Wirtualizacja zasobów oraz alokacja przepływów w sieciach sterowanych programowo przy jednoczesnym zapewnieniu wymagań QoS

Chodacki, Maksymilian maksymilian.chodacki@gmail.com
Grzanka, Antoni antoni.grzanka@gmail.com Leniart, Eryk eryk.leniart@gmail.com
Niedziałkowski, Adam adam.niedzialkowski@gmail.com

17 Stycznia 2017

1 Wstęp

Sieci sterowane programowo (SDN, z ang. Software Defined Networks) to koncept, który coraz bardziej zyskuje na wadze w dzisiejszym świecie architektury systemów i sieci. Koncepcja oddzielenia warstwy kontrolującej sieć od mechanizmów związanych z transmisją danych rozwiązuje wiele problemów, przed którymi stają codziennie architekci sieci. Jednym z wyzwań, przed którym stają operatorzy chcący zaimplementować w swojej sieci mechanizm SDN jest sprawiedliwy podział zasobów pomiędzy swoich klientów (dzierżawców) oraz zapewnienie takiej alokacji przepływów, która zapewni płynne działanie sieci. Jest to krytycznie ważna kwestia, ponieważ w rzeczywistych zastosowaniach, każdy z klientów (dzierżawców) sieci ma swoje wymagania odnośnie świadczonych przez operatora usług, których niespełnienie może wiązać się z poważnymi konsekwencjami finansowymi. Do tego typu parametrów zaliczamy jitter, opóźnienie przesyłu, packet loss czy [cos tam jeszcze].

W pracy [1] autorzy proponują kompleksowe rozwiązanie powyższego problemu, obejmujące dwustopniowy model optymalizacyjny (wirtualizacja zasobów poprzez podział sieci oraz maksymalizacja minimalnego przepływu).

W poniższej pracy autorzy skupili się na optymalizacji sposobu alokacji przepływów względem innych parametrów niż te zaproponowane w [1], co może bardziej realistycznie odpowiadać zapotrzebowaniom klientów. W tym celu przeanalizowano algorytm zaproponowany w [1], dokonano jego implementacji w języku ILOG, a także rozszerzono go, proponując dwa alternatywne modele ostatniej fazy działania algorytmu.

2 Podział zasobów

Pierwszą częścią algorytmu zaproponowanego w [1] jest podział sieci pomiędzy dzierżawców (klientów) tak, aby możliwie ich od siebie odizolować. Dopiero na tak przygotowanej, podzielonej sieci dokonuje się alokacji przepływów oraz sprawdzenia wymagań dotyczących parametrów Quality of Service.

Celem optymalizacji w pierwszym modelu jest maksymalna izolacja infrastruktury wykorzystywanej przez różnych dzierżawców, co można osiągnąć np. przez minimalizację łącz współdzielonych przez różnych klientów:

$$\min \sum_{t_1 \in T} \sum_{t_2 \in T} \sum_{a \in A} x_{at_1} x_{at_2} \tag{1}$$

W celu zapewnienia łączności każdego z węzłów z dowolnym innym dodano ograniczenia powodujące zbudowanie drzewa rozpinającego na grafie:

$$\forall_{t \in T} \sum_{a \in A} x_{at} \geqslant N - 1 \tag{2}$$

$$\forall_{t \in T} \forall_{Z \in V : Z \subset V, Z \neq \emptyset} \sum_{a \in S_n} x_{at} \geqslant 1 \tag{3}$$

gdzie Z to zbiór podgrafów V a S_n to zbiór wszystkich łącz mających swój początek w Z a koniec w $V \setminus Z$. Powyższe ujęcie problemu zbudowania drzewa rozpinającego znane jest jako *cutset formulation* i zapewnia, że w każdym zbudowanym drzewie wszystkie węzły będą miały łączność z dowolnym innym.

Z uwagi na to, że implementacja wymagała wykorzystania dwóch łącz dla każdego połączenia, należało dodać również ograniczenie zabraniające jakiemukolwiek dzierżawcy korzystanie z łącza tylko w jedną stronę:

$$\forall_{t \in T} \forall_{Z \in V : Z \subset V, Z \neq \emptyset} \forall_{a \in S_n} x_{at} x_{a't} \tag{4}$$

gdzie $x_{a't}$ oznacza łącze komplementarne do x_{at} łączące te same węzły w drugą stronę, tj. $\forall_{a \in A : a=(i,j)} \exists_{a' \in A : a'=(j,i)}$.

