DASHbed: a testbed Framework for Large Scale Empirical Evaluation of Real-Time DASH in Wireless Scenarios

Darijo Raca, Yusuf Sani, Cormac J. Sreenan, Jason J. Quinlan

Summary

Goal: Create a highly customizable real-time framework for testing HAS algorithms in a wireless environment

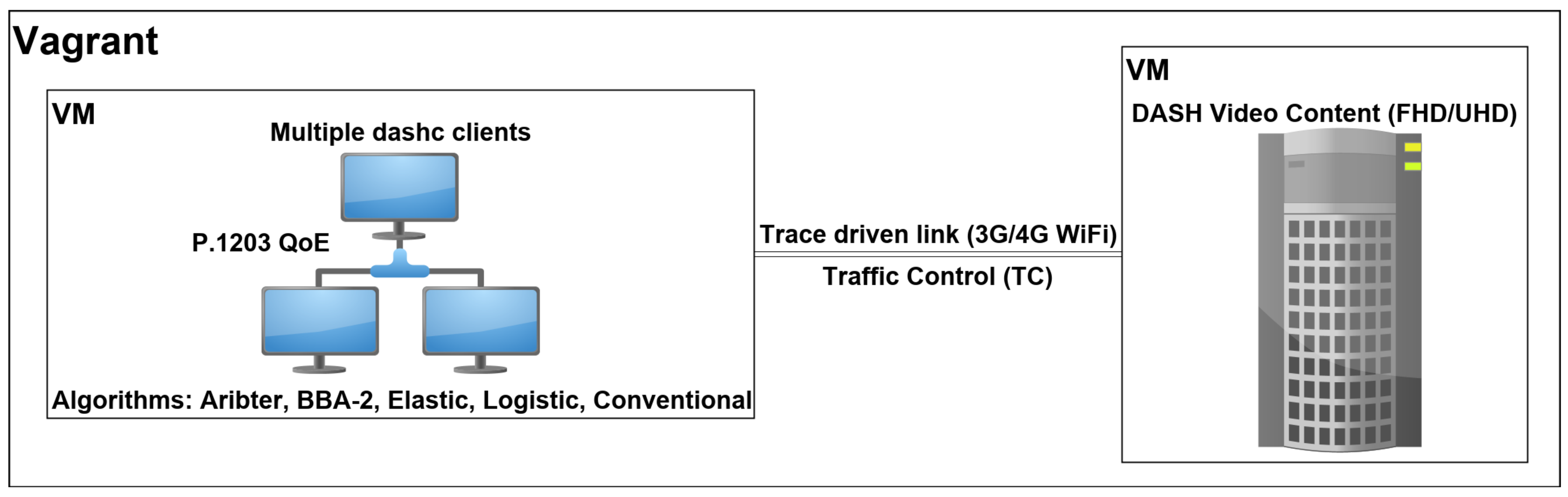
Approach: We use our previously released *dashc* headless HAS player, and create a validation testbed using Vagrant (a light weight VM environment)

Contribution: *DASHbed* offers a means of running large-scale experiments with a hundred competing players. We supplement the proposed framework with a dataset consisting of results for five HAS algorithms tested in various evaluated scenarios. The dataset showcases the abilities of *DASHbed* and presents the adaptation metrics per segment in the generated content (such as switches, buffer-level, P.1203 values, delivery rate, stall duration, etc.)

DASHbed

- Trace driven framework for large scale HAS evaluation in wireless scenarios:
  - WiFi
  - 3G
  - 4G
- Five Algorithms implemented in *dashc:*
  - *Conventional*
  - *Elastic*
  - *Arbiter*
  - *BBA-2*
  - *Logistic*
- Options for implementing both a physical hardware testbed and a virtual testbed
- Modified *dashc* headless player to provide P.1203 QoE values
- Generated logs are provided per downloaded segment, shown in table below
- Output logs provide a means of validating HAS algorithms

Framework Architecture



- Framework is run on one physical machine
- We use Vagrant for creating framework box
- Client and Server are implemented as virtual machines (Ubuntu 18.04)

