Using Informed Access Network Selection to Improve HTTP Adaptive Streaming Performance

Theresa Enghardt
TU Berlin
MPI for Informatics
theresa@inet.tu-berlin.de

Thomas Zinner
NTNU - Norwegian University of Science and Technology
thomas.zinner@ntnu.no

Anja Feldmann
MPI for Informatics
anja@mpi-inf.mpg.de

ABSTRACT

As end-user devices often have multiple access networks available, choosing the most suitable network can help to improve application performance and user experience. However, selecting the best access network for HTTP Adaptive Streaming (HAS) is non-trivial, e.g., due to complex interactions between network conditions and the Adaptive Bit-Rate algorithm (ABR), which adapts to network conditions by selecting which video representation to load. In this paper, we propose to use an application-informed approach, Informed Access Network Selection (IANS), to select the most suitable access network for each video segment. We evaluate the impact of IANS on HAS performance in a testbed under a variety of network conditions and using different workloads. We find that IANS improves HAS performance substantially, in particular in cases where the available downstream capacity is low. In the Capacity Decrease scenario, where capacity decreases drastically during the video load, IANS can improve the estimated Mean Opinion Score (MOS) compared to using a single network from 2.1 to 2.8. We compare IANS to MPTCP using the Lowest-RTT-first scheduler, which continues to use a low downstream capacity network, resulting in lower performance. This confirms that IANS can improve video streaming performance.

CCS CONCEPTS

• **Networks** → *Network performance analysis*; *Network dynamics*; *Network experimentation.*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys'20, June 8–11, 2020, Istanbul, Turkey
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6845-2/20/06…$15.00
https://doi.org/10.1145/3339825.3391865

KEYWORDS

Access networks, Multi-Access Connectivity, Video performance evaluation

ACM Reference Format:
Theresa Enghardt, Thomas Zinner, and Anja Feldmann. 2020. Using Informed Access Network Selection to Improve HTTP Adaptive Streaming Performance. In *11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3339825.3391865

1 INTRODUCTION

Adverse network conditions can degrade Quality of Experience (QoE) for applications such as HTTP Adaptive Streaming (HAS). For example, insufficient or varying downstream capacity leads to interrupted playback or frequent quality switches. To overcome such adverse network conditions, an end-user device should switch to a different access network if available, e.g., from WiFi to cellular or vice versa. While switching between networks or using multiple access networks has the potential to improve performance, this potential is not yet used. Even today, many end-user devices use WiFi by default and fall back to cellular only if WiFi is not available, even though the performance via WiFi may be inferior to the performance via cellular. Aggregating the downstream capacity of both networks is possible using Multipath TCP (MPTCP) [18]. However, MPTCP is not always available, e.g., due to a lack of server-sided support [2]. Moreover, while MPTCP distributes data to send over different paths, e.g., via multiple access networks, on the server-side, it is typically application-agnostic, i.e., independent of transfer size or available downstream capacity, and is unable to consider any client-side information in its decisions.

To overcome these limitations, Informed Access Network Selection (IANS) [6] is an application-aware approach that enables hosts to select the best suitable access network(s). For each new connection or transfer, an application communicates its needs, e.g., whether it prefers a network with short latency or high downstream capacity. An IANS policy then matches application needs to the network(s) with the desired network performance characteristics, if available. Prior work [6] evaluates the benefits of IANS for Web browsing and finds that IANS can shorten overall load times by loading small resources via a network with short latency and large resources via a network with high downstream capacity. However, the same approach cannot be used for HAS, which typically loads a series of large resources. Here, it is not sufficient to simply pick the network with higher capacity. Instead, the host also needs to ensure that sufficient downstream capacity remains available over time, and otherwise switch to another network. This is challenging as the actual downstream capacity is variable, especially in mobile scenarios. Yet, prior work assumes that the available downstream capacity is constant over time and shared between all concurrent resource loads. Therefore, this paper introduces new IANS Policies which are necessary to use IANS for HAS.

When designing IANS Policies for HAS, which network characteristics to optimize for is an open question, due to complex interactions between ABRs and network conditions [3]: Choose the network with the highest available downstream capacity to optimize for the highest video quality, or choose the network with the least downstream capacity variation to prevent video playout interruptions? To shed light on this problem and to determine which approach yields the best IANS Policy, this paper proposes and compares multiple IANS Policies for HAS. In our evaluation, we use multiple ABRs, which we do not modify.

The contributions of this paper are as follows: (1) We design three IANS Policies, which select the most suitable access network(s) for each video segment: The Optimist Policy, the Pessimist Policy, and the Selective MPTCP Policy. (2) We implement these policies within the Socket Intents prototype [6]. (3) We evaluate the impact of IANS on HAS performance using a systematic study in a testbed, where we compare our IANS Policies vs. using a single access network or using MPTCP. Hereby, we use three different videos in eight different network scenarios. We find that the Pessimist Policy reduces playback interruptions and leads to better QoE in most scenarios. In the Capacity Decrease scenario, we find that using application-agnostic MPTCP degrades performance, while the Selective MPTCP Policy achieves good QoE. Based on our results, we conclude that a combination of the Pessimist Policy and the Selective MPTCP Policy is the most promising approach for IANS.

2 RELATED WORK

Access Network Performance. Streaming videos with a high representation bitrate requires a high available downstream capacity on the path between the server and the end-user device. For 2011, Sommers et al. [23] find that WiFi provides higher up- and downstream capacity and more consistent performance compared to cellular. For the time period between September 2013 to May 2014, Deng et al. [5] compare WiFi and LTE. They find that LTE outperforms WiFi 40% of the time, with a potential capacity difference of more than 10 Mbit/s in either up- or downstream. More recently, industry reports [16] suggest an increase in upstream and downstream capacity for fixed (WiFi uplinks) and mobile (cellular) networks. Often, fixed networks have a higher capacity than mobile, but this depends on geographic region and network provider.

Utilizing Multiple Access Networks. Mobile data offloading can shift traffic from cellular to WiFi networks and improve video quality [11], however, it requires support from the network. MPTCP [8] utilizes multiple access networks by aggregating their capacity. It can improve performance for downloading large files, see, e.g., Raiciu et al. [18]. However, MPTCP is application-agnostic and does not take application or user preferences into account. Moreover, for many applications, the scheduling of data across the different network paths mainly occurs on the server-side, where some application information is only available at the client-side. To make MPTCP application-aware, Corbillon et al. [4] propose a cross-layer scheduler for MPTCP optimized for video streaming. MP-DASH [9] is an overlay to MPTCP which uses application information to selectively enable the secondary MPTCP path, potentially reducing cellular data usage. However, both approaches require support within the kernel of both client and server, while IANS does not modify MPTCP and, therefore, only requires user-space changes on the client. Moreover, both cross-layer MPTCP schedulers involve more extensive modifications to the application than IANS, i.e., they require changes to the ABR or insight into the video content. While some mobile OSes [1] have client-sided MPTCP support, server-sided support is often lacking [2] even for vanilla MPTCP in practice. Evensen et al. [7] propose an HTTP-based approach that distributes video segment loads across multiple networks using HTTP range requests. While this approach does not require any server modifications, it is tightly integrated with the video streaming system and quality adaptation algorithm. Lai et al. [14] design a system to switch between WiFi and cellular on a mobile device to mitigate network disruptions and satisfy application requirements. While their approach is limited to mobile apps that use HTTP, IANS supports all applications that use TCP or UDP. Moreover, their approach does not leverage MPTCP.

Communicating Application Needs. While most contemporary Operating Systems (OSes) support the usage of multiple access networks at the same time, existing approaches are not standardized and often proprietary. For example, mobile OSes often implement a centralized connection manager [28] which enables the use of the cellular network on a per-application basis. More fine-grained decisions require the centralized decision logic to be aware of application requirements or workload properties, such as video segment

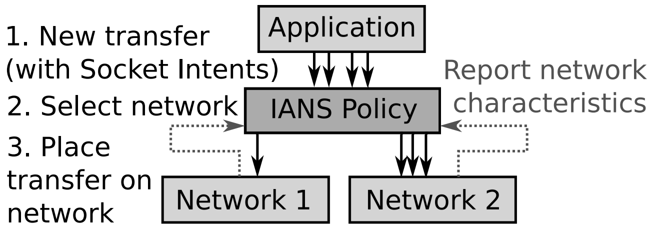

1. New transfer (with Socket Intents)
2. Select network
3. Place transfer on network
Application
IANS Policy
Report network characteristics
Network 1
Network 2

Figure 1: IANS concept.

size or buffer status. However, the Socket API, which is the de-facto standard, does not include such information. Therefore, there is a need for enhanced networking APIs which allow applications to specify their requirements for a new connection or transfer. While NEAT [13] focuses on selecting between different transport protocols, IANS [6] selects between access networks to improve application performance. Building on both of these approaches, the IETF Transport Services Working Group is standardizing an enhanced networking abstraction [26].

