Lab 3: Adaptive Video and QoS

SCOM/ SRSI 2020-21

A. Aguiar, FEUP/ DEEC

Part 1: Analyse MPD file and get acquainted with adaptation algorithms

Identify the fields in http://www.digitalprimates.net/dash/streams/gpac/mp4-main-multi-mpd-AVNBS.mpd
Q: How many video and audio codecs are available?
Q: Which is the segment size?
Q: Which audio and video datarates are available?
Q: Which options would you have for an available bandwidth of 500kbps, 1.5Mbps, 10Mbps?

Go to the web page of the MPEG-DASH Industry Forum: https://dashif.org. Under the tab Software, in the topic Reference Client, you can access a list of DASH clients for trials. Choose the most recent one for this work.
Then, play the video https://dash.akamaized.net/akamai/bbb_30fps/bbb_30fps.mpd and observe what happens for different controls (adaptation with different algorithms).
Use wireshark to observe the playing of the video stream.
Q: Describe the data transfer profile of the video stream, preferably with a plot.

Part 2: Design a bottleneck experiment to observe quality degradation in the video you are watching

Play the video stream, and some time into the stream, start generating traffic with iperf (https://iperf.fr/), a traffic generator tool that can also be used to generate traffic and to estimate available bandwidth. You need to set up an iperf server in another machine and make sure that the traffic shares a queue with the video stream. Plan where to place the iperf server and client with respect to the video client, so that their traffic interferes with/ steals bandwidth from the video flow. For this, make a diagram of the network, identifying the nodes, links and their transmission rates, queues involved and which flows share them. This plan should be done before continuing with the experiment.

Start an iperf stream with UDP at increasing data rates and observe the choices of the player. Finish the iperf UDP traffic before the end of the stream, and observe what happens. Show your observations using the wireshark trace analysis tool, mark the timestamps at which you added and changed cross traffic. Justify what you are seeing.
Observe the behaviour of the video player. Which data rates is the player choosing? How often are segments requested? How does the player tell the server which rate to use?
Start TCP cross traffic after ~2 minutes, for 2 minutes.
Observe the choices of the player and answer the same questions as above.

Start an iperf stream with UDP, sequentially increasing data rates for periods of ~2min. Finish the iperf UDP traffic before the end of the stream. Observe the choices of the player and answer the same questions as above. Were the behaviours different for TCP and UDP cross-traffic? How? Why? Did you expect this behaviour?
Mark the timestamps at which you added and changed cross traffic. Describe your observations, and explain what happened using the wireshark trace analysis tool. To show the impact of the crosstraffic, you can show and interpret the throughput of the TCP flow.

Part 3: Implement bandwidth reservation for the video flow

Checkout the Linux tool traffic control: http://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html
We will use only tc, not tcng.

In the node that is the bottleneck, i.e. where your cross-traffic is sharing a link with the video flow, implement a queueing policy that enables traffic differentiation.

- You should create a token bucket so that the video flow has a certain bandwidth guarantee. Identify which parameters you need to define for the flow and the traffic reservation, and which quantities you need to know, e.g. available link bandwidth.
- Beware that you must mark packets so that tc can handle them.

Re-run the experiments with cross-traffic that you performed before, document them using the TCP throughput plots, and explain the observations.

More information on tc and linux packet forwarding

• J. Vehent, Journey to the Center of the Linux Kernel: Traffic Control, the QoS, http://jve.linuxwall.info
• W. Almesberger, Linux Network Traffic Control – Implementation Overview (see moodle)
• https://www.funtoo.org/Traffic_Control
• http://wiki.openwrt.org/doc/howto/packet.scheduler/packet.scheduler
• http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm