POLITECHNIKA GDAŃSKA
Wydział Elektroniki, Telekomunikacji i Informatyki

Krzysztof Nowak

Algorytmy wywłaszczania ścieżek w sieciach MPLS

Rozprawa doktorska

Promotor:
dr hab. inż. Sylwester Kaczmarek, prof. nadzw. PG
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Gdańsk, 2011

Mojej żonie Agnieszce

SPIS TREŚCI

1. WSTĘP

1.1. Istota problemu

Poruszane w tej pracy zagadnienie wywłaszczania jest istotnym narzędziem inżynierii ruchu. Korzysta się z niego w sieciach telekomunikacyjnych, w których utworzenie nowego połączenia lub ścieżki wymaga dokonania rezerwacji pasma[1], w sytuacjach, gdy w danym momencie brakuje wolnego pasma. Wywłaszczanie polega na usunięciu, lub inaczej wywłaszczeniu, jednego lub więcej istniejących połączeń o niższym priorytecie, w celu zwolnienia pasma brakującego do utworzenia nowego połączenia o wyższym priorytecie. Usunięte połączenia nie muszą zostać stracone – zostają w miarę dostępnych zasobów utworzone ponownie na alternatywnych trasach.

Wywłaszczanie jest techniką opcjonalną, ułatwiającą gospodarowanie zasobami sieciowymi, zwłaszcza w sieciach o dużej liczbie połączeń, w których niezbędne jest zautomatyzowanie procesu tworzenia ścieżek. Nie jest to mechanizm nowy, gdyż znany był już w tradycyjnych sieciach telekomunikacyjnych z komutacją kanałów. Dopiero jednak wraz z wprowadzeniem do nowoczesnych sieci transportowych kanałów wirtualnych o definiowanej przepustowości, wywłaszczenie stało się zagadnieniem nieporównanie bardziej złożonym.

W niniejszej pracy zagadnienie wywłaszczania jest omawiane w powiązaniu z technologią wieloprotokołowej komutacji etykietowej, czyli Multiprotocol Label Switching (MPLS) [30,96]. Została ona opracowana w pierwszej połowie lat dziewięćdziesiątych, jako uniwersalny mechanizm transportu przeznaczony szczególnie dla sieci opartych na protokole IP. W sieciach IP/MPLS strumienie pakietów są transportowane w ścieżkach, identyfikowanych za pomocą etykiet, którymi są identyfikatory liczbowe, umieszczone w dodatkowym nagłówku dołączanym do pakietu. Ścieżkom przyporządkowuje się priorytety, a często również rezerwuje określone pasmo na całej drodze połączeniowej. Z tego względu w dalszej części pracy mówimy o wywłaszczaniu ścieżek.

Zagadnienie wywłaszczania, jako jeden z kluczowych elementów inżynierii ruchu [9] w sieciach MPLS, posiada dość dobrze opisane procedury postępowania [7,84,91]. To, co nastręcza najwięcej trudności to problem wyboru konkretnych ścieżek, które mają zostać wywłaszczone. Chodzi o to, aby z jednej strony po usunięciu wybranych ścieżek zapewnić pasmo potrzebne nowej ścieżce, a z drugiej strony spowodować jak najmniejsze straty w przenoszonym ruchu. Zwykle dąży się do tego, aby zminimalizować liczbę usuwanych ścieżek lub ich pasmo. Metoda umożliwiająca dokonanie w tym zakresie optymalnego wyboru [35] jest niestety niemożliwa do praktycznego zastosowania ze względu na zbyt dużą złożoność obliczeniową. Mówiąc ściślej, aby wybrać optymalny zbiór ścieżek przeznaczonych do wywłaszczenia należy przeanalizować wszystkie możliwe ich kombinacje, co prowadzi do

[1] Przyjęło się używać słowa *pasmo* dla określenia *przepływności*; zostało to wprowadzone wraz z technologią ATM.

niekontrolowanego czasu wykonania i jest przez to nie do zaakceptowania w jakichkolwiek praktycznych realizacjach. W praktyce wykorzystuje się więc algorytmy nieoptymalne, których zaletą jest przewidywalny i akceptowalny czas wykonania.

Wywłaszczanie może być przeprowadzane w sposób lokalny lub globalny. W pierwszym przypadku decyzja o wyborze ścieżek przeznaczonych do wywłaszczenia podejmowana jest zwykle w lokalnym węźle, natomiast w drugim przypadku decyzją zajmuje się zwykle wydzielony blok w sieci. Większość ze znanych algorytmów to rozwiązania lokalne [20,70,72,82] bazujące tylko na informacjach dostępnych lokalnie w każdym węźle. Alternatywne algorytmy globalne [35,71] do wyboru ścieżek wykorzystują informacje zebrane z całej zarządzanej domeny MPLS. Podejście globalne umożliwia dokonanie potencjalnie lepszego wyboru, ale wymaga zebrania i analizy większej ilości danych. Wśród pytań, na które nie udzielono dotąd odpowiedzi jest to, czy powszechnie wykorzystywane algorytmy lokalne wywłaszczania dają wyniki porównywalne z algorytmami globalnymi.

W niniejszej pracy zebrano wyniki badań nad algorytmami wywłaszczania w warunkach zbliżonych do tych, jakie występują w rzeczywistych sieciach, przy utrzymującym się wysokim poziomie rezerwacji pasma łączy. Podjęto próbę określenia zysku, jaki może zostać osiągnięty poprzez zastosowanie metod globalnych wywłaszczania w stosunku do metod lokalnych. W tym celu zaproponowano konkurencyjny algorytm globalny i sprawdzono, w jakim stopniu osiągane za jego pomocą efekty pozwalają poprawić jakość wywłaszczania. Uzyskanie odpowiedzi na to pytanie jest niezwykle istotne, gdyż każde wywłaszczenie ścieżki może spowodować chwilową lub długotrwałą utratę części komunikacji w sieci. Im lepiej działa algorytm wywłaszczania, tym te straty są mniejsze. W praktyce przekłada się to na jakość usług oferowanych przez dostawcę, rzutuje na jego wiarygodność a pośrednio także na jego wyniki finansowe.

1.2. Cel i teza pracy

Celem pracy jest analiza i praktyczna weryfikacja jakości wywłaszczania oferowanej przez istniejące, w większości lokalne, algorytmy i porównanie ich z wynikami osiąganymi przy użyciu zaproponowanego algorytmu globalnego. Pod pojęciem oceny jakości rozumiemy tu porównanie wyników pomiarów wielkości takich jak średnia liczba wywłaszczonych ścieżek lub średnia ilość wywłaszczonego pasma.

Aby zrealizować ten cel konieczne jest przybliżenie zagadnienia wywłaszczania, naświetlenie problemu badawczego i pokazanie sposobu realizacji wywłaszczania w sieciach MPLS, z uwzględnieniem używanych do tego celu protokołów oraz procedur. W celu przeprowadzenia porównania i dokonania prawidłowej oceny algorytmów należy omówić istniejące rozwiązania, przeanalizować ich sposób działania i dokonać wstępnej analizy. Bazą do tego powinno być formalne zdefiniowanie istotnych wielkości, dzięki czemu uzyska się jednolity i jednoznaczny opis, a w dalszej części ocenę jakościową algorytmów.

Kluczowym elementem pracy jest zaproponowany algorytm wywłaszczania, który używa heurystyki do wyboru możliwe najkorzystniejszego zbioru ścieżek, z punktu widzenia zadanej funkcji celu. Jest to algorytm globalny, korzystający z danych o ścieżkach utworzonych w całej domenie, co potencjalnie daje możliwość dokonania lepszego wyboru niż oferują to algorytmy lokalne. Projektując algorytm dążono do uzyskania jednocześnie wysokiej jakości wyboru ścieżek jak też akceptowalnej złożoności obliczeniowej. Ta ostatnia cecha przekłada się na czas wykonania i jest krytycznym elementem każdego realizowanego algorytmu, gdyż bez udowodnienia akceptowalnego czasu wykonania dla najgorszego przypadku nie można wprowadzić algorytmu do pracy w rzeczywistych sieciach z gwarancją jakości usług.

Aby przeprowadzić wiarygodne badania algorytmów, należy dysponować odpowiednim narzędziem analitycznym lub pomiarowym. W odniesieniu do wywłaszczania, ze względu na duże rozmiary badanych sieci zdecydowanie lepszym rozwiązaniem jest wykorzystanie metody symulacyjnej. Zdecydowano się na implementację własnego symulatora umożliwiającego wszechstronne badanie mechanizmów inżynierii ruchu. W procesie jego projektowania i implementacji szczególny nacisk położono na przenośność, łatwość rozbudowy i szybką interpretację wyników.

Przyjęto metodę badawczą polegającą na przeprowadzeniu badań symulacyjnych ogólnie dostępnych algorytmów, przede wszystkim w zakresie liczby powodowanych wywłaszczeń i ilości traconego pasma. Jednym z najważniejszych celów było przy tym określenie, jak zmieniają się wyniki uzyskane przy użyciu różnych algorytmów wywłaszczania w różnych warunkach. Oczekiwanym efektem badań powinno być określenie z dużym prawdopodobieństwem, jak zachowują się poszczególne algorytmy w warunkach najbardziej zbliżonych do rzeczywistych.

Celem głównym pracy jest potwierdzenie prawdziwości **postawionej tezy**, która brzmi następująco:

Zaproponowany przez autora algorytm globalny umożliwia bardziej efektywne wywłaszczanie niż umożliwiają to inne dostępne obecnie algorytmy, w tym najbardziej popularne algorytmy lokalne, uznawane za wysoce efektywne.

Aby osiągnąć ten cel główny postawiono następujące cele pomocnicze.

1. Ocena dostępnych algorytmów wywłaszczania:
   - dokonanie przeglądu istniejących algorytmów,
   - wybór do dalszej analizy kilku najpopularniejszych algorytmów,
   - przeprowadzenie analizy sposobu działania wybranych algorytmów,
   - sprawdzenie możliwości użycia algorytmów w sieciach MPLS i ewentualna modyfikacja pod tym kątem,
   - ujednolicenie i sformalizowanie opisu wybranych algorytmów,
   - dokonanie niezależnej analizy złożoności obliczeniowej wybranych algorytmów.
2. Opis zaproponowanego algorytmu wywłaszczania:

- prezentacja zasady działania algorytmu,
- przeprowadzenie analizy porównawczej własnego i istniejących algorytmów,
- analiza złożoności obliczeniowej.

3. Przygotowanie środowiska pomiarowego:
   - określenie założeń do środowiska symulacyjnego,
   - utworzenie programu symulującego sieci MPLS,
   - implementacja wybranych algorytmów,
   - implementacja graficznego środowiska do tworzenia scenariuszy badań,
   - dokonanie charakterystyki utworzonych aplikacji,
   - weryfikacja wiarygodności symulatora.
4. Przeprowadzenie symulacji i analiza wyników pomiarów:
   - określenie scenariuszy badań symulacyjnych,
   - zdefiniowane kryteriów porównawczych,
   - przeprowadzenie symulacji,
   - zebranie i interpretacja wyników pomiarów.

Realizacja tak postawionych celów wraz z niezbędnym wprowadzeniem do tematu wymagała odpowiedniej struktury pracy, przedstawionej w kolejnej części pracy.

1.3. Struktura pracy

Wszystkie postawione cele znalazły swoje odzwierciedlenie w strukturze niniejszej pracy, która jest w dalszej części skonstruowana następująco.

W rozdziale drugim przedstawiono ogólne zasady, na których oparta jest architektura sieci MPLS. Rozpoczęto od genezy powstania koncepcji przełączania etykietowego, po czym omówiono te jej cechy, które są istotne dla zrozumienia zasady działania rutera IP/MPLS i sposobu tworzenia ścieżek. Szczególny akcent położono na przedstawienie protokołów tworzenia ścieżek i wyjaśnienie zasad komutacji pakietów. Na tej bazie możliwe było wprowadzenie w kolejnym rozdziale pojęcia wywłaszczania.

Trzeci rozdział poświęcono procedurom wywłaszczania i realizującym je algorytmom. Omówiono w nim ogólny schemat realizacji wywłaszczania, a następnie bardziej szczegółowo przedstawiono procedury postępowania towarzyszące algorytmom wywłaszczania. Wprowadzono klasyfikację metod na lokalne i globalne oraz przedstawiono różnice w obu koncepcjach. Wymieniono najważniejsze parametry jakościowe, niezbędne dla porównania różnych algorytmów. Przedstawiono także stan standaryzacji związanej bezpośrednio z wywłaszczaniem i wprowadzono niezwykle istotne pojęcie priorytetu ścieżki.

Dalsza część trzeciego rozdziału została poświęcona istniejącym algorytmom. Rozpoczęto od zdefiniowania istotnych wielkości i parametrów opisujących algorytmy, po czym przedstawiono i scharakteryzowano poszczególne ogólnie dostępne algorytmy. Opisy uzupełniono

o ujednolicony pseudokod umożliwiający lepsze zrozumienie zasady działania i ewentualną niezależną implementację. Rozdział zamyka wstępna analiza porównawcza algorytmów.

W rozdziale czwartym omówiono opracowany przez autora algorytm wywłaszczania. Wyjaśniono jego cechy i sposób działania. Przedstawiono jego pseudokod w formie zgodnej z pozostałymi algorytmami i dokonano jego wstępnej analizy, w oparciu o wcześniej wprowadzone, jednolite kryteria ocen.

Po dokonaniu prezentacji algorytmów możliwe było przedstawienie środowiska badawczego, którym jest zrealizowany przez autora program symulacyjny. Temu poświęcony jest rozdział piąty. Omówiono w nim założenia do modelu symulacyjnego a następnie przedstawiono sposób, w jaki je zrealizowano, tworząc uniwersalny symulator mechanizmów inżynierii ruchu w sieciach MPLS. Omówiono schemat blokowo-funkcjonalny programu oraz wymieniono zaimplementowane metryki służące do oceny jakości algorytmów. Na zakończenie przedyskutowano przeprowadzone analizy potwierdzające wiarygodność programu, a szczególnie poprawność uzyskiwanych z jego pomocą wyników.

Badaniom algorytmów poświęcono rozdział szósty. Przedstawiono warunki, w jakich prowadzono symulacje i zaprezentowano użyte topologie sieciowe. Następnie, w zasadniczej części rozdziału dokonano analizy uzyskanych wyników badań oraz postawiono wypływające z nich istotne wnioski. Na zakończenie podsumowano rezultaty badań i odniesiono się do tezy postawionej w pierwszym rozdziale pracy.

W ostatnim, siódmym rozdziale przedstawiono wnioski końcowe, podsumowano dokonania i nakreślono kierunki przyszłych badań. Pracę uzupełniono o załączniki zawierające istotne materiały, których nie umieszczono w zasadniczej części pracy. Dołączono bibliografię oraz spis symboli i listę akronimów użytych w tekście.

2. TECHNOLOGIA MPLS

Niniejszy rozdział zawiera omówienie najważniejszych aspektów technologii MPLS, niezbędnych dla zrozumienia specyfiki działania metod wywłaszczania. Przedstawiono czynniki, jakie wpłynęły na ewolucję sieci IP w kierunku IP/MPLS a następnie omówiono najważniejsze założenia architektury MPLS. Omówiono także sposób zestawiania ścieżek oraz dokonano porównania przeznaczonych do tego protokołów.

2.1. Historia

Przełom lat osiemdziesiątych i dziewięćdziesiątych ubiegłego wieku w dziedzinie techniki komputerowej przyniósł gwałtowny rozwój sieci Internet, w którym roczny wzrost przesyłanego ruchu przekraczał 100% [37]. Jego szybkość i skala zaskoczyły nawet twórców związanych z nim technologii, a jednocześnie były wyzwaniem dla producentów sprzętu sieciowego. Równolegle ze wzrostem ilości serwerów sieciowych pojawiały się coraz to nowe aplikacje i usługi, które wymagały coraz większego pasma. Aby sieć była w stanie temu podołać, przeprowadzano rozbudowę sieci szkieletowych oraz zwiększano przepływności łączy. Przewidywano wówczas, że istniejące rutery mogą wkrótce nie sprostać prognozowanej szybkości rozwoju sieci i związanemu z tym wzrostowi rozmiaru tablic rutingu. W tej sytuacji poszukiwano rozwiązań umożliwiających przyspieszenie przełączania pakietów IP poprzez zmianę zasady kierowania ruchem. W połowie lat dziewięćdziesiątych szczególne nadzieje pokładano w technikach grupowania pojedynczych połączeń IP w większe strumienie, które możnaby komutować zbiorczo i dzięki temu znacznie ograniczyć rozmiary tablic kierowania ruchem oraz przyspieszyć proces komutacji [75].

Odpowiednie rozwiązania istniały już w tym czasie w sieciach telekomunikacyjnych, gdzie z powodzeniem stosowano technikę Asynchronous Transfer Mode (ATM) [81,90], w której przesyłane dane były dzielone na komórki opatrzone identyfikatorami VCI i VPI. Odmiennie od adresów IP, które pozostają niezmienne w czasie transportu pakietu, wartości VCI/VPI podlegają zmianie na kolejnych łączach, dzięki czemu nie muszą być unikalne w skali sieci. Wartości identyfikatorów są ustalane lokalnie przez przełączniki ATM, co umożliwia optymalizację ich przydziału pod kątem szybkości przełączania i zmniejszenia rozmiarów tablic kierowania ruchem. Technika ATM doczekała się wielu udanych implementacji, a mimo tego nie zyskała pełnej popularności w sieciach IP ze względu na wyższą cenę urządzeń, większy stopień komplikacji, a także niewielki rozmiar komórki, wynoszący 53 bajty, z czego 48 przypadało na dane. Ta ostatnia cecha powodowała konieczność dzielenia pakietów IP na małe kawałki, co zwiększało czas przetwarzania i prawdopodobieństwo strat pakietów w sieci. Niemniej jednak na przełącznikach ATM oparto pierwsze rozwiązania sprzętowe zmierzające do przyspieszenia komutacji IP, do których należały *Cell Switching Router* (CSR) firmy Toshiba, *IP Switching* firmy Ipsilon i *Aggregate Router-based IP Swi-*

tching (ARIS) firmy IBM [30].

Opracowane w pierwszej połowie lat dziewięćdziesiątych rozwiązanie *Tag Switching* [93] firmy Cisco uznaje się za pierwowzór dla architektury MPLS, z którego zapożyczono jej podstawowe zasady. Pakiet IP na wejściu do sieci został poprzedzony dodatkowym nagłówkiem (tagiem), który określał przynależność do zbiorczego strumienia i był używany do kierowania pakietów, z pominięciem adresu IP. Ciekawą cechą tego rozwiązania było to, że w przypadku zastosowania do transportu przełączników ATM, tag mógł być umieszczony w nagłówku komórki w polach VCI i VPI.

W roku 1997 w ramach organizacji IETF utworzono grupę roboczą „mpls" [42], która rozpoczęła prace nad nową technologią przełączania pakietów dla sieci IP, nazwaną Multiprotocol Label Switching. Najważniejszym dokumentem opracowanym przez grupę jest *Multiprotocol Label Switching Architecture*, który w roku 2001 stał się standardem RFC 3031 [96]. W ramach dalszych prac opracowano szereg specyfikacji uzupełniających, w tym dedykowany protokół dystrybucji etykiet Label Distribution Protocol (LDP) [5] oraz specyfikację RSVP-TE [7], będącą rozszerzeniem protokołu Resource Reservation Protocol (RSVP) [22] o mechanizmy tworzenia ścieżek.

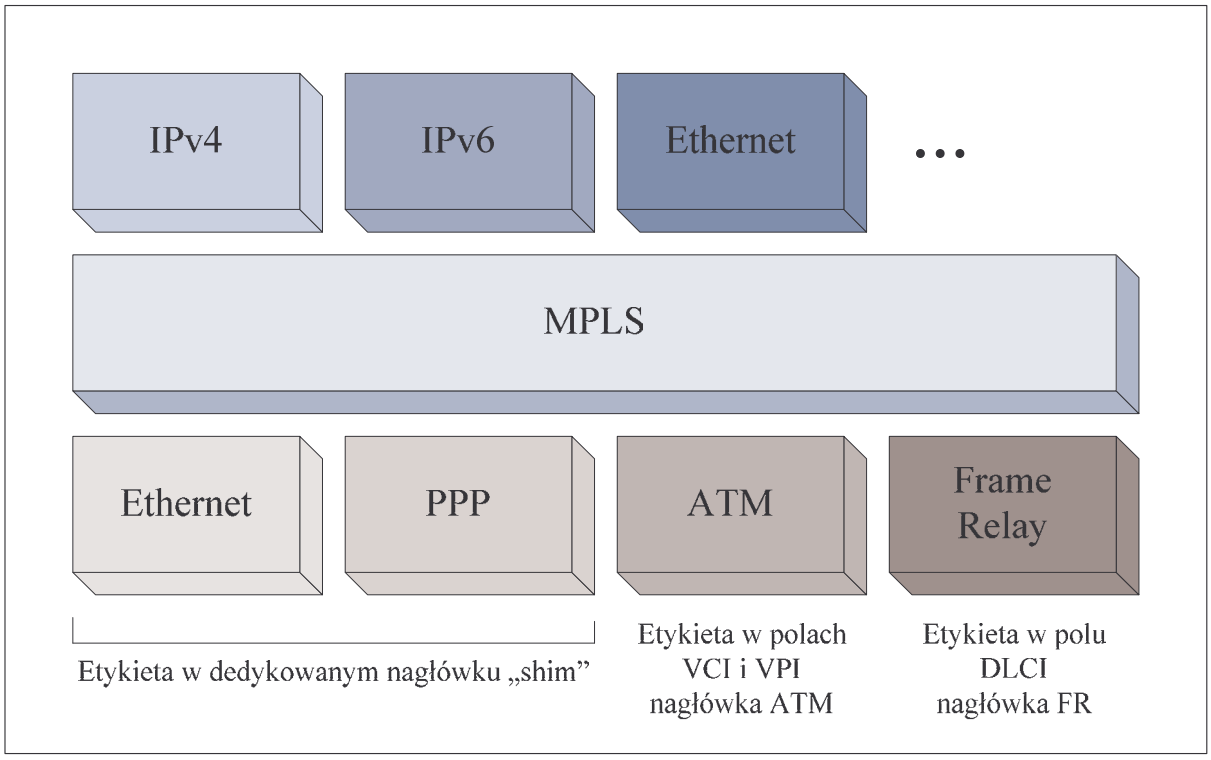
Tymczasem równolegle z rozwojem techniki MPLS nastąpił znaczący postęp w szybkości działania ruterów, wynikający z zastosowania procesorów nowej generacji. W rezultacie główny czynnik, jaki wymusił rozpoczęcie prac nad MPLS, tj. przyspieszenie komutacji, okazał się nieaktualny. Niejako przy okazji odkryto jednak, że nowa technika daje niedostępne dotąd w sieciach IP możliwości zarządzania ruchem,. W sieciach opartych na MPLS dużo łatwiej tworzy się sieci wirtualne a ponadto możliwe jest zarządzanie ruchem poprzez rezerwację pasma i przydzielanie ścieżkom priorytetów. Rozwój MPLS przyniósł kolejne zaawansowane narzędzia, jak mechanizmy szybkiego przełączania ruchu na wypadek awarii (ang. *fast rerouting*) lub koncepcja użycia MPLS jako uniwersalnej platformy transportowej dla popularnych technologii warstwy łącza (Ethernet, ATM, Frame Relay). Technologia MPLS stała się szeroko akceptowana i została wdrożona w sieciach wielu operatorów telekomunikacyjnych oraz dostawców usług sieciowych. Obecnie MPLS jest niemal synonimem inżynierii ruchu w sieciach IP [76].

Od kliku lat w rozwój architektury MPLS zaangażowana jest organizacja ITU-T, czego efektem jest powstanie wymagań i standardów opisujących architekturę określaną wcześniej jako Transport MPLS (T-MPLS) [88], na podstawie której wspólnie z IETF opracowano architekturę nazywaną obecnie MPLS Transport Profile (MPLS-TP) [66,67]. Zalecenia ITU-T z jednej strony upraszczają architekturę sieci poprzez ograniczenie ilości opcji i wariantów, a z drugiej strony rozszerzają ją w zakresie m.in. wykrywania uszkodzeń lub separacji sieci sygnalizacji i danych. Umożliwia to dostosowanie architektury MPLS do standardów obecnych w tradycyjnych sieciach transportowych, opartych na technologiach takich jak SDH/SONET lub ATM.

2.2. Architektura MPLS

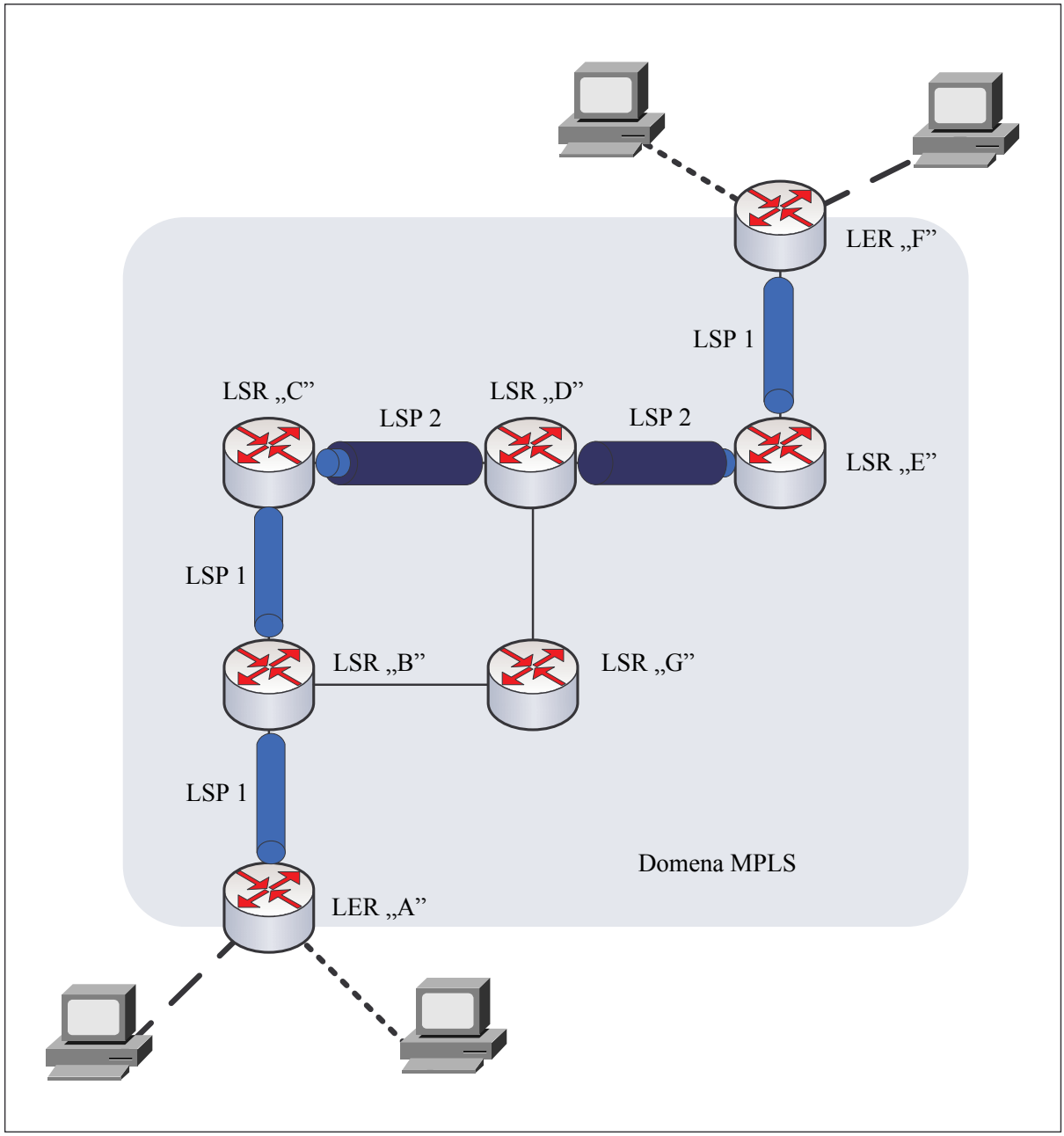
Technologia wieloporotokołowego przełączania etykietowego MPLS (ang. *Multiprotocol Label Switching*) jest przeznaczona w ogólności do transportu różnego typu pakietów (stąd określenie „wieloprotokołowe”). Od początku swojego istnienia najważniejszym jej przeznaczeniem było wykorzystanie w sieciach IP, choć ostatnio sporą popularność zdobywają także techniki emulacji łącza PWE3 (ang. *Pseudo Wire Emaulation Edge-to-Edge*) [24,25,106], w których sieci MPLS używane są do transportu ruchu typu ATM, Frame Relay a nawet Ethernet. Ta ostatnia możliwość jest szczególnie interesująca, gdyż umożliwia łatwe tworzenie wirtualnych sieci LAN łączących oddalone od siebie lokalizacje.

Już we wczesnych specyfikacjach odnoszących się do warstwy łącza, w ramach architektury MPLS zdefiniowano sposób jej implementacji w technologiach Ethernet i PPP (ang. *Point to Point Protocol*), a także ATM i Frame Relay (rys. 2.1). W ten sposób niezależnie od posiadanej przez operatora technologii transportowej, możliwa była stosunkowo łatwa migracja do MPLS bez konieczności wymiany całej infrastruktury sieciowej.



Rys. 2.1. Model warstwowy współpracy MPLS z protokołami warstw łącza i sieci.

Sieci MPLS są sieciami typu połączeniowego, w których transport dowolnego pakietu jest możliwy tylko po zestawieniu na całej drodze tzw. ścieżki LSP (ang. *Label Switched Path*). Przynależność danego pakietu do ścieżki jest określona przez jego etykietę. Węzły w sieci MPLS są określane mianem ruterów LSR (ang. *Label Switching Router*), wśród których wyróżnia się rutery brzegowe LER (ang. *Label Edge Router*). Ruter LER ma pełną funkcjonalność rutera LSR, a oprócz tego zapewnia enkapsulację i dekapsulację pakietów IP (rys. 2.2).

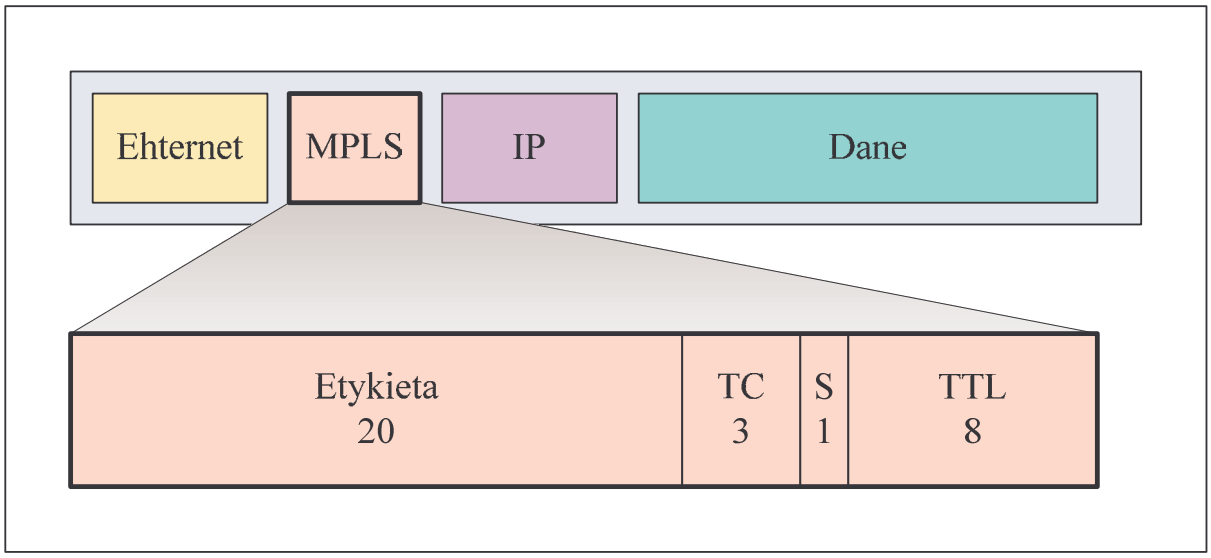


Rys. 2.2. Elementy architektury MPLS i rola ścieżek LSP.

Każdy przesyłany w sieci MPLS pakiet IP zostaje wyposażony w nagłówek, który przypisuje go do ścieżki LSP. Na łączach typu Ethernet lub PPP dedykowany nagłówek MPLS obejmuje 4 bajty, które zostają „wciśnięte” pomiędzy nagłówek ramki w warstwie łącza a nagłówek datagramu IP – stąd pochodzi angielska nazwa *shim header* (*shim* oznacza klin). Powstały w ten sposób pakiet MPLS został przedstawiony na rys. 2.3. Inaczej tworzy się pakiet MPLS, gdy warstwa łącza zbudowana jest w technologii ATM lub Frame Relay. Nie wprowadza się wówczas dodatkowego nagłówka, ale wykorzystuje istniejące pola nagłówków ATM lub Frame Relay do przenoszenia etykiety. Dokładny opis pól i ich rola zawarte są w RFC 3032 [95] wraz ze zmianami i uzupełnieniami [1,94].

Dedykowany nagłówek MPLS zawiera następujące pola:

- etykieta (20 bitów), określająca przynależność do ścieżki,
- klasa ruchu (TC, 3 bity), do której należy pakiet,
- spód stosu (S, 1 bit), będący wskaźnikiem ostatniej etykiety na stosie oraz
- czas życia pakietu (TTL, 8 bitów), zawierający wartość zmniejszaną o jeden w każdym ruterze; po osiągnięciu wartości zero pakiet jest kasowany z powodu podejrzenia powstania pętli.



Rys. 2.3. Struktura pakietu MPLS w sieci Ethernet (liczby określają rozmiar pól w bitach).

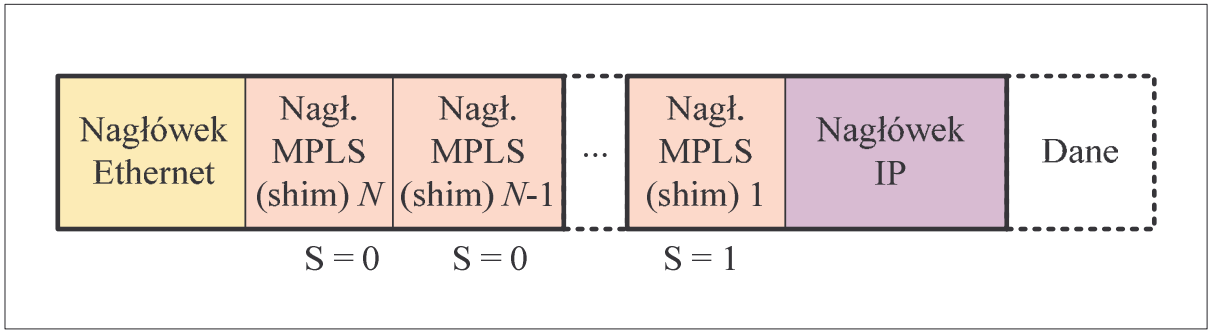
Etykieta (ang. *label*) jest identyfikatorem liczbowym, który jest podstawową informacją adresową i jest używany do kierowania pakietu MPLS wewnątrz domeny. Etykiety, w przeciwieństwie do adresów IP, mają znaczenie lokalne, to znaczy podlegają zmianie przez rutery na każdym łączu na trasie ścieżki. O tym, jakie konkretnie wartości etykiet przypisuje się pakietom, decydują wyłącznie sąsiednie rutery na etapie tworzenia ścieżki. Przestrzeń możliwych wartości etykiet to 10^{20} na jednym łączu lub w całej domenie (ang. *per-interface* lub *per-platform*), zależnie od wybranej strategii przydziału etykiet. Zatem na jednym łączu lub w domenie teoretycznie może istnieć około miliona ścieżek, choć nie zawsze cały zakres jest dostępny. W przypadku implementacji sieci MPLS na urządzeniach ATM lub Frame Relay pole etykiety jest krótsze niż 20 bitów i zakres etykiet jest przez to odpowiednio mniejszy. Poza tym niezależnie od używanej technologii niektóre implementacje ograniczają liczbę dostępnych etykiet. Dodatkowo etykiety o wartościach od 0 do 15 zostały przez IETF zarezerwowane do celów specjalnych. Część z nich została już zdefiniowana [69,92,95], a inne pozostają do wykorzystania w przyszłości.

Pole TC umożliwia oznaczenie klasy ruchu, do której należy pakiet. We wcześniejszych opracowaniach pole to było oznaczane jako eksperymentalne (EXP), jednak dokument RFC 5462 [4], ustala znaczenie tego pola jako Traffic Class. Jest to z jednej strony zgodne z początkowymi zamierzeniami projektantów architektury MPLS, a z drugiej strony jest usank-

cjonowaniem stanu faktycznego.

Pole TTL (ang. *Time To Live*) w nagłówku MPLS ma podobne przeznaczenie, jak jego odpowiednik w nagłówku IP [85]. Jego najważniejszym celem jest wykrywanie powstania zapętleń i eliminowanie pakietów zbyt długo przebywających w sieci. Sposób przetwarzania i kontroli pola TTL z nagłówka MPLS może się różnić od analogicznego postępowania w sieciach IP i dlatego został objęty osobnymi regulacjami [1].

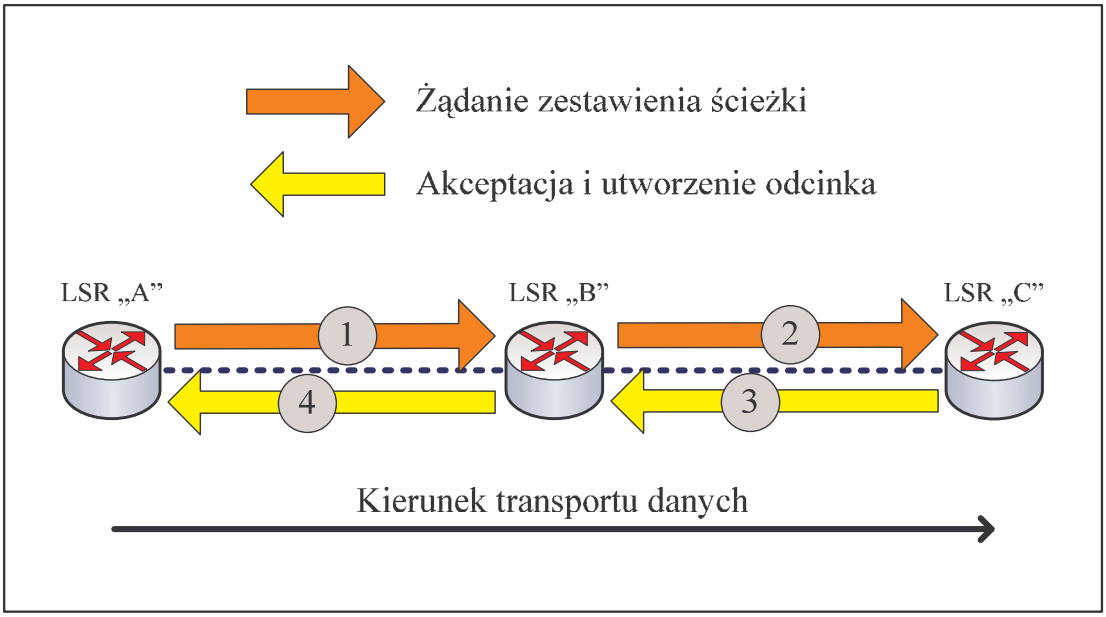
Architektura MPLS zapewnia wsparcie dla sieci hierarchicznych, tzn. umożliwia tworzenie ścieżek wewnątrz ścieżek (tuneli), co m. in. ułatwia kierowanie ruchem oraz budowanie sieci wydzielonych. Funkcjonalność ta jest oparta na mechanizmie stosu etykiet (ang. *label stack*). Tworzenie stosu polega na umieszczeniu w pakiecie MPLS następujących po sobie kilku nagłówków MPLS (rys. 2.4). Tylko nagłówek ze szczytu stosu, znajdujący się zaraz za nagłówkiem ramki warstwy łącza jest brany pod uwagę przy określaniu drogi, a kolejne wykorzystuje się dopiero, gdy same staną się szczytem stosu po usunięciu etykiet z wyższych poziomów stosu. Gdy pozostaje ostatnia etykieta ze stosu to wartość pola S równa się 1. Po jej usunięciu pakiet przestaje być pakietem MPLS i wówczas zwykle opuszcza domenę MPLS.



Rys. 2.4. Implementacja stosu etykiet poziomu *N* z użyciem nagłówków dedykowanych.

Standard definiujący architekturę MPLS [96] ze względów historycznych opisuje różne tryby tworzenia ścieżek, z których obecnie najczęściej wykorzystywany jest *pulled-conditional* (rys. 2.5), który zapewnia utworzenie kompletnej ścieżki. Tworzenie ścieżki inicjowane jest w ruterze wejściowym (ang. *ingress*), skąd żądanie utworzenia kolejnych części ścieżki jest wpierw kierowane krok po kroku w kierunku rutera wyjściowego (ang. *egress*), a następnie w drodze powrotnej akceptowane aż do osiągnięcia rutera wejściowego. Procedura taka ma szereg zalet. Po pierwsze, umożliwia narzucenie drogi, jaką kierowana jest ścieżka, a tym samym jawne jej kontrolowanie. Po drugie, w drodze powrotnej możliwe jest zarezerwowanie pasma dla ścieżki. Po trzecie, potwierdzenie utworzenia pełnej drogi połączeniowej trafia do rutera wejściowego. W ten sposób w ruterze wejściowym, który inicjuje tworzenie ścieżki, jest gwarancja, że ścieżka została utworzona od początku do końca, a na każdym łączu zarezerwowane jest żądane pasmo. Charakterystyczna dla oryginalnej architektury MPLS jest zasada tworzenia ścieżki tylko w jednym kierunku [8]. Aby uzyskać komunikację dwu-

kierunkową, konieczne jest utworzenie drugiej ścieżki skierowanej w przeciwną stronę. Zasada ta została zmieniona w nowszych zaleceniach dotyczących rozszerzonych architektur opartych na MPLS, wśród których najważniejsze to GMPLS [12,14] i MPLS-TP [67,88]. Wymaga się w nich, aby ścieżka została utworzona w obu kierunkach.

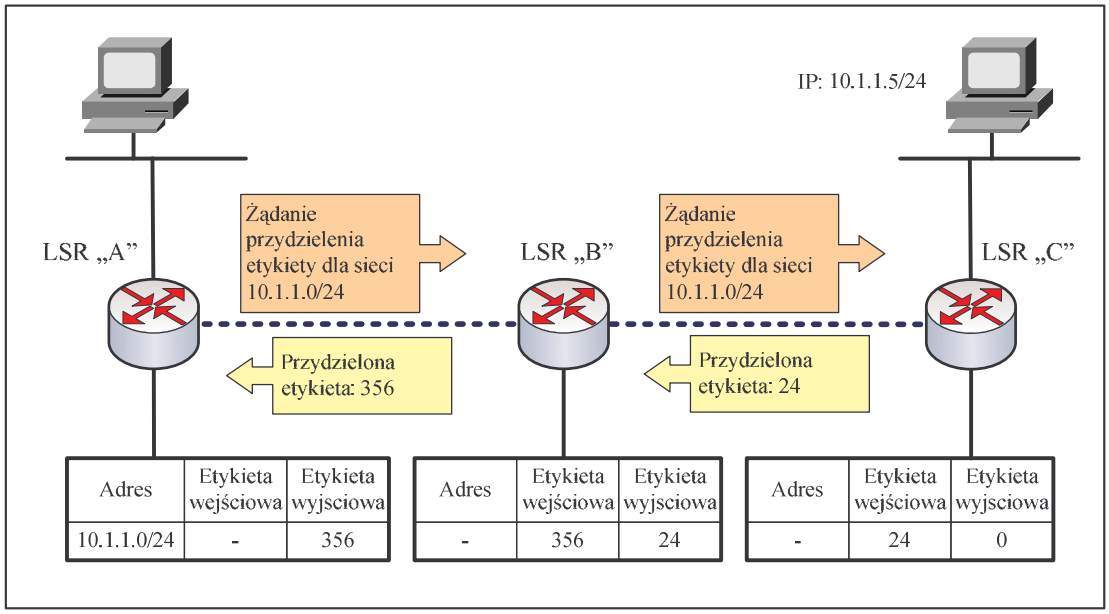


Rys. 2.5. Sposób tworzenia ścieżki w trybie *pulled-conditional*. Cyfry określają sekwencję poszczególnych wiadomości.

Za wymianę wiadomości sygnalizacyjnych związanych z utworzeniem nowej ścieżki jest odpowiedzialny wybrany przez operatora protokół tworzenia ścieżek, zwany protokołem dystrybucji etykiet. Niezależnie od użytego protokołu, tworzenie ścieżki odbywa się w podobny sposób. Ruter wejściowy określa najpierw trasę ścieżki i wysyła żądanie przydzielenia etykiety do następnego rutera. Ten określa kolejny ruter na drodze połączeniowej i przekazuje do niego podobne żądanie, aż wiadomość dotrze do rutera wyjściowego znajdującego się najbliżej docelowej sieci. Ruter wyjściowy określa wartość etykiety i informuje o niej ruter poprzedni. Podobnie każdy ruter w kierunku powrotnym podejmuje decyzję o wartości etykiety i informuje o niej sąsiada w kierunku do rutera wejściowego. Jeśli zażądano także rezerwacji pasma, wówczas przed odesłaniem odpowiedzi każdy z ruterów dokonuje stosownej rezerwacji na własnym łączu na drodze ścieżki. Po zakończeniu procedury utworzona w ten sposób ścieżka jest zarejestrowana w tablicach kierowania ruchem poszczególnych ruterów (rys. 2.6). Stąd rutery posiadają informację, że przed przesłaniem pakietu do następnego rutera należy mu przydzielić otrzymaną od sąsiada wartość etykiety.

Komutacja pakietów odbywa się tak, jak pokazano w przykładzie pokazanym na rys. 2.7. Ruter wejściowy po otrzymaniu pakietu IP na podstawie docelowego adresu IP i ewentualnie innych parametrów przydziela go do klasy Forwarding Equivalence Class (FEC). Choć w ogólności możliwe jest do tego celu wykorzystanie dowolnych informacji z pakietu IP, to

zgodnie z najnowszymi zaleceniami, w celu przyspieszenia komutacji, opcjonalne pola nagłówka IP są w tym procesie ignorowane [99]. Po określeniu klasy FEC, na podstawie odpowiedniego wpisu w tablicy FTN (FEC-to-NHLFE, NHLFE = Next Hop Label Forwarding Entry) nadaje się pakietowi etykietę i kieruje go do odpowiedniego portu wyjściowego w kierunku kolejnego rutera. Ten po otrzymaniu zaetykietowanego pakietu odszukuje parę (etykieta, port wejściowy) w tablicy ILM (Incoming Label Map), na podstawie której dokonuje wymiany etykiety na inną i przekazuje pakiet dalej. Taka procedura jest powtarzana, aż pakiet dotrze do rutera wyjściowego, który odczyta z własnej tablicy ILM polecenie usunięcia etykiety (etykieta zerowa), po czym prześle pozbawiony etykiety pakiet dalej zgodnie z tablicą rutingu IP. Taki sposób transportu danych przez domenę MPLS jest niewidoczny dla urządzeń końcowych, choć może spowodować zmniejszenie dopuszczalnego rozmiaru pakietu MTU (ang. *Maximum Transfer Unit*) o wielokrotność czterech bajtów, zależnie od największego rozmiaru stosu etykiet.



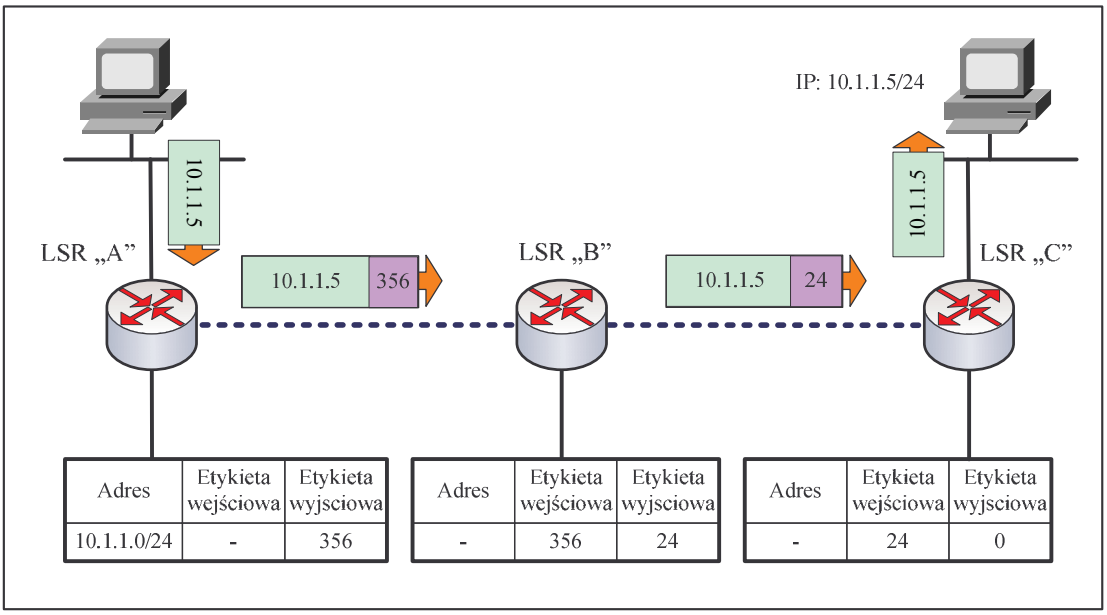
Rys. 2.6. Mechanizm tworzenia tablic kierowania ruchem w ruterach LSR.

Należy dodać, że w najnowszych zaleceniach ITU-T określających profil transportowy MPLS (MPLS-TP) [21,66,67] przedstawiono wymagania na sieci transportowe MPLS stawiane przez operatorów telekomunikacyjnych:

- tworzenie ścieżek dwukierunkowych o jednakowych lub różnych pasmach w obu kierunkach,
- fizyczną separację warstwy transportowej i sterowania,
- niezależność od użytych protokołów sygnalizacyjnych (patrz rozdz. 2.3),
- możliwość rezygnacji z protokołów sygnalizacyjnych (sterowanie przez systemy zarządzania),
- tworzenie ścieżek poprzez kilka domen (homogenicznych lub heterogenicznych),

– wsparcie dla różnych typów sieci klienckich, w tym IP, MPLS, FR i ATM.

Wiele z wymienionych wymagań zostało już zrealizowanych w ramach architektury GMPLS [16,33,57,58,64]. Można oczekiwać, że architektura MPLS-TP wyznaczy jeden z ważnych kierunków rozwoju technologii MPLS w ciągu najbliższych kilku lat.



Rys. 2.7. Sposób przełączania etykiet w ruterach LSR.

2.3. Protokoły sygnalizacyjne

W każdej praktycznej realizacji architektura MPLS bazuje na wielu protokołach, wśród których najważniejsze to protokół rutingu i protokół zestawiania ścieżek, zwany protokołem dystrybucji etykiet.

Protokoły rutingu w sieciach MPLS odpowiadają za rozgłoszenie topologii sieci wraz z informacjami dotyczącymi dostępnych na łączach zasobów. Aktualnie taką rolę spełniają protokoły OSPF-TE [55], IS-IS-TE [98] oraz BGP-TE [77], które bazują na standardowych protokołach znanych z sieci IP, rozbudowanych o mechanizmy rozsyłania dodatkowych informacji o łączach, koniecznych dla wsparcia inżynierii ruchu (TE).

Protokoły zestawiania ścieżek umożliwiają rozgłaszanie etykiet i rezerwację pasma. Zależnie od decyzji operatora i możliwości oprogramowania używanego w ruterach, jest to jeden z protokołów: RSVP-TE, LDP lub CR-LDP. W dalszej części zostaną omówione cechy poszczególnych protokołów.

Oryginalny protokół RSVP (Resource Reservation Protocol) [22] został opracowany w połowie lat 90-tych, w celu zapewnienia mechanizmu rezerwacji zasobów dla poszczególnych strumieni IP. Nigdy w tej postaci nie stał się szeroko rozpowszechniony, ale dzięki otwartej definicji możliwe było jego rozszerzenie o mechanizmy tworzenia ścieżek LSP.

W ten sposób jako RSVP-TE [7] stał się najbardziej popularnym protokołem dystrybucji etykiet.

W celu zestawienia nowej ścieżki ruter wejściowy wysyła do rutera wyjściowego wiadomość PATH, zawierającą między innymi żądanie LABEL_REQUEST umieszczone w obiekcie SESSION. Opcjonalne obiekty to: EXPLICIT_ROUTE (ERO), zawierający zapis żądanej drogi, RECORD_ROUTE (RRO), nakazujący rejestrację listy węzłów pośrednich i zwrócenie jej do rutera wejściowego, oraz SESSION_ATRIBUTE, zawierający między innymi priorytety ścieżki.

Ruter wyjściowy po otrzymaniu wiadomości PATH, odsyła z powrotem wiadomość RESV, która zawiera między innymi obiekt LABEL i opcjonalnie RECORD_ROUTE. Obiekt LABEL zawiera etykietę, która zostaje przydzielona dla ścieżki na danym łączu, natomiast RECORD_ROUTE listę węzłów pośrednich.

Protokół RSVP-TE jest oparty na RSVP, który był w zamierzeniu przeznaczony do obsługi krótkich sesji. Jest oparty bezpośrednio na warstwie IP i nie zapewnia wiarygodnej komunikacji z sąsiadami. W konsekwencji w sieciach MPLS, gdzie ścieżki są z założenia połączeniami długookresowymi, wymagane jest periodyczne odświeżanie informacji o utworzonych ścieżkach. Jeśli w określonym przedziale czasu nie nastąpi odświeżenie, wówczas przyjmuje się, że ścieżka została usunięta. Model taki w MPLS określany jest mianem *soft-state*. Jego wadą jest duża ilość przesyłanych danych sygnalizacyjnych, proporcjonalna do liczby ścieżek i częstości odświeżania. Rodzi to wątpliwości dotyczące skalowalności protokołu RSVP-TE. Niemniej jednak zaproponowano pewne udoskonalenia, które spowodowały zmniejszenie ilości informacji niezbędnej do utrzymania ścieżek [15].

Protokół LDP (ang. *Label Distribution Protocol*) [5] został opracowany specjalnie do celów rozgłaszania etykiet, poprzez rozwinięcie protokołu TDP (ang. *Tag Distribution Protocol*) firmy Cisco Systems, opracowanego dla technologii *tag switching*.

Utworzenie ścieżki w LDP odbywa się następująco. Ruter wejściowy wysyła żądanie przydzielenia etykiety w wiadomości Label Request, zawierającej aktualną długość ścieżki *(hop count)* i opcjonalnie listę ruterów dla wykrywania pętli. Ruter wyjściowy odsyła informację zwrotną w wiadomości Label Mapping, zawierającą wymaganą etykietę i definicję ścieżki, obejmującą prefiksy docelowych adresów IP oraz ewentualnie inne dane umożliwiające zakwalifikowanie pakietu IP do klasy FEC skojarzonej z utworzoną ścieżką.

Oryginalny protokół LDP bazuje przy wyznaczaniu tras na dostępnych protokołach rutingu IP, uniemożliwiając odgórne wskazanie drogi przez operatora lub ruter wejściowy. Nie umożliwia również rezerwacji pasma ścieżki. Z tych powodów nie jest możliwe wykorzystywanie go do najistotniejszych metod inżynierii ruchu, w tym technik protekcji ścieżek. W odpowiedzi na te niedostatki zaproponowano *Constrained-based Routing Label Distribution Protocol* (CR-LDP) [43], w którym wprowadzono niezbędne rozszerzenia.

Wszystkie protokoły: RSVP-TE, LDP i CR-LDP mają wiele cech wspólnych, gdyż wykorzystują podobne zasady tworzenia ścieżek. Są jednak trzy obszary, w których ujawniają się podstawowe różnice pomiędzy nimi.

1. *Sposób wyboru drogi.* Protokół LDP zawsze wybiera drogę zgodnie z informacją uzyskaną od protokołu rutingu. Protokoły RSVP-TE i CR-LDP pozwalają natomiast na autorytatywne ustalenie drogi lub wykorzystanie udoskonalonych protokołów rutingu, które w procesie wyboru drogi biorą pod uwagę dostępne pasmo.
2. *Warstwy komunikacyjne.* Protokół RSVP-TE jest oparty bezpośrednio na warstwie IP, używając „zawodnej" komunikacji bez potwierdzeń, natomiast LDP i CR-LDP wykorzystują wiarygodne połączenia TCP do utrzymywania sesji związanych ze stanem ścieżek oraz UDP (multicast) do nawiązania kontaktu z sąsiadami.
3. *Tryb sygnalizacji.* Protokoły LDP i CR-LDP wykorzystują model *hard-state*, w którym ścieżki są utrzymywane do momentu jawnego ich skasowania z użyciem odpowiedniej wiadomości sygnalizacyjnej. W stanie stabilnym żadne informacje o ścieżkach nie muszą być wymieniane. Z kolei w protokole RSVP-TE wymagane jest periodyczne odświeżanie informacji o istniejących ścieżkach. Skasowanie ścieżki może odbywać się po prostu poprzez zaprzestanie rozgłaszania informacji o tej ścieżce, choć odpowiedni typ wiadomości do tego celu został również przewidziany [22].

Najczęściej wykorzystywanym do tworzenia ścieżek jest obecnie protokół RSVP-TE. Protokół LDP, mimo, iż został opracowany specjalnie dla sieci MPLS, stracił na znaczeniu ze względu na brak rutingu źródłowego i brak możliwości rezerwacji pasma. Opracowanie jego rozszerzenia CR-LDP okazało się spóźnione i nie zmieniło tej sytuacji, a sam protokół nie jest nawet dostępny w urządzeniach wielu czołowych dostawców sprzętu, włączając Cisco Systems i Juniper Networks. W efekcie organizacja IETF ogłosiła w dokumencie RFC 3468 [6] zaprzestanie prac nad rozwojem protokołu LDP. Zdecydowano jednocześnie o wyborze RSVP-TE jako protokołu sygnalizacyjnego zalecanego dla sieci MPLS. Nie oznacza to zakazu stosowania protokołu LDP, którego największą zaletą jest prostota działania. Może on być nadal wykorzystywany tam, gdzie jego cechy są wystarczające.

2.4. Podsumowanie

Koncepcja sieci MPLS została opracowana w celu przyspieszenia przełączania pakietów w ruterach IP. Pomimo, że we współczesnych ruterach szybkość przełączania nie jest ograniczeniem, to sieci MPLS posiadają szereg zalet, które predestynują je do zastosowania jako platforma do wprowadzania mechanizmów inżynierii ruchu dla IP, takich jak rezerwacja pasma, sterowanie trasą i mechanizmy odtwarzania po awarii.

Typowym sposobem enkapsulacji pakietu IP w pakiet MPLS jest umieszczenie dodatkowego nagłówka „shim" o rozmiarze czterech bajtów pomiędzy nagłówek ramki Ethernet a nagłówek datagramu IPv4 lub IPv6, choć w warstwie łącza przewidziano również możli-

wość współpracy z protokołami PPP, ATM i Frame Relay. Najważniejszym polem nagłówka MPLS jest etykieta, która przyporządkowuje zbiorczy strumień IP do klasy FEC, która z kolei wiąże go z określoną ścieżką LSP. Etykieta ma formę identyfikatora liczbowego, który ma znaczenie lokalne, gdyż jego wartość zmienia się na każdym kolejnym łączu na drodze ścieżki. Z punktu widzenia warstwy IP sieć MPLS dostarcza tuneli i zapewnia przeźroczysty transport pakietów.

Budowanie ścieżki odbywa się za pomocą wybranego protokołu dystrybucji etykiet. Najpopularniejszym obecnie protokołem jest RSVP-TE, który powstał w wyniku rozszerzenia protokołu RSVP o mechanizmy dystrybucji etykiet. Zapewnia on nie tylko utworzenie ciągłej ścieżki od rutera wejściowego (ang. *ingress*) do wyjściowego (ang. *egress*), ale też określenie z góry drogi połączeniowej i rezerwację pasma. Jego wadą jest zasada *soft-state*, która wymaga periodycznego odświeżania informacji o wszystkich utworzonych ścieżkach, choć w tym zakresie wprowadzono udoskonalenia w celu zmniejszenia ilości wymienianej informacji sygnalizacyjnej. Pozostałe rzadziej używane protokoły dystrybucji etykiet to LDP i jego udoskonalona wersja CR-LDP. Niezależnie od używanego protokołu ścieżka tworzona jest tylko w jedną stronę, co dla zapewnienia dwukierunkowej komunikacji wymaga utworzenia na każdej drodze połączeniowej dwóch przeciwnie skierowanych ścieżek.

W kolejnym rozdziale pokazane zostanie, jak implementuje się wywłaszczanie w sieciach MPLS i jak w tym celu wykorzystuje się priorytet ścieżki. Przedstawione zostaną też ogólnie dostępne algorytmy wywłaszczania i co wynika z ich porównania. Zagadnienia te są bezpośrednio związane z tematem i celem pracy.

3. MECHANIZMY I ALGORYTMY WYWŁASZCZANIA W SIECIACH MPLS

W poprzednim rozdziale omówiono architekturę sieci MPLS w zakresie umożliwiającym teraz na znacznie szersze omówienie jednego z mechanizmów inżynierii ruchu w sieciach MPLS, jakim jest wywłaszczanie. Ten rozdział rozpoczęto od ogólnego omówienia techniki wywłaszczania, po czym dokładniej scharakteryzowano typowe procedury z nią związane. Omówiono najważniejsze regulacje dotyczące wywłaszczania zawarte w dokumentach RFC, a następnie przedstawiono zbiór pojęć ułatwiających opis algorytmów. W zasadniczej części tego rozdziału szczegółowo omówiono znane algorytmy heurystyczne oraz algorytmy optymalne. Na zakończenie dokonano porównania omawianych algorytmów w oparciu o zdefiniowane kryteria.

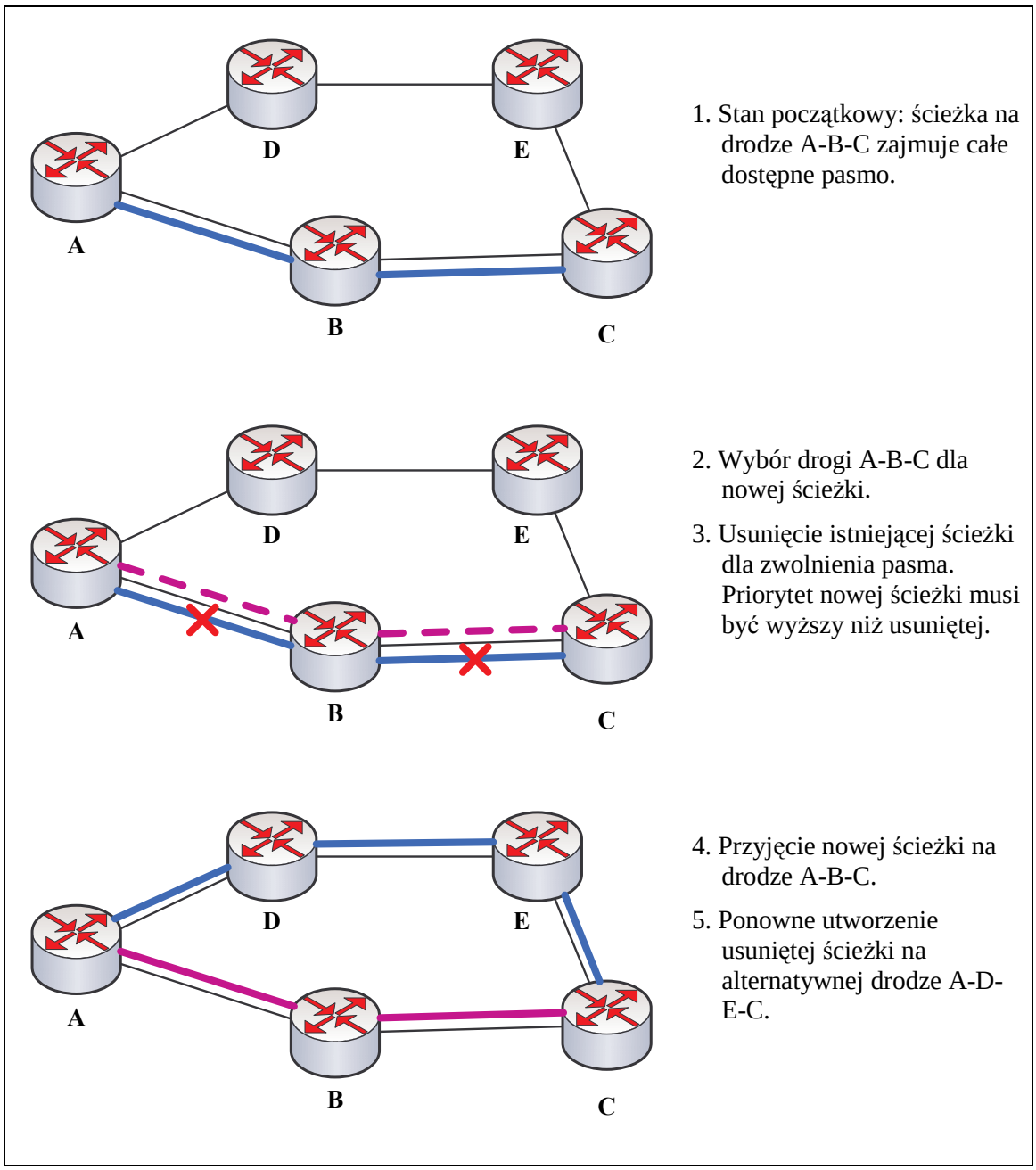
3.1. Wprowadzenie

W obrębie sieci MPLS połączenia IP są grupowane w strumienie i kierowane utworzonymi wcześniej ścieżkami LSP. Przez każde łącze fizyczne może przebiegać wiele logicznie niezależnych od siebie ścieżek. Jednak mimo tej niezależności ruch transportowany w ścieżkach rywalizuje o dostęp do wspólnego łącza fizycznego, a to, podobnie jak w tradycyjnej sieci IP, wprowadza zjawiska takie jak straty pakietów i zmienne opóźnienie pakietów. O ile w ogólności nie jest możliwe zupełne uniknięcie strat, to należy dążyć do ich ograniczania, a jednym ze sposobów jest rezerwacja określonego pasma dla ścieżki.

Rezerwacja pasma oznacza wydzielenie dla ścieżki części wolnego pasma łącza, a zatem po każdej rezerwacji zmniejsza się ilość pasma dostępnego dla ścieżek tworzonych później. W konsekwencji w pewnym momencie może zabraknąć pasma dla nowych zapotrzebowań i będą one odrzucane. Szczególnym problemem jest to wtedy, gdy brakuje pasma dla ścieżki o wysokim priorytecie, której utworzenie jest konieczne. Możliwym rozwiązaniem w takiej sytuacji jest wywłaszczanie, które polega na usunięciu jednej lub więcej utworzonych wcześniej ścieżek o niższym priorytecie po to, aby zwolnić przydzielone im pasmo i umożliwić utworzenie ścieżki o wyższym priorytecie. W niewielkich sieciach możliwe jest wykonanie tego ręcznie przez interwencję administratora po przeanalizowaniu dostępnych ścieżek. Może to być jednak zadanie bardzo żmudne i narażone na błędy, a w większych sieciach wręcz niemożliwe. Niezbędna jest zatem automatyzacja tego procesu i tym właśnie zajmują się algorytmy wywłaszczania.

Na rys. 3.1 przedstawiono na przykładzie zasadę działania wywłaszczeń w sieci MPLS. Dla uproszczenia założono, że dla istniejącej ścieżki utworzonej na drodze A-B-C zarezerwowano całe pasmo dostępne na łączach A-B i B-C, tak że nie ma możliwości dokonania na tej samej trasie żadnej innej rezerwacji (1). Dodatkowo założono, że istniejąca ścieżka ma umiarkowane wymagania na wielkość opóźnienia i może w razie potrzeby być przeniesiona

na nieco dłuższą trasę. W takiej sytuacji pojawia się nowe żądanie rezerwacji pasma dla ścieżki przenoszącej ruch wymagający najkrótszej możliwej drogi od węzła A do C, a więc A-B-C (2). Ponieważ zgodnie z założeniem na trasie A-B-C nie ma wolnego pasma, zatem w celu uniknięcia odrzucenia nowego żądania, istniejąca ścieżka musi zostać wywłaszczona. Zostaje ona tymczasowo usunięta (3), a w jej miejsce przyjęta zostaje nowa ścieżka (4). Następnie usunięta ścieżka zostaje przyjęta ponownie, ale już na innej trasie, w tym przypadku A-D-E-C (5). Po pomyślnym zakończeniu procedury obie ścieżki są ulokowane w sieci, mając zagwarantowane potrzebne pasmo i trasę spełniającą ich wymagania.



Rys. 3.1. Istota wywłaszczania. Przyjęcie nowej ścieżki o wyższym priorytecie (fioletowa) kosztem istniejącej ścieżki (niebieska).

Mechanizm wywłaszczania związany jest z pojęciem priorytetów ścieżek. Aby dana ścieżka mogła wywłaszczyć inną, musi mieć od niej wyższy priorytet. O przypisaniu priorytetów do ścieżek decyduje dostawca sieci na podstawie kontraktu z klientem.

Dość oczywistym warunkiem wywłaszczania jest to, że na drodze wybranej dla nowej ścieżki przynajmniej na jednym z łączy wolne pasmo jest niewystarczające. Nie stosuje się wywłaszczania, gdy pasmo jest wystarczające, tzn. w celu jedynie optymalizacji zasobów, gdyż każde przesunięcie ścieżki na nową drogę może być odczuwalne przez użytkowników sieci i objawiać się choćby chwilową utratą komunikacji.

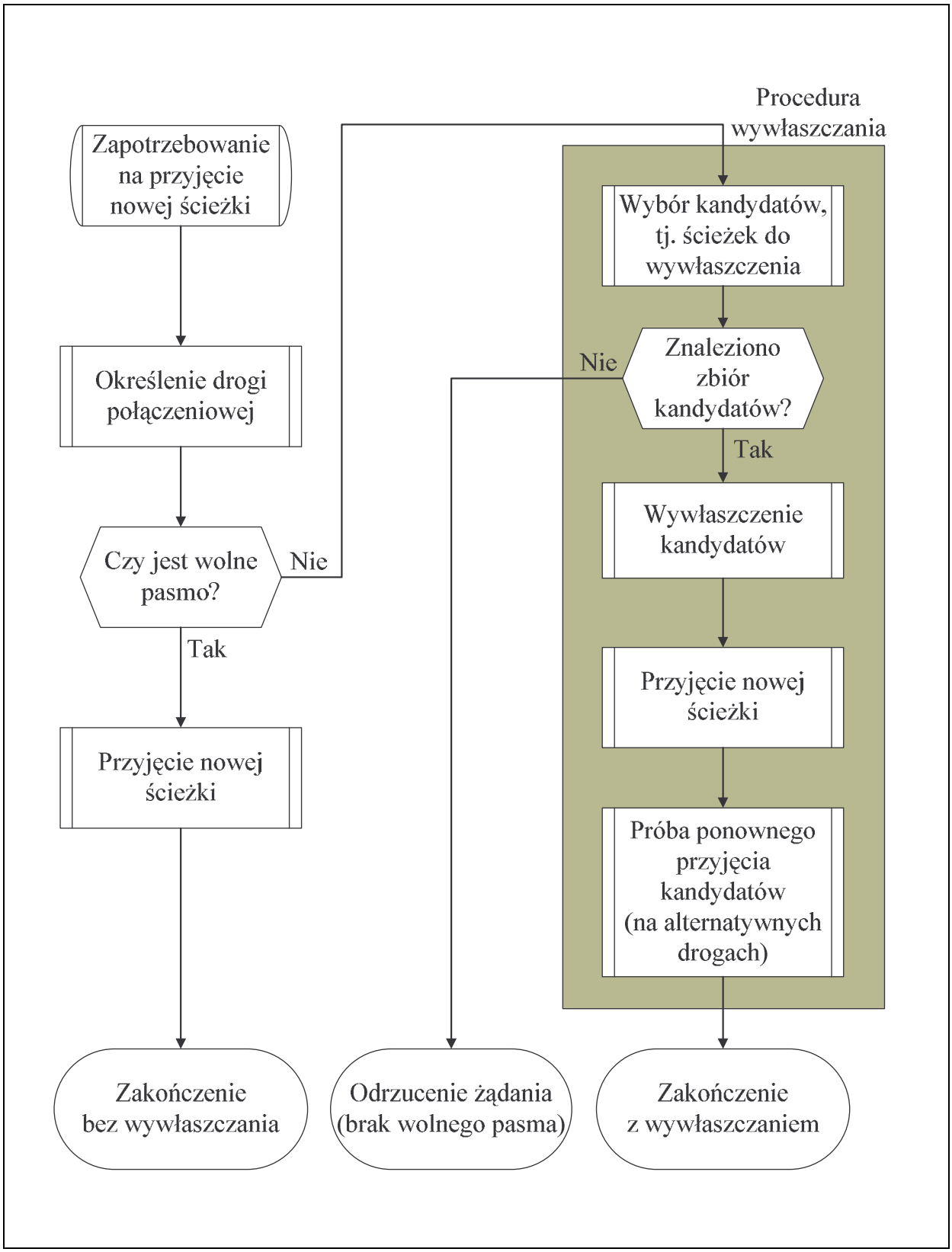
Alternatywą dla wywłaszczania jest podjęcie próby wyboru alternatywnej drogi dla nowej ścieżki, zwykle dłuższej od wybranej w pierwszej kolejności [40,49]. Korzyścią jest oczywiście zmniejszenie liczby wywłaszczeń, ale takie podejście może skutkować nieoptymalnym wykorzystaniem zasobów. Stanie się tak, jeśli nowa ścieżka przenosząca ruch wymagający najlepszej możliwej trasy zostanie przyjęta bez wywłaszczania na dłuższej trasie, gdyż najkrótsze trasy będą zajęte przez ścieżki o niższym priorytecie, dla których sposób wyboru trasy może nie mieć istotnego znaczenia.

Można powiedzieć, że rolą wywłaszczania jest uporządkowanie dostępu do zasobów w zależności od wymagań określonych dla poszczególnych ścieżek, minimalizując wpływ kolejności ich tworzenia. Jest to istotna korzyść i ważny aspekt inżynierii ruchu, dlatego wywłaszczanie jest dziś udostępniane przez najważniejszych dostawców sprzętu sieciowego. Należy jednak być świadomym, że dzieje się to kosztem zakłóceń obsługi w ruchu o niższym priorytecie.

3.2. Procedury wywłaszczania

Schemat postępowania przy przyjęciu nowej ścieżki z uwzględnieniem wywłaszczania został pokazany na rys. 3.2. Po wyznaczeniu drogi połączeniowej dla nowej ścieżki następuje sprawdzenie, czy na wybranej trasie dostępne jest wymagane pasmo. Jeśli tak, wówczas ścieżka zostaje utworzona bez wywłaszczania. Jeśli natomiast pasma brakuje, wówczas uruchomiona zostaje procedura wywłaszczania. Jej pierwszym i najważniejszym etapem jest algorytm wyboru ścieżek (inaczej kandydatów) przeznaczonych do wywłaszczenia. Kandydatów wybiera się w pętli tak długo, aż uzyska się pasmo wystarczające na przyjęcie nowej ścieżki. Jeśli procedura się powiedzie, wówczas następuje usunięcie, czyli wywłaszczenie, kandydatów i utworzenie w ich miejsce nowej ścieżki. Następnie podejmuje się próbę ponownego utworzenia usuniętych ścieżek na drogach alternatywnych. Jeśli algorytm nie znajduje ścieżek, które można wywłaszczyć aby uzyskać brakujące pasmo, to ścieżka nie może zostać utworzona a żądanie zostaje odrzucone.

Warto zauważyć, że ponowne utworzenie wywłaszczonych ścieżek może doprowadzić do wywłaszczenia kolejnych ścieżek. Mamy wówczas do czynienia ze zjawiskiem kaskady wywłaszczeń, które zostało opisane w rozdziale 3.4.6.



Rys. 3.2. Ogólna procedura utworzenia ścieżki z uwzględnieniem wywłaszczania.

Procedura selekcji kandydatów ma szansę zakończyć się powodzeniem pod warunkiem, że na wszystkich łączach, gdzie konieczne jest zwolnienie brakującego pasma, istnieje przynajmniej jeden zbiór ścieżek zapewniający jego odzyskanie. Jeśli na którymś łączu suma

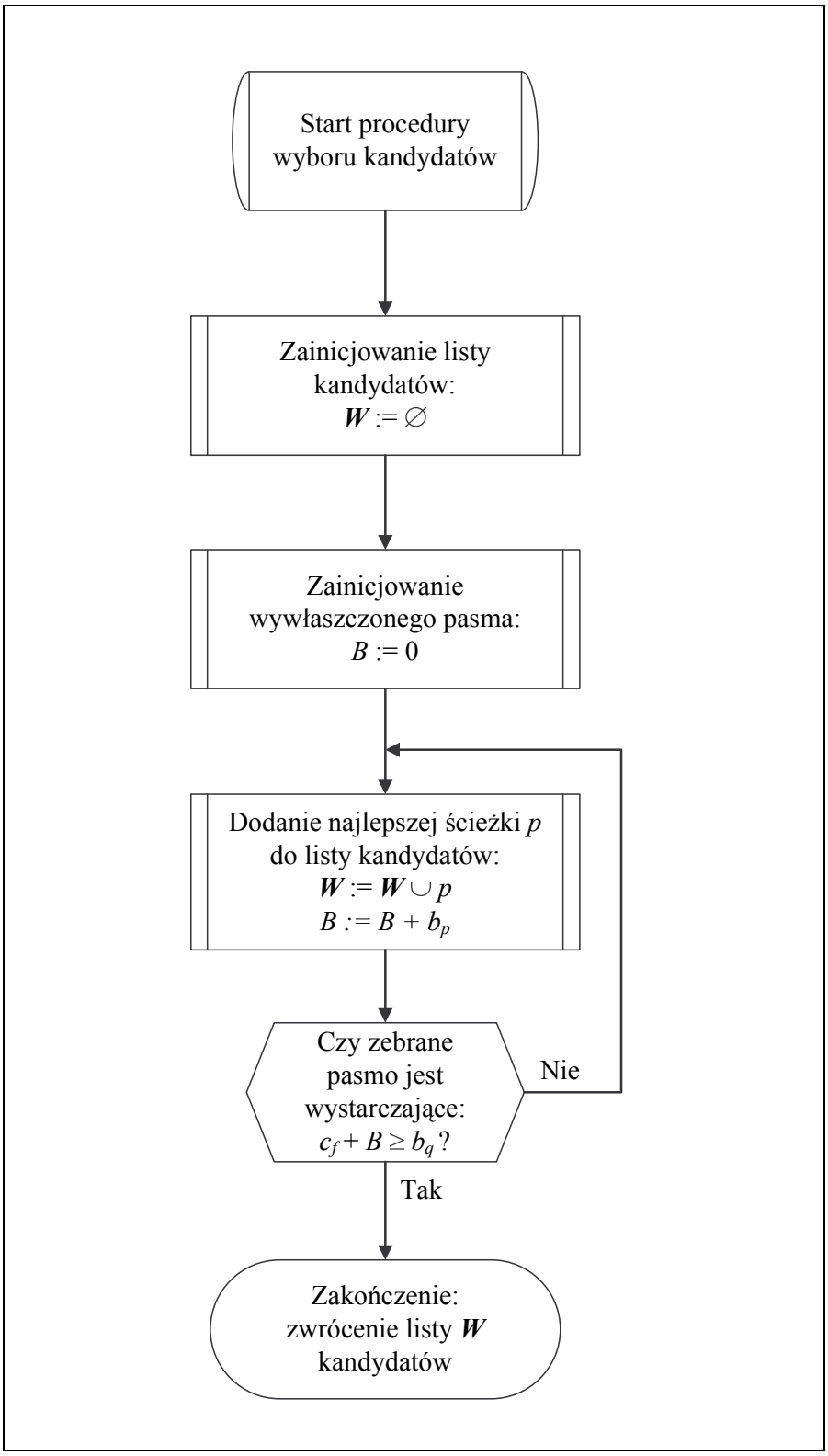
pasm zarezerwowanych dla wszystkich potencjalnych kandydatów jest na to zbyt mała, wówczas nawet wywłaszczenie wszystkich dostępnych ścieżek nie doprowadziłoby do zwolnienia wymaganego pasma. Jeśli rozpoczęto by wywłaszczanie kandydatów zanim algorytm zakończy się sukcesem, to przeprowadzone wywłaszczenia spowodowałyby niepotrzebną uciążliwość dla użytkowników sieci. Ryzyko wystąpienia takiej sytuacji minimalizuje się poprzez wyposażenie algorytmu rutingu w informacje o poziomach pasma dostępnego dla ścieżek z podziałem na priorytety, tzn. uwzględniając pasmo wszystkich potencjalnych kandydatów do wywłaszczenia. Nie można takiej sytuacji jednak zupełnie wykluczyć, gdyż w praktyce informacje w bazie algorytmu rutingu mogą nie być aktualne, a wówczas algorytm może wskazać drogę, która w rzeczywistości już nie posiada odpowiedniej ilości pasma. Dodatkową przyczyną niepowodzenia może być fakt braku kandydatów na innych łączach wymagających wywłaszczania. Ewentualność taką można ograniczyć przez użycie globalnych algorytmów wywłaszczania, ale nie da się jej uniknąć jeśli wykorzystujemy algorytm lokalny, który nie ma informacji o sytuacji na dalszych łączach na drodze połączeniowej.

O ile ogólny schemat działań prowadzących do wywłaszczania jest dobrze zdefiniowany [7,84,91] i nie wymaga pogłębionych badań, o tyle sposób wyboru kandydatów jest zagadnieniem złożonym, którym różnią się poszczególne algorytmy. Trudność polega na tym, że pasma poszczególnych ścieżek zwykle różnią się od siebie, a przez to nie można z góry określić ani tego, ile ścieżek należy wybrać, ani jakie pasma powinny mieć wybrane ścieżki. Łatwo bowiem wyobrazić sobie sytuację, w której kilka mniejszych ścieżek lepiej spełni wymaganie na brakujące pasmo niż jedna większa ścieżka, która zwolniłaby zbyt dużo pasma.

Większość z istniejących algorytmów działa według schematu przedstawionego na rys. 3.3. Procedura rozpoczyna się od pustej listy $\boldsymbol{W}$ kandydatów, do której w każdym przebiegu pętli dodaje się wybraną ścieżkę p o paśmie b_p, która najlepiej pasuje do zapotrzebowania określonego przez nową ścieżkę o paśmie b_q. Po każdej takiej operacji oblicza się bilans pasma, tj. określa się, jak zmieni się wolne pasmo, jeśli się wywłaszczy wszystkich kandydatów z listy $\boldsymbol{W}$. Wprowadzono tu symbole: B na oznaczenie zebranego pasma i c_f na oznaczenie wolnego pasma na łączu. Jeśli bilans jest dodatni, tzn. pasmo po dokonaniu wywłaszczeń będzie wystarczające dla przyjęcia nowej ścieżki, wówczas pętla zostaje zakończona a lista kandydatów jest kompletna, co pozwala zakończyć działanie algorytmu. Najważniejsza i najtrudniejsza zarazem do rozstrzygnięcia kwestia to wybranie najodpowiedniejszego w danej sytuacji kandydata do wywłaszczenia.

Istnieją dwa odmienne podejścia do problemu wyboru ścieżek przeznaczonych do wywłaszczenia, wynikające z zasięgu algorytmu. Zasięg określa to, jak szeroką wiedzą na temat zarządzanej domeny posługuje się algorytm:

- algorytm lokalny obejmuje pojedyncze łącze,
- algorytm globalny obejmuje całą domenę.



Rys. 3.3. Ogólna procedura wyboru ścieżek do wywłaszczenia.

Algorytm lokalny przeprowadza analizę w oparciu o stan na bieżącym łączu bez uwzględniania wpływu podejmowanej decyzji na pozostałe łącza ścieżki. Metoda tego typu nie wymaga dostępu do informacji o długości wybieranych ścieżek oraz o ścieżkach dostępnych na innych łączach. Ograniczenie zasięgu do pojedynczego łącza umożliwia znaczne

uproszczenie algorytmu, jednak kosztem zmniejszenia efektywności rozwiązania w skali całej domeny. Dane wymagane do realizacji metody lokalnej obejmują:

- pasma zarezerwowane dla dostępnych ścieżek,
- priorytety dostępnych ścieżek,
- wolne pasmo na danym łączu (porcie).

Algorytm globalny bierze pod uwagę szerszy kontekst, gdyż „widzi" całą domenę i podejmuje decyzję o wyborze kandydatów po przeanalizowaniu stanu na całej drodze połączeniowej. Takie podejście potencjalnie umożliwia osiągnięcie lepszych rezultatów w sytuacji braku pasma na więcej niż jednym łączu, jednak kosztem zwiększenia złożoności obliczeniowej. W stosunku do metody lokalnej następujące dane mogą być dodatkowo brane pod uwagę:

- trasy dostępnych ścieżek,
- wolne pasma na łączach na wybranej drodze.

Przedstawiony podział jest analogiczny do tego, jaki wprowadzono do klasyfikacji protokołów rutingu IP. Odpowiednikiem algorytmów lokalnych są metody typu *distance-vector* (np. RIP) natomiast odpowiednikiem metod o zasięgu domeny są protokoły typu *link-state* (np. IS-IS, OSPF).

Podział algorytmów na lokalne i globalne może sugerować, że pierwsze z nich są realizowane jako metody zdecentralizowane, a drugie jako scentralizowane. W rzeczywistości zasięg i sposób realizacji są od siebie niezależne. To sposób realizacji określa, gdzie w sieci wykonywany jest algorytm. W przypadku metod scentralizowanych jest to dedykowany blok, w którym określa się ścieżki przeznaczone do wywłaszczania. Z kolei w metodach zdecentralizowanych każdy ruter podejmuje decyzję niezależnie. Algorytm globalny może być elementem zarówno metody scentralizowanej, jak i zdecentralizowanej. Różnica wynika z tego, czy wszystkie wymagane przez algorytm informacje są dostępne w każdym ruterze, czy tylko w jednym, dedykowanym urządzeniu w domenie. Z kolei do realizacji algorytmu lokalnego naturalnym podejściem jest metoda zdecentralizowana. Podejście scentralizowane, choć teoretycznie możliwe do zrealizowania, byłoby tu jednak nielogiczne.

Warto zauważyć, że w kontekście ostatnich prac IETF nad architekturą *Path Computation Element* (PCE) [34] algorytmy globalne zyskują na znaczeniu. Zgodnie z koncepcją PCE, zadania związane z wyznaczeniem tras w domenach MPLS i GMPLS mogą zostać wyniesione poza ruter brzegowy, do dedykowanego serwera w domenie. Ten na podstawie określonych przez ruter brzegowy wymagań zwraca informację o trasie. Umożliwia to wykonywanie złożonych czasowo zadań bez angażowania mocy obliczeniowej procesorów rutera. Architektura PCE wydaje się być odpowiednia także do wykonywania obliczeń związanych z wywłaszczaniem, analogicznie do roli bloków centralnych w projekcie serwera sterowania połączeniami [54].

Implementując dowolny algorytm, niezależnie od zasięgu, definiuje się w nim w sposób

jawny lub niejawny miarę porównawczą dla oceny poszczególnych zbiorów kandydatów. Chodzi o wybranie najlepszego zbioru kandydatów spośród wszystkich dostępnych ścieżek. Ocena zależy od wybranego kryterium porównawczego, którym może być:

- minimalizacja liczby wywłaszczonych ścieżek,
- minimalizacja sumy pasm wywłaszczonych ścieżek,
- minimalizacja sumy pasm sieciowych wywłaszczonych ścieżek (iloczynu pasm i długości ścieżek),
- maksymalizacja (w sensie liczbowym) sumy priorytetów wywłaszczonych ścieżek,
- minimalizacja długości wywłaszczonych ścieżek.

Lista powyższa nie jest zamknięta i możliwe jest zdefiniowane innych kryteriów. Dodatkowo zwykle możliwe jest łączenie kilku kryteriów, np. minimalizacja liczby ścieżek, a w przypadku dostępności kilku równoważnych zbiorów, minimalizacja pasma. Należy dodać, iż niektóre metody umożliwiają użytkownikowi dokonanie wyboru preferowanego kryterium.

To, na ile algorytm globalny umożliwi wybranie lepszego zbioru kandydatów niż algorytm lokalny, zależy od liczby łączy, na których nastąpić ma wywłaszczenie. Jeśli dotyczy to tylko jednego łącza, wtedy algorytm lokalny poradzi sobie równie dobrze jak globalny. W przypadku, gdy łączy wymagających wywłaszczenia jest więcej, wówczas dobry algorytm globalny w ogólności powinien dawać lepsze rezultaty.

Udowodniono [35] bardzo ważne twierdzenie mówiące o tym, że zarówno problem minimalizacji liczby ścieżek, jak i minimalizacji pasma w procesie wywłaszczania o zasięgu globalnym jest NP-trudny [59,78]. Innymi słowy, algorytm optymalny w ogólnym przypadku musi dokonać porównania wszystkich możliwych kombinacji ścieżek. Złożoność obliczeniowa takiej optymalnej metody jest zbyt duża aby zastosować ją w praktyce. Konieczne jest zatem poszukiwanie algorytmów heurystycznych, które dają zwykle wynik gorszy od optymalnego, ale za to oferują akceptowany czas wykonania. Algorytm optymalny można zastosować w ograniczonym zakresie jedynie w warunkach testowych jako podstawę porównawczą dla metod heurystycznych.

3.3. Wywłaszczanie w świetle standardów

Przed omówieniem poszczególnych dostępnych algorytmów, zostanie przedstawiony stan wymagań i zaleceń określających sposób przeprowadzenia wywłaszczeń.

W odniesieniu do sieci MPLS, najwcześniejsze, choć częściowo już nieaktualne założenia dotyczące wywłaszczania ścieżek zostały zawarte w dokumencie RFC 2702 [9], w którym zdefiniowano cztery atrybuty ścieżek: *preemptor enabled* (1), *non-preemptor* (2), *preemptable* (3) oraz *non-preemptable* (4). Właściwość (1) oznacza możliwość spowodowania wywłaszczenia innych ścieżek natomiast (2) brak takiej możliwości. Z kolei (3) odpowiada za

możliwość zostania wywłaszczonym przez inną ścieżkę, a (4) wyklucza taką ewentualność. Za domyślne uznaje się atrybuty (2) i (4), które wspólnie wykluczają powstanie wywłaszczeń.

W tym samym dokumencie określa się, że ścieżki posiadają priorytet, który w połączeniu z odpowiednim atrybutem może spowodować wywłaszczenie. Określono także pięć niezbędnych warunków wywłaszczenia ścieżki p_2 przez ścieżkę p_1:

1. p_1 ma wyższy priorytet niż p_2,
2. p_1 domaga się zasobów zajętych przez p_2,
3. zasobów jest zbyt mało, aby przyjąć jednocześnie ścieżki p_1 i p_2,
4. p_1 posiada atrybut *preemptor enabled* oraz
5. p_2 posiada atrybut *preemptable*.

W świetle późniejszych rekomendacji atrybuty typu *preemptor* i *preemptable* należy uznać za nieaktualne. Ich rolę zajęły regulacje określone w standardzie RSVP-TE [7], który jest najpopularniejszym narzędziem używanym do rezerwacji pasma w sieciach MPLS. Zdefiniowano w nim dwa typy priorytetów, deklarowanych w procesie tworzenia ścieżki:

- priorytet utworzenia (*setup priority*), oraz
- priorytet utrzymania (*holding priority*).

Znaczenie tych parametrów jest następujące. Priorytet utworzenia określa priorytet ścieżki na etapie jej tworzenia, tzn. określa zdolność do wywłaszczenia innych ścieżek. Z kolei priorytet utrzymania określa priorytet przypisany ścieżce po jej utworzeniu, tzn. decyduje, w jakim stopniu dana ścieżka jest „odporna” na wywłaszczenie przez inne ścieżki. Oba priorytety określane są wartościami liczbowymi z zakresu od 0 do 7, przy czym 0 oznacza najwyższy priorytet a 7 – najniższy. Wymaga się, aby priorytet utworzenia nie był wyższy niż priorytet utrzymania, gdyż w przeciwnym wypadku grozi to powstaniem nieskończonej pętli wzajemnych wywłaszczeń. W niektórych praktycznych realizacjach jako domyślny uznaje się priorytet utworzenia równy 7 oraz priorytet utrzymania równy 0, co gwarantuje brak możliwości spowodowania wywłaszczeń jak też pełną odporność na wywłaszczenia ze strony innych ścieżek niezależnie od ich priorytetu.

Dla porządku warto dodać, iż koncepcja wywłaszczania istniała już w oryginalnym protokole RSVP zaproponowanym wcześniej niż powstała technologia MPLS, gdzie zdefiniowano parametry *Preemption Priority* oraz *Defending Priority* [41]. Oba priorytety mogły przyjmować wartości z zakresu liczb 16-bitowych bez znaku, przy czym wyższa wartość odpowiadała wyższemu priorytetowi. W kontekście sieci MPLS specyfikację tę należy uznać za nieaktualną na rzecz standardu RSVP-TE.

W odniesieniu do samej procedury wywłaszczania standard RSVP-TE określa, że ścieżka, która podlega wywłaszczeniu, zostaje natychmiast skasowana. Obecnie podejście takie określane jest jako *hard preemption*, w przeciwieństwie do nowszej koncepcji, tzw. *soft preemption*, zdefiniowanego w RFC 5712 [65]. W tym ostatnim podejściu ścieżka przeznaczona do wywłaszczenia nie jest od razu kasowana, ale zostaje skierowana na nową drogę. Ścieżka

na starej drodze pozostaje jeszcze przez pewien czas aktywna, dzięki czemu nie zostają stracone pakiety, które w momencie wywłaszczenia znajdują się na starej trasie. Możliwość taka jest już udostępniona przez czołowych producentów ruterów MPLS [3,36]. Należy jednak podkreślić, iż wykorzystywanie techniki *soft preemption* nie daje żadnej gwarancji uniknięcia przerw w ruchu. Mogą one wciąż się zdarzyć z powodu zaburzenia kolejności dostarczania pakietów [29] lub przejściowego powstania nadmiernych strat.

Mechanizm wywłaszczania opisano także w specyfikacji protokołu CR-LDP [43]. Tutaj również użyto analogicznych do RSVP-TE pojęć *setupPriority* i *holdingPriority* o wartościach z zakresu od 0 do 7, gdzie 0 oznacza najwyższy priorytet ścieżki. Dokument zawiera zalecenie, aby domyślne wartości obu parametrów wynosiły 4.

Przedstawione zalecenia mają swoje odzwierciedlenie w zasadzie działania algorytmów wywłaszczania w sieciach MPLS. Można je podsumować następująco.

1. Każda ścieżka posiada dwa typy priorytetów: priorytet utworzenia (*setup*) i priorytet utrzymania (*holding*).
2. Priorytety są liczbami całkowitymi z zakresu od 0 do 7, tzn. jest osiem różnych poziomów priorytetów. Wartość 0 oznacza najwyższy, a 7 najniższy priorytet.
3. Priorytet utrzymania nie może być niższy (liczbowo wyższy) od priorytetu utworzenia (ale może być równy).
4. Aby ścieżka p_1 mogła dokonać wywłaszczenia ścieżki p_2 muszą być spełnione trzy warunki:
   - priorytet utworzenia p_1 musi być wyższy (liczbowo niższy) od priorytetu utrzymania p_2,
   - p_1 domaga się zasobów zajętych przez p_2, oraz
   - nie ma możliwości przydzielenia zasobów dla p_1 i p_2 jednocześnie.

3.4. Definicje

Jednoznaczny opis algorytmów nie byłby możliwy bez omówienia i formalnego zdefiniowania używanych pojęć i wielkości. W niniejszej części zebrano te definicje, które są niezbędne do jednoznacznego opisu algorytmów wywłaszczania i ich jakości.

3.4.1. Jakość algorytmów

Oceniając **jakość algorytmów** oceniamy tu zdolność do generowania takich zbiorów kandydatów do wywłaszczenia, dla których koszt wywłaszczenia jest jak najmniejszy.

Definicja kosztu wywłaszczania jest zależna w dużej mierze od technologii, w jakiej algorytm jest zaimplementowany. Jeśli będzie to sieć, w której usunięcie ścieżek jest kosztowne z technologicznego lub biznesowego punktu widzenia, wówczas koszt będzie prawdopodobnie zdefiniowany jako liczba wywłaszczeń. W innej sytuacji może to być priorytet wywłaszczonych ścieżek. Najczęściej spotykane, choć nie jedyne, definicje kosztów to:

- liczba wywłaszczeń, w którym dąży się do minimalizacji liczby wybranych kandydatów,
- wywłaszczone pasmo, w którym dąży się do minimalizacji sumarycznego pasma kandydatów,
- wywłaszczone pasmo sieciowe, w którym kosztem jest iloczyn pasma ścieżki i liczby łączy, na których jest utworzona, czego celem jest częstsze wywłaszczanie ścieżek krótkich przy zachowaniu ścieżek dłuższych, dla których wywłaszczenie jest bardziej złożone,
- priorytety wywłaszczonych ścieżek, w którym dąży się do wyboru w pierwszej kolejności ścieżek o najniższym priorytecie.

Jeśli algorytm pozwala na wybór definicji kosztów, wówczas decyzję podejmuje zwykle administrator sieci, zwany tutaj użytkownikiem, bazując na znajomości technologii i otoczenia biznesowego w ramach zarządzanej domeny MPLS. Mówimy wówczas, że użytkownik ma możliwość podania **kryterium wyboru kandydatów**.

Zdefiniowana w ten sposób jakość jest jedną z kluczowych cech algorytmów z punktu widzenia użytkownika, gdyż przekłada się bezpośrednio na wysokość kosztów ponoszonych przez niego w efekcie przeprowadzonych wywłaszczeń. Innymi istotnymi cechami algorytmów są jego złożoność obliczeniowa a także łatwość implementacji.

Jakość jako cecha algorytmu wynika w największej części ze sposobu wyboru ścieżek przeznaczonych do wywłaszczenia, a więc tego, w jaki sposób algorytm został zaprojektowany. Niezależnym od algorytmu czynnikiem wpływającym na jego jakość może być sposób definicji kosztu przez użytkownika, o ile algorytm daje taką możliwość, a jednocześnie przełożenie kosztu na sposób konfiguracji algorytmu nie jest oczywiste. Na jakość może mieć również wpływ sposób implementacji, o ile dokonano zmiany w stosunku do oryginalnej specyfikacji algorytmu.

Jakość algorytmów jest trudna do oszacowania w drodze teoretycznej analizy jego zasady działania, ale może być zbadana np. przy użyciu metod symulacyjnych. Pewną trudność w tym przypadku sprawia określenie, czy osiągany wynik świadczy o dobrej czy złej jakości, gdyż nie dysponujemy wartościami odniesienia. Możliwe jest jednak określenie jakości algorytmów poprzez badania porównawcze. W ten sposób określa się, że algorytm, który osiągnął niższy średni koszt wywłaszczania, jest lepszy.

3.4.2. Sieć i ścieżki

Sieć jest opisana przez graf ($\boldsymbol{V}$, $\boldsymbol{E}$, $\boldsymbol{C}$):

- $\boldsymbol{V}$ jest zbiorem węzłów sieci; $\boldsymbol{V} = \{v_1, v_2, ..., v_n\}$, gdzie $N=|\boldsymbol{V}|$ jest liczbą węzłów,
- $\boldsymbol{E}$ jest zbiorem łączy w sieci; $\boldsymbol{E} = \{e_1, e_2, ..., e_l\}$, gdzie $L=|\boldsymbol{E}|$ jest liczbą łączy; wszystkie łącza są jednokierunkowe,
- $\boldsymbol{C}$ jest zbiorem pasm (przepływności) łączy fizycznych w sieci; $\boldsymbol{C} = \{c_1, c_2, ..., c_L\}$; jeśli przepływności wszystkich łączy są równe to oznaczamy je symbolem c.

Gęstość d sieci jest to stosunek liczby łączy fizycznych (jednokierunkowych) do liczby węzłów sieci.

$$d = \frac{L}{N} \tag{3.1}$$

Długość l ścieżki jest rozumiana jako liczba łączy pośrednich, z których składa się droga (trasa) ścieżki LSP, od rutera wejściowego do wyjściowego.

Podobnie zdefiniowana jest długość drogi. Jest to liczba łączy, jakie dzielą dwa rutery LSR przy wykorzystaniu określonej drogi. Pojęcie długości drogi może dotyczyć konkretnej ścieżki, ale może też być użyte bez powiązania ze ścieżką.

Ścieżka p jest opisana przez cztery wielkości $P_p = (R_p, b_p, s_p, h_p)$:

- R_p jest listą łączy, na których ścieżka została utworzona, $R_p = [e_p^{(1)}, e_p^{(2)}, ..., e_p^{(l_p)}]$, gdzie $l_p = |R_p|$ jest długością ścieżki, wszystkie ścieżki są jednokierunkowe,
- b_p jest pasmem (przepływnością) zarezerwowanym dla ścieżki,
- s_p jest priorytetem utworzenia (*setup*) ścieżki, $s_p \in \{0, 1, ..., 7\}$, przy czym 0 oznacza najwyższy priorytet,
- h_p jest priorytetem utrzymania (*holding*) ścieżki, $h_p \in \{0, 1, ..., 7\}$, przy czym 0 oznacza najwyższy priorytet.

Zbiór $\boldsymbol{P}$ zawiera ścieżki utworzone w sieci $\boldsymbol{P} = \{P_1, P_2, ..., P_Z\}$, gdzie Z jest liczbą ścieżek.

Aby uprościć zapis pewnych definicji, zdefiniujmy ogólną funkcję testującą σ(), która dla danej funkcji logicznej $f()$ przyjmie wartość jeden wtedy, gdy funkcja $f()$ będzie prawdziwa, a zero w przeciwnym wypadku, tj. gdy warunek w funkcji $f()$ nie będzie spełniony.

$$\sigma(f()) = \begin{cases} 1 & \text{gdy } f() \text{ jest prawdziwe} \\ 0 & \text{gdy } f() \text{ jest nieprawdziwe} \end{cases} \tag{3.2}$$

3.4.3. Wolne pasmo i dostępne pasmo

Wolne pasmo A_f w sieci: $A_f = [a_f^{(1)}, a_f^{(2)}, ..., a_f^{(L)}]$ oznacza pasma na poszczególnych łączach, jakie nie zostały jeszcze zarezerwowane i są dostępne dla nowej ścieżki bez potrzeby wywłaszczania.

Zajęte pasmo A_{oc} w sieci, $A_{oc} = [a_{oc}^{(1)}, a_{oc}^{(2)}, ..., a_{oc}^{(L)}]$ jest pasmem zarezerwowanym dla ścieżek na poszczególnych łączach.

$$a_{oc}^{(e)} = \sum_{p \in \boldsymbol{P}} \sigma\left(e \in R_p\right) b_p \tag{3.3}$$

Dostępne pasmo A w sieci określa pasma dostępne na poszczególnych łączach: $A = [A^{(1)}, A^{(2)}, ..., A^{(L)}]$, gdzie $A^{(e)}$ to pasmo dostępne na łączu e, $e \in \{1, ..., L\}$. Dostępne pasmo jest ilo-

ścią pasma, jakie może zostać przydzielone nowej ścieżce przy uwzględnieniu wywłaszczania. Pasmo dostępne jest tablicą ośmioelementową $A^{(e)} = [a_0^{(e)}, a_1^{(e)}, ... a_7^{(e)}]$. Na tej bazie definiuje się pasmo $a_s^{(e)}$ dostępne na łączu e dla ścieżki o priorytecie utworzenia s, które jest równe wartości wolnego pasma $a_f^{(e)}$ na łączu e powiększonej o sumę pasm ścieżek o priorytecie utrzymania h_p niższym (liczbowo wyższym) niż s:

$$a_s^{(e)} = a_f^{(e)} + \sum_{p \in \boldsymbol{P}} \sigma(e \in R_p)\,\sigma(h_p > s)\,b_p\,, \tag{3.4}$$

gdzie $s \in \{0, 1, ..., 7\}$.

Ponieważ każda ścieżka o priorytecie utworzenia równym s może wywłaszczyć każdą ścieżkę o priorytecie utrzymania h większym liczbowo od s, zatem pasmo dostępne dla ścieżek o priorytecie s, dla $s < 7$, jest zawsze większe lub równe pasmu dostępnemu dla ścieżki o priorytecie s+1, zatem:

$$a_0^{(e)} \geq a_1^{(e)} \geq ... \geq a_7^{(e)}. \tag{3.5}$$

Nowa ścieżka o najniższym priorytecie utworzenia $s = 7$ nie może spowodować wywłaszczenia żadnej ścieżki, więc pasmo dostępne dla takiej ścieżki jest równe z definicji wolnemu pasmu, tj. $a_7^{(e)} = a_f^{(e)}$.

Pasmo $b_p^{(e)}$ zajęte przez ścieżkę p na łączu e jest równe pasmu ścieżki p, jeśli ścieżka istnieje na łączu e, lub zero, jeśli ścieżka p nie przechodzi przez łącze e. Zakładamy tu, że pasmo ścieżki na całej jej długości jest jednakowe.

$$b_p^{(e)} = \begin{cases} b_p & \text{dla } e \in R_p \\ 0 & \text{dla } e \notin R_p \end{cases} \tag{3.6}$$

3.4.4. Wywłaszczanie

Niech nowa ścieżka q, która może spowodować wywłaszczenia, jest opisana przez $P_q = (R_q, b_q, s_q, h_q)$.

Lista $\hat{R} = \{\hat{e}_1, \hat{e}_2, ...\}$ zawiera te łącza należące do R_q, na których wolne pasmo jest mniejsze niż pasmo b_q, tzn. na których niezbędne jest wykonanie wywłaszczeń. Liczbę z łączy należących do $\hat{R}$ określa się jako liczbę łączy z wywłaszczaniem.

$$z = \left|\hat{R}\right| \tag{3.7}$$

Zbiór $\boldsymbol{K} \subset \boldsymbol{P}$ potencjalnych kandydatów do wywłaszczenia obejmuje te ścieżki ze zbioru $\boldsymbol{P}$ dla których spełnione są warunki:

- priorytet utrzymania h jest liczbowo wyższy priorytetowi utworzenia s_q nowej ścieżki,

- dzielą przynajmniej jedno łącze z nową ścieżką q oraz
- posiadają zarezerwowane niezerowe pasmo.

Można to formalnie opisać następująco:

$$\boldsymbol{K} = \{p : p \in \boldsymbol{P} \wedge h_p > s_q \wedge R_p \cap \hat{R} \neq \varnothing \wedge b_p > 0\}. \tag{3.8}$$

Dla danego łącza e zbiór $\boldsymbol{K_e} \subset \boldsymbol{K}$ zawiera te ścieżki ze zbioru potencjalnych kandydatów, które przechodzą przez łącze e.

$$\boldsymbol{K_e} = \{p : p \in \boldsymbol{K} \wedge e \in \hat{R}\} \tag{3.9}$$

Zbiór $\boldsymbol{W} \subset \boldsymbol{K}$, $\boldsymbol{W} = \{w_1, w_2, ...\}$, kandydatów do wywłaszczenia zawiera, na zakończenie algorytmu wywłaszczania, ścieżki przeznaczone do wywłaszczenia po to, aby można było przyjąć nową ścieżkę q i zarezerwować dla niej żądaną ilość pasma b_q.

Liczba M wywłaszczeń to liczba kandydatów, tj. ścieżek, które należy wywłaszczyć, aby przyjąć nową ścieżkę, zgodnie z wynikiem działania danego algorytmu.

$$M = |\boldsymbol{W}| \tag{3.10}$$

Wywłaszczone pasmo B jest sumą pasm kandydatów, tj. ścieżek przeznaczonych do wywłaszczenia.

$$B = \sum_{p \in \boldsymbol{W}} b_p \tag{3.11}$$

Wywłaszczone pasmo sieciowe B_n jest sumą iloczynów pasm i długości kandydatów (długość jest określona jako liczba łączy l_p, przez które przechodzi ścieżka-kandydat).

$$B_n = \sum_{p \in \boldsymbol{W}} b_p l_p \tag{3.12}$$

$$l_p = \left| R_p \right| \tag{3.13}$$

Bilans pasma $S = [S_1, S_2, ...]$ określa, w trakcie lub po zakończeniu działania algorytmu, różnicę pomiędzy pasmem jakie zostanie lub zostało wywłaszczone a minimalnym pasmem wymaganym do przyjęcia nowej ścieżki q :

$$S_e = \sum_{p \in \boldsymbol{W}} \beta(p,e) + a_f^{(e)} - b_q, \tag{3.14}$$

gdzie $e \in \hat{R}$, natomiast $\beta(p,e)$ jest funkcją zwracającą pasmo ścieżki p w przypadku, gdy ścieżka p przechodzi przez łącze e, lub zero w przeciwnym wypadku:

$$\beta(p,e) = \begin{cases} b_p & \text{gdy } e \in R_p \\ 0 & \text{gdy } e \notin R_p \end{cases}. \tag{3.15}$$

Globalny bilans pasma S_g osiąga wartość nieujemną tylko wtedy, gdy bilans pasma S jest nieujemny dla każdego łącza $e \in \hat{R}$, tzn. gdy zbiór kandydatów zapewnia pasmo niezbędne do przyjęcia nowej ścieżki (jest to warunek na sukces procedury wywłaszczania). Jest obliczany poprzez wybór najmniejszej wartości bilansu pasma S_e na łączach wymagających wywłaszczania na drodze połączeniowej:

$$S_g = \min_{e \in \hat{R}} S_e. \tag{3.16}$$

Poszukiwane pasmo $B_o = [B_o^{(1)}, B_o^{(2)}, \ldots]$ określa wartość pasma niezbędną dla przyjęcia nowej ścieżki q, na początku działania algorytmu:

$$B_o^{(e)} = \max\left(0, b_q - a_f^{(e)}\right), \tag{3.17}$$

gdzie $e \in \hat{R}$.

Wymagane pasmo $B_r = [B_r^{(1)}, B_r^{(2)}, ...]$ określa, w trakcie działania algorytmu, ile pasma na poszczególnych łączach należy jeszcze pozyskać w drodze powiększenia zbioru $\boldsymbol{W}$ o kolejnych kandydatów:

$$B_r^{(e)} = \max\left(0, -S_e\right), \tag{3.18}$$

gdzie $e \in \hat{R}$.

W oparciu o tablicę B_r można zdefiniować wymagane pasmo sieciowe B_{rnet} jako sumę wymaganych pasm na wszystkich łączach należących do drogi połączeniowej, wybranej dla ścieżki.

$$B_{rnet} = \sum_{e \in \hat{R}} B_r^{(e)} \tag{3.19}$$

Nadmiarowe (stracone) pasmo $B_x = [B_x^{(1)}, B_x^{(2)}, ...]$ oznacza, po zakończeniu działania algorytmu, pasmo, które zostanie lub zostało wywłaszczone ponad wymagane minimum, na wszystkich łączach e należących do drogi połączeniowej ścieżki p, tzn. również tych, na których nie brakuje wolnego pasma. Nadmiarowe pasmo jest wynikiem niedokładności algorytmu i/lub braku zbioru ścieżek, które dokładnie pokrywałyby się z zapotrzebowaniem na brakujące pasmo na wszystkich łączach:

$$B_x^{(e)} = \max\left(0, \sum_{p \in \boldsymbol{W}} \beta(p,e) + a_f^{(e)} - b_q\right), \tag{3.20}$$

gdzie $e \in R_p$.

Podobnie jak dla wymaganego pasma, zdefiniować można nadmiarowe (stracone) pasmo sieciowe jako sumę pasm straconych na wszystkich łączach należących do drogi połączeniowej wybranej dla nowej ścieżki.

$$B_{xnet} = \sum_{e \in R_p} B_x^{(e)} \tag{3.21}$$

Ponieważ algorytmy wywłaszczania zwykle nie zajmują się łączami, na których pasmo jest wystarczające, pasmo stracone sieciowe może nie oddawać w pełni jakości algorytmów w zakresie minimalizacji wywłaszczonego pasma. Z tego powodu zdefiniowano dodatkowo nadmiarowe pasmo lokalne B_{xloc}, obejmujące tylko te łącza, na których pasmo jest niewystarczające dla przyjęcia nowej ścieżki.

$$B_{xloc} = \sum_{e \in \hat{R}} B_x^{(e)} \,. \tag{3.22}$$

W praktyce przy ocenie algorytmów wywłaszczania przydatne jest zdefiniowanie wielkości względnych: pasma straconego $b_x^{(e)}$ na łączu e, pasma straconego sieciowego b_{xnet} i pasma straconego lokalnego b_{xloc}. Są to wartości odniesione do wymaganego pasma $B_o^{(e)}$ lub wymaganego pasma sieciowego B_{xnet}.

$$b_x^{(e)} = \frac{B_x^{(e)}}{B_o^{(e)}} \tag{3.23}$$

$$b_{xnet} = \frac{B_{xnet}}{B_{rnet}} \tag{3.24}$$

$$b_{xloc} = \frac{B_{xloc}}{B_{rnet}} \tag{3.25}$$

Do celów oceny wpływu algorytmu na warunki ruchowe w sieci zdefiniowano poziom rezerwacji pasma $b_b^{(e)}$ dla łącza e, jako stosunek ilości pasma zarezerwowanego dla wszystkich ścieżek przechodzących przez łącze e do pasma tego łącza.

$$b_b^{(e)} = \frac{a_{oc}^{(e)}}{c_e} \tag{3.26}$$

Analogicznie zdefiniować można globalny poziom B_b rezerwacji pasma jako stosunek sumy pasm zarezerwowanych na wszystkich łączach do sumy pasm łączy:

$$B_b = \frac{\sum_{e \in E} a_{oc}^{(e)}}{\sum_{e \in E} c_e}, \tag{3.27}$$

co w przypadku identycznych pasm łączy w sieci równych c daje następującą zależność:

$$B_b = \frac{\sum_{e \in E} a_{oc}^{(e)}}{L \cdot c}. \tag{3.28}$$

Do opisu niektórych algorytmów przydatne jest zdefiniowanie tablicy B_{hold} zawierającej sumę pasm zajętych przez wszystkich potencjalnych kandydatów, dla których priorytet utrzymania h_p jest równy danemu h:

$$B_{hold}[h] = \sum_{p \in \boldsymbol{K}} \sigma(h_p = h) b_p , \tag{3.29}$$

gdzie $h \in \{0, 1, \ldots, 7\}$.

Podobnie, można zdefiniować tablicę pasma B_{pr} możliwego (dostępnego) do wywłaszczenia, zdefiniowaną jako sumę pasm ścieżek (potencjalnych kandydatów), które mogą być wywłaszczone przez ścieżkę o danym priorytecie:

$$B_{pr}[h] = \sum_{h'=h+1}^{7} B_{hold}[h'] = \sum_{p \in \boldsymbol{K}} \sigma(h_p > h) b_p , \tag{3.30}$$

gdzie $h \in \{0, 1, \ldots, 7\}$.

Z definicji wywłaszczania wynika, że $B_{pr}[7] = 0$, tzn. pasmo dostępne do wywłaszczenia dla ścieżki o najniższym priorytecie utworzenia równym 7 musi być równe 0, gdyż ścieżka o najniższym możliwym priorytecie utworzenia nie może wywłaszczyć żadnej innej ścieżki, w tym także innej ścieżki o priorytecie utrzymania równym 7.

3.4.5. Warunki poprawności wywłaszczania

Aby dowolna ścieżka q mogła wywłaszczyć dowolną inną ścieżkę p muszą być jednocześnie spełnione następujące trzy warunki.

1. Istnieje przynajmniej jedno łącze e należące do obu ścieżek, na którym brakuje wolnego pasma dla ścieżki q, co możemy zapiać następująco:

$$a_f^{(e)} < b_q , \tag{3.31}$$

dla dowolnego $e \in (R_q \cap R_p)$.

2. Priorytet utworzenia s_q ścieżki q jest wyższy (liczbowo mniejszy) od priorytetu utrzymania h_p ścieżki p.

$$s_q < h_p \tag{3.32}$$

3. Dodatkowym warunkiem jest pozytywne zakończenie algorytmu wywłaszczania. Warunek ten oznacza, że musi istnieć przynajmniej jeden zbiór $\boldsymbol{W}$ kandydatów taki, że wywłaszczenie ścieżek należących do $\boldsymbol{W}$ umożliwi odzyskanie wolnego pasma w ilości wymaganej do przyjęcia ścieżki q. Formalnie to ujmując dla każdego łącza e, na którym brakuje pasma do przyjęcia ścieżki q, musi być spełniona następująca nierówność.

$$a_f^{(e)} + \sum_{w \in \boldsymbol{W}} b_w \beta(w,e) \geq b_q, \tag{3.33}$$

dla każdego $e \in \hat{R}$.

3.4.6. Kaskada i łańcuch wywłaszczeń

Ze zjawiskiem kaskady wywłaszczeń mamy do czynienia wtedy, gdy wywłaszczona ścieżka podczas jej ponownego utworzenia na nowej drodze powoduje kolejne wywłaszczenia innych ścieżek. Te z kolei ścieżki mogą spowodować kolejne wywłaszczenia, itd. W efekcie może w sieci dojść do niekontrolowanego, masowego wywłaszczenia wielu ścieżek. Jest to zjawisko wysoce niekorzystne, gdyż powoduje nie tylko utratę dużych ilości ruchu z powodu przeniesienia ścieżek, ale także gwałtowny wzrost ilości ruchu sygnalizacyjnego w sieci.

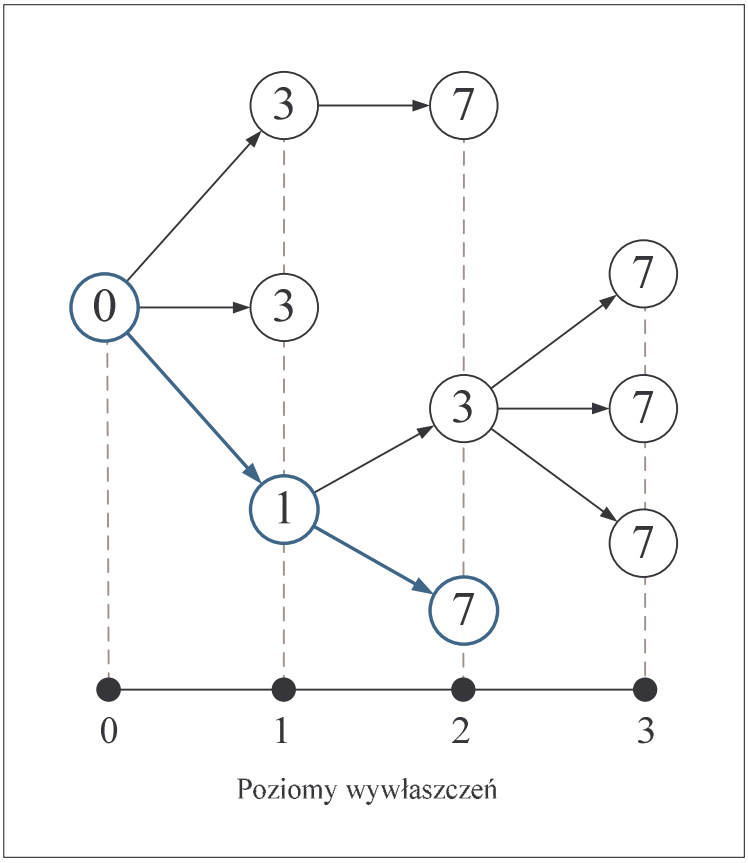
Każda kaskada rozpoczyna się w momencie, gdy tworzona jest w sieci nowa ścieżka (ścieżka inicjująca), która nie była wcześniej wywłaszczona. Jeśli utworzenie tej ścieżki spowoduje wywłaszczenie ścieżek o niższym priorytecie, wówczas wywłaszczone ścieżki mogą same spowodować wywłaszczenie innych ścieżek o jeszcze niższym priorytecie. W ten sposób mamy do czynienia z kolejnymi poziomami wywłaszczeń. Liczba poziomów będzie zawsze mniejsza od liczby priorytetów ścieżek zdefiniowanych w sieci (por. wzór 3.5), których zgodnie ze standardem [7] nie może być więcej niż osiem. Zatem maksymalna liczba poziomów może wynieść siedem, gdy utworzenie nowej ścieżki odpowiada poziomowi zerowemu.

Należy pamiętać, iż każda ścieżka może potencjalnie wywłaszczyć kilka innych ścieżek. Zatem kaskada wywłaszczeń może mieć postać drzewa, podobnego jak na rys. 3.4, a liczba ścieżek biorących udział w kaskadzie może być bardzo duża. Oczywiście o kaskadzie mówimy tylko wtedy, gdy poziom wywłaszczania osiągnie wartość dwa lub więcej.

Ze zjawiskiem kaskady wywłaszczeń wiąże się bezpośrednio pojęcie **łańcucha wywłaszczeń**, który odpowiada pojedynczej gałęzi w kaskadzie. Łańcuch wywłaszczeń jest to sekwencja wywłaszczeń spowodowanych pośrednio lub bezpośrednio przez ścieżkę inicjującą. Długość łańcucha jest o jeden mniejsza od liczby ścieżek w łańcuchu, np. jeśli nowo tworzo-

na ścieżka powoduje wywłaszczenie, a wywłaszczona ścieżka już nie powoduje kolejnych wywłaszczeń, wówczas mamy najkrótszy możliwy łańcuch o długości jeden, składający się z dwóch ścieżek. Bazując na pojęciu łańcucha, można zdefiniować długość kaskady wywłaszczeń jako długość najdłuższego łańcucha wywłaszczeń w kaskadzie. Ostatnią z istotnych definicji jest liczba ścieżek w kaskadzie wywłaszczeń, równa liczbie wszystkich ścieżek wywłaszczonych bezpośrednio lub pośrednio przez ścieżkę inicjującą.

Dla zilustrowania wprowadzonych pojęć, na rys. 3.4 przedstawiono schematycznie przykład kaskady wywłaszczeń. Ścieżka inicjująca wywłaszczenie o priorytecie utworzenia równym 0 powoduje wywłaszczenie trzech ścieżek o priorytetach 3, 3 i 1. Wywłaszczona ścieżka o priorytecie 1 przy próbie ponownego utworzenia powoduje kolejne wywłaszczenia dwóch ścieżek o priorytetach 3 i 7, z których pierwsza również wywłaszcza trzy inne ścieżki. Na tym przykładzie widać, że ścieżka inicjująca spowodowała wywłaszczenie w sumie dziewięciu ścieżek, mimo, że tylko trzy z nich zostały wywłaszczone bezpośrednio. Liczba ścieżek w kaskadzie jest tu równa dziewięć, natomiast długość kaskady wynosi trzy. Przykładowy łańcuch obejmujący ścieżki o priorytetach 0, 1 i 7 ma długość dwa.



Rys. 3.4. Zjawisko kaskady wywłaszczeń – przykład; strzałki oznaczają kolejne wywłaszczenia, a numery w węzłach drzewa – priorytety ścieżek. Kolorem niebieskim wyróżniono przykładowy łańcuch wywłaszczeń.

Aby formalnie zdefiniować pojęcia związane z kaskadą wywłaszczeń, wprowadzimy pojęcie zbioru $\boldsymbol{X}^{(p)}$ ścieżek wywłaszczonych bezpośrednio przez ścieżkę p. Zbiór $\boldsymbol{X}^{(p)}$ jest równy zbiorowi $\boldsymbol{W}$ utworzonemu w efekcie próby utworzenia ścieżki p.

Zbiór $\boldsymbol{D}^{(p)}$ ścieżek w kaskadzie wywłaszczeń spowodowanej przez ścieżkę p zawiera wszystkie ścieżki, jakie zostały bezpośrednio lub pośrednio wywłaszczone w efekcie utworzenia ścieżki p.

$$\boldsymbol{D}^{(p)} = \boldsymbol{X}^{(p)} \cup \sum_{l \in \boldsymbol{X}^{(p)}} \boldsymbol{D}^{(l)} \tag{3.34}$$

Liczba $H^{(p)}$ ścieżek w kaskadzie wywłaszczeń zapoczątkowanym przez ścieżkę p jest określona jako liczność zbioru $\boldsymbol{D}^{(p)}$.

$$H^{(p)} = \left| \boldsymbol{D}^{(p)} \right| \tag{3.35}$$

Zdefiniujmy funkcję poprzednika prev(p) ścieżki p, zwracającą bądź jednoelementowy zbiór zawierający ścieżkę q, która spowodowała wywłaszczenie ścieżki p, bądź zbiór pusty, gdy ścieżka p nie została wywłaszczona.

$$\text{prev}(p) = \begin{cases} \{q\} & \text{gdy } \exists_q (p \in \boldsymbol{X}^{(q)}) \\ \varnothing & \text{w przeciwnym wypadku} \end{cases} \tag{3.36}$$

Można teraz zdefiniować poziom $m^{(p)}$ wywłaszczenia ścieżki p jako (dotychczasową) długość łańcucha wywłaszczeń, określoną tuż po wywłaszczeniu ścieżki p.

$$m^{(p)} = \begin{cases} 0 & \text{dla prev}(p) = \varnothing \\ m^{(\text{prev}(p))} + 1 & \text{dla prev}(p) \neq \varnothing \end{cases} \tag{3.37}$$

Długość $g^{(p)}$ kaskady wywłaszczeń spowodowanej przez ścieżkę p definiujemy jako największy poziom wywłaszczenia wśród ścieżek biorących udział w kaskadzie wywłaszczeń.

$$g^{(p)} = \max_{l \in D^{(p)}} (m^{(l)}) \tag{3.38}$$

3.5. Dostępne algorytmy

Wywłaszczanie jako metoda udostępniania pasma w sytuacji braku wolnych zasobów jest znane w sieciach telekomunikacyjnych od dawna. Dopóki jednak nie zaczęto implementować sieci pakietowych z kanałami o różnej przepustowości, dopóty nie było potrzeby prowadzenia bardziej zaawansowanych badań w tym kierunku. Dopiero na początku lat dziewięćdziesiątych pojawiły się pierwsze opracowania szerzej omawiające ten temat.

W tej części pracy omówione zostaną znane algorytmy heurystyczne, które kosztem nieoptymalnego wyboru oferują akceptowalną złożoność obliczeniową. Dla uzupełnienia zostaną przedstawione i omówione algorytmy optymalne, które wprawdzie z powodu nadmiernej złożoności obliczeniowej nie mogą być praktycznie zastosowane, ale pozwalają uzyskać pełniejszy obraz możliwych rozwiązań.

Omówione algorytmy heurystyczne zaczerpnięto z najczęściej cytowanych publikacji dotyczących wywłaszczania. Najpierw krótko je scharakteryzowano, aby następnie opisać je dokładniej. Przyjęto konwencję nazewniczą w postaci skrótów nazwisk autorów.

1. **GarGop** (Garay-Gopal, 1992) [35]. Jest to pierwsze ogólnie dostępne opracowanie dotyczące wywłaszczania połączeń w sieciach pakietowych. Zawarto w nim dwa niezależne algorytmy globalne przeznaczone do minimalizacji liczby wywłaszczonych ścieżek i do minimalizacji wywłaszczonego pasma, bez uwzględnienia priorytetów ścieżek. Algorytmy zostały opracowane dla sieci ATM, ale mogą być zaadoptowane do wykorzystania w sieciach MPLS.
2. **Pey** (Peyravian, 1994) [82]. W tym artykule przedstawiono pojedynczy, stosunkowo prosty algorytm lokalny, przeznaczony do minimalizacji zarówno liczby wywłaszczeń jak i wywłaszczonego pasma.
3. **OliSco** (de Oliveira-Scoglio, 2002) [72]. W artykule zawarto jeden z pierwszych algorytmów przeznaczonych z założenia dla sieci MPLS. Do tej publikacji odnoszą się autorzy większości późniejszych prac dotyczących wywłaszczania. Jest to prosta, uniwersalna i szybka metoda lokalna oparta na sortowaniu ścieżek-kandydatów. Algorytm ten został później upowszechniony jako dokument informacyjny RFC 4829 [70].
4. **BlaMeL** (Blanchy-Mélon-Leduc, 2003) [20]. W artykule opisano prosty algorytm lokalny, w którym nadrzędnym kryterium wyboru jest priorytet kandydatów. W dalszej kolejności minimalizuje się liczbę ścieżek a w ostatniej kolejności stracone pasmo.

W kilku kolejnych podrozdziałach dokładniej zaprezentowano i poddano analizie wymienione algorytmy. Kod programów je realizujących, zaczerpnięty z oryginalnych artykułów, został ujednolicony pod względem użytych oznaczeń i zapisany w języku pseudo-pascal, w którym niektóre instrukcje i procedury zostały dla zachowania przejrzystości zapisane w języku naturalnym. Dodatkowym uproszczeniem w stosunku do języka programowania Pascal jest zastosowanie notacji matematycznej dla zapisania operacji na zbiorach i sekwencjach. Dla poprawy jakości opisów zachowano oznaczenia wprowadzone w rozdziale 3.4 a linie kodu zostały ponumerowane.

3.5.1. Algorytm GarGop (Garay-Gopal)

Pierwsze szeroko dostępne analityczne ujęcie zagadnienia wywłaszczania zawarto w [35]. Artykuł ten został wprawdzie napisany pod kątem zastosowania w sieciach ATM, ale

dzięki podobieństwu obu technik wiele treści pozostaje aktualnych również dla sieci IP/MPLS.

Autorzy zaproponowali proste algorytmy heurystyczne o zasięgu globalnym, oddzielnie dla przypadku jednakowych i różnych pasm ścieżek. Dla potrzeb wywłaszczania w sieciach MPLS interesujący wydaje się jedynie przypadek nierównych pasm ścieżek i tylko ten będzie tu rozważany. Występuje on w dwóch wariantach. Pierwszy – GarGop/RC (ang. *relocation count*) – minimalizuje liczbę wywłaszczeń, natomiast drugi – GarGop/BW (ang. *bandwidth*) – minimalizuje pasmo wywłaszczonych ścieżek.

Algorytm minimalizacji liczby ścieżek został przedstawiony na rys. 3.5. Główna procedura *algorithm_GarGop* działa następująco. Po inicjalizacji (linie 2-4) uruchamia się główną pętlę (linie 5-11), która jest powtarzana tak długo, aż brakujące pasmo B_{rnet} osiągnie wartość zero. W linii 6 następuje sprawdzenie, czy w zbiorze $\boldsymbol{K}$ potencjalnych kandydatów znajduje się przynajmniej jedna ścieżka. Jeśli nie, wówczas algorytm jest przerywany z powodu braku wystarczającej liczby potencjalnych kandydatów.

```
1   procedure algorithm_GarGop ()
2     W := ∅                          # wyzerowanie zbioru kandydatów
3     K := cand_preselect_on_route ()  # wyznaczenie zbioru
                                       # potencjalnych kandydatów
4     calculate_bw_balance ()          # wyliczenie początkowego
                                       # bilansu pasma
5     repeat
6       if K = ∅ then return ∅         # zakończ, jeśli nie ma więcej
                                       # potencjalnych kandydatów
7       p := best_path (K)             # wybierz kandydata z listy,
8       W := W ∪ {p}                   # dodaj go do listy
9       K := K \ {p}                   # i usuń z listy wyboru
10      update_bw_balance (p)          # sprawdź ile pasma jeszcze brakuje
11    until B_rnet = 0                 # zakończ gdy brakujące pasmo jest zerowe
12    return W                         # zwróć listę kandydatów
13  end
14
15  procedure cand_preselect_on_route ()
16    K := ∅
17    for each p in P do               # dla każdej ścieżki
18      if h_p > s_q                   # która może zostać wywłaszczona
19      and path_on_route (R_p, p) then
```

```
20          K := K ∪ {p}                    # dodaj ścieżkę do listy
                                            # potencjalnych kandydatów
21  end
22
23  procedure calculate_bw_balance ()
24      B_rnet := 0                         # zainicjowanie brakujące pasma sieciowego
25      for each e in R̂ do                  # dla każdego łącza na wybranej drodze
26          S_e := a_f^(e) – b_q            # oblicz bilans pasma na łączu
27          B_r^(e) := max (0, –S_e)        # uaktualnij brakujące pasmo na łączu e
28          B_rnet := min (B_rnet, B_r^(e)) # uaktualnij brakujące pasmo sieciowe
29      end for
30  end
31
32  function best_path (K)
33      max_sum := 0                        # wyzeruj wskaźnik najlepszej ścieżki
34      for each p in K                     # dla każdego potencjalnego kandydata
35          sum := 0                        # wyzeruj sumę dla ścieżki
36          for each e in R̂
37              sum := sum + min [B_r^(e), β(p,e)b_p]
                    # na łączach gdzie występuje ścieżka zsumuj pasmo ścieżki
                    # ograniczone do wymaganego pasma
38          end for
39          if sum > max_sum then       # jeśli aktualna suma jest największa
40              max_sum := sum          # uaktualnij wskaźnik najlepszej ścieżki
41              best_p := p             # zapamiętaj najlepszą ścieżkę
42          end if
43      end for
44      return best_p                   # zwróć zapamiętaną najlepszą ścieżkę
45  end
46
47  procedure update_bw_balance (p)
48      for each e in R̂
49          S_e := S_e + β(p,e)b_p      # uaktualnij bilans pasma i brakujące
50          B_r^(e) := max (0, –S_e)    # pasmo o pasmo wybranej ścieżki
51      end for
52  end
```

Rys. 3.5. Algorytm GarGop/RC dla minimalizacji liczby wywłaszczonych ścieżek.

Najważniejszymi krokami algorytmu są przypisanie zmiennej *p* najlepszej ścieżki (linia 7), dodanie jej do listy kandydatów (linia 8) i usunięcie z listy potencjalnych kandydatów (linia 9). Po tym następuje uaktualnienie bilansu pasma (linia 10), tzn. pasmo wybranej ścieżki zostaje dodane do bilansu pasma S_e na każdym łączu *e*, przez które przechodzi ścieżka, i na którym brak jest wolnego pasma. Gdy wartość brakującego pasma spadnie do zera (linia 11) następuje wyjście z pętli i na zakończenie zbiór ***W*** kandydatów zostaje zwrócony jako wynik działania algorytmu.

O wyborze ścieżek do wywłaszczania decyduje kluczowa dla algorytmu funkcja *best_path*. W omawianym algorytmie za najlepszego kandydata do wywłaszczania uznaje się ścieżkę, dla której wartość metryki $\alpha(p)$ (wzór 3.39) jest największa. Aby ją obliczyć, na każdym łączu wymagającym wywłaszczania, na którym ścieżka *p* jest obecna, wybiera się mniejszą z dwóch wartości: brakujące pasmo $B_r^{(e)}$ i pasmo ścieżki b_p, i sumuje po tych łączach.

$$\alpha(p) = \sum_{e \in \hat{R}} \min(B_r^{(e)}, b_p) \tag{3.39}$$

Procedura *best_path* rozpoczyna się od wyzerowania zmiennej *max_sum* (linia 33), po czym dla każdej ścieżki z listy potencjalnych kandydatów ***K*** oblicza się (linie 35-38) metrykę $\alpha(p)$ przechowywaną w zmiennej *sum*. Obliczona wartość jest następnie porównywana z dotychczasową największą wartością *max_sum*. Jeśli nowa metryka jest większa, wówczas następuje uaktualnienie *max_sum* oraz przypisanie aktualnej ścieżki *p* do zmiennej *best_p* (linie 40-41). Po przejrzeniu wszystkich potencjalnych kandydatów zwrócona zostaje ścieżka o najwyższej wartości metryki $\alpha(p)$ (linia 44).

Druga postać algorytmu GarGop, utworzona dla przypadku minimalizacji ilości wywłaszczonego pasma, różni się od poprzedniej jedynie inną definicją procedury *best_path*. Tym razem minimalizacji podlega metryka $\chi(p)$ (wzór 3.40). Odpowiednio zmodyfikowana procedura została pokazana na rys. 3.6.

$$\chi(p) = \frac{\sum_{e \in \hat{R}} \max(b_p - B_r^{(e)}, 0)}{\sum_{e \in \hat{R}} \min(B_r^{(e)}, b_p)} \tag{3.40}$$

Z racji przeznaczenia dla sieci ATM, oryginalna metoda nie uwzględnia priorytetów ścieżek. Nie jest to problemem, gdyż z racji użycia zbioru ***K*** ograniczono wybór kandydatów do tych ścieżek, które mają priorytet utrzymania niższy niż priorytet utworzenia nowej ścieżki *q*. Tak zostało to z resztą określone w ogólnej definicji zbioru ***K*** (wzór 3.8).

Algorytm GarGop jest algorytmem zachłannym, który raz wybranej ścieżki nie usuwa już z listy kandydatów. Jest to cecha, która umożliwia uruchamianie procedury usuwania ścieżek już w trakcie działania algorytmu, jeśli tylko się zagwarantuje, że wybór zakończy się sukcesem. Może to mieć znaczenie w praktycznych realizacjach, jeśli czas potrzebny do zakończe-

nia działania algorytmu jest nie do pominięcia, np. z powodu długiego czasu dostępu do danych o ścieżkach.

```
1   function best_path (K)
2     min_div := ∞                  # inicjalizuj wskaźnik najlepszej ścieżki
3     for each p in K               # dla każdego potencjalnego kandydata
4       sum1 := sum2 := 0           # wyzeruj wskaźniki sum
5       for each e in R̂             # na każdym łączu ścieżki
                                    # zsumuj nadwyżkę pasma (sum1)
                                    # oraz pasmo wykorzystane (sum2)
6         sum1 := sum1 + max (b_p – B_r^(e), 0)
7         sum2 := sum2 + min (B_r^(e), b_p)
8       end for
9       div := sum1 / sum2          # oblicz metrykę ścieżki
10      if div < min_div then       # jeśli aktualna metryka jest najmniejsza
11        min_div := div            # uaktualnij wskaźnik najlepszej ścieżki
12        best_p := p               # zapamiętaj aktualną ścieżkę
13      end if
14    end for
15    return best_p                 # zwróć ścieżkę o najmniejszej metryce div
16  end
```

Rys. 3.6. Algorytm GarGop/BW dla minimalizacji ilości wywłaszczonego pasma.

Dokładniejsza analiza wariantu GarGop/RC pozwala zauważyć, że algorytm nie bierze pod uwagę pasma ścieżek w sytuacji, gdy to pasmo jest większe niż pasmo wymagane. Nie są przez to rozróżniane ścieżki o dużym i małym paśmie, co ma negatywny wpływ na wywłaszczone pasmo. Może się zdarzyć, że potrzebne jest zwolnienie małej ilości pasma, a algorytm wybierze wtedy dowolną, być może największą ścieżkę. Takie działanie jest uzasadnione tym, że wariant ten jest nastawiony na minimalizację liczby wywłaszczeń. Algorytm wykonuje więc to, do czego został przeznaczony ale nic więcej. Nie jest tu możliwe nałożenie dodatkowych wymagań.

Odnośnie wariantu GarGop/BW daje się zauważyć, że również tutaj zdolność do wyróżniania ścieżek jest ograniczona. W przypadku, gdy na każdym z łączy pasmo ścieżki jest mniejsze niż pasmo wymagane, metryka $\chi(p)$ ma wartość zero, która jest wartością minimalną. Użyte kryterium wyboru powoduje, że wybierana jest jedna z takich ścieżek bez względu na pasmo. To ma z kolei przełożenie na pewną nieokreśloność tej metody i zależność wyniku od kolejności analizy ścieżek.

Niewątpliwą zaletą metody GarGop w obu wariantach jest prostota i niewielka złożoność

obliczeniowa. Przekłada się to na stosunkowo niewielkie wymagania na moc procesora, a to ma znaczenie w przypadku sieci o dużej intensywności żądań utworzenia ścieżek.

3.5.2. Algorytm Pey (Peyravian)

Algorytm Pey został opracowany zanim pojawiły się sieci MPLS, ale podobnie jak poprzednio omawiany GarGop może być odpowiednio zaadoptowany. W przeciwieństwie do GarGop jest to algorytm o zasięgu lokalnym. Autor proponuje pojedynczy algorytm bez możliwości wyboru kryterium. Jest on w zamyśle zaproponowany tak, aby minimalizować zarówno liczbę wywłaszczeń jak i wywłaszczone pasmo, czyli dwa czynniki zwykle uznawane za najważniejsze.

Odpowiednia procedura została przedstawiona na rys. 3.7. Jest to algorytm lokalny, dlatego zadaniem głównej pętli *for* (linie 3-17) jest powtórzenie procedury wyboru zbioru kandydatów na każdym łączu, na którym brakuje wolnego pasma. W pierwszym kroku w pętli następuje obliczenie bilansu pasma na bieżącym łączu (linie 4-5), wybór potencjalnych kandydatów (linia 6), a następnie uruchomienie pętli *while* odpowiedzialnej za wybór kompletu kandydatów na danym łączu. Pętla jest powtarzana tak długo, dopóki brakuje pasma.

```
1   procedure algorithm_Pey ()
2     W := ∅                          # wyzerowanie zbioru kandydatów
3     for each e in R̂                 # dla każdego łącza na drodze ścieżki
4       S_e := a_f^(e) – b_q          # oblicz bilans pasma
5       B_r^(e) := max (0, –S_e)      # oblicz wymagane pasmo
6       K_e := cand_preselect_on_link (e)   # wyznaczenie lokalnego zbioru
                                            # potencjalnych kandydatów
7       while B_r^(e) ≠ 0 do          # powtarzaj, dopóki brakuje pasma
8         if K_e = ∅ then return ∅    # wyjdź, jeśli brakuje kandydatów
9         p := best_path (K_e)        # wybierz najlepszą ścieżkę
10        W := W ∪ {p}                # dodaj ścieżkę do listy kandydatów
11        K_e := K_e \ {p}            # usuń z listy potencjalnych kandydatów
12        S_e := S_e + b_p            # uaktualnij bilans pasma
13        B_r^(e) := max (0, –S_e)    # oraz wymagane pasmo
14      end while
15    end for
16    return W
17  end
18
19  function best_path (K_e)
20    best_bw := ∞                    # zainicjuj metrykę najlepszej ścieżki
```

```
21      best_p := ∅                     # wyzeruj wskaźnik najlepszej ścieżki
22      for each p in K_e               # dla każdego potencjalnego kandydata
23          if b_p >= B_r^(e)           # jeśli pasmo ścieżki nie jest mniejsze
                                        # od pasma wymaganego
24          and b_p < best_bw then      # i jest mniejsze od dotychczas najlepszego
25              best_bw:= b_p           # zapamiętaj pasmo ścieżki
26              best_p := p             # i zachowaj wskaźnik do ścieżki
27          end if
28      end for
29      if best_p ≠ ∅ then              # jeśli znaleziono pojedynczą ścieżkę
30          return best_p               # to ją zwróć
31      best_bw := 0                    # wyzeruj najlepsze pasmo ścieżki
32      for each p in K_e               # dla każdego potencjalnego kandydata
33          if b_p > best_bw then       # jeśli pasmo ścieżki jest większe niż najlepsze
34              best_bw:= b_p           # zapamiętaj pasmo ścieżki
35              best_p := p             # i zachowaj wskaźnik do ścieżki
36          end if
37      end for
38      return best_p                   # zwróć najlepszą ścieżkę
39  end
40
41  procedure cand_preselect_on_link (e)
42      K_e := ∅
43      for each p in P do                  # dla każdej ścieżki
44          if h_p > s                      # która może zostać wywłaszczona
45          and path_on_link (e, p) then    # i leży na łączu e
46              K_e := K_e ∪ {p}            # dodaj ścieżkę do lokalnej listy
                                            # potencjalnych kandydatów
47  end
```

Rys. 3.7. Algorytm Pey.

Po wejściu do pętli *while* następuje sprawdzenie, czy w zbiorze potencjalnych kandydatów $\boldsymbol{K_e}$ znajdują się ścieżki (linia 8), po czym wywołana zostaje funkcja *best_path*, aby wybrać najlepszą ścieżkę z listy $\boldsymbol{K_e}$ (linia 9). Po tym następuje aktualizacja listy ścieżek do wywłaszczenia, listy potencjalnych kandydatów oraz bilansu pasma (linie 10-13). Struktura pętli *while* jest podobna jak w przypadku algorytmu globalnego, jednak w tym przypadku lista potencjalnych kandydatów ogranicza się do ścieżek dostępnych na aktualnym łączu, podczas gdy w algorytmach globalnych bierze się pod uwagę ścieżki na wszystkich łączach

należących do $\hat{R}$. Po zagwarantowaniu wystarczającego pasma następuje opuszczenie pętli *while* i w razie potrzeby powtórzenie wyboru dla kolejnego łącza w pętli *for*. Na zakończenie następuje zwrócenie zbioru $\boldsymbol{W}$ zawierającego ścieżki wybrane na wszystkich łączach (linia 16).

Podobnie jak dla wielu innych metod, cechą wyróżniającą ten algorytm jest sposób wyboru najlepszej ścieżki w funkcji *best_path* (linie 19-39). W celu wyboru ścieżki o paśmie jak najbardziej zbliżonym do wymaganego wybór odbywa się w dwóch etapach. W pierwszym (linie 22-28) wybrana zostaje pojedyncza ścieżka o najmniejszym paśmie wystarczającym dla zaspokojenia brakującego pasma. Jeśli żadna taka ścieżka nie zostanie znaleziona, wówczas następuje drugi etap polegający na wybraniu ścieżki o największym paśmie (linie 31-37).

Omówiona postać procedury *best_path* jest zgodna z oryginalnym opracowaniem [82]. Jak łatwo zauważyć, można poprawić szybkość działania tego algorytmu poprzez użycie pojedynczej pętli z bardziej złożonym warunkiem. To może mieć znaczenie wtedy, gdy w danej implementacji przeglądanie ścieżek zajmuje stosunkowo dużo czasu. Odpowiednio zmodyfikowana procedura *best_path_mod* została przedstawiona na rys. 3.8.

```
1   function best_path_mod (K_e)
2     best_bw := 0
3     best_p := ∅
4     for each p in K_e
5       if (b_p >= B_r^(e) and (best_bw < B_r^(e) or b_p < best_bw) )
6       or (b_p < B_r^(e) and b_p > best_bw) then
7         best_bw:= b_p
8         best_p := p
9       end if
10    end for
11    return best_p
12  end
```

Rys. 3.8. Modyfikacja algorytmu Pey z pojedynczą pętlą wyboru najlepszego kandydata.

Należy zwrócić uwagę, iż ogólna procedura realizująca algorytm Pey z rys. 3.7 w liniach 1-17 została przedstawiona w taki sposób, aby dać lepsze porównanie z algorytmem globalnym GarGop. W praktyce jednak najbardziej prawdopodobne jest zaimplementowanie algorytmu lokalnego jako metody zdecentralizowanej. Wówczas pętla *for* odpowiadająca za obsługę kolejnych łączy (linie 3-15) nie będzie istniała w sposób jawny. Jej funkcję spełni uruchomienie lokalnych instancji algorytmu w kolejnych ruterach na drodze połączeniowej, przy czym musi być ono wykonane jedno po drugim. Jest to ważne, aby algorytm w danym ruterze bazował na wywłaszczeniach przeprowadzonych na poprzednich łączach. Może się

bowiem zdarzyć, że jedna ze ścieżek wywłaszczonych na wcześniejszym łączu zwolniła przy okazji wystarczającą ilość pasma na aktualnym łączu, a wtedy rozpoczynanie kolejnego wywłaszczania jest już niepotrzebne.

Zaletą algorytmu Pey jest jego prostota. Jego działanie polega na minimalizacji liczby wywłaszczonych ścieżek przy jednoczesnym ograniczaniu wywłaszczonego pasma. Określając kolejnych kandydatów do wywłaszczenia dąży on przy tym do wyboru ścieżek o paśmie najbardziej zbliżonym do wymaganego, kosztem jak najmniejszej liczby ścieżek. W ten prosty sposób algorytm osiąga bardzo dobrą wydajność pod warunkiem, że wymagane kryterium to liczba wywłaszczeń. Dokładniejsza analiza prowadzi do wniosku, że algorytm Pey jest w istocie optymalny w ramach pojedynczego łącza w zakresie minimalizacji liczby wywłaszczeń.

Algorytm Pey nie posiada wariantów, zatem nie wymaga też konfiguracji. To może być jego zaletą, ale też jest oczywistym ograniczeniem, bo nie umożliwia dostosowanie go do indywidualnych wymagań.

3.5.3. Algorytm OliSco (de Oliveira, Scoglio)

Jedną z pierwszych publikacji dotyczących wywłaszczania dla sieci MPLS jest [72]. W artykule tym opisano prosty algorytm o zasięgu lokalnym oparty na sortowaniu ścieżek. Metoda polega na obliczaniu dla każdej ze ścieżek odpowiednio zdefiniowanej metryki liczbowej. Następnie tworzona jest lista dostępnych ścieżek posortowana zgodnie z rosnącymi wartościami ich metryk. Wybór kandydatów do wywłaszczenia polega na pobieraniu z czoła listy ścieżek tak długo, aż zebrane zostanie brakujące pasmo. Ponieważ jest to algorytm lokalny, zatem w razie potrzeby procedura jest powtarzana na kolejnym łączu na drodze połączeniowej w tych ruterach, gdzie wciąż brakuje pasma dla nowej ścieżki.

Algorytm jest oparty na uniwersalnej metryce $m(p,e)$, rozumianej jako koszt wywłaszczenia danej ścieżki p na łączu e. Jest ona zdefiniowana w oparciu o cztery parametry X_1, X_2, X_3 i X_4 o wartościach zależnych od zadanego kryterium wyboru kandydatów:

$$m(p,e) = X_1 \cdot \left(8 - h_p\right) + X_2 \cdot \frac{1}{b_p} + X_3 \cdot \left(b_p - B_o^{(e)}\right)^2 + X_4 \cdot b_p , \qquad (3.41)$$

gdzie X_1, X_2, X_3 i X_4 są nieujemnymi współczynnikami służącymi do wyboru kryterium. Jeśli któryś ze współczynników ma wartość zero, wówczas dany parametr nie jest w ogóle brany pod uwagę przy wyborze kandydatów do wywłaszczenia. Rola poszczególnych współczynników jest następująca:

- X_1 odpowiada za minimalizację priorytetu ścieżki,
- X_2 odpowiada za minimalizację liczby wywłaszczeń,
- X_3 odpowiada za minimalizację wywłaszczonego pasma,
- X_4 odpowiada za minimalizację wywłaszczeń dużych ścieżek.

Na rys. 3.9 przedstawiono implementację algorytmu w pseudokodzie. Po wyzerowaniu zbioru kandydatów (linia 2) następuje zainicjowanie listy L, przechowującej wpisy (ścieżka p, metryka m, pasmo b). Lista jest sortowana według rosnącej wartości metryki m, a w razie istnienia dwóch takich samych wartości metryk według rosnącego pasma. Dalej w fazie inicjalizacji następuje obliczenie początkowego bilansu pasma i wymaganego pasma (linie 4-5), oraz wyznaczenie zbioru potencjalnych kandydatów na danym łączu. Następnie dla każdej ścieżki z listy $\boldsymbol{K_e}$ następuje obliczenie metryk (linia 8) i dodanie ścieżki do listy L (linia 9).

```
1   procedure algorithm_OliSco ()
    # algorytm lokalny, dla uproszczenia podano procedurę dla
    # pojedynczego węzła, powtarzaną w razie potrzeby w następnych węzłach
2     W := ∅                          # inicjuj listę kandydatów
3     L := ∅                          # inicjuj listę potencjalnych kandydatów
                                      # sortowaną po metryce m,
                                      # w razie identycznych m po b_p
4     S_e := a_f^(e) – b_q            # oblicz początkowy bilans pasma
5     B_r^(e) := max (0, –S_e)        # oraz brakujące pasmo
6     K_e := cand_preselect_on_link (e)     # wyznaczenie lokalnego zbioru
                                            # potencjalnych kandydatów
7     for each p in K_e               # dla każdego potencjalnego kandydata
8       m := X_1*(8 – h_p) + X_2*(1/b_p) + X_3*sqr (b_p – B_o^(e)) + X_4*b_p
                                      # wylicz metrykę ścieżki
9       L ← (p, m, b_p)               # i dodaj do listy sortowanej
10    end for
11    B_orig := B_r^(e)               # zapamiętaj brakujące pasmo
12    while B_r^(e) ≠ 0 do            # dopóki brakuje pasma
13      if L = ∅ then return ∅        # wyjdź jeśli brak potencjalnych kandydatów
14      m := L[0].m                   # pobierz metrykę ścieżki z czoła listy
15      if |L|=1 or m < L[1].m then       # jeśli pojedyncza najmniejsza metryka
16        p := L[0].p                     # to zapamiętaj ścieżkę z czoła listy
17      else begin                    # w przypadku równych metryk
18        L' := ∅                     # utwórz zbiór ścieżek o jednakowej metryce
19        for each p in L such that L[].m = m do
                                      # zachowaj ścieżki o jednakowej metryce
20          L' := L' ∪ {p}            # i sortuj według pasma rosnąco
21        for each p in L' do         # dla każdej ścieżki o jednakowej metryce
```

```
22              if b_p > B_orig then      # jeśli pasmo ścieżki jest większe niż
                                          # początkowe brakujące pasmo
23                  return {p}            # wywłaszcz tylko p
24          for each p in L' do           # dla każdej ścieżki o jednakowej metryce
25              if b_p > B_r^(e) then     # jeśli pasmo ścieżki jest większe niż
                                          # aktualne brakujące pasmo
26                  break                 # dodaj p do W
27          for each p in reversed (L') do
                                          # przeglądaj w odwrotnej kolejności
28              W := W ∪ {p}              # dodaj do listy kandydatów
29              L:= L \ {p}               # usuń z listy sortowanej
30              S_e := S_e + b_p          # uaktualnij bilans pasma
31              if S_e > 0 then return W  # jeśli bilans pozytywny to zakończ
32          end for
33          goto eval_balance             # wyskocz z pętli do obliczenia bilansu pasma
34      end if
35      W := W ∪ {p}                      # dodaj do listy kandydatów
36      L:= L \ {p}                       # usuń z listy sortowanej
37      S_e := S_e + b_p                  # uaktualnij bilans pasma
38  eval_balance:
39      B_r^(e) := max (0, –S_e)          # uaktualnij brakujące pasmo
40  end while
41  return W                              # zwróć listę kandydatów
42 end
```

Rys. 3.9. Algorytm OliSco.

Główna pętla *while* (linie 12-40) jest wykonywana tak długo, aż pasmo zajęte przez kandydatów będzie wystarczające do przyjęcia nowej ścieżki. W ramach pojedynczego przebiegu pętli najpierw następuje sprawdzenie (linia 13), czy w zbiorze L znajdują się jeszcze ścieżki do wyboru, co jest warunkiem znalezienia wystarczającego pasma. Następnie zostaje pobrana pierwsza ścieżka z listy L i dodana do listy $\boldsymbol{W}$ kandydatów, o ile jej metryka jest unikalna (linie 15-16). Jeśli natomiast na czele listy znajdują się przynajmniej dwie ścieżki o identycznych metrykach, wówczas następuje dodanie wszystkich ścieżek o tej samej metryce do oddzielnej listy L' (linie 19-20) i analiza całego zbioru L'. Jeśli znaleziona zostanie ścieżka o paśmie większym od pasma wymaganego na samym początku (zmienna B_{orig}), wówczas tylko ta jedna ścieżka jest zwracana jako zbiór kandydatów (linie 21-23). W dalszej kolejności (przy jednakowych metrykach) poszukiwana jest w zbiorze L' ścieżka o paśmie większym do aktualnego pasma wymaganego. Jeśli taka zostanie znaleziona (linie 24-26), wówczas jest

dodawana do listy kandydatów, a jeśli nie, to ścieżki z listy *L'* dodawane są do kandydatów w kolejności od największego pasma (linie 27-32), za każdym razem sprawdzając, czy nie zebrano już wymaganego pasma (warunek w linii 31). Po każdym dodaniu ścieżki do zbioru $\boldsymbol{W}$ kandydatów, następuje usunięcie go z listy *L* i uaktualniane wartości bilansu pasma. Po określeniu kompletnej listy $\boldsymbol{W}$ kandydatów algorytm zostaje zakończony.

Rozwinięciem tego algorytmu jest metoda adaptacji pasma [72], w której zanim rozpocznie się procedura wywłaszczania, następuje sprawdzenie, czy jest możliwość zredukowania pasma istniejących ścieżek, korzystając np. z możliwości elastycznego przydziału pasma dla klas AF mechanizmu DiffServ [19]. W takim przypadku zmniejszenie zajętego pasma odbywa się proporcjonalnie dla wszystkich ścieżek. Jest to podejście bardziej „łagodne" od wywłaszczania, a jego zaletą – zmniejszenie liczby wywłaszczeń w sieci.

W oparciu o algorytm OliSco opracowano również metodę globalną wywłaszczania o nazwie PREPATH [71]. Celem algorytmu jest odszukanie ścieżek, których droga najbardziej pokrywa się z drogą nowej ścieżki. Algorytm przyjmuje stałe kryterium wyboru, którym jest w pierwszej kolejności priorytet ścieżki, następnie liczba wywłaszczeń, a w ostatniej kolejności wywłaszczone pasmo. Przedstawiona implementacja jest jednak kłopotliwa w realizacji i cechuje ją duża złożoność obliczeniowa. Należy dodać, że algorytm PREPATH przestał być rozwijany przez autorów [74].

Algorytm OliSco jest bodaj najbardziej znanym algorytmem z racji wielu odniesień w publikacjach naukowych. Jego największą zaletą jest możliwość dostosowania do indywidualnych wymagań obejmujących pasmo ścieżek, liczbę wywłaszczeń oraz priorytety wybranych ścieżek. Algorytm ten został również opublikowany w formie dokumentu IETF jako RFC 4829 [70]. Należy jednak zaznaczyć, że posiadając status „informational" nie jest i nie pretenduje do roli żadnego standardu IETF, a ma znaczenie jedynie informacyjne [23].

Złożoność obliczeniowa algorytmu OliSco nie jest duża, choć jego implementacja może być nieco trudniejsza od GarGop i Pey. Wynika to z dodatkowych procedur wykonywanych w przypadku napotkania ścieżek o tych samych metrykach, włączając w to konieczność budowania i analizy dodatkowej listy *L'*. Oczywiście możliwe jest zaimplementowanie algorytmu w prostszej postaci, z pominięciem dodatkowych sprawdzeń dla przypadku jednakowych metryk, co jednak spowoduje pogorszenie osiąganych rezultatów.

3.5.4. Algorytm BlaMeL (Blanchy, Mélon, Leduc)

W artykule [20] zaproponowano prosty mechanizm wywłaszczania o zasięgu lokalnym. Opisana metoda zakłada następujące kryterium wyboru kandydatów do wywłaszczenia. Po pierwsze, spośród dostępnych ścieżek wybiera się te o najniższym priorytecie. W drugiej kolejności o wyborze decyduje mniejsza liczba wywłaszczeń, tzn. preferowany jest zbiór zawierający mniejszą liczbę kandydatów, a w ostatniej kolejności decyduje niższa wartość sumy wywłaszczonego pasma.

Implementację algorytmu BlaMeL pokazano na rys. 3.10. Po zainicjowaniu zbioru kandydatów $\boldsymbol{W}$ i obliczeniu początkowego bilansu pasma (linie 3-4) następuje zainicjowanie (linie 5-8) i obliczenie (linie 10-13) tablicy B_{hold} zawierającej sumy pasm kandydatów z podziałem na ich priorytety utrzymania oraz związane z nią listy $\boldsymbol{K_p}$ ścieżek o określonym priorytecie. Zatem zamiast pojedynczej listy kandydatów $\boldsymbol{K_e}$, wyznaczonej w linii 9, operuje się tu zestawem ośmiu list potencjalnych kandydatów. W linii 14 wprowadzono zmienną r określającą aktualnie analizowany priorytet. Ścieżki wybierane są w kolejności malejącego priorytetu, zatem jego początkowa wartość wynosi 7.

```
1   procedure algorithm_BlaMeL ()
    # algorytm lokalny, dla uproszczenia podano procedurę dla
    # pojedynczego węzła, powtarzaną w razie potrzeby w następnych węzłach
2     W := ∅                          # inicjuj listę kandydatów
3     S_e := a_f^(e) – b_q            # oblicz początkowy bilans pasma
4     B_r^(e) := max (0, –S_e)        # oraz brakujące pasmo
5     for h in [0..7] do              # dla każdego priorytetu utrzymania
6       B_hold[h] := 0                # wyzeruj pasmo dostępne dla wywłaszczenia
7       K_p[h] := ∅                   # inicjuj zbiór potencjalnych kandydatów
                                      # pogrupowany według priorytetów
8     end
9     K_e := cand_preselect_on_link (e)  # wyznaczenie lokalnego zbioru
                                         # potencjalnych kandydatów
10    for each p in K_e do              # dla każdego potencjalnego kandydata
11      B_hold[h_p] := B_hold[h_p] + b_p  # uaktualnij pasmo ścieżek
12      K_p[h_p] := K_p[h_p] ∪ {p}        # dodaj do listy ścieżek,
                                          # posortowanej według pasma malejąco
13    end
14    r := 7                          # rozpocznij od najniższego priorytetu
15    while B_r^(e) ≠ 0 do            # dopóki brakuje pasma
16      if r ≤ s_q then               # jeśli za mało dostępnego pasma
                                      # w ścieżkach o niższym priorytecie
17        return ∅                    # wtedy wywłaszczanie jest nieudane
18      if B_r^(e) ≥ B_hold[r] then   # jeśli suma pasm dla danego priorytetu r
                                      # nie przekracza brakującego pasma
19        W := W ∪ K_p[r]             # dodaj wszystkie ścieżki o priorytecie r
20        S_e := S_e + B_hold[r]      # uaktualnij bilans pasma
21        r := r – 1                  # przejdź do niższego priorytetu
```

```
22          end
23          else                          # nie wszystkie ścieżki o priorytecie r
                                          # mają być dodane do listy kandydatów
24              while B_r^(e) ≠ 0 do      # dopóki brakuje pasma
25                  i := 0                # wyzeruj licznik ścieżek
26                  while i < |K_p[r]| –1 and K_p[r][i+1] > B_r^(e) do
                                          # znajdź najmniejszą ścieżkę o paśmie
                                          # większym od wymaganego (jeśli istnieje)
                                          # lub pierwszą z listy (w przeciwnym wypadku)
27                      i := i + 1
28                  p := K_p[r][i]        # zapamiętaj wybraną ścieżkę
29                  K_p[r] := K_p[r] \ {p}  # usuń ścieżkę z listy
30                  W := W ∪ {p}          # dodaj ścieżkę do kandydatów
31                  S_e := S_e + b_p      # uaktualnij bilans pasma
32              end while
33          end if
34          B_r^(e) := max (0, –S_e)      # oblicz brakujące pasmo
35      end while
36      return W                          # zwróć zbiór kandydatów
37  end
```

Rys. 3.10. Algorytm BlaMeL.

Główna pętla *while*, powtarzana do czasu uzyskania wymaganego pasma, zawiera się w liniach od 15 do 35. Każdy przebieg pętli rozpoczyna się od sprawdzenia, czy bieżąca wartość priorytetu r nie jest niższa lub równa priorytetowi nowej ścieżki (linia 16). Spełnienie warunku oznacza, że pomimo zebrania wszystkich ścieżek o priorytecie utrzymania liczbowo wyższym od priorytetu utworzenia nowej ścieżki nie uzyskano wymaganej ilości pasma, a zatem niespełniony jest warunek na powodzenie wywłaszczania (wzór 3.33) i zwrócony zostaje pusty zbiór kandydatów. Następnie sprawdzany jest warunek, czy suma pasm dostępnych dla bieżącego priorytetu jest mniejsza lub równa wymaganemu pasmu. Jeśli tak, wówczas wszystkie ścieżki z odpowiedniego zbioru $\boldsymbol{K_p}$ są dodawane do listy kandydatów (linie 18-22). Jeśli natomiast pasmo zbioru jest większe niż wymagane, wówczas ścieżki ze zbioru są dodawane pojedynczo w następujący sposób. Najpierw wybierana jest najmniejsza ścieżka spośród tych, których pasmo jest większe od wymaganego (linie 26-27). W przypadku gdy wszystkie ścieżki mają pasmo niższe od wymaganego, do listy kandydatów dodawana jest pierwsza ścieżka z listy (linia 28). Następnie wybrana ścieżka zostaje usunięta z odpowiedniego zbioru $\boldsymbol{K_p}$, dodana do zbioru kandydatów $\boldsymbol{W}$ i obliczony zostaje bilans pasma (linie 29-31). Procedura zostaje zakończona po zebraniu wymaganego pasma i zwrócony zostaje

zbiór kandydatów.

Algorytm BlaMeL jest prostym algorytmem o zasięgu lokalnym ukierunkowanym na minimalizację priorytetów wywłaszczonych ścieżek a w dalszej kolejności liczbę wywłaszczeń i wywłaszczonego pasma. Należy zauważyć, że taki wybór podstawowego kryterium umożliwia łatwe zbudowanie algorytmu optymalnego w ramach pojedynczego łącza. Prostota koncepcji algorytmu została osiągnięta kosztem pewnej komplikacji wykorzystywanych struktur danych, gdyż w ogólności wymaga utworzenia ośmiu niezależnych list potencjalnych kandydatów.

Słabością metody BlaMeL jest zupełny brak możliwości sterowania kryterium wyboru kandydatów. W konsekwencji, jeśli użytkownik nie stawia priorytetu ścieżek na pierwszym miejscu, ten algorytm nie będzie spełniał jego oczekiwań.

3.5.5. Algorytm dokładny (optymalny)

Dokładny lub optymalny algorytm wywłaszczania to taki, który mając daną drogę przeznaczoną dla nowej ścieżki, w oparciu o ilości wolnego pasma i listę ścieżek utworzonych na poszczególnych łączach, wybiera najlepszy możliwy zbiór ścieżek przeznaczonych do wywłaszczenia. Rozstrzygnięcie tego, który zbiór jest najlepszy, opiera się na zadanej funkcji kosztu wywłaszczania, którym może być np. liczba wywłaszczonych ścieżek lub ilość wywłaszczonego pasma. Algorytm optymalny zawsze zwraca zbiór kandydatów, dla którego zadany koszt jest najmniejszy. Z tego powodu zwykle nie istnieje jeden idealny zbiór kandydatów do wywłaszczenia, ale jest tyle zbiorów ile możliwych kryteriów wyboru. Tym samym możliwe jest zdefiniowane wielu różnych algorytmów optymalnych, zależnie od wybranego kryterium. Niezależnie od tego różny może być zasięg algorytmu, tzn. wybór optymalny może dotyczyć całej domeny lub tylko pojedynczego łącza. W pierwszym przypadku będzie to zwykle metoda realizowana w sposób scentralizowany, a w drugim jako metoda zdecentralizowana.

Prostszym w analizie przypadkiem są optymalne algorytmy lokalne, które są wykonywane krok po kroku na kolejnych łączach wymagających wywłaszczania, a decyzja podejmowana jest wyłącznie w oparciu o stan na pojedynczym łączu. Dwa przykłady takich algorytmów dla sieci ATM przedstawiono w [83]. Pierwszy przyjmuje za cel minimalizację liczby wywłaszczonych ścieżek, drugi natomiast ma na celu minimalizację ilości wywłaszczonego pasma. Oba algorytmy zostaną tu omówione w oparciu o ich wersje zaadoptowane przez autora do zastosowania w sieciach MPLS.

Algorytm optymalny dla minimalizacji liczby wywłaszczonych ścieżek został przedstawiony na rys. 3.11. Po zainicjowaniu zbioru $\boldsymbol{W}$ kandydatów oraz obliczeniu bilansu pasma (linie 2-4) wyznaczona zostaje lista $\boldsymbol{K_e}$ potencjalnych kandydatów na danym łączu (linia 5). Następnie uruchamiana jest pętla *while* (linie 6-22) wykonywana tak długo, aż brakujące pasmo $B_r^{(e)}$ zmaleje do zera. Przebieg pętli odbywa się w dwóch etapach. Najpierw wyszukuje się pojedynczą ścieżkę o najmniejszym paśmie nie mniejszym od aktualnie wymaganego (li-

nie 8-12). Jeśli pojedyncza ścieżka nie zostanie znaleziona, wówczas wykonywany jest etap drugi (linie 13-17), w którym wybiera się ścieżkę o największym paśmie. Jeśli na danym etapie dostępna jest więcej niż jedna ścieżka o tym samym paśmie, wówczas wybiera się tą o niższym priorytecie. Po dokonaniu wyboru następuje przesunięcie ścieżki ze zbioru $\boldsymbol{K_e}$ do $\boldsymbol{W}$ i uaktualnienie bilansu pasma (linie 18-21). Po zebraniu wymaganego pasma następuje zakończenie procedury i zwrócenie zbioru kandydatów.

Złożoność obliczeniowa tego algorytmu to $O(|\boldsymbol{K_e}|^2)$ na każde łącze wymagające wywłaszczania, gdzie $|\boldsymbol{K_e}|$ to liczba potencjalnych kandydatów. Jest to unikalny przykład algorytmu optymalnego, który ma akceptowalną złożoność obliczeniową dzięki temu, że jest algorytmem lokalnym i nie zakłada minimalizacji pasma w przypadku istnienia kliku możliwych zbiorów o tej samej liczbie ścieżek.

Porównanie omówionego algorytmu z algorytmem Pey prowadzi do wniosku, że te algorytmy są w istocie bardzo podobne. Pey jest algorytmem optymalnym lokalnie pod względem liczby wywłaszczonych ścieżek. Jego jakość jest więc dobrym poziomem odniesienia przy interpretacji wyników badań pozostałych algorytmów.

```
1   procedure algorithm_OptRC_local ()
    # algorytm lokalny, dla uproszczenia podano procedurę dla
    # pojedynczego węzła, powtarzaną w razie potrzeby w następnych węzłach
2     W := ∅                          # inicjuj listę kandydatów
3     S_e := a_f^(e) – b_q            # oblicz początkowy bilans pasma
4     B_r^(e) := max (0, –S_e)        # oraz brakujące pasmo
5     K_e := cand_preselect_on_link (e)     # wyznaczenie lokalnego zbioru
                                            # potencjalnych kandydatów
6     while B_r^(e) ≠ 0 do            # dopóki brakuje pasma
7       i := 0                        # oznaczenie pierwszego przebiegu pętli for
8       for each k in K_e             # dla każdego potencjalnego kandydata
9         if i = 0 and b_k ≥ B_r^(e) then p := k
                                      # zapamiętaj pierwszą ścieżkę ze zbioru K_e
10        if i > 0 and b_k ≥ B_r^(e) and
11          (b_k < b_p or (b_k = b_p and h_k > h_p)) then p := k
                                      # wybierz najmniejszą ścieżkę o paśmie
                                      # większym niż wymagane pasmo
12      end for
13      if i = 0 then                 # jeśli brak pojedynczej ścieżki
                                      # o paśmie większym niż wymagane
14        i := 1, p := K_e[0]
```

```
15          for each k in K_e do      # wybierz największą dostępną ścieżkę
16            if b_k > b_p or (b_k = b_p and h_k > h_p)) then p := k
17        end if
18        K_e := K_e \ {p}            # usuń ścieżkę z listy
19        W := W ∪ {p}                # dodaj ścieżkę do kandydatów
20        S_e := S_e + b_p            # uaktualnij bilans pasma
21        B_r^(e) := max (0, –S_e)    # oblicz brakujące pasmo
22      end while
23      return W                      # zwróć zbiór kandydatów
24  end
```

Rys. 3.11. Algorytm optymalny (lokalny) dla minimalizacji liczby wywłaszczeń.

Optymalny algorytm lokalny dla minimalizacji wywłaszczonego pasma został przedstawiony na rys. 3.12. Dokonuje on sprawdzenia wszystkich możliwych kombinacji w zbiorze potencjalnych kandydatów, poczynając od pojedynczej ścieżki, następnie analizując wszystkie pary, potem wszystkie 3-elementowe kombinacje itd., a kończąc na pojedynczym zbiorze zawierającym wszystkie dostępne ścieżki. Kolejne kombinacje są porównywane z najlepszym znalezionym dotychczas zbiorem, dla którego suma pasm jest najmniejsza, ale jednocześnie większa lub równa brakującemu pasmu. Złożoność obliczeniowa tego algorytmu jest bardzo duża i wynosi $O(|\boldsymbol{K_e}|\cdot 2^{|\boldsymbol{K_e}|-1})$, co tym samym uniemożliwia jego praktyczne zastosowanie.

```
1   procedure algorithm_OptBw_local ()
    # algorytm lokalny, dla uproszczenia podano procedurę dla
    # pojedynczego węzła, powtarzaną w razie potrzeby w następnych węzłach
2     K_e := cand_preselect_on_link (e)  # wyznaczenie lokalnego zbioru
                                         # potencjalnych kandydatów
3     k := |K_e|                    # liczba potencjalnych kandydatów
4     min_sum := min_prio := ∞      # inicjuj metryki
5     length (best_set) := 0        # wyzeruj najlepszy zbiór kandydatów
6     for r := 1 to k do            # dla każdej możliwej liczności
                                    # zbioru kandydatów
7       for n := 0 to r-1 do
8         set[n] := n + 1           # inicjuj tablicę kolejnymi liczbami
                                    # naturalnymi od 1 do r
9       for i :=1 to k!/(k – r)!r! do      # dla każdej kombinacji r ścieżek
```

```
10          sum_bw := sum_prio := 0        # wyzeruj sumy
11          for n := 0 to r-1 do
12             p := K_e[set[n]-1]          # przypisz ścieżkę należącą do zbioru
13             sum_bw := sum_bw + b_p      # uaktualnij sumę pasma
14             sum_prio := sum_prio + h_p  # uaktualnij sumę priorytetów
15          end for
16          if sum_bw < min_sum and sum_bw ≥ B_r^(e) then
                                           # jeśli suma pasm ścieżek jest wystarczająca
                                           # i mniejsza od dotychczas znalezionych
17             best_set := set             # zapamiętaj ten zbiór ścieżek jako najlepszy
18          if sum_bw = min_sum and sum_bw ≥ B_r^(e)
19          and sum_prio > min_prio then
                                           # jeśli suma pasm ścieżek równa dotychczas
                                           # najlepszej ale sumarycznie niższe priorytety
20             best_set := set             # zapamiętaj tez zbiór jako lepszy
            # generuj kolejną kombinację ścieżek
21          pos := r
22          while pos > 0 and set[pos] = k – r + pos + 1 do
23             pos := pos – 1
24          if set[pos] < k – r + pos + 1 then
25             set[pos] := set[pos] + 1
26          spos := pos + 1
27          for pos := spos to r do
28             set[pos] := set[pos-1] + 1
29       end for
30    end for
31    W := ∅                               # inicjuj listę kandydatów
32    for pos := 0 to length (best_set)-1 do
33       W := W ∪ K[set[pos]-1]            # przepisz ścieżki ze zbioru set do listy kandydatów
34    return W                             # zwróć listę kandydatów
35 end
```

Rys. 3.12. Algorytm optymalny lokalny dla minimalizacji wywłaszczonego pasma.

Algorytmy o zasięgu domeny są znacznie bardziej złożone, gdyż wymagają sprawdzenia wszystkich ścieżek obecnych na wszystkich łączach na drodze połączeniowej, na których występuje niedostatek pasma. W dostępnej literaturze nie natrafiono na przykład takiego algorytmu. Implementacja opracowana przez autora została przedstawiona na rys. 3.13. Zastosowano tu parametr *variant* określający kryterium optymalnego wyboru: *MIN_RC* ozna-

cza minimalizację liczby wywłaszczeń, a *MIN_BW* minimalizację wywłaszczonego pasma. Algorytm działa na zasadzie wygenerowania wszystkich możliwych kombinacji kandydatów i porównaniu ich, po czym najlepszy wybór zostaje zwrócony jako lista kandydatów do wywłaszczenia.

```
1   procedure algorithm_Opt_domain (variant: {MIN_RC, MIN_BW})
2     W := ∅                                  # inicjuj listę kandydatów
3     K := cand_preselect_on_route ()         # wybierz potencjalnych kandydatów
4     k := |K|                                # liczba potencjalnych kandydatów
5     for r := 0 to k-1 do                    # dla każdej możliwej liczby kandydatów
                                              # r+1 to aktualna liczba kandydatów
6       if W ≠ ∅                              # znaleziono przynajmniej jeden zbiór
7       and variant = MIN_RC then             # metoda minimalizacji
                                              # liczby wywłaszczeń
8         break                               # procedura zakończona sukcesem
9       pos := 0
10      set[0] := -1                          # inicjalizacja zbioru kandydatów
11      repeat
12        spos := set[pos]
13        while pos ≤ r do
            # wygeneruj pierwszy lub kolejny zbiór
            # zaczynając od zwiększenia numeru na pozycji pos
14          spos := spos + 1
15          set[pos] := spos
16          pos := pos + 1
17        end while
18        if set[r] < k then
19          Y := ∅                            # aktualnie analizowany zbiór kandydatów
20          for pos := 0 to r-1 do
21            Y := Y ∪ K[set[pos]-1]          # utwórz zbiór kandydatów
                                              # według numerów zapisanych w set
22          if is_balance_positive (Y)        # zbiór zapewnia brakujące pasmo
23          and found_better_set (W, Y, variant) then
                                              # znaleziono zbiór lepszy od W
24            W := Y                          # zapamiętaj aktualny zbiór
25        end if
26        pos := r                            # rozpocznij od ostatniej pozycji
```

```
27          if set[pos] ≥ k-1 then          # osiągnięto największy numer ścieżki
                                            # na pozycji pos
28              pos := pos – 1              # przejdź na poprzednią pozycję
29              while pos > 0 and set[pos] ≥ k – (r – pos) – 1 do
                                            # przejdź do pozycji, na której
                                            # można zwiększyć numer ścieżki
30                  pos := pos – 1
31          end if
32      until set[0] ≥ k – r                # zakończ gdy na pierwszej pozycji
                                            # osiągnięto największy możliwy numer ścieżki
33  end for
34  return W                                # zwróć listę kandydatów
35  end
36
37  function is_balance_positive (Y)
38      for each e in R̂ do                  # oblicz bilans początkowy
39          S_e := a_f^(e) – b_q            # ile pasma brakuje na łączu

40      # dodaj pasmo jakie można uzyskać z wywłaszczenia
41      for each p in Y do                  # dla każdej ścieżki ze zbioru Y
42          for each e in R̂ do              # na każdym łączy na wybranej drodze
43              if e ∈ R_p then             # ścieżka-kandydat zajmuje łącze e
44                  S_e := S_e + b_p        # powiększ bilans o pasmo ścieżki p

45      # sprawdź obliczony bilans na każdym łączu
46      for each e in R̂ do                  # na każdym łączu na wybranej drodze
47          if S_e < 0 then
48              return FALSE                # błąd: zbiór Y zwalnia za mało pasma
49
50      return TRUE                         # sukces: zbiór Y zwalnia
                                            # wystarczającą ilość pasma
51  end
52
53  function found_better_set (W, Y, variant)
54      better := FALSE
55      if W = ∅ then                       # jest to pierwszy akceptowalny zbiór
```

```
56        return TRUE
57     if variant = MIN_RC then # minimalizuj liczbę wywłaszczeń
58        if |Y| < |W| then          # Y zawiera mniej ścieżek do wywłaszczenia
59           return TRUE
60        if |Y| = |W| then          # jeśli oba zbiory zawierają
                                     # taką samą liczbę kandydatów
61           if sum_bw(Y) < sum_bw(W) then    # decyduje mniejsze pasmo
62              return TRUE
63           if sum_bw(Y) = sum_bw(W)         # w przypadku jednakowych pasm
64           and sum_prio(Y) > sum_prio(W) then       # decydują niższe priorytety
65              return TRUE
66        end if
67     else begin                               # minimalizuj wywłaszczone pasmo
68        if sum_bw(Y) < sum_bw(W) then       # zbiór Y zawiera mniejsze pasmo
69           return TRUE
70        if sum_bw(Y) = sum_bw(W)  then      # jeśli sumy pasm są identyczne
71           if |Y| < |W| then          # decyduje mniejsza liczba wywłaszczeń
72              return TRUE
73           if |Y| = |W|               # jeśli liczby wywłaszczeń są jednakowe
74           and sum_prio(Y) > sum_prio(W) then       # decydują niższe priorytety
75              return TRUE
76        end if
77     end if
78     return FALSE               # zbiór W pozostaje lepszy od aktualnego
79  end
```

Rys. 3.13. Uniwersalny algorytm optymalny o zasięgu domeny.

Po zainicjowaniu zmiennych i zbiorów globalnych (linie 2-4) rozpoczyna się główna pętla wyboru kandydatów (linie 5-33). Każde powtórzenie tej pętli oznacza zwiększenie liczności zbioru kandydatów, rozpoczynając od zbiorów zawierających pojedyncze ścieżki, a kończąc na pojedynczym zbiorze zawierającym wszystkie ścieżki ze zbioru potencjalnych kandydatów ***K***.

Każdy przebieg pętli rozpoczyna się od sprawdzenia (linie 6-7), czy w poprzednim przebiegu odnaleziono zbiór kandydatów. Jeśli tak jest i zadanym kryterium wyboru była minimalizacja liczby wywłaszczeń, wówczas algorytm zostaje zakończony z sukcesem, gdyż kolejne powtórzenie spowodowałoby wygenerowanie zbioru o większej liczbie kandydatów, a więc gorszego w świetle wybranego kryterium. Jeśli warunek nie jest spełniony, wówczas następuje kontynuacja przebiegu wewnątrz pętli.

Pętla w liniach 11-32 odpowiada za wygenerowane i sprawdzenie jakości kolejnego zbioru kandydatów o tej samej liczności, zadanej przez zewnętrzną pętlę *for* (linie 5-33) w zmiennej *r*. W pętli *while* (linie 13-17) następuje wygenerowanie listy wskaźników do elementów ze zbioru ***K***, a ściślej numerów pozycji w zbiorze. Przykładowo, mając danych dziesięciu potencjalnych kandydatów i generując zbiór *set* o liczności 5 (*r*=4), generowane są kolejno ciągi, począwszy od [0 1 2 3 4], następnie [0 1 2 3 5] aż po [5 6 7 8 9]. W kolejnym etapie następuje utworzenie dodatkowego zbioru ***Y*** (linie 19-21), zawierającego ścieżki wybrane zgodnie z aktualną zawartością zbioru *set*. Jeśli dany zbiór ***Y*** jest lepszy od poprzedniego (sprawdzenie w liniach 22-23), wówczas zostaje zapamiętany jako aktualny zbiór kandydatów ***W***. Na koniec następuje przygotowanie do wygenerowania kolejnego zbioru *set* (linie 27-31).

Po wyjściu z obu pętli, tj. po sprawdzeniu wszystkich możliwych kombinacji potencjalnych kandydatów, następuje zwrócenie najlepszego znalezionego zbioru ***W*** kandydatów (linia 34). Ponieważ jest to algorytm globalny, więc obliczenie i sprawdzenie bilansu pasma w funkcji *is_balance_positive* jest wykonywane na każdym łączu wymagającym wywłaszczania.

3.5.6. Inne opracowania

Zupełnie odmienne podejście do wywłaszczania połączeń (w sieciach ATM) przedstawiono w artykule [13]. Jedynym kryterium wyboru ścieżek-kandydatów jest w tym przypadku czas, który upłynął od ich utworzenia. Przedstawiono szeroką analizę teoretyczną szeregu algorytmów opartych na tej zasadzie. Wydaje się jednak, że podejście takie nie mieści się w koncepcji sieci MPLS, która z założenia jest siecią szkieletową a ścieżki połączeniami długoterminowymi, bez określonego rozkładu czasów trwania.

Podobne podejście zaproponowano w [2], określając dla każdej ścieżki czas, jaki upłynął pomiędzy jej utworzeniem a wywłaszczeniem, i na tej podstawie wyznaczając oczekiwany stopień zadowolenia klienta. Bazuje się tu na przykładowych czasach trwania transmisji dla połączeń określonego typu, która jednak, jak przyznają autorzy, jest znana jedynie użytkownikowi końcowemu. Niewątpliwie satysfakcja użytkownika jest bardzo istotnym aspektem wywłaszczania, choć dyskusyjne są założenia o określonym czasie wystarczającym na wykonanie pojedynczego transferu danych w ścieżce.

Autorzy artykułu [102] z kolei zaproponowali udoskonalenie algorytmu CSPF (ang. *Constraint Shortest Path First*) w taki sposób, aby ten w procesie wyboru drogi dla nowej ścieżki brał pod uwagę pasmo zajęte przez ścieżki o niższym priorytecie i obsługiwał mechanizm wywłaszczania. Jako algorytmu wywłaszczania autorzy używają omówionego wcześniej algorytmu optymalnego dla minimalizacji liczby wywłaszczeń (rys. 3.11). W artykule omówiono również istotne dla wywłaszczania elementy protokołów RSVP-TE, CR-LDP i OSPF, a także przedstawiono szereg własności metryk i algorytmów zaproponowanych dla CSPF.

Nieco podobną zasadę zastosowano w [56]. Tu również już w procesie wyboru trasy bierze się pod uwagę pasmo dostępne dla ścieżki o określonym priorytecie, uwzględniając ewen-

tualne wykonywanie wywłaszczeń. Dodatkowo modyfikuje się funkcję kosztu łączy tak, aby uwzględnić priorytety potencjalnych kandydatów do wywłaszczenia oraz pasmo niezbędne do wywłaszczenia.

Zagadnienie rutingu CSPF świadomego istnienia wywłaszczeń opisano także w [108], w którym dodatkowo zaproponowano wariant algorytmu V-PREPT zaadoptowany z [73]. Wyniki symulacyjne uzupełniono aspektami praktycznej implementacji metody z użyciem protokołu RSVP-TE.

Jedne z najprostszych algorytmów wywłaszczania zaproponowano w [100,101]. Oba algorytmy są oparte na zasadzie losowego wyboru ścieżek do wywłaszczenia z dostępnego zbioru potencjalnych kandydatów. Nie ma możliwości wpływania na sposób wyboru ścieżek.

Bazując na oczywistym stwierdzeniu, że kolejność żądań utworzenia ścieżek ma wpływ na liczbę wywłaszczeń, zaproponowano [27] sortowanie żądań według pasma a następnie tworzenie ścieżek w odpowiedniej kolejności. Zaprezentowano wyniki symulacyjne liczby odrzuconych zgłoszeń i maksymalnego obciążenia łączy, bazując na pomiarach jednej wybranej topologii. W konkluzji podkreślono, że sortowanie kolejności żądań według pasma umożliwia zmniejszenie liczby odrzuceń. W praktyce takie podejście jest ograniczone do specyficznych zastosowań, w których z góry znane są wszystkie żądania.

Istnieje kilka opracowań, w których autorzy dążą do minimalizacji liczby wywłaszczeń poprzez taki przydział zasobów, aby w ogóle unikać wywłaszczeń lub minimalizować ich liczbę. W sytuacji braku wolnego pasma jednym ze sposobów jest podjęcie próby zmniejszenia przydzielonego pasma tym ścieżkom, które przenoszą ruch elastyczny i w ten sposób uniknięcie wywłaszczeń [72,105]. Innym sposobem na ograniczenie liczby wywłaszczeń jest unikanie mocno obciążonych łączy przy wybieraniu trasy dla nowej ścieżki [10,32].

Dwa proste algorytmy wywłaszczania ścieżek ukierunkowane na sieci ze wsparciem dla technologii Differentiated Services (DiffServ) [19] zaprezentowano w [97]. Zakładają one wykorzystanie jednego z dwóch modeli *Maximum Allocation Bandwidth Constraints Model* (MAM) [11,63] lub *Russian Dolls Bandwidth Constraints Model* (RDM) [62], w których pula pasma, jakie może być zarezerwowane dla ścieżek o danym priorytecie, jest administracyjnie ograniczona.

Jednym z praktycznych problemów związanych z wywłaszczaniem jest niebezpieczeństwo zbyt częstego wywłaszczania ścieżek o niskim priorytecie utrzymania. Jedną z propozycji rozwiązania tego problemu jest ograniczenie częstości wywłaszczania za pomocą żetonów (tokenów) przydzielanych ścieżkom i odbieranych w momencie wywłaszczenia [26].

Zagadnienie wywłaszczania w kontekście praktyki operatora telekomunikacyjnego omówiono w [38], gdzie w oparciu o istnienie czterech klas jakości obsługi: złotej (*gold*), srebrnej (*silver*), ekonomicznej (*economy*) i bez gwarancji (*best-effort*) wprowadzono dodatkowe ograniczenia na możliwość wzajemnego wywłaszczania pomiędzy ścieżkami z poszczególnych klas, np. unikanie wywłaszczeń ścieżek z klasy ekonomicznej przez ścieżki z klasy

srebrnej.

Należy również nadmienić, że podejmowano próby rozwiązania problemu wywłaszczania za pomocą mechanizmów logiki rozmytej i algorytmów genetycznych [31,104].

3.6. Złożoność obliczeniowa

Przeprowadzono analizę złożoności obliczeniowej omówionych wcześniej czterech algorytmów heurystycznych. Na jej potrzeby dokonano redukcji przedstawionego wcześniej kodu do instrukcji niezbędnych do określenia złożoności. Zwykle do tego typu analizy wystarczy wyodrębnienie głównej pętli procedury realizującej wybór kandydatów. Dla ułatwienia analizy poszczególnym pętlom *for*, *while* i *repeat-until* przyporządkowano w komentarzach do kodu oznaczenia literowe *A*, *B*, *C* itd.

3.6.1. Algorytm GarGop

Algorytm GarGop/RC (rys. 3.5) redukuje się do postaci przedstawionej na rys. 3.14. Usunięto z niego wszystkie instrukcje znajdujące się poza główną pętlą. Dla łatwiejszej analizy włączono niezbędne części kodu wywoływanych funkcji.

```
1   repeat
2     # best_path ()
3     for each p in K
4       for each e in R̂
5       end for
6     end for
7     # update_bw_balance (p)
8     for each e in R̂
9     end for
10  until B_r = 0
```

Rys. 3.14. Analiza złożoności obliczeniowej algorytmu GarGop/RC, krok 1/2.

Już nawet pobieżna analiza pozwala poczynić dalsze uproszczenia. Mianowicie, instrukcje zawarte w funkcji *update_bw_balance* można pominąć, gdyż wykonywane są w szeregu z bardziej złożoną pętlą w funkcji *best_path*. Zatem ostateczna, zredukowana postać algorytmu GarGop/RC jest taka, jak na rys. 3.15.

Pętla *C* algorytmu GarGop/RC będzie więc wykonywana tyle razy, ile wynosi iloczyn liczby kandydatów (pętla *A*), liczby potencjalnych kandydatów (pętla *B*) i liczby łączy wymagających wywłaszczenia (pętla *C*). Ograniczeniem górnym dla tych trzech wielkości są odpowiednio liczba ścieżek *Z*, liczba ścieżek *Z* i liczba węzłów *N*. Zatem złożoność obliczeniowa algorytmu ma postać $\boldsymbol{O(NZ^2)}$.

```
1   repeat                  # A
2     # best_path
3     for each p in K       # B: wykonywane dla każdej potencjalnej
                            #    ścieżki-kandydata
4       for each e in R̂     # C: w rzeczywistości wykonywane tyle razy,
5                           #    ile jest łączy z wywłaszczaniem, zwykle 1
6       end for
7     end for
8   until B_r = 0           # A: wykonywane tak długo, aż zebrany zostanie
                            #    cały zbiór kandydatów
```

Rys. 3.15. Analiza złożoności obliczeniowej algorytmu GarGop/RC, krok 2/2.

Warto dodać, że w praktyce w sieciach złożonych z kilkudziesięciu węzłów liczba łączy wymagających wywłaszczania zwykle nie przekracza dwóch [46] a zatem czynnik N jest do pominięcia w tym i pozostałych analizowanych algorytmach.

Algorytm **GarGop/BW** (rys. 3.6) różni się od GarGop/RC obliczeniami zawartymi w funkcji *best_path*. Liczba przejść pętli C jest w obu algorytmach identyczna, a zatem złożoność algorytmu GarGop/BW jest taka sama jak dla GarGop/RC, tzn. $\boldsymbol{O(NZ^2)}$.

3.6.2. Algorytm Pey

Algorytm Pey (rys. 3.7) na potrzeby analizy złożoności został zredukowany do docelowej postaci pokazanej na rys. 3.16.

```
1   for each e in R̂         # A: wykonywana dla każdego łącza z wywłaszczaniem
2     while B_r^(e) ≠ 0 do  # B: wykonywana aż zebrany zostanie zbiór kandydatów
3       # best_path ()
4       for each p in K_e   # C: wykonywana dla każdego potencjalnego kandydata
5       end for             # C
6     end while             # B
7   end for                 # A
```

Rys. 3.16. Analiza złożoności obliczeniowej algorytmu Pey.

Na tej podstawie natychmiast otrzymujemy liczbę wykonań wewnętrznej pętli C równą iloczynowi liczby łączy wymagającej wywłaszczenia (pętla A), liczby kandydatów (pętla B) i liczby potencjalnych kandydatów (pętla C). Biorąc górne ograniczenia dla tych wielkości jako, odpowiednio, liczbę węzłów N, liczbę ścieżek Z i liczbę ścieżek Z otrzymujemy wartość złożoności obliczeniowej w postaci $\boldsymbol{O(NZ^2)}$.

3.6.3. Algorytm OliSco

Dekonstrukcja algorytmu OliSco (rys. 3.9) została przeprowadzona w dwóch etapach. Analizę rozpoczęto od postaci rozwiniętej, zawierającej wszystkie warianty oryginalnej procedury (rys. 3.17).

```
1   for each e in R̂          # A: wykonywana dla każdego łącza z wywłaszczaniem
2     while B_r^(e) ≠ 0 do   # B: wykonywana aż zebrany zostanie zbiór kandydatów
3       if () then ()        # w przypadku różnych metryk
4       else                 # w przypadku identycznych metryk
5         for each p in L do     # C1: wykonywana dla różnych metryk
6           for each p in L' do  # C2: jak wyżej
7           for each p in L' do  # C3: jak wyżej
8           for each p in L' do  # C4: jak wyżej
9         end if
10    end while              # B
11  end for                  # A
```

Rys. 3.17. Analiza złożoności obliczeniowej algorytmu OliSco, krok 1/2.

W przypadku ścieżek o różnych metrykach wewnętrzna pętla *B* będzie wykonywana tyle razy, ile wynosi iloczyn liczby łączy z wywłaszczaniem (pętla *A*) i liczby kandydatów (pętla *B*). Natomiast w przypadku jednakowych metryk ścieżek redukujemy cztery pętle *C1*…*C4* do pojedynczej pętli *C*, gdyż w analizie złożoności obliczeniowej pomija się współczynniki całkowite [60]. Otrzymujemy w ten sposób postać algorytmu przedstawioną na rys. 3.18.

```
1   for each e in R̂          # A: wykonywana dla każdego łącza z wywłaszczaniem
2     while B_r^(e) ≠ 0 do   # B: wykonywana aż zebrany zostanie zbiór kandydatów
3       if () then ()        # w przypadku różnych metryk
4       else                 # w przypadku identycznych metryk
5         for each p in L' do    # C: wykonywana dla różnych metryk
6       end if
7     end while              # B
8   end for                  # A
```

Rys. 3.18. Analiza złożoności obliczeniowej algorytmu OliSco, krok 2/2.

Łatwo jest zauważyć, że liczba przejść w pętli *C* powoduje odpowiednie zmniejszenie liczby przejść w pętli *B* (odpowiednio większa liczba ścieżek dodanych do listy kandydatów). Zatem wynik dla jednakowych metryk jest taki sam jak dla różnych metryk, tj. iloczyn liczby

łączy z wywłaszczaniem i liczby kandydatów. Daje to złożoność algorytmu równą ***O*(*NZ*)**.

3.6.4. Algorytm BlaMeL

Algorytm BlaMeL (rys. 3.10) został zredukowany na potrzeby analizy złożoności obliczeniowej do postaci przedstawionej na rys. 3.19.

```
1   for each e in R̂          # A: wykonywana dla każdego łącza z wywłaszczaniem
2     while B_r^(e) ≠ 0 do   # B: wykonywana dla każdego priorytetu
3                            #    aż zebrany zostanie zbiór kandydatów
4       if B_r^(e) ≥ B_hold[r] then
5       end
6       else
7         while B_r^(e) <> 0 do     # C: wykonywana dla ścieżek
8                                   #    o tym samym priorytecie
9           while i < |K_p[r]| –1                 # D: przeszukiwana lista
10            and K_p[r][i+1] > B_r^(e) do        #    dla priorytetu r
11          end while                             # D
12      end if
13    end while          # B
14  end for              # A
```

Rys. 3.19. Analiza złożoności obliczeniowej algorytmu BlaMeL, krok 1/2.

Pętla *C* może być w rozważaniach pominięta, gdyż każde jej przejście powoduje zmniejszenie liczby przejść pętli *B*, w przeciwieństwie do pętli *D*, która ma bezpośredni wpływ na czas wykonania. W najgorszym przypadku, gdy wszystkie ścieżki-kandydaci mają identyczny priorytet, otrzymujemy postać algorytmu przedstawioną na rys. 3.20.

```
1   for each e in R̂ do       # A: wykonywana dla każdego łącza z wywłaszczaniem
2     while B_r^(e) ≠ 0 do   # B: wykonywana dla każdego priorytetu
3                            #    aż zebrany zostanie zbiór kandydatów
4       while i < |K_p[r]| –1 and K_p[r][i+1] > B_r^(e) do   # D: przeszukiwana lista
5                                                            #    dla priorytetu r
6     end while          # B
7   end for              # A
```

Rys. 3.20. Analiza złożoności obliczeniowej algorytmu BlaMeL, krok 2/2.

Zatem liczba powtórzeń pętli *D* algorytmu BlaMeL jest iloczynem liczby łączy z wy-

właszczaniem (pętla *A*), liczby kandydatów (pętla *B*) i liczby potencjalnych kandydatów (pętla *D*). To daje złożoność obliczeniową równą $\boldsymbol{O(NZ^2)}$.

3.7. Analiza porównawcza

Mając podbudowę w postaci charakterystyki poszczególnych algorytmów możliwe jest dokonanie porównania ich cech i możliwości, bez czego owa charakterystyka byłaby w istocie niepełna. W tym celu wybrano cztery cechy, według których zdecydowano się dokonać oceny opisanych algorytmów.

1. *Zasięg algorytmów* określa, czy jest to algorytm lokalny czy globalny. W pierwszym przypadku decyzja podejmowana jest w oparciu o informacje dostępne na pojedynczym łączu wyjściowym, a w drugim bierze się pod uwagę dane ze wszystkich łączy na trasie ścieżki, w granicach domeny administracyjnej. Z uwagi na jakość osiąganych rezultatów większe możliwości mają algorytmy o zasięgu globalnym, co odbywa się kosztem większej ilości wymaganych informacji oraz większej złożoności implementacyjnej i obliczeniowej.

2. *Elastyczność kryteriów wyboru* umożliwia określenie celu algorytmu zależnie od preferencji użytkownika, choćby w ograniczonym zakresie. Niektóre metody na sztywno definiują funkcję kosztu i nie pozostawiają żadnych możliwości konfiguracyjnych, podczas gdy inne zapewniają możliwość wpływania na kryteria wyboru ścieżek.

3. *Złożoność obliczeniowa* jest szacunkową, względną miarą czasu wykonywania algorytmu, rozważaną dla najgorszego możliwego przypadku. Określa ona to, czy dana metoda nadaje się do zastosowania w rzeczywistej sieci i jak zmienia się jej szybkość w zależności od rozmiaru danych wejściowych, np. liczby potencjalnych kandydatów. Przyjmuje się, że algorytmy o złożoności potęgowej (n, n^2, n^3, itd.) są akceptowalne, natomiast algorytmy o złożoności wykładniczej (a^k, gdzie a jest stałą, zwykle równą 2) już nie. Oczywiście im mniejsza złożoność (mniejszy wykładnik potęgi), tym lepiej.

4. *Jakość wyboru kandydatów* przejawia się w zdolności do minimalizacji kosztu wywłaszczania, zgodnie z definicjami z rozdz. 3.4.1.

Dodatkową cechą, która wprawdzie może być nieistotna w wielu implementacjach, ale jest warta omówienia, jest *świadomość priorytetów*, czyli taki sposób wyboru kandydatów, w którym ścieżki o wyższym priorytecie są wybierane rzadziej. O ile dany algorytm posiada tę cechę, to albo uznaje priorytety ścieżek za najważniejszy czynnik wyboru kandydatów (algorytm BlaMeL) lub pozwala na uwzględnienie go w parametrach algorytmu (algorytm OliSco).

Odpowiednie zestawienie oparte na trzech pierwszych cechach i świadomości priorytetów zawarto w tabeli 3.1. Czwarta cecha, określająca jakość wyboru, zostanie zweryfikowana w rozdziale szóstym, w którym opisano wyniki badań symulacyjnych. Z informacji zawartych w tabeli wynikają następujące wnioski. Z wyjątkiem algorytmu GarGop pozostałe mają za-

sięg lokalny, a zatem biorą pod uwagę tylko stan na bieżącym łączu. Z kolei algorytmy GarGop i Pey nie biorą pod uwagę priorytetu wywłaszczanych ścieżek, choć trzeba zaznaczyć, iż w dwóch pozostałych algorytmach różnie podchodzi się do wagi priorytetów, bo w BlaMeL jest to najważniejsze kryterium, a w OliSco konfigurowalne. Należy zaznaczyć, iż nawet jeśli algorytm nie bierze pod uwagę priorytetów, jego implementacja w sieci MPLS musi wykluczać z listy potencjalnych kandydatów ścieżki o wyższym priorytecie utrzymania niż priorytet utworzenia nowej ścieżki. Jeśli chodzi o elastyczność kryteriów, to tylko metoda OliSco oferuje taką możliwość, a na dodatek umożliwia dowolne ich określenie z użyciem kilku zmiennych. Można tu dodać, że oba omówione warianty algorytmu GarGop oparte są na dwóch różnych procedurach, które można zastosować zależnie od potrzeby, jednak trudno to traktować jako podejście elastyczne z punktu widzenia definicji kryteriów. W odniesieniu do złożoności obliczeniowej, dla wszystkich opisywanych metod jest ona akceptowalna, a szczególnie wyróżnia się tu algorytm OliSco, który cechuje najmniejsza złożoność.

Tabela 3.1. Porównanie algorytmów wywłaszczania.

Algorytm	Zasięg	Świadomość priorytetów	Elastyczność kryteriów wyboru	Złożoność obliczeniowa
GarGop	Globalny	Nie	Nie	$O(NZ^2)$
Pey	Lokalny	Nie	Nie	$O(NZ^2)$
OliSco	Lokalny	Tak	Tak	$O(NZ)$
BlaMeL	Lokalny	Tak	Nie	$O(NZ^2)$

Warto dodać, iż wszystkie omówione algorytmy heurystyczne należą do klasy algorytmów zachłannych [28], gdyż wybór pada zawsze na tą ścieżkę, która na danym etapie algorytmu ma największą wartość zadanej metryki. W efekcie ścieżka dodana do listy kandydatów nie jest już z niej usuwana. Jedynie algorytm OliSco dopuszcza taką sytuację, ale tylko w przypadku, gdy wśród rozpatrywanych potencjalnych kandydatów są ścieżki o identycznej metryce i pasmo jednej z nich przekracza pasmo wymagane w momencie rozpoczęcia działania algorytmu (patrz rys. 3.9, linie 21-23). Jak zobaczymy w rozdziale 4, algorytm może zupełnie odejść od tej zasady i dowolnie usuwać kandydatów wcześniej dodanych do listy.

Przedstawione zestawienie pozwala zauważyć trywialny fakt, że liczba dostępnych i jednocześnie uniwersalnych algorytmów nie jest duża, co w połączeniu z wysokim stopniem skomplikowania zagadnienia daje możliwości poszukiwania doskonalszych rozwiązań. Bazując na przedstawionym zestawieniu żaden z algorytmów nie jest zdecydowanie lepszy od innych. Bliski wysokiej ocenie jest algorytm OliSco ze względu na świadomość priorytetów i elastyczność kryteriów wyboru, choć ma zasięg lokalny, a jego prostota pozostawia wątpli-

wości odnośnie jakości generowanych rozwiązań.

Przedstawioną charakterystykę można podsumować stwierdzeniem, że wprawdzie wszystkie omówione algorytmy wydają się być akceptowalne pod kątem praktycznego zastosowania, to jednak nie dysponujemy efektywnym obliczeniowo i elastycznym algorytmem o zasięgu domeny. Efektem prac nad spełnieniem tych wymagań jest algorytm opisany w kolejnym rozdziale.

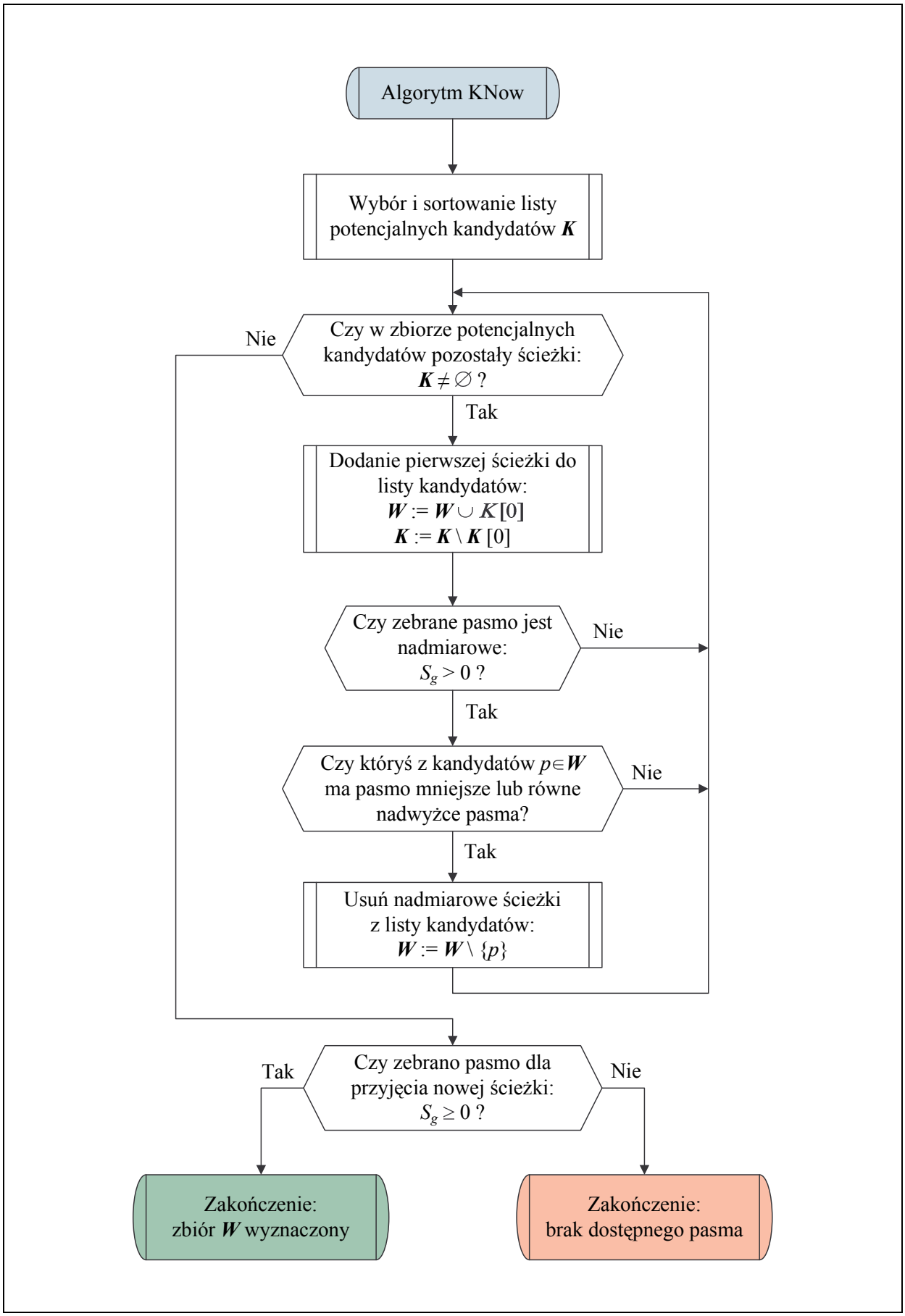
4. PROPONOWANY ALGORYTM

W ramach badań nad mechanizmami wywłaszczania ścieżek opracowano i przetestowano własny algorytm heurystyczny. Przeprowadzone badania symulacyjne wykazały jego wysoką jakość w sieciach o różnej wielkości i topologii. Proponowany algorytm obejmuje zasięgiem domenę i umożliwia wybór kryterium wywłaszczania. Cechuje go złożoność obliczeniowa podobnego rzędu jak pozostałych algorytmów heurystycznych.

4.1. Opis algorytmu

W trakcie badań istniejących heurystycznych algorytmów wywłaszczania zdecydowano się na opracowanie nowego algorytmu globalnego, który byłby konkurencyjny w stosunku do istniejących algorytmów przy zachowaniu akceptowalnej złożoności obliczeniowej. W większości znanych algorytmów kolejne ścieżki wybiera się w pętli i dodaje kolejno do zbioru kandydatów. Proponowany algorytm działa inaczej. Tu kolejne ścieżki są dodawane do listy kandydatów (bez żadnych specyficznych warunków wstępnych), ale po każdym dodaniu ścieżki sprawdzany jest bilans pasma. Jeśli jest on dodatni, cały zbiór kandydatów jest przeglądany pod kątem nadmiarowych ścieżek, tzn. takich, które można usunąć ze zbioru kandydatów przy zachowaniu zdolności do przyjęcia nowej ścieżki. Oczekuje się, iż taki sposób działania umożliwi bardziej efektywny wybór kandydatów niż ma to miejsce w przypadku opisanych wcześniej algorytmów. Proponowany algorytm został nazwany KNow, dla zachowania konwencji nazewniczej użytej dla opisanych wcześniej algorytmów heurystycznych.

Istota działania najważniejszej części algorytmu została zilustrowana schematem przedstawionym na rys. 4.1. W pierwszym kroku tworzony jest zbiór potencjalnych kandydatów $\boldsymbol{K}$ obejmujący ścieżki z każdego łącza wymagającego wywłaszczania. Zbiór jest zaimplementowany w postaci listy, która może być sortowana w określony sposób, co ma pośrednio wpływ na kolejność wyboru kandydatów do wywłaszczenia. Następnie rozpoczyna się pętla, w której pierwsza ścieżka (z czoła listy $\boldsymbol{K}$) jest dodawana do zbioru kandydatów $\boldsymbol{W}$. Po każdym dodaniu nowego kandydata sprawdza się, czy pasmo wybranych dotąd kandydatów nie jest nadmiarowe, tzn. większe niż jest potrzebne do przyjęcia nowej ścieżki. Jeśli jest, wówczas przegląda się po kolei zbiór kandydatów $\boldsymbol{W}$ w poszukiwaniu ścieżek, których pasmo jest mniejsze niż aktualna nadwyżka (bilans) pasma. Jeśli znalezione zostaną ścieżki spełniające ten warunek, wówczas są usuwane z listy kandydatów (na schemacie dla uproszczenia pokazano wybór pojedynczej nadmiarowej ścieżki p). Pętla jest powtarzana tak długo, dopóki w zbiorze $\boldsymbol{K}$ znajdują się potencjalni kandydaci. Gdy zbiór $\boldsymbol{K}$ jest pusty, wtedy następuje sprawdzenie ostatecznego bilansu pasma i zakończenie procedury, przy czym warunkiem pozytywnego zakończenia procedury jest nieujemna wartość globalnego bilansu pasma S_g. Wartość ujemna oznacza brak możliwości przyjęcia nowej ścieżki z powodu braku dostępnego pasma.



Rys. 4.1. Schemat działań zasadniczej części algorytmu KNow.

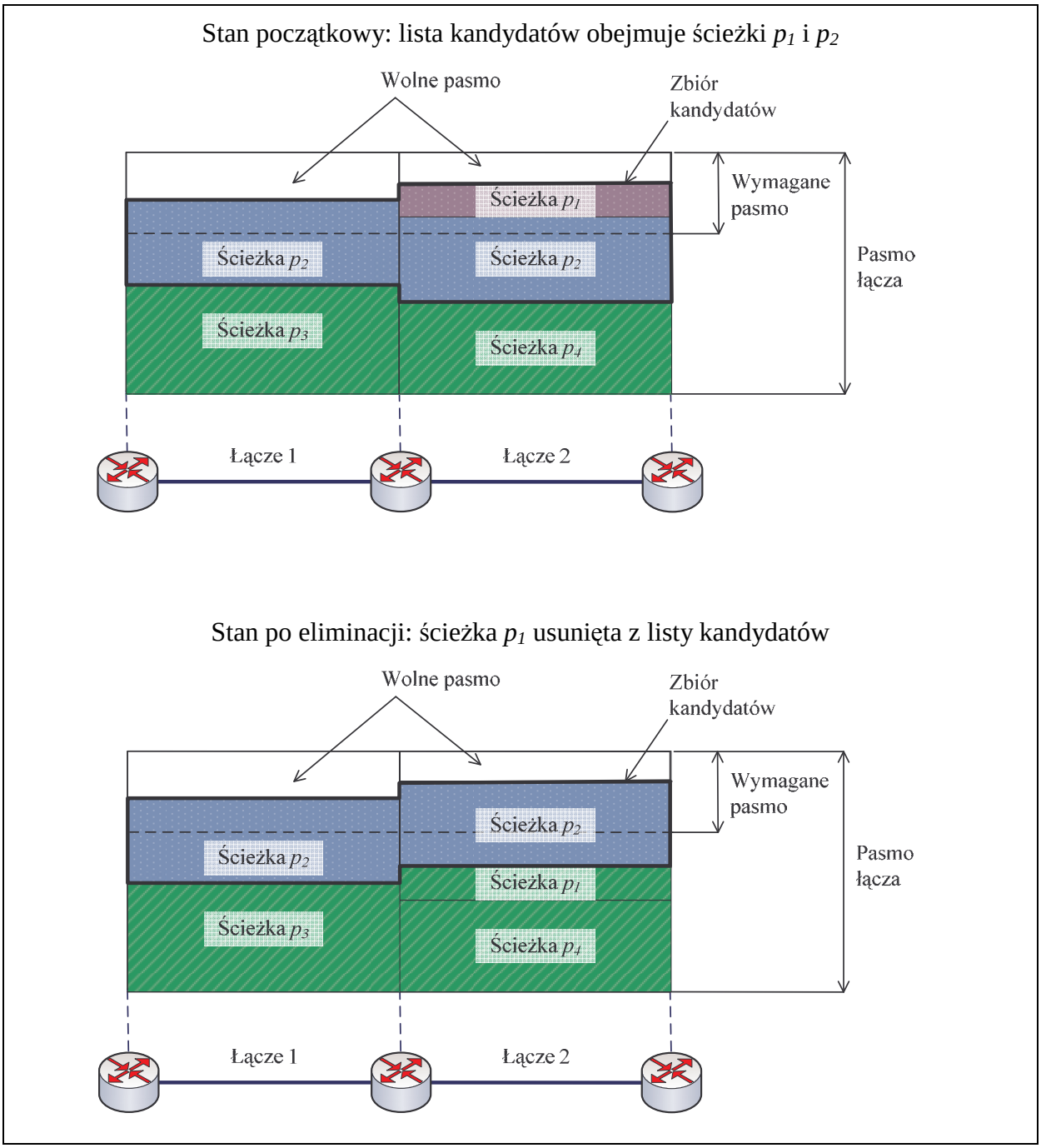
Kluczową dla algorytmu KNow jest procedura eliminacji, czyli usuwania nadmiarowych ścieżek z listy kandydatów. Jej celem jest sprawdzenie, czy nadmiar zgromadzonego pasma umożliwia usunięcie którejś ze ścieżek. W tym celu po każdym dodaniu kandydata do zbioru $\boldsymbol{W}$ na wszystkich ścieżkach z tego zbioru testowany jest warunek, który formalnie zapisano w postaci wzoru 4.1. Jego znaczenie jest następujące. Jeśli po dodaniu kandydata do zbioru $\boldsymbol{W}$ istnieje w tym zbiorze dowolna ścieżka p, dla której na każdym łączu e wymagającym wywłaszczania i należącym jednocześnie do trasy tej ścieżki R_p, aktualna wartość pasma straconego $B_x^{(e)}$ (równa bilansowi pasma S_e) przekracza lub jest równa pasmu b_p tej ścieżki, wówczas ścieżka zostaje usunięta z listy kandydatów.

$$\underset{p\subset \boldsymbol{W}}{\exists} \underset{e\in(\hat{R}\cap R_p)}{\forall} S_e \geq b_p \Rightarrow \boldsymbol{W} := \boldsymbol{W} \setminus \{p\} \tag{4.1}$$

Aby lepiej zrozumieć sens takiego sposobu działania algorytmu, na rys. 4.2 przedstawiony został stosowny przykład. Ścieżki p_1 i p_2 zostały kolejno dodane do zbioru kandydatów $\boldsymbol{W}$. Jest to stan początkowy, w którym następuje uruchomienie procedury eliminacji. Obliczono w niej, że bilans pasma, czyli wolne pasmo powiększone o sumę pasm kandydatów p_1 i p_2, jest na łączu 2 większe od pasma ścieżki p_1. Inaczej mówiąc, ścieżka p_1 powinna zostać usunięta z listy kandydatów, gdyż ścieżka p_2 samodzielnie zapewnia pasmo wystarczające dla przyjęcia nowej ścieżki. Zatem po zakończeniu eliminacji w zbiorze kandydatów pozostaje tylko ścieżka p_2, co jest zgodne z intuicyjnym wyborem.

