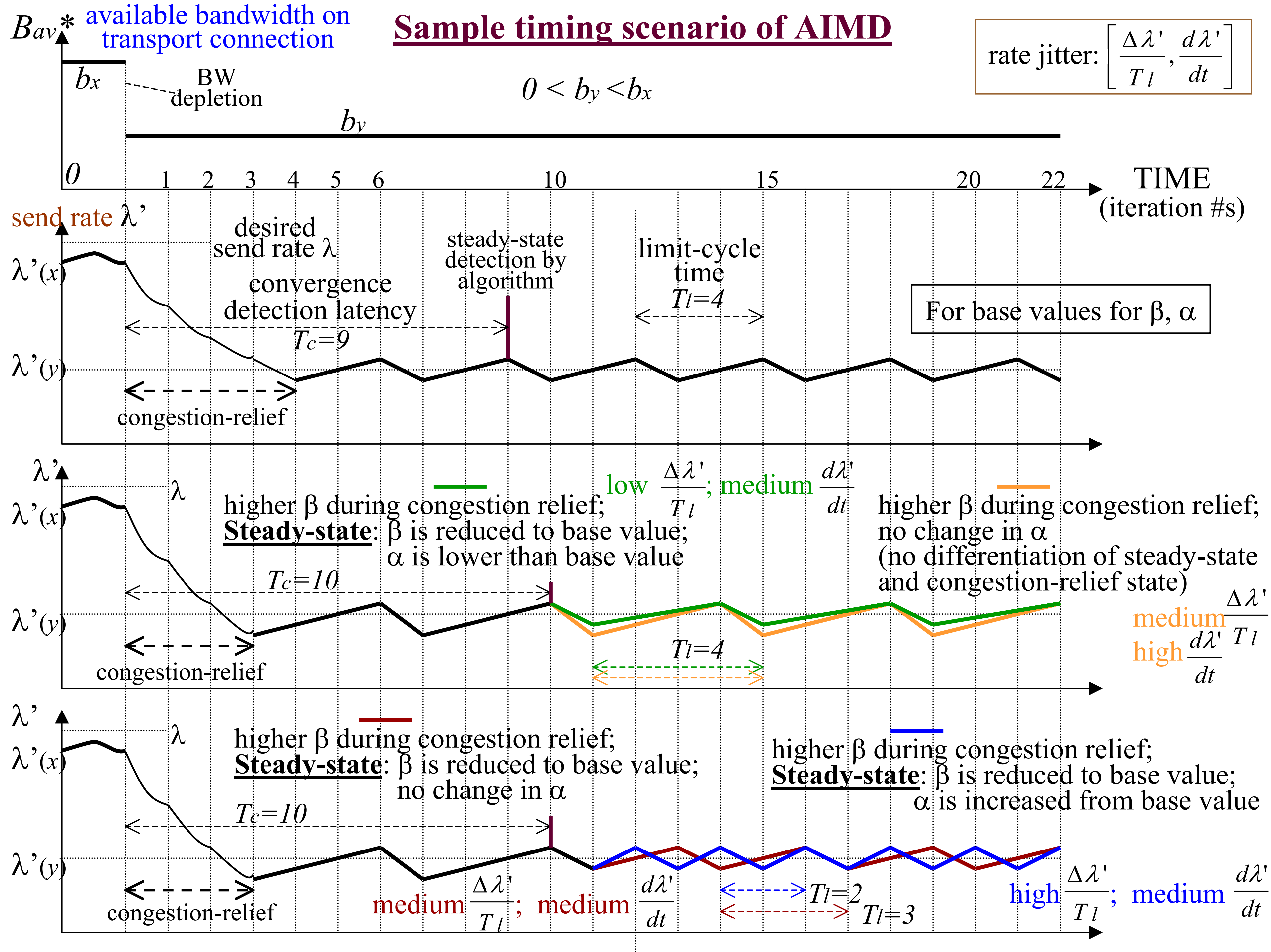
FAM: flow aggregation module (packet mux/demux, AIMD-based rate control, . .)

EDSI: end-user device signaling interface

AIMD-computed aggregate data send rate λ' (*bps*) over one or more control epochs such that $L < \delta$: say, δ=0.007