3 BACKGROUND

Our approach to using multiple access networks to improve HTTP Adaptive Streaming (HAS) performance is based on Informed Access Network Selection (IANS) [6], for which prior work only covers Web performance. First, we revisit the key concepts of IANS. Then, we describe HAS.

3.1 IANS Concepts

To select the most suitable access network(s), IANS [6] learns about application needs, communicated as Socket Intents, and network performance characteristics. Based on this information, IANS Policies select the network to use for a new connection or transfer.

Communicating Application Needs: Socket Intents. To enable IANS, applications provide hints about what they know, expect, or want to achieve regarding their own traffic by setting Socket Intents [21]. Different from Quality of Service (QoS) requirements, Intents are taken into account in a best-effort manner. They do not guarantee specific performance characteristics for the application but help to use the available network resources more suitably. For example, if a transfer benefits most from short latency, Intents may indicate to prefer the network with the shortest current latency for this transfer. Applications express this preference using the Traffic Category Intent, which they can set to Query for short latency or Bulk for high capacity. Alternatively, applications can set the Size to be Received of an upcoming transfer, which allows the IANS Policy to automatically decide whether to optimize for short latency or high capacity [6].

Selecting Networks: IANS Policies. Based on Socket Intents as well as current network performance characteristics, IANS Policies select the best suitable access network(s) for a new transfer, see Figure 1. (1) An application specifies its Intents for each new transfer. (2) An IANS policy decides to use one or both of the available networks. (3) The new transfer is placed on the chosen access network. IANS continuously monitors the current network performance characteristics of the available networks. As both application needs and network conditions are diverse, there is not a single best strategy to select an access network for all scenarios. Therefore, different IANS Policies exist. While prior work has proposed the Threshold Policy [6] to speed up Web browsing, this approach is insufficient for HAS. Therefore, we design different IANS Policies for HAS. When to switch between IANS Policies or how to combine them is out of scope for this paper.

3.2 HTTP Adaptive Streaming

HAS divides audio and video content into segments of a certain length. Each segment is encoded as different representations, e.g., different quality levels, and stored on a Web server. A HAS client can load the representation that best matches the available resources, such as the device type, screen resolution, and network conditions. To find out which representations exist for specific content, a client first loads a manifest file, which includes a list of representations with their resolutions and bitrates. The client chooses the initial representation for the first segment, loads this segment and decodes the content into the playout buffer. Once there is enough content in the playout buffer, the client starts playout out the content to the user in parallel to downloading more segments.

After each segment, the client can switch to a different representation of the content using an Adaptive Bit-Rate algorithm (ABR). ABRs typically try to select the highest possible representation, i.e., the best possible video quality, while preventing stalling. Stalling occurs when the buffer runs out and media playout is interrupted, e.g., because the client fails to load segments fast enough. To select the best representations, ABRs often utilize information such as throughput estimates and playout buffer level. In our evaluation, we focus on buffer-based ABRs, which are robust to fluctuations in throughput estimates, and, therefore, typically utilized for Video on Demand use cases. In particular, we use the following algorithms:

- **Buffer Based Approach (BBA)** [12]: BBA determines the next representation to be loaded based on the current buffer level. BBA-0 starts with the lowest representation and keeps loading this representation as long as the buffer level is low. If the buffer level is high, BBA-0 switches to a higher representation based on a linear function of the buffer level. While BBA-1 accounts for variable segment sizes and BBA-2 optimizes the start phase of the playout, in our investigations we rely on BBA-0 because our video player supports it.

- **Buffer Occupancy based Lyapunov Algorithm (BOLA)** [25]: Similar to BBA, BOLA determines the next representation based on the current buffer level. Hereby, it computes a function based on the buffer level using a utility maximization function, which aims to minimize stalling and maximize video quality. The function includes a parameter to set the relative importance of stalling to video quality.

4 SELECTING A NETWORK FOR HAS

To realize IANS for video streaming, we focus on HTTP Adaptive Streaming (HAS) because HAS allows us to choose a network for each video segment. Hereby, IANS is aware of the application needs, i.e., the representation selected by the ABR for each segment, and selects the most suitable access network. If the available access networks do not provide sufficient downstream capacity, the ABR can adapt by switching to a lower representation. Due to this flexibility to adapt to network conditions, we see a high potential for IANS to improve HAS performance. To achieve this goal, first, we enable the video player to express its needs for each video segment, i.e., its Socket Intents for each transfer. Then, we design three IANS Policies to optimize for high video quality, for low stalling risk, or to combine the available networks.

4.1 Socket Intents for HAS

For each transfer our HAS client initiates, it expresses its needs as Socket Intents. For loading metadata such as initial manifest files, which are usually small in size, latency has a high impact on load time. Therefore, the client sets the TRAFFIC CATEGORY to QUERY, so a network with short current latency will be selected. For loading video segments, which are larger in size, the available downstream capacity often has a major impact on load time. Therefore, the client sets the TRAFFIC CATEGORY to BULK to signal that downstream capacity is important. Moreover, the client also sets the BITRATE RECEIVED based on the representation that the ABR has selected for the video segment to enable the IANS Policy to estimate load times on different networks. Finally, to inform the IANS Policy of the maximum allowed load time to avoid stalling, the client sets the DURATION to the current buffer level.

4.2 IANS Policies: Design Choices

Based on the Socket Intents for each transfer, our IANS Policies select an access network with the goal of achieving short load times. To explore the question of what to optimize for, we design three different IANS Policies: The OPTIMIST POLICY aims to achieve the highest possible video quality by selecting the network with the highest available downstream capacity. The PESSIMIST POLICY aims to minimize the risk of stalling by selecting a network with sufficient and stable available downstream capacity. The SELECTIVE MPTCP POLICY aims to combine the downstream capacity of all available networks while preventing overloading networks with low downstream capacity.

The currently available downstream capacity, which is critical for all three IANS Policies, may fluctuate during the video load. Therefore, we capture multiple downstream capacity estimates on different time scales: For the recently observed downstream capacity c_{mid}, we use the maximum observed data rate, $DRate_{max}$, during the last 10 seconds. We choose this time window because it captures the capacity observed during the last few segment loads, since video segments often have a playout duration of 2 or 4 seconds [15] and segment load times are usually shorter than segment playout durations. For networks that were not used for the last few segment loads, the observed $DRate_{max}$ may be lower than the actual achievable $DRate_{max}$, so c_{mid} may not be accurate. Therefore, to capture what capacities a network has provided in the past and to identify long-term tendencies, we record c_{long} as the $DRate_{max}$ seen during the last 60 seconds and $c_{verylong}$ for the last 600 seconds. Finally, to see whether the downstream capacity on a recently used network has decreased on short notice, we record c_{short} as the $DRate_{max}$ seen during the last second.

Of these estimates, initially, all IANS Policies use c_{mid}. For each transfer, the OPTIMIST POLICY and PESSIMIST POLICY first calculate the expected load time on each network based on c_{mid} using a calculation similar to the THRESHOLD POLICY [6]. Hereby, the policies estimate the transfer size based on the BITRATE RECEIVED of the representation for the next video segment as well as the segment duration. The network with the shortest expected load time t_{mid} becomes the candidate network, net_{cand}. Then, instead of directly selecting net_{cand}, the OPTIMIST POLICY and PESSIMIST POLICY take the other downstream capacity estimates into account: The OPTIMIST POLICY considers switching to an alternative network based on higher $c_{verylong}$, see Section 4.3. The PESSIMIST POLICY switches if the c_{short} on net_{cand} is too low, see Section 4.4. The SELECTIVE MPTCP POLICY selectively enables MPTCP if sufficient capacity is available, see Section 4.5. It only uses c_{mid} as MPTCP uses all available networks, and, thus, it should generate accurate c_{mid} estimates for all networks.

