DOI: 10.15199/59.2023.6.1

Nowe protokoły transportowe w sieciach IP. Analiza porównawcza wybranych mechanizmów w protokołach SCTP i QUIC

New transport protocols in IP networks: Comparative analysis of selected mechanisms in SCTP and QUIC protocols

Prof. dr hab. inż. JÓZEF WOŹNIAK
Katedra Teleinformatyki
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Od początku historii sieci komputerowych wdrażających ideę komutacji pakietów, czyli nowego – w stosunku do realizowanego w publicznych sieci telekomunikacyjnych – sposobu przekazu danych i zasad logicznego zestawiania połączeń, zainicjowanego i zaimplementowanego na szerszą skalę w sieci ARPANET ((*Department of Defense*) *Advanced Research Project Agency* ([1]), obserwujemy ciągłe pojawianie się i rozwój protokołów tworzących, ewentualnie uzupełniających architekturę TCP/IP (*Transmission Control Protocol/Internet Protocol*), która z biegiem lat stała się powszechnie akceptowanym światowym standardem.

STRESZCZENIE: Zestaw protokolarny TCP/IP opisuje podstawową koncepcję organizacji pracy Internetu. Z tej racji jest przedmiotem zainteresowania oraz stałych analiz zarówno operatorów, użytkowników, jak też badaczy zagadnień sieciowych, a wreszcie i projektantów, by reagować na pojawiające się nowe potrzeby. Działania takie, przynajmniej częściowo, są wymuszane zarówno przez wymagania nowych multimedialnych aplikacji - na najwyższym poziomie architektury sieciowej, jak też zupełnie nowe możliwości realizacji przekazów - związane z nowymi technologiami transmisyjnymi i technikami odbioru, pozwalającymi, przykładowo na zrównoleglenie przekazów, czy też bezstratne przełączanie pomiędzy kilkoma interfejsami. W artykule pokrótce zasygnalizowano oczekiwania i wymagania związane, w szczególności z nowymi „wielobiektowymi" aplikacjami, jak też ograniczenia wynikające z ogromnej bezwładności, obserwowanej po stronie infrastruktury transportowej sieci IP. Mając na uwadze wspomniane uwarunkowania dokonano charakterystyki wybranych protokołów transportowych typu *end-to-end*, poświęcając główną uwagę dwóm protokołom realizującym przekazy wielostrumieniowe, a mianowicie SCTP oraz QUIC.
Zaprezentowano zarówno wykorzystywane w obu protokołach struktury danych, jak też porównano podstawowe procedury związane z ich pracą. Dokonano też zestawienia cech charakterystycznych obu protokołów, wskazując na zakres ich użyteczności oraz scenariusze użycia.
SŁOWA KLUCZOWE: Protokoły transportowe, przegląd rozwiązań, charakterystyka wybranych rozwiązań: TCP, MPTCP, Protokoły wielostrumieniowe: QUIC, SCTP, analiza porównawcza, scenariusze użycia

ABSTRACT: The TCP/IP protocol set is the basic concept of organizing the work of the Internet. For this reason, it is the subject of interest and constant analysis by operators, users, network researchers and designers in order to respond to emerging new needs. Such actions, at least partially, are forced both by the requirements of new multimedia applications - at the highest level of network architecture, as well as completely new possibilities of transmitting messages - related to new transmission technologies and reception techniques, allowing, for example, for parallelization of messages and lossless switching(Handover) between several interfaces.
The article briefly indicates the expectations and requirements related, in particular, to new "multi-object" applications, as well as the limitations resulting from the huge inertia observed on the side of the IP network transport infrastructure. Taking into account the above-mentioned conditions, the selected end-to-end transport protocols were characterized, focusing mainly on two protocols implementing multi-stream transfers, namely SCTP and QUIC.
The data structures used in both protocols are presented, as well as the basic procedures related to their work are compared. The characteristic features of both protocols are also compared, indicating the scope of their usefulness and use scenarios.
KEYWORDS: Transport protocols, overview of solutions, characteristics of selected solutions: TCP, MPTCP, Multi-stream protocols: QUIC, SCTP, comparative analysis, usage scenarios

Zwykle, w tym kontekście, przywołuje się pięć najważniejszych cech tej architektury związanych, bądź z protokołem IP (*Internet Protocol*) i warstwą sieciową, bądź też z warstwą transportową i protokołem TCP (*Transmission Control Protocol*) [2], [3], [4]. Są to:

1. Globalnie unikatowa adresacja;
2. Datagramowość i bezstanowość;
3. Możliwość segmentacji i ponownego składania strumienia danych;
4. Porządkowanie napływających jednostek danych;
5. Niezawodność i integralność przekazu danych.

Dwie, a w zasadzie trzy pierwsze cechy dotyczą funkcjonowania podsieci komunikacyjnej i protokołów warstwy internetowej – tj. komutacji pakietów-datagramów i uniwersalnej adresacji, pozwalających na efektywne ale i elastyczne zestawianie tras w sieci IP. Środkowa funkcjonalność związana jest z jednej strony z paradygmatem niepodzielności adresów IP w przenoszonych datagramach, z drugiej zaś z ograniczeniami, na maksymalny rozmiar MTU (*Maximum Trannsmission Unit*), narzucanymi przez urządzenia pośredniczące w wymianie, względnie wynikające z wymagań po stronie aplikacji. Dwie ostatnie cechy wiążemy zwykle ze wspomnianym protokołem TCP. Nie zawsze bowiem można założyć, że to, co wprowadzimy do sieci w węźle źródłowym, dotrze do miejsca przeznaczenia w niezmienionej postaci. Nie można też zakładać, że w ogóle tam dotrze! Wpływ na to może mieć szereg przyczyn związanych z czynnikami przypadkowymi, jak np. uszkodzenia urządzeń i/lub łączy, przepełnianie się buforów, czy silne zakłócenia lub też celowymi działaniami osób trzecich. Rezultatem tego mogą być zniekształcone dane, czy serie pakietów, które zostaną utracone. Protokół TCP jest wystarczająco „inteligentny" (w sensie odpowiednich mechanizmów i metod algorytmicznych), aby wykryć dane, które zostały uszkodzone (poza przypadkami nieuprawnionych ale skutecznych manipulacji) lub utracone podczas przekazu przez sieć. Uszkodzone dane są odrzucane, a mechanizmy TCP gwarantują ich retransmisję, dopóki nie zostaną poprawnie odebrane i udanie potwierdzone.

Zestaw TCP/IP może oczywiście realizować wiele innych zadań. Tym niemniej wydaje się, że te pięć elementów stanowi podstawę komunikacji. Inne funkcje, w tym te, które są częścią protokołów TCP/IP, jak i te opracowywane komercyjnie, stanowiące uzupełnienie zestawu TCP/IP, bazują na tym fundamencie.

Aby nie pozostawiać tematu bez niedomówień dodajmy, że w klasycznym stosie TCP/IP, mamy również i drugi ważny protokół transportowy, czyli bezpołączeniowy, ale szybki UDP (*User Datagram Protocol*). Można też stwierdzić, że oba te protokoły to bez przesady „konie pociągowe" Internetu, odpowiedzialne za transport wiadomości - komunikatów generowanych przez nieograniczoną wręcz liczbę współczesnych aplikacji.

TCP obsługuje przy tym zdecydowaną większość aplikacji i jest używany w miliardach urządzeń, będąc nieprzerwanie od lat najczęściej i najszerzej wykorzystywanym protokołem transportu sieciowego, jaki powstał do tej pory. Podkreślmy jednocześnie, że bezpołączeniowy protokół UDP [5], któremu dalej nie będziemy poświęcali już miejsca (chociaż będzie nam stale „towarzyszył"), odgrywa ważną rolę przy obsłudze aplikacji typu: wideokonferencje, VoIP (*Voice over IP*), a także DNS (*Domain Name Systems*), DHCP (*Dynamic Host Configuration Protocol*), czy SNMP (*Simple Network Management Protocol*). Dodajmy też, że protokołów warstwy transportowej, standaryzowanych przez IETF (*Internet Engineering Task Force*) i publikowanych w dokumentach RFC jest oczywiście znacznie więcej, co najmniej kilkanaście zasługujących na uwagę, ale mających zdecydowanie ograniczone znaczenie [6].

TCP cechuje niewątpliwie duża elastyczność. Związane jest to z faktem, że w standardowym rozwiązaniu możliwe jest stosunkowo proste modyfikowanie szeregu mechanizmów związanych z kontrolą przepływu i/lub zarządzaniem mechanizmami przeciążeniowymi. Specyfikacja TCP nie jest więc sztywnym standardem.

Chociaż wszystkie pakiety - segmenty danych, wymieniane w ramach sesji, używają standardowych pól nagłówka TCP, to sposób wykorzystania tych pól oraz interpretacja pozyskiwanych danych sprawiają, że możliwe jest tworzenie elastycznych implementacji TCP, które zachowują się względem siebie radykalnie inaczej. Dotyczy to wielkości okien oraz skalowalności ich rozmiarów, różnorodności mechanizmów reagowania na przeciążenia w sieci, czy np. sposobów sterowania przepływem, także z wykorzystaniem opcjonalnych powiadomień selektywnych.

Dokonując nieco głębszych analiz współczesnego Internetu zauważamy, że protokół TCP, pomimo znacznej elastyczności, ma swoje poważne wady, szczególnie w przypadku świadczenia multimedialnych usług internetowych, których znaczenie jest coraz większe.

Jedna z najistotniejszych wad TCP związana jest z „jednostrumieniowością" i bajtową orientacją przekazu realizowanych w protokole sesji połączeniowych.

Jedna z najistotniejszych wad TCP związana jest z „jednostrumieniowością" i bajtową orientacją przekazu (z zachowaniem stosownej ich kolejności – *offset*) realizowanych w protokole sesji połączeniowych. Tymczasem większość stron internetowych nie jest już prostymi, monolitycznymi obiektami. Najczęściej zawierają one wiele oddzielnych komponentów, w tym obrazy, skrypty, niestandardowe ramki, arkusze stylów i tym podobne. Każdy z nich jest oddzielnym „obiektem" sieciowym, a jeżeli używamy przeglądarki wyposażonej w oryginalną implementację protokołu HTTP, to każdy obiekt powinien zostać załadowany w nowej sesji TCP, nawet gdy jest obsługiwany z tego samego adresu IP [7] . Narzuty związane z konfiguracją zarówno nowej sesji TCP, jak i nowej sesji TLS (*Transport Layer Security*) (do kryptograficznego zabezpieczania strumieni po stronie aplikacji) dla każdego odrębnego obiektu sieciowego w złożonym zasobie sieciowym stają się wtedy znaczne. Naturalną konsekwencją tego jest traktowanie takich podstrumieni jako części pojedynczego strumienia w sesji TCP.

Tym samym odrzucenie któregokolwiek pakietu-segmentu TCP spowoduje blokowanie dostarczania do aplikacji danych z innych podstrumieni (tzw. blokowanie początku kolejki – *HoL - Head of Line Blocking*)).

Tego typu sytuacje generują potrzebę przekazów wielostrumieniowych, czyli wydzielania w ramach jednego pakietu części przenoszących dane z różnych obiektów, także w ramach jednej sesji i niezależną obsługę danych z różnych podstrumieni i obiektów.

Inną pożądaną funkcjonalnością staje się obecnie możliwość przesyłania danych wieloma ścieżkami, co umożliwia obecny stan rozwoju sprzętu, szczególnie po stronnie urządzeń przenośnych, wyposażanych typowo w dwa lub więcej interfejsy (*multihoming*). Ta swoista redundancja możliwości transmisyjnych może też być wykorzystywana przy wspieraniu mobilności użytkowników.

Rozważając te i szereg innych wymagań, można stwierdzić, że dla coraz większej liczby aplikacji protokół TCP okazuje się niewystarczający, szczególnie wtedy, gdy:

- wymagana jest niezawodność bez gwarancji kolejności dostarczania jednostek danych,
- konieczne jest przesyłanie krótkich komunikatów, generowanych przez różne procesy po stronie aplikacji,
- wskazany jest przekaz wielu strumieni w jednym połączeniu (sesji), co wymagać będzie odejścia od traktowania strumienia jako sekwencji bajtów, na rzecz sekwencji wiadomości (krótkich lub długich),
- wskazane jest przekazywanie powiadomień selektywnych, dotyczących poszczególnych wiadomości w ramach „segmentu-komunikatu",
- pożądane jest wsparcie dla hostów z wieloma interfejsami sieciowymi tzw. multihoming,
- konieczne jest szybkie przełączanie i wsparcie mobilności, w coraz częściej spotykanym heterogenicznym środowisku sieci bezprzewodowych,
- wymagana jest wysoka odporność na ataki typu blokada/odmowa usługi DoS (*Denial of Service*).

Pojawia się zatem pytanie. Dlaczego nie zacząć od nowa i nie zdefiniować nowy protokół transportowy, który wyeliminuje wady TCP?

Odpowiedź udzielana na tak postawione pytanie jest częściowo zaskakująca ale i prosta. Na przeszkodzie w znacznym stopniu stoi NAT (*Network Address Translation*)! (por. np. [7], [8]). W dalszej części rozważań kilkakrotnie odwoływać się będziemy do analiz zaczerpniętych z zacytowanych powyżej opracowań.

Translacje adresów sieciowych NAT (*Network Address Translation*) zmieniły założenie, że urządzenia sieciowe nie analizują zawartości pakietu (a dokładniej, to translacje portów w NAT zmieniły to założenie). NAT to urządzenia sieciowe, które sięgają do „wnętrza" pakietów IP i przepisują adresy portów używane przez TCP i UDP. Co zrobić, jeżeli pakiet IP zawiera wartość identyfikatora protokołu transportowego typu end-to-end, która nie jest ani TCP, ani UDP? Większość NAT po prostu odrzuca pakiet na podstawie paradygmatu bezpieczeństwa, zgodnie z którym „to, czego nie rozpoznajesz, może być szkodliwe". „Pragmatycznym" rozwiązaniem jest zatem takie, że urządzenia typu NAT ograniczają wybór protokołów transportowych aplikacji w publicznym Internecie do zaledwie dwóch: TCP i UDP. Dodajmy też, że owo „pragmatyczne" podejście stosowane jest również w routerach i firewallach w publicznych sieciach, co sprawia, że w Internecie mamy (w praktyce) tylko TCP i UDP. Wielcy internetowi „gracze", jak Google, przewidują, że w bliżej nieokreślonej przyszłości możliwy będzie tylko HTTPS(!), gdyż TCP i UDP są zbyt niskopoziomowe i zbyt zależne od sprzętu i jego twórców.

Konkluzją płynącą z powyższej analizy jest to, że infrastruktura internetowa charakteryzuje się ogromną inercją we wprowadzaniu zmian, bądź też jest niewystarczająco podatna na proponowane zmiany. Tym niemniej w przestrzeni aplikacji otwierają się nowe możliwości. Aplikacje nie muszą czekać, aż architektura sieci obejmie obsługę określonego protokołu transportowego ani też czekać na wdrożenie wymaganych bibliotek pomocniczych. Aplikacje mogą samodzielnie rozwiązywać te problemy. Zysk w elastyczności i efektywności takiego podejścia może być znaczny.

Taki właśnie, bardzo nowatorski, a jednocześnie pragmatyczny punkt widzenia przyjęli projektanci protokołu QUIC (*Quick UDP Internet Connections*). Rozwiązaniem wybranym w QUIC jest podejście bazujące na UDP. O ile wybór UDP jako „podkładowego" protokołu typu end-to-end dla QUIC (nie jest to już skrót nazwy, ale nazwa własna protokołu) był wyborem podyktowanym wspomnianą nieelastycznością licznej grupy urządzeń NAT w publicznym Internecie i ich zbiorową niezdolnością do przyjęcia nowych protokołów, to obsługa pakietów UDP przez NAT ma też dalsze implikacje dla QUIC. Dokładniejszy opis protokołu QUIC zostanie zaprezentowany w dalszej części artykułu. Korzystać przy tym będziemy z kilku dokumentów, w tym między innymi: [7], [9],[10] .

Drugim wybranym przez nas do dalszej analizy rozwiązaniem, pozwalającym na realizację większości sformułowanych wcześniej wymagań, stawianych protokołom transportowym, będzie SCTP (*Stream Control Transmission Protocol*). W przypadku SCTP problem konwersji zachodzących w NAT pozostawiono bez pełnego rozwiązania, licząc zapewne na istotne zmiany w akceptacji protokołu przez przyszłe urządzenia sieciowe. Podobnie jak w odniesieniu do QUIC skorzystamy głównie z dokumentów RFC [8], [11,] [12],[13].

ANALIZA PORÓWNAWCZA WYBRANYCH MECHANIZMÓW W PROTOKOŁACH SCTP I QUIC

We wstępie wskazano na dużą elastyczność protokołu TCP, ukazując jednocześnie jego dość istotne mankamenty, w zestawieniu z wymaganiami współczesnych aplikacji. Mając to na uwadze określono nowe, pożądane funkcjonalności protokołów transportowych.

Wśród nowych propozycji aspirujących do roli rozwiązań, co najmniej alternatywnych, dla TCP dostrzegamy kilka niewątpliwie ciekawych propozycji, w tym MPTCP (*Multipath TCP*) oraz wspomniane już: QUIC i SCTP.

Podstawowe założenia wielościeżkowego protokołu MPTCP, zdefiniowane w kilku dokumentach (np. [6],[14],[15])

obejmują nowe, w stosunku do TCP, mechanizmy sterowania transmisją TCP, na które składają, między innymi:

- System podprzepływów, używany do rozpoznawania i zbierania danych z wielu standardowych połączeń TCP (ścieżek z jednego hosta do drugiego). Podprzepływy są identyfikowane podczas trójstronnego uzgadniania sesji TCP. Po uzgodnieniu aplikacja może dodać lub usunąć niektóre podprzepływy.

A ponadto:

- Opcja MPTCP DSS (*Data Sequence Signal*) - zawierająca numer sekwencyjny danych i numer powiadomienia co umożliwia odbieranie danych z wielu podprzepływów w oryginalnej kolejności, bez jakichkolwiek modyfikacji;
- Zmodyfikowany protokół retransmisji nadzorujący kontrolę zatorów (przeciążenia - *congestion)* i niezawodność (reliability).

MPTCP zapewnia przy tym:

- obsługę dodawania/usuwania ścieżek (na przykład w przypadku utraty połączenia lub wystąpienia stanów przeciążenia),
- kompatybilność ze starszym sprzętem TCP (np. z niektórymi zaporami sieciowymi, które mogłyby automatycznie odrzucać połączenia TCP, gdyby numery sekwencyjne nie były kolejne),
- zdefiniowanie uczciwej strategii kontroli przeciążenia (w tym: *scheduling* i *load balancing*) pomiędzy różnymi łączami i różnymi hostami (szczególnie w przypadku tych, które nie obsługują MPTCP).

Wielościeżkowość w MPTCP nie gwarantuje jednak pożądanej w nowych, szczególnie przeglądarkowych aplikacjach obsługi wielostrumieniowości.

Tę pożądaną cechę posiadają natomiast dwa pozostałe z wymienionych powyżej protokołów, i to właśnie im poświęcimy naszą uwagę.

W dalszej części, korzystając z licznych dokumentów standaryzacyjnych przybliżymy podstawowe procedury realizowane zarówno w protokole *Quick UDP Internet Connections* (QUIC), coraz szerzej wykorzystywanym w pracy przeglądarek internetowych do obsługi aplikacji webowych, jak też w pozostającym nieco „w cieniu", ale ciekawym produkcie, jakim jest protokół *Stream Control Transmission Protocol* (SCTP). Protokół ten został opracowany, w swej pierwotnej postaci, w roku 2000 (RFC 2960 [11]) z przeznaczeniem do transportu komunikatów sygnalizacyjnych (sygnalizacja SS7) w publicznej komutowanej sieci telefonicznej (*Public Switched Telephone Network* - PTSN) za pośrednictwem datagramów IP. W późniejszym okresie został także (po części) zaadaptowany do pracy w sieciach IP, jako protokół transportowy ogólnego przeznaczenia

Pomimo dość odmiennych scenariuszy wykorzystania, charakterystycznych dla obu tych protokołów oraz znacznej „dysproporcji" w „powszechności" użycia w klasycznych sieciach IP, rozwiązaniom tym poświęcimy uwagę w dalszej części artykułu. Skorzystamy (w zasadniczej części rozważań) z analizy porównawczej zawartej w dokumencie [8]. Wspomagając się także innymi dokumentami standaryzacyjnymi RFC, przedstawimy podobieństwa i różnice między dwoma wspomnianymi protokołami transportowymi: Stream *Control Transmission Protocol* (SCTP) i *Quick UDP Internet Connections* (QUIC). Dodajmy dla porządku (ponownie), że QUIC nie jest już skrótem ale nazwą własną protokołu.

W kolejnych fragmentach niniejszego artykułu zaprezentujemy podstawowe procedury i mechanizmy związane z zestawianiem połączeń, przekazem danych, sterowaniem przepływem danych, a także zamykaniem połączeń w protokołach SCTP oraz QUIC. Poprzedzimy te rozważania opisem wybranych, charakterystycznych dla obu protokołów, struktur danych (szereg innych struktur zostanie zaprezentowanych podczas omawiania specyficznych mechanizmów i funkcjonalności implementowanych w obu protokołach).

