Quality of Service w sieciach LAN/WAN

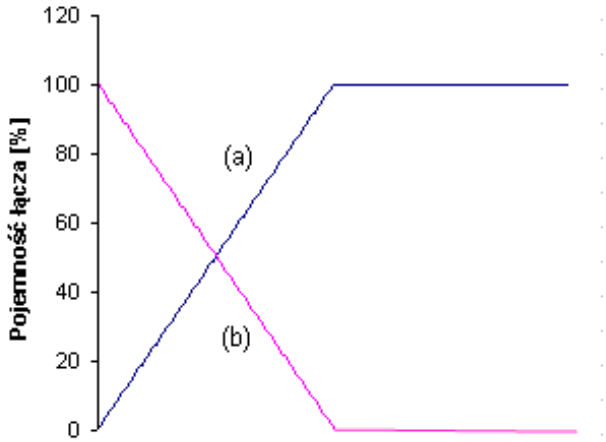
1. Metody zapewnienia Quality of Service w sieciach LAN/WAN

Klasycznym dziś rozwiązaniem wielu sieci jest połączenie sieci lokalnych z siecią rozległą Internet, wykorzystujące do komunikacji protokół IP za pośrednictwem urządzeń zwanych routerami. W sieci o takiej architekturze znajduje się określona ilość użytkowników, którzy współdzielą łącze z siecią rozległą, czyli więc rywalizują o dostęp do niej między sobą. Standardową zasadą dostępu do zasobów sieci rozległej, w sieciach lokalnych o tego typu architekturze, jest zasada „kto pierwszy, ten lepszy", czyli obsługa metodą FIFO (First In First Out). Zasada ta oznacza, że pierwszeństwo do zasobów sieci rozległej ma użytkownik, który pierwszy wyśle swoje żądanie. W okresie powstawania sieci Internet i jej początkowych rozwojów, dane przesyłane przez tą sieć były nieznacznych rozmiarów. Założenie takiej metody dostępu do sieci, pozwalało na maksymalizację czasu dostarczenia informacji. Z czasem ta sytuacja zaczęła się jednak zmieniać. Użytkownicy coraz częściej korzystają ze zdalnie uruchamianych aplikacji, żądają szybkiego dostępu do olbrzymich archiwów danych, przesyłają dźwięk i obraz w czasie rzeczywistym, co oznacza, że przesyłane dane są dużych rozmiarów. Łącze z siecią rozległą stało się więc „wąskim gardłem", które ma znacznie mniejsze możliwości przesyłu informacji niż zapotrzebowanie użytkowników współdzielących to łącze. Pobranie takich danych o dużych rozmiarach przez użytkownika poprzez łącze z siecią Internet wymaga pewnej ilości czasu, a co się z tym wiąże oraz zgodnie z metodą FIFO dostępu do tego łącza, powoduje to całkowite zajęcie tego łącza aż do momentu zakończenia pobrania tych danych. Przykładowo pobranie pliku o objętości 700 MB za pośrednictwem łącza o pojemności 1 [Mbit/s] zajmuje około 90 minut. Przy zastosowaniu metody FIFO pobranie takiego filmu przez jednego z użytkowników w sieci z pełną prędkością łącza, oznacza więc blokadę dostępu do tego łącza przez około 90 minut, aż do ukończenia jego ściągnięcia. W czasie tym żaden inny użytkownik nie może korzystać z zasobów sieci Internet. Należy przy tym zaznaczyć, że prędkość ściągania danych zależy jednak nie tylko od posiadanego łącza z siecią rozległą, ale od aktualnej pojemności każdego z poszczególnych łącz pośredniczących przy poborze danych. Oznacza to, że prędkość ściągania danych jest taka, jaką umożliwia jedno z łącz pośredniczących o najniższej aktualnie przepustowości.

W sieciach, w których znajduje się dość duża ilość użytkowników, przy pobieraniu przez nich danych o niedużej wielkości, ale za to przy dużej intensywności pobierania tych

danych, może także wystąpić sytuacja wypełnienia możliwości dostępnej przepustowości łącza z siecią Internet.

Zasada FIFO powoduje więc, co zostało uwidocznione na rys. 1, że im większa ilość danych będzie przepływać przez łącze z siecią rozległą (linia a), tym możliwość ilości pobrania nowych danych będzie proporcjonalnie mniejsza (linia b), aż do stanu, w którym przy 100% zajętości łącza, możliwości pobrania nowych danych nie ma.



Rys. 1. Wpływ metody FIFO na żądania poboru danych z sieci Internet do użytkowników w sieci lokalnej.

Źródło: Opracowanie własne.

Rywalilizacja użytkowników o przepustowość wpływa także na inne parametry jak straty pakietów i opóźnienie. Przesyłana ilość danych wypełniająca możliwości przepustowości łącza, powoduje dłuższą obsługę w buforach interfejsów sieciowych, co prowadzi do opóźnienia w dotarciu pakietów do odbiorcy, jak również może nastąpić całkowita utrata pakietu, kiedy bufor jest przepełniony. Szczególnie istotne staje się więc w sieciach o omawianej architekturze zapewnienie odpowiedniego podziału między użytkowników parametru jakim jest przepustowość.

Charakterystyczne dla sieci o omawianej architekturze jest to, że ruch między siecią lokalną a siecią rozległą, jest w przeważającej części ruchem unicastowym, przy czym w tej wymianie informacji występuje dość duża ilość tych strumieni unicastowych. Transmisja danych między tymi sieciami jest więc bardzo niejednorodna, występuje bardzo duża ilość różnorodnych strumieni danych od różnorodnych nadawców do różnorodnych odbiorców, dotyczącej dużej ilości różnorodnych usług przesyłanych między tymi sieciami. Usługi te można najczęściej w pewien sposób schierarchizować w randze ważności. Ważniejsze z reguły dla użytkowników na przykład jest szybsze ściągnięcie strony WWW niż szybsze

pobieranie danego pliku danych, których transmisja odbywa się w tym samym czasie. Istnieje więc także poza podziałem łącza między użytkowników, problem zapewnienia jakości transmisji dla danych usług, które są priorytetyzowane względem innych. Przy czym ta priorytetyzacja usług, polega na fakcie, że użytkownik jest skłonny mieć zapewnione lepsze parametry transmisji dla danej usługi, kosztem innych usług.

Popularne w dzisiejszej nauce modele klasy „End-To-End" (model IntServ, DiffServ, protokół MPLS) są mechanizmami „ponad protokołem IP", nie wchodzącymi w skład tego protokołu, mającymi za zadanie zapewnienie jakości danych parametrów na całej drodze od nadawcy do odbiorcy. Są one mechanizmami bardziej wysokopoziomowymi, mającymi rozwiązać problem klasyfikacji ruchu i następnie propagacji do innych węzłów informacji odnośnie konieczności rezerwacji danych zasobów dla tego ruchu.

Metody te zakładają wsparcie ich mechanizmów przez wszystkie, bądź przynajmniej przez większość węzłów w sieci dla ich skutecznego działania. Pozwalają na bardzo dobre osiągnięcie parametrów jakości dla danego priorytetyzowanego ruchu. Jednak sieć Internet na dzień obecny nie wspiera tychże rozwiązań, ich zasięg zostaje ograniczony do sieci lokalnych i ich działanie można odczuć tylko wewnątrz tychże sieci. Ponieważ jednak większość wymiany informacji odbywa się z siecią rozległą, najczęściej najdalszym elementem na którym można zapewniać jakość usług są obrzeża tej sieci lokalnej, czyli jej router brzegowy. W związku z tym metody klasy „End-To-End" nie mają możliwości większych zastosowań dla sieci o omawianej architekturze. Istnieje jednakże potrzeba zapewnienia jakości usług dla wcześniej przedstawianych problemów, czyli problemu podziału przepustowości łącza między użytkowników, oraz zapewnienie szczególnej jakości dla istotnych usług sieciowych, kosztem innych usług.