4.3 Optimist Policy: Considering an Alternative Based on Best Case

Algorithm 1: Optimist Policy.

```
1  Function optimistPolicy ($net_{cand}$, $t_{buffer}$, $t_{long}$, $t_{verylong}$):
2      $net_{alt} \leftarrow$ network with shortest $t_{verylong}$
3      if $t_{buffer} = 0$ then    // Playout not started yet
4          return $net_{alt}$      // Possibly safe to try
5      if $net_{alt}$ not used for last 3 segments then
6          if $t_{long_{alt}} < \frac{2}{3} \cdot t_{buffer}$ then
7              return $net_{alt}$      // Possibly safe to try
8          else if
             $t_{long_{cand}} > \frac{2}{3} \cdot t_{buffer}$ and $t_{long_{alt}} < t_{long_{cand}}$ then
9              return $net_{alt}$      // Not safe, but better
10         else if $net_{alt}$ not used last 10 segments then
11             return $net_{alt}$
12     return $net_{cand}$         // We have not switched
```

Having first determined a candidate network net_{cand}, the Optimist Policy tries to optimize for the highest available capacity. Hereby, it determines whether to switch from net_{cand} to an alternative network net_{alt} according to Algorithm 1. First, it determines whether there is any net_{alt} with a better $c_{verylong}$, therefore, a shorter "best case" load time estimate $t_{verylong}$. In this case, the Optimist Policy switches to net_{alt} if it deems net_{alt} safe to try, i.e., if playout has not started yet, so there is no risk of stalling, or if net_{alt} has not been used for at least three segments and provides an acceptable c_{long}. We choose three segments since the duration of three segments of 4 seconds each exceeds the c_{mid} time of 10 seconds. Here, the Optimist Policy considers c_{long} acceptable if $t_{long_{alt}}$, i.e., the load time on net_{alt} based on c_{long}, is below $\frac{2}{3}$ of the buffer level, so the load is unlikely to stall even with a safety margin. We use a safety margin on $\frac{2}{3}$ because, in our tests, we observe a good balance between expected fluctuations in load time and sufficient flexibility to switch when using this value. If $t_{long_{alt}}$ does not satisfy this condition, we check if $t_{long_{cand}}$ does. If not, neither net_{cand} not net_{alt} is "safe" against stalling, but the Optimist Policy still picks net_{alt} if it has a shorter $t_{long_{alt}}$. Otherwise, in cases in which $t_{long_{alt}}$ is outdated, i.e., net_{alt} was not picked for more than 10 segments, the Optimist Policy selects net_{alt} to give it a chance. We choose the duration of 10 segments based on the segment duration of 4 seconds, so the load times for 10 segments exceed our *longtermEstimate* of 60 seconds by a factor of 1.5. Finally, if the Optimist Policy has not decided to switch to net_{alt}, it stays with net_{cand}.

4.4 Pessimist Policy: Considering an Alternative Based on Worst Case

Algorithm 2: Pessimist Policy.

```
1  Function pessimistPolicy ($net_{cand}$, $t_{segment}$, $t_{buffer}$, $t_{short}$, $t_{mid}$, $t_{long}$):
2      if $t_{short_{cand}} > t_{buffer}$ or $t_{segment}$ then   // Concerned
3          $net_{alt} \leftarrow$ Network with shortest $t_{short}$
4          if $t_{short_{alt}} < t_{buffer}$ then
5              return $net_{alt}$      // Finishes early enough
6      else if $net_{cand}$ used for last segment and
         $t_{short_{cand}} > \frac{4}{3} \cdot t_{buffer}$ then   // Be more concerned
7          if $t_{short_{alt}} < t_{short_{cand}}$ or $t_{long_{alt}} < t_{long_{cand}}$ then
8              return $net_{alt}$      // More ready to switch
9      return $net_{cand}$             // We have not switched
```

Similar to the Optimist Policy, the Pessimist Policy first uses c_{mid} and designates the network with the shortest expected load time as the candidate net_{cand}. Instead of directly selecting net_{cand}, the Pessimist Policy considers switching to another network only if it deems net_{cand} unable to provide sufficient capacity to avoid stalling, see Algorithm 2. First, the Pessimist Policy considers the "worst case" load time on net_{cand}, $t_{short_{cand}}$, based on c_{short} seen during the last second. If $t_{short_{cand}}$ is longer than either t_{buffer} or $t_{segment}$, the Pessimist Policy becomes concerned about stalling. It switches to an alternative network net_{alt} with a shorter t_{short} if $t_{short_{alt}}$ is shorter than t_{buffer}. In this case, it hopes that the segment load will be finished before the buffer runs out, which avoids stalling. If the Pessimist Policy has missed this opportunity to switch, it performs another check if net_{cand} was used for the most recent segment, so $t_{short_{cand}}$ is likely to be accurate. Here, if $t_{short_{cand}}$ is longer than $\frac{4}{3}$ times the t_{buffer}[1], i.e., the segment load may not finish within the deadline including a safety margin, the Pessimist Policy is even more concerned about stalling and, thus, more ready to switch: It switches if net_{alt} provides either a better t_{short} or t_{long}. If neither of the estimates is better for the net_{alt}, the Pessimist Policy stays with the net_{cand}.

4.5 Selective MPTCP Policy

Instead of always selecting a single network to use for a transfer, the Selective MPTCP Policy enables MPTCP for some transfers according to Algorithm 3. As MPTCP provides the most benefits for large transfers, the Selective MPTCP Policy only enables MPTCP when the category is

[1]We use a safety margin of $\frac{4}{3}$ because, in our tests, we observe that the policy accurately detects long load times that lead to stalling when using this value, while for smaller margins this detection becomes inaccurate and for longer margins the policy becomes inflexible.

Algorithm 3: Selective MPTCP Policy.

```
  Input: Transfer with category, $bitrate_{segment}$, $t_{segment}$,
         $t_{buffer}$, $use_{TLS}$
  Networks $n \in \mathcal{N}$ with $RTT_{min}$, reuse, $c_{short}$, $c_{mid}$, $c_{long}$,
  $c_{verylong}$
  Output: Network to use for transfer
1 $net_{short} \longleftarrow$ network with shortest $RTT_{min}$
2 if category = QUERY then
3     return $net_{short}$                                  // No MPTCP
4 else if category = BULKTRANSFER then
5     $c_{min} \longleftarrow$ lowest $c_{mid}$                       // Last 10 seconds
6     if ($t_{buffer} \leq 10$ and $c_{min} > bitrate_{segment}$) or
       ($t_{buffer} > 10$ and $c_{min} > bitrate_{segment}/2$) then
7         return all networks, use MPTCP with first
           subflow on $net_{short}$
8     else          // Insufficient capacity for MPTCP
9         return only network with highest $c_{mid}$
```

set to BULKTRANSFER. Moreover, the Selective MPTCP Policy only enables MPTCP when sufficient downstream capacity is available on all networks. Otherwise, the overhead of MPTCP connections and saturating the congestion window may overwhelm a network with insufficient downstream capacity [6]. Therefore, in the Selective MPTCP Policy, we compare the c_{mid} on the lowest capacity network, c_{min}, to the next representation bitrate, $bitrate_{segment}$. We enable MPTCP if the c_{min} is higher than $bitrate_{segment}$, or if is higher than half the $bitrate_{segment}$ but there is sufficient buffer available which reduces the risk of stalling. We consider a buffer level of 10 seconds sufficient to reduce the risk of stalling here because we have seen this level to be adequate in our tests. Otherwise, we use only the network with the highest downstream capacity without any MPTCP to avoid overloading the lower downstream capacity network.

5 IMPLEMENTATION

We implement the Optimist Policy, the Pessimist Policy, and the Selective MPTCP Policy within the Socket Intents prototype[2]. To enable an application to use these IANS Policies, we modify the video player provided by GPAC[3], an open-source cross-platform multimedia framework, version 0.7.2-DEV. In particular, we enable the GPAC player to set Socket Intents for each of its transfers, e.g., for each video segment, as explained in Section 4.1. Hereby, we modify the player to use the Socketconnect API of the Socket Intents prototype [6]. Through this API, the player specifies the Traffic Category as Query for all manifest files and initial segments. Then, for each video segment, the player sets the Bitrate Received of the next representation to load and the maximum allowed Duration of the transfer as the current buffer level.

[2]The code is available at https://github.com/fg-inet/socket-intents/.

[3]See http://www.gpac.io/. The modified player is available at https://github.com/fg-inet/gpac

Table 1: Representation bitrates and resolutions.

Avg. bitrate (kbps)			Resolution		
RB	BBB	V	RB	BBB	V
201	218	210	480x360	480x360	480x360
395	378	433	480x360	480x360	854x480
500	509	574	854x480	854x480	854x480
892	783	811	854x480	1280x720	1280x720
1498	1474	1422	1280x720	1280x720	1280x720
1992	2087	1861	1280x720	1920x1080	1440x1080
2996	3936	3523	1920x1080	1920x1080	1440x1080

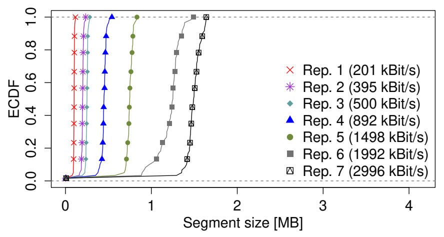
Once our modified GPAC player initiates a new transfer, an IANS Policy is called to select the best access network for each new connection or transfer. The IANS Policy bases this decision upon the Socket Intents of the transfer as well as the current network performance characteristics as detailed in Section 4. While the Socket Intents are communicated by the application, the current network performance characteristics are continuously gathered by the prototype based on the current traffic. The prototype estimates latency based on Smoothed Round Trip Times (SRTTs) of the current TCP connections, for which it periodically queries the TCP stack. To estimate the available downstream capacities, the prototype periodically reads network interface counters and calculates data rates based on the counter increase. To avoid overestimating capacities due to transient traffic spikes, the prototype calculates a smoothed average of counter increases within the last 1 second. Then, to estimate total capacity, the prototype uses the maximum of the smoothed average data rates across a configurable time window, e.g., 10 seconds for the t_{mid}.