λ' is determined from the current send rate λ and available bandwidth B_{av}*

λ' is split as λ'_a , λ'_b , λ'_c at sink end-point



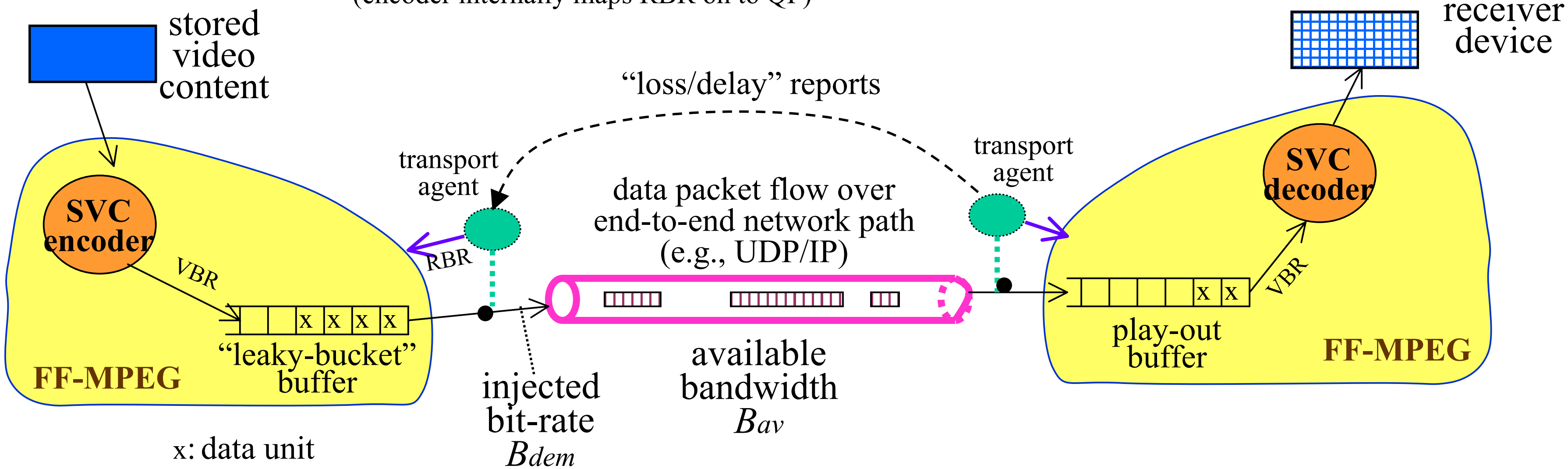
Sample timing scenario of AIMD
Bav* available bandwidth on transport connection
rate jitter: [Δλ'/Tl, dλ'/dt]
bx
BW depletion
0 < by < bx
by
0
1 2 3 4 5 6 10 15 20 22
TIME (iteration #s)
send rate λ'
desired send rate λ
steady-state detection by algorithm
limit-cycle time Tl=4
λ'(x)
convergence detection latency Tc=9
For base values for β, α
λ'(y)
congestion-relief
λ'
λ
low Δλ'/Tl; medium dλ'/dt
higher β during congestion relief;
Steady-state: β is reduced to base value;
α is lower than base value
higher β during congestion relief;
no change in α
(no differentiation of steady-state and congestion-relief state)
Tc=10
medium Δλ'/Tl
high dλ'/dt
Tl=4
congestion-relief
higher β during congestion relief;
Steady-state: β is reduced to base value;
no change in α
higher β during congestion relief;
Steady-state: β is reduced to base value;
α is increased from base value
Tc=10
Tl=2
Tl=3
medium Δλ'/Tl; medium dλ'/dt
high Δλ'/Tl; medium dλ'/dt
congestion-relief

Virtualized Network Functions (VNF) Pertinent to Video Content delivery Under BW Constraints

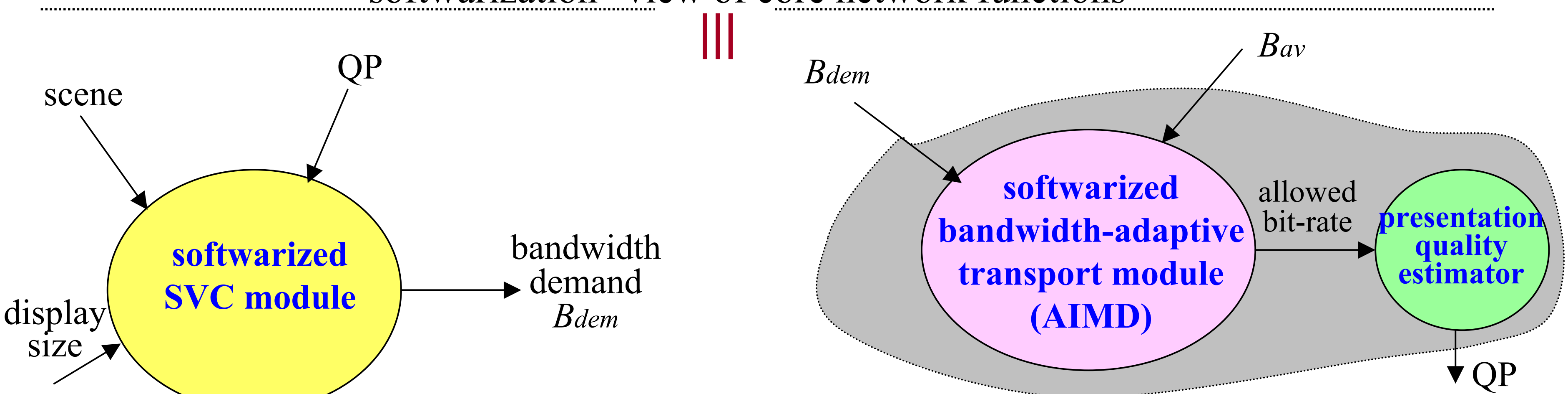
SVC: scalable (fine-granular) video control

