



**AGH**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI**

KATEDRA INFORMATYKI

Praca dyplomowa magisterska

*Strumieniowanie adaptacyjne danych multimedialnych z  
wykorzystaniem standardu DASH-MPEG*

Autor:

Kierunek studiów:

Opiekun pracy:

*Piotr Borowiec*

*Informatyka*

*dr inż. Łukasz Czekierda*

Kraków, 2014

*Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczanie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.*

*Serdecznie dziękuję ...*

## Spis treści

<b>1. Wprowadzenie</b>	7
<b>2. Zarys problemu</b>	9
2.1. Wstęp	9
2.2. Problem strumieniowania danych	9
2.3. Podsumowanie	10
<b>3. Przegląd istniejących rozwiązań</b>	11
3.1. Wstęp	11
3.2. Transmission Control Protocol	11
3.3. Datagram Congestion Control Protocol	12
3.4. Real-time Transport Protocol i RTP Control Protocol	14
3.5. Podsumowanie	15
<b>4. Standard DASH-MPEG</b>	17
4.1. Wprowadzenie	17
4.2. Zarys historyczny standardu	17
4.3. Systemy implementujące DASH	17
4.4. Hierarchiczna struktura modelu danych	18
4.5. Osie czasu	20
4.6. Model referencyjny klienta dla metryki DASH	20
4.7. Profile	21
4.8. Podsumowanie	21
<b>5. Opis projektu i implementacji</b>	23
5.1. Algorytm z artykułu	23
5.2. Własny algorytm	23
5.3. Implementacja	23
5.4. Problemy i rozwiązania	23
<b>6. Wdrożenie i testy</b>	25
<b>7. Podsumowanie</b>	27

<b>A. Konfiguracja przełącznicy.....</b>	<b>29</b>
<b>Bibliografia.....</b>	<b>31</b>

# **1. Wprowadzenie**

- Cel pracy - skąd się wziął pomysł, dlaczego praca jest tworzona
- Struktura - krótki opis dotyczący kolejnych rozdziałów
- Zakres merytoryczny - ramy dla projektu
- Wkład własny - krótki opis co się zrobiło i jak przebiegały prace.



## **2. Zarys problemu**

### **2.1. Wstęp**

Poniższy rozdział zawiera opisy problemów związanych ze strumieniowaniem danych w sieciach komputerowych. Aplikacje przeznaczone do celów strumieniowania można zbudować na podstawie wielu protokołów i standardów. Rozdział skupia się na wadach istniejących rozwiązań w kontekście strumieniowania danych multimedialnych oraz danych interaktywnych.

### **2.2. Problem strumieniowania danych**

Protokoły oparte o datagramy, takie jak User Datagram Protocol pozwalają na wysłanie strumienia danych jako serii pakietów. Takie rozwiązanie jest proste i wydajne, ale UDP nie daje żadnych gwarancji, że pakiety dotrą nienaruszone, w odpowiedniej kolejności i na czas do odbiorcy. Brak mechanizmów kontrolujących powyższe czynniki oznacza, że jakość strumienia może być słaba - dane przez niego przenoszone będą niepoprawne, spóźnione lub odbiorca nie otrzyma ich wcale. Część z tych problemów została rozwiązana w Datagram Congestion Control Protocol (zob. 3.3), który wprowadza kontrolę przeciążeń. Pozwala to na kompromis pomiędzy możliwością wykorzystania łącza (UDP wykorzystuje łącze w sposób bardziej agresywny niż TCP i DCCP), a adaptacją do zmieniającego się dostępnego pasma. UDP nie posiada mechanizmów adaptacji, TCP reaguje szybciej od DCCP na zmienność pasma. DCCP pozwala na kontrolowanie ilości utraconych pakietów w transmisji i jest TCP friendly, co oznacza, że może wykorzystać przepustowość nie większą niż dwukrotna przepustowość jaką wykorzystałoby TCP w tych samych warunkach. Protokoły datagramowe są dobrze przystosowane do transmisji interaktywnych w których utrata pojedynczych pakietów nie jest problematyczna, natomiast wymagane jest małe opóźnienie (np. gry komputerowe z trybem multiplayer).

Niezawodne protokoły (np. Transmission Control Protocol) gwarantują poprawność i kolejność danych. Zwykle mechanizmy zapewniające powyższe funkcjonalności są oparte na licznikach czasu i retransmisjach, co może powodować spore opóźnienia w transmisji. Strumienie oparte na takich protokołach są wrażliwe na zmieniającą się dostępną przepustowość sieci i ilość danych, które dostarczają do odbiorcy, może być zmienna w czasie. Dlatego aplikacje oparte na technologiach tego typu zwykle implementują bufor, które pozwalają na zmniejszenie wpływu zmienności strumienia na działanie programu.