Na zakończenie rozważań podamy też (częściowo za [7]) ogólne wnioski dotyczące cech oraz scenariuszy użycia obu protokołów. Dokonamy także szerszego porównania cech funkcjonalnych obu protokołów z cechami protokołu TCP - taktowanego jako referencyjny.

STRUKTURY PAKIETÓW W SCTP ORAZ QUIC

W nazewnictwie jednostek danych (*Protocol Data Units - PDUs*), tworzonych na różnych poziomach architektury TCP/IP, używane są różne określenia. Najpowszechniej przyjmuje się, że w Warstwie Łącza Danych przesyłane są RAMKI. Z kolei w Warstwie Internetowej - Sieciowej dominuje określenie DATAGRAMU, ew. zamiennie PAKIETU. W przypadku Warstwy Transportowej i protokołu TCP jest to SEGMENT, chociaż w odniesieniu do UDP częściej nazywamy taką jednostkę PAKIETEM. W Warstwie Aplikacji mamy zwykle do czynienia z WIADOMOŚCIAMI.

Z uwagi na dość zróżnicowane nazewnictwo, stosowane przez różnych autorów, w dalszej części artykułu przyjmiemy, że interesujące nas jednostki danych - PDUs (Protocol Data Units), przesyłane w QUIC oraz SCTP nazwiemy PAKIETAMI. Jak wspomniano wcześniej PAKIETY (pisane dalej z małej litery) będą przenosiły wiadomości - komunikaty związane zwykle z kilkoma strumieniami generowanymi przez aplikację, lub grupę aplikacji.

Części pakietów w QUIC nazywane będą przez nas (zgodnie z zapisami w dokumentach RFC) „ramkami" (*FRAMES*), natomiast w SCTP „fragmentami" (*CHUNKS*) (ale można też znaleźć opisy w języku polskim w postaci „porcji/części").

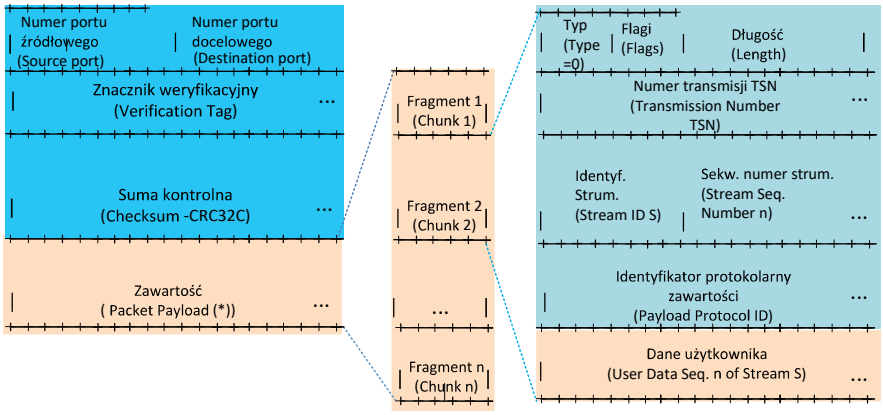
Struktury pakietów zarówno w SCTP, jak i w QUIC różnią się od tych z protokołu TCP, w którym mamy jeden „uniwersalny" nagłówek dla wszystkich typów segmentów. Zamiast tego w SCTP/QUIC korzystamy z dedykowanych fragmentów-ramek do konfiguracji połączeń, transmisji danych, rozłączania, czy też przekazu powiadomień selektywnych SACK.

Pakiety SCTP

Ogólna koncepcja struktury pakietu SCTP (RFC4960 [12]) pokazana jest na rys. 1. Pakiet obejmuje wspólny nagłówek, z dołączonymi „fragmentami" (*chunks*).

Znaczenie poszczególnych pól wspólnego nagłówka oraz nagłówków fragmentów (typu DATA), przenoszących zarówno informacje kontrolne jak i dane przesyłane z aplikacji, przedstawiono na rys. 2.

» *Rys. 1. Ogólna struktura pakietu SCTP obejmująca wspólny nagłówek oraz jeden lub więcej fragmentów (chunks – części/porcji) z własnymi nagłówkami*



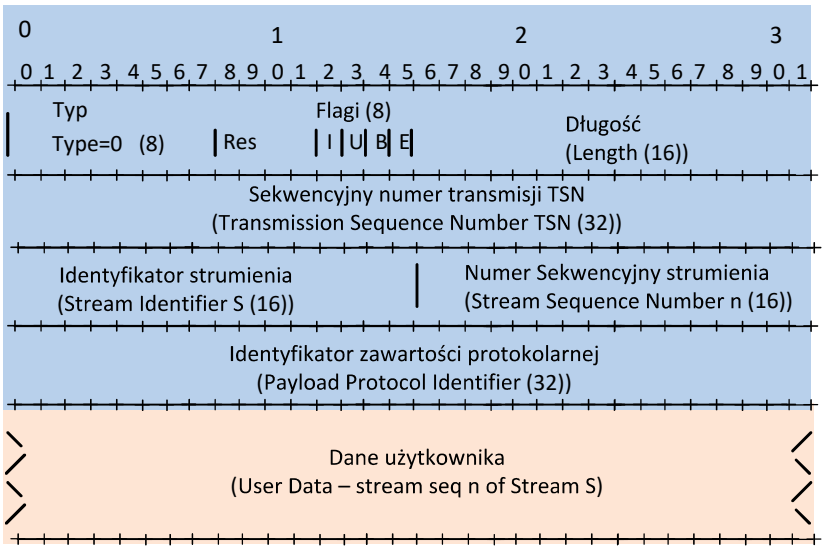
» *Rys. 2. Struktura pakietu (PDU) SCTP ze wspólnym nagłówkiem i z dołączonymi fragmentami/częściami (chunks) typu DATA*

Z uwagi na podobne zadanie obu protokołów QUIC i SCTP, tj. zrównoleglonę obsługę wielu strumieni, struktura pakietu SCTP jest w zasadzie analogiczna do struktury zaprezentowanego i omawianego w dalszej części „wzorcowego" pakietu QUIC z krótkim nagłówkiem (rys. 4).

Wspólny nagłówek SCTP zawiera pola z adresami portów, znacznik weryfikacyjny (identyfikator/etykietę - *Tag*) i sumę kontrolną pakietu. Nagłówek ten liczy łącznie 12 bajtów, w porównaniu z 20 bajtami z TCP. Suma kontrolna w SCTP jest 4-bajtowa, podczas, gdy w TCP zajmuje ona 2 bajty. Znacznik weryfikacyjny jest wykorzystywany jako identyfikator asocjacji (nawiązywanego połączenia - powiązania dwóch urządzeń końcowych).

W protokole SCTP przewidziano szereg specyficznych komunikatów i typów fragmentów, definiowanych poprzez wartość z zakresu od 0 do 255 w polu Typ Fragmentu. Są to w szczególności:

0 - Payload Data (DATA)
1 - Initiation (INIT)
2 - Initiation Acknowledgement (INIT ACK)
3 - Selective Acknowledgement (SACK)
4 - Heartbeat Request (HEARTBEAT)
5 - Heartbeat Acknowledgement (HEARTBEAT ACK)
6 - Abort (ABORT)
7 - Shutdown (SHUTDOWN)
8 - Shutdown Acknowledgement (SHUTDOWN ACK)
9 - Operation Error (ERROR)
10 - State Cookie (COOKIE ECHO)
11 - Cookie Acknowledgement (COOKIE ACK)
12 - Reserved for Explicit Congestion Notification Echo (ECNE)
13 - Reserved for Congestion Window Reduced (CWR)
14 - Shutdown Complete (SHUTDOWN COMPLETE)
15 to 62 - reserved by IETF
63 - IETF-defined Chunk Extensions
64 to 126 - reserved by IETF
127 - IETF-defined Chunk Extensions
128 to 190 - reserved by IETF
191 - IETF-defined Chunk Extensions
192 to 254 - reserved by IETF
255 - IETF-defined Chunk Extensions.



» *Rys. 3. Struktura fragmentu SCTP typu DATA Chunk*

W ramach pakietu ze wspólnym nagłówkiem wysyłana może być zmienna liczba fragmentów, aż do maksymalnej możliwej, czyli ograniczonej przez rozmiar *Maximum Transmission Unit* (MTU). Wszystkie fragmenty mogą być przy tym łączone w ramach pakietów z wyjątkiem INIT, INIT ACK i SHUTDOWN COMPLETE (których nie wolno łączyć w pakiety). Wiadomości reprezentujące DANE mogą być dzielone tak, aby zmieściły się w MTU pakietów SCTP (może to wymuszać konieczną segmentację wiadomości). Podkreślmy, że odmiennie niż TCP czy QUIC, SCTP jest protokołem typu *message-oriented*.

Podstawowym typem jest oczywiście Fragment DATA, pokazany zarówno na rys. 2, jak i w pełniejszej postaci na rysunku 3. Ma on zarezerwowaną wartość pola Typ Fragmentu (pole 8-bitowe) równą 0.

W przypadku pola z Flagami, bit U ustawiony na 1 wskazuje, że dane są nieuporządkowane (nie wymagają odtwarzania kolejności - *unordered delivery service*), a wartość SSN numeru sekwencyjnego strumienia (*Stream Sequence Number*) można zignorować.

Z kolei bit B=1 informuje o początkowym fragmencie (nie podlegającym segmentacji) w strumieniu, podczas, gdy bit E=1 wskazuje na końcowy fragment.

Pole Length (16 bitów) wskazuje długość fragmentu DANYCH w bajtach, w tym 16 bajtów pól kontrolnych, począwszy od pola Typ.

Jak podkreślono wcześniej SCTP został zaprojektowany do przenoszenia wielu strumieni danych w pojedynczych pakietach. Poszczególne strumienie, po zestawieniu połączenia, mają więc przyporządkowane identyfikatory (*Stream IDs*) wskazujące strumień, do którego należy dany fragment (*chunk*). Kolejny parametr związany ze strumieniem to SSN (*Stream Sequence Number*) podający kolejny numer fragmentu/porcji (*chunk*) w ramach powiązanego strumienia (identyfikowanego przez wspomniany *Stream ID*), a także określa kolejność, w jakiej komunikaty będą dostarczane do aplikacji - ULP (chyba że ma to być przekaz nieuporządkowany). Zauważmy jednocześnie, że fragmenty typu DATA mogą, ale nie muszą, mieć odwzorowanie jeden do jednego z komunikatami - wiadomościami generowanymi przez proces nadawczy (dopuszczana jest procedura fragmentacji). Numer SSN zawsze zaczyna się od 0 i wzrasta do 65535 (z „zawijaniem" po osiągnięciu wartości maksymalnej). Zwiększa się on sekwencyjnie przez cały okres istnienia powiązania (asocjacji).

Poprzedzający te pola numer TSN (*Transmission Sequence Number*) określa kolejny numer transmisji, pozwalający na jednoznaczne numerowanie transmisji pakietów, a tym samym i zawartych w nich fragmentów, w celu kontrolowania przesyłu danych. Numery te są przywoływane i powtarzane podczas retransmisji pakietów, aby zapewnić integralny i niezawodny transfer danych. Procedury sterowania przepływem oraz kontroli przeciążeniowej, z wykorzystaniem TSN, będą tematem jednego z dalszych fragmentów artykułu.

Wartości TSN mieszczą się w zakresie od 0 do 4294967295 i mogą zaczynać się od losowej wartości z tego zakresu.

Identyfikator zawartości protokołu (*Payload Protocol Identifier*) nie jest używany przez sam protokół SCTP, ale jest przeznaczony do użycia przez urządzenia pośredniczące.

Pole danych użytkownika (*User Data*) zawiera dane z aplikacji, które są uzupełniane na końcu do 4-bajtowej granicy bajtami zerowymi.

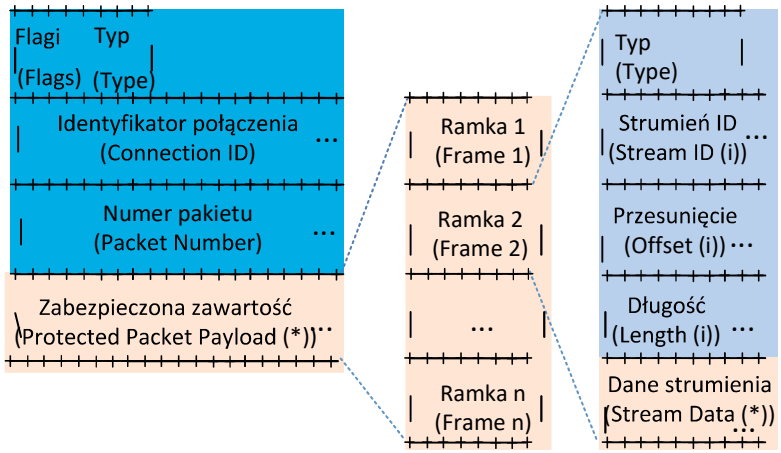
Pakiety QUIC

Odmiennie niż w SCTP w pakietach QUIC wykorzystywane są dwie formy nagłówka: długa i krótka. Do początkowej wymiany informacji, w tym zestawiania i konfigurowania połączenia wraz z uzgadnianiem współdzielonego klucza kryptograficznego i procesem negocjacji wykorzystywanej wersji protokołu używane są długie nagłówki, którym nieco więcej miejsca poświęcimy w dalszej części artykułu.

Po skonfigurowaniu połączenia gro informacji, w tym dane pochodzące z poszczególnych strumieni związanych z aplikacją przenoszone będą przez pakiety o krótkim nagłówku z dołączonymi ramkami typu STREAM (odpowiednikami fragmentów typu DATA SCTP). Struktura pakietu i ramki STREAM pokazana jest na rys.4. Podobnie jak w SCTP pakiety QUIC mogą przenosić jednocześnie dane z kilku strumieni.

Wspomniany krótki nagłówek, obejmuje 1-bajtowe pole Flag, pozwalające między innymi na zdefiniowanie typów ramek. Fakt przenoszenia w pakiecie ramek STREAM określa 5-bitowe pole Typ z wartościami z zakresu 0x10-0x17.

Wspólny nagłówek zawiera ponadto 8-bajtowy identyfikator danego połączenia z urządzeniem docelowym (Connection ID), a także numer pakietu (dopuszcza się zróżnicowaną długość tego pola: 1, 2 lub 4 bajty). Wypełnienie pakietu stanowią zabezpieczone kryptograficznie (z użyciem TLS- *Transport Layer Security*) ramki typu STREAM, z danymi odwzorowującymi generowane przez aplikację wiadomości. Zawierają one również 1-bajtowe pole Typu. Trzy mniej znaczące bity tego pola, informują o organizacji ramki, sygnalizując obecność w niej kolejnych pól (oraz sposób ich wypełnienia). 0x04



» *Rys. 4. Struktura przykładowego pakietu QUIC z krótkim wspólnym nagłówkiem, przenoszącego dane aplikacji w ramkach typu STREAM*

jest bitem OFF, a jego ustawienie na 1, oznaczać będzie istotność wartości w polu przesunięcia (offset field). W przypadku ustawienia tego bitu na 0, przesunięcie danych ramki (*offset*) będzie zerowe, bądź też ramka nie będzie zawierała żadnych danych. Z kolei bit na pozycji 0x02 jest związany z długością (length) ramki. Jego ustawienie na 1 sygnalizuje obecność tego pola w ramce, natomiast wartość 0 tego bitu, oznacza, że dane rozciągają się do końca pakietu. Wreszcie 0x01 jest bitem FIN. Jeżeli jest on ustawiony na 1, to sygnalizuje ostatnią ramkę w strumieniu.

Identyfikator strumienia (*Stream ID*) wskazuje strumień, do którego należy ramka, a także określa, czy strumień jest dwukierunkowy czy jednokierunkowy oraz kto - serwer, czy klient - utworzył strumień. Jeżeli drugim, najmniej znaczącym bitem jest 1, strumień jest jednokierunkowy, gdy jego wartość jest zerem, to strumień jest dwukierunkowy.

Z kolei, jeżeli najmniej znaczący bit to 1, wówczas jest to interpretowane, jako zainicjowanie strumienia przez serwer. Jeżeli, natomiast jego wartość jest zerowa, wówczas to klient zainicjował strumień.

Pole przesunięcia (*offset*) wskazuje wartość przesunięcia pierwszego bajtu przenoszonego przez ramkę STREAM, a pole długości podaje długość danych w strumieniu. W jednym pakiecie QUIC, o wspólnym nagłówku, można przesyłać wiele ramek typu STREAM.

Dodajmy jeszcze, że w początkowych fazach połączenia pomiędzy polami Connection ID oraz Packet Number mogą pojawiać się także dwa inne pola:

- 32-bitowy znacznik, reprezentujący wersję protokołu QUIC. Występuje on tylko w kierunku Klient->Serwer, o ile ustawiona jest flaga *WERSJA QUIC* oraz 32-bajtowy ciąg (*Diversification Nonce*) jednorazowo generowany przez serwer i używany tylko w kierunku Serwer->Klient, informujący, że serwer będzie w stanie wygenerować unikalne klucze dla danego połączenia. Jest to szczególnie istotne podczas kryptograficznego uzgadniania 0-RTT QUIC. W przypadku, gdyby ponownie została przesłana wiadomość Client Hello (CHLO), z tym samym identyfikatorem połączenia - Connection ID, mogłoby to prowadzić do wyboru tego samego klucza w dwóch różnych połączeniach. Wartość wygenerowana przez serwer uniemożliwia klientowi wygenerowanie tych samych kluczy dla dwóch różnych połączeń.

W przypadku przejścia w tryb zabezpieczonej wymiany danych pole to jest usuwane. „Regularne" pakiety zawierają bowiem dane objęte zabezpieczeniami AEAD (*Authenticated Encryption with Associated Data*).

W protokole QUIC, podobnie jak w SCTP, zdefiniowano kilkanaście rodzajów ramek, w tym między innymi [16]:

- wspomniane powyżej ramki STREAM Frames - do przenoszenia zawartości strumieni generowanych przez aplikacje oraz
- MAX_DATA Frame - do określenia maksymalnego okna odbiorczego,
- MAX_STREAM_DATA Frame - do określenia maksymalnej ilości danych w ramach strumienia,
- MAX_STREAM_ID Frame - do opisu maksymalnej liczby strumieni,
- ACK Frame - do efektywnego sterowania przepływem oraz kontroli poprawności wymienianych pakietów.

Istotną rolę pełni też liczna grupa innych ramek, w tym: NEW_CONNECTION_ID Frame, STOP_SENDING Frame, ACK Block PADDING Frame, RST_STREAM Frame, CONNECTION_CLOSE Frame, APPLICATION_CLOSE Frame , PING Frame, BLOCKED Frame, STREAM_BLOCKED Frame, STREAM_ID_BLOCKED Frame, a także kilka innych.

PODSTAWOWE CECHY ZESTAWIANYCH POŁĄCZEŃ W SCTP I W QUIC ([8], [9], [10], [12], [13])

Oba protokoły są zorientowane połączeniowo. Tym niemniej do uzgadniania zasad wymiany informacji między obiektami wykorzystują one komunikaty oraz mechanizmy zlokalizowane w różnych warstwach stosu TCP/IP, co mocno rzutuje na szczegóły implementacyjne. SCTP to „typowy" protokół warstwy transportowej, podczas gdy w QUIC kody funkcjonalności transportowych zostały przeniesione do aplikacji i są wykonywane w przestrzeni użytkownika.

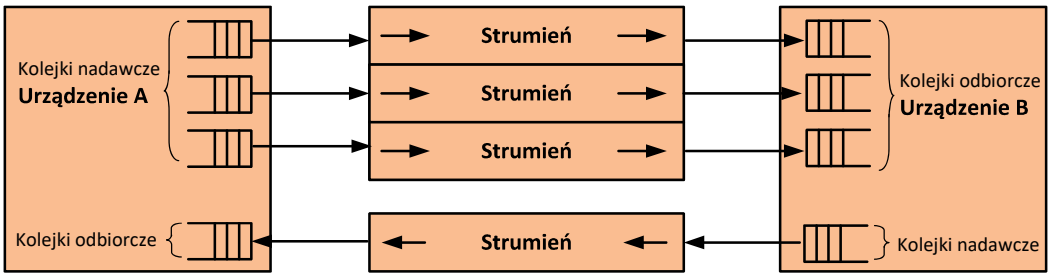
Konsekwencją tego odmiennego „usytuowania" są bardzo istotne różnice projektowe. Tradycyjne protokoły warstwy transportowej są częścią stosu TCP/IP i ich implementacja jest możliwa wyłącznie za akceptacją twórców jądra systemu operacyjnego, co wiąże się z bardzo długim i nierównoczasowym wdrożeniem w różnych systemach. Protokół implementowany w warstwie aplikacyjnej jest niezależny od systemu i jego implementacja może być wieloplatformowa i wdrażana natychmiastowo, niezależnie od cyklu pojawiania się nowych wersji jądra systemu u różnych producentów.

Sesje połączeniowe w SCTP

Połączenie SCTP określane jest mianem powiązania (asocjacji - *association*) między dwoma punktami końcowymi, z których każdy jest zdefiniowany przez zestaw adresów IP i numer portu. Podczas gdy podstawowy adres IP jest typowo wykorzystywany przez każdy punkt końcowy, powiązane ze sobą hosty - punkty końcowe, mogą też poinformować urządzenie współpracujące o rozszerzonym zestawie adresów, możliwych do użycia podczas przesyłania pakietów (wsparcie dla *multihomingu*).

Identyfikator strumienia (Stream ID) wskazuje strumień, do którego należy ramka, a także określa, czy strumień jest dwukierunkowy czy jednokierunkowy oraz kto - serwer, czy klient - utworzył strumień.

Odmiennie od rozwiązania TCP, gdzie wykorzystuje się jeden nagłówek do przekazu całej zawartości segmentu, w SCTP zaprojektowano wiele oddzielnych struktur danych zwanych fragmentami, czy też częściami/porcjami (*chunks*) do przenoszenia zarówno informacji sterujących o zasocjowanych - powiązanych strumieniach, jak i poszczególnych wiadomości generowanych przez aplikację.



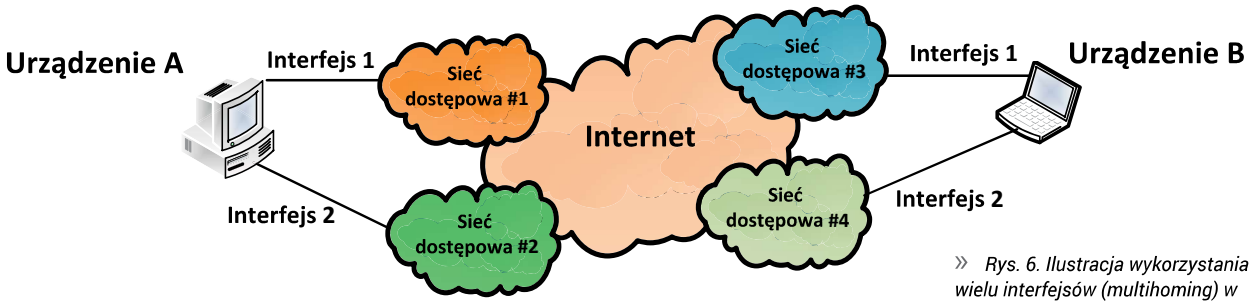
» *Rys.5. Ilustracja obsługi wielu strumieni w SCTP (wg [17])*

SCTP komunikuje się z protokołem warstwy wyższej ULP (*Upper Layer Protocol*), czyli z aplikacją, poprzez wymianę prymitywów komunikacyjnych ASSOCIATE i SHUTDOWN. Dzięki nim aplikacje są w stanie komunikować się z SCTP w celu konfiguracji, bądź usunięcia powiązania.

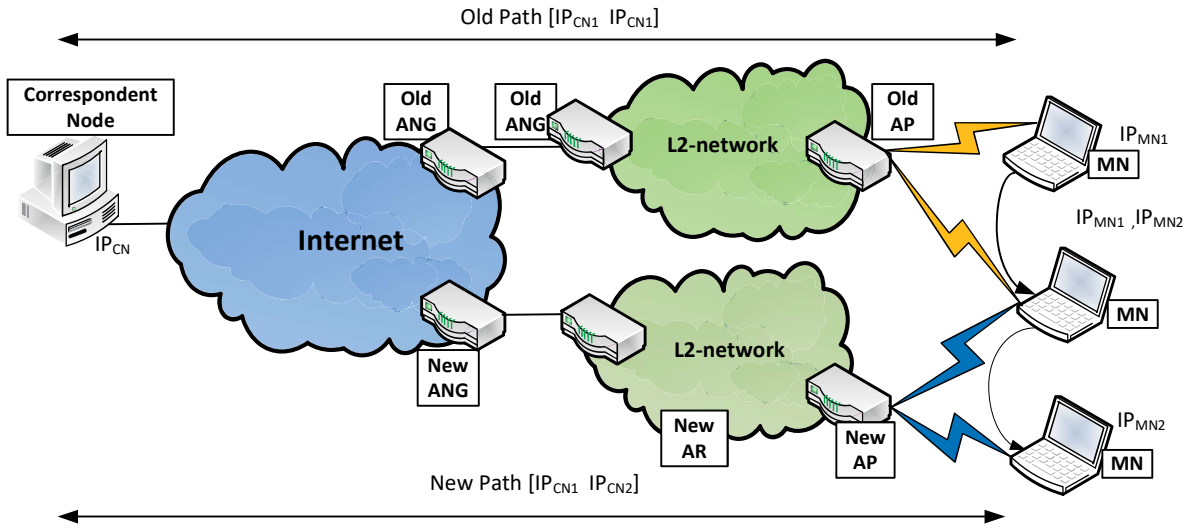
SCTP może obsługiwać wiele strumieni (*multistreaming*) wytwarzających komunikaty (wiadomości), pozwalając każdemu z dwóch punktów końcowych negocjować liczbę strumieni, które można obsłużyć w ramach konfigurowanego powiązania i zapewnić kolejność dostarczania wiadomości w strumieniu podrzędnym do ULP (do aplikacji), z użyciem numeru sekwencyjnego strumienia podrzędnego. W sposób poglądowy pokazano to na rys. 5.

SCTP pozwala też na wykorzystywanie dwóch lub więcej interfejsów jednocześnie (*multihoming*) (por. rys. 6). Funkcjonalność ta sprawia, że protokół ten może jest brany pod uwagę także w kontekście bezstratnego przełączania między sieciami wykorzystującymi różne techniki dostępowe (patrz rys. 7). Interesujące studium zarządzania mobilnością w SCTP można znaleźć np. w [17].

Aby zapewnić niezawodne dostarczanie komunikatów dla wszystkich strumieni, protokół SCTP przypisuje poszczególnym fragmentom danych unikalny sekwencyjny numer transmisji (*Transmission Sequence Number*). Należy przy tym pamiętać, że numer TSN jest określany dla asocjacji (powiązania), a nie dla każdego strumienia podrzędnego. Działa on w analogiczny sposób jak numer sekwencyjny (*Sequence Number*) w TCP, zapewniając niezawodne dostarczanie informacji, z tym wyjątkiem, że pierwszy numeruje pakiety - bloki danych (z jednym lub wieloma fragmentami), a drugi wskazuje



» *Rys. 6. Ilustracja wykorzystania wielu interfejsów (multihoming) w pracy urządzeń (wg [17])*



» *Rys. 7. Przykładowy scenariusz wykorzystanie SCTP do zarządzania mobilnością - z węzłem korespondującym (CN) z jednym interfejsem oraz węzłem mobilnym (MN) z dwoma interfejsami. Oznaczenia na rysunku: ANG - access network gateway, AP - access point, AR - access router, NAR - new access router, OAR - old access router (wg [17])*

na numery bajtów danych (w TCP), gwarantując właściwe ich porządkowanie (*offset*). SCTP wykorzystuje też te same, co TCP, mechanizmy unikania i kontroli przeciążenia oraz podobny schemat selektywnego potwierdzania, z tą różnicą, że SCTP ma dedykowany komunikat SACK, podczas gdy TCP wykorzystuje do tego celu pole opcji.

Sesje połączeniowe w QUIC

Połączenie QUIC to powiązanie między dwoma punktami końcowymi zdefiniowanymi przez parę adresów IP i numerów portów (obsługa *multihoming/multipath* w QUIC jest wciąż w fazie rozwoju [18] [19]).

Adresy IP i numery portów mogą przy tym ulegać zmianie w czasie połączenia. Podstawowa różnica między QUIC, a TCP lub SCTP, polega na tym, że QUIC jest protokołem transportowym w przestrzeni użytkownika (protokołem transportowym warstwy aplikacji). Dzięki temu możliwa jest szybka rewizja - modyfikacja parametrów konfiguracyjnych protokołu. By zapewnić taką funkcjonalność protokołu, podczas zestawiania połączenia w QUIC realizowany jest szybki proces negocjacji, który obejmuje określenie wersji obsługiwanej przez oba punkty końcowe oraz zabezpieczenia kryptograficzne, realizowane przez protokół TLS, w celu zapewnienia bezpiecznego połączenia.

W ramach połączenia QUIC strumienie mogą być „uaktywniane" w dowolnym momencie, z wyłączeniem fazy usuwania połączenia. Każdy punkt końcowy może zainicjować strumień podrzędny, który może być dwukierunkowy lub jednokierunkowy. QUIC dziedziczy model danych strumienia bajtów TCP. Dedykowane struktury kontrolne tzw. „ramki" służą do komunikacji i przenoszenia bajtów danych między punktami końcowymi.

Podobnie jak SCTP, QUIC posiada dedykowaną ramkę - komunikat SACK (nazywany formalnie QUIC ACK) do przenoszenia potwierdzeń selektywnych, chociaż semantyka QUIC ACK różni się od semantyki SCTP w istotny sposób. QUIC ACK informuje, które pakiety zostały dostarczane do docelowego punktu końcowego - pakiety niepotwierdzone są uważane za utracone. Żaden pakiet QUIC nie jest nigdy retransmitowany (z tym samym numerem), a numery pakietów (także retransmitowanych – w identycznej lub zmienionej postaci) rosną monotonicznie. Informacja powyższa oznacza, że na podstawie odebranej ramki - komunikatu (S)ACK punkt końcowy QUIC może wywnioskować, które wysłane przez ten punkt „ramki" bajtowe (w ramach pakietu QUIC) nie zostały odebrane. Aby zapewnić niezawodne dostarczanie danych w kolejności każdego strumienia bajtów do aplikacji, nadawca będzie retransmitował ramki bajtów, które nie zostały potwierdzone. Nowe ramki mogą „przepakowywać" brakujące dane (by zachować integralność danych). Każdorazowo będą one transmitowane w pakietach o narastających numerach (zatem odmiennie niż w TCP, czy SCTP).

Jako kolejną różnicę w stosunku do SCTP, wskazuje się i to, że QUIC realizuje kontrolę przepływu zarówno na podstawie parametrów dla całego połączenia (sesji), jak i parametrów poszczególnych strumieni, podając maksymalną ilość danych dozwolonych do przesłania w połączeniu (analogicznie do *window size* w TCP), a także przypadającą na strumień. Jeśli punkt końcowy wyśle więcej danych niż zadeklarowano, całe połączenie jest usuwane.

ZESTAWIANIE POŁĄCZEŃ W SCTP I QUIC([8], [9], [10], [12], [13])

Do konfiguracji połączenia protokół SCTP stosuje 4-stronne uzgadnianie (*4-way handshake*) z cyfrowo podpisanym plikiem cookie stanu w celu zapobiegania atakom typu „odmowa usługi" (SYN-flood). Plik stanu (STATE COOKIE) jest wysyłany przez serwer w odpowiedzi na komunikat INIT klienta i zawiera wszystkie informacje o stanie, których serwer potrzebuje, aby upewnić się, że powiązanie jest prawidłowe. Obejmuje to kod uwierzytelniania wiadomości (MAC - *Message Authentication Code*) [20], znacznik czasu i „czas życia" plików cookie. Plik cookie zawiera wszystkie informacje potrzebne do konfiguracji powiązania SCTP, tak więc SCTP po stronie serwera nie musi przechowywać informacji o kojarzonym (asocjowanym) kliencie.

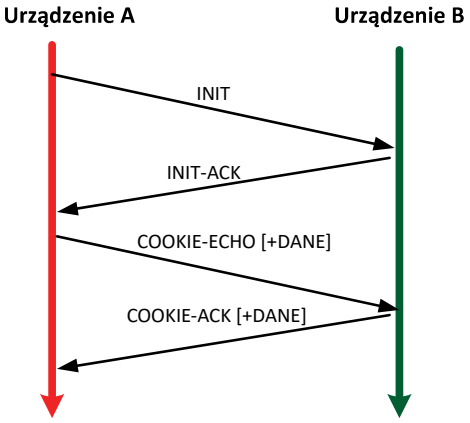
W przypadku konfiguracji połączenia w protokole QUIC ma miejsce bezpośrednie powiązanie procesu negocjacji klucza TLS z uzgadnianiem warunków transportu. Zapewnia to zestawienie bezpiecznego połączenie w ramach 1-RTT - z udaną negocjacją wersji i 0-RTT - w przypadku wznowienia połączenia (nieco szerszy opis zostanie przedstawiony w dalszej części tego artykułu). Podczas początkowej konfiguracji połączenia serwer przekazuje klientowi kryptograficzny plik cookie znany jako token adresu źródłowego (adres IP klienta i znacznik czasu) w celu weryfikacji adresu źródłowego. Wysyła również konfigurację serwera zawierającą długoterminową wartość współdzielonego klucza publicznego Diffie-Hellmana i preferencje serwera. Informacje te mogą być wykorzystane do zestawiania kolejnych połączeń. Zapewnia to bezpieczny i skuteczny sposób nawiązywania połączeń, ale w przeciwieństwie do tradycyjnych plików cookie syn-cookie, które zapobiegają atakom typu syn-flood, a które są przeznaczone do jednorazowego użytku, długoterminowe pliki cookie QUIC mogą stwarzać potencjalne możliwości nowych rodzajów ataków (np. atak powtórzeniowy). Serwer QUIC przyjmuje również bezstanową walidację adresów, a plik cookie przechowuje wszystkie stany niezbędne do kontynuacji połączenia.

Zestawianie połączenia w SCTP

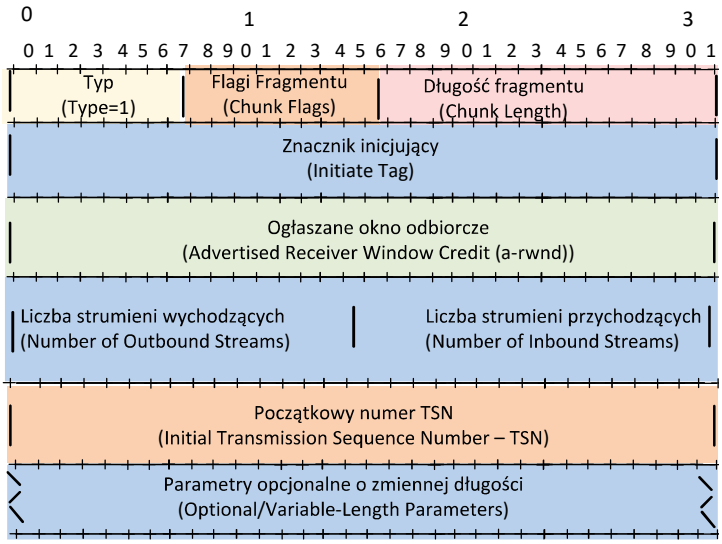
Koncepcyjnie SCTP jest podobny do TCP, w którym połączenie jest definiowane przez parę adresów IP i numerów portów. Tym niemniej SCTP różni się od TCP, definiując zestaw możliwych adresów IP (więcej niż jeden) i numerów portów we wspólnym nagłówku.

Ma to na celu ułatwienie realizacji funkcji *multihoming* SCTP, ponieważ wiadomości mogą być wysyłane (lub odbierane) na dowolny z tych adresów, nawet jeżeli określono adres podstawowy.

Inicjowanie powiązania (asocjacji) pomiędzy dwoma punktami SCTP (por. rys. 8) rozpoczyna się, gdy jeden punkt końcowy, np. SCTP A, wysyła fragment/porcję (*chunk*) typu



» *Rys. 8. Przykład nawiązania połączenia SCTP z 4-stronnym powiadamianiem (4-way handshake)*



» *Rys. 9. Struktura fragmentu SCTP typu INIT*

INIT do innego punktu końcowego (SCTP B). Fragment INIT (rys. 9) zawiera wartość znacznika inicjującego (*Initiate Tag*), która jest liczbą losową z zakresu od 1 do 4294967295 (ciąg 32-bitowy). Zostanie on, w kolejnym kroku, użyty przez odbiorcę - punkt końcowy SCTP B, we fragmencie INIT ACK Chunk, jako znacznik weryfikacyjny (*Verification Tag*), do potwierdzania wszystkich pakietów w ramach zrealizowanego powiązania - asocjacji. Może on być również używany jako początkowy numer TSN. Drugi punkt końcowy, SCTP B, generuje i przesyła (w ramach INIT ACK) stanowy plik cookie. Punkt końcowy SCTP A odpowie wtedy fragmentem COOKIE ECHO, który może również zawierać fragmenty DANYCH (*DATA chunks*). W reakcji na to punkt końcowy SCTP B potwierdza odbiór fragmentu COOKI ECHO fragmentem COOKIE ACK, który może być również powiązany z innymi fragmentami DANYCH. Aplikacja (ULP) każdego z punktów końcowych zostanie następnie powiadomiona o pomyślnym ustanowieniu powiązania.

W ramach fragmentów INIT i INIT ACK, przesyłanych przez punkty końcowe, uzgadniana jest liczba strumieni wychodzących i przychodzących zaakceptowanych przez każdy z nich. Punkty końcowe przyjmą przy tym wartości minimalne (liczby strumieni) spośród liczby preferowanych strumieni wychodzących z danego punktu końcowego i możliwych do odbioru (czyli przychodzących) po stronie drugiego punktu końcowego. Wszystkie strumienie w SCTP są jednokierunkowe.

Podawana jest również, w ramach ogłaszanego okna odbiorczego, wielkość kredytowanego „przekazu".

Jak wspomniano powyżej w procesie tworzenia powiązania istotne znaczenie odgrywa znacznik inicjujący (ciąg 32-bitowy), który jest następnie używany przez odbiorcę do potwierdzania wszystkich pakietów w polu znacznika weryfikacyjnego dla ustanowionego powiązania - asocjacji. Zmienna inicjująca może przyjmować dowolną wartość (ze wspomnianego przedziału) z wyjątkiem 0. W przypadku przyjęcia wartości 0, odbiornik musi zareagować fragmentem ABORT.

Fragment INIT zawiera też pole z opcjonalnymi parametrami. Możliwych jest wiele sposobów wypełnienia tego pola.

Jeden ze „scenariuszy" przewiduje podanie dodatkowych adresów IPv4 lub IPv6 - związanych z wykorzystywanymi przez dany punkt końcowy interfejsami. Adresu w tym polu nie podajemy, gdy nadawca ma tylko jeden interfejs, z którym można się skontaktować i z którego powinny być przesyłane DANE w ramach asocjacji .

Po skonfigurowaniu połączenia możliwe jest, że SCTP będzie korzystał z opcji segmentacji swoich fragmentów/porcji (*chunks*), aby w szczególności uniknąć niedozwolonej fragmentacji adresów IP. Ewentualna fragmentacja (segmentacja) jest dokonywana tylko przez urządzenie źródłowe, a punkt końcowy dokonuje ponownego złożenia podzielonych fragmentów po ich odebraniu.

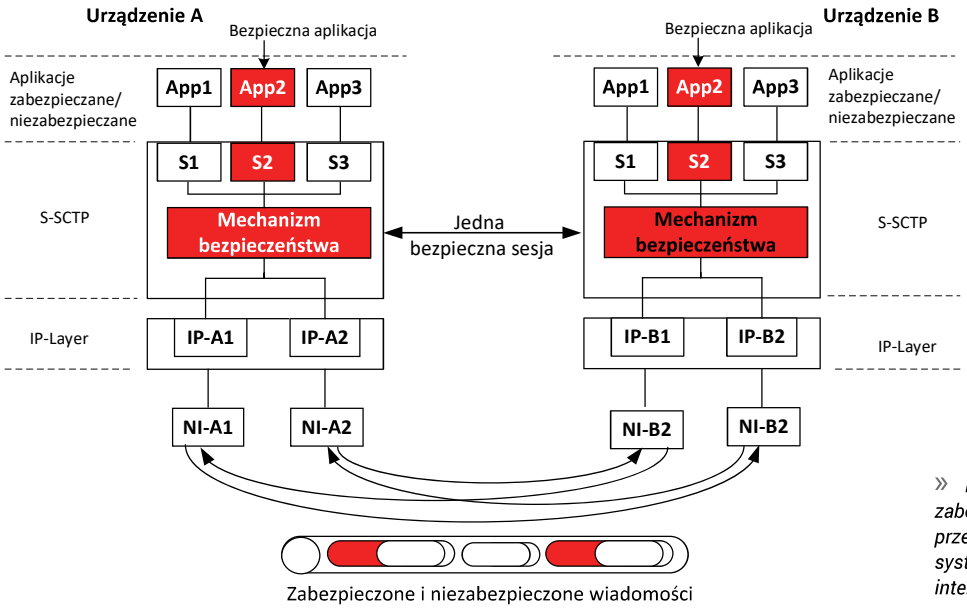
Kryptograficzne zabezpieczanie w SCTP

W procesie zestawiania połączeń ważną rolę odgrywają też procedury wzajemnego uwierzytelniania oraz zabezpieczania kryptograficznego. Procedury te nie są jednakże opisane w samym SCTP. Podobnie jak nie przewiduje ich protokół TCP.

Zakłada się zatem, że realizacja tych funkcji, czyli gwarancji bezpieczeństwa end-to-end będzie realizowana bądź poniżej SCTP, czyli za pośrednictwem IPsec [21], bądź też powyżej SCTP, czyli z wykorzystaniem TLS [22].

Ta druga koncepcja została zaprezentowana, w sposób dopasowany do potrzeb SCTP w RFC 3436 [23]. *Transport Layer Security over Stream Control Transmission Protocol* opisuje użycie protokołu TLS przez SCTP. Protokół TLS został oryginalnie zaprojektowany do działania nad protokołem transportowym, zorientowanym na przesył strumienia bajtów, zapewniającym niezawodne i uporządkowane dostarczanie danych. Dlatego też TLS jest używany głównie jako dodatek do protokołu TCP.