Najprostszym mechanizmem pozwalającym na sterowanie ruchem w sieciach LAN/WAN jest filtrowanie tego ruchu, czyli całkowite odrzucanie ruchu niepożądanego. Opiera się ono na zestawie statycznych reguł, które zezwalają bądź blokują dostęp do określonych usług. Zablokowanie pewnych usług użytkownikom (np. ściąganie danych za pośrednictwem sieci Peer-to-Peer) pozwolić może na zdecydowane obniżenie ilości transmitowanych danych, ograniczając ją do usług istotnych, jednakże nie eliminuje ono sytuacji zatorów sieci, zmniejszając tylko prawdopodobieństwo ich wystąpienia.

Do podstawowych wad filtrowania można zaliczyć:

- ograniczenie dostępnych usług. Filtrowanie jest mechanizmem, który pozwala poprawić jakość korzystania jednych usług, kosztem innych. W efekcie sieć

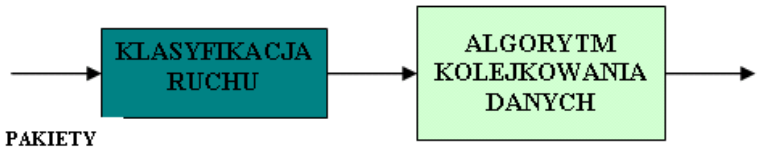
zostaje przeznaczona do z góry określonego zestawu usług, z których za jej pośrednictwem można skorzystać.

- niewykorzystanie przepustowości łącza w sytuacji kiedy ruch nieodrzucany przez filtrację nie wypełnia całej dostępnej przepustowości. W rezultacie pasmo, które może być potencjalnie dostępne pozostaje niewykorzystane.
- możliwość zdominowania łącza jednego rodzaju ruchem który jest akceptowany. Filtrowanie nie jest w stanie zapewnić, że pakiety związane z ruchem, który został przepuszczony i tak nie zdominują łącza i nie zajmą całej dostępnej przepustowości łącza.

Przedstawione powyżej wady powodują ze mechanizm filtracji danych jest niedoskonały i nie nadaje się dostatecznie do zapewnienia jakości w sieciach LAN/WAN. Może być stosowany jedynie jako rozwiązanie tymczasowe.

Zdecydowanie lepszym mechanizmem zapewnienia jakości w sieciach o omawianej architekturze jest kształtowanie ruchu z wykorzystaniem algorytmów kolejkowania danych.

Metody klasy „End-To-End" jak zostało przedstawione, zajmują się sposobem klasyfikacji ruchu i propagacją rezerwacji zasobów między węzłami sieci, nie zajmują się jednak samymi niskopoziomowymi mechanizmami, które miałyby za zadanie osiągnięcie żądanych parametrów jakości. Do rozwiązania tychże problemów stosuje się różnorodne algorytmy zajmujące się odpowiednim zarządzaniem buforem, w którym przebywają pakiety IP, zwane algorytmami kolejkowania danych. Kształtowanie ruchu w sieciach LAN/WAN, jest więc uproszczonym modelem metod klasy „End-To-End". Metody te nie zakładają możliwości propagacji rezerwacji zasobów do innych węzłów poza siecią lokalną, oraz zakładają klasyfikację ruchu na podstawie danych zawartych w nagłówkach poszczególnych protokołów. Taki uproszczony model dotyczący routerów brzegowych został przedstawiony na Rys. 2.

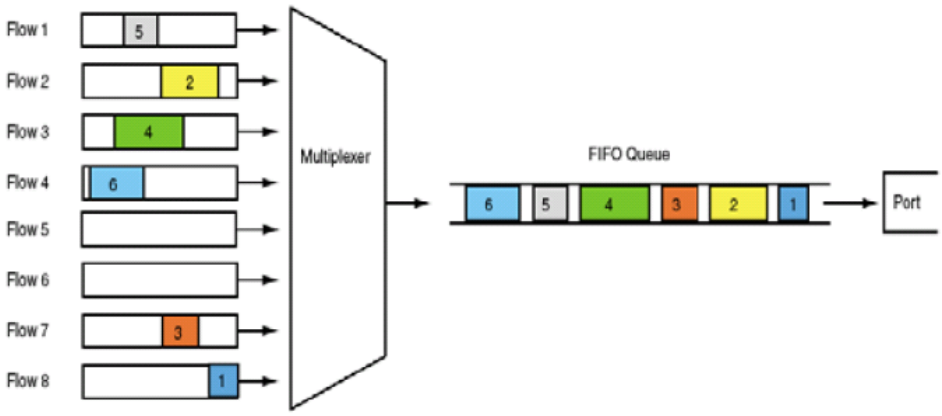


Rys. 2. Model kształtowania ruchu dla routerów brzegowych w sieciach LAN/WAN.
Źródło: Opracowanie własne

2. Algorytmy kolejkowania danych

Dane wpływające do urządzenia sieciowego umieszczane są w znajdującym się w nim *buforze*, czyli swoistej „poczekalni", przed dalszym ich wysłaniem lub przetworzeniem. W przypadku węzłów sieciowych ich zadaniem jest wygładzanie prędkości napływających danych, w warunkach różnych przepustowości łącz sieciowych sieci, między którymi ten węzeł pośredniczy. Algorytmy kolejkowania danych w tego typu urządzeniach sieciowych zajmują się sposobem obsługi danych przechodzących przez bufor. Dzielą one bufor na pewną ilość kolejek, które są następnie obsługiwane według określonego algorytmu postępowania.

Standardowym algorytmem kolejkowania implementowanym w urządzeniach sieciowych jest algorytm „First In First Out", czyli „pierwszy wszedł, pierwszy wychodzi", co jest de facto brakiem jakiegokolwiek postępowania z napływającymi danymi (Rys. 3).



Rys. 3. Schemat działania algorytmu kolejkowania „First In First Out".
Źródło: [1]

Jest to algorytm, w którym występuje tylko jedna kolejka w buforze. Pakiety, które pierwsze wchodzą do bufora, pierwsze go opuszczają, w kolejności takiej, w jakiej do niego wpłynęły. Efekt działania tego algorytmu na przepływ danych w sieciach LAN/WAN został przedstawiony w poprzednim rozdziale. Jest on związany z ograniczoną objętością bufora, który w momencie zapełnienia odrzuca przychodzące pakiety, do momentu, aż nie zostanie w nim zwolnione jakiekolwiek miejsce. Jak zauważono w [2, 3], mechanizm ten powodować może także powstawanie dużych opóźnień, co ma bardzo niekorzystny wpływ na ruch aplikacji czasu rzeczywistego, które rośnie wraz ze stopniem zapełnienia bufora przez napływające pakiety. Algorytm ten traktuje wszystkie pakiety równorzędnie i nie daje możliwości ustalenia żadnych specjalnych warunków dla danego wyodrębnionego ruchu.

Mechanizm ten ma też jednak pewne zalety, literatura [1, 3] podaje, że jest on bardzo prostą techniką, nie wymagającą specjalistycznej wiedzy i zużywającą najmniejsze zasoby sprzętowe. Opóźnienie powstające w kolejce jest tu także bardzo przewidywalne, gdyż wynika ono ze stopnia jej zapełnienia.

Zgodnie z [8], można podsumować, że algorytm ten w wyraźny sposób nie dostarcza żadnych mechanizmów kształtowaniem ruchu, przychodzące pakiety przekazuje dalej w sposób taki, w jaki do niego dotarły. Z powyższego faktu wynikła konieczność opracowania innych algorytmów, pozwalających na określone sterowanie przepływem danych.