Jeżeli transmisja powinna zostać dostarczona do wielu odbiorców to wykorzystanie unicast'ów jest niewydajne. Pakiety przenoszące informacje muszą być powielane i adresowane do każdego z odbiorców osobno. W celu poprawy wykorzystania łącza możliwe jest skorzystanie z multicast'ów. Dzięki temu transmisja może dotrzeć do wielu odbiorców bez zbędnego przesyłania tych samych informacji wielokrotnie przez to samo łącze. Znaczącą wadą tego rozwiązania jest fakt, że Internet obecnie nie jest przystosowany do przesyłania multicast'ów na szeroką skalę. Implementacja multicast'ów zajmuje zasoby na urządzeniach sieciowych i potrzebuje protokołów routingu i kontrolowania położenia odbiorców w sieciach (np. PIM - Protocol Independent Multicast oraz IGMP - Internet Group Management Protocol). Kolejny problem w transmisji grupowej stanowi brak możliwości wyboru danych do strumieniowania po stronie klienta. Klient nie może ponownie odtworzyć fragmentu meczu, który dopiero obejrzał lub przewinąć części strumieniowanego filmu do przodu. Możliwe jest jednak udostępnienie klientowi tych funkcjonalności przy użyciu serwerów cache'ujących i odtwarzaczy buforujących otrzymywane dane.

Innym podejściem do strumieniowania danych charakteryzują się protokoły typu P2P (Peer-to-peer). W rozwiązaniach opartych na tej technologii nie ma "wąskich gardeł" w postaci centralnych serwerów. Dane znajdujące się u jednego lub więcej użytkowników mogą być bezpośrednio przesłane do odbiorcy. Każdy odbiorca może strumieniować dane od wielu użytkowników, co pozwala na rozłożenie obciążenia sieci i stacji wysyłających. To podejście również ma swoje wady w szczególności dotyczące bezpieczeństwa i jakości danych które odbiorca otrzymuje.

## 2.3. Podsumowanie

Każda z technologii ma zalety z których aplikacja może skorzystać oraz wady, które aplikacja powinna korygować lub zmniejszać ich wpływ na jakość strumieniowania. Bardzo istotny jest wybór odpowiedniej technologii na podstawie wymagań jakie powinna spełniać transmisja.

## 3. Przegląd istniejących rozwiązań

### 3.1. Wstęp

Istnieje wiele podejść do zagadnienia jakim jest strumieniowanie danych. Najbardziej popularnym spośród opisywanych poniżej jest para protokołów: Real-time Transport Protocol i RTP Control Protocol [2]. W niniejszym rozdziale opisane zostaną również protokoły: Transmission Control Protocol [9], Datagram Congestion Control Protocol [4] oraz standard DASH-MPEG [10]. W każdym z powyższych przypadków zostaną zaprezentowane metody pozwalające na adaptację tych protokołów/standardów do warunków panujących w sieci.

### 3.2. Transmission Control Protocol

TCP jest protokołem o charakterze połączeniowym mającym sporo zastosowań. Należy do warstwy transportowej modelu ISO/OSI i jest wykorzystywany podczas wymiany poczty elektronicznej (SMTP, IMAP), przeglądania stron internetowych (HTTP, HTTPS), transferu plików (FTP) oraz pozwala na zdalny dostęp do zasobów (SSH) [12]. Oferuje dostarczanie informacji w sposób niezawodny; sprawdza dane pod względem poprawności (sumy kontrolne) oraz kolejności (numery sekwencyjne) [9].

Protokół TCP jest ukierunkowany na niezawodne i bezbłędne dostarczanie danych, co w pewnych przypadkach może powodować opóźnienia. Z tego powodu UDP (w połączeniu z RTP) jest uważane za lepszy wybór przy strumieniowaniu interaktywnych danych multimedialnych.

Mechanizmy kontroli przeciążeń i przepływu w TCP pozwalają strumieniom na szybkie dostosowanie się do zmieniających się warunków w sieci (zmiennej dostępnej przepustowości). Do regulacji wielkości strumieni wykorzystywane jest okno. Okno jest mechanizmem określającym jak wiele danych można wysłać bez otrzymania potwierdzeń odbioru. Wielkość okna TCP  $W$  jest obliczana w sposób następujący [12]:

$$W = \min(cwnd, awnd)$$

gdzie:

- $cwnd$  (*congestion window*) - wielkość okna obliczona na podstawie obciążenia sieci,
- $awnd$  (*advertised window*) - wielkość okna obliczona na podstawie obciążenia odbiorcy (przesyłana w komunikatach *window update*,

Kontrola przepływu zapobiega przeciążeniu odbiorcy i może być oparta na regulacji wielkości okna TCP lub na ustaleniu górnej dopuszczalnej przepustowości dla stacji wysyłającej dane. Stacja odbierająca może zmieniać wielkość okna stacji wysyłającej za pomocą wiadomości *window update* [12]. Jeżeli stacja odbiorcza nie nadaje z przetwarzaniem pakietów, może zmniejszyć wielkość okna ogłaszając w wiadomościach do wysyłającego. Zmniejszenie okna będzie równoznaczne ze spowolnieniem stacji wysyłającej - zmniejszona zostanie ilość pakietów, które można wysłać nie otrzymując ich potwierdzeń odbioru.

