# An ns-3 distribution supporting MPTCP and MPEG-DASH obtained by merging community models

Vitalii Poliakov, Damien Saucez, Lucile Sassatelli

# An ns-3 distribution supporting MPTCP and MPEG-DASH obtained by merging community models

Vitalii Poliakov
Université Côte d'Azur, CNRS, I3S, France
poliakov@i3s.unice.fr

Damien Saucez
Université Côte d'Azur, Inria, France
damien.saucez@inria.fr

Lucile Sassatelli
Université Côte d'Azur, CNRS, I3S, France
sassatelli@i3s.unice.fr

## ABSTRACT

MPEG-DASH and MPTCP are two technologies growing in interest and put together they promise greater quality of experience for video consumers and better network resource usage. However, while independent MPEG-DASH and MPTCP implementations exist for ns-3, they are not directly usable together as they suffer from incompatibilities. In this work, we introduce a new ns-3 distribution that packages the AMuSt Framework DASH implementation and the ns-3 MPTCP implementation from University of Sussex such that they can be used together to nourish the flourishing research on Internet video streaming.

## 1 INTRODUCTION

As volumes of online video streaming continue to grow, researchers seek ways to make it more efficient. Apart from developing better client- or server-side solutions, the research on the video delivery over the network remains a difficult-to-approach domain owing to the need to run large-scale experiments. Such experiments should involve modern video delivery technologies like HTTP Adaptive Streaming (HAS) [1, 6] and Content Delivery Networks (CDN), as well as complex network topologies. For these reasons, most of the research in this domain remains done by teams in collaboration with large CDNs and content providers (e.g., Akamai and Netflix).

Despite that ns-3 is an alternative to having a real deployment, existing ns-3 models are quite diverse and have a rather loose compatibility between each other. It results that researchers often have to spend efforts to make several ns-3 modules gleaned from the Internet work together.

MPEG-DASH and MPTCP are two technologies that one would naturally study together. However, even though ns-3 community provides MPEG-DASH and MPTCP implementations, it happens that they are not directly usable due to incompatibilities. In this work, we present an ns-3 distribution that packages the AMuSt Framework DASH implementation [5] and the ns-3 MPTCP implementation from University of Sussex [4] such that they can be used together to nourish the flourishing research on Internet video streaming.

In the following, we present the efforts that were needed to package MPTCP and MPEG-DASH models together into ns-3. As we show further, the seemingly simple task of merging community models turned out to be tricky as they were based on different ns-3 releases. In order to make the two models work together, we first had to overcome the incompatibilities between different ns-3 versions, and then to extend the MPTCP implementation of ns-3 to support functionalities needed by MPEG-DASH module.

This work is made as part of a work on simulating a large-scale multipath-enabled video delivery system which is now under development. The source code of the resulting distribution is available on GitHub at:
https://github.com/vitaliipoliakov/ns3-dash-mptcp

## 2 SELECTING MODELS FOR MULTIPATH AND MPEG-DASH

Multipath transport protocol today is represented by Multipath TCP (MPTCP). Several implementations of it exist in ns-3 [3]. Naturally, one would want to use the newest and most advanced implementation by Coudron and Secci [3]; however, in spite of offering benefits like full compliance with MPTCP specifications and compatibility with the ns-3 TCP socket API, we have found the current release of this implementation to be working unreliably in our scenario (excessive DUPACK's and inadequate subflow management when used with data transmissions bigger than 1 MB). Given our time frame, we moved to an older implementation developed by the University of Sussex [4]. This older implementation, though being developed for ns-3.19, has proven to work reliably with big amounts of data to transmit, and also responding reasonably well to path asymmetry (both bandwidth and delay) and it turned out to be faster for us to adapt it to our needs than modifying the implementation by Coudron and Secci.

A plethora of MPEG-DASH models have been implemented by the ns-3 community; in our work we use the implementation of Kreuzberger et al. [5], the AMuST-DASH framework, because it is directly built on top of a largely adopted Linux DASH library developed by Bitmovin [1]. This ns-3 module has been developed for ns-3.24.

## 3 PUTTING MPTCP AND MPEG-DASH TOGETHER

In the following we describe the main manipulations that we did to make our two models work together.

### 3.1 Modifications for MPTCP

The selected implementation of MPTCP seems to be focused on making the connection initiator to be the transmitting entity, therefore some of the features of the socket do not work when the other party has to transmit data as well. In addition to that, the implementation does not directly allow

[1] https://github.com/bitmovin/libdash

one to send real data from applications. Here we discuss modifications which mitigate those limitations; unless otherwise noted, they only concern the file `src/internet/model/mp-tcp-socket-base.cc` (together with its header file) of the original distribution.

*3.1.1 Connection receiver's limitations.* Distinction between the connection initiator and connection receiver is represented by socket's methods which can only be called by the particular endpoint according to the logical flow of MPTCP. The methods specific to the connection receiver are underdeveloped, so the following modifications are required to allow it to transmit data.

The `ProcessSynRcvd` method sets up the MPTCP socket upon receiving an incoming session and respond back; however, the implementation seems to never change the default value (0 Bytes) of congestion window of the socket. We initialize the congestion window to a value of one (1) MSS as it is done for the connection initiator methods; note that it is, however, not an optimal value [2].

The `SendPendingData` checks whether the currently selected MPTCP subflow has already been established; otherwise, the next subflow is selected. However, in the latter case the connection receiver's socket fails to change the subflow and hence ceases to transmit. We have not been able to identify the cause of this, though calling `getSubflowToUse` method inside of the (mentioned above) conditional clause solves the issue without any side effects.

