# A Resource Allocation Controller for Cloud-based Adaptive Video Streaming

**Luca De Cicco**, Saverio Mascolo, Dario Calamita

Politecnico di Bari, Dipartimento di Ingegneria Elettrica e dell'Informazione

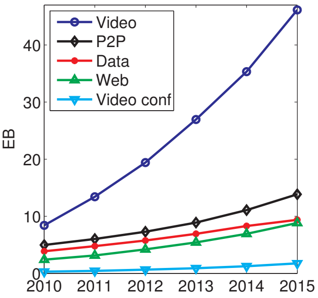
MCN 2013 - Budapest, Hungary

13 June 2013

# Motivation

## Two ongoing trends (Cisco VNI)

- *Video is booming*: video applications today account for more than half of the global traffic
- *Mobile is growing*: mobile data traffic will be half of global traffic in 2017




YouTube Broadcast Yourself™
NETFLIX
Akamai HD Network
hulu

# The challenge

## Main Goal

Design a cloud-based platform for massive distribution of adaptive videos

# The challenge

## Main Goal

Design a cloud-based platform for massive distribution of adaptive videos

## Issues

1. Bandwidth is unpredictable in best-effort Internet
2. Mobile devices have limited CPU and display resolution
3. User demand is highly time-varying

# The challenge

## Main Goal

Design a cloud-based platform for massive distribution of adaptive videos

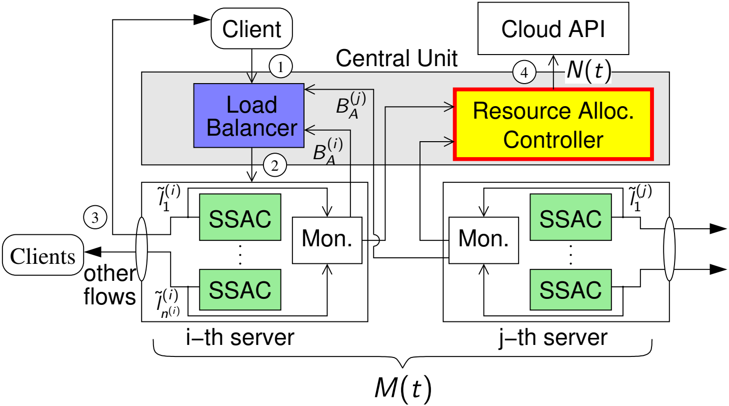
## Issues

1. Bandwidth is unpredictable in best-effort Internet
2. Mobile devices have limited CPU and display resolution
3. User demand is highly time-varying

## Design Goals

1. *Issues 1 and 2* ⇒ **Implement video adaptivity**
2. *Issue 3* ⇒**Resource Allocation** to dynamically turn on/off servers

# The proposed Control Plane




## Architecture

- One Central Unit
- $M(t)$ servers

## Controllers

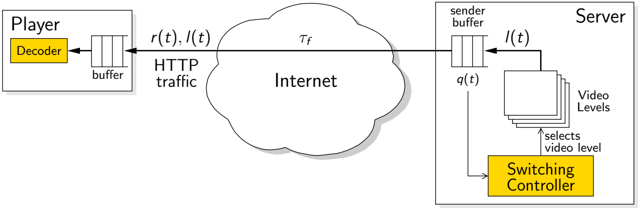
- Stream Switching Adaptation Controller (per-flow)
- Load balancer (centralized)
- Resource Allocation Controller (centralized)

# Stream Switching Adaptation Controller

## Stream-switching approach

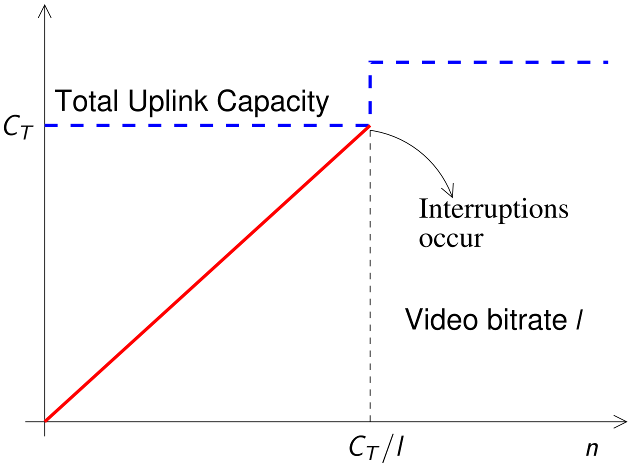
The video is available at different resolutions and bitrates, a controller selects the video to be streamed

## Quality Adaptation Controller (QAC) - ACM MMSYS 2011




- Adaptation logic is executed at the **server** (in the Cloud)
- The video flow behaves as any TCP greedy flow
- Fairness is inherited by TCP congestion control

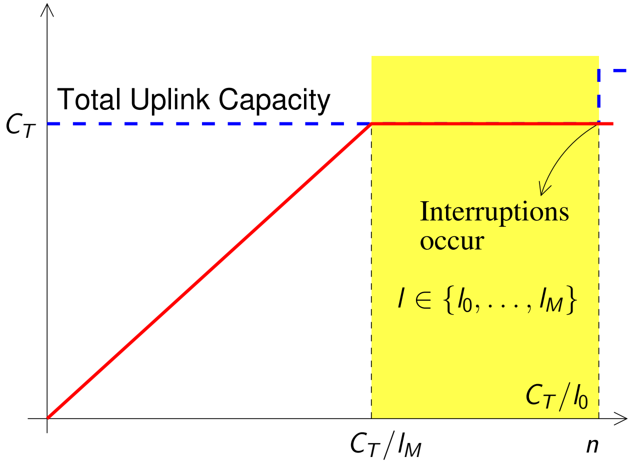
# Inelastic videos




**Fact**

If video is not adaptive, the delivery network **must be always overprovisioned** to prevent playback interruptions

# Elastic videos




- We can **work at** 100% **uplink channel utilization**
- **But**: users will not receive the maximum video level anymore
- **Action**: increase the number of servers to increase uplink capacity

# Why flows do not get the maximum video level?

Where's the bottleneck?

1. At the Server. **Can act** on these flows by turning ON machines.
2. At the Client. Cannot act on these flows (threated as a disturbance)

# Why flows do not get the maximum video level?

Where's the bottleneck?

1. At the Server. **Can act** on these flows by turning ON machines.
2. At the Client. Cannot act on these flows (threated as a disturbance)

- The goal of the RAC is to steer to zero the number of uplink-limited flows $n_{UL}(t)$
- We need to estimate $n_{UL}(t)$

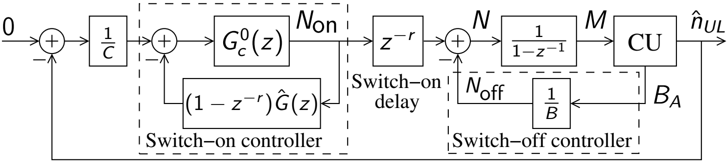
# limited flows = # uplink-limited flows + # client limited flows

$$n_L(t) = n_{UL}(t) + n_{CL}(t)$$

- The CU measures $n_L(t)$ easily
- A variable threshold mechanism estimates $n_{CL}(t)$ (details in the paper)

# The Resource Allocation Controller

- **Switch-on Controller**: steers $\hat{n}_{UL}(t)$ to zero (control-loop set point)
- **Switch-off Controller**: turns off servers when the goal of the switch-on controller is reached




## Switch-on controller

- PD controller: $G_c^0(z) = K_p + K_d(1 - z^{-1})$
- The Smith predictor compensates the effect of the switch-on delay
- The model used in the SP is an integrator (tf from N to M)

## Switch-off controller

It turns off (if $N_{on} = 0$) a number of machines equal to $B_A/B$

# Simulations

## Simulator

- based on CDNSim
- implements the control modules and a module monitoring CPU costs

## Metrics

- Fraction of flows obtaining the maximum level: $\alpha(t) = 1 - n_L(t)/n(t)$
- *Total Servers costs* $C_c(t)$