Kontrola przeciążeń zapobiega nadmiernemu zalaniu pakietami sieci znajdującej się pomiędzy stacjami końcowymi. TCP nie wykorzystuje jawnych metod wykrywania obciążenia sieci. Podstawną do założenia, że sieć może być przeciążona są ginące pakiety. Algorytm Slow Start wykorzystuje utracone pakiety jako wyznacznik obciążenia sieci. Jego zadaniem jest wykrycie dostępnej przepustowości bez narażania sieci na przeciążenie. W tym celu przesyła pewną ustaloną ilość segmentów i oczekuje na ich potwierdzenie. Z każdą transmisją zakończoną sukcesem ilość wysyłanych segmentów bez potwierdzania jest zwiększana wykładniczo aż do momentu w którym pakiety zaczną ginać. Drugi algorytm - Congestion Avoidance - pozwala na wykrycie dodatkowego pasma po fazie powolnego startu. Zachowuje się podobnie jak algorytm Slow Start, ale ilość segmentów jest zwiększana liniowo. [9]

### 3.3. Datagram Congestion Control Protocol

Protokół DCCP należy do warstwy transportowej i wspiera dwukierunkowe połączenia typu punkt-punkt. Cechą charakterystyczną tego protokołu jest niezawodne nawiązywanie i zrywanie połączenia oraz negocjacja parametrów połączenia. Sama transmisja oparta jest na przesyłaniu datagramów bez potwierdzeń. DCCP posiada kilka mechanizmów kontroli przeciążeń.

Każdy ze wspieranych przez DCCP mechanizmów kontroli przeciążeń posiada identyfikator CCID (Congestion Control Identifier). W trakcie negocjacji parametrów połączenia stacje końcowe wymieniają posortowaną listę akceptowalnych CCID (od najbardziej pożądanego do najmniej pożądanego), a następnie wybierają mechanizm, którym będą posługiwać się w trakcie transmisji [6]. RFC standaryzuje dwa mechanizmy kontroli przeciążeń:

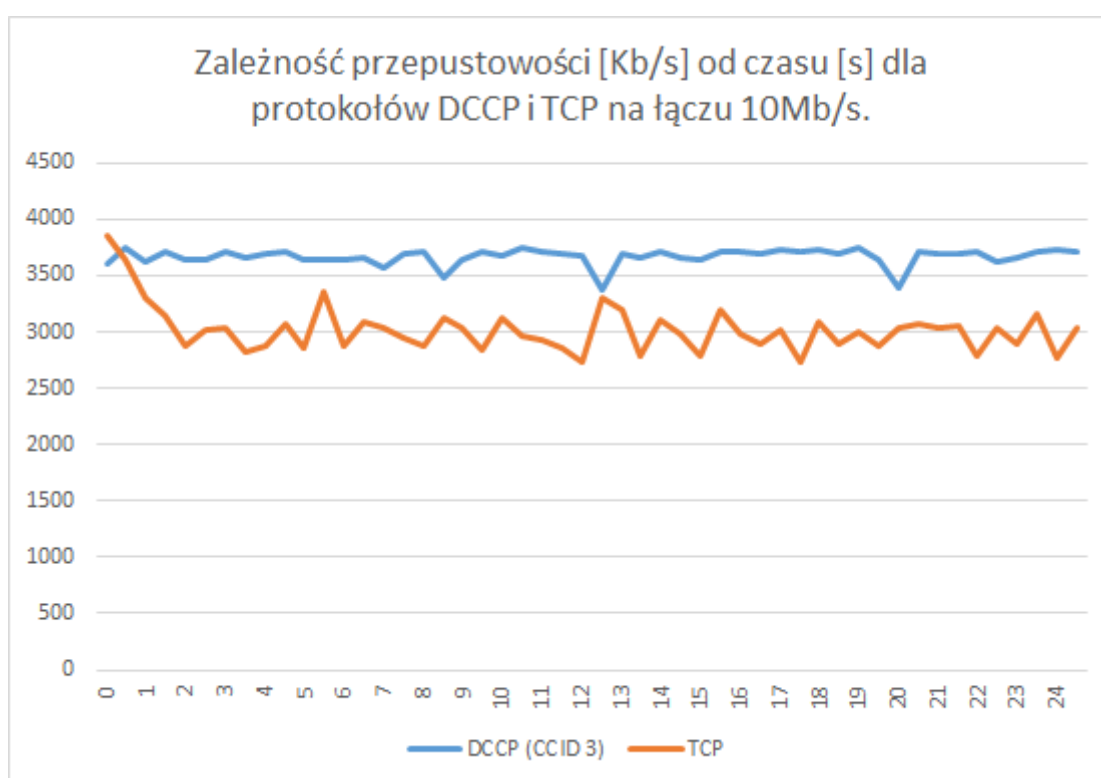
- CCID 2 - TCP-like Congestion Control
- CCID 3 - TCP-Friendly Rate Control

Pozostałe numery CCID (0-1 oraz 4-255) pozostają do tej pory niewykorzystane.

Mechanizm TCP-like Congestion Control korzysta z AIMD (Additive Increase/Multiplicative Decrease). Jeżeli TCP wykorzystuje AIMD to po otrzymaniu każdej poprawnej wiadomości ACK od odbiorcy, nadawca powiększa wielkość *cwnd* (wielkość okna) o  $\frac{1}{cwnd}$ . W razie utraty pakietów okno jest pomniejszane dwukrotnie [12]. Omawiany mechanizm zachowuje także liczniki czasu znane z TCP oraz algorytm Slow Start [11]. TCP-like Congestion Control powinno być wykorzystywane przez aplikacje

w których bardziej preferowane jest maksymalne wykorzystanie łącza od utrzymania stabilnego pasma (np. transfer plików) [5].