*3.1.2 Transmitting real data.* The `Add` and `Retrieve` methods implemented for the socket's TX and RX buffers (file `mp-tcp-typedefs.cc`) do not allow handling actual user data. In fact, they accept/return merely the amount of bytes as an argument/output correspondingly. As MPEG-DASH is based on packet contents, we have therefore added the `AddRealData` and `RetrieveRealData` methods, which effectively handle real data in the MPTCP socket.

*3.1.3 Work in progress.* The RTO functionality of the socket implementation does not seem to be finished on the connection receiver side: the corresponding method (`ReTxTimeout`) can only be executed when called by the initiator. Removing this limitation, nevertheless, allows the socket to run without visible issues, though it has to be tested more.

Upon receiving the unordered data the socket proceeds with storing it in the right order. One of the conditional clauses checks if the Data Sequence Number (DSN) of the received mapping is smaller than the last stored DSN. In case it is, it checks the same for corresponding Subflow Sequence Numbers (SSN) and exits the program if the last is false. To the best of our knowledge, in MPTCP the latter condition cannot be false if the former condition is true; nevertheless, we have observed this happen for unknown reasons. Unable to solve it, we have removed the last condition and have not observed any issues with data ordering since then.

### 3.2 Modifications for AMuSt-DASH

The selected implementation of MPTCP does not replace the stock TCP socket, but instead comes in complement to it. Therefore, all applications written willing to use MPTCP must be updated to use this specific multipath socket – including the AMuSt-DASH model. As mentioned before, the MPTCP we use makes it fairly easy to convert applications for multipath: though the API itself is rather different, the specifics of MPTCP protocol – like subflow establishment – are done entirely by the socket itself without any need to program them in the application. As a result, the application will follow exactly the same steps for establishing/receiving an MPTCP connection than for TCP but with minor differences in methods called, as explained below.

First, each method of the application handling a socket has to cast it to an MPTCP socket using a `DynamicCast` directive.

Second, the selected implementation of MPTCP does not have a `Send(Ptr<Packet>)` method. Instead, one needs to fill the TX buffer with data using `FillBuffer`, and then call the `SendBufferedData` method to initiate data transmission.

## 4 COMPILING THE MERGED DISTRIBUTION

The two models mentioned above have been developed for different ns-3 versions which are not fully compatible with each other. This leads to compilation errors once the files are merged together; here we discuss them and explain fixes.

Since the implementation of MPTCP is rather complicated and spans across multiple files, we have decided to use its distribution as a substrate and merge the AMuSt-DASH (which is an application) into it. This requires copying the AMuSt-DASH-specific files (i.e., models, headers, and helpers) from `applications` module into the distribution supporting MPTCP, without forgetting to update `src/applications/wscript` and `src/wscript` scripts to correctly build the modified modules.

The header file `src/internet/model/tcp-socket.h` has been changed between ns-3.19 and ns-3.24 as the TCP states declaration has been rethought; the version included in ns-3.19 (MPTCP) declares the states outside of the `TcpSocket` class, while the one in ns-3.24 (DASH) has them inside. We have adopted the newer version of the file, which hence required updating seldom references to the TCP states in the TCP/MPTCP socket of the substrate distribution such that they are accessed from the `TcpSocket` namespace.

Next, AMuSt-DASH requires custom string handling functions: `string_ends_width()`, `zlib_compress_string()`, and `zlib_decompress_string()`. These functions are implemented in file *src/core/model/string.cc* of the original AMuSt-DASH distribution and can safely be added in the module. It has to be noted, however, that two last functions depend on ZLib library [2], so its support has to be added in the `src/core/wscript` (exactly as it is done in the DASH distribution).

[2] https://zlib.net/

## 5 CONCLUSION

In spite of the several unclear points (that we have mentioned above), and the fact that the resulting distribution is based on ns-3.19, our tests until this moment suggest that the distribution is rather stable and usable for performing experiments involving MPEG-DASH and MPTCP.

Ultimately, we would like to create a stand-alone module holding the mentioned implementations of MPEG-DASH and MPTCP, that could be used with an ns-3 distribution of choice. Unfortunately, the implementation of MPTCP requires modifications to essential files such as `tcp-l4-protocol.cc`, which might differ between releases of ns-3. Therefore, creating a proper module of this kind will most likely require having MPTCP supported by the official releases of the simulator.

## REFERENCES


[1] M Christopher. 2015. MPEG-DASH vs. Apple HLS vs. Microsoft Smooth Streaming vs. Adobe HDS. (2015).

[2] Jerry Chu, Yuchung Cheng, Nandita Dukkipati, and Matt Mathis. 2013. Increasing TCP's initial window. (2013).

[3] Matthieu Coudron and Stefano Secci. 2017. An implementation of multipath TCP in ns3. *Computer Networks* 116 (2017), 1–11.

[4] Morteza Kheirkhah, Ian Wakeman, and George Parisis. 2015. Multipath-TCP in ns-3. *arXiv preprint arXiv:1510.07721* (2015).

[5] Christian Kreuzberger, Daniel Posch, and Hermann Hellwagner. 2016. AMuSt Framework - Adaptive Multimedia Streaming Simulation Framework for ns-3 and ndnSIM. (2016).

[6] Iraj Sodagar. 2011. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia* 18, 4 (2011), 62–67.