6 EVALUATION METHOLODOGY

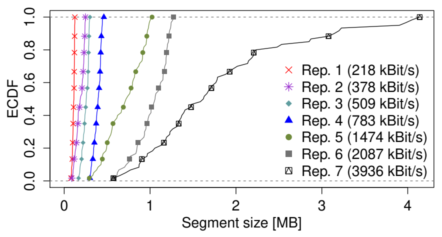
We study the benefits of our IANS Policies for HAS for different video workloads and network scenarios. To enable a systematic evaluation we use a testbed where we have full control over the network performance characteristics.

6.1 Workload

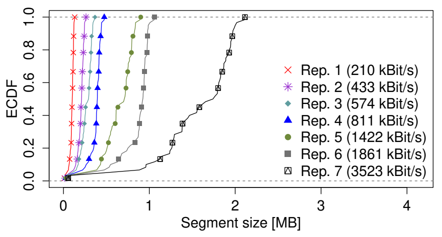
As different workload properties, such as different file size distributions, may influence the effect of IANS on HAS performance, our study includes different workloads. For comparability to prior work, we utilize a well-known HAS data set [15], from which we select three videos of different genres: "Red Bull Playstreets" (RB) as a sports video, "Big Buck Bunny" (BBB) as an animation movie, and "Valkaama" (V) as a live-action movie. We choose videos of different genres as, even while using the same video encoding algorithm,



(a) "Red Bull Playstreets" (RB).



(b) "Big Buck Bunny" (BBB).



(c) "Valkaama" (V).

Figure 2: ECDFs of video segment sizes for different videos in our workload.

the amount of data to be transmitted varies due to differences in content. While these videos have overall durations of around 10 or around 90 minutes, we limit our experiment workload to a fixed subsequence within each video, which simplifies comparing results for different videos. In particular, we choose the first four minutes of each video, as this duration lies within the application range of the P.1203 model [17] (between 1 and 5 minutes), which we use to estimate QoE. The videos are split into segments of 4 seconds.

Each video is available in different representations, i.e., different screen resolutions and target bitrates. We use seven representations for each workload, see Table 1, which correspond to typical resolutions and bitrates used by commercial HAS providers [10]. Our lowest chosen representation has a target bitrate of around 200 kBit/s with a screen resolution of 480x360 pixels. Lower representations correspond to a very small screen resolution, i.e., 320x240 pixels, compared to the display size of our client, which is 1920x1080. Loading a representation with such a small resolution leads to poor QoE even if no stalling events occur. The highest representation we choose corresponds to a high representation in the original workload which provides the same resolution as the screen resolution of our client, 1920x1080. Loading this representation is likely to provide a high QoE [22] and is still possible for network scenarios with high downstream capacity, see Section 6.2.

We depict the video segment size distributions for the different videos and chosen quality representations in Figure 2. For RB, the segment size distribution for each representation remains rather constant with only minor deviations, see Figure 2a. In contrast, segment sizes vary significantly for the other video workloads, particularly for high-quality representations, see Figure 2b and Figure 2c. Here, within the same representation, some segments are much larger than others. As it is common to estimate the segment size based on the average encoding bitrate, such a high spread in segment sizes may lead to inaccurate estimates.

6.2 Network Scenarios

To compare video loads for different IANS Policies under different network conditions in a systematic evaluation, we

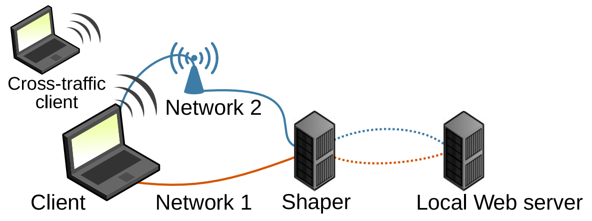


Figure 3: Testbed setup.

use the testbed shown in Figure 3. This testbed represents a scenario where the access network is the performance bottleneck and where the client can reach the server over two different access networks.

To show the IANS behavior under different network conditions, network 1 provides constant downstream capacity, while the downstream capacity on network 2 fluctuates. We focus on varying the downstream capacity because it has a high impact on video segment load times. While in practice, downstream capacity changes occur on all available networks, the simplification of keeping downstream capacity on network 1 constant allows us to explore a variety of variation patterns on network 2 while limiting the overall number of scenarios, therefore, maintaining sufficient computational tractability of the results. To provide constant downstream capacity for network 1, here, the client is connected to the traffic shaper via a wired link using 1 Gbit/s Ethernet. As the wired link only adds minimal delay and no congestion for the traffic, the traffic shaper clearly dominates the network performance characteristics of network 1. To emulate a wireless access network with fluctuating downstream capacity on network 2, here, we use a WiFi AP, so the network performance characteristics are influenced by both the traffic shaper and the effects of the wireless link. The client and WiFi AP run Linux kernel version 4.19, while the server and shaper run version 3.18. The client and server run a modified version of the Linux kernel that supports MPTCP [2].

To include a range of different network behaviors in our study, we vary the capacity on network 2 in two ways: We modify the downstream capacity of the traffic shaper over time, which emulates reductions due to RSS fluctuations in a wireless network, and we introduce cross-traffic to the WiFi and bottleneck link, which emulates downstream capacity reductions due to concurrent traffic. To avoid further capacity variations due to side effects out of our control, we keep the client and AP stationary and use an otherwise unoccupied channel on the 5 GHz band for our wireless link. We use 802.11n with two spatial streams and a 40MHz channel. We show the different network scenarios used in our study in Table 2.

For the **variable capacity scenarios**, we set the median downstream capacity based on factors of the lowest representation of one of our video workloads, 218 kBit/s, for both networks. Since a downstream capacity of 218 kBit/s may be insufficient for loading the lowest representation due to overhead, we scale up the downstream capacity by factors of 1.5, 2, 2.5, 4, 5, and 10 to allow loading different representations. While on network 1 we set the downstream capacity to be constant, we let the downstream capacity on network 2 vary around the median using different variation patterns with a different coefficient of variation c_v. In particular, we vary the downstream capacity according to six different variation patterns seen by HAS sessions using 3G networks in mobile scenarios [19]. Here, we use a factorial design of all combinations of median downstream capacity and capacity variation pattern. As latency, we use 80 ms, as this is the latency seen in the mobile scenarios in which the downstream capacity traces were taken [19].

For the **cross-traffic scenarios**, we keep the downstream capacity on network 1 constant at 2 Mbit/s, which is sufficient for loading a high representation of the video, e.g., with a sufficient screen resolution. Network 2 provides downstream capacity of 5 Mbit/s, which enables us to load an even higher quality representation of the video, but we also introduce TCP cross-traffic to network 2. To impose cross-traffic to the WiFi and bottleneck link, we request files of different sizes using another client on network 2. Here, we impose a self-similar load of TCP flows to fully utilize the shaper link using Harpoon [24][4]. In particular, we study the impact of 1, 2, 3, 4, or 8 concurrent TCP sessions, each of which load files of varying sizes using Harpoon. To see if latency influences our results, here, we study both scenarios with 10 ms and 100 ms of additional latency on both networks.

6.3 Performance Metrics

To evaluate HAS performance, during playout of the video, we log initial playout delay, start and end timestamps for all segment loads, the representation level at which each segment is played out, the buffer status and download rate based on which the ABR has chosen to load this representation, and timestamps at which all frames were rendered. We compute the frequency and duration of stalling events of the video playout both based on the download timestamps of the segments and based on the render times of the frames. As the render times indicate both stalling events due to long segment load times and stalling events unrelated to network conditions, e.g., due to decoder delays in the player, in our evaluation we use the stalling events based on download timestamps, which only include stalling events due to long segment load times.

In addition to collecting streaming metrics, such as initial playout delay, number and duration of stalling events, played out video representations, as well as their oscillations, we compute QoE estimates from these metrics. To limit potential biases of the used QoE model, we use two different models: ITU-T P.1203 [17, 20][5] and the Cumulative Quality Model (CQM)[6] [27]. We use two models to make our results robust to the effects of a single model, as we have seen artifacts of the P.1203 model, see Footnote 8. We compare the MOS values computed using P.1203 to the MOS values computed using CQM at the end of each video load and find we can draw the same conclusions based on both models. In particular, the relative differences of the median MOS values for two different IANS Policies or scenarios are similar when using either P.1203 or CQM. This confirms that our results are robust to the used QoE model. However, we note that the absolute MOS values vary for the two models. Although we use the same audiovisual quality scores as input to both P.1203 and CQM, we observe that the MOS scores computed using P.1203 are generally higher than using CQM by between 0.3 and 0.5. For example, while in theory, the MOS

Table 2: Emulated network scenarios.

Property	**Levels** (variable capacity scenarios)	**Levels** (cross-traffic scenarios)
Median downstream capacity:	218, 327, 436, 545, 872, 1090, or 2180 kBit/s.	2 or 5 MBit/s.
c_v for downstream capacities:	0, 0.3, 0.34, 0.45, 0.49, 0.6, or 0.7.	0.
Additional latency:	80 ms.	10 or 100 ms.
Concurrent TCP sessions:	None.	1, 2, 3, 4, or 8.

[4]We configured Harpoon to generate TCP flows with an average total throughput similar to the shaped downstream capacity, whereby the file sizes follow a Pareto distribution with alpha=1.2 and shape=1500 bytes and the inter-connection times follow an exponential distribution with a mean of one second.

[5]We use the code provided at https://github.com/itu-p1203/itu-p1203 in mode 0.

[6]We use the code provided at https://github.com/TranHuyen1191/CQM with Tran's Window quality model.

can range between 1 and 5, the highest MOS value we observe is 4.3 for P.1203 and 3.9 for CQM. In this case, the highest available representation with an audiovisual score of about 4.1 is played out continuously and no stalling occurs. Here, P.1203 slightly increases the final MOS score because there are no stalling events. In contrast, CQM considers recent minimum, maximum, and average quality scores and produces a slightly lower MOS.

6.4 Course of Experiments

We load the videos in our workload using the IANS-enabled video player, see Section 5. As ABRs, we select BBA-0 [12] and BOLA [25]. We run experiments with different ABRs to make our results less dependent on any particular ABR. While our workload data set [15] is widely used, it does not include any audio. Thus, we emulate audio by periodically loading a file of 100 KB in parallel to the video segments, which corresponds to audio at a bitrate of 192 kBit/s with a duration of 4 seconds. We compare the following access network selection policies to load the video segments: Loading the video using only a single network, using MPTCP for all transfers, and using three IANS Policies: The Optimist Policy, see Section 4.3, the Pessimist Policy, see Section 4.4, and the Selective MPTCP Policy, see Section 4.5. For MPTCP, we use the Lowest-RTT-first scheduler. Note that more advanced cross-layer schedulers would require modifications to both the server and the application, therefore, they would not be directly comparable with IANS within the same experimental setup. For the audio segments, our IANS Policies choose the network which is not currently used for the video segments. This has the side effect of getting performance estimates for these networks. We repeat our experiments five times for all videos.

We run each experiment for a fixed duration because this allows us to directly compare different experiment runs with each other: For the variable capacity scenarios, downstream capacity varies over time according to the same pattern during each video load. Therefore, each video load experiences the same changes in network conditions at the same point in time during the experiment. We fix our experiment duration at 240 seconds as this duration satisfies our criteria: One the one hand, it is sufficient to load enough video segments for our IANS Policies to show effects. On the other hand, the length of the video content is within the application range of the P.1203 model [17]. Note that a fixed experiment duration implies that we may load less content for experiments in which stalling occurs. For such cases, we may have to compare MOS values computed based on different durations of video content. To limit possible biases, we only include video loads with between 120 and 240 seconds of content in our results. We choose this duration in analogy to our results for CQM: Here, the relative differences between two video loads after the first 120 seconds of content are usually similar to the differences after loading the full 240 seconds. Furthermore, we look at MOS values computed using P.1203 based on shorter content durations, i.e., for the first 120, 150, 180, and 210 seconds of content for each video load. For the cross-traffic scenarios, the results look identical to MOS values based on 240 seconds of content. For the variable capacity scenarios, our results still hold true even for shorter content durations, with two exceptions: In the capacity decrease scenarios, shorter content durations do not capture the decrease in capacity. In some scenarios, we observe outliers due to weighting factors and fixed thresholds in the P.1203 model, cf. Footnote 8. We repeat our experiment 5 times for each combination of scenario, policy, and ABR. In our evaluation, we compute the median MOS with confidence intervals of the median for each combination of scenario and policy, i.e., for up to 15 MOS values.

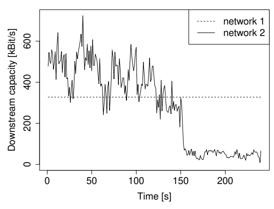
7 EVALUATION RESULTS

Next, we evaluate the benefits of IANS for HAS. First, we discuss a single scenario, the Capacity Decrease Scenario, in detail. Then, we show the results of our systematic study for scenarios with variable capacity. Finally, we show the results for our cross-traffic scenarios.

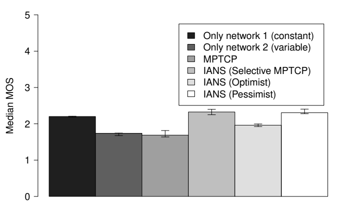
7.1 Capacity Decrease Scenario: In-Depth Discussion

First, we focus on a scenario in which downstream capacity for network 1 stays constant during the video load, but downstream capacity for network 2 decreases drastically. Such a scenario may occur, e.g., if a mobile device moves out of range of a WiFi AP or if it stays within the range, but experiences a low RSS. Our motivation for starting with this scenario is to illustrate how IANS adapts to the downstream capacity changes. Figure 4 shows the shaped downstream capacity as well as our results for the RB video. Results for the other videos are similar. In this scenario, we shape downstream capacities according to Figure 4a, i.e., 327 kBit/s throughout the experiment on network 1, while the downstream capacity on network 2 varies around the same median with a c_v of 0.7 with a sharp decrease in capacity: Initially, downstream capacity on network 2 is higher than on network 1, but after around 150 seconds, the downstream capacity on network 2 decreases to between 30 and 60 kBit/s.

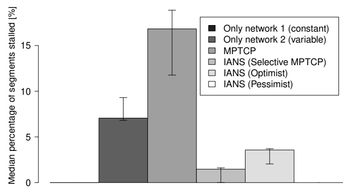
Figure 4b shows the QoE as MOS computed using the ITU-T P.1203 model for all ABRs. Here, IANS outperforms a single network and MPTCP, whereby the Pessimist Policy and the Selective MPTCP Policy yield the highest MOS values. These QoE improvements occur because both the Pessimist Policy and the Selective MPTCP Policy reduce stalling



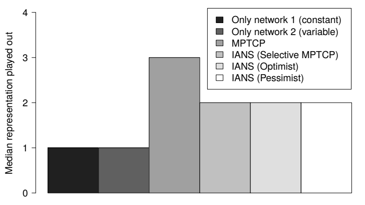
(a) Shaped capacity.



(b) QoE (median MOS with CI).



(c) Stalling (median with CI).



(d) Representations (median).

Figure 4: Capacity Decrease scenario.

events, see Figure 4c. Here, IANS recognizes the decreased downstream capacity on network 2 and, therefore, uses only network 1 after downstream capacity decreases. Stalling may still occur in cases where the downstream capacity decreases at the same time at which the IANS Policy decides which network to use for a transfer, therefore, IANS cannot detect the downstream capacity change for this transfer yet. In contrast, MPTCP continues to use both network 1 and network 2 for all transfers, which leads to stalling for all segments loaded after the capacity decrease. This leads to a mean stalling percentage of about 15% of segments for MPTCP because it is able to load about 10 more segments after the capacity decrease, all of which stall the playout. In contrast, network 2 is only able to load 3 more segments during the remainder of the experiment due to very long load times, which results in a lower overall percentage of stalled segments around 8%.

In Figure 4d, we show that IANS not only reduces stalling, but also enables to load a higher mean representation than any single network. While MPTCP is able to load an even higher mean representation, this does not result in a higher QoE due to frequent stallings. The Optimist Policy achieves a lower QoE than the other IANS Policies: While it allows higher representations of the video to be loaded, it still sees several stalling events. We find that these stalling events occur because, after the downstream capacity on network 2 decreases, the Optimist Policy attempts to use network 2 for every fourth segment because it has seen high downstream capacity on this network in the past. As the changes in QoE reflect changes in streaming metrics such as stalling and played out representations, the remainder of the paper focuses on QoE metrics.