TCP-Friendly Rate Control jest mechanizmem uznawanym za niezachłanny przy dzieleniu łącza z innymi strumieniami o charakterze TCP. Niezachłanność tego mechanizmu oznacza, że wykorzystuje on przepustowość nie większą niż dwukrotna wartość przepustowości, która zostałaby wykorzystana przez TCP w tych samych warunkach. TCP-Friendly Rate Control charakteryzuje się także dużo mniejszą zmiennością wartości przepustowości w czasie w porównaniu do TCP, co sprzyja zastosowaniu DCCP z CCID 3 w strumieniowaniu danych multimedialnych. ‘Wygładzenie’ przepustowości transmisji jest okupione wolniejszą niż w TCP szybkością przystosowywania się strumienia do zmian w dostępnym paśmie [8].



Rysunek 3.1: Wykres zależności przepustowości od czasu dla protokołów TCP i DCCP.

Rysunek 3.1 przedstawia porównanie działania protokołów TCP i DCCP (CCID 3). W trakcie testu wykorzystano oprogramowanie D-ITG (Distributed Internet Traffic Generator) [16]. Test został przeprowadzony z wykorzystaniem dwóch komputerów i przełącznicy Cisco Catalyst 3550 połączonych w jedną sieć LAN i polegał na jednoczesnym uruchomieniu dwóch strumieni (TCP i DCCP) pomiędzy stacjami końcowymi. Próbkowanie przepustowości przeprowadzono co 0,5 s. Na przełącznicy zaimplementowano politykę pozwalającą na ustalenie przepustowości pomiędzy stacjami na 10Mb/s (zob. Dodatek A).

Tabela z rysunku 3.2 przedstawia porównanie średniej przepustowości i odchylenia standardowego dla protokołów TCP i DCCP. Przeprowadzony test potwierdza zachowanie protokołu DCCP (CCID 3)

Protokół	Średnia przepustowość	Odchylenie Standardowe
TCP	3020 Kb/s	209
DCCP (CCID 3)	3703 Kb/s	74

Rysunek 3.2: Tabela porównawcza protokołów TCP i DCCP.

opisane w RFC 5348 [8].

### 3.4. Real-time Transport Protocol i RTP Control Protocol

RTP (Real-time Transport Protocol) pozwala na transport danych w systemach i aplikacjach czasu rzeczywistego wspierających interaktywne transmisje audio i video. Zwykle wykorzystuje UDP jako protokół transportowy, ale może działać nad innymi protokołami warstwy transportowej modelu ISO/OSI (DCCP, TCP - [2, 6]). Jeżeli sieć w której działa RTP wspiera multicasting to RTP pozwala na wysłanie danych do wielu odbiorców jednocześnie. RTP nie dostarcza mechanizmów QoS (Quality of Service), ani nie gwarantuje dostarczenia wysłanych danych na czas do odbiorcy.

Protokół RTCP (Real-time Control Protocol) stanowi protokół kontrolny dla RTP. Bazuje na okresowej transmisji pakietów kontrolnych do wszystkich uczestników sesji. Przekazuje informacje od odbiorców do nadawców na temat jakości transmisji. Pakiety RTCP zawierają identyfikator źródła (Canonical Name) oraz pozwalają na ustalenie liczby uczestników sesji, co pozwala na obliczenie z jaką częstotliwością należy wysyłać pakiety kontrolne.

Nazwa	Typ	Zalecane pasmo
GSM	audio	13 Kb/s
G728	audio	16 Kb/s
G729	audio	8 Kb/s

Rysunek 3.3: Przykładowe typy pakietów RTP.

RTP przenosi dane, które zwykle wymagają ustalonej przepustowości. Dzięki temu prawdopodobieństwo, że strumień RTP będzie systematycznie zajmował coraz większe pasmo jest niewielkie. Z drugiej strony, strumień nie może też zostać ograniczony bez uszczerbku na jakości transmisji danych, szczególnie jeżeli transmisja ma charakter interaktywny. Pasma potrzebne do sprawnej transmisji zależy od rodzaju przesyłanych danych. Rodzaj przesyłanych danych można identyfikować na podstawie pola Payload Type (zob. rysunek 3.3) w nagłówku pakietu RTP. Z każdym typem związany jest profil RTP opisujący parametry transmisji (w tym wymaganą przepustowość) [3]. Protokół RTP nie posiada odpowiednich mechanizmów kontroli przeciążeń, dlatego powinien korzystać z mechanizmów dostępnych w protokołach niższych warstw (np. DCCP CCID 3).