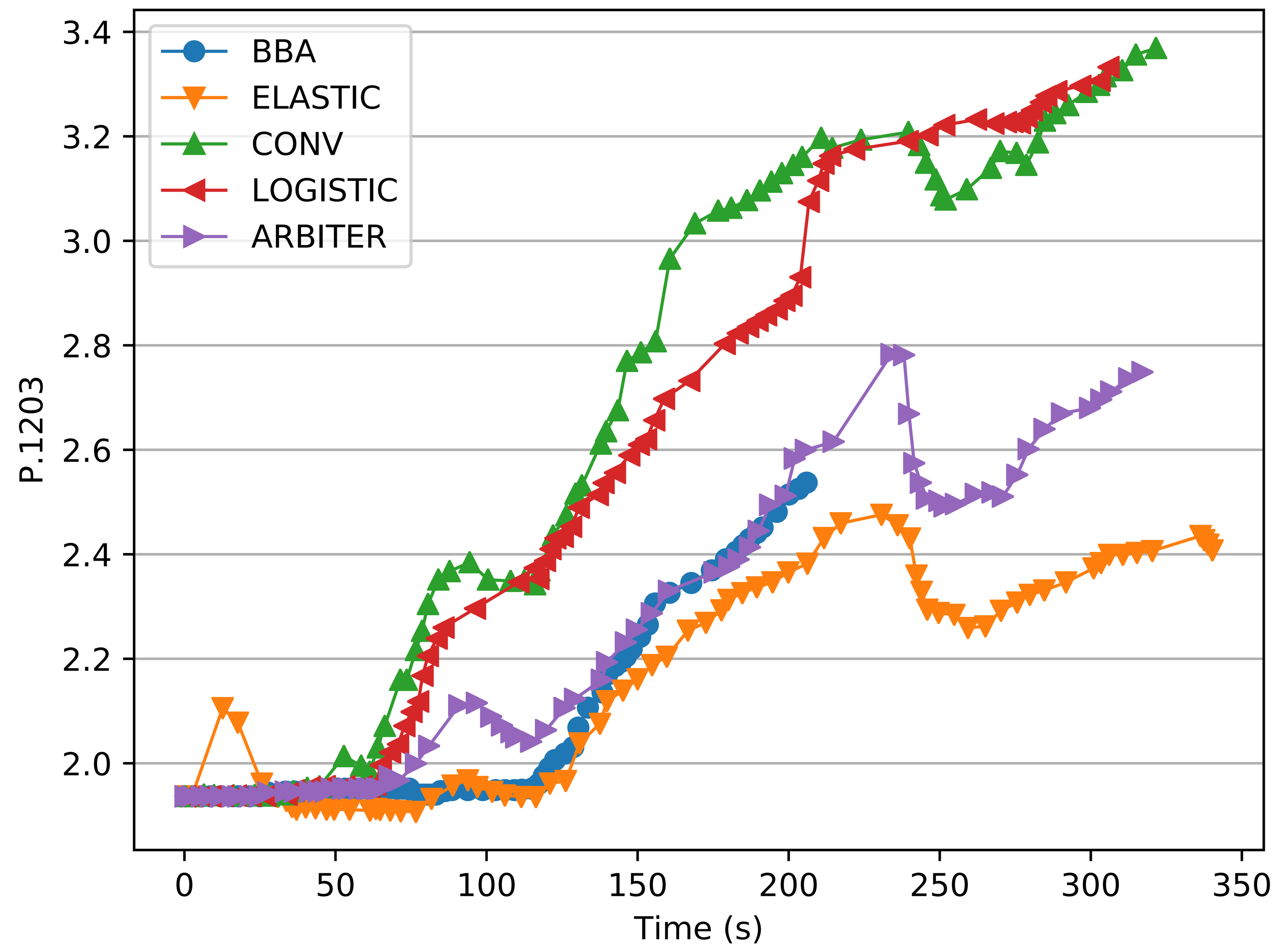
Template for running DASHbed framework

```
#python3 runExperimentsDASHCMD.py interface \
numOfCClients numRuns streamingDuration \
[list of algorithms] segmentDuration [movieNames]
```

Further information and build instructions available at:
http://www.cs.ucc.ie/misl/research/datasets/dashbed/

Dataset

P.1203 QoE for all five algorithms in a single case scenario (4 seconds segment duration)



# of clients	# of algorithms	# of runs	segment dur. (s)
1	1	5	2, 4, 8
4	1	5	2, 4, 8
5	5	5	2, 4, 8
10	5	5	2, 4, 8

Seg_#	Arr_time	Del_Time	Stall_Dur	Rep_Level	Del_Rate	Act_Rate	Byte_Size	Buff_Level	RTT	Codec	Width	Height	FPS	Seg_Dur	Start	P.1203 QoE
1	109	109	0.000000	232	9070	248	124131	4.000	0.005969	h264	320	240	24	4.000	0.000	1.936882
2	1375	59	0.000000	232	18704	276	138452	8.000	0.006278	h264	320	240	24	4.000	4.000	1.936882
3	3116	533	0.000000	4275	39881	5323	2661696	11.466	0.034981	h264	1920	1080	24	4.000	8.000	2.669001
4	4621	268	0.000000	4275	47542	3187	1593595	15.198	0.075885	h264	1920	1080	24	4.000	12.000	2.997820
5	6012	113	0.000000	4275	53917	1524	762041	19.085	0.016670	h264	1920	1080	24	4.000	16.000	3.331018

{d.raca, ys8, cjs, j.quinlan}@cs.ucc.ie, Department of Computer Science, University College Cork, Ireland

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1892.