Advanced Topics in Communication Networks
9th December 2013, TIK, ETH Zurich

Exploiting SDN for Video Content Distribution

Panagiotis Georgopoulos
[panos@tik.ee.ethz.ch]

Overview

Theme : How can we use SDN to improve Video Content Distribution?

VoD Traffic Benefits Network and Users — OpenFlow-based in-network caching service for Video-on-Demand traffic

Live Video Traffic Benefits Users — OpenFlow-assisted QoE Fairness Framework (QFF)

Present :
- Why ?
- What is it?

The process :
- Requirements Analysis
- Design & Implementation
- Evaluation results

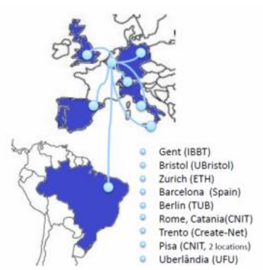
OFELIA

- An EU collaborative project within the FP7 ICT work programme
- 17 partners (industry and academia)
- Project duration: 3 years (Sept 2010 – Sept 2013)
- Total Cost: 6.3M€, EC Funding: 4.45M€
- **Establish SDN in Europe :**
  - **Create an OpenFlow Testbed across Europe**
  - **Linking Infrastructure and Applications**

Applications
OpenFlow
Network Infrastructure

OFELIA Testbed

- 10 OpenFlow-enabled islands federated together
- Publicly available for experiments
- Allows for experimentation on multi-layer and multi-technology networks
- A unique experimental facility that allows researchers not only to experiment "on" a test network but to control and extend the network itself precisely and dynamically with the use of OpenFlow

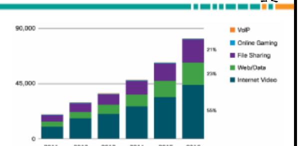


Video-on-Demand Use case for OFELIA

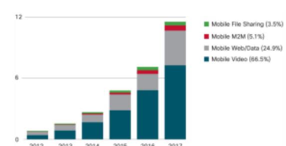
- **We wanted to (i.e. our goals)**
  1) Design and implement an OpenFlow-assisted Video-on-Demand service based on transparent caching
  2) Evaluate and demonstrate the benefits of OpenFlow on a VoD service by running inter-island experiments over the OFELIA testbed across Europe
     - Consider both Network & User perspectives

Motivation : Why Video?

- In the UK visits to online video sites have **grown by 36%** in one year (Sept 2010 - Sept 2011) [1]
- Globally, **Internet video traffic was 57% of all consumer Internet traffic** in 2012 and will be 69% in 2017 [2]
- **Mobile video traffic** exceeded 50% for the first time in 2012 [3]
- Mobile video will increase 16-fold between 2012 and 2017
- Two-thirds (~66%) of the world's mobile data traffic will be video by 2017



Global consumer Internet traffic in Petabytes per month [2]



Mobile consumer Internet traffic in Exabytes per month [3]

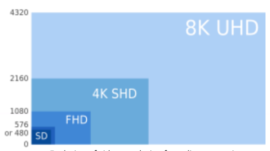
[1] Hitwise (2011)
[2] Cisco VNI Global Forecast (2012)
[3] Cisco VNI Mobile Forecast (2013)

Motivation : Why Video-on-Demand?

- With a **VoD service** (e.g. BBC iPlayer, Netflix, Amazon's LOVEFiLM) consumers can retrieve previously recorded content at a different time that the content was initially made available

- **VoD traffic will triple by 2017 :** equivalent to 6 billion DVDs per month [1]
- **Internet video to TV traffic** doubled in 2011, will **increase six fold by 2016** [1]
- **High-Definition VoD surpassed Standard-Definition VoD** in 2011 [requires ~1-8 Mbps]
  - By 2016, HD Internet video will comprise 79% of VoD [1]

- **Trend to improve video quality even more :**
  - **Moving to Ultra-HD (4K - 8K) and 3DTV :** 4 times higher resolution than HD [requires ~20-600 Mbps]



Evolution of video resolution for online streaming

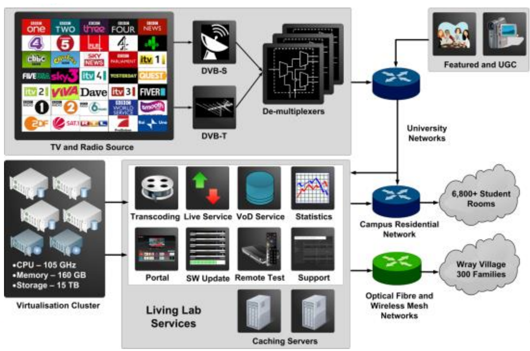
[1] Cisco VNI Global Forecast (2012)

So… what does all this mean?

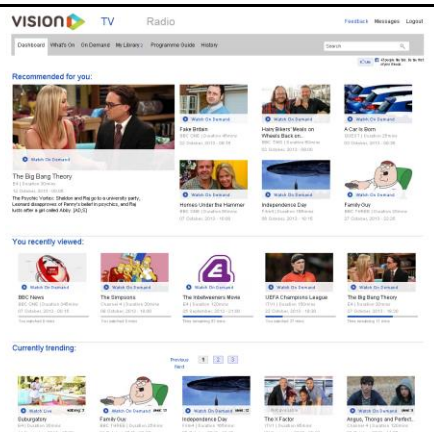
- Video streaming (live and on-demand) is fast becoming an **essential part of consumers' lives**

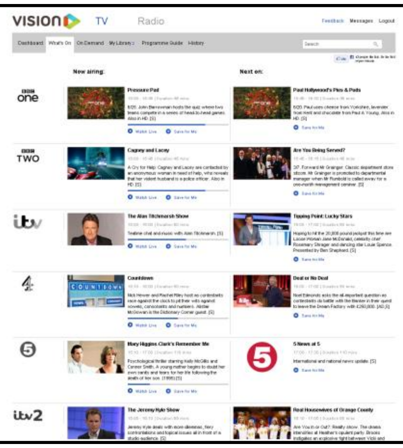
- The network has to transfer an **enormous amount of video traffic** (~45.000 PB/m. in 2016)
  - Big strain on the network ; high-throughput requirements end-to-end (especially with HD)
  - Quickly and reliably to the user ; high Quality of Experience (QoE)

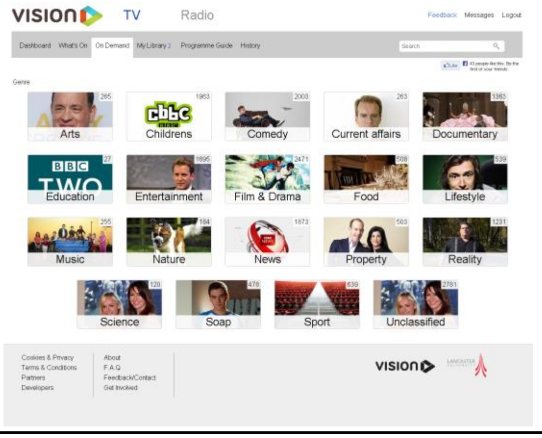
Living Lab Infrastructure at Lancaster University



Quick Live Overview of Vision IPTV







Challenge in Video-on-Demand (aka the Problem)

VoD Distribution Efficiency

- VoD requests **handled naively** – independent flow per request
- These are duplicated minutes, hours or days later (by same or different user)
- Identical delivery of media objects through the same network segments
- End-to-end capacity of network infrastructure must grow continuously to match the increasing number of Internet video users
- The increasing popularity of VoD and especially of HD content worsens this

- What is NOT a solution :
  - Multicast : VoD requests are not for the same content at the same time
  - Peer-to-Peer : Limited storage and uplink resources on user devices (peers) – cannot guarantee high QoE for the users

Key Characteristics of Video-on-Demand

- High-throughput end-to-end
  - Not just high egress capacity at origin video servers, but also adequate bandwidth available in all networks in between video source and users
- Distance matters between source VoD server and user
  - (Standard) TCP used for VoD can become bottleneck as it requires ACKs for every window of data packets sent
  - TCP's throughput is inversely related to network latency or RTT

Distance (Server to User)	Network RTT	Typical Packet Loss	Throughput	4GB DVD Download Time
Local: <100 mi.	1.6 ms	0.6%	44 Mbps (high quality HDTV)	12 min.
Regional: 500–1,000 mi.	16 ms	0.7%	4 Mbps (basic HDTV)	2.2 hrs.
Cross-continent: ~3,000 mi.	48 ms	1.0%	1 Mbps (SD TV)	8.2 hrs.
Multi-continent: ~6,000 mi.	96 ms	1.4%	0.4 Mbps (poor)	20 hrs