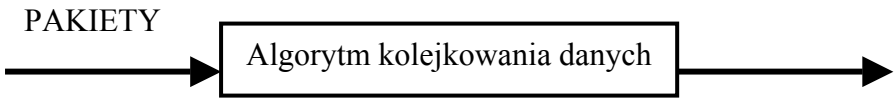
Wśród algorytmów kolejkowania ruchu sieciowego opracowanym jako rozwinięcie algorytmu „First In First Out", można wyróżnić:

a) algorytmy szeregowania pakietów – które zajmują się zmianą kolejności pakietów w buforze, mające na celu osiągnięcie określonej jakości obsługi,
b) algorytmy przeciwdziałania przeciążeniom – mające przeciwdziałać zapełnieniu się bufora, skutkującym zatorem łącza.

Algorytmy te z kolei mogą być algorytmami [5]:

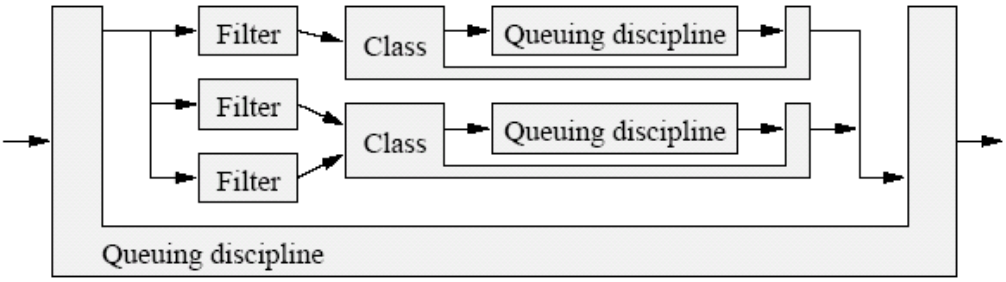
a) bezklasowymi,
b) opartymi o klasy.

Algorytmy bezklasowe charakteryzują się tym, że działają samodzielnie. Modelowo w ramach obsługi w obrębie jednego interfejsu sieciowego nie można połączyć ich z innymi algorytmami kolejkowania (Rys. 4). Pakiet po obsłużeniu przez dany algorytm w buforze, opuszcza interfejs sieciowy i zostaje przekazany dalej.



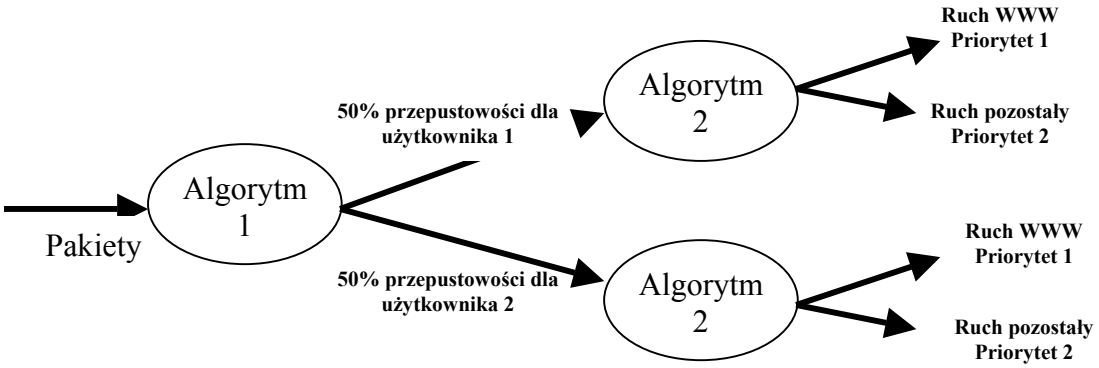
Rys. 4. Budowa algorytmu kolejkowania danych bezklasowego.
Źródło: [5]

Algorytmy klasowe zaproponowane w [9, 10] posiadają natomiast możliwość połączenia ich z innymi algorytmami w obrębie jednego interfejsu sieciowego, także opartymi o klasy, bądź bezklasowymi (Rys. 5). Ruch wchodzący do takiego algorytmu zostaje skierowany do odpowiedniej kolejki, po obsłudze zgodnie z jego założeniem zostaje następnie przekierowany do kolejnego algorytmu obudowanego jako klasa, gdzie także zostają utworzone kolejki według założenia tego następnego algorytmu i zgodnie z jego mechanizmami obsłużony.



Rys. 5. Budowa algorytmu kolejkowania danych opartego o klasy.
Źródło: [5]

Dzięki takiemu mechanizmowi można zbudować zintegrowany algorytm, który jest w stanie zapewnić daną pożądaną obsługę ruchu sieciowego na kilka różnych sposobów naraz. *Przykładowo:* potrzebą użycia takich algorytmów może być sytuacja, kiedy należy najpierw przepustowość podzielić według danych założeń między użytkowników (np. każdemu dokładnie po równo), czym zajmuje się jeden algorytm, następnie w ramach każdego użytkownika kolejny algorytm zajmuje się podziałem jego ruchu nadając pierwszeństwo dostarczenia danych według pewnych ustalonych priorytetów (Rys. 6).



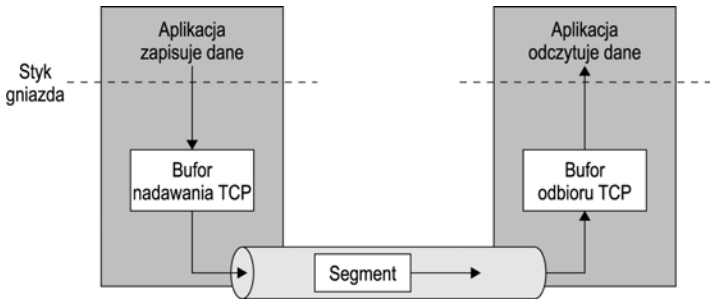
Rys. 6. Przykład zastosowania algorytmu kolejkowania opartego o klasy.
Źródło: Opracowanie własne.

2.1. Algorytmy szeregowania pakietów

Najczęstszym celem stosowania algorytmów kolejkowania danych w sieciach LAN/WAN jest odpowiednia obsługa użytkowników, rozumiana jako podział zasobów łącza z siecią rozległą według ustalonych zasad między tych użytkowników. Algorytmy kolejkowania zajmujące się takim podziałem, nazywa się algorytmami szeregowania pakietów. Głównym zadaniem tych mechanizmów jest w praktyce podział parametru, jakim jest przepustowość w sytuacjach zatorów sieci. Przy czym ten podział zasobów rozpatrywany

jest w warunkach skrajnych, jakim są przeciążenia sieci. Jeśli przeciążenie sieci nie występuje, podział przepustowości nie jest tak istotny, gdyż pozostaje wolna część łącza i nie istnieje wtedy konieczność jego podziału. Konieczność ta powstaje w momencie, kiedy wykorzystywana jest całkowita przepustowość łącza sieciowego, przez co nie mogą zostać obsłużone kolejne żądania.

Algorytmy te wykorzystują jedną z właściwości protokołu TCP, który pozwala na dostosowanie prędkości przesyłu danych od nadawcy do odbiorcy, do warunków panujących w sieci. Zadaniem tego mechanizmu jest wysyłanie pakietów w zależności od rozmiaru bufora u odbiorcy, co ma zagwarantować, iż bufor po stronie odbiorcy nie zostanie przepełniony i pakiety nie będą tracone (Rys 7).



Rys. 7. Bufory nadawania i odbioru.
Źródło: [28].

