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: *Piotr Borowiec*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Łukasz Czekierda*

Kraków, 2014

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję ...

Spis treści

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. Problematyka strumieniowania danych

2.1. Wstęp

Poniższy rozdział skupia się na problematyce strumieniowania danych multimedialnych w sieciach komputerowych. Omawia niepożądane zjawiska występujące podczas transmisji danych oraz zmienność warunków panujących w sieciach komputerowych. Zwraca uwagę na zależność jakości i szybkości transmisji danych od konfiguracji sieci i urządzeń sieciowych oraz na zalety wykorzystania transmisji grupowych w przypadku komunikacji jeden do wielu.

2.2. Zajwiska występujące podczas transmisji danych

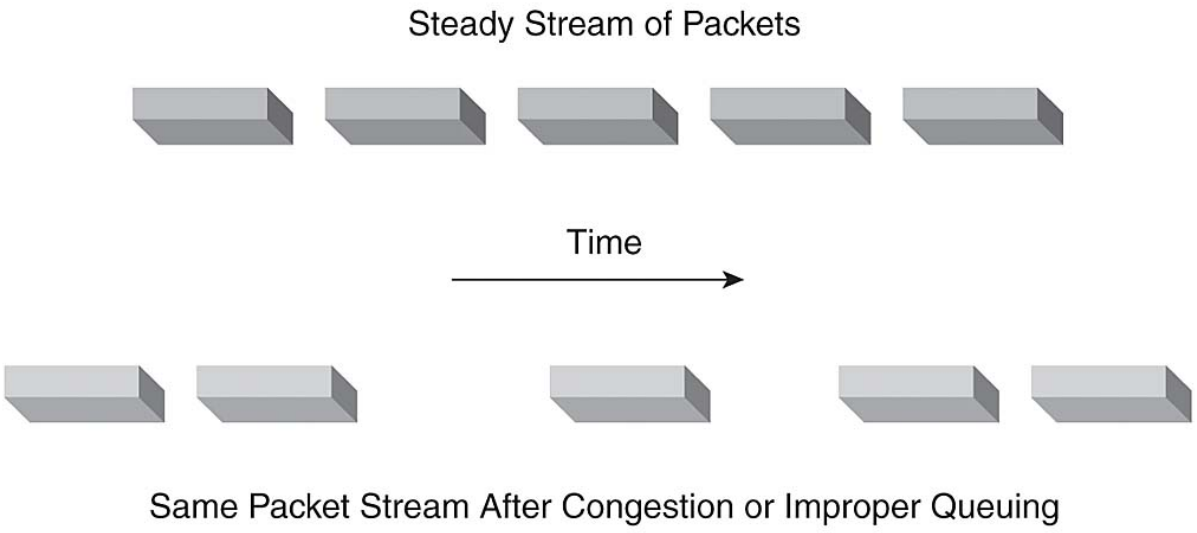
Aplikacje zajmujące się strumieniowaniem danych narażone są na wiele zjawisk związanych z wymianą informacji w sieciach komputerowych. Do zjawisk, które mogą spowodować największe zakłócenia podczas transmisji danych należą: opóźnienia, jitter oraz utrata danych przesyłanych przez sieć.

Opóźnienie w trakcie przesyłania danych wynika przede wszystkim z czasu transmisji danych przez media fizyczne oraz z czasu przetwarzania danych przez urządzenia sieciowe znajdujące się na trasie od nadawcy do odbiorcy. Nowoczesne urządzenia sieciowe przełączają ruch z szybkością bliską *wire-speed*[1], wykorzystując w tym celu specjalne układy scalone *ASIC*[2]. Kolejnym powodem występowania opóźnień jest duży ruch w sieci. Urządzenia sieciowe przetwarzające dane zwykle wykorzystują bufory znajdujące się na interfejsach w celu przechowania danych przychodzących zanim zostaną one skierowane na interfejsy wyjściowe. Jeżeli ruch przechodzący przez urządzenie jest na tyle duży, że urządzenie pracuje z maksymalną szybkością, to bufory zaczną zapełniać się danymi. Taka sytuacja prowadzi do opóźnień w transmisji - dane muszą czekać określony czas zanim zostaną przetworzone przez urządzenie. Parametr opisujący opóźnienie występujące na drodze od nadawcy do odbiorcy i z powrotem to tzw. *Round Trip Time*. Do pomiaru opóźnienia może posłużyć program Ping dostępny na większości systemów operacyjnych.

Drugie zjawisko, jitter, zobrazowane zostało na rysunku 2.1. Jitter występuje, gdy dane w sieciach pakietowych wysłane w równych odstępach czasu (górna część rysunku) docierają do odbiorcy w różnych odstępach czasu (dolna część rysunku). Wahania opóźnienia pakietów należących do tej samej transmisji

[1] Maksymalna szybkość wykorzystywanego standardu komunikacyjnego.

[2] Application Specific Integrated Circuit - układy scalone przeznaczone do wykonywania z góry ustalonego zadania.

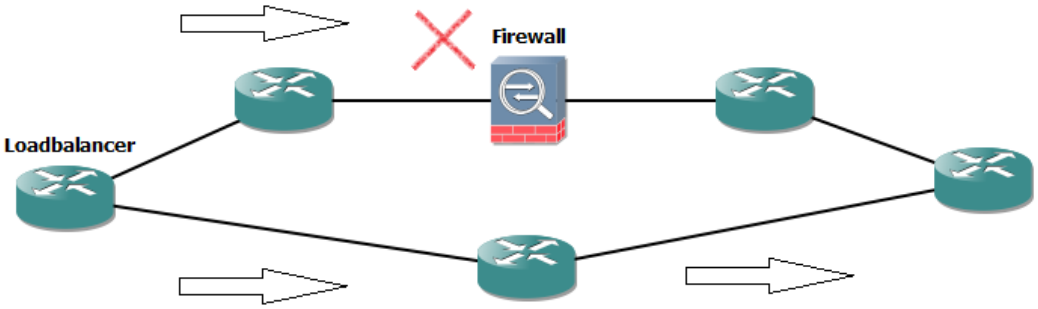


Rysunek 2.1: Wizualizacja zjawiska jitter, źródło: [19]

mogą być spowodowane obciążeniem sieci lub wyborem różnych ścieżek dla tych danych przez urządzenia sieciowe. Na jitter podatne są przede wszystkim aplikacje pozwalające na komunikację głosową. W sieciach w których działają takie aplikacje zwykle wdrożone jest Quality of Service. Dzięki temu dane głosowe otrzymują wysoki priorytet i są przekazywane przez urządzenia sieciowe w pierwszej kolejności. Zmniejsza to zarówno opóźnienie jak i jitter dla transmisji głosowych. W celu dalszego zmniejszenia wpływu jitter na działanie transmisji, implementowane są bufory w aplikacji klienta. Takie rozwiązanie pozwala na powtórne "wygładzenie" strumienia pakietów, ale zwiększa całkowite opóźnienie transmisji.

Utrata danych w sieciach komputerowych może nastąpić z wielu powodów. W warstwie fizycznej możliwa jest zmiana przesyłanych sygnałów wskutek interferencji zewnętrznych lub niedoskonałości samego medium. W przypadku warstwy łącza danych i najszerzej stosowanego standaru jakim jest Ethernet utrata ramki może być następstwem kolizji, niezgodności sumy kontrolnej CRC[3] z wartością policzoną na urządzeniu sieciowym lub pętli. Pętle w warstwie drugiej mogą doprowadzić do *broadcast storms* i w rezultacie sprawić, że sieć będzie niedostępna. Z kolei w warstwie sieciowej utrata pakietów może nastąpić na skutek braku odpowiednich tras na urządzeniach pośredniczących w wymianie danych lub zaimplementowanych środków bezpieczeństwa (zapory, filtry, itd...). Innym możliwym powodem utraty danych są konfiguracje urządzeń sieciowych. Niepoprawne lub zbyt restrykcyjne polityki zaimplementowane na urządzeniach sieciowych mogą spowodować całkowitą lub częściową utratę danych i stanowić wyzwanie przy próbie poprawy działania tych urządzeń. Urządzenia ograniczające komunikacje mogą znajdować się pod kontrolą innego podmiotu administracyjnego, co dodatkowo komplikuje proces poprawy działania sieci. Rysunek 2.2 przedstawia przykład w którym konfiguracja urządzeń powoduje częściową utratę danych. Loadbalancer równoważy obciążenie pomiędzy dwoma trasami. Na górnej trasie

[3]Cyclic Redundancy Check - sposób tworzenia skrótu z danych obliczany w celu możliwości późniejszego ustalenia ich spójności.



Rysunek 2.2: Błędna konfiguracja urządzeń sieciowych powoduje częściową utratę danych.

znajduje się zapora, która filtruje ruch i rezultacie tylko część pakietów (przesłana trasą dolną) dociera do odbiorcy.

Również w warstwie sieciowej mogą występować pętle i protokoły działające na tym poziomie powinny być wyposażone w mechanizmy pozwalające na usuwanie danych krążących w takich pętlach. W przypadku protokołu IP zdefiniowane jest pole TTL (Time to live) określające maksymalną liczbę urządzeń sieciowych, które mogą przetworzyć pakiet zanim dotrze on do odbiorcy. Przy każdym przetworzeniu pakietu przez urządzenie wartość pola jest zmniejszana o 1 (maksymalna wartość tego pola to 255) i jeżeli spanie do 0 to pakiet jest usuwany z sieci.

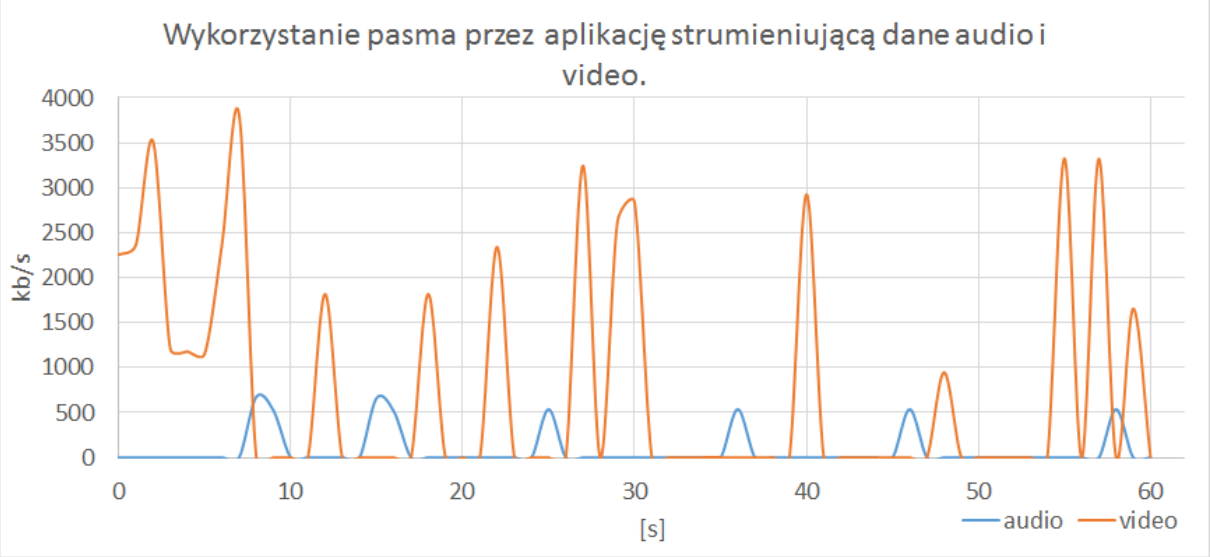
2.3. Zmienność warunków panujących w sieciach komputerowych

Szybkość transmisji danych zależy bezpośrednio od przepustowości łączy znajdujących się pomiędzy nadawcą i odbiorcą. Do sieci zwykle podłączonych jest wielu użytkowników wymieniających pomiędzy sobą dane, często w sposób nieregularny. Część łączy pomiędzy użytkownikami musi być współdzielona, co oznacza, że całkowita dostępna przepustowość na tych łączach może być zmienna w czasie. Zakres i częstotliwość z jaką zmienia się dostępne pasmo zależy od charakteru transmisji danych prowadzonych przez użytkowników, a utraty danych i ich retransmisje prowadzą do jeszcze większych nieregularności przepustowości sieci.