3 Model alokacji przepływu

Naszym kolejnym krokiem było zaimplementowanie modelu alokacji przepływów, w którym funkcją celu jest maksymalizacja minimalnego przepływu, czyli zrównoważenie ruchu generowanego przez przepływy dzierżawców na wszystkie dostępne łącza:

$$\sum_{n\in N} \max_{\{X^n_{f,i,j};\forall(i,j)\in E^f_n, f\in F^n\}} \min_{f\in F^n} \lambda^n_f \tag{5}$$

W modelu zostało zaimplementowanych pięć ograniczeń. Pierwsze z nich zapewnia, że suma wartości przepływów na danym łączu nie przekroczy jego pojemności:

$$\sum_{n\in N}\sum_{f\in F^n} X^n_{f,i,j} \leqslant c_{ij} \tag{6}$$

Kolene ograniczenie nazywane konserwacją przepływów zapewnia, że dla danego węzła niebędącego węzłem docelowym, dla dowolnego przepływu suma wartości wychodzących przepływów jest równa sumie wartości przychodzących przepływów plus wartość danego przepływu jeśli jest to węzeł źródłowy dla tego przepływu. W innym wypadku suma ta równa się zero:

$$\begin{gathered}\sum_{j;(i,j)\in E^f_n} X^n_{f,i,j} - \sum_{j;(j,i)\in E^f_n} X^n_{f,i,j} = \lambda^n_f \parallel_{\{i=s^n_f\}} \\ \forall i \neq d^n_f, f \in F^n, n \in N\end{gathered} \tag{7}$$

Następne ograniczenie zapewnia, że przepływ nie może zostać rozdzielony na dwa lub więcej łączy:

$$X^n_{f,i,j} = 0 \quad \forall(i,j) \notin E^f_n, f \in F^n, n \in N \tag{8}$$

Czwarte ograniczenie związane jest z zapewnieniem jakości obsługi. Zapewnia [POWTORZENIE] ono, że iloczyn straty pakietów na łączach przypisanych do danego przepływu nie przekroczy ustalonej maksymalnej wartości:

$$\prod_{(i,j)\in E^f_n} p^n_{f,i,j} < p^{max}_{n,f} \tag{9}$$

Ostatnie ograniczenie nie jest opisane przez autorów, zostało dodane przez nas na potrzeby naszej struktury danych wejściowych, a dokładniej podzielonego grafu pomiędzy dzierżawców. Ograniczenie to zapewnia, że dla danego dzierżawcy, każdy jego przepływ nie będzie wykorzystywał łącz, które do niego nie należą:

$$\forall_{f\in F^n} X_{f,i,j} = 0 \quad \text{jeśli} \quad x^f_{i,j} = 0 \tag{10}$$

4 Główny skrypt

Skrypt główny jest zasadniczo podzielony na kilka części. Jego szczegółowy algorytm działania przedstawiony jest na rysunku 1. Po pierwsze, uruchamia on model podziału sieci pomiędzy dzierżawców. W wyniku tego otrzymujemy podgrafy dla każdego z klientów, które są następnie przekazywane do modelu alokacji przepływów jako stała wejściowa. W przypadku znalezienia rozwiązania, dla danej podzielonej sieci sprawdzane są parametry jakości obsługi takie jak opóźnienie oraz jitter. W przypadku nie spełnienia tych wymagań, algorytm bierze następne najlepsze rozwiązanie z drugiego modelu i dla niego zostają sprawdzone parametry QoS. Jeśli dla żadnego rozwiązania z puli nie otrzymamy zadowalających rezultatów, algorytm powraca do pierwszego modelu i wybiera z jego puli rozwiązań kolejną najbardziej optymalnie podzieloną sieć, a następnie przekazuje te dane do modelu drugiego. Cykl ten powtarzany jest do znalezienia pasującego rozwiązania.

Dodatkowo z powodu ograniczeń języka OPL musieliśmy zaimplementować własne funkcje takie jak obliczanie wariancji czy sortowanie.

5 Testy

Implementację opisanych wcześniej modeli testowaliśmy na trzech różnych topologiach - małej, średniej oraz dużej. Początkowo sprawdzaliśmy działanie modeli oraz skryptu tylko na bardzo małej topologii. Było taki ze względu na możliwość szybkiej weryfikacji poprawności działania kodu, a przy większych topologiach nie bylibyśmy w stanie jednoznacznie określić czy dany model działa tak, jak zakładaliśmy.

Testowanie polegało na sprawdzaniu czy zostały spełnione zakładane przez nas wcześniej warunki. Początkowo były to sprawdzenie, czy każdy z dzierżawców posiada połączenie do wszystkich węzłów oraz czy liczba wspólnych krawędzi dla wszystkich dzierżawców jest jak najmniejsza. Stopień trudności w weryfikowaniu poprawności rozwiązania rósł wraz

z każdym dodamym dzierżawcą, węzłem lub połączeniem, dlatego większość testów przeprowadziliśmy dla topologii składającej się z sześciu węzłów, dwóch dzierżawców i ośmiu połączeń. Była ona na tyle rozbudowana, że na pewnych etapach projektu pojawiały się błędy w działaniu naszych modeli, a z drugiej strony na tyle przejrzysta, że byliśmy w stanie w niedługim czasie zweryfikować poprawność działania modeli.