W uzupełnieniu podanego opisu na rys. 4.3 pokazano uproszczoną postać pseudokodu dla tej części algorytmu, która jest odpowiedzialna za eliminację nadmiarowych kandydatów. W pętli (linie 1-11) sprawdzana jest każda ścieżka p ze zbioru kandydatów $\boldsymbol{W}$. Na każdym łączu wymagającym wywłaszczania (pętla for w liniach 2-6) sprawdzany jest warunek na zachowanie ścieżki w zbiorze kandydatów, tj. czy bilans pasma S_e ma mniejszą wartość niż pasmo ścieżki b_p (pod warunkiem, że łącze e należy do trasy R_p ścieżki). Jeśli na którymś łączu warunek jest spełniony, wówczas następuje skok do linii 10 w celu ominięcia procedury usuwania kandydata i zaczyna się sprawdzanie kolejnej ścieżki. Jeśli jednak odwrotny warunek z linii 3 nie jest spełniony dla wszystkich badanych łączy, wówczas ścieżka powinna być usunięta z listy kandydatów, co następuje w linii 8, po czym w linii 9 następuje uaktualnienie bilansu pasma S_e.

Należy zauważyć, że w procedurze eliminacji w takiej postaci założono, że każda ścieżka-kandydat przechodzi przez przynajmniej jedno łącze wymagające wywłaszczania. Gdyby tak nie było, wówczas warunek w linii 3 nie mógłby być spełniony i ścieżka byłaby eliminowana bez względu na bilans pasma. Taka sytuacja nie wystąpi, gdyż z założenia w zbiorze kandydatów mogą znajdować się tylko te ścieżki, które przechodzą przez przynajmniej jedno łącze wymagające wywłaszczania.



Rys. 4.2. Ilustracja zasady działania procedury eliminacji w algorytmie KNow: ścieżka p_1 dodana wcześniej do listy kandydatów cała mieści się poniżej linii wymaganego pasma, a zatem nie powinna być wywłaszczana, ale usunięta z listy kandydatów.

Algorytm w postaci pełnego pseudokodu został przedstawiony na rys. 4.4. Lista kandydatów ***W*** zostaje zainicjowana (linia 2) i wyznaczony zostaje zbiór ***K*** potencjalnych kandydatów (na całej długości drogi), a lista sortowana jest zgodnie z zadanym parametrem *variant* (linia 3). Następnie (linia 4) obliczony zostaje początkowy bilans pasma S_e równy dla każdego łącza e różnicy między wolnym pasmem $a_f^{(e)}$ a pasmem nowej ścieżki b_q (por. wzór 3.14):

$$S_e = a_f^{(e)} - b_q \,. \tag{4.2}$$

```
1    for each p in W do              # sprawdź wszystkich aktualnych kandydatów
2      for each e in R̂ do            # dla każdego łącza wymagającego wywłaszczania
3        if e ∈ R_p and S_e < b_p then      # warunek odwrotny, na brak eliminacji
4          goto check_next_candidate  # jeśli spełniony, sprawdź kolejnego kandydata
5        end if
6      end for
7      # eliminacja: ścieżka p jest nadmiarowa, zostanie usunięta z listy kandydatów
8      W := W \ {p}                   # usuń ścieżkę z listy kandydatów
9      update_bw_balance ()           # uaktualnij bilans pasma S_e
10   check_next_candidate:
11   end for
```

Rys. 4.3. Fragment algorytmu KNow odpowiedzialny za procedurę eliminacji.

Właściwa część algorytmu składa się z dwóch części. Pierwsza (linie 5-16), pomocnicza, jest uruchamiana tylko wtedy, gdy brakuje pasma tylko na jednym łączu (warunek w linii 5). Przeprowadza się wówczas poszukiwanie pojedynczego rozwiązania, tj. pojedynczej ścieżki o najmniejszym paśmie zapewniającym przyjęcie nowej ścieżki, o ile taka istnieje. Odbywa się to poprzez przejrzenie wszystkich potencjalnych kandydatów w pętli for (linie 9-15). W zmiennych *sgl_cand*, *sgl_bw* i *sgl_surplus* zapisuje się odpowiednio wybranego kandydata, jego pasmo i osiąganą wartość straconego pasma. W drugiej części algorytmu (linie 17-33), wykonywanej niezależnie od liczby łączy wymagających wywłaszczania, następuje charakterystyczne dla algorytmu poszukiwanie najlepszego zbioru ścieżek. Dla każdej ścieżki *p* należącej do zbioru potencjalnych kandydatów ***K*** wykonuje się procedurę składającą się z dwóch kroków.

1. Ścieżka zostaje dodana do listy kandydatów (linie 18-19).
2. Nadmiarowe ścieżki zostają usunięte z listy kandydatów (linie 20-28).

W pierwszym kroku ścieżka *p* zostaje dodana do listy kandydatów, co skutkuje zwiększeniem wartości bilansu pasma na łączach znajdujących się na trasie ścieżki *p*. Jeśli bilans na którymś z łączy osiągnął przez to poziom większy od zera, to tak skonstruowany zbiór kandydatów ***W*** spowoduje wywłaszczenie nadmiarowego pasma. To z kolei wskazuje, że zbiór kandydatów może zawierać zbyt wiele ścieżek i wyzwala krok drugi.

W drugim kroku następuje sprawdzenie każdej ścieżki *l* dodanej do tej pory do zbioru kandydatów. Jeśli pasmo ścieżki *l* jest mniejsze niż bilans pasma na każdym łączu wymagającym wywłaszczania (warunek w linii 23), wówczas ta ścieżka jest usuwana z listy kandydatów (linie 24-25).

Na zakończenie procedury następuje sprawdzenie, czy pasmo uzyskane w efekcie wywłaszczenia wybranych kandydatów jest wystarczające do przyjęcia nowej ścieżki (linie 30-33). Jedynym powodem niepowodzenia może tu być zbyt mały zbiór potencjalnych kandyda-

tów, tzn. suma pasm wszystkich potencjalnych kandydatów nie wystarcza do przyjęcia nowej ścieżki.

```
1   procedure algorithm_KNow (variant)
2     W := ∅                                  # inicjuj listę kandydatów
3     K := cand_preselect_on_route (variant)  # utwórz listę
                                              # potencjalnych kandydatów
4     calculate_bw_balance ()                 # oblicz początkowy bilans pasma
5     if |R̂| = 1 then                         # wywłaszczanie tylko na jednym łączu
6       # szukaj pojedynczego rozwiązania
7       sgl_surplus := sgl_bw := 0            # inicjalizuj metryki dla
8       sgl_cand := ∅                         # pojedynczego rozwiązania
9       for each p in K do                    # dla każdego potencjalnego kandydata
10        if b_p ≥ B_r^(e) and (sgl_cand = ∅ or b_p < sgl_bw) then
          # jeśli pasmo pojedynczej ścieżki jest wystarczające
          # i jest mniejsze od pasma najmniejszej zapamiętanej ścieżki
11          sgl_cand := p                   # zapamiętaj ścieżkę i jej parametry
12          sgl_bw := b_p
13          sgl_surplus := b_p - B_r^(e)
14        end if
15      end for
16    end if                                # koniec części dla pojedynczego łącza
      # POCZĄTEK PROCEDURY DODAWANIA I ELIMINACJI KANDYDATÓW
17    for each p in K do                    # dla każdego potencjalnego kandydata
18      W := W ∪ {p}                        # dodaj p bezwarunkowo do listy kandydatów
19      update_bw_balance ()                # uaktualnij bilans pasma
20      if surplus_generated () then        # występuje nadmiar zebranego pasma
21        # w pętli for eliminacja nadmiarowych ścieżek z listy kandydatów
22        for each l in W do                # sprawdź wszystkich aktualnych kandydatów
23          if path_under_surplus (l) then  # ten kandydat jest nadmiarowy
24            W := W \ {l}                  # usuń ścieżkę z listy kandydatów
25            update_bw_balance ()          # uaktualnij bilans pasma
26          end if
27        end for
28      end if
29    end for                               # wybór kandydatów zakończony
      # KONIEC PROCEDURY DODAWANIA I ELIMINACJI KANDYDATÓW
```

```
30    for each e in R̂ do          # sprawdź bilans na każdym łączu
31       if S_e < 0 then          # procedura zakończona niepowodzeniem
32          return ∅
33    end for
34    # procedura zakończona sukcesem
35    if sgl_cand ≠ ∅             # jeśli wystarcza pojedyncza ścieżka
36    and S_g > sgl_surplus then  # i zapewnia mniejszą nadwyżkę pasma
37       return sgl_cand          # zwróć pojedyncze rozwiązanie
38    else return W               # zwróć listę kandydatów
39 end
40
41 procedure cand_preselect_on_route (variant)
42    K := ∅                      # lista sortowana zgodnie z variant
43    for each p in P do          # dla każdej ścieżki
44       if h_p > s               # która może zostać wywłaszczona
45       and path_on_route (R_p, p) then
46          K := K ∪ {p}          # dodaj ścieżkę do listy
                                   # potencjalnych kandydatów
47 end
48
49 procedure calculate_bw_balance ()
50    # oblicz bilans początkowy
51    for each e in R̂ do          # na każdym łączu na wybranej drodze
52       S_e := a_f^(e) – b_q      # od wolnego pasma
                                   # odejmij pasmo nowej ścieżki
53       B_r^(e) := max (0, –S_e)  # brakujące pasmo to odwrotność nadmiaru
54    end for
55    # dodaj pasmo kandydatów
56    for each p in W do          # dla każdego kandydata
57       for each e in R̂ do       # na każdym łączu na wybranej drodze
58          S_e := S_e + b_p      # dodaj do bilansu pasmo ścieżki
59    # uaktualnij brakujące pasmo
60    for each e in R̂ do          # dla każdego łącza na wybranej drodze
61       B_r^(e) := max (0, –S_e)  # przelicz brakujące pasmo
62 end
63
64 procedure update_bw_balance (p)
```

```
65      for each e in R̂ do          # dla każdego łącza na wybranej drodze
66          S_e := S_e + β(p,e)b_p  # uaktualnij bilans pasma
67          B_r^(e) := max (0, –S_e)  # uaktualnij brakujące pasmo
68      end for
69  end
70
71  procedure surplus_generated ()
72      for each e in R̂ do          # dla każdego łącza na wybranej drodze
73          if S_e > 0 then         # znaleziono nadmiarowe pasmo
74              return TRUE
75      end for
76      return FALSE                # nie ma nadmiarowego pasma
77  end
78
79  procedure path_under_surplus (p)
80      for each e in R̂ do          # dla każdego łącza na wybranej drodze
81          if b_p ≥ S_e then       # ścieżka nie jest nadmiarowa
82              return FALSE
83      end for
84      return TRUE                 # ścieżka jest nadmiarowa na każdym łączu
85  end
```

Rys. 4.4. Algorytm KNow.

4.2. Analiza algorytmu

Porównanie istniejących metod heurystycznych w rozdziale trzecim zakończono stwierdzeniem, że nie dysponujemy algorytmem, który posiadałby następujące cztery pożądane cechy.

1. *Efektywność obliczeniowa*, umożliwiająca zastosowanie w rzeczywistych sieciach o dużej liczbie węzłów i ścieżek.
2. *Elastyczność konfiguracyjna*, umożliwiająca określenie przez użytkownika kryteriów wyboru kandydatów (patrz rozdz. 3.4.1).
3. *Zasięg obejmujący całą domenę*, aby wykorzystać znajomość topologii sieci i położenia ścieżek do dokonania wyboru możliwie najlepszego z punktu widzenia całej domeny.
4. *Jakość* przejawiająca się uzyskiwaniem dobrych wyników w porównaniu z innymi badanymi algorytmami heurystycznymi (patrz rozdz. 3.4.1).

W oparciu o te cechy zostanie przeprowadzona analiza zaproponowanego algorytmu.

4.2.1. Efektywność obliczeniowa

Efektywność obliczeniową charakteryzuje się poprzez badanie złożoności obliczeniowej. Złożoność wielomianowa (n, n^2, n^3, itd.) jest akceptowalna, natomiast wykładnicza (2^n) wyklucza zastosowanie algorytmu w praktyce. Dla zaproponowanego algorytmu wykonano podobną analizę, jaką zrobiono wcześniej dla omówionych algorytmów heurystycznych. Na rys. 4.5 zebrano te elementy kodu algorytmu KNow, które są istotne dla określenia złożoności, przy czym do kodu włączono pętle zawarte w wywoływanych procedurach.

```
1   for each p in K do              # A: dla każdego potencjalnego kandydata
2     # update_bw_balance ()
3     for each e in R̂ do            # B: dla każdego łącza na wybranej drodze
4     end for
5     # surplus_generated ()
6     for each e in R̂ do            # C: dla każdego łącza na wybranej drodze
7     end for
8     if … then
9       for each l in W do          # D: dla wszystkich aktualnych kandydatów
10        # path_under_surplus (l)
11        for each e in R̂ do        # E: dla każdego łącza na wybranej drodze
12        end for
13        if … then
14          # update_bw_balance ()
15          for each e in R̂ do      # F: dla każdego łącza na wybranej drodze
16          end for
17        end if
18      end for
19    end if
20  end for
```

Rys. 4.5. Analiza złożoności obliczeniowej algorytmu KNow, krok 1/3.

Algorytm składa się więc aż z sześciu pętli i posiada cztery poziomy zagnieżdżenia, choć możliwe jest tu wykonanie kilku redukcji. Po pierwsze, pętla *B* wykonuje się w szeregu z pętlą *C*, więc na potrzeby wyliczeń złożoności obliczeniowej może być pominięta. Po drugie, pętla *F* również wykonuje się w szeregu z pętlą *E*, co także czyni ją zbędną w dalszej analizie. Usuwając także nieistotne warunki *if* w liniach 8 i 13 otrzymujemy postać pokazaną na rys. 4.6.

```
1   for each p in K do          # A: dla każdego potencjalnego kandydata
2     for each e in R̂ do        # C: dla każdego łącza na wybranej drodze
3     end for
4     for each l in W do        # D: dla wszystkich aktualnych kandydatów
5       for each e in R̂ do      # E: dla każdego łącza na wybranej drodze
6       end for
7     end for
8   end for
```

Rys. 4.6. Analiza złożoności obliczeniowej algorytmu KNow, krok 2/3.

Można teraz zauważyć, że pętla *C* jest wykonywana w szeregu z pętlami *D* i *E*, zatem po jej pominięciu otrzymujemy ostateczną postać przedstawioną na rys. 4.7.

```
1   for each p in K do          # A: dla każdego potencjalnego kandydata
2     for each l in W do        # D: dla wszystkich aktualnych kandydatów
3       for each e in R̂ do      # E: dla każdego łącza na wybranej drodze
4       end for
5     end for
6   end for
```

Rys. 4.7. Analiza złożoności obliczeniowej algorytmu KNow, krok 3/3.

Stąd otrzymujemy następującą liczbę powtórzeń pętli: (liczba kandydatów)^2 · (liczba łączy z wywłaszczaniem). To daje złożoność obliczeniową $\boldsymbol{O(NZ^2)}$, identyczną jak dla większości omówionych wcześniej algorytmów, a więc jak najbardziej akceptowalną.

Analizując średnią liczbę przebiegów w realnych sytuacjach, jest ona równa iloczynowi liczby potencjalnych kandydatów, średniej liczby kandydatów i liczby łączy z wywłaszczaniem. Jeśli weźmiemy pod uwagę to, że w praktyce zwykle średnia liczba łączy z wywłaszczaniem nie przekracza 2 [46], wówczas można ten czynnik pominąć. Z kolei liczba kandydatów do wywłaszczenia zwykle nie przekracza kilku ścieżek i w niewielkim stopniu zależy od warunków badań. Możemy więc podsumować, że złożoność obliczeniowa jest proporcjonalna do liczby potencjalnych kandydatów. Pozwala to z wysokim prawdopodobieństwem uznać algorytm za efektywny obliczeniowo, a przez to możliwy do zastosowania w praktycznych realizacjach.

4.2.2. Elastyczność konfiguracyjna

Elastyczność konfiguracyjna w przypadku proponowanego algorytmu wyraża się poprzez możliwość zmiany sposobu sortowania zbioru ***K*** potencjalnych kandydatów, zaimplemento-

wanego w postaci listy. Nie wymaga to znaczącej rozbudowy implementacji metody, a jedynie wprowadzenia wariantowości w funkcji dodającej ścieżkę do zbioru ***K***. Ogólna zasada jest taka, że typy ścieżek preferowanych do wyboru jako kandydatów powinny być umieszczane na końcu listy. Przykładowo, w celu minimalizacji liczby wywłaszczeń należy wybierać ścieżki duże, a więc sortować zbiór ***K*** według pasma rosnąco. Z kolei aby zminimalizować stracone pasmo należy preferować wybór małych ścieżek, a więc sortować zbiór ***K*** według pasma malejąco. Natomiast posortowanie zbioru ***K*** według wartości priorytetu rosnąco spowoduje, że preferowane będzie wywłaszczanie ścieżek o wysokim liczbowo, czyli niskim priorytecie.

4.2.3. Zasięg algorytmu

Trzecim pożądanym kryterium dobrego algorytmu jest zasięg obejmujący domenę. Proponowany algorytm z założenia posiada tę cechę, gdyż w procesie wyboru ścieżek analizuje stan na całej drodze połączeniowej, w ramach zarządzanej domeny. Dzięki temu w sytuacji gdy wywłaszczenie następuje na więcej niż jednym łączu, możliwe jest osiągnięcie lepszego rezultatu niż przy użyciu metody lokalnej.

4.2.4. Jakość algorytmu

Ostatnią, choć kluczową, cechą algorytmu jest jakość, rozumiana (zgodnie z definicją w rozdz. 3.4.1) jako zdolność do wyboru takich zbiorów kandydatów ***W***, dla których średni koszt wywłaszczenia będzie niski. Na tym etapie jednak pozostawimy tę cechę bez oceny, gdyż trudno jest w przypadku tego algorytmu dokonać teoretycznej analizy jakościowej. Bardziej uzasadnione jest przeprowadzenie badań symulacyjnych poszczególnych algorytmów i porównanie ze sobą otrzymanych wyników. Temu poświęcone zostały dwa kolejne rozdziały pracy, w których opisano warunki przeprowadzonych badań a następnie zaprezentowano i omówiono wyniki.

5. ŚRODOWISKO BADAWCZE

W poprzednich rozdziałach omówiono szereg algorytmów wywłaszczania i przeprowadzono ich wstępną charakterystykę. Jej uzupełnienie obejmujące analizę jakości algorytmów musi być oparte na wiarygodnych badaniach porównawczych. W niniejszym rozdziale omówiono cechy algorytmów, jakie można badać, zastosowane metody badawcze, oraz narzędzia, jakich w tym celu użyto.

5.1. Badane algorytmy i miary ich oceny

Do celów badawczych wybrano opisane wcześniej algorytmy heurystyczne. Dodatkowo dla metod GarGop i OliSco wybrano po dwa warianty, dla minimalizacji liczby wywłaszczeń i dla minimalizacji wywłaszczonego pasma, oznaczone przyrostkami odpowiednio /RC (ang. *Relocation Count*) i /BW (ang. *Bandwidth*). W ten sposób otrzymano siedem algorytmów bądź ich wariantów.

1. **GarGop/RC**: algorytm GarGop dla minimalizacji liczby wywłaszczeń (rys. 3.5).
2. **GarGop/BW**: algorytm GarGop dla minimalizacji wywłaszczonego pasma (rys. 3.6).
3. **Pey**: algorytm Pey z pojedynczą pętlą (rys. 3.7, 3.8).
4. **OliSco/RC**: algorytm OliSco (rys. 3.9) dla minimalizacji liczby wywłaszczeń (wzór 3.41: X_1=0, X_2=1, X_3=0, X_4=0).
5. **OliSco/BW**: algorytm OliSco (rys. 3.9) dla minimalizacji wywłaszczonego pasma (wzór 3.41: X_1=0, X_2=0, X_3=1, X_4=0).
6. **BlaMeL**: algorytm BlaMeL (rys. 3.10).
7. **KNow**: algorytm KNow (rys. 4.4) z listą kandydatów sortowaną według pasma, malejąco.

Jest wiele różnych kryteriów, według których dokonuje się oceny jakościowej algorytmów wywłaszczania, ale najważniejszymi są liczba wywłaszczeń i pasmo wywłaszczonych ścieżek. W praktyce stosuje się różne modyfikacje tych wielkości, takie jak:

- średnia liczba wywłaszczeń $\overline{M}$,
- średnie stracone pasmo $\overline{B_x}$,
- średnie stracone pasmo sieciowe $\overline{B_{xnet}}$,
- średnia wartość metryki złożonej $\overline{Q}$, która uwzględnia jednocześnie liczbę wywłaszczeń i stracone pasmo sieciowe, (zdefiniowanej wzorem 5.1),
- średnie wywłaszczone pasmo $\overline{B}$,
- średnie wywłaszczone pasmo sieciowe $\overline{B_{net}}$,
- średnie względne pasmo stracone $\overline{b_x}$,
- średnie względne pasmo stracone sieciowe $\overline{b_{xnet}}$.

Są jeszcze inne wielkości, jakie można mierzyć, choć zwykle należy je traktować jako wielkości obrazujące skalę zmian powodowanych w sieci poprzez wywłaszczenia, a nie jako

cechy określonych metod, np.:

- średni poziom $\overline{b_b}$ rezerwacji pasma na poszczególnych łączach,
- średnia długość $\overline{g}$ kaskady wywłaszczeń,
- średnia liczba $\overline{H}$ ścieżek w kaskadzie wywłaszczeń,
- prawdopodobieństwo p_{pre} wywłaszczenia, tj. prawdopodobieństwo spowodowania wywłaszczeń przez nowo tworzoną ścieżkę (5.2),
- prawdopodobieństwo p_{rej} odrzucenia żądania utworzenia ścieżki, spowodowanej brakiem zasobów (5.3).

Niektóre z wymienionych wielkości nie zostały wcześniej zdefiniowane a inne wymagają tutaj dodatkowego komentarza.

Średnia liczba wywłaszczeń $\overline{M}$ przekłada się na liczbę operacji, jakie należy wykonać, aby przyjąć nową ścieżkę. Zwykle zależy nam na minimalizacji tej wielkości, gdyż każda wywłaszczona ścieżka to przerwa w ruchu odczuwalna przez klienta. Ponadto każdą wywłaszczoną ścieżkę należy ponownie utworzyć, co oznacza zwiększenie ruchu sygnalizacyjnego w sieci i dodatkowe obciążenie ruterów. Cechą dobrego algorytmu jest więc dążenie do minimalizacji tej wielkości.

Średnia wartość straconego pasma $\overline{B_x}$ ma również niebagatelne znaczenie. Im większe ścieżki zostają wywłaszczone, tym potencjalnie większy ruch zostaje zakłócony, a co za tym idzie, straty dla operatora są większe. Zależy nam zatem, aby algorytm wywłaszczania w ten sposób wybierał ścieżki, aby nie zwalniać dużo więcej pasma, niż jest rzeczywiście potrzebne dla przyjęcia nowej ścieżki. Oczywiste jest, że strat pasma nie da się zupełnie wyeliminować, ale stosunek pasma wywłaszczonego do wymaganego powinien być możliwie mały.

Oba omówione parametry są dobrymi wyznacznikami jakości metody, gdyż oddają bezpośrednio to, czego od metod oczekujemy. Należy jednak pamiętać, że w praktyce ważniejsze niż konkretne wartości tych parametrów są różnice w wynikach osiąganych przez poszczególne algorytmy, które zostały zmierzone przy zachowaniu identycznych warunków pomiarowych i dla tych samych sieci.

Pojawia się pytanie, czy możliwe jest jednoczesne utrzymanie na niskim poziomie obu wielkości, tj. liczby wywłaszczeń i straconego pasma. W celu zbadania tego zagadnienia zdefiniowano złożoną metrykę Q, która uwzględnia zarówno liczbę wywłaszczeń jak też wywłaszczone pasmo sieciowe. Przykładową prostą definicję przedstawiono w postaci wzoru (5.1). Tak skonstruowana metryka Q zawiera się w przedziale (0;1], przy czym im algorytm jest lepszy w zakresie liczby wywłaszczeń i wywłaszczonego pasma, tym wyższa wartość metryki.

$$Q = \frac{B_{rnet}}{MB_{xnet}} \tag{5.1}$$

Komentarza wymagają także wielkości opisujące wywłaszczone pasmo sieciowe B_{net} i stracone pasmo sieciowe B_{xnet}. Wychodzi się tu z założenia, że wywłaszczenie dłuższej ścieżki jest bardziej szkodliwe niż wywłaszczenie ścieżki krótszej, gdyż zajmuje więcej czasu i wymaga zaangażowania większej liczby ruterów. Aby to uwzględnić, mnoży się pasma wywłaszczonych ścieżek przez ich długości. Ten parametr jest często pomijany przy omawianiu metod o zasięgu lokalnym, gdyż te nie biorą pod uwagę długości wywłaszczonych ścieżek, a przez to nie mają wpływu na jego wartość.

Prawdopodobieństwo p_{pre} spowodowania wywłaszczenia określa się jako stosunek liczby k_{crpr} utworzeń ścieżek, które kończą się spowodowaniem wywłaszczenia, do ogólnej liczby k_{cr} utworzonych ścieżek.

$$p_{pre} = \frac{k_{crpr}}{k_{cr}} \tag{5.1}$$

Prawdopodobieństwo p_{rej} odrzucenia żądania utworzenia ścieżki to stosunek liczby k_{rej} odrzuconych żądań do ogólnej liczby k_{req} żądań utworzenia ścieżek.

$$p_{rej} = \frac{k_{rej}}{k_{req}} \tag{5.1}$$

5.2. Założenia do modelu symulacyjnego

Mając zdefiniowane kryteria oceny, można określić założenia dla środowiska, w jakim zostaną przeprowadzone badania algorytmów. Z uwagi na brak dostępu do środowiska badawczego opartego na rzeczywistych sieciach MPLS, weryfikację algorytmów przeprowadzono w oparciu o badania symulacyjne. Cechy, jakimi powinien charakteryzować się model symulacyjny to:

- budowa modułowa dla zapewnienia łatwego jego rozszerzania o nowe funkcje i algorytmy,
- możliwość badania dowolnej struktury sieci o wielkości przynajmniej 100 węzłów,
- wejście (opis sieci i parametry badań) oraz wyjście (wyniki) w postaci plików tekstowych,
- wyniki nie powinny wymagać dalszego złożonego przetwarzania statystycznego.

Najważniejszym z przyjętych założeń jest otwartość implementacji, przejawiająca się łatwym rozszerzaniem modelu o nowe funkcjonalności. W szczególności chodzi o to, aby móc stosunkowo szybko przeprowadzić badania nowych algorytmów lub rozszerzyć możliwości badawcze o nowe rodzaje pomiarów. Implementacja powinna również umożliwiać, po niezbędnych rozszerzeniach, prowadzenie badań nad innymi mechanizmami inżynierii ruchu, np.

protokołów dynamicznego rutingu. Dzięki temu wysiłek poniesiony na implementację zaowocuje w postaci uniwersalnego narzędzia, które będzie mogło być rozwijane i służyć badaniom wielu różnych aspektów sieci MPLS.

Zmiana struktury badanej sieci musi odbywać się poza samym narzędziem, w oparciu o plik konfiguracyjny. Program powinien w oparciu o te dane utworzyć i zasymulować strukturę sieci składającą się z zadanej liczby węzłów i źródeł, połączonych w określoną strukturę łączami o zadanych parametrach, obejmujących szybkość i opóźnienie.

Wejście i wyjście programu powinno być oparte na plikach tekstowych. Plik wejściowy powinien opisywać topologię sieci oraz parametry symulacji. Powinien podlegać kontroli składni i przynajmniej podstawowemu sprawdzeniu spójności zdefiniowanej sieci, aby zapewnić wczesne wykrycie błędów w opisach sieci. Wyniki pomiarów w postaci raportów o ujednoliconej strukturze powinny zostać skierowane do określonych w konfiguracji plików tekstowych.

Wyniki pomiarów powinny zawierać wartości średnie uzupełnione o przedziały ufności, które są niezbędne do oceny jakości uzyskanych wyników [39,61]. Aby to zapewnić, symulacja musi się składać z szeregu odcinków pomiarowych, po zakończeniu których zebrane zostaną wyniki częściowe. Program powinien dokonać ich automatycznego przetworzenia w celu wyznaczenia wartości średnich i przedziałów ufności. Liczba i długość odcinków pomiarowych oraz długość niezbędnego odcinka rozbiegowego powinny być określone w pliku wejściowym.

5.3. Realizacja modelu symulacyjnego

Mając dane założenia należało zdecydować się na wybór narzędzia symulacyjnego. Możliwy był wybór jednego z ogólnodostępnych symulatorów, np. *ns2* [103] lub *OMNeT++* [86]. Alternatywą było opracowanie własnego rozwiązania. Obie możliwości mają swoje zalety i wady. W przypadku wyboru istniejącej aplikacji ma się do dyspozycji gotowy produkt testowany przez dużą społeczność użytkowników. Z kolei własna implementacja umożliwia utworzenie środowiska spełniającego wszystkie postawione wymagania i łatwą do rozbudowy, a ponadto zapewniającą pełną kontrolę nad implementacją. Po rozważeniu dostępnych możliwości, zdecydowano się na zrealizowanie własnego symulatora sieci MPLS. Ważnymi argumentami przy podjęciu takiej decyzji były brak algorytmów wywłaszczania w najpopularniejszych projektach oraz wcześniejszy sukces implementacji symulatora algorytmów sterowania przyjęciem zgłoszeń dla sieci ATM [48,68].

Powstały program o nazwie *msim* jest rozwojowym symulatorem sieci MPLS, umożliwiającym z założenia przeprowadzenie badań różnych mechanizmów inżynierii ruchu. Umożliwia on zarówno prowadzenie szczegółowych symulacji na poziomie pakietów, jak też symulacji długookresowych zjawisk na poziomie zgłoszeń. Dzięki modułowej architekturze i wykorzystaniu mechanizmów obiektowości i dziedziczenia łatwo można go rozszerzyć

o nowe funkcjonalności. Symulator, napisany w języku C++, został uruchomiony i przetestowany w systemach operacyjnych Microsoft Windows XP oraz Linux (dystrybucja Debian 5 „lenny”), z wykorzystaniem kompilatorów Borland C++ Builder oraz GNU gcc. Program posiada uproszczony interfejs użytkownika w postaci konsoli tekstowej, na którą kierowane są informacje o postępie symulacji i dodatkowe komunikaty, w tym informacje o wykrytych błędach (rys. 5.1).

```
MPLS Network Simulator, ver. 0.4.8, Krzysztof Nowak, 2003-2010
Simulation definition file used: a06_050_176.cfg
Preparing simulation environment.
 *** Service menu activated. Press [Esc] to open. ***
Environment created, ready to simulate.
Pass 1/8 [None]     >....>....>
Pass 2/8 [GarGop/RC]>....>....>
Pass 3/8 [GarGop/BW]>....>....>
Pass 4/8 [Pey]      >....>....>
Pass 5/8 [OliSco/RC]>....>....>
Pass 6/8 [OliSco/BW]>....>....>
Pass 7/8 [BlaMeL]   >....>....>
Pass 8/8 [KNow]     >....>....>
Finished, no problems encountered.
```

Rys. 5.1. Typowe informacje kierowane na konsolę przez symulator *msim*.

Uzupełnieniem symulatora jest program o nazwie *Vims* z graficznym interfejsem użytkownika wspomagającym projektowanie badanych sieci i umożliwiającym automatyczne generowanie na ich podstawie plików konfiguracyjnych. Aplikacja działa w systemie operacyjnym Microsoft Windows XP i posiada interfejs okienkowy z możliwością pracy na wielu mapach sieci jednocześnie. Oferuje ona dodatkowo funkcję analizy statystycznej sieci, obejmującej rozkłady gęstości połączeń i długości najkrótszych tras (rys. 5.2).

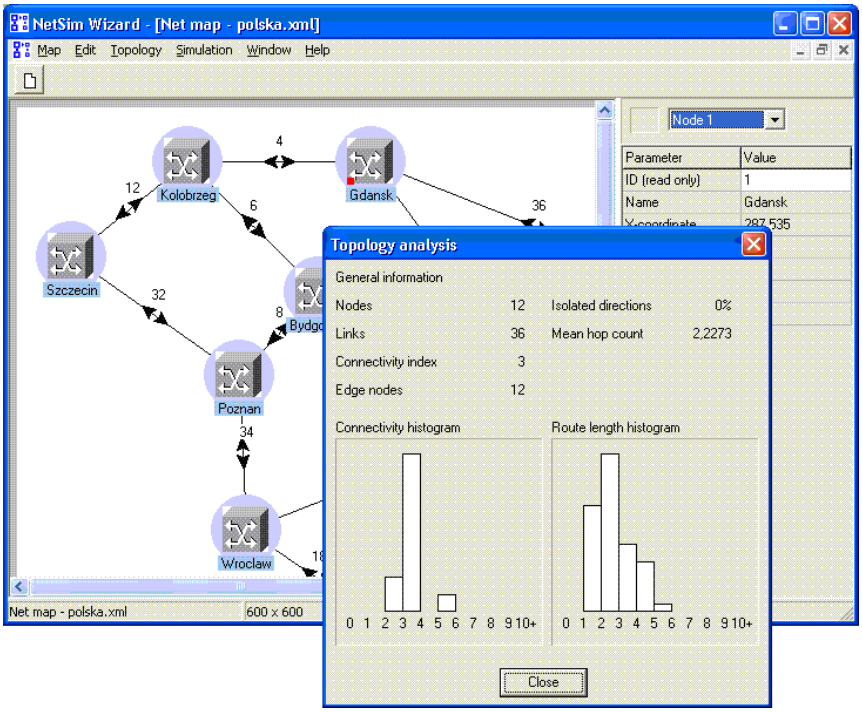
5.3.1. Zasada działania

Program przeprowadza symulację zdarzeniową z czasem dyskretnym (ang. *descret event simulation*). Symulacja składa się odcinka rozbiegowego i zadanej przez użytkownika liczby odcinków symulacyjnych. Wyniki pomiarów po każdym odcinku symulacyjnym są zapamiętane i po wykonaniu ostatniego odcinka przeliczone w celu uzyskania wartości średniej i wariancji, co jest niezbędne do określenia przedziałów ufności uzyskanych wyników.

Struktura blokowo-funkcjonalna symulatora została przestawiona na rys. 5.3. Najważniejsze obiekty wchodzące w skład programu to:

- *menedżer symulacji*, odpowiadający za właściwy przebieg procesu symulacji,
- *menedżer kolejki zdarzeń*, odpowiadający za przechowywanie i dystrybucję zdarzeń,
- *menedżer konfiguracji*, przechowujący informacje o sieci i odpowiadający za jej tworzenie,

- *menedżer połączeń*, zarządzający bazą danych o ścieżkach,
- *generator raportów*, przechowujący i przetwarzający statystyki,
- *menedżer topologii*, zajmujący się generowaniem żądań tworzenia ścieżek w czasie trwania symulacji.

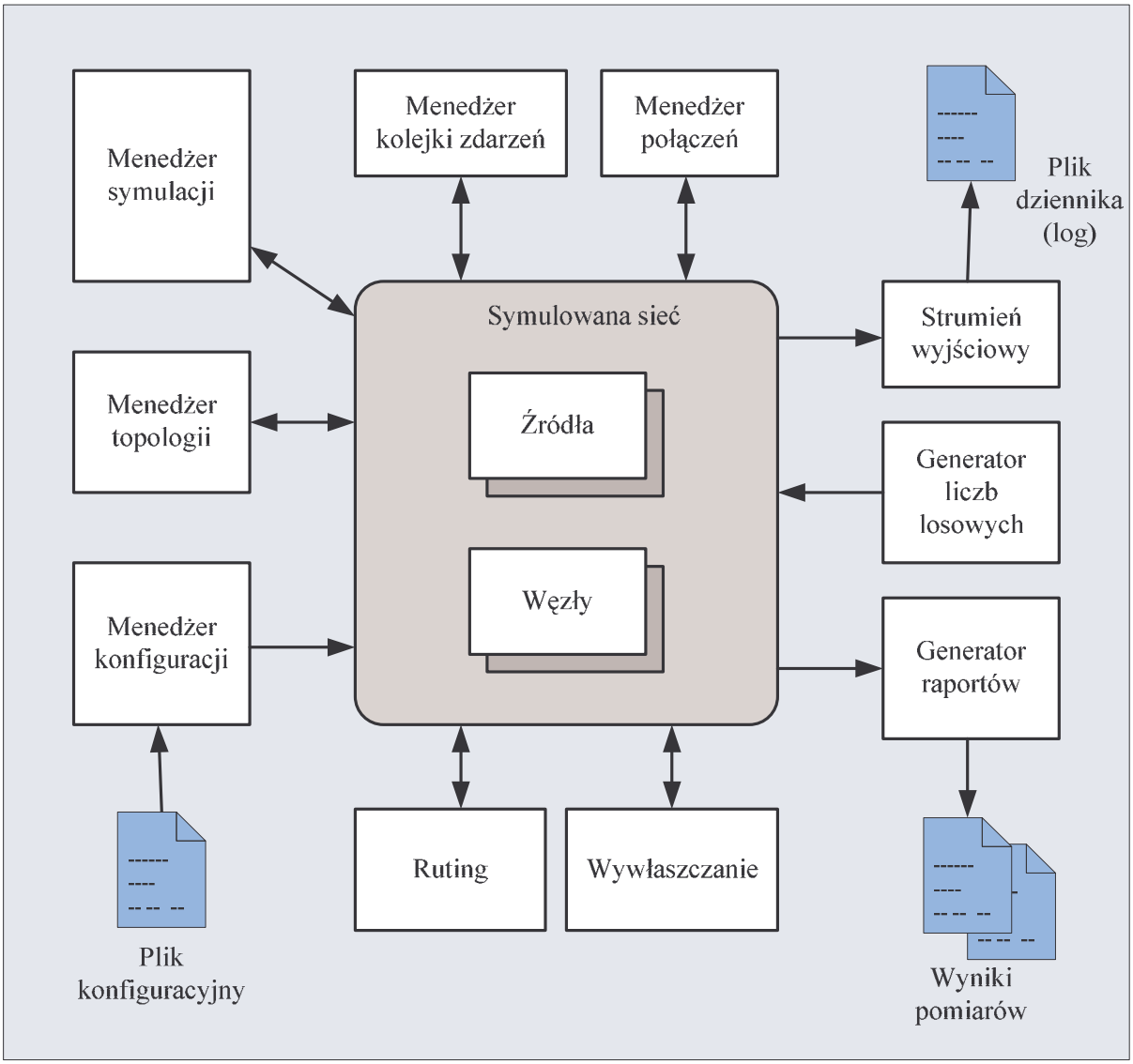


Rys. 5.2. Aplikacja *Vims* wspomagająca tworzenie plików konfiguracyjnych.

Menedżer symulacji jest najważniejszym obiektem programu. Odpowiada za utworzenie większości pozostałych obiektów sterujących, w tym menedżera kolejki zdarzeń oraz menedżera konfiguracji. Interpretuje parametry wywołania (z linii poleceń systemu operacyjnego) i na ich podstawie inicjuje pozostałe obiekty. W czasie symulacji odpowiada za podział czasu symulacji na odcinki. Koordynuje też automatyczne powtórzenia symulacji.

Menedżer kolejki zdarzeń przechowuje zdarzenia generowane przez obiekty sieci i obiekty sterujące. Koordynuje również dostarczanie zarówno zdarzeń kierowanych do poszczególnych obiektów jak też zdarzeń rozgłoszeniowych, kierowanych do wszystkich obiektów symulacji. Do jego najważniejszych zadań należy też sterowanie bieżącym czasem symulacji.

Menedżer konfiguracji odczytuje plik wejściowy, który zawiera opis warunków symulacji, w tym topologię sieci, warunki ruchowe, typ i czas trwania symulacji oraz rodzaj generowanych wyników pomiarów. Wszystkie odczytane dane zostają wczytane do pamięci i udostępnione w postaci struktur danych innym obiektom symulatora. Najważniejszym jego zadaniem jest utworzenie symulowanej sieci i takie powiązanie logiczne obiektów, aby tworzyły kompletną strukturę węzłów, źródeł i łączy, zdolną do przeprowadzenia symulacji.



Rys. 5.3. Schemat blokowo-funkcjonalny symulatora *msim*.

Menedżer połączeń zajmuje się tworzeniem i kasowaniem ścieżek, w czym współpracuje z blokami rutingu i wywłaszczania. Przechowuje też informacje o wszystkich ścieżkach utworzonych w sieci.

Generator raportów przechowuje w czasie trwania symulacji statystyki generowane przez symulowane obiekty. Po każdym odcinku pomiarowym przelicza je, a po zakończeniu symulacji tworzy raporty z wynikami pomiarów. W przypadku, gdy symulacja składa się z kilku powtórzeń, odpowiednio łączy wyniki z poszczególnych powtórzeń, aby utworzyć zbiorcze raporty.

Menedżer topologii zajmuje się tworzeniem i kasowaniem obiektów symulowanej sieci w czasie trwania symulacji. Poprzez kreowanie nowych źródeł powoduje generowanie żądań utworzenia ścieżek i w ten sposób umożliwia prowadzenie masowych badań algorytmów wywłaszczania. W przeciwieństwie do pozostałych obiektów sterujących jest tworzony przez

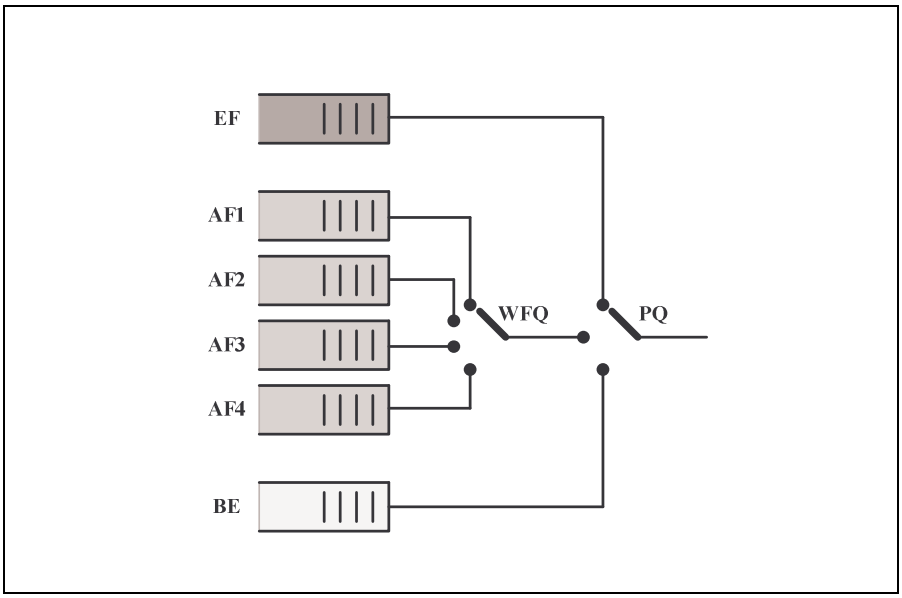
menedżera konfiguracji, a nie menedżera symulacji, gdyż jest blokiem opcjonalnym, tworzonym tylko w razie obecności odpowiedniego parametru w pliku konfiguracyjnym.

5.3.2. Symulowana sieć

Symulowana sieć jest modelem rzeczywistej sieci IP/MPLS, na którym testowane są mechanizmy inżynierii ruchu. W programie symulacyjnym sieć składa się z dowolnej (choć ograniczonej z góry) liczby następujących obiektów:

- węzeł, który jest odpowiednikiem rutera IP/MPLS,
- źródło, odpowiadające strumieniowi pakietów wprowadzanych do sieci IP/MPLS,
- łącze, będące modelem łącza fizycznego lub logicznego (tunelu) typu punkt-punkt.

Węzeł jest modelem nieskończenie szybkiego rutera IP/MPLS z buforami na wyjściu. Każdemu łączu wyjściowemu przydzielony jest konfigurowalny system buforowy, którym może być pojedyncza kolejka FIFO (ang. *First In First Out*) lub zestaw oddzielnych buforów dla każdej klasy ruchu z przełącznikami typu WFQ (ang. *Weighted Fair Queuing*) oraz PQ (ang. *Priority Queuing*), tak jak to pokazano na rys. 5.4.



Rys. 5.4. Typowy system buforowy wykorzystywany
w symulacjach na poziomie pakietów.

Węzeł definiuje się poprzez jego zadeklarowanie w pliku konfiguracyjnym i przydzielenie mu interfejsów sieciowych. Przykładowa definicja wygląda następująco.

```
node: id=1, name=Gdansk, x=287.535, y=32, size=1, igrp=1, ogrp=1;
addport: node=1, port=1, link=1, bufset=1;
addport: node=1, port=2, link=3, bufset=1;
addport: node=1, port=3, link=35, bufset=1;
```

Oprócz identyfikatora (*id*) oraz opisu (*name*) zdefiniowane zostały opcjonalne współrzędne (*x* i *y*), a także względny rozmiar (*size*) węzła oraz jego przyporządkowanie do grup wejściowych (*igrp*) i wyjściowych (*ogrp*). Te ostatnie parametry (*igrp*, *ogrp*) są używane przez menedżera topologii i w tym przypadku oznaczają, że w danym węźle mogą się zarówno rozpoczynać, jak i kończyć ścieżki generowane losowo. Do węzła dołączono trzy porty (interfejsy), skojarzone z łączami (*link*) o identyfikatorach 1, 3 i 35 (zdefiniowanymi oddzielnie). Wszystkie interfejsy posiadają identyczny system buforowy (*bufset*) o identyfikatorze 1 (zdefiniowany oddzielnie).

Źródła pełnią rolę generatorów zagregowanych strumieni ruchu. Ich działanie zależy od zdefiniowanego w pliku konfiguracyjnym poziomu symulacji. Dla poziomu zgłoszeń aktywność źródeł ogranicza się do wygenerowania żądania przydziału ścieżki od węzła dostępowego do innego węzła w sieci. Parametry żądania obejmują identyfikator jednego z predefiniowanych typów źródeł i związaną z nim klasę ruchu, co umożliwia określenie wymaganego pasma i warunków jakościowych QoS, w tym dopuszczalne prawdopodobieństwo strat pakietów oraz maksymalne opóźnienie. Nawet jeśli badania prowadzone są tylko na poziomie zgłoszeń, wymagane parametry QoS mogą być używane przez blok rutingu do wyznaczenia trasy spełniającej zadeklarowane warunki.

W przypadku, gdy symulacja odbywa się na poziomie pakietów, wtedy źródło oprócz wygenerowania żądania przydzielenia ścieżki tworzy i wysyła do węzła dostępowego pakiety IP o określonych rozkładach długości i rozkładach ich generacji.

```
source: id=1, type=1, link=22, destip=10.0.0.4, gcnt=0;
srctype: id=1, cos=1, mpl=100, ldt=none,
mspd=8e6, spddt=none, intens=100, sgend=expon, sspdd=expon;
cos: id=1, tos=ef, loss=1e-6, delay=0.05, setup=1;
```

W podanym przykładzie zdefiniowano źródło skojarzone z typem (*type*) 1, dołączone do łącza (*link*) 22, generujące pakiety do adresu IP (*destip*) 10.0.0.4. Dodatkowo pojawił się parametr określający ile pakietów ma być wygenerowanych za jednym razem (*gcnt*). Jeśli ten parametr ma wartość 0, to źródło działa tylko na poziomie zgłoszeń, tzn. jego aktywność ogranicza się do wygenerowania żądania przydzielenia ścieżki. Skojarzony ze źródłem typ (*srctype*) zawiera numer klasy (*cos*), średnią długość pakietu (*mpl*), średnią szybkość generowanego strumienia (*mspd*) wyrażoną w bitach na sekundę i odpowiadające obu wielkościom typy rozkładów (*ldt* i *spddt*). Pozostałe parametry są używane przez menedżera topologii do określenia częstości tworzenia źródeł danego typu. Klasa ruchu skojarzona z tym typem zawiera między innymi definicje parametrów QoS, tj. dopuszczalne straty pakietów (*loss*) oraz dopuszczalne opóźnienie (*delay*) na całej długości ścieżki, a poza tym priorytet utworzenia ścieżki (*setup*). Priorytet utrzymania (*holding*) nie został zdefiniowany, a zatem jest w tym przykładzie określony niejawnie jako równy wartości priorytetu utworzenia.

Łącze w programie symulacyjnym jest obiektem statycznym, tzn. nie generuje żadnych zdarzeń a jest jedynie strukturą danych, wykorzystywaną przez obiekty węzłów i źródeł do poznania swojego sąsiedztwa oraz do określenia szybkości transferu danych i opóźnienia propagacyjnego. Przykładowa definicja łącza wygląda następująco.

```
link: id=1, name=Gda_War, bw=1.55e+08, pt=0.001246, outn=11, outp=1;
```

Definicja łącza obejmuje identyfikator (*id*), opis (*name*), pasmo (*bw*) wyrażone w bitach na sekundę, czas propagacji (*pt*), oraz numery węzła (*outn*) i portu (*outp*) po drugiej stronie łącza. Zdefiniowane w ten sposób łącza są jednokierunkowe, co wynika z założeń opisujących architekturę MPLS [96]. W razie potrzeby definiuje się dodatkowe łącze skierowane w przeciwną stronę.

5.3.3. Przygotowanie badań

Przygotowanie i przeprowadzenie badań wywłaszczania z użyciem symulatora składa się zwykle z następujących etapów.

1. Zdefiniowanie scenariusza badań.
2. Utworzenie pliku konfiguracyjnego.
3. Uruchomienie serii symulacji.
4. Przetwarzanie i interpretacja wyników.

W każdym z tak zdefiniowanych etapów użytkownik jest wspomagany przez aplikacje *msim* i *Vims*. W dalszej części opisano dokładniej rolę obu programów w poszczególnych etapach.

Zdefiniowanie scenariusza badań polega między innymi na zbudowaniu topologii sieci, określeniu rodzaju pomiarów, wybraniu badanych algorytmów, określeniu wystarczającego czasu trwania symulacji. Na tym etapie aplikacja wspomagająca *Vims* umożliwia przygotowanie topologii w jednym z trzech trybów:

- ręczny,
- automatyczny,
- poprzez import i konwersję z innego formatu.

Ręczne tworzenie sieci polega na umieszczeniu na planszy węzłów oraz źródeł i połączeniu zdefiniowanych obiektów za pomocą łączy o określonych parametrach. Jest to najbardziej czasochłonny sposób, ale umożliwiający utworzenie dowolnej topologii. Sposób automatyczny polega na wygenerowaniu losowej topologii używając jednej z zaimplementowanych odmian algorytmu Waxman'a [107], co jest szczególnie przydatne w tworzeniu dużych sieci. Z kolei tworzenie topologii poprzez import wykorzystuje się, jeśli dysponujemy plikiem z opisem sieci utworzonej przez zewnętrzne narzędzie. Program został wyposażony w procedurę importu plików XML zawierających topologie dostępne poprzez stronę internetową instytutu ZIB (Zuse-Institut Berlin) [18].

Niezależnie od wybranego sposobu tworzenia sieci, po jego zakończeniu niezbędne jest utworzenie pliku konfiguracyjnego. Ten etap wymaga wiele uwagi, gdyż zawartość pliku kompletnie definiuje środowisko badawcze, a ewentualna pomyłka może skutkować uzyskaniem niewłaściwych wyników. Na tym etapie wsparcie aplikacji *Vims* jest kluczowe, gdyż umożliwia automatyczne wygenerowanie gotowego do użycia pliku konfiguracyjnego, przy użyciu istniejącego szablonu. Dzięki temu unika się pomyłek w opisie topologii, a czas potrzebny na utworzenie pliku jest ograniczony do minimum. W praktyce tworzenie plików opisujących złożone sieci bez takiego wspomagania byłoby bardzo uciążliwe, czasochłonne i narażone na błędy. Po wygenerowaniu, plik jest dostępny do przeglądania i edycji, więc w razie potrzeby można go dodatkowo dostosować do indywidualnych potrzeb, choć część opisująca topologię rzadko wymaga dalszych zmian. Gdy plik jest gotowy, można przystąpić do symulacji.

Wykonanie pojedynczej symulacji może być niewystarczające. Zwykle potrzebne jest porównanie kilku wyników dla oceny różnic pomiędzy określonymi algorytmami, topologiami lub warunkami ruchowymi. Program symulacyjny umożliwia automatyczne powtarzanie symulacji, w oparciu o pojedynczy plik, w którym za pomocą specjalnej składni określone parametry definiuje się w postaci serii poszczególnych wartości, używanych w kolejnych powtórzeniach. Dodatkowe ułatwienie wynikające z takiego trybu pracy symulatora to automatyczne zbieranie wyników z kolejnych powtórzeń i umieszczenie ich we wspólnej tabeli w pliku raportu. Znacznie przyspiesza to przetwarzanie i porównywanie wyników.

Po wykonaniu serii symulacji należy zwykle przetworzyć wyniki, a następnie dokonać ich interpretacji. Program wspomaga przetwarzanie wyników poprzez opisany wcześniej mechanizm łączenia wyników z całej serii symulacji. Dodatkowo program oblicza przedziały ufności i odpowiednio formatuje listę wyników. Wszystko to dzieje się automatycznie przed zakończeniem pracy symulatora i dzięki temu powstałe pliki mogą być użyte bezpośrednio do generowania wykresów i tabel.

Opisana charakterystyka nie wyczerpuje możliwości i cech programów *msim* i *Vims*. Więcej informacji na ten temat można odnaleźć w [45,47,49,53].

5.3.4. Pomiary

Wybór pomiarów dokonywanych przez program symulacyjny odbywa się poprzez ich specyfikację w pliku konfiguracyjnym. Pomiary odbywają się w następujący sposób. W ramach pojedynczego odcinka symulacyjnego wszystkie obiekty zdolne do generowania pomiarów, np. węzeł lub źródło, generują związane z poszczególnymi zdarzeniami próbki liczbowe. Każda pojedyncza próbka, która jest powiązana z generującym ją obiektem i typem zdarzenia, zostaje przekazana za pośrednictwem menedżera raportów do rejestru powiązanego z konkretnym typem pomiaru. Bieżąca wartość rejestru zostaje następnie zmodyfikowana wartością próbki w sposób, jaki został zdefiniowany dla zdarzenia związanego z próbką. Może to być jedna lub więcej operacji typu:

- uśrednienie wartości poszczególnym próbek,
- zliczanie liczby wystąpień próbek,
- sumowanie wartości wszystkich próbek,
- określenie wartości minimalnej lub maksymalnej.

Najczęściej wykonuje się pomiary wielkości należących do dwóch pierwszych typów, np. średniej liczby wywłaszczeń koniecznych do utworzenia jednej ścieżki albo liczby ścieżek utworzonych z użyciem wywłaszczeń. Na zakończenie każdego odcinka symulacyjnego rejestry zawierają częściowe wyniki pomiarów zebrane w ciągu tego odcinka. Wyniki częściowe nie mogą być jeszcze użyte do interpretacji badanych sieci, gdyż bez znajomości wyników z innych odcinków nie jest możliwe określenie przedziałów ufności.

Po zebraniu wyników częściowych trafiają one do powiązanych z nimi dwóch dodatkowych rejestrów. W pierwszym przechowuje się wartość średnią wyników częściowych, a w drugim wartość średnią kwadratów wyników częściowych, która służy potem do obliczenia przedziału ufności metodą *t*-Studenta. Wyznaczone w ten sposób pary (wartość średnia i przedział ufności) są wynikami generowanymi przez program i mogą być użyte do interpretacji badanych zjawisk.

W programie zaimplementowano kilkadziesiąt różnych typów pomiarów, wśród których są następujące (wartości średnie oznaczają tu wartości uśrednione po zdarzeniach a nie po odcinkach):

- liczba tworzonych (zagregowanych) źródeł w sieci, co odpowiada liczbie żądań zestawienia ścieżki,
- liczba przyjętych i odrzuconych żądań zestawienia ścieżki,
- średnia liczba wywłaszczonych ścieżek M,
- średnia wartość względnego straconego pasma lokalnego $b_x^{(e)}$ i sieciowego b_{xnet},
- średnia wartość metryki złożonej Q,
- średni poziom rezerwacji pasma $b_b^{(e)}$,
- średnia długość kaskady wywłaszczeń $g^{(p)}$,
- liczba niepowodzeń realokacji wywłaszczonych ścieżek,
- średnia długość ścieżki l,
- liczba utworzeń ścieżek powodujących powstanie kaskady wywłaszczeń,
- średnia liczba ścieżek w kaskadzie wywłaszczeń $H^{(p)}$.

Wszystkie wielkości mierzone są zbiorczo oraz z podziałem na klasy tworzonych ścieżek. Dla każdej wielkości możliwe jest uzyskanie przedziału ufności, przy zadanym poziomie ufności, który domyślnie ustalony jest na wartość 0,95.

5.3.5. Weryfikacja modelu

Samodzielnie zrealizowany model symulacyjny powinien być z założenia traktowany jako niewiarygodny i podlegać szeregowi badań weryfikujących, które umożliwią wychwycenie błędów i uzasadnią wysoki stopień zaufania do uzyskanych przez program wyników.

Krytyczne podejście od własnego dzieła, choć trudne do osiągnięcia, jest najlepszym sposobem na prawidłową jego weryfikację w oparciu o dobre metody testowe.

W trakcie rozwijania programu prowadzono testowanie modelu korzystając z następujących metod i narzędzi.

1. Weryfikacja funkcjonowania programu w oparciu o pliki śladu.
2. Testy regresyjne na zgodność wyników po wprowadzeniu poważniejszych zmian.
3. Subiektywna analiza generowanych wyników.

Weryfikacje w oparciu o plik śladu (*log*) polegały w skrócie na wnikliwej jego analizie i porównaniu zawartości z przeprowadzoną niezależnie analizą oczekiwanego zachowania programu. Znajomość danych wejściowych, takich jak pasmo łączy, wybrana droga, lista dostępnych ścieżek oraz priorytety ścieżek, rozbudowana o pełną wiedzę na temat implementacji algorytmów umożliwiła weryfikację, czy kroki podejmowane przez program są oczywistą konsekwencją zaprogramowanego algorytmu. Jeśli działanie programu nie pokrywało się z oczekiwanym, wówczas konieczne było wyjaśnienie, skąd wynikają różnice, a w szczególności czy nie są wynikiem błędu programu. Proces takiego sposobu badań bywa żmudny, ale jest niezbędny do uzyskania wysokiego poziomu wiarygodności programu.

Przeprowadzone testy regresyjne miały na celu wyeliminowanie wtórnych błędów, tzn. takich, które mogły się pojawić w efekcie poprawiania przetestowanych już wcześniej bloków programu. Testy te polegały na powtórzeniu badań symulacyjnych przy użyciu dokładnie tych samych parametrów symulacji i topologii sieci, co badania przeprowadzone wcześniej. Korzystano tu z opcji programu umożliwiającej start symulacji z zadaną wartością początkową generatora liczb losowych. W ten sposób przy zadaniu tych samych warunków program powinien wygenerować wyniki pomiarów identyczne z wynikami odniesienia zebranymi wcześniej. Takie badania mogą być przeprowadzone w dużej części automatycznie łącznie z porównaniem zawartości plików raportów. Ewentualna różnica oznacza, że identyczne warunki początkowe prowadzą do innych wyników, co zwykle oznacza błąd w programie. To, czy błąd zawarty jest w nowej wersji czy w wersji odniesienia, wymaga indywidualnego sprawdzenia. Zdarzają się jednak sytuacje, gdy takie odstępstwo nie oznacza błędu, np. użycie innego kompilatora powoduje wykorzystanie innego sposobu inicjowania generatora liczb pseudolosowych, a to prowadzi do wyników nieco się różniących od wyników odniesienia. Inną tego typu sytuacją jest zmiana częstości lub sekwencji odczytu generatora liczb losowych, np. dodatkowy odczyt generatora w czasie symulacji przez nowy kod powoduje, że sekwencja liczb odczytywana przez pozostałe funkcje zmienia się, co też prowadzi do różnic w wynikach. Jednak nawet w takich przypadkach wyniki powinny być do siebie zbliżone, w granicach uzyskanych przedziałów ufności.

Badania oparte na analizie wyników polegały między innymi na sprawdzeniu, czy uzyskiwane wyniki mieszczą się w oczekiwanym przedziale oraz jaki wpływ na generowane wyniki mają zmiany pewnych parametrów. Ten rodzaj testów był w dużej części oparty na

doświadczeniu osoby przeprowadzającej badania a przez to ich wynik jest subiektywny. Mimo tego badania tego typu, prowadzone w sposób ciągły stanowią niezbędne uzupełnienie omówionych wcześniej metod.

Dzięki dążeniom do wyjaśnienia przyczyn wszelkich niejasności wyeliminowano szereg błędów programu i uzyskano z czasem wysoki poziom zaufania do wyników uzyskiwanych przez program. Biorąc pod uwagę prawa Murphy'ego, takie zaufanie musi pozostać ograniczone, jednak wyniki przeprowadzonych testów dają podstawę do optymizmu w tym zakresie. Nie bez znaczenia jest też fakt, iż koncepcja modelu i wyniki wygenerowane przez program zostały pozytywnie zweryfikowane przez recenzentów i zaprezentowane na szeregu krajowych i międzynarodowych konferencji naukowych oraz opublikowane na łamach czasopism [44,45,46,47,49,50,51,52]. Program został również praktycznie wykorzystany w projekcie badawczym dotyczącym koncepcji serwerów sterowania połączeniem w sieciach MPLS i ASON/GMPLS [53].

5.4. Zastosowanie

Zaimplementowany symulator jest uniwersalnym narzędziem, umożliwiającym prowadzenie badań nad różnymi metodami inżynierii ruchu. Szczególny nacisk położono na badania algorytmów wywłaszczania.

Program był wielokrotnie aktualizowany i rozbudowywany, a jego aktualna postać umożliwia wszechstronne badanie algorytmów wywłaszczania, obejmując pomiary czasów działania, generowanie histogramów, badanie cech kaskad wywłaszczania czy pomiary obciążenia sieci.

Symulator *msim* i aplikacja wspomagająca *Vims* okazały się bardzo przydatne w prowadzonych badaniach. Najważniejsze rezultaty uzyskane za ich pomocą zostały przedstawione i zinterpretowane w kolejnym rozdziale.

6. BADANIA I OMÓWIENIE WYNIKÓW

6.1. Opis warunków badań

Aby uzyskać miarodajne wyniki, konieczne było zaplanowanie warunków, w jakich badane będą algorytmy, a w szczególności wybrać topologie sieci oraz określić ilość i strukturę generowanego ruchu. Należało również zdecydować, które parametry algorytmów są najbardziej istotne i powinny wchodzić w skład kryteriów porównawczych.

Zdecydowano się na przeprowadzenie badań w oparciu o jedenaście różnych topologii, które podzielono na dwie serie. Do pierwszej wybrano sześć sieci o różnych wielkościach przy zachowaniu podobnej gęstości *d*, a do drugiej pozostałe pięć sieci, różniących się gęstością.

6.1.1. Topologie sieci

Wybór określonej topologii sieci może potencjalnie mieć wpływ na uzyskane wyniki. Dla osiągnięcia wysokiego poziomu wiarygodności wyników uzasadnione jest więc przeprowadzenie badań dla różnorodnych sieci. Podjęto przy tej okazji próbę określenia zależności wyników od parametrów sieci, o ile takie zależności istnieją.

Rozmiar sieci, rozumiany jako liczba węzłów w sieci, może mieć wpływ na wyniki badań poprzez różną wartość średniej długości drogi. Im droga jest dłuższa, tym więcej łączy może wymagać wywłaszczeń. Zatem im większa sieć, tym większe jest prawdopodobieństwo uzyskania znaczącej różnicy w wynikach w zależności od zasięgu algorytmu, tj. od tego, czy dany algorytm bierze pod uwagę dane tylko z pojedynczego łącza (lokalny), czy z całej domeny (globalny).

Gęstość sieci *d* określa się przez średnią liczbę łączy wychodzących od jednego węzła (wzór 3.1). Duża gęstość oznacza, że węzeł jest statystycznie lepiej skomunikowany z sąsiadami, a przez to ma potencjalnie większe możliwości wyboru drogi obejściowej. Może to mieć wpływ na prawdopodobieństwo wystąpienia wywłaszczenia lub przywrócenia wywłaszczonych ścieżek na alternatywnych drogach.

Starano się wybrać topologie możliwie bliskie tym, jakie spotykane są w sieciach operatorów telekomunikacyjnych. Źródłem informacji była biblioteka SNDlib [18], zgromadzona i udostępniona na serwerze sieciowym instytutu Zuse w Berlinie (Zuse-Institut Berlin, ZIB). Tabela 6.1 zawiera zestawienie parametrów sieci wybranych do badań, które dodatkowo przedstawiono graficznie na rys. 6.1 i 6.2.

Wybrane topologie zgodnie z założeniami podzielono na dwie serie. Do pierwszej (A) zaliczono sześć sieci o podobnej gęstości połączeń (od 2,93 do 3,60) ale różnej wielkości (od 12 do 50 węzłów). Do drugiej (B) wybrano pięć sieci o podobnej liczbie łączy (od 82 do 102) ale różnej gęstości połączeń (od 2,93 do 9,00). Tabela 6.2 zawiera zestawienie zaplanowanych scenariuszy badań. Dla wszystkich sieci ustalono średnią długość życia ścieżki na 1 godzinę.

Wartość ta może być dyskusyjna, jednak należy zauważyć, że z punktu widzenia badań nad algorytmami wywłaszczania istotna jest intensywność strumienia żądań tworzenia ścieżek. W praktyce więc nie tyle ważna jest konkretna wartość długości życia ścieżek, ile jej relacja do intensywności zgłoszeń, a tą również wyrażono w jednostkach na godzinę. Konkretne wartości długości życia oraz pasma ścieżek wybierano losowo z rozkładem wykładniczym.

Tabela 6.1. Parametry topologii wybranych do badań.

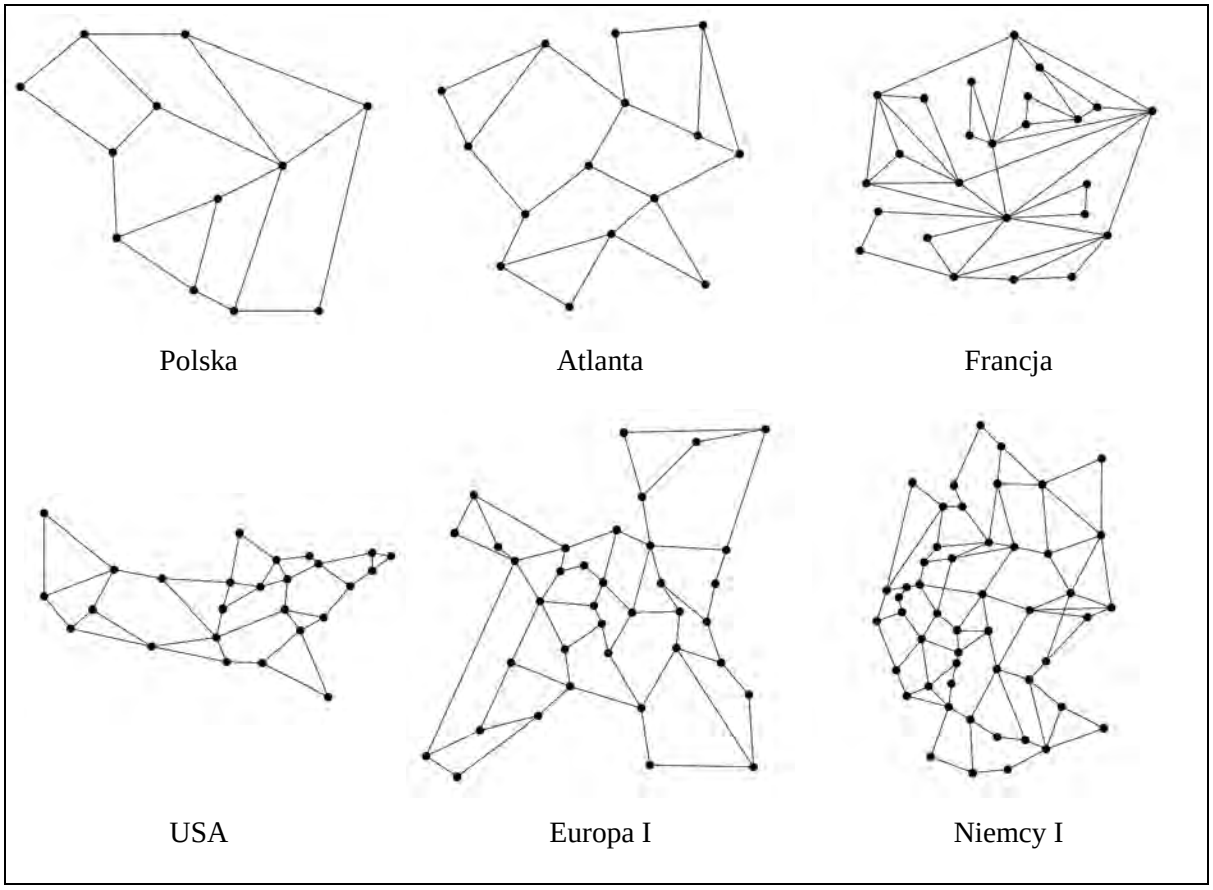
Lp.	Nazwa sieci	Liczba węzłów *N*	Liczba łączy (jednokier.) *L*	Przepływność łączy *c* [MBit/s]	Gęstość sieci *d*	Nazwa w bibliotece SNDlib
1	Polska	12	36	155	3,00	Polska
2	Atlanta	15	44	1000	2,93	Atlanta
3	Francja	25	90	2500	3,60	france
4	USA	26	84	64	3,23	janos-US
5	Europa I	37	114	7560	3,08	cost266
6	Niemcy I	50	176	40	3,52	germany50
7	Europa II	28	82	20	2,93	nobel-eu
8	Telecom Austria	24	102	504000	4,25	ta1
9	Nowy Jork	16	98	1000	6,13	newyork
10	Di-Yuan	11	84	1	7,64	di-yuan
11	Niemcy II	10	90	80000	9,00	dfn-bwin

6.1.3. Przedmiot badań

Dokonanie oceny algorytmów jest możliwe po wybraniu zbioru wielkości opisujących ich jakość z uwzględnieniem różnych definicji kosztów wywłaszczania (patrz rozdz. 3.4.1). O ile zbiór możliwych wielkości jest duży (patrz rozdz. 3.5.4), to w praktyce do oceny algorytmów używa się jedynie kilku parametrów. W niezależnych publikacjach [35,83] dokonuje się oceny w oparciu o liczbę wywłaszczeń i wywłaszczone pasmo (lokalne lub sieciowe). Za bardzo istotne należy uznać również czasy działania algorytmów. Opierając się dodatkowo na wynikach wcześniejszych analiz [50,51] zdecydowano, że do oceny algorytmów wystarczające są wyniki pomiarów pięciu następujących wielkości.

1. Średnia liczba wywłaszczeń: $\overline{M}$.
2. Średni indeks pasma straconego sieciowego: $\overline{b_{NET}}$ (wzór 6.1).
3. Średni indeks pasma straconego lokalnego: $\overline{b_{LOC}}$ (wzór 6.2).
4. Średnia metryka złożona: $\overline{Q}$.
5. Rozkład czasów działania algorytmów.

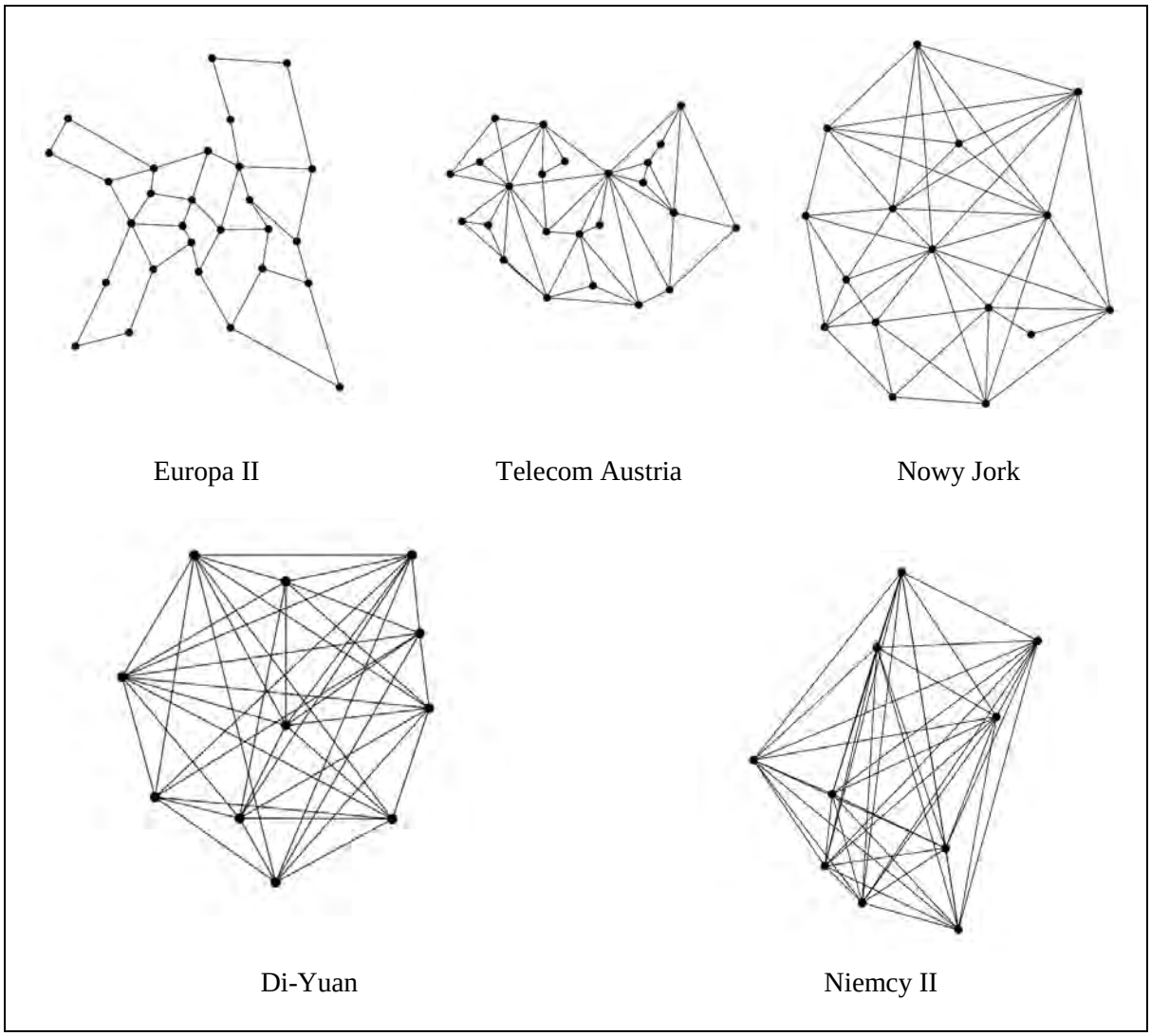
Trzy pierwsze parametry, tj. średnia liczba wywłaszczeń i średnie pasma stracone (sieciowe i lokalne), reprezentowane tu w postaci indeksów pasma, to najważniejsze parametry, charakteryzujące jakość algorytmów. Średnia liczba wywłaszczeń jest brana pod uwagę, gdy wymagana jest minimalizacja liczby usuwanych ścieżek, natomiast pasmo stracone charakteryzuje zdolność algorytmu do wybierania ścieżek najbardziej odpowiadających brakującemu pasmu. Czwarty parametr, średnia metryka złożona, umożliwia ujęcie w jednym parametrze liczby wywłaszczeń i ilości straconego pasma sieciowego, co znacznie ułatwia porównywanie algorytmów.



Rys. 6.1. Topologie o różnej wielkości wybrane do badań.

Ostatnia wielkość, tj. rozkład czasów działania algorytmu, wskazuje na rzeczywisty czas potrzebny na wybór listy kandydatów do wywłaszczenia. Jest to ważny parametr decydujący o możliwości praktycznej realizacji danej metody i stanowiący istotne uzupełnienie teoretycznych rozważań przeprowadzonych w oparciu o złożoność obliczeniową (patrz rozdział 3.6). Porównanie ze sobą czasów daje informację o tym, które z algorytmów wymagają

mniejszych zasobów procesora, co przekłada się na możliwość obsługi większej liczby zgłoszeń w określonym czasie lub możliwość użycia tańszej platformy sprzętowej. Analiza czasów musi wykraczać poza pomiar wartości średniej, gdyż rozkład czasów działania dla poszczególnych algorytmów może znacznie się różnić.



Rys. 6.2. Topologie o różnej gęstości wybrane do badań.

Do celów prezentacji wyników wprowadzono pojęcia indeksów pasma lokalnego b_{LOC} i sieciowego b_{NET}, zdefiniowane jako odwrotność straconego pasma, odpowiednio lokalnego i sieciowego:

$$b_{LOC} = \frac{1}{b_{xloc}} = \frac{B_{rnet}}{B_{xloc}}, \tag{6.1}$$

$$b_{NET} = \frac{1}{b_{xnet}} = \frac{B_{rnet}}{B_{xnet}}. \tag{6.2}$$

Taki sposób prezentacji wyników wziął się z powodów czysto praktycznych, gdyż tylko w ten sposób można było uzyskać akceptowalne wartości przedziałów ufności. Uzyskiwane wyniki samego pasma straconego były narażone na przekłamania spowodowane sporadycznymi bardzo dużymi wartościami pasm ścieżek (generowanymi losowo), co powodowało, że niejednokrotnie przedział ufności przekraczał wartość zmiennej, powodując nieprzydatność takich rezultatów. Zdefiniowany wcześniej indeks pasma nie ma tej wady. Zatem wykresy zawierające wartość indeksu pasma należy interpretować w ten sposób, że im większa wartość, tym lepiej, tzn. tym mniejsze pasmo jest stracone w wyniku wywłaszczania.

Tabela 6.2. Scenariusze badań.

Seria badań	Topologia bazowa	Liczba priorytetów	Średnie pasmo ścieżki [Mbit/s]	Intensywność zgłoszeń [1/godz.]	Średnia liczba ścieżek na łączu
A	Polska	3	6,5	130	24
	Atlanta	3	40	170	25
	Francja	3	99	330	25
	USA	3	2,5	250	26
	Europa I	3	316	320	24
	Niemcy I	3	1,5	500	26
B	Europa II	3	0,8	260	25
	Telecom Austria	3	20000	400	25
	Nowy Jork	3	40	450	25
	Di-Yuan	3	0,04	560	25
	Niemcy II	3	3200	750	25

6.2. Wyniki badań - seria A

Wyniki przeprowadzonych badań przyniosły szereg ciekawych wniosków. W niniejszym podrozdziale przedstawiono i skomentowano najbardziej interesujące rezultaty uzyskane w ramach badań topologii serii A, tzn. sieci różniących się wielkością przy zachowaniu podobnej gęstości. Uzyskane wyniki przedstawiono w postaci wykresów, na których umiesz-

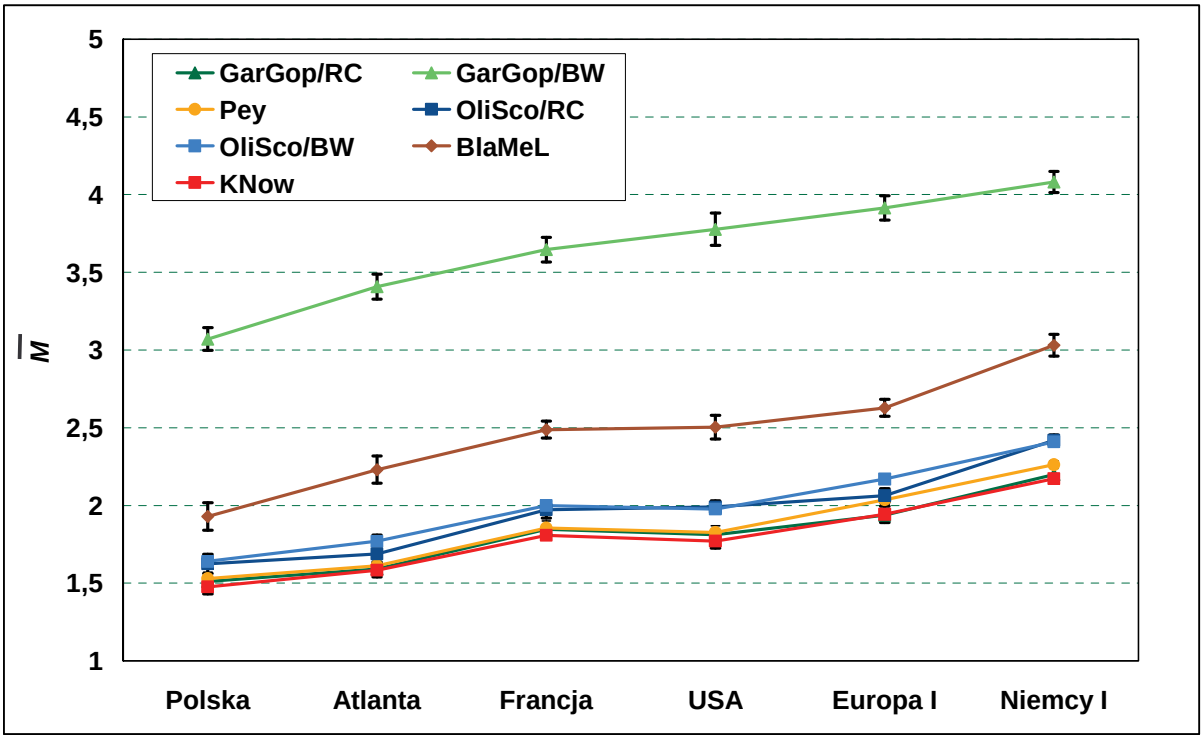
czono również przedziały ufności. Te uzyskano metodą *t*-Studenta z poziomem ufności równym 0,95 przy użyciu procedur wbudowanych w program symulacyjny.

6.2.1. Liczba wywłaszczeń

Jedną z najważniejszych, jeśli nie najważniejszą miarą jakości algorytmów jest średnia liczba wywłaszczeń $\overline{M}$. Parametr ten określa, ile średnio ścieżek zostaje usuniętych w efekcie wykonania procedury wywłaszczania dla przyjęcia pojedynczej ścieżki. Oczywiście ta wielkość powinna być jak najmniejsza, gdyż każde usunięcie ścieżki skutkuje chwilową lub trwałą utratą określonej liczby połączeń.

Na rys. 6.3 przedstawiono wykres liczby wywłaszczeń dla sieci z serii A, tj. dla sieci różniących się liczbą węzłów. Już pobieżna analiza pozwala zaobserwować następujące zależności:

- różne algorytmy i ich warianty prowadzą do bardzo różnych wyników oraz
- liczba wywłaszczeń rośnie wraz ze wzrostem rozmiaru sieci.



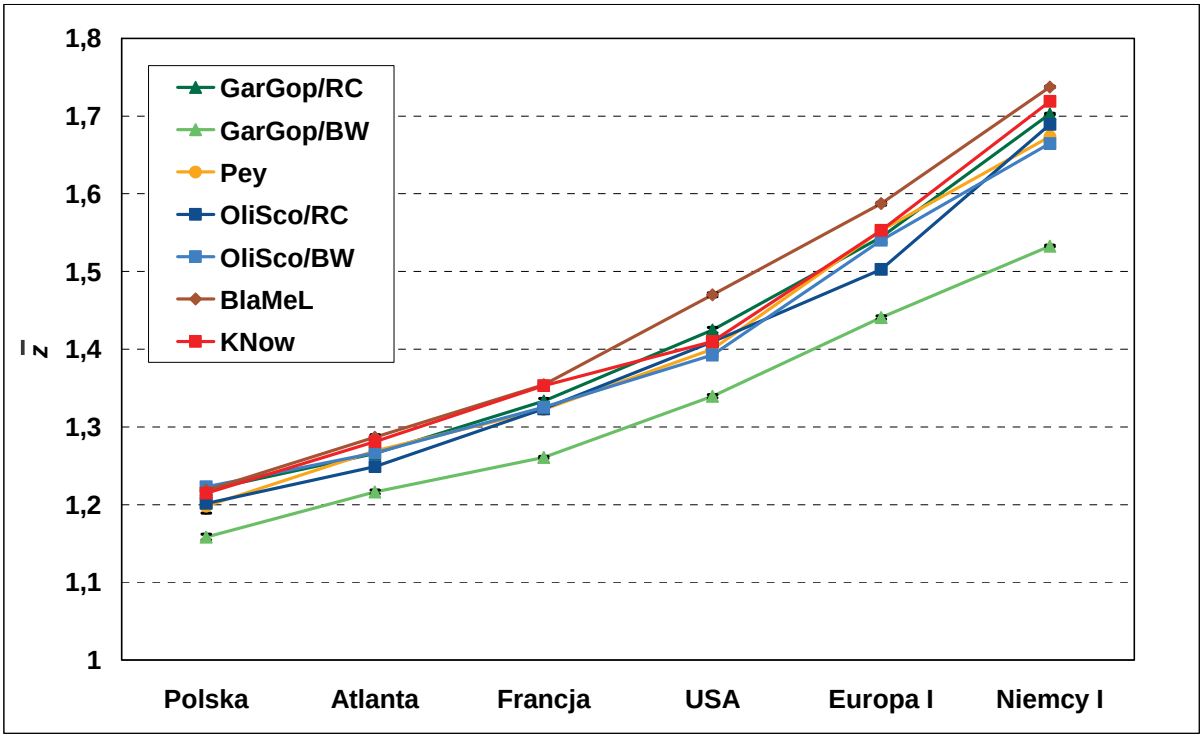
Rys. 6.3. Zależność średniej liczby wywłaszczeń *M* od rozmiaru sieci (seria A).

Istnienie różnic w wynikach jest oczywiste, gdyż algorytmy działają w różny sposób i w różny sposób definiują funkcję kosztu wywłaszczania. Zaskakiwać może jednak rozpiętość wyników, np. dla topologii „Polska” w zakresie od około 1,5 do około 3,1. Zauważyć przy tym można, że najgorszy wynik generuje algorytm GarGop/BW, którego priorytetem nie jest minimalizacja liczby wywłaszczeń, ale wywłaszczonego pasma. Choć słabe rezultaty w tym zestawieniu nie dziwią, to jednak warto zdawać sobie sprawę, że użycie algorytmu GarGop/BW może powodować średnio ponad dwukrotny wzrost liczby wywłaszczeń

w stosunku do większości pozostałych algorytmów.

Zaobserwowano także, że średnia liczba wywłaszczeń zwiększa się wraz ze wzrostem rozmiaru sieci. Taka zależność nie jest zaskakująca, gdyż w większych sieciach częściej występują długie ścieżki, a to prowadzi do większej liczby łączy, na których należy przeprowadzić wywłaszczenie. Potwierdzają to wyniki przedstawione na rys. 6.4, z których wynika, że średnia liczba łączy z wywłaszczaniem rzeczywiście rośnie wraz ze wzrostem liczby węzłów sieci, osiągając w tej serii badań wartości od około 1,2 do około 1,7. Można więc w tym miejscu podsumować tą część badań następującym wnioskiem.

Wniosek 6.1. Średnia liczba wywłaszczeń rośnie wraz ze wzrostem liczby węzłów w sieci, przy zachowaniu podobnej gęstości sieci.



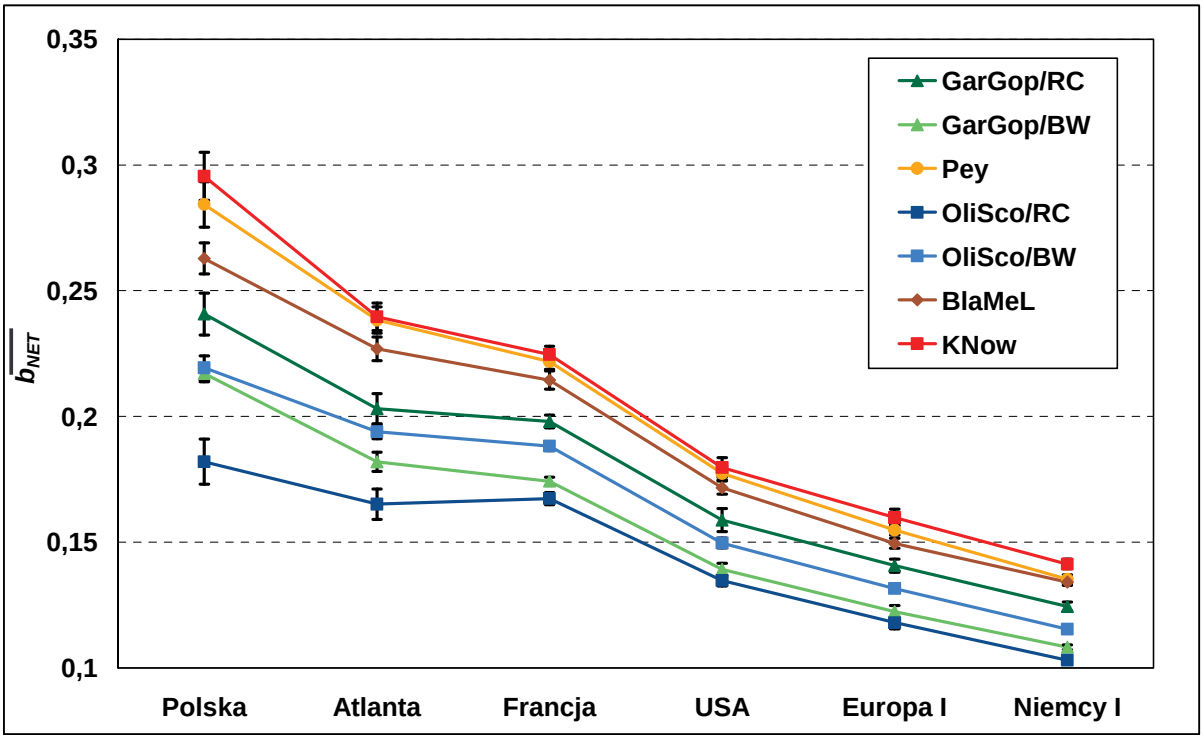
Rys. 6.4. Średnia liczba $\bar{z}$ łączy wymagających wywłaszczenia w funkcji rozmiaru sieci (seria A). Przedziały ufności mają tu pomijalne wartości.

6.2.2. Wywłaszczone pasmo

Drugim najczęściej ocenianym kryterium porównawczym algorytmów jest wywłaszczone pasmo. Parametr ten jest o tyle istotny, że większe wywłaszczone pasmo oznacza więcej straconych pakietów danych i mniejszy ruch obsłużony przez sieć.

Przedstawione wcześniej słabe wyniki algorytmu GarGop/BW w odniesieniu do liczby wywłaszczeń pozwalają przypuszczać, że znacznie lepiej wypadnie on w zestawieniach porównujących wywłaszczone pasmo. Wyniki przedstawione na rys. 6.5 i 6.6 przeczą jednak tym przewidywaniom. Co więcej, rezultaty osiągane przez ten algorytm znajdują się wśród

dwóch najgorszych wyników. To pozwala wysnuć wniosek, że w takiej postaci algorytm GarGop/BW nie sprawdza się w żadnym z dwóch istotnych kryteriów wyboru kandydatów, tj. liczbie wywłaszczeń i ilości wywłaszczonego pasma.



Rys. 6.5. Średni indeks wywłaszczonego pasma sieciowego b_{NET} w funkcji rozmiaru sieci (seria A).

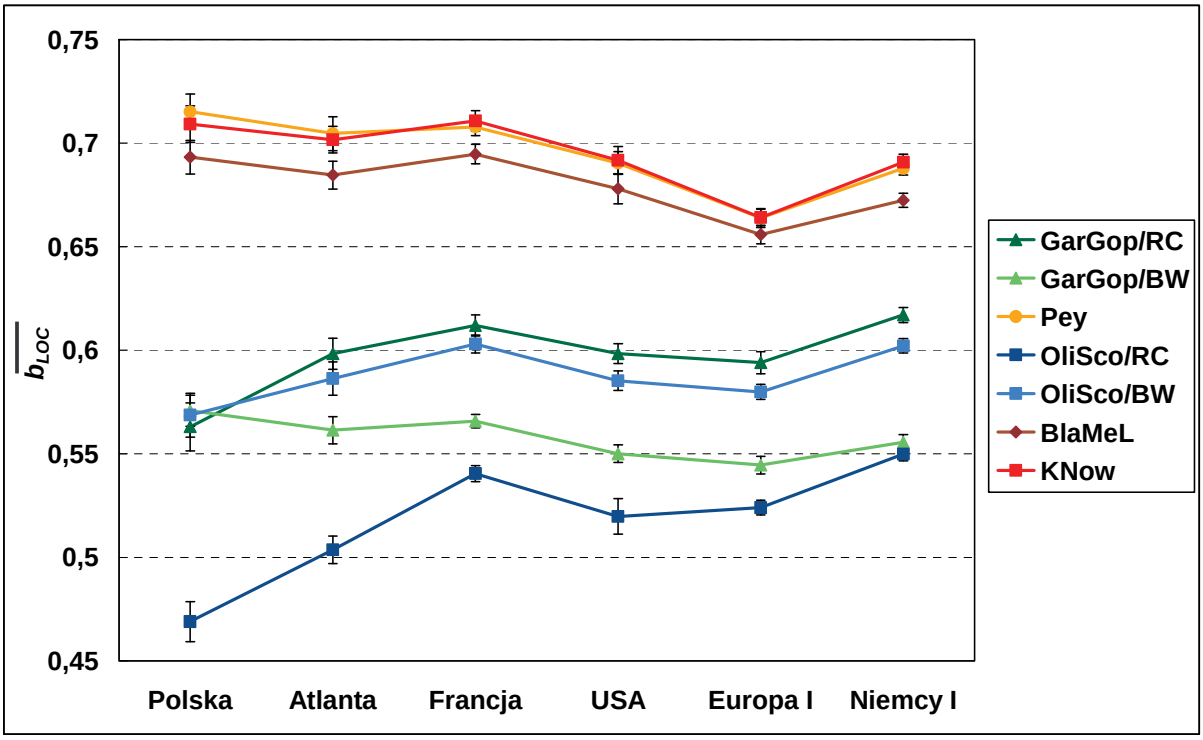
W tym zestawieniu zaskakujący jest fakt, że większość algorytmów minimalizujących liczbę wywłaszczeń umożliwia równocześnie minimalizowanie wywłaszczonego pasma. To pozwala wyciągnąć ważny wniosek, że minimalizacja liczby wywłaszczeń i minimalizacja wywłaszczonego pasma nie muszą być w sprzeczności. Ma to wielkie znaczenie praktyczne dla problemu wyboru najlepszego algorytmu i jego wariantu, gdyż eliminuje problem decyzji, które z dwóch kryteriów są w sieci najważniejsze – liczba wywłaszczeń czy stracone pasmo. Przeprowadzone badania wskazują, że istnieją takie algorytmy, które dobrze sprawdzają się w obu kategoriach. Ocenie metod pod tym kątem służy metryka złożona Q, opisana w kolejnej części.

Wniosek 6.2. Dobre algorytmy wywłaszczania umożliwiają minimalizację jednocześnie liczby wywłaszczeń i ilości wywłaszczonego pasma.

6.2.3. Metryka złożona

Metryka złożona Q umożliwia ocenę zdolności algorytmów do podejmowania decyzji zapewniających osiągnięcie jednocześnie małej liczby wywłaszczeń i małego straconego pa-

sma. Jest prosta w interpretacji, gdyż większa wartość oznacza lepszą jakość wywłaszczania w obu tych obszarach.



Rys. 6.6. Średni indeks wywłaszczonego pasma lokalnego b_{LOC} w funkcji rozmiaru sieci (seria A).

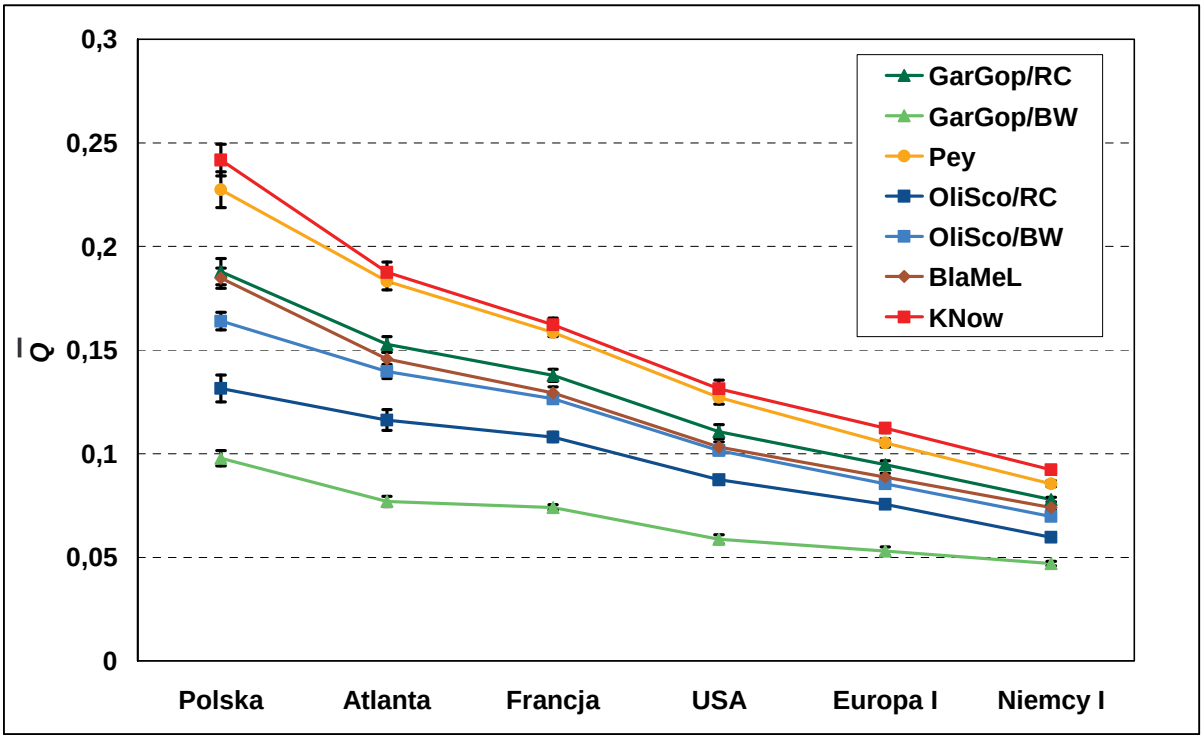
Wykres wyników badanych algorytmów w tym zakresie przedstawiono na rys. 6.7. Najlepsze wyniki osiąga tu algorytm globalny KNow, a nieco gorszy jest algorytm Pey, który zapewnia rezultaty optymalne pod względem liczby wywłaszczeń w skali lokalnej. Pozostałe algorytmy generują wyraźnie gorsze wyniki, co jest szczególnie widoczne dla algorytmów GarGop/BW i OliSco/RC, odbiegających znacząco od pozostałych rezultatów.

6.3. Wyniki badań - seria B

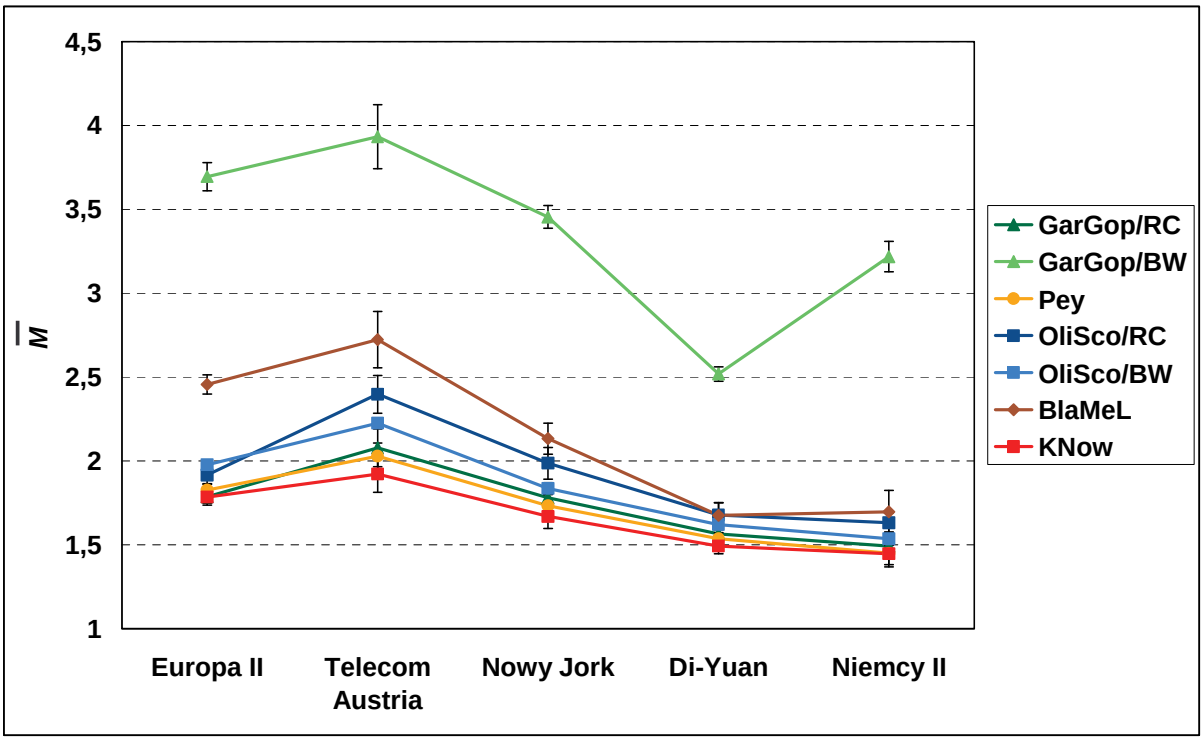
Seria badań B została dobrana tak, aby reprezentować rzeczywiste sieci o różnych gęstościach przy zachowaniu podobnej liczby łączy. Podobnie jak w serii A, poziom ufności wyników jest równy 0,95.

6.3.1. Liczba wywłaszczeń

O ile badania w serii A wykazały wyraźny wzrost liczby wywłaszczeń przy wzroście rozmiaru sieci, o tyle w serii B nie otrzymano jednoznacznej zależności od gęstości sieci. Wyniki przedstawione na rys. 6.8 wskazują na to, że charakter zmienności jest zależny od typu algorytmu. Można jednak stwierdzić, że zwykle w sieciach o większej gęstości jakość wywłaszczania jest lepsza, tzn. zmniejsza się liczba wywłaszczeń.



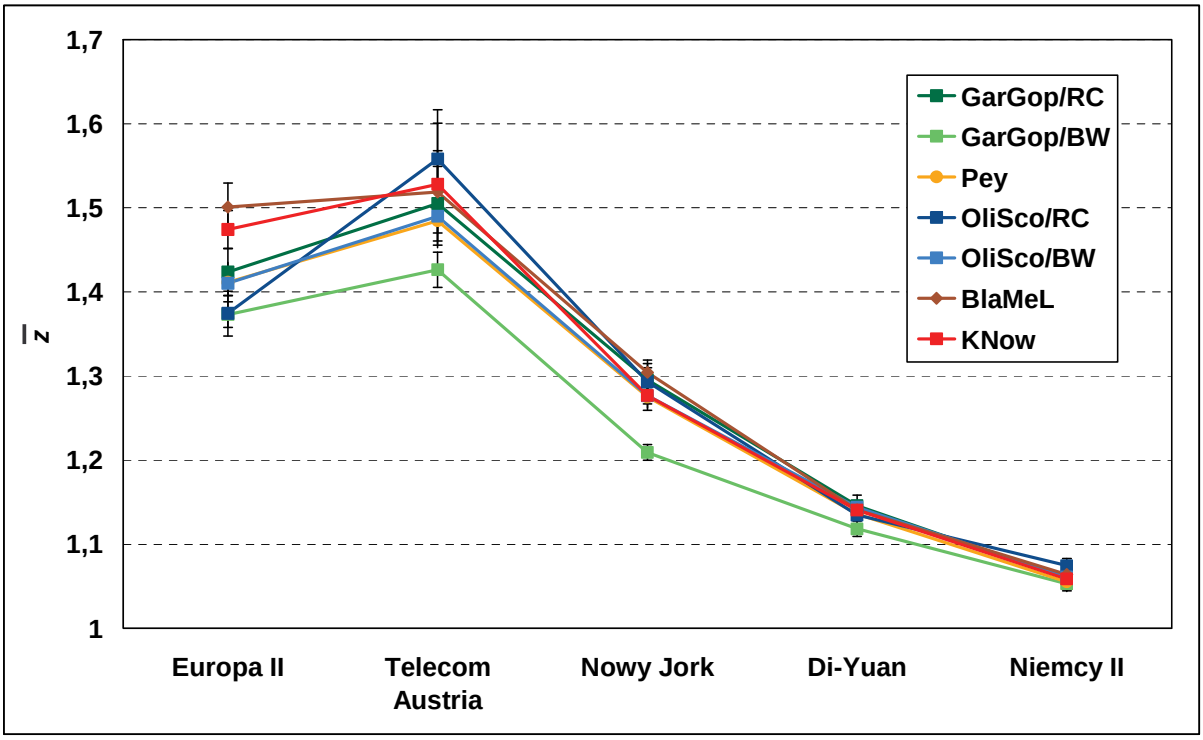
Rys. 6.7. Średnia wartość metryki złożonej *Q* w funkcji rozmiaru sieci (seria A).



Rys. 6.8. Zależność średniej liczby wywłaszczeń *M* od gęstości sieci (seria B).

Powodem takiego charakteru zmian jest to, że dla gęstszych topologii średnia długość drogi się zmniejsza, a przy tym zmniejsza się prawdopodobieństwo tego, że wywłaszczanie wykonuje się na więcej niż jednym łączu na drodze połączeniowej. To z kolei prowadzi do

powstania mniejszej liczby wywłaszczeń. Potwierdzeniem jest wykres z rys. 6.9 przedstawiający średnią liczbę łączy z wywłaszczaniem, który wykazuje bardzo podobną zmienność. Należy jednak zaznaczyć, że wybrane topologie o większej gęstości charakteryzują się mniejszą liczbą węzłów, co w świetle wyników badań w serii A, sprzyja zmniejszeniu liczby wywłaszczeń.

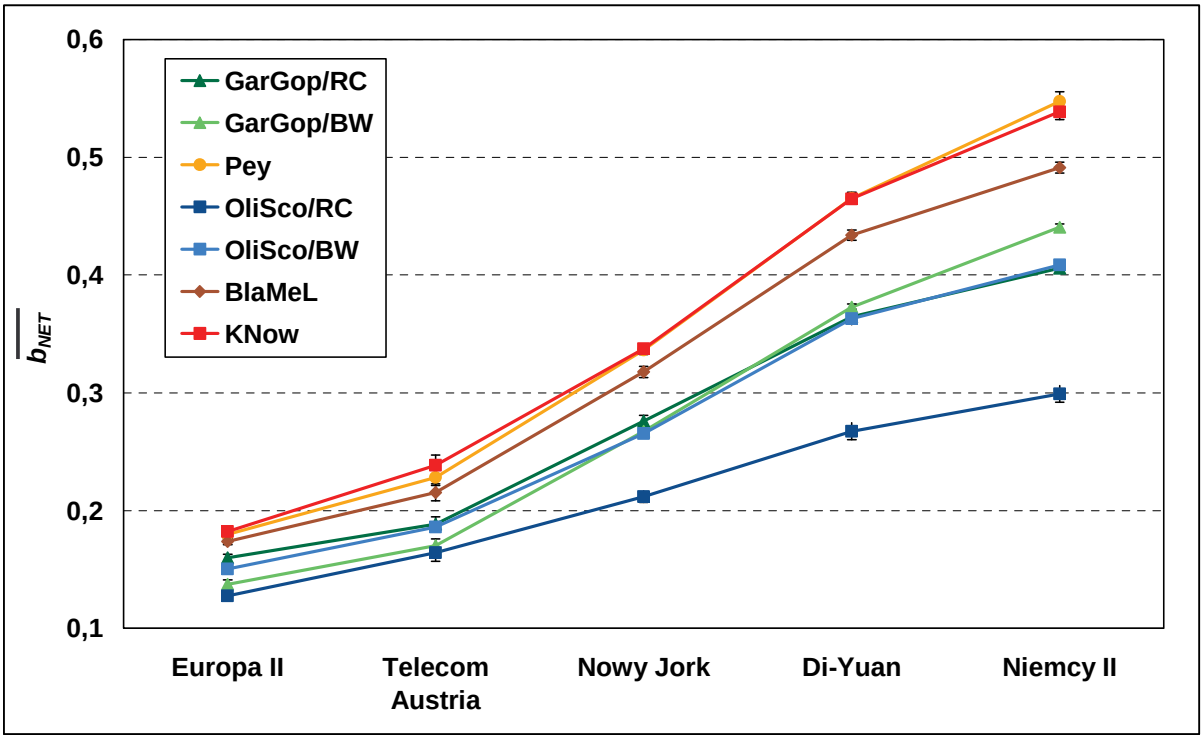


Rys. 6.9. Średnia liczba $\bar{z}$ łączy wymagająca wywłaszczania dla topologii w serii B.

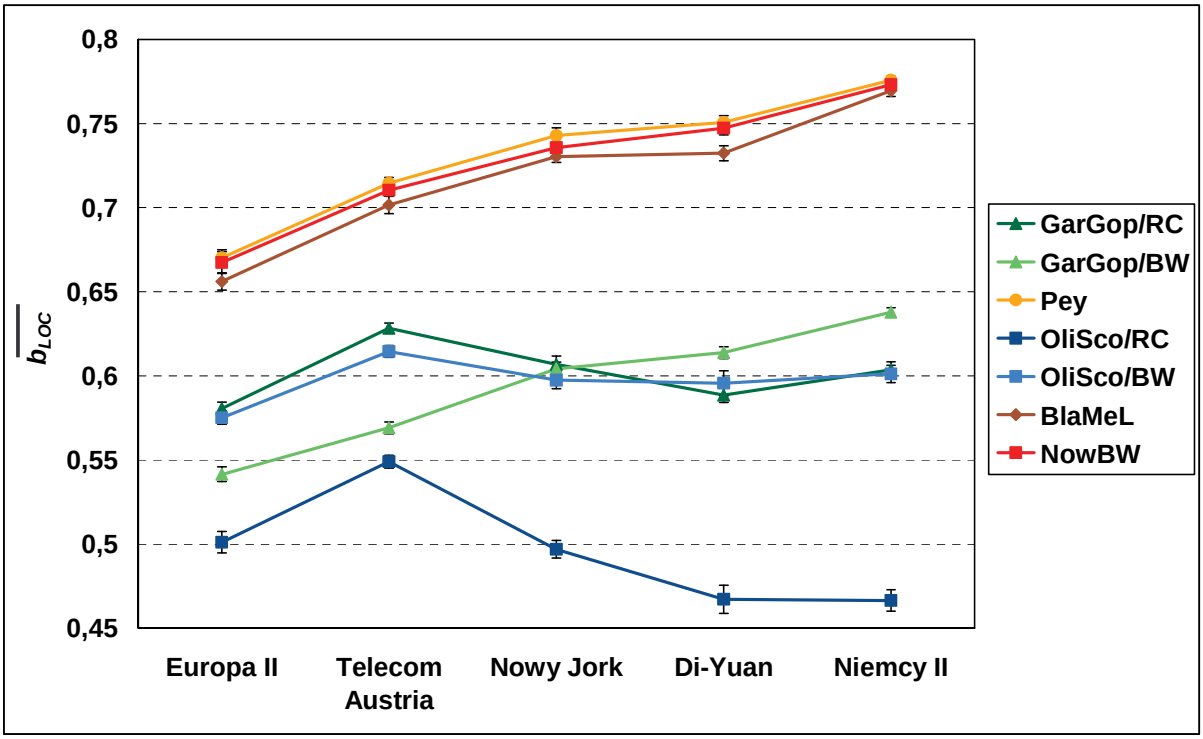
Analiza jakości poszczególnych algorytmów w zakresie minimalizacji liczby wywłaszczonych ścieżek potwierdza rezultaty uzyskane wcześniej. Najlepsze wyniki uzyskuje się poprzez zastosowanie algorytmów KNow i Pey, natomiast wyniki algorytmu GarGop/BW są również w tym zestawieniu raczej nie do zaakceptowania.

6.3.2. Wywłaszczone pasmo i metryka złożona

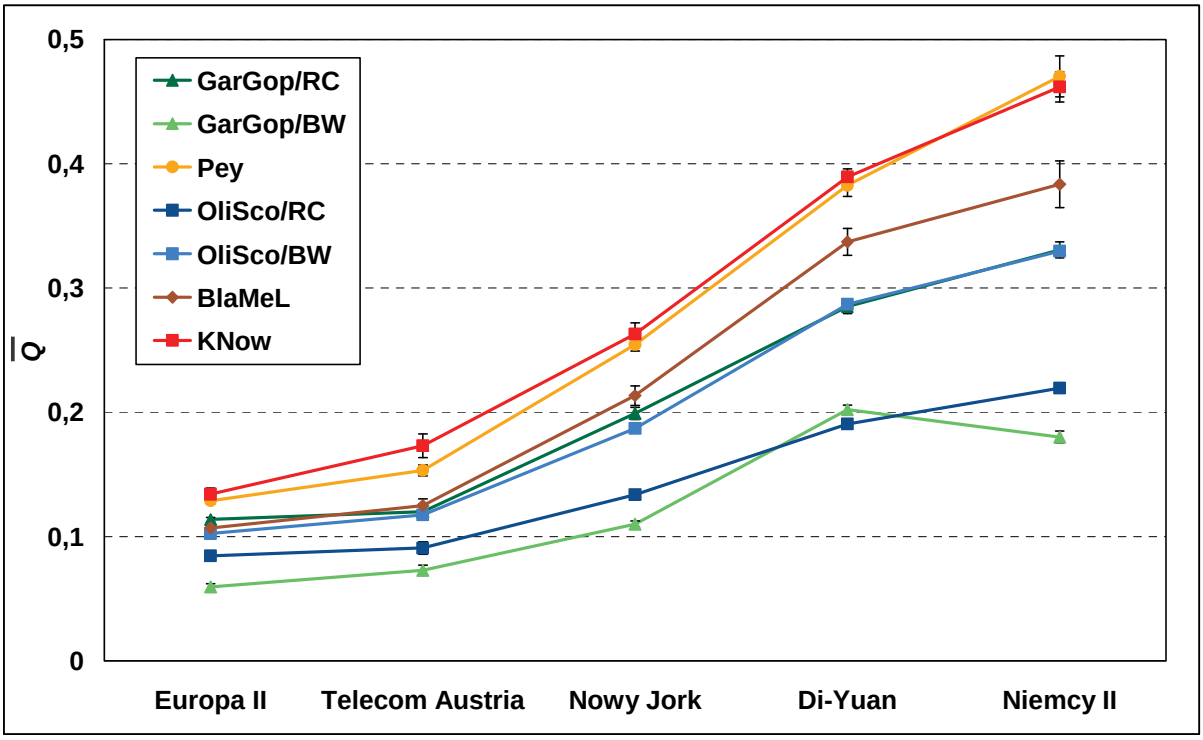
W odniesieniu do wywłaszczonego pasma wyniki przedstawione na rys. 6.10 i 6.11 wskazują, że przy większej gęstości jakość wywłaszczania dla najlepszych metod znacznie się poprawia, tzn. zdecydowanie zmniejsza się stracone pasmo. Odpowiednio większa jest też wartość metryki złożonej (rys. 6.12), której zmienność w tym przypadku zależy znacznie bardziej od pasma niż od liczby wywłaszczeń. Najlepsze rezultaty po raz kolejny osiągają algorytmy KNow i Pey, przy bardzo zbliżonych wynikach.



Rys. 6.10. Średni indeks wywłaszczonego pasma sieciowego b_{NET} w funkcji gęstości sieci (seria B).



Rys. 6.11. Średni indeks wywłaszczonego pasma lokalnego b_{LOC} w funkcji gęstości sieci (seria B).



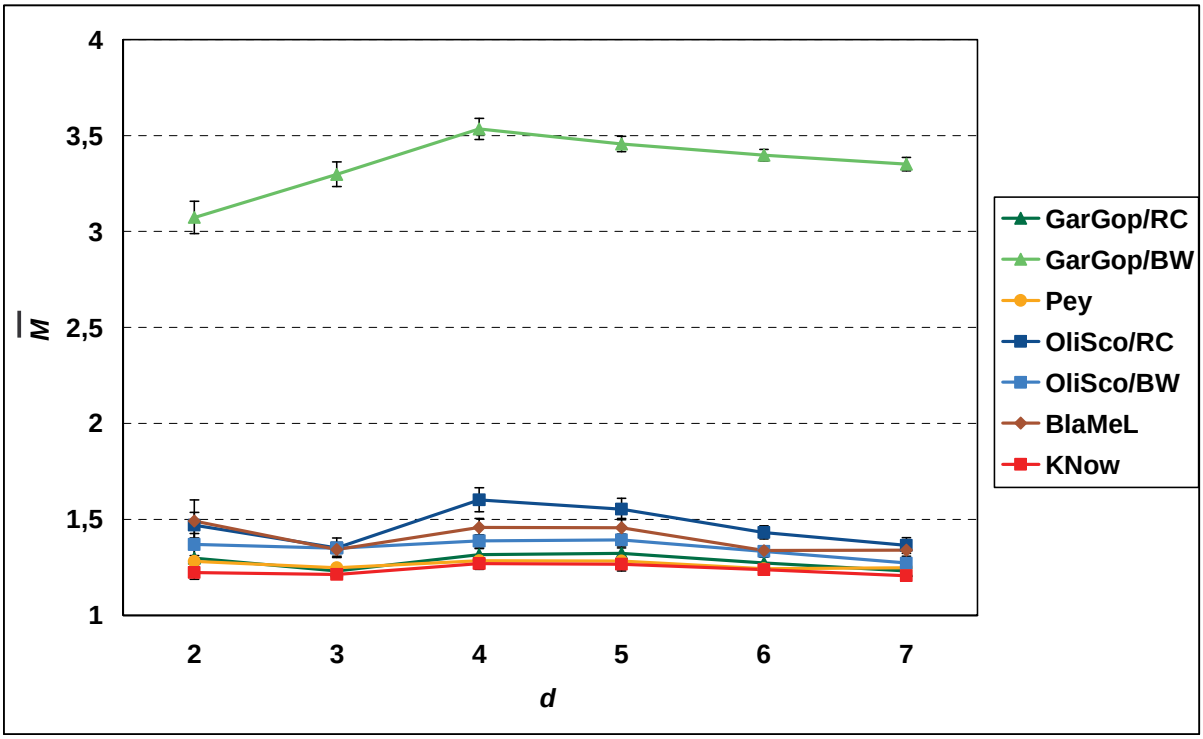
Rys. 6.12. Wartości średnie metryki złożonej *Q* w funkcji gęstości sieci (seria B).

6.3.3. Przypadek topologii regularnej

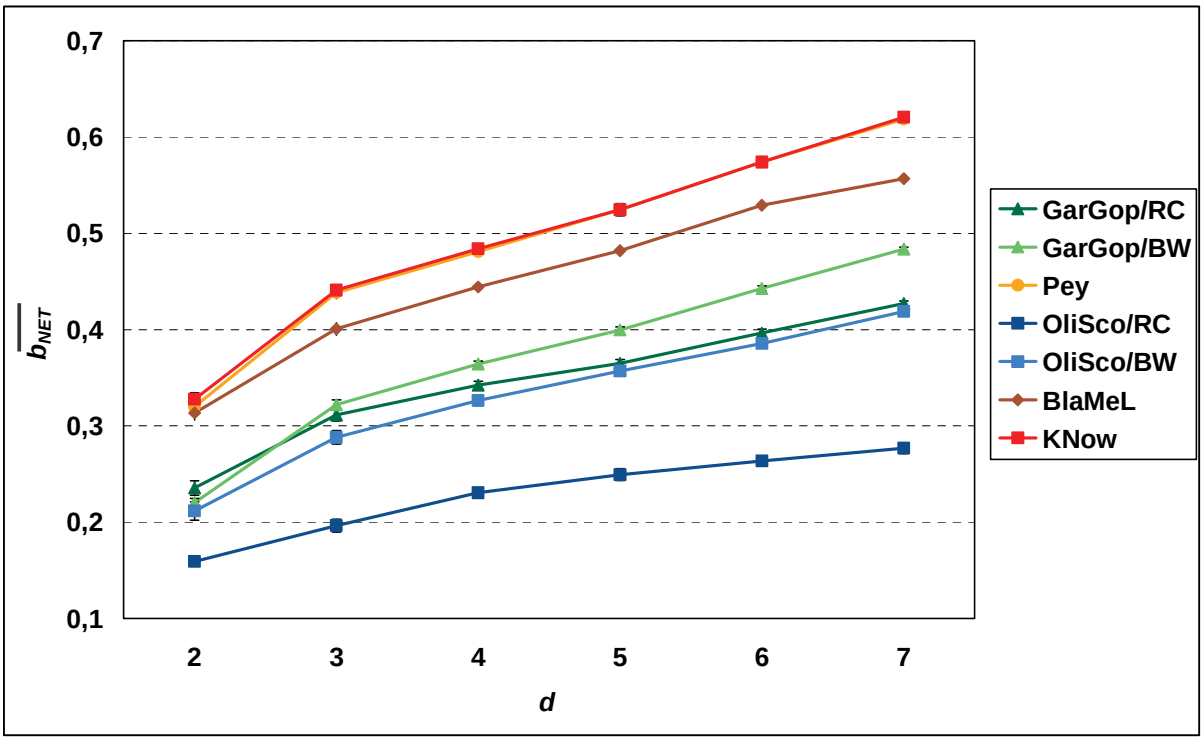
Wyniki przedstawione w ramach serii B odnoszą się do rzeczywistych sieci, które wybrane zostały z ograniczonego zbioru. Z tego powodu topologie te różnią się nie tylko gęstością, ale również liczbą węzłów. To może nasuwać podejrzenia, że uzyskane wyniki nie są rezultatem zmiany jedynie gęstości, ale też funkcją rozmiaru sieci, zwłaszcza, że charakterystyki zmian liczby wywłaszczeń i wywłaszczonego pasma zgadzają się z charakterystyką otrzymaną dla sieci o różnej liczbie węzłów w serii A. Aby rozstrzygnąć tę wątpliwość, przeprowadzono badania topologii regularnej składającej się z ośmiu węzłów, w której zwiększano liczbę łączy międzywęzłowych, począwszy od pierścienia (d = 2), a skończywszy na połączeniu każdy-z-każdym (*full mesh*, d = 7). W ten sposób uzyskano zmienność jedynie gęstości i bardziej jednoznaczne wyniki.

Uzyskane rezultaty wskazują na pewną zmienność liczby wywłaszczeń (rys. 6.13) oraz zdecydowaną poprawę indeksu straconego pasma (rys. 6.14) i metryki złożonej (rys. 6.15) wraz ze wzrostem gęstości sieci regularnej. Porównując te wyniki z wynikami z serii B można zatem uznać, iż wykazywany tam spadek liczby wywłaszczeń jest jednak efektem raczej zmniejszania się rozmiaru sieci. Można też podsumować, że zmiana gęstości sieci bez zmiany liczby węzłów nie powoduje znacznych różnic w liczbie wywłaszczeń. Na tej podstawie sformułowano wniosek określający zależności jakości wywłaszczania od zmiany gęstości sieci.

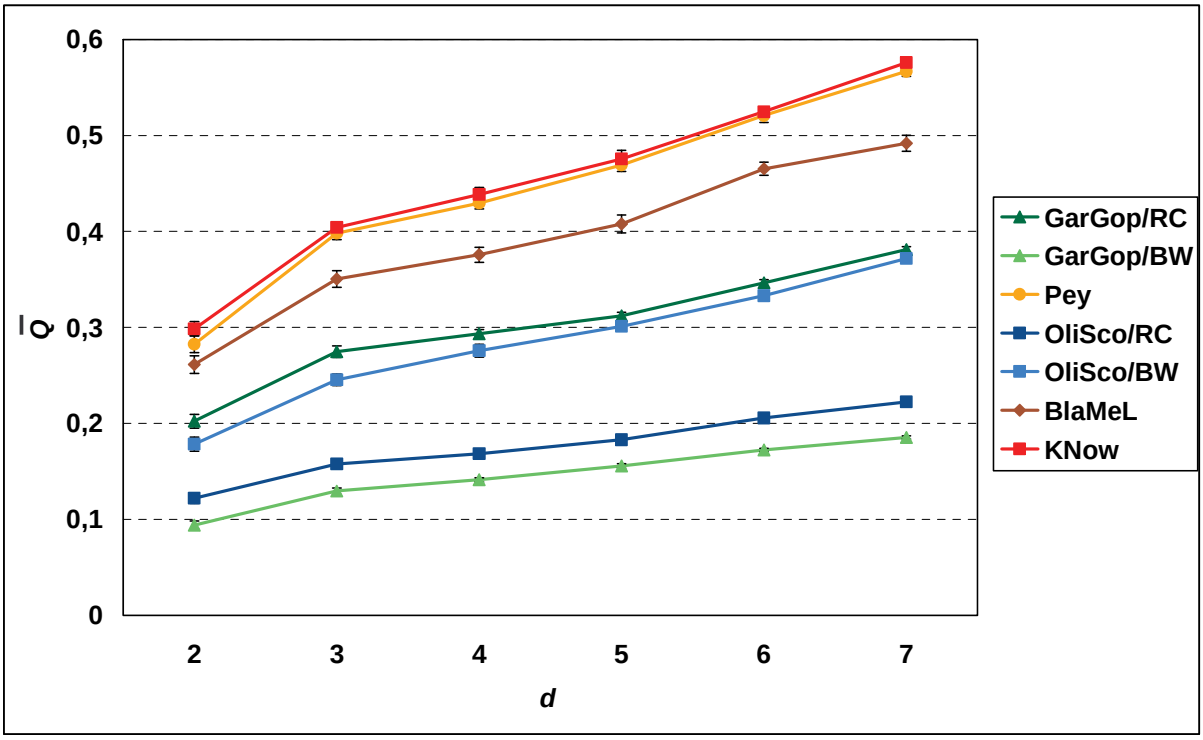
Wniosek 6.3. Wzrost gęstości sieci przy zachowaniu stałego obciążenia łączy powoduje zmniejszenie straconego pasma i brak znaczących zmian liczby wywłaszczeń.



Rys. 6.13. Średnia liczba wywłaszczeń M w funkcji gęstości d sieci regularnej.



Rys. 6.14. Średni indeks straconego pasma sieciowego b_{NET} w funkcji gęstości d sieci regularnej.



Rys. 6.15. Średnia wartość metryki złożonej *Q* w funkcji gęstości *d* sieci regularnej.

6.4. Wyniki pozostałych badań

6.4.1. Czasy działania algorytmów

Pomiary czasów działania algorytmów, a ściślej czasów działania procedur wyboru kandydatów do wywłaszczenia są uzupełnieniem rozważań teoretycznych określających złożoność obliczeniową (rozdziały 3.6 i 4.2.1) i służą ich praktycznej weryfikacji. Pomiary przeprowadzono dla dwóch najbardziej różniących się rozmiarem sieci z serii A, tzn. topologii „Polska” (12 węzłów) i „Niemcy I” (50 węzłów).

Badania wykonano na komputerze osobistym typu laptop z procesorem Intel Core i7 640M 2,8 GHz w systemie Linux Debian 5 (jądro w wersji 2.6) osadzonym na platformie wirtualnej Oracle Sun VirtualBox. Do pomiarów czasów wykorzystano dwukrotne wywołanie funkcji `gettimeofday()`, zwracającej czas systemowy na wejściu i na wyjściu z procedury wyboru ścieżek. Wyniki przedstawiono w postaci histogramu zależności średniej liczby próbek *n* od czasu wykonania *t*. Wartości liczbowe czasów *t* podano w mikrosekundach czasu rzeczywistego, choć należy pamiętać, że konkretne wartości nie mają tu większego znaczenia. Zdecydowanie bardziej istotne jest porównanie ze sobą rozkładów czasów dla poszczególnych algorytmów.

Wyniki pomiarów dla topologii „Polska” przedstawiono na rys. 6.16. Obserwowane różnice nie są duże i dla wszystkich mierzonych algorytmów zdecydowana większość wywołań kończy się przed upływem 100µs, a najczęściej pomiędzy 30µs a 60µs, w zależności od ba-

danego algorytmu. Czasy powyżej 120µs zdarzają się rzadko i stanowią od 2% do 6% wyników. Porównując poszczególne algorytmy można stwierdzić, że statystycznie nieco więcej czasu na obliczenia wymaga algorytm KNow. Pozostałe algorytmy osiągają zbliżone rezultaty.

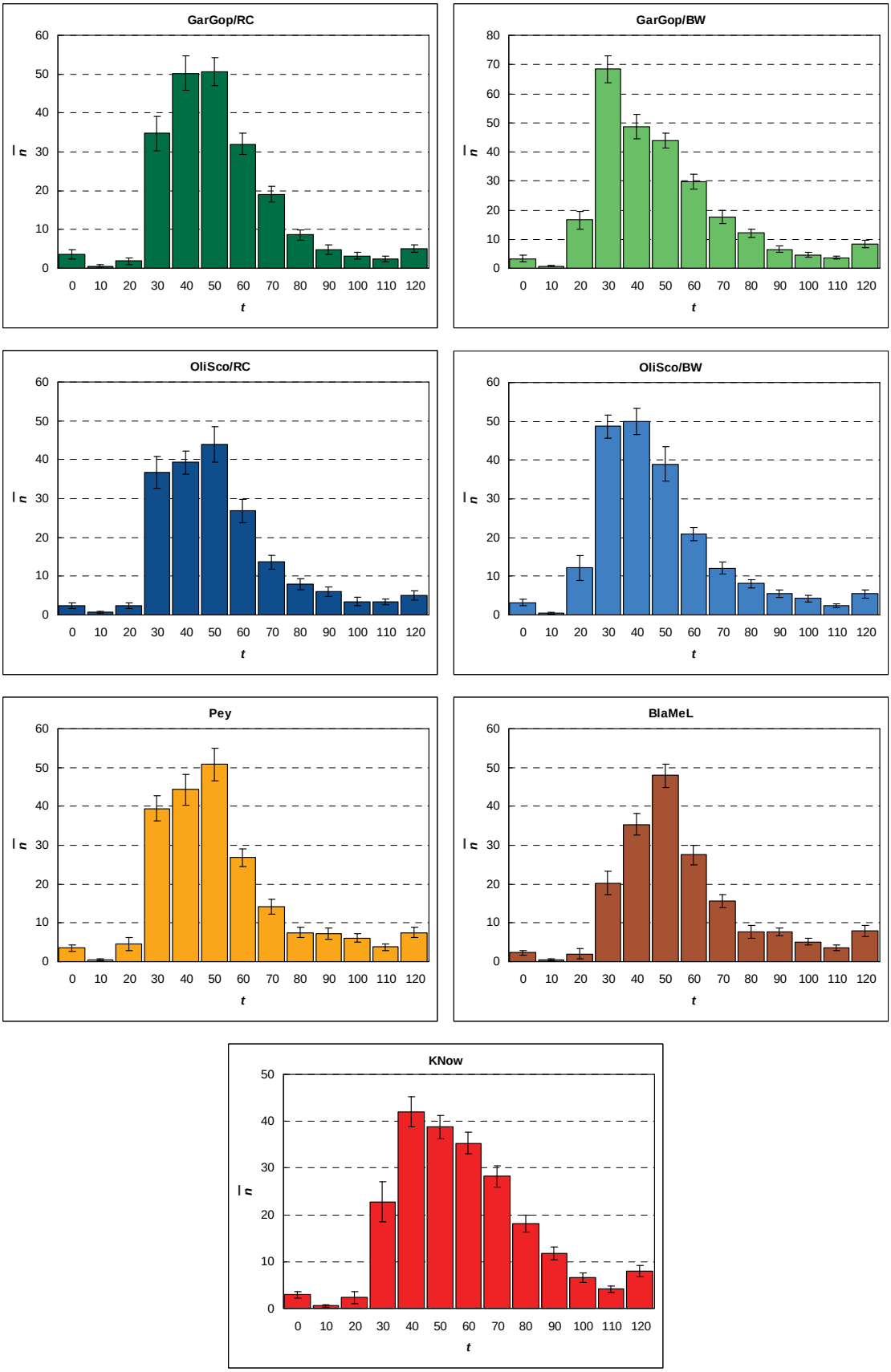
Ciekawych informacji dostarcza przedstawione na rys. 6.17 porównanie czasów dla dużej topologii „Niemcy I", złożonej z 50 węzłów. Tutaj czasy działania algorytmów są znacznie większe. Jedynie poniżej 10% wywołań mieści się w czasach poniżej 100µs i mimo, że większość wywołań kończy działanie w czasie do 150µs, to co dziesiąte wywołanie kończy się dopiero po upływie 400µs lub 500µs, zależenie od algorytmu. Najszybszymi algorytmami są OliSco/BW oraz GarGop/BW, choć jak pokazano wcześniej te algorytmy nie należą do najlepszych, zarówno w zakresie liczby wywłaszczeń, jak i wywłaszczonego pasma. Algorytm KNow ponownie uzyskuje jedne z najdłuższych statystycznie czasów, choć jednocześnie ta metoda ma jeden z najmniejszych odsetek czasów powyżej 600µs. Oznacza to, że algorytm KNow, jak też oba warianty GarGop, a więc algorytmy o zasięgu globalnym, są najbardziej przewidywalne pod kątem maksymalnego czasu działania.

Interesującym zjawiskiem, jakie można zaobserwować dla topologii „Niemcy I" dla części algorytmów jest pojawienie się lokalnego drugiego maksimum, które przypada na przedział czasu 200-250µs. Można przypuszczać, że wynika to ze stosunkowo częstej konieczności przeprowadzania wywłaszczeń na przynajmniej dwóch łączach. Zachowanie takie obserwujemy szczególnie wyraźnie dla algorytmów Pey, BlaMeL i OliSco/RC. Są to jednocześnie algorytmy, dla których obserwujemy stosunkowo dużo przypadków czasów działania powyżej 600µs (około 5%).

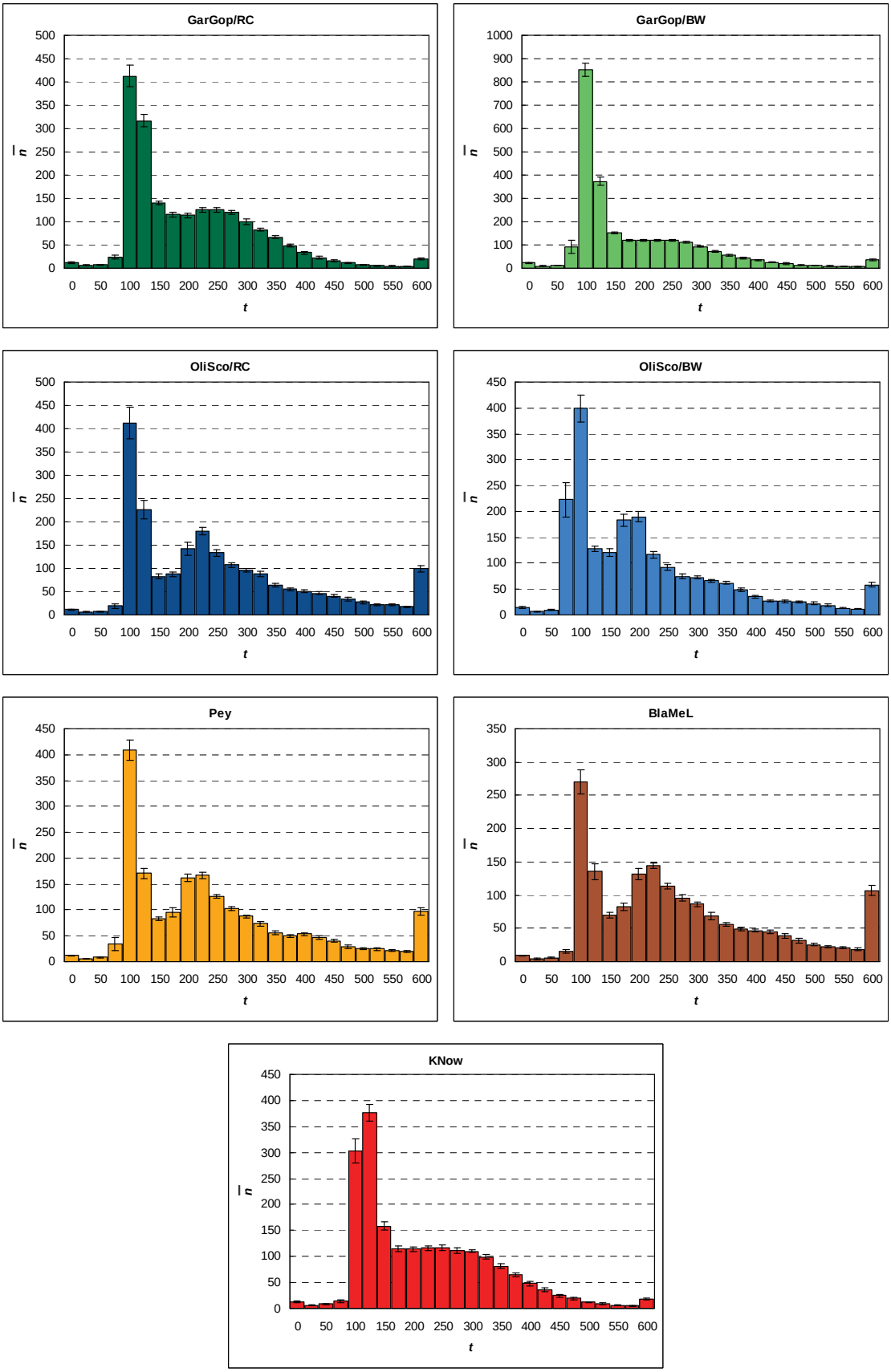
Warto przywołać wyniki testów jakościowych, tzn. liczby wywłaszczeń, wywłaszczonego pasma i metryki ogólnej i zestawić je z czasami działania. Porównanie takie pozwala wnioskować, że jakościowo najlepsze metody, tj. Pey i KNow, to również te, które wykazują najdłuższe czasy wykonania. Spośród nich algorytm Pey częściej kończy wyznaczanie ścieżek szybciej, ale znacznie gorzej radzi sobie z przypadkami wymagającymi dłuższych obliczeń. Algorytm KNow należy z kolei do tych, które wykazują najszybsze „wygaszanie" długich czasów wykonania.

W załączniku B zaprezentowano omówione wyniki w postaci tabeli z rozkładem procentowym dla poszczególnych czasów.

Wniosek 6.4. Najlepsze algorytmy wymagają najdłuższych czasów działania, natomiast najszybsze algorytmy prowadzą do słabych lub średnich wyników wywłaszczania.



Rys. 6.16. Histogramy czasów wykonania *t* algorytmów dla topologii Polska (skala w mikrosekundach czasu rzeczywistego).



Rys. 6.17. Histogramy czasów wykonania *t* algorytmów dla topologii Niemcy I (skala w mikrosekundach czasu rzeczywistego).

6.4.2. Wywłaszczanie lokalne i globalne

Podział na algorytmy globalne i lokalne wynika z sposobu analizy dostępnych ścieżek i wyboru spośród nich kandydatów do wywłaszczenia. W algorytmach lokalnych bierze się pod uwagę tylko dane dostępne lokalnie w węźle, a więc pasmo i priorytet ścieżki, natomiast algorytmy globalne operują na danych z całej domeny. Są dzięki temu w stanie dokonać lepszego wyboru szczególnie w przypadku, gdy wywłaszczanie jest efektem braku pasma na więcej niż jednym łączu na wybranej drodze.

Przedstawione dotąd wyniki najlepszych algorytmów wskazują na stosunkowo niewielką przewagę algorytmu globalnego KNow nad algorytmem lokalnym Pey. Jednocześnie zaobserwowano, iż tylko dla sieci obejmujących więcej niż 30 węzłów średnia liczba łączy z wywłaszczaniem jest większa niż 1,5. Zatem w przypadku mniejszych sieci zwykle wywłaszczanie występuje tylko na jednym łączu. To z kolei uniemożliwia wykorzystanie potencjału algorytmów globalnych, gdyż w przypadku gdy wywłaszczanie odbywa się tylko na jednym łączu zarówno algorytmy globalne jak i lokalne w ogólności nie różnią się w działaniu. Jedyną przewagą tych pierwszych może być wtedy wybór ścieżek w oparciu o dodatkowe atrybuty niedostępne lokalnie, jak długość ścieżki lub typy zajętych zasobów.

W trakcie badań prowadzono kontrolne śledzenie sposobu działania symulowanych algorytmów. Odkryto wówczas możliwość istnienia dodatkowego czynnika ograniczającego rezultaty osiągane przez algorytmy globalne w większych sieciach. Otóż według tych obserwacji, w nieregularnych sieciach drogi niezależnych ścieżek rzadko się pokrywają, a zatem małe jest prawdopodobieństwo znalezienia pojedynczej ścieżki obejmującej wszystkie łącza wymagające wywłaszczania, a to jest warunek na uzyskanie przewagi przez algorytmy globalne. Dla sprawdzenia tego zjawiska przeprowadzono dodatkowe pomiary określające, jaka ilość spośród dostępnych kandydatów pokrywa się z łączami wymagającymi wywłaszczania. Pominięto przy tym nieistotne dla problemu przypadki wywłaszczeń na pojedynczym łączu. Uzyskane wyniki potwierdziły zauważone wcześniej zjawisko. Rzeczywiście, dla topologii „Polska” około 84% wszystkich ścieżek obejmowało tylko pojedyncze łącze z wywłaszczaniem, a dla topologii „Niemcy I” było to około 78%. Zatem jedynie około 15-20% dostępnych ścieżek może przynieść korzyść z wywłaszczania globalnego. Przytoczone wyniki zawarto w tabeli 6.3.

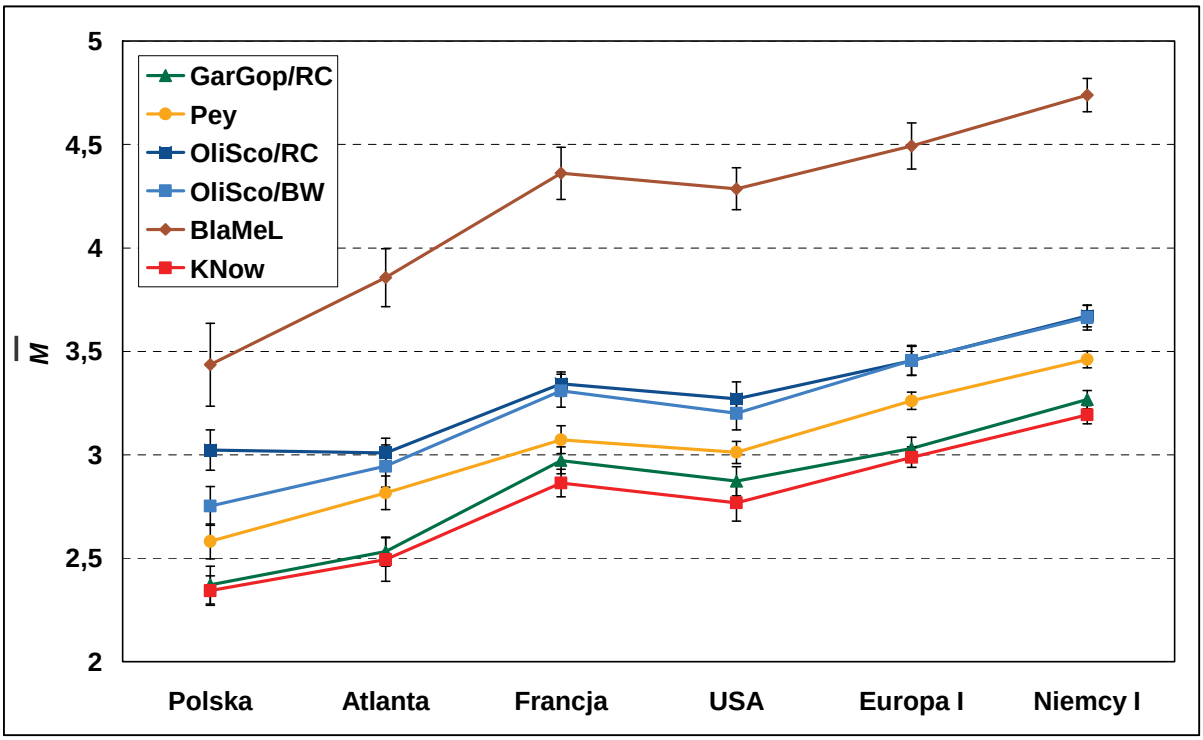
Mając na uwadze wymienione czynniki ograniczające jakość wyboru algorytmów globalnych, zdecydowano się na przeprowadzenie dodatkowych badań z uwzględnieniem tylko tych wywłaszczeń, w których występowały przynajmniej dwa łącza wymagające wywłaszczenia. Celem tych badań była ocena zasadności używania metod globalnych wywłaszczania, a jednocześnie skonfrontowanie tezy sformułowanej we wstępie do niniejszej pracy.

Na rys. 6.18, 6.19 i 6.20 przedstawiono odpowiednio wyniki średniej liczby wywłaszczeń M, średniego indeksu straconego pasma sieciowego b_{NET} oraz wartości średnie metryki złożonej Q dla tylko tych przypadków, gdy na drodze połączeniowej występują przynajmniej dwa

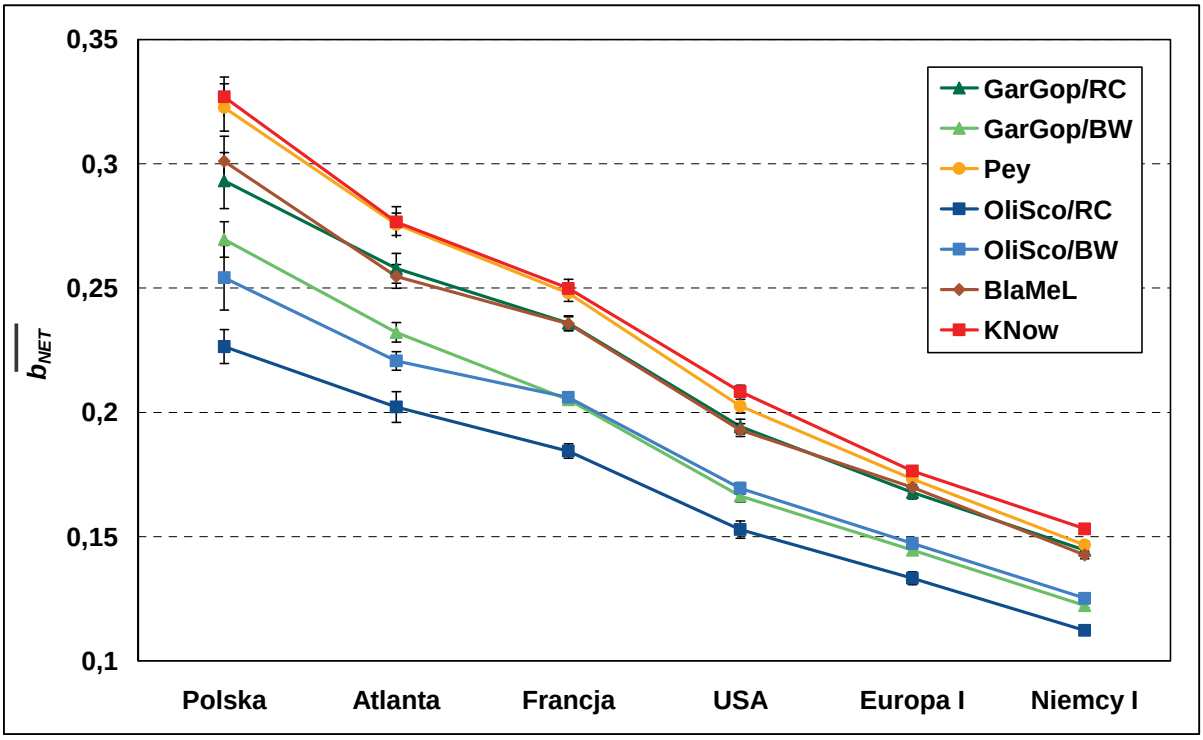
łącza wymagające wywłaszczania. Wyniki wskazują, że rzeczywiście w tych przypadkach algorytmy globalne okazują się bardziej skuteczne od lokalnych. Przewaga algorytmu KNow nad algorytmem Pey jest w tych przypadkach dużo wyraźniejsza. Co więcej, znacznie lepsze wyniki osiąga także drugi z badanych algorytmów globalnych – GarGop/RC, który jest nawet lepsza od algorytmu Pey. To pozwala wysnuć następujące wnioski.

Tabela 6.3. Średnia liczba ścieżek przechodzących przez daną liczbę łączy wymagających wywłaszczania. Uwzględniono tylko przypadki wywłaszczania na przynajmniej dwóch łączach. Poziom ufności 0,95.

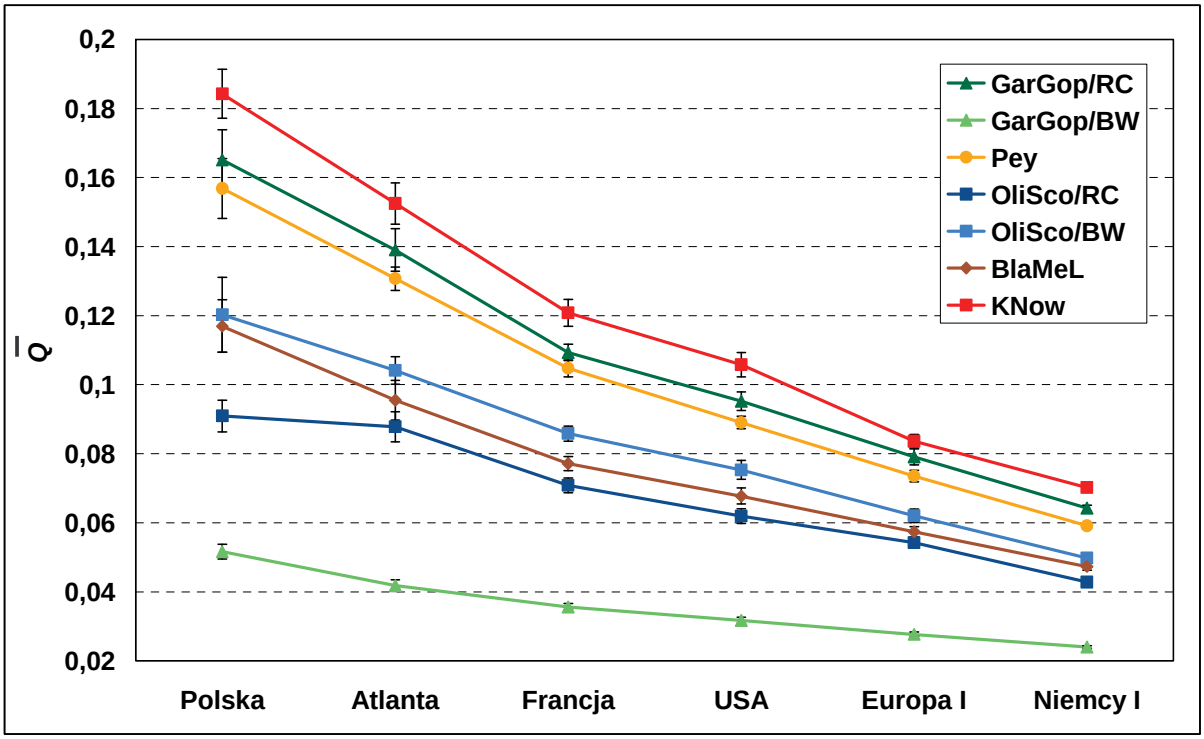
	Polska			Niemcy I		
Liczba łączy	Liczba ścieżek	Przedział ufności	Procentowy udział ścieżek	Liczba ścieżek	Przedział ufności	Procentowy udział ścieżek
1	355,095	±76,5206	84,2	5412,32	±1820,43	78,1
2	61,68	±12,6943	14,6	1182,39	±288,975	17,1
3	4,3125	±1,0098	1,0	256,09	±23,1986	3,7
4	0,5025	±0,366803	0,1	77,08	±8,53902	1,1



Rys. 6.18. Zależność średniej liczby wywłaszczeń *M* od rozmiaru sieci, dla wywłaszczeń wymaganych na więcej niż jednym łączu na drodze połączeniowej (*z*>1). Pominięto algorytm GarGop/BW, który osiągnął dużo gorsze rezultaty (*M*>6).



Rys. 6.19. Zależność średniego indeksu straconego pasma sieciowego b_{NET} od rozmiaru sieci, dla wywłaszczeń wymaganych na więcej niż jednym łączu na drodze połączeniowej (z>1).



Rys. 6.20. Zależność wartości średniej metryki złożonej Q od rozmiaru sieci, dla wywłaszczeń wymaganych na więcej niż jednym łączu na drodze połączeniowej (z>1).

Wniosek 6.5a. Czynnikiem ograniczającym jakość algorytmów globalnych jest mała średnia liczba łączy wymagających wywłaszczania na drodze wybranej dla nowej ścieżki.

Wniosek 6.5b. Dodatkowym czynnikiem ograniczającym jakość algorytmów globalnych są rzadkie przypadki pokrywania się tras ścieżek z łączami wymagającymi wywłaszczania na więcej niż jednym łączu.

6.5. Wnioski

6.5.1. Porównanie jakości wywłaszczania

Badania symulacyjne przeprowadzone dla rzeczywistych topologii sieci o różnych wielkościach i gęstościach wykazały znaczne różnice w jakości poszczególnych algorytmów wywłaszczania. Wykazano przy tym, że w przypadku algorytmów istniejących w różnych wariantach niezbędny jest wybór właściwego z nich, gdyż niektóre z nich mogą prowadzić do rezultatów znacznie odbiegających od średnich wyników pozostałych algorytmów. Co więcej, jeden z badanych algorytmów globalnych, GarGop/BW, wykazał swoją nieprzydatność w badanych warunkach.

Analiza wyników symulacji prowadzi do wniosków, że im większa liczba węzłów sieci, tym większa średnia liczba wywłaszczeń (patrz wniosek 6.1), natomiast wzrost gęstości sieci wpływa na zmniejszenie straconego pasma (patrz wniosek 6.3).

Jeden z najciekawszych wniosków jakie wypływają z badań to odkrycie, że najlepsze algorytmy, takie jak KNow i Pey, cechują się doskonałą uniwersalnością (patrz wniosek 6.2). Osiągają one najlepsze wyniki w zakresie dwóch najważniejszych kryteriów, tj. liczby wywłaszczeń i wywłaszczonego pasma. Dzięki temu nie ma potrzeby dobierania algorytmu i jego parametrów pod kątem wymagań sieci, o ile celem jest minimalizacja liczby wywłaszczeń lub minimalizacja straconego pasma. To z kolei zdecydowanie poprawia komfort pracy z takimi algorytmami w praktycznych realizacjach.

Biorąc pod uwagę wszystkie przeprowadzone badania najlepszym algorytmem jest zaproponowany przez autora niniejszej pracy algorytm globalny KNow. Zarówno w zakresie minimalizacji liczby wywłaszczeń jak i minimalizacji straconego pasma osiągnięte rezultaty są bądź najlepsze bądź równe najlepszemu z pozostałych algorytmów, w granicach przedziału ufności. W porównaniu z najbardziej znanym algorytmem OliSco, algorytm KNow zapewnia zdecydowanie lepsze rezultaty w zakresie liczby wywłaszczeń i straconego pasma. Wyniki pomiarów wskazują na porównywalny z innymi algorytmami czas działania i rzadsze niż w przypadku innych algorytmów przypadki długich czasów wykonania. Te fakty pozwalają stwierdzić, że postawiona we wstępie **teza pracy została uwodniona.**

Analizując pozostałe algorytmy, bardzo dobre wyniki oferuje metoda lokalna Pey, która

została opracowana zanim powstały sieci MPLS i została przez autora zaadoptowana do tej technologii. Jest to realizacja optymalnego algorytmu lokalnego w zakresie liczby wywłaszczeń, który dodatkowo dąży do minimalizacji wywłaszczonego pasma. W porównaniu z tą metodą inne algorytmy lokalne dedykowane dla sieci MPLS wykazały dużo gorszą jakość.

6.5.2. Metody globalne i lokalne

Zgodnie z oczekiwaniem, zastosowanie zaproponowanego algorytmu globalnego spowodowało poprawę jakości wywłaszczania, rozumianej jako zmniejszenie średniej liczby wywłaszczeń i wywłaszczonego pasma, w stosunku do najlepszego z badanych algorytmów lokalnych. Jednocześnie skala tej poprawy, rzędu 10%, nie jest tak duża jak możnaby tego oczekiwać. Wykazano, że przyczyna takiego stanu rzeczy leży w następujących dwóch czynnikach, które ograniczają zysk osiągany przez zastosowanie algorytmów globalnych.

1. Większość wywłaszczeń odbywa się na pojedynczym łączu (porównaj rys. 6.4). To oznacza, że w większości wywłaszczeń nie ma możliwości skorzystania z potencjału jaki mają algorytmy globalne. Pokazano, że po uwzględnieniu tylko tych przypadków, gdy liczba łączy z wywłaszczaniem jest większa od jednego, wyniki nie tylko jednego, ale dwóch algorytmów globalnych, KNow i GarGop/RC, są lepsze od najlepszego algorytmu lokalnego Pey (patrz wniosek 6.5a).
2. Trasy niezależnych od siebie ścieżek rzadko podążają tą samą drogą. W efekcie osłabia się efekt zastosowania algorytmu globalnego, gdyż stosunkowo rzadko występuje korzystna dla wywłaszczania globalnego sytuacja, w której wystarczające jest wywłaszczenie pojedynczej ścieżki obejmującej wszystkie łącza wymagające wywłaszczania (patrz wniosek 6.5b).

Co warte podkreślenia, oba te czynniki wzajemnie wzmacniają negatywny wpływ na jakość algorytmów globalnych. Wynika to z faktu, że drugi czynnik objawia się szczególnie w dużych sieciach, w których nieco mniejsze znaczenie ma czynnik pierwszy, gdyż jak pokazano, im większa sieć, tym większa średnia liczba łączy wymagających wywłaszczenia.

Pomimo tych ograniczeń pokazano, że zaproponowany algorytm KNow radzi sobie dobrze zarówno w małych jak też dużych sieciach i osiąga najlepsze wyniki we wszystkich badanych sieciach, niezależnie od wielkości i gęstości.

6.5.3. Czasy działania

Rzeczywiste czasy działania algorytmów wykazują dużą zależność od rozmiaru sieci. Im większa sieć, tym dłuższe czasy działania, niezależnie od wybranego algorytmu. Rozrzut jednostkowych czasów działania również zwiększa się w przypadku dużych sieci.

W odniesieniu do różnic w czasach działania dla poszczególnych algorytmów, są one stosunkowo niewielkie. Najdłuższe czasy działania wykazują te algorytmy, które osiągają najlepsze wyniki jakościowe. Zatem dłuższy czas działania jest ceną za osiągnięcie lepszych wyników jakościowych (patrz wniosek 6.4).

Spośród najlepszych jakościowo, algorytm Pey jest najszybszy. Z kolei algorytm KNow wykazuje znacznie mniejszy rozrzut jednostkowych czasów niż w przypadku algorytmu Pey. Algorytm KNow kosztem nieco większego średniego czasu wykonania rzadziej niż inne algorytmy wykazuje przypadki szczególnie długich czasów wykonania.

7. PODSUMOWANIE

Celem postawionym we wstępie do niniejszej pracy było porównanie wydajności dostępnych algorytmów wywłaszczania i porównanie ich z wynikami osiąganymi przez zaproponowany algorytm heurystyczny. Postawiono tezę o lepszej efektywności zaproponowanego algorytmu niż ta, jaką oferują inne najbardziej popularne algorytmy. Cele pomocnicze pracy obejmowały:

- dokonanie oceny dostępnych algorytmów wywłaszczania,
- wykonanie opisu zaproponowanego algorytmu,
- przygotowanie środowiska pomiarowego, oraz
- przeprowadzenie badań symulacyjnych i analizy wyników.

W ramach przeprowadzonych badań nad algorytmami wywłaszczania udało się osiągnąć wszystkie zamierzone cele, a w szczególności wykonano następujące zadania.

1. Udowodniono postawioną tezę w oparciu o wyniki badań symulacyjnych.

Wyniki badań wskazują na wysoką jakość opracowanego przez autora globalnego algorytmu wywłaszczania KNow. Pokazano, iż użycie zaproponowanego algorytmu pozwala uzyskać wyniki lepsze od uzyskiwanych z użyciem pozostałych najlepszych lokalnych i globalnych algorytmów heurystycznych. Badania oparto na dwóch najważniejszych kryteriach wyboru kandydatów, tj. średniej liczby wywłaszczeń i średniego wywłaszczonego pasma. Stwierdzono, że poprawa jakości w stosunku do najlepszego z pozostałych algorytmów w większości przypadków nie przekracza 10%. W porównaniu z najpopularniejszym algorytmem dedykowanym dla sieci MPLS, w wariantach OliSco/RC i OliSco/BW, zaproponowany algorytm zapewnia około 20% mniej wywłaszczeń i około 40% poprawę indeksu wywłaszczonego pasma.

Zaproponowany przez autora algorytm wydaje się być dobrą alternatywą dla innych algorytmów heurystycznych, gdyż spełnia cztery pożądane cechy dobrego algorytmu, tj. ma zasięg globalny, pozwala na zadanie kryterium wyboru kandydatów, posiada typową i akceptowalną złożoność obliczeniową oraz wykazuje się wysoką jakością wyboru kandydatów, zwykle lepszą od wszystkich pozostałych algorytmów poddanych badaniom.

Badania potwierdzające dobre wyniki algorytmu przeprowadzono z użyciem kilkunastu różnych topologii, włączając w to dostępne topologie sieci operatorów telekomunikacyjnych jak i topologie regularne. Wyboru topologii dokonano w ten sposób, aby zweryfikować wpływ różnego rozmiaru oraz różnej gęstości sieci.

2. Samodzielnie wykonano narzędzia symulacyjne i poprawnie je zweryfikowano.

Wszystkie badania przeprowadzono używając programu *msim*, zaprojektowanego i zaimplementowanego od podstaw przez autora. Jest to otwarty na rozszerzenia symulator ukierunkowany na prowadzenie badań mechanizmów inżynierii ruchu w sieciach MPLS. Jego najważniejsze cechy są następujące:

- budowa obiektowa,
- możliwość kompilacji w systemach Windows i Linux,
- możliwość prowadzenia symulacji na poziomie połączeń lub pakietów,
- automatyzacja serii symulacji oraz zintegrowane przetwarzanie wyników.

Program *msim*, rozwijany przez kilka lat, podlegał ewolucji i udoskonaleniom, dzięki którym stał się narzędziem bardzo użytecznym i realizującym wszystkie zamierzone rodzaje badań. W efekcie długotrwałego procesu testowania i poprawiania programu osiągnięto wysoki poziom zaufania do wyników generowanych przez program.

Wykonano także niezmiernie pomocną aplikację *Vims*, umożliwiającą łatwe tworzenie i modyfikowanie badanych sieci, automatyczne generowanie plików konfiguracyjnych dla symulatora, a także wczytywanie plików topologii dostępnych w innych formatach i generowanie plików graficznych z mapami topologii.

3. Odkryto interesujące zjawiska związane z wywłaszczaniem.

W efekcie analizy wyników badań symulacyjnych odkryto kilka zaskakujących lub nieoczywistych wniosków:

- można by oczekiwać, że dwa najważniejsze kryteria wyboru kandydatów, tj. minimalizacja liczby wywłaszczeń i minimalizacja ilości wywłaszczonego pasma są wzajemnie sprzeczne, tzn. jedno uzyskuje się kosztem drugiego; tymczasem wyniki badań wskazują na to, że najlepsze algorytmy pozwalają na uzyskanie dobrych wyników jednocześnie dla obu wymienionych kryteriów,
- przewaga najlepszej metody globalnej KNow, operującej na całej drodze połączeniowej jednocześnie, nad najlepszą metodą lokalną Pey, obejmującą działaniem tylko pojedyncze łącze, nie jest tak duża, jak można by tego oczekiwać; powodem jest to, że zwykle konieczność wywłaszczania występuje na pojedynczym łączu, a drogi różnych ścieżek rzadko pokrywają się na więcej niż jednym łączu,
- nie zweryfikowano pozytywnie argumentu o tym, że przewaga algorytmów lokalnych wywłaszczania bierze się ze zdecentralizowanego charakteru sieci IP/MPLS i trudności implementacyjnych algorytmów globalnych; odkryto, iż przyczyna atrakcyjności algorytmów lokalnych nie może być raczej wynikiem niedostatecznego zysku z dodatkowych nakładów poniesionych na implementację algorytmów globalnych; siła algorytmów globalnych poza lepszą jakością wywłaszczania leży w możliwości zaimplementowani jako metoda scentralizowana w oddzielnym urządzeniu w sieci, pozwalając na odciążenie procesorów ruterów i wykorzystanie zaawansowanych kryteriów wyboru ścieżek przeznaczonych do wywłaszczania.

4. Dokonano porównania kilku niezależnych algorytmów heurystycznych.

Praca jest jedynym znanym zestawieniem czterech niezależnych algorytmów wywłaszczania, zawierającym opis zasady działania poszczególnych algorytmów i wyniki badań symulacyjnych. Najważniejsze osiągnięcia w tym zakresie to:

- ujednolicenie opisu algorytmów, obejmujące użyte symbole i kod źródłowy,
- opracowanie metodologii określania na potrzeby badań symulacyjnych warunków ruchowych w sieciach o dowolnej strukturze,
- dokonanie niezależnej analizy złożoności obliczeniowej,
- prezentacja wyników pomiarów czasów działania algorytmów w postaci histogramów,
- dostosowanie algorytmów GarGop i Pey do pracy w technologii MPLS,
- udoskonalenie algorytmu Pey w zakresie złożoności obliczeniowej,
- zaproponowanie przykładowego algorytmu optymalnego o zasięgu domeny.

Przeprowadzone badania wypełniają zatem lukę, jaką dotąd był brak ogólnie dostępnych wyników porównawczych dostępnych algorytmów wywłaszczania. Przedstawione w niniejszej pracy rezultaty umożliwiły weryfikację jakości algorytmów w realnych scenariuszach sieciowych. Wyniki uzyskane dla różnych topologii i warunków ruchowych wykazują powtarzalność i pozwalają na stawianie wniosków obarczonych niewielkim ryzykiem błędów.

Rozszerzenie badań na algorytmy, które są starsze niż technologia MPLS, umożliwiło skorzystanie z doświadczenia naukowców badających zjawisko wywłaszczania na długo przed pojawieniem się sieci MPLS. W szczególności jeden z algorytmów, Pey, opracowany dla sieci ATM, okazał się być algorytmem lepszym od wszystkich badanych algorytmów lokalnych dedykowanych dla sieci MPLS.

Można mieć nadzieję, że niniejsze opracowanie ułatwi prowadzenie niezależnych badań porównawczych i doprowadzi do poszerzenia listy publikacji w zakresie wywłaszczania w sieciach MPLS. Sprzyja temu ujednolicony przegląd istniejących algorytmów, zbiór definicji dotyczących wywłaszczania a także opracowany przez autora sposób określania warunków ruchowych w dowolnej sieci (patrz załącznik A).

Pomimo wielu ciekawych wniosków jakie przyniosły badania, temat wywłaszczania pozostawia jeszcze wiele otwartych zagadnień. Jednym z nich jest opisane w rozdziale 3.4.6 zjawisko kaskady wywłaszczeń i jego znaczenie w praktycznych realizacjach. Niezmiernie istotny jest związany z tym problem efektu wielokrotnego wywłaszczania ścieżek o niskim priorytecie, nawet w obrębie pojedynczej kaskady wywłaszczeń. Dalszych badań wymaga opracowanie metod pozwalających na ograniczenie takich niekorzystnych zjawisk.

Innym kierunkiem dalszych badań jest poszerzenie obszaru badań na technologie MPLS-TP i Generalized MPLS (GMPLS). W pierwszej z nich sieci MPLS ewoluują w kierunku operatorskich sieci transportowych, natomiast w drugiej protokoły i algorytmy opracowane dla sieci MPLS są wykorzystywane do nowoczesnego sterowania ruchem w sieciach opartych na technologiach SDH [17], ASON/DWDM [79,80] lub OTN [87,89]. Jest to również długofalowy kierunek dalszego rozwoju programu symulacyjnego.

BIBLIOGRAFIA

[1] Agarwal P., Akyol B., „Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks", RFC 3443, January 2003.

[2] Ahmad I., Kamruzzaman J., Aswathanarayaniah S., „Revenue Aware Preemption Policy in Multimedia Communication Networks", w: *Proc. of IEEE International Conference on Multimedia and Expo (ICME) 2005*, Amsterdam, pp. 1374-1377.

[3] Alvarez S., „QoS for IP/MPLS Networks", Cisco Press, 2006, pp. 59.

[4] Andersson L. Asati R., „Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, February 2009.

[5] Andersson L., ed., Minei I., ed., Thomas B., ed., „LDP Specification", RFC 5036, October 2007.

[6] Andersson L., Swallow G., „The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols", RFC 3468, February 2003.

[7] Awduche D., Berger L., Gan D., Li T., Srinivasan V., Swallow G., „RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[8] Awduche D., Hannan A., Xiao X., „Applicability Statement for Extensions to RSVP for LSP-Tunnels", RFC 3210, December 2001.

[9] Adwuche D., Malcolm J., Agogbua J., O'Dell M., McManus J., „Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

[10] Ahmad I., Kamruzzaman J., Aswathanarayaniah S., „Preemption-Aware Routing for QoS-Enabled Networks", w: *Proc. of IEEE GLOBECOM 2005*, pp. 715-720.

[11] Ash J., „Max Allocation with Reservation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering & Performance Comparisons", RFC 4126, June 2005.

[12] Ashwood-Smith P., ed., Berger L., ed., „Generalized Multi-Protocol Label Switching (GMPLS) Signaling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions", RFC 3472, January 2003.

[13] Bar-Noy A., Canetti R., Kutten S., Mansour Y., Schieber B., „Bandwidth allocation with preemption", w: *Proc. of the 27th ACM Symposium on Theory of Computing*, 1995, pp. 616-625.

[14] Berger L., ed., „Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[15] Berger L., Gan D., Swallow G., Pan P., Tommasi F., Molendini S., „RSVP Refresh Overhead Reduction Extensions", RFC 2961, April 2001.

[16] Berger L., Takacs A., Caviglia D., Fedyk D., Meuric J., „GMPLS Asymmetric Bandwidth Bidirectional Label Switched Paths (LSPs)", RFC 5467, March 2009.

[17] Bernstein G., Mannie E., Sharma V., Gray E., „Framework for Generalized Multi-Protocol Label Switching (GMPLS)-based Control of Synchronous Digital Hierarchy/Synchronous Optical Networking (SDH/SONET) Networks", RFC 4257, December 2005.

[18] Biblioteka topologii SNDlib, Zuse-Institut Berlin, http://sndlib.zib.de/.

[19] Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W., „An Architecture for Differentiated Service", RFC 2475, December 1998.

[20] Blanchy F., Melon L., Leduc G., „Routing in a MPLS network featuring preemption mechanisms", w: *Proc. of ICT 2003*, 2003.

[21] Bocci M., ed., Bryant S., ed., Frost D., ed., Levrau L., Berger L., „A Framework for MPLS in Transport Networks", RFC 5921, July 2010.

[22] Braden R., ed., Zhang L., Berson S., Herzog S., Jamin S., „Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[23] Bradner S., „The Internet Standards Process - Revision 3", BCP 9, RFC 2026, October 1996.

[24] Bryant S., ed., Morrow M., Swallow G., Cherukuri R., Nadeau T., Harrison N., Niven-Jenkins B., „Application of Ethernet Pseudowires to MPLS Transport Networks", RFC 5994, October 2010.

[25] Bryant S., ed., Pate P., ed., „Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[26] Chaieb I., Le Roux J.-L., Cousin B., „A New Pre-emption Policy For MPLS-TE Networks", w: *Proc. of IEEE International Conference on Networks (ICN) 2007*, pp. 394-399.

[27] Chaieb I., Le Roux J.-l., Cousin B., „Improved MPLS-TE LSP path computation using preemption", w: *Proc. of IEEE Global Information Infrastructure Symposium (GIIS)*, Marrakesz, Maroko, 2007, pp. 218-221.

[28] Cormen T. H., Leiserson Ch. E., Rivest R. L., Stein C., „Wprowadzenie do algorytmów", Wydawnictwa Naukowo-Techniczne, Warszawa 2004.

[29] Czarkowski M., Kaczmarek S., „Simulation model for evaluation of packet sequence changed order of stream in DiffServ network", w: *Advances in Electronics and Telecommunications*, vol. 1, no. 2 (2010), pp. 30-34.

[30] Davie B. S., Rekhter Y., „MPLS: Technology and Applications", Morgan Kaufmann, 2000.

[31] Din N. M., Abidin H. Z., Rahman S. F. A., Fisal N., „A Fuzzy LSP Regulator for Preemption Control in a DiffServ-Aware MPLS Internet", w: *Proc. of IEEE International Conference on Networks (ICN) 2005*, pp. 415-420.

[32] Dogar F. R., Uzmi Z. A., Baqai S. M., „PFR: A Distributed Preemption Strategy for Improved QoS in Multi-Class Networks", w: *Workshop on Distributed Autonomous Network Management Systems (DANMS), co located with IEEE ICAC 2006*, Dublin, Ireland, 2006.

[33] Farrel A., Bryskin I., „GMPLS: Architecture and Applications", The Morgan Kaufmann Series in Networking, Elsevier, 2006.

[34] Farrel A., Vasseur J.-P., Ash J., „A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[35] Garay J. A., Gopal I. S., „Call preemption in communication networks", w: *Proc. of INFOCOM '92*, pp. 1043-1050, Florencja 1992.

[36] Garrett A., „JUNOS cookbook", O'Reilly Media, Inc., 2006, pp. 506.

[37] Gromov G., „Roads and Crossroads of the Internet History”, http://www.netvalley.com/cgi-bin/intval/net_history.pl?chapter=8, 05/06/2011.

[38] Grover W. D., „Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking”, Prentice Hall Professional, 2003, pp. 371-376.

[39] Hartmann A. K., „A Practical Guide To Computer Simulation”, World Scientific Publishing Company, 2009.

[40] Hau L. Ch., Soong B.-H., Bose S. K., „A New Preemption Strategy with Re-routing Control for MPLS Networks”, w: *Proc. of 2005 Fifth International Conference on Information, Communications and Signal Processing*, Bangkok, pp. 1515-1519.

[41] Herzog S., „Signaled Preemption Priority Policy Element”, RFC 2751, January 2000.

[42] IETF MPLS Working Group (Grupa Robocza MPLS), http://datatracker.ietf.org/wg/mpls/charter/

[43] Jamoussi B., ed. i in., „Constraint-Based LSP Setup using LDP”, RFC 3212, January 2002.

[44] Kaczmarek S., Nowak K., „A New Heuristic Algorithm for Effective Preemption in MPLS Networks”, w: *Workshop on High Performance Switching and Routing*, Poznań 2006, pp. 337-342, doi:10.1109/HPSR.2006.1709731.

[45] Kaczmarek S., Nowak K., „A simulation tool for traffic engineering methods and QoS evaluation of MPLS Networks”, w: *SIMUTools'09: Proceedings of 2nd International Conference on Simulation Tools and Techniques*, Rome 2009, doi:10.4108/ICST.SIMUTOOLS2009.5689.

[46] Kaczmarek S., Nowak K., „Comparison of centralized and decentralized preemption in MPLS Networks”, w: *Proceedings of 14th Polish Teletraffic Symposium*, Zakopane, 2007, pp. 259-268.

[47] Kaczmarek S., Nowak K., „Model symulacyjny i badania sieci MPLS”, w: *Zeszyty Naukowe Wydz. ETI, Technologie Informacyjne*, Gdańsk 2004, nr 4, ss. 487-494.

[48] Kaczmarek S., Nowak K., „Model symulacyjny sterowania przyjęciem zgłoszenia (CAC) w sieci ATM”, w: *Referaty, Poznańskie Warsztaty Telekomunikacyjne PWT 2001*, Poznań 2001.

[49] Kaczmarek S., Nowak K., „Ocena wpływu metody kierowania ruchem i topologii na jakość usług w sieci MPLS”, w: *Materiały Konferencyjne. XX Krajowe Sympozjum Telekomunikacji`2004*, Bydgoszcz 2004, tom B, ss. 31-37.

[50] Kaczmarek S., Nowak K., „Performance Evaluation of Preemption Algorithms in MPLS Networks”, w: *JET International Journal of Electronics and Telecommunications*, 2011, vol. 57, no. 2, pp. 169-175, http://ijet.czasopisma.pan.pl/38-international-journal-of-electronics-and-telecommunications/wydania/70-no-2--2011.

[51] Kaczmarek S., Nowak K., „Performance of LSP preemption methods in different MPLS networks”, w: *Proceedings of the 5th Polish-German Teletraffic Symposium PGTS`08*, Berlin 2008, pp. 195-204.

[52] Kaczmarek S., Nowak K., „Symulacja pracy sieci MPLS”, w: *Materiały konferencyjne. V Ogólnopolska Konferencja „INTERNET - Wrocław 2003*, Wrocław 2003, ss. 102-111.

[53] Kaczmarek S, red., Czarkowski M., Młynarczyk M., Narloch M., Nowak K., w: „Opracowanie struktury funkcjonalnej dla warstwy serwerów sterowania połączeniem. Określenie protokołów komunikacji pomiędzy serwerami warstwy sterowania połączeniem oraz między tą warstwą a warstwami przyległymi", rap. 2.3, PBZ-MNiSW-02-II/2007, Politechnika Gdańska, Gdańsk 2009.

[54] Kaczmarek S., red., Młynarczyk M., Nowak K., w: „Usługi i sieci teleinformatyczne następnej generacji – aspekty techniczne, aplikacyjne i rynkowe", rap. 2.4, w: *Architektury i protokoły sieciowe*, PBZ-MNiSW-02-II/2007, Politechnika Gdańska, Gdańsk 2010.

[55] Katz D., Kompella K., Yeung D., „Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.

[56] Ke Y., Lin Z., Hui-min Z., „A Preemption-aware Path Selection Algorithm for DiffServ/MPLS Networks", w: *Proc. of IEEE Workshop on IP Operations and Management*, 2004, pp. 129-133.

[57] Kompella K., ed., Rekhter Y., ed., „Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, October 2005.

[58] Kompella K., Rekhter Y., „Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

[59] Kubale M., „Introduction to Computational Complexity and Algorithmic Graph Coloring", Gdańskie Towarzystwo Naukowe, Gdańsk 1998.

[60] Kubale M., „Łagodne wprowadzenie do analizy algorytmów", Politechnika Gdańska, Gdańsk 2004.

[61] Law A., „Simulation Modeling and Analysis", Hcgraw Hill Higher Education, 2006.

[62] Le Faucheur F., ed., „Russian Dolls Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering", RFC 4127, June 2005.

[63] Le Faucheur F., Lai W., „Maximum Allocation Bandwidth Constraints Model for Diffserv-aware MPLS Traffic Engineering", RFC 4125, June 2005.

[64] Mannie E., ed., „Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, October 2004.

[65] Meyer M., ed., Vasseur J. P., ed., „MPLS Traffic Engineering Soft Preemption", RFC 5712, January 2010.

[66] Minei I., Lucek J., „MPLS-Enabled Applications: Emerging Developments and New Technologies", Wiley Series on Communications Networking & Distributed Systems, John Willey and Sons, Ltd, 2011.

[67] Niven-Jenkins B., ed., Brungard D., ed., Betts M., ed., Sprecher N., Ueno S., „Requirements of an MPLS Transport Profile", RFC 5654, September 2009.

[68] Nowak K., „Sterowanie przyjęciem zgłoszenia (CAC) w sieci ATM", praca dyplomowa, Politechnika Gdańska, Gdańsk 2000.

[69] Ohta H., „Assignment of the 'OAM Alert Label' for Multiprotocol Label Switching Architecture (MPLS) Operation and Maintenance (OAM) Functions", RFC 3429, November 2002.

[70] Oliveira J. C. de, ed., Vasseur J. P., ed., Chen L., Scoglio C., „Label Switched Path (LSP) Preemption Policies for MPLS Traffic Engineering", RFC 4829, April 2007.

[71] Oliveira J. C. de, Scoglio C., Akyildiz I. F., Smith J. A., Uhl G., „A New Topology-Aware LSP Preemption Policy for Diffserv-Based MPLS Networks", w: *Proc. of IEEE NETWORKS 2002*, pp. 283-294, Atlanta, 2002.

[72] Oliveira J. C. de, Scoglio C., Akyildiz I. F., Uhl G., „A New Preemption Policy for DiffServ-Aware Traffic Engineering to Minimize Rerouting", w: *Proceedings of IEEE INFOCOM 2002*, vol. 2, pp. 695-704, June 2002.

[73] Oliveira J. C. de, Scoglio C., Akyildiz I. F., Uhl G., „New preemption policies for DiffServ-aware traffic engineering to minimize rerouting in MPLS networks", *IEEE/ACM Trans. Networking*, vol. 12, no. 4, August 2004, pp. 733-745.

[74] Oliveira J. C. de, z prywatnej korespondencji (e-mail) z autorem.

[75] Oryginalne uzasadnienie dla powstania grupy roboczej "mpls", http://www.ietf.org/proceedings/97apr/97apr-final/xrtftr90.htm

[76] Osborne E., Simha A., „Traffic Engineering with MPLS", Cisco Press, 2002.

[77] Ould-Brahim H., Fedyk D., Rekhter Y., „BGP Traffic Engineering Attribute", RFC 5543, May 2009.

[78] Papadimitriou Ch. H., „Złożoność obliczeniowa", z ang. „Computational Complexity", Wydawnictwa Naukowo-Techniczne, Warszawa 2002.

[79] Papadimitriou D., Drake J., Ash J., Farrel A., Ong L., „Requirements for Generalized MPLS (GMPLS) Signaling Usage and Extensions for Automatically Switched Optical Network (ASON)", RFC 4139, July 2005.

[80] Papadimitriou D., ed., „Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control", RFC 4328, January 2006.

[81] Perrot H. G., „An Introduction to ATM Networks", Willey, 2001.

[82] Peyravian M., „Providing Different Levels of Network Availability in High-speed Networks", w: *Proc. of GLOBECOM '94*, 1994, pp. 941–945.

[83] Peyravian M., Kshemkalyani A. D., „Connection Preemption: Issues, Algorithms, and a Simulation Study", w: *Proc. of INFOCOM 1997*, pp. 143-151.

[84] Poretsky S., „Connection Precedence and Preemption in Military Asynchronous Transfer Mode (ATM) Networks", w: *Proc. of MILCOM '98*, 1998, pp. 86-90.

[85] Postel J., „Internet Protocol", RFC 791, September 1981.

[86] Projekt OMNeT++, http://www.omnetpp.org/.

[87] Recommendation ITU-T, G.709/Y.1331, „Interfaces for the Optical Transport Network", 12/2009.

[88] Recommendation ITU-T, G.8110.1/Y.1370.1, „Architecture of Transport MPLS (T-MPLS) layer network", 11/2006.

[89] Recommendation ITU-T, G.872, „Architecture of optical transport networks", 11/2001.

[90] Recommendation ITU-T, I.150, „B-ISDN asynchronous transfer mode functional characteristics”, 02/1999.

[91] Recommendation ITU-T, I.255.3, „Multi-Level Precedence and Preemption Service (MLPP)”, 07/1990.

[92] Recommendation ITU-T, Y.1711, „Operation & Maintenance mechanism for MPLS networks”, 02/2004.

[93] Rekhter Y., Davie B., Katz D., Rosen E., Swallow G., „Cisco Systems' Tag Switching Architecture Overview”, RFC 2105, February 1997.

[94] Rosen E., „Removing a Restriction on the use of MPLS Explicit NULL”, RFC 4182, September 2005.

[95] Rosen E., Tappan D., Fedorkow G., Rekhter Y., Farinacci D., Li T., Conta A., „MPLS Label Stack Encoding”, RFC 3032, January 2001.

[96] Rosen E., Viswanathan A., Callon R., „Multiprotocol Label Switching Architecture”, RFC 3031, January 2001.

[97] Shan T., Yang O. W. W., „Bandwidth Preemption Algorithms for Differentiated Service aware Traffic Engineering”, w: *Proc. of IEEE Globecom 2005*, pp. 536-539.

[98] Smit H., Li T., „Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE)”, RFC 3784, June 2004.

[99] Smith D., Mullooly J., Jaeger W., Scholl T., „Label Edge Router Forwarding of IPv4 Option Packets”, RFC 6178, March 2011.

[100] Stanisic V., Devetsikiotis M., „A Dynamic Study of Providing Quality of Service Using Preemption Policies with Random Selection”, w: *Proc. of IEEE International Conference on Communications (ICC) 2003*, vol. 3, pp. 1543-1546.

[101] Stanisic V., Devetsikiotis M., „A Framework for Providing Quality of Service Using Preemption Policies”, CACC Technical report, November 2002.

[102] Szviatovszki B., Szentesi A., Jüttner A., „Minimizing Re-Routing in MPLS Networks with Preemption-Aware Constraint-Based Routing”, *Spec. issue of the Computer Communications Journal on Advances in Performance Evaluation of Computer and Telecommunications Networking*, vol. 25(11-12), May 2003, pp. 1076-1084.

[103] The Network Simulator – ns-2, http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[104] Vieira R. C., Guardieiro P. R., „A Proposal and Evaluation of a LSP Preemption Policy Implemented with Fuzzy Logic and Genetic Algorithms in a DiffServ/MPLS Test-bed”, w: *Proc. of International Conference on Communications (ICC) 2005*, vol. 1, pp. 109-114.

[105] Weber S., Oliveira J.C. de, Dasgupta S., Willman B., Zhao Z., „Combined Preemption and Adaptation in Next Generation Multiservice Networks”, w: *Proc. of IEEE International Conference on Communications (ICC) 2006*, pp. 670-675.

[106] Xiao X., ed., McPherson D., ed., Pate P., ed., „Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)”, RFC 3916, September 2004.

[107] Zegura E. W., Calvert K. L., Donahoo M. J., A Quantitative Comparison of Graph-based Models for Internet Topology", w: *IEEE/ACM Transactions on Networking*, vol. 3, issue 6, December 1997, pp. 770-783.

[108] Zhu M., He X., Ye W., „Minimizing Preemption Cost for Diffserv-Aware MPLS Traffic Engineering", w: *Proc. of First International Conference on Communications and Networking in China*, 2006, pp. 1-5.

SPIS SYMBOLI I OZNACZEŃ

A	tablica pasm dostępnych na poszczególnych łączach przy uwzględnieniu wywłaszczania
$A^{(e)}$	tablica pasm dostępnych na łączu *e* dla ścieżek o poszczególnych priorytetach
A_f	tablica ilości wolnego pasma na poszczególnych łączach
$a_f^{(e)}$	wolne pasmo na łączu *e*
A_{oc}	tablica pasm zarezerwowanych dla ścieżek na poszczególnych łączach
$a_{oc}^{(e)}$	suma pasm zarezerwowanych dla ścieżek na łączu *e*
$a_s^{(e)}$	pasmo dostępne dla ścieżek o priorytecie utworzenia *s* na łączu *e*
B	wywłaszczone pasmo
B_b	globalny (dla wszystkich łączy) poziom rezerwacji pasma
$b_b^{(e)}$	poziom rezerwacji pasma na łączu *e*
B_{hold}	tablica pasm zarezerwowanych dla ścieżek o poszczególnych priorytetach
b_{LOC}	indeks straconego pasma lokalnego (odwrotność względnego straconego pasma lokalnego)
B_n	wywłaszczone pasmo sieciowe
b_{NET}	indeks straconego pasma sieciowego (odwrotność względnego straconego pasma sieciowego)
B_o	tablica pasm poszukiwanych na poszczególnych łączach
$B_o^{(e)}$	pasmo poszukiwane na łączu *e*
b_p	pasmo zarezerwowane dla ścieżki *p*
$b_p^{(e)}$	pasmo zarezerwowane dla ścieżki *p* na łączu *e*
B_{pr}	tablica pasm, jakie mogą być wywłaszczone przez ścieżki o poszczególnych priorytetach
B_r	tablica wymaganych pasm na poszczególnych łączach
$B_r^{(e)}$	wymagane pasmo na łączu *e*
B_{rnet}	wymagane pasmo sieciowe (suma pasm wymaganych na kolejnych łączach)
B_{tot}	sumaryczne pasmo jakie może być zarezerwowane w całej sieci
B_x	tablica nadmiarowych (straconych) pasm na kolejnych łączach
$B_x^{(e)}$	nadmiarowe (stracone) pasmo na łączu *e*
$b_x^{(e)}$	względne nadmiarowe (stracone) pasmo na łączu *e*
B_{xloc}	nadmiarowe (stracone) pasmo lokalne (suma pasm nadmiarowych na łączach na drodze połączeniowej, na których brakuje wolnego pasma)
b_{xloc}	względne nadmiarowe (stracone) pasmo lokalne
B_{xnet}	nadmiarowe (stracone) pasmo sieciowe (suma pasm nadmiarowych na wszystkich łączach należących do drogi połączeniowej)
b_{xnet}	względne nadmiarowe (stracone) pasmo sieciowe

b_n	średnie pasmo zarezerwowane dla ścieżek należących do *n*-tej klasy
C	tablica pasm (przepływności) łączy
c	pasmo (przepływność) łącza (przy jednakowych pasmach łączy)
c_e	pasmo (przepływność) łącza *e*
c_n	średnie pasmo (przepływność) rezerwowane dla źródeł należących do *n*-tej klasy
C_{net}	sumaryczna ilość pasma (przepływności) jaką może być obciążona sieć
$\boldsymbol{D^{(p)}}$	zbiór ścieżek w kaskadzie wywłaszczeń spowodowanej przez ścieżkę *p*
d	gęstość sieci
E	zbiór łączy jednokierunkowych w badanej sieci
e	łącze
$e_p^{(n)}$	*n*-te łącze, na którym utworzona jest ścieżka *p*
$\hat{e}_n$	*n*-te łącze na drodze ścieżki, na którym brakuje wolnego pasma
$g^{(p)}$	długość kaskady wywłaszczeń spowodowanej przez ścieżkę *p*
$H^{(p)}$	liczba ścieżek w kaskadzie spowodowanej przez ścieżkę *p*
h	priorytet utrzymania ścieżki
h_p	priorytet utrzymania ścieżki *p*
$\boldsymbol{K}$	zbiór ścieżek-potencjalnych kandydatów do wywłaszczenia na całej wybranej drodze
k_{cr}	liczba utworzonych ścieżek
k_{crpr}	liczba ścieżek, których utworzenie spowodowało wywłaszczenie
$\boldsymbol{K_e}$	zbiór ścieżek-potencjalnych kandydatów do wywłaszczenia na łączu *e*
k_{rej}	liczba odrzuconych żądań utworzenia ścieżek
k_{req}	liczba żądań utworzenia ścieżek
L	liczba łączy jednokierunkowych w sieci
l_p	długość ścieżki *p* (liczba łączy)
M	liczba wywłaszczeń (ścieżek przeznaczonych do wywłaszczenia)
m	metryka ścieżki w algorytmie OliSco
$m^{(p)}$	poziom wywłaszczenia ścieżki *p* (w kaskadzie lub łańcuchu)
N	liczba węzłów sieci
n	liczba próbek (dot. histogramów czasów wykonania)
$\boldsymbol{P}$	zbiór ścieżek utworzonych w sieci
P_p	cztery wielkości (R_p, b_p, s_p, h_p) opisujące ścieżkę *p*
p_{pre}	prawdopodobieństwo spowodowania wywłaszczenia przez ścieżkę
p_{rej}	prawdopodobieństwo odrzucenia żądania utworzenia ścieżki
Q	metryka złożona (jakości algorytmów wywłaszczania)
q_n	względny udział *n*-tej klasy w generowanym ruchu
$\hat{R}$	lista łączy, na których wolne pasmo jest mniejsze od pasma danej ścieżki
$\bar{r}$	średnia długość drogi w sieci

R_p	lista łączy, na których utworzona jest ścieżka p
S	tablica bilansów pasma na łączach na wybranej drodze
s	priorytet utworzenia ścieżki
S_e	bilans pasma na łączu e
S_g	globalny bilans pasma
s_p	priorytet utworzenia ścieżki p
t	zmierzony czas wykonania algorytmu
t_n	średnia długość trwania połączeń należących do n-tej klasy
$\boldsymbol{V}$	zbiór węzłów badanej sieci
v	węzeł
$\boldsymbol{W}$	zbiór ścieżek-kandydatów do wywłaszczenia
w_i	i-ta ścieżka-kandydat do wywłaszczenia
$\boldsymbol{X}^{(p)}$	zbiór ścieżek wywłaszczonych bezpośrednio przez ścieżkę p
Z	liczba ścieżek w sieci
z	liczba łączy na drodze połączeniowej, na których wymagane jest wywłaszczanie
λ_n	intensywność żądań utworzenia ścieżek należących do n-tej klasy
ρ_n	natężenie ruchu dla ścieżek należących do n-tej klasy

WYKAZ AKRONIMÓW

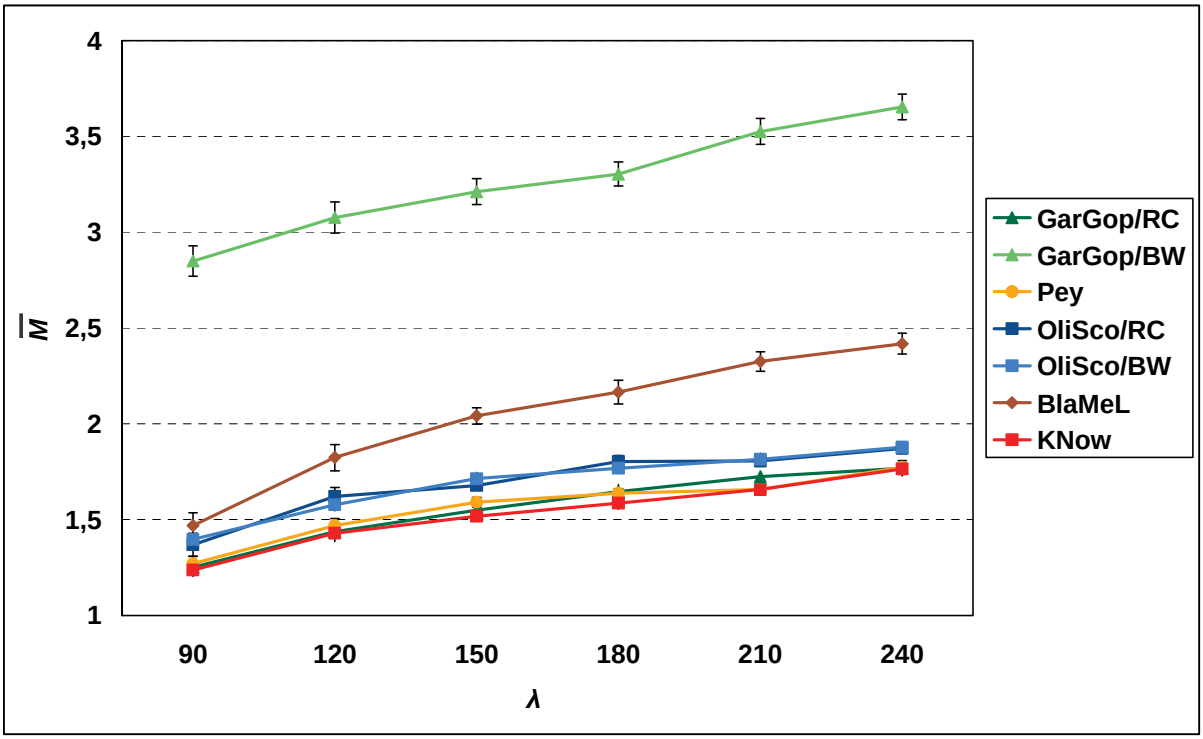
AF	Assured Forwarding Per-Hop Behavior
ATM	Asynchronous Transfer Mode
ARIS	Aggregate Router-based IP Switching
ASON	Automatically Switched Optical Network
BGP-TE	Border Gateway Protocol with Traffic Engineering extensions
BW	Bandwidth
CoS	Class of Service
CR-LDP	Constraint-based Routing Label Distribution Protocol
CSPF	Constrained Shortest Path First
CSR	Cell Switching Router
DLCI	Data Link Connection Identifier
DWDM	Dense Wavelength-Division Multiplexing
ERO	Explicit Route Object
FEC	Forwarding Equivalence Class
FIFO	First In First Out
FR	Frame Relay
FTN	FEC-To-NHLFE; Forwarding Equivalence Class To Next Hop Label Forwarding Entry
GMPLS	Generalized Multiprotocol Label Switching
GNU	GNU's Not Unix
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IETF	Internet Engineering Task Force
ILM	Incoming Label Map
IS-IS	Intermediate System To Intermediate System
IS-IS-TE	Intermediate System To Intermediate System with Traffic Engineering extensions
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switched Patch
LSR	Label Switching Router
MAM	Maximum Allocation (Bandwidth Constraints) Model
MPLS	Multiprotocol Label Switching
MPLS-TP	Multiprotocol Label Switching – Transport Profile
MTU	Maximum Transfer Unit

NHLFE	Next Hop Label Forwarding Entry
NP	Non-Polynomial
OSPF	Open Shortest Path First
OSPF-TE	Open Shortest Path First with Traffic Engineering extensions
OTN	Optical Transport Network
PCE	Path Computation Element
PPP	Point-to-Point Protocol
PWE3	Pseudo Wire Emaulation Edge to Edge
QoS	Quality Of Service
RC	Relocation Count
RDM	Russian Dolls (Bandwidth Constraints) Model
RFC	Request For Comments
RIP	Routing Information Protocol
RRO	Record Route Object
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol with Traffic Engineering extensions
SDH	Synchronous Digital Hierarchy
TC	Traffic Class
TCP	Transmission Control Protocol
TDP	Tag Distribution Protocol
TE	Traffic Engineering
TTL	Time To Live
UDP	User Datagram Protocol
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
WFQ	Weighted Fair Queuing
XML	Extensible Markup Language
ZIB	Zuse-Institut Berlin

ZAŁĄCZNIK A. METODA WYZNACZANIA ILOŚCI RUCHU W SIECI

Jedno z pytań, na jakie należało odpowiedzieć przy dobieraniu warunków badań było to, czy ilość ruchu wprowadzanego do sieci, która przekłada się na częstość występowania wywłaszczeń, ma wpływ na uzyskane wyniki. Jeśli tak jest, wówczas pojawia się problem istnienia dodatkowej zmiennej, jaką należy brać pod uwagę przy interpretacji wyników badań. To może utrudniać prowadzenie badań i stawiać pod znakiem zapytania ich wiarygodność.

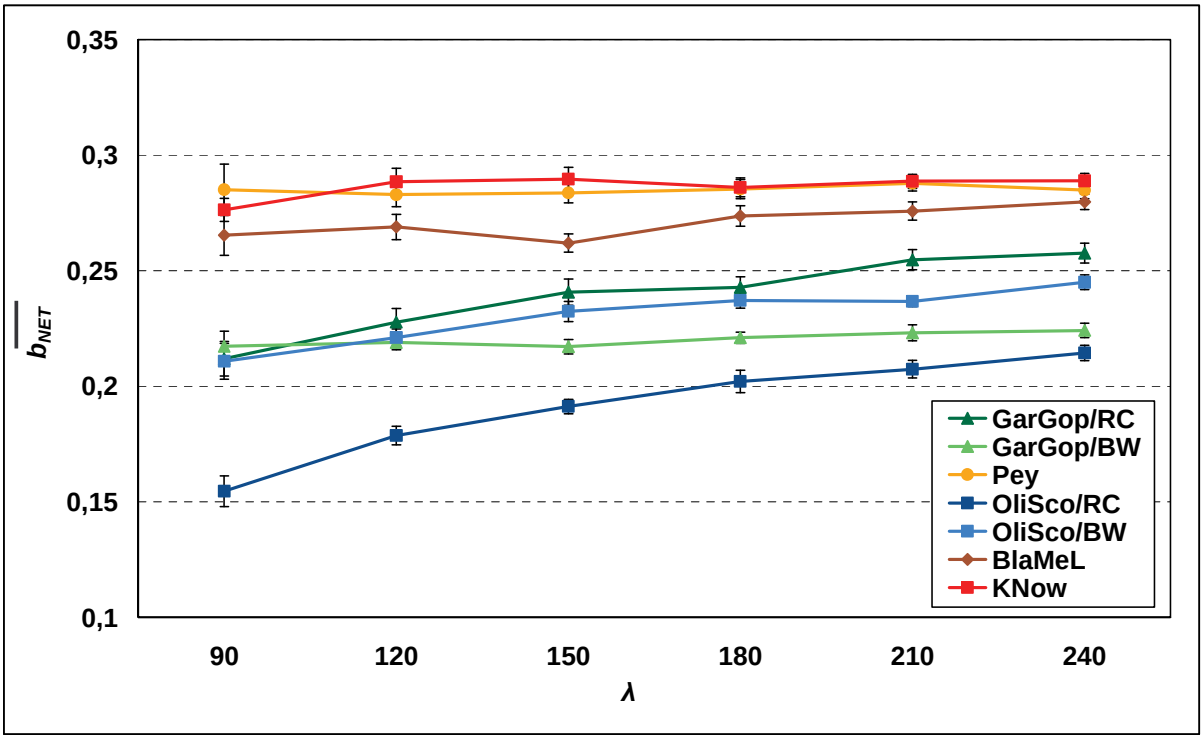
Aby zweryfikować to zagadnienie na drodze eksperymentalnej przeprowadzono serię badań topologii „Polska", w których badano, jak zmieniają się rezultaty przy zwiększaniu intensywności zgłoszeń, tzn. żądań tworzenia ścieżek. Na rys. A.1 i A.2 przedstawiono wyniki tych badań, zawierające odpowiednio średnią liczbę wywłaszczeń M i średni indeks straconego pasma sieciowego b_{NET}. O ile stracone pasmo nie zmienia się znacząco dla najlepszych algorytmów, to liczba wywłaszczeń wykazuje dość istotne zmiany, rosnąc od około 1,3 dla 90 zgł./godz. do ok. 1,8 dla 240 zgł./godz.



Rys. A.1. Topologia „Polska" - zależność średniej liczby wywłaszczeń M od intensywności zgłoszeń λ.

Należy zatem mieć świadomość, że rezultaty osiągane przez badane algorytmy rzeczywiście zależą od ilości ruchu, jaki jest generowany w sieci. W przypadku konkretnej topologii nie jest to duży problem, gdyż wszystkie algorytmy bada się w tych samych warunkach ruchowych. Większym problemem jest przeprowadzenie wiarygodnych badań porównawczych

dla różnych topologii. Zaproponowane rozwiązanie to opracowanie uniwersalnej procedury generowania podobnych warunków ruchowych niezależnie od topologii.



Rys. A.2. Topologia „Polska” - zmiany wartości średniej indeksu pasma straconego sieciowego b_{NET} od intensywności zgłoszeń λ.

Dodatkowym powodem badań nad ilością wprowadzanego do sieci ruchu była potrzeba uzyskania warunków zapewniających powstanie masowych wywłaszczeń bez potrzeby żmudnego prowadzenia badań testowych metodą prób i błędów. Dla osiągnięcia optymalnych warunków badań konieczne jest osiągnięcie wysokiego poziomu rezerwacji pasma na łączach i utrzymywanie takiego stanu przez cały czas trwania symulacji. Celem jest spowodowanie zajęcia przynajmniej części łączy na poziomie około 0,9. Zbyt małe obciążenie skutkuje brakiem dostatecznej liczby wywłaszczeń co generuje niewiarygodne wyniki, a z kolei wygenerowanie zbyt dużego zapotrzebowania na pasmo nie jest korzystne, gdyż niemal każde utworzenie ścieżki spowoduje powstanie wywłaszczeń. To jest sytuacja spoza obszaru normalnego funkcjonowania sieci, a raczej wynikająca z przekierowań ścieżek spowodowanych powstaniem uszkodzenia w sieci, i leży poza obszarem objętym badaniami w niniejszej pracy.

Dokładne określenie zapotrzebowania sieci jest w ogólności trudne, ale z analizy topologii można uzyskać pewne oszacowane od góry wartości ruchu, jakie należy generować, a następnie w razie potrzeby zmodyfikować je eksperymentalnie. Maksymalna, teoretyczna ilość pasma, jaką można zarezerwować w sieci odpowiada stanowi, w którym wszystkie łącza są w pełni zajęte. Prowadzi to do określenia dla danej sieci obciążenia maksymalnego C_{net} (A.1).

$$C_{net} = \sum_{e \in E} c_e \,. \qquad \text{(A.1)}$$

Biorąc pod uwagę to, że ścieżka składa się z kilku łączy, można określić sumaryczną wielkość B_{tot} zapotrzebowania na pasmo ścieżki, jakie należy wygenerować, aby całkowicie obciążyć sieć.

$$B_{tot} = \frac{C_{net}}{\bar{r}}, \tag{A.2}$$

gdzie $\bar{r}$ jest średnią długością drogi w sieci. W przypadku, gdy wszystkie łącza mają identyczną przepływność c, otrzymujemy zależność (A.3).

$$B_{tot} = \frac{C_{net}}{\bar{r}} = \frac{|E|}{\bar{r}} c = \frac{L \cdot c}{\bar{r}}. \tag{A.3}$$

Pozostaje jeszcze określić średnią długość drogi, przy czym zależnie od rozkładu źródeł ruchu w sieci bierze się pod uwagę wszystkie węzły lub tylko węzły brzegowe. Aby wyznaczyć średnią długość drogi, należy wyznaczyć wszystkie pary różnych węzłów brzegowych, a następnie zsumować długości dróg pomiędzy węzłami w parze i podzielić przez liczbę par. Zakładamy tutaj, że wszystkie węzły są ze sobą połączone bezpośrednio lub pośrednio, tzn. pomiędzy dowolnymi dwoma węzłami można odnaleźć drogę połączeniową. Zatem średnią długość drogi określa się jako:

$$\bar{r} = \frac{\sum_{e,f \in V, e \neq f} \mathrm{dist}(e, f)}{L(L-1)}, \tag{A.4}$$

gdzie dist(*e*,*f*) jest długością najkrótszej drogi (liczba łączy) pomiędzy parą węzłów *e* i *f*. W praktyce średnia długość drogi dla określonej sieci była obliczana przez program wspomagający tworzenie symulowanych sieci, a do wzoru (A.4) była podstawiana uzyskana w ten sposób wartość liczbowa.

Mając dane całkowite pasmo B_{tot} należy określić parametry źródeł należących do poszczególnych klas ruchu. Średnie pasmo b_n zarezerwowane w dowolnym momencie przez wszystkie źródła należące do *n*-tej klasy powinno wynosić:

$$b_n = q_n B_{tot}, \quad q_n \in (0;1), \sum_n q_n = 1 \tag{A.5}$$

gdzie q_n to względny udział *n*-tej klasy w całkowitym ruchu generowanym w sieci. Parametry źródeł danej klasy, tj. rezerwowane pasmo c_n, intensywność zgłoszeń λ_n i średni czas trwania połączeń t_n, należy dobrać według zależności (A.6).

$$c_n = \frac{b_n}{\rho_n} = \frac{b_n}{\lambda_n t_n} \tag{A.6}$$

Wartości w ten sposób otrzymane są teoretycznym limitem zakładającym idealnie równomierne obciążenie wszystkich łączy, które mogą być przeszacowane i spowodować więcej wywłaszczeń, niż się oczekuje. Jest to jednak prosta metoda, dzięki której można wyznaczyć wartość odniesienia, skorygowaną później eksperymentalnie. W praktyce uzyskane w ten sposób wartości były zwykle odpowiednie do badań algorytmów wywłaszczania, a użycie opisanej procedury istotnie skróciło czas potrzebny na przeprowadzenie badań.

ZAŁĄCZNIK B. ROZKŁADY CZASÓW WYKONANIA ALGORYTMÓW WYWŁASZCZANIA

W rozdziale 6.4.1 zamieszczono histogramy czasów wykonania badanych algorytmów wywłaszczania w środowisku symulacyjnym. Dla lepszej ilustracji przedstawiono tu rozkłady tych czasów w postaci tabeli, w których określono procentowy udział próbek należących do poszczególnych przedziałów.

Dodatkowym ułatwieniem w interpretacji wyników jest oznaczenie kolorowym tłem tych przedziałów, w których znalazło się najwięcej próbek. Kwalifikacji poszczególnych wartości do odpowiedniego koloru dokonano przy użyciu wartości progowych. Dla topologii „Polska", wybrano cztery wartości, wynoszące 3%, 5%, 8% i 16%, a dla topologii „Niemcy I", trzy progi 3%, 5% i 8%, przy czym przekroczenie wyższego progu odpowiada bardziej intensywnemu kolorowi.

Tabela B.1. Topologia „Polska" - procentowy udział próbek w poszczególnych zakresach czasów wykonania algorytmów (kolorami o rosnącej intensywności wyróżniono zakresy zawierające odpowiednio przynajmniej 3, 5, 8 i 16% próbek).

Czas [µs]	GarGop/ RC	GarGop/ BW	Pey	OliSco/ RC	OliSco/ BW	BlaMeL	KNow
0	1,66	1,25	1,64	1,23	1,51	1,25	1,33
10	0,25	0,30	0,21	0,37	0,24	0,27	0,23
20	0,88	6,27	2,13	1,23	5,76	1,06	1,06
30	16,03	25,91	18,25	19,17	22,94	11,04	10,30
40	23,19	18,43	20,52	20,50	23,57	19,26	18,94
50	23,37	16,63	23,50	22,93	18,38	26,13	17,52
60	14,78	11,30	12,40	14,02	9,82	14,99	15,94
70	8,80	6,65	6,50	7,10	5,69	8,50	12,73
80	3,97	4,57	3,47	4,10	3,82	4,20	8,20
90	2,19	2,45	3,33	3,16	2,60	4,17	5,33
100	1,48	1,71	2,82	1,80	1,98	2,81	2,98
110	1,09	1,38	1,73	1,75	1,16	1,99	1,87
120	2,31	3,15	3,49	2,64	2,55	4,33	3,57

Tabela B.2. Topologia „Niemcy I” - procentowy udział próbek w poszczególnych zakresach czasów wykonania algorytmów (kolorami o rosnącej intensywności wyróżniono zakresy zawierające odpowiednio przynajmniej 3, 5 i 8% próbek).

Czas [µs]	GarGop/ RC	GarGop/ BW	Pey	OliSco/ RC	OliSco/ BW	BlaMeL	KNow
0	0,63	0,89	0,61	0,55	0,71	0,54	0,61
25	0,34	0,39	0,27	0,31	0,31	0,26	0,30
50	0,40	0,46	0,42	0,35	0,45	0,36	0,42
75	1,22	3,63	1,71	0,93	10,90	0,90	0,69
100	21,23	33,68	20,42	19,80	19,53	15,95	15,30
125	16,32	14,73	8,52	10,91	6,27	8,00	18,99
150	7,22	5,97	4,17	3,99	5,89	4,14	8,00
175	5,94	4,72	4,76	4,21	8,96	4,89	5,76
200	5,83	4,71	8,12	6,80	9,29	7,77	5,76
225	6,44	4,73	8,34	8,66	5,71	8,54	5,84
250	6,44	4,76	6,28	6,41	4,48	6,68	5,87
275	6,16	4,36	5,17	5,16	3,65	5,63	5,61
300	5,15	3,69	4,38	4,61	3,56	5,10	5,56
325	4,26	2,82	3,66	4,21	3,22	4,03	5,00
350	3,43	2,28	2,80	3,08	3,04	3,29	4,10
375	2,51	1,71	2,49	2,67	2,39	2,88	3,24
400	1,73	1,35	2,69	2,43	1,73	2,74	2,42
425	1,18	0,98	2,33	2,23	1,33	2,67	1,82
450	0,82	0,78	2,01	1,97	1,30	2,28	1,24
475	0,59	0,55	1,45	1,67	1,23	1,87	0,98
500	0,39	0,45	1,26	1,32	1,06	1,52	0,59
525	0,28	0,37	1,22	1,08	0,91	1,31	0,45
550	0,23	0,30	1,09	1,03	0,65	1,23	0,30
575	0,21	0,25	0,97	0,83	0,57	1,11	0,24
600	1,06	1,43	4,88	4,78	2,86	6,31	0,92