Rysunek 2.3 obrazuje różnice w transmisjach danych audio oraz video. Do testu wykorzystano dwa laptopy z kartami sieciowymi, przełącznicę Cisco Catalyst 3500 oraz oprogramowanie VLC i Wireshark. Laptopy i przełącznica zostały połączone w jedną sieć LAN. Jeden z laptopów posłużył jako serwer HTTP na którym znajdowały się pliki multimedialne. Na drugim laptopie zainstalowano programy VLC[4] i Wireshark[5]. Wykres wygenerowano na podstawie informacji zgromadzonych przez Wireshark

[4]VLC to rozbudowany program do odtwarzania danych multimedialnych. Posiada funkcjonalność strumieniowania plików audio i video.

[5]Wireshark to tzw. *Sniffer*, czyli program pozwalający na przechwycenie i rozkodowanie pakietów pojawiających się na interfejsie urządzenia.



Rysunek 2.3: Porównanie transmisji audio i video.

(funkcja “IO Graphs”) w ciągu 60 sekund ciągłego strumieniowania danych za pomocą VLC.

Transmisje audio zajmują znaczne mniejszą część pasma i charakteryzują się stałym jego wykorzystaniem. Zajętość pasma dla transmisji video jest większa, zmnienna w czasie i zależy od sposobu kodowania przesyłanych danych. Plik video zwykle kodowany jest wewnątrz-ramkowo i między-ramkowo. Kodowanie wewnątrz-ramkowe pozwala na zmniejszenie wielkości pojedynczej ramki. Stopień kompresji zależy zarówno od cech ramki jak i od algorytmu kodującego. Kodowanie między-ramkowe polega zakodowaniu ramki na podstawie ramek sąsiadujących i pozwala na największy stopień kompresji, gdy obraz video pozostaje statyczny (kolejne ramki są prawie identyczne lub bardzo podobne).

Poza charakterystyką przesyłanych danych na transmisję mogą wpływać cechy protokołów z których ta transmisja korzysta. Protokoły wykorzytywane przez aplikacje do komunikacji w sieci należą do warstwy transportowej i mogą zostać sklasyfikowane według kilku kategorii. Pierwszy rodzaj podziału dotyczy zachowania się protokołów w przypadku utraty danych. Protokoły niezawodne posiadają mechanizmy retransmisji utraconych danych i implementują liczniki czasu, po których upływie dane zostają uznane za zaginione. Dzięki temu dane są przekazywane w całości do odbiorcy, ale takie podejście powoduje dodatkowe narzuty na komunikację. Protokoły zawodne nie obsługują mechanizmów retransmisji i niedają żadnych gwarancji, że dane zostaną dostarczone do odbiorcy. Kolejny podział protokołów moża przeprowadzić ze względu na sposób inicjalizacji komunikacji i jej zakończenia. Protokoły połączeniowe przed wysłaniem danych nawiązują połączenie w celu ustalenia, czy odbiorca jest gotowy do przyjęcia przesyłanych danych. W trakcie inicjalizacji połączenia możliwa jest wymiana wartości parametrów w celu usprawnienia dalszej komunikacji. Protokoły bezpołączeniowe nie sprawdzają dostępności odbiorcy. Ich rola polega jedynie na wysłaniu danych do ustalonego adresata, bez względu na to czy jest w stanie te dane odebrać, czy też nie. Brak możliwości sprawdzenia dostępności adresata może prowadzić do przesyłania danych, które nie mogą zostać odebrane. Kolejne dwie kategorie dotyczą zachowania się

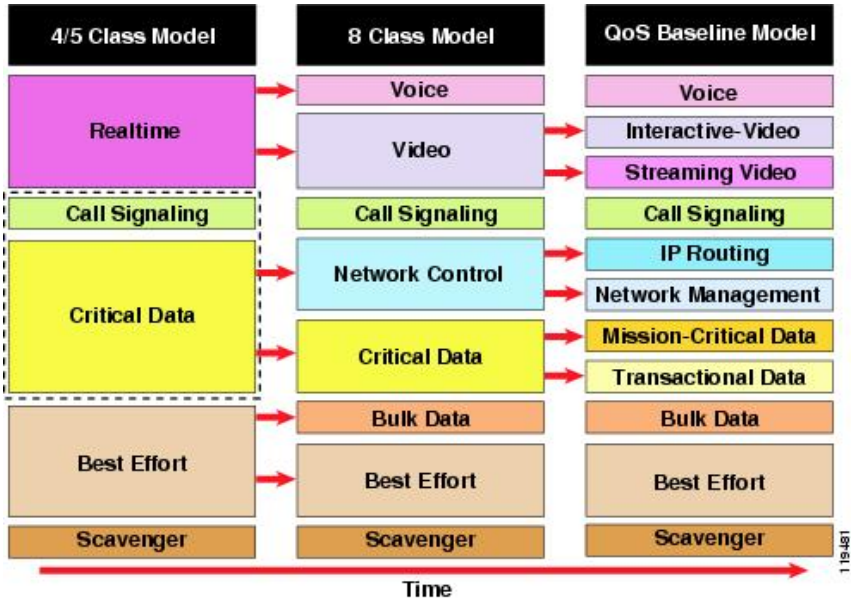
protokołów w przypadku przeciążenia sieci lub odbiorcy. Protokoły reagujące na zwiększony ruch w sieci implementują mechanizmy kontroli przeciążeń. Jeżeli protokół wykryje, że sieć jest przeciążona to ograniczy szybkość z jaką przesyła dane. Protokoły nie implementujące kontroli przeciążeń zachowują się bardziej agresywnie przy konkurowaniu o dostępne pasmo, jednak w przypadku wykorzystania całej dostępnej przepustowości możliwa jest utrata dużej części przesyłanych danych - istnieje ryzyko, że urządzenia sieciowe odrzucą dane, ponieważ nie nadążają z ich przetwarzaniem. Z kolei mechanizmy kontroli przepływu zabezpieczają odbiorców przed zalaniem danymi. Jeżeli urządzenie nadawcy wysyła dane szybciej niż urządzenie odbiorcy jest w stanie je odebrać to istnieje ryzyko przepełnienia buforów po stronie odbierającej. Taka sytuacja jest niepożądana, ponieważ może doprowadzić do utraty danych i pogorszenia jakości transmisji.

2.4. Wpływ konfiguracji urządzeń sieciowych na strumienie danych

Konfiguracja Quality of Service może wywierać znaczący wpływ na jakość i szybkość przesyłania danych. QoS to zbiór usług pozwalających na:

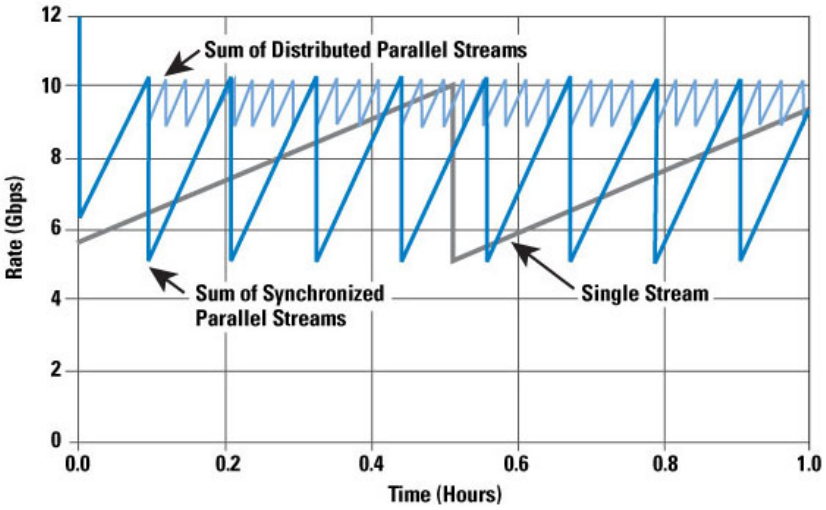
- priorytetyzację wybranych klas ruchu,
- kontrolę opóźnienia,
- kształtowanie dostępnego pasma,
- wybór sposobu odrzucania danych.

Kontrola opóźnienia i priorytetyzacja ruchu jest możliwa poprzez klasyfikację ruchu oraz zastosowanie różnych polityk dla różnych klas ruchu. Klasyfikacja ruchu może odbywać się w oparciu o wykorzystywane protokoły lub aplikacje działające na dobrze znanych portach. Rozpoznanie protokołu na podstawie zawartości pojedynczego pakietu może okazać się niemożliwe (np. RTP), dlatego narzędzia wykorzystywane w tym celu analizują grupy pakietów. Porównując różnice w nagłówkach kolejnych pakietów możliwe jest ustalenie z pewnym prawdopodobieństwem protokołu używanego w danej transmisji. Przykładem narzędzia działającego w ten sposób jest Network Based Application Recognition (NBAR) firmy Cisco. Po zaklasyfikowaniu ruchu, urządzenie sieciowe obsługuje dane zgodnie z zaimplementowanymi politykami. Polityki mogą określać kolejność obsługi danych przez urządzenie, kolejność ich odrzucania w przypadku przeciążenia oraz wpływać na opóźnienie wywołane dużym ruchem w sieci. Rysunek 2.4 przedstawia referencyjny model QoS. Dane znajdujące się wyżej powinny posiadać wyższy priorytet od danych znajdujących się niżej w hierarchi i być obsługiwane w pierwszej kolejności. Podział ten wynika z faktu, że dane czasu rzeczywistego (rozmowy telefoniczne, video-konferencje) są wrażliwe na jitter, opóźnienie i utratę pakietów. Ruch generowany przez protokoły routingu i dotyczący zarządzania siecią również wymaga szybszej obsługi, co przyspieszy konwergencję sieci. Dane typu best effort oraz bulk do których należy zdalne tworzenie kopii zapasowych nie nakładają ograniczeń na jitter i opóźnienie. Dlatego też mogą być obsługiwane z niższym priorytetem.



Rysunek 2.4: Referencyjny model Quality of Service

Zmniejszanie lub zwiększanie dostępnej przepustowości może odnosić się do całego ruchu przechodzącego przez urządzenie sieciowe, bądź tylko do konkretnych klas tego ruchu. Przykładowo możliwe jest ograniczenie ruchu nie posiadającego kontroli przeciążeń w celu usprawnienia transmisji reagujących na zmiany w dostępnym paśmie. Implementacja polityki tego typu pozwoli na zapewnienie bardziej stabilnego podziału łącza pomiędzy transmisje wykorzystujące zróżnicowane protokoły.

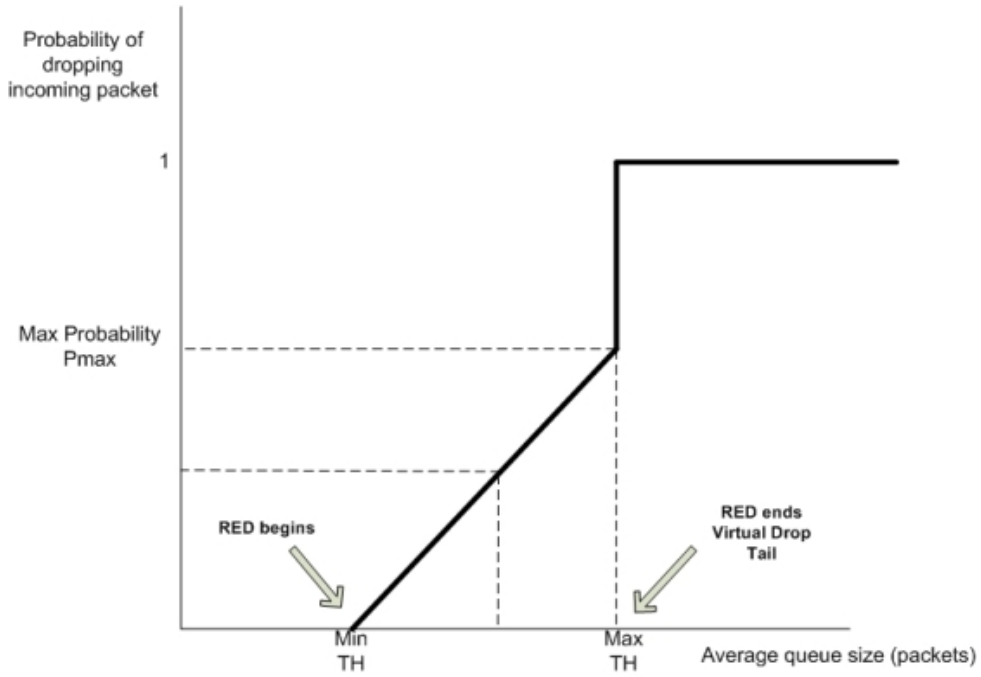


Rysunek 2.5: Problem synchronizacji strumieni.

Mechanizmy decydujące o odrzucaniu części przychodzących danych ze względu na zbyt duże natężenie ruchu również mogą wywierać znaczny wpływ na transmisje danych. W najprostszym przypadku, gdy urządzenie nie nadąża z przetwarzaniem informacji, dane są odrzucane. Takie zachowanie może

prowadzić do niepożądanych zjawisk. Jednym z takich zjawisk jest synchronizacja strumieni TCP zobrazowana na rysunku 2.5.

TCP wykorzystuje mechanizmy kontroli przepływu i przeciążeń w celu zabezpieczenia sieci i odbiorców przed przeciążeniem. Mechanizmy te nadają transmisji specyficzny charakter. Z upływem czasu dane do strumienia wkładane są z coraz większą szybkością, a w chwili wykrycia przeciążenia szybkość ta gwałtownie spada (pojedynczy strumień na rysunku 2.5). Jeżeli w sieci działa kilka strumieni i każdy z nich zacznie tracić dane w tym samym momencie to wspomniane mechanizmy ulegną synchronizacji i nie będzie możliwe pełne wykorzystanie dostępnego pasma (suma zsynchronizowanych strumieni na rysunku 2.5). Ryzyko globalnej synchronizacji strumieni można niwelować poprzez zastosowanie mechanizmu Random Early Detection (RED) zaprezentowanego na rysunku 2.6.



Rysunek 2.6: Mechanizm Random Early Detection, źródło [21]

Mechanizm RED bazuje na długości kolejki (bufora) interfejsu urządzenia sieciowego. Jeżeli ilość danych w kolejce przekroczy wartość Min TH (minimum threshold) to rozpoczyna się proces odrzucania części danych. Dane mogą zostać odrzucone z liniowym prawdopodobieństwem określonym na podstawie zajętości kolejki. Jeżeli długość kolejki przekroszy Max TH (maximum threshold) wszystkie dane pojawiające się na interfejsie zostaną odrzucone. Probabilistyczne podejście do odrzucania danych zapobiega synchronizacji mechanizmów kontroli i pozwala na lepsze wykorzystanie dostępnego pasma.

2.5. Dostarczanie danych do wielu odbiorców

Jeżeli transmisja powinna zostać dostarczona do wielu odbiorców to wykorzystanie komunikacji punkt-punkt jest niewydajne. Pakiety przenoszące informacje muszą być powielane i adresowane do

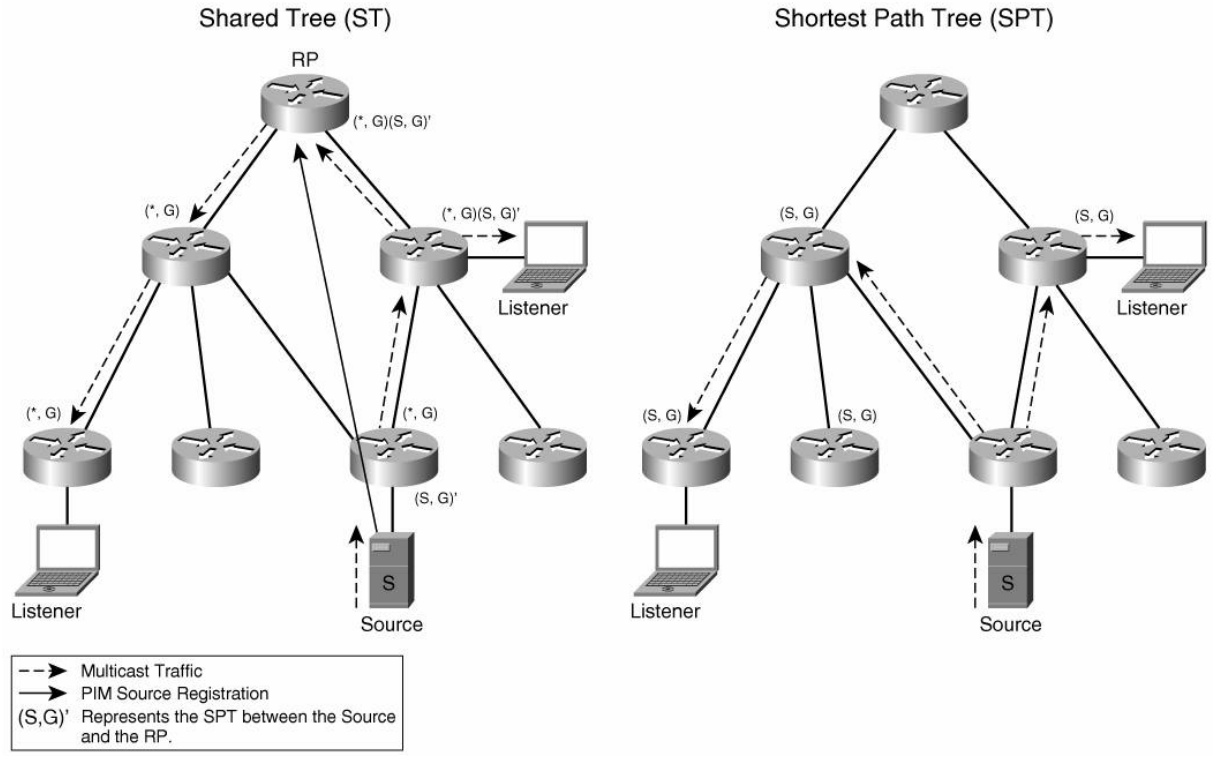
każdego z odbiorców osobno. W celu poprawy wykorzystania łącza możliwe jest skorzystanie z technologii multicast, dzięki której transmisja może dotrzeć do wielu odbiorców bez zbędnego przesyłania tych samych inforamcji wielokrotnie przez to samo łącze.

Najbardziej rozpowszechnionymi protokołami pozwalającymi na transmisje multicast są Protocol Independent Multicast (PIM) oraz Internet Group Management Protocol (IGMP). Protokół IGMP pozwala użytkownikom na powiadamianie urządzeń sieciowych o zamiarze przyłączenia się lub odejścia[6] z grupy multicast. Informacje uzyskane z protokołu IGMP oraz z protokołów routingu unicast są wykorzystywane przez PIM do budowy drzewa rozpinającego sieć w której działa (zob. rysunek 2.7). PIM może działać w kilku z poniżej przedstawionych trybów:

– Sparse mode - polega na budowie współdzielonego drzewa (ST) dla grupy, którego korzeniem jest RP (Rendezvous Point). Zapewnia dobre skalowanie w dużych sieciach.

– Dense mode - polega na budownie najkrótszego drzewa (SPT) poprzez zalewanie, a następnie odcinanie gałęzi sieci w których nie ma odbiorców.

– Source specific mode - tryb pozwalający na odbiór transmisji grupowej od konkretnego źródła.



Rysunek 2.7: Drzewa budowane przez różne tryby protokołu PIM.

Znaczącą wadą tego rozwiązania jest fakt, że Internet obecnie nie jest przystosowany do przesyłania danych multicast na szeroką skalę. Implementacja tej technologii zajmuje zasoby na urządzeniach sieciowych i wymaga protokołów routingu i kontrolowania położenia odbiorców w sieciach (PIM, IGMP).

[6]Powiadamianie o opuszczaniu grupy jest dostępne od drugiej wersji protokołu IGMP.

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 i odtwarzaczy buforujących otrzymywane dane.

Innym podejściem do strumieniowania danych charakteryzują się protokoły typu P2P (Peer-to-peer). W rozwiązaniach opartych na tej technologi 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.

2.6. Podsumowanie

W powyższym rozdziale opisano zjawiska związane z transmisją danych w sieciach komputerowych (jitter, opóźnienie, utrata pakietów) oraz wyjaśniono dlaczego sieci charakteryzują się zmiennością dostępnego pasma. Przedstawiony został wpływ konfiguracji urządzeń sieciowych na strumieniowanie danych oraz podejścia, które można zastosować w przypadku strumieniowania tych samych danych do wielu odbiorców.

Strumieniowanie adaptacyjne, polegające na dostosowaniu bitrate danych multimedialnych do warunków panujących w sieci ma znaczą przewagę nad pozostałymi technologiami - nie jest ograniczone przez zasięg administracyjny organizacji. Technologie Quality of Service i multicast wymagają implementacji na wszystkich urządzeniach pomiędzy nadawcą i odbiorcą. Nie istnieje jednak wymóg implementacji tych technologii i sieci należące do ISP zwykle nie udostępniają takich funkcjonalności. Natomiast strumieniowanie adaptacyjne wymaga jedynie wsparcia ze strony serwera oraz aplikacji klienta i działa w sposób transparentny dla pośrednich urządzeń sieciowych.

3. Przegląd istniejących rozwiązań

3.1. Wstęp

Istnieje wiele protokołów dostarczających funkcjonalności strumieniowania 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 RTSP [10]. W rozdziale zostną również zaprezentowane metody pozwalające na adaptację tych protokołów do warunków panujących w sieci.

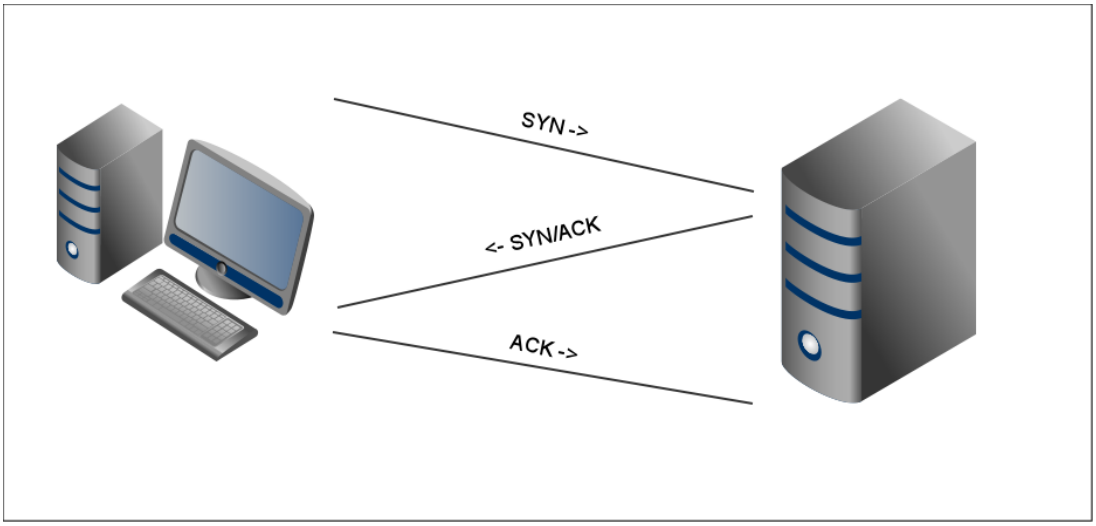
3.2. Transmission Control Protocol

TCP 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) [13]. Oferuje dostarczanie informacji w sposób niezawodny; sprawdza dane pod względem poprawności (sumy kontrolne) oraz kolejności (numery sekwencyjne) [9].