## Considered controllers

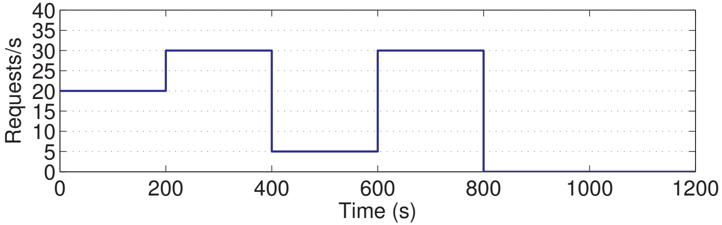
- *The proposed PD controller* with $K_p = -0.7$, $K_d = -0.3$
- The proposed controller without the Smith predictor
- *Feed forward controller*: $N(t_k) = n(t_k)/C - M(t_k)$ (difference between the number of servers that should be ON to provide maximum quality and the number of active server)

# Scenarios

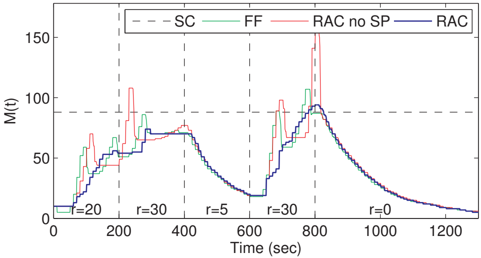
## Scenarios

- Client downlink is not the bottleneck $\Rightarrow n_{CL}(t) = 0$
- 16% of users have a downlink channel not allowing maximum video level ($n_{CL}(t) \neq 0$):

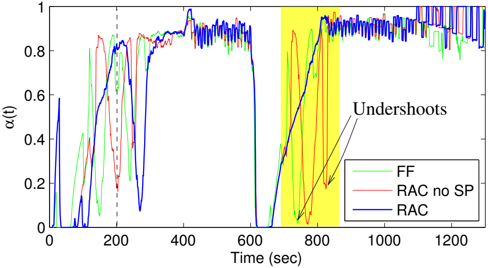
## Request arrival (Poisson with variable intensity $r(t)$)



# Results: client limited flows ($\hat{n}_{CL} = 0$)



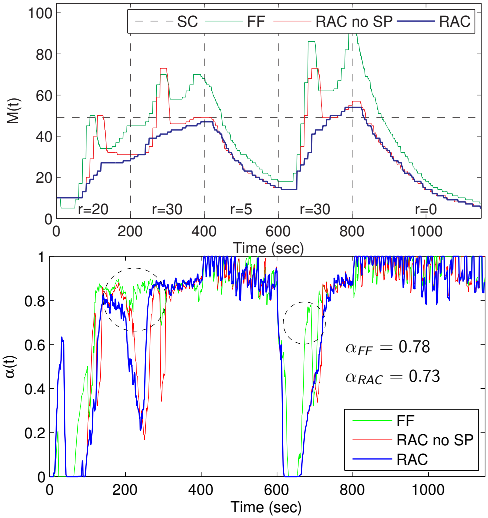

- Number of active servers over time is smooth with RAC
- Other controllers exhibit overshoots when $r$ increases
- Machines are turned on, but the effect on $n_{UL}$ is measured only after the switch-on delay




- Overshoots waste resources, undershoots hurt QoE (less videos receiving max video level)
- RAC is worse than FF in terms of $\alpha$ only during transients when $r$ increases

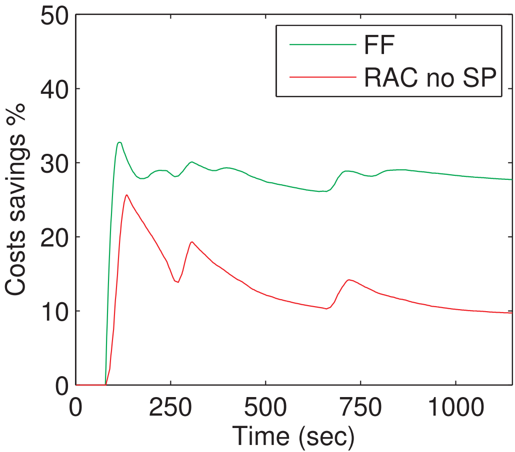
# Results: client limited flows ($\hat{n}_{CL} \neq 0$)

16% of flows with 1Mbps connection $\Rightarrow$ expected maximum $\alpha = 0.84$