Summary of single scenario: IANS can detect a persistent decrease in downstream capacity and use a network with a more stable downstream capacity, thus, reduce stalling and improve QoE. In particular, the Pessimist Policy and the Selective MPTCP Policy provide good results for the Capacity Decrease scenario.

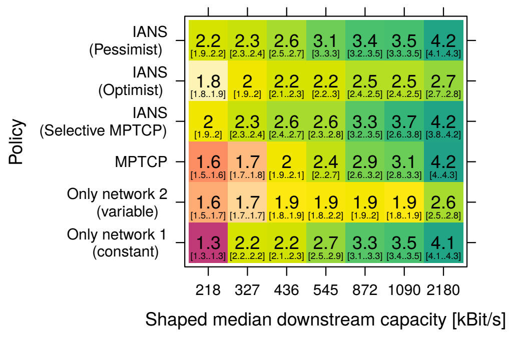
7.2 Systematic Study of Variable Capacity Scenarios

Next, we present the results of our systematic study of scenarios with variable capacity, in which we keep the downstream capacity on network 1 constant and vary the downstream capacity on network 2 during each run, recall Table 2. In total, our study consists of 42 variable capacity scenarios, whereby each scenario corresponds to a combination of median downstream capacity and capacity variation pattern. In this paper, we show the results for two capacity variation patterns: The Capacity Decrease scenarios ($c_v = 0.7$), introduced in Section 7.1, and the "ferry" scenarios ($c_v = 0.49$). We focus on these scenarios because the results for the other variable capacity scenarios are similar to the "ferry" scenario.

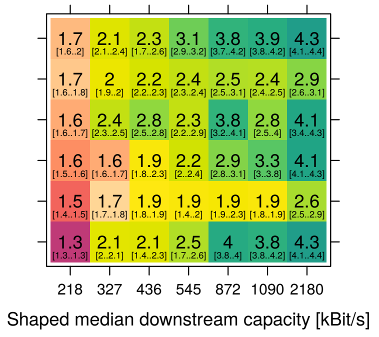
For each scenario, we show the QoE achieved by different access network selection policies using heatmaps, see Figure 5. Each subplot represents a combination of video and capacity variation pattern. Within each subplot, we scale the median downstream capacities, so each column corresponds to a single scenario and shows the achieved QoE for different access network selection policies. Hereby, IANS Policies are displayed at the top and results for using only a single network or using MPTCP for all transfers are displayed below. Each heatmap entry shows the QoE as MOS values computed using the ITU-T P.1203 model. Since the different ABRs often yield a similar QoE, we show the median MOS for all ABRs for the same scenario and access network selection policy. Furthermore, each heatmap entry contains the median and the corresponding confidence interval. Note, the size of the confidence interval can be large, i.e., with MOS differences of more than 1, because for some video loads, the P.1203 model penalizes the computed MOS due to frequent representation switches[7]. The color schema (same for all plots) ranges from violet and red for MOS values below 2 over light yellow for MOS values between 2 and 2.5 to green for MOS values of 2.5 or more. Overall, in Figure 5, green dominates the results for IANS and MPTCP as well as for scenarios with high downstream capacities, whereas red and violet are more common for using a single network and for low downstream capacities.

For the RB video in the Capacity Decrease scenarios, shown in Figure 5a, IANS yields MOS improvements for scenarios with low downstream capacities, i.e., of 545 kBit/s or less. For

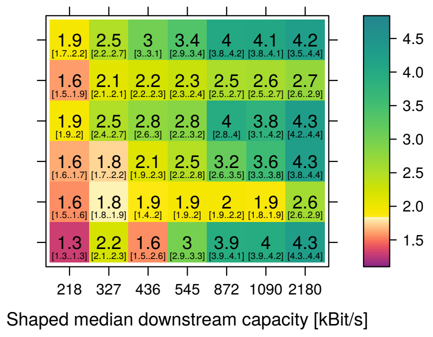
[7]In particular, we find that P.1203 heavily penalizes playouts in which the played out representation changes more frequently than every 30 seconds even for cases in which two playouts are otherwise identical, i.e., they include the same number of stalling events and similar played out representations.



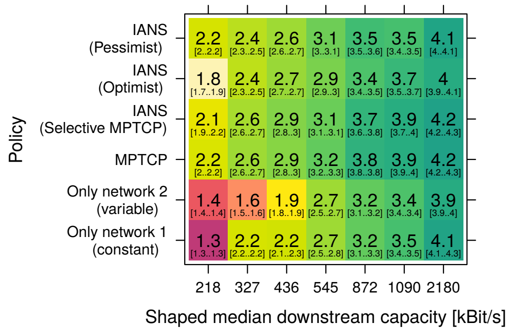
(a) RB, Capacity Decrease (c_v =0.7).



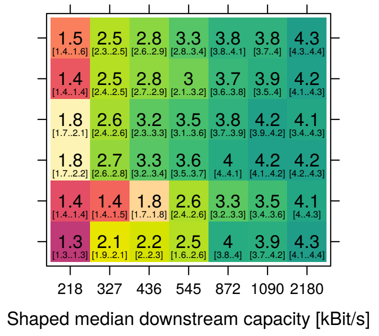
(b) BBB, Capacity Decrease (c_v =0.7).



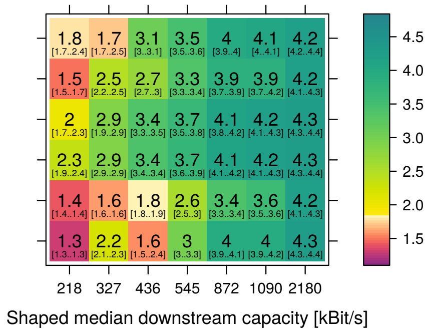
(c) V, Capacity Decrease (c_v =0.7).



(d) RB, Ferry (c_v =0.49).



(e) BBB, Ferry (c_v =0.49).



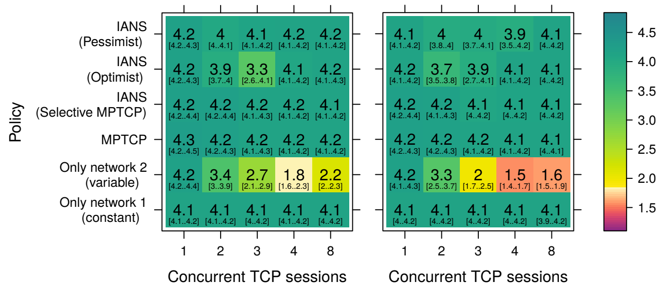
(f) V, Ferry (c_v =0.49).

Figure 5: Median MOS with confidence intervals for all ABRs and two variable capacity scenarios.

the scenario with a median capacity of 218 kBit/s, both the Pessimist Policy and the Selective MPTCP Policy provide a median MOS of 2 or more for RB, while a single network and MPTCP yield a MOS of 1.6 or less. While MPTCP hurts performance due to stalling, IANS is able to improve the MOS because it is aware of downstream capacity changes, e.g., the decrease in downstream capacity on network 2. Therefore, IANS loads video segments over the higher downstream capacity network 1, which reduces stalling events or avoids stalls entirely, as seen previously in Section 7.1.

For the BBB video, see Figure 5b, IANS leads to worse performance than RB for the lowest downstream capacity scenario with a median downstream capacity of 218 kBit/s. This is because BBB's lowest representation has a slightly higher bitrate than RB, so there is insufficient capacity to load even the lowest representation without stalling. For scenarios with downstream capacities between 327 kBit/s and 545 kBit/s, similar to RB, both the Pessimist Policy and the Selective MPTCP Policy are able to select a network with sufficient downstream capacity and, therefore, improve the MOS compared to using a single network or MPTCP. Surprisingly, for BBB, the Selective MPTCP Policy performs worse for 545 and 1090 kBit/s than for 436 and 872 kBit/s because stalling events occur for the former cases. Here, the Selective MPTCP Policy enables MPTCP based on the assumption that sufficient capacity is available for the selected representation, however, the loaded video segment is unusually large. This occurs only for BBB, as BBB has the highest spread in video segment sizes, recall Figure 2b. Future work may fine-tune the Selective MPTCP Policy for such workloads. Results for V are similar to RB, see Figure 5c, except for a low MOS score for the 436 kBit/s scenario and network 1. Here, we see an interaction between the shaped capacity and the representations' segment sizes for this particular video, which causes the ABR to switch representations frequently, which causes P.1203 to penalize MOS.