### 3.5. Podsumowanie

W powyższym rozdziale opisano kilka podejść do problemu strumieniowania danych multimedialnych. Opisane zostały protokoły TCP, DCCP, RTP/RTCP oraz standard DASH-MPEG. Szczególną uwagę poświęcono mechanizmom adaptacji do zmiennej przepustowości sieci i cechom jakie powinny posiadać protokoły przeznaczone do strumieniowania danych. Do cech tych należą:

- kontrola przeciążeń i przepływu (TCP)
- jednostajne wykorzystanie łącza (DCCP CCID3)
- kanał zwrotny przynoszący informacje na temat sesji (RTCP)
- możliwość dostosowania parametrów transmisji do warunków w sieci (DASH, TCP)



## **4. Standard DASH-MPEG**

### **4.1. Wprowadzenie**

Poniższy rozdział opisuje standard Dynamic Adaptive Streaming over HTTP (DASH). Po zarysie historycznym i odniesieniu do pokrewnych technologii zaprezentowany zostaje podrozdział 4.3 zawierający zwięzły opis działania systemów wspierających strumieniowanie danych multimedialnych z wykorzystaniem DASH. Kolejne podrozdziały skupiają się na wybranych aspektach i funkcjonalnościach tego standardu takich jak hierarchiczna struktura modelu danych multimedialnych, synchronizacja strumieni oraz profile.

### **4.2. Zarys historyczny**

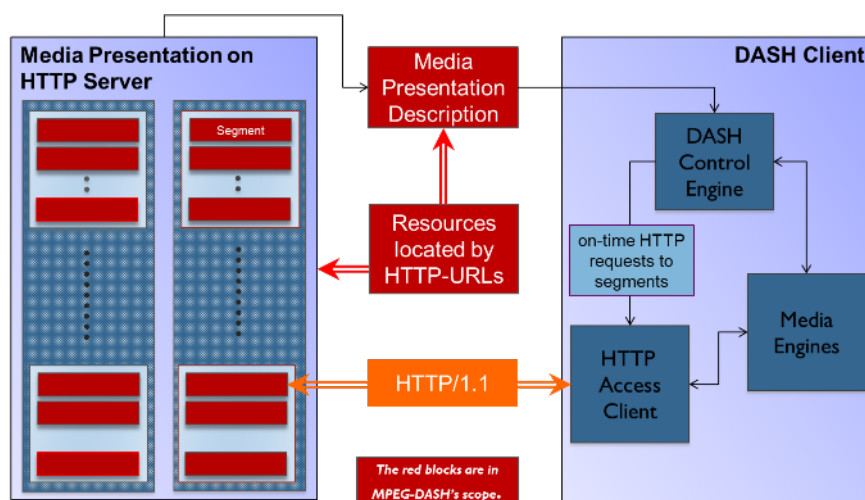
Dynamic Adaptive Streaming over HTTP jest adaptacyjną techniką strumieniowania danych w sieciach komputerowych. Prace nad standardem prowadzone przez grupę MPEG rozpoczęły się w 2010 roku. W 2011 standard DASH pojawił się jako draft, a międzynarodowym standardem stał się w kwietniu 2012 dzięki publikacji ISO/IEC 23009-1:2012 [10]. W lipcu 2013 wprowadzono do standardu poprawki.

DASH jest technologią spokrewnioną z Microsoft Smooth Streaming [13], Adobe Systems HTTP Dynamic Streaming [14] oraz Apple Inc. HTTP Live Streaming [15].

### **4.3. Systemy implementujące DASH**

W podejściu stosowanym w standardzie DASH logika strumieniowania zostaje w całości przeniesiona na aplikację klienta. Rolę serwera DASH może pełnić zwykły serwer HTTP. Plik multimedialny, który będzie podlegał operacji strumieniowania, należy odpowiednio przygotować. Plik powinien być dostępny w kilku wersjach różniących się jakością (a co za tym idzie również wielkością). Każda kopia powinna zostać podzielona na segmenty o równej długości. Na rysunku 4.1 kopie pliku multimedialnego składają się z segmentów (czerwone prostokąty) umieszczonych na serwerze HTTP (lewa strona rysunku). Od długości segmentów zależeć będzie szybkość przystosowywania się aplikacji DASH do zmieniającej się przepustowości sieci. Tak przygotowane pliki należy opisać za pomocą XML w postaci dokumentu MPD (Media Presentation Description). Plik MPD zawiera adresy URL poszczególnych segmentów, informacje na temat kodowania danych, rozdzielczości oraz wymaganych przepustowości.





Rysunek 4.1: Standard MPEG-DASH i przykładowa architektura klienta DASH, źródło: [17]