Aby dane były dostarczane w sposób wiarygodny, klient musi otrzymać od odbiorcy potwierdzenie każdego wysłanego przez siebie segmentu danych. Ponieważ klient musi czekać na potwierdzenie od serwera przed nadaniem kolejnego segmentu, proces ten może prowadzić do wolnego przesyłu danych oraz niepełnego wykorzystania zasobów sieciowych. Aby zminimalizować czas jałowy sieci i zapewnić wydajny i wiarygodny przesył danych, TCP wykorzystuje ideę *okien przesuwnych* (*sliding window*). W oknie przesuwnym przed oczekiwaniem na potwierdzenie nadawanych jest kilka segmentów. Liczba segmentów, jaką nadawca może wysłać w określonym połączeniu, zanim otrzyma potwierdzenie od odbiorcy wskazujące, iż ten otrzymał przynajmniej jeden segment danych, nosi nazwę *okna nadawania (send window)*. Okno to ma stały rozmiar, a wszystkie segmenty mieszczące się wewnątrz okna możemy nadać nie czekając na potwierdzenie. Gdy nadawca otrzyma potwierdzenie pierwszego segmentu w oknie nadawania, okno przesuwa się i kolejny segment zostaje wysłany. Jeśli nadawca otrzyma potwierdzenie dla kilku segmentów, na przykład trzech, okno odpowiednio przesuwa się i zostają wysłane trzy segmenty. Jednakże liczba segmentów, które

można wysłać, jest zależna od okna odbioru. Po stronie serwera proces aplikacji odczytuje dane z bufora z określoną prędkością, wobec czego rozmiar okna odbioru jest zależny od szybkości, z jaką dane są odczytywane. Gdy serwer wysyła potwierdzenie segmentów danych do klienta, razem z potwierdzeniem ogłaszany jest rozmiar okna. Klient wysyła następnie segmenty z okna nadawania tak, by nie przepełnić okna odbioru po stronie serwera.

Przeciążenia powodują opóźnienia w dostarczaniu danych. Sytuacja staje się jeszcze gorsza, gdy protokół TCP stosuje odliczanie dopuszczalnego czasu i ponowne transmisje w przypadku utraconych segmentów. Aby uniknąć przeciążenia, klient musi „pamiętać" rozmiar okna odbioru. Ponadto rozmiar okna nadawania jest zmniejszany w zależności od poziomu przeciążenia. Takie zmniejszone okno nadawania nosi nazwę *limitu okna podczas przeciążenia* lub *okna przeciążenia*. Okno przeciążenia jest mechanizmem kontroli przeciążeń, wymuszanym przez nadawcę i opartym na szacunku przeciążenia sieci według nadawcy. Z drugiej strony okno odbioru jest mechanizmem kontrolnym stosowanym przez odbiorcę i opartym na ocenie dostępnej objętości wolnego miejsca w buforze. Dopuszczalny rozmiar okna jest zawsze mniejszą z dwóch wartości: okna odbioru ogłoszonego przez odbiorcę i okna przeciążenia.

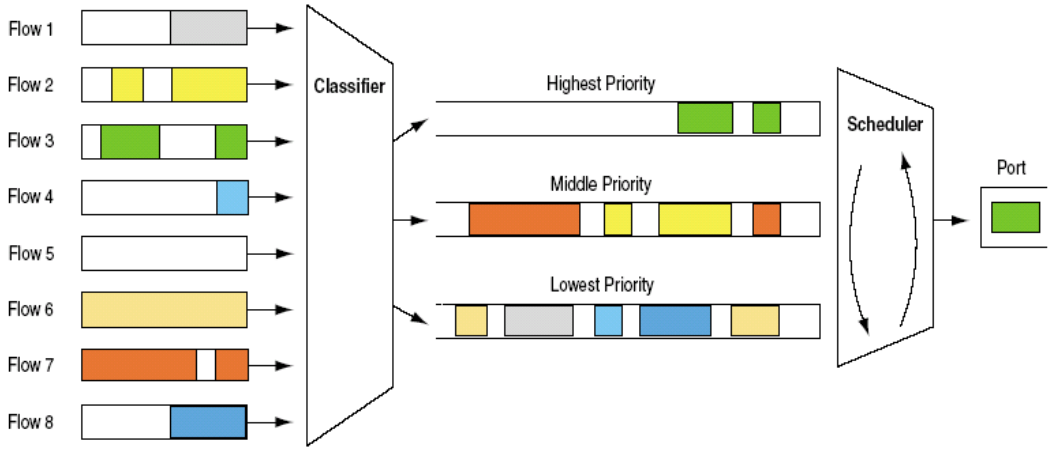
W stanie, gdy nie występują przeciążenia, rozmiar okna przeciążenia jest równy rozmiarowi okna odbiorcy. Jednakże w razie zatorów rozmiar okna jest zmniejszany. Do oszacowania rozmiaru okna przeciążenia protokół TCP stosuje następującą strategię:

1. Redukcja okna przeciążenia o połowę po każdej utracie segmentu.
2. Jeśli straty dalej występują, rozmiar okna zmniejszany jest wykładniczo.
3. W ostateczności transmisja zostaje ograniczona do pojedynczych segmentów, a dopuszczalne czasy oczekiwania przed retransmisją są nadal podwajane. [28]

Dzięki opisanemu mechanizmowi możliwe jest sterowanie prędkością wysyłanych danych od nadawcy, i tą właściwość wykorzystują algorytmy szeregowania pakietów. Po utworzeniu odrębnej kolejki o określonej objętości dla danego rodzaju ruchu sieciowego, jeśli ta kolejka zostanie zapełniona, następuje odrzucanie kolejnych napływających do niej pakietów, a przez to zmniejszenie prędkości ich wysyłania przez nadawcę.

Najprostszym algorytmem szeregowania pakietów, będącym niewielkim rozwinięciem algorytmu „First In First Out" pozwalającym na bardzo ograniczony podział łącza, jest algorytm *Priority Queuing*. Umożliwia on podział ruchu na trzy lub cztery grupy kierowane do trzech lub czterech różnych kolejek ustawionych hierarchicznie, priorytetyzując w ten sposób jeden ruch względem innego (Rys. 8). Każda ta kolejka jest obsługiwana zgodnie z metodą „First In First Out", jakkolwiek pakiet nie może opuścić jej dopóki kolejka

hierarchicznie wyższa nie jest opróżniona. Poza możliwością dość ograniczonej priorytetyzacji algorytm ten posiada wszystkie wady i zalety algorytmu „First In First Out".



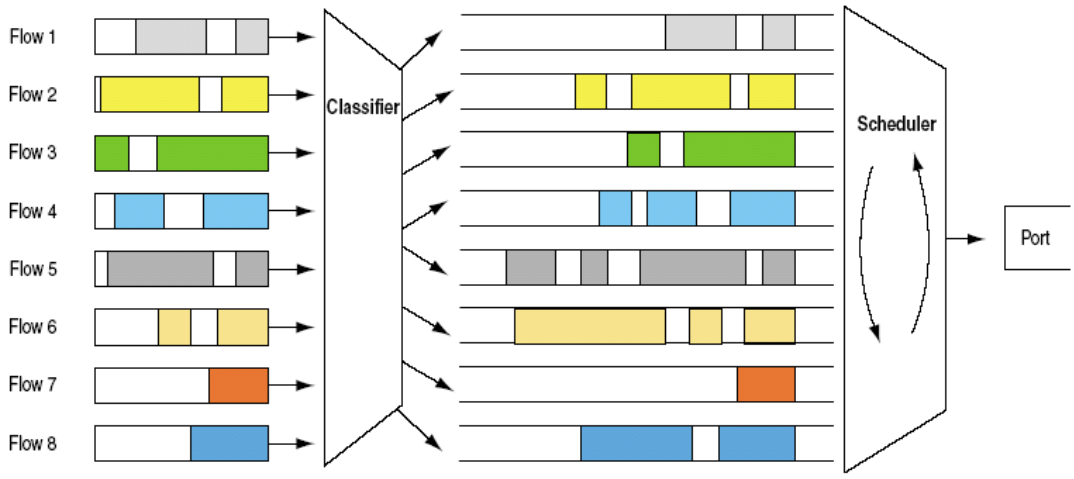
Rys. 8. Schemat działania algorytmu kolejkowania Priority Queuing.
Źródło: [1]

Wśród pozostałych algorytmów szeregowania pakietów, można wyróżnić:

a) algorytmy sprawiedliwego kolejkowania,
b) algorytmy ograniczania przepustowości,
c) algorytmy hierarchicznego podziału łącza.