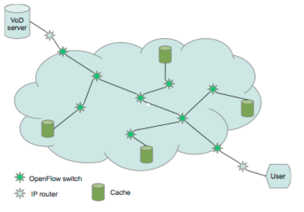
Effect of Distance on Throughput and Download Time [1]

- We need a solution that :
  - Ensures high-throughput end-to-end
  - Minimizes distance between source video content server and user

SDN?

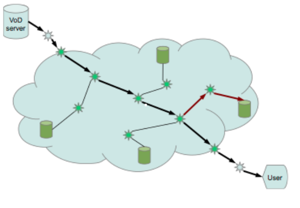
[1] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: a Platform for High-Performance Internet Applications. SIGOPS Oper. Syst. Rev., 44(3):2–19, 2010.

Video-on-Demand Content Caching with OpenFlow



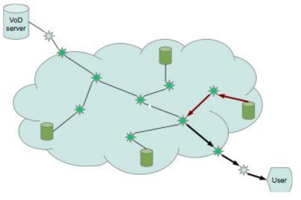
An OpenFlow network with peripheral content caches

Video-on-Demand Content Caching with OpenFlow



First interaction: Content silently copied to cache

Video-on-Demand Content Caching with OpenFlow



Later interactions: Content retrieved from cache

Caching Requirements (1)

An **OpenFlow-based content caching** architecture should satisfy the following **functional requirements** :

1. Should identify cacheable content without any significant impact on the user's request
2. Should cache content transparently to the user
3. Should deliver content transparently to the user
4. Should retain the underlying content delivery mechanism to avoid fundamental changes to the service
5. Should be content agnostic
6. Should be easily integrated in a production network
7. Should be able to use multiple cache instances
8. Should be able to add or remove cache instances without service interruption

Worth thinking :
How could we do the above without SDN?

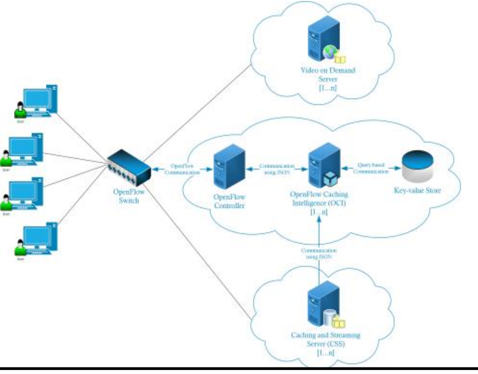
Caching Requirements (2)

An **OpenFlow-based content caching** should satisfy the following **non-functional requirements** :

1. Should optimize network utilization. For example, it should not unreasonably cache content that is infrequently requested and thus increase the network utilisation unnecessarily
2. Should adjust its run-time functionality and improve the users' QoE by maintaining a high level view of the network based on run-time metrics (e.g. buffering times etc.)
3. Should support load balancing between carefully and strategically located in-network caches

Worth thinking :
How could we do the above without SDN?

OpenFlow-assisted In-network Caching Architecture



Entities (1)

- Any hardware or software OpenFlow Switch
  - Must be able to communicate with the VoD server, the CSS(s) and the OpenFlow controller, but not necessarily directly

- Primary source for the video assets
  - Could be located anywhere on the Internet (reachable by IP)

- Any kind of OpenFlow Controller (e.g. Floodlight, NOX, POX)
  - Should be reachable by the OpenFlow Switch
  - Runs L2 learning switch : allows the switch to forward on MAC-to-Port pairing
  - Exposes a JSON-RPC Flow Pusher interface to OCI

Entities (2) : OpenFlow Caching Intelligence (OCI)

- Orchestrator of in-network caching functionality
  1. Provides a JSON-RPC interface to retrieve requests for content to be cached in a highly flexible and configurable fashion (Cache as a Service)
     - Used by network administrator or even content providers (via SLAs)
     - Supports regular expressions to fine tune requests for content (e.g. particular video, all videos from a domain, a type of video from any domain (n.b. with later versions of OpenFlow))

METHOD	PARAMETERS	RESULT
start-expr	{ "expr" : <expr> }	<boolean>
stop-expr	{ "expr" : <expr> }	<boolean>
list-expr-all	None	[{'expr': <expr>, 'port': <port>}, ...]

  2. Implements the caching logic : what should be cached where at each point in time
     - Enhanced to support resource monitoring and load-balancing

Entities (2) : OpenFlow Caching Intelligence (OCI)

- Orchestrator of in-network caching functionality

3. Provides a JSON-RPC interface to manage the resources of the available caches in the network : Handles addition/removal of caches at run-time

METHOD	PARAMETERS	RESULT
hello	{ "host" : <host>, "port" : <port> }	<node-id>
Keep-alive	{ "node-id" : <node-id> }	<boolean>
goodbye	{ "node-id" : <node-id> }	<boolean>

4. Manages the OpenFlow switches of the network via the Controller
   - Adding/removing flows to switches via the Flow Pusher API of the controller so that users' requests are served appropriately

URI	DESCRIPTION	ARGUMENTS
/wm/staticflowentrypusher/json	Add/Delete static flow	HTTP POST data (add flow), HTTP DELETE (for deletion)

Entities (3)

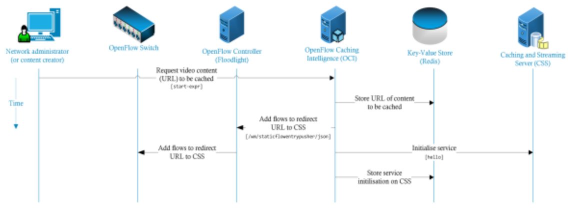
- Key-Value Store to maintain a list of :
  - All names of videos that have been requested for caching
  - Videos that have been cached and where
  - Status of CSS (online/offline, reachable etc.), their location and resources

- Caching and Streaming Server (CSS)
  - Multiple CSS instances in the network, possibly connected directly to the switch and consequently to the user : Lower latency and faster response times (high QoE)
  - Three operations :
    1. Communicate its status to the OCI
    2. Caching content that is requested from the user
    3. Stream content that is being already cached

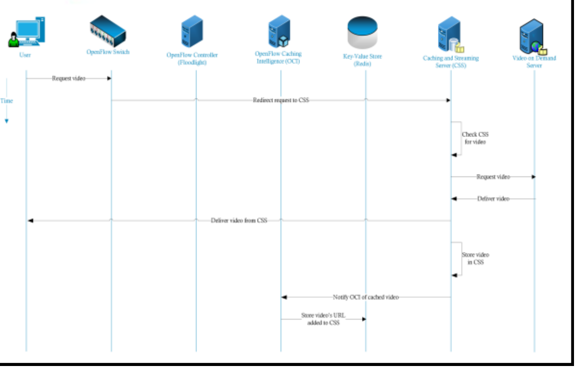
Three Essential Operations

1. Handle requests for content to be cached (Cache as a Service)
   - From network admins/content providers
2. Serve user requests for content that has not been cached yet
   - Serve user and cache content for future use
3. Serve user requests for content that is in a network's cache

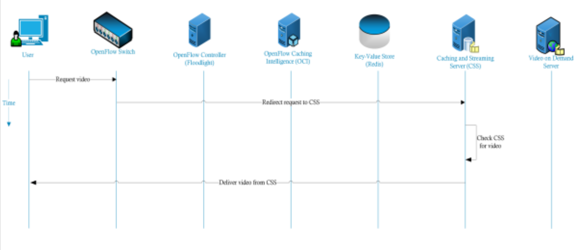
Handle Requests for Content to be Cached



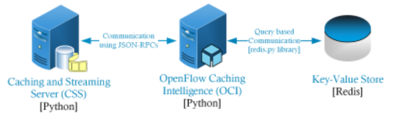
Serve User Requests for Content that has not been Cached yet (cache-miss)



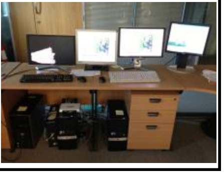
Serve User Requests for Content that is in the Network's Cache (cache-hit)



Implementation (1)



