AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I ELEKTRONIKI

KATEDRA TELEKOMUNIKACJI

Praca dyplomowa

magisterska

Imię i nazwisko	**Szymon Wieloch**
Kierunek studiów	Elektronika i Telekomunikacja
Temat pracy dyplomowej	**Metody zapewniania niezawodności dla sieci tolerujących opóźnienia i przerwania**
Opiekun pracy	dr inż. Piotr Chołda

Kraków, rok 2010

Spis treści

1 Wstęp

1.1 Cel pracy dyplomowej

Celem tej pracy dyplomowej jest:

- Ukazanie problemów, jakie pojawiają się w sieciach, gdzie występują duże opóźnienia propagacji sygnału lub w czasie transmisji często mają miejsce przerwania łączności.
- Przedstawienie protokołów służących transmisji danych, radzących sobie w tak trudnych sytuacjach oraz ukazanie najważniejszych ich cech.
- Przedstawienie mechanizmów wznawiania pracy oraz różnych protokołów rutingu radzących sobie w niektórych specyficznych środowiskach.
- Stworzenie modelu sieci tolerującej przerwania i opóźnienia — uproszczenie przedstawionych protokołów do poziomu, który będzie umożliwiał implementację symulatora oraz przeprowadzenie symulacji w rozsądnym czasie, ale jednocześnie zachowa wszystkie istotne cechy protokołów.
- Zaprojektowanie oraz implementacja symulatora sieci *DTN*, umożliwiającego porównanie statystyczne działania sieci wykorzystujących różne protokoły do przesyłania danych.
- Przeprowadzenie symulacji dla zarówno typowych, jak i skrajnych przypadków, aby móc odpowiedzieć na pytanie, kiedy opłaca się stosować dany protokół i jaką efektywnością się on charakteryzuje w zależności od struktury sieci i charakterystyki istniejących elementów.

1.2 Skrót „DTN", czyli kolizja nazw

Pierwotnie skrót *DTN* oznaczał sieć tolerującą opóźnienia (ang. *Delay-Tolerant Network*). Jednak ministerstwo obrony Stanów Zjednoczonych zdecydowało się na rozpoczęcie pracy nad protokołami tolerującymi przerwania w transmisji. Sieć taka nazwana została *Disruption-Tolerant Network*, co oczywiście wprowadza konflikt nazw. Dodatkowo sytuację komplikuje fakt, że oba typy sieci wykorzystują dokładnie te same protokoły — zazwyczaj są to *Bundle* oraz *LTP* opisane dalej. Praca ta zajmuje się obydwoma typami sieci. Zamiast wprowadzać nazwę *Delay-and-Disruption-Tolerant Network*, którego nie da się znaleźć praktycznie w żadnej

pozycji związanej z tematem, zdecydowałem się pozostać przy nazwie *DTN*, przy czym pod tym skrótem rozumiem sieć tolerującą zarówno przerwania, jak i opóźnienia. **[1,10]**

1.3 Konwencja oznaczeń używanych w tej pracy

Nazwa

Nazwa angielska lub tłumaczenie

[n] – odwołanie do pozycji bibliografii o numerze n.

2 Teoria

2.1 Po co nowy rodzaj sieci?

2.1.1 Potrzeba specyficznych sieci

Mimo istnienia wielu różnych typów sieci telekomunikacyjnych okazuje się, że ciągle istnieją dosyć specyficzne przypadki, z którymi zupełnie nie radzą sobie znane, typowe protokoły telekomunikacyjne. Dwa takie przypadki, które będą rozważane w tej pracy dyplomowej, to duże opóźnienia występujące podczas propagacji sygnału oraz przerwania transmisji. Przypadki te zazwyczaj nie są rozróżniane, ponieważ protokoły je obsługujące działają w obu spośród nich.

W ww. warunkach zdecydowanie najlepiej działają protokoły oparte na transmisji pakietów, lub — mówiąc ogólniej — paczek danych. Dzięki temu, że dzielą one większą ilość danych na mniejsze fragmenty oraz dzięki sumom kontrolnym, lub innym mechanizmom, umożliwiają kontrolę poprawności odebranych danych, łatwe staje się ustalenie, jaka część danych dotarła prawidłowo, a jaka nie. Łatwe jest także poinformowanie drugiej strony o konieczności retransmisji określonego fragmentu. Takie właśnie podejście w *Internecie* przyczyniło się do niesamowitej popularności tej sieci. **[1]**

Podejścia oparte na transmisji całej wiadomości (komutacja wiadomości) czy utworzeniu znanego z tradycyjnej telefonii „czystego" kanału transmisji (komutacja kanałów), w którym nie działa dodatkowy protokół dzielący dane na pakiety, nie sprawdzają się. Komutacja wiadomości okazuje się nieefektywna z powodu dużego narzutu, jaki wprowadza retransmisja (przesłanie ponownie całej wiadomości). W przypadku częstych przerwań prawdopodobieństwo uszkodzenia długiej wiadomości jest bardzo duże, duży jest więc także narzut na retransmisje, natomiast w przypadku dużych opóźnień kilkukrotna retransmisja oznacza drastyczne zwiększenie opóźnienia. Komutacja kanałów także okazuje się nieefektywna ze względu na brak możliwości prostego poinformowania drugiej strony o miejscu wystąpienia błędu i później odebrania retransmitowanych danych. **[1]**

2.1.2 Środowiska dla których brakuje obecnie dobrych rozwiązań

Aktualnie brakuje protokołów telekomunikacyjnych, które potrafiłyby sobie poradzić w następujących sytuacjach:

2.1.2.1 Duże opóźnienia

2.1.2.1.1 Transmisja międzyplanetarna

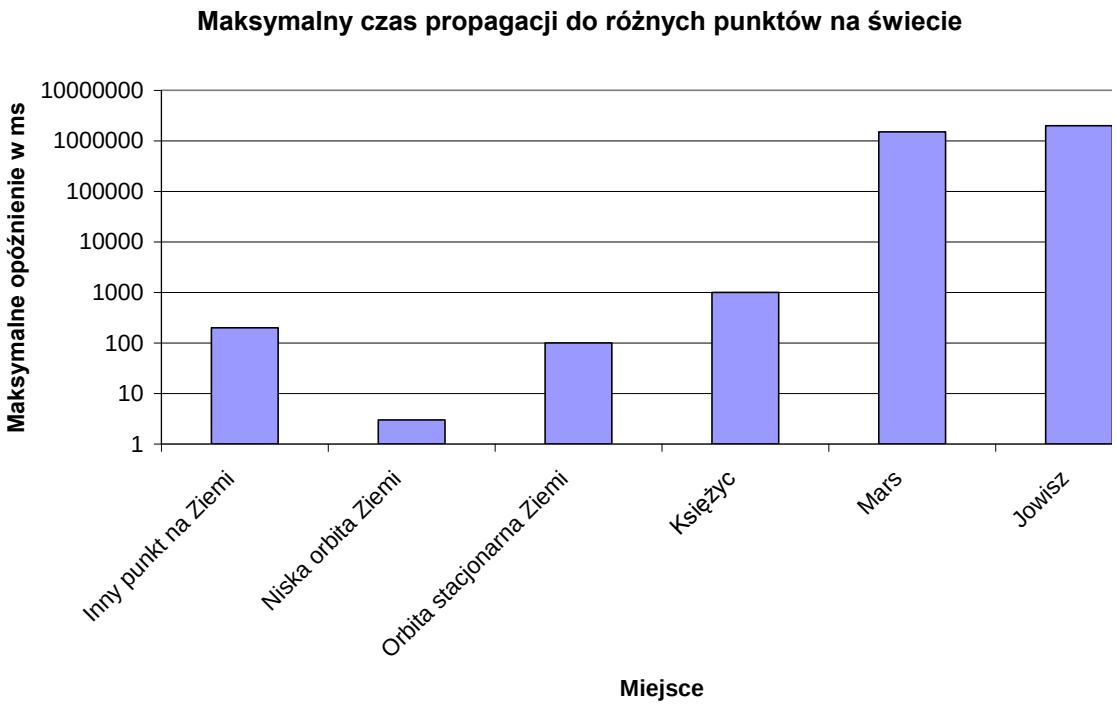
Szczególnym problemem jest brak możliwości transmitowania dużej ilości danych z Marsa z pracujących tam sond, a także dla bazy na Księżycu oraz wyprawy załogowej na Marsa, które obecnie są w fazie planowania. W grę wchodzą oczywiście także pozostałe planety oraz badane obiekty kosmiczne. W takich sieciach czas propagacji sygnału często sięga godzin (rysunek 2-1). **[1]**

2.1.2.1.2 Sieci podwodne

W wodzie ze względu na ogromne tłumienie fal elektromagnetycznych i możliwość transmisji tylko na odległości rzędu centymetrów, wykorzystywane są sieci oparte na transmisji dźwięku. Niestety, szybkość rozprzestrzeniania się dźwięku w wodzie to tylko 1,5 km/s, co oznacza, że transmisja na duże odległości wiąże się z wielosekundowymi opóźnieniami. **[1]**

2.1.2.1.3 Sieci sensorowe

Ze względu na stosunkowo duże opóźnienia występujące w sieciach sensorowych w węzłach (zazwyczaj dla oszczędności oraz z powodu ograniczonego dostępu do źródeł energii mają one bardzo prostą budowę i wykorzystują najprostsze protokoły) oraz często bardzo dużą liczbę przeskoków na drodze, uzyskiwane opóźnienia są stosunkowo duże. Częstym problemem są także uszkodzenia węzłów albo po prostu przestoje spowodowane brakiem zasilania, które bardzo często pochodzi z baterii słonecznych. **[1]**



Rysunek 2-1: Opóźnienia występujące podczas transmisji do różnych celów z Ziemi wynikające z ograniczonej szybkości propagacji fali elektromagnetycznej wg. [4]

2.1.2.2 Przerwania transmisji

2.1.2.2.1 Zebranet oraz inne projekty wykorzystujące tzw. „muły danych”

Zebranet to projekt mający na celu bliższe zbadanie życia afrykańskich zebr. Wiszące na szyjach zwierząt urządzenia wykorzystują *GPS* (ang. *Global Positioning System* – Globalny System Pozycjonowania, **[17]**) do zapisania co jakiś czas pozycji zwierzęcia. Problemy rodzi system przesyłania danych do naukowców. Aktualne rozwiązanie polega na tym, że gdy zebry zbliżają się na odległość równą zasięgom nadajników, zaczynają one automatycznie wymieniać między sobą wszystkie dane. Dzięki temu wystarczy później trafić tylko na jedną z zebr, aby zdobyć dane tego zwierzęta oraz wszystkich, z którymi miało kontakt. Ponieważ zwierzęta zbliżają się do siebie często na krótki okres, więc przerwania w czasie transmisji występują bardzo często, zaś opóźnienie dostarczenia danych do naukowców może sięgnąć lat. **[1]**

„Muły danych” to znacznie szersza koncepcja. Zakłada ona, że między źródłem danych, a miejscem docelowym transmisji porusza się jakiś obiekt, który od pierwszego urządzenia może te dane odbierać, a drugiemu przekazywać. Obecnie w Wielkiej Brytanii testowany jest system wymiany informacji między autobusami miejskimi. Autobusy te generują losowe dane, ponieważ testowana jest sama efektywność takiego rozwiązania, jednak łatwo sobie wyobrazić np. system analizowania zakorkowania poszczególnych dróg oparty na *GPS* oraz kilku dodatkowych węzłach sieci umieszczonych np. przy głównych skrzyżowaniach, które odbierałyby dane od autobusów i przekazywałyby je do centrów analizy. Taki system pozwalałby bardzo szybko obserwować zmiany w płynności poruszania się po mieście autobusów, a więc także wszystkich innych pojazdów, po wprowadzeniu zmian w organizacji ruchu oraz znajdować i usuwać miejsca szczególnie blokujące ruch. Obecnie miasto często inwestuje duże sumy w przebudowę dróg, które wcale nie okazują się najistotniejsze z punktu widzenia wpływu na sprawność ruchu w mieście. Nadal brakuje dobrego systemu analizującego ruch miejski na bieżąco. Kiedy autobusy mijają się na szybkiej drodze, czas, kiedy mają ze sobą łączność nie przekracza kilku sekund, dlatego często następują krótkie połączenia, z którymi tego typu sieci muszą sobie radzić.

2.1.2.2.2 Militarne sieci taktyczne

Sieci wojskowe są nietypowe pod tym względem, że można się spodziewać, że wróg będzie robił wszystko, żeby zakłócić transmisję. Większość wojskowych urządzeń w momencie stwierdzenia braku łączności zmienia częstotliwość na taką, która nie jest zakłócana (istnieją specjalne procedury ustalania, jaka to będzie częstotliwość). Od momentu nawiązania łączności protokoły powinny natychmiast wznowić transmisję danych. **[1]**

2.1.2.2.3 Urządzenia przemieszczające się

W niektórych sytuacjach konieczne jest umieszczenie nadajników na pojazdach poruszających się w zróżnicowanych terenie. Wymiana informacji pomiędzy takimi urządzeniami jest utrudniona ze względu na przeszkody pojawiające się na linii nadajnika i odbiornika. Przerwanie i wznawianie połączenia może odbywać się kilka razy na sekundę. Taka sytuacja ma też miejsce w zastosowaniach wojskowych, ale także w przypadku każdej komunikacji odbywającej się w górzystym lub miejskim terenie, czego przykładem może być sieć sensorowa badająca stan zdrowia pacjenta. Przemieszczając się po różnych obszarach miasta może on co chwila nawiązywać i tracić łączność z siecią, bądź też mieć kontakt tylko z mułami danych

(mogą to być np. środki komunikacji miejskiej), których zadaniem będzie przetransportować pobrane dane do ośrodka zdrowia. **[1]**

Aby poradzić sobie z takimi specyficznymi środowiskami potrzebujemy stworzyć nowy typ sieci, mającej swoje własne zasady komunikacji i przesyłania danych dostosowane do wymagań, jakie stawia specyficzne środowisko. Aby zrozumieć, dlaczego nie możemy wprost przenieść sprawdzonych i dobrze znanych mechanizmów np. z sieci *Internet*, wystarczy przyjrzeć się działaniu protokołu *TCP* (ang. *Transmission Control Protocol* — protokół kontroli transmisji) w sieci z częstymi przerwaniami czy też dużymi opóźnieniami.

2.1.3 Protokół TCP

TCP jest najpopularniejszym protokołem transportowym stosowanym obecnie w sieci *Internet*. *TCP* przesyła dane jako strumień bajtów, dzieląc je na mniejsze części, tzw. okna. Okna to po prostu pewna liczba bajtów, porcja danych. *TCP* gwarantuje niezawodny transport pomiędzy dwoma komunikującymi się węzłami. Szczególnie ta ostatnia cecha przyczyniła się do zastosowania tego protokołu wszędzie tam, gdzie przesyłane są duże ilości danych, które muszą zostać dostarczone niezawodnie do celu, natomiast nie występuje potrzeba dostarczania informacji w czasie rzeczywistym. Przykładami mogą być: transport plików (np. *FTP*, ang. *File Transfer Protocol* — protokół transferu plików), transport stron internetowych (np. *HTTP*, ang. *Hypertext Transfer Protocol* — protokół przesyłania dokumentów hipertekstowych) czy przesyłaniu tekstu (np. *telnet* — standard protokołu komunikacyjnego używanego do obsługi odległego terminala).

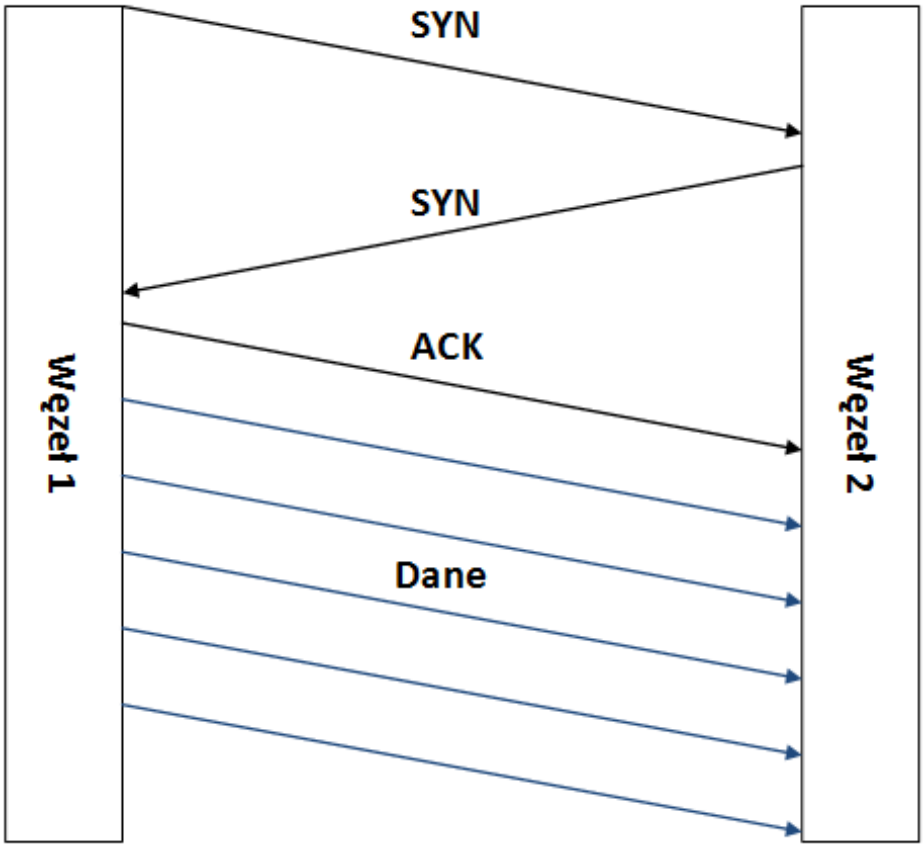
2.1.4 Działanie protokołu TCP przy dużych opóźnieniach

2.1.4.1 Maksymalne opóźnienie

TCP w domyślnej implementacji czeka na odpowiedź od drugiej strony maksymalnie 2 minuty. Jeśli w tym czasie nie dostanie żadnego sygnału podtrzymującego sesję, rozłącza się przekazując aplikacji informację o niepowodzeniu wysłania danych. W opisywanych w tej pracy środowiskach 2 minuty to zdecydowanie za mało. Oczywiście, wartość tę można zwiększyć, jednak nie rozwiąże to pozostałych problemów. **[1,5,14,15]**

2.1.4.2 Nawiązywanie połączenia

Nawiązanie połączenia przy opóźnieniach powyżej dwóch minut jest w ogóle niemożliwe. Jeśli jednak zwiększymy limit czasowy, to przekonamy się, że *TCP* co prawda zaczyna działać, ale staje się bardzo nieefektywny. Aby nawiązać połączenie przez *TCP* konieczne jest trzykrokowe potwierdzenie (ang. *three-way handshake*). Przykład (rysunek 2-2) przedstawia przypadek, w którym węzeł 1 chce połączyć się z węzłem 2 w celu wysłania określonych danych. W tej sytuacji węzeł przeprowadza najpierw procedurę synchronizacji, a później, wraz z sygnałem *ACK* potwierdzającym procedurę synchronizacji, rozpoczyna wysyłanie danych. Łatwo zauważyć, że konieczne jest dwukrotne przesłanie danych do synchronizacji przez sieć (nie licząc samego procesu wysyłania danych), co niestety oznacza, że łączny czas ustanawiania połączenia jest w przybliżeniu równy dwukrotnemu czasowi propagacji. Na przykład, dla połączenia z Marsem (4-20 min. czasu propagacji w zależności od położenia obu planet) oznacza to średnio prawie pół godziny dodatkowego, niepotrzebnie straconego czasu. **[1,5,14,15]**



Rysunek 2-2: Ustanawianie połączenia w TCP.

2.1.4.3 Rozłączanie

Rozłączenie w ramach *TCP* składa się natomiast z czterostronnej komunikacji (ang. *four-way „Goodbye" handshake*). Jest to mniej kłopotliwe, niż nawiązywanie łączności, ponieważ same dane już dotarły do celu, a sesja *TCP* musi się tylko upewnić, że więcej danych już nie ma i że wszystkie segmenty zostały odebrane poprawnie. Nie jest to więc duży problem z punktu widzenia opóźnień, natomiast rodzi to duże problemy, kiedy w trakcie rozłączania pojawi się przerwanie — nawet jeśli przesłanie danych przebiegło prawidłowo i wszystkie segmenty zostały odebrane bez błędów, to bez rozłączenia sesji odbiornik nie jest w stanie stwierdzić, że dotarły już wszystkie dane. Zazwyczaj aplikacja korzystająca z połączenia dostaje w takiej sytuacji informację od systemu operacyjnego, że w trakcie połączenia nastąpił błąd, a nie że jest to koniec danych. **[1,5,14,15]**

2.1.4.4 Konieczność stosowania potwierdzeń

Jedną z najważniejszych cech *TCP*, która praktycznie wyklucza używanie tego protokołu w sieciach, w których występują duże opóźnienia, jest przesyłanie danych za pomocą „przesuwnego okna" (ang. *sliding window*). Polega to na tym, że *TCP* dzieli informację na mniejsze elementy zwane oknami, które dzielone są dopiero na pakiety i nie rozpocznie przesyłania kolejnego okna, dopóki nie dostanie informacji potwierdzającej poprawne odebranie danych przed odbiornik. Do momentu uzyskania takiego potwierdzenia nadajnik nie robi zupełnie nic. W kontekście wielominutowych opóźnień oznacza to, że znaczna większość czasu może być poświęcana na oczekiwanie na potwierdzenia. W czasie czekania na

potwierdzenie łącze jest zupełnie nie wykorzystywane, a mogłoby w tym czasie przesyłać dane. **[1,5,14,15]**

2.1.4.5 Wolny start i rozmiar okna

TCP opracowano dla środowiska, w którym opóźnienia są bardzo małe, dlatego wyposażony został w bardzo użyteczny mechanizm wolnego startu, który tuż po rozpoczęciu transmisji ustala rozmiar okna na bardzo małą wartość. W czasie transmisji okno to rośnie systematycznie, aż w którymś momencie bufor jednego z ruterów na drodze pakietów przepełni się ze względu na zbyt szybki przesył danych lub natłok spowodowany innymi transmisjami. W takiej sytuacji pakiety nadmiarowe są odrzucane, zaś do węzła wysyłającego dane zostanie zwrócony pakiet *ICMP* z informacją o odrzuceniu części pakietów (*ICMP Source Quench* — tłumienie źródła). W tym przypadku węzeł odbierający dane wyśle informację o otrzymaniu tylko części danych. Po upłynięciu pewnego okresu protokół *TCP* założy, że wysłane wcześniej dane nie dotarły w całości, ponieważ zostały odrzucone z powodu natłoku. Dlatego ograniczy rozmiar okna dwukrotnie i wyśle dane ponownie. **[1,5,14,15]**

Ten system jest bardzo efektywny w klasycznym *Internecie*, ponieważ pozwala dobrze dopasować szybkość transmisji poszczególnych połączeń do przepływności łącza. Niestety, przy czasach propagacji na poziomie minut rozpędzenie się *TCP* do możliwości łącza będzie zajmować całe godziny.

W kontekście sieci tolerujących opóźnienia i przerwania nie ma także co myśleć poważnie o wykorzystywaniu protokołu *ICMP* do przyspieszania transmisji (pakiety *ICMP* mogą być wykorzystywane do wysyłania informacji o odrzuceniu pakietów, dzięki czemu retransmisja zawartości okna może się odbyć wcześniej), ponieważ na drodze pakietów zazwyczaj nie ma ruterów, które mogłyby odrzucić dane i wysłać o tym informację. Dane są odrzucane zazwyczaj z powodu awarii któregoś z elementów sieci lub po stwierdzeniu przekłamań w pakiecie, co uniemożliwia odczytanie adresu węzła wysyłającego. Podobna sytuacja ma miejsce także w sieciach składających się z elementów uproszczonych, takich jak sieci sensorowe, w których węzły często nie obsługują *ICMP* ze względu na oszczędność kosztów produkcji i zużycia energii.

2.1.5 Działanie protokołu TCP przy przerwaniach

2.1.5.1 Ucinanie okien

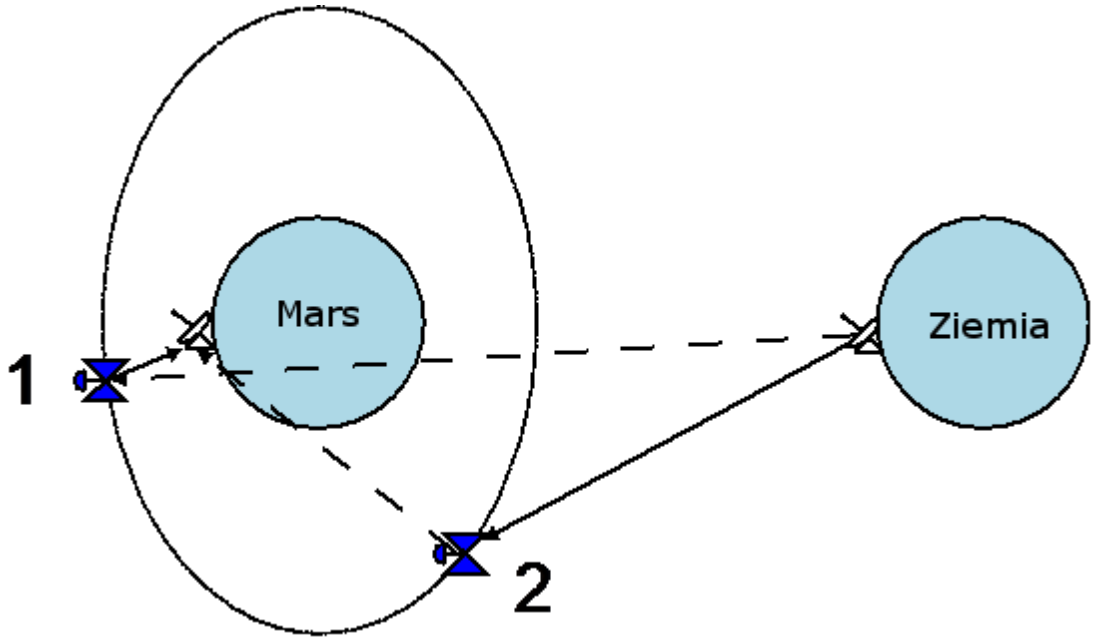
Dane w oknie transportowane są w segmentach, czyli porcjach danych. Jeden segment przesyłany jest w jednym pakiecie *IP*. Jeżeli okno nie dojdzie całe do odbiornika, tylko jeden z pierwszych segmentów zostanie odrzucony z powodu przekłamania w danych lub (co jest bardziej interesujące z naszego punktu widzenia) krótkiej przerwy w transmisji, to retransmisja obejmie ponownie prawie całe okno. Jest to spowodowane tym, że węzeł odbierający w wiadomościach potwierdzających odebranie danych wysyła numer ostatniego poprawnie odebranego segmentu. Tym razem jednak okno — zupełnie niepotrzebnie — zostanie pomniejszone, ponieważ nadajnik założy, że brak potwierdzenia to efekt przepełnienia się bufora rutera. W ten sposób przy częstych przerwach wypadkowa szybkość transmisji będzie wolniejsza w okresie, kiedy łącze będzie sprawne, mimo że zasoby będą w pełni dostępne. **[1,5,14,15]**

2.1.5.2 Retransmisje

Jeżeli szybkość transmisji miałaby być optymalna, to okna danych powinny być stosunkowo duże. Dzięki temu czas transmisji danych byłby dużo większy od czasu propagacji sygnału na łączu. I tutaj pojawia się problem: *TCP* w klasycznej wersji nie ma mechanizmu selektywnego potwierdzania przychodzących pakietów. Potwierdzany jest ostatni odebrany poprawnie pakiet, zaś dane retransmitowane są począwszy od następnego segmentu. Nie jest to złe rozwiązanie w tradycyjnej sieci internetowej, jednak w rozważanych środowiskach czas retransmisji może być bardzo duży. Rozwiązaniem tego problemu jest mechanizm *SACK* opisany dalej. Należy jednak pamiętać, że *SACK* nie jest obligatoryjnym rozwiązaniem w sieci internetowej oraz że nie rozwiązuje on wszystkich problemów związanych z naturą *TCP*, jaką jest transmisja strumienia danych. **[1,5,14,15]**

2.1.5.3 Połączenie typu koniec-koniec (ang. end-to-end)

Protokół *TCP* wymaga ze swojej natury bezpośredniego połączenia pomiędzy pierwszym a ostatnim węzłem. Zupełnie nie sprawdza się wtedy, kiedy na drodze połączenia występują cykliczne przerwy w transmisji. Jedną z typowych sytuacji, kiedy występują bardzo regularne przerwy, jest połączenie sondy na Marsie z odbiornikiem na Ziemi (rysunek 2-3). Pomiędzy nimi najczęściej występuje satelita krążący dookoła planety, który odbiera od sondy dane tylko wtedy, kiedy znajduje się nad nią (ze względu na ograniczenia w wielkości anten oraz mocy nadawczej najkorzystniej jest, kiedy satelita znajduje się stosunkowo blisko planety, co automatycznie oznacza dosyć szybkie krążenie satelity dookoła planety, aby zrównoważyć siłę ciążenia). Z kolei satelita może przesyłać dane w kierunku Ziemi tylko wtedy, kiedy znajduje się z odpowiedniej strony Marsa — jeżeli ten będzie znajdował się na linii w kierunku Ziemi, to transmisja jest niemożliwa. Gdy sonda znajduje się po przeciwnej stronie Marsa w stosunku do Ziemi, to bezpośrednie połączenie nigdy nie będzie możliwe, zaś transmisja może przebiegać



Rysunek 2-3: Przykład sytuacji, kiedy kontakt między nadajnikiem (Ziemia) i odbiornikiem (Mars) jest niemożliwy, ale istnieje pośrednie połączenie poprzez satelitę. Linia ciągła oznacza możliwość transmisji danych, linia przerywana oznacza brak takiej możliwości.

tylko za pomocą buforowania danych w satelicie i ich przesyłania w momencie uzyskania łączności z celem informacji. **[1,5,14,15]**

2.1.6 Próby ulepszenia TCP

2.1.6.1 SACK

Jedną z wad *TCP* jest konieczność retransmitowania całego okna w przypadku wystąpienia błędu w jednym z transmitowanych segmentów. W zastosowaniach, w których koszt transportu danych jest bardzo wysoki, a częste błędy w transmitowanych danych powodowałyby częste retransmisje całego okna, tradycyjna wersja protokołu *TCP* jest nieefektywna. Najlepszym przykładem jest transmisja satelitarna. Opóźnienie wynosi od kilku do około 100 milisekund w zależności od orbity satelity. Stosunkowo często występują błędy (przekłamania bitów) podczas transmisji. Koszt energetyczny transportu danych jest stosunkowo duży, szczególnie wtedy, kiedy satelita transmituje dane w kierunku Ziemi, ponieważ jedynym źródłem zasilania satelity są baterie słoneczne o ograniczonej mocy. Dla sieci o dużym koszcie retransmisji danych stworzono rozszerzenie *TCP* —*SACK* (ang. *Selective Acknowledgement* — selektywne potwierdzanie). Jego działanie polega na zmianie sposobu potwierdzania odebrania segmentów. Domyślnie *TCP* potwierdza ostatni poprawnie otrzymany segment. Wykorzystując *SACK*, węzeł odbierający dane wysyła potwierdzenie odebrania każdego z segmentów. Węzeł wysyłający otrzymuje informacje, które z segmentów zostały poprawnie odebrane i ma możliwość retransmisji tylko tych segmentów, które nie dotarły poprawnie do węzła odbierającego dane. *SACK* umożliwia zwiększenie rozmiaru okna, ponieważ ewentualna retransmisja nie oznacza już konieczności powtórnego wysyłania całego okna tylko pojedynczych segmentów. Daje to wzrost efektywnej szybkości transmisji w sieciach o dużym czasie propagacji danych. **[16]**

SACK jest zaimplementowany i obsługiwany w wielu rozwiązaniach stosowanych w tradycyjnej sieci *Internet*. Mechanizm ten został już zaimplementowany w popularnym systemie *Windows 98*. Nie jest jednak obligatoryjny i ciągle powstają jeszcze rozwiązania, które z niego nie korzystają (przykładem mogą być telefony komórkowe). *SACK* jest używany przede wszystkim w komunikacji satelitarnej i kilku środowiskach o podobnej charakterystyce. *SACK* umożliwia zwiększenie okna oraz efektywniejszą retransmisję danych, co sprawia, że w kontekście dużych opóźnień i częstych błędów transmisja staje się efektywniejsza od tradycyjnego *TCP*. *SACK* niestety nie sprawdza się w środowiskach o bardzo dużych opóźnieniach. W kontekście transmisji międzyplanetarnej czas propagacji sięga kilkudziesięciu minut. W celu zrozumienia, w czym tkwi problem z *TCP* i mechanizmem *SACK*, konieczne jest wybiegnięcie do przodu z informacjami przedstawionymi w tej pracy. W sieciach *DTN* dane przekazywane są w postaci niezależnych paczek danych. Podczas transportu paczek danych przez łącze *TCP* dane transmitowane są w postaci strumienia bajtów. Cechą *TCP* jest to, że przekazuje dane warstwie aplikacji dopiero po skompletowaniu ciągłej części strumienia danych. Wyobraźmy sobie, że przez sieć przesłane zostaje duże okno zawierające 100 paczek danych, które dzielone jest na pewną liczbę segmentów. Niestety, pierwszy segment w tym oknie zostaje uszkodzony w czasie transmisji i do odbiornika docierają tylko pozostałe segmenty. *TCP* potwierdzi otrzymane segmenty używając *SACK* i będzie oczekiwać na retransmisję brakującego segmentu. Jednak do momentu otrzymania tego segmentu dane nie zostaną przekazane warstwie aplikacji. 99

paczek danych, które mogłyby już być przetwarzane przez węzeł, oczekiwać będzie na jeden brakujący fragment. Zaś czas oczekiwania dla sieci *DTN* może sięgnąć wieku. **[16]**

2.1.7 Koncepcja sieci globalnej

Internet był wielkim sukcesem w łączeniu urządzeń telekomunikacyjnych na całym globie. Wykorzystywany stos protokołów *TCP/IP* zapewnia zadowalające wykorzystanie łączy, stosunkowo niski narzut informacji oraz bardzo tanie węzły pośredniczące w przekazywaniu danych (rutery). Jego powstanie rozpoczęło marzenia o jednolitej i taniej sieci przydatnej do wszystkich możliwych zastosowań komunikacyjnych. Wraz z kolejnymi rozszerzeniami i powstającymi protokołami stało się możliwe przesyłanie przez *Internet* zarówno sygnalizacji dla tradycyjnej telekomunikacji (*SIGTRAN*, ang. *SIGnaling TRANsport* — transport sygnalizacji), zastąpienie tradycyjnej telefonii telefonią *IP* (*VoIP*, ang. *Voice over IP* – głos przez *IP*), jak również pełne strumieniowanie mediów zarówno przez *broadcast* (radia internetowe), jak również przeznaczonych dla konkretnego użytkownika (ang. *Video on Demand* — wideo na życzenie). Jednak — jak to już zostało pokazane — klasyczny *Internet* nie wszędzie znajduje zastosowanie. Nie jest on np. odporny na przerwy w łączności czy też bardzo duże opóźnienia. Oprócz *Internetu* z konieczności musi istnieć wiele wyspecjalizowanych (i przez to dużo droższych) sieci, dostosowanych do bardziej wymagających środowisk. Sieci te radzą sobie m.in. z dużą liczbą błędów transmisji, celowymi zakłóceniami (aplikacje wojskowe), dużymi opóźnieniami, komunikacją jedynie w pewnych okresach czasu (np. satelity krążące dookoła planet) czy dużą asymetrią związaną z szybkością transmisji. **[2]**

Podsumowując: istnieje wiele sieci (nazwijmy je regionalnymi), w których proces przekazywania danych jest stosunkowo homogeniczny i sprawny. Dopóki sieci te nie wchodzą we wzajemną interakcję, nie występują problemy. Zaczynają się one wtedy, kiedy próbuje się przesyłać dane z jednego typu sieci do drugiego. Zapewnienie współpracy jest możliwe tylko dzięki agentom tłumaczącym jeden format przesyłania na drugi oraz dodatkowo działającym jako bufor dla niekompatybilnych opóźnień występujących w różnych sieciach.

Koncepcja sieci tolerującej opóźnienia i przerwania, czyli *DTN*, obejmuje m.in. pomysł stworzenia sieci globalnej, która umożliwi przesyłanie danych w uniwersalnym formacie między istniejącymi już sieciami regionalnymi. Oznacza to tworzenie dodatkowej warstwy w stosie protokołów, tworzącej paczki danych w formacie niezależnym od sieci. Paczki te będą przenoszone w sposób natywny (tzn. specyficzny dla danej sieci). Dla przykładu mobilne sondy na odległej planecie (np. Mars lub jeszcze odleglejszej) mogą przesyłać dane zgodnie z jednym protokołem transmisji do istniejącej stacji zbiorczej, ta z kolei może w innym formacie przesyłać dane do satelity okrążającego planetę. Trzecim sposobem przesyłania danych (trzecim typem sieci) jest transmisja międzyplanetarna. Oczywiście dane zostaną odebrane przez ziemskiego satelitę, który w sposób dostosowany do warunków lokalnych (takich jak atmosfera) wyśle te dane w odpowiedniej formie do nadajnika na Ziemi. Po odebraniu interesujących informacji nadajnik będzie próbować przekazać je do osób odpowiedzialnych za dany projekt, do czego najprawdopodobniej wykorzysta istniejącą i popularną sieć *Internet* wysyłając informacje np. przez e-mail. Każdy z opisanych etapów w tym przykładzie to inny typ sieci, inny sposób adresacji, korekcji błędów, a także inne charakterystyki czasowe.

Łatwo zauważyć, że sieci *DTN* muszą wymieniać dane w jednym uniwersalnym formacie i być gotowe do dopasowania się do różnych opóźnień i przerwań występujących w każdym z typów sieci. Konieczne jest także odwzorowanie adresów globalnych na adresy specyficzne dla danej sieci. Szczegółowo cechy sieci globalnej zostaną opisane w następnej sekcji.

2.1.8 Cechy sieci globalnej

Celem tego działu jest przedstawienie cech, jakimi musi się charakteryzować sieć globalna, aby spełniać swoją rolę. Są to jedynie ogólne założenia i koncepcje, bez zagłębiania się w szczegóły. W dalszych rozdziałach pokazane jest, w jaki sposób sieć *DTN* implementuje przedstawione tutaj wymagania w stosunku do sieci globalnej.

2.1.8.1 Paczki danych

Komutacja pakietów, dobrze znana z *Internetu*, okazała się bardzo efektywnym sposobem komunikacji. Polega ona na tym, że dane dzielone są na części, każda z tych części jest adresowana, a następnie przesyłana przez sieć. Podobne podejście jest najkorzystniejsze w sieci globalnej, gdzie dane przesyłane są w paczkach (ang. *bundle*). Paczki te jednak nie są pakietami znanym z sieci *IP*, są raczej wiadomościami wymienianymi pomiędzy węzłami. Każda sieć może stosować własne metody transportu paczek danych, tzn. każda sieć wykorzystuje dostępne w niej protokoły transportowe do transportu paczek danych. Protokół odpowiedzialny za podział danych na paczki musi znajdować się powyżej warstwy transportowej modelu stosu *TCP/IP* lub *ISO OSI*. Jednocześnie jest to warstwa niższa od warstwy aplikacji. Takie podejście sprawia, że konieczne jest wprowadzenie nowej warstwy rezydującej powyżej warstwy transportowej i poniżej warstwy aplikacji, która z założenia ma być obsługiwana tylko w węźle początkowym i końcowym (tabela 2-1). **[2]**

Poruszające się w sieci paczki danych mogą przechodzić fizycznie po różnych nośnikach (np. fale radiowe rozchodzące się w powietrzu, napięcie między dwoma przewodnikami w kablu, fala elektromagnetyczna wewnątrz światłowodu, fale dźwiękowe w wodzie itd.), różnych warstwach łącza danych (np. *Ethernet*, *PPP*, *802.11n*), po różnych protokołach warstwy sieciowej (np. *IP*, *AppleTalk*) i różnych protokołach warstwy transportowej (np. *TCP*, *UDP*, *SCTP*). Przez każdą sieć dane mogą być przenoszone w sposób charakterystyczny dla danej sieci (np. przy przechodzeniu przez sieć *IP* dane mogą całą drogę przemierzać za pośrednictwem jednego połączenia *TCP*), natomiast na stykach różnych sieci, a także w miejscach, gdzie chcemy zmienić np. protokół transportowy, konieczne jest umieszczenie ruterów warstwy sieci globalnej, tzn. takich, które trasują paczki danych sieci globalnej.

2.1.8.2 Agenci

Agenci to rutery nowej warstwy. Zajmują się przełączaniem paczek danych między sieciami. Obsługują także buforowanie danych. Sensem istnienia agentów jest scalanie dwóch sieci, czyli tłumaczenie protokołów warstw 1-3 modelu *TCP/IP* lub 1-4 *ISO OSI* na ich odpowiedniki w drugiej sieci. Bardzo ważnym aspektem związanym z działaniem agentów jest tłumaczenie uniwersalnej adresacji stosowanej w sieci globalnej na adresację stosowaną w danej sieci (adresacja jest opisana w dalszej części pracy). **[2]**

Tabela 2-1: Położenie warstwy sieci globalnej w modelu warstw TCP/IP. Jak widać, warstwa sieci globalnej może wykorzystywać wiele różnych protokołów ze wszystkich niższych warstw.

Warstwa	Przykładowe protokoły należące do danej warstwy
Aplikacji	▪ Aplikacja wykorzystująca protokół warstwy sieci globalnej
Sieci globalnej	▪ Bundle (opisany w dalszej części pracy)
Transportowa	▪ TCP ▪ UDP ▪ SCTP (ang. *Stream Control Transmission Protocol* — protokół sterowania transmisją strumieniową) ▪ LTP (opisany w dalszej części tej pracy)
Sieciowa	▪ IP
Łącza danych	▪ Ethernet ▪ PPP (ang. *Point to Point Protocol* — protokół łączący dwa punkty) ▪ 802.11g
Fizyczna	▪ Różne sposoby kodowania bitów

2.1.8.3 Odporność na przerwania i opóźnienia

Tworzenie sieci uniwersalnej oznacza, że musi być ona kompatybilna z wszystkimi typami sieci, czyli musi obsługiwać wszystkie najgorsze przypadki istniejące w obsługiwanych podtypach sieci, a co za tym idzie musi być odporna na przerwy w transmisji i zakłócenia. Łatwo można sobie wyobrazić sytuację, w której paczki danych przesyłane są w sieci międzyplanetarnej, gdzie opóźnienia sięgają nawet godzin, albo w której paczki danych przechodzą przez łącze mające kilkugodzinne opóźnienie i trafiają do węzła, którego pozostałe łącza są chwilowo niedostępne. Odrzucenie tych danych oznaczałoby konieczność powtórnego przesyłania danych przez wszystkie sieci na drodze połączenia, co wiązałoby się z dużymi kosztami, dlatego sieć globalna musi umieć sobie radzić z przechowywaniem danych (paczek) przynajmniej w najbardziej newralgicznych punktach sieci, gdzie spodziewamy się problemów z łącznością. **[2]**

2.1.8.4 Dzielenie sieci (ang. Network Partitioning)

Pojęcie dzielenia sieci jest znane także pod nazwą przerywanego kontaktu — w sieci globalnej zakłada się, że bezpośrednie połączenie między dwoma węzłami może w ogóle nie istnieć. Może natomiast istnieć przerywany kontakt, który polega na okresowej dostępności każdego łącza. Dane mogą być między węzłami przesyłane etapami — mogą zostać przetransportowane najdalej, jak to możliwe, a później oczekiwać na dostępność kolejnego fragmentu drogi. Łącza mogą być dostępne cyklicznie, przez co nigdy nie będzie istnieć bezpośrednie połączenie pomiędzy węzłami, ale będzie istnieć możliwość transportu danych poprzez ich magazynowanie na węzłach i oczekiwanie na dostępność kolejnego łącza. Takiej sytuacji nie przewiduje m.in. protokół *TCP*, gdzie bezpośrednie dwustronne połączenie musi być obecne w

czasie transmisji. Przerwanie jednego z łączy na drodze do celu prowadzi do podzielenia drogi do przebycia na kilka części i w efekcie do przerywanego kontaktu. Agenci tłumaczący jeden format transportu danych na inny przy okazji działają jako węzły przechowujące paczki danych. Tutaj dochodzimy do drugiego ważnego pojęcia: podejście magazynuj-i-przekaż (ang. *store-and-forward*). **[2]**

2.1.8.5 Magazynuj-i-przekaż

W tradycyjnej sieci *IP* pakiety po zrutowaniu natychmiast są wysyłane dalej. Jeśli węzeł docelowy nie jest dostępny (np. łącze prowadzące do niego uległo awarii), to dane natychmiast są odrzucane. Sieć *DTN* wprowadza inny sposób transmisji danych. Jeżeli połączenie nie istnieje po zrzutowaniu paczki danych, ponieważ chwilowo została przerwana łączność, to możliwe jest magazynowanie informacji aż do momentu wznowienia połączenia. Podobnie dane mogą być przetrzymywane w węźle aż do chwili uzyskania potwierdzenia ich odebrania przez kolejny węzeł, co usuwa konieczność istnienia połączenia „koniec-koniec". **[1,2]**

2.1.8.6 Zapewnianie niezawodności

W sieci globalnej konieczne jest założenie, że pośredniczące sieci mogą nie być doskonałe — mogą albo przekazywać dane z błędami, albo po drodze gubić paczki danych. Dlatego stosowane w niej protokoły muszą zarówno mieć sumy kontrolne *CRC* (ang. *Cyclic Redundancy Check* — suma kontrolna oparta na kodach cyklicznych), jak również dawać możliwość retransmisji uszkodzonych lub zgubionych paczek. Konieczne jest także założenie, że w wyniku uszkodzenia nie będą dostępne niektóre łącza. Ponieważ może się tak zdarzyć, że niedostępne będą wszystkie łącza prowadzące do celu i pochodzące z danej sieci, dlatego też sieć globalna musi być wyposażona we własne protokoły rutingu i dawać możliwość zmiany trasy na inną, czasem przechodzącą przez inną sieć. **[2]**

2.2 Sieci DTN

Rozwiązaniem wszystkich nakreślonych wyżej problemów wydaje się nowa koncepcja sieci tolerującej przerwania i opóźnienia. Zakłada ona, że gdy dane przesyłane są przez sieć, nastąpić może w dowolnym momencie wyłączenie lub uszkodzenie węzła, uszkodzenie łącza, zaś opóźnienia transmisji danych wprowadzane przez sieć praktycznie nie są ograniczone — sieć powinna dostosować się do istniejących warunków i przesyłać dane w takiej formie i ilości, jak jest to możliwe. **[1]**

W praktyce realizacja takiego rozwiązania polega na dzieleniu danych na mniejsze paczki i przesyłaniu ich w sieci, aż dane dotrą do adresata. Węzeł, który dostanie dane, a który nie ma możliwości przekierowania ich natychmiast do adresata np. ze względu na niedostępność łącza lub awarię następnego węzła, ma możliwość przechowania tych danych do momentu rozwiązania problemu lub zmiany rutingu w taki sposób, żeby dane zostały skierowane do adresata inną drogą. Jest to zupełnie inne zachowanie niż w przypadku istniejącej obecnie sieci internetowej, gdzie pakiet *IP* w momencie nie odnalezienia ścieżki do adresata zostałby natychmiast odrzucony, a informacja o tym w postaci komunikatu *ICMP* zostałaby odesłana do źródła danych. Te same dane mogą być wysyłane jednocześnie różnymi drogami, co często zwiększa prawdopodobieństwo dostarczenia danych, jednak zwiększa także zużycie zasobów sieci oraz wymusza konieczność sprawdzania, czy dany węzeł ma już określoną porcję danych. Aby osiągnąć ostatnią funkcjonalność każda paczka danych musi mieć przypisywany unikalny

identyfikator, który zapewnia, że wystarczy porównać ze sobą identyfikatory dwóch paczek, aby ustalić, czy zawierają te same dane.

Rozwijaniem i standaryzacją protokołów sieci *DTN* zajmuje się grupa *DTNRG* (ang. *DTN Research Group*). Aktualnie istnieją dwa główne protokoły sieci *DTN* **[9]**:

1. Protokół *Bundle*, który działa w warstwie sieci globalnej. Cechuje się tworzeniem logicznego połączenia pomiędzy węzłem końcowym i początkowym. Obejmuje zarówno adresację i dzielenie danych na paczki, jak i niezawodność dostarczania danych (odpowiednik 3 i 4 warstwy *ISO OSI*).

2. Protokół *Licklider* (zwany także *LTP*, *Licklider Transport Protocol*), którego działanie ogranicza się do zapewniania łączności między dwoma bezpośrednio połączonymi węzłami oraz zapewnienia niezawodnego dostarczenia danych przez to łącze. Poza tym ma także funkcje związane z optymalizacją szybkości transmisji (warstwy 2 oraz 4 modelu *ISO OSI*).