2.1.1. Algorytmy sprawiedliwego kolejkowania

Dla rozwiązania powyżej przedstawionych problemów algorytmu „First In First Out", Nagle w [4] zaproponował ideę algorytmów „*sprawiedliwego kolejkowania*" (*ang. Fair Queuing*), w których utrzymywane są oddzielne kolejki dla pakietów z określonych źródeł. Kolejki te zostają następnie obsługiwane metodą *„Round-Robin"*, która zakłada cykliczne opróżnianie kolejek po jednym pakiecie (Rys. 9).



Rys. 9. Schemat działania algorytmów sprawiedliwego kolejkowania.
Źródło: [1]

Powstało bardzo wiele algorytmów sprawiedliwego kolejkowania, których główną cechą jest „sprawiedliwy" podział transmisji danych. Przy czym ta *sprawiedliwość* w różnych algorytmach *może być różnie rozumiana*, może to być:

a) *równy podział przepustowości między nadawców* (klasyfikowanie do kolejki pakietów z tym samym adresem źródłowym),
b) *równy podział przepustowości między odbiorców* (klasyfikowanie do kolejki pakietów z tym samym adresem docelowym),
c) *równy podział przepustowości między strumienie danych* (klasyfikowanie do kolejki pakietów z tym samym adresem źródłowym i docelowym, protokołem i numerem portu źródłowego i docelowego).

Najpopularniejsze algorytmy kolejkowania danych z rodziny „sprawiedliwych", to poza samym Fair Queuing:

a) SFQ (*ang. Stochastic Fairness Queuing*) [6],
b) WFQ (*ang. Weighted Fair Queuing*) [7],
c) WRR (*ang. Weighted Round Robin*) [11],
d) DRR (*ang. Deficit Round Robin*) [12],

Powstałe w tej klasyfikacji kolejki są następnie obsługiwane w sposób cykliczny za pomocą mechanizmu „Round-Robin". Przy czym każda kolejka ma swoją określoną pojemność, po której przekroczeniu, kolejne nadchodzące pakiety zostaną odrzucone, do momentu zwolnienia w niej miejsca. Niektóre algorytmy kolejkowania pozwalają ponadto na przypisywanie określonym kolejkom wag liczbowych, co pozwala na to, że mimo iż wszystkie kolejki są obsługiwane cyklicznie, niektóre z nich, te o wyższej wadze, mogą być

obsługiwane częściej. Przykładem takich algorytmów jest algorytm WFQ (*ang. Weighted Fair Queuing*) [7].

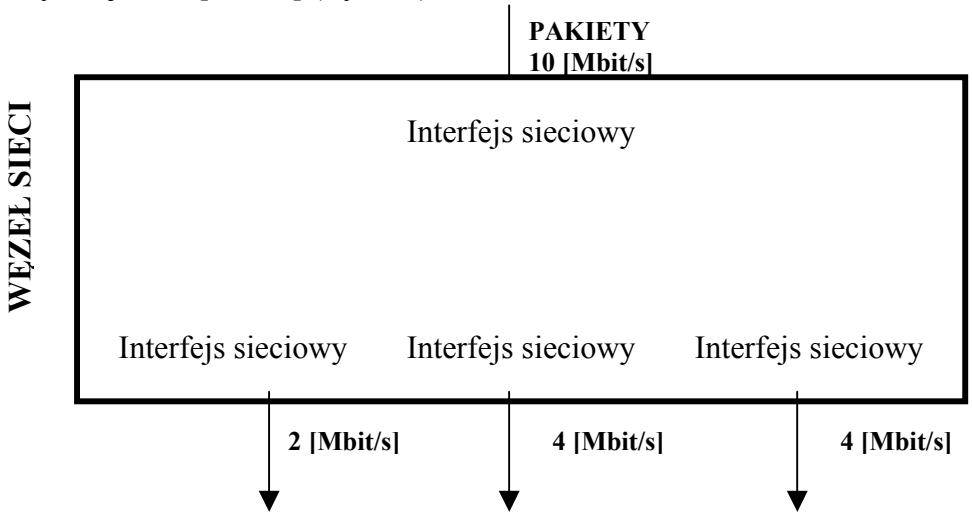
Zaletą algorytmów sprawiedliwego kolejkowania w stosunku do algorytmu „First In First Out", jest możliwość równego podziału łącza, bądź między nadawców, bądź odbiorców, bądź sesje danych. Taki podział zapobiega możliwościom powstania zatorów łącza.

Wadą tych rozwiązań jest brak możliwości ustalania określonej przepustowości dla danego ruchu. Z wyjątkiem algorytmów ważonych, nie ma możliwości priorytetyzacji danego strumienia danych. Również, jeśli któraś z powstałych kolejek będzie posiadała większy stopień zapełnienia, czas ich obsługi zwiększy się powodując wzrost opóźnienia.

2.1.2. Algorytmy ograniczania przepustowości

Algorytmy sprawiedliwego kolejkowania pozwalają na równy podział przepustowości, przy czym może on być pomiędzy nadawcami, odbiorcami bądź sesjami danych. Istnieje jednak czasem potrzeba rozdziału przepustowości pomiędzy podsieci na różne, a zarazem bardzo konkretne wartości, bądź też spowolnienia transmisji danych wychodzących z węzła w innych celach. Takim podziałem zajmują się algorytmy ograniczania przepustowości. Istotą ich działania jest ograniczenie przepływu pakietów tak, aby opuszczały one bufor interfejsu sieciowego nie szybciej niż dana ustalona wartość.

Przykładowo: potrzebą użycia tego typu algorytmów, może być sytuacja, gdy istnieje konieczność rozdzielenia przepustowości łącza o wielkości 10 [Mbit/s] pomiędzy trzy podsieci, gdzie dwie podsieci mogą transmitować dane nie szybciej niż 4 [Mbit/s], a jedna podsieć nie szybciej niż 2 [Mbit/s] (Rys. 10).

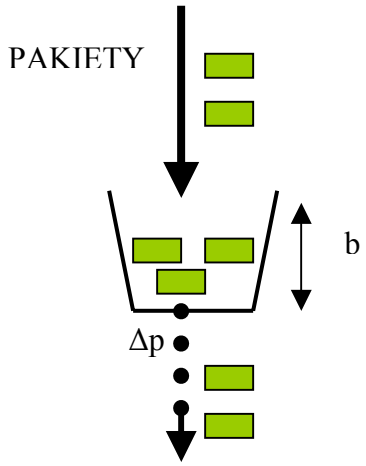


Rys. 10. Schemat działania algorytmów ograniczania przepustowości.

Źródło: Opracowanie własne.

Dla tego sposobu kształtowania danych zaproponowano w [18] model *„cieknącego wiadra”* (*ang. Leaky Bucket*), utożsamiany też często niesłusznie z modelem *„wiadra z żetonami”* (*ang. Token Bucket*).