- Python based Implementation : using Redis Key-Value Store, Floodlight Controller
- Open-source (soon available on github)
- Video Content :
  - Video files are big ; we need chunk based video files that provide flexibility & scalability
  - MP4 chunks don't work well ; either web browsers don't play them properly or have to load the full video before start playing it ☹



Implementation (2) : MPEG-DASH

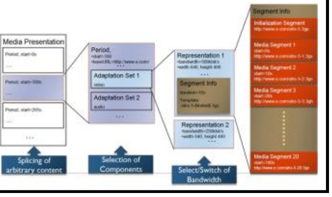
- **DASH: Dynamic Adaptive Streaming over HTTP**
  - Audio/video agnostic
  - Chunked media facilitates swapping between bitrates
  - Adaptive to network bandwidth
  - Can be delivered using conventional HTTP servers
  - Standardised and has support from industry

Chunks

Fetch from different CSS

Increase QoE

- Uses a **media presentation description (MPD)** describes segments information :
  - Timing, URL, media characteristics such as video resolution and bit rates

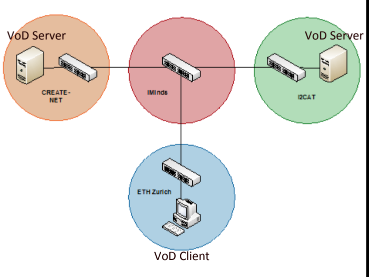


Thomas Stockhammer , Qualcomm Incorporated

Evaluation on the OFELIA testbed

Conceptual Evaluation Setup

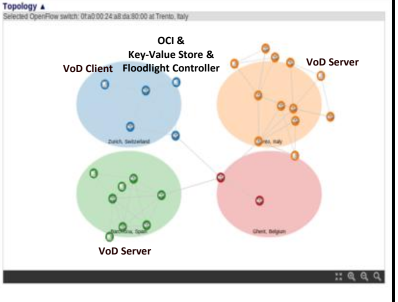
- **Topology :** Deployed our architecture on three OFELIA islands distributed geographically
  - Switzerland : ETH Zurich
  - Italy : Create-NET
  - Spain : i2CAT
- Over 120 inter-island (federated) experiments over the OFELIA testbed



Evaluation on the OFELIA testbed

Evaluation Setup on Expedient

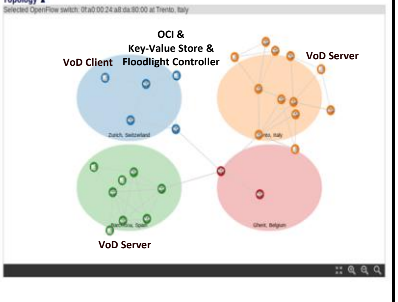
- **Three Scenarios :**
  - Without cache (baseline)
  - With cache (cache-miss)
  - With cache (cache-hit)
- **Experiments**
  - Big Buck Bunny : ~10min. reference video
  - 20 VoD requests of each scenario with both VoD servers



Evaluation on the OFELIA testbed

Evaluation Setup on Expedient

- **Evaluation Criteria :**
  - Startup delay (QoE metric)
  - External link network utilisation (content fetched from cache)
  - Caching hits/miss



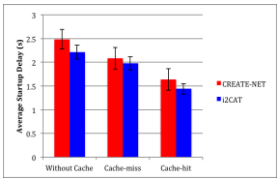
Results

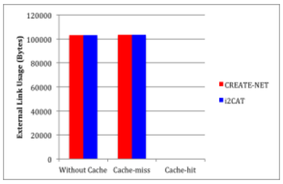
	CREATE-NET (Italy)			i2CAT (Spain)		
	Without Cache	Cache-miss	Cache-hit	Without Cache	Cache-miss	Cache-hit
Average Startup Delay (s)	2.484	2.088	1.639	2.212	1.982	1.441
Improvement over Baseline (%)	-	16.02	34.02	-	10.40	34.85
Standard Deviation (σ)	0.208	0.225	0.226	0.145	0.138	0.109
External Link Usage (Bytes)	105,734,144	105,827,872	0	105,734,144	105,827,872	0

- **Key results :**
  - In tests over both islands we **reduced the startup delay** up to **35% -> increased QoE for end-user**
  - External **link utilisation reduced to virtually zero** (only background traffic remained)
    - Indicatively, the full streaming of our ~10min video saved ~100MBytes for just one client session