TCP jest protokołem o charakterze połączeniowym. Oznacza to, że przed transmisją danych następuje nawiązanie połączenia pomiędzy nadawcą i odbiorcą. Inicjalizacja połączenia nosi nazwę *Three-way handshake* i jest przedstawiona na rysunku 3.1. W celu nawiązania połączenia klient wysyła segment TCP z ustawioną flagą SYN. W odpowiedzi serwer przesyła segment z ustawionymi flagami SYN i ACK. W celu potwierdzenia połączenia i zakończenia inicjalizacji klient przesyła segment z ustawioną flagą ACK. W trakcie wymiany tych segmentów ustalana jest początkowa wartość numeru sekwencyjnego pozwalającego na indeksowanie przesyłanych danych.

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 zmienijących 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 obliczna w sposób następujący [13]:



Rysunek 3.1: Procedura nawiązania połączenia protokołu TCP.

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

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 regualcji 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* [13]. Jeżeli stacja odbiorcza nie nadąża z przetwarzaniem pakietów, może zmniejszych wielkość okna ogłaszną 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 ginąć. 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

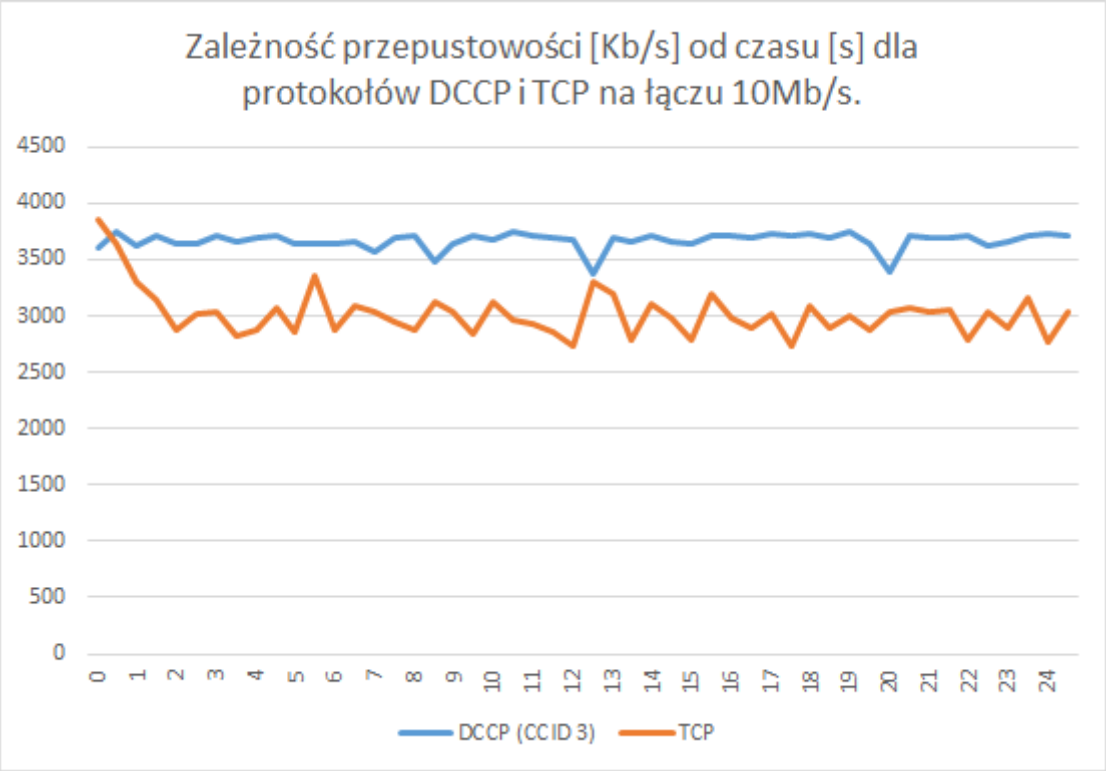
– CCID 3 - TCP-Friendly Rate Control

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 [13]. Omawiany mechanizm zachowuje także liczniki czasu znane z TCP oraz algorytm Slow Start [12]. 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 3.2 przedstawia porównanie działania protokołów TCP i DCCP (CCID 3). W trakcie testu wykorzystano oprogramowanie D-ITG (Distributed Internet Traffic Generator) [17]. 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 $0,5s$. Na przełącznicy zaimplementowano politykę pozwalającą na ustalenie przepustowości pomiędzy stacjami na 10Mb/s (zob. Dodatek A).



Rysunek 3.2: Wykres zależności przepustowości od czasu dla protokołów TCP i DCCP.

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

Rysunek 3.3: Tabela porównawcza protokołów TCP i DCCP.

Tabela z rysunku 3.3 przedstawia porównanie średniej przepustowości i odchylenia standardowego dla protokołów TCP i DCCP. Przeprowadzony test potwierdza zachowanie protokołu DCCP (CCID 3) 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 odbior-

ców do nadawców na temat jakości transmisji. Pakiety RTCP zawierają indentyfikator 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 3.4: 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. Pasmo potrzebne do sprawnej transmisji zależy od rodzaju przesyłanych danych. Rodzaj przesyłanych danych można identyfikować na podstawie pola Payload Type (zob. rysunek 3.4) 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. Real Time Streaming Protocol

Protokół RTSP jest stanowy i należy do warstwy aplikacji modelu ISO/OSI. Dostarcza funkcjonalności pozwalającej na kontrolowanie sesji multimedialnych[1] pomiędzy serwerem a klientem. Sam protokół nie służy do przesyłania danych, a jedynie do przekazywania komend pozwalających na zmianę stanu sesji konkretnego klienta na serwerze. Jako kanał do przekazania danych za pomocą strumieniowania można wykorzystać wspomnianą parę protokołów RTP i RTCP.

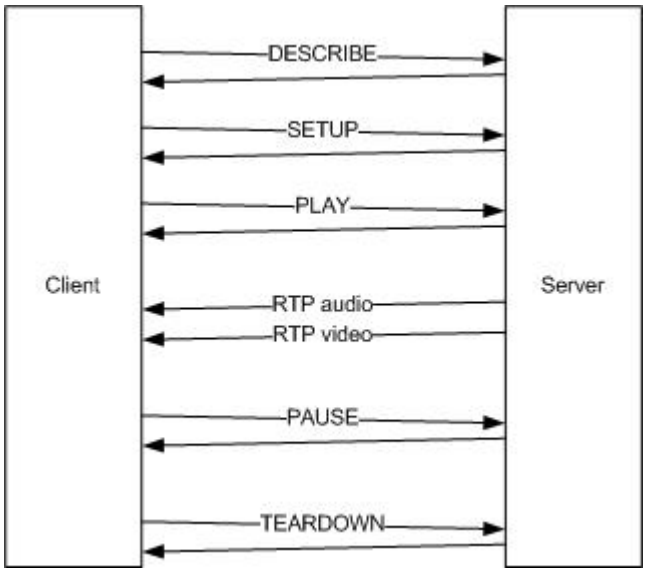
Klient może przekazywać swoje żądania do serwera za pośrednictwem komend w formie teksowej, opisanych w RFC 2326 [10]. Komendy, które implementują serwery RTSP można podzielić na wymagane i opcjonalne. Do wymaganych należą:

- OPTIONS - żądanie określenia zbioru implementowanych komend przez serwer,
- SETUP - żądanie opisu w jaki sposób będzie przebiegał transport danych (unicast/multicast, porty, protokoły),
- PLAY - żądanie rozpoczęcia transmisji danych,
- TEARDOWN - żądanie zakończenia sesji,

[1] sesja multimedialna może składać się z jednego lub kilku strumieni danych

Komendy opcjonalne nie są wymagane w implementacji w pełni funkcjonalnego serwera i należą do nich:

- DESCRIBE- żądanie opisu pliku znajdującego się na serwerze (często w formie SDP[2])
- ANNOUNCE - może zostać przesłane od klienta do serwera jak i od serwera do klienta. W pierwszym przypadku zawiera opis danych znajdujących się pod określonym URL. W drugim przypadku pozwala na aktualizację opisu sesji w czasie rzeczywistym (przydatne w razie dodania/usunięcia strumienia danych w ramach sesji),
- PAUSE - żądanie zatrzymania jednego lub kilku strumieni danych,
- RECORD - żądanie zapisu danych przesyłanych w ramach sesji (przydatne do tworzenia nagrań z videokonferencji),
- REDIRECT - komenda przesyłana od serwera do klienta z informacją na temat lokalizacji poszukiwanego zasobu pod innym adresem,
- SET_PARAMETER i GET_PARAMETER - para komend wykorzystywana do otrzymywania i ustawiania parametrów sesji.



Rysunek 3.5: Przykładowa sesja RTSP, źródło: [23]

Rysunek 3.5 pokazuje przebieg przykładowej sesji RTSP. Klient przesyła żądanie opisu zasobu znajdującego się na serwerze używając komendy DESCRIBE, a następnie używając SETUP ustala sposób transmisji danych. Po wymianie tych informacji, klient jest gotowy do przyjęcia strumienia i zawiadamia o tym serwer komendą PLAY. Po otrzymaniu tej komendy serwer rozpoczyna strumieniowanie danych. W trakcie strumieniowania, klient może wysłać komendę PAUSE wstrzymującą przepływ danych. W celu zakończenia połączenia wysyłana jest komenda TEARDOWN.

[2]Session Description Protocol - tekstowy format opisu zasobów.

3.6. Podsumowanie

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

- kontrola przeciążeń (TCP, DCCP) oraz przepływu (TCP),
- jednostajne wykorzystanie łącza (DCCP CCID3),
- kanał zwrotny przenoszący informacje na temat sesji (RTCP, RTSP),
- możliwość dostosowana parametrów transmisji do warunków w sieci (TCP),
- funkcjonalność pozwalająca na sterowanie strumieniem danych (RTSP).

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. Jest to technologia spokrewniona z Microsoft Smooth Streaming [14], Adobe Systems HTTP Dynamic Streaming [15] oraz Apple Inc. HTTP Live Streaming [16].