For the "ferry" scenario, shown in Figure 5d, all IANS Policies outperform using either single network, e.g., improving a MOS of 2.2 or 1.9 on network 1 or network 2 to a MOS of 2.7 or 2.6 using the Optimist Policy and Pessimist Policy or even 2.9 using the Selective MPTCP Policy. Here, MPTCP yields similar MOS improvements as the Selective MPTCP Policy, as there is always sufficient capacity available to use MPTCP. For BBB, see Figure 5e, we again see low MOS values for the Optimist Policy and the Pessimist Policy for the 218 kBit/s scenario, as both policies can only use a single network at a time, which, here, provides an insufficient capacity to load even the lowest representation. For scenarios with capacities between 327 kBit/s and 545 kBit/s, IANS yields even higher MOS improvements for BBB than for RB. At 872 kBit/s and above, the Optimist Policy and Pessimist Policy yield a similar performance as using network 1, and



(a) RB, Latency = 10 ms. **(b) RB, Latency = 100 ms.**

Figure 6: Median MOS with confidence intervals for RB with variable TCP cross-traffic.

only the SELECTIVE MPTCP POLICY or MPTCP is able to improve performance. Finally, V generally yields similar results as RB and BBB, see Figure 5f. Again, we see a few cases of interactions between the workload representation and the shaped capacity, similar to the Capacity Decrease scenario.

Summary of systematic study: (1) IANS improves QoE in cases with low available downstream capacity by avoiding stalling and enabling to load higher video representations. In such scenarios, the PESSIMIST POLICY often improves QoE compared to using only a single network and outperforms the OPTIMIST POLICY in most cases. (2) While MPTCP improves QoE in most variable capacity scenarios, it hurts performance for the Capacity Decrease scenario. (3) In such a scenario, the SELECTIVE MPTCP POLICY improves QoE by avoiding using the low downstream capacity network. This shows the potential of using IANS to selectively enable MPTCP only for cases in which it benefits performance. Going forward, IANS should combine the PESSIMIST POLICY with the SELECTIVE MPTCP POLICY by allowing it to enable MPTCP.

7.3 Cross-Traffic Scenarios

After varying the shaped downstream capacity on network 2, we now focus on scenarios in which we keep the shaped downstream capacity constant but introduce variable TCP cross-traffic, recall Table 2. Here, network 1 provides a constant downstream capacity of 2 Mbit/s without any cross-traffic, while downstream capacity on network 2 is constant at 5 Mbit/s with cross-traffic on the bottleneck link. We vary the latency on both networks and show results for scenarios with short latencies (10 ms) and with large latencies (100 ms).

Figure 6 shows a heatmap of the QoE for different cross-traffic scenarios and ABRs when loading the RB video. Figure 6a shows the results for short latency scenarios and Figure 6b for large latency scenarios. Within each subplot, we show scenarios with 1, 2, 3, 4, and 8 concurrent harpoon sessions loading files of varying sizes. Each heatmap entry shows the achieved median MOS with confidence intervals for a specific scenario and access network selection policy. The color scheme is the same as in Section 7.2.

IANS yields a high MOS even with concurrent cross-traffic of up to 8 sessions, while cross-traffic decreases the MOS for network 2. This confirms that IANS can detect the decreased downstream capacity due to cross-traffic and, therefore, uses network 1 or MPTCP. In some cases, the OPTIMIST POLICY decreases performance, as it attempts to use network 2 occasionally. Results for BBB and V are similar. However, for BBB, we see an interaction between the shaped downstream capacity on network 1 and the BOLA_BASIC ABR. Here, BOLA_BASIC continuously switches representations, which leads to a decreased MOS on network 1.

Summary for cross-traffic scenarios: For cases with high current cross-traffic on a network, IANS avoids using this network, thus, it avoids MOS degradations.

8 CONCLUSION

Mobile users still experience poor video quality due to bad network performance. This mostly stems from using one single access network although multiple different networks would be available. Recent developments like Informed Access Network Selection (IANS) allow overcoming this limitation by choosing between multiple networks dynamically. IANS allows defining policies to appropriately decide which network to select based on application requirements and network performance characteristics.

In this paper, we design three IANS Policies to improve the performance of HTTP Adaptive Streaming (HAS) and evaluate their performance improvements in a controlled testbed. We find that IANS is able to improve HAS performance whereby it yields the highest performance benefits in scenarios with low available downstream capacity. Also, in a scenario with decreasing capacity, IANS can improve the estimated MOS from 2.1 to 2.8 while outperforming MPTCP, which continues to use a network with insufficient capacity.

Future work should further explore the combination of IANS and MPTCP. Moreover, there is potential for optimization by not just selecting between different access networks, but also between different endpoints, e.g., hosted on different CDN nodes. Finally, the potential of IANS should be explored for more applications, such as messaging and file uploads.

ACKNOWLEDGMENTS

This work has been supported by Leibniz Prize project funds of DFG - German Research Foundation: Gottfried Wilhelm Leibniz-Preis 2011 (FKZ FE 570/4-1).

The original concept of IANS and earlier versions of the Socket Intents prototype were developed in collaboration with Philipp Tiesel.

REFERENCES

[1] Olivier Bonaventure. 2018. Apple uses Multipath TCP. Retrieved April 04, 2020 from http://blog.multipath-tcp.org/blog/html/2018/12/15/apple_and_multipath_tcp.html.

[2] Olivier Bonaventure. 2018. Which servers use Multipath TCP? Retrieved April 04, 2020 from http://blog.multipath-tcp.org/blog/html/2018/12/19/which_servers_use_multipath_tcp.html.

[3] Valentin Burger, Thomas Zinner, Lam Dinh-Xuan, Florian Wamser, and Phuoc Tran-Gia. 2018. A generic approach to video buffer modeling using discrete-time analysis. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 2s (2018), 33.

[4] Xavier Corbillon, Ramon Aparicio-Pardo, Nicolas Kuhn, Géraldine Texier, and Gwendal Simon. 2016. Cross-layer scheduler for video streaming over MPTCP. In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 7.

[5] Shuo Deng, Ravi Netravali, Anirudh Sivaraman, and Hari Balakrishnan. 2014. Wifi, lte, or both?: Measuring multi-homed wireless internet performance. In *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 181–194.

[6] Theresa Enghardt, Philipp S. Tiesel, Thomas Zinner, and Anja Feldmann. 2019. Informed Access Network Selection: The Benefits of Socket Intents for Web Performance. In *International Conference on Network and Service Management (CNSM)*. IEEE, 295–300. https://doi.org/10.23919/CNSM46954.2019.9012714

[7] Kristian Evensen, Dominik Kaspar, Carsten Griwodz, Pål Halvorsen, Audun Hansen, and Paal Engelstad. 2011. Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 57–68.

[8] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental). , 64 pages. https://doi.org/10.17487/RFC6824

[9] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 129–143.

[10] YouTube Help. 2019. Live encoder settings, bitrates, and resolutions. Retrieved June 25, 2019 from https://support.google.com/youtube/answer/2853702?hl=en.

[11] Donghyeok Ho, Gi Seok Park, and Hwangjun Song. 2018. Mobile data offloading system for video streaming services over SDN-enabled wireless networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 174–185.

[12] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2015. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM Computer Communication Review* 44, 4 (2015), 187–198.

[13] Naeem Khademi, David Ros, Michael Welzl, Zdravko Bozakov, Anna Brunstrom, Gorry Fairhurst, Karl-Johan Grinnemo, David Hayes, Per Hurtig, Tom Jones, et al. 2017. NEAT: a platform-and protocol-independent internet transport API. *IEEE Communications Magazine* 55, 6 (2017), 46–54.

[14] Zeqi Lai, Yong Cui, Yimin Jiang, Xiaomeng Chen, Y Charlie Hu, Kun Tan, Minglong Dai, and Kai Zheng. 2017. Wireless network instabilities in the wild: Prevalence, App (non) resilience, and OS remedy. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 1–10.

[15] Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 89–94.

[16] Ookla. 2018. Speedtest Market Reports. Retrieved May 21, 2019 from https://www.speedtest.net/reports/.

[17] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1. In *Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, Erfurt. https://doi.org/10.1109/QoMEX.2017.7965631

[18] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. 399–412.

[19] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 114–118.

[20] Werner Robitza, Steve Göring, Alexander Raake, David Lindegren, Gunnar Heikkilä, Jörgen Gustafsson, Peter List, Bernhard Feiten, Ulf Wüstenhagen, Marie-Neige Garcia, Kazuhisa Yamagishi, and Simon Broom. 2018. HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software. In *9th ACM Multimedia Systems Conference*. Amsterdam. https://doi.org/10.1145/3204949.3208124

[21] Philipp S. Schmidt, Theresa Enghardt, Ramin Khalili, and Anja Feldmann. 2013. Socket Intents: Leveraging Application Awareness for Multi-access Connectivity. In *ACM CoNEXT*. ACM, New York, NY, USA, 295–300. https://doi.org/10.1145/2535372.2535405

[22] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2015. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials* 17, 1 (2015), 469–492.

[23] Joel Sommers and Paul Barford. 2012. Cell vs. WiFi: on the performance of metro area mobile connections. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 301–314.