W przypadku TLS over SCTP, rozszerzenie to wykorzystuje jedną sesję TLS na strumień. Potencjalnie prowadzi



» *Rys. 10. Ilustracja częściowego zabezpieczania fragmentów (chunks) przesyłanych w pakietach SCTP pomiędzy systemami końcowymi wyposażonymi w dwa interfejsy (dual-homed) (wg [24])*

to do problemów z wydajnością, gdy powiązanie-asocjacja obejmuje wiele strumieni. Każda wiadomość musi być bowiem zabezpieczona protokołem TLS przed jej wysłaniem do SCTP. Tym samym, jeżeli aplikacja wysyła wiele krótkich wiadomości, każda z nich jest osobno zabezpieczona. Powoduje to większy narzut w porównaniu z rozwiązaniem, gdyby zabezpieczano kompletny pakiet SCTP, zawierający kilka powiązanych wiadomości. Dodatkowo, ponieważ każdy rekord TLS zależy od stanu poprzedniego rekordu, usługa SCTP związana z przekazem wiadomości bez zapewniania kolejności dostarczenia (unordered delivery service) nie będzie przez TLS obsługiwana. W modelu TCP/IP protokół TLS znajduje się nad warstwą transportową, więc nie może też chronić fragmentów (chunks) kontrolnych SCTP ani wspólnego nagłówka pakietu SCTP, gdyż są one dodawane po przekazaniu danych przez TLS do SCTP.

Zaletą TLS over SCTP, w porównaniu z SCTP over IPsec jest, że rozwiązanie to może efektywnie mieszać ruch zabezpieczony i niezabezpieczony w ramach jednego powiązania SCTP. Użytkownik TLS może również w pełni korzystać z funkcji multihomingu i rozszerzeń bezpieczeństwa w warstwie IP.

Użycie protokołu SCTP w powiązaniu ze standardowymi protokołami bezpieczeństwa (TLS lub IPsec) prowadzi do znacznych ograniczeń i potencjalnych nieefektywności. Ani TLS, ani IPsec nie obsługują przy tym wszystkich funkcji SCTP. Z uwagi na wielostrumieniowość w górnych warstwach dostępu do usług i *multihoming* w warstwach dolnych dostępu do usług, można stwierdzić, że brak w pełni zintegrowanego rozwiązania rodzi szereg niedogodności, w tym potencjalnie nieefektywne działanie (w większości scenariuszy). Dlatego w literaturze można znaleźć szereg kompleksowych, w pełni zintegrowanych z SCTP rozszerzeń bezpieczeństwa [24], [25], [26]. W propozycji sformułowanej w [24] i nazywanej Secure-SCTP, bezpieczna sesja S-SCTP jest inicjowana po ustanowieniu normalnego powiązania - asocjacji SCTP. Jeśli jeden punkt końcowy nie obsługuje rozszerzenia S-SCTP lub konfiguracja bezpiecznej sesji nie powiedzie się, np. z powodu błędnych certyfikatów aplikacja może zdecydować, czy chce skorzystać z niezabezpieczonej asocjacji, czy też zamknie takie powiązanie. Podstawowa koncepcja rozwiązania S-SCTP polega na tym, że powiązanie ma tylko jedną wspólną bezpieczną sesję dla wszystkich strumieni danych w przypadku *multistreamingu* i dla wszystkich adresów w scenariuszu *multihomingu*. Aby to osiągnąć, mechanizm bezpieczeństwa musi zapewnić integrację pomiędzy górnym blokiem funkcjonalnym SCTP, który wykonuje grupowanie fragmentów SCTP w pakiety SCTP (*bundling*), a dolnym blokiem funkcjonalnym, który dokonuje wyboru ścieżek sieciowych, poprzez wybór adresu docelowego dla wysyłanego pakietu SCTP, jak pokazano na rys. 10.

Użycie protokołu SCTP w powiązaniu ze standardowymi protokołami bezpieczeństwa (TLS lub IPsec) prowadzi do znacznych ograniczeń i potencjalnych nieefektywności.

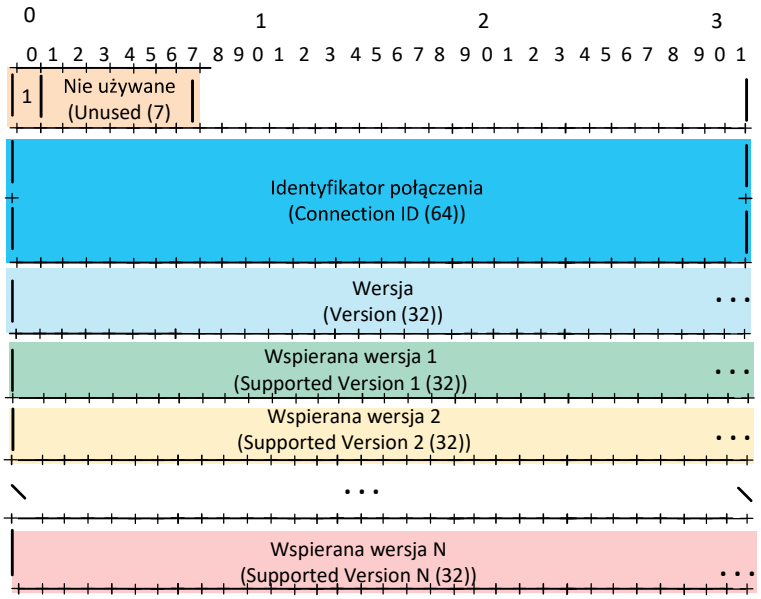
Inną ważną i standaryzowaną metodą zabezpieczania pakietów SCTP jest *Datagram Transport Layer Security over Stream Control Transmission Protocol* (DTLS over SCTP). Rozwiązanie to zdefiniowano w RFC 6083 [27].

Uzupełnioną wersję DTLS over SCTP opisano, jako rozwiązanie ulepszone i alternatywne, w dokumencie [28]. Wersja ta zapewnia wzajemne uwierzytelnianie, poufność przekazu, ochronę integralności i częściową ochronę przed powtórzeniami dla aplikacji korzystających z protokołu SCTP i pozwala aplikacjom klient/serwer zapewnić prywatność komunikacji oraz zapobiegać podsłuchiwaniu, a także wykrywać manipulacje lub fałszowanie wiadomości. Ochrona DTLS jest realizowana w dodatkowej warstwie adaptacji DTLS/SCTP poniżej API do komunikacji z aplikacją (i z przekazem parametrów bezpieczeństwa), a nad SCTP API do warstwy z implementacją SCTP i z obsługą SCTP-AUTH (*Authenticated Chunks for the Stream Control Transmission Protocol*) [29].

Zestawianie połączenia QUIC

Typowe połączenie QUIC rozpoczyna się od negocjacji wersji protokołu, używanej przez oba punkty końcowe. Pakiet początkowy z tzw. „długim" nagłówkiem jest wysyłany przez klienta, aby ustalić, czy oba punkty końcowe obsługują tę samą wersję. W przypadku, gdy wersja klienta nie jest akceptowana przez serwer, wysyła on pakiet Version Negotiation i proces uzgadniania jest kontynuowany. Strukturę pakietu QUIC Version Negotiation ilustruje rys. 11.

Po uzgodnieniu wersji protokołu przez oba punkty końcowe rozpoczyna się faza transportu, poprzedzona procedurą wzajemnego uwierzytelniania i uzgadniania zabezpieczeń kryptograficznych (*cryptographic handshake*).



» *Rys. 11. Pakiet QUIC typu Version Negotiation*

Kryptograficzne zabezpieczenia end-to-end w QUIC

W dzisiejszym Internecie „skończyło się" wzajemne zaufanie i nieco naiwne podejście, związane z przekonaniem, że wszystkie strony wymiany zachowują się fair. Aplikacje nie ufają już innym aplikacjom. Nie ufają platformie, na której znajduje się aplikacja, ani współdzielonym bibliotekom, które implementują podstawowe funkcje. Nie chcą też czekać, aż platforma będzie obsługiwać nowe funkcje protokołów transportowych. Aplikacje nie mają już zaufania do sieci, że zachowa ona ich tajemnice. Coraz więcej funkcji i usług jest włączanych z powrotem do aplikacji, a to, co jest przekazywane z aplikacji, jest w miarę możliwości ukrywane pod osłoną prywatności (poprzez szyfrowane).

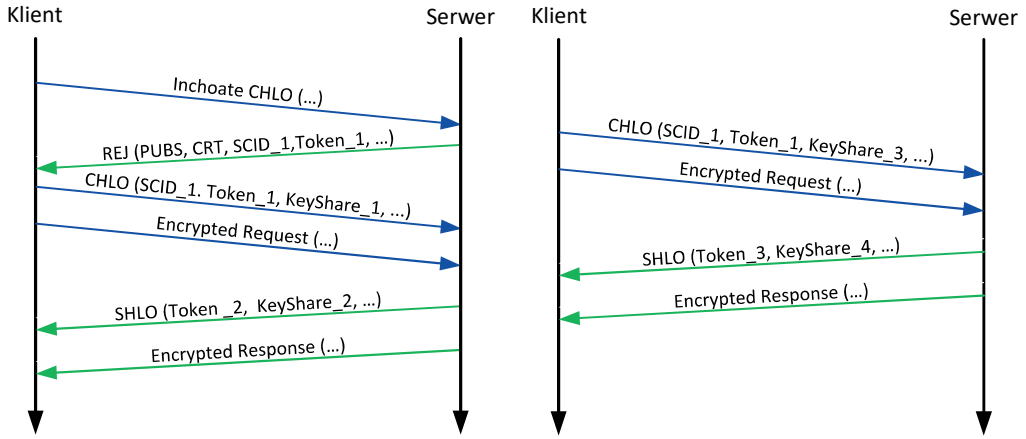
Tym samym utworzenie bezpiecznego połączenia do przekazu danych w zawodnej sieci Internet jest jednym z podstawowych wymagań. Realizacja procedur uwierzytelniania i ustalenia wspólnego klucza kryptograficznego wiąże się jednak ze znacznymi opóźnieniami, ze względu na konieczną wymianę stosunkowo dużej liczby komunikatów protokolarnych. Czas ten liczony do pierwszego przekazu danych z pełną ochroną kryptograficzną (pochodzących z aplikacji), wyrażany jest zwykle za pomocą parametru RTT (*Round-Trip-Time*). Nawet bardzo wydajne protokoły uwierzytelniania wymagają do tego wymiany co najmniej dwóch komunikatów - i tym samym czasu określanego jako 1-RTT. Klasyczny protokół TCP oddziela wymianę parametrów bezpieczeństwa i transportu, wymagając co najmniej 2-RTT do ustanowienia bezpiecznego połączenia [30].

Pod tym względem QUIC jest wyjątkowo efektywny. Oferuje on dwa tryby konfiguracji połączeń: z tzw. zerowym czasem RTT (0-RTT) oraz z pojedynczym czasem RTT (1-RTT) - z pełną procedurą uzgadniania (*handshake*) parametrów połączeń wraz z ustaleniem współdzielonego klucza kryptograficznego, jak pokazano na rysunku 12.

Pełne uzgadnianie, określane jako procedura typu 1-RTT, jest wymagane dla początkowego (pierwszego) połączenia pomiędzy klientem i serwerem.

Uzgadnianie takie rozpoczyna się od wysłania przez klienta komunikatu wstępnego, oznaczonego na rys. 12a, jako *Inchoate Client Hello*. Zawiera on losowo wygenerowany identyfikator połączenia (*ConnID*), który jest używany dalej przez obie strony w odniesieniu do tego połączenia. *ConnID* jest częścią publicznego nagłówka pakietu QUIC.

Identyfikator ten odgrywa bardzo ważną rolę, gdyż umożliwia np. migrację połączenia na nowy adres IP i/lub port punktu końcowego. Staje się to konieczne po ponownym



» *Rys. 12. Ilustracja procedur zestawiania bezpiecznych połączeń w QUIC. a) zgodnie z trybem 1-RTT, b) zgodnie z trybem 0-RTT. Oznaczenia: CHLO - Client Hello, SHLO - Server Hello (wg [31])*

powiązaniu z NAT lub gdy do punktu końcowego zostanie przypisany nowy adres IP [31].

W odpowiedzi, serwer przesyła komunikat odrzucenia (REJ - *Reject*) zawierający dane wymagane do uwierzytelnienia tożsamości serwera i ustanowienia bezpiecznego połączenia, a także token do uwierzytelnienia publicznie widocznego adresu IP klienta.

Przy kolejnych połączeniach między daną parą klient-serwer może być użyty tryb skrócony 0-RTT. Właśnie taki tryb, 0-RTT, opracowany dla wersji TLS 1.3 [22] zaproponowano w QUIC. Umożliwia on dwóm stronom szybkie ustanowienie wspólnego klucza kryptograficznego i utworzenie bezpiecznego kanału do transmisji.

Jest to możliwe z wykorzystaniem informacji zapamiętanych (*cached*) z poprzednich połączeń. To zachowanie jest sygnalizowane ponownym użyciem, w komunikacie Client Hello (CHLO) identyfikatora konfiguracji serwera (SCID_1) i tokena (*cookie*) adresu źródłowego (STK - *Source Address Token_2*) z pierwszego połączenia (pokazanego na rysunku 12a), przy kolejnym (np. drugim) konfigurowaniu połączenia (patrz rysunek 12b). Jest to parametr kluczowy przy „eliminacji" dodatkowego opóźnienia RTT.

Parametr SCID opisuje 16-bajtowy identyfikator, który umożliwia obiektom odwoływanie się do określonej konfiguracji serwera, podczas gdy token opisuje zaszyfrowany i uwierzytelniony blok, który jest nieprzezroczysty dla klienta. Token zawiera publicznie widoczny adres IP klienta i znacznik czasu widziany przez serwer (por. rys. 12b).

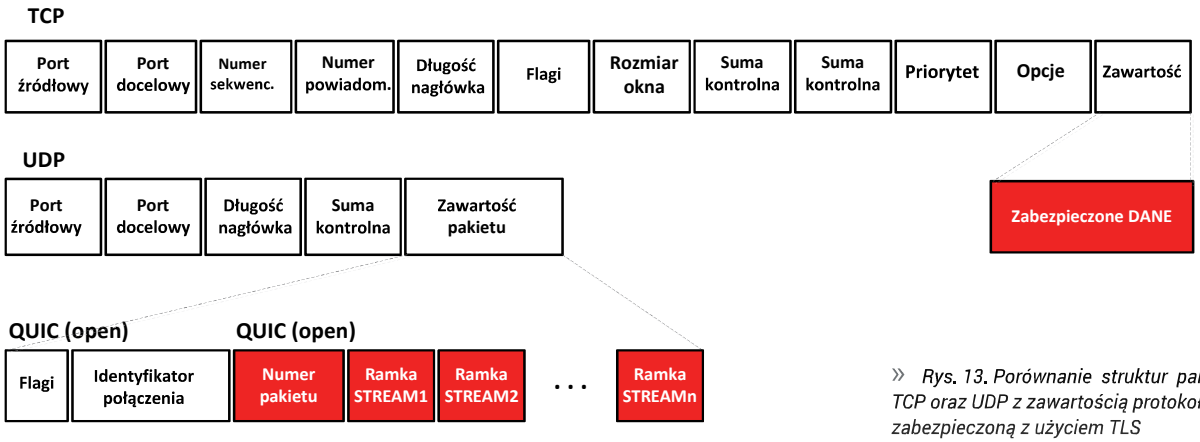
Należy jednak dodać, że, wykorzystanie procedury 0-RTT wiąże się z pewnym ryzykiem, w szczególności może dojść do przechwycenie, przez atakującego, danych z aplikacji. Mając to na uwadze część aplikacji korzystających z QUIC domyślnie nie umożliwia wznawiania połączenia w trybie 0-RTT [32].

Podczas procedury kryptograficznego uzgadniania każdy punkt końcowy anonsuje parametry transportowe, które definiują parametry początkowe dla połączenia. Parametry te obejmują wartości, opisujące maksymalną ilość danych, które mogą być przesyłane w ramach każdego strumienia, a także maksymalną ilość danych przypadającą na połączenie. Wartości te są aktualizowane przez cały okres istnienia połączenia, aby ułatwić kontrolę przepływu. Po nawiązaniu połączenia strumienie podrzędne można tworzyć w dowolnym momencie okresu istnienia połączenia. QUIC obsługuje przy tym jednokierunkowe i dwukierunkowe strumienie podrzędne.

Jak jasno wynika z odwołań do TLS, QUIC używa szyfrowania typu *end-to-end*. Szyfrowanie to jest wykonywane praktycznie na pełnej zawartości pakietu UDP. Tym samym po zakończeniu procedur kryptograficznych przez TLS niewiele z późniejszych pól pakietów QUIC jest pozbawionych takich zabezpieczeń (rys. 13). To co pozostaje niezabezpieczone w QUIC, to flagi publiczne i identyfikator połączenia. Zgodnie z wcześniejszymi opisami identyfikator umożliwia odbiorcy powiązanie pakietu z punktem końcowym bez odszyfrowywania jego całej zawartości. Część pakietu QUIC bez ochrony obejmuje również zestaw flag publicznych, używanych, między innymi podczas ustanawiania sesji QUIC, która to część może być później pominięta.

Pozostała część pakietu QUIC obejmuje prywatne pola kontrolne i zawartość (*payload*). Są one zaszyfrowane i nie są bezpośrednio dostępne dla np. podsłuchującego użytkownika. Ta prywatna sekcja zawiera w szczególności numer sekwencyjny pakietu. Pole to służy do wykrywania zduplikowanych i brakujących pakietów. Zabezpieczona część obejmuje również wszystkie parametry wymagane do kontroli przepływu, w tym «ogłaszaną" przez punkt końcowy wielkość okna odbiorczego.

Jest to jedna z najważniejszych i niewątpliwie krytycznych różnic między TCP i QUIC. W przypadku protokołu TCP kontrolne części protokołu są niezabezpieczone, między innymi po to by elementy pośredniczące w sieci mogły sprawdzać adresy portów (i wnioskować o typie aplikacji), a także o stanie przepływu (flow control) w ramach połączenia. Obserwacja sekwencji pakietów TCP, nawet biorąc pod uwagę jedynie pakiety przesyłane w jednym kierunku w ramach połączenia, pozwala urządzeniom pośredniczącym w sieci (w tym nieuprawnionym) na ocenę czasu transmisji w obie strony (RTT) i szybkości/intensywności transmisji danych. Podobnie, jak w przypadku NAT, daje to potencjalną możliwość zmian (ale i manipulowania) parametrów przekazu, w tym rozmiaru okna odbiorczego, czy strumienia powiadomień ACK i w ten sposób ograniczać przepływ, zmniejszając szybkość transferu, w sposób niewidoczny dla obu punktów końcowych. Umieszczenie wszystkich tych informacji kontrolnych w zaszyfrowanej części pakietu QUIC zapewnia,



» *Rys. 13. Porównanie struktur pakietów TCP oraz UDP z zawartością protokołu QUIC - zabezpieczoną z użyciem TLS*

że żaden element sieci nie ma bezpośredniego wglądu do tych informacji i tym samym żaden element sieci nie może manipulować przepływem w połączeniu.

Można więc przyjąć, że QUIC narzuca perspektywę przyjętą w latach 80. oznaczającą, że protokół transportowy typu end-to-end nie jest w zasadzie współdzielony z siecią. Wszystko, co widzi sieć, to datagramy bezstanowe, a punkty końcowe mogą bezpiecznie zakładać, że informacje zawarte w polach kontroli transportu typu end-to-end są przesyłane przez sieć w sposób chroniący je przed inspekcją (wglądem do zawartości) i zmianami przez strony trzecie.

STRUMIENIE W SCTP ORAZ QUIC QUIC ([8], [9], [10], [12], [13])

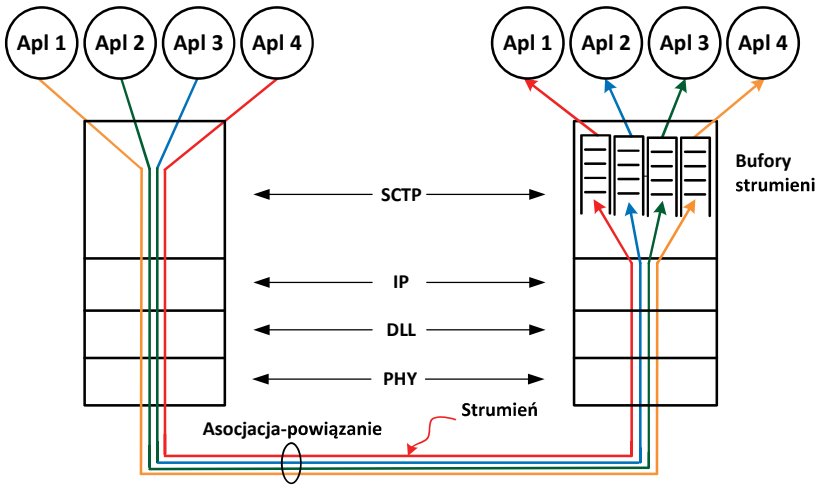
Jedną z głównych cech obu omawianych protokołów jest multipleksowanie strumieni. Problem z TCP polega na tym, że utracony pakiet może blokować dostarczanie do aplikacji zawartości danych dla wszystkich (pod)strumieni. Wynika to z faktu, że TCP używa jednobajtowej abstrakcji strumienia. Oznacza to przesyłanie wszystkich danych zgodnie z kolejnością bajtów, sygnalizowaną w polu Sequence Number. SCTP i QUIC rozwiązują ten problem, obsługując strumienie multipleksowane w samym protokole (i pakietach danych). Jeśli pakiet, któregoś z tych protokołów, przenoszący zawartości dla wybranych strumieni, zostanie odrzucony, to kolejne pakiety przenoszące inne strumienie bajtów, nie muszą być blokowane, a ich zawartości mogą być, zgodnie ze swoją kolejnością nadal dostarczane do aplikacji (bez blokady).

W SCTP dane pochodzące ze strumieni generowanych przez różne procesy po stronie aplikacji przesyłane będą w pakietach w postaci fragmentów (*chunks*). Kolejne transmisje pakietów będą posiadały identyfikatory TSN (*Transmission Sequence Number*) o sekwencyjnie narastających wartościach. Każdy pakiet zawierać przy tym może fragmenty danych z różnych strumieni, identyfikowanych (niezależnie od sekwencyjnego identyfikatora transmisji) przez identyfikatory strumieni (SID - *Stream ID*). Jeżeli fragment danych z jednego strumienia zostanie utracony, dane z innych strumieni nadal powinny być odbierane i przekazywane do aplikacji. Sekwencyjny numer transmisji wykorzystywany będzie zarówno w procedurach sterowania przepływem, jak też szybkiej retransmisji po utracie pakietu (ew. ich większej liczby). Największa wartość TSN, opisująca poprawny odbiór pakietu, będzie potwierdzana zwrotnie w postaci tzw. skumulowanej wartości ACK (*Cummulative ACK*). Jeżeli pakiet zostanie utracony, wszystkie pakiety z TSN wyższym, niż utracony, nie mogą zostać „w pełni" zaakceptowane, dopóki ten utracony nie zostanie ponownie przesłany i zaakceptowany. Informacje o brakujących pakietach, ale jednocześnie ze wskazaniem na kolejne poprawnie odebrane są przesyłane we fragmentach typu SACK - z selektywnym powiadamianiem.

Podobnie jest w przypadku QUIC, gdzie przyjmuje się dwa poziomy numeracji. Dane użytkownika są jednoznacznie identyfikowane za pomocą identyfikatora strumienia i przesunięcia (*offset*), a także monotonicznie rosnącego numeru sekwencyjnego pakietów. Numer sekwencyjny pakietów jest używany do kontroli przeciążenia i wykrywania strat oraz numeruje wszystkie pakiety (SCTP nie numeruje pakietów kontrolnych - z zawartością sterującą). Selektywne powiadamianie w QUIC, potwierdza numer sekwencyjny ostatniego odebranego pakietu, a QUIC retransmituje utracony pakiet przy użyciu nowego numeru sekwencyjnego. W rezultacie okno przeciążenia może otworzyć się na więcej pakietów, a utracony pakiet nie wpływa na odbiór pakietów następujących po nim, unikając w ten sposób problemu blokowania HoL. Pomimo, że ramka QUIC SACK wskazuje pakiety, które zostały utracone, to protokół QUIC nie retransmituje utraconych pakietów z tymi samymi numerami. Tym samym w procedurach QUIC musi być zachowane wewnętrzne odwzorowanie, która ramka (*frame*) strumienia i w jakim pakiecie uległa utracie, aby wiedzieć, którą (które) z nich należy ponownie przesłać, co oczywiście wprowadza dodatkowe przetwarzanie.

Strumienie w SCTP

Konfiguracja połączenia SCTP wymaga zadeklarowania liczby strumieni oraz ich identyfikatorów w momencie tworzenia asocjacji - powiązania. SCTP nie ma funkcji uruchamiania strumieni w trakcie asocjacji, ponieważ nie rozróżnia strumieni inicjowanych przez klienta i inicjowanych przez serwer. Poglądowa ilustracja asocjacji SCTP, strumieni oraz ich obsługi w urządzeniu końcowym pokazana jest na rys. 14. Strumienie utrzymują się zatem, bez zmian, przez cały okres istnienia aktywnego powiązania. Fragment INIT SCTP (por. rys. 9) deklaruje dwa pola - z liczbą strumieni wychodzących (OS) i liczbą strumieni przychodzących (NIS) (odpowiednio: *Outbound Streams* (OS) i *Number of Inbound Streams* (NIS)), aby efektywnie wynegocjować liczby strumieni, które mają zostać utworzone



» *Rys. 14. Poglądowa ilustracja asocjacji SCTP, strumieni oraz ich obsługi w urządzeniu końcowym (na podstawie: [33])*

podczas powiązania. Liczba strumieni nie jest przy tym negocjowana w tradycyjnym sensie. Zamiast tego przyjmowana jest minimalna liczba spośród strumieni żądanych i strumieni oferowanych. Na przykład, jeżeli odbiorca anonsuje 5 strumieni w swoim polu NIS, a nadawca anonsuje 10 strumieni w swoim polu OS, liczba strumieni do odbioru wyniesie 5. Jeżeli jedno z tych pól zostanie ustawione na 0, powiązanie będzie usunięte. W sytuacji, gdy nadawca zostanie ograniczony do mniejszej liczby strumieni niż żądał, może poinformować swoją warstwę aplikacji, że nie udało się zabezpieczyć wymaganej liczby strumieni, a aplikacja może zdecydować o kontynuowaniu lub przerwaniu powiązania. Strumienie SCTP są tylko jednokierunkowe, a numer sekwencyjny każdego strumienia musi zaczynać się od 0. Ponieważ strumienie są tworzone podczas konfiguracji powiązania, w przypadku konieczności zmiany liczby strumieni powiązanie należy usunąć i ponownie skonfigurować.

Strumienie w QUIC

QUIC może nie tylko obsługiwać wiele różnych „profili" strumieniowych, ale także może obsługiwać różne profile strumieniowe w ramach jednej kompleksowej sesji QUIC. Nie jest to oczywiście nowa koncepcja, a protokół HTTP/2 jest tego dobrym przykładem na poziomie aplikacji, pozwalając na multipleksowanie i „ramkowanie" strumieni w celu przenoszenia wielu przepływów danych w pojedynczym strumieniu danych transportowych. Jednak pojedynczy strumień transportowy TCP, używany przez HTTP/2, może napotkać blokowanie HoL, w wyniku czego wszystkie strumienie danych „doświadczają wspólnego opóźnienia" w pojedynczej sesji TCP. Jeżeli jeden ze strumieni się zablokuje, to w praktyce wpłynie to na wszystkie strumienie danych zawarte w przesyłanych segmentach, które również ulegną wstrzymaniu.

Jednym z powodów multipleksowania wielu przepływów danych między tymi samymi dwoma punktami końcowymi w protokole HTTP/2 było zmniejszenie nakładu pracy związanego z konfiguracją zabezpieczeń TLS dla każdej sesji TCP. Może to stanowić poważny problem, gdy każdy z poszczególnych strumieni wysyła małe „obiekty" - ciągi danych, co może skutkować tym, że komponent uzgadniania TCP i TLS w pobieraniu złożonego obiektu internetowego dominuje zarówno w całkowitym czasie pobierania, jak i wielkości przesyłanych danych.

QUIC pozwala na nieco inną formę multipleksowania, w której każdy strumień danych w pakiecie może wykorzystywać swój własny stan przepływu od końca do końca, a przerwa w jednym strumieniu nie powinna wpływać na jakikolwiek inny równocześnie obsługiwany i przesyłany strumień.

QUIC ma funkcję uruchamiania lub wstrzymywania strumienia w dowolnym momencie połączenia (oprócz zerwania połączenia). Parzystość identyfikatora strumienia umożliwia QUIC tworzenie nowych strumieni po stronie klienta lub serwera bez konieczności przeprowadzania negocjacji między każdą ze stron w celu ustalenia identyfikatora. Ponieważ o parzystości decyduje najmniej znaczący bit w polu identyfikatora, klient wybiera tylko parzyste identyfikatory strumienia, a serwer wybiera tylko nieparzyste identyfikatory strumienia. Strumienie podrzędne QUIC mogą obsługiwać przekazy (strumienie) jednokierunkowe i dwukierunkowe, co jest określane przez ustawienie drugiego najmniej znaczącego bitu w identyfikatorze strumienia. Bity rezerwacji umożliwiają uruchamianie strumieni w dowolnym momencie w czasie trwania połączenia bez konieczności negocjacji. Każdy punkt końcowy jest ograniczony do własnego, nienakładającego się zakresu identyfikatorów, co eliminuje potrzebę negocjowania identyfikatora w celu uniknięcia konfliktów. Protokół definiuje kilka parametrów transportowych i typów ramek do precyzyjnego określenia i dalszego kontrolowania zachowania strumieni. Maksymalny rozmiar możliwych do transmisji danych (w ramach określonego strumienia - opisanego przez Stream ID) jest przesyłany w ramce MAX_STREAM_DATA i ogłaszany przez odbiornik w celu płynnego sterowania przepływem poszczególnych strumieni. Jeśli urządzenie spróbuje wysłać więcej danych niż jest dopuszczone przez odbiorcę, połączenie zostanie przerwane. MAX_STREAM_ID to podobna ramka przesyłana przez odbiornik w celu wskazania maksymalnej liczby strumieni dozwolonych w połączeniu. Jeśli jeden z hostów zainicjuje strumień z identyfikatorem strumienia wyższym niż ogłoszone maksimum, połączenie zostanie zakończone. Nadawca może poinformować odbiorcę, że nie jest w stanie wysłać więcej danych lub rozpocząć nowy strumień - korzystając z ramek STREAM_BLOCKED i STREAM_ID_BLOCKED.

Odbiorca może też ogłaszać nowe limity danych dla całego połączenia i strumieni w dowolnym momencie połączenia i jest zobowiązany do przestrzegania tych ograniczeń, np. odbiorca nie może ogłosić wyższego limitu i odrzucić go, gdy nadawca rozpocznie nadawanie. Ważną uwagą dotyczącą implementacji jest to, że w przypadku, gdy pakiet QUIC zostanie odrzucony, każdy strumień znajdujący się w tym pakiecie zostanie zablokowany. Do implementacji QUIC należy określenie liczby strumieni wysyłanych w pakiecie, aby ograniczyć występowanie blokowania HoL. QUIC musi zatem równoważyć, lub wręcz optymalizować wysyłanie danych do wszystkich swoich „ujść" licząc się z możliwością zablokowania strumienia, gdy zostanie odrzucony pakiet. Gdy punkt końcowy strumienia zakończy przesyłanie swoich danych, to ustawia bit FIN w swojej ostatniej ramce STREAM lub w ramce po ostatniej ramce STREAM, aby wskazać, że strumień jest zamknięty w kierunku punktu końcowego, w wyniku czego strumień jest w połowie zamknięty. Gdy oba punkty końcowe wyślą ramki STREAM z ustawionym bitem FIN, strumień zostaje całkowicie zamknięty.

TCP zapewniał aplikacjom abstrakcję niezawodnego uporządkowanego strumienia bajtów (dzięki funkcji *offset* i numeracji: *sequence number*). QUIC zapewnia podobną abstrakcję dla aplikacji, określaną w QUIC jako strumienie. Zasadnicza różnica polega na tym, że TCP implementuje jeden profil, czyli jeden wspólny *payload*, podczas gdy pojedyncza sesja QUIC może obsługiwać wiele profili strumieniowych.

Wspomniane wcześniej strumienie dwukierunkowe realizują transakcje klienta i serwera w „dopasowanym kontekście", tak jak jest to wymagane w przypadku konwencjonalnych transakcji żądania/odpowiedzi np. protokołu HTTP/1. Oczekuje się, że klient otworzy i utworzy dwukierunkowy strumień z serwerem, a następnie wyśle żądanie w strumieniu,

co wygeneruje odpowiednią odpowiedź z serwera. Możliwe jest, że serwer zainicjuje dwukierunkowy strumień typu *push* do klienta, który zawiera odpowiedź bez początkowego żądania. Informacje kontrolne są obsługiwane za pomocą jednokierunkowych strumieni kontrolnych, w których jedna strona może przekazać wiadomość drugiej tak szybko, jak to jest tylko możliwe.

FRAGMENTACJA W PROTOKOŁACH SCTP ORAZ QUIC QUIC ([8], [9], [10], [12], [13])

W klasycznym modelu pracy podsieci komunikacyjnej, korzystającej z protokołu IPv4 zakładano, ze router, który odebrał datagram większy niż MTU na ścieżce do kolejnego routera, miał dwie możliwości: odrzucić datagram, jeżeli w jego nagłówku ustawiony był bit flagi „Nie Fragmentuj" oraz przesłać do źródła komunikat protokołu ICMP (*Internet Control Message Protocol*) wskazujący na stan „Wymagana Fragmentacja", lub podzielić datagram i przesłać go dalej z wymaganym mniejszym MTU.

Założenia przyjęte w protokole IPv6 sprawiają, że routery obsługujące ten protokół nie mają możliwości fragmentacji. Tym samym to na protokołach warstw wyższych spoczywa konieczność dostarczania jednostek danych o wymaganych długościach (urządzenia końcowe IPv6 muszą określić optymalną wartość MTU za pomocą funkcji *Path MTU Discovery*).

W przypadku SCTP, zgodnie z wykorzystywanym tam modelem danych mamy do czynienia z przesyłaniem wiadomości zdefiniowanych przez aplikację. Problemem może więc okazać się przekaz zbyt długich wiadomości, wykraczających poza wymiary MTU. Tym samym segmentacja pakietów, a raczej fragmentów, przenoszących dane z aplikacji okazuje się niezbędna.

Z kolei w modelu QUIC przyjmuje się, analogicznie do protokołu TCP, model dwukierunkowego strumienia bajtów. Takie rozwiązanie ułatwia przenoszenie danych aplikacji z protokołu TCP do QUIC, a ponadto uwalnia QUIC od fragmentacji wiadomości, o którą musi zadbać SCTP.

Fragmentacja w SCTP

Rozwiązanie realizowane przez lata w sieciach IP, zakładające możliwość adaptacji rozmiaru pakietu poprzez jego fragmentację w urządzeniach sieciowych było ważną koncepcją. Nadawca nie musiał być bowiem świadomy ograniczeń, które mogą obowiązywać na ścieżce. Jakakolwiek fragmentacja i ponowne składanie pakietów na poziomie sieci były niewidoczne dla transferu jednostek danych w przekazie od krańca do krańca. Takie podejście nie wydaje się być w dalszym ciągu uzasadnione. W szczególności przyjęcie założenia o niepodzielności datagramów IPv6, sprawia, że to nadawcy muszą zapewnić, że ich pakiety będą mogły docierać do punktów przeznaczenia bez żadnych dodatkowych zmian, czyli bez obsługi fragmentacji w niższych warstwach stosu TCP/IP.

Aby uniknąć podziału datagramów, a w szczególności fragmentacji adresów IP, SCTP segmentuje własne fragmenty (*chunks*), tak aby jego pakiety mieściły się w MTU (*Maximum Transmission Unit*) na wykorzystywanej ścieżce. Ponieważ SCTP bazuje na wiadomościach, jako jednostkach danych, musi określić, w jaki sposób segmentować i ponownie składać przesyłane wiadomości (*payload*), aby zapewnić poprawną i efektywną realizację przekazu. Oznacza to, że musi zachować niefragmentowane nagłówki i obsługiwać ponowne składanie danych po ich odbiorze. SCTP osiąga to poprzez wykorzystanie flag bitowych w nagłówkach fragmentów DATA Chunk i wartości numerycznych w swoich polach TSN i SSN. Bit B ustawiony na 1 wskazuje, że ta „porcja" danych jest pierwszym fragmentem wiadomości użytkownika. Bit E ustawiony na 1 wskazuje, że porcja jest ostatnim fragmentem wiadomości użytkownika.

Jeśli zarówno B, jak i E są ustawione na 1, wiadomość nie jest fragmentowana. Jeżeli zarówno B, jak i E są ustawione na 0, „fragment" jest środkowym elementem wiadomości użytkownika. Pole TSN wskazuje Numer Sekwencyjny Transmisji bloku DANYCH, który jest używany do identyfikacji i potwierdzania pomyślnie odebranych fragmentów. Każdy fragment DANYCH w pakiecie współdzieli sekwencyjny numer TSN oraz stosowny SSN, podczas gdy każdy „posegmentowany" fragment DANYCH (*DATA Chunk*) będzie miał inny sekwencyjny numer TSN, ale ten sam SSN wśród wszystkich fragmentowanych DATA Chunks przenoszących tę samą wiadomość. Odbiorca będzie wtedy w stanie potwierdzić wszystkie otrzymane fragmenty za pomocą odpowiedniego numeru TSN i odtworzyć oryginalne wiadomości, dopasowując fragmenty danych do zawartości o tym samym numerze SSN. Wymaga to oczywiście stosunkowo dużego nakładu na przetwarzanie.

Fragmentacja w QUIC

Pakiet QUIC musi zawierać całe ramki i nie może zawierać ramek podzielonych między pakiety. Pakiet QUIC musi też „dopasowywać zawartość" do MTU ścieżki. QUIC może zmieniać rozmiar pakietów bez potrzeby stosowania skomplikowanych mechanizmów śledzenia fragmentów wiadomości, jak to ma miejsce w SCTP, ponieważ każda zawartość danych - *payload* QUIC jest tylko strumieniem bajtów, a taki strumień można łatwo regulować poprzez zmianę pola przesunięcia bajtów (*offset*) w ramce STREAM. W QUIC nie ma wiadomości do fragmentacji, ponieważ dane są już dostępne w najbardziej „granularnej" (atomowej) formie. Rzeczywisty rozmiar pakietu QUIC jest określany przez implementację protokołu i sposób, w jaki zachowuje się korzystająca z niego aplikacja. Bieżące wersje QUIC nie zawierają zbyt wielu szczegółów na temat zapełniania pakietów QUIC ramkami, poza zaleceniem pakowania jak największej liczby ramek, aby zminimalizować narzut protokolarny przypadający na pakiet i tym samym obniżyć koszt przetwarzania. Musi być jednakże zapewniana swoista równowaga. Jeśli w pakiecie jest zbyt wiele ramek, a pakiet zostanie utracony, wówczas wszystkie związane z nim strumienie będą blokowane. Z kolei jeżeli jest za mało ramek, zwiększa się wspomniany narzut protokolarny na pakiet.

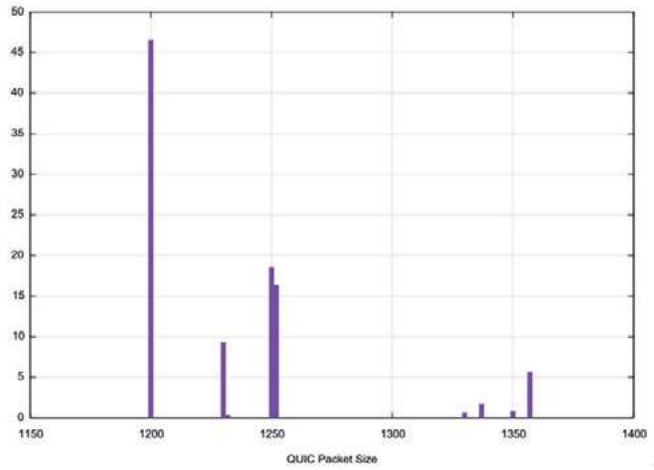
Przyjmuje się, że w przypadku wykorzystania do transmisji poprzez sieć protokołu IPv4 maksymalny rozmiar pakietu QUIC wynosi 1350 bajtów. Dodanie ośmiu bajtów dla nagłówka

UDP, 20 bajtów dla IPv4 i 14 bajtów dla ramki Ethernet (nie wliczając pola CRC - *Cyclic Redundancy Check*) oznacza, że ramka Ethernet przenosząca pakiet QUIC ma rozmiar 1392 bajtów. Nie ma przy tym innego, szczególnego uzasadnienia, ani wyjaśnienia dla wyboru 1350, poza wynikami testów empirycznych w publicznym Internecie [7] .

W przypadku protokołu IPv6 maksymalny rozmiar pakietu QUIC jest zmniejszany o 20 bajtów do 1330 bajtów. Wynikowa ramka ethernetowa, w warstwie dostępowej, ma nadal taki sam rozmiar ze względu na większy nagłówek datagramu IPv6 (nagłówek IPv6 to 40 bajtów – nie uwzględniając pól opcji - chociaż z mniejszą liczbą pól).

Jako ciekawostkę, a jednocześnie uzupełnienie do przedstawionych powyżej danych, przytoczmy informację ze strony [34] prowadzonej przez wielokrotnie cytowanego (w innym kontekście) Goeff'a Hustona.

Jak wynika z opublikowanych tam danych pomiarowych (por. rys. 15), związanych z przekazami QUIC, pakiety o długości 1200 bajtów, zaobserwowano w 46% sesji. Kolejnymi najpopularniejszymi rozmiarami było: 1250 bajtów (18,5% sesji) i 1252 bajty (16,4% sesji). W żadnej sesji nie odnotowano pakietu dłuższego niż 1357 bajtów.



» *Rys.15. Rozkład długości pakietów QUIC (na podstawie [34])*

STEROWANIE PRZEPŁYWEM W SCTP I QUIC QUIC ([8], [9], [10], [12], [13])

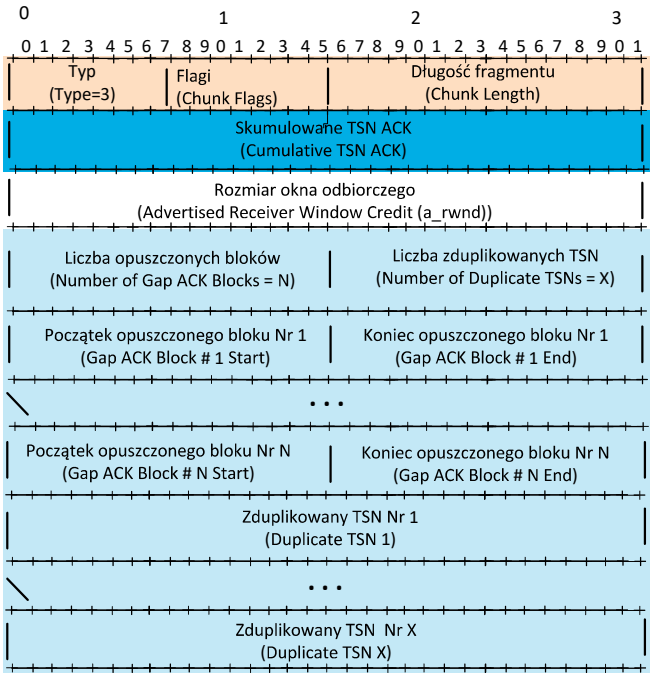
Kontrola przepływu to proces związany z dostosowywaniem intensywności wysyłania danych w zależności od dostępnej przestrzeni buforowej i mocy obliczeniowej w odbiorczym urządzeniu końcowym. Nadawca i odbiorca wymieniają informacje o dostępnych rozmiarach swoich buforów za pośrednictwem wielkości tzw. okien odbiorczych, które wskazują, ile danych mogą obsłużyć przy ciągłym charakterze wymiany (sygnalizują tym samym wielkość „kredytowanego" ruchu). Nadawca dysponujący wiedzą o zasobach pamięci odbiorcy dostosowuje szybkość, czy też intensywność wysyłania danych (rozmiary własnego okna nadawczego) do okna odbiorcy i tym samym unika wysyłania większej ilości danych, niż odbiorca jest w stanie przetworzyć. Przy takiej strategii strona odbiorcza nie jest narażona na odrzucanie napływających danych z powodu przepełnienia bufora.

Kontrola przepływu jest po części związana z kontrolą przeciążenia, ale gdy kontrola przeciążenia koncentruje się na zapobieganiu przeciążeniu (natłokowi) w sieci i szybkiemu reagowaniu na straty pakietów (traktowane zwykle jako symptomy natłoku), to kontrola przepływu koncentruje się głównie na zapewnieniu płynnej realizacji przekazu - wymiany danych. Powszechnie wykorzystywaną koncepcją kontroli przepływu jest przesuwane okno i jego rozmiar, czyli „ogłaszane" okno odbiorcze (*advertised receiver window*), za pośrednictwem, którego punkt końcowy podaje liczbę bajtów, które może przyjąć, a tym samym które może przesłać jego wysyłający odpowiednik. Oba protokoły SCTP i QUIC stosują formę przesuwanego okna. W SCTP, w przeciwieństwie do QUIC, nie ma przy tym kontroli przepływu dla pojedynczego strumienia, a sterowanie przepływem odbywa się dla połączenia (asocjacji).

Sterowanie w SCTP

Sterowanie przepływem w protokole SCTP odbywa się wyłącznie na zasadzie per asocjacja, przy użyciu mechanizmów podobnych do protokołu TCP, zgodnie z klasycznymi definicjami obowiązującymi w TCP *Flow and Congestion Control (*[35],[36]). Gdy odbiorca wysyła fragment SACK (*SACK Chunk*) (por. rys. 16), zawiera on pole o nazwie *Advertised Receiver Window Credit* (a_rwnd). Wartość ta reprezentuje dostępną pojemność w buforze odbiornika.

Ponieważ pakiety SACK mogą być odbierane poza kolejnością, nadawca niekoniecznie zakłada, że ma określoną ilość miejsca w buforze do wysłania. W trakcie zestawiania



» *Rys.16. Fragment SCTP typu SACK*

połączenia - asocjacji przez SCTP każdy punkt końcowy otrzyma wartość a_rwnd (*advertised receiver window size*) w ramach swojego wymienianego dwustronnie fragmentu, czy to INIT (por. wcześniejszy rys. 9), czy też INIT ACK i przyjmie tę wartość jako rzeczywisty wymiar okna odbiorczego (rwnd) odpowiedniego punktu końcowego. W miarę upływu czasu i postępu w realizacji asocjacji - powiązania każdy punkt końcowy będzie odejmował rozmiary fragmentów DANYCH, które są wysyłane lub retransmitowane do współpracującego punktu końcowego, od pierwotnie uzgodnionej wartości rwnd. Dzieje się tak, ponieważ nadawca zakłada, że przestrzeń buforowa partnera zostanie zajęta przez przesłane porcje danych. Każdy punkt końcowy uwzględnia również rozmiar fragmentów DANYCH, które są przygotowane do retransmisji. Z każdym komunikatem SACK, otrzymanym przez punkt końcowy, aktualizuje on swoją wartość okna nadawczego, w szczególności zgodnie z a_rwnd w SACK, pomniejszoną o wszelkie zaległe bajty z brakujących fragmentów, które nie zostały jeszcze potwierdzone.

Sterowanie w QUIC

W nieco inny sposób realizowane jest sterowanie przepływem w QUIC. Odbywa się ono zarówno w odniesieniu do połączenia (sesji), jak i strumienia podrzędnego. Najważniejszymi parametrami kontroli przepływu w QUIC są parametry transportowe MAX_DATA i MAX_STREAM_DATA. Te dwa parametry są wymieniane i uzgadniane podczas zestawiania połączenia, wykorzystując do tego odpowiednie komunikaty kontrolne, które mogą być przesyłane po zestawieniu połączenia. Po zaanonsowaniu wartości ww. parametrów przez jeden punkt końcowy drugi musi je zaakceptować. MAX_DATA wskazuje maksymalną ilość danych, które mogą być przesłane w połączeniu. Z kolei MAX_STREAM_DATA wskazuje, analogicznie, maksymalną ilość danych, które mogą być wysłane w ramach strumienia. Do zadań każdego punktu końcowego należy rozdział napływających danych między jego strumienie. W miarę upływu czasu trwania połączenia i wymiany wiadomości w ramach strumienia, punkty końcowe będą zwykle deklarować wyższe wartości MAX_DATA i MAX_STREAM_DATA, aby kontrolować, a jednocześnie poprawiać efektywność przepływu z/do partnera (rozszerzanie okna nadawczego). Jeżeli jednak którakolwiek z tych zmiennych zostanie „zignorowana" przez nadawcę, w którymkolwiek ze strumieni, całe połączenie zostanie zerwane. Wyjątek dotyczy Strumienia 0, który jest zarezerwowany dla uzgadniania kryptograficznego podczas konfiguracji. Żadne przesłanie „bajtu" w Strumieniu 0 nie jest wliczane do limitów parametrów transportowych. Ponieważ QUIC wykorzystuje paradygmat strumienia bajtów, a przesunięcia bajtów są przekazywane w ramkach STREAM, stopień wykorzystania danych (w ramach okna) można łatwo obliczyć w obu punktach końcowych, rejestrując największe odebrane przesunięcia bajtów (*offset*). W praktyce gwarantuje to brak możliwości złamania uzgodnionego porozumienia przez punkt końcowy, chyba że wystąpił błąd w implementacji lub jest to efektem złośliwego ataku.

NIEZAWODNOŚĆ I KONTROLA PRZECIĄŻEŃ W SCTP I QUIC QUIC ([8], [9], [10], [12], [13])

Kontrola przeciążenia to proces wykrywania i ograniczania stanów nadmiernego obciążenia (natłoku - *congestion*), czy też zatorów (*deadlocks*) w sieci, czego symptomem są pakiety tracone na ścieżkach pomiędzy źródłem, a ujściem danych. Zbyt duże obciążenie w sieci może powodować nie tylko utratę pakietów ale też znaczny wzrost opóźnienia, a w konsekwencji liczne retransmisje - co pogarsza wydajność i jakość usług w sieci. Nadawca i odbiorca wykorzystują różne algorytmy i techniki do monitorowania warunków sieciowych i odpowiedniego dostosowywania szybkości wysyłania. Główne cele kontroli przeciążeniowej to unikanie wspomnianych zatorów, a także sprawiedliwy podział zasobów sieciowych pomiędzy konkurującymi przepływami oraz optymalizacja przepustowości i wydajności sieci, a tym samym gwarantowanie niezawodnego transferu danych.

Zarówno QUIC, jak i SCTP zapewniają niezawodne dostarczanie, a także odpowiednie formy kontroli przeciążenia. SCTP zapożycza większość koncepcji kontroli przeciążeniowej z protokołu TCP, a QUIC wykorzystuje zarówno mechanizmy protokołu TCP, jak i własne [37].

Bloki ACK, w QUIC, wskazują zarówno zakresy numerów pakietów, które zostały poprawnie odebrane (*ACK Range*) - do największego potwierdzonego numeru (Largest Acknowledged) włącznie, jak też informują (w polach GAP) o lukach w odebranych pakietach.

Jest to rozwiązanie nieco odmienne od skumulowanego wskaźnika TSN ACK w SCTP, który w sposób ciągły śledzi najmniejszą potwierdzoną wartość TSN (*Cummulative TSN ACK*).

Obydwa rozwiązania zapewniają selektywne powiadamianie o postępach w realizowanych przekazach, wskazując także na braki pakietów obserwowane w punktach końcowych. Pozwala to na szybkie reagowanie na utraty pakietów oraz efektywne wdrażanie procedur przeciążeniowych.

Kontrola przeciążeń w SCTP

SCTP, jako typowy protokół transportowy zorientowany połączeniowo zapewnia niezawodne dostarczanie danych, w kolejności ich wysyłania, dzięki użyciu identyfikatora TSN. W przeciwieństwie do Numeru Pakietu (*Packet Number*) wykorzystywanego w QUIC, wartości TSN nie muszą narastać monotonicznie. Numery TSN pozwalają na identyfikowanie i potwierdzanie bloków fragmentów przez odbiorcę. Jeżeli nadawca nie otrzyma potwierdzenia w określonym czasie to wie, które fragmenty (*chunks*) należy ponownie przesłać - ze względu na powiązany z nimi numer TSN. Numery TSN są więc używane do śledzenia brakujących fragmentów, a fragmenty są retransmitowane z tym samym numerem TSN, które miały, gdy zostały pierwotnie usunięte – stracone.

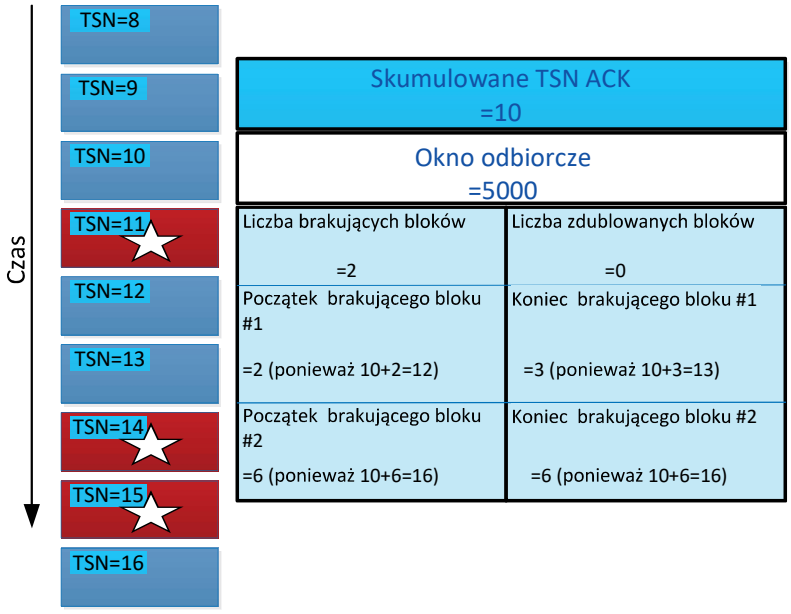
Ponieważ numeracja TSN jest niezależna od SSN (czyli numeracji Fragmentów w danym strumieniu), mechanizm przekazu zgodnego z kolejnością generacji (*in-order delivery*) dla poszczególnych strumieni jest oddzielony od niezawodnego (*reliable*) mechanizmu dostarczania pakietów (segmentów danych). SSN kontroluje dostarczanie danych we właściwej kolejności do ULP (aplikacji), natomiast TSN

kontroluje niezawodne dostarczanie segmentów między punktami końcowymi. TSN jest również niezależny od przenoszonego w segmencie strumienia. Protokół SCTP śledzi też skumulowaną wartość TSN ACK (*Cumulative TSN ACK*), reprezentującą ostatni numer TSN (numer segmentu - PDU), który punkt końcowy otrzymał przed przerwą (np. utratą segmentu) w przekazie kolejnej serii wartości TSN.

Każdy TSN poniżej skumulowanej wartości TSN ACK jest zatem potwierdzony przez docelowy punkt końcowy. Jeśli odbiorca ma luki w numerach TSN, które nie zostały odebrane, potwierdzi tylko te, które poprawnie zaakceptował, pozostawiając nadawcy ustalenie, czego brakuje.

Aby poprawnie realizować te procedury (w tym procedurę szybkiej retransmisji (*Fast Retransmit*)) urządzenie odbiorcze przesyła do nadawcy fragmenty typu *SACK Chunks* (por. rys. 16), potwierdzające otrzymanie kolejnych pakietów z narastającymi numerami TSN. W tym kontekście ciekawy jest sposób wykorzystania pól Fragmentu SACK, czyli Bloku GAP (składającego się z kilku pól), informującego o zauważonych brakach fragmentów i pakietów (zgodnie z numeracją TSN). Jednocześnie informacje te pozwalają na potwierdzanie sekwencji wartości TSN powyżej skumulowanego TSN ACK. Blok GAP wskazuje liczby utraconych TSN (ew. także na pojawiające się duplikaty) oraz zakresy numerów TSN, które nie są potwierdzane przez odbiorcę.

Tym niemniej, nieco zaskakująco Gap ACK Block Start wskazuje segment, o wyższym numerze TSN niż z TSN ACK, potwierdzonym przez SCTP adresata, ale najbliższy tego segmentu, a Gap ACK Block End końcową wartość TSN segmentów odebranych poprawnie - w sposób ciągły (w ramach okna). Na podstawie przesłanej zawartości SACK nadawca identyfikuje brakujące numery TSN i w reakcji na to ponownie przesyła segmenty (PDUs – *Protocol Data Units*) związane z brakującymi numerami TSN. Pozwala to na skokowy wzrost wartości Skumulowanego TSN ACK - przesyłanego przez odbiorcę.



» *Rys.17. Ilustracja wykorzystania poszczególnych pół w bloku GAP we fragmencie typu SACK*

By zilustrować tę procedurę rozważmy sytuację, gdy protokół SCTP A przesyła pakiety z numerami TSN 8, 9 i 10 (por. rys. 17). Po ich odbiorze SCTP B potwierdza ten fakt we Fragmencie SACK ze skumulowaną wartością TSN ACK równą 10. Przyjmijmy dalej, że kolejny pakiet z TSN równym 11 zostaje utracony, natomiast dwa kolejne, tj. z numerami TSN równymi 12 i 13 zostaną przez SCTP B odebrane poprawnie. SCTP B prześle zatem do SCTP A powiadomienie SACK z niezmienioną skumulowaną wartością TSN ACK=10, informując jednocześnie o jednym utraconym pakiecie (stosowne pole we fragmencie), podając w polu GAP ACK, w części Start wartością równą 2 i w polu End wartość równą 3, wskazując tym samym, że segmenty z TSN=12 (10+2) oraz TSN=13 (10+3) zostały odebrane poprawnie. Taka informacja stanowi dla SCTP A wskazówkę, że powinna zostać powtórzona transmisja pakietu z TSN 11. Po jego pozytywnym odbiorze w kolejnym SACK, przesłanym przez SCTP B, pojawi się wtedy skumulowana wartość TSN ACK=13.

W kwestii ogólnej idei można przyjąć, że kontrola przeciążeń w SCTP jest zarządzana przez analogiczne mechanizmy, które wykorzystuje TCP, takie jak powolny start (*Slow Start*), unikanie przeciążenia (*Congestion Avoidance*), szybka retransmisja (*Fast Retransmit*), czy czas retransmisji RTO. Podstawowe zasady kontroli przeciążeń w SCTP zostały sformułowane w dokumencie RFC 2581 [36]. Dodajmy jednak, że kontrola przeciążeniowa w SCTP odnosi się do całego powiązania (asocjacji), a nie do indywidualnych strumieni. Jednocześnie w przypadku korzystania z *multihomingu* (wielu interjfejsów) protokół SCTP będzie utrzymywał oddzielne wartości okien przeciążeniowych cwnd (*Congestion Window*) dla każdego adresu docelowego. Po stwierdzeniu symptomów przeciążenia SCTP przechodzi do fazy *Slow Start*.

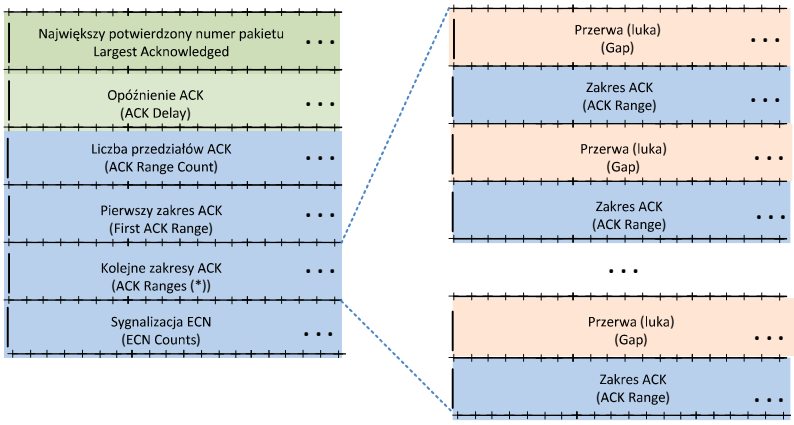
Kontrola przeciążeń w QUIC

QUIC zapewnia integralne i niezawodne dostarczanie danych w kolejności wysyłania dzięki zastosowaniu pola przesunięcia bajtów (*offset*) w ramkach STREAM. Jeśli pakiet zostanie odrzucony, poszczególne ramki zawarte w pakiecie zostaną ponownie przesłane, ale niekoniecznie cały pakiet. Oznacza to, że zostanie utworzony nowy pakiet, z nowym numerem, a utracone ramki zostaną w nim powtórzone - wszystkie ew. ich część (wówczas pozostałe zostaną np. włączone do kolejnego pakietu). Numeracja pakietów QUIC zawsze narasta monotonicznie, czyli innymi słowy, zduplikowany numer pakietu nigdy się nie pojawi, co ułatwia odróżnienie potwierdzeń jednostek retransmitowanych od oryginalnych. Jest to zgodne z koncepcją pełnej kontroli i identyfikacji poszczególnych strumieni. Informacje o utraconych pakietach przesyłane są w QUIC za pośrednictwem ramek ACK, mających cechy i możliwości podobne do fragmentów SACK SCTP. Tym niemniej realizacja procedur retransmisji jest nieco inna. W szczególności w QUIC utracone pakiety QUIC nie są retransmitowane

z tymi samymi numerami, ani też zawartości retransmitowanych pakietów nie muszą obejmować tych samych ramek.

To do implementacji QUIC należy decyzja, ile jednostek danych (pakietów) należy użyć do ponownego wysłania straconych ramek. Ponadto, ponieważ punkty końcowe wiedzą, których wysłanych pakietów i ramek brakuje (informują o tym pola Gap oraz ACK Range - pokazane na rys. 18), wiedzą też jakich bajtów brakuje w uporządkowanym ciągu (*offset*).

Pole *Largest Acknowledged* wskazuje największy numer potwierdzonego pakietu. Pola *ACK Range* wskazują numery pakietów o ciągłej numeracji odebrane przed pakietem z *Largest ACK*. Z kolei pola *Gap* informują o niepotwierdzonych pakietach (pojedynczych, bądź ich seriach)



» *Rys. 18. Format Ramki QUIC ACK*

Pole ACK Delay, w ramce QUIC ACK, wskazuje, mierzony w mikrosekundach, czas odbioru pakietu o największym numerze - *Largest Acknowledged*, co ułatwia dokładny pomiar RTT. Mechanizm sterowania przepływem w QUIC śledzi zawsze najnowszy numer pakietu, przy czym, jak wspomniano wcześniej, numery pakietów rosną monotonicznie, co pozwala na łatwe śledzenie kolejnych transmisji.