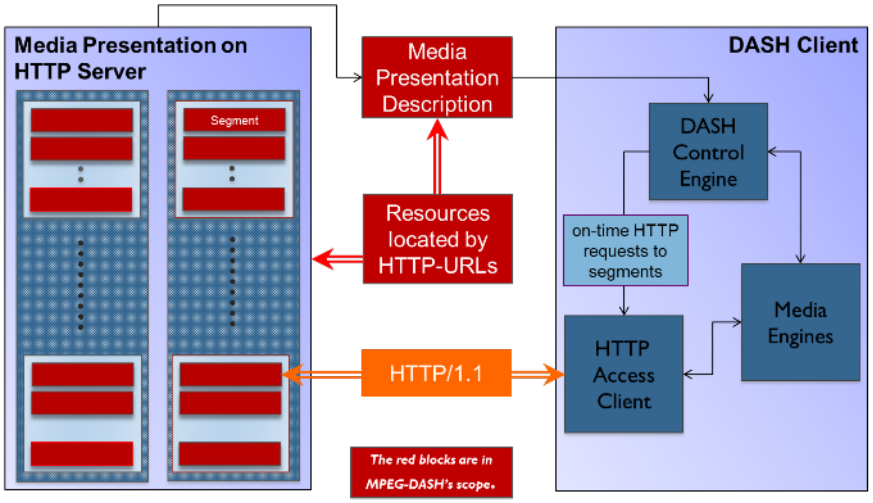
Prace nad standardem prowadzone przez grupę MPEG rozpoczęły się w 2010 roku. W 2011 standard DASH pojawił się jako draft, a międzynarodowym standardem stał się w kwietniu 2012 dzięki publikacji ISO/IEC 23009-1:2012 [11]. Pierwsza demonstracja na szerszą skalę systemów opartych na DASH miała miejsce podczas igrzysk w Londynie w 2012 roku. Wdrożony tam system pozwalał 1000 użytkowników na żywo śledzić wydarzenia sportowe za pomocą laptopów, smartfonów i tabletów. W lipcu 2013 wprowadzono do standardu poprawki.

Do obecnych implementacji DASH zaliczają się:

- DASH Player i wtyczka do VLC, które powstały na uniwerstytecie ITEC w Klagenfurt (październik 2011),
- biblioteki libdash (open-source) oraz bitdash (rozwiązanie komercyjne) należące do austriackiej firmy bitmovin GmbH (styczeń 2013),
- zestaw narzędzi do przetwarzania plików multimedialnych oraz zbiór odtwarzaczy (dla HTML5, Flash, Silverlight, Chromecast) należący do castLabs GmbH (marzec 2014).

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 serwer HTTP (np. Apache lub Nginx) na którym umieszczone zostaną dane przeznaczone do strumieniowania.



Rysunek 4.1: Standard MPEG-DASH i przykładowa architektura klienta DASH, źródło: [18]

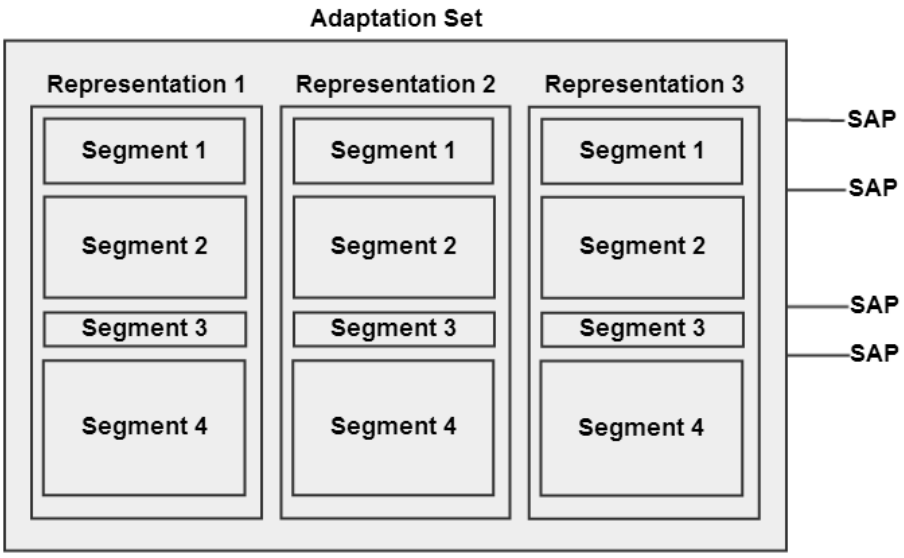
Dane multimedialne, które będą podlegały operacji strumieniowania, należy odpowiednio przygotować. Powinny one być dostępne w kilku wersjach[1] różniących się jakością (a co za tym idzie również wielkością). Każda kopia powinna zostać podzielona na ponumerowane części (segmenty), które razem tworzą spójną całość. Na rysunku 4.1 prostokąt po lewej stronie symbolizuje serwer WWW. Kopie pliku multimedialnego (niebieskie prostokąty wewnątrz serwera) składają się z segmentów (czerwone prostokąty).

Od długości segmentów zależy szybkość przystosowywania się aplikacji DASH do zmieniających się warunków w sieci (np. przepustowości). Aplikacja klienta może dokonać przełączenia pomiędzy wersjami strumieniowanych danych tylko podczas tzw. *Stream Access Points* (SAP). Są to pozycje w strumieniu danych multimedialnych, które nie wymagają informacji zawartych we wcześniejszych segmentach w celu odtwarzania późniejszych segmentów. Zwykle te pozycje znajdują się pomiędzy kolejnymi segmentami w celu uproszczenia procedur adaptacyjnych i ich implementacji. Innym częstym zabiegiem upraszczającym strumieniowanie i odtwarzanie danych jest wyrównanie czasu trwania odpowiadających sobie segmentów dla całego zbioru (*Adaptation Set*) wersji danych multimedialnych (*Representation*)[2]. Powyższy przykład został zobrazowany na rysunku 4.2.

Przygotowane dane należy opisać za pomocą języka XML w postaci dokumentu MPD (Media Presentation Description). Plik MPD pozwala aplikacji klienta na poznanie struktury danych znajdujących się na serwerze. Zawiera adresy URL poszczególnych segmentów, czasy ich trwania, informacje na temat

[1] Akceptowalne jest przygotowanie tylko jednej wersji danych, ale uniemożliwi to operację strumieniowania adaptacyjnego.
[2] Elementy *Representation* oraz *Adaptation Set* zostały szerzej opisane w podrozdziale 4.4.

kodowania danych, rozdzielczości oraz minimalnych przepustowości potrzebnych do strumieniowania danych i ich odtwarzania w czasie rzeczywistym. Dokument MPD musi zostać dostarczony do aplikacji klienta, która następnie parsuje jego zawartość i rozpoczyna strumieniowanie danych. W trakcie strumieniowania, aplikacja klienta wybiera w sposób dynamiczny wersję danych i pobiera kolejne segmenty za pomocą protokołu HTTP.



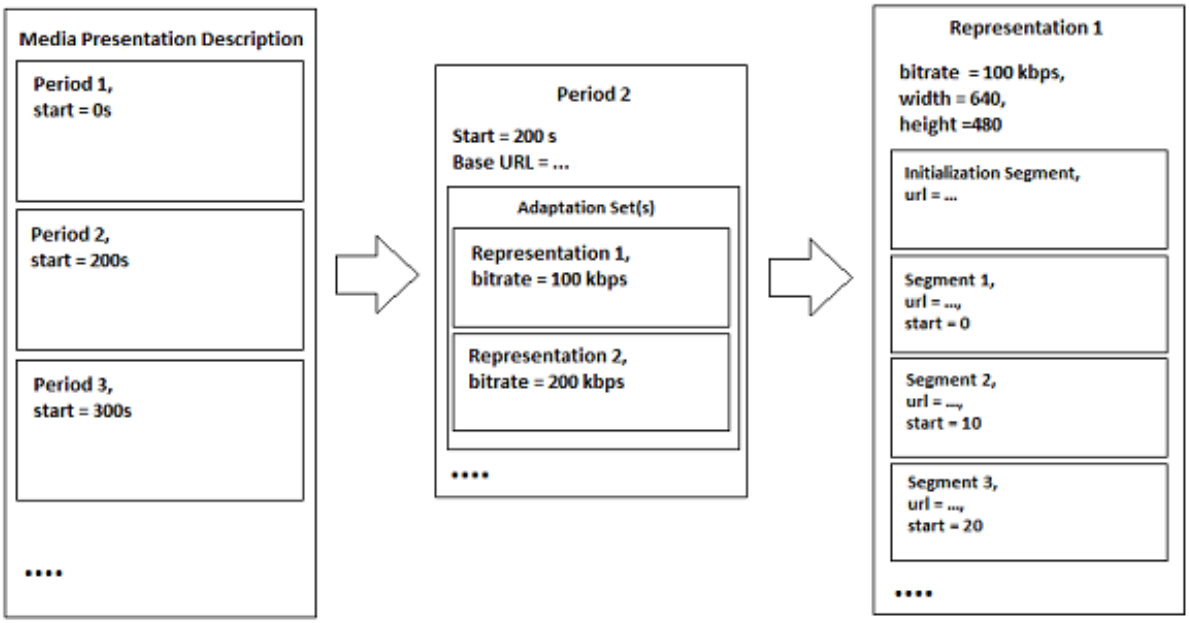
Rysunek 4.2: Wizualizacja struktury danych pozwalającej na efektywne strumieniowanie adaptacyjne.

Dzięki hierarchicznemu ułożeniu danych na serwerze i funkcjonalności logicznego odseparowania danych do osobnych strumieni możliwe jest dokonanie personalizacji przesyłanych danych. Przykładowo, jeżeli plik video znajdujący się na serwerze posiada kilka różnych wersji językowych napisów to możliwy jest automatyczny wybór odpowiednich napisów na podstawie preferencji klienta lub języka ustawionego na jego urządzeniu.

Standard DASH działa niezależnie od sposobu kodowania danych i jest łatwo implementowalny w Internecie, ponieważ jest oparty na stosie TCP/IP. Dzięki wykorzystaniu protokołu HTTP może również współpracować z istniejącą infrastrukturą w postaci filtrów, zapór, urządzeń NAT i cache [18].

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 4.3). Lewy prostokąt symbolizuje plik MPD, który zawiera kilka elementów typu *Period* (środkowy prostokąt na rysunku 4.3) 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:



Rysunek 4.3: Struktura modelu danych, na podstawie [11]

– zbiór dostępnych wersji plików multimedialnych,

– wersje językowe dla dźwięku,

– wersje językowe napisów,

– sposób kodowania danych,

– itd...

Parametry te pozostają niezmienne w pojedynczym elemencie *Period*, ale mogą występować różnice pomiędzy kolejnymi elementami tego typu. Wszystkie elementy *Period* posiadają znacznik czasowy pozwalający na 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 4.3). Dane multimedialne w ramach *Representation* są wystarczające do odtworzenia pełnego i spójnego pliku multimedialnego zanim został podzielony na części (*Segments*). 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[3]) lub innych czynników.

Dane multimedialne w ramach *Representation* są podzielone na wiele części typu *Segments*. 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. Możliwy jest dalszy podział danych w obrębie pojedynczego elementu *Segment* na *Subsegments*. W celu umożliwienia synchronizacji podczas odtwarzania danych wykorzystywany jest *Segment index* dostarczający informacji o położeniu *Subsegments* na osi czasu prezentacji (zob. 4.5.1). *Segment index* ma znaczenie lokalne w obrębie pojedynczego elementu *Segment*. Aplikacja klienta ma możliwość pobrania *Segment index* i następnie wysłania zapytań HTTP dotyczących jedynie wybranych *Subsegments*.

4.5. DASH timelines

Standard DASH definiuje dwie osie czasu. Pierwsza z nich, opisana w sekcji 4.5.1 stanowi główny komponent standardu odpowiadający za synchronizację pobieranych danych multimedialnych. *Segment availability timeline* pozwala na sprawne zarządzanie dostępnością segmentów potrzać transmisji w czasie rzeczywistym.

4.5.1. Media presentation timeline

Pierwsza z dwóch osi czasu 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 4.3) 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* elemntu *Period* do którego dany *Segment* należy oraz wartości atrybutu *start* dla danego elementu typu *Segment*.