[24] Joel Sommers, Hyungsuk Kim, Paul Barford, and Paul Barford. 2004. Harpoon: a flow-level traffic generator for router and network tests. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 32. ACM, 392–392.

[25] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[26] Brian Trammell, Michael Welzl, Theresa Enghardt, Gorry Fairhust, Mirja Kuehlewind, Colin Perkins, Philipp Tiesel, Christopher Wood, and Tommy Pauly. 2020. *An Abstract Application Layer Interface to Transport Services (work in progress)*. Internet Draft draft-ietf-taps-interface-06. IETF.

[27] Huyen T. T. Tran, Nam Pham Ngoc, Tobias Hoßfeld, Michael Seufert, and Truong Cong Thang. 2019. Cumulative Quality Modeling for HTTP Adaptive Streaming. arXiv:arXiv:1909.02772

[28] M. Wasserman and P. Seite. 2011. Current Practices for Multiple-Interface Hosts. RFC 6419 (Informational). , 21 pages. https://doi.org/10.17487/RFC6419

A REPRODUCIBILITY OF OUR RESULTS

This appendix explains the steps necessary to reproduce the results in this paper. It describes where to find the software that is necessary to use IANS for HAS, how to run an experiment, how to evaluate the resulting data, and where to find the dataset that we use in our evaluation.

The experiment scripts and references to all required source code and data is available at https://github.com/fg-inet/MMSys2020_Informed-Access-Network-Selection. The DOIs for persistent storage of our artifacts are:

- **Data**: https://doi.org/10.5281/zenodo.3756984
- **Code**: https://doi.org/10.5281/zenodo.3757021

A.1 Overview of Artifacts

Related to this paper, we present the following artifacts:

- The **Socket Intents prototype, which implements Informed Access Network Selection (IANS)**, including the IANS Policies used for HAS: The Optimist Policy, the Pessimist Policy, and the Selective MPTCP Policy. The Socket Intents prototype is available on Github: https://github.com/fg-inet/socket-intents/
- Our **modified version of the GPAC video player**, which can use the Socket Intents prototype to load videos using HAS, thus, taking advantage of the prototype. Our fork of the GPAC player is available on Github: https://github.com/fg-inet/gpac
- Scripts that we used to **run our experiments and to evaluate the results**. These scripts are available on Github: https://github.com/fg-inet/MMSys2020_Informed-Access-Network-Selection or at https://doi.org/10.5281/zenodo.3757021
- **A Virtual Machine (VM)** on which both the Socket Intents prototype and the modified GPAC player are installed. This VM is available on Zenodo, along with the dataset of the paper's evaluation: https://doi.org/10.5281/zenodo.3756984
- The **dataset** on which the evaluation in the paper was based. The dataset is available on Zenodo, along with the virtual machine: https://doi.org/10.5281/zenodo.3756984

A.2 Experiment Runs Using the Virtual Machine

The virtual machine (VM) available on Zenodo has both the Socket Intents prototype and the modified GPAC player installed. Furthermore, it contains the same experiment scripts that were used in the paper. Next, we describe how to run an experiment using our VM.

(1) Download the VM from Zenodo: https://doi.org/10.5281/zenodo.3756984 (File name: **ians-video.ova**)
(2) Import the virtual machine as an appliance into virtualbox or similar suitable software. Boot the machine and log in using the following credentials: User name: **osboxes**, password: **osboxes.org**
(3) Once booted, open a terminal: Click on "Activities" in the top left corner, search for "Terminal".
(4) (Optional) Adjust parameters in stream_video.sh, e.g., video playout duration, Adaptive BitRate algorithm (ABR) to use, and IANS Policies to use. This will greatly influence the duration of the test. Note that the VM does not support Multipath TCP (MPTCP), so running this policy will not have an effect.
(5) (Optional) Choose a workload by creating a text file containing the URL of an MPEG-DASH .mpd file. By default, the script will load the "Red Bull" video [15].
(6) Run `./run.sh` (or, if you want to use a different URL file, `./run.sh default.conf $URLFILE` using a different URL file) to run a test. By default, the test will only load the first 60 seconds and only use BOLA_O as ABR, as the test becomes very long otherwise. By default, the test will first load the video using only network interface 1, then network interface 2, then both using the Optimist Policy, then both using the Pessimist Policy. Note that for the RB video, it may take a long time until any non-blank frame is painted because the VM is very slow.
(7) Find the experiment output directories using `cd data/; ls` - each experiment run corresponds to one directory. See Section A.4 for instructions how to analyze the data.

A.3 Experiment Runs Using Your Own Setup

Prerequisites:

- A Linux machine (tested on Debian and Ubuntu) and being able to access a server hosting the video workload via at least two network interfaces. Note that you may have to configure policy routing so that outgoing packets are routed over the correct interfaces.
- The build-essentials package for building C code, Python3 (including packages scipy and matplotlib), R (including packages zoo, lattice, and viridis, plus dependencies), wget, and all dependencies of the software installed below.

Minumum steps necessary to prepare your setup for running experiments:

(1) Build and install the Socket Intents prototype, see README.md in the relevant repository. Create config files for each IANS Policy you want to use as shown in the "Testing the Socket Intents Prototype" section.

(2) Build and install the modified GPAC player, see repository
(3) Download the P.1203 repository and the necessary dependencies, such as python3-matplotlib and python3-scipy. Create a symbolic link from the evaluation script directory, performance-test/video/eval, to the directory called itu_p1203 within the P.1203 repository. Note: Make sure you do not create this link to the top-level directory of P.1203, but the subdirectory that contains the actual code. The script needs to be able to execute this code, which you can test, e.g., by invoking `python3 -m itu_p1203 $FILE` from the command line.
(4) From from the evaluation script directory, performance-test/video/eval, create a symbolic link to your "data" directory. E.g. run the following in the directory: `ln -s /home/yourname/data data`
(5) From the top-level directory of this repo, create the following symbolic links:

```
ln -s performance-test/ video/load_stuff.sh
load_stuff.sh
ln -s performance-test/video/stream_video.sh
stream_video.sh
ln -s performance-test/ video/run.sh run.sh
```

A.4 Analysis of the Data

This section describes how to reproduce plots similar to the ones shown in the evaluation. The repository includes many more scripts, such as to produce time series of the logged data. Note that the MOS values (QoE) plotted here is computed using the P.1203 model.

(1) In performance-test/video/eval (either on the VM or on your own machine with this repository checked out), execute `./stallings.R` to compute stall events and durations.
(2) Run `./dump_json.py` to compute produce the QoE values using P.1203.
(3) Run `./plot_qoe.R` to produce barplots for the QoE.
(4) Run `./heatmap_video.R` to produce heatmaps.

Note: If you have multiple runs in data/, you may want to explicitly select the run(s) to plot or compute data for, e.g., to only compute data for the first run, you must execute `./stallings.R 1 1` (start with the first run and only compute data for this one run). To plot data for the first four runs, execute `./plot_qoe.R 1 4`

To compute and plot MOS values using the CQM model instead, do the following.

Note: This required Matlab.

(1) (Similar to above) In performance-test/video/eval (either on the VM or on your own machine with this repository checked out), execute ./stallings.R to compute stall events and durations.
(2) Run prepare_cqm_data.py to compute the initial delays, segment quality arrays and arrays of stallings and stalling durations, which are required as input to the CQM model.
(3) Use the Matlab scripts read_my_csv.m and the scripts in the CQM repository to compute the actual QoE values.
(4) Run cqm_heatmap.R to produce heatmaps.

A.5 Structure of the Dataset Used in the Paper

The dataset of the original experiment runs contains logs produced by the Socket Intents prototype and our modified GPAC player during experiments conducted between 20th June and 02nd September 2019.

In these experiments, we load the first four minutes of each of the following videos:

- "Big Buck Bunny" (BBB), an animation video,
- "Red Bull Playstreets" (RB), a sports video, and
- "Valkaama" (V), a movie.

We conduct our experiments in a fully controlled testbed, using both scenarios with variable network capacity and with cross-traffic.

For easier handling, our dataset is split into the following files:

- **variable_bbb.tar.gz**: Results for the variable capacity scenarios when loading "Big Buck Bunny",
- **variable_rb.tar.gz**: Results for the variable capacity scenarios when loading "Red Bull Playstreets",
- **variable_v.tar.gz**: Results for the variable capacity scenarios when loading "Valkaama", and
- **crosstraffic.tar.gz**: Results for the cross-traffic scenarios when loading all videos.

Each .tar.gz file contains different the different runs as directories. Each directory name corresponds to one run, annonated with the date and time at which the experiment started and with the network conditions emulated in the network scenario for this particular run.

To analyze data, please refer to Section A.4.