W modelu tym (Rys. 11) nadchodzące pakiety są buforowane w „wiadrze” o określonej głębokości *b*. Jeśli wypełnią one dopuszczalny rozmiar wiadra *b*, kolejne nadchodzące pakiety zostają odrzucane, do momentu zwolnienia miejsca w wiadrze.

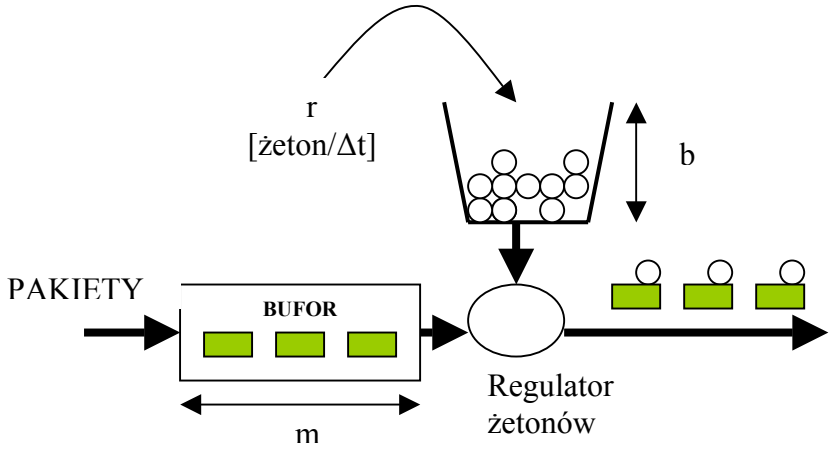


Rys. 11. Model cieknącego wiadra.

Źródło: Opracowanie własne.

Pakiety umieszczone w wiadrze opuszczają je następnie ze stałą prędkością *Δp*, dzięki czemu zostaje osiągnięty efekt, w którym pakiety zostają przesyłane dalej do sieci z prędkością nie większą niż pozwala na to ustalona wartość *Δp*. Jednak jeśli nadchodzące pakiety są różnych rozmiarów, prędkość transmisji będzie ściśle uzależniona od rozmiaru tych pakietów. Mechanizm ten można przyrównać do wiadra z wodą, z którego przez dziurkę w spodzie równomiernie wyciekają krople wody.

Bardziej elastycznym mechanizmem jest model „*wiadra z żetonami*” (*ang. Token Bucket*) (Rys. 12), wykorzystujący do kształtowania ruchu żetony.



Rys. 12. Model wiadra z żetonami.

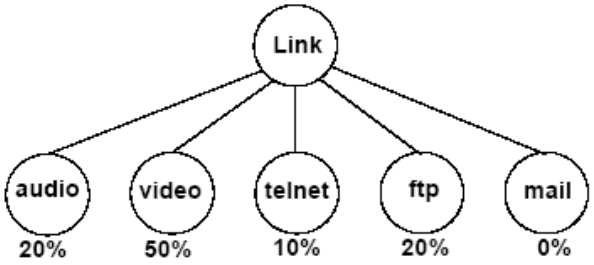
Źródło: Opracowanie własne, na podstawie [14, 15, 16, 17].

Przepływ żetonów jest definiowany przez dwie wartości: napływ żetonów *r* (ich produkcja) w stałym odstępie czasu *Δt*, oraz głębokość wiadra *b*, czyli ilość żetonów, jakie może ono pomieścić. Jeśli „wiadro" jest pełne, nadchodzące żetony zostają odrzucane. Każdy żeton umożliwia przesłanie określonej ilości bitów danych lub jeden pakiet danych. Ażeby pakiety zostały przesłane dalej, muszą one otrzymać jeden żeton na jeden pakiet, bądź ilość żetonów na jeden pakiet, adekwatną do rozmiarów informacji w nim zawartych. Mechanizm ten pozwala więc, na dokładne ustalenie ograniczenia prędkości transmisji w odniesieniu do różnorodnej wielkości napływających pakietów, na co nie pozwalał model „cieknącego wiadra". Jeżeli w wiadrze brakuje żetonów, pakiet jest buforowany do momentu pojawienia się odpowiadającej mu ilości żetonów, przy czym bufor ma swoją objętość *m* po przekroczeniu której, kolejne nadchodzące pakiety są odrzucane. Jeśli w wiadrze występuje nadwyżka żetonów, pakiety mogą przez chwilę być wysyłane nieco szybciej, z prędkością $b + r/\Delta t$ [19], niż wynika to z ustanowionego ograniczenia, wykorzystując tą chwilową nadwyżkę żetonów.

Jak zauważono w [15], w implementacjach często spotyka się oba te modele połączone. Najczęściej stosuje się model, w którym dane z „wiadra żetonów" trafiają do „cieknącego wiadra", zwanym „*wiadrem żetonów z kontrolą prędkości cieknącego wiadra*" (*ang. Token Bucket with Leaky Bucket rate control*) [13]. Mechanizm ten ma za zadanie wygładzić chwilową nadwyżkę wysyłanych pakietów z „wiadra żetonów". Przykładem algorytmu kolejkowania stosującego ten mechanizm jest algorytm TBF (*ang. Tocken Bucket Filter*) opisany w [20].

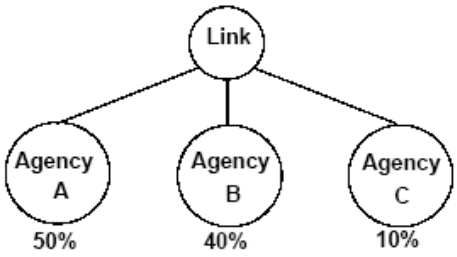
2.1.3. Algorytmy hierarchicznego podziału łącza

Algorytmy hierarchicznego podziału łącza zostały zaproponowane w [9] i miały na celu połączenie zalet wszystkich dotąd omawianych metod. Pozwalają one na dokładnie określony podział parametrów jakości, przede wszystkim przepustowości łącza, pomiędzy zdefiniowane klasy ruchu. Klasa ruchu wg. [21] jest „pewnym agregatem ruchu grupowanym razem dla celów jakichś sposobów zarządzania nim". Klasy te mogą być wyodrębniane na podstawie różnych parametrów znajdujących się w nagłówkach protokołów, np. adres IP nadawcy lub odbiorcy, port nadawcy lub odbiorcy, itd. *Przykładowo:* może to być podział przepustowości na klasy usług, w którym ruch audio otrzymuje 20% prędkości łącza, ruch video 50%, ruch telnet 10%, ruch ftp 20%, a ruch usługi e-mail 0% (Rys. 13).



Rys. 13. Przykładowy podział łącza pomiędzy klasy usług.
Źródło: [9].

Klasy usług mogą wyznaczać nie tylko same konkretne usługi, ale także może to być podział zasobów pomiędzy podsieci czy konkretnych odbiorców lub nadawców. *Przykładowo:* może to być podział prędkości łącza na trzy podsieci, w którym podsieć „Agency A" otrzymuje 50% przepustowości, podsieć „Agency B" 40%, oraz podsieć „Agency C" 10% (Rys. 12).