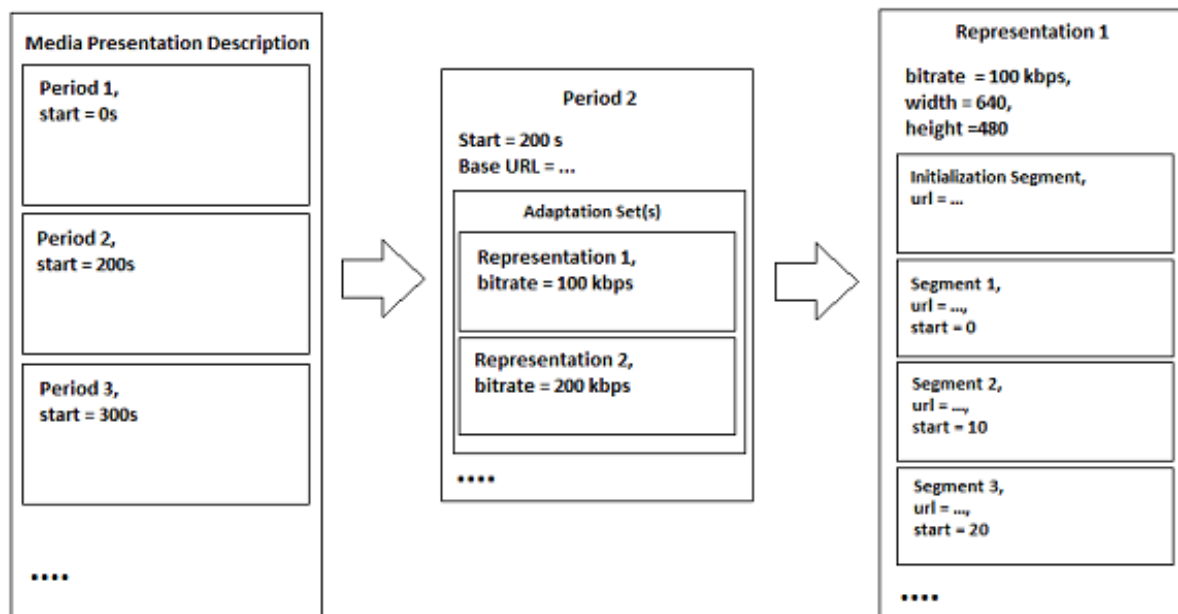
Następnie MPD musi zostać dostarczony klientowi, który parsując jego zawartość otrzyma informacje na temat danych multimedialnych, które ma strumieniować. W trakcie strumieniowania, aplikacja klienta wybiera w sposób dynamiczny wersję danych i pobiera kolejny segment z pomocą protokołu HTTP. Standard DASH działa niezależnie od sposobu kodowania danych i jest łatwo implementowalny w Internecie. Dzięki wykorzystaniu protokołu HTTP może współpracować z istniejącą infrastrukturą w postaci filtrów, zapór, urządzeń NAT i cache [17].

## 4.4. Hierarchiczna struktura modelu danych

Struktura modelu danych multimedialnych jest opisywana plikiem Media Presentation Description za pomocą języka XML i ma charakter hierarchiczny (zob. rysunek 4.2). Lewy prostokąt symbolizuje plik MPD, który zawiera kilka elementów typu *Period* (środkowy prostokąt na rysunku 4.2) oraz może zawierać informacje na temat profilu (zob. 4.7). Elementy *Period* reprezentują ciągłą część danych multimedialnych, które charakteryzują się takim samym zestawem parametrów. Do zestawu tych parametrów mogą należeć:

- zbiór dostępnych wersji plików multimedialnych (zbiór różnych bitrate),
- wersje językowe (dla dźwięku),
- wersje językowe napisów,
- sposób kodowania danych,
- itd...

Parametry te pozostają niezmiennie w pojedynczym elemencie *Period*, ale mogą występować różnice pomiędzy różnymi elementami. Wszystkie elementy *Period* posiadają znacznik czasowy pozwalający na



Rysunek 4.2: Struktura modelu danych, na podstawie [10]

ustalenie ich kolejności, a dane znajdujące się w kolejnych elementach tworzą ciągły strumień danych multimedialnych.

Dane znajdujące się w pojedynczym elemencie *Period* mogą zostać rozdzielone pomiędzy kilka elementów o nazwie *Adaptation Set*. *Adaptation Set* reprezentuje jedną lub kilka wersji komponentów danych multimedialnych. Przykładowo, możliwe jest odseparowanie danych video i audio poprzez przechowywanie ich w osobnych elementach *Adaptation Set*. Dane tekstowe (napisy) również mogą zostać wydzielone w podobny sposób. Rezygnacja z multipleksacji danych pozwala na wprowadzenie kilku wersji danych (audio lub napisy dostępne w różnych wersjach językowych).

Każdy *Adaptation Set* zawiera jeden lub kilka elementów typu *Representation* (prawy prostokąt na rysunku 4.2). Typowo, dane multimedialne w poszczególnych elementach *Representation* różnią się jakością, co w rezultacie wpływa na ich wielkość. W trakcie strumieniowania aplikacja klienta może wybierać pomiędzy różnymi wersjami danych na podstawie warunków panujących w sieci komputerowej (dostępna przepustowość, RTT<sup>1</sup>) lub innych czynników.

Dane multimedialne w ramach *Representation* mogą zostać podzielone na wiele części typu *Segments*. Są to najmniejsze części danych jakie klient może pobrać za pomocą metody HTTP GET. Każdy *Segment* ma określony czas trwania odpowiadający czasowi odtwarzania danych w nim zawartych oraz znacznik czasowy pozwalający na ustalenie ich właściwej kolejności. Elementy *Segment* należące do tego samego *Representation* zwykle mają porównywalny czas trwania.

<sup>1</sup>Round Trip Time - minimalny czas potrzebny na komunikację od nadawcy do odbiorcy i z powrotem.