- Large overprovisioning in the case of feed forward controller
- RAC w/o SP performs better but shows overshoots when requests rate increases
- RAC outperforms other controllers in terms of costs (saves 10%) and pays a slight performance degradation (4%)

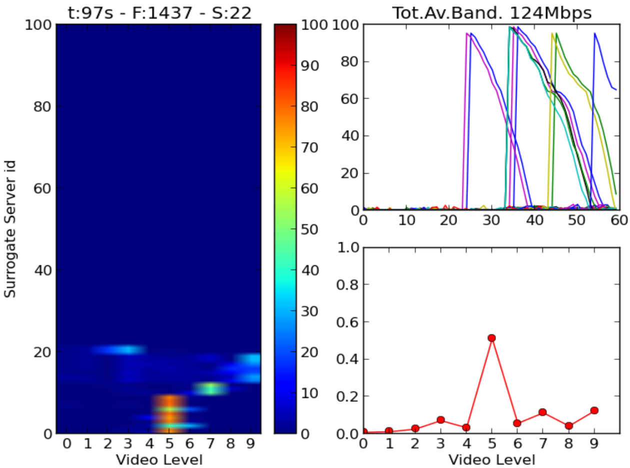
# Cost savings ($n_{CL} \neq 0$)



# Let's see RAC in motion

## Heat map

- Warmer color at (x,y) ⇒many flows are receiving level $x$ by server $y$
- Ideal: dark blue (0) everywhere except for a bright evenly colored bar at level 9




## Levels pdf

- Fraction of flows obtaining level $x$
- Ideal: zero for $x < 9$, one for $x = 9$

# Conclusions

- We have proposed a Resource Allocation Controller for cloud-based adaptive video streaming
- Feedback control theory is employed to compute the number of servers to turn on/off
- The RAC strives to minimize delivery network costs while delivering the maximum video quality
- The RAC controller saves up to 30% CPU costs while paying a small performance quality degradation during transients
- Future work: make the system distributed

## Thanks!

# Questions

? ? ?

? ?

?

?

?

?

# BACKUP SLIDES

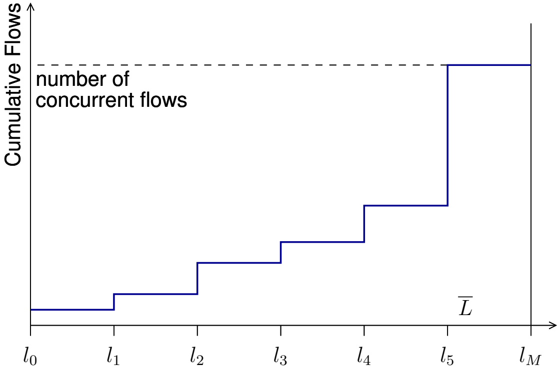
# Estimating $n_{UL}(t)$

**Estimating the number of uplink-limited flows**

- $\overline{L}$ to estimate $n_L(t)$ (limited flows)
- $\underline{L}(t)$ to estimate $\hat{n}_{CL}(t)$
- $\hat{n}_{UL}(t) = n_L(t) - \hat{n}_{CL}(t)$

**Ideally**

$n_{UL}(t) = 0$ with the minimum number of servers



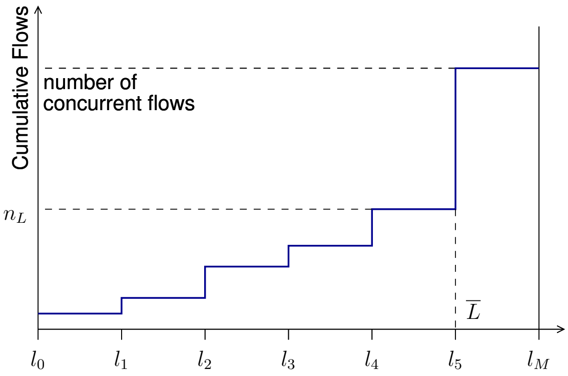
# Estimating $n_{UL}(t)$

## Estimating the number of uplink-limited flows

- $\overline{L}$ to estimate $n_L(t)$ (limited flows)
- $\underline{L}(t)$ to estimate $\hat{n}_{CL}(t)$
- $\hat{n}_{UL}(t) = n_L(t) - \hat{n}_{CL}(t)$