Results

Average Startup Delay **External Link Usage**





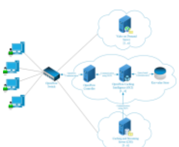
- 35% Improvement even in a bandwidth reach environment (OFELIA testbed)
- Reinforced by relatively low standard deviation values
- Greater improvements would be possible on next generation OpenFlow switches where packet processing will take place on the hardware path

Advantages of our VoD In-network Caching Architecture

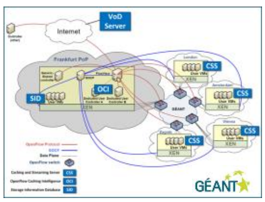
1. Provides an **interface for cacheable content** in an "**open**", **highly-configurable, controllable** and **flexible** manner
2. **Centrally controlled caching** : efficient load balancing, allows pre-caching of frequent content
3. **Easily deployable in a production network** : the underlying delivery video mechanism will remain the same in an OpenFlow network (existing hardware and software can be retained, no fundamental changes in service)
4. Fully **transparent to the user** : no need to install any extra software or have to sacrifice any of his local network or storage to be able to stream HD content with high efficiency.
5. **Caching very close to the user** :
   a) **Reduces network utilisation** as requests are served locally : minimize the amount of packets that are required to traverse the network from the source media provider to the user
   b) The **video QoE of the end-user will improve**, as the user will experience lower latency, smaller buffering times and higher video quality as content is now located locally

Summary

1. Designed and implemented an **OpenFlow-assisted Video-on-Demand** service based on efficient, transparent and highly configurable caching



2. Evaluated and demonstrated the benefits of OpenFlow on a VoD service by running inter-island experiments over the OFELIA testbed across Europe
   - Improved both **network efficiency** and **user experience**



Live Video Traffic

How to provide a user centric, but network-wide, Quality of Experience (QoE) Fairness on Adaptive Video Streaming ?

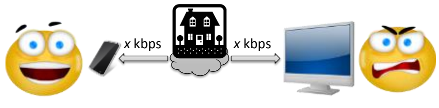
Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. ***Towards Network-wide QoE Fairness using OpenFlow-assisted Adaptive Video Streaming***. In: ACM SIGCOMM 2013 Workshop on Future Human-Centric Multimedia Networking (FhMN), 16 August, 2013, Hong Kong, China.

The Problem

- Adaptive Video Streaming (e.g. MPEG-DASH) aims to increase QoE and maximise connection utilisation (supporting chunks encoded at different bitrates)
- Many implementations are bursty and unstable in nature and naively estimate available bandwidth from a one-sided client perspective
  - No account of other devices in the network
  - Results in unfairness ; video streams fight over link's capacity which causes network congestion (video quality degradation, frame freezing etc.) and potentially lowers QoE for all clients
- **Counter productive!**

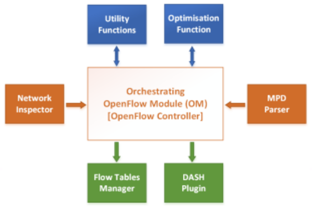
Potential Solution

- Split available bandwidth to current users on the network ?
- But **naïve network resource fairness** (equal split) is **unfair** :
  - You could easily satisfy a user watching a video on his smartphone, but it is much harder for an HD TV

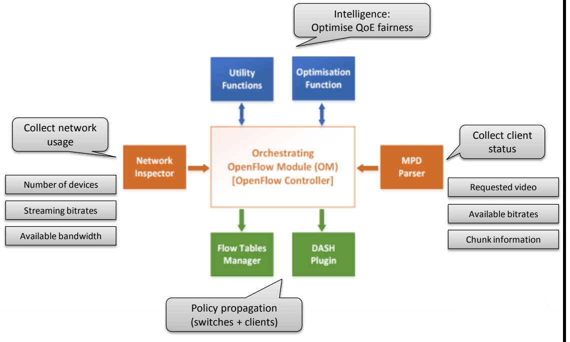


Our solution : OpenFlow-assisted QoE Fairness Framework (QFF)

- Aims to provide a user-centric fair-share of network resources and fairly maximise the QoE of multiple competing clients in a shared network environment
  - QFF monitors video streams of all clients in a network and dynamically allocates network resources to each device. Avoid user-agnostic decisions ; no blindly dividing bandwidth between active users
  - Use of SDN to provide the network-wide view and the control plane to orchestrate this functionality



