Модел. и анализ информ. систем. Т. **21**, № **6** (2014) 57–70

УДК 519.681

О некоторых задачах реконфигурирования программно-конфигурируемых сетей

Захаров В.А.[1], Чемерицкий Е.В.[2]

Московский государственный университет им. М.В. Ломоносова
119991, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52
Центр прикладных исследований компьютерных сетей

e-mail: zakh@cs.msu.su, echemeritskiy@arccn.ru

получена 14 ноября 2014

Ключевые слова: программно-конфигурируемая сеть, коммутатор, контроллер, правило коммутации, пакет, реконфигурация, маршрут, постусловие, инвариант

Разработка алгоритмов реконфигурирования сетей является важным направлением развития программного обеспечения для телекоммуникационных сетей нового поколения — программно-конфигурируемых сетей. Частный случай проблемы реконфигурирования сетей — это задача плавного восстановления заданной сетевой конфигурации, после того как некоторые правила коммутации пакетов были удалены из таблиц коммутаторов (например, по истечении срока их активности). В данной статье проведено исследование этой задачи в рамках формальной модели программно-конфигурируемых сетей, предложены корректные и безопасные алгоритмы восстановления сетевых конфигураций и показано, что в общем случае задачу плавного восстановления конфигураций нельзя решить без обращения к правилам коммутации с приоритетами.

1. Введение

Как только возникла концепция программно-конфигурируемых телекоммуникационных сетей (Software Defined Networks, SDN) [1], был организован целый ряд проектов, направленных на создание языков высокого уровня и инструментальных средств для программирования SDN: NetCore [2], Maestro [3], Procera [4]. И хотя в этих проектах исследователи старались охватить разные аспекты задачи автоматизации администрирования сетей, все они основываются на общей фундаментальной идее многоуровневого программирования. Чтобы реализовать в полной мере эту идею, нужно решить немало задач. Некоторые из них — это хорошо изученные традиционные задачи системного программирования; при выборе методов их решения

[1]Работа поддержана грантом РФФИ 12-01-00706.
[2]Работа поддержана фондом «Сколково», грант N 79, июль, 2012.

разработчики сталкиваются, скорее, с "трудностями изобилия", нежели с отсутствием подходящих средств. Но имеется также ряд совершенно новых задач, связанных с особенностями управления SDN; эти проблемы ранее не рассматривались, и некоторые из них даже не имеют адекватной математической формализации. Исследование и решение этих проблем, главными из которых являются вопросы разработки математических моделей и алгоритмов корректного и эффективного администрирования SDN, особенно важно для создания развитых программных библиотек высокоуровневых языков сетевого программирования.

Одна из наиболее важных задач управления SDN — это проблема реконфигурирования сети, