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, Rebecca Batat

[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 \textit{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]

Published papers in:
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]

QoS parameters
--- e.g., content access latency in CDN

Network application

Agent layer implementing service interface for S

map protocol state onto service interface state

Distributed processes implementing protocol $P_i(S)$: $i=1,2$

protocol $P_i(S)$: active instance protocol $P_2(S)$ is inactive

Repertoire of optimistic and pessimistic algorithms

A single-shoe doesn’t fit all sizes!!

Distributed realization of system infrastructure ‘resources’

E.g., placement of mirror sites in a CDN

--- abstracted as mapping function

$R = F_i([q_x,q_y,\ldots],E^*)$

Exercise resources

Asynchronous processes implementing protocol (i.e., algorithm)
Management control of quality of QoS support mechanisms

Target network system

\[ S \]

[controlled system: algorithms, resources, adaptation logic, etc]

desired QoS \((q)\)

attained QoS \((q')\)

external environment \(E^*\)

prescribed QoS \(q\)

achieved QoS \((q')\)

analyze service-layer algorithms and resource allocations

run-time selection of algorithm

Decision-Engine
(realized by management entity)

System instance 1 is more resilient than instance 2, in dealing with environment condition \(e_h\);
Instance 2 is more resilient than instance 3.

\[ \text{A single-shoe doesn’t fit all sizes!!} \]
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:

$$E^* = E(y_k) \cup E(n_k) \cup E(c_k)$$

- Parameters that the designer knows about (known “knowns”)
- Parameters that the designer does not currently know about (knowable “unknowns”)
- 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

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]

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

900 video frames (H.264) played out @ 25 fps

Bit-rate generated by video encoder (\(\lambda\) in kbps)

QP

TIME in secs

Source-1: QP=26, visual_quality = GOOD

Source-2: QP=28, visual_quality = BETTER

Source-3: QP=28, visual_quality = BETTER

900 video frames (H.264) played out @ 25 fps

Bit-rate generated by video encoder (\(\lambda\) in kbps)

QP

TIME in secs
**Iterative adjustment of video bit-rate to effect congestion-relief**

**Additive Increase Multiplicative Decrease (AIMD)**

“available bandwidth” on a transport path is unknown

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

\[
\lambda_{\text{new}} = \lambda_{\text{cur}} - \beta \cdot L \quad \text{when} \quad L > \delta_h, \\
\text{where } \beta > 0
\]

\[
\lambda_{\text{new}} = \lambda_{\text{cur}} + \alpha \quad \text{when} \quad L < \delta_l, \\
\text{where } \alpha > 0
\]

\(L\): observed “packet loss ratio”

\(\delta_l, \delta_h\): Acceptable loss thresholds

\([\delta_l < \delta_h \text{ 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

\[ \lambda' = BAVT (\lambda, B_{av}^*) \]

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

\( \lambda' \) is determined from the current send rate \( \lambda \) and available bandwidth \( B_{av}^* \)

\( \lambda' \) is split as \( \lambda'_a, \lambda'_b, \lambda'_c \) at sink end-point

Reduced signaling overhead: \( O(N) \)

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

EDSI: end-user device signaling interface

source \( V_c \) serves \( D_{x}, D_{y} \)

source \( V_a \) serves \( D_{z} \)

source \( V_b \) serves \( D_{u} \)

[equipped with H.264 encode capability]

[signaling of encoder run-time parameters [quantization, bit_rate, …]]

[signaling of encoder meta-data [codec_type, frame_rate, …]]

packet flow (for video downloads)

Transport path over cloud network (say, TCP)

\( L = \text{net}(\lambda, B_{av}^*) \)

Aggregate send rate \( \lambda' \)

End-to-end packet loss signaling \((L)\)

Reduced signaling overhead: \( O(N) \)
Sample timing scenario of AIMD

Available bandwidth on transport connection

\[ B_{av} \]

Send rate \( \lambda' \)

Desired send rate \( \lambda \)

Convergence detection latency

\( T_c = 9 \)

Steady-state: \( \beta \) is reduced to base value; \( \alpha \) is lower than base value

Limit-cycle time \( T_l = 4 \)

For base values for \( \beta, \alpha \)

\[ \frac{\Delta \lambda'}{T_l} \]

Rate jitter:

\[ \frac{d\lambda'}{dt} \]

TIME (iteration #s)

0 < \( b_y < b_x \)

0

1 2 3 4 5 6 10 15 20 22

Sample timing scenario of AIMD
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

FF-MPEG

SVC encoder

VBR

RBR

“leaky-bucket” buffer

x: data unit

transport agent

injected bit-rate \( B_{dem} \)

data packet flow over end-to-end network path (e.g., UDP/IP)

available bandwidth \( B_{av} \)

transport agent

play-out buffer

receiver device

FF-MPEG

SVC decoder

VBR

“loss/delay” reports

available bandwidth \( B_{av} \)

transport agent

presentation quality estimator

allowed bit-rate

softwarized bandwidth-adaptive transport module (AIMD)

bandwidth demand \( B_{dem} \)

softwarized SVC module

display size

scene

QP

QP
User-participatory control of video send rates

H.264-encoded data flow (say) $f_1$, $f_2$, ..., $f_N$ at flow aggregation point send data packets into core transport network. AIMD protocol peers report $\lambda$: user-level attempted data flow rate (aggregated). Sustainable rate $\lambda' < \lambda$.

Mechanism to detect bandwidth-hogging users, and evict them from session.

Software stubs set reference parameters to use video encoder & network bandwidth.

User-assisted control of video data flow & network BW (including “last-mile” segment).

‘fuel-gauge’-type of BW availability indicator

DASHBOARD [visual display]

GUI-internal mapping

SLIDING BAR FOR CONTROL

OK cancel

QoS space

Min. acceptable rate $\lambda_i$(min) Max. budgeted rate $\lambda'_i$

Send rate $\lambda'_i$

REDUCE

IMPROVE

set reference parameters to use video encoder & network bandwidth
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 always keeps video chunks up-to-date

master server (hosts video content)

HTTP requests from clients to pull video content are directed to these nodes

Video encode & decode instances, treated as NFV/SDN functions
[peer modules running on overlay mirror site device & client device]

CDN overlay distribution path segment (realized over a data transport path)
[for asynchronous push of video chunks from \( R \) to mirror sites]

Transport path segment to carry data flows (say, UDP or TCP connection)

AIMD-level data flow ingress/egress points

AIMD-level data flow ingress/egress points

Video data transport from content mirror sites & content rendering at client devices
(over last-mile access links)

Video encode & decode instances, treated as NFV/SDN functions
[peer modules running on overlay mirror site device & client device]

Non-participant node & link

ca-cj: Client devices

--- local attachment (via access network)

--- Video data transport from content mirror sites & content rendering at client devices (over last-mile access links)