Our solution : OpenFlow-assisted QoE Fairness Framework (QFF)

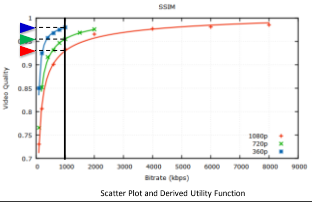


QFF's Core Intelligence : Utility Function

- Utility Function provides a model that maps the bitrate of a particular video to the QoE delivered on that specific device
  - We obtained QoE for each video sequence using objective video quality assessment that employ a functional model of the human visual system (Structural Similarity Index (SSIM), Video Quality Metric (VQM))

Resolution	Video Bitrate (kbps)
1080p	100, 200, 600, 1000, 2000, 4000, 6000, 8000
720p	100, 200, 400, 600, 800, 1000, 1500, 2000
360p	100, 200, 400, 600, 800, 1000

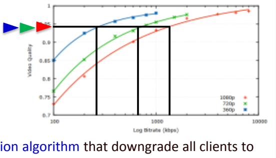
- Utility Function proved that :
  - Relationship between bitrate and perceptual quality is not linear
  - Equal division of bandwidth between different resolutions results in QoE unfairness



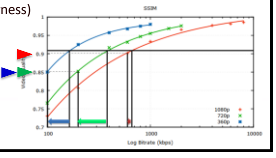
Scatter Plot and Derived Utility Function

QFF's Core Intelligence : Optimisation Function

- Optimisation Function finds the optimum bitrate for each streaming video device in the network that results in equivalent QoE levels for all devices
  - But the utility functions are not continuous, i.e. we don't have available encodings for all possible bitrates

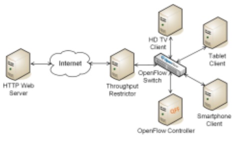


- Implemented branch and bound optimisation algorithm that downgrade all clients to the maximum feasible bitrate (max-min fairness)
  - Very modest computational overhead < 0.3sec for optimising 100 Utility Functions with 10 different bitrates each

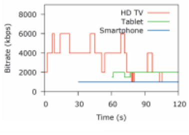


Proof-of-concept Evaluation

- Around a home networking scenario (transferable to campus network, corporate networks etc.)
- Three different DASH-enabled devices : smartphone (360p), Tablet (720p), HDTV (1080p)



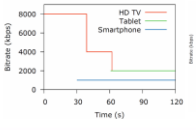
DASH-JS



Network instability :

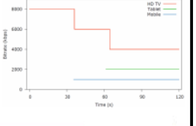
Bitrate changes : 18-31 (av.23)

EQUAL B/W (control)



Network stability: Bitrate changes : 2, but HDTV gets penalized (lower QoE)

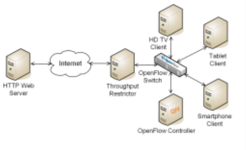
QFF



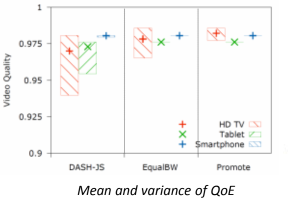
Network stability & QoE fairness across devices

Proof-of-concept Evaluation

- Around a home networking scenario (transferable to our campus network)
- Three different DASH-enabled devices : smartphone (360p), Tablet (720p), HDTV (1080p)



- QFF produces increased mean QoE and reduced QoE variance (particularly for the HDTV)



Mean and variance of QoE

Summary

Theme : Exploiting SDN for Video Content Distribution

- Aims to optimise network utilization and increase user's QoE by reducing start up and buffering times and increasing video quality levels

OpenFlow-based in-network caching service for Video-on-Demand traffic

VoD Traffic Benefits Network and Users

- Aims to provide a user-centric fair-share of network resources and fairly maximise the QoE of multiple competing clients in a shared network environment

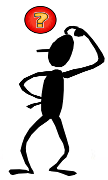
OpenFlow-assisted QoE Fairness Framework (QFF)

Live Video Traffic Benefits Users

One final thought…

Could we have done all this

without SDN ?



Thank you ! Questions?