RBR: agent-allocated bit-rate for source
(encoder internally maps RBR on to QP)

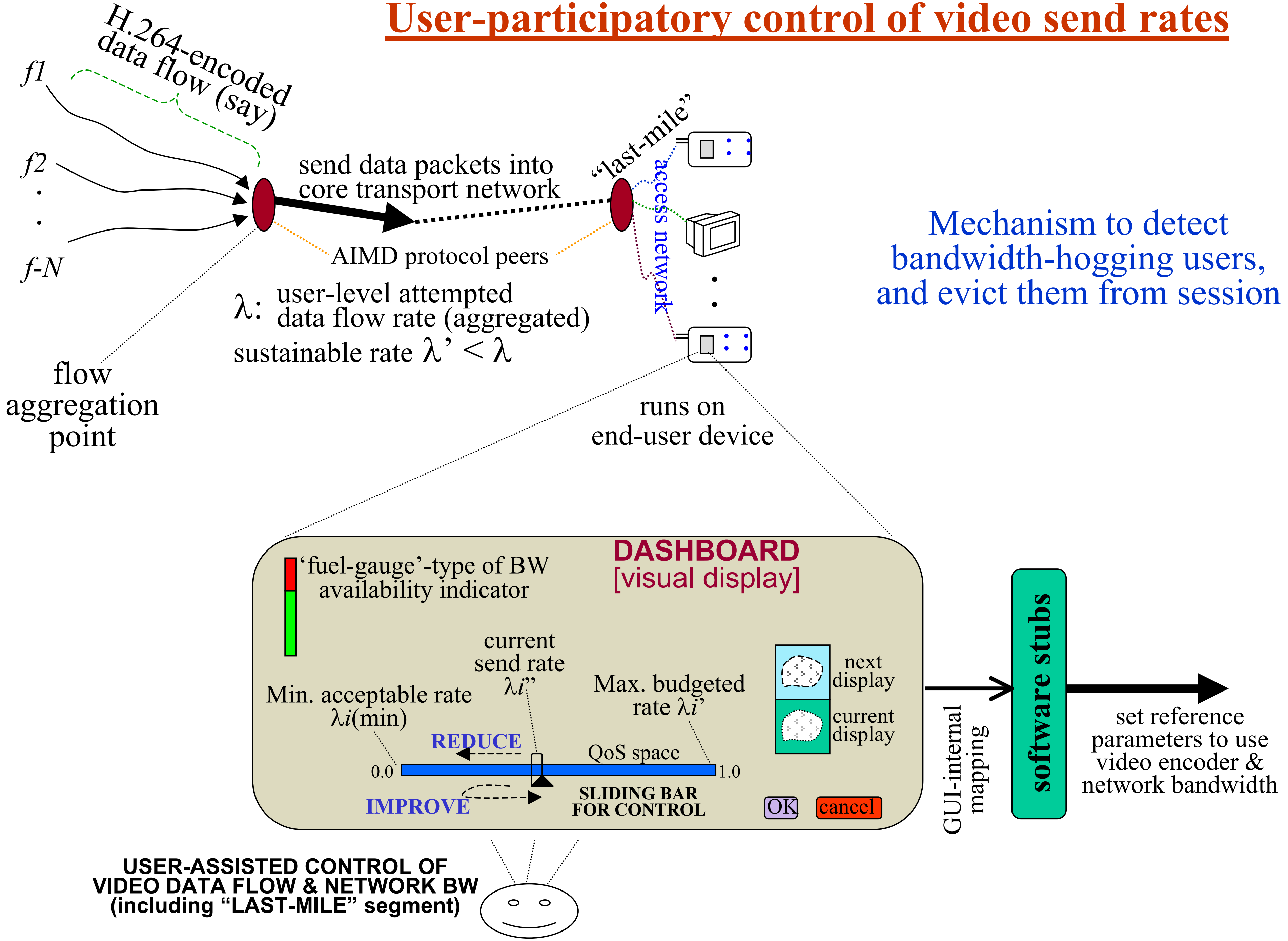
VBR: variable bit-rate output

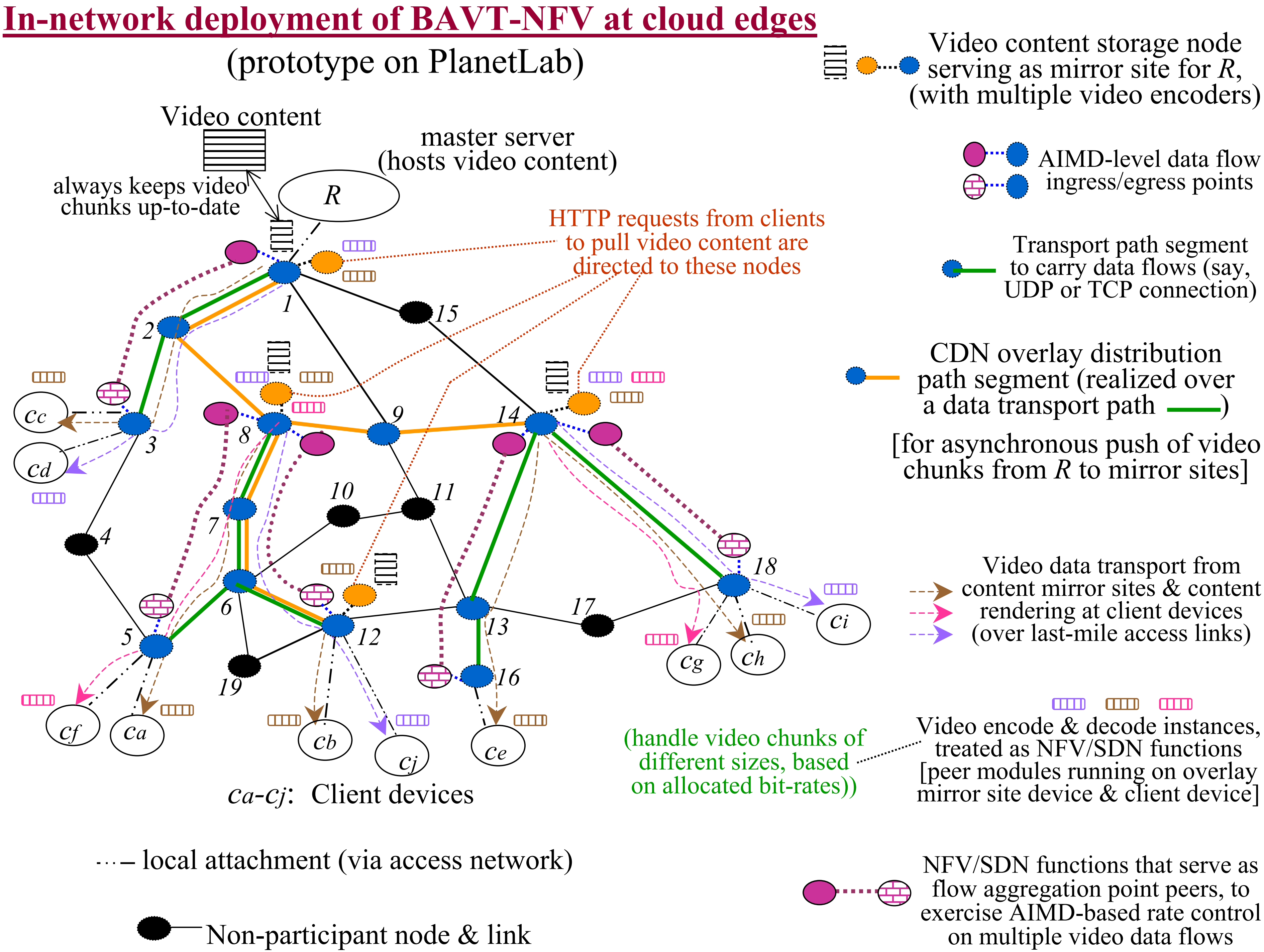


"softwarization" view of core network functions



User-participatory control of video send rates





In-network deployment of BAVT-NFV at cloud edges
(prototype on PlanetLab)
Video content storage node serving as mirror site for R, (with multiple video encoders)
Video content
master server (hosts video content)
R
always keeps video chunks up-to-date
AIMD-level data flow ingress/egress points
HTTP requests from clients to pull video content are directed to these nodes
Transport path segment to carry data flows (say, UDP or TCP connection)
CDN overlay distribution path segment (realized over a data transport path)
[for asynchronous push of video chunks from R to mirror sites]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
ca
cb
cc
cd
ce
cf
cg
ch
ci
cj
Video data transport from content mirror sites & content rendering at client devices (over last-mile access links)
(handle video chunks of different sizes, based on allocated bit-rates))
Video encode & decode instances, treated as NFV/SDN functions [peer modules running on overlay mirror site device & client device]
ca-cj: Client devices
local attachment (via access network)
NFV/SDN functions that serve as flow aggregation point peers, to exercise AIMD-based rate control on multiple video data flows
Non-participant node & link