Zwróćmy przy tym uwagę na to, że Largest ACK nie odpowiada największemu numerowi pakietu, związanemu z ciągiem wyłącznie pozytywnych decyzji, jak to miało miejsce w przypadku TCP. QUIC odnotowuje tę wartość jako największą bieżącą, ale w polu bądź w polach ACK Range oraz Gap wskazuje na serie pakietów odebranych poprawnie (ale poprzedzających Lagest ACK), a także przedzielające je pojedyncze lub wielokrotne straty pakietów (Gap).

Wartość z pola Gap jest liczbą całkowitą i wskazuje liczbę pakietów w bloku utraconych, czyli niepotwierdzonych pakietów, poprzedzających poprawnie odebrany. Z kolei ACK *Range Length* opisuje długość takiego bloku (z poprawnie odebranymi pakietami). Podobnie jak w przypadku SCTP w QUIC ACK w polach Gap i ACK *Range Length* podawane są względne wartości, z wykorzystaniem, w tym przypadku działań odejmowania, a kolejne bloki ACK Range są uporządkowane malejąco względem Largest ACK, wskazując na coraz niższe wartości numerów pakietów. Blok ACK Range potwierdza właściwy ciąg pakietów, z zakresu poniżej największej wartości potwierdzonej w danej sekwencji (bloku).

W zasadzie bardzo zbliżona procedura wykorzystywana jest w SCTP, przy czym w SACK SCTP śledzi się, odmiennie niż w QUIC, najniższy ciągły TSN zawarty w polu *Cummulative TSN ACK* (SCTP może retransmitować TSN o danym numerze, co nie jest realizowane w przypadku QUIC). Kolejne wartości Gap ACK Start oraz Gap ACK End, w protokole SCTP, wskazują narastająco (ale także z przyjęciem względnych wartości odniesionych do *Cummulative TSN ACK*) brakujące oraz potwierdzone pakiety.

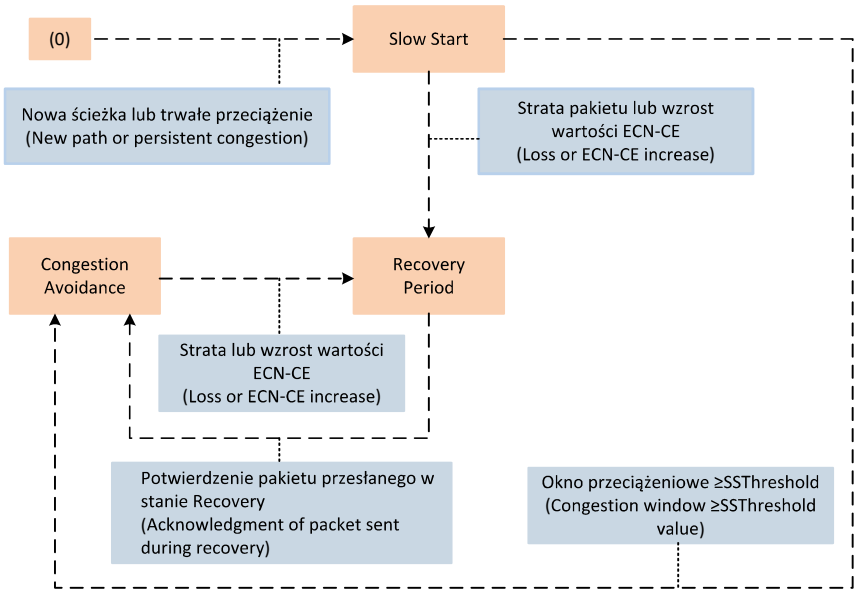
Przy omawianiu SCTP sygnalizowano wykorzystywanie w tym protokole do kontroli przeciążeń mechanizmów analogicznych do TCP. Podobnie jest w QUIC, jednak z pewnymi modyfikacjami. QUIC upraszcza kontrolę przeciążenia i wykrywania strat, rozdzielając swoje źródło informacji, gwarantujące niezawodne (*reliable*) dostarczanie: identyfikator strumienia i przesunięcia bajtów (*offset*), od źródła informacji o sekwencyjności (*in-order delivery*) transmisji: monotonicznie rosnąca numeracja pakietów.

Jak wspomniano każdy z pakietów QUIC może zawierać kilka ramek, z których każdą można uznać za odpowiadającą, do pewnego stopnia, pakietowi IP. QUIC wykrywa utratę, bazując na całych pakietach (co jest odpowiednikiem zbioru pakietów IP) - dla każdego pakietu z potwierdzeniem ACK wszystkie ramki przenoszone w tym pakiecie są uważane za odebrane. Ramki przenoszone w pakiecie uważa się za „bezwzględnie" utracone jeżeli pakiet ten nie zostanie potwierdzony po potwierdzeniu wysłanego później pakietu i po osiągnięciu określonego progu zliczającego kolejne transmisje pakietów, a także przekroczeniu limitu czasu.

Przy omawianiu SCTP sygnalizowano wykorzystywanie w tym protokole do kontroli przeciążeń mechanizmów analogicznych do TCP.

Potwierdzenie QUIC ACK wskazuje, że pakiet wysłany później został dostarczony poprawnie, a przyjęte w mechanizmach kontroli przepływu i przeciążeń progi czasowe zapewniają pewną tolerancję w akceptacji niewłaściwej kolejności pakietów. Zbyt szybkie deklarowanie utraty pakietów prowadzić może do niepotrzebnych retransmisji i tym samym skutkować obniżoną wydajnością działania mechanizmów kontroli. Mechanizmy te mogą jednocześnie wpływać na wielkość wspomnianego progu, dotyczącego akceptacji pakietów bez właściwej kolejności [36].

Należy dodać, że w QUIC ACK przewidziano też pole pozwalające zasygnalizować możliwość bezpośredniego przekazu informacji „przeciążeniowych" typu *Explicit Congestion Notification* (ECN), czyli działań wymuszających ograniczenie lub wzrost intensywności przekazu przez źródło (w protokole TCP możliwość wykorzystania ECN sygnalizowana była w polu znaczników). Jeśli ścieżka połączeniowa została sprawdzona pod kątem możliwości obsługi jawnego - bezpośredniego powiadomienia o przeciążeniu (ECN),



» *Rys. 19. Mechanizm sterowania przeciążeniowego w QUIC - jego stany i przejścia między nimi*

QUIC może wykorzystać do tego celu wartości elementu Congestion Experienced (CE) w nagłówku datagramu IP. Stanowi to jedną z różnic, w stosunku do TCP oraz SCTP. Nie tylko bowiem utrata pakietu ale i wzrost wartości ECN-CE powodują przejście z procedury (stanu) powolnego startu (*Slow Start*) do procedur związanych ze stanem Recovery (por. rys.19).

Poszczególne stany, przejścia między nimi oraz związane z tym procedury opisano w RFC 9002 [37].

Nadawca, korzystający z algorytmu *NewReno* (jeden z powszechnie stosowanych, także w TCP, mechanizmów reagowania na stany przeciążenia w sieci (*Congestion Control*) [38], [39], [40], [41]) realizuje procedurę powolnego startu (*Slow Start*), gdy rozmiar okna przeciążeniowego (*Congestion Window*) jest poniżej wartości progowej *Slow-StartThreshold*. Procedura ta jest wdrażana zarówno wtedy, gdy rozpoczynamy proces nadawania, lub gdy wchodzimy w stan „trwałego" (przedłużającego się przeciążenia - *persistent congestion*). Ma to miejsce wtedy, kiedy nadawca stwierdzi utratę wszystkich pakietów wysłanych w odpowiednio długim przedziale czasu.

W stanie *Slow Start* okno przeciążeniowe jest zwiększane (tradycyjnie) o liczbę bajtów potwierdzanych w kolejnych RTT. Powoduje to wykładniczy wzrost rozmiaru tego okna.

Gdy pakiet zostanie utracony (co jest traktowane jako symptom przeciążenia), względnie, gdy drugie urządzenie końcowe sygnalizuje wzrost wartości ECN-CE nadawca wchodzi w stan wychodzenia ze stanu przeciążenia (*Recovery*).

W stanie tym nadawca ustawia próg powolnego startu na połowę wartości okna przeciążeniowego związanego z momentem wykrycia utraty, a okno przeciążeniowe nie ulega zmianie w odpowiedzi na nowe straty lub wzrost wartości ECN-CE. Stan recovery kończy się, gdy pakiet wysłany już w okresie wychodzenia z przeciążenia zostaje potwierdzony. Różni się to nieco od definicji wychodzenia ze stanu przeciążenia w protokole TCP, w którym stan ten kończy się w momencie potwierdzenia utraconego segmentu-pakietu, który sygnalizował ten stan.

Nadawca NewReno przechodzi do stanu Congestion Avoidance i realizuje swoje transmisje w tym stanie (unikania przeciążenia) zawsze, gdy okno przeciążeniowe jest równe lub większe niż *SSThreshold*, i oczywiście, gdy nie znajduje się on w stanie Recovery.

Nadawca chcący uniknąć przeciążenia stosuje metodę „addytywnego" zwiększania i „multiplikatywnego" zmniejszania (AIMD) rozmiaru okna, co ogranicza wzrost okna przeciążenia do co najwyżej jednego maksymalnego rozmiaru pakietu dla każdego potwierdzonego okna przeciążeniowego. Nadawca opuszcza stan *Congestion Aviodance* i wchodzi w okres wychodzenia z przeciążenia (*Recovery*), gdy pakiet zostanie utracony lub gdy wartość ECN-CE zgłoszona przez urządzenie korespondujące wzrośnie.

Przyjmuje się jednocześnie możliwość ignorowania strat pakietów, których nie można rozszyfrować. Dotyczy to w szczególności pakietów, które mogą napłynąć do drugiego urządzenia przed pełnym pozyskaniem kluczy kryptograficznych, czyli pakietów wymienianych w fazie ustalania kluczy oraz procedur uwierzytelniania, związanych z wymianami w ramach 0-RTT lub 1-RTT.

Podsumowując ten fragment rozważań można stwierdzić, że zarówno SCTP, jak i TCP, łączą niezawodne dostarczanie i zapewnianie kolejności transmisji w ramach jednego źródła informacji, jakim jest TSN. Algorytmy kontroli przeciążenia stosowane w TCP, takie jak powolny start, szybka retransmisja i zegary RTT, są nadal używane w QUIC, z tą różnicą, że są przystosowane do używania numerów pakietów. Ponadto odnotowuje się kilka innych drobnych zmian.

ZAMYKANIE - USUWANIE POŁĄCZEŃ W SCTP I QUIC QUIC ([8], [9], [10], [12], [13])

Procedury zamykania połączenia (*Connection Teardown*) są silnie uwarunkowane aplikacyjnie. Z racji ulokowania procedur QUIC w warstwie aplikacji zamykanie połączeń jest prostsze. W SCTP wymagane są procedury trójstronnego uzgadniania (*3-way handshake*).

Zamykanie asocjacji w SCTP

By zakończyć połączenie SCTP, punkty końcowe realizują 3-stronne uzgodnienie (*3-way handshake*) zamknięcia powiązania. Aplikacja inicjuje ten proces przesyłając komunikat podstawowy (*Primitive*) typu ZAMKNIĘCIE (SHUTDOWN) do niższej warstwy, a następnie czeka na potwierdzenie wszystkich wysłanych fragmentów (*chunks*) i/lub retransmisję brakujących. W kolejnym kroku punkt końcowy wyśle Fragment SHUTDOWN (*SHUTDOWN Chunk*), aby zainicjować czyste zamknięcie powiązania po potwierdzeniu, że strona partnerska otrzymała wszystkie wysłane dane. Po odebraniu Fragmentu SHUTDOWN partnerski punkt końcowy przestanie akceptować dane ze swojej aplikacji (ULP) i potwierdzi otrzymanie wszystkich danych, a następnie odpowie poleceniem SHUTDOWN-ACK. Na koniec inicjujący punkt końcowy wyśle fragment SHUTDOWN-Complete, aby zamknąć powiązanie

Zamykanie sesji w QUIC

W momencie zamykania połączenia w QUIC, punkt końcowy wysyła ramkę zamykającą CONNECTION_CLOSE lub APPLICATION_CLOSE do swojego punktu partnerskiego i wchodzi w stan zamknięcia, w którym ignoruje wszystkie stany wewnętrzne z wyjątkiem wymaganych do utworzenia ramek zamykających. Jeśli po odebraniu ramki istnieją otwarte strumienie podrzędne, strumienie te są niejawnie zamykane. Jeśli inicjator zamknięcia odbierze pakiety, będąc w stanie zamknięcia, odpowiada na każdy z nich ramką zamykającą. Odbiorca ramki zamykającej wchodzi w stan usuwania połączenia, w którym nie wysyła już pakietów i ignoruje/odrzuca inny stan wewnętrzny. Zanim odbiornik wejdzie w stan usuwania, może również wysłać ramkę zamykającą. Zazwyczaj zerwanie połączenia QUIC jest uzgodnieniem dwukierunkowym, chyba że inicjator utracił pakiety. Innym sposobem, w jaki połączenie może zostać zamknięte, jest niejawny brak aktywności sieciowej, co powoduje przekroczenie limitu czasu w punktach końcowych.

INNE RÓŻNICE POMIĘDZY QUIC I SCTP

SCTP obsługuje *multihoming*. W szczególności punkt końcowy może zgłosić wiele adresów IP za pomocą pól fragmentów (*chunks*) INIT lub INIT ACK. Protokół może więc ustanowić wielościeżkowe połączenie z drugim punktem końcowym. Gdy w jednym z połączeń przekroczony zostanie limit czasu, fragment (*chunk*) może zostać ponownie przesłany przez inne aktywne połączenie, zwiększając odporność połączenia SCTP.

Oryginalny QUIC nie obsługuje funkcji multihomingu. Trwają jednakże prace nad wielościeżkowym rozszerzeniem QUIC Draft w celu dodania funkcji wielościeżkowej do protokołu QUIC ([18], [19]).

QUIC bardzo przypomina połączenie TCP, TLS i HTTP/2/3. Pakiety QUIC są zawsze szyfrowane (z wyjątkiem nagłówka publicznego) i uwierzytelniane (w tym nagłówek publiczny). Szyfrowanie zapobiega analizowaniu informacji o przeciążeniu, przez węzły pośrednie na trasie i wprowadzaniu przez nie zmian, co obecnie stanowi problem dla protokołu TCP. Nagłówek publiczny jest wymagany albo do routingu, albo też poprawnej identyfikacji pakietu, co wymaga by był niezaszyfrowany. Zawartość pakietu jest również w pełni uwierzytelniona, co zapobiega manipulacjom w sieci. Jakakolwiek modyfikacja pakietu QUIC spowoduje zerwanie połączenia.

Z kolei sam protokół SCTP nie oferuje szyfrowania ani uwierzytelniania, podobnie jak protokół TCP.

STAN WDRAŻANIA I WYKORZYSTANIA SCTP I QUIC

Jeżeli chodzi o sieć IP, to wdrożenie SCTP napotyka na szereg trudności [7]. Podstawową stanowi fakt, że urządzenia w sieci IP, na przykład bramy NAT, nie obsługują dobrze protokołu SCTP. By było to możliwe bramy NAT muszą zostać zaktualizowane, aby były zgodne z SCTP. Wiadomo jednak, że modyfikacja węzłów pośredniczących (*routerów*) jest bardzo kosztowna, a dostawcy usług internetowych muszą mieć wyraźne uzasadnienie biznesowe, podejmując decyzje o aktualizacji urządzeń. W rzeczywistości niektóre zapory (*firewalls*) przepuszczają (jak wspomniano wcześniej) tylko pakiety TCP lub UDP, co ogranicza SCTP do bardzo małej przestrzeni zastosowań. Biorąc pod uwagę fakt, że MPTCP może zaspokoić takie potrzeby, motywacja do wdrażania SCTP jest mniejsza. Największym mankamentem jest to, że w przeciwieństwie do MPTCP, interfejsy API gniazd SCTP znacznie różnią się od TCP, a programiści muszą aktualizować swój kod źródłowy, aby wdrożyć SCTP, co znacznie utrudnia szerokie wdrożenie SCTP.

Z uwagi na wspomniany brak obsługi protokołu SCTP przez praktycznie wszystkie rozwiązania NAT typu *„legacy”* podejmowano szereg prób rozwiązania tego problemu. W szczególności w dokumencie RFC 6951 [44] zaproponowano i opisano prostą metodę enkapsulacji pakietów protokołu SCTP w pakietach UDP. Takie, nieco sztuczne, rozwiązanie pozwala na użycie SCTP w sieciach z rozwiązaniami NAT, które nie obsługują SCTP. Wspomniany dokument prezentuje jednakże wyłącznie funkcjonalności wymagane w stosie SCTP, które pozwalają na enkapsulację pakietów SCTP w pakietach UDP. Niestety w propozycji tej nie zdefiniowano w pełni mechanizmów, które są wymagane do komunikowania się ze sobą dwóch hostów końcowych przez porty UDP. W szczególności nie są proponowane mechanizmy pozwalające określić, czy drugie urządzenie końcowe korzysta z enkapsulacji UDP, ani też mechanizmy określające, który zdalny numer portu UDP może zostać użyty. Narzuca się przy tym uwaga, że rozwiązanie to przypomina, do pewnego stopnia, podejście zaproponowane w QUIC.

Zaprojektowany przez Google QUIC jest obecnie szeroko stosowany w „klientach” Chrome uzyskujących dostęp do usług Google. Działając na szczycie UDP, ma wsparcie po stronie większości urządzeń sieciowych. Dzięki temu jest zdecydowanie bardziej przyjazny dla węzłów pośredniczących niż SCTP. Ponieważ QUIC działa w warstwie aplikacji, zakłada się, że może być aktualizowany znacznie częściej niż stos TCP w jądrze. QUIC udostępnia nowe API, które nie jest w pełni przezroczyste dla starszych aplikacji, co sprawia, że podobnie jak w przypadku SCTP, programiści muszą przetworzyć kod źródłowy, aby umożliwić wcześniejszym

aplikacjom korzystanie z QUIC. Tym niemniej dzięki wsparciu potentatów na runku usługodawców Internetu i społeczności internetowych wdrażanie QUIC jest relatywnie szybkie i obiecujące.

Dodajmy na zakończenie (za [7]), że QUIC nie ma zastąpić protokołu TCP. QUIC bazuje na ciągłej dostępności protokołu TCP. Ilekroć bowiem QUIC napotka na błąd krytyczny, taki jak np. fragmentacja pakietu QUIC HELLO, zamierzoną reakcją QUIC jest zamknięcie połączenia. Ponieważ sam QUIC znajduje się w przestrzeni aplikacji, a nie w jądrze, aplikacja po stronie klienta może zostać bezpośrednio poinformowana o zamknięciu połączenia QUIC i może ponownie otworzyć połączenie z serwerem przy użyciu konwencjonalnego protokołu transportowego TCP, z jego „obciążeniami".

WNIOSKI KOŃCOWE

Oba omawiane protokoły, czyli zarówno SCTP jak i QUIC zaadoptowały szereg funkcji z traktowanego przez nas jako referencyjny protokołu TCP, a projektanci obu ciekawych rozwiązań niewątpliwie korzystali z wcześniejszych doświadczeń związanych z pracami nad projektowaniem protokołów. QUIC i SCTP mają pewne podobieństwa i różnice. Jako podsumowanie naszych rozważań zaprezentujemy, sygnalizowane dość wybiórczo cechy obu protokołów.

Stream Control Transmission Protocol i jego krótka charakterystyka:

- **Podstawowa wersja SCTP** została przedstawiona w RFC 2960, w 2000 roku. W opracowaniu standardu uczestniczyły m.in. firmy: Motorola, Cisco, Siemens, Ericsson. Kolejne wersje SCTP zostały opisane w RFC 4960/9260.
- **SCTP zapewnia obsługę wielu strumieni danych**: Protokół umożliwia obsługę wielu logicznych strumieni wiadomości generowanych przez aplikację. Jednocześnie zapewnienia prawidłową kolejność wiadomości/komunikatów w ramach strumienia. Pozwala na unikanie blokowania wszystkich strumieni w przypadku wystąpienia strat.
- **Realizuje transfer danych zorientowany na przekaz wiadomości/ komunikatów** (*message oriented*): Przekaz danych użytkownika realizowany jest w postaci wiadomości/komunikatów, z zachowaniem struktur danych generowanych na poziomie aplikacji.
- **SCTP oferuje multihoming dla zapewnienia redundacji w sieci**: Funkcjonalność ta realizowana jest poprzez możliwość wykorzystywania przez urządzenia końcowe SCTP wielu adresów IP (w praktyce wiąże sie to zwykle z kilkoma różnymi technologicznie interfejsami). Pozwala to na przesyłanie wiadomości za pośrednictwem jednostek danych - SCTP PDU (w postaci tzw. Fragmentów (*Chunks*)) różnymi - alternatywnymi trasami.
- **Zapewnia ochrona przed atakami typu Odmowa Usługi** (*Denial of Service* - DoS): W SCTP wprowadzono kilka mechanizmów w procedurze zestawiania połączenia, chroniących przed atakami DoS i zapobiegające zalewaniu w wyniku zajmowania pamięci podczas nieuprawnionego otwierania sesji.
- **Dopuszcza fragmentację/segmentację**: Implementowany w SCTP mechanizm określania dopuszczalnej, na danej ścieżce, maksymalnej wartości MTU (*Maximum Transmission Unit*) pozwala na wcześniejszą segmentację Fragmentów (*Chunks*), dopasowując ich rozmiar do dopuszczalnej wartości MTU.
- **Umożliwia szybkie reagowanie na utraty pakietów**: W SCTP realizuje się procedury potwierdzania poprawnie odebranych jednostek danych, z jednoczesną możliwością wskazania brakujących (dzięki selektywnemu potwierdzaniu SACK), co pozwala na szybszą ich retransmisję. Mechanizmy SCTP zapobiegają też duplikowaniu jednostek.
- **Implementuje pełny zestaw mechanizmów zapobiegania** przeciążeniom w sieci: W SCTP wykorzystywane są procedury analogiczne do stosowanych w TCP (*Slow Start, Congestion Avoidance, czy Fast Retransmit*).

Krótka charakterystyka QUIC:

- **Quick UDP Internet Connections** (QUIC) to protokół transportowy ogólnego przeznaczenia implementowany w warstwie aplikacji, pierwotnie zaprojektowany przez Jima Roskinda w Google i wdrożony w 2012 r. Został ogłoszony publicznie w 2013 r. jako rozwiązanie rozszerzone i opisany przez IETF w dokumentach RFC 9000/9312 [9], [10].
- **QUIC łączy uzgadnianie transportu i zabezpieczeń kryptograficznych**, wykorzystując kryptograficzne pliki cookie do wznowienia połączenia, minimalizując opóźnienia w zestawianiu i szyfrowaniu połączenia.
- **Pakiety QUIC są zawsze szyfrowane** (z wyjątkiem nagłówka publicznego) i uwierzytelniane (w tym nagłówek publiczny). QUIC rozwiązuje również problemy bezpieczeństwa nieodłącznie związane z umożliwieniem wymiany danych podczas uzgadniania 0-RTT, poprzez użycie tokena bezpieczeństwa do sprawdzania poprawności adresu.
  - *QUIC używa długoterminowego kryptograficznego pliku cookie i identyfikatora połączenia*, co niestety otwiera możliwości nowych typów ataków. Równoważenie kompromisów między zyskami, a stratami jest zawsze częścią projektu protokołu.
- **QUIC działa w przestrzeni użytkownika**, umożliwiając szybkie wdrażanie i eksperymentowanie. Ponieważ działa poprzez UDP, jest kompatybilny z większością implementacji węzłów pośredniczących.
  - QUIC zapewnia również mechanizm rezerwowy dla normalnego uzgadniania TCP w przypadku, gdy jedna ze stron nie obsługuje protokołu.
  - *QUIC wdraża, w pełnym zakresie, mechanizmy kontroli przeciążenia*.
- Dzięki multipleksowaniu strumieni (wielostrumieniowości) brakujące ramki jednego **strumienia nie blokują dostarczania danych w innych strumieniach**, unikając problemu HOL-Blocking,

  Przesyłanie danych z wielu strumieni w pojedynczych pakietach wprowadza jednakże dodatkowe przetwarzanie – by zachować wewnętrzne odwzorowywanie, pomiędzy numerem ramki w danym strumieniu i numeracją pakietów (i by wiedzieć, które ramki należy przesłać ponownie).
- **QUIC wykorzystuje do sterowania przepływem i kontroli przeciążeń więcej elementów sygnalizacyjnych niż TCP**, co sprawia, że QUIC dostarcza więcej informacji dla algorytmów kontroli przeciążenia i efektywniej oraz w bardziej elastyczny sposób reaguje na niepożądane stany w sieci.

» *Tab. 1. Podobieństwa i różnice pomiędzy protokołami TCP, QUIC i SCTP (na podstawie: [45])*

Protokół	TCP (*Transmission Control Protocol*)	QUIC	SCTP (*Stream Control Transmission Protocol*)
Niezawodność	Niezawodne dostarczanie danych z mechanizmami wykrywania błędów, retransmisji i potwierdzania	Niezawodne dostarczanie danych z mechanizmami wykrywania błędów, retransmisji i potwierdzania	Niezawodne dostarczanie danych z mechanizmami wykrywania błędów, retransmisji i potwierdzania
Rodzaj połączenia logicznego	Zorientowane połączeniowo	Zorientowane połączeniowo	Zorientowane połączeniowo
Porządkowanie kolejności dostarczania	Gwarantuje uporządkowaną dostawę pakietów danych	Gwarantuje uporządkowaną dostawę pakietów danych	Gwarantuje uporządkowaną dostawę pakietów danych, jednocześnie umożliwia „mieszanie ruchu” - *ordered* oraz *unordered delivery service*
Narzut protokolarny – wspólny nagłówek/ dodatkowe nagłówki lub opcje	Nagłówek TCP obejmuje 20 B w polach stałych oraz do 40 B w polach opcji	Nagłówek pakietu QUIC typu DATA obejmuje 10-14 B w polach stałych (w tzw. nagłówku krótkim) oraz od 4-11 B w pojedynczym nagłówku ramki	Nagłówek pakietu SCTP przenoszącego dane obejmuje 12B we wspólnym nagłówku i 16 B w pojedynczym nagłówku fragmentu DATA
Aplikacje (główne)	*Web browsing, email transfer, file transfer* (FTP)	Web browsing	Sieci telekomunikacyjne, voice and video over IP, przekaz sygnalizacji w sieciach 4/5G
Sterowanie przepływem	Do sterowania przepływem wykorzystywany jest mechanizm okna przesuwnego, a ponadto rozmiar okna oraz numery sekwencyjne przesyłanych i potwierdzanych bajtów	Implementuje zasady sterowania z TCP. Sterowanie odbywać na dwóch poziomach - całego połączenia oraz poszczególnych strumieni	Implementuje zasady sterowania z TCP. Odmiennie niż w TCP numeracji podlegają wiadomości. Sterowanie jest realizowane w odniesieniu do całego powiązania - asocjacji.
Kontrola przeciążeń	Wdraża mechanizmy kontroli przeciążeń w celu optymalizacji wydajności sieci (*Slow Start, Cong. Avoidance*, czy *Fast Retransmit*)	Wdraża mechanizmy kontroli przeciążeń wykorzystywane w TCP (i kilka własnych) w celu optymalizacji wydajności sieci	Wdraża mechanizmy kontroli przeciążeń wykorzystywane w TCP w celu optymalizacji wydajności sieci
Wykrywanie błędów	Wykrywa i retransmituje utracone lub uszkodzone pakiety (opcjonalnie selektywne ACK)	Wykrywa i retransmituje utracone lub uszkodzone pakiety (z SACK)	Wykrywa i retransmituje utracone lub uszkodzone pakiety (z SACK)
Przekaz zorientowany na wiadomości	Nie (transfer zorientowane na bajty)	Nie (transfer zorientowany jest na bajty)	Tak, oferuje dostarczanie zorientowane na wiadomości
Wielostrumieniowość (*multistreaming*)	Brak (wyłącznie pojedynczy strumień jest obsługiwany)	Tak, obsługuje jednoczesną transmisję wielu strumieni	Tak, obsługuje jednoczesną transmisję wielu strumieni
Wielointerfejsowość (*multihoming*)	Brak takiej funkcjonalności	Brak takiej funkcjonalności (do tej pory). Możliwa jest jednak szybka migracja na nowy adres IP/port w przypadku zmiany sieci,, dzięki stałemu identyfikatorowi Connection ID.	Tak, obsługuje wiele adresów IP w celu poprawy efektywności i zapewnienia odporności na awarie poprzez nadmiarowość (adresy deklarowane są w fazie inicjowania połączenia (fragmenty INIT)
Wielościeżkowość (*multipath*)	Brak takiej funkcjonalności	Brak takiej funkcjonalności (do tej pory)	Tak (dzięki funkcjonalności multi-homingu)
Równoważenie obciążenia	Brak takiej funkcjonalności	Brak (aplikacja musi optymalizować długość i zawartość pakietów z ramkami DATA)	Możliwe (proponowane w szeregu rozwiązaniach - w przypadku przekazów wielościeżkowych)
Potencjalne wsparcie mobilności	Brak	Brak (parametr Connection ID ułatwia migrację do nowego adresu IP)	Tak (możliwe wsparcie w przypadku transferów wielościeżkowych i dzięki funkcjonalności multihomingu)
Ochrona kryptograficzna	Nie jest realizowana w ramach protokołu (wymagana jest niezależna ochrona za pośrednictwem TLS)	Ochrona TLS jest zintegrowana z przesyłem danych z aplikacji	Nie jest realizowana standardowo (wymagana jest niezależna ochrona za pośrednictwem TLS, np. TLS/ DTLS over SCTP)
Ochrona przed atakami typu DOS/SYN	Brak	Częściowa	Tak (dzięki procedurom w fazie zestawiania połączenia)
Obsługa przez urządzenia NAT	Realizowana standardowo	Realizowana standardowo	Utrudniona (szczególnie w rozwiązaniach typu „legacy”)

» *Rys.20. Wykorzystanie QUIC w serwisach webowych (na podstawie [46])*

Pełniejsze porównanie zaprezentowanych w artykule nowych propozycji protokołów transportowych z przyjętym przez nas jako referencyjny, protokołem TCP zawarto w tabeli 1.

Protokół QUIC został zaprojektowany jako protokół ogólnego przeznaczenia. Tym niemniej jego pierwsze wdrożenie związane było z obsługą protokołu HTTP(S) w przeglądarkach Google Chrom. Pomimo, że kolejne wersje tego standardu jednoznacznie deklarują, że jest on protokołem transportowym ogólnego przeznaczenia, to w dalszym ciągu jego sztandarowe zastosowania sprowadzają się do obsługi złożonych aplikacji przeglądarkowych.

W opracowaniu Geoff'a Hustona [34] zawarto szereg interesujących danych na temat użycia QUIC (czyli pełnej integracji z protokołem HTTP/3) w popularnych przeglądarkach, Zgodnie z oczekiwaniami liderem jest Chrom z ponad 52,2% udziałem. Znaczny udział QUIC (i HTTP/3) jest też widoczny w odniesieniu do przeglądarek Safari, (44,6%). Pozostałe przeglądarki, w tym Firefox, Edge, czy też Opera korzystały z protokołu QUIC (HTTP/3) w zdecydowanie mniejszym stopniu.

Ciekawa jest także informacja związana ze wspieraniem HTTP/3 (i tym samym możliwością obsługi protokołu QUIC), przez znane systemy operacyjne i z ich procentowym udziałem (sumującym się do 100%). Liderami pod tym względem są Android (47,7%) oraz iOS (44,5%). System Windows (nowa wersja) ma mniej niż 10% udział. Inne platformy w tym Linux, czy Win7/8 mają w takim podziale niewielki udział.

Należy jednak dodać, że zakres zastosowań QUIC ciągle się poszerza. Można tutaj wymienić np. DNS (*Domain Name System*)-over-QUIC - do rozpoznawania nazw, czy też SMB (Server Message Block) over QUIC - oferujący usługę SMB VPN.

Aktualne statystyki związane z wykorzystaniem QUIC wskazują, że korzysta niego ok. 7,5-9% (z pewnymi i wahaniami) serwisów przeglądarkowych, przy czym w październiku 2023 roku było to 7,6% [46] (ilustruje to rys. 20).

Prezentowane statystyki ulegają jednakże bardzo szybkim zmianom. Statystyki W3Tech.com pokazują np. ok 30% udział HTTP/3 i QUIC na 10 milionach najpopularniejszych witryn WWW. Usługodawcy chmurowi jak np. *Cloudflare* wprowadzili HTTP/3 dla wszystkich swoich klientów poprzez *reverese proxy* na styku Internet-Chmura [47]. Pamiętajmy, że HTTP/3 to też QUICK, a w przypadku HTTP/3 nie ma innego protokołu transportowego.

Z kolei protokół SCTP, dalej oczekuje na swoją „szansę" w klasycznych sieciach IP, pozostając, pomimo podjęcia szeregu kroków w kierunku jego akceptacji do przekazu danych, bardzo ważnym protokołem sygnalizacyjnym w płaszczyźnie sterowania, w praktycznie wszystkich generacjach sieci telefonii komórkowej.

W systemach 3G protokół SCTP jest wykorzystywany do przekazu komunikatów Systemu Sygnalizacji Nr 7 (*Signaling System No 7* - SS7 [48]), pochodzących głównie z SCCP (*Signaling Connection Control Part*). Wymiana pakietów SCTP odbywa się pomiędzy urządzeniem IP *Signaling Point* (IPSP), typu *Signaling Gateway* (SG), a innymi IPSPs w sieci.

W systemach trzeciej i czwartej generacji (4G LTE) podstawową aplikacją oferującą usługi AAA (*Authentication, Authorization, and Accounting*) w sieciach operatorskich stał się protokół Diameter. W celu zapewnienia niezawodnego przesyłu komunikatów tego protokołu, pomiędzy elementami sieci, w jej warstwie transportowej wykorzystywany jest powszechnie protokół SCTP. Drugim protokołem warstwy transportowej, zaproponowanym w RFC 6733 [49] jest TCP.

Kluczowym zadaniem SCTP w sieciach 5. generacji (5G) jest, z kolei przenoszenie strumieni sygnalizacyjnych w ramach interfejsu N2. W szczególności SCTP został przewidziany do realizacji przekazów pomiędzy węzłem NG-RAN (stacją bazową nowej generacji - gNodeB w radiowej sieci dostępowej - RAN) i komponentem usługowym tej sieci - AMF (*Access and Mobility Function*) [42], [43], [50]. Opis zadań, wymienianych komunikatów oraz rodzajów powiązań pomiędzy NG-RAN i AMF zdefiniowano w [51], [52]. Jako ciekawostkę podajmy, że wartość numeru portu docelowego przydzielonego przez IANA do użycia w odniesieniu do aplikacji NG AP to 38412.

Podsumowując nasze rozważania można uznać, że oba zaprezentowane rozwiązania, różniące się pod względem implementacyjnym, mają szereg ciekawych cech (w wielu przypadkach wspólnych), które to cechy pozwalają im odgrywać ważne role we współczesnych sieciach wykorzystujących protokół IP oraz architekturę TCP/IP, chociaż w zdecydowanie odmiennych scenariuszach użycia. ●

LITERATURA

1. https://en.wikipedia.org/wiki/DARPA
2. Internet Protocol. DARPA Internet Program Protocol Specification. RFC: 791, September 1981 (ed. Jon Postel)
3. J. Postel (ed): Transmission Control Protocol. DARPA Internet Program Protocol Specification. RFC: 793 (Replaces: RFC 761), September 1981
4. M. Sportack: TCP/IP First-Step. Cisco Press. 1st ed. Dec 3, 2004
5. J. Postel(ed): User Datagram Protocol. RFC 768, August 1980
6. https://en.wikipedia.org/wiki/Transport_layer

7. G. Huston: A quick look at QUIC. The ISP Column - A monthly column on things Internet. March, 2019, https://www.potaroo.net/ispcol/2019-03/quic.pdf
8. Arun Joseph, Tianxiang Li, Zihao He, Yong Cui, Lixia Zhang: A Comparison between SCTP and QUIC. QUIC working group. Internet-Draft Intended status: Informational, September 2018
9. J. Iyengar, M. Thomson (eds.): QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021
10. M. Kühlewind, B. Trammell: Manageability of the QUIC Transport Protocol. RFC 9312, September 2022
11. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson: Stream Control Transmission Protocol. RFC 2960, October 2000
12. R. Stewart (Ed.): Stream Control Transmission Protocol. RFC 4960, September 2007
13. R. Stewart, M. Tüxen, K. Nielsen: Stream Control Transmission Protocol. RFC 9260, June 2022
14. A. Ford, C. Raiciu, M. Handley, O. Bonaventure: - TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, January 2013
15. A. Ford, C. Raiciu, M. Handley, O. Bonaventure, C. Paasch: TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684, March.2020
16. J. Iyengar, M. Thomson: QUIC: A UDP-Based Multiplexed and Secure Transport. draft-ietf-quic-transport-08. December 2017
17. Ł. Budzisz: Stream Control Transmission Protocol (SCTP), a proposal for seamless handover management at the transport layer in heterogeneous wireless networks. Universitat Politecnica de Catalunya ` Department of Signal Theory and Communication Radio Communication Group. Ph.D. Dissertation, Barcelona, 2009
18. Q. De Coninck, O. Bonaventure. 2021. Multipath Extensions for QUIC (MP-QUIC). Internet-Draft draft-deconinck-quic-multipath-07. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-deconinck-quic-multipath-07 Work in Progress
19. Q. De Coninck: The packet number space debate in multipath QUIC. ACM SIGCOMM Computer Communication Review, Volume 52, Issue 3, July 2022, pp 2–9 https://doi.org/10.1145
20. H. Krawczyk, M. Bellare, R. Canetti: HMAC: Keyed-Hashing for Message Authentication. - RFC2104, February 1997
21. E. Barker, Q. Dang, S. Frankel, K. Scarfone, P. Wouters: Guide to IPsec VPNs. NIST Special Publication 800-77 Revision 1. June 2020
22. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 – draft-ietf-tls-tls13- 18. https://tools.ietf.org/html/draft-ietf--tls-tls13-18, October 2016
23. A. Jungmaier, E. Rescorla and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002
24. A. Hohendorf, E. Rathgeb, M. Tüxen: Secure End-to-End Transport over SCTP. Computer Science, June 2006
25. S. Lindskog, A. Brunström: An End-to-End Security Solution for SCTP. Third International Conference on Availability, Reliability and Security. March 2008
26. S. Lindskog, A. Brunström: Secure Socket SCTP: A Multi-layer End-to-End Security Solution. Computer Science. 2008
27. M. Tuexen, R. Seggelmann, E. Rescorla: Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP). RFC 6083, January 2011
28. M. Westerlund, J. Preuss Mattsson, C. Porfiri: Datagram Transport Layer Security (DTLS) over Stream Control Transmission Protocol (SCTP). draft-ietf-tsvwg-dtls-over-sctp-bis-07. October 2023
29. M. Tuexen, R. Stewart, P. Lei, P, E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", RFC 4895, August 2007
30. https://venturebeat.com/business/google-plans-to-propose-its-quic-network-protocol-which-delivers-http-over-udp-as-an-internet-standard
31. E. Sy, C. Burkert, H. Federrath, M. Fischer A QUIC Look at Web Tracking. Proceedings on Privacy Enhancing Technologies ; 2019 (3):255–266
32. A. Ghedini: Even faster connection establishment with QUIC 0-RTT resumption. 20.11.2019. https://blog.cloudflare.com/even-faster-connection-establishment-with-quic-0-rtt-resumption
33. M. Atiquzzaman, W. Ivancic: Evaluation of SCTP multistreaming over wireless/satellite links. Proceedings. 12th International Conference on Computer Communications and Networks. Dallas, TX. October 2020-22, 2003]).
34. G. Huston: A look at QUIC use. Jul 2022, https://blog.apnic.net/2022/07/11/a-look-at-quic-use/
35. W. Stevens: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001. January 1997
36. M. Allman, V. Paxson, W. Stevens: TCP Congestion Control. RFC 2581, April 1999
37. J. Iyengar, I. Swett (eds.): QUIC Loss Detection and Congestion Control. Internet Engineering Task Force (IETF). RFC: 9002, May 2021
38. S. Floyd, T. Henderson: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC2582, April 1999
39. M. Allman, V. Paxson, E. Blanton, "TCP Congestion Control", RFC5681, September 2009
40. T. Henderson, S. Floyd, A. Gurtov, A, Y. Nishida: The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582, April 2012
41. M. Hoeft, J. Woźniak: Evaluation of Connectivity Gaps Impact on TCP Transmissions in Maritime Communications. Computer Networks. Springer International Publishing, 2017, p.91-105 (eds.: P. Gaj, A. Kwiecień, M.Sawicki)
42. 5G: NG-RAN - NG signalling transport. (3GPP TS 38.412 version 15.0.0 Release 15). ETSI TS 138 412 V15.0.0. 2018-07
43. System architecture for the 5G System (5GS). Technical Specification Group Services and System Aspects. 3rd Generation Partnership Project. Stage 23GPP TS 23.501 V18.3.0 (2023-09) (Release 18)
44. M. Tuexen, R. Stewart: UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication. RFC 6951, May 2013
45. J. Woźniak, K. Nowicki: The Need for New Transport Protocols on the Internet. AUTOMATYKA, ELEKTRYKA, ZAKŁÓCENIA. Vol. 14, nr 3 (53) 2023
46. Usage statistics of QUIC for websites. https://w3techs.com/technologies/details/ce-quic (October 2023)
47. https://w3techs.com/technologies/overview/site_element
48. Specifications of the Signalling System No 7. ITU-T Recommendation Q.700. 03/1993
49. V. Fajardo, J. Arkko, J. Loughney, G. Zorn: Diameter Base Protocol . RFC 6733, October 2012
50. J. Woźniak (ed.): Tendencje w rozwoju polskiej i światowej telekomunikacji i teleinformatyki. Warszawa, Wyd. WAT 2020
51. System architecture for the 5G System (5GS). Technical Specification Group Services and System Aspects. Stage 2 3GPP TS 23.501 V18.3.0 (2023-09) (Release 18)
52. Procedures for the 5G System (5GS). 3GPP TS 23.502 version 16.7.0 Release 16), ETSI TS 123 502 V16.7.0 (2021-01).