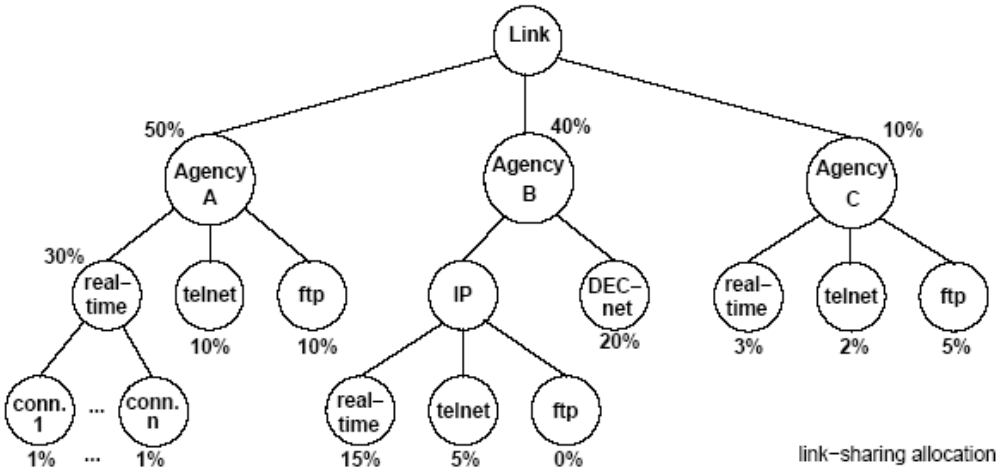


Rys. 14. Przykładowy podział łącza pomiędzy odbiorców.
Źródło: [9].

Algorytmy hierarchicznego podziału łącza umożliwiają nie tylko podział bądź tylko na usługi, bądź tylko odbiorców, czy nadawców. Największą ich zaletą jest fakt, że mogą

pozwolić na podział najpierw według odbiorców, czy nadawców, a następnie w ramach ich wyodrębnionego ruchu jeszcze podział na konkretne usługi. Może to być także podział najpierw na całe podsieci, potem w ramach tych podsieci na konkretnych nadawców, lub odbiorców i w ich ramach jeszcze na ruch danych usług.

Przykładowo: algorytmy te mogą rozwiązać kwestie następującego podziału nadającego określoną gwarancję przepustowości. Całą przepustowość łącza należy najpierw podzielić na trzy podsieci, w stosunku podsieć „Agency A" 50%, „Agency B" 40%, a następnie „Agency B" 10%. W ramach podsieci „Agency A" Ruch musi zostać rozłożony w stosunku z przydzielonych 50%, dla ruchu usług czasu rzeczywistego (ang. real-time) 30%, dla ruchu usługi telnet 10%, oraz dla ruchu usługi ftp 10%. Przydzielone 40% dla podsieci „Agency B" należy rozdzielić na kolejne dwie podsieci „IP" oraz „DECnet" po 20% dla każdej, a następnie z 20% dla podsieci „IP", 15% dla ruchu usług czasu rzeczywistego, 5% dla ruchu usługi telnet, oraz 0% dla ruchu usługi e-mail. Dla podsieci „Agency C" z przydzielonych 10% z kolei, należy rozłożyć dla ruchu usług czasu rzeczywistego 3%, dla ruchu usługi telnet 2%, oraz dla ruchu usługi ftp 5% (Rys. 13).



Rys. 15. Struktura hierarchicznego podziału łącza.
Źródło: [9].

Struktura hierarchicznego podziału łącza ma charakter struktury drzewa. Na samej górze tej struktury znajduje się *„pień"* (*ang. root*), który reprezentuje całkowitą przepustowość łącza. W drzewie tym wyróżnia się następnie *„klasy rodzice"* (*ang. parent class*), oraz *„klasy dzieci"* (*ang. child class*). Związane jest to z faktem, że algorytmy hierarchicznego podziału łącza są algorytmami opartymi o klasy, który to mechanizm został

wyjaśniony na początku rozdziału. Klasa rodzic jest nadrzędna w stosunku do klas dzieci. Zawiera ona w sobie zgodnie z mechanizmem klasowych algorytmów kolejkowania danych, kolejne algorytmy kolejkowania, którymi są klasy dzieci. Klasa dziecko, może być więc z kolei klasą rodzicem dla następnych zawartych w sobie algorytmów, jedynie „pień" nie posiada swojej klasy rodzica. W algorytmach hierarchicznego podziału łącza następuje podział przepustowości klasy pnia na klasy rodzice, które następnie są dzielone na klasy dzieci, przy czym przepustowość tego podziału musi być sumarycznie nie większa niż posiada klasa rodzic.

Jeśli jedna klasa nie wykorzystuje swojej przydzielonej przepustowości, w algorytmach hierarchicznego podziału łącza istnieje możliwość „pożyczenia" tych niewykorzystywanych zasobów innym klasom. W związku z tym mechanizmem, można wyróżnić *statyczny oraz dynamiczny podział łącza*. Statyczny podział nie uwzględnia możliwości oddania niewykorzystywanych zasobów innym klasom, natomiast dynamiczny podział taką sytuację umożliwia.

Do ostatniej klasy dziecka może być także podpięty jeszcze całkowicie inny algorytm kolejkowania, na przykład algorytm Fair Queuing, który zapewni równy podział pomiędzy sesje danych w ramach tej klasy dziecka. Standardowo do ostatniej klasy dziecka podpięty jest algorytm „First In First Out", jeśli nie zostanie w jego miejsce umieszczony inny.

Ze względu na największe możliwości z punktu widzenia obsługi użytkowników, algorytmy hierarchicznego podziału łącza są obecnie najczęściej stosowanymi podstawowymi algorytmami kolejkowania danych w sieciach LAN/WAN. Najpopularniejszymi algorytmami są tutaj:

- CBQ (ang. Class Based Queuing) [9],
- HTB (ang. Hierarchical Token Bucket) [22],
- HFSC (ang. Hierarchical Fair Service Curve) [23],
- HPFQ (ang. Hierarchical Packet Fair Queuing) [24].

2.2. Algorytmy zapobiegania zatorom

Odrębną grupą algorytmów kolejkowania danych są algorytmy zapobiegania zatorom, których przedstawicielem jest algorytm „*losowego wczesnego wykrywania*" (*ang. Random Early Detection*) zaproponowany w [25]. Pozostałe istniejące algorytmy zapobiegania zatorom stanowią modyfikację tego mechanizmu.

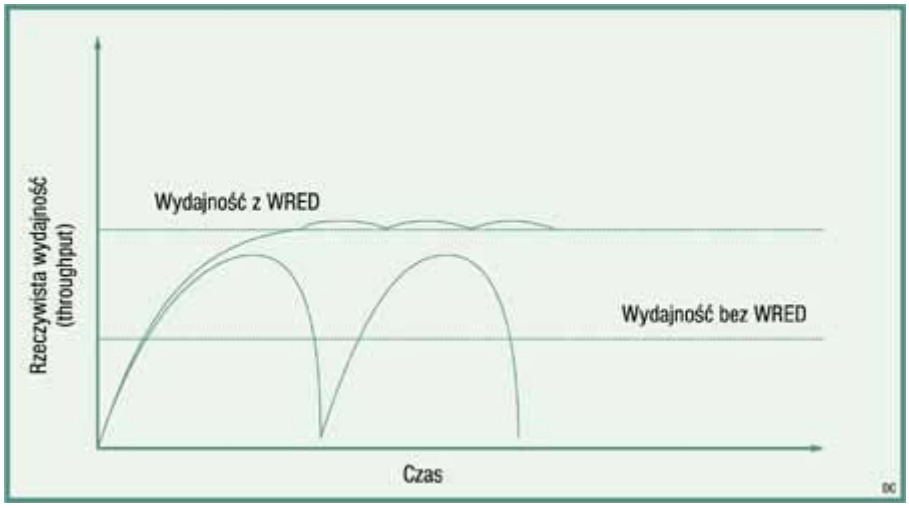
Jak zostało wcześniej opisane, algorytmy szeregowania pakietów są rozpatrywane z punktu widzenia podziału zasobów w sytuacjach przeciążenia sieci. Nadchodzące pakiety są

w tych mechanizmach buforowane w powstałych kolejkach. Jeśli jednak nadmiar napływających danych jest długotrwały, pojemność kolejki jest niewystarczająca do zachowania wszystkich tych przychodzących pakietów, zatem nadmiarowa część musi zostać odrzucona. W przypadku sesji TCP spowoduje to ponowną retransmisję tychże pakietów, co może jeszcze wzmocnić zjawisko przeciążenia.