## Ideally

$n_{UL}(t) = 0$ with the minimum number of servers



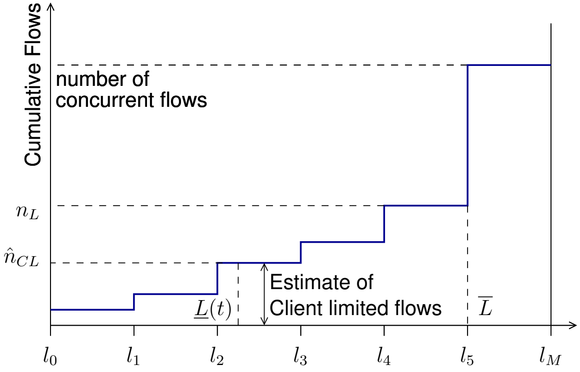
# Estimating $n_{UL}(t)$

Estimating the number of uplink-limited flows

- $\overline{L}$ to estimate $n_L(t)$ (limited flows)
- $\underline{L}(t)$ to estimate $\hat{n}_{CL}(t)$
- $\hat{n}_{UL}(t) = n_L(t) - \hat{n}_{CL}(t)$

Ideally

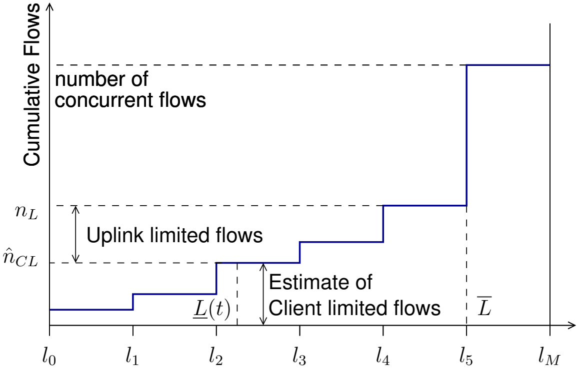
$n_{UL}(t) = 0$ with the minimum number of servers



# Estimating $n_{UL}(t)$

## Estimating the number of uplink-limited flows

- $\overline{L}$ to estimate $n_L(t)$ (limited flows)
- $\underline{L}(t)$ to estimate $\hat{n}_{CL}(t)$
- $\hat{n}_{UL}(t) = n_L(t) - \hat{n}_{CL}(t)$




## Ideally

$n_{UL}(t) = 0$ with the minimum number of servers

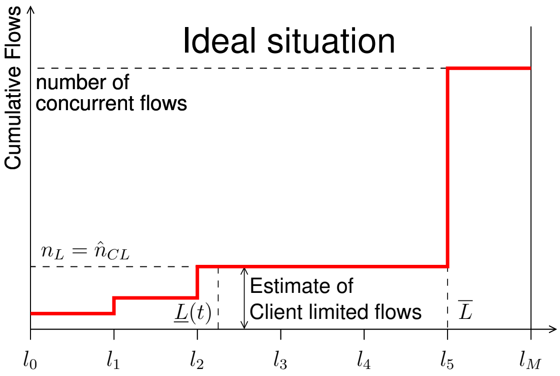
# Estimating $n_{UL}(t)$

Estimating the number of uplink-limited flows

- $\overline{L}$ to estimate $n_L(t)$ (limited flows)
- $\underline{L}(t)$ to estimate $\hat{n}_{CL}(t)$
- $\hat{n}_{UL}(t) = n_L(t) - \hat{n}_{CL}(t)$

Ideally

$n_{UL}(t) = 0$ with the minimum number of servers



# The Threshold $\underline{L}(t)$

### Definition

Every flow getting an average video level less than $\underline{L}(t)$ is considered as client limited

### Fair Level

$$l_f(t) = \min(B/n(t), l_M)$$

Fair level all $n(t)$ flows should get in the case $n_{CL}(t) = 0$.

### The threshold $\underline{L}(t)$

$$\underline{L}(l_f(t), \alpha(t)) = l_f(t) + \alpha(t) \cdot (l_M - l_f(t))$$

where $\alpha$ is the number of flows getting the maximum level.
Bandwidth limited clients leave bandwidth to other clients with the effect of increasing their average levels