## 4.5. Oś czasu

Standard DASH definiuje dwie osie czasu. Pierwsza z nich pozwala na synchronizację odtwarzania komponentów danych multimedialnych (audio/video/text). Metadane wymagane do synchronizacji można uzyskać z pliku MPD. Elementy *Period* oraz *Segment* posiadają atrybut “start” (zob. rysunek 4.2) pozwalający na określenie czasu w którym należy rozpocząć odtwarzanie zawieranych przez nie danych. W celu obliczenia bezwzględnego czasu w którym należy odtworzyć dany *Segment* należy obliczyć sumę składającą się z wartości atrybutu “start” elementu *Period* do którego dany *Segment* należy oraz wartości atrybutu “start” dla danego elementu typu *Segment*.

Druga z osi czasu jest wykorzystywana do wskazywania aplikacji klienta okien czasowych w jakich dostępne są dane multimedialne. Okna te nazywane są “Segment Availability Times”. Aplikacja klienta powinna sprawdzić dostępność danych (reprezentowanych przez *Segment* posiadający własny URL) przed próbą ich pobrania. W przypadku modeli “On Demand” takich jak VoD<sup>2</sup> plik MPD nie ulega zmianie i okna czasowe dla wszystkich danych multimedialnych są jednakowe. Takie podejście pozwala klientowi na odtwarzanie danych znajdujących się w dowolnym miejscu na osi czasowej (możliwość wprowadzenia powtórek, przewijania transmisji do przodu lub do tyłu). Dla usług “na żywo”, gdzie plik MPD jest aktualizowany w czasie działania aplikacji klienckich, okna czasowe charakteryzują się ograniczonym czasem życia związanym z położeniem danych multimedialnych na osi czasu. Użytkownik ma wtedy możliwość pobrania jedynie najnowszych danych (możliwość pobrania starszych danych jest limitowana wielkością okna czasowego - “Segment Availability Time”).

## 4.6. Model referencyjny klienta dla metryki DASH

System zgodny z modelem referencyjnym dla metryki DASH posiada trzy punkty obserwacyjne (*Observation Points*) w których dokonywane są pomiary dotyczące jakości transmisji danych multimedialnych i pracy aplikacji (zob. rysunek 4.3).



Rysunek 4.3: Model referencyjny klienta dla metryki DASH, źródło: [10]

<sup>2</sup>Video On Demand - usługa pozwalająca na odtwarzanie danych multimedialnych w wybranym przez użytkownika czasie, późniejszym niż czas emisji.

Pierwszy punkt obserwacyjny znajduje się na styku połączenia sieciowego z komponentem aplikacji klienta odpowiedzialnym za zgłaszanie zapotrzebowania na dane i ich pobierania. W trakcie pracy tego komponentu monitorowane są:

- połączenia TCP - docelowy adres IP, czas rozpoczęcia, trwania i zakończenia połączenia,
- sekwencja przesłanych wiadomości HTTP - czas transmisji, zawartość, połączenie TCP związane z transmisją,
- odpowiedzi HTTP - czas otrzymania, zawartość nagłówka.

Drugi z punktów obserwacyjnych znajduje się pomiędzy komponentem odpowiedzialnym za pobieranie danych, a komponentem przetwarzającym otrzymane dane. Przetwarzanie danych może polegać na ich demultipleksowaniu (audio/video) lub dekodowaniu. W tym punkcie obserwacyjnym zbierane są informacje na temat kodowanych danych takie jak:

- typ danych - video, audio, tekst, itd...,
- czas dostarczenia danych,
- czas dekodowania danych,
- poziom wypełnienia bufora dla pobieranych danych,
- numer identyfikacyjny elementu *Representation* z którego dane pochodzą.

Ostatni z punktów obserwacyjnych znajduje się pomiędzy komponentem przetwarzającym dane oraz komponentem, który prezentuje przetworzone dane użytkownikowi.

- typ danych,
- czas prezentowania danych wynikający z metadanych pliku MPD,
- czas rzeczywisty prezentowania danych (timestamp),
- numer identyfikacyjny elementu *Representation* z którego dane pochodzą.

Tabele przedstawiające dokładną specyfikację metryk można znaleźć w specyfikacji standardu DASH-MPEG ([10]) w dodatku D.

## 4.7. Profile

## 4.8. Podsumowanie

niedomagania i aspekty nad którymi warto się skupić.