W porównaniu do algorytmów szeregowania pakietów, które zajmują się kwestią jak się zachować w sytuacjach przeciążenia sieci, algorytmy zapobiegania zatorom zajmują się kwestią jak się zachować, aby do tego przeciążenia nie dopuścić. Efekt ten jest osiągany poprzez losowe odrzucanie pakietów, mające na celu spowolnić transmisję danych wybranych sesji TCP. Takie spowolnienie transmisji jest możliwe tylko w przypadku ruchu TCP, który posiada zaimplementowane mechanizmy dostosowania prędkości przesyłu danych do warunków sieci.

Zadaniem algorytmów zapobiegania zatorom jest więc takie sterowanie buforem, aby nie mogła nastąpić sytuacja przeciążenia łącza, czyli sytuacja, gdy kolejny nadchodzący pakiet do bufor zostaje odrzucony ze względu na jego przepełnienie. Efekt ten jest osiągany poprzez ciągłe obliczanie średniej długości kolejki i porównywanie jej z dwoma progami: minimalnym i maksymalnym. Jeśli średnia wielkość kolejki znajduje się poniżej minimalnego progu wówczas ani jeden przybywający pakiet nie będzie odrzucony. Jeśli średnia wielkość kolejki znajduje się powyżej progu maksymalnego, wtedy każdy nowo przybyły pakiet zostaje odrzucony. Jeśli średnia znajduje się pomiędzy wartościami tych progów, wówczas pakiety są odrzucane bazując na obliczeniach prawdopodobieństwa dokonywanych na podstawie średniej wielkości kolejki. Czym bardziej więc wielkość kolejki zbliża się do wartości maksymalnego progu, tym więcej pakietów jest odrzucanych. Taki mechanizm powoduje, że połączenia zużywające aktualnie więcej przepustowości są bardziej narażone na odrzucanie ich pakietów.

Algorytmy zapobiegania zatorom są dużo bardziej przydatne w węzłach sieci szkieletowych, gdzie występuję przepływ danych o bardzo dużych przepustowościach. Utworzony bufor dla tak dużego ruchu, musiałby być bardzo dużych wielkości, aby nie powodować zbyt dużego odrzucania pakietów, a to z kolei powodowałoby duże zwiększenie opóźnienia. Efekt retransmisji danych ma także dużo większy wpływ przy tak dużych prędkościach przesyłu danych. Dodatkowo algorytmy te są w stanie poradzić sobie z „wybuchami” ruchu, czyli nagłym jego zwiększeniem, gdyż zaczynają one odrzucać pakiety zanim jeszcze bufor zostanie zapełniony, co ma także dużo większe znaczenie przy dużych prędkościach w sieciach szkieletowych.



Rys. 16. Wpływ stosowania mechanizmu WRED na efektywną transmisję. *Źródło: [26]*.

Najbardziej znaną odmianą mechanizmu RED jest algorytm „*ważonego losowego wczesnego wykrywania*" (*ang. Weighted Random Early Detection*) [27]. Algorytm ten potrafi rozpoznawać różne klasy ruchu, dla których z osobna są definiowane progi wypełnienia kolejki oraz maksymalne prawdopodobieństwo odrzucenia pakietu. Mechanizm ten pozwala na zapobieganie przeciążeniom, poprzez ograniczanie w pierwszej kolejności ruchu mniej istotnego.

BIBLIOGRAFIA:

[1] Astuti D.: Packet handling. Technical report. Department of Computer Science, University of Helsinki, 2003.

[2] Cisco: Internetworking technology handbook. http://www.cisco.com

[3] Semeria C.: Supporting differentiated service classes: queue scheduling disciplines. Juniper Networks

[4] Nagle J.: On Packet Switches With Infinite Storage. RFC970

[5] Almesberger W.: Linux network traffic control – Implementation overview. EPFL ICA. April 1999

[6] P. Mckenney. Stochastic Fair Queuing. In Internetworking: Reserach and Experience, Vol. 2, January 1991

[7] Demers A., Keshav S., Shenker S.: Analisys and Simulation of a Fair Queuing Alghorithm. Internetworking Research and Experience. October 1990.

[8] Guerin R., Peris. V.: Quality-of-service in packet networks: basic mechanisms and directions. Computer Networks Volume: 31, Issue: 3, February 1999

[9] Floyd S., Jacobson V.: Link-sharing and resource management models for packet networks. IEEE/ACM Transactions on Networking. Vol.3, No. 4 (Aug. 1995)

[10] Clark D. D., Jacobson V.: Flexible and Efficient Resource Management for Datagram Network. Unpublished manuscript. April 1991

[11] Katevenis M., Sidiropoulos S., Courcoubetis C.: Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip. IEEE Journal on Selected Areas in Communications, Vol. SAC-9, No. 8, October 1991

[12] Shredhar M., Varghese G.: Efficient fair queueing using deficit round robin. IEEE/ACM Trans. Networking (4) 3. November 1994

[13] Partridge C.: Token bucket with leaky bucket rate control. Gigabit Networking, Addison-Wesley, Reading, MA. 1994.

[14] Astuti D.: Packet handling. Technical report. Department of Computer Science, University of Helsinki, 2003.

[15] Zhao W., Olshefski D., Schulzrinne H.: Internet Quality of Service – An overview. Columbia University Technical Report, Feb. 2000.

[16] Maceluch R., Kasprzyk P., Domański A.: Dynamiczny przydział pasma użytkownika sieci z wykorzystaniem usługi QoS w systemie LINUX. ZN Pol. Śl. Studia Informatica Vol. 24, No 2A (53), Gliwice 2003.

[17] Czachórski T.: Modele kolejkowe w ocenie efektywności sieci i systemów komputerowych. Wyd. Politechniki Śląskiej 1999.

[18] Tuner J. S.: New Directions in Communications. IEEE Communications Magazine, 24(10), October 1986.

[19] Liu E. Y.: On End-to-End Performance of Multi-service Concatenation. Proceedings of IBM Center for Advanced Study Conference (CASCON'98). November 1998.

[20] Wagner K.: Short Evaluation of Linux's Token Bucket Filter (TBF) Queuing Discipline. May 2001.

[21] Anker T., Bergman E., Dolev D., Gelbourt I.: Hierarchical Bandwidth Sharing Made Simple. Technical Report. The Hebrew University of Jerusalem. February 2002

[22] Devera M.: Hierarchical Token Bucket. http://luxik.cdi.cz/~devik/qos/htb/

[23] Stoica I., Zhang H., Eugene Ng T. S.: A Hierarchical Fair Service Curve Algorithm for

link-sharing, real-time and priority services. Proceedings of SIGCOMM'97. Carnegie Mellon University.

[24] Benett J. C. R., Zhang H.: Hierarchical packet fair queuing algorithms. Proc. SIGCOMM, August 1996

[25] Floyd S., Jacobson, V.: Random Early Detection gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, vol. 1, no. 4, August 1993.

[26] Szarecki R.: QoS – jakość transmisji w Internecie. Telenet Forum – Numer specjalny. Styczeń 2001.

[27] Weighted Random Early Detection. http://www.cisco.com/univercd/cc/td/ /doc/product/software/ios112/ios112p/gsr/wred_gs.htm

[28] Scrimger R., LaSalle P., Leitzke C., Parihar M., Gupta M.: TCP/IP – Biblia. Wyd. Helion. Gliwice 2002.