2.2.1 Odporność na opóźnienia

Odporność na opóźnienia na pojedynczym łączu realizowana jest przez protokoły transportowe, takie jak *LTP* (opisany dalej). Ze względu na niedostępność danego łącza czy węzła dane mogą być magazynowane w węzłach do momentu pojawienia się drogi do celu, co prowadzi do dalszych opóźnień. Łączny czas opóźnień może być stosunkowo duży, ale sieć *DTN* nie jest tworzona z myślą o szybkim transporcie, tylko raczej o niezawodnym dostarczeniu danych w trudnych warunkach. Limity czasowe ze względu na charakter sieci i duże opóźnienia między poszczególnymi węzłami zawsze ustawiane są na bardzo dużym poziomie, przy czym wartości te nie są specyfikowane w żadnym z protokołów — ze względu na możliwość funkcjonowania każdego protokołu w bardzo szerokim zakresie różnych sieci o bardzo różnych parametrach, w każdym przypadku sensowne mogą być zupełnie inne wartości. W skrajnych przypadkach (sieci międzyplanetarne) mogą to być nawet miesiące. W przypadku przekroczenia limitu czasowego dla pojedynczej paczki danych, jest ona odrzucana, co zapewnia protokół *Bundle*. **[1]**

2.2.2 Odporność na przerwania

Dane dzielone są na mniejsze paczki, które przesyłane są przez sieć niezależnie od siebie. Dzieleniem danych na paczki i gwarantowaniem niezawodnego ich dostarczenia zajmują się protokoły warstwy sieci globalnej, takie jak protokół *Bundle* (opisany dalej). Aby uniknąć odrzucania danych w momencie braku połączenia, dane są magazynowane w buforze węzła i przekazywane dalej, kiedy tylko pojawi się taka możliwość. Protokoły rutingu działające w sieci *DTN* są projektowane w taki sposób, aby umożliwić przesłanie wiadomości inną drogą, jeżeli założona wcześniej nie będzie dostępna. **[1]**

2.3 Protokoły dla sieci DTN

2.3.1 Protokół Bundle

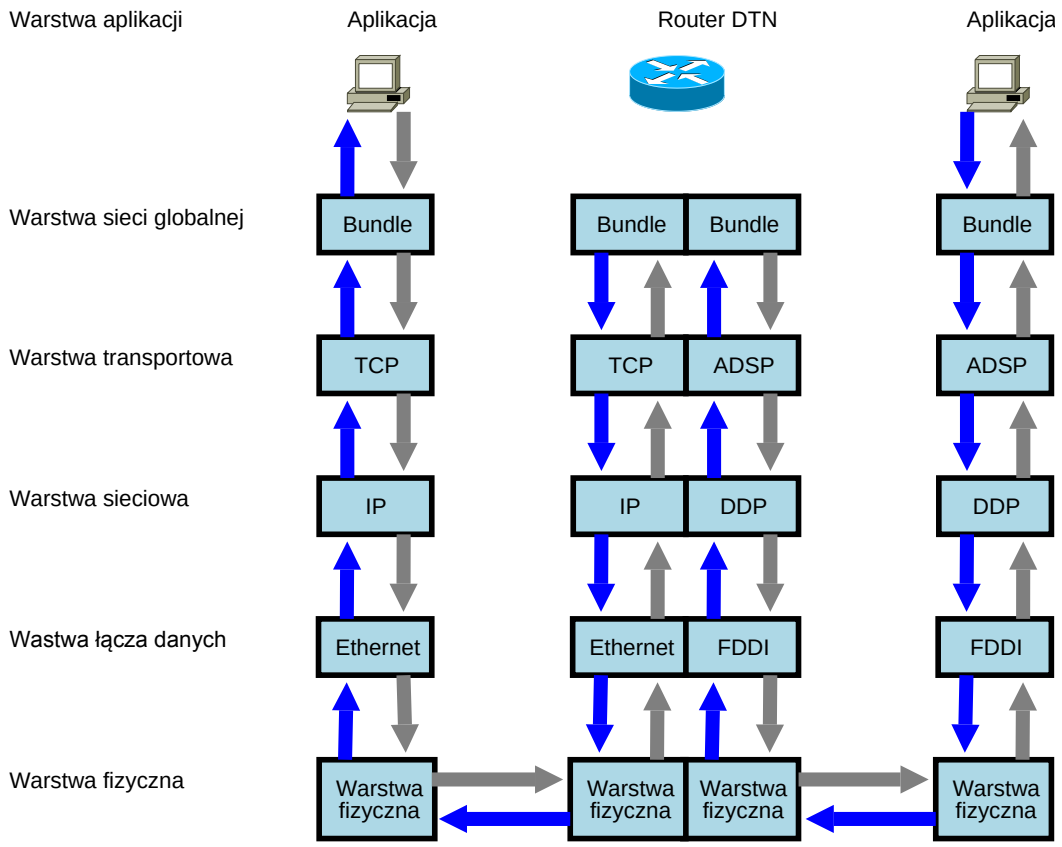
Protokół *Bundle* (ang. *bundle* — paczka, tobołek) to protokół, którego głównym zadaniem jest podział danych na paczki (odpowiednik pakietów z sieci *IP*). Odpowiada on także za przeprowadzanie potwierdzeń i retransmisji. Protokół *Bundle* zdefiniowany jest w dokumencie

IETF RFC 5050. Dokument ten ma status dokumentu roboczego (ang. *draft*), co oznacza, że przed stworzeniem oficjalnego standardu mogą zostać w nim wprowadzone zmiany. **[8]**

2.3.1.1 Warstwa działania

Protokół *Bundle* w pewnym sensie tworzy nową warstwę w modelu *ISO OSI*; znajduje się pomiędzy warstwą transportową, a warstwą sesji. Główną ideą stojącą za protokołem *Bundle* jest wykorzystanie już istniejących sieci, a więc istniejącej adresacji i protokołów transportowych. Dlatego ma kilka cech charakterystycznych dla już istniejących niższych warstw: niezależną adresację hostów (warstwa 3) oraz zapewnianie retransmisji uszkodzonych lub zgubionych pakietów — potwierdzanie odebrania pakietu (typowa funkcja warstwy 4, chociaż czasem podobną funkcję spełnia także warstwa 2 potwierdzająca odebranie ramek). **[1,2,14]**

W czasie transportu dane znajdujące się w paczkach mogą się przemieszczać między różnymi typami sieci, np. z protokołów wykorzystywanych w sieciach sensorowych mogą być przenoszone przez protokół *IP*. W tym celu adresacja zawarta w paczkach jest odwzorowywana na specyficzną dla danej sieci i na podstawie tego adresu następuje dalsze rutowanie paczek danych. Rysunek 2-4 przedstawia przykład przechodzenia pakietu *Bundle* pomiędzy sieciami opartymi na *IP* oraz *AppleTalk*. Węzeł początkowy znajdujący się w sieci *IP* próbuje przesłać dane do węzła w sieci *AppleTalk*. W tym celu tworzy pakiety *Bundle*, a następnie rutuje je.



Rysunek 2-4: Przykład przechodzenia paczki danych przez różne protokoły stosowane w sieci oraz tłumaczenia protokołów przez agentów. Na rysunku — patrząc od lewej strony — paczki danych przychodzą do rutera przez stos TCP/IP, wychodzą zaś przez stos protokołu AppleTalk. Na podstawie [2].

Rezultatem rutingu jest odnalezienie drogi prowadzącej przez ruter *DTN*. Aby przesłać do niego dane węzeł początkowy łączy się za pomocą protokołu *TCP* z ruterem i przekazuje mu pakiety *Bundle*. Ruter dokonuje rutingu otrzymanych pakietów i podłącza się za pomocą protokołu *ADSP* (odpowiednik *TCP* w sieciach *AppleTalk*) do węzła końcowego. Następnie przekazuje mu pakiety *Bundle*. Dane przenoszone są przez protokół *DDP* (ang. *Datagram Delivery Protocol* — protokół dostarczający data gramy), który jest odpowiednikiem protokołu *IP*. **[2,11]**

2.3.1.2 Adresacja

Węzły końcowe identyfikowane są za pomocą identyfikatorów punktów końcowych *EID* (ang. *Endpoint Identifier*), które powinny być unikalne globalnie. Obecnie wykorzystywane są adresy *URL* (ang. *Universal Resource Lokator*), czyli nazwy tekstowe postaci „dtn://nazwaserwera.pl" lub podobne. Adresy te są odwzorowywane na adresy specyficzne dla danej sieci. **[1, 8]**

2.3.1.3 Kontakt

Bundle korzysta z pojęcia „kontaktu". Kontakt między dwoma najbliższymi węzłami istnieje wtedy, kiedy istnieje możliwość wysyłania między nimi danych bez oczekiwania (tzn. łącze i drugi węzeł są zdatne, a wysyłane dane docierają do odbiorcy natychmiastowo, jedynym ograniczeniem jest tylko czas propagacji i transmisji danych). Należy nadmienić, że kontakt może być ustalany przez inne warstwy, np. wykrycie braku nośnej realizowane przez warstwę łącza danych lub też „odgórne" ustalenie, że dwa węzły będą się komunikować tylko w określonych porach, np. satelita Marsa ma łączność z Ziemią tylko wtedy, kiedy nie jest zasłonięty przez planetę, co oznacza, że obydwie strony wiedzą, kiedy nastąpi rozłączenie i powtórne połączenie. **[1,9]**

2.3.1.4 Czas życia

Każdy z pakietów ma przypisany czas ważności. Jeżeli czas ten upłynie, to pakiet jest traktowany jako nieaktualny i odrzucany. Jest to odpowiednik *TTL* (ang. *Time To Live*) znanego z sieci *IP*. Jednak wartością nie jest liczba pozostałych przeskoków, ale moment czasowy, kiedy paczka jest odrzucana. Takie rozwiązanie wymusza synchronizację czasu w sieci *DTN*. **[1,8,9]**

2.3.1.5 Przekazywanie odpowiedzialności za retransmisję

Inną ważną cechą protokołu *Bundle* jest przekazywanie odpowiedzialności za niezawodne dostarczenie informacji. Niektóre węzły (ang. *custodians*) odebrawszy pewną porcję informacji mogą odesłać potwierdzenie odebrania danych do węzła początkowego i przejąć odpowiedzialność za ewentualną retransmisję paczek danych w sytuacji, gdyby wystąpiły jakieś problemy z ich dalszym doręczeniem. Zabezpiecza to przed sytuacją, w której za retransmisję nadal odpowiedzialny byłby węzeł początkowy, który może znajdować się dużo dalej (w sensie czasowym i przestrzennym). Należy pamiętać, że na łączach w sieci *DTN* mogą występować przerwania i duże opóźnienia, co bardzo niekorzystnie odbijałoby się na wydajności całej sieci, gdyby retransmisje odbywały się zawsze od węzła początkowego do końcowego. Przy obecnym podejściu możliwe jest np. w komunikacji międzyplanetarnej ustalenie, że satelita krążący wokół Marsa będzie węzłem przyjmującym odpowiedzialność za dalsze niezawodne dostarczanie danych. Krążąc dookoła planety może on raz odbierać dane od nadajnika na planecie, a kiedy indziej transmitować je w kierunku Ziemi. Dzięki przejmowaniu odpowiedzialności za retransmisję, ewentualne powtórne wysłanie informacji będzie się

odbywać tylko pomiędzy Marsem a nadajnikiem lub pomiędzy nadajnikiem a Ziemią, ale nigdy nie będzie mieć miejsca retransmisja od Marsa do Ziemi. **[1,8]**

2.3.1.6 Raporty

Kolejną cechą protokołu *Bundle* jest tworzenie raportów, czyli sygnalizacja wydarzeń związanych z określoną paczką danych. Do paczek danych dodawane jest specjalne pole, które zawiera listę węzłów oraz flagi wydarzeń, o których powinny być poinformowane dane węzły. Aktualnie obsługiwane wydarzenia to:

- odebranie paczki danych przez węzeł końcowy,
- przejęcie odpowiedzialności za retransmisję danych przez węzeł,
- przekierowanie paczki danych,
- dostarczenie paczki danych — wysyłane kiedy paczka danych jest dostarczana do swojego miejsca przeznaczenia (przez instancję protokołu *Bundle*, który jest „połączony" z aplikacją),
- usunięcie paczki danych w sytuacji błędu lub przekroczenia czasu życia.

W zależności od poziomu niezawodności, jaki chce zapewnić aplikacja, może ona różnie poustawiać odpowiednie flagi i w efekcie być informowana o tym, czy np. paczka danych dotarła na miejsce (czy też nie) i w razie potrzeby retransmitować dane. Wszystkie raporty wysyłane są w nowych paczkach danych, przy czym zakłada się, że nowy raport nie może być generowany jako efekt działania na raportach (podobna sytuacja ma miejsce w protokole *ICMP*, zabezpiecza to przed wysyłaniem raportów o raportach o raportach itd.). **[1,8]**

2.3.1.7 Słownik

W protokole Bundle adresy węzłów końcowych mogą być stosunkowo długie, a możliwe jest, że dany adres jest wykorzystywany w kilku miejscach np. jako źródło paczki danych oraz jako węzeł, który powinien być informowany o usunięciu czy dostarczeniu paczki danych. Wielokrotne podawanie tego samego adresu wprowadza niepotrzebny narzut, dlatego wprowadzono słownik, czyli proste odwzorowanie krótkiego identyfikatora liczbowego na długi adres. Dalej liczba ta będzie określana mianem odwołania, identyfikatora lub krótko *ID*. W paczce danych nigdzie nie jest podawany bezpośrednio adres węzła, tylko wcześniej opisane odwołanie. **[1,8]**

2.3.1.8 Nagłówek

Tabela 2-2 przedstawia budowę nagłówka protokołu *Bundle*.

Dodatkowe informacje o elementach nagłówka:

Flagi przetwarzania — określają, czy dana paczka danych jest tylko fragmentem (w przypadku sieci o ramkach mniejszych od paczki danych może ona zostać podzielona na części), czy zawiera dane itp.

Flagi *Class Of Service* (ang. klasa usług) —określają priorytet paczki danych.

Flagi *Status Report Request* (ang. żądanie raportu statusu) —określają, w reakcji na jakie wydarzenia mają być wysyłane raporty.

Tabela 2-2: Budowa głównego nagłówka protokołu Bundle. Na podstawie [9].

		Bajt			
		1	2	3	4
Wiersz	1	Wersja	Flagi przetwarzania	Flagi COS (Class of Service)	Flagi SRR (Status Report Request)
	2	Długość nagłówka (w bajtach)			
	3	Odwołanie do węzła docelowego			
	4	Odwołanie do węzła źródłowego			
	5	Odwołanie do węzła odbierającego raport			
	6	Odwołanie do węzła przejmującego odpowiedzialność za niezawodne dostarczenie			
	7	Czas utworzenia			
	8				
	9	Czas życia			
	10	Długość słownika			
	11..*N*	Tablica słownika			
	N + 1	Wielkość fragmentu (opcjonalne)			
	N + 2	Wielkość danych aplikacji (opcjonalne)			

2.3.1.9 *Rozszerzenia*

Każda paczka danych składa się z głównego nagłówka oraz pewnej liczby nagłówków rozszerzeń, np. nagłówka danych czy nagłówka zabezpieczeń. Kolejne rozszerzenia mogą być dodawane w momencie zgłoszenia zapotrzebowania na nie. Dodatkowe nagłówki umieszczane są w pakiecie *Bundle* tuż po głównym nagłówku. Ostatnim nagłówkiem jest zazwyczaj nagłówek danych (aczkolwiek dane mogą nie być wysyłane np. w paczkach przenoszących potwierdzenia). **[1,9]**

2.3.2 Protokół Licklider

LTP (*Licklider Transmission Protocol*) jest protokołem pośrednim między protokołem transportowym a protokołem łącza danych, chociaż działa często opierając się np. na *UDP*. Protokół *LTP* zdefiniowany został w dokumencie IETF RFC 5326. Dokument ten ciągle ma status roboczy. *LTP* umożliwia przesyłanie danych pomiędzy dwoma bezpośrednio połączonymi węzłami w sytuacji, gdy istnieją między nimi bardzo duże opóźnienia lub częste przerwania transmisji. Ważnym aspektem *LTP* jest możliwość zawieszenia połączenia (np. potwierdzeń) w sytuacji, gdy wiadomo, że nastąpi przerwa w łączności. **[1,18]**

W szczególności najistotniejszymi cechami *LTP* są:

2.3.2.1 *Segmentacja*

Gdy *LTP* otrzymuje blok danych z wyższych warstw, dzieli go na segmenty. Następnie przesyłane one są przez sieć. Węzeł docelowy może generować potwierdzenia, aby zapewnić niezawodne dostarczenie danych. Ostatni segment składający się na blok danych jest oznaczany jako koniec bloku. Jego otrzymanie sprawia, że węzeł docelowy wysyła raport z

odbioru bloku, w którym określa, które dane udało mu się poprawnie otrzymać. Otrzymanie takiego raportu oczywiście powoduje retransmisję wszystkich brakujących danych. **[1,7]**

2.3.2.2 Zapewnianie niezawodności

Ponieważ różne dane mają różne priorytety, *LTP* pozwala określić tzw. czerwone i zielone segmenty, czyli obszary danych, które muszą być dostarczone niezawodnie i są w razie błędu retransmitowane oraz takie, których błąd jest do zaakceptowania i które nie są retransmitowane. **[1,7,18]**

2.3.2.3 Segmenty anulujące

W przypadku zbyt wielu błędów lub gdy zbyt wiele razy nie nastąpiła reakcja z drugiej strony w wyznaczonym czasie, możliwe jest zerwanie sesji *LTP* Służą do tego tzw. segmenty anulujące (ang. *cancellation segments*). **[1, 18]**

2.3.2.4 Limity czasowe (ang. timeout)

LTP wykorzystuje potwierdzenia odebrania danych i automatycznie ustala pewne limity czasowe, po których dane są retransmitowane lub sesja jest uznawana za zerwaną. Jednak *LTP* ze swojej natury zakłada, że mogą istnieć z góry ustalone przerwy w komunikacji (np. noc dla sieci sensorowych, które zasilane są energią z baterii słonecznych). W skrajnych sytuacjach czas przeznaczony na transmisję może wynosić na przykład 1 minutę co godzinę. Dla *LTP* ilość czasu, jaka upłynęła, jest sumą okresów, kiedy powinno istnieć połączenie. Jeśli więc na odpowiedź przeznaczonych jest np. 5 minut, to w podanym przykładzie tak naprawdę czas, po jakim protokół uzna, że nie nastąpiło potwierdzenie, wyniesie 5 godzin. Czas mierzony jest bardziej jako czas trwania obustronnego połączenia niż jako rzeczywisty upływ czasu. **[1, 18]**

Ze względu na możliwość wykorzystania *LTP* w różnych środowiskach, specyfikacja nie definiuje żadnych wartości limitów czasowych. Implementacja przeznaczona do konkretnej sieci może dopasować te wartości, a także np. dynamicznie uzależnić je od różnych parametrów, np. jakości łącza. **[7]**

2.3.2.5 Rozszerzalność

Podobnie jak protokół *Bundle*, *LTP* ma mechanizmy umożliwiające rozszerzanie tego protokołu. Istnieje możliwość dodawania dowolnej liczby nagłówków (doczepiane są one przed wysyłanymi danymi) oraz końcówek (ang. *trailer* — dosłownie przyczepa, dodawane są po wysłanych danych). Zdefiniowane już rozszerzenia umożliwiają m.in. sprawdzanie integralności danych (lepsze niż standardowe sumy kontrolne *CRC*), uwierzytelnianie, czy ciasteczka (ang. *cookies*). **[1,19]**

2.3.2.6 Struktura ramki

Tabela 2-3 przedstawia strukturę ramki protokołu *LTP*.

Tabela 2-3: Struktura ramki protokołu LTP. Na podstawie [7].

		Bity	
		pierwsze 4bity	kolejne 4 bity
Numer bajtu	1	Numer wersji	Flagi typu segmentu
	2	Identyfikator sesji	
	3		
	4	Liczba nagłówków	Liczba końcówek
	5..*M*	Nagłówki	
	M+1..*N*	Zawartość segmentu	
	N+1..*O*	Końcówki	

2.3.2.7 Zawartość segmentu

Zawartość segmentu zawiera nie tylko same dane przesyłane pomiędzy węzłami, ale także metainformacje związane z zawartością. M.in. zawiera informacje o tym, który fragment transmitowanego bloku danych znajduje się w ramce *LTP*. Informacja ta ma postać numeru w bloku danych pierwszego transmitowanego bajtu danych (ang. *offset*) oraz liczby transmitowanych danych.

2.3.2.8 Struktura rozszerzeń

Wszystkie rozszerzenia mają prostą strukturę opartą na dobrze znanym w telekomunikacji schemacie *TLV* (ang. *Type-Length-Value* — typ-długość-wartość). Pierwszy bajt oznacza typ rozszerzenia, kolejne bajty to długość, ostatecznie pojawia się wartość rozszerzenia. **[7,18]**

2.3.3 Protokoły alternatywne

Mimo że *Bundle* i *LTP* są zdecydowanie najpopularniejszymi protokołami proponowanymi dla sieci tolerujących przerwania i opóźnienia, istnieją także pomysły alternatywne. Jednym z nich jest stworzenie raczej nowej warstwy opartej na protokole *UDP* (jak wiadomo, na tym protokole już obecnie funkcjonują kolejne warstwy, np. *RTP* — protokół służący strumieniowaniu mediów). Najpoważniejszym konkurentem dla obecnej struktury sieci *DTN* wydaje się stworzenie protokołu, który dawałby podobne możliwości jak protokół *Bundle*, ale z drugiej strony nie wymagał modyfikacji istniejącej sieci internetowej i zużywałby prawdopodobnie mniej zasobów od protokołu *Bundle*. Należy się spodziewać, że takie rozwiązanie byłoby zdecydowanie tańsze. **[4]**

Koncepcja taka ciągle jest w fazie rozważania, nie udało się jeszcze rozwiązać kilku istotnych problemów, takich jak czas ważności adresów *IP* (które w sieci internetowej często są przydzielane tymczasowo przez np. protokół *DHCP* i mogą się zmieniać, zaś jako stałe identyfikatory używane są raczej nazwy domenowe) czy działanie serwerów *DNS* w kontekście dużych opóźnień. **[4]**

2.4 Niezawodność w sieciach DTN

Problematykę niezawodności w sieciach DTN można podzielić na kilka składowych:

- zapewnianie istnienia drogi łączącej dwa węzły,
- zapewnianie niezawodności dostarczenia paczek danych,
- zapewnianie, że dostarczone dane nie zawierają przekłamań.

2.4.1 Zapewnianie istnienia drogi łączącej dwa węzły

Zapewnianie istnienia drogi łączącej dwa węzły jest związane z zapobieganiem rezultatów uszkodzeń węzłów istniejących w sieci, a także (w przypadku np. sieci sensorowych) wyłączeń urządzeń spowodowanym np. brakiem energii z baterii słonecznych w nocy. Podobne problemy mogą wystąpić z istniejącymi połączeniami, np. kabel łączący węzły może ulec przerwaniu lub burza może sprawić, że kanał radiowy nie będzie dostępny. Podobne problemy bardzo często występują w sieciach militarnych, gdzie węzły w postaci nadajników w pojazdach lub kieszeni żołnierza przemieszczają się po zróżnicowanym terenie, co często prowadzi do ograniczenia transmisji spowodowanego zasłonięciem anten przez górę lub budynki. Dlatego też w momencie, kiedy jeden węzeł będzie próbował wysłać wiadomość do drugiego węzła znajdującego się po drugiej stronie sieci, może się okazać, że jedno z łączy lub jeden z węzłów, przez które przebiega połączenie, jest niedostępny. Tego typu problemy można rozwiązać na dwa sposoby: albo zwiększać niezawodność urządzeń, powielając poszczególne elementy (dzięki czemu zmniejszy się szansa, że urządzenia lub łącza będą niedostępne) lub zwiększając wymagania projektowe (np. większa moc anten może znacznie zmniejszyć prawdopodobieństwo zerwania transmisji przy przesłonięciu trasy sygnału przez jakiś budynek), albo zapewniając taki protokół rutingu, który wykryje ewentualną niedostępność łącza czy węzła i przekieruje przychodzące dane przez inne istniejące połączenia.

Niniejsza praca nie zajmuje się zagadnieniami związanymi z polepszaniem fizycznej niezawodności urządzeń i łączy. Skupia się na szukaniu rozwiązań przy istniejących parametrach fizycznych sprzętu. W szczególności rozpatrywane będą różne podejścia do rutingu i wpływ opóźnień oraz przerwań na sprawność dostarczania danych w zależności od przyjętego protokołu rutingu i typu sieci, w której jest on uruchomiony. Można założyć, że protokoły rutingu będą się sprawdzać w różny sposób w zależności od struktury sieci.

2.4.2 Zapewnianie niezawodnego dostarczania paczek danych

Zapewnienie niezawodnego dostarczania paczek danych jest bezpośrednio związane z działającym protokołem warstwy transportowej (lub podobnymi mechanizmami istniejącymi w innych warstwach). W typowej sieci dane dzielone są na różnego typu paczki — ramki, pakiety itp. Są to więc fragmenty większej porcji danych, które są dostarczane niezależnie lub tylko częściowo zależnie od siebie. Np. w sieci *IP* dane dzielone są na segmenty przez protokół *TCP* i przesyłane w osobnych pakietach *IP*. Oczywiście istnieje szansa, że pakiety zostaną albo zgubione w sieci (np. protokół warstwy łącza może odrzucić przychodzącą ramkę w sytuacji, gdy wykryje uszkodzenie danych, możliwe jest także, że odrzucenie nastąpi w węźle w wyniku przepełnienia się bufora). W takiej sytuacji *TCP* nie dostaje potwierdzenia otrzymania określonych danych i rozpoczyna retransmisję. I to jest właśnie działanie zmierzające do niezawodnego dostarczenia paczek danych, tzn. zapewnienia, że używany protokół będzie przetrzymywał dane do chwili uzyskania potwierdzenia, a w razie jego nieotrzymania nastąpi retransmisja.

Retransmisje w sieci występują w praktyce na więcej niż jednym poziomie. Np. transmitowany pakiet protokołu *Bundle* może być transmitowany pomiędzy ruterami warstwy sieci globalnej przy pomocy protokołów transportowych gwarantujących niezawodne dostarczenie danych

(np. *TCP*). W ten sposób w sieci występują zarówno retransmisje zgubionych segmentów protokołu *TCP*, jak również retransmisje pakietów *Bundle*.

Analizując działanie protokołów rutingu i protokołu *Bundle* można zauważyć, że występuje współdziałanie między zapewnianiem niezawodności na tym poziomie, a protokołem rutingu. Niektóre protokoły rutingu (np. opisany w dalszej części pracy protokół epidemiczny) tworzą wiele kopii wiadomości i tym samym zwiększają niezawodność dostarczenia danej paczki do adresata. Nie można też nie zauważyć, że niektóre protokoły rutingu zwiększają stopień obciążenia sieci (np. przepełnianie się bufora) i tym samym mogą wpływać na odrzucanie paczek danych. Dlatego ten poziom zapewniania niezawodności będzie uwzględniany w tej pracy, a działanie protokołu *Bundle* zostanie przebadane symulacyjnie.

2.4.3 Zapewnianie dostarczania danych bez przekłamań

Nawet wtedy, kiedy wszystkie paczki dotrą na miejsce, nie mamy jeszcze gwarancji, że dane zostały poprawnie dostarczone. W trakcie transportu danych przez sieć mogły nastąpić różnego typu przekłamania. Występują trzy typowe podejścia do tego zagadnienia:

- Akceptacja uszkodzonych danych — np. w sieci *PSTN* (ang. *Public Switched Telephon Network* — publiczna komutowana sieć telefoniczna), czyli w tradycyjnej sieci telefonicznej, dane w ogóle nie są weryfikowane i cokolwiek pojawi się w kanale, kierowane jest do słuchawki użytkownika. Ewentualne zakłócenia nie mają większego znaczenia, ponieważ użytkownik telefonu i tak nie dostrzeże niewielkich szumów.
- Kody nadmiarowe — czasami wykorzystywane są kody nadmiarowe, które dodają do przesyłanych danych nadmiarowe informacje umożliwiające korekcję ewentualnych błędów. To podejście jest szczególnie korzystne w sieciach z bardzo dużym opóźnieniem występującym w sieciach *DTN* — w razie błędów nie jest konieczna czasochłonna retransmisja, poprawne dane da się odtworzyć na podstawie tego, co dotarło.

  W tej pracy interesujące jest szczególnie samo dostarczanie danych, nie zaś zaawansowane metody kodowania nadmiarowego, dlatego ten temat zostanie pominięty. W razie potrzeby dane nadmiarowe mogą być dodawane przez warstwy wyższe. Ta praca będzie analizować tylko odsetek dostarczonych danych oraz liczbę przekłamań występujących w dostarczonych danych.

  Należy jeszcze dodać, że przekłamania mogą występować zarówno w danych, jak i w nagłówkach danych, co najczęściej skutkuje odrzuceniem całej paczki. W ten sposób możliwe jest nie tyle uszkodzenie danych, co odrzucenie całej paczki.

- Brak akceptacji dla uszkodzeń i odrzucanie nieprawidłowych danych — w takiej sytuacji dane są zawsze retransmitowane aż do uzyskania prawidłowych danych bez uszkodzeń, albo do momentu przekroczenia pewnej liczby powtórzeń, co najczęściej kończy w ogóle sesję. Ten fakt jest szczególnie interesujący z punktu widzenia badania metod wznawiania pracy w sieciach *DTN* i dlatego przeprowadzona zostanie symulacja retransmisji we wszystkich istniejących warstwach.

2.5 Ruting

Jednym z ważniejszych problemów w sieciach *DTN* jest ruting, czyli algorytm decydujący o kierowaniu otrzymanych informacji do dostępnych kanałów transmisyjnych. W sieci *DTN* — nietypowo — protokół rutingu podejmuje także decyzję o przechowywaniu kopii danych do momentu pojawienia się łączności oraz o zwielokrotnianiu wiadomości. Protokoły rutingu muszą wykrywać przerwania transmisji w sieci (wynikłe z uszkodzeń bądź wyłączenia węzłów lub łączy) i kierować dane w taki sposób, żeby jednak dotarły one do miejsca przeznaczenia. Całość komplikuje fakt, że z powodu dużych opóźnień węzeł nigdy nie wie, jaka jest aktualna sytuacja w sieci; przy założeniu, że stosowany protokół rutingu rozpropagowuje informacje o stanie sieci, informacje docierają do innych węzłów w przybliżeniu po czasie propagacji łącza. Poza tym dane w sieciach *DTN* mogą być składowane do momentu wznowienia połączenia i ich przekierowywanie jest silnie uzależnione od sytuacji w sieci. Nie zostały jeszcze zdefiniowane żadne uniwersalne protokoły, które radziłyby sobie z większością występujących w sieciach *DTN* sytuacji, natomiast zaprezentowane zostały niektóre z podejść, które wydają się szczególnie przydatne w niektórych przypadkach szczególnych. Potencjalne zastosowania sieci *DTN* są tak różnorodne, że bardzo trudne wydaje się znalezienie jednego protokołu rutingu, który potrafiłby sobie poradzić ze wszystkimi występującymi w sieci sytuacjami. Prace nad uniwersalnym protokołem rutingu ciągle trwają i znane są najczęściej sprawdzające się podejścia do rutingu, które — niestety — nie są jeszcze zadowalające.

2.5.1 Składowanie danych

Jednym z typowych problemów pojawiających się w Siechach *DTN* jest postępowanie z danymi zaadresowanymi do węzła, który jest aktualnie niedostępny. Dwa główne podejścia do rozwiązania tego zagadnienia, to ruting optymistyczny i ruting pesymistyczny. **[3]**

2.5.1.1 Ruting optymistyczny

Ruting optymistyczny zakłada, że kiedy nie istnieje znane połączenie z celem, to dane są przechowywane w buforze. Optymistycznie zakłada się, że za chwilę ponownie pojawi się łączność i że dane będą mogły być przesłane dalej. To podejście wydaje się bardziej efektywne, szczególnie w sieciach *DTN*, jednak nakłada ogromne wymagania co do ilości pamięci w węzłach. **[3]**

2.5.1.2 Ruting pesymistyczny

Ruting pesymistyczny zakłada, ze skoro obecnie ścieżka nie jest znana, to dane należy natychmiast odrzucić. To podejście jest stosowane w sieciach opartych na *IP*. **[3]**

2.5.2 W kierunku uniwersalnego protokołu rutingu

W typowej sytuacji protokół rutingu musi spełnić kilka wymagań, które decydują o tym, czy jego stosowanie będzie efektywne, czy też w jakiś sposób zasoby będą marnowane, a dane nie zawsze będą docierały do miejsca przeznaczenia. Oczywiście nie jest możliwe stworzenie jednego protokołu rutingu, który nadawałby się do wszystkich zastosowań, jednak łatwo zauważyć, że w sieciach *DTN* znaczną część sieci stanowią bardzo podobne do siebie rozwiązania a stworzenie jednego, parametryzowanego protokołu rutingu umożliwi znaczne uproszczenie struktury sieci oraz realnych kosztów (implementacja, szkolenia dla serwisantów, znajomość typowych problemów). Taki protokół musiałby spełniać następujące wymagania**[3]**:

2.5.2.1 Samokonfigurowalność

Protokół rutingu powinien po podłączeniu i podstawowym skonfigurowaniu (np. wybranie protokołu rutingu i sąsiadów) sam nawiązać łączność i od działających ruterów pobrać informację o topologii sieci i kierunkach przekierowywania paczek danych. W innym przypadku administrator po wprowadzeniu nawet drobnej zmiany w sieci musiałby modyfikować konfigurację wszystkich ruterów, co byłoby bardzo pracochłonne. Funkcja taka jest także potrzebna do dynamicznego wykrywania uszkodzeń i przekazywania innym węzłom informacji o aktualnie dostępnych ruterach i ścieżkach. **[3]**

2.5.2.2 Adaptacja do różnych parametrów sieci

Protokół rutingu musi zapewniać wysoką wydajność w szerokim zakresie typów łączności (między ruterami mogą istnieć zarówno trwałe łącza internetowe, łącza o dużej liczbie przerwań, dużym opóźnieniu, a także o łączności istniejącej tylko w pewnych określonych momentach czasowych). Gwarantuje to, że będzie możliwe połączenie sieci o różnych charakterystykach bez potrzeby zmieniania „po drodze" protokołu rutingu i bez potrzeby analizowania, który w danej sytuacji będzie korzystny. **[3]**

2.5.2.3 Niskie zużycie zasobów sieciowych

Protokół rutingu powinien efektywnie korzystać z zasobów sieciowych, w tym z**[3]**:

- buforów — nadmierne magazynowanie informacji w węzłach, które zwiększa niezawodność dostarczania danych może spowodować, że po dostarczeniu zbyt dużej ilości informacji i przepełnieniu bufora następne przychodzące dane będą odrzucane;
- łączy — zbyt częste retransmisje zmniejszają przepustowość sieci;
- Zasobów energetycznych — transmisja na zbyt duże odległości przez systemy radiowe może wyczerpać baterie węzła i uczynić go na pewien okres niedostępnym.

2.5.2.4 Metryka

Protokół rutingu powinien wykorzystywać metrykę w celu estymacji kosztów transmisji danych określoną ścieżką oraz w celu porównania kosztów transportu danych różnymi ścieżkami. Metryka powinna zostać zdefiniowana w taki sposób, żeby była użyteczna dla innych protokołów rutingu. Powinien także wykorzystywać algorytm *SPF* (ang. *Shortest Path First* — algorytm najkrótszej ścieżki) lub jego odmianę w celu przesyłania danych najkrótszą droga. **[3]**

W przypadku sieci *DTN* wartością oddającą koszt przesłania wiadomości jest czas potrzebny na dostarczenie wiadomości do następnego węzła. Im krótszy czas dostarczenia wiadomości, tym mniej zasobów jest wykorzystywanych (np. bufory). Zwiększa się także niezawodność dostarczania danych, ponieważ jest więcej zasobów dostępnych dla innych wiadomości. Jednak zdefiniowano także bardziej szczegółowe i precyzyjne metryki: **[3]**

2.5.2.4.1 MED

MED (ang. *Minimum Expected Delay* — minimalne oczekiwane opóźnienie) — najprostsza metryka, która zakłada, że czas kolejkowania wynosi 0, zaś średnia sumy czasu transmisji, propagacji oraz czekania jest zawsze precyzyjnie znana. Jest to po prostu oczekiwany czas przesłania wiadomości z jednego węzła do drugiego. **[3]**

2.5.2.4.2 ED

ED (ang. *Earliest Delivery* — najwcześniejsze dostarczenie) — zakłada, że czas kolejkowania wynosi 0, czas propagacji i czas transmisji są znane precyzyjnie. W przeciwieństwie do *MED* metryka ta wymaga znajomości czasów kontaktów — brane są pod uwagę tylko te ścieżki, na których w przyszłości wystąpi taki układ kontaktów, że będzie możliwe dostarczenie wiadomości, pozostałe ścieżki są pomijane. **[3]**

2.5.2.4.3 EDLQ

EDLQ (ang. *Earliest Delivery with Local Queuing* — najwcześniejsze dostarczenie z lokalnym kolejkowaniem) – do czasu *ED* dodawana jest także estymata czasu kolejkowania w węźle lokalnym. **[3]**

2.5.2.4.4 EDAQ

EDAQ (ang. *Earliest Delivery with All Queues* — najwcześniejsze dostarczenie z wszystkimi kolejkami) — do czasu *ED* dodawany jest czas oczekiwania przez wiadomość w kolejkach we wszystkich węzłach na badanej drodze. Wartość ta obliczana jest na podstawie aktualnych informacji dostarczanych przez poszczególne węzły. **[3]**

2.5.3 Porównywanie protokołów rutingu

Aby móc porównywać różne protokoły rutingu, musimy zdefiniować, jakie rezultaty działania uznajemy za szczególnie pożądane, a jakie za szczególnie szkodliwe. Da to nam możliwość oceny różnych strategii rutingu (opisanych dalej) i ustalenia drogą symulacji, który protokół w danym środowisku okazuje się najkorzystniejszy do określonych zastosowań.

Trzy czynniki decydujące o sprawności danego protokołu rutingu, to:

2.5.3.1 Skuteczność dostarczania danych

Skuteczność dostarczania danych można zdefiniować jako odsetek informacji, które zostały wysłane i dostarczone do celu. W przypadku nieefektywnie działającego protokołu rutingu możliwe jest, że część danych będzie tracona, mimo iż w sieci będzie istnieć fizyczna możliwość przetransportowania tych informacji. Ze względu na charakter niniejszej pracy, ten czynnik jest kluczowy, natomiast nie można przyjąć, że jest jedyny, jaki ma znaczenie, ponieważ w tej sytuacji z góry byłoby wiadomo, że najlepszymi protokołami rutingu byłyby oparte na podejściu epidemicznym (opisanym w dalszej części pracy), które dążą do tworzenia dużej liczby kopii każdej paczki danych. W praktyce wielokrotne kopiowanie tej samej wiadomości zupełnie nie pokrywa się z możliwościami istniejących i planowanych sieci telekomunikacyjnych. **[3]**

2.5.3.2 Opóźnienie

Opóźnienie to czas po jakim dane docierają do celu od momentu ich wysłania. Jest to czas średni i może być zależny nie tylko od czasów propagacji, transportu i kolejkowania, jak to ma typowo miejsce w sieci internetowej, ale także od czasu oczekiwania na wystąpienie kontaktu oraz ewentualnych retransmisji (znacznie częstszych w sieci *DTN* w stosunku do *Internetu*). **[3]**

2.5.3.3 Informacje nadmiarowe

Niektóre protokoły rutingu wysyłają wielokrotnie różne ramki, a także kierują je przez dłuższe łącza. W efekcie zmienia się liczba dokonywanych między węzłami transmisji (jedna transmisja w tym przypadku jest rozumiana jako wysłanie pojedynczej paczki informacji między dwoma

najbliższymi węzłami). Takie transmisje nadmiarowo zużywają zasoby i dlatego są niepożądane. Do tego dochodzi sam narzut związany z transmisją danych protokołu rutingu. **[3]**

2.6 Podejścia do przekierowywania wiadomości

Najważniejszym czynnikiem decydującym o skuteczności danego protokołu rutingu jest niewątpliwie metoda przekierowywania paczek danych, które docierają do węzła korzystającego z danego protokołu rutingu. Ten właśnie czynnik jest nie tylko najistotniejszy w sieci *DTN*, ale także najtrudniejszy do dopasowania do poszczególnych środowisk ze względu na ogromną różnorodność typów sieci i zależności czasowych występujących w różnych sieciach. Istnieje jednak kilka typowych podejść do rutingu, które sprawdzają się różnie w różnych sieciach i dlatego z założenia będą implementowane w różnych protokołach rutingu przeznaczonych do zupełnie różnych środowisk. Możliwe jest nawet, że dwa protokoły rutingu korzystające z tego samego podejścia będą drastycznie różnić się sposobem działania.

2.6.1 Podejście „wyroczni"

W pewnych sytuacjach struktura sieci jest bardzo dobrze znana (np. misje kosmiczne). Stan całej sieci można traktować jako deterministyczny — z góry wiemy kiedy oraz z kim będzie łączność. Protokoły rutingu oparte na tym podejściu przewidują, kiedy nastąpi kontakt pomiędzy dwoma najbliższymi węzłami i dlatego przekazując paczki danych z wyprzedzeniem kierują je tam, gdzie za jakiś czas nastąpi połączenie. Nazwa „podejście wyroczni" wzięła się stąd, że węzeł podejmujący decyzję o transmisji paczki danych przewiduje stan sieci w przyszłości. **[1]**

2.6.2 Podejście oparte na modelu

W niektórych sieciach występują specyficzne sytuacje, które pozwalają przewidywać opłacalność transmisji danych przez określone łącze. Np. tam, gdzie węzły umieszczone są na pojazdach poruszających się w terenie, zazwyczaj przerwy w łączności są krótkotrwałe, dlatego znacznie efektywniejszy jest ruting optymistyczny. Podejście oparte na modelu zakłada, że jesteśmy w stanie dla danej sieci stworzyć model występujących w niej połączeń. Np. w ww. przypadku absolutna większość zerwań łączności to zerwania bardzo krótkotrwałe, dlatego w tym akurat typie sieci protokół rutingu powinien koncentrować się na przechowywaniu danych do momentu ponownego nawiązania połączenia, a nie przekierowywać dane do innych węzłów, licząc, że one jakoś odnajdą drogę. Ponieważ model tworzony jest na potrzeby jednej sieci, więc tak tworzone protokoły rutingu nie mogą być uniwersalne i nie mają zastosowania nigdzie indziej poza daną siecią. **[1]**

2.6.3 Podejście epidemiczne

Podobnie jak wirusy, które rozprzestrzeniają się we wszystkich kierunkach kopiując się wielokrotnie, aż zainfekują wszystkie organizmy, paczki danych mogą być wysyłane do wszystkich znanych węzłów (pod warunkiem, że dany węzeł nie ma jeszcze tych danych). W przypadku niepowodzenia transmisji lub niedostępności łącza dane mogą być przechowywane. Gdy dostępne staną się połączenia z węzłami, do których paczka jeszcze nie została wysłana, natychmiast jest to robione. Dodatkowym wymaganiem stawianym przez podejście epidemiczne jest konieczność rozróżniania danych, a więc zapewnienia unikalnego identyfikatora dla każdej paczki, co umożliwia sprawdzenie, czy dana paczka już znajduje się w

tym węźle. To podejście jest bardzo atrakcyjne tylko wtedy, kiedy połączenia między poszczególnymi węzłami tworzą się i znikają w sposób losowy, a ilość przesyłanych danych jest bardzo niewielka. Przykładem zastosowania jest projekt *Zebranet*, w sieci którego połączenia między węzłami (zebrami oraz badaczami) występują zupełnie przypadkowo i na różny okres czasu, a przesyłane dane (współrzędne pobytu zebr) zajmują bardzo mało miejsca. Podejście epidemiczne jest o tyle dobre, iż gwarantuje, że jeśli tylko powstanie kiedykolwiek w sieci odpowiednia kombinacja połączeń umożliwiająca przesłanie danych, to dane te zostaną przesłane. Niezawodność ta jest jednak osiągana ogromnym kosztem związanym przede wszystkim z magazynowaniem kopii danych na każdym istniejącym węźle i stąd zastosowanie w praktyce tego protokołu jest mocno ograniczone. **[1]**

2.6.4 Podejście szacunkowe

To podejście zakłada, że jesteśmy w stanie w jakiś sposób oszacować prawdopodobieństwo dostarczenia danych przez określoną drogę do celu. Np. w sieciach z autobusami miejskimi używanymi jako muły danych dostarczenie danych do centrum miasta jest bardziej prawdopodobne, jeśli przekażemy dane do autobusu zmierzającego w tym kierunku. Inny sposób szacowania prawdopodobieństwa dostarczenia danych to wielokrotne wysyłanie informacji różnymi drogami i zbieranie statystyk ich dostarczenia. W ten sposób budowana jest tablica efektywności dostarczania danych do danego węzła przy użyciu różnych dróg. **[1]**

Mając oszacowane prawdopodobieństwa dostarczenia danych przez dostępne łącza, można wysłać tę samą paczkę danych kilkakrotnie przez najlepsze z dostępnych ścieżek. Podejście to pozwala znacznie ograniczyć ilość przesyłanych informacji w stosunku do podejścia epidemicznego, a jednocześnie nadal zapewnia bardzo duże prawdopodobieństwo udanego dostarczenia danych. **[1]**

2.6.5 Podejście oparte na wymazywaniu (ang. erasure)

Podejście oparte na wymazywaniu wykorzystuje w ciekawy sposób kodowanie i dzielenie danych na części. Zgodnie z tą koncepcją dane są kodowane i dzielone na części, a następnie kilkoma drogami wysyłane do celu. Do odebrania i zdekodowania danych nie potrzebne jest odebranie wszystkich części, wystarczy pewna ich liczba (np. 5 z 8). To podejście wymaga bardzo dużej mocy obliczeniowej węzła wysyłającego i odbierającego dane, poza tym atrakcyjne jest tylko wtedy, kiedy istnieje co najmniej kilka dróg dostania się do celu, przy czym drogi te są potencjalnie niepewne. Jakkolwiek podejście to wydaje się dosyć dziwne, udowodniono, że znacznie ogranicza ono opóźnienie w najgorszym przypadku. **[1]**

Podejście to może być wykorzystane także w tzw. sieciach *peer-to-peer* do wymiany plików. Są one oparte na protokole *IP*. W takich sieciach bardzo często zdarza się, że jedna z wielu części danych jest długo niedostępna, ponieważ osoba ją udostępniająca nie podłącza się do sieci. W nowym podejściu dane byłyby przechowywane są w takiej formie, że użytkownik nie musiałby ściągać żadnej części konkretnie. Wystarczyłoby zebrać kilka spośród krążących w sieci fragmentów danych, aby odtworzyć oryginalny plik.

2.6.6 Podejście oparte na kontroli ruchu

Podejście oparte na kontroli ruchu zakłada, że węzły mają wpływ na zmianę topologii sieci. Wyobraźmy sobie sieć, w której istnieje pewna liczba mułów danych — są to węzły poruszające

się pomiędzy istniejącymi węzłami statycznymi. W momencie gdy węzeł statyczny potrzebuje wysłać dane, wysyła silny (w celu objęcia dużego obszaru), ale jednocześnie krótki (w celu oszczędności energii) sygnał wywoławczy, po którym zbliża się do niego jeden z dostępnych mułów danych. Po znalezieniu się w zasięgu nadajnika dającego dużą przepływność danych, informacje są transmitowane, a następnie muł danych wyrusza w kierunku węzła docelowego. Podejście to wydaje się bardzo prymitywne, jednak udowodniono, że jest wiele sytuacji, w których się sprawdza. **[1]**

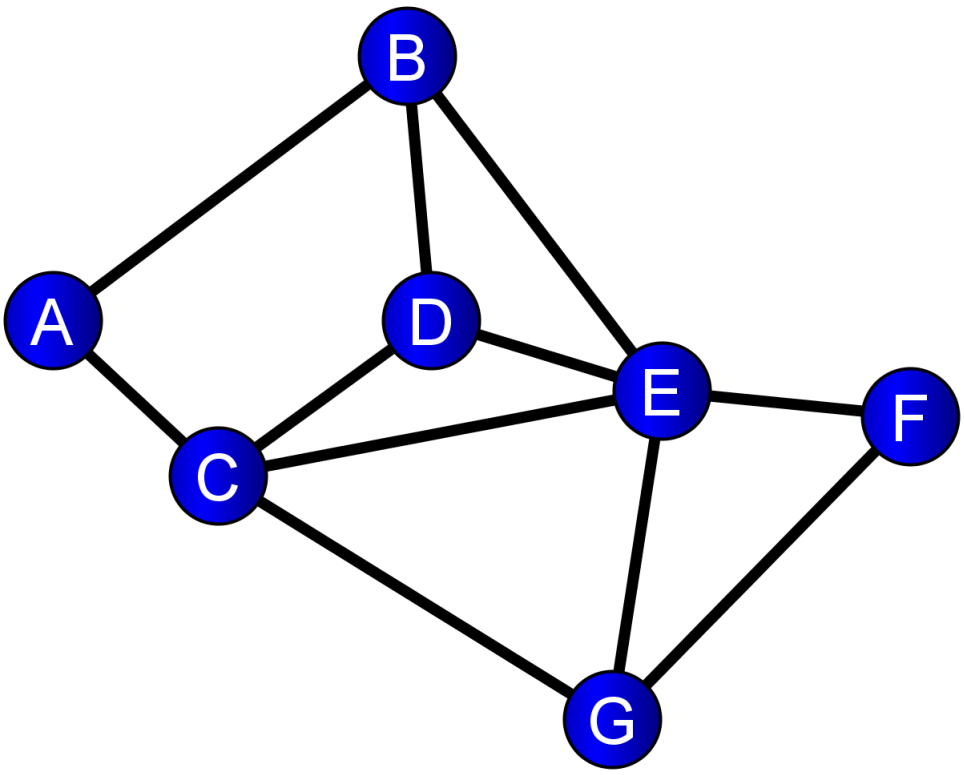
2.7 Podział protokołów rutingu

Wprowadźmy teraz dodatkowe rozróżnienia związane z bardziej typowymi protokołami rutingu; istnieje kilka strategii, które mogą być wykorzystane w celu poprawy właściwości danego protokołu rutingu. Aby lepiej zrozumieć ich znaczenie, użyjmy przykładowej sieci *DTN* (rysunek 2-5).

Dla uproszenia, w rozpatrywanym przypadku wszystkie łącza są dwustronne, zaś kontakt pomiędzy dwoma najbliższymi węzłami ma charakter probabilistyczny, tzn. łącza stają się dostępne i niedostępne w losowych odstępach czasu.

2.7.1 Replikacja

W przypadku strumienia replikacji wiadomość może być powielana i wiele jej kopii może być wysyłanych przez sieć. Jedną ze skrajności są protokoły rutingu oparte na podejściu epidemicznym, które kopiują wiadomość przy każdej nadarzającej się okazji. Zaletą tego podejścia jest największe prawdopodobieństwo dostarczenia informacji oraz najmniejsze



Rysunek 2-5: Przykładowa struktura sieci DTN.

opóźnienie. Natomiast wadą zdecydowana nadmiarowość wykorzystania zasobów. Drugą skrajnością jest wykorzystanie tylko jednej kopii. Jest to jednak niebezpieczne, ponieważ w razie uszkodzenia któregoś z elementów sieci wiadomość jest tracona. Natomiast to podejście cechuje się minimalnym wykorzystaniem istniejących zasobów. Większość protokołów rutingu w *DTN* znajduje się gdzieś pośrodku — tworzonych jest kilka kopii wiadomości. Najbardziej powszechnym scenariuszem jest wysłanie jednej kopii przez sieć i pozostawienie drugiej w celu ewentualnej retransmisji. Możliwe jest także wybranie kilku spośród dostępnych dróg przez które wysyłane są kopie wiadomości. **[3]**

2.7.2 Propagacja wiedzy o sieci

Protokół rutingu może wymieniać informacje nt. struktury i cech sieci. Pierwszą skrajnością jest sytuacja, gdy protokół rutingu nie wymienia żadnych informacji. W momencie rutowania wiadomości znany jest tylko stan bezpośrednio przylegających łączy. Drugą skrajnością jest wiedza każdego węzła o wszystkich występujących w sieci awariach (znajomość aktualnej struktury sieci). Pierwszy przypadek jest przede wszystkim łatwy do implementacji, natomiast stosunkowo nieefektywny szczególnie dla dużych sieci. W miarę przechodzenia do drugiej skrajności zwiększa się złożoność implementacji oraz wykorzystanie łącza przez protokoły rutingu. Przesyłanie dużej ilości informacji o sieci nie zawsze też jest tak efektywne, jak mogłoby się wydawać, ponieważ ze względu na duże opóźnienia istniejące w sieciach *DTN*, duża liczba przybywających informacji jest nieaktualna. Gdzieś pośrodku plasuje się np. transmisja informacji o prawdopodobieństwie istnienia połączenia. **[3]**

2.7.3 Strategie rozprzestrzeniania informacji

Ta grupa strategii związana jest z momentem, kiedy węzeł decyduje się przekazać dane dalej.

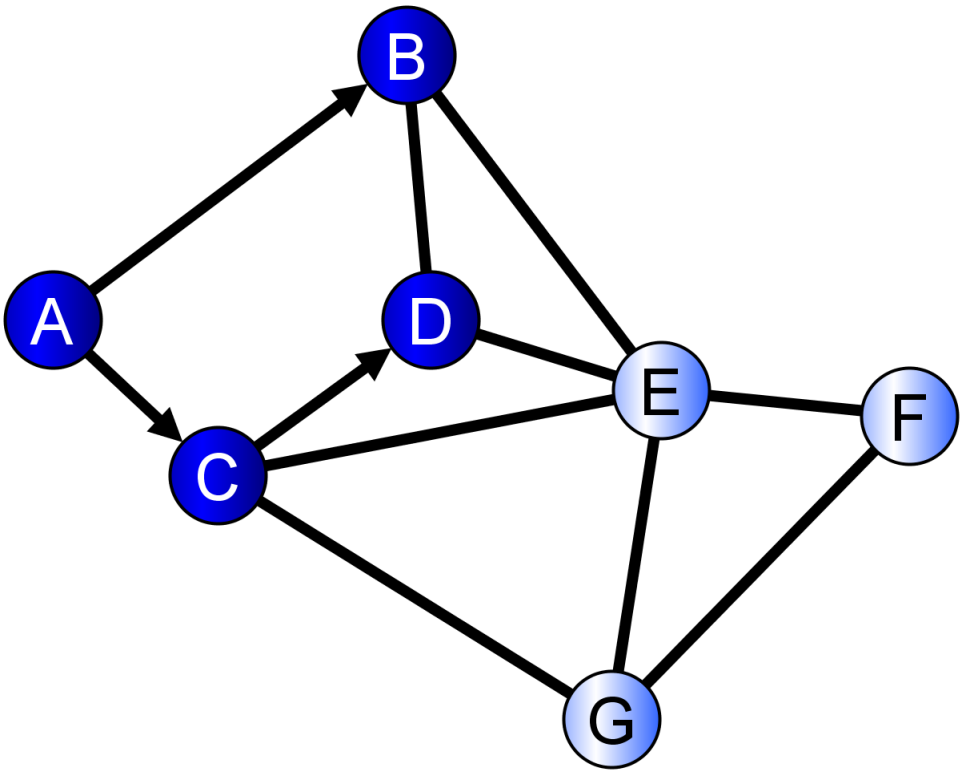
2.7.3.1 Kontakt bezpośredni

Strategia przekazywania danych podczas bezpośredniego kontaktu polega na czekaniu, aż między dwoma węzłami będzie występować bezpośrednie połączenie (kontakt). Dopiero wtedy następuje transmisja danych. Nie ma prób transportu wiadomości do węzłów pośrednich w nadziei, że połączenie między węzłem pośrednim oraz docelowym pojawi się wcześniej, niż połączenie bezpośrednie. Rysunek 2-6 graficznie przedstawia sposób działania tego algorytmu. **[3]**

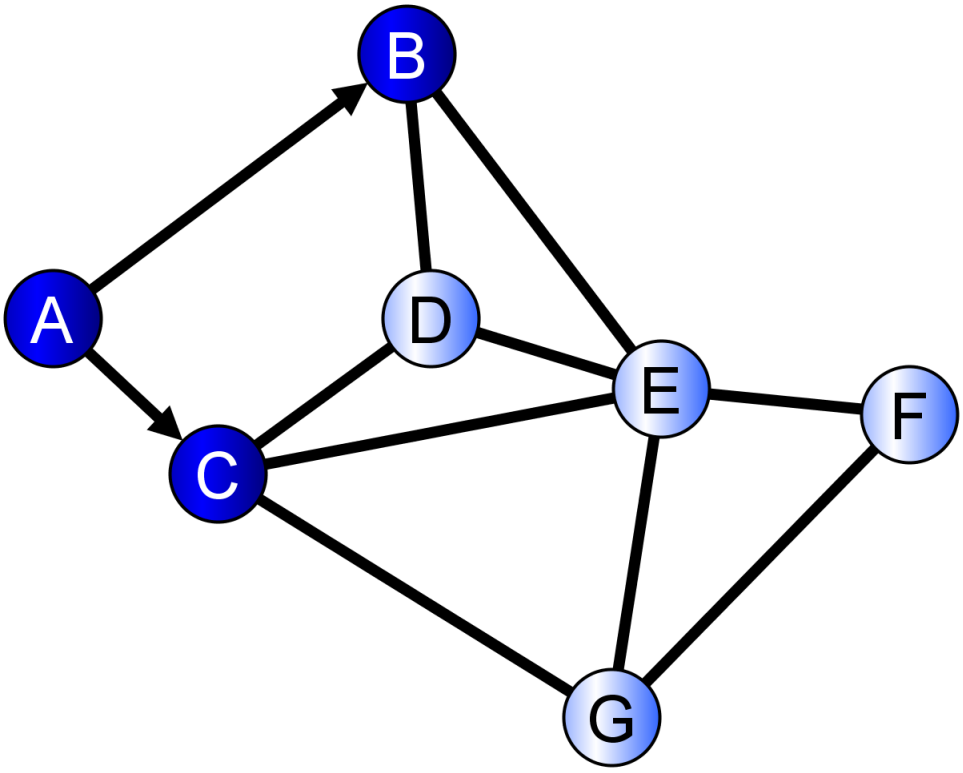
2.7.3.2 Kopiowanie do określonej liczby sąsiednich węzłów

Strategia kopiowania do określonej liczby polega na wysyłaniu kopii wiadomości do *n* najbliższych węzłów, z którymi węzeł początkowy będzie mieć kontakt, przy czym n to parametr strategii. Następnie każdy z tych *n* + 1 węzłów (początkowy oraz *n* mających już wiadomość) będzie wysyłać wiadomość do celu tylko w momencie uzyskania bezpośredniego kontaktu. Ta strategia znacznie zwiększa szybkość dostarczania wiadomości do celu. Jeżeli *p* to prawdopodobieństwo wystąpienia kontaktu między dwoma węzłami, to przy dużej liczbie węzłów i małym *p* prawdopodobieństwo wystąpienia połączenia między jednym z węzłów mających wiadomość a węzłem docelowym, wynosi:

$$p' = (n+1)p$$



Rysunek 2-6: Kopiowanie tylko w przypadku bezpośredniego kontaktu. Wysyłającym węzłem jest A. Tylko węzły B, C i D są dostępne, dlatego tylko tam może nastąpić wysłanie wiadomości. Na podstawie [3].



Rysunek 2-7: Kopiowanie wiadomości do najbliższych węzłów w celu zwiększenia prawdopodobieństwa kontaktu. A jest węzłem wysyłającym wiadomość. Na podstawie [3].

co automatycznie zmniejsza $n + 1$ razy średni czas czekania na wystąpienie kontaktu. Sposób

działania tego algorytmu przedstawia graficznie rysunek 2-7. **[3]**

2.7.3.3 Rozprzestrzenianie oparte na strukturze drzewa

Rozprzestrzenianie oparte na strukturze drzewa jest podobne do kopiowania do *n* sąsiednich węzłów, przy czym strategia ta daje możliwość dalszego kopiowania informacji do następnych węzłów. Wykorzystywane mogą być różne ograniczenia nałożone na transmisję — m.in. ograniczenie liczby przeskoków od źródła do *k* węzłów lub ograniczenie łącznej liczby kopii do *l*, gdzie *k* i *l* to parametry konfiguracyjne tej strategii. Graficzną prezentację sposobu działania tego algorytmu przedstawia rysunek 2-8. **[3]**

2.7.3.4 Podejście epidemiczne

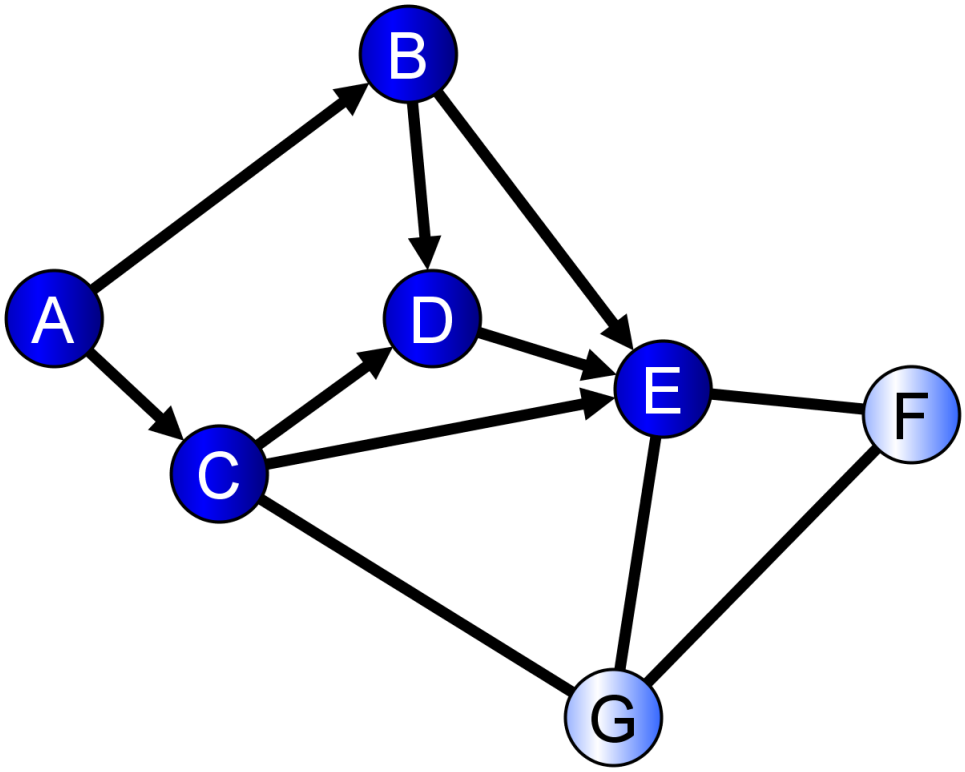
Podejście epidemiczne polega na systematycznym kopiowaniu wiadomości we wszystkich kierunkach, do wszystkich węzłów, z którymi występuje kontakt. Zasadę działania podejścia epidemicznego przedstawia rysunek 2-9.**[3]**

2.7.4 Strategie przekierowywania

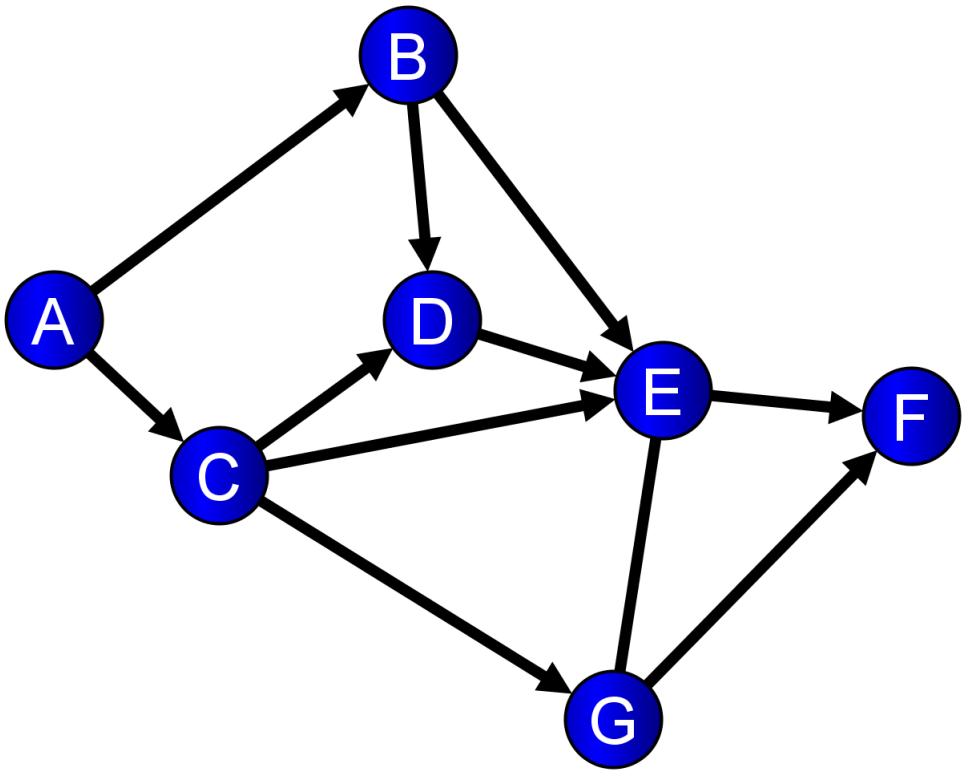
Strategie przekierowywania decydują, w jaki sposób węzeł podejmuje decyzje o wyborze ścieżki do kolejnych węzłów na drodze do celu.

2.7.4.1 Ruting oparty na lokalizacji

W przypadku rutingu opartego na lokalizacji paczki danych przekierowywane są w oparciu o położenie geograficzne węzłów. Z założenia paczka jest kierowana do tego węzła, którego



Rysunek 2-8: Replikacja zgodnie ze strukturą drzewa. A jest węzłem wysyłającym wiadomość. W celu zwiększenia prawdopodobieństwa dostarczenia wiadomości, kopiuje dane do kilku węzłów, z którymi jest w kontakcie. Węzły wybierane są zgodnie z algorytmem rozprzestrzeniania opartego na strukturze drzewa, co oznacza, że dane kopiowane są nie tylko do bezpośrednio połączonych węzłów (B, C), ale także do węzłów położonych w sieci w większej odległości (w sensie liczby przeskoków). Na podstawie [3].



Rysunek 2-9: Rozprzestrzenianie epidemiczne wiadomości. Węzeł A wysyła wiadomość. W tym celu transmituje ją do bezpośrednio połączonych węzłów (B, C). Węzły przechowują kopię wiadomości, a jednocześnie transmitują ją do węzłów, z którymi same są połączone (D, E, G) z wyłączeniem węzła, od którego otrzymały tę wiadomość (A). W ten sposób wiadomość rozprzestrzenia się po całej sieci, aż w końcu trafi do adresata. Na podstawie [3].

koordynaty są najbliższe koordynatom węzła docelowego (spośród tych węzłów, z którymi istnieje w danym momencie kontakt). Wymaga to więc bardzo mało informacji o strukturze sieci (położenie własne, węzła docelowego oraz węzłów, z którymi węzeł przekierowujący jest w kontakcie), co można prosto osiągnąć np. przy wykorzystaniu *GPS*. **[3]**

Niestety nie ma gwarancji, że węzły, które są blisko siebie w sensie geograficznym będą mogły się komunikować np. z powodu przeszkody na linii połączenia radiowego. Drugim problemem jest potencjalna zmienność koordynat w sytuacji, gdy węzły są mobilne. Kiedy więc wiadomość dotrze już do miejsca docelowego, może się okazać, że węzeł przeniósł się już do zupełnie innego miejsca w sieci, dlatego węzły przemieszczające się muszą informować o zmianach swojego położenia całą sieć. Często jest to nieefektywne.

2.7.4.2 Przekierowywanie gradientowe

Przekierowywanie gradientowe opiera się na obliczaniu dla każdego przylegającego łącza gradientu, tj. stosunku różnicy odległości do węzła docelowego do metryki łącza. Im większy gradient, tym szybciej pakiet będzie zbliżać się do węzła docelowego będąc przesłanym danym łączem. Każdy węzeł zna topologię sieci, przy czym nie zna aktualnego stanu wszystkich łączy i węzłów. Na podstawie dostępnych informacji węzeł rutujący jest w stanie wyliczyć odległość każdego węzła w sieci od węzła docelowego i wybrać najlepsze spośród dostępnych łączy (tzn. takie, który ma najwyższy gradient). Przemieszczająca się paczka danych podąża zgodnie ze ścieżką największych gradientów aż do węzła docelowego. **[3]**

2.7.4.3 Przekierowywanie oparte na metryce łączy

Przekierowywanie oparte na metryce łączy jest podobne do znanego z sieci *IP* protokołu *OSPF* — węzeł musi znać cała strukturę sieci włączając w to chwilowy stan łączy i węzłów, a następnie za sprawą algorytmu Dijkstry obliczana jest najkrótsza ścieżka do wybranego węzła. Najważniejszą składową metryki dla sieci *DTN* jest niezawodność dostarczania danych, poza nią znaczenie ma m.in. opóźnienie. Istnieje większa liczba metryk, które mogą być używane w zależności od typu sieci i specyficznych potrzeb konkretnych aplikacji wykorzystujących daną sieć (np. mniejsze opóźnienie kosztem niezawodności). **[3]**

2.7.5 Moment ustalania topologii sieci

Strategie ustalania topologii sieci definiują, kiedy zdobywana jest wiedza nt. struktury sieci.

2.7.5.1 Podejście oparte na tworzeniu tablicy rutingu

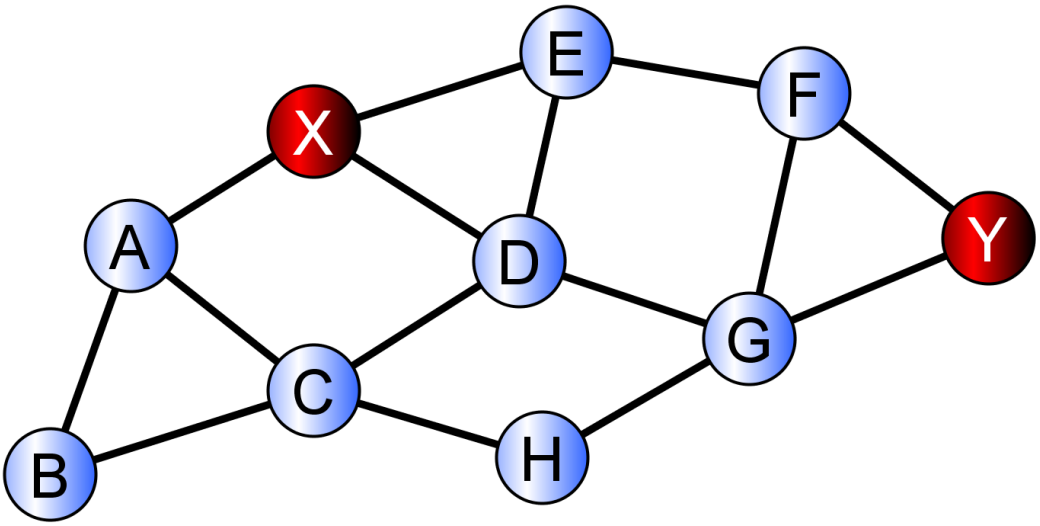
Protokoły rutingu znane z sieci *IP* tworzą tablicę rutingu, którą modyfikują w momencie otrzymania nowych informacji o stanie sieci. To rozwiązanie sprawdza się w sieci *IP*, gdzie liczba przesyłanych danych użytecznych jest bardzo duża i dlatego też narzut informacji pochodzących od protokołów rutingu jest akceptowalny. W strategii tej trasa do celu znana jest cały czas, zaś tablica rutingu jest zawsze aktualna w takim stopniu, jak to tylko możliwe. W momencie przybycia paczki danych do węzła, natychmiast może on wysłać ją przez dostępne łącze.

2.7.5.2 Wykrywanie struktury sieci w momencie transmisji danych

Istnieją środowiska, gdzie istnieje duża liczba węzłów, węzły często ulegają zmianom stanu takim jak wyłączenie lub uszkodzenie, wymieniają się stosunkowo małą ilością informacji, zaś opóźnienia występujące między węzłami są niewielkie. Przykładem może być sieć sensorowa. W takich środowiskach doskonale sprawdzają się protokoły wykrywające strukturę sieci dopiero w momencie transmisji danych. Najpopularniejszym protokołem tego typu jest *AODV* (ang. *Ad hoc On-Demand Distance Vector* — wektor dystansu na żądanie w sieciach typu ad hoc). Jego główną koncepcją jest wyszukiwanie połączenia przez węzeł będący źródłem danych. Trasa jest wyszukiwana i zestawiana dopiero w momencie, kiedy węzeł chce wysłać dane. Utrzymywana jest tylko przez pewien krótki czas. **[3,6,12,13]**

Całość procesu wyszukiwania ścieżki przebiega w kilku fazach. Na samym początku węzeł źródłowy wysyła do wszystkich węzłów w otoczeniu (ang. *broadcast*) pakiet *RREQ* (ang. *Route Request* — żądanie trasy) informujący o chęci wysłania danych do węzła docelowego. Węzły, które odebrały *RREQ* odpowiadają informując tym samym, że istnieje połączenie prowadzące w tym kierunku i przesyłają wiadomość dalej. Gdy węzeł docelowy odbierze wiadomość *RREQ* wysyła natychmiast odpowiedź *RREP* (ang. *Route Reply* — odpowiedź z trasą) — ponieważ wszystkie połączenia zostały przed chwilą wykryte, więc wiadomość ta dotrze bez większych problemów do źródła danych, które rozpocznie transmisję. **[3,6,12,13]**

Na rysunku poniżej (rysunek 2-10) węzeł X chce wysłać dane do węzła Y.

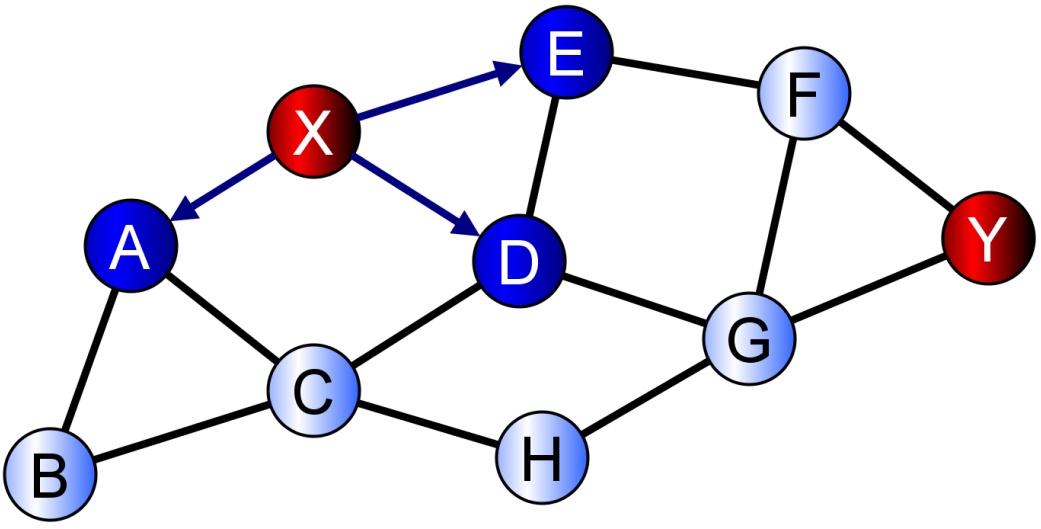


Rysunek 2-10: Faza 1 AODV: węzeł X decyduje się wysłać informacje do węzła Y. Na podstawie [6].

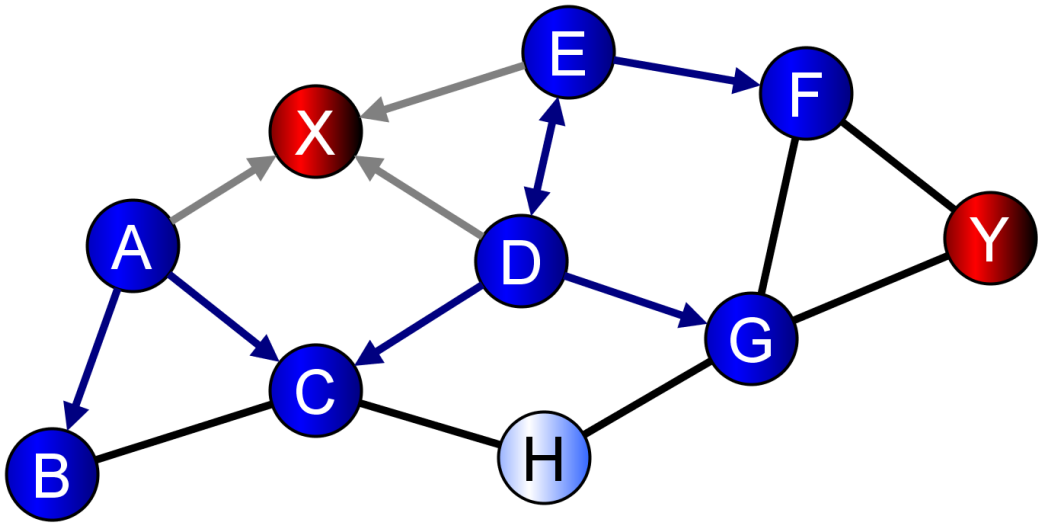
Węzeł X wysyła *RREQ* do najbliższych węzłów. Poniższy rysunek (rysunek 2-11) prezentuje drugą fazę działania protokołu AODV. Niebieskie strzałki oznaczają wysyłanie pakietu *RREQ*.

Węzły potwierdzają otrzymanie *RREQ* i przesyłają pakiet dalej. Ponieważ *AODV* jest wykorzystywany w sensorowych sieciach radiowych, które nie prowadzą ukierunkowanej transmisji, dlatego też pakiet jest wysyłany także wstecznie do węzła, skąd przybył pakiet *RREQ*.

Otrzymanie pakietu *RREQ* z jakiegoś węzła oznacza możliwość transmisji wiadomości do tego węzła (łącza są zawsze dwukierunkowe) oraz — za jego pośrednictwem — do węzła X. W ten sposób węzły ustanawiają połączenie z węzłem X. Mechanizm ten prezentuje graficznie rysunek 2-12. Szare strzałki oznaczają wiedzę węzłów o możliwości połączenia z



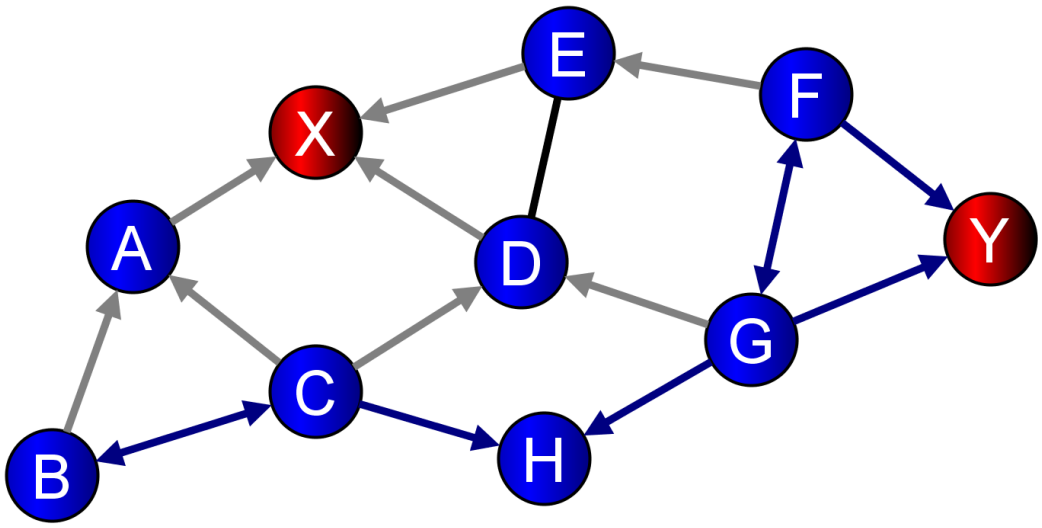
Rysunek 2-11: Faza 2 AODV: węzeł X wysyła RREQ do najbliższych węzłów. Na podstawie [6].



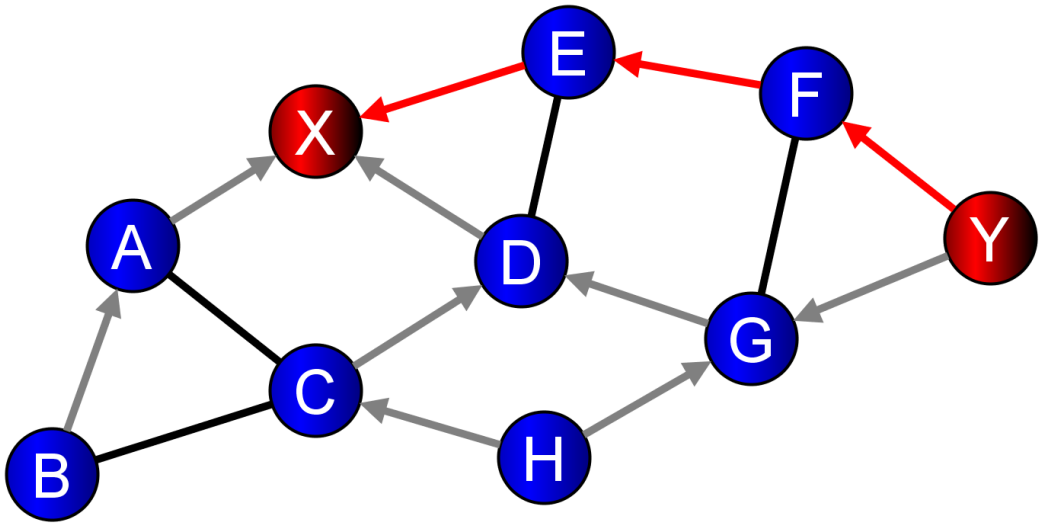
Rysunek 2-12: Faza 3 AODV: węzły wysyłają potwierdzenia oraz przesyłają RREQ dalej. Na podstawie [6].

węzłem X.

Pakiet *RREQ* dociera w końcu do węzła Y, z którym węzeł X chciał ustanowić połączenie. Tę sytuację prezentuje rysunek 2-13. Należy zwrócić uwagę, że węzeł C dostał dwie wiadomości *RREQ* od węzłów A oraz D, ale druga z kolei wiadomość jest ignorowana. Podobnie ignorowane będą wiadomości z węzłów B oraz H, które będą otrzymane za jakiś czas. Węzeł C pamięta, że dostał już pakiet *RREQ* od węzła X, dlatego nie wysyła wiadomości powtórnie powstrzymując w ten sposób zalew wiadomości. Zazwyczaj wybór drogi spośród kilku źródeł, z których przybywa ten sam pakiet *RREQ*, podyktowany jest kolejnością przychodzenia wiadomości — szybsze otrzymanie wiadomości oznacza krótszą drogę do węzła X. W ten sposób każdy z węzłów ustanawia kierunek przełączania przychodzących do niego danych zmierzających do węzła X (wpis w tabeli rutingu) przez



Rysunek 2-13: Faza 4 AODV: RREQ dociera do węzła docelowego Y. Na podstawie [6].



Rysunek 2-14: Faza 5 AODV — węzeł Y wysyła pakiet RREP do węzła X korzystając z utworzonej na węzłach tymczasowej tablicy rutingu. Na podstawie [6].

jedno z dostępnych łączy.

Rysunek 2-14 przedstawia ostatnią fazę działania protokołu *AODV*. Ostatecznie węzeł Y odpowiada pakietem *RREP* — ponieważ węzły E i F znają już topologię sieci i wiedzą, w jaki sposób można dotrzeć do węzła X, więc bez problemu przekazują pakiet *RREP* do celu. Pakiet ten jednocześnie poinformuje węzły E i F o drodze do węzła Y. Następnie rozpoczyna się normalna transmisja między węzłami X i Y przez węzły E i F.

3 Tworzenie aplikacji symulującej działanie sieci DTN

3.1 Model sieci DTN

Aby móc napisać program symulujący działanie sieci *DTN*, potrzebujemy najpierw stworzyć model sieci *DTN*. Model jest znacznym uproszczeniem rzeczywistego sposobu funkcjonowania elementów sieci, co sprawia, że uzyskiwane przez symulację opartą na modelu wyniki mogą nie pokrywać się z rzeczywistymi wynikami osiąganymi podczas testów polowych. Stworzenie oraz wykorzystywanie modelu do symulacji jest jednak konieczne ze względu na bardzo dużą komplikację istniejących protokołów i urządzeń. Symulowanie działania protokołów zgodnie z ich definicją doprowadziłoby do drastycznego zwiększenia czasu potrzebnego na implementację symulatora, a także do zwiększenia czasu symulacji powyżej akceptowalnych wartości. Symulacja złożonych stosów protokołów w większych sieciach za pomocą dostępnych obecnie narzędzi jest często wolniejsza od rzeczywiście działających urządzeń, co wynika najczęściej z faktu równoległego przetwarzania informacji w sieci przez wiele urządzeń (i wiele warstw w pojedynczym urządzeniu, np. karta sieciowa pracuje równolegle z procesorem) oraz symulacji działającej na jednym procesorze i współdziałającej z innymi operacjami wykonywanymi przez system operacyjny. Dobry model sieci *DTN* podany został w **[3]**, jednak jest on zbyt ogólny jak na potrzeby symulacji. Poniższa wersja jest bardzo szerokim rozwinięciem, dopasowaniem do potrzeb symulacji oraz — w niewielkim zakresie — modyfikacją wspomnianego modelu.

3.1.1 Elementy sieci

Niniejszy podrozdział identyfikuje występujące w sieci elementy fizyczne, które tworzą strukturę sieci *DTN*.

3.1.1.1 Węzły

Głównymi elementami sieci są węzły. Są to elementy adresowalne, tzn. mają przypisany adres umożliwiający innym węzłom wysyłanie do nich danych. Węzły połączone są pewną liczbą łączy. Ze względu na symulowanie niezawodności elementów sieci, węzły są elementami ulegającymi uszkodzeniom oraz wyłączającymi się zgodnie z mechanizmem opisanym w dalszej części tej pracy. Węzły zapewniają buforowanie dla transportowanych wiadomości. Węzeł wysyła wiadomości potwierdzające otrzymanie określonej porcji informacji, może także przejmować na siebie odpowiedzialność za ewentualną późniejszą retransmisję pakietów (patrz opis protokołu *Bundle*). Przejmowanie tej odpowiedzialności jest cechą niektórych węzłów i jest ustalanie w konfiguracji. Każdy węzeł obsługuje jeden protokół rutingu.

3.1.1.2 Łącza

Pojęcie łącza w sieci *DTN* zmienia swoje znaczenie w stosunku do tego, co znane jest z sieci Internet. Przez łącze będzie rozumiane połączenie pomiędzy dwoma węzłami warstwy sieci globalnej, przy czym w praktyce połączenie to może przechodzić np. poprzez wiele ruterów pracujących w sieci *IP*. Wszystkie łącza w sieci są dwukierunkowe. Każde łącze umożliwia transmisję tylko między dwoma węzłami (nie istnieje rozgłaszanie wiadomości znane z sieci radiowych). Łącza mogą działać lub nie działać — uszkadzają się i wyłączają na podobnych zasadach jak węzły. Mechanizm opisany jest w dalszej części tej pracy. Jeżeli łącze jest dostępne, to węzeł z jednej strony łącza ma możliwość wysyłania danych do węzła po drugiej stronie. Jeden węzeł może mieć wiele łączy z innym węzłem, np. może istnieć łącze o niskim koszcie przesyłania danych, ale zawodne, a jako zabezpieczenie niezawodne łącze o dużym koszcie transferu informacji. Łącza są elementami parametryzowanymi. Ich cechami są: przepływność, opóźnienie oraz obsługiwany protokół transportowy. Zakładam, że na każdym łączu działa taki protokół i węzły nie muszą już znać sposobu, w jaki zostały do nich przetransportowane paczki danych. Protokół transportowy określa m.in. narzut informacji, a także efektywność przesyłania danych w zależności od liczby błędów oraz opóźnienia łącza.

3.1.1.3 Źródła i cele informacji

Wiadomości generowane w sieci są oczywiście losowe. Źródłem wiadomości oraz jej celem mogą być dowolne dwa węzły istniejące w sieci. Pod uwagę brane będzie jedynie wysyłanie wiadomości pomiędzy dwoma węzłami. Wysyłanie tej samej wiadomości do wielu celów (ang. *multicasting*) czy też do wszystkich węzłów (ang. *broadcasting*) nie będzie brane pod uwagę.

Tworzenie nowej porcji danych odbywa się w źródłach wiadomości przypisanych bezpośrednio do węzłów. Źródło wiadomości co pewien czas (zgodnie z zadanym rozkładem prawdopodobieństwa) generuje pewną liczbę bajtów (także zgodnie z zadanym rozkładem prawdopodobieństwa), następnie wybiera adresata informacji. Ponieważ w sieci często zdarzają się sytuacje, kiedy węzły komunikują się niesymetrycznie, tzn. komunikacja pomiędzy parą węzłów może być dużo częstsza lub węzły te będą przesyłać znacznie więcej informacji między sobą w porównaniu z resztą sieci, dlatego źródło informacji zawiera konfigurowalną tabelę prawdopodobieństw wysłania wygenerowanych danych do określonych węzłów (jest to

tabela o elementach postaci [identyfikator węzła; prawdopodobieństwo wygenerowania danych do węzła]). Symulacja generowania wiadomości o zmiennej wielkości w zależności od węzła jest pominięta ze względu na nadmierną komplikację. W tym miejscu należy zwrócić uwagę, że to źródło wiadomości ustala, jaki będzie rozmiar paczek danych wysyłanych przez sieć — teoretycznie rozmiar ten może być różny, co może mieć wpływ na charakterystykę sieci, dlatego też parametr ten jest konfigurowalny.

Wszystkie informacje w sieci ostatecznie kierowane są do któregoś z węzłów. W węźle odbierającym dane można przeprowadzić analizę statystyczną, która jest głównym celem symulacji. W tym celu wyróżniony został kolejny element sieci — cel informacji, który jest komponentem węzła. Ze względu na temat tej pracy dyplomowej najważniejszym czynnikiem, jaki nas interesuje, jest niezawodność dostarczenia danych oraz opóźnienie. Dodatkowe statystyki tworzone są także w pozostałych elementach sieci, przez które przechodzą paczki danych. Wszystkie statystyki udostępniane są przez jednolity system pobierania statystyk opisany dalej.

3.1.2 Problematyka dostępności

W sieci *DTN* spotykamy się z różnymi sytuacjami związanymi z niedostępnością łączy czy węzłów (pozostałe elementy traktuje się jako stuprocentowo niezawodne lub po prostu jako części łączy i węzłów ulegające awariom wraz z nimi). Podczas awarii dane mogą zostać stracone, albo przechowane do momentu naprawienia. Poza awariami istnieje jeszcze możliwość wystąpienia przestoju spowodowanego najczęściej brakiem zasilania w przypadku węzłów (np. zasilanie może pochodzić z promieni słonecznych), zaś w przypadku łączy jest to najczęściej spowodowane przerwaniem połączenia radiowego przez przeszkodę. Przestoje mogą być zarówno przewidywalne, jak również całkowicie losowe. Osobnym problemem jest wykrywanie niedostępności łączy oraz propagacja informacji o znanych momentach wyłączeń w sieci.

3.1.2.1 Awarie i przestoje

Zakładam, że niedostępność elementów można podzielić na dwa rodzaje: awarie oraz przestoje. Awarie to wydarzenia całkowicie losowe mające przypisane rozkłady prawdopodobieństwa wystąpienia oraz późniejszej naprawy. Przestoje są wydarzeniami całkowicie przewidywalnymi. Dla celów symulacji momenty wystąpienia przestojów są losowane zgodnie z zadanym rozkładem prawdopodobieństwa przed symulacją, a następnie informacje te są udostępniane elementom sieci w czasie symulacji. Pozostałe możliwości zostają pominięte.

Bardzo ważnym rozróżnieniem związanym z awariami i przestojami jest utrata lub zachowanie danych zgromadzonych w buforach w momencie wystąpienia niedostępności elementu. Przestoje należy traktować w inny sposób niż uszkodzenia, ponieważ są to zdarzenia przewidywalne i prawidłowe z punktu widzenia działania sieci. W przypadku łączy zakładam, że dane, które są transportowane w momencie wystąpienia przestoju, dotrą do przeznaczenia mimo utraty kontaktu. W przypadku węzłów zakładam, że węzeł w stanie przestoju przechowuje zgromadzone dane.

Jeśli zaś chodzi o uszkodzenia, są to z założenia wydarzenie niepożądane, związaniem z fizycznym błędnym funkcjonowaniem elementu, dlatego zakładam, że wszelkie dane są tracone. Łącze traci wszelkie dane, które znajdowały się w kanale, węzeł natomiast traci całą zawartość bufora.

Należy tu zwrócić uwagę na duże uproszczenie: tak naprawdę uszkodzenie łącza może wystąpić w każdym punkcie łącza. Część danych, która została już przesłana przez punkt uszkodzenia, powinna dotrzeć do odbiornika. To oznacza, że średnio połowa danych transmitowanych w chwili wystąpienia awarii powinna mimo dotrzeć do miejsca przeznaczenia. Taki poziom symulacji jednak jest bardzo trudny do przeprowadzenia i dlatego ta cecha łączy zostanie pominięta.

3.1.2.2 Detekcja przestojów i awarii

Niedostępność łącza jest bardzo ważną informacją dla protokołów rutingu. Daje ona możliwość przekierowywania informacji na łącza, które są zdatne. Uszkodzenie łącza zazwyczaj wykrywa się w warstwie 2 modelu *TCP/IP* czy *ISO OSI*. Najczęściej realizowane jest za pomocą cyklicznie wysyłanych informacji sprawdzających stan łącza (ang. *keep alive*), których niewykrycie równoznaczne jest z utratą łączności. Można zauważyć, że wykrycie uszkodzenia węzła następuje po okresie równym czasowi propagacji. Podobna sytuacja ma miejsce w przypadku łączy, z tą różnicą, że uszkodzenie łącza może nastąpić w dowolnym jego punkcie, co mocno komplikuje symulację. Dla uproszczenia zakładam, że detekcja uszkodzenia łącza także odbywa się po czasie równym połowie czasu propagacji (odpowiada to sytuacji, gdy uszkodzenie łącza miało miejsce w jego środku).

Natomiast wykrycie przestojów łączy i węzłów — jako przewidywalne — odbywa się natychmiast, tak jak gdyby druga strona wiedziała, kiedy nastąpi rozłączenie.

Oczywiście w taki sam sposób odbywa się detekcja naprawienia łączy i węzłów.

3.1.2.3 Propagacja wiedzy o przyszłym wyłączeniu

Skoro wszystkie przestoje są znane, to wiadomości o planowanej niedostępności łączy czy węzłów powinny być przesyłane przez sieć przez specjalne pakiety zarządzające. Dla potrzeb symulacji zakładam, że żadne dane nie są transportowane, tylko wiedza o niedostępności łączy jest po prostu dostępna pozostałym węzłom w dowolnym momencie. Jest to dosyć duże uproszczenie, jednak narzut przesyłanych informacji o momencie wystąpienia wyłączenia jest minimalny, zaś węzły mogą wymieniać się informacjami o momencie przestoju na długo przed samym przestojem, więc w praktyce zawsze by o nim wiedziały. Dlatego można założyć, że każdy węzeł precyzyjnie zna chwile przestojów wszystkich elementów sieci.

Wiedza węzła o momencie wyłączenia elementów sieci nie zmienia faktu, że wiele protokołów rutingu z tej wiedzy nie korzysta.

3.1.3 Protokoły działające w sieci

3.1.3.1 Warstwa transportowa i niższe warstwy działające na danym łączu

Ponieważ czas dokładnej symulacji protokołów warstw niższych niż *Bundle* byłby bardzo duży, dlatego wszystkie protokoły niższych warstw symulowane są w dużym uproszczeniu. Zakłada

się, że na każdym łączu istnieje stos 3 pierwszych warstw modelu *TCP/IP* lub *ISO OSI* (spełniają one podobne funkcje). W zależności od używanego protokołu występują różne przebiegi czasowe w zależności od opóźnienia oraz współczynnika przekłamań bitów, które charakteryzują dane łącze. W aplikacji będą symulowane trzy protokoły, które zostały tak uogólnione, aby móc za ich pomocą symulować typowe protokoły występujące w sieci (możliwość dodatkowej konfiguracji np. narzutu dodawanego przez dany protokół). Trzema symulowanymi protokołami są:

- *TCP*,
- *UDP*,
- *LTP*.

3.1.3.1.1 TCP

Łącze korzystające z protokołu *TCP* zapewnia niezawodne dostarczenie danych do węzła docelowego. Symulowane są:

- Nawiązywanie połączenia — właściwa transmisja po nawiązaniu kontaktu rozpoczyna się dopiero po ustanowieniu połączenia, co w przypadku *TCP* oznacza co najmniej dwukrotne przesłanie informacji synchronizacyjnych.
- Dopasowywanie transmisji do możliwości łącza — dane wysyłane są wolniej na początku transmisji, tj. rozmiar okna jest bardzo mały. Wraz z upływającym czasem i ilością wysyłanych informacji wielkość okna i szybkość transmisji ulegają zwiększeniu. W praktyce rozmiar okna *TCP* rośnie wykładniczo 2 razy po każdym udanym dostarczeniu okna do węzła po drugiej stronie łącza. W sytuacji niedostarczenia okna do celu jego rozmiar zmniejszany jest dwukrotnie. W rzeczywistych działających implementacjach protokołu *TCP* po wykryciu pierwszego zgubienia lub uszkodzenia danych rozmiar okna zaczyna wzrastać liniowo, a nie wykładniczo. W aplikacji symulowane jest to za pomocą zmniejszenia mnożnika zwiększania okna z 2 do 1,1.
- Retransmisje — po otrzymaniu okna wysyłane jest potwierdzenie. W przypadku uszkodzenia danych zawartych w oknie (uszkodzenie łącza lub przekłamanie bitowe) dane są retransmitowane w całości. Nie istnieje więc system potwierdzania ostatniego poprawnie odebranego segmentu. Proces jest kontynuowany aż do momentu dostarczenia całości wiadomości.
- Ograniczenia czasowe i ograniczenia powtórzeń — nie jest symulowany maksymalny czas pomiędzy wysłaniem informacji a otrzymaniem potwierdzenia jej otrzymania (wynoszący domyślnie 2 minuty), po czym nastąpiłoby przerwanie transmisji. Zakładam, że przerwanie transmisji w ogóle nie może wystąpić, druga strona zawsze czeka w nieskończoność na przybycie wiadomości i zawsze retransmituje dane w przypadku nie uzyskania potwierdzenia w wyznaczonym przedziale czasowym.
- Brak konieczności istnienia fizycznego połączenia — mało znaną cechą protokołu *TCP* jest brak transmisji jakichkolwiek pakietów podtrzymujących połączenie gdy żadne dane nie są transmitowane. Dlatego możliwe jest fizyczne odłączenie kabla łączącego komputer z *Internetem* na pewien okres czasu, a następnie podpięcie go i kontynuowanie transmisji. Mechanizm ten może być użyty do niezrywania połączenia *TCP* gdy z góry wiadomo, że łącze lub któryś z węzłów będzie mieć przestój. Wystarczy, żeby warstwa aplikacji w tym czasie powstrzymała się od transmisji.

3.1.3.1.2 UDP

Protokół *UDP* nie gwarantuje niezawodnego dostarczania danych. Na podobnej zasadzie zachowuje się np. protokół *Bundle* działający bezpośrednio na warstwie *Ethernetu*.

Modyfikując ustawienia symulacji protokołu *UDP* (takie jak rozmiar nagłówka) można uzyskać wyniki zgodne z symulacją protokołu *Ethernet*. Symulowane jest tylko gubienie pakietów — w przypadku przekłamania bitów wewnątrz przesyłanych danych natychmiast następuje odrzucenie całości wiadomości.

3.1.3.1.3 LTP

Protokół *LTP* jest przeznaczony do wykorzystania na łączach o dużym opóźnieniu. Symulowane są:

- Zdolność *LTP* do zawieszania połączenia — w momencie wyłączenia łącza dane, które były w trakcie wysyłania, nie są później retransmitowane od początku; transmisja rozpoczyna się od momentu, w którym się zakończyła.
- Limity czasowe zerwania sesji są zaniedbywane, retransmisje prowadzone są do skutku lub do momentu, kiedy nastąpi uszkodzenie łącza.
- Wszystkie przesyłane segmenty są oznaczane jako „czerwone", to znaczy, że muszą być niezawodnie dostarczone do celu.
- Niezależna transmisja pakietów *Bundle* — *LTP* jako protokół transportowy służy do transmisji tablicy bajtów. Specyfikacja *LTP* nie definiuje, czy wewnątrz takiej tablicy ma się znajdować jeden pakiet *Bundle*, czy też wiele pakietów jeden po drugim. W tej pracy zakładam, że każdy pakiet *Bundle* transmitowany jest niezależnie.

3.1.3.2 Wiadomości

Każda wiadomość transmitowana w sieci ma postać tablicy bajtów. Wiadomość dzielona jest na paczki danych (mniejsze jednostki), które z kolei umieszczane są w pakietach protokołu *Bundle*.

3.1.3.3 Czasy opóźnień

Czas transmisji pakietu *Bundle* z węzła początkowego do końcowego może zostać podzielony na cztery typy:

3.1.3.3.1 Czas oczekiwania

Czas oczekiwania to czas, przez jaki musi oczekiwać wiadomość w węźle do momentu, kiedy zostanie uzyskany kontakt. Zależy on od ewentualnego uszkodzenia łącza, przestojów i momentu przybycia wiadomości.

3.1.3.3.2 Czas kolejkowania

Czas kolejkowania jest to czas, przez który wiadomość musi oczekiwać w kolejce, aż wysłane zostaną komunikaty o wyższym priorytecie (np. te które przybyły wcześniej).

3.1.3.3.3 Opóźnienie transmisji

Opóźnienie transmisji jest to czas, jaki zajmuje wysłanie wiadomości. Czas ten jest bezpośrednio zależny od przepływności łącza.

3.1.3.3.4 Opóźnienie propagacji

Opóźnienie propagacji jest to czas, jaki upływa pomiędzy wysłaniem danego sygnału a jego dotarciem do odbiornika po drugiej stronie łącza.

3.1.3.4 Bundle

Jedynym symulowanym protokołem warstwy sieci globalnej będzie protokół *Bundle*. Wszystkie dane wymieniane w sieci będą mieć formę ramki protokołu *Bundle*. Ze względu na dużą komplikację tego protokołu konieczne było przyjęcie wielu uproszczeń:

- Adresacja będzie opierać się na prostych identyfikatorach sieciowych, a nie adresach *URL* stosowanych w rzeczywistej sieci *Bundle*. Identyfikatorem będzie krótki tekst — nazwa węzła. Ze względu na optymalizację, wewnętrznie aplikacja i tak będzie używać raczej wskaźników na instancję klasy reprezentującą konkretny węzeł.
- Każda paczka danych zawiera określony czas życia, po upływie którego jest odrzucana.
- Ponieważ protokołem *Budle* przesyła się większe partie danych, które następnie są dzielone na mniejsze części, dlatego paczki danych są numerowane, aby można było stwierdzić, kiedy do celu dotarły wszystkie. Podobnie, każda paczka jest jednoznacznie identyfikowalna, co jest konieczne ze względu na sprawdzanie, czy dana paczka już znajduje się w buforze.
- Węzły mogą przejmować odpowiedzialność za dalszą transmisję danych. Przejmowanie takiej odpowiedzialności jest w symulacji cechą węzła.
- Istnieją trzy typy pakietów *Bundle* — dane, raporty oraz informacje o rutingu (w rzeczywistości raporty dodawane są do normalnej paczki danych, nie ma osobnego typu wiadomości). Raporty przenoszą potwierdzenia odebrania wiadomości, a także informację o przejęciu odpowiedzialności za retransmisję przez węzły. Informacje o rutingu są generowane przez protokoły rutingu, a ich odebranie nie jest nigdy potwierdzane.
- Liczba prób retransmisji, czas po jakim następuje próba retransmisji oraz czas życia każdej z paczek danych jest parametryzowany, ponieważ w zależności od środowiska inne parametry są znaczące. Na przykład w komunikacji międzyplanetarnej czas, po jakim nastąpi retransmisja, powinien przekraczać dwukrotność czasu propagacji. W innym przypadku retransmisja następowałaby wcześniej, niż fizycznie możliwe jest dotarcie potwierdzenia. Dla komunikacji między Ziemią a Marsem czas, po którym powinna nastąpić retransmisja, wynosi około 40 minut. W kontekście stosowania protokołu *Bundle* w sieci *Internet*, gdzie przeciętne opóźnienie nie przekracza kilkudziesięciu milisekund, retransmisja powinna następować po około kilkuset milisekundach.

3.1.4 Fizyczne ograniczenia sieci

3.1.4.1 Zdolność do wysłania całości wiadomości

W zależności od rozmiaru danych, które należy wysłać do przeznaczenia, decyzje o rutingu mogą brać pod uwagę ustalony czas trwania kontaktu i szacować, czy w danym czasie możliwe będzie wysłanie całej wiadomości, czy też lepiej będzie poszukać innej drogi. Zakładam, że każda z paczek danych jest traktowana niezależnie, a jej rozmiar i czas trwania transferu jest mały w stosunku do czasów trwania kontaktu, co automatycznie oznacza, że tego typu informacje nie będą uwzględniane.

3.1.4.2 Bufory

W węzłach występuje kilka typów buforów:

- Bufor wyjściowy łącza — jest to bufor, do którego pakiety *Bundle* trafiają po zrutowaniu w oczekiwaniu na swoją kolej w transmisji przez łącze. Wartość tego bufora jest konfigurowalna i będzie modyfikowana dla różnych symulacji. Dzięki temu można

sprawdzić płynność transmisji danych w sieci *DTN*. Jeżeli bufor się przepełni, to nowe pakiety są odrzucane.

- Bufor protokołu *Bundle* — jest to bufor przechowujący paczki danych na wypadek wystąpienia konieczności retransmisji. Jego rozmiar jest ograniczony tylko możliwościami maszyny, na której wykonuje się symulacja.
- Bufor protokołów transportowych — jest to bufor, w którym przechowywane są dane, które protokół transportowy pobrał z bufora wyjściowego łącza i przechowuje w celach ewentualnej retransmisji. Dla *TCP* rozmiar tego bufora nigdy nie przekroczy rozmiaru maksymalnego okna, dla *LTP* jego rozmiar nigdy nie przekroczy dwukrotności iloczynu przepływności łącza i czasu propagacji łącza.
- Bufor protokołów rutingu — niektóre protokoły rutingu muszą przechowywać pakiety. Przykładem może być protokół *AODV*, który przed przekazaniem pakietu do bufora wyjściowego potrzebuje najpierw rozgłosić wiadomość *RREQ* i poczekać na przyjście wiadomości *RREP*. W międzyczasie pakiet jest składowany w buforze protokołu rutingu.

3.1.4.3 Zużycie energii oraz mocy obliczeniowej

Czasami dodatkowym problemem może być koszt energetyczny wysłania wiadomości. Np. wysłanie wiadomości do drugiego węzła za pośrednictwem węzła pośredniego może dawać znaczną oszczędność energii (ilość potrzebnej energii rośnie z kwadratem odległości). Podobnie, do przetworzenia skomplikowanych protokołów może być potrzebna duża moc obliczeniowa, a nie każda sieć ma węzły wyposażone w wystarczająco mocne procesory. Te aspekty zostają całkowicie pominięte w tej pracy.

3.1.5 Protokoły rutingu

Każdy z węzłów obsługuje jeden z protokołów rutingu. Każdy z węzłów może obsługiwać inny protokół rutingu. Czasami wykorzystanie różnych protokołów rutingu może powodować problemy, np. protokół *AODV* może nie odnaleźć drogi w sieci, gdy występują węzły nieobsługujące wiadomości *RREQ* i *RREP*. Dlatego większość badanych scenariuszy będzie się opierać na wykorzystaniu tylko jednego protokołu rutingu w całej sieci.

Protokoły rutingu mogą wymieniać między sobą wiadomości na temat stanu sieci, aczkolwiek nie wszystkie protokoły rutingu tego wymagają. W celu uproszczenia zakładam, że węzły znają całą strukturę sieci, a przekazują sobie jedynie informacje o tym, które łącza są dostępne, a także kiedy wystąpią kontakty (lub ich brak), lub też jakie są prawdopodobieństwa tych wydarzeń.

Mechanizm rutingu oparty jest na schemacie znanym z sieci *Internet*:

1. Kiedy pojawia się nowy pakiet *Bundle*, jest on przekazywany protokołowi rutingu.
2. Protokół rutingu wybiera interfejs (lub interfejsy w przypadku zwielokrotniania wiadomości), do bufora którego przekazuje pakiet albo odrzuca go gdy nie może odnaleźć drogi.
3. Jeżeli pakiet przekazany zostanie do któregoś z buforów wyjściowych łączy, to oczekuje tam na swoją kolej wysłania lub na dostępność łącza (w zależności od tego, czy łącze jest dostępne w czasie rutowania i czy nie nastąpią zmiany stanu łącza w czasie oczekiwania na wysłanie).

Do symulowania wybrałem następujące protokoły rutingu:

- ruting statyczny,
- protokół epidemiczny,
- protokół gradientowy,
- protokół gradientowy z wyborem dostępnej drogi i odrzucaniem pakietów,
- protokół oparty na *AODV*,
- protokół oparty na *OSPF*,
- protokół przewidujący kontakty.

3.1.5.1 Ruting statyczny

Ruting statyczny to najprostsza forma rutingu, w której przełączanie wiadomości odbywa się na podstawie tabeli rutingu podanej przez operatora.

3.1.5.2 Ruting epidemiczny

Węzeł obsługujący protokół rutingu epidemicznego wysyła odebraną wiadomość do wszystkich swoich sąsiadów (tzn. przez wszystkie swoje łącza). Pamiętane są wszystkie wysłane wiadomości, dzięki czemu gdy węzeł otrzyma tę samą wiadomość powtórnie, nie będzie jej niepotrzebnie przetwarzać ani powtórnie wysyłać.

3.1.5.3 Ruting gradientowy

W sytuacji zastosowania rutingu gradientowego każde łącze ma przypisaną metrykę, która dla prostoty określona jest wyłącznie przez wartość oczekiwaną czasu propagacji łącza. Protokół gradientowy używając algorytmu Dijkstry wyznacza odległość wszystkich przyległych do niego węzłów (tzn. takich, do których dysponuje łączami). Odległość jest rozumiana jako suma metryk używanych łączy. Następnie wyznacza gradient, który definiowany jest następująco:

$$g = \frac{\Delta S}{M}$$

Gdzie:
ΔS — różnica pomiędzy odległością (w sensie sumy metryk) od węzła docelowego węzła rutującego i węzła sprawdzanego,
M — metryka łącza.

Następnie wybierane jest to łącze, które ma największy gradient, niezależnie od tego, czy jest dostępne, czy też nie. Dlatego gdy łącze nie jest dostępne pakiet będzie oczekiwać w buforze wyjściowym na dostępność łącza. Jeżeli gradient jest ujemny, to skierowanie pakietu w tym kierunku oznaczałoby oddalanie się od węzła i dlatego taka droga nie jest brana pod uwagę.

3.1.5.4 Ruting gradientowy z wyborem dostępnej drogi i odrzucaniem pakietów

Ruting gradientowy z automatycznym wyborem dostępnej drogi działa podobne do normalnego rutingu gradientowego. Protokół wyznacza gradienty dla wszystkich dostępnych łączy. Następnie wybiera łącze o największym dodatnim gradiencie. Jednak pod uwagę brane są tylko te łącza, które są dostępne w momencie rutowania pakietu. Jeżeli takie łącze nie zostanie odnalezione, to pakiet jest odrzucany.

3.1.5.5 Protokół oparty na AODV

Zachowanie protokołu rutingu opartego na *AODV* jest bardzo podobne do wcześniejszego opisu protokołu *AODV* (rozdział 2.7.5.2). Po otrzymaniu żądania zrutowania pakietu węzeł źródłowy rozgłasza pakiet *RREQ* do wszystkich węzłów, które są do niego bezpośrednio połączone. Węzły te rozgłaszają go dalej. W końcu pakiet dociera do węzła docelowego, który odpowiada wiadomością *RREP*. Otrzymanie przez węzeł wiadomości *RREP* i *RREQ* powoduje utworzenie wpisu w tabeli rutingu. Dzięki temu węzły pośredniczące są w stanie kierować wymianą danych pomiędzy węzłem początkowym i końcowym.

Wszystkie znane trasy tracą ważność po pewnym konfigurowalnym okresie. Dzięki temu ewentualne retransmisje lub transmisje po dłuższym okresie czasu będą się odbywać tylko po nowo wykrytej trasie.

3.1.5.6 Protokół oparty na OSPF

OSPF (ang. *Open Shortest Path First* — w wolnym tłumaczeniu „pierwszeństwo ma najkrótsza ścieżka") to protokół rutingu stosowany w sieci *Internet*. W uproszczeniu jego główną cechą jest utrzymywanie pełnej wiedzy o stanie obszaru sieci przez wszystkie węzły. Dzięki temu każdy z węzłów może znaleźć optymalną drogę przy użyciu algorytmu Dijkstry. *OSPF* jest bardzo skomplikowanym protokołem, dlatego wersja symulowana w aplikacji jest mocno uproszczona. Przyjmuje się, że wysyła i odbiera on dwie wiadomości:

- Informacja o zmianie stanu łącza — wysyłana jest przez węzeł w momencie wykrycia awarii lub naprawy łącza. Wiadomość ta jest rozgłaszana do wszystkich węzłów w sieci.
- Cykliczna synchronizacja — informacja o stanie łączy może się zdesynchronizować w niektórych węzłach np. jeśli nie dotarły do nich pakiety z aktualizacją stanu łącza. Dlatego co pewien czas każdy z węzłów rozgłasza pakiet ze stanem wszystkich swoich łączy.

Łącza, które są w stanie przestoju lub które prowadzą do węzła w stanie przestoju, są traktowane jako niedostępne. Jeżeli nie zostanie odnaleziona droga prowadząca do celu, to pakiet jest odrzucany.

Protokół ten wykorzystuje ruting pesymistyczny i wymaga bezpośredniego kontaktu między węzłem początkowym a końcowym.

3.1.5.7 Protokół przewidujący kontakty

Każda instancja protokołu rutingu przewidującego kontakty ma dostęp do informacji o znanych momentach wystąpienia przestojów. Dlatego może próbować przewidzieć, jakie dostępne połączenia będą występować w sieci w przyszłości na poszczególnych węzłach i na tej podstawie może szukać optymalnej trasy. Oczywiście taki ruting nie jest w stanie w pełni poradzić sobie z awariami, które mogą pojawiać się w sposób nieprzewidywalny.

W prostej implementacji tego protokołu zakładam, ze węzły znają całą strukturę sieci. Co więcej, znają też momenty włączania i wyłączania węzłów (jeżeli np. satelita krążący dookoła jakiejś planety raz na jakiś czas jest niedostępny ze względu na zasłonięcie planetą, to moment tego zasłonięcia jest przewidywalny zarówno dla satelity, jak i stacji, z którą się komunikuje; stąd wymiana informacji między węzłami nie jest konieczna).

Paczka danych zmierzając do miejsca docelowego może zatrzymać się po drodze ze względu na uszkodzenie jednego z elementów sieci. Wtedy trafia ona do bufora wyjściowego łącza i oczekuje na naprawę łącza. Jest to więc ruting optymistyczny.

3.2 Symulacja

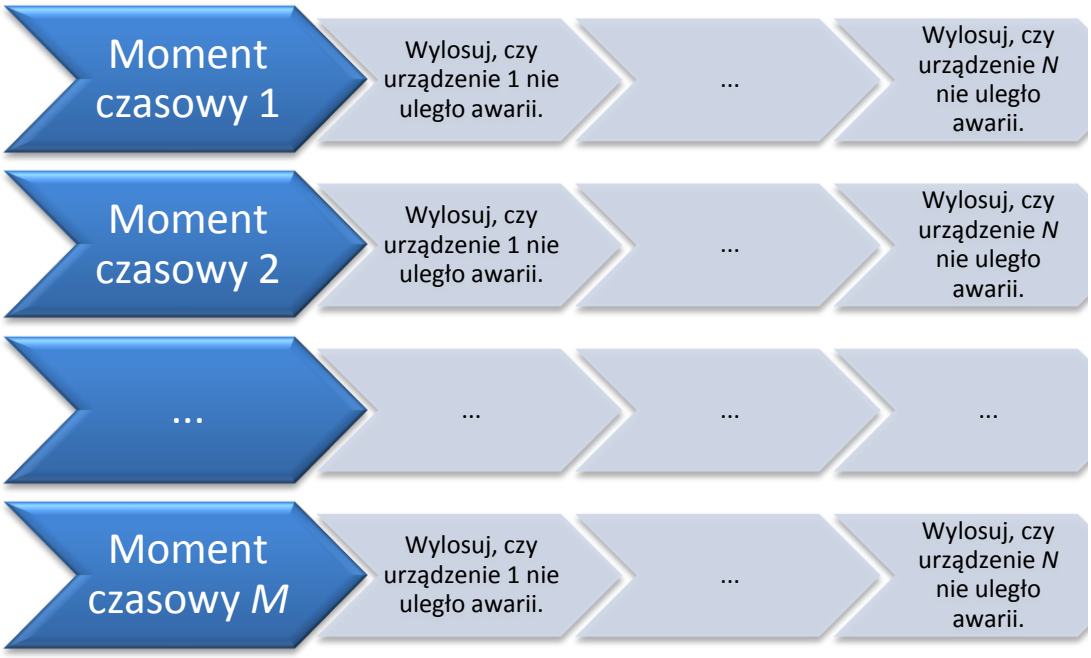
Symulacja procesów zachodzących w sieciach telekomunikacyjnych jest zadaniem trudnym zarówno ze względu na złożoność protokołów, jak również ze względu na spore wymagania związane z szybkością działania procesu symulacji. Dostępne narzędzia symulujące działanie sieci *IP* przeprowadzają symulację wielokrotnie wolniej, niż ten proces przebiega w rzeczywistości. Wynika to z faktu, że komputer musi jednocześnie symulować pracę wykonywaną normalnie przez wiele elementów sieci, w tym kilkaset węzłów, karty telekomunikacyjne, węzły pośrednie (takie jak rutery) i jeszcze przeprowadzać analizę statystyczną wyników, aby zaprezentować wszystko użytkownikowi.

Zastosowanie niektórych zaawansowanych algorytmów pozwala znacznie przyspieszyć proces symulacji. Szczególnie trudnymi zagadnieniami są: synchronizacja wydarzeń w czasie oraz symulowanie różnych rozkładów prawdopodobieństwa.

3.2.1 Symulacja czasu oraz losowości zdarzeń

Założenia projektowe są takie, że urządzenia mogą uszkadzać się i być naprawiane zgodnie z pewnymi rozkładami prawdopodobieństwa. W zależności od wylosowanych wartości zdarzenia w sieci następują w różnym czasie. Symulacja musi zarówno zachować kolejność zdarzeń, Zapewnić występowanie zdarzeń zgodnie z zadanymi rozkładami prawdopodobieństw oraz wykonywać wszystkie operacje możliwie najszybciej. Najprostszym sposobem symulacji kolejnych losowych zdarzeń w sieci byłoby przyjęcie niewielkiego przedziału czasowego, a więc założenie ziarnistości czasu. Wszystkie zdarzenia występowałyby tylko we wcześniej ustalonych momentach czasowych. Na przykład, jeśli dane urządzenie uszkadza się raz do roku, to automatycznie przy przedziale czasu na poziomie 1 ms (1 rok = 31 536 000 000 ms) prawdopodobieństwo zepsucia się urządzenia w tym przedziale czasu wynosi $3{,}17\times10^{-10}$. Oznacza to, że przyjmując powyższy model statystycznie wykonalibyśmy 31 536 000 000 losowań, zanim jedno z nich zakończyłoby się wylosowaniem uszkodzenia urządzenia. W ten sposób jednak nie da się symulować pracy sieci telekomunikacyjnej, ponieważ potrzebowalibyśmy mocy obliczeniowej tysięcy komputerów, aby wykonać nawet najprostszą symulację. Dodatkowym utrudnieniem byłaby ziarnistość czasu; w sieci telekomunikacyjnej najmniejszy przedział na poziomie 1 ms mógłby się okazać zbyt duży. Sposób działania tego algorytmu prezentuje rysunek 3-1.

Drugie, znacznie lepsze i zaimplementowane w ramach tej pracy rozwiązanie to uprzednie losowanie momentu wystąpienia danego wydarzenia (w tworzonej aplikacji odpowiada za to klasa bazowa *RandomGenerator*, która — w zależności od konkretnej implementacji — generuje czas o różnych rozkładach prawdopodobieństwa) oraz zapisywanie go w globalnym zbiorze wydarzeń. Pewien element symulujący czas (w aplikacji jest to klasa *Timer*) pobiera ze zbioru zapisanych wydarzeń to, które ma najmniejszą wartość czasową oraz uruchamia związane z danym wydarzeniem urządzenie. Kiedy np. urządzenie uszkadza się, program symulujący automatycznie losuje czas naprawienia urządzenia i dodaje go do globalnego zbioru



Rysunek 3-1: Przykład prostego i nieefektywnego systemu symulacji uszkadzania się i naprawiania *N* urządzeń.

zdarzeń. Ze względu na szybkość dodawania oraz usuwania elementów zbiór ten powinien być zaimplementowany jako kolejka priorytetowa, co daje złożoność algorytmu obsługi pojedynczego wydarzenie na poziomie O(ln *N*), gdzie *N* to liczba aktualnie zapisanych wydarzeń. Konieczna jest także funkcja usuwania dodanego wydarzenia, ponieważ urządzenie może uprzednio zadeklarować, że ma w danym momencie czasowym np. wysłać jakieś dane, ale jeżeli wcześniej nastąpi uszkodzenie węzła i utrata wszystkich danych zapisanych w buforze, to węzeł nie będzie już miał nic do wysłania. Rysunek 3-2 przedstawia zasadę działania klasy *Timer*.

3.2.2 Symulacja rozkładu czasu

Aby móc symulować czas oraz losowe zachodzenie zdarzeń w sieci konieczne jest symulowanie różnych rozkładów prawdopodobieństwa wystąpienia danego zdarzenia w czasie. Niestety, praktycznie wszystkie biblioteki programistyczne umożliwiają losowanie co najwyżej wartości zmiennoprzecinkowych z zakresu [0;1] o liniowym rozkładzie prawdopodobieństwa. Konieczne jest uzyskanie rozkładu pożądanego za pomocą rozkładu dostępnego.

Szukamy funkcji postaci

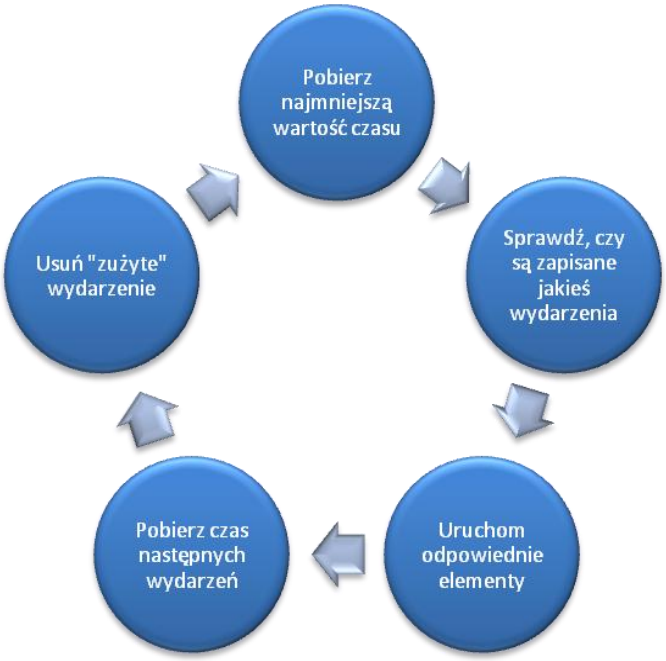
$$t = h(w)$$

gdzie *w* to wylosowana liczba z przedziału [0;1].

Dane są:

$g_t(t) = f(t)$ — gęstość rozkładu prawdopodobieństwa uzyskania wartości czasu t

$$g_w(w) = \begin{cases} 1 \; dla \; w \in [0;1] \\ 0 \; dla \; w \in (-\infty;0) \cup (1;+\infty) \end{cases}$$



Rysunek 3-2: Ilustracja działania klasy Timer — zaawansowana symulacja czasu.

Prawdopodobieństwa uzyskania odpowiadających wartości muszą być sobie równe, zatem:

$g_t(t)dt = g_w(w)dw$

Po obustronnym scałkowaniu otrzymujemy:

$G_t(t) = G_w(w)$ gdzie G — dystrybuanty odpowiadających rozkładów prawdopodobieństwa

$t = G_t^{-1}(G_w(w))$

$\boldsymbol{h(w) = G_t^{-1}(w)}$ **gdzie** $\boldsymbol{G_t(t) = \int_{-\infty}^{t} g_t(t)dt}$

Np. dla rozkładu wykładniczego o postaci:

$g_t(t) = \frac{1}{A}e^{-\frac{t}{A}}$ gdzie *A* — wartość oczekiwana wylosowanego czasu

Otrzymujemy:

$G_t(t) = 1 - e^{-\frac{t}{A}}$

$G_t^{-1}(t) = -A\ln(1-t)$

$\boldsymbol{h(w) = -A\ln(1-w)}$

3.2.3 Symulacja przekłamań

Kiedy w danej ramce występuje przekłamanie, dzięki sumie kontrolnej *CRC* jest możliwe wykrycie błędu i odrzucenie wadliwej ramki. Jeśli jednak ramka ma 10000 bitów, zaś dany jest współczynnik przekłamywania bitów, to mamy podobny problem z wydajnością, jak w przypadku wydarzeń losowych umieszczonych w czasie. Musielibyśmy przeprowadzić 10 000

osobnych losowań. Rozwiązanie tego problemu jest jednak proste, ponieważ interesują nas tylko dwa przypadki związane bezpośrednio z odrzucaniem całej ramki — albo przekłamaniu nie ulega żaden bit, albo przekłamaniu ulega dowolna liczba bitów, ale co najmniej jeden. Prawdopodobieństwo odrzucenia całej ramki wynosi:

$$P_o = 1 - (1 - BER)^n$$

Gdzie:

n — liczba bitów w ramce
BER — prawdopodobieństwo przekłamania pojedynczego bitu.

3.2.4 Symulowanie generowania ruchu w sieci

Generowanie ruchu ma postać generowania pewnej liczby bajtów do wysłania. Każdy węzeł co określony odstęp czasu (zadany jednym z obsługiwanych rozkładów czasu) generuje pewną liczbę bajtów do wysłania (oczywiście liczba ta także ma pewien rozkład prawdopodobieństwa, podobny do rozkładu czasu, przy czym wartość zaokrąglana jest do liczby całkowitej), a następnie wysyłana do jednego z węzłów. Wtedy następuje cały proces dzielenia danych na paczki, opakowywania pakietem *Bundle* i kierowania ich zgodnie z protokołem rutingu do określonych łączy. W rozpatrywanej symulacji zakładam, że węzeł każdorazowo dokonuje wyboru węzła docelowego i że są to decyzje całkowicie od siebie niezależne (tzn. na dany wybór węzła nie mają wpływu poprzednie decyzje). W sieci często istnieje sytuacja, w której jeden węzeł wysyła informacje częściej do niektórych węzłów, a rzadziej do innych. Aby umożliwić przeprowadzenie tego typu symulacji, źródło informacji zawiera wewnętrzną tabelę prawdopodobieństw. Każdemu węzłowi (z wyjątkiem wysyłającego) przypisane jest prawdopodobieństwo wysłania tych wiadomości właśnie do niego.

3.3 Rozwiązania programistyczne

Przygotowując się do tworzenia większej aplikacji, każdy programista staje przed wieloma problemami związanymi z zaprojektowaniem struktury oraz funkcji aplikacji. Symulator sieci *DTN* stawia dodatkowe wymagania związane z samym procesem symulacji, prowadzeniem statystyk umożliwiających uzyskanie odpowiedzi na stawiane pytania czy też procesem konfiguracji sieci. Najistotniejsze elementy z punktu widzenia inżynierii oprogramowani postanowiłem skrótowo opisać poniżej.

3.3.1 Język programowania

Aplikacja tworzona jako praca magisterska ma małe szanse stania się używaną powszechnie na całym świecie przez miliony użytkowników, dlatego stosowanie języków niskiego poziomu, takich jak *C* lub *C++,* które co prawda dają istotne przyspieszenie procesu symulacji, ale jednocześnie znacznie wydłużają sam proces tworzenia oprogramowania, nie wydaje się uzasadnione. Za optymalny język programowania dla takiego projektu uznałem język *C#.* Jego najważniejsze cechy zestawiono poniżej.

- Pełna obiektowość.
- Pełne zarządzenie pamięcią, czyli tzw. zbieracz śmieci (ang. *garbage collector*), który uwalnia programistę od samodzielnego zwalniania pamięci i jednocześnie gwarantuje brak tzw. wycieków pamięci (ang. *memory leaks*).

- Ze względu na wzorowanie się twórców języka *C#* na języku *Java*, *C#* posiada wszystkie wbudowane w *Javę* ułatwienia programistyczne, np. wbudowaną klasę *string* reprezentującą napis; wszystkie klasy automatycznie dziedziczą po klasie *object*, przejmując w ten sposób metody pozwalające np. w czasie działania programu pobrać nazwę klasy (zostało to wykorzystane w systemie logowania).
- Dzięki daleko idącej optymalizacji *C#* jest tylko o 10-20% wolniejszy od języka *C++*, co w porównaniu z *Javą* wolniejszą w zależności od aplikacji 1,4-20 razy, jest bardzo dobrym wynikiem.
- *C#* działa na platformie *.Net*, dzięki czemu daje dostęp do największej na świecie biblioteki programistycznej, aktualnie dostępnej tylko pod system operacyjnym *Windows*, ale już wkrótce, dzięki projektowi *Mono*, dostępnej praktycznie na wszystkich platformach, w tym w pierwszej kolejności na *Linuxie*. Oznacza to, że program bez żadnych modyfikacji już wkrótce będzie można skompilować praktycznie na dowolnym systemie operacyjnym.

3.3.2 Komunikacja z użytkownikiem

Aplikacja czasem musi poinformować użytkownika o nieprawidłowości wprowadzonych danych lub o stanie wewnętrznym, którym może być np. procent wykonanej symulacji. Do tego celu wykorzystane zostanie standardowe wyjście aplikacji (domyślnie jest to konsolka tekstowa).

W celu sprawdzania sposobu działania aplikacji i poprawności realizacji działań, przydatne jest także wygenerowanie dzienników, które będą informować o najważniejszych wydarzeniach zachodzących wewnątrz aplikacji. Zazwyczaj chcemy po prostu wiedzieć, że jakiś węzeł uległ uszkodzeniu, protokół rutingu zmienił wpisy w tablicy rutingu, ze względu na ograniczenie bufora odrzucone zostały niektóre dane itp. Ważne jest także, aby rejestrowane były informacje o czasie symulacji oraz identyfikator konkretnego elementu w sieci. Nie jest to rozwiązanie konieczne, aby symulator działał, ale pozwala na bardzo proste sprawdzanie poprawności działania symulowanej sieci w momencie tworzenia aplikacji.

Oczywiście generowanie wspomnianych informacji jest czasochłonne i może drastycznie ograniczać szybkość wykonywania symulacji. Optymalna sytuacja jest wtedy, kiedy mamy do dyspozycji dwie wersje: jedną wersję testową (ang. *debug*), która kompiluje się w taki sposób, że dodaje rejestrowanie do wszystkich występujących zdarzeń wewnątrz aplikacji oraz drugą wersję przeznaczoną dla użytkowników (ang. *release*), która jest zoptymalizowana pod względem szybkości wykonywania i nie tworzy żadnych dzienników.

3.3.3 Konfiguracja

Optymalny system konfiguracji to taki, który jednocześnie daje użytkownikowi możliwość ręcznej, tekstowej konfiguracji programu, sieci telekomunikacyjnej oraz ustawień, a także mając budowę drzewiastą umożliwia grupowanie obiektów i przekazywanie każdemu z nich tylko tego fragmentu drzewa, który jest mu potrzebny do skonfigurowania się. Dzięki ostatniej cesze łatwo można uzyskać profesjonalne, obiektowe rozdzielenie wiedzy o konfiguracji na poszczególne elementy i późniejszą prostą rozbudowę funkcji.

Powyższe wymagania spełnia format *XML* i z tych powodów został on zaadaptowany na potrzeby aplikacji. Poza ww. cechami *XML* daje także możliwość prostego tworzenia walida

torów. Ponadto istnieje dużo gotowego, darmowego oprogramowania umożliwiającego edycję i walidację plików, co znacznie uprasza proces tworzenia plików.

Ze względów praktycznych konfiguracja podzielona została na 4 części, z czego 3 to pliki konfiguracyjne. Podział konfiguracji na części umożliwia np. symulowanie tej samej sieci przy różnych ustawieniach symulacji — możliwe jest wykorzystanie innego pliku z inną konfiguracją symulacji.

3.3.3.1 Konfiguracja za pomocą linii poleceń

Konfiguracja za pomocą linii poleceń Jest rozwiązaniem stosunkowo niewygodnym, ponieważ wymaga wielokrotnego podawania tych samych ustawień, dlatego została wykorzystana tylko do wprowadzania głównych plików z konfiguracją (jeśli nie są wykorzystywane domyślne).

3.3.3.2 Konfiguracja globalna

Wiele protokołów lub elementów w sieci jest konfigurowalnych. Przykładami mogą być czas po jakim protokół Bundle retransmituje niepotwierdzone dane czy rozmiar bufora wyjściowego łączy. Jest to konfiguracja globalna dla całej symulacji. Domyślna nazwa pliku z tą konfiguracją to *Configuration.xml*. Plik umożliwia modyfikację praktycznie wszystkich wykorzystywanych w aplikacji stałych związanych z protokołami.

3.3.3.3 Konfiguracja sieci

Konieczne jest przekazanie aplikacji struktury sieci, która ma być symulowana. Definicja sieci domyślnie znajduje się w pliku *Network.xml*. Zawiera opis m.in. węzłów, łączy, źródeł informacji i stosowanego rutingu. Należy pamiętać, że samo podanie np. protokołu transportowego wykorzystywanego na łączu nie definiuje jeszcze w pełni zachowania tego elementu. Jego zachowanie jest dodatkowo uzależnione od konfiguracji globalnej.

3.3.3.4 Konfiguracja statystyk

Sensem symulacji jest zebranie statystyk umożliwiających analizę sieci. Definicję typu i ilości gromadzonych danych zwiera domyślnie plik *Statistic.xml*. Statystyki generowane są przez instancje klas pobierających dane z elementów udostępniających interfejs do pobierania statystyk. Każda instancja jest konfigurowana osobno, określane są m.in. elementy, z których pobierane są dane, pobierane parametry, format i nazwa pliku wyjściowego oraz czas pobierania informacji.

3.3.4 Identyfikacja elementów

Ze względu na potrzeby konfiguracji oraz łączenia ze sobą definiowanych w konfiguracji elementów sieci część urządzeń otrzymuje unikalny identyfikator (klasa *Identificable*) oraz automatycznie zapisuje się do globalnego zbioru par [identyfikator, element]. Umożliwia to łatwe łączenie się elementów oraz tworzenie ich statystyk, np. łącze o odpowiedniej przepływności jest doczepiane do elementów o identyfikatorach „Węzeł1” oraz „Węzeł2” zgodnie z konfiguracją sieci zawartą w jednym z plików konfiguracyjnych, zaś statystyki pobierane mogą być np. z łącza o identyfikatorze „Łącze międzyplanetarne” zgodnie z konfiguracją statystyk. Identyfikatory są tekstowe. Ze względu na stosunkowo wolny czas przetwarzania napisów w programie identyfikatory te są używane tylko podczas pobierania

statystyk (jest to stosunkowo rzadka czynność) oraz podczas konfiguracji, kiedy są zamieniane na odpowiadające im referencje.

3.3.5 Statystyki

Tworzenie statystyk jest konieczne, żeby wykonywanie symulacji dało w ogóle jakieś wyniki. Idea tworzenia statystyk jest oparta na systemie symulacji czasu w aplikacji. Statystyki są pobierane z podanych w konfiguracji elementów w określonych przez użytkownika momentach czasowych. Użytkownik wybiera, ile będzie elementów pobierających statystyki, jaki będzie format ich zapisywania (obsługiwany jest m.in. format odczytywany przez program *Excel*), podaje rozkład czasu, co jaki będzie przeprowadzane pobieranie statystyk (w szczególności regularnie co stały odstęp czasu lub jednokrotnie) podaje nazwę pliku, w którym mają być zapisane dane, podaje identyfikatory elementów, których statystyki mają być zapisywane, a następnie podaje, jakie parametry mają być zapisywane.

Koncepcja parametrów opiera się na założeniu, że każdy z elementów może mieć unikalny zestaw parametrów, które zwraca. Dlatego udostępniając parametry przetwarza je do jednolitego formatu postaci pary: [nazwa parametru, wartość], gdzie „nazwa parametru" to identyfikator tekstowy. Tabela 3-1 przedstawia przykład zestawu parametrów, jakie mogą cechować pojedyncze urządzenie w sieci.

3.3.6 Model klas w UML

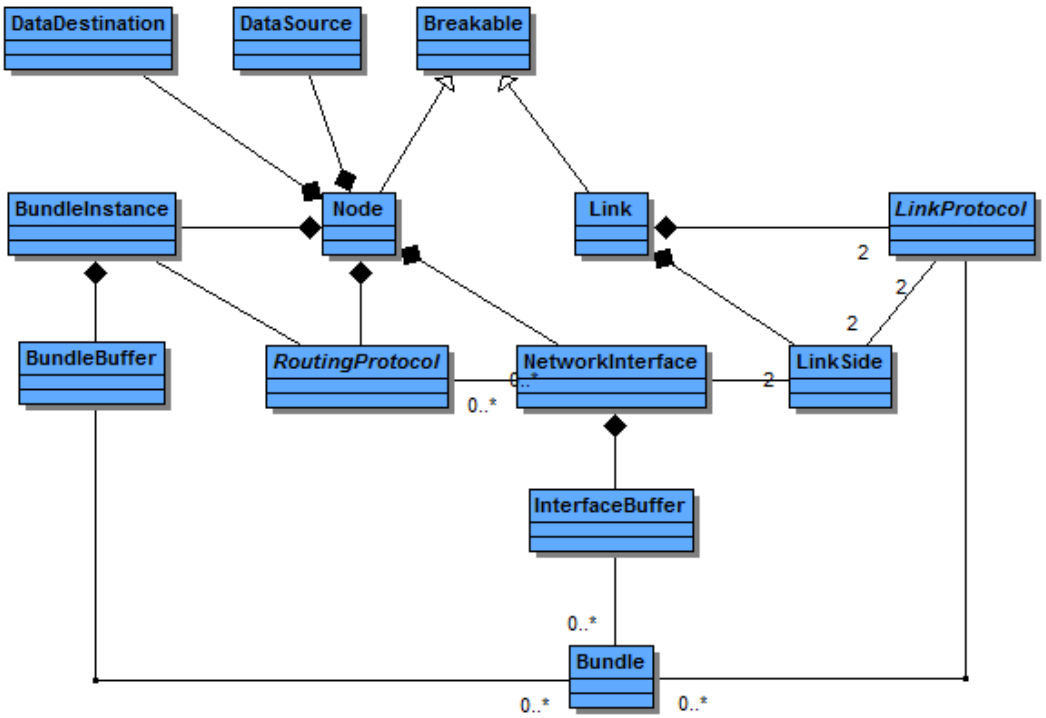
Symulator jest prostą aplikacją, jednowątkową i wykorzystującą jedynie operacje synchroniczne. Dlatego projektowanie wewnętrznej struktury komunikacji nie ma większego sensu. Natomiast warto zidentyfikować najważniejsze klasy i ich strukturę, co znacznie przyspiesza późniejszą implementacją i ogranicza nieprzemyślane decyzje skutkujące w późniejszym czasie kosztownymi modyfikacjami.

Do pracy załączam jedynie model podstawowych klas w języku *UML*, które przedstawiają strukturę sieci *DTN* (rysunek 3-3), strukturę synchronizacji czasu symulacji (rysunek 3-4), model protokołów transportowych wykorzystywanych w sieci *DTN* (rysunek 3-5), model protokołów rutingu wykorzystywanych w sieci *DTN* (rysunek 3-6), model generowania statystyk (rysunek 3-7) oraz model generatorów losowych (rysunek 3-8). Język *UML* definiuje znaczenie każdego elementu, oraz relacji występujących między elementami. Z tego powodu do każdego elementu dołączony jest tylko opis działania poszczególnych elementów.

3.3.6.1 Struktura sieci DTN

Tabela 3-1: Przykład parametrów, jakie może mieć jedno z urządzeń

Nazwa parametru	Wartość
Identyfikator	Węzeł główny
Jest zepsute	Nie
Łączny czas awarii	450 s
Łączny czas poprawnego działania	1004876 s
Ilość odebranych informacji	45 MB



Rysunek 3-3: Struktura sieci DTN.

Rysunek 3-3 przedstawia strukturę sieci *DTN*.

Breakable — klasa bazowa dla elementów, które ulegają uszkodzeniom i przestojom.

Node — reprezentuje węzeł w sieci *DTN*.

Link — reprezentuje łącze w sieci *DTN*.

LinkSide —reprezentuje jedną stronę łącza. Węzły wysyłają paczki danych wprowadzając je do jednej strony łącza, a następnie dane po drugiej stronie łącza przekazywane są do interfejsu sieciowego węzła (*NetworkInterface*)

LinkProtocol — jest to protokół wykorzystywany do transport danych (transportowy) na danym łączu.

Bundle — reprezentuje pakiet *Bundle*.

BundleInstance — instancja protokołu *Bundle*. Obsługuje przetwarzanie pakietów *Bundle*, retransmisje i wysyłanie potwierdzeń. Odpowiada także za przejmowanie odpowiedzialności za retransmisje.

BundleBuffer — bufor protokołu *Bundle*, gdzie przechowywane są paczki danych na wypadek konieczności retransmisji.

NetworkInterface — reprezentuje połączenie węzła z łączem.

InterfaceBuffer — jest to element przechowujący paczki danych dla których znaleziono trasę aż do momentu, kiedy łącze, przez które mają być wysłane dane, będzie wolne.

DataSource — źródło danych, element generujący ruch w sieci.

DataDestination — jest to element, do którego trafiają paczki danych po dotarciu do węzła docelowego. Jego głównym zadaniem jest prowadzenie statystyk i sprawdzanie, czy dana paczka została dostarczona już wcześniej.

RoutingProtocol — element, którego zadaniem jest rutowanie paczek danych zgodnie z wybranym protokołem rutingu.

3.3.6.2 Synchronizacja czasu symulacji

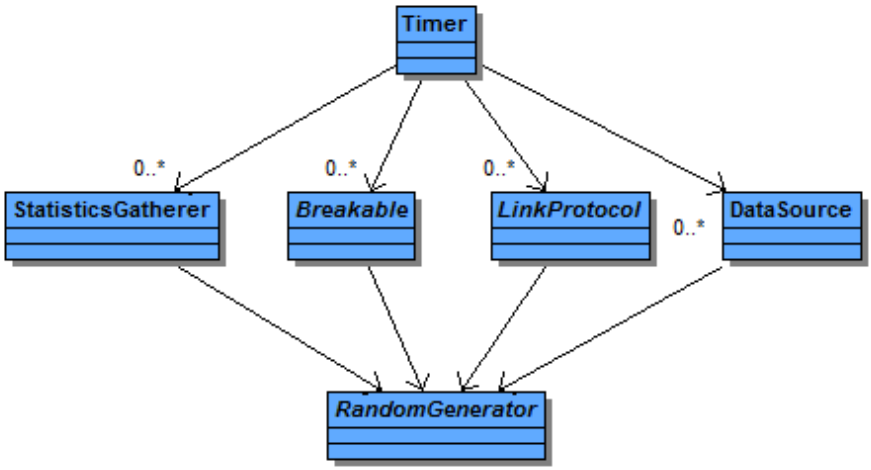
Rysunek 3-4 przedstawia model synchronizacji czasu symulacji.

Timer — jest to klasa odpowiedzialna za synchronizację czasową. Przechowuje kolekcję momentów czasowych oraz przypisanych im akcji. Elementy, które chcą zarejestrować jakieś wydarzenia w określonym momencie w przyszłości (w przyszłości z punktu widzenia czasu symulacji), przekazują klasie *Timer* moment czasowy oraz element reprezentujący akcję do wykonania (który fizycznie implementowany jest jako tzw. delegat, czyli wskaźnik na metodę). Klasa *Timer* pobiera z kolekcji zarejestrowanych akcji tę, która związana jest z najwcześniejszym momentem czasowym, wykonuje zadaną akcję, usuwa akcję z kolekcji, pobiera następną akcję.

LinkProtocol — protokół transportowy działający na łączu. Rejestruje w klasie *Timer* m.in. zdarzenia związane z dostarczeniem paczki danych po czasie propagacji.

DataSource —źródło danych. Rejestruje w klasie *Timer* moment czasowy (wygenerowany uprzednio przez generator losowy), w którym wygenerowana zostanie kolejna porcja danych.

Breakable — klasa bazowa dla elementów sieci, które ulegają uszkodzeniom i przestojom.



Rysunek 3-4: Struktura synchronizacji czasu symulacji. Klasy DataSource, Breakable, LinkProtocol i StatisticsGatherer są przykładowymi elementami rejestrującymi wydarzenia w klasie Timer w celu uzyskania synchronizacji czasowej.

Rejestruje w klasie *Timer* wydarzenia związane z naprawieniem i uszkodzeniem elementów sieci.

StatisticsGatherer — jest to element pobierający statystyki z elementów sieci. Statystyki pobierane są zgodnie z rozkładem losowym czasu symulacji podanym przez użytkownika. Klasa ta rejestruje w klasie *Timer* moment najbliższego pobrania statystyk.

RandomGenerator — generator losowy, który z zadanym rozkładem prawdopodobieństwa generuje losowy moment czasowy.

3.3.6.3 Protokoły wykorzystywane w symulacji

Rysunek 3-5 przedstawia model protokołów transportowych. Rysunek 3-6 przedstawia model protokołów rutingu.

LinkProtocol —reprezentuje protokół transportowy działający na łączu w sieci *DTN*.

TcpProtocol — reprezentuje protokół *TCP*.

LtpProtocol — reprezentuje protokół *LTP*.

UdpProtocol — reprezentuje protokół *UDP*.

RoutingProtocol — reprezentuje protokół rutingu działający na danym węźle w sieci *DTN*.

GradientWithRedirectionRoutingProtocol — reprezentuje gradientowy protokół rutingu z wyborem dostępnej drogi i odrzucaniem pakietów.

StaticRoutingProtocol— reprezentuje statyczny protokół rutingu.

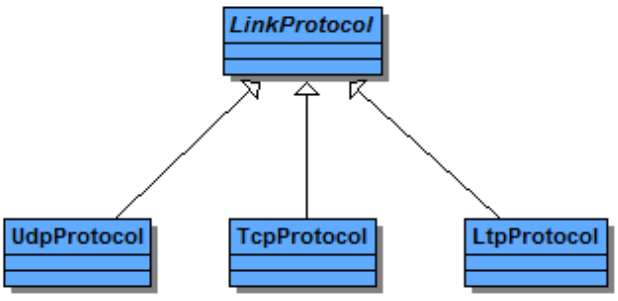
EpidemicRoutingProtocol — reprezentuje epidemiczny protokół rutingu.

GradientRoutingProtocol — reprezentuje gradientowy protokół rutingu.

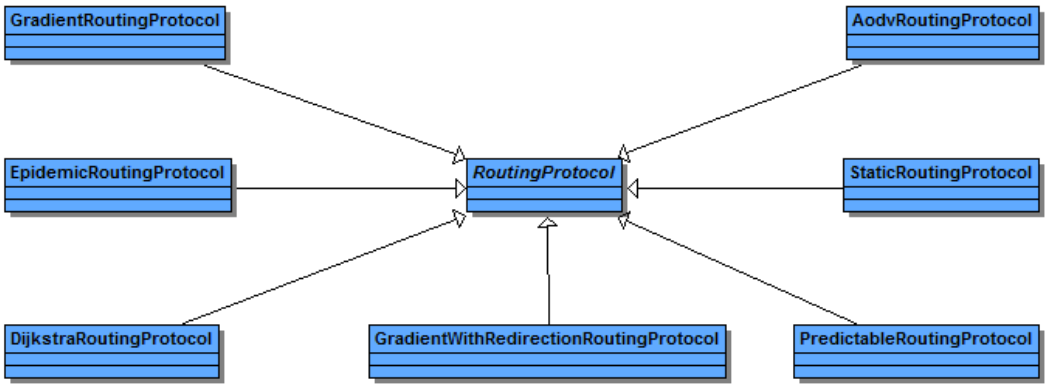
DijkstraRoutingProtocol — reprezentuje protokół rutingu oparty na protokole *OSPF*.

AodvRoutingProtocol — reprezentuje protokół rutingu oparty na protokole *AODV*.

PredictableContactRouting —reprezentuje protokół rutingu oparty na przewidywalności



Rysunek 3-5: Model protokołów transportowych wykorzystywanych w symulacji sieci DTN.



Rysunek 3-6: Model protokołów rutingu wykorzystywanych w symulacji sieci DTN.

kontaktów.

3.3.6.4 Generowanie statystyk

Ze względu na dużą liczbę elementów, które generują statystyki, pokazane zostały tylko przykładowe klasy dziedziczące po *StatisticsGenerator*. Rysunek 3-7 przedstawia model generowania statystyk w symulatorze sieci *DTN*.

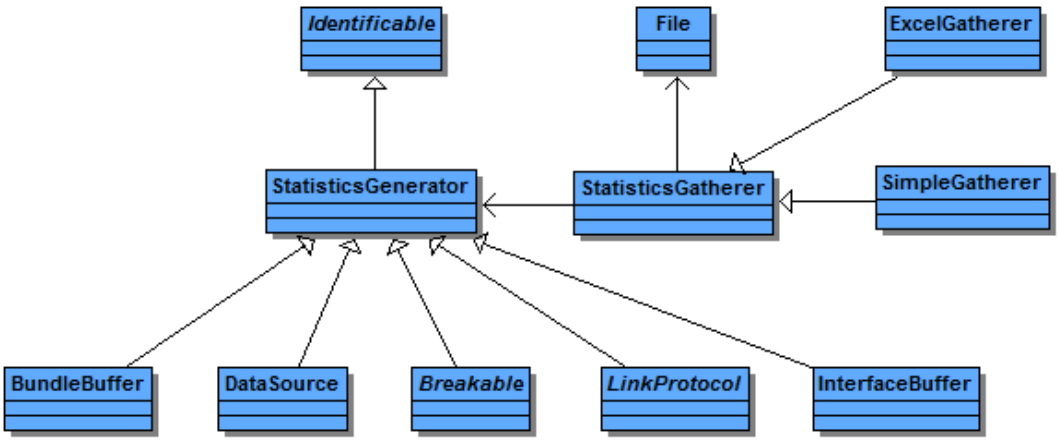
File — reprezentuje plik na dysku, do którego zapisywane są statystyki.

StatisticsGatherer —jest to element pobierający dane statystyczne zgodnie z konfiguracją podaną przez użytkownika.

SimpleGatherer — zapisuje statystyki do pliku w formie zrozumiałej dla człowieka.

ExcelGatherer — zapisuje statystyki do pliku w formacie, który umożliwia łatwe zaimportowanie danych do programu *Excel*.

BundleBuffer— generuje statystyki związane z liczbą przechowywanych pakietów *Bundle*,



Rysunek 3-7: Model generowania statystyk w sieci DTN. Nie wszystkie klasy dziedziczące po StatisticsGenerator zostały pokazane ze względu na ich zbyt dużą liczbę.

liczbą potwierdzonych pakietów, liczbą pakietów odrzuconych z powodu kilkukrotnego braku potwierdzenia itp.

DataSource — generuje statystyki o liczbie wygenerowanych danych.

Breakable — klasa bazowa dla elementów sieci *DTN*, które ulegają uszkodzeniom i przestojom. Generuje statystyki związane z awariami i przestojami elementu, który po niej dziedziczy.

LinkProtocol — protokół transportowy działający na łączu. Generuje statystyki związane z transportem pakietów przez pojedyncze łącze. W zależności od implementacji generuje także informacje specyficzne dla protokołu, np. rozmiar okna *TCP*.

Link — reprezentuje łącze w sieci *DTN*.

InterfaceBuffer — przechowuje paczki danych po zrutowaniu aż do momentu, kiedy określone łącze będzie wolne i będzie można wysłać przez nie paczkę danych. Generuje statystyki związane z zajętością bufora, liczbą odrzuconych paczek danych z powodu przepełnienia bufora itp.

3.3.6.5 Generatory losowe

Rysunek 3-8 przedstawia model generatorów losowych wykorzystywanych w symulacji.

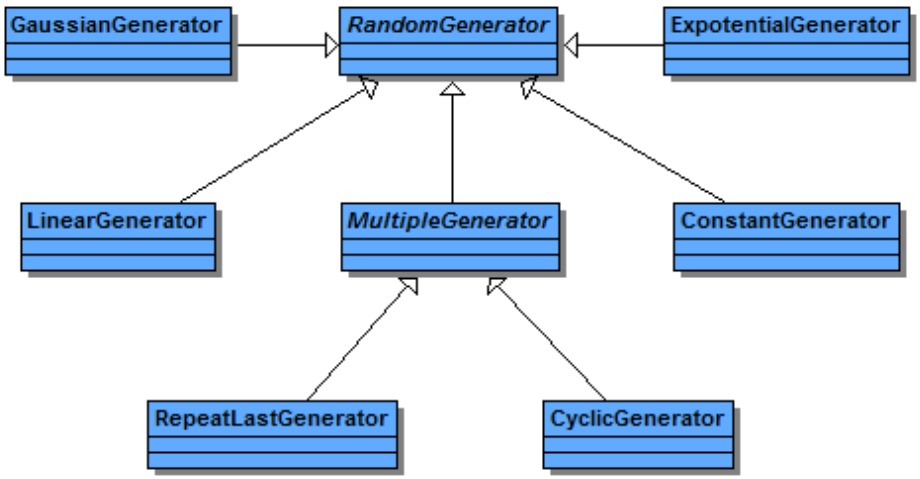
RandomGenerator — klasa bazowa dla generatorów losowych o różnych rozkładach prawdopodobieństwa.

GaussianGenerator — reprezentuje rozkład normalny.

ExpotentialGenerator —reprezentuje rozkład wykładniczy.

LinearGenerator — reprezentuje rozkład liniowy.

ConstantGenerator —reprezentuje rozkład stały (losowana jest zawsze ta sama wartość).



Rysunek 3-8: Model generatorów losowych wykorzystywanych w symulacji.

MultipleGenerator —klasa bazowa dla klas generujących wartości losowe w oparciu o wiele wewnętrznych generatorów losowych. Klasa ta przechowuje wewnętrznie kolekcję klas *RandomGenerator*, które są wykorzystywane do wygenerowania jednej wartości losowej.

RepeatLastGenerator — klasa ta zwraca wartość losową pochodzącą z kolejnych generatorów znajdujących się w kolekcji, aż dotrze do ostatniego generatora losowego. Następnie zwraca już tylko wartość losową pochodzącą z ostatniego generatora losowego.

CyclicGenerator —klasa ta zwraca wartość losową pochodzącą z kolejnych generatorów znajdujących się w wewnętrznej kolekcji. Po użyciu wartości z ostatniego generatora losowego, cykl zaczyna się ponownie od pierwszego generatora losowego w kolekcji.

4 Symulacja oraz analiza wyników

Niniejszy rozdział ma na celu zdefiniowanie interesujących nas zależności pomiędzy różnymi parametrami sieci, które będą mierzone w trakcie symulacji; zdefiniowanie konkretnych przypadków, które będą symulowane w sieci tolerującej przerwania i opóźnienia, a następnie — po wykonaniu odpowiednich symulacji — przedstawienie wyników w formie tabel i wykresów, co powinno umożliwić analizę zależności pomiędzy poszczególnymi parametrami sieci i wyciągnięcie podstawowych wniosków związanych ze skutecznością i zakresem stosowania poszczególnych protokołów, w tym najbardziej nas interesujących protokołów rutingu odpowiedzialnych za kierowanie przepływem danych w sieciach *DTN*.

4.1 Scenariusze symulacji

Ze względu na temat tej pracy najważniejszą poszukiwaną zależnością jest niezawodność dostarczania danych dla różnych protokołów w zależności od charakterystyki sieci (m.in. opóźnienia występujące w sieci, rozmiar sieci, ilość generowanych danych i częstość występowania niezdatności) oraz koszt uzyskania określonej niezawodności (mierzony m.in. w opóźnieniu dostarczenia danych i ilości nadmiarowych danych wysyłanych przez sieć). W szczególności interesujące jest porównanie wydajności protokołów transportowych w trudnych warunkach, które są typowe dla sieci *DTN* oraz porównanie efektywności protokołów routingu w konfiguracjach sieci, które występują w rzeczywistych sieciach, które są przystosowane do unikalnego środowiska, w którym muszą dobrze funkcjonować.

Ten rozdział ma za zadanie tylko zdefiniowanie scenariuszy, które następnie będą symulowane, a ich wyniki przedstawione w kolejnych rozdziałach. Należy zwrócić uwagę, że na tym etapie nie zawsze możliwe jest określenie stałych użytych w symulacji. Np. czas symulacji da się określić dopiero po przeprowadzeniu kilku testów, które pozwolą określić taką wartość czasu symulacji, żeby czas działania symulatora zmieścił się w granicach akceptowalnych dla osoby przeprowadzającej symulację. W przeciwnym razie mogłoby się okazać, że zgodnie z przyjętymi założeniami symulacja musi trwać np. rok. Stałe, które będą ustalane empirycznie, przedstawione zostaną w kolejnych rozdziałach razem z wynikami symulacji, natomiast celem tego rozdziału jest ustalenie relacji między stałymi (np. czas propagacji łącza dużo mniejszy od czasu do awarii łącza).

4.1.1 Protokoły transportowe

W sieci *DTN* występują następujące zjawiska, z którymi muszą sobie radzić protokoły transportowe:

- przerwania łączności,
- duży czas propagacji,
- przekłamania w transportowanych danych.

4.1.1.1 Przerwania łączności

Przerwania łączności dla niektórych protokołów (np. dla *TCP*) powodują zmniejszenie szybkości transportowanych danych. Duże znaczenie ma tutaj częstotliwość występowania przerwań oraz czas trwania przerwy w łączności. Np. *TCP* zmniejsza rozmiar okna dwukrotnie po każdym stwierdzeniu, że nie nadeszło potwierdzenie odebrania wysłanych danych. Im dłuższa przerwa w czasie działania łącza, tym bardziej zmniejszy się szybkość działania tego protokołu.

Sieć badana w tym scenariuszu składa się z dwóch węzłów oraz jednego łącza pomiędzy nimi (rysunek 4-1). Pierwszy węzeł generuje taką ilość danych, żeby w 100% zająć dostępne łącze. Łącze ulega awariom oraz naprawom zgodnie z rozkładem wykładniczym, przy czym czas dostępności łącza równy jest czasowi jego niedostępności.

W takim przypadku każdy protokół transportowy może uzyskać średnią przepływność dostarczania danych na poziomie najwyżej 50% przepływności łącza. Od tej przepływności należy jeszcze odjąć narzut związany z warstwami łącza oraz sieciową. Celem scenariusza jest sprawdzenie, jaką rzeczywistą efektywność mają poszczególne protokoły rutingu, tj. jaką ilość danych są w stanie dostarczyć do węzła po drugiej stronie łącza w zadanym czasie w zależności od częstotliwości występowania przestojów łącza.

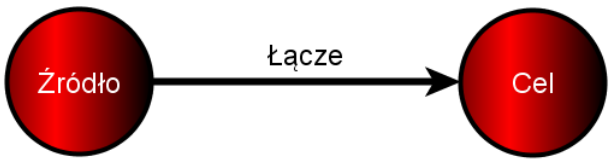
4.1.1.2 Duży czas propagacji

Duży czas propagacji może znacznie wpływać na szybkość transportu danych na łączu.

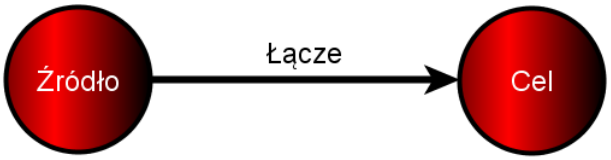
W scenariuszu tym badana sieć składa się z dwóch węzłów oraz jednego łącza znajdującego się między węzłami (rysunek 4-2). Pierwszy węzeł generuje taką ilość danych, żeby w 100% zająć dostępne łącze. Celem scenariusza jest zbadanie ilości danych dostarczonych do węzła drugiego w zależności od czasu propagacji łącza.

4.1.1.3 Przekłamania w transportowanych danych

Jednym z podstawowych zadań protokołów transportowych jest zapewnianie niezawodności dostarczonych danych (*TCP* i *LTP*) lub wykrywanie, że dane zostały uszkodzone (*UDP*), co pozwala na ich odrzucenie. Zadaniem tego scenariusza jest przetestowanie wpływy zakłóceń na efektywność dostarczania danych, tj. wpływu współczynnika *BER* na ilość dostarczonych



Rysunek 4-1: Struktura sieci podczas badania odporności protokołów transportowych na przerwy w łączności.



Rysunek 4-2: Struktura sieci podczas badania wpływu dużego czasu propagacji na efektywność protokołów transportowych.

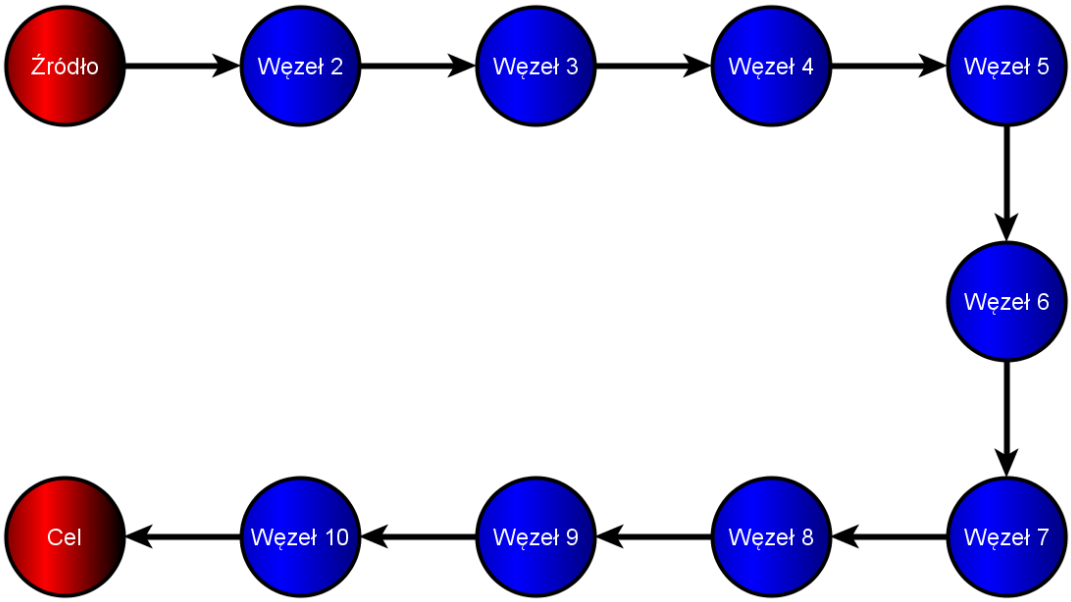
danych.

Sieć służąca do przetestowania tego zjawiska składa się z 10 łączy oraz jedenastu węzłów, które tworzą jedną linię (rysunek 4-3). Pierwszy węzeł generuje taką ilość danych, żeby w pełni obciążyć łącze i wysyła je do ostatniego węzła. Na każdym łączu część danych ulega przekłamaniom zgodnie z zadanym dla symulacji współczynnikiem *BER*. W przypadku protokołów *TCP* i *LTP* dane te są retransmitowane, w przypadku protokołu *UDP* dane są odrzucane. Normalnie dla *UDP* niezawodność dostarczenia danych zostałaby osiągnięta na poziomie warstwy protokołu *Bundle*, jednak w tej symulacji retransmisje zostają wyłączone, ponieważ interesują nas wyłącznie protokoły transportowe.

4.1.2 Protokoły rutingu

Nie istnieje jeden protokół rutingu, który sprawdzałby się w każdej możliwej sieci *DTN*. Dlatego celem przeprowadzenia symulacji jest ustalenie, który z protokołów rutingu najlepiej sprawdza się w określonym środowisku. Szczególnie interesującymi dla nas przypadkami są sieci, które albo już funkcjonują, albo są rozważane jako potencjalne rozwiązania.

Różnice pomiędzy poniższymi scenariuszami sprowadzają się różnic w następujących czynnikach:



Rysunek 4-3: Struktura sieci podczas badania wpływu współczynnika BER na efektywność protokołów transportowych.

- ilość generowanych danych,
- używany protokół transportowy,
- awarie,
- przestoje,
- retransmisje ,
- czas życia pakietów *Bundle*,
- czas propagacji łączy,
- badane parametry,
- struktura sieci.

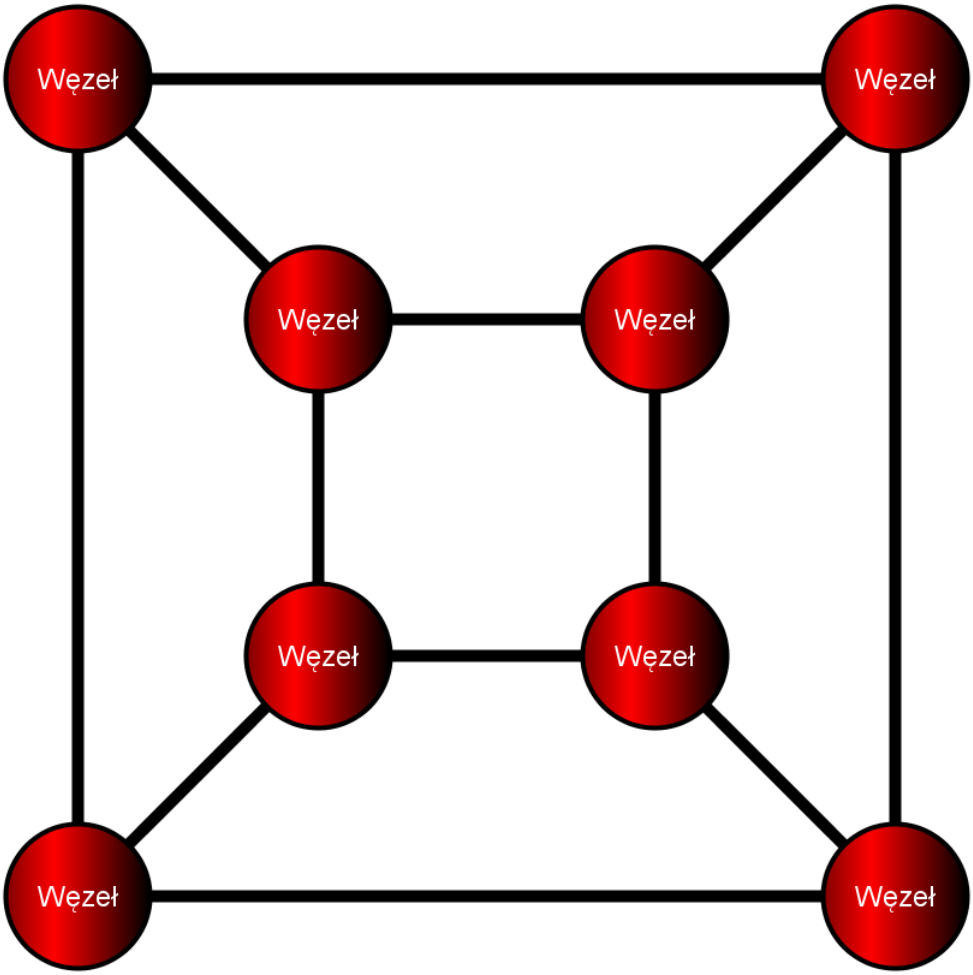
4.1.2.1 Tradycyjna sieć internetowa

Jako punkt odniesienia warto rozpocząć od sieci, której charakterystyka pokrywa się z najpopularniejszą obecnie siecią internetową. Sieci takie cechują się małymi opóźnieniami, rzadkimi awariami, brakiem przewidywalnych wyłączeń oraz bardzo dużym obciążeniem, sięgającym granic możliwości sieci. Na potrzeby symulacji obciążenie sieci zostanie ustawione na poziomie ok. 50% teoretycznej możliwości sieci (uwzględniając tylko przepływności łączy). Miernikiem efektywności protokołu jest przede wszystkim fakt dostarczenia danych, a w drugiej kolejności mały narzut transmitowanych informacji oraz małe opóźnienie dostarczania danych. W sieci występują retransmisje w warstwie protokołu *Bundle* (trzykrotne), przy czym czas retransmisji jest większy niż najdłuższa możliwa droga paczki danych od węzła początkowego do docelowego oraz czas dostarczenia potwierdzenia. Czas życia równy jest czasowy retransmisji. Protokołem transportowym jest protokół *TCP*. Dla pełnej szybkości działania protokół *TCP* potrzebuje bufora wyjściowego łącza o rozmiarze kilka razy większym od rozmiaru maksymalnego okna, jednak dla większego podobieństwa do sieci internetowej zostanie on ustawiony na poziomie 0,5 MB (przy rozmiarze maksymalnego okna 1 MB). Nie powinno to mieć większego wpływu na *TCP* z powodu bardzo małych opóźnień występujących w sieci.

Struktura sieci oparta jest na sześcianie (rysunek 4-4). Sześcian jest wyjątkowo korzystnym kształtem sieci, który zapewnia symetrię, a jednocześnie gwarantuje, że każdy węzeł będzie mieć co najmniej trzy drogi łączące go z pozostałą częścią sieci. Każdy węzeł generuje ruch z równym prawdopodobieństwem do każdego węzła w sieci.

4.1.2.2 Sieć sensorowa

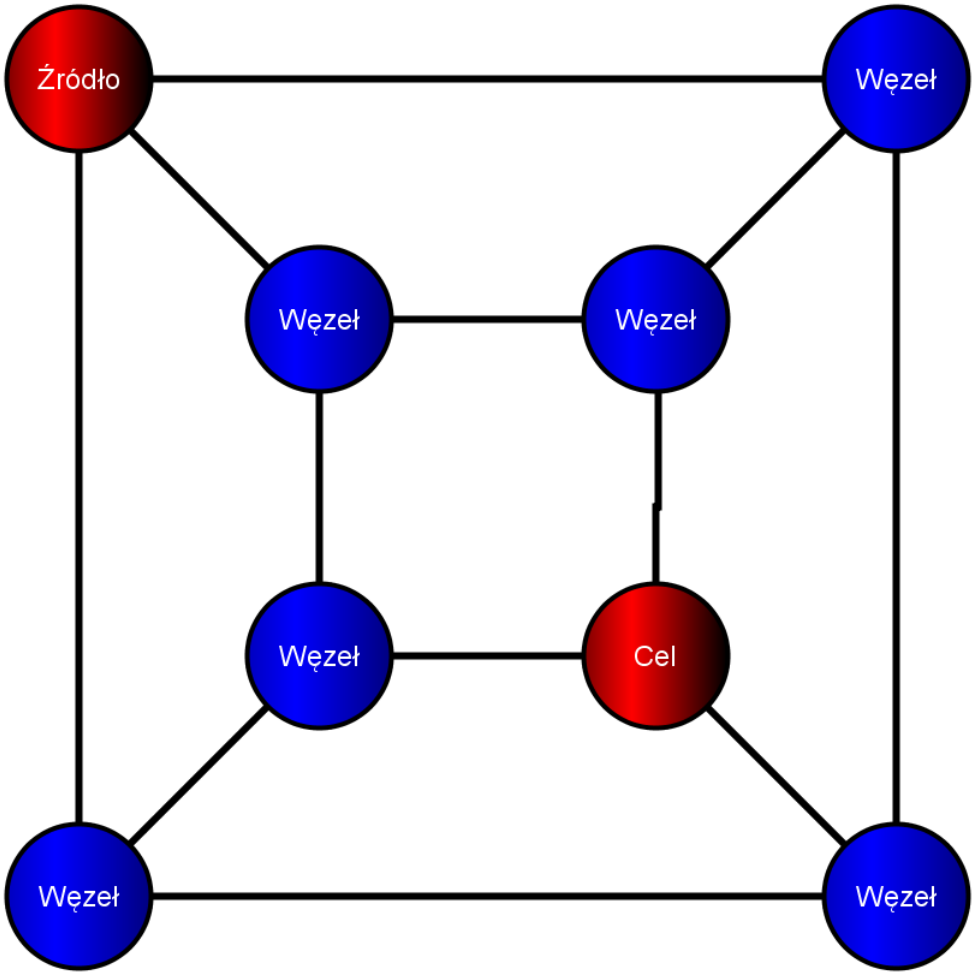
Sieci sensorowe charakteryzują się bardzo małą ilością transportowanych danych. Dane najczęściej generowane są w średnich porcjach, natomiast ich generacja następuje bardzo rzadko. Innymi cechami są relatywnie małe opóźnienia na łączach oraz bardzo duża liczba awarii węzłów. Parametr *TTB* łącza (ang. *Time To Break* — czas do awarii) jest dużo większy niż średni odstęp czasowy między wygenerowaniem danych. Przestoje nie występują. Liczba retransmisji zostaje ustalona na trzy, zaś czas życia paczki danych równy jest czasowi retransmisji. Największym problemem w sieci jest duży koszt wysyłania informacji (węzły mają ograniczone zasoby energetyczne) i to ten czynnik będzie wykorzystywany do porównania efektywności protokołów. W sieciach sensorowych tradycyjnie stosuje się protokół *AODV*, który wyszukuje drogę do celu dopiero w momencie wygenerowania nowych danych. Jest to okupione dużymi opóźnieniami w dostarczaniu danych ale sieci sensorowe nie mają wymagań



Rysunek 4-4: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci Internet.

związanych z małymi opóźnieniami. Natomiast duże znaczenie w sieci ma niezawodność dostarczenia danych oraz łączna liczba bitów transmitowanych w sieci (dlatego w rzeczywistych sieciach sensorowych nie sprawdzają się protokoły rutingu, które utrzymują w sposób ciągły wiedzę nt. struktury sieci wymieniając się przy tym informacjami o zmianach struktury). Wykorzystywanym protokołem transportowym jest *LTP*.

Przyjęta struktura sieci ponownie jest oparta na sześcianie (rysunek 4-5). Cała komunikacja występuje między dwoma węzłami znajdującymi się w przeciwnych rogach sześcianu. Taka struktura sieci daje 3 łącza wychodzące z każdego węzła. Najbliższe połączenie do węzła docelowego ma metrykę równą potrójnej metryce jednego łącza. Istnieje kilka równoległych dróg, które mogą być wykorzystane do przesłania pakietu do węzła docelowego. Protokół rutingu powinien wybrać taką, która zapewni paczce dotarcie do celu z dużym prawdopodobieństwem. Węzeł początkowy i docelowy nie ulegają awarii, aby zapewnić poprawne zbieranie statystyk i próby retransmisji. Pozostałe węzły ulegają awariom zgodnie z wykładniczym rozkładem prawdopodobieństwa. Wpływ danych wysyłanych przez inne węzły jest pomijalny i dlatego zostaje zupełnie zignorowany. W sieci jedynym elementem generującym dane jest węzeł źródłowy, a jedynym je odbierającym jest węzeł docelowy. Protokołem wykorzystywanym do transportu danych jest *UDP*, ponieważ przy małym czasie



Rysunek 4-5: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci sensorowej.

propagacji i braku przekłamań powinien on zagwarantować dostarczanie danych najmniejszym kosztem.

4.1.2.3 Sieci z mułami danych

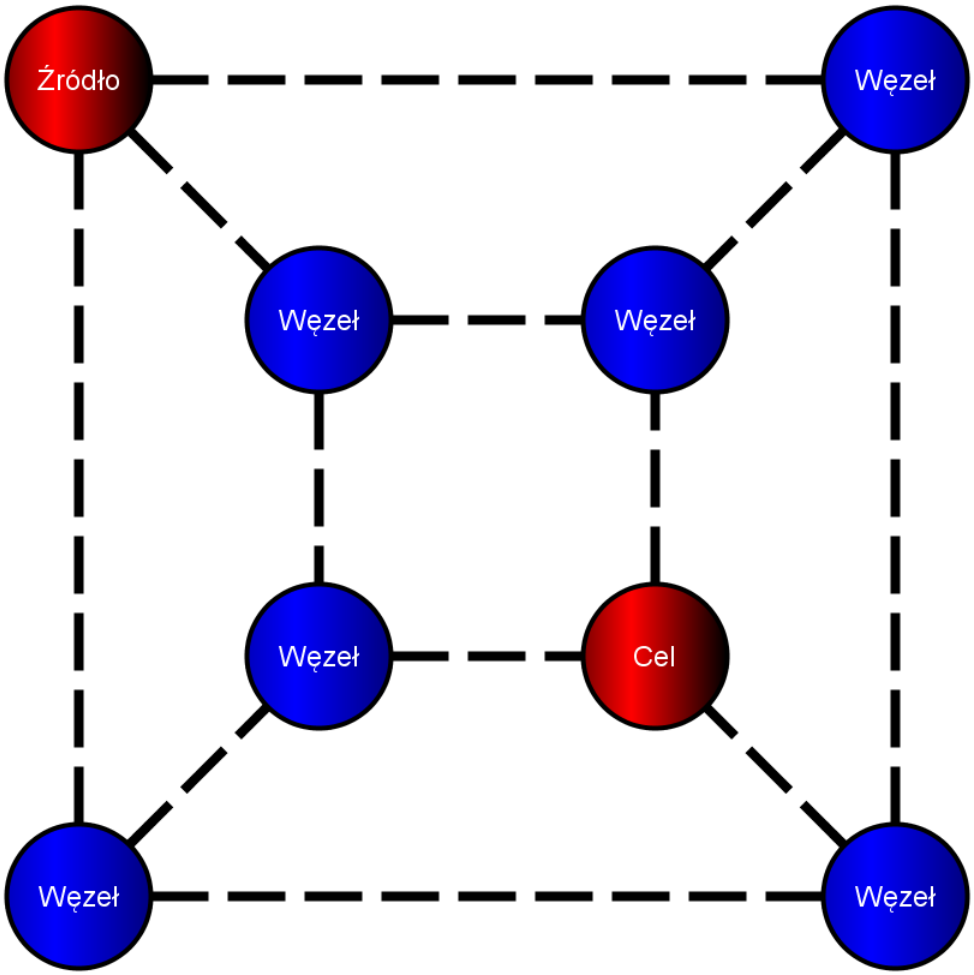
W sieciach opartych na mułach danych występują ruchome węzły, które uzyskują kontakt z pozostałymi węzłami co pewien okres czasu. Okres ten jest przewidywalny i dlatego dobrym sposobem symulacji są przestoje łączy między węzłami. Każde łącze przez większość czasu jest wyłączone. Średni okres włączenia każdego łącza jest dużo mniejszy od średniego okresu wyłączenia. Nie występują natomiast uszkodzenia łączy i węzłów ani przestoje węzłów. Włączenia i wyłączenia łączy następują zgodnie z rozkładem wykładniczym. W takiej sieci dane generowane są rzadko i w małych ilościach, zaś połączenie między węzłami ma najczęściej postać połączenia radiowego na krótką odległość, co gwarantuje minimalne opóźnienia. Ponieważ w sieci nie występują ani awarie elementów, ani przekłamania bitów w czasie transmisji, dlatego nie są potrzebne retransmisje ani na poziomie protokołu transportowego, ani na poziomie protokołu *Bundle*. Jako protokół transportowy wykorzystać można *UDP*. Czas życia paczki danych można uznać za nieskończony, tj. żadna paczka danych nie jest odrzucana przed dostarczeniem jej do celu. Jedyną sytuacją, w której mogłoby nastąpić odrzucenie paczki, jest przepełnienie bufora wyjściowego łącza i dlatego jego rozmiar jest ustawiony na wartość

praktycznie nieskończoną. Przy takich warunkach istnieje potencjalnie tylko jedna możliwość niedostarczenia paczki z danymi – wadliwe działanie protokołu rutingu, który w określonej sieci nie jest w stanie odnaleźć drogi do celu. Odsetek dostarczonych danych jest najważniejszym czynnikiem badanym w sieci. Inne istotne czynniki to czas dotarcia wiadomości do celu oraz dodatkowy narzut informacji związany ze zwielokrotnianiem danych lub informacjami o rutingu.

Struktura sieci ponownie opiera się na sześcianie (rysunek 4-6). Sieć z mułami danych jest symulowana w sposób, który uniemożliwia wskazanie w strukturze sieci mułów danych. Wpływ danych wysyłanych przez inne węzły jest pomijalny i dlatego zostaje zupełnie zignorowany. W sieci jedynym elementem generującym dane jest węzeł źródłowy, a jedynym je odbierającym jest węzeł docelowy.

4.1.2.4 Sieci z bardzo rzadkimi i jednocześnie nieprzewidywalnymi kontaktami między węzłami

Sieć *Zebranet* jest siecią wyjątkowa pod tym względem, że łączność między węzłami występuje w całkowicie losowych i nieprzewidywalnych momentach. Podobna sytuacja występuje w niektórych innych sieciach. Dobrym sposobem symulacji tego rodzaju zachowania są awarie

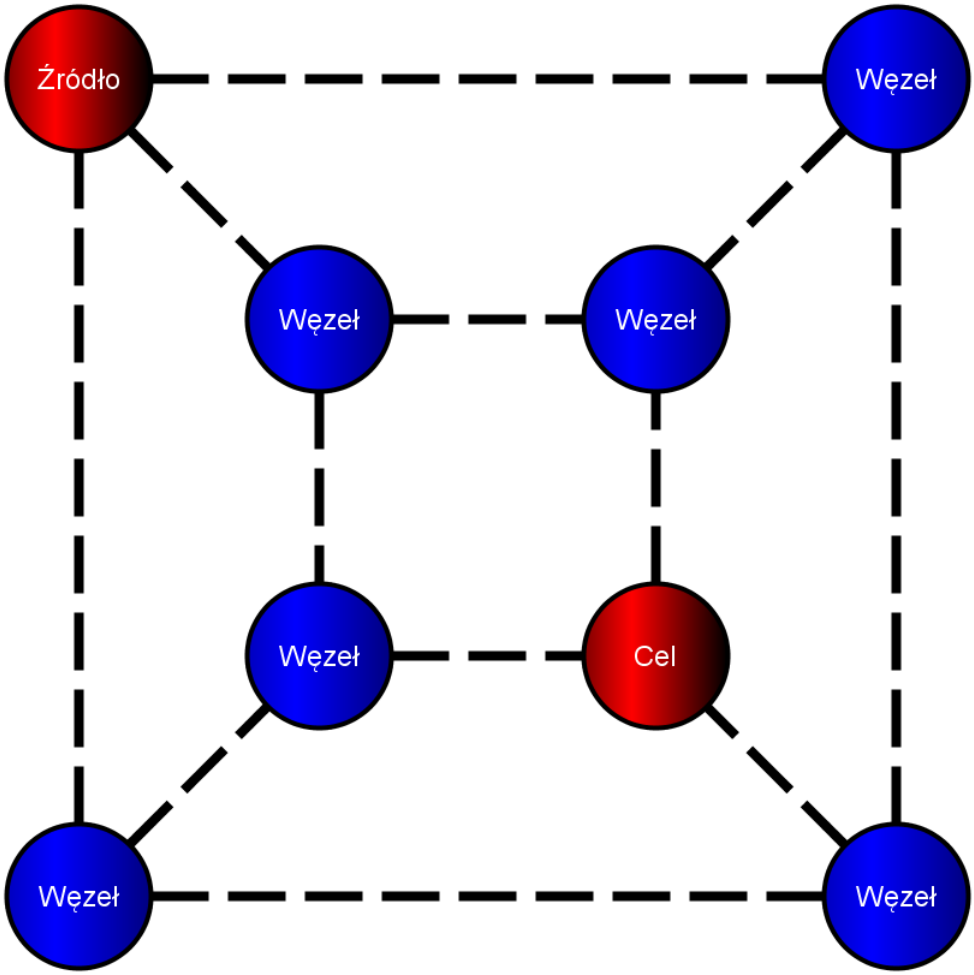


Rysunek 4-6: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci z mułami danych. Linie przerywane oznaczają łącza, które przez większość czasu są w stanie przestoju.

łączy przy jednoczesnym braku przestojów i awarii węzłów, przy czym średni okres awarii jest dużo większy od średniego okresu poprawnego działania. W sieci bardzo rzadko występuje bezpośredni kontakt między węzłami. Jego brak może być dla wielu protokołów problemem, dlatego głównym badanym czynnikiem jest odsetek danych, które dotarły do celu. Innym istotnym czynnikiem jest czas dotarcia wiadomości do celu. Nie jest natomiast istotny koszt transportu informacji, ponieważ z założenia transport odbywa się na bardzo małe dystanse, zaś ilość transmitowanych danych jest bardzo mała.

Protokołem wykorzystywanym do transportu danych w sieci jest *UDP*. Liczba retransmisji zostaje ustalona na wartość trzech. Czas życia może zostać ustawiony na wartość praktycznie nieskończoną, ponieważ z założenia nie da się przewidzieć momentu pojawienia się kontaktu między węzłami i pakiety *Bundle* będą musiały czekać długi okres czasu.

Do symulacji wykorzystana będzie sieć o strukturze sześcianu, przy czym tylko dwa najodleglejsze od siebie węzły będą brać udział w generowaniu i odbieraniu danych (rysunek 4-7).



Rysunek 4-7: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci z rzadkimi i nieprzewidywalnymi kontaktami między węzłami. Przerywane linie reprezentują łącza, które przez większość czasu są w stanie awarii.

4.1.2.5 Sieci międzyplanetarne

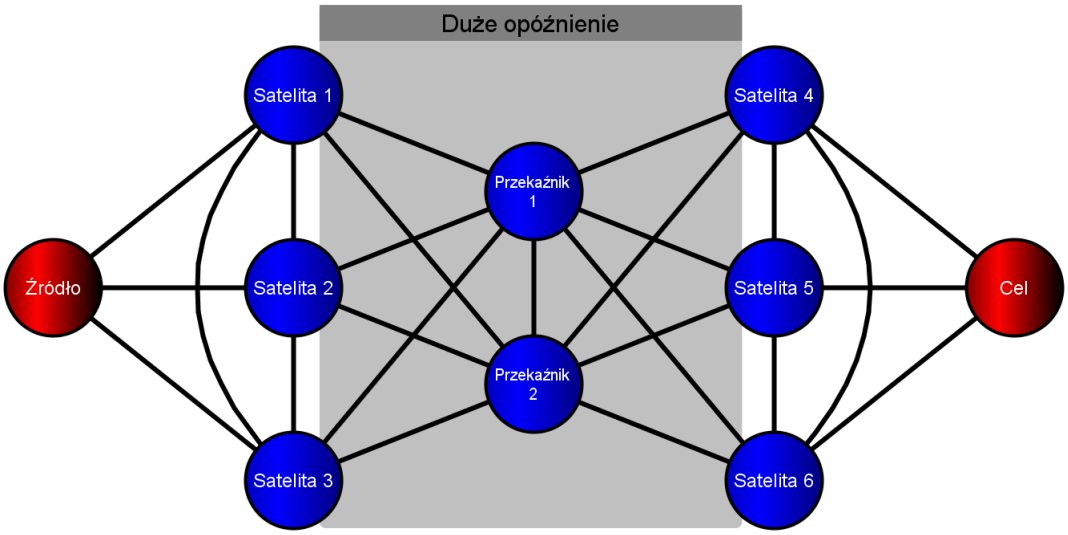
Główną cechą sieci międzyplanetarnych jest bardzo duże opóźnienie występujące w sieci. Drugą cechą jest bardzo duży koszt transportu danych np. przez satelity, których jedynym źródłem zasilania są baterie słoneczne. Ilość energii wykorzystanej na transmisję danych jest głównym czynnikiem określający efektywność protokołu rutingu. Dla prostoty można przyjąć, że ilość transmitowanych danych jest proporcjonalna do ilości energii zużytej przez węzeł na transmisję. Opóźnienia związane z transportem nie są czynnikiem krytycznym. Naukowcy analizujące dane otrzymane z innych części kosmosu są skłonni poczekać trochę dłużej na uzyskanie rezultatów ich badań. Czas transportu danych jest i tak dużo mniejszy od czasu umieszczenia poszczególnych elementów systemu badawczego w kosmosie.

Z racji bardzo dużych opóźnień występujących na łączach stosowanymi protokołami transportowymi mogą być tylko *UDP* oraz *LTP*, przy czym *LTP* powinien lepiej spełniać swoje zadanie chociażby z tego powodu, że był projektowany w celu usprawnienia komunikacji w kosmosie. *LTP* gwarantuje niezawodność dostarczania danych, natomiast nie gwarantuje, że węzeł, który ulegnie uszkodzeniu, nie straci tych danych. Dlatego w sieci stosowane będą wielokrotne retransmisje, które będą kontynuowane aż do końca symulacji. Czas życia pojedynczego pakietu zostanie ustawiony na wartość praktycznie nieskończoną, dzięki czemu dane nigdy nie zostaną utracone z powodu jego upłynięcia.

Średnia ilość danych generowanych w jednostce czasu jest dużo mniejsza niż przepływność łączy występujących w sieci.

Sieć składa się z jednego niezawodnego nadajnika danych znajdującego się na odległej planecie, trzech satelitów, które mogą przekazać dane dalej w kosmos, dwóch stacji bazowych, które mogą być wykorzystane do wzmocnienia sygnału po drodze oraz trzech odbiorników w postaci satelitów, które mogą dostarczać dane na Ziemię, do niezawodnego odbiornika jakim jest ośrodek badawczy (rysunek 4-8). Czasy propagacji na łączach pomiędzy satelitą a obiektami naziemnymi są relatywnie małe, natomiast bardzo duże są czasy propagacji na łączach pomiędzy satelitami a stacjami przekazującymi dane do Ziemi. Oczywiście protokoły rutingu można w tej sytuacji przetestować dopiero po wprowadzeniu awarii i wyłączeń elementów. Najciekawszym elementem wydaje się zdolność protokołów rutingu do radzenia sobie z wybieraniem optymalnej drogi przy nieprzewidywalnych awariach oraz bardzo dużych opóźnieniach, aczkolwiek w sieci występują także przewidywalne wyłączenia poszczególnych elementów.

Satelita zazwyczaj krąży nad planetą, dlatego łączność z nim jest przerywana w sposób przewidywalny. Dobrym sposobem symulacji są przestoje łączy z satelitą. W danym momencie działa łącze tylko do jednego spośród trzech satelitów (dzieje się to na zmianę co pewien okres). Łącza z satelitami nie ulegają awariom, satelity nie ulegają ani awariom, ani przestojom. Natomiast nieprzewidywalne awarie występują na łączach pomiędzy satelitami a przekaźnikami, co umożliwia sprawdzenie odporności protokołów rutingu na awarie łączy. Każde łącze jest dostępne przez ok. 80% czasu, przy czym awarie i naprawy występują zgodnie z rozkładem wykładniczym. Okres awarii jest kilkakrotnie większy od czasu propagacji łącza. Węzły pracujące jako przekaźniki są węzłami niezawodnymi.



Rysunek 4-8: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci międzyplanetarnej.

4.1.2.6 Sieć z dużymi opóźnieniami

Celem tego rozdziału jest zbadanie wyidealizowanej sieci, w której występują bardzo duże opóźnienia, ale praktycznie nie pojawiają się żadne awarie ani przestoje. Ilość generowanych danych jest mała w stosunku do przepływności łączy. Cechą charakterystyczną sieci z dużymi opóźnieniami jest czas propagacji na łączach dużo większy od średniego okresu pomiędzy generacją nowych danych. Węzły generują ruch losowo do dowolnego innego węzła w sieci. Przy takich warunkach można mieć praktycznie pewność, że wiadomość dotrze do celu. Pytanie brzmi: jaki będzie koszt dotarcia danej wiadomości do węzła docelowego, tj. jaki będzie narzut wiadomości wysyłanych przez protokoły rutingu oraz jak długo wiadomość będzie zajmować zasoby sieci (tj. jaki będzie jej czas dotarcia do celu). Dla urozmaicenia scenariusza i dla większej ogólności testów, badana sieć ma budowę asymetryczną (rysunek 4-9), z różnymi wartościami opóźnienia przypisanymi do poszczególnych łączy.

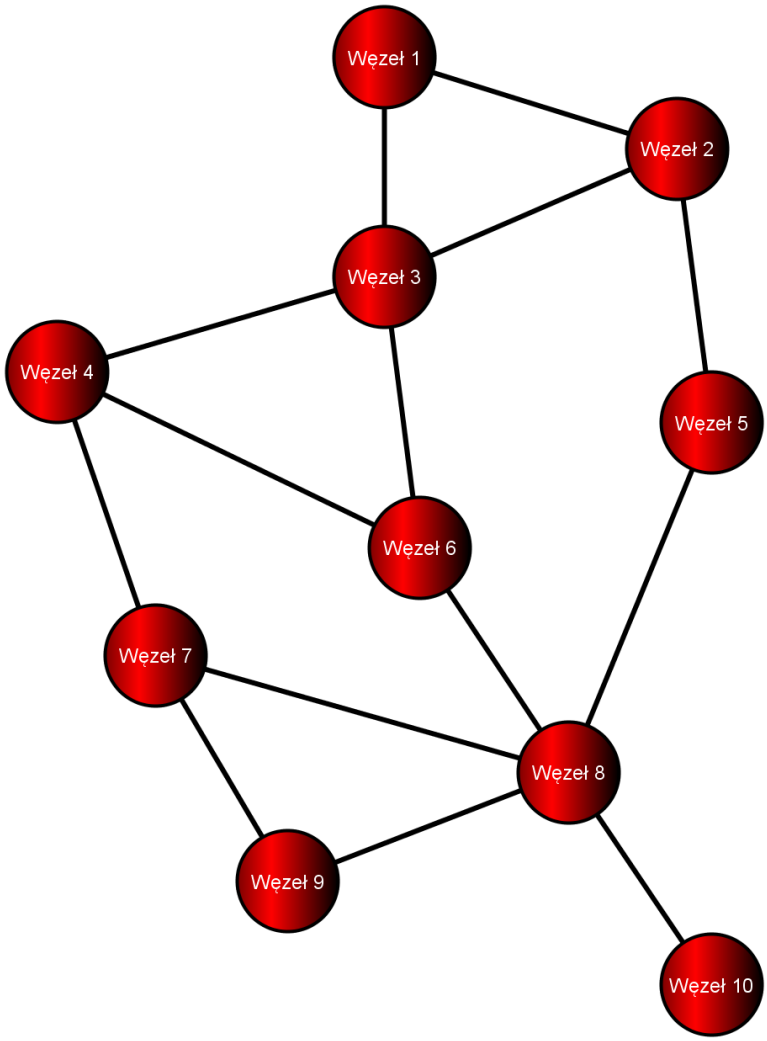
Średnia ilość wygenerowanych danych w jednostce czasu będzie mniejsza od przepływności łączy, co pozwoli uniknąć odrzucania danych przy przepełnieniu bufora (jego rozmiar i tak będzie ustawiony na wartość praktycznie nieskończoną). Z powodu braku potrzeby retransmisji danych na łączu używanym protokołem będzie *UDP*.

W sieci nie są potrzebne retransmisje, zaś czas życia pojedynczej wiadomości może być nieskończony.

4.1.3 Wpływ przekazywania odpowiedzialności za retransmisję na wydajność sieci

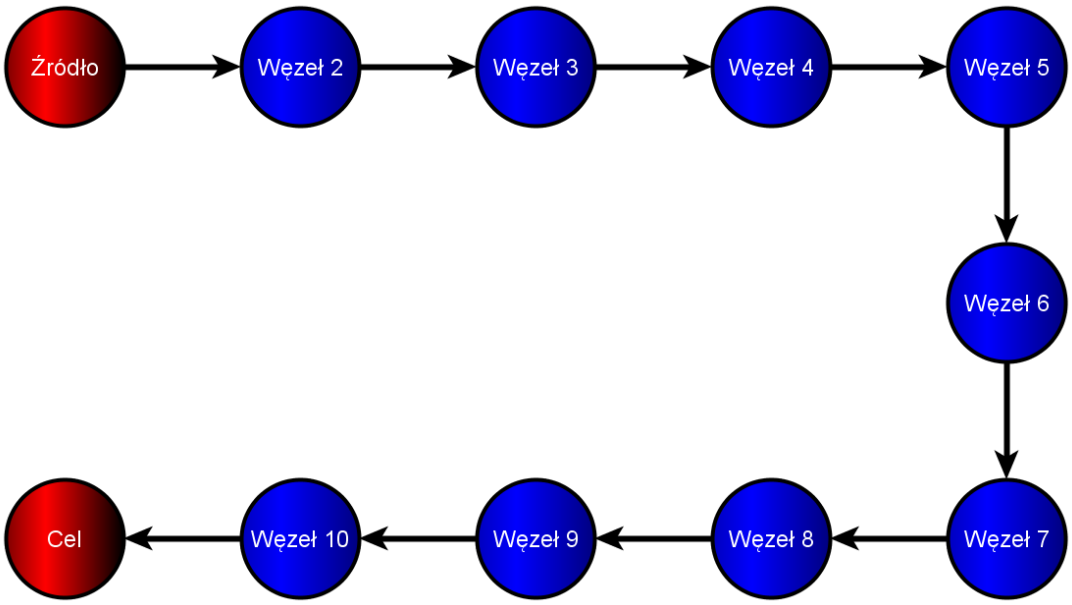
Zadaniem niniejszego scenariusza jest porównanie wydajności sieci, w której używanym protokołem transportowym jest *UDP* oraz występują przekłamania bitów w czasie transmisji danych w zależności od tego, czy węzły przejmują odpowiedzialność za retransmisję danych.

Struktura sieci jest identyczna z odpowiadającym scenariuszem, w którym badana była odporność protokołów *UDP*, *LTP* i *TCP* na przekłamania bitów. Jedenaście węzłów połączonych



Rysunek 4-9: Struktura sieci dla scenariusza badającego efektywność protokołów rutingu w sieci o dużych czasach propagacji sygnału w łączach.

jest dziesięcioma łączami (rysunek 4-10). Protokołem transportowym jest *UDP*. Symulacja przeprowadzana jest w dwóch wariantach. W pierwszym wariancie dane transmitowane są zwyczajnie przez sieć, tj. węzeł początkowy wysyła dane do węzła docelowego i oczekuje na potwierdzenie odebrania pakietu *Bundle* od tego węzła. W przypadku odrzucenia pakietu z powodu przekłamania bitów retransmisja następuje z pierwszego węzła. W drugiej wersji każdy węzeł przejmuje odpowiedzialność za retransmisję danych. Jeżeli konieczna będzie retransmisja, to nastąpi ona zawsze z ostatniego węzła, do którego dotarł pakiet. Ponieważ badany jest wpływ przekazywania odpowiedzialności za retransmisję danych, więc w sieci muszą występować retransmisje. Jest to zasadnicza różnica w porównaniu ze scenariuszem, w którym była badana odporność protokołów transportowych na przekłamania bitów. Liczba retransmisji powinna być praktycznie nieskończona, tj. każdy pakiet powinien być transmitowany do skutku. Przy takim sposobie działania sieci 100% danych powinno być dostarczonych do węzła docelowego. Natomiast dwa warianty będą najprawdopodobniej z różną częstością retransmitować dane. W celu porównania wydajności wariantów mierzona będzie liczba transmitowanych danych potrzebnych do dostarczenia danych do celu.



Rysunek 4-10: Struktura sieci dla scenariusza badającego wpływ przejmowania odpowiedzialności za retransmisję przy gubieniu pakietów.

4.2 Analiza wyników

W obliczeniach wykorzystywana była ufność *Θ* cechy *X* mającej w populacji rozkład normalny o wartości oczekiwanej m, która zdefiniowana jest następująco:

$$P(\bar{X} - \theta < m < \bar{X} + \theta) = 95\%$$

4.2.1 Protokoły transportowe

Próby na kilku sieciach pokazały, że optymalny czas symulacji wynosi mniej więcej 10^5 s przy szybkości łącza 10^5 B/s. Przy takiej konfiguracji czas przeprowadzenia jednej symulacji trwa ok. 3 min. Dłuższy czas symulacji nie pozwalał na przeprowadzenia dużej liczby symulacji w dostępnym czasie, natomiast jej skrócenie powodowało zmniejszenie wiarygodności uzyskiwanych wyników. Kilka prób pokazało także znikomy wpływ warunków początkowych na rezultaty symulacji. Tzn. podczas symulacji działania protokołów transportowych i zadanym czasie symulacji niepotrzebny okazał się czas na rozbieg dla sieci, aby uzyskać wiarygodne wyniki. Symulacja trwa na tyle długo, że wpływ warunków początkowych był pomijalny.

Symulacja retransmisji pakietów protokołu *Bundle* była zupełnie niepotrzebna w zadanym scenariuszu i została wyłączona. Każda paczka danych po zrutowaniu trafiała do bufora wyjściowego danego łącza. Jeśli bufor był przepełniony, to paczka danych była odrzucana. Jeśli natomiast był wolny, to paczka trafiała na koniec bufora i oczekiwała na swoją kolej przekazania do protokołu transportowego. Ilość generowanych danych przekraczała możliwości łącza, co gwarantowało ciągłe zapełnienie bufora wyjściowego i łatwość oceny protokołu na podstawie liczby dostarczonych bajtów w czasie symulacji.

Ze względu na sposób symulacji protokołu *TCP*, konieczne było ustawienie rozmiaru bufora wyjściowego łącza na poziomie 8 MB. Spowodowane było to maksymalnym rozmiarem okna *TCP*, który w symulacji został ustawiony na 1 MB. Duży bufor wyjściowy gwarantował płynność

przekazywania danych z bufora wyjściowego do warstwy *TCP*, gdzie dane były wysyłane w oknie.

W każdym z badanych scenariuszy istniała tylko jedna możliwa droga przekazywania danych. Dlatego też stosowanymi protokołami były zawsze albo statyczny, albo gradientowy. Oba dawały zawsze identyczne rezultaty.

4.2.1.1 Przerwania łączności

Czas symulacji wynosił 10^5 s. Czas propagacji łącza był stały i wynosił 1 s. Łącze ulegało awarii i było naprawiane z rozkładem wykładniczym i z wartością oczekiwaną z przedziału od 2,5 s do 1280 s zmienną dla każdej z symulacji. Przedział ten został wyznaczony eksperymentalnie.

Tabela 4-1 przedstawia liczbę dostarczonych bajtów w czasie symulacji. Tabela 4-2 przedstawia średnie opóźnienie dostarczenia paczki danych, które jest liczone od momentu wygenerowania paczki danych do momentu jej dotarcia do drugiego węzła. Rysunek 4-11 przedstawia graficznie zależność pomiędzy wartością oczekiwaną czasu trwania awarii łącza a dostarczonymi danymi. Rysunek 4-12 przedstawia graficznie zależność pomiędzy wartością oczekiwaną czasu trwania awarii a średnim opóźnieniem.

Wykresy pokazują, że protokoły *UDP* oraz *LTP* są tylko w bardzo niewielkim stopniu zależne od częstotliwości awarii łącza. Tak naprawdę na wyniki wpływ mógł mieć sposób symulacji awarii łącza, który zakłada, że awaria powoduje anulowanie transportu wszystkich danych transmitowanych w łączu (podczas gdy awaria łącza występuje zazwyczaj w pojedynczym miejscu, a tym samym wszystkie dane, które znajdują się w łączu poza tym miejscem, docierają do celu). Można się spodziewać, że wpływ częstotliwości awarii na działanie protokołów *LTP* i *UDP* jest w rzeczywistości jeszcze mniejszy dla małych wartości oczekiwanych czasu trwania awarii (w symulowanym przedziale 2,5-10 s) niż to wynika z uzyskanych pomiarów. W pozostałym przedziale oba protokoły przetransportowały blisko 50% przekazanych im danych, a więc znalazły się blisko maksymalnej teoretycznej przepływności możliwej do uzyskania w tym scenariuszu. Wskazuje to na efektywne wykorzystanie łącza, a więc oba protokoły sprawdzają się przy częstych przerwaniach łączności. Jednocześnie opóźnienie w dostarczaniu

Tabela 4-1: Dostarczone dane I średnie opóźnienie różnych protokołów transportowych dla różnych okresów niedostępności łącza.

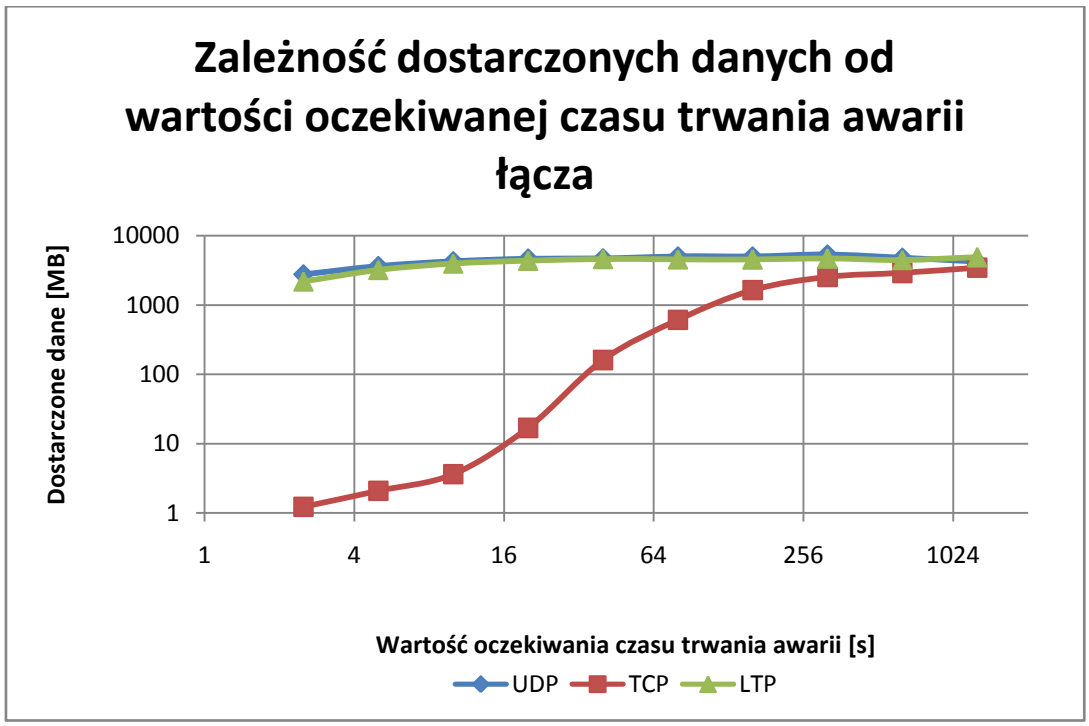
Wartość oczekiwana czasu trwania awarii [s]	Dostarczone dane [B]		
	UDP	TCP	LTP
1280	4218586789	3444464869	4881817721
640	4816402115	2895832229	4459450137
320	5331490232	2546142252	4683363399
160	5026246076	1643320033	4537630590
80	5031288349	612053880	4554013397
40	4712657901	161038227	4608487364
20	4630480516	16901528	4339641307
10	4265667948	3622356	3965969197
5	3634238632	2084424	3212068985
2,5	2764984635	1225976	2190567066

Tabela 4-2: Opóźnienie dostarczania paczki danych.

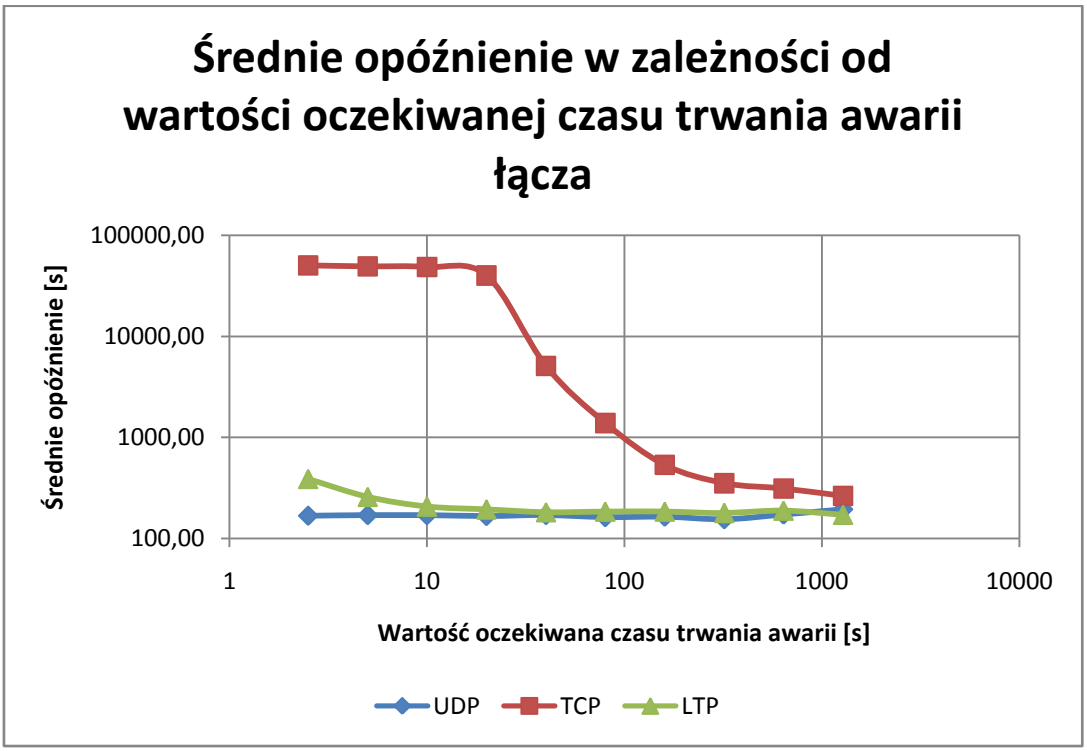
TTB [s]	Średnie opóźnienie [s]			Estymator średniego odchylenia [s]			Ufność [s]		
0	UDP	TCP	LTP	UDP	TCP	LTP	UDP	TCP	LTP
1280	195,35	264,64	172,55	566,98	657,44	456,11	1,11	1,43	0,83
640	173,49	313,04	189,03	318,07	554,53	400,71	0,58	1,31	0,76
320	155,82	353,12	179,20	195,65	502,21	260,50	0,34	1,27	0,48
160	164,92	536,58	185,49	158,71	662,34	182,95	0,29	2,08	0,35
80	163,07	1389,17	184,94	109,90	1418,97	128,86	0,20	7,30	0,24
40	171,01	5104,46	182,33	81,07	3465,59	95,75	0,15	34,77	0,18
20	167,48	40102,50	194,21	56,72	21698,14	69,64	0,11	671,75	0,13
10	170,79	48666,31	207,24	42,93	29066,01	53,55	0,08	1939,22	0,11
5	170,77	49409,60	259,63	29,71	28841,01	52,03	0,06	2527,98	0,12
2,5	168,43	50525,45	390,25	20,71	28905,31	67,64	0,05	3332,55	0,19

danych było relatywnie małe w porównaniu z *TCP*. Główną składową opóźnienia był czas oczekiwania danych w buforze wyjściowym łącza o pojemności 8 MB, co przy zadanej szybkości transmisji oznacza 80 s. Drugą składową był czas oczekiwania danych w buforze na naprawienie łącza (po wykryciu uszkodzenia łącza dane nie są dalej transmitowane). Protokół *LTP* uzyskał trochę gorszą charakterystykę opóźnień niż protokół *UDP*, ale nie była to różnica mająca duże znaczenie.

Natomiast widoczny jest bardzo silny wpływ wartości oczekiwanej czasu trwania awarii na protokół *TCP*. Poza pierwszym startem protokół *TCP* zwiększa swoje okno o 10% w momencie otrzymania poprawnego potwierdzenia dostarczenia danych oraz zmniejsza je o 50% w



Rysunek 4-11: Zależność dostarczonych danych od wartości oczekiwanej czasu trwania awarii łącza dla różnych protokołów transportowych.



Rysunek 4-12: Zależność średniego opóźnienia dostarczania pakietu Bundle od wartości oczekiwanej czasu trwania awarii łącza dla różnych protokołów transportowych.

momencie braku potwierdzenia. Gdy często występują przerwania, po każdym naprawieniu łącza okno (a więc i szybkość transmisji) zwiększają się z małą szybkością, natomiast po awarii okno i szybkość transmisji zmniejszają się z dużo większą szybkością. Jeżeli przerwania są bardzo częste, to *TCP* rozpędza się do bardzo małej szybkości zanim nastąpi awaria łącza. Dlatego średnia szybkość jest bardzo mała dla częstych awarii łącza.

Rozważając przypadek od strony teoretycznej, *TCP* powinien stać się efektywny dopiero wtedy, gdy czas rozpędzania się protokołu do najwyższej szybkości będzie dużo większy od czasu do awarii łącza.

TCP ma także zdecydowanie największe opóźnienie dostarczania danych. Główną składową jest tutaj oczekiwanie w buforze. Jest to związane z bardzo niską szybkością działania *TCP* dla częstych awarii łącza. Kiedy dane są transportowane z niską prędkością, to mija dużo czasu zanim dane z końca bufora wyjściowego łącza zostaną wysłane do węzła po drugiej stronie łącza.

4.2.1.2 Duży czas propagacji

Szybkość łącza wynosiła 10^5 B/s. Wartość czasu propagacji łącza była modyfikowana dla poszczególnych symulacji.

Tabela 4-3 przedstawia ilość dostarczonych danych w zależności od czasu propagacji łącza dla badanych protokołów transportowych. Rysunek 4-13 przedstawia graficznie ilość dostarczonych danych w zależności od czasu propagacji dla różnych protokołów transportowych.

Jak widać na wykresie, czas propagacji łącza ma bardzo niewielki wpływ na protokoły *UDP* oraz *LTP*. Oba protokoły uzyskały wynik zbliżony do maksymalnej teoretycznej możliwej przepływności łącza, tj. ok. 98%. Osiągnięcie teoretycznej wartości nie było możliwe ze względu na narzut nagłówków warstw łącza, sieciowej i transportowej. Natomiast czas propagacji łącza miał bardzo duży wpływ na szybkość działania protokołu *TCP*. Łatwo to zrozumieć, jeśli weźmie się pod uwagę, że czas dostarczenia okna z jednego węzła do drugiego jest równy:

$$T_d = 2T_p + T_{to} + T_{tp}$$

Gdzie:

T_p – czas propagacji,
T_{to} – czas transmisji okna,
T_{tp} – czas transmisji potwierdzenia.

Dla dużych rozmiarów okna zachodzi zależność:

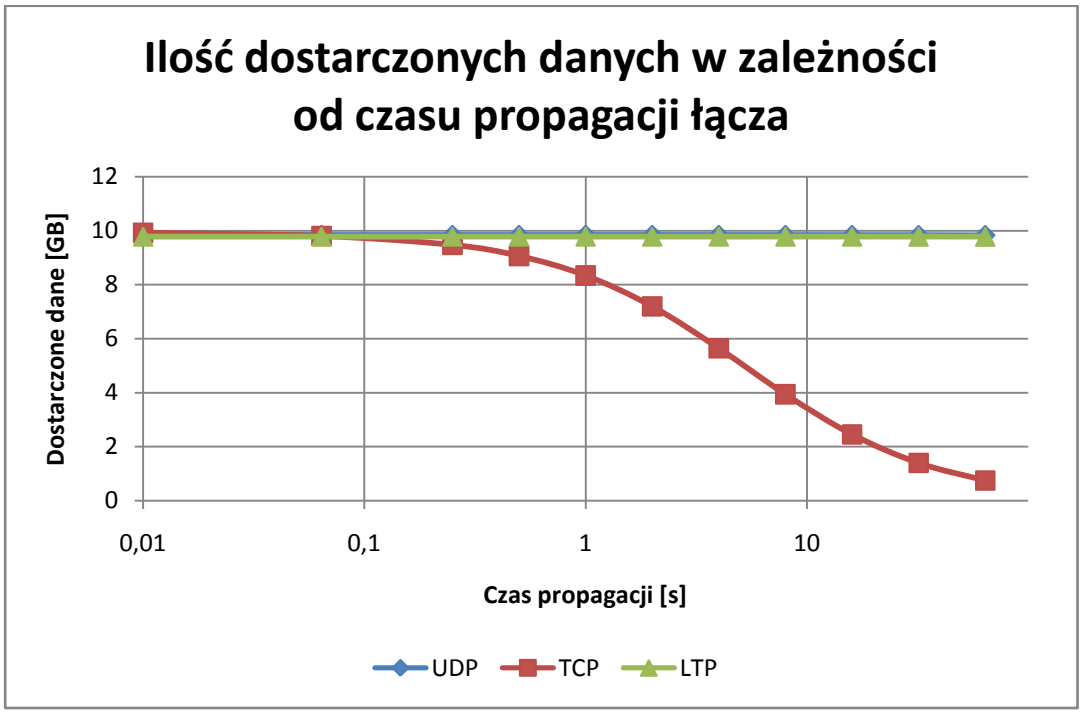
$$T_{to} \gg T_{tp}$$

Otrzymujemy równość:

$$T_d = 2T_p + T_{to}$$

Tabela 4-3: Ilość dostarczonych danych w zależności od czasu propagacji łącza dla różnych protokołów transportowych.

Czas propagacji [s]	Dostarczone dane [B]		
	UDP	TCP	LTP
64	9827130038	744029763	9775954138
32	9830277829	1394146530	9779086836
16	9831849609	2452159943	9780650154
8	9832634299	3936940517	9781434844
4	9833030875	5639828758	9781822958
2	9833224932	7191723386	9782021246
1	9833324076	8337814823	9782118559
0,5	9833372448	9059235647	9782165100
0,25	9833397834	9468785031	9782190486
0,064	9833414758	9798009713	9782209241
0	9833423220	9917708301	9782213472



Rysunek 4-13: Zależność ilości dostarczonych danych od czasu propagacji łącza dla różnych protokołów transportowych.

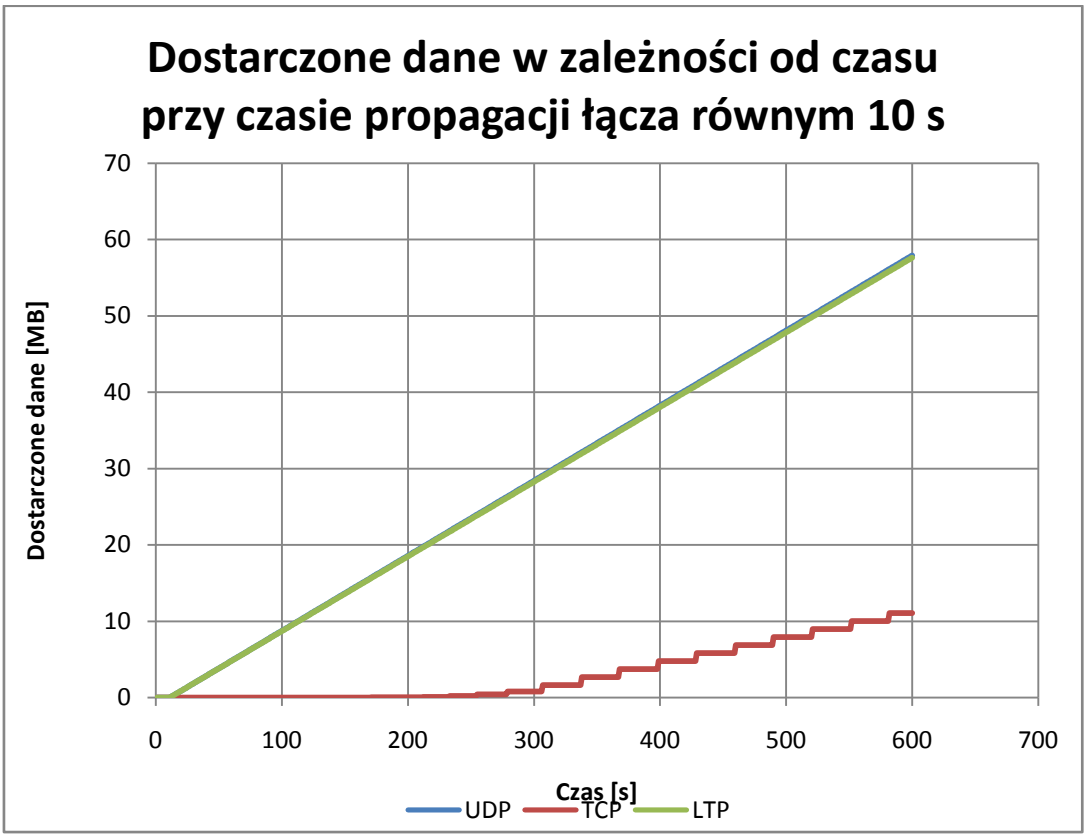
Widać więc, że dla dużego czasu propagacji główną składową jednego cyklu protokołu *TCP* jest czas propagacji, natomiast dla małych wartości jest to czas transmisji okna. W obu przypadkach przetransportowana jest taka sama ilość danych — aktualne okno *TCP*.

Dla $T_p << T_{to}$ *TCP* jest protokołem działającym efektywnie, dla $T_p >> T_{to}$ staje się protokołem nieefektywnym. Wartość graniczna jest w przybliżeniu uzyskiwana, gdy równie dużo czasu jest poświęcanie transmisji danych, co ich propagacji przez łącze, tj. gdy:

$$2T_p = T_{to}$$

co ma miejsce dla $T_p \approx 5$ s. Wykres pokazuje, że teoretyczne rozważania pokrywają się z uzyskanymi rezultatami.

Rysunek 4-14 prezentuje zależność ilości dostarczonych danych od czasu przy czasie propagacji 10 s. Wykres ten dobrze prezentuje zasadę działania symulowanych protokołów. *UDP* oraz *LTP* od samego początku działają z pełną szybkością i dostarczają stałą liczbę danych w jednostce czasu, dlatego ich wykres ma postać linii prostej. Natomiast *TCP* obsługuje wolny start. Dlatego początek wykresu ma postać eksponenty. W którymś momencie *TCP* osiąga maksymalny rozmiar okna i od tego momentu następuje liniowy przyrost ilości dostarczonych danych. Należy jeszcze nadmienić, że przyrosty danych występują skokowo w momencie dostarczenia okna. Stąd wykres przypomina swoim kształtem schody.



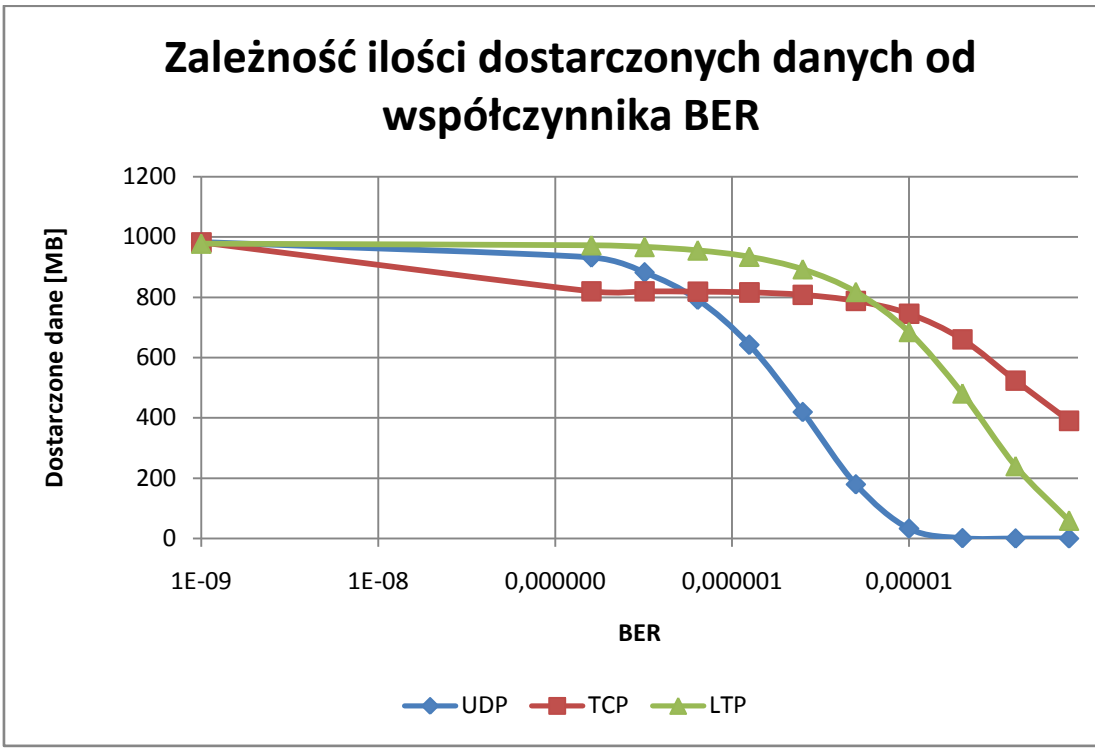
Rysunek 4-14: Ilość dostarczonych danych w zależności od czasu przy czasie propagacji łącza 10 s dla różnych protokołów transportowych.

4.2.1.3 *Przekłamania w transportowanych danych*

Szybkość łącza wynosiła 10^5 B/s. Czas propagacji dla wszystkich łączy wynosił 0,001 s. Wartość współczynnika *BER* dla wszystkich łączy była modyfikowana dla poszczególnych symulacji. Tabela 4-4 przedstawia ilość dostarczonych danych w zależności od wartości współczynnika *BER* podczas symulacji. Rysunek 4-15 przedstawia graficznie wyniki uzyskane w tym

Tabela 4-4: Liczba dostarczonych bajtów w zależności od współczynnika BER dla różnych protokołów.

BER	Dostarczone dane [bajty]		
	UDP	TCP	LTP
0,00008	0	390344147	59169895
0,00004	0	523500012	239908253
0,00002	1119721	660189837	481151683
0,00001	33024825	745226600	685266099
0,000005	179417031	788025883	817538084
0,0000025	419521406	808075198	893128526
0,00000125	642103455	816205332	934591528
0,00000064	791809007	818865628	955355029
0,00000032	881909282	819453908	966756589
0,00000016	931652654	819525436	972439929
0,000000001	983477163	982024856	978191306



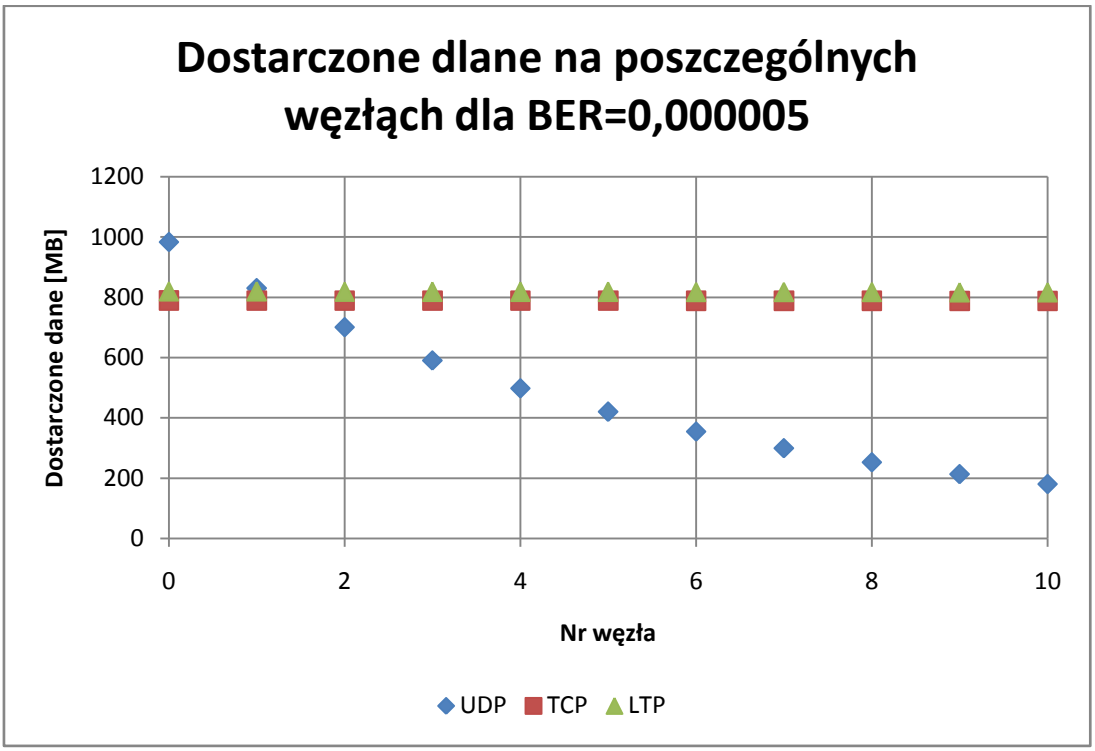
Rysunek 4-15: Zależność ilości dostarczonych danych od współczynnika BER dla różnych protokołów transportowych.

scenariuszu.

Jak widać na wykresie, protokołem który najszybciej pokazuje wpływ stopy błędów na szybkość transmisji danych, jest protokół *TCP*, który niepotrzebnie zmniejsza rozmiar okna (a więc także szybkość transmisji) po braku otrzymania potwierdzenia odebrania całego okna. Ta cecha protokołu *TCP* jest wadą w przypadku małej stopy błędów, natomiast okazuje się ogromną zaletą przy jej dużych wartościach. *TCP* jako jedyny protokół potrafi zmniejszyć rozmiar transmitowanych jednorazowo danych, dzięki czemu zmniejszeniu ulega prawdopodobieństwo przekłamania danych w oknie. W tej sytuacji *TCP* oscyluje wokół rozmiaru okna, przy którym wartość oczekiwana zwiększenia rozmiaru okna po prawidłowym dostarczeniu go do węzła docelowego jest równa wartości oczekiwanej zmniejszenia rozmiaru okna przy nieprawidłowym dostarczeniu okna do węzła docelowego (tj. dostarczeniu go z błędami, co w symulacji jest równoważne brakiem wysłania potwierdzenia).

Przy dużej stopie błędów zdecydowanie najgorzej radzi sobie protokół *UDP*, który nie dysponuje żadnym mechanizmem retransmisji danych przy wykryciu błędu.

Rysunek 4-16 prezentuje jeszcze jedną podstawową różnicę miedzy protokołami transportowymi mającymi mechanizm retransmisji i tymi, które jej nie mają. Jak widać na wykresie, protokoły *TCP* oraz *LTP* mają mniejszą szybkość transmisji przy badanej stopie błędów, natomiast każde poszczególne łącze działa z tą samą szybkością i niezawodnie transmituje wszystkie otrzymane dane. Dlatego liczba dostarczonych danych nie zmienia się na



Rysunek 4-16: Ilość dostarczonych danych przez różne protokoły na poszczególnych węzłach dla BER=0,000005.

żadnym z węzłów. Natomiast w przypadku protokołu *UDP* część danych jest tracona przy przejściu przez każde kolejne łącze. Dlatego liczba danych docierających do poszczególnych łączy jest coraz mniejsza. W ten sposób na kolejnych łączach istnieje już rezerwa szybkości, tj. łącza nie działają z pełną szybkością, ponieważ nie otrzymują wystarczająco dużo danych. Przepływność łącza jest marnowana.

Protokół *UDP* jest szczególnie niewydajny, kiedy dane przechodzą przez dużą liczbę łączy. Ilość danych dostarczonych do kolejnych węzłów maleje wykładniczo.

Kolejną bardzo niekorzystną cechą *UDP* jest to, że jeśli paczka danych nie dotrze do węzła docelowego, to wystąpi retransmisja danych z węzła początkowego, co znaczy, że te same dane będą ponownie transmitowane przez całą sieć i mogą ponownie zostać odrzucone z powodu przekłamań. W przypadku protokołów gwarantujących niezawodne dostarczenie danych takie retransmisje w ogóle by nie występowały.

4.2.1.4 Wnioski

Symulacja protokołów transportowych wykazała wyższość protokołu *LTP* nad innymi protokołami transportowymi w typowych sytuacjach, jakie występują w sieciach *DTN*. Chociaż należy nadmienić, że występują pewne specyficzne sytuacje, w których inne protokoły uzyskują nieco lepsze wyniki. Protokół *UDP* ma minimalnie mniejszy narzut transmitowanych danych. Natomiast protokół *TCP* lepiej sobie radzi przy bardzo dużej liczbie przekłamań występujących w danych transmitowanych przy małym opóźnieniu.

LTP osiąga największą przewagę nad pozostałymi badanymi protokołami transportowymi dla ekstremalnie dużych wartości czasu propagacji łącza oraz przy dużej liczbie gubionych (lub

uszkadzanych) danych. Jednocześnie *LTP* powinien być jedynym protokołem, który dobrze radzi sobie w sytuacji, gdy jednocześnie występują duże opóźnienia i częste straty danych. Taki przypadek nie był badany ze względu na bardzo dużą liczbę przypadków do sprawdzenia (konieczne byłoby przeprowadzenie symulacji przy jednoczesnej modyfikacji czasu propagacji I np. wartości *BER*, czyli w dwóch wymiarach).

4.2.2 Protokoły rutingu

Symulacja działania rutingu statycznego została pominięta, ponieważ wymaga ona ingerencji człowieka w konfigurację sieci. Efektywność statycznego protokołu rutingu zależy od zdolności człowieka do skonfigurowania określonej sieci i dlatego nie jest mierzalna. Do symulacji pozostało 6 protokołów rutingu, przy czym dla zwięzłości poszczególnym protokołom przypisano skrócone nazwy, co było konieczne do przedstawienia wykresów:

- Gradientowy (gradientowy).
- Gradientowy z wyborem dostępnej drogi i odrzucaniem pakietów (przekierowujący).
- Oparty na *AODV* (AODV).
- Oparty na *OSPF* (SPF).
- Ze zdolnością do przewidywania przestojów (przewidujący).
- Oparty na epidemicznym rozprzestrzenianiu wiadomości (epidemiczny).

4.2.2.1 Sieć internetowa

Czas propagacji łącza został ustawiony na wartość 10 ms. Przepływność łączy została ustawiona na 10^5 B/s, co oddaje rzeczywistą szybkość istniejących łączy internetowych. Występowanie awarii i napraw łączy ma rozkład wykładniczy o wartości oczekiwanej odpowiednio 10^3 s oraz 10 s, co znaczy, że łącze jest niedostępne przez ok. 1% czasu. Ze względu na istnienie wielu dróg do celu nie powinna się pojawić sytuacja (albo przynajmniej będzie ona bardzo rzadka), kiedy nie istnieje żadna droga do celu.

Ilość generowanych danych można wyliczyć w następujący sposób: w badanej sieci każdy węzeł może wysyłać dane do 3 węzłów oddalonych od niego o jedno łącze, 3 węzłów oddalonych od niego o dwa łącza oraz do jednego węzła oddalonego od niego o trzy łącza (podane odległości przedstawiają najkrótszą istniejącą trasę). Średnia liczba łączy przez które przechodzi jeden pakiet (przy założeniu braku awarii łącza) to:

$$L = \frac{3 \times 1 + 3 \times 2 + 1 \times 3}{7}$$

$$L \approx 1{,}72$$

8 węzłów ma generować taki ruch na 12 łączach, aby łącze było obciążone w 50% (należy pamiętać, że łącze ma dwa kierunki transportu danych):

$$8 \times S = \frac{2 \times 50\% \times 12 \times 10^5}{L}$$

$$S \approx 8{,}6 \times 10^5\ B/s$$

Uwzględniając dodatkowy narzut protokołów warstwy transportowej, sieciowej i łącza danych można uznać, że średnio każdy węzeł powinien generować ok. 8×10^4 B/s, które z równym prawdopodobieństwem będą trafiać do każdego z pozostałych węzłów.

Czas symulacji po kilku testowych uruchomieniach ustawiony został na wartość 10^5 s. Dłuższy czas nie był możliwy, ponieważ niektóre algorytmy stosowane przez protokoły rutingu były tak obciążające dla procesora, że pojedyncza symulacja zajmowała dwie godziny. Warunki początkowe w sieci można było pominąć, ponieważ stan sieci staje się stabilny po bardzo krótkim czasie rzędu kilku sekund, co w relacji do czasu symulacji jest pomijalne.

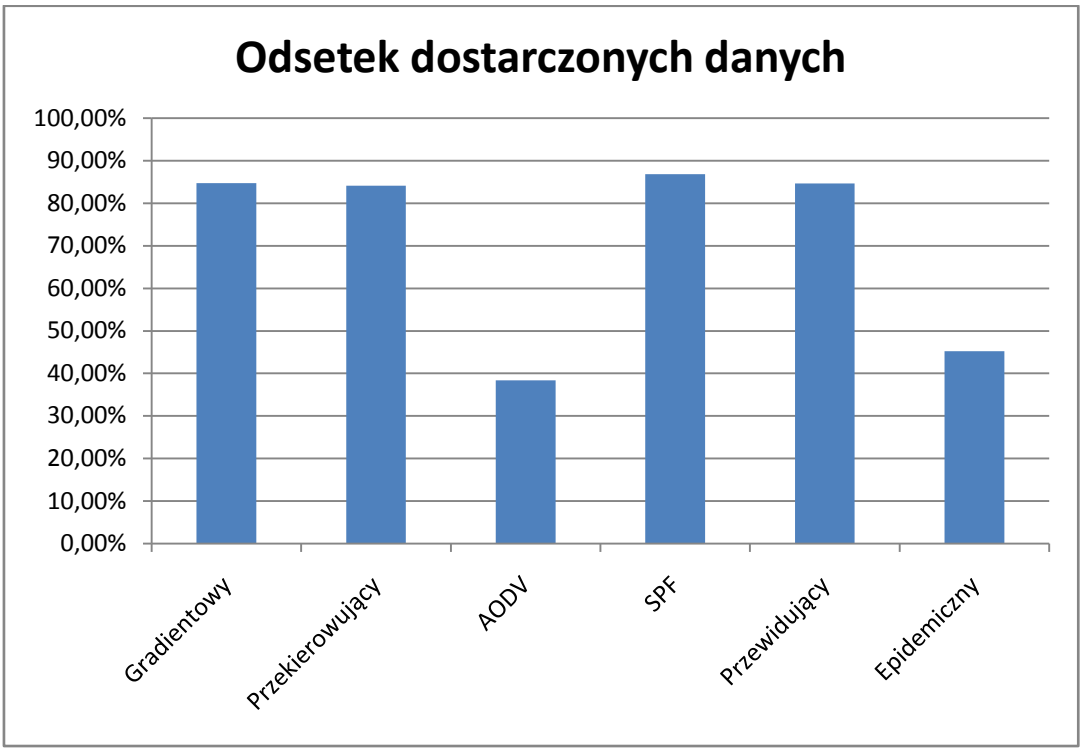
4.2.2.1.1 Niezawodność dostarczania danych

Tabela 4-5 przedstawia zestawienie ilości dostarczonych danych przez poszczególne protokoły. Rysunek 4-17 prezentuje graficznie porównanie niezawodności dostarczania danych przez poszczególne protokoły rutingu.

Widać wyraźnie, że protokoły rutingu nie radzą sobie z niezawodnym dostarczaniem danych do węzła końcowego. Głębsza analiza danych z symulacji pozwoliła ustalić, że pakiety gubione są w buforze wyjściowym łączy. Jest to spowodowane przepełnieniem się bufora danymi. Główną przyczyną jest najprawdopodobniej brak funkcji rozdzielania wysyłania danych na różne łącza (ang. *load balancing*). Powoduje to, że dany węzeł kieruje dane znacznie częściej do wybranych łączy niż do innych. Analiza szczegółowych danych z sieci potwierdziła nierównomierne wykorzystanie poszczególnych łączy. Chwilowo większa liczba danych, które węzeł próbuje wysłać przez łącze lub awaria łącza sprawiają, że bufor wyjściowy się przepełnia. Kolejne przybywające dane są odrzucane. Odrzucenie danych powoduje oczywiście retransmisję, która dodatkowo zajmuje łącza. Im więcej retransmisji, tym w większym stopniu są zajęte łącza i częściej dochodzi do przepełnienia bufora itd. Najlepszy rezultat osiągnął protokół SPF, który dostarczał dane z blisko 87% skutecznością. Nie jest to dobry rezultat, ale pozwala zauważyć, że sieci *DTN* nie zostały zaprojektowane do dużych przepływności i o ile nie przeniesiemy wielu mechanizmów z sieci *Internet* do *DTN*, o tyle sposób transmisji będzie bardzo nieefektywny. Należy jeszcze zauważyć, że efektywności dostarczania danych nie da się podnieść zwiększając liczbę retransmisji, ponieważ dodatkowe retransmisje obciążyłyby dodatkowo łącze.

Tabela 4-5: Ilość wygenerowanych i dostarczonych danych dla różnych protokołów rutingu w sieci internetowej.

Protokół rutingu	Wygenerowane dane [B]	Dostarczone dane [B]	Odsetek dostarczonych danych
Gradientowy	63968047490	54184725883	84,71%
Przekierowujący	63990258187	53837103776	84,13%
AODV	64030915451	24553863723	38,35%
SPF	63935196660	55515452868	86,83%
Przewidujący	63899892729	54112117671	84,68%
Epidemiczny	63896171695	28898456694	45,23%



Rysunek 4-17: Odsetek dostarczonych danych dla różnych protokołów rutingu w sieci internetowej.

Protokoły gradientowy, przekierowujący, SPF i przewidujący mają zbliżoną niezawodność dostarczania danych do węzła docelowego, natomiast protokoły epidemiczny oraz AODV dają wyraźnie gorsze rezultaty. Powodem, dla którego protokół epidemiczny osiąga tak złe rezultaty, jest nadmierne zwielokrotnianie wiadomości, co prowadzi do szybkiego przepełniania się buforów i odrzucania bardzo dużego odsetka wiadomości. Powodem, dla którego AODV nie sprawdza się w tym scenariuszu, była m.in. trudność z odnalezieniem drogi do węzła docelowego. Na 16 036 466 wygenerowanych porcji danych (które później były także retransmitowane) 8 796 521 razy zdarzało się, że protokołowi w ogóle nie udało się odnaleźć drogi. Przyczyną było gubienie wiadomości AODV w przepełnionych buforach. Poza tym AODV ma tendencję do nieregularnego wysyłania danych — pewna liczba danych jest generowana i oczekuje na odnalezienie ścieżki w sieci. Kiedy ścieżka zostanie już odnaleziona, na wysłanie czeka bardzo dużo danych, co daje rezultat w postaci wysyłania danych dużymi paczkami, które łatwiej przepełniają bufor.

4.2.2.1.2 Transmitowane dane

Tabela 4-6 prezentuje zestawienie ilości wysyłanych bajtów na łączach dla różnych protokołów rutingu. Uwzględniając znajomość przepływności poszczególnych łączy (10^5 B/s), czas symulacji (10^5 s), liczbę łączy (12) oraz pamiętając, że łącza są dwukierunkowe, możemy łatwo wyznaczyć maksymalną teoretyczną liczbę transmitowanych danych, która jest iloczynem przepływności łączy, czasu symulacji, liczby węzłów oraz liczby kierunków transmisji danych w pojedynczym łączu:

$$D = 2 \times 12 \times 10^5 \times 10^5$$

Tabela 4-6: transmitowane dane różnych protokołów rutingu w sieci internetowej.

Protokół rutingu	Transmitowane dane [B]	Procent zajętości łącza
Gradientowy	205285463742	85,54%
Przekierowujący	196332867745	81,81%
AODV	130955659922	54,56%
SPF	217278651366	90,53%
Przewidujący	207037119367	86,27%
Epidemiczny	239602092754	99,83%

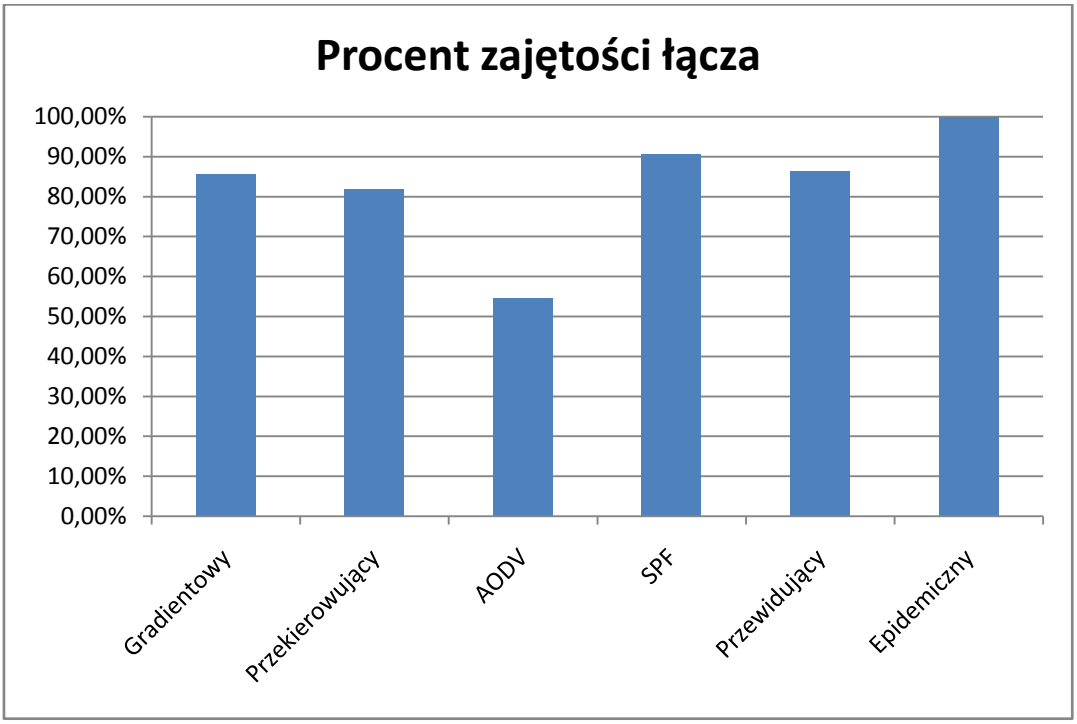
$D = 2{,}4 \times 10^{11} B$

Uwzględniając ten parametr możemy łatwo wyznaczyć procent zajętości łączy dla danego protokołu (wartość ta podana została w tabeli).

Rysunek 4-18 przedstawia graficzną prezentację rezultatów z tabeli.

Widać, że ponownie protokoły gradientowy, przekierowujący, SPF i przewidujący dają podobne rezultaty, natomiast AODV i epidemiczny wyraźnie się od nich różnią. Protokół epidemiczny w blisko 100% zużywa wszystkie zasoby sieci, tym samym okazuje się bardzo nieefektywny w tym scenariuszu. Natomiast AODV zajmuje bardzo mało zasobów, co ma bezpośredni związek z tym, że duży odsetek paczek danych albo nie jest w ogóle wysyłany przez sieć (ponieważ nie odnaleziono drogi), albo jest wysyłany, ale w nieefektywny sposób i następuje odrzucenie większości wysłanych danych podczas przekazywania ich do bufora.

Protokół SPF, który dał najlepsze rezultaty pod względem efektywności dostarczania danych, tutaj pokazuje swoje wady — zużywa dodatkowe zasoby sieci na przekazywanie informacji o jej



Rysunek 4-18: Procent zajętości łącza dla różnych protokołów rutingu w sieci internetowej.

stanie. Zużycie jest wyraźnie większe w porównaniu z protokołami gradientowym, przekierowującym i przewidującym, które dają podobne rezultaty jeśli chodzi o niezawodność dostarczania danych. Jednak w sieci Internet koszt transmisji danych jest bardzo niski i dlatego SPF nadal wydaje się najlepszy spośród testowanych protokołów.

Warto zauważyć, że praktycznie każdy protokół przekroczył zakładane 50% obciążenia łącza. Było to głównie spowodowane retransmisjami.

4.2.2.1.3 Opóźnienie

Tabela 4-7 przedstawia zestawienie opóźnień dostarczania paczek danych dla różnych protokołów rutingu. Rysunek 4-19 przedstawia graficznie dane nt. średniego opóźnienia.

Nie licząc protokołu epidemicznego, wszystkie protokoły dają bardzo zbliżone rezultaty jeśli chodzi o średni czas transportu paczki danych od źródła do celu. Wartość tego czasu wynika głównie z oczekiwania w buforze wyjściowym łącza na wysłanie wiadomości, a nie ze sposobu wyboru ścieżki. Dlatego między protokołami nie ma dużych różnic.

Jedynym protokołem, który dał wyraźnie inne wyniki od pozostałych protokołów, jest protokół epidemiczny. Protokół ten transmituje tak dużą liczbę danych, że każdy bufor wyjściowy łącza w sieci jest pełny niemal w 100%. Dlatego paczka danych, która ostatecznie dociera do celu, po drodze musi oczekiwać maksymalny czas w każdym buforze.

4.2.2.1.4 Wnioski

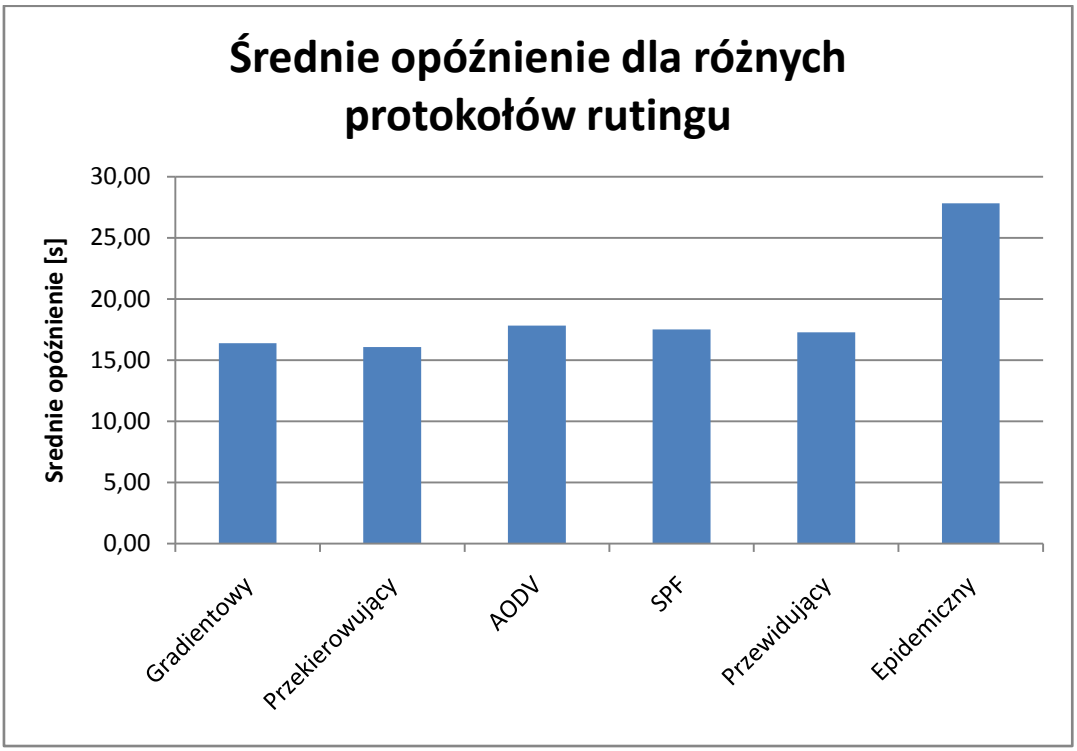
Sieci *DTN* nie są zaprojektowane do obsługi dużych przepływności i dlatego nie radzą sobie dobrze w takich warunkach. W sieciach *DTN* zbliżonych charakterystyką do sieci internetowej najlepiej sprawdza się protokół SPF, który jest także wykorzystywany w sieci *Internet*. Protokoły gradientowy, przekierowujący i przewidujący dają bardzo zbliżone rezultaty do SPF i są dla niego zadowalającym zamiennikiem. Natomiast protokoły AODV i epidemiczny nie sprawdzają się zupełnie, a w szczególności nie są skalowalne ze względu na rozgłaszanie wiadomości. Wyklucza to ich użycie w dużych sieciach.

4.2.2.2 Sieć sensorowa

Czas propagacji łączy został ustawiony na poziomie 10 ms. Przepływność łączy została ustawiona na 10^5 B/s. Łącza ulegały uszkodzeniom i naprawom z rozkładem o wartości oczekiwanej 10 s. Oczywiście oznacza to, że łącze było sprawne średnio przez 50% czasu. Węzeł źródłowy generował dane średnio co 10^2 s, przy czym porcja danych miała rozmiar

Tabela 4-7: Opóźnienie dostarczania paczek danych dla różnych protokołów rutingu w sieci internetowej.

Protokół rutingu	Średnie opóźnienie[s]	Odchylenie standardowe opóźnienia [s]	Ufność [s]
Gradientowy	16,39	11,43	0,00327
Przekierowujący	16,08	11,59	0,00341
AODV	17,82	14,70	0,00840
SPF	17,50	11,33	0,00316
Przewidujący	17,28	11,71	0,00337
Epidemiczny	27,83	15,00	0,00886



Rysunek 4-19: Średnie opóźnienie dla różnych protokołów rutingu w sieci internetowej.

średnio 10^4 B.

Eksperymenty z czasem trwania symulacji pozwoliły określić jego optymalną wartość na poziomie 5×10^6 s. Dłuższy okres nie był możliwy ze względu na symulację protokołu SPF, który transmituje informacje w momencie zmian w sieci. Zmiany te były znacznie częstsze od generowania danych przez węzły i dlatego w tym samym czasie symulacji aplikacja potrzebowała wykonywać wielokrotnie więcej obliczeń, co znacznie wydłużało czas trwania symulacji.

4.2.2.2.1 Niezawodność dostarczania danych

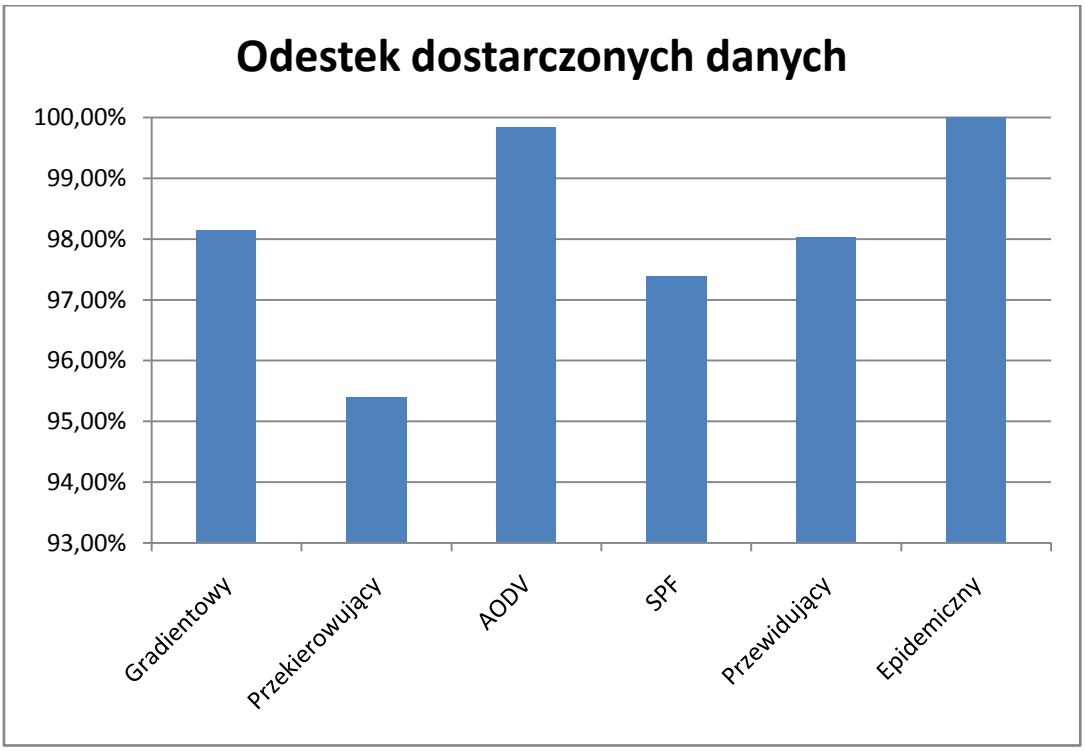
Tabela 4-8 zawiera zestawienie ilości poprawnie dostarczonych danych w sieci w relacji do wygenerowanych danych.

Tabela 4-8: Dostarczone dane dla różnych protokołów rutingu w sieci sensorowej.

Protokół rutingu	Wygenerowane dane [B]	Dostarczone dane [B]	Odestek dostarczonych danych
Gradientowy	501489015	492181618	98,14%
Przekierowujący	499357640	476387560	95,40%
AODV	498641497	497851375	99,84%
SPF	500417555	487373996	97,39%
Przewidujący	501221770	491338929	98,03%
Epidemiczny	503666278	503666278	100,00%

Rysunek 4-20 przedstawia graficznie odsetek dostarczonych danych dla różnych protokołów rutingu.

Protokół epidemiczny uzyskał bardzo dobry wynik – 100% niezawodność dostarczania danych. Dobry wynik osiągnął także protokół AODV. Protokoły przewidujący, gradientowy i SPF osiągnęły średni wynik, natomiast zdecydowanie najgorszy okazał się protokół przekierowujący. Prawie 5% danych nie dotarło ostatecznie do celu mimo trzykrotnego wysyłania spowodowanego retransmisjami. Krótka analiza dodatkowych danych pozwala na znalezienie przyczyny nieefektywnego dostarczania danych w przypadku protokołu przekierowującego. Protokół ten odrzuca paczkę danych, kiedy nie istnieje żadna droga do celu dostępna w czasie rutowania. Ponieważ statystycznie każdy węzeł miał od jednej do trzech dróg zbliżających pakiet do celu, zaś w 50% każdy z pośrednich węzłów mógł być niedostępny, więc bardzo często rutowana paczka danych była odrzucana z powodu nieznalezienia drogi do celu. Przyczyną nieefektywnego działania SPF jest natomiast nieefektywne przekazywanie informacji o stanie sieci do poszczególnych węzłów. Aż 17 790 229 razy paczka danych była tracona z powodu awarii węzła, w którym oczekiwała na naprawienie łącza. Jest to bardzo duża liczba w porównaniu z 148 818 paczkami z danymi, jakie w ogóle zostały wygenerowane przez węzeł źródłowy. Absolutną większość odrzucanych danych stanowiły informacje o rutingu. Zanim sieć zsynchronizowała informacje o swoim stanie (dzięki regularnemu rozsyłaniu informacji o aktualnym stanie łączy każdego węzła), następowała kolejna awaria i kolejne dane były tracone. Brak pełnej synchronizacji doprowadzał do wysyłania paczek w błędnym kierunku lub do nie odnajdywania drogi, która istniała (co nastąpiło 58 489 razy, dla porównania AODV – tylko 13 442).



Rysunek 4-20: Odsetek dostarczonych danych dla różnych protokołów rutingu w sieci sensorowej.

Znając przepływność łączy (10^5 B/s), czas symulacji (5×10^6 s) oraz liczbę łączy (12) łatwo wyliczyć maksymalną możliwą liczbę bajtów, jakie mogły zostać przetransmitowane w sieci:

$$D = 2 \times 12 \times 10^5 \times 5 \times 10^6$$

$$D = 1{,}2 \times 10^{13} B$$

Tabela 4-9 przedstawia zestawienie ilości transmitowanych danych dla różnych protokołów. Rysunek 4-21 przedstawia graficzną prezentację danych o procentowej zajętości łącza.

Pod względem liczby transmitowanych danych zdecydowanie najgorszy okazał się protokół SPF. Jako jedyny transmituje on dane także wtedy, kiedy źródło danych nie próbuje niczego wysłać przez sieć. Ponieważ w sieciach sensorowych uszkodzenia łączy występują częściej od próby wysłania danych, dlatego SPF okazuje się zupełnie nieprzydatny. Protokół epidemiczny, który uzyskał bardzo dobrą niezawodność dostarczania danych, tutaj prezentuje swoje wady. Zwielokrotnianie wysyłanej wiadomości okazuje się bardzo kosztowne z punktu widzenia liczby transmitowanych danych. Protokół gradientowy i przewidujący uzyskały podobne rezultaty. Najlepsze wyniki dały protokół AODV oraz protokół przekierowujący. Mogłoby się wydawać, że protokół AODV, jako protokół, który potrzebuje rozpropagować żądanie odnalezienia drogi w sieci, będzie transmitować większą liczbę danych od protokołu gradientowego czy przewidującego. Okazuje się jednak, że jest on bardziej efektywny od protokołu gradientowego. Przyczyna leży w niezawodności dostarczania danych przez AODV oraz w grupowym wysyłaniu wielu paczek na raz. Protokół AODV zestawia ścieżkę, która zostaje sprawdzona przed wysłaniem danych. Dlatego jest ona pewna i dlatego też występuje bardzo mało retransmisji. W czasie działania symulacji tylko 19 080 razy wystąpiła potrzeba retransmisji danych. Dla porównania, protokół gradientowy w tym samym czasie potrzebował 101 154 retransmisji, ponieważ niektóre paczki albo zostały odrzucone, albo były przechowywane w buforze wyjściowym łącza czekając na naprawę węzła po drugiej stronie.

Tabela 4-9: Transmitowane dane dla różnych protokołów rutingu w sieci sensorowej.

Protokół rutingu	Transmitowane dane [B]	Procent zajętości łącza
Gradientowy	2422599100	0,0202%
Przekierowujący	2035611092	0,0170%
AODV	2163944992	0,0180%
SPF	40141829281	0,3345%
Przewidujący	2400395803	0,0200%
Epidemiczny	7101213611	0,0592%

Rysunek 4-21: Procent zajętości łącza dla różnych protokołów rutingu w sieci sensorowej.

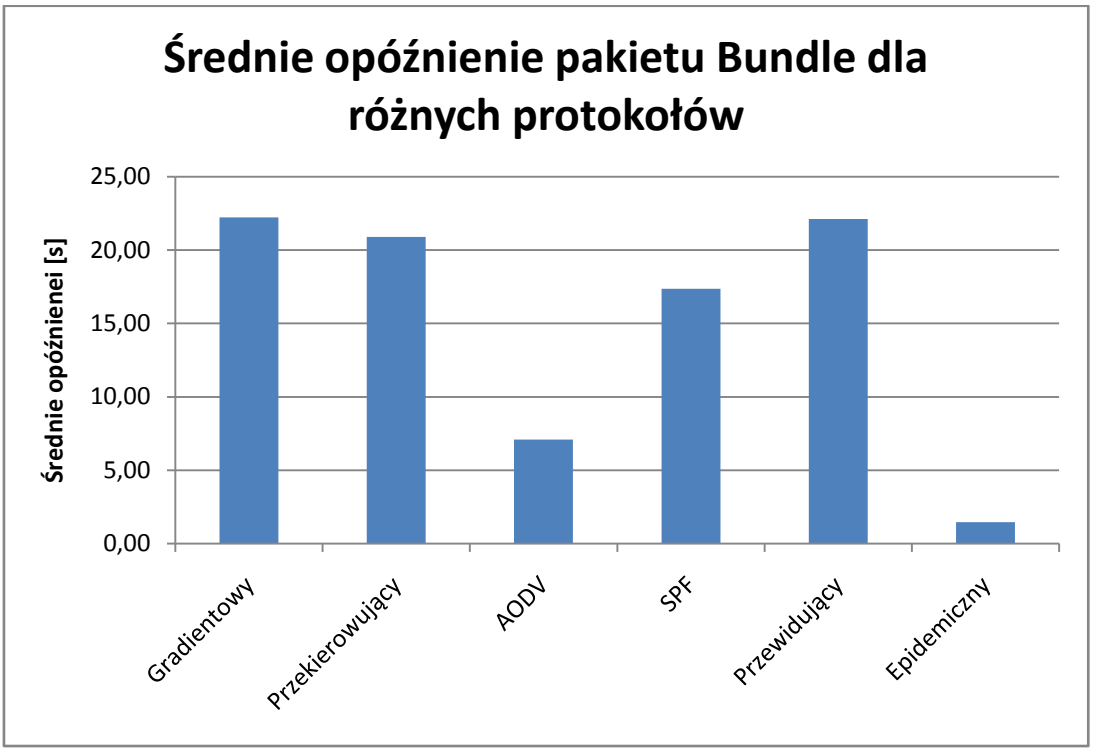
4.2.2.2.2 Opóźnienia

Tabela 4-10 przedstawia zestawienie danych dotyczących opóźnienia transmisji paczki danych dla różnych protokołów. Rysunek 4-22 przedstawia graficznie dane o średnim opóźnieniu dostarczania pakietu Bundle.

Jak widać, protokoły gradientowy, przekierowujący oraz przewidujący bardzo wolno dostarczają dane do węzła docelowego. Nieco lepiej radzi sobie SPF, jednak z powodu niedoskonałości dostarczania informacji o strukturze sieci dane nie zawsze kierowane są tam, gdzie powinny. Znacznie lepiej radzi sobie protokół AODV, który wysyła paczki danych tylko wtedy, kiedy odnajdzie zdatną drogę w sieci. Dzięki temu paczki danych prawie nigdy nie oczekują w buforze wyjściowym łącza na naprawienie węzła po drugiej stronie łącza. Protokół epidemiczny okazał się w tej sferze bezkonkurencyjny, uzyskując rezultat prawie pięć razy lepszy od drugiego w kolejności AODV. Różnica pomiędzy AODV a protokołem epidemicznym

Tabela 4-10: Opóźnienie dla różnych protokołów rutingu w sieci sensorowej.

Protokół rutingu	Średnie opóźnienie[s]	Odchylenie standardowe opóźnienia [s]	Ufność [s]
Gradientowy	22,23	29,96	0,152
Przekierowujący	20,90	32,26	0,163
AODV	7,10	18,15	0,092
SPF	17,36	29,63	0,150
Przewidujący	22,11	29,73	0,150
Epidemiczny	1,46	3,36	0,017



Rysunek 4-22: Średnie opóźnienie pakietu Bundle dla różnych protokołów rutingu w sieci sensorowej.

bierze się z dwóch powodów: po pierwsze protokół epidemiczny wykorzystuje wszystkie dostępne ścieżki równocześnie i wystarczy, aby tylko jedna z nich była zdatna. Dzięki temu praktycznie zawsze dane docierają do celu tak szybko, jak tylko fizycznie jest to możliwe. Drugi powód jest taki, że w przypadku AODV węzeł wysyłający dane musi najpierw wysłać wiadomość z żądaniem zestawienia ścieżki do węzła docelowego, później musi poczekać, aż ta wiadomość dotrze, a w drugą stronę zostanie wysłany pakiet z informacją o zestawieniu drogi. Dopiero wtedy rozpoczyna się wysyłanie pierwszego pakietu.

Przedstawione dane związane były ze średnim czasem transportu pojedynczej paczki protokołu *Bundle* od momentu jej wysłania do momentu dotarcia do węzła docelowego. Zauważmy jednak, że nie jest to jedyne opóźnienie, jakie istnieje w sieci. Często bardziej użyteczną informacją może być czas od wygenerowani określonej porcji danych do momentu jej odebrania w węźle docelowym. Zauważmy, że są to dwie różne wartości, ponieważ określona porcja danych może być wysyłana w wielu pakietach protokołu *Bundle* po retransmisjach.

Tabela 4-11 przedstawia zestawienie opóźnienia przesyłania porcji danych przez poszczególne protokoły rutingu. Rysunek 4-23 przedstawia graficznie wartości średniego opóźnienia porcji danych.

Opóźnienie porcji danych jest bardzo podobne do opóźnienia pakietu *Bundle*. Jedyna większa różnica występuje dla protokołu epidemicznego, dla którego wystąpił przyrost opóźnienia na poziomie 15%. Przyrost ten wynika z relacji czasu retransmisji (50 s) do średniego opóźnienia pakietu *Bundle*. Jeśli protokół epidemiczny nie dostarczył danych przy pierwszym wysłaniu wiadomości, to dodatkowy czas związany z oczekiwaniem na retransmisję wpływał na opóźnienie dużo bardziej niż dla innych protokołów.

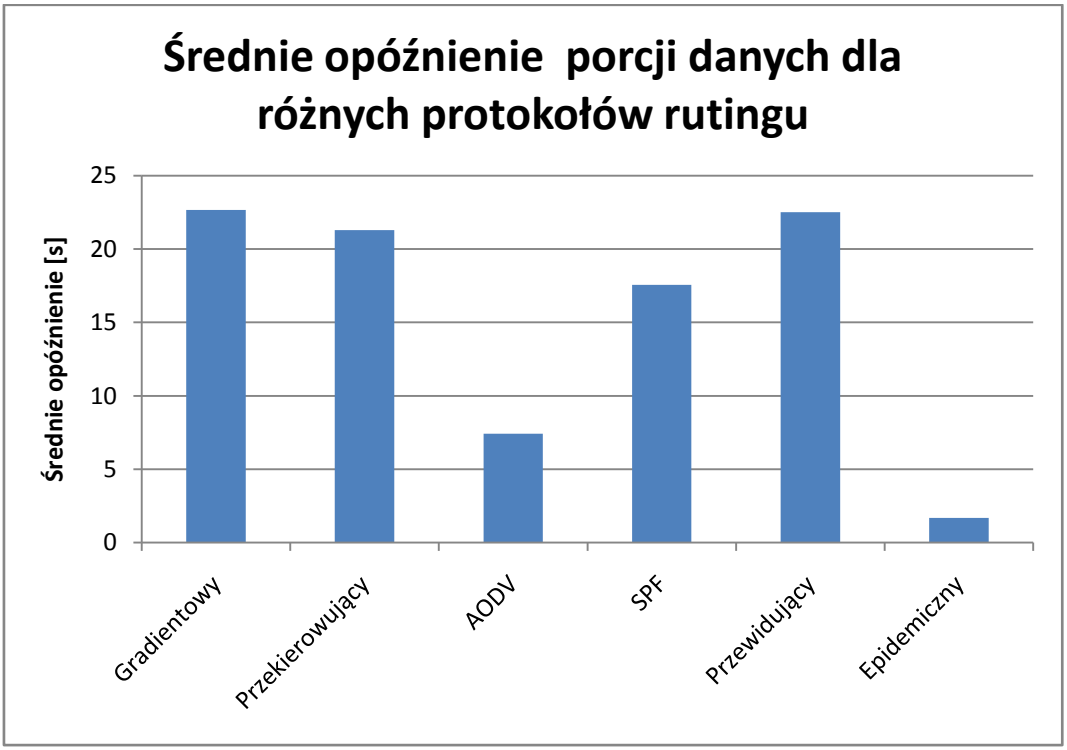
Tabela 4-11: Opóźnienie dostarczania porcji danych do węzła docelowego dla różnych protokołów rutingu w sieci sensorowej.

Protokół rutingu	Średnie opóźnienie [s]	Odchylenie standardowe opóźnienia [s]	Ufność [s]
Gradientowy	22,67	30,30	0,153
Przekierowujący	21,28	32,54	0,165
AODV	7,41	18,77	0,095
SPF	17,57	29,91	0,151
Przewidujący	22,52	30,10	0,152
Epidemiczny	1,68	4,70	0,024

4.2.2.3 Sieci z mułami danych

Każde z łączy występujących w sieci z mułami danych jest dostępne przez okres średnio 10 s. Czas przestoju wynosi 100 s. Oznacza to, że dane mogą być transportowane przez łącze średnio przez 9% czasu. Szybkość łączy to 10^5 B/s. Węzeł źródłowy generuje średnio 10^4 B danych co 20 s. Eksperymentalne symulacje pokazały, że przy takich parametrach nigdy nie występuje przepełnienie sieci, w szczególności nie występuje ono podczas symulacji protokołu epidemicznego, który zwielokrotnia wysyłane dane. Czas symulacji eksperymentalnie został ustalony na 2×10^7 s. Przy tej wartości najwolniej symulowany protokół AODV potrzebuje 1,5 godziny na symulację. Warunki początkowe ponownie mogą zostać zaniedbane, ponieważ czas symulacji jest wielokrotnie większy od czasu, po którym stan sieci stabilizuje się.

Dla symulacji istotna była konfiguracja symulacja protokołu AODV. Przy przyjętym podejściu



Rysunek 4-23: Średnie opóźnienie porcji danych dla różnych protokołów rutingu w sieci sensorowej.

protokół oczekuje do 200 s na odnalezienie drogi dla pojedynczego pakietu.

4.2.2.3.1 Wyszukiwanie drogi przez protokół rutingu

Ze względu na charakterystykę sieci, niemożliwe było odrzucenie danych z powodu przepełnienia bufora, przekłamań bitów występujących podczas transmisji danych, awarii węzłów i łączy. Dlatego jedyną przyczyną, dla której dane nie dotarły do celu, mogło być nieodnalezienie drogi przez protokół rutingu.

Tabela 4-12 przedstawia zestawienie liczby wygenerowanych paczek danych oraz liczbę odrzucanych pakietów protokołu *Bundle* z powodu nieodnalezienia dla nich drogi w sieci.Rysunek 4-24 przedstawia graficznie odsetek dostarczonych paczek danych.

Protokoły gradientowy, przewidujący oraz epidemiczny osiągnęły 100% skuteczność dostarczania danych, co było do przewidzenia. Niewielka różnica pomiędzy wygenerowanymi a dostarczonymi pakietami widoczna w tabeli spowodowana jest tym, że podczas zbierania statystyk niektóre pakiety *Bundle* ciągle podróżowały przez sieć. Protokół AODV uzyskał średni wynik. Dla tego protokołu duże znaczenie miał maksymalny czas oczekiwania na znalezienie drogi w sieci (zanim uznał on, że droga nie została znaleziona i odrzucił pakiety). Skuteczność dostarczania danych na poziomie 70% bynajmniej nie czyni z AODV protokołu efektywnego, jednak przy wielokrotnych retransmisjach możliwe teoretycznie stałoby się uzyskanie akceptowalnych rezultatów.

Protokoły przekierowujący oraz SPF okazały się skrajnie nieefektywne w testowanej sieci. Są one w stanie przesyłać dane tylko wtedy, kiedy istnieje bezpośredni kontakt między węzłem początkowym a docelowym. Ponieważ w sieci zdarzało się to bardzo rzadko, więc żaden z wymienionych protokołów nie był w stanie uzyskać dobrego rezultatu.

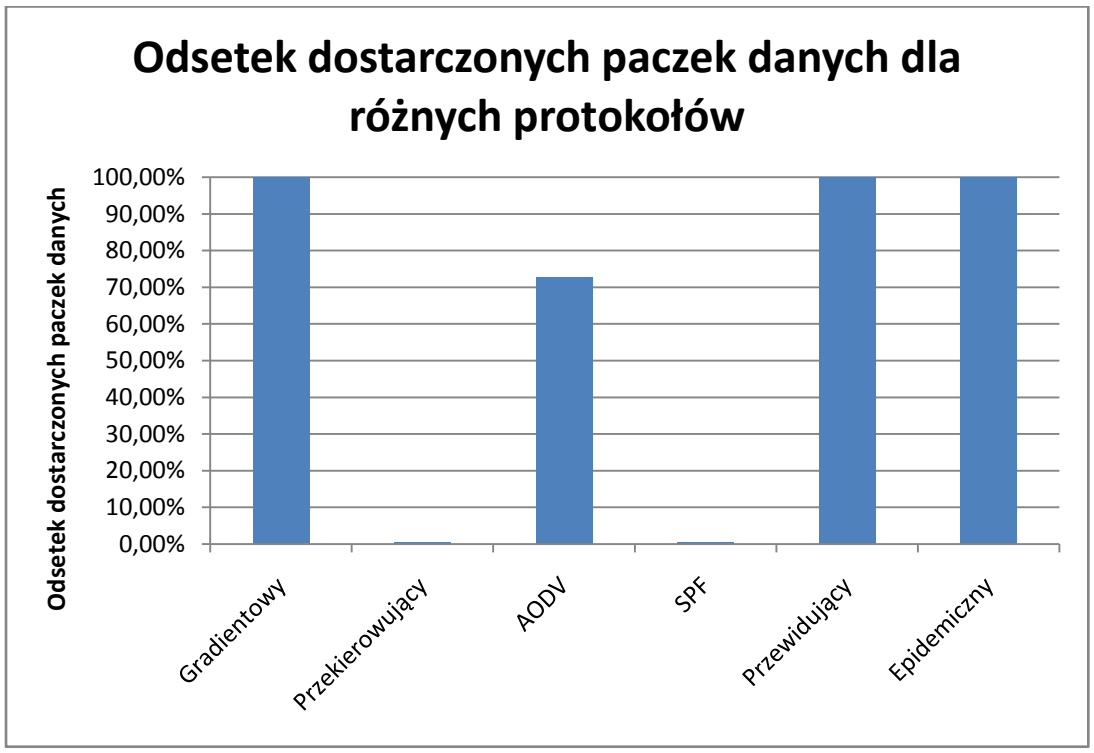
4.2.2.3.2 Opóźnienie dostarczania paczki danych

Tabela 4-13 przedstawia zestawienie danych charakteryzujących opóźnienie dostarczania paczki danych w sieci. Rysunek 4-25 przedstawia graficznie średnie opóźnienie dostarczania paczki danych dla różnych protokołów rutingu.

Widać, że najlepszy rezultat osiągnęły protokoły przekierowujący oraz SPF. Ich bardzo dobry wynik został jednak osiągnięty sztucznie – protokoły te odrzucały dane, gdy nie istniało

Tabela 4-12: Informacje o wygenerowanych i dostarczonych danych dla różnych protokołów rutingu w sieci z mułami danych.

Protokół rutingu	Liczba wygenerowa-nych paczek danych	Liczba dostarczonych paczek danych	Liczba nie odnalezio-nych rut	Odsetek dostarczonych paczek danych
Gradientowy	2975709	2975671	0	100,00%
Przekierowujący	2974016	12121	2963556	0,41%
AODV	2969577	2159972	1169032	72,74%
SPF	2968607	12598	2956304	0,42%
Przewidujący	2984438	2984435	0	100,00%
Epidemiczny	2975567	2975565	0	100,00%



Rysunek 4-24: Odsetek dostarczonych danych dla różnych protokołów rutingu w sieci z mułami danych.

bezpośrednie połączenie między węzłami początkowym i docelowym. Dlatego pakiety nigdy nie oczekiwały w buforze wyjściowym węzłów.

Najlepszymi protokołami pod względem rozmiaru opóźnień okazały się protokoły przewidujący oraz epidemiczny. Ich rezultat jest prawie identyczny i równy minimalnemu opóźnieniu, jakie mogło być osiągnięte w sieci. Mają też one ten sam odsetek dostarczonych danych. Aby lepiej móc porównać te protokoły, konieczne jest uwzględnienie dodatkowych współczynników, szczególnie narzutu transmitowanych danych.

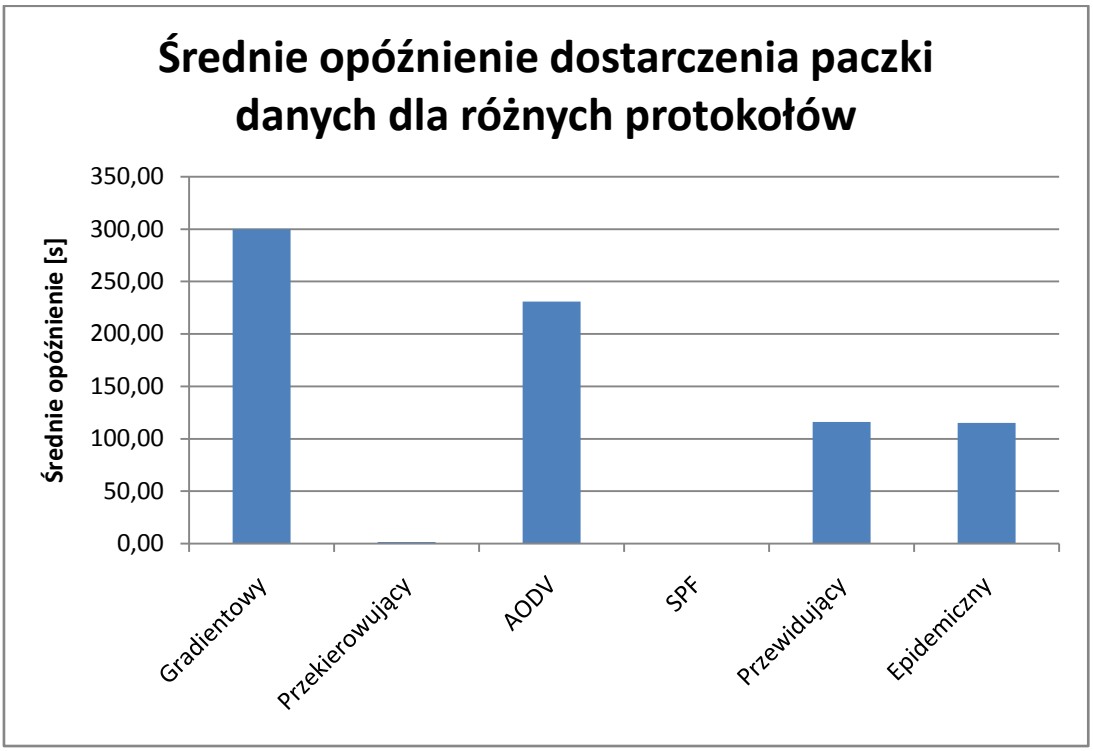
Protokoły AODV oraz gradientowy uzyskały najgorsze wyniki. Protokół AODV zyskuje przewagę nad protokołem gradientowym dzięki możliwości skierowania danych do celu wieloma drogami. Natomiast protokół gradientowy czeka po prostu aż dane łącze zacznie być zdatne. Nie zmienia raz wybranej trasy nawet jeśli istnieją inne, lepsze drogi.

4.2.2.3.3 Transmitowane dane

Tabela 4-14 przedstawia zestawienie liczby transmitowanych bajtów w sieci dla różnych protokołów rutingu. Rysunek 4-26 przedstawia graficznie ilość transmitowanych danych dla

Tabela 4-13: Opóźnienie dostarczania paczki danych dla różnych protokołów rutingu w sieci z mułami danych.

Protokół rutingu	Średnie opóźnienie [s]	Odchylenie standardowe [s]	Ufność [s]
Gradientowy	300,00	185,02	0,210
Przekierowujący	1,57	20,85	0,371
AODV	231,02	200,63	0,268
SPF	0,27	4,98	0,087
Przewidujący	115,98	80,94	0,092
Epidemiczny	115,15	69,44	0,079



Rysunek 4-25: Średnie opóźnienie dostarczania paczki danych dla różnych protokołów rutingu w sieci z mułami danych.

różnych protokołów rutingu.

Liczba transmitowanych danych w sieci z założenia nie jest istotnym czynnikiem w badanym rodzaju sieci. Widać jednak, że występuje tutaj duża różnica pomiędzy podobnymi dotychczas protokołami tj. przewidującym i epidemicznym. Protokół epidemiczny potrzebował wysłać przez łącze prawie pięć razy więcej danych niż protokół przewidujący.

Protokoły przekierowujący oraz SPF uzyskały najlepsze rezultaty, jednak po raz kolejny ich dobry wynik związany jest z odrzucaniem pakietów, gdy nie istnieje bezpośrednie połączenie między węzłami początkowym i docelowym.

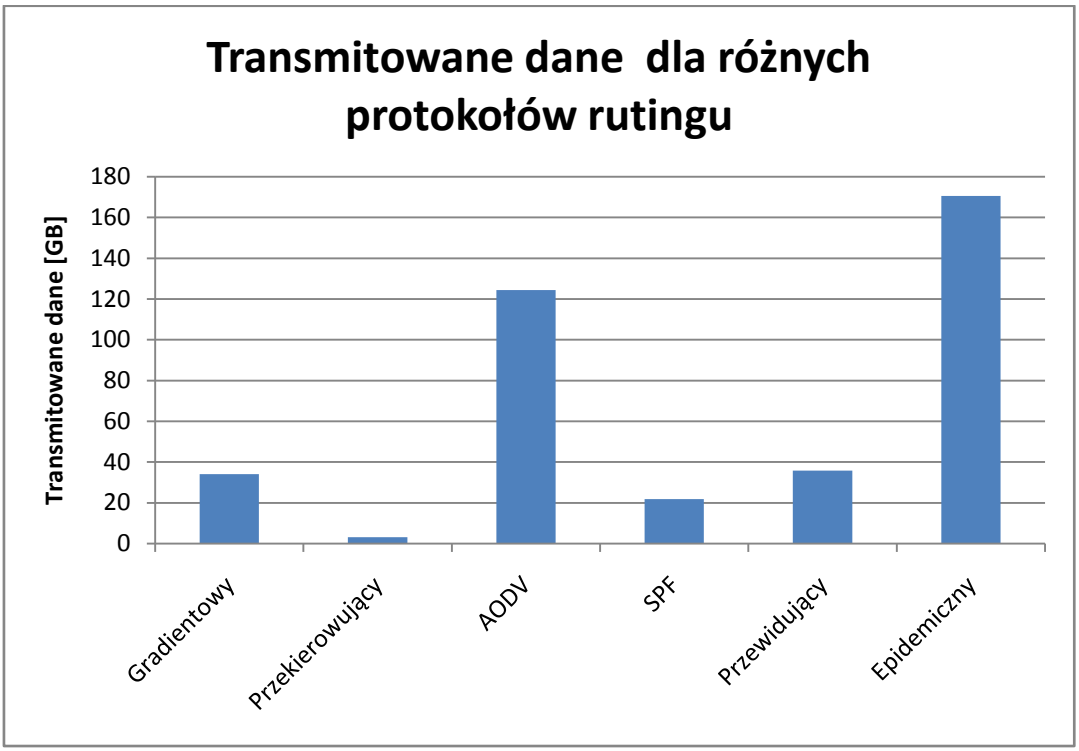
Protokół AODV pokazuje kolejną wadę — znaczny narzut wiadomości związanych z rutingiem.

4.2.2.3.4 Wnioski

Najlepszym protokołem w sieciach z mułami danych okazał się protokół przewidujący. Bardzo

Tabela 4-14: Transmitowane dane dla różnych protokołów rutingu w sieci z mułami danych.

Protokół rutingu	Transmitowane dane [B]
Gradientowy	34111214604
Przekierowujący	3139382310
AODV	124443240530
SPF	21816265187
Przewidujący	35883441788
Epidemiczny	170508307757



Rysunek 4-26: Transmitowane dane dla różnych protokołów rutingu w sieci z mułami danych.

dobry rezultat uzyskał także protokół epidemiczny, który może być z powodzeniem zastosowany w sieciach, gdzie duży narzut transmitowanych danych nie ma znaczenia. Jego dodatkowymi zaletami mogą być prostota implementacji oraz brak konieczności gromadzenia wiedzy o czasie występowania wyłączeń.

4.2.2.4 Sieci z bardzo rzadkimi i jednocześnie nieprzewidywalnymi kontaktami między węzłami

W sieci z bardzo rzadkimi i jednocześnie nieprzewidywalnymi kontaktami między węzłami średni czas dostępności łącza w sieci wynosił 1 s. Średni czas niedostępności (awarii) wynosił 100 s. Łatwo stwierdzić, że łącze było dostępne przez zaledwie 1% czasu. Przepływność łączy została ustawiona na 10^5 B/s. Źródło danych generuje 10^4 B co 200 s. Czas symulacji został ustawiony na 5×10^6 s. Czas retransmisji został ustawiony na 10^3 s, co gwarantuje, że stan sieci w momencie retransmisji jest niezależny od stanu w momencie pierwszej, nieudanej próby wysłania paczki danych.

Dla symulacji istotna była konfiguracja symulacji protokołu AODV. Przy przyjętym podejściu protokół oczekuje do 200 s dla pojedynczego pakietu na odnalezienie drogi.

4.2.2.4.1 Odsetek dostarczonych danych

Tabela 4-15 zawiera ilość wygenerowanych i dostarczonych do celu danych. Rysunek 4-27 prezentuje graficznie odsetek dostarczonych danych dla różnych protokołów rutingu.

Protokół SPF w czasie działania całej symulacji dostarczył do celu tylko jeden pakiet. Protokół przekierowujący także uzyskał bardzo słaby rezultat. Protokoły te zdecydowanie nie sprawdzają się w przedstawionym scenariuszu. Protokół AODV uzyskał średni rezultat. Pomimo trzech prób wysłania każdej paczki danych udało mu się dostarczyć tylko 84% danych.

Tabela 4-15: Wygenerowane i dostarczone dane dla różnych protokołów rutingu w sieci z rzadkimi i nieprzewidywalnymi kontaktami między węzłami.

Protokół rutingu	Liczba wygenerowanych danych [B]	Liczba dostarczonych danych [B]	Odsetek dostarczonych danych
Gradientowy	252255085	251842849	99,84%
Przekierowujący	249234770	42110	0,02%
AODV	247887560	208729098	84,20%
SPF	245987340	4096	0,00%
Przewidujący	250249172	249879916	99,85%
Epidemiczny	252192347	252192207	100,00%

Prawdopodobnie rezultat można by poprawić zwiększając liczbę retransmisji, co zwiększyłoby czas dostarczenia danych do celu, ale wprowadziłoby duży narzut transmitowanych wiadomości (retransmisje).

Protokoły gradientowy, przewidujący oraz epidemiczny uzyskały bardzo podobne rezultaty, bliskie lub równe 100%. Najprawdopodobniej fakt, że protokoły gradientowy i przewidujący nie uzyskały 100% jest spowodowany ciągle transportowanymi paczkami danych, tzn. w momencie przerwania symulacji część pakietów była jeszcze transportowana przez sieć.

Podany scenariusz nie pozwolił sprawdzić efektywności protokołów w sieci bardzo podobnej do *Zebranet*. W takiej sieci problemem było w ogóle dostarczenie danych do węzła docelowego. Tymczasem w badanym scenariuszu czas symulacji był tak duży, że wcześniej czy

Rysunek 4-27: Odsetek dostarczonych danych dal różnych protokołów rutingu w sieci o rzadkich i nieprzewidywalnych kontaktach między węzłami.

później każde łącze było naprawiane i otwierała się droga do kolejnego węzła na drodze. Można by rozważyć inny rodzaj symulacji — o czasie symulacji porównywalnym z czasem niedostępności łącza. Symulacja byłaby przeprowadzana wielokrotnie, a później można by sprawdzić odsetek przypadków, kiedy protokołowi rutingu udało się odnaleźć drogę do celu w czasie symulacji. W opisanym scenariuszu prawdopodobnie protokół epidemiczny uzyskałby znacznie lepszy rezultat niż pozostałe protokoły.

4.2.2.4.2 Średnie opóźnienie

Tabela 4-16 zawiera zestawienie danych związanych z opóźnieniem dostarczania paczki danych do węzła docelowego. Rysunek 4-28 przedstawia graficznie średnie opóźnienie dostarczania paczki danych dla różnych protokołów.

Protokół SPF uzyskał najlepszy średni czas dostarczenia porcji danych do węzła docelowego. Jednak jego rezultat powstał w sposób sztuczny — w czasie symulacji tylko raz powstało bezpośrednie połączenie pomiędzy węzłem początkowym i docelowym i wygenerowany w tym momencie pakiet został błyskawicznie przetransportowany do węzła docelowego.

Najgorszy rezultat uzyskał protokół przekierowujący, jednak ze względu na bardzo małą liczbę dostarczonych paczek danych trudno jest uznać ten rezultat za wiarygodny. Protokół przekierowujący i tak nie może być używany w tego rodzaju sieci ze względu na mały odsetek dostarczonych danych.

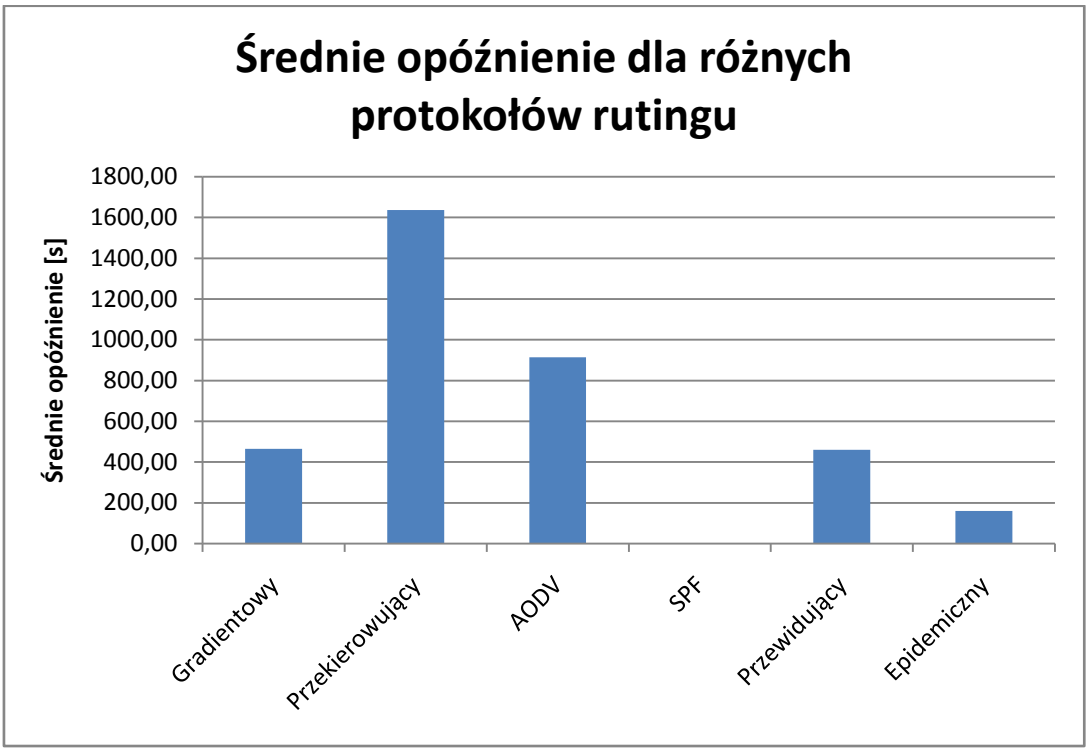
Rezultat osiągnięty przez AODV pokazuje kolejną wadę zastosowania tego protokołu w sieci o rzadkich kontaktach między węzłami.

Protokoły gradientowy i przewidujący uzyskały relatywnie dobry rezultat, który pozwala zastanawiać się nad zastosowaniem tych protokołów w niektórych rodzajach sieci o rzadkich kontaktach pomiędzy węzłami.

Zdecydowanie najlepszy rezultat uzyskał protokół epidemiczny. Jego wynik jest prawie trzykrotnie lepszy niż protokołów gradientowego i przewidującego. Rezultat ten został osiągnięty dzięki próbom wysłania danych wszystkimi istniejącymi ścieżkami równolegle. Paczka danych trafiała do celu tą trasą, na której momenty pojawienia się zdatności łączy dawały możliwość dostarczenia pakietu w najkrótszym czasie.

Tabela 4-16: Opóźnienie dostarczania paczki danych dla różnych protokołów rutingu w sieci z rzadkimi i nieprzewidywalnymi kontaktami między węzłami.

Protokół rutingu	Średnie opóźnienie [s]	Odchylenie standardowe [s]	Ufność [s]
Gradientowy	464,5292017	401,2188704	2,874040792
Przekierowujący	1636,593202	504,5650008	285,4792784
AODV	914,5589714	767,1048307	6,035905263
SPF	0,13203	0	---
Przewidujący	460,7761775	398,7740974	2,867735445
Epidemiczny	159,8885496	86,75622517	0,621031562



Rysunek 4-28: Średnie opóźnienie dostarczania paczek danych dla różnych protokołów rutingu w sieci o rzadkich i nieprzewidywalnych kontaktach między węzłami.

4.2.2.4.3 Wnioski

Najlepszym protokołem rutingu w sieci o rzadkich i nieprzewidywalnych kontaktach między węzłami jest protokół epidemiczny. Nie tylko dostarcza dane niezawodnie do celu, ale także robi to w najkrótszym czasie. Niestety, protokół ten jest absolutnie nieskalowalny. W dużych sieciach jako alternatywę można rozważyć protokół gradientowy i przewidujący.

4.2.2.5 Sieci międzyplanetarne

Czas propagacji sygnału z Ziemi do Marsa waha się w przedziale 4-20 min. Jako punkt odniesienia można uznać 10 min. jako rząd czasu propagacji, jaki występuje w sieciach międzyplanetarnych. Dlatego czas propagacji wszystkich łączy z dużym opóźnieniem zostaje ustawiony na 600 s. Okresy awarii łączy z założenia mają mieć wartość kilkakrotnie większą od czasu propagacji łącza. Dlatego średni okres awarii łącza zostaje ustawiony na 5×10^3 s, a średni okres sprawnego działania na wartość 2×10^4 s, co gwarantuje średnią dostępność łączy na poziomie 80% czasu. Satelita znajdujący się na niskiej orbicie okołoziemskiej potrzebuje około 90 min. na okrążenie planety. Dlatego okres włączenia każdego z łączy między satelitą a planetą zostaje ustawiony na 30 min. Czas propagacji na tych łączach jest względnie niski i zostaje ustawiony na 10^{-3} s. Przepływności wszystkich łączy zostały ustawione na 10^5 B/s.

Czas retransmisji powinien być większy niż suma czasów propagacji wszystkich łączy na drodze od węzła źródłowego do docelowego i z powrotem (lub do najbliższego węzła przejmującego odpowiedzialność za retransmisję) oraz sumy średnich okresów niedostępności występujących na poszczególnych łączach. Dzięki odsunięciu retransmisji w czasie można uzyskać pewność, że stan sieci w momencie retransmisji będzie niezależny od stanu sieci w momencie pierwszej transmisji. Dla sieci międzyplanetarnej czas ten został ustalony na 300 minut ($1{,}8\times10^4$ s).

Teoretyczna przepływność możliwa do uzyskania w sieci jest bliska 10^5 B/s (przy założeniu, że protokół rutingu będzie w stanie odnaleźć drogę w istniejącej strukturze sieci). Aby zapewnić odpowiednią rezerwę, zostaje przyjęte, że węzeł źródłowy generuje średnio 100 B/s. W praktyce zostaje to zrealizowane poprzez generację średnio 10^4 B co 100 s.

4.2.2.5.1 Transmitowane dane

Tabela 4-17 przedstawia zestawienie liczby transmitowanych przez poszczególne protokoły bajtów. Rysunek 4-29 przedstawia graficznie ilość transmitowanych danych przez poszczególne protokoły rutingu.

Zgodnie z założeniami symulacji, wszystkie paczki danych są wielokrotnie retransmitowane, dlatego każdy z protokołów rutingu osiągnął 100% skuteczności dostarczania danych do węzła docelowego. Każdy z nich osiągnął to jednak innym kosztem. W kontekście sieci międzyplanetarnych najważniejszym czynnikiem jest zmniejszenie liczby transmitowanych danych. Jak widać na powyższym wykresie, protokoły epidemiczny i przekierowujący mają bardzo duży narzut transmitowanych informacji. Przyczyną narzutu pierwszego jest zwielokrotnianie pojedynczego pakietu *Bundle*. Liczba zwielokrotnianych pakietów jest proporcjonalna do liczby łączy występujących w sieci. Ponieważ w sieci występują 24 łącza, więc narzut wysyłanych informacji jest bardzo duży. W przypadku protokołu przekierowującego przyczyną dużego narzutu są wielokrotne retransmisje. Pierwsze połączenie pomiędzy planetą a satelitą pakiet kierowany przez ten protokół pokonuje bez problemów, ponieważ zawsze dostępne jest jedno z istniejących łączy do któregoś satelity. W 4% przypadków pakiet jest odrzucany w satelicie — gdy oba łącza do ruterów są uszkodzone. W mniej niż 1% przypadków pakiet jest odrzucany na przekaźniku — gdy wszystkie trzy łącza do satelitów są uszkodzone. Gdy pakiet trafia do satelity, to istnieje 67% szansy, że jest to satelita, który nie może aktualnie transmitować danych do odbiornika na planecie. Dlatego w 67% przypadków pakiet jest odrzucany.

Zdecydowanie najlepszy rezultat osiągnęły protokoły przewidujący oraz gradientowy. Różnica między nimi jest nieistotna.

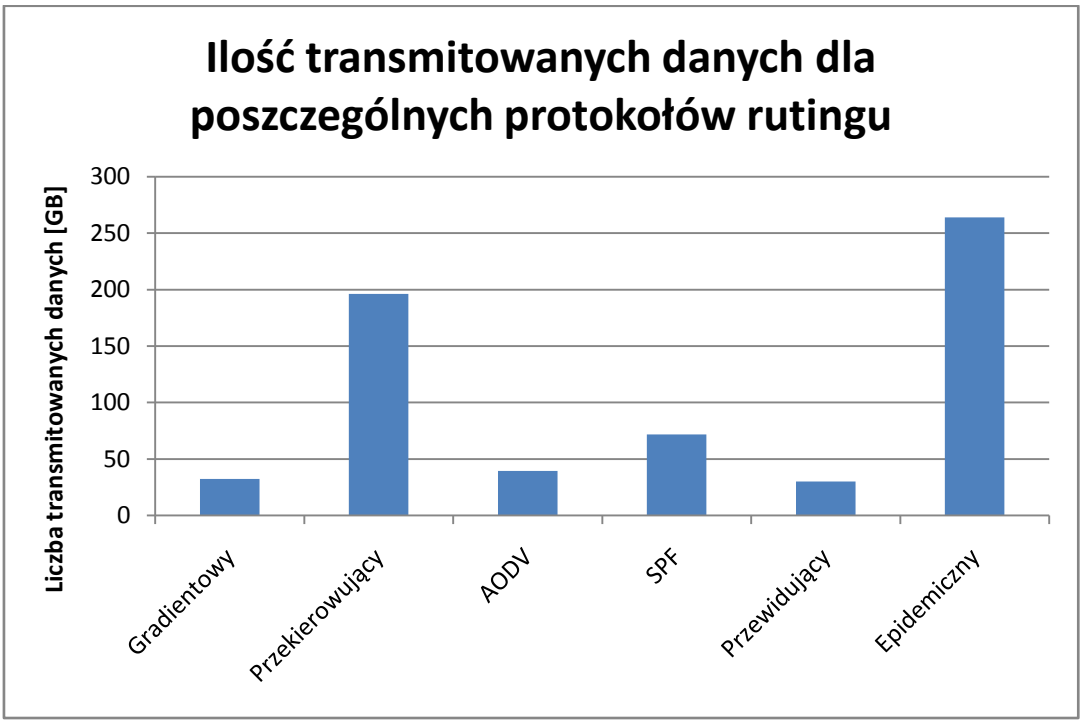
4.2.2.5.2 Opóźnienia

Tabela 4-18 prezentuje średnie opóźnienie dostarczenia paczki danych w sieci dla różnych protokołów. Rysunek 4-30 przedstawia średnie opóźnienie dostarczania paczki danych dla różnych protokołów rutingu.

Wyjątkowo oś pionowa przedstawiona została w skali logarytmicznej ze względu na bardzo

Tabela 4-17: Transmitowane dane dla różnych protokołów rutingu w sieci międzyplanetarnej.

Protokół rutingu	Transmitowane dane [B]
Gradientowy	32203348931
Przekierowujący	196228178213
AODV	39544929802
SPF	71745931251
Przewidujący	30108001022
Epidemiczny	263949054562



Rysunek 4-29: Transmitowane dane dla różnych protokołów rutingu w sieci międzyplanetarnej.

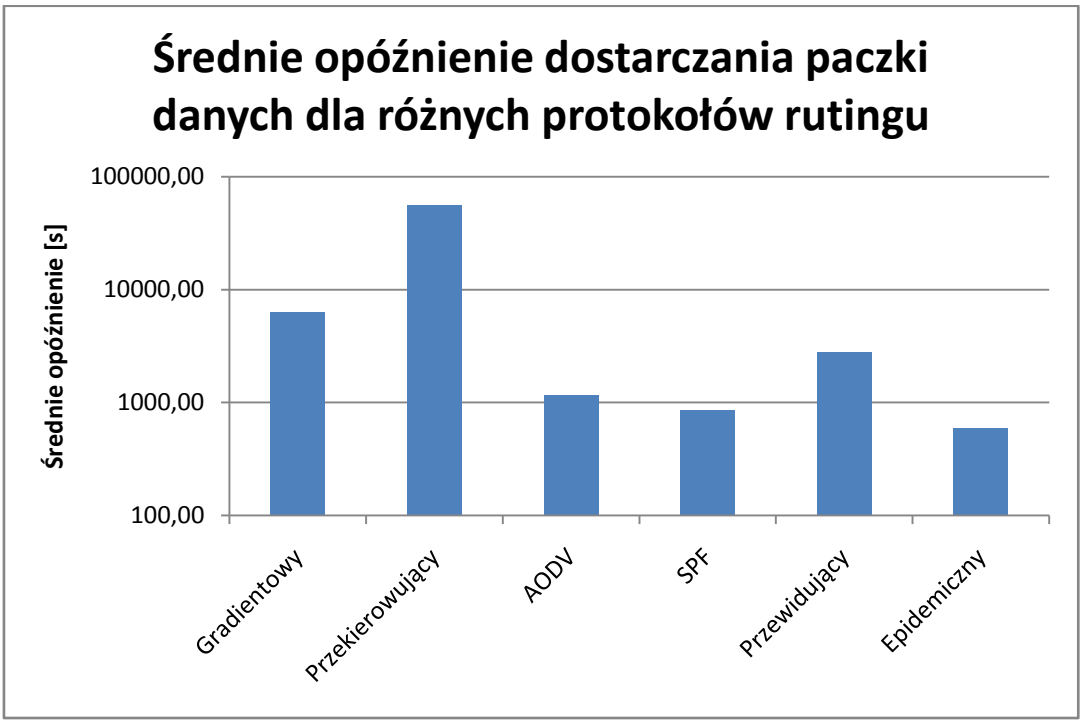
duży rozrzut uzyskanych wyników.

Najlepszy rezultat osiągnął protokół epidemiczny, jednak narzut transmitowanych informacji przekreśla ten protokół w zastosowaniach międzyplanetarnych. Nieco lepszy rezultat uzyskały protokoły AODV oraz SPF. Oba osiągnęły średnie rezultaty pod względem liczby transmitowanych danych. Tutaj pokazują swoje zalety — kosztem liczby transmitowanych danych zmniejszają średnie opóźnienie dostarczania wiadomości.

Protokoły przewidujący oraz gradientowy uzyskały nie najlepsze rezultaty pod względem opóźnienia występującego w sieci. Jednak znaczne mniejsza liczba transmitowanych danych nadal sprawia, że to te dwa protokoły wydają się najlepsze dla sieci międzyplanetarnych. Protokół przewidujący wydaje się nieco lepszy od protokołu gradientowego jeśli chodzi o średnie opóźnienie (występuje również minimalna różnica w liczbie transmitowanych danych na korzyść protokołu przewidującego). Protokół przekierowujący uzyskał bardzo słaby rezultat spowodowany koniecznością wielokrotnych retransmisji. Jest on zupełnie nieprzydanych w

Tabela 4-18: Opóźnienie dostarczania paczki danych dla różnych protokołów rutingu w sieci międzyplanetarnej.

Protokół rutingu	Średnie opóźnienie [s]	Odchylenie standardowe [s]	Ufność [s]
Gradientowy	6270,28	4641,09	2,9400
Przekierowujący	56759,14	73000,91	18,7337
AODV	1161,56	1481,26	0,8468
SPF	850,57	1525,51	0,6474
Przewidujący	2771,88	4123,88	2,7017
Epidemiczny	600,57	21,75	0,0048



Rysunek 4-30: Średnie opóźnienie dostarczania paczki danych dla różnych protokołów rutingu w sieci międzyplanetarnej.

kontekście sieci międzyplanetarnych.

4.2.2.5.3 Wnioski

Najlepszymi protokołami w kontekście sieci międzyplanetarnych okazały się protokoły, które nie transmitują żadnych informacji związanych z rutingiem, tylko na podstawie z góry danej wiedzy o sieci wyszukują optymalną drogę, a później oczekują na naprawienie lub włączenie poszczególnych łączy.

Interesujące rezultaty dały także protokoły AODV oraz SPF, które narzut transmitowanych informacji zamieniają na znaczne zmniejszenie średniego opóźnienia dostarczania paczek danych w sieci. Protokoły te mogą być rozważane w sieciach, gdzie wartość opóźnienia ma duże znaczenie.

4.2.2.6 Duże opóźnienia

Przepływności wszystkich łączy w sieci zostały ustawione na wartość 10^5 B/s. Każde źródło danych generowało średnio 10^4 B średnio co 10 s, co dawało średnią liczbę wygenerowanych danych na poziomie 10^3 B/s. Czasy propagacji poszczególnych łączy w sieci zostały ustawione na wartość losową z przedziału 100-1000 s. Istotnym elementem konfiguracji protokołów był czas, po jakim protokół AODV uznawał znalezioną trasę za nieaktualną. Wartość ta wynosiła 100 s. Czas symulacji po kilku próbach ustawiony został na 10^6 s. Tym razem to protokół epidemiczny był tym, którego symulacja trwała zdecydowanie najdłużej.

Ze względu na charakterystykę sieci, wszystkie protokoły uzyskały 100% skuteczność dostarczania danych do celu. Retransmisje nigdy nie były konieczne.

4.2.2.6.1 Transmitowane dane

Tabela 4-19 przedstawia zestawienie ilości transmitowanych danych dla poszczególnych testowanych protokołów rutingu. Rysunek 4-31 prezentuje graficznie ilość transmitowanych danych dla poszczególnych protokołów rutingu.

Łatwo zauważyć, że — poza protokołem epidemicznym — wszystkie protokoły wysłały praktycznie taką samą ilość danych. Narzut wiadomości wysyłanych przez protokoły AODV oraz SPF był nieistotny. Natomiast protokół epidemiczny wysłał około 8 razy więcej danych niż pozostałe protokoły. Stąd wniosek, że protokół epidemiczny nie powinien być stosowanych w sieciach, gdzie głównym problemem są duże opóźnienia łączy, natomiast rzadkością są przestoje elementów.

4.2.2.6.2 Opóźnienie

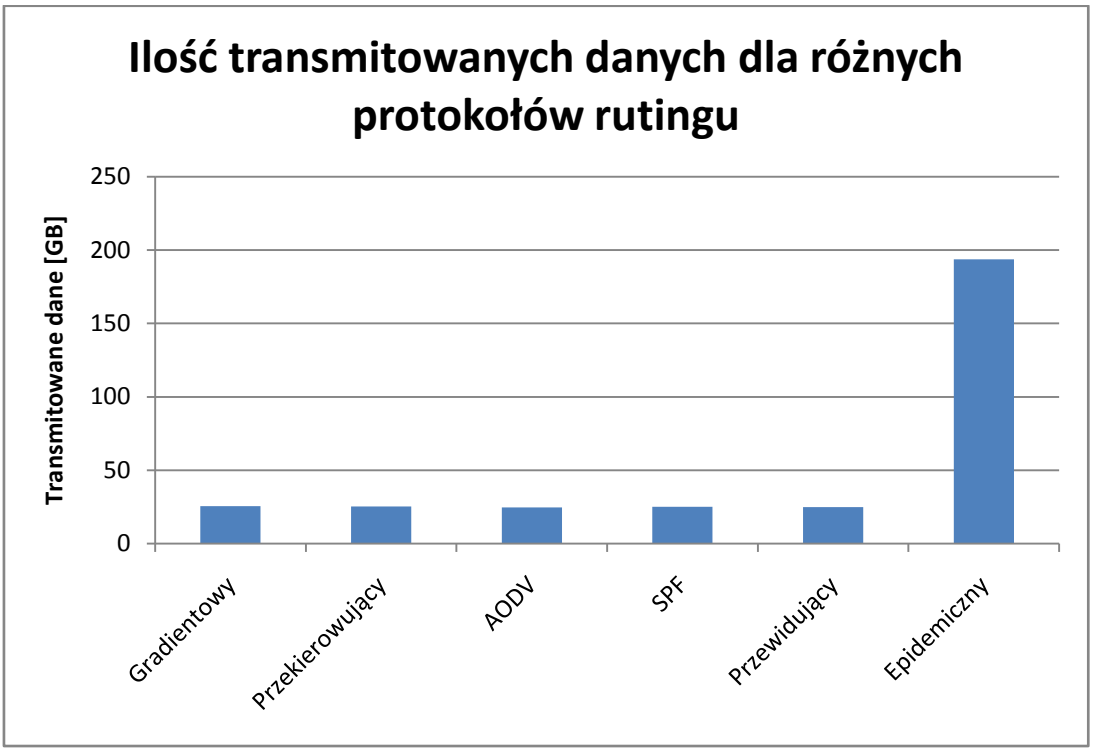
Tabela 4-20 przedstawia zestawienie statystyk związanych z opóźnieniami dostarczania pakietu protokołu *Bundle* w badanej sieci. Oczywiście ze względu na 100% skuteczność dostarczania danych opóźnienie dostarczania pakietu *Bundle* jest równoważne z opóźnieniem dostarczania paczki danych. Rysunek 4-32 przedstawia graficznie wartości średniego opóźnienia dostarczania pakietu *Bundle*.

Widać, że wszystkie protokoły uzyskały bardzo zbliżone rezultaty. Dodatkowy narzut informacji wysyłany przez protokoły SPF oraz epidemiczny w żaden sposób nie poprawił efektywności protokołu.

Bardzo ciekawy jest rezultat uzyskany przez protokół AODV. Średni czas dostarczenia pakietu *Bundle* w sieci wyniósł dla wszystkich protokołów prawie 900 s. Intuicja podpowiada, że gdy protokół AODV szuka drogi dla pakietu, to najpierw rozsyła pakiet *RREQ*, a później oczekuje na otrzymanie wiadomości *RREP* od węzła docelowego. Dlatego wydaje się, że średnie opóźnienie dostarczenia pakietu *Bundle* w sieci powinno wynieść około trzykrotną wartość opóźnienia dla innych protokołów. Jednak wyniki wyraźnie pokazują, że protokół ten osiąga czas opóźnienia tylko o około 1% gorszy od pozostałych protokołów. Aby zrozumieć, co się dzieje w sieci, musimy przypomnieć sobie, w jaki sposób protokół AODV szuka drogi do celu. Jest to realizowane poprzez rozgłaszanie pakietu *RREQ*. Otrzymanie pakietu *RREQ* sprawia, że wszystkie węzły w sieci zapamiętują drogę do węzła rozgłaszającego. Jedno poszukiwanie drogi do dowolnego celu przez danych węzeł oznacza, że cała sieć zostanie poinformowana o najlepszej drodze do węzła szukającego drogi. Kiedy dany węzeł próbuje wysłać jakiś pakiet *Bundle* przez sieć, to najprawdopodobniej w jego tablicy rutingu będzie już wpis o drodze do węzła docelowego, który powstał po tym, jak węzeł docelowy rozgłosił pakiet *RREQ*. W

Tabela 4-19: Transmitowane dane dla różnych protokołów rutingu w sieci o dużych opóźnieniach.

Protokół rutingu	Transmitowane dane [B]
Gradientowy	25485145555
Przekierowujący	25304078930
AODV	24706434845
SPF	24994617456
Przewidujący	24765773380
Epidemiczny	193723299999



Rysunek 4-31: Transmitowane dala dla różnych protokołów rutingu w sieci o dużych opóźnieniach.

badanej sieci działanie protokołu AODV polega na systematycznym rozgłaszaniu przez węzły swojej pozycji. W takich warunkach protokół AODV zachowuje się podobnie do protokołu SPF.

4.2.2.6.3 Wnioski

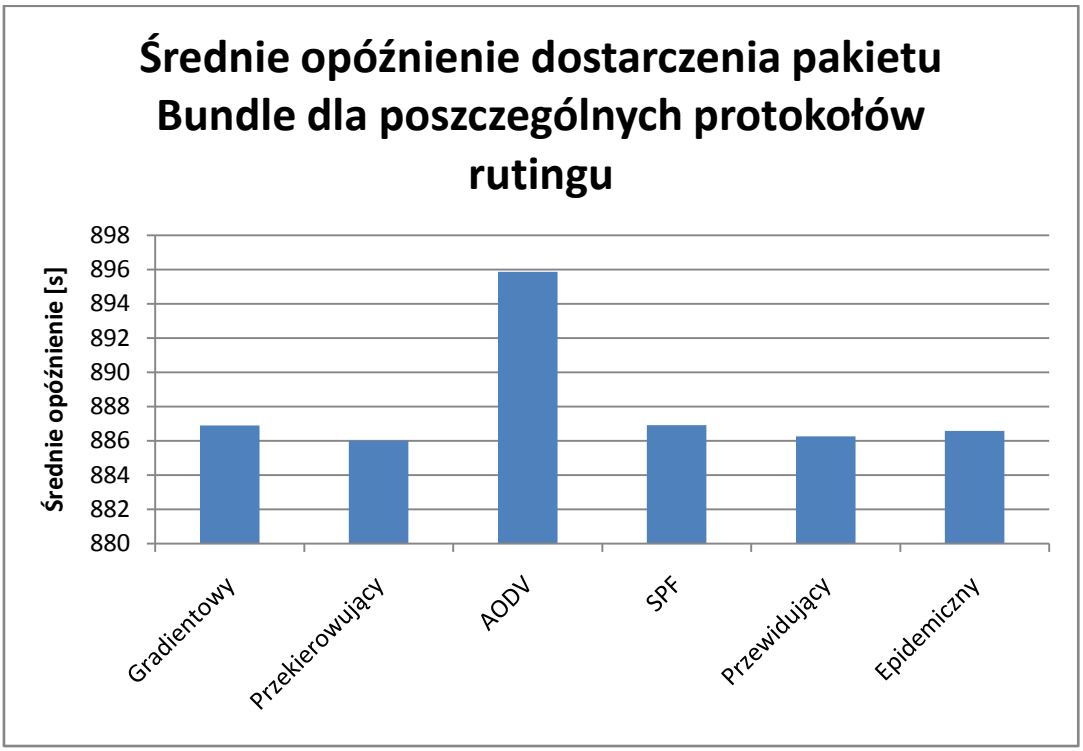
Spośród badanych protokołów trudno jest wyłonić najlepszy, ponieważ wiele protokołów uzyskało bardzo podobny rezultat. Na pewno nie należy stosować protokołu epidemicznego, który w kontekście sieci z małą liczbą przestojów nie sprawdza się ze względu na bardzo duży narzut i jednocześnie brak możliwości zwiększenia efektywności poprzez zwielokrotnianie pakietów.

Za najlepsze uznać można ex aequo protokoły gradientowy, przekierowujący, przewidujący oraz SPF.

Zachowanie protokołu AODV sugeruje możliwość opracowania jeszcze jednego nowego

Tabela 4-20: Opóźnienie dostarczania pakietu Bundle dla różnych protokołów rutingu w sieci o dużych opóźnieniach.

Protokół rutingu	Średnie opóźnienie[s]	Odchylenie standardowe [s]	Ufność [s]
Gradientowy	886,90	445,37	0,505
Przekierowujący	886,00	445,49	0,504
AODV	895,86	450,04	0,510
SPF	886,92	445,29	0,504
Przewidujący	886,26	445,66	0,504
Epidemiczny	886,59	445,37	0,504



Rysunek 4-32: Średnie opóźnienie dostarczania pakietu Bundle dla różnych protokołów rutingu w sieci o dużych opóźnieniach.

podejścia do rutingu, w którym węzeł cyklicznie rozgłaszałby pakiet z informacją o rutingu. Jego otrzymanie przez dany węzeł byłoby równoznaczne z utworzeniem wpisu w tablicy rutingu w tym węźle. Łącze, po którym dotarł pakiet z informacją o rutingu, byłoby traktowane jako najlepsza droga (jeśli ten sam pakiet dotarłydo węzła po kilku łączach, to łącze, po którym dotarł jako pierwszy, jest łączem o najmniejszym opóźnieniu i przez to najlepsze). Taki protokół nie był symulowany i dlatego trudno jest ocenić jego efektywność w kontekście sieci *DTN*.

4.2.2.7 Wnioski końcowe

Przeprowadzone symulacje jednoznacznie pokazały, że nie istnieje jeden protokół rutingu, który mógłby być używany w każdej sieci *DTN*. W zależności od parametrów sieci różne protokoły osiągają różne rezultaty.

4.2.2.7.1 Protokół gradientowy

Protokół gradientowy okazał się protokołem odpornym na występowanie dużych opóźnień w sieci. Natomiast nie jest odporny na częste przestoje łączy i węzłów. Dane wysłane tym protokołem bardzo często oczekują w buforze wyjściowym łącza na koniec przestoju łącza lub węzła, który powinien być następnym elementem na drodze do węzła docelowego.

Protokół gradientowy nie radzi sobie także z dużą liczbą danych transportowanych w sieci, ponieważ nie obsługuje w żaden sposób dzielenia strumienia pakietów na różne łącza (ang. *load balancing*). Przy dużej liczbie danych duży odsetek pakietów jest odrzucany.

4.2.2.7.2 Protokół przekierowujący

Protokół przekierowujący bardzo często dawał bardzo słabe wyniki. Głównym problemem było zbyt częste odrzucanie paczek danych. Protokół przekierowujący radzi sobie wyłącznie z

sytuacjami, w których istnieje bezpośredni kontakt między węzłem początkowym a docelowym. Kiedy pakiet *Bundle* dociera do węzła, z którego w danym momencie nie można transmitować danych, to pakiet jest odrzucany.

Protokół przekierowujący uzyskiwał lepsze czasy opóźnienia dostarczania pakietu *Bundle* od protokołu gradientowego. Było to spowodowane tym, że kiedy pakiet docierał do węzła, z którego najlepsza droga prowadząca do celu była niedostępna, to protokół gradientowy oczekiwał na dostępność najlepszej drogi, a protokół przekierwujący kierował pakiet mniej korzystną drogą. Zazwyczaj było to opłacalne.

Można wyobrazić sobie protokół rutingu, który byłby kombinacją protokołów gradientowego i przekierowującego. Protokół taki kierowałby pakiety do celu najlepszą drogą (o największym gradiencie) spośród tych łączy, które są dostępne. Jeżeli żadna droga nie byłaby dostępna, to protokół wybierałby łącze o największym gradiencie i umieszczał pakiet w buforze wyjściowym wybranego łącza, gdzie czekałby na dostępność łącza.

4.2.2.7.3 Protokół AODV

Protokół AODV radził sobie najlepiej z sieciami, w których ilość transmitowanych danych była mała w porównaniu z możliwościami sieci. Nie radził sobie zupełnie tam, gdzie ilość transmitowanych danych była bardzo duża.

Co ciekawe, protokół AODV radził sobie zupełnie nieźle w sieci międzyplanetarnej oraz w sieci o dużych opóźnieniach, chociaż należy zauważyć, że działanie protokołu AODV w sieci o dużych opóźnieniach opiera się na innej zasadzie niż było to zakładane podczas projektowania tego protokołu dla sieci sensorowych. Niemniej metoda działania okazała się skuteczna.

4.2.2.7.4 Protokół SPF

Protokół SPF nie sprawdzał się przede wszystkim w sieciach, gdzie nie występował bezpośredni kontakt między węzłem początkowym i końcowym. Drugim scenariuszem, w którym SPF pokazał swoje wady, była sieć sensorowa. W sieci tej awarie występowały częściej niż generacja danych, a ponieważ protokół SPF wysyłał uaktualnienia po wykryciu każdej zmiany struktury sieci, więc narzut wysyłanych uaktualnień był wielokrotnie większy od wysyłanych danych.

Protokół SPF sprawdził się najlepiej tam, gdzie jest stosowany jego bardziej dopracowany odpowiednik, *OSPF* — w sieci internetowej.

4.2.2.7.5 Protokół przewidujący

Protokół przewidujący uzyskiwał rezultaty bardzo podobne do protokołu gradientowego. Różnice występowały tylko wtedy, kiedy w sieci miały miejsce przewidywalne wyłączenia elementów. Wtedy protokół przewidujący radził sobie lepiej od protokołu gradientowego i dlatego można uznać, że jest on lepszym protokołem niż protokół gradientowy. Protokół przewidujący szczególnie dobrze radzi sobie w sieciach z mułami danych.

4.2.2.7.6 Protokół epidemiczny

Protokół epidemiczny okazał się protokołem, który osiąga zdecydowanie najmniejsze opóźnienia i największą niezawodność dostarczania danych spośród wszystkich badanych

protokołów. Niestety, jest również protokołem, który ma największy narzut transmitowanych danych. Protokół epidemiczny sprawdzał się tam, gdzie przepływność łączy była dużo większa od średniej ilości danych, które były transmitowane przez te łącze oraz tam, gdzie narzut transmitowanych danych miał małe znaczenie. Lepiej radził sobie także w małych sieciach niż w dużych (wystarczy porównać sieć o strukturze sześcianu mającą 12 łączy i sieć międzyplanetarną o 24 łączach).

4.2.3 Wpływ przekazywania odpowiedzialności za retransmisję na wydajność sieci

Czas propagacji łączy ustawiony został na 0,001 s. Czas retransmisji ustawiony został na 1 s — jest to czas, po którym z pewnością dane zdążą zostać wysłane przez węzeł początkowy (dane mogą trafić na koniec pełnego bufora wyjściowego) i pokonać drogę do węzła docelowego, a jednocześnie jest to wartość znacznie mniejsza od czasu symulacji, który ustawiony został na 10^5 s. Węzeł początkowy generował średnio 10^4 B co 1 s, zaś przepływność wszystkich łączy wynosiła 10^6 B/s.

Podczas symulacji okazało się, że od *BER* równego 0,00002 w wariancie bez przejmowania odpowiedzialności za retransmisje do węzła docelowego zaczął docierać tylko niewielki odsetek wysyłanych danych. Powodem było to, że bardzo duży odsetek pakietów *UDP* był gubiony podczas przechodzenia do kolejnych węzłów i nawet przy wielokrotnych retransmisjach nie udało się dostarczyć danych do węzła docelowego. Dlatego dane od tego punktu są już niemiarodajne. Wariant z przejmowaniem retransmisji był wydajny w znacznie szerszym zakresie i w przedziale *BER* badanym w tym scenariuszu nie dało się zauważyć podobnego zjawiska. Podstawowy wniosek z tego faktu jest taki, że przy przejmowaniu odpowiedzialności za retransmisję sieć jest w stanie poradzić sobie ze znacznie większą wartością współczynnika *BER*.

4.2.3.1 Transmitowane dane

Tabela 4-21 przedstawia zestawienie ilości danych transmitowanych w sieci. Niemiarodajne

Tabela 4-21: Transmitowane dane dla wariantu z przejmowaniem odpowiedzialności za retransmisję i bez niego.

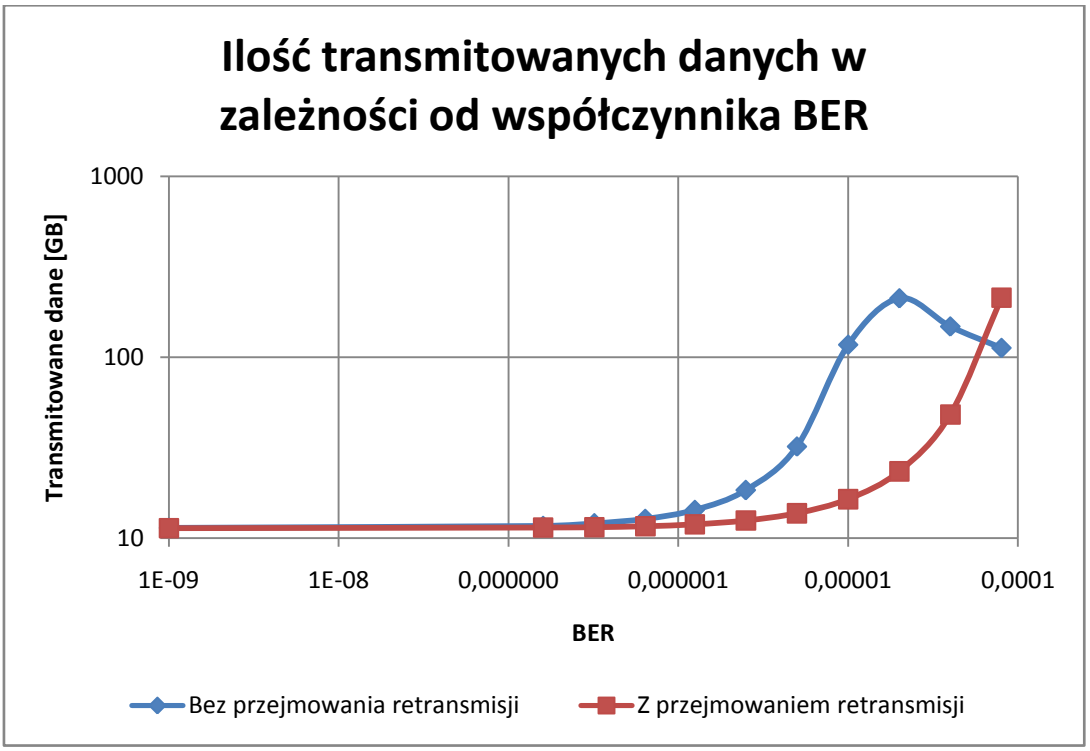
BER	Transmitowane dane [B]		Stosunek
	Bez przejmowania retransmisji	Z przejmowaniem retransmisji	
0,00008	112711818297	213698759315	0,527
0,00004	148108747545	48277171858	3,068
0,00002	211751564239	23439518994	9,034
0,00001	117186335689	16433488538	7,131
0,000005	32143710799	13741947959	2,339
0,0000025	18419722494	12536033503	1,469
0,00000125	14357672503	11948512176	1,202
0,00000064	12790043450	11669492536	1,096
0,00000032	12060899172	11525803409	1,046
0,00000016	11716284504	11453086732	1,023
0,000000001	11400192719	11384195616	1,001

dane zaznaczone zostały na szaro. Rysunek 4-33 prezentuje ilość transmitowanych danych w zależności od współczynnika *BER* dla obu wariantów.

Widać, że dla bardzo małych wartości współczynnika *BER* (tj. dla małej liczby pakietów gubionych podczas transportu przez sieć) wariant z przejmowaniem odpowiedzialności za retransmisję osiąga takie same rezultaty jak wariant bez przejmowania odpowiedzialności. Wraz ze wzrostem wartości współczynnika *BER* rośnie liczba transmitowanych danych dla obu wariantów, jednak tempo wzrostu dla wariantu z przejmowaniem odpowiedzialności jest wyraźnie niższe niż dla drugiego wariantu. W ostatnim wiarygodnym punkcie wariant z przejmowaniem odpowiedzialności za retransmisję potrzebował wysłać ponad siedem razy mniej danych aby osiągnąć ten sam efekt — dostarczenie pakietu *Bundle* do węzła docelowego oraz potwierdzenia od węzła docelowego do węzła źródłowego.

4.2.3.2 Wnioski

Badany scenariusz pokazał, że implementacja przekazywania odpowiedzialności za retransmisję w sieciach, gdzie paczki danych praktycznie nie są gubione, jest bezcelowa. W takich warunkach sieć radzi sobie równie dobrze w obu wariantach. Natomiast wraz ze wzrostem liczby gubionych pakietów zdecydowanie rośnie znaczenie przejmowania odpowiedzialności za retransmisję. Przy pewnym odsetku gubionych pakietów dostarczanie danych do węzła docelowego jest możliwe tylko wtedy gdy węzły w sieci przejmują odpowiedzialność za retransmisję pakietów.



Rysunek 4-33: Transmitowane dane w zależności od współczynnika BER dla wariantu z przejmowaniem odpowiedzialności za retransmisję i bez niego.

5 Podsumowanie

5.1 Symulacja

5.1.1 Model

Najważniejszym problem podczas tworzenia symulatora sieci *DTN* był brak dobrego modelu takiej sieci. Jeśli nawet istnieją pewne praktyczne rozwiązania, które funkcjonują w rzeczywistym świecie, to ich wewnętrzny sposób działania jest bardzo słabo opisany, zaś opracowanie teoretyczne jest praktycznie niedostępne. Brak dobrego modelu zaowocował oparciem się w tej pracy na niektórych rozwiązaniach znanych z sieci *Internet*, które nie zawsze okazywały się trafione. Efektem odkrycia problemu z brakiem powtórnego rutowania pakietów po wykryciu zmian w sieci jest propozycja zmiany organizacji rutowania dla sieci *DTN* opisana dalej.

Inne elementy modelu okazały się bardzo dobre. Sprawdził się m.in. model generowania opóźnień, model awarii i wyłączeń czy też modele działania poszczególnych protokołów rutingu. Bardzo dobry okazał się model wydarzeń zachodzących w sieci (opisana wcześniej klasa *Timer*).

5.1.2 Czas trwania

Czas symulacji bardziej złożonych scenariuszy często okazywał się bardzo duży. Praktyka pokazała, że symulacja o czasie symulacji rzędu 10^6 sekund trwała około godziny, czyli mniej niż 300 razy szybciej od czasu rzeczywistego. Aby zasymulować sieć składającą się ze 100 węzłów należałoby zaopatrzyć się w sprzęt komputerowy z górnej półki i dużą dawkę cierpliwości. Rozwiązaniem tego problemu wydaje się wprowadzenie uproszczeń do sieci.

Symulacja działania sieci *DTN* sprowadzała się do symulacji dwóch podstawowych elementów: protokołów transportowych oraz protokołów rutingu. W wielu symulacjach parametr *BER* był równy 0 i dane były niezawodnie dostarczane do węzła docelowego. Łącze działało wtedy jak idealny tunel dla wiadomości. W takiej sytuacji złożona i czasochłonna symulacja działania protokołu *LTP* lub innego protokołu transportowego była zbędna. Jedna z możliwości przyspieszenia symulacji polega na stworzeniu uproszczonego protokołu transportowego, którego zadaniem byłoby niezawodne dostarczanie danych przy niskim czasowym koszcie symulacji. Protokół taki mógłby następnie być stosowany w tych scenariuszach, których efekt nie zależał od precyzji symulacji protokołu transportowego.

5.2 Rozwiązania w sieci DTN

Doświadczenia z protokołami rutingu pokazują, że rozwiązania stosowane obecnie w sieci *DTN* ciągle jeszcze są mocno niedojrzałe. O ile udało się już dosyć dobrze zdefiniować protokół transportowy (*LTP*) oraz protokół warstwy sieci globalnej (*Bundle*), o tyle ciągle brakuje protokołu rutingu, który byłby standardem stosowanym w dużej części sieci. Wyniki zebrane w tej pracy mogą sugerować, że stworzenie takiego standardu może być trudne, jeśli nie niemożliwe.

Jednym z powodów niedojrzałości rozwiązań w sieci *DTN* jest mała liczba osób zajmujących się takimi sieciami. Dla upowszechnienia idei sieci *DTN* należałoby zgromadzoną wiedzę przełożyć

na co najmniej jedno powszechnie dostępne rozwiązanie techniczne. Tymczasem sieć *DTN* pozostaje terminem znanym tylko niewielkiej grupie specjalistów, głównie naukowców szukających rozwiązań dla tworzonych przez siebie sieci przeznaczonych do unikalnych rozwiązań.

5.3 Propozycje innowacji

Ta część pracy ma za zadanie przedstawienie pomysłów na nowe rozwiązania, których nie znalazłem w trakcie zgłębiania wiedzy o sieci *DTN*, a które na podstawie doświadczeń zebranych w czasie pisania tej pracy wydają się wartościowe.

5.3.1 Organizacja buforów i rutingu w węzłach

Procedura trasowania w symulatorze oparta był o zasadę działania ruterów w sieci *Internet*. Doświadczenia zebrane w czasie pisania tej pracy pokazują, że jest to rozwiązanie nieefektywne.

Rozważmy sieć, w której z węzła A do węzła B prowadzą dwie trasy. Stosowanym protokołem rutingu jest SPF. Węzeł A wysyłając pakiet przeprowadza następujące operacje:

1. Zrutuj pakiet *Bundle*.
2. Umieść pakiet w buforze wyjściowym wybranego przez protokół rutingu łącza.
3. Poczekaj aż pojawi się możliwość wysłania pakietu.
4. Wyślij pakiet.

Zauważmy, że jeśli w monecie rutowania dostępne było łącze pierwsze, a drugie łącze było uszkodzone, to protokół SPF wybierze pierwsze łącze. Załóżmy, że w czasie gdy pakiet oczekuje w buforze wyjściowym na wysłanie, następuje awaria łącza pierwszego, za to zaczyna działać łącze drugie. Niestety, pakiet pozostał w buforze wyjściowym pierwszego łącza. Nie zostanie skierowany do drugiego łącza. Takie zachowanie jest poprawnie w kontekście sieci internetowej, gdzie występuje większy koszt stworzenia skomplikowanego mechanizmu przerzucania pakietów z jednego bufora do drugiego (z tą różnicą, że rutery w sieci internetowej odrzucają zawartość buforów, co już zupełnie nie sprawdziłoby się w sieci *DTN*) niż powtórnego wysłania pakietu od węzła początkowego.

Rozważmy teraz inne rozwiązanie: istnieje tylko jeden wspólny bufor wyjściowy dla wszystkich łączy. Elementy tego bufora to nie tylko pakiety *Bundle*, ale także opisujące je informacje:

- Numer pakietu — bufor powinien mieć charakter kolejki *FIFO*, dlatego że ważne jest utrzymanie w nim kolejności wysyłanych elementów. Każdy pakiet *Bundle*, który jest po raz pierwszy rutowany, otrzymuje kolejny numer, który definiuje kolejność jego wychodzenia z bufora.
- Przypisane łącze — jeśli pakiet zostanie zrutowany, to do pakietu może zostać przypisana informacja o łączu, przez które powinien zostać wysłany. Nie jest to jednak konieczne. W buforze mogą się także znajdować pakiety, dla których droga nie została odnaleziona, ale może zostać odnaleziona w przyszłości.
- Informacja o wykorzystywaniu przez protokół transportowy — ma postać wartości binarnej tak/nie i wskazuje, czy protokół transportowy przypisanego łącza rozpoczął już transmisję

danego pakietu przez łącze. Jest to forma zabezpieczenia przed zmianą trasy dla pakietów, które są już transmitowane przez protokół transportowy.

Przedstawmy teraz procedurę obsługi pakietu:

1. Zrutuj pakiet *Bundle*.
2. Umieść pakiet w buforze wyjściowym z przypisanym łączem i nowym numerem pakietu.
3. Jeżeli pojawi się zmiana stanu sieci (np. którejś łącze ulegnie uszkodzeniu) lub protokół rutingu otrzyma od sieci nowe informacje nt. stanu sieci, to zrutuj ponownie wszystkie pakiety, które nie zostały wysłane przez protokół transportowy. Nie zmieniaj numeru przypisanego do pakietu.
4. Kiedy protokół transportowy na danym łączu jest w stanie wysłać nowy pakiet, to znajduje w buforze wyjściowym element, który jest przypisany do danego łącza, nie jest jeszcze wysłany przez protokół transportowy i jednocześnie jego numer jest najmniejszy spośród wszystkich pakietów w buforze. Następnie oznacza pakiet jako wysłany przez protokół transportowy i wysyła go przez sieć.
5. Jeśli otrzymanie pakietu zostanie potwierdzone przez węzeł, do którego protokół transportowy wysłał pakiet, to wpis w buforze wyjściowym zostaje usunięty.
6. Jeśli w trakcie działania protokołu transportowego następuje awaria łącza i nie jest możliwe uzyskanie potwierdzenia, że pakiet został odebrany, to pakiet jest oznaczany jako niewysłany przez protokół transportowy i zostaje ponownie zrutowany po wykryciu awarii łącza.

Taki algorytm transportowania danych po pierwsze umożliwia dynamiczne kierowanie pakietów różnymi drogami, a po drugie rozwiązuje problem zwracania pakietów przez protokół transportowy, gdy nie zostaną one poprawnie dostarczone.

5.3.2 Nowy protokół rutingu

Doświadczenie z protokołem AODV działającym w sieci o dużych opóźnieniach pozwala rozważyć nowy protokół. Na potrzeby tego opisu nazwijmy go protokołem aktywnie rozgłaszającym, ponieważ główną cechą będzie rozgłaszanie przez węzły swojej pozycji.

Jeżeli rozważymy typową sieć z kilkoma węzłami i danymi, które muszą zostać przesłane od węzła źródłowego do węzła docelowego, to większość protokołów rutingu działa w taki sposób, że to węzeł wysyłający dane jest aktywny w zestawianiu drogi do węzła docelowego. Przykładem może być protokół AODV, który w takiej sytuacji rozgłasza wiadomość *RREQ* do poszczególnych węzłów. W przypadku protokołu aktywnie rozgłaszającego jest inaczej. Ideą tego protokołu jest aktywne rozgłaszanie swojej pozycji w sieci.

Załóżmy, że nowy węzeł podłączony zostaje do sieci. Pozostałe węzły nie znają drogi do niego. Dlatego zadaniem tego węzła będzie poinformowanie innych węzłów o istnieniu trasy prowadzącej do niego. W tym celu rozgłosi on pakiet, który na potrzeby tego opisu nazwiemy *RINF* (ang. *route information* — informacja o trasie). Pakiet ten zostanie rozgłoszony do wszystkich węzłów w sieci. W kontekście sieci *DTN* najlepszą metryką łączy jest opóźnienie. Pakiet *RINF* będzie docierać do poszczególnych węzłów różnymi drogami. Jeśli dany węzeł otrzyma pakiet *RINF* przez trzy różne łącza, które są do niego podłączone, to pierwszy w

kolejności pakiet musiał dotrzeć do tego węzła drogą o najmniejszym opóźnieniu. W ten sposób dany węzeł wprowadzi trasę do swojej tablicy rutingu.

W sieci *DTN* mogą występować awarie i przestoje łączy i dlatego konieczne jest cykliczne rozgłaszanie pakietów *RINF* w celu aktualizacji wiedzy nt. struktury sieci.

Pakiet *RINF* może dotrzeć do węzła różnymi drogami i jednocześnie może się zdarzyć, że pakiety z różnych rozgłoszeń będą docierać do węzła na przemian. To znaczy, że starszy pakiet *RINF* może przybyć po nowszym. Dlatego pakiety *RINF* powinny zawierać znacznik czasowy (ang. *timestamp*), który będzie ustawiany na podstawie czasu rozgłoszenia. Węzeł w swojej tablicy rutingu powinien zapisywać znacznik czasowy wprowadzanych tras. Gdy węzeł otrzymuje nowy pakiet *RINF*, powinien najpierw sprawdzić, czy istnieje już trasa do danego węzła i czy znacznik czasowy trasy jest nowszy lub równy znacznikowi pakietu *RINF*. Jeśli tak, to pakiet *RINF* powinien zostać odrzucony.

Protokół ten ma jedną zasadniczą wadę — nie jest zdolny do natychmiastowego rozpoczęcia swojej pracy. Węzeł podłączony do sieci musi poczekać na wiadomości z informacjami o rutingu przychodzące od innych węzłów.

5.3.3 Zwiększenie odporności protokołu LTP na przekłamania bitów

Symulacja działania protokołów transportowych przy dużej stopie przekłamań bitów pozwoliła zauważyć proste zjawisko: przy dużej stopie przekłamań bitów bardziej efektywne są te protokoły transportowe, które dane wysyłają w mniejszych, niezależnie potwierdzanych porcjach. Im większa stopa przekłamań, tym mniejszą porcję danych powinno się wysyłać.

LTP daje możliwość zmiany rozmiaru segmentu danych bez modyfikacji protokołu. Wystarczy w inny sposób dzielić dane na segmenty, co odwzorowywane jest w opisie zawartości danych. Zauważmy, że opis danych zawartych w pojedynczym pakiecie *LTP* ma postać pary [offset; rozmiar danych]. Protokół *LTP* mógłby, jako opcjonalny nagłówek w raportach, wysyłać informację o liczbie stwierdzonych błędnych pakietów w ciągu dłuższego czasu obserwacji. Pozwoliłoby to nadajnikowi na zmniejszenie rozmiaru wysyłanej porcji danych po wykryciu, że główną przyczyną gubienia danych są przekłamania w dochodzących pakietach *LTP*. Porcje danych, które byłyby retransmitowane, byłyby dzielone na mniejsze segmenty. Np. pojedynczy retransmitowany pakiet byłby dzielony na dwie mniejsze części, które byłyby wysyłane osobno.

6 Bibliografia

1. **Stephen Farrell, Vinny Cahill.** *Delay- and Disruption-Tolerant Networking.* 2006 Artech House Inc., USA.

2. **Forest Warthman.** Delay-Tolerant Networks; A tutorial. 2003 Warthman Associates, USA.

3. **Evan P. C. Jones**. Practical Routing in Delay-Tolerant Networks. Technical report. 2006 Electrical & Computer Engineering, University of Waterloo, USA.

4. **Scott Burleigh, Adrian Hooke, Leigh Torgerson, Kevin Fall, Vint Cerf, Bob Durst, Keith Scott, Howard Weiss.** Delay-Tolerant Networking: An Approach to Interplanetary Internet. Technical report. 2003 Worcester Polytechnic Institute, USA.

5. **Rob Scrimger, Paul LaSalle, Clay Leitzke, Mridula Parihar, Meeta Gupta.** TCP/IP. Biblia. 2002 Helion SA, Polska.

6. **Romit Roy Choudhury, Pradeep Kyasanur.** Sensor Network Routing. Presentation. Duke–Pratt School of Engineering, USA.

7. Licklider Transsmission Protocol, http://irg.cs.ohiou.edu/ltp/

8. **IETF RFC 5050**. Bundle Protocol Specification. Listopad 2007

9. Delay Tolerant Networking Research Group, 2008, http://www.dtnrg.org/wiki

10. Delay-tolerant networking, 2009, http://en.wikipedia.org/wiki/Delay_Tolerant_Networking

11. AppleTalk,2008, http://pl.wikipedia.org/wiki/Appletalk

12. AODV, 2003, http://moment.cs.ucsb.edu/AODV/aodv.html

13. Ad hoc On-Demand Distance Vector Routing, 2009, http://en.wikipedia.org/wiki/AODV

14. **IETF RFC 793**, Transmission Control Protocol, 1981.

15. **IETF RFC 1323**, TCP Extensions for High Performance, 1992.

16. **IETF RFC 2018**, TCP Selective Acknowledgment Options, 1996.

17. Global Positioning System, http://www.gps.gov/

18. **IETF RFC 5326.** Licklider Transmission Protocol — Specification. Wrzesień 2008.
19. **IETF RFC 5327.** Licklider Transmission Protocol — Security Extensions. Wrzesień 2008.

7 Spis tabel

8 Spis ilustracji