## **5. Opis projektu i implementacji**

- Krótka prezentacja algorytmu zaimplementowanego z artykułu
- Prezentacja własnego algorytmu lub poprawek do algorytmu z punktu poprzedniego
- Opis implementacji programu klienta (diagram klas, sekwencji...)
- Przedstawienie ewentualnych problemów przy implementacji i zastosowanych rozwiązań

### **5.1. Algorytm z artykułu**

### **5.2. Własny algorytm**

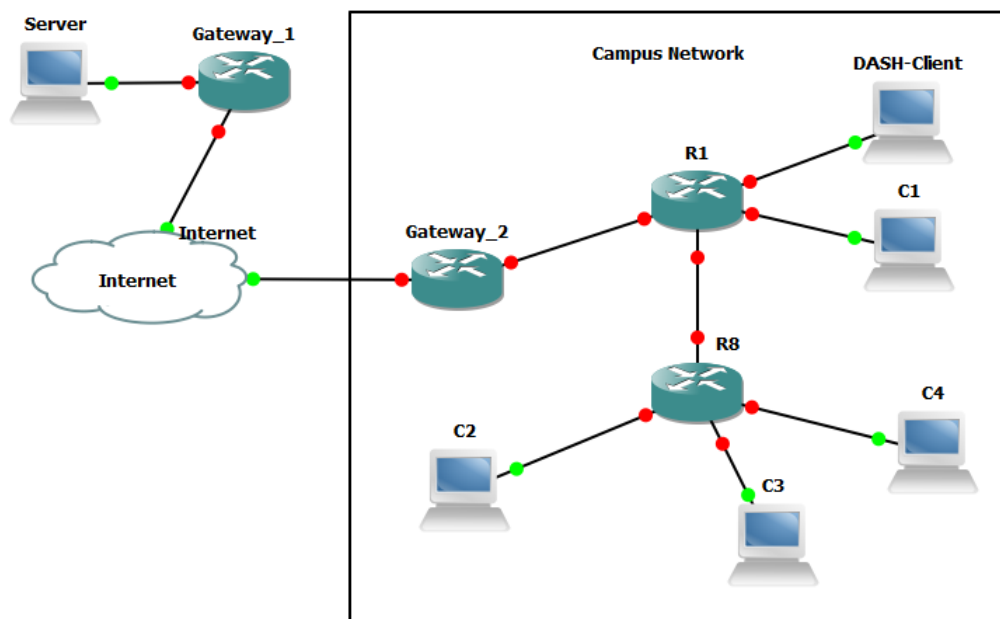
### **5.3. Implementacja**

### **5.4. Problemy i rozwiązania**



## 6. Wdrożenie i testy

- Opis instalacji i uruchamiania programu klienta
- Opis przeprowadzonych testów działania/testów porównawczych
- Pokazanie w jaki sposób skonfigurować sieć lokalną (multicasty) w celu zmniejszenia jej obciążenia przy transmisjach jeden-do-wielu. Tylko klient DASH łączy się do serwera. Pozostałe komputery w sieci kampusowej odbierają transmisję od klienta DASH.







## **7. Podsumowanie**

- Podsumowanie pracy - co się udało, co się nie udało
- Podkreślenie wkładu własnego
- Dalsze możliwości rozwijania pracy



## A. Konfiguracja przełącznicy

Poniżej znajduje się konfiguracja przełącznicy wykorzystywanej do testów. Pominięta została konfiguracja interfejsów FastEthernet 3-24 oraz interfejsów GigabitEthernet.

```
!  
version 12.2  
no service pad  
service timestamps debug uptime  
service timestamps log uptime  
no service password-encryption  
!  
hostname Switch  
!  
!  
no aaa new-model  
ip subnet-zero  
!  
mls qos aggregate-policer aggpolicer 10000000 8000 exceed-action drop  
mls qos  
!  
no file verify auto  
spanning-tree mode pvst  
spanning-tree extend system-id  
!  
!  
!  
vlan internal allocation policy ascending  
!  
class-map match-all tcpmap  
    match access-group 145  
!
```

```
!  
policy-map test  
  class tcpmap  
    police aggregate aggpolicer  
!  
!  
!  
interface FastEthernet0/1  
  switchport mode dynamic desirable  
  service-policy input test  
!  
interface FastEthernet0/2  
  switchport mode dynamic desirable  
  service-policy input test  
!  
interface Vlan1  
  no ip address  
  shutdown  
!  
ip classless  
ip http server  
!  
!  
!  
!  
access-list 145 permit tcp any any  
!  
control-plane  
!  
!  
line con 0  
line vty 5 15  
!  
!  
end
```

## Bibliografia

- [1] Guibin Tian, Yong Liu *Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming*. Polytechnic Institute of New York University
- [2] Request For Comment 3550: RTP: A Transport Protocol for Real-Time Applications
- [3] Request For Comment 3551: RTP: RTP Profile for Audio and Video Conferences with Minimal Control
- [4] Request For Comment 4340: Datagram Congestion Control Protocol
- [5] Request For Comment 4341: Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control
- [6] Request For Comment 5762: RTP and the Datagram Congestion Control Protocol (DCCP)
- [7] Request For Comment 5681: TCP Congestion Control
- [8] Request For Comment 5348: TCP Friendly Rate Control (TFRC): Protocol Specification
- [9] Request For Comment 793: Transmission Control Protocol
- [10] Information technology — Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats
- [11] Request For Comment 2581: TCP Congestion Control
- [12] TCP/IP Illustrated, Volume 1, Second Edition Kevin R. Fall, W. Richard Stevens
- [13] <http://www.microsoft.com/silverlight/smoothstreaming>
- [14] <http://www.adobe.com/pl/products/hds-dynamic-streaming.html>
- [15] <https://developer.apple.com/streaming/>
- [16] <http://traffic.comics.unina.it/software/ITG/>
- [17] <http://dashif.org/mpeg-dash/>