4.5.2. Segment availability timeline

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 modelów “On Demand” takich jak VoD[4] 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 wprowadze-

[3]Round Trip Time - minimalny czas potrzebny na komunikację od nadawcy do odbiorcy i z powrotem.

[4]Video On Demand - usługa pozwalająca na odtwarzanie danych multimedialnych w wybranym przez użytkownika czasie, późniejszym niż czas emisji.

nia powtórek, przewijana 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 4.4).

Rysunek 4.4: Model referencyjny klienta dla metryki DASH, źródło: [11]

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 zapełnienia 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 ([11]) w dodatku D.

4.7. DASH profiles

Standard DASH definiuje kilka różnych profili identyfikowanych poprzez URN[5] i dopuszcza możliwość zdefiniowania własnego profilu identyfikowalnego za pomocą URN lub URL. Pojęcie profilu DASH jest konceptualnie podobne do profilu RTP/RTCP (zob. 3.4) - jest to zestaw parametrów określających transmisję. Profile DASH mogą definiować ograniczenia lub wymogi odnoszące się zarówno do danych multimedialnych jak i do struktury tych danych znajdującej się na serwerze. W przypadku danych multimedialnych profil może określać np: obsługiwane kodeki[6], kontenery[7] oraz rozdzielczość. Ograniczenia nałożone na strukturę danych dotyczą zawartości plików MPD: ilości reprezentacji (*Representation*), czasu trwania i wielkości segmentów, dostępnych bitrate, itp. Poniżej opisane zostały dwa z sześciu profili zdefiniowanych przez standard DASH. Specyfikacje pozostałych profili można znaleźć w [11].

4.7.1. On Demand profile

Ten profil został stworzony w celu dostarczenia podstawowych funkcjonalności w modelach On Demand. Cechą charakterystyczną tego profilu jest wymóg, by każdy element *Representation* dostarczał pojedynczy *Segment* podzielny na wiele elementów *Subsegment*. Początek każdego z elementów typu *Subsegment* powinien stanowić *Stream Access Point*. Oznacza to, że aplikacja klienta może bez przeszkód rozpocząć odtwarzanie danych multimedialnych od dowolnego wybranego elementu *Subsegment*.

[5] Uniform Resource Name - jednolity format nazewnictwa zasobów.

[6] urządzenie lub program, który potrafi przekształcić strumień danych. Służy do kodowania i dekodowania strumieni danych. Przykłady: MP3, H.263, MPEG-2.

[7] kontenery multimedialne poza danymi zawierają metadane potrzebne do właściwego odtwarzania danych multimedialnych. Przykłady: AVI, OGG, MKV.

Aplikacja klienta może wcześniej zostać zainicjalizowana metadanymi znajdującymi się w odpowiednim *Initialization Segment* lub każdy z *Media Segments* może posiadać własne dane inicjalizujące.

Profil narzuca również pewne ograniczenia dotyczące struktury danych multimedialnych. Plik MPD nie może ulegać aktualizacjom w trakcie strumieniowania (tak jak to ma miejsce w przypadku transmisji na żywo). Wszystkie elementy *Representation* muszą posiadać swoje *Base URL* oraz posiadać parameter *subsegmentStartWithSAP* ustawiony na wartość true. Ponadto dla dowolnych dwóch elementów *Representation* w tym samym *Adaptation Set* żadne dwa elementy *Segment* nie mogą na siebie zachodzić (pod względem czasu odtwarzania) jeżeli ich numery sekwencyjne są różne (paramter *segmentAlignment* ustawiony na true).

Wprowadzenie powyższych ograniczeń pozwala na skalowalne i wydajne wykorzystanie serwerów HTTP oraz ułatwia przełączanie pomiędzy wersjami danych multimedialnych w trakcie strumieniowania.

4.7.2. Live profile

Profil Live przeznaczony jest do obsługi strumieniowania danych multimedialnych na żywo. Wymogiem profilu jest krótki czas trwania elementów typu *Segment*, standard nie precyzuje jednak tego wymogu za pomocą mierzalnych wartości. Od długości elementów *Segment* zależeć będzie opóźnienie z jakim aplikacje klienckie będą otrzymywać dane. Im dłuższy *Segment*, tym dłużej będzie trwało zebranie danych, ich zakodowanie, a następnie przesłanie do klientów.

Podobnie jak w przypadku profilu On Demand, profil Live nakłada ograniczenie, dzięki któremu przełączanie pomiędzy różnymi wersjami tych samych danych jest ułatwione (*segmentAlignment*). Ponadto, każda reprezentacja musi zawierać jeden *Initialization Segment* oraz co najmniej jeden *Media Segment*.

4.8. Podsumowanie

DASH oferuje bogatą funkcjonalność w zakresie strumieniowania danych multimedialnych. Może zostać wykorzystany zarówno w systemach On Demand jak i do transmisji na żywo. Jako standard nie definiuje żadnych algorytmów strumieniowania adaptacyjnego, ale opisuje środowisko w którym takie algorytmy znalazły zastosowanie. Stos protokołów, który wykorzystuje decyduje o jego zaletach i wadach. Wykorzystanie protokołu TCP może powodować spore opóźnienia w transmisji, co zmniejsza przydatność DASH przy transmisjach na żywo. Zastosowanie protokołu HTTP pozwala na łatwe wdrożenie implementacji opartych na DASH w infrastrukturach sieciowych.

Dalsza część pracy skupia się na zagadnieniach algorytmów strumieniowania adaptacyjnego, testach ich skuteczności oraz problematyce pomiarów warunków panujących w sieciach komputerowych. Pomiary wykonywane w trakcie działania aplikacji nie mogą wpływać na sie. Niesie to za sobą konieczność oparcia tych pomiarów na parametrach transmisji wykorzystywanych do pobierania niewielkich segmentów danych multimedialnych.

5. Opis projektu i implementacji

5.1. Wprowadzenie

W ramach projektu powstał odtwarzacz multimedialny umożliwiający adaptację strumieniowania w oparciu o standard DASH-MPEG. Poniższy rozdział zawiera informacje opisujące algorytm wykorzystany do implementacji strumieniowania adaptacyjnego. Pomysł algorytmu oparty o wykorzystanie strukturę kontrolera PID został zaczerpnięty z artykułu "Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming" [1]. Dalsza część rozdziału skupia się na szczegółach implementacyjnych odtwarzacza oraz narzędziach i bibliotekach wykorzystanych do jego stworzenia.

5.2. Opis algorytmu

Prezentowany poniżej algorytm służy do wygładzenia procesu adaptacji strumienia danych multimedialnych. Oznacza to, że przy dużej zmienności warunków panujących w sieci alogrytm będzie starał utrzymać jakość przesyłanych danych (reprezentację) na stabilnym poziomie.

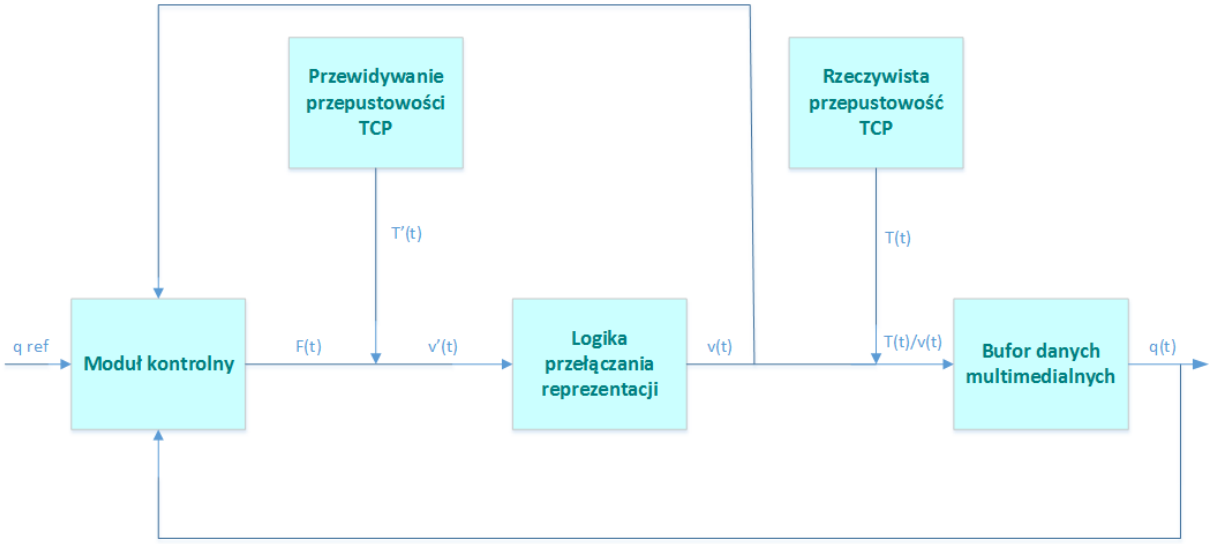
Branie pod uwagę jedynie dostępnej przepustowości z danej chwili podczas wyboru konkretnej reprezentacji danych może prowadzić do ciągłych i widocznych dla użytkownika zmian w jakości oglądanego materiału. Oparcie algorytmu na idei kontrolera PID pozwala na kompensację różnic pomiędzy wartością dostępnej przepustowości oraz bitrate wybranej reprezentacji i zwiększa jego tolerancję na chwilowe zmiany w dostępnym paśmie. Kontroler PID wykorzystuje pętlę sprzężenia zwrotnego i składa się z trzech komponentów:

– P *(proportional)* - odpowiada za kompensację różnicy bieżącej,

– I *(integral)* - kompensuje akumulację różnic z przeszłości,

– D *(derivative)* - kompensuje przyszłe przewidywane różnice.

Ogólna zasada działania mechanizmu adaptacji została przedstawiona na rysunku 5.1 oraz w tabeli objaśniającej wykorzystane oznaczenia 5.2. Dalsza część podrozdziału została podzielona na sekcje opisujące wybrane elementy zaimplementowanego rozwiązania.



Rysunek 5.1: Schemat adaptacji w oparciu o kontroler PID, na podstawie: [1]

Oznaczenie	Objaśnienie
q ref	referencyjna (docelowa) zajętość bufora danych.
F(t)	czynnik adaptacji.
T'(t)	przewidywana przepustowość w chwili t.
v'(t)	przewidywany bitrate danych multimedialnych.
v(t)	bitrate danych po adaptacji i kwantyzacji.
T(t)	zmierzona przepustowość w chwili t.
q(t)	poziom zajętości bufora danych w chwili t.

Rysunek 5.2: Tabela oznaczeń z rysunku 5.1.

5.2.1. Moduł kontrolny

Moduł kontrolny wykorzystuje kilka parametrów do wyliczenia czynika adaptacji *F(t)* potrzebnego do ustalenia wstępnego bitrate danych *v'(t)*. Czynnik adaptacji *F(t)* jest iloczynem dwóch wartości:

– *Czynnik zajętości bufora* - liczony na podstawie zajętości referencyjnej *q ref* oraz zajętości rzeczywistej *q(t)*. Im większa różnica pomiędzy tymi dwoma wielkościami, tym większa będzie szybkość adaptacji. W wypadku, gdy wartości będą zbliżone lub równe, powyższy czynnik będzie miał neutralny wpływ na wartość czynnika adaptacji.

– *Czynnik trendu bufora* - obliczany na postawie kolejnych wartości *q(t)*. Jeżeli w danych w buforze przybywa to jest to sygnał do zwiększenia wartości czynnika trendu bufora. Jeżeli bufor jest opróżniany to ten czynnik odpowiednio maleje. Utrzymywanie się poziomu zajętości bufora na jednakowym poziomie powoduje, że trend bufora nie wpływa na wartość czynnika adaptacji.

5.2.2. Przewidywanie przepustowości TCP

Algorytm ma do dyspozycji dwa sposoby przewidywania przyszłej przepustowości łącza TCP. Możliwy jest wybór metody predykcji podczas uruchamiania aplikacji odtwarzacza multimedialnego.

Pierwsze metoda polega na liczeniu średniej z kilku ostatnich przechowywanych pomiarów przepustowości. Odrzucane są skrajne pomiary, a wielkość bufora na przetrzymywane pomiary jest konfigurowalna.

Druga metoda wykorzystuje wykładniczą średnią kroczącą. Zaletą metody jest szybkie dostosowanie się do długofalowych trendów zmian w przepustowości i dodatkowa ochrona przed krótkotrwałymi zmianami w paśmie.

5.2.3. Logika przełączania reprezentacji

Moduł odpowiedzialny za przełączanie reprezentacji został opisany za pomocą poniższego pseudokodu:

Algorithm 1 Logika przełączania reprezentacji

1: $v'(t) = F(t) \cdot T'(t)$
2: **if** $q(t) < \frac{q_{ref}}{2}$ **then**
3: $\quad v(t) = Q(T(t-1))$
4: $\quad return$
5: **else if** $v'(t) > v(t-1)$ **then**
6: $\quad Counter++$
7: $\quad$ **if** $Counter > m$ **then**
8: $\quad\quad v(t) = Q(T'(t))$
9: $\quad\quad Counter = 0$
10: $\quad\quad return$
11: $\quad$ **end if**
12: **else if** $v'(t) < v(t-1)$ **then**
13: $\quad Counter = 0$
14: **end if**
15: $v(t) = v(t-1)$
16: $return$

Funkcja Q jest funkcją kwantyzacji. Bitrate *v'(t)* obliczany na podstawie przewidywanej przepustowości oraz czynnika adaptacji (pierwsza linia) może przyjmować dowolne wartości. Istnieje jednak skończona liczba reprezentacji o określonych wartościach bitrate. Funkcja Q odwzorowuje dowolny bitrate na wartość największą, ale nie większą niż argument funkcji i pochodzącą ze zbioru składającego się z bitrate dostępnych reprezentacji. W drugiej linii aktualny poziom bufora jest porównywany z poziomem referencyjnym i jeżeli będzie co najmniej dwukrotnie mniejszy to istnieje ryzyko, że z bufora

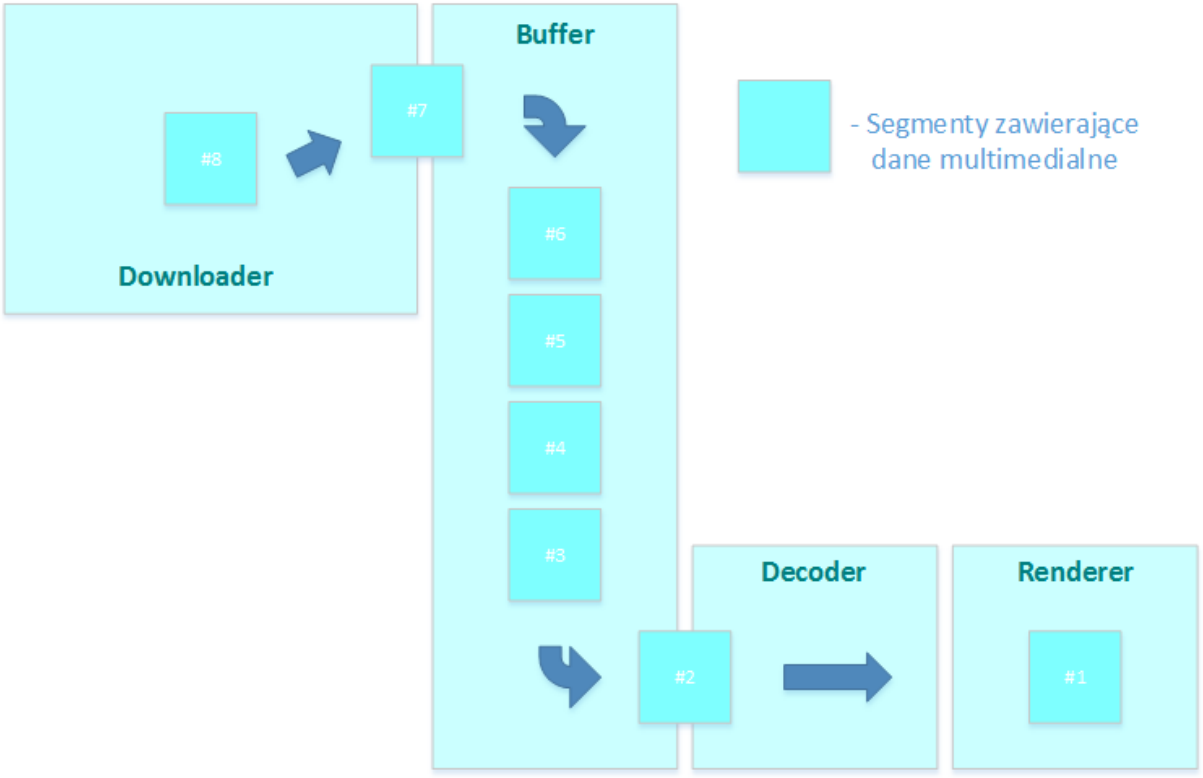
wyciągnięte zostaną wszystkie dane multimedialne. W celu redukcji tego zagrożenia za wynikowy bitrate przyjmowana jest skwantyzowana wartość rzeczywistej przepustowości zmierzona podczas pobierania poprzedniego segmentu danych.

Jeżeli warunek z linii 2 nie zachodzi to w następnej kolejności sprawdzane jest czy przewidywany bitrate jest większy niż bitrate ostatnio pobranego segmentu (linia 5). Za każdym razem gdy warunek z linii nr 5 jest spełniony, zwiększana jest wartość licznika *Counter*, który po przekroczeniu wartości *m* jest zerowany, a wynikowy bitrate jest ustalany jako skwantyzowana wartość przewidywanej przepustowości (linia 8).

Jeżeli warunki z linii 2 i 5 nie są spełnione to sprawdzane jest czy przewidywany bitrate jest mniejszy od bitrate poprzednio pobranego segmentu. Jeżeli tak to licznik jest zerowany, a wynikowy bitrate jest taki sam jak w poprzedniej iteracji algorytmu.

5.3. Wysokopoziomowa koncepcja rozwiązania

Odtwarzacz multimedialny składa się z 4 komponentów przedstawionych na rysunku 5.3.



Rysunek 5.3: Koncepcja rozwiązania

Pierwszy z komponentów, *Downloader*, odpowiada za komunikację z serwerem HTTP na którym znajdują się dane multimedialne przeznaczone do strumieniowania. Jest różnież odpowiedzialny za zbieranie danych związanych z transmisją:

– czas rozpoczęcia pobierania segmentu danych,

– czas zakończenia pobierania segmentu danych,

– wielkość pobranego segmentu danych,

– pozostałe elementy metryki wyszczególnione w standardzie DASH-MPEG.

Downloader po zakończeniu pobierania segmentu danych multimedialnych przekazuje go do bufora.

Ze względu na wielowątkowość aplikacji, bufor synchronizuje dostęp do przechowywanych segmentów danych multimedialnych. Architektura wykorzystująca model producenta i konsumenta pozwala na niezależne działanie komponentu *Downloader* i komponentów zajmujących się przetwarzaniem i prezentacją danych.

Dane multimedialne są wyjmowane z bufora i dekodowane - zajmuje się tym *Decoder*. Po przetworzeniu dane przekazywane są do komponentu *Renderer* odpowiedzialnego za prezentację zdekodowanych danych użytkownikowi.

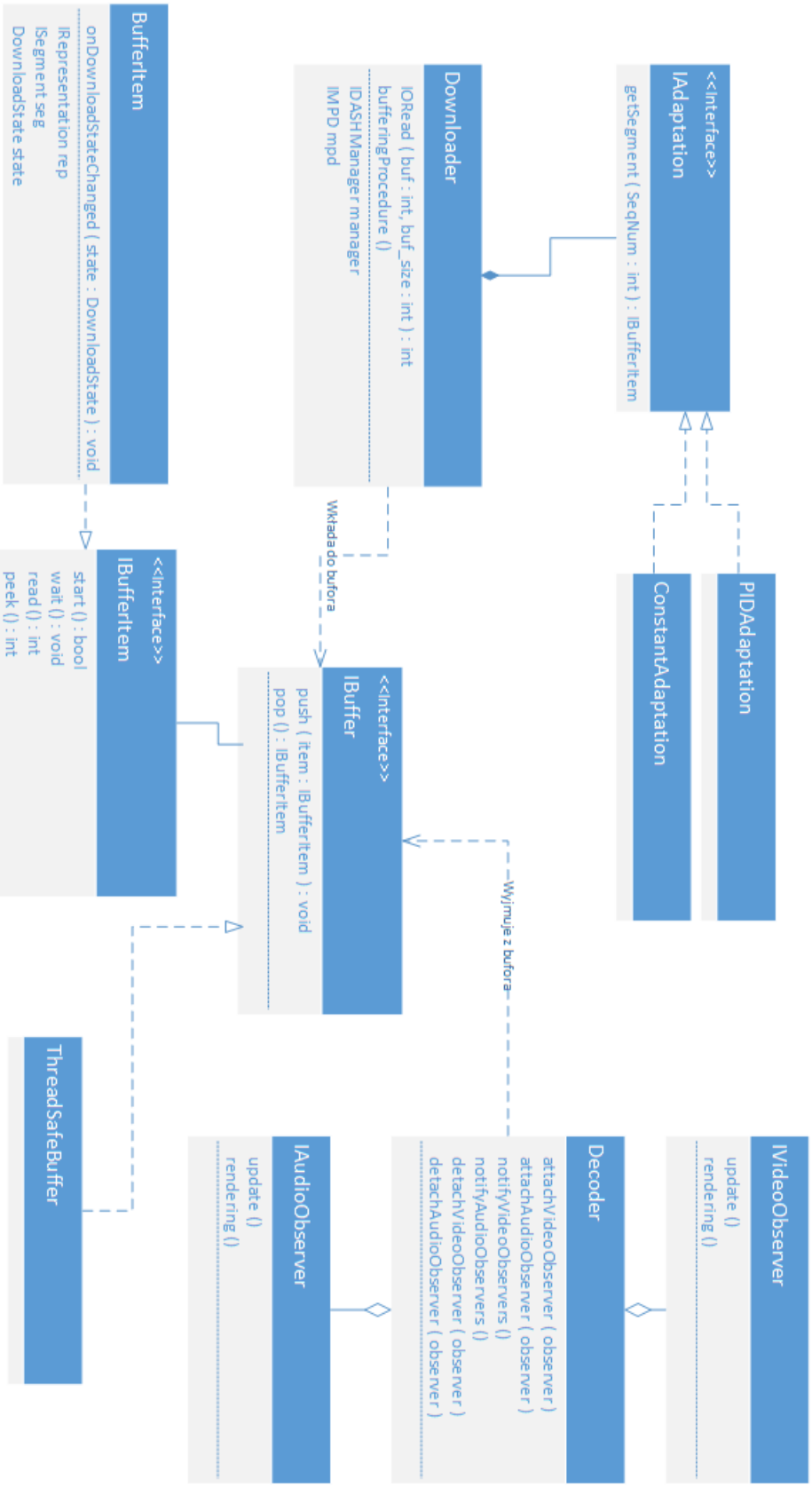
5.4. Implementacja projektu odtwarzacza

Diagram klas na rysunku 5.4 przedstawia najważniejsze klasy i powiązania pomiędzy nimi. Poniższy podrozdział skupia się na wyjaśnieniu przeznaczenia poszczególnych klas oraz ich wzajemnych zależności.