Na podstawie przeprowadzonych testów stwierdzamy, że nasza implementacja działa w ten sam sposób, w jaki zamierzaliśmy. Partycjonowanie sieci odbywa się prawidłowo - nie zdarza się, aby któryś z dzierżawców nie miał dostępu do któregoś z węzłów. Liczba wspólnych krawędzi jest najmniejsza z możliwych - tutaj pewność dobrego działania mamy tylko dla mniejszych topologii, w których byliśmy w stanie ręcznie sprawdzić wszystkie możliwości i określić minimalną liczbę wspólnych krawędzi. Kolejną testowaną funkcjonalnością było przydzielanie łączy do konkretnych przepływów. Sprawdzaliśmy, czy przez każde łącze płynie taki ruch, który będzie w stanie się w nim zmieścić bez strat wynikających z jego łącza - ten warunek także był spełniony dla naszych przypadków testowych.

6 Rozszerzenie

Po zaimplementowaniu podstawowego modelu postanowiliśmy podejść do tematu rozszerzenia problemu na dwa sposoby: dodaniu nowych ograniczeń oraz zastosowaniu go w innej formie. Zaproponowanym przez nas ograniczeniem jest zapewnianie minimaalnej wartości przepływu, a nowym zastosowaniem jest wprowadzenie kosztu jako głównej metryki optymalizacji sieci.

6.1 Zapewnienie minimalnych wartości przepływu

Istotnym aspektem nie poruszanym przez autorów pracy jest zapewnienie minimalnej wartości przepływu. Autorzy odnieśli się do problemu zapewnienia wszystkim przepływom dodatniej wartości (w praktyce wszystie przepływy będą miały równą wartość). Pozwala to na sprawiedliwy podział zasobów, natomiast w przypadku jeżeli różnym przepływom chcemy zapewnić różne minimalne wartości potrzebne jest rozszerzenie problemu o nowe dane (minimalną liczbę danych dla każdego przepływu) oraz dodatkowe ograniczenie:

$$\forall_{t \in T} \forall_{f \in F_t} \quad \lambda_{tf} \geqslant f.minimal \tag{11}$$

6.2 Koszt przesyłu danych

W przypadku nowego zastosowania postanowiliśmy postawić na praktyczne podejście. Rozszerzylismy model o koszt przesyłanych danych i zmieniliśmy funkcję celu tak, by zrównoważyć koszt przepływów:

$$\sum_{t \in T} \max_{f \in F_t} \sum_{a \in A_f} a.cost \tag{12}$$

6.3 Wyniki roszerzonych modeli

Poniżej przedstawiamy porównanie wyników modeli rozszerzonych i podstawowego wyliczonych w środowisku CPLEX. Tabela zawiera wyniki dla trzech zestawów danych: sieci małej (4 węzły), średniej (6 węzłów) i dużej (17 węzłów).

Tabela 1: Mała topologia

Mała topologia	model podstawowy	model z minimalnym przepływem	model z minimalnym kosztem
Jitter	0	0	0
Delay	2,9	2,9	4,83
Sumaryczny przepływ	10	10	5
czas wykonania	31	37	22

Tabela 2: Średnia topologia

Średnia topologia	model podstawowy	model z minimalnym przepływem	model z minimalnym kosztem
Jitter	0,0023	0,0004	0
Delay	1,92921099	2,45	3,3
Sumaryczny przepływ	50	50	2
czas wykonania	71	88	89

Tabela 3: Duża topologia

Duża topologia	model podstawowy	model z minimalnym przepływem	model z minimalnym kosztem
Jitter	6,51E-05	6,51E-05	4,93E-32
Delay	3,57498538	3,82498538	4,83
Sumaryczny przepływ	80	80	5
czas wykonania	917096	908099	908099

7 Podsumowanie

W opisanym projekcie skupiliśmy się na problemie optymalizacji sieci przy użyciu algorytmów zaproponowanych przez azjatyckich naukowców. Udało nam się zaimplementować ich rozwiązanie i sprawdzić, czy rzeczywiście ono działa. Założeniem projektu była minimalizacja liczby wspólnych łączy dzielonych przez kilku dzierżawców. Uzyskane wyniki pozwalają nam stwierdzić, że projekt został zrealizowany poprawnie, spełnia wszystkie pierwotne założenia. W trakcie problemu zmierzyliśmy się z kilkoma niebanalnymi problemami - jak na przykład linearyzacja ograniczeń, przez co musieliśmy poświęcić na zrobienie projektu więcej czasu niż zakładaliśmy, z drugiej strony jednak mieliśmy okazję więcej się nauczyć.

Literatura

Lin, Shih-Chun and Wang, Pu and Luo, Min, *Jointly Optimized QoS-aware Virtualization and Routing in Software Defined Networks*, Comput. Netw. Vol. 96, Elsevier North-Holland, Inc. February 2016.