Obiekt klasy *Downloader* jest odpowiedzialny za komunikację z serwerem HTTP i pobieraniem kolejnych segmentów danych multimedialnych. Podczas uruchamiania odtwarzacza należy podać lokalizację pliku MPD (Media Presentation Description), który następnie jest pobierany i parsowany przez obiekt implementujący interfejs *IDASHManager* należący do biblioteki libdash. Pobieranie konkretnego segmentu jest delegowane do osobnego wątku wykonującego instrukcje funkcji *bufferingProcedure*. W celu wybrania reprezentacji z której należy skorzystać przy pobieraniu segmentu danych multimedialnych *Downloader* korzysta z klas implementujących *IAdaptation*. Segmenty, których pobieranie zakończyło się sukcesem wkładane są do bufora implementującego interfejs *IBuffer*.

Logika adaptacji strumienia polegająca na wyborze kolejnej wersji danych multimedialnych (ich reprezentacji) jest zawarta w klasach implementujących interfejs *IAdaptation*. Obiekt klasy *Downloader* korzysta z funkcji *getSegment* w celu uzyskania nowego obiektu implementującego interfejs *BufferItem*, który na obecnym etapie cyklu życia reprezentuje jeszcze nie pobrany segment danych. Obiekty klas implementujących *IAdaptation* zajmują się wyborem reprezentacji, tworzeniem nowego obiektu obudowującego wybrany segment danych oraz przekazaniem go do obiektu *Downloader*. Odtwarzacz pozwala na wybór algorytmu adaptacji. Obecnie możliwy jest wybór jednego z dwóch mechanizmów adaptacji:

– *InteractiveAdaptaion* - pozwala na sterowanie wyborem reprezentacji w czasie rzeczywistym. Możliwy jest wybór dowolnej z reprezentacji opisanych w pliku MPD, którego lokalizacja została



Rysunek 5.4: Diagram klas

podana przy starcie odtwarzacza. W przypadku wyboru nieprawidłowej reprezentacji, odtwarzacz zwraca informację o błędzie i pracuje z wykorzystaniem poprzedniej reprezentacji.

– *ConstantAdaptation* - wybór reprezentacji należy podać przy uruchamianiu odtwarzacza i pozostaje on niezmienny przez cały czas działania aplikacji. Jeżeli reprezentacja nie znajduje się w zbiorze dostępnych reprezentacji opisanych plikiem MPD to aplikacja odtwarzacza zostaje zamknięta i zwrócona zostaje informacja o błędzie.

– *PIDAdaptaion* - adaptacja strumienia opierająca się o kontroler PID i algorytm opisany w podrozdziale 5.2. Adaptacja zachodzi automatycznie i nie wymaga interwencji użytkownika.

Obiekty klas implementujących *IBufferItem* stanowią kontener na dane pobierane z serwera. Składają się przede wszystkim z obiektów klas należących do biblioteki libdash oraz zawierają stan pozwalający na stwierdzenie, czy segment, który obiekt reprezentuje został już pobrany. Można na nich wykonywać operacje rozpoczęcia pobierania (*start*), oczekiwania na zakończenie pobierania (*wait*) oraz operacje podglądania danych w segmencie i ich czytania.

Klasa *ThreadSafeBuffer* stanowi implementację interfejsu *IBuffer*. Obiekt tej klasy pozwala na synchronizowany dostęp wielu wątków do kolejki zawierającej elemnty implementujące *IBufferItem*. Wykorzystany został współbieżny wzorzec projektowy producent-konsument. Producentem jest obiekt klasy *Downloader*, natomiast konsumentami mogą być obiekty typu *Decoder*. Procedura *push* pozwala na wkładanie elementów na początek bufora, natomiast procedura *pop* zwraca ostatni element przy okazji usuwając go z bufora.

Decoder zajmuje się dekodowaniem danych zawartych w kolejnych segmentach. Po zdekodowaniu segmentu danych multimedialnych powiadamiane są obiekty odpowiedzialne za prezentację danych użytkownikowi. Klasy obiektów odpowiedzialnych za prezentację implementowane są jako obserwatory obiektów klasy *Decoder*. *Decoder* pozwala na rejestrację i odrejestrowanie wielu obiektów pozwalających na renderowanie danych audio oraz video.

5.5. Wykorzystane narzędzia i biblioteki

Projekt odtwarzacza powstał w języku C++ i z wykorzystaniem Microsoft Visual Studio 2010. Budowanie aplikacji odbywa się za pomocą programu CMake. Odtwarzacz korzysta z bibliotek libav, libcurl, libxml2, sdl, zlib, boost oraz biblioteki libdash implementującej standard DASH-MPEG. Wykorzystany został system kontroli git.

6. Wdrożenie i testy

6.1. Instalacja odtwarzacza

Aplikacja została napisana na systemie Windows 7 (64 bit) i pakiet instalacyjny zawiera większość bibliotek potrzebnych do uruchomienia odtwarzacza na tym systemie. Działanie odtwarzacza nie zostało przetestowane na systemach Unix/Linux, iOS oraz wcześniejszych wersjach systemu Windows.

W celu instalacji aplikacji odtwarzacza użytkownik musi wcześniej zainstalować bibliotekę boost w wersji 1.55.0. Instalacja odtwarzacza polega na jego zbudowaniu za pomocą programu CMake (w opcjach należy podać ścieżkę do zainstalowanej wcześniej biblioteki boost). Po zbudowaniu aplikacji ze źródeł, program jest gotowy do działania. Interakcja z programem odbywa się z pomocą linii komend. Przykładowa komenda służąca do kompilacji źródeł odtwarzacza znajduje się w dodatku B.

6.2. Instrukcja użytkownika

Uruchamianie i sterowanie działaniem odtwarzacza odbywa się za pomocą linii komend. Na listingu 6.1 przedstawiono tekst pomocy wyświetlającej się po uruchomieniu aplikacji z opcją *-h*.

Listing 6.1: Skrócona instrukcja użytkowania odtwarzacza.

```
...\bin>clientdash.exe -h
Allowed options:
  -h [ --help ]                        produce help message
  -u [ --url ] arg                     specify mpd url
  -b [ --buffer ] arg (=30)            specify buffer size
  -l [ --log ] arg (=clientdash.log)   specify log path
  -a [ --adaptation ] arg (=Constant) specify adaptation algorithm
  -r [ --repID ] arg (=0)              specify representation for constant
                                       adaptation
  -t [ --base_t ] arg (=1000000)       specify base throughput for PID
                                       adaptation
```

Opcje służące do sterowania działaniem programu można podawać zarówno w formie skróconej (*-h*) oraz pełnej (*–help*). Jedynym wymaganym argumentem nie posiadającym domyślnej wartości jest

adres URL pliku Media Presentation Description (*-u* lub *–url*). Odtwarzacz, po sparsowaniu zawartości tego pliku będzie mógł poznać lokalizację i strukturę danych multimedialnych do których użytkownik chciałby uzyskać dostęp.

Kolejnym z możliwych do określenia parametrów jest wielkość bufora na dane multimedialne, która jest liczona w ilości segmentów. Jeżeli czas trwania każdego segmentu wynosi dwie sekundy i bufor może pomieścić 30 segmentów to całkowity czas playback wynosi 60 sekund.

Parametr *-l* określa lokalizację pliku do którego zbierane są logi z działania programu. Domyślnie plik nazywa się *clientdash.log* i znajduje się w tym samym katalogu co aplikacja odtwarzacza. W celu zmiany pliku logowania należy podać jego nazwę poprzedzoną pełną ścieżką.

Odtwarzacz pozwala na wybór metody adaptacji. Dostępne są trzy wartości dla opcji *-a*:

– Constant - wartość domyślna oznaczająca, że aplikacja będzie korzystać tylko z jednej reprezentacji. Do określenia z której reprezentacji program ma korzystać można użyć opcji *-r*. Domyślną wartością jest najniższa dostępna reprezentacja (reprezentacje zazwyczaj numerowane są od 0). W przypdaku wyboru reprezentacji, która nie znajduje się w zbiorze dostępnych reprezentacji pliku Media Presentation Description program wyświetli stosowny komunikat i zakończy działanie.

– Interactive - tryb interaktywny pozwalający na wybór reprezentacji z której odtwarzacz ma korzystać w czasie rzeczywistym. Podawanie reprezentacji odbywa się za pomocą konsoli. W przypadku wyboru nieistniejącej reprezentacji przez użytkownika, aplikacja wyświetli stosowny komunikat i będzie nadal korzystać z poprzedniej reprezentacji.

– PID - tryb wykorzystujący algorytm oparty o kontroler PID opisany w podrozdziale 5.2. W tym trybie dobór reprezentacji i przełączanie jest wykonywane automatycznie w oparciu o przepustowość łącza. Dostępna przepustowość jest liczona na podstawie czasu pobierania poprzednich segmentów danych oraz ich wielkości. Przy wyborze tego mechanizmu adaptacji użytkownik powinien również podać spodziewaną przepustowość łącza (*-t*). Nie jest to wymagany parametr, ale pozwala na szybszą stabilizację działania algorytmu. Im lepiej użytkownik oszacuje początkową przepustowość tym lepiej dobierane będą reprezentacje w początkowej fazie działania aplikacji. Paramter *-t* nie posiada długofalowego wpływu na działanie aplikacji.

6.3. Testy działania aplikacji

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
```

B. Przykład kompilacji źródeł odtwarzacza

Na listingu B.1 przedstawiono komendę z której korzysta program Microsoft Visual Studio 2010 Professional w celu kompilacji źródeł odtwarzacza multimedialnego.

Listing B.1: Kompilacja źródeł odtwarzacza przez program MVS 2010.

```
...> cmake /I"C:\boost_1_55_0" /Zi /nologo /W3 /WX- /O2 /Oi /Oy- /GL
/D "WIN32"
/D "NDEBUG"
/D "_CONSOLE"
/D "_UNICODE"
/D "UNICODE"
/Gm- /EHsc /GS /Gy /fp:precise /Zc:wchar_t /Zc:forScope
/Fp"C:\...\libdash\intermediate\clientdash\ReleaseWin32\clientdash.pch"
/Fa"C:\...\libdash\intermediate\clientdash\ReleaseWin32\"
/Fo"C:\...\libdash\intermediate\clientdash\ReleaseWin32\"
/Fd"C:\...\libdash\intermediate\clientdash\ReleaseWin32\vc100.pdb"
/Gd /analyze- /errorReport:queue
```

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 Pro

le 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] Request For Comment 2326: Real Time Streaming Protocol

[11] Information technology — Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats

[12] Request For Comment 2581: TCP Congestion Control

[13] TCP/IP Illustrated, Volume 1, Second Edition Kevin R. Fall, W. Richard Stevens

[14] http://www.microsoft.com/silverlight/smoothstreaming

[15] http://www.adobe.com/pl/products/hds-dynamic-streaming.html

[16] https://developer.apple.com/streaming/

[17] http://traffic.comics.unina.it/software/ITG/

[18] http://dashif.org/mpeg-dash/

[19] http://www.networkworld.com/redesign08/subnets/cisco/031708-ch7-voip.html

[20] http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-2/gigabit_tcp.html

[21] http://www.ciscopress.com/articles/article.asp?p=352991&seqNum=8

[22] http://www.cisco.com/c/en/us/support/docs/collaboration-endpoints/spa901-1-line-ip-phone/108736-pqa-108736.html

[23] http://www.w3.org/2008/WebVideo/Fragments/wiki/UA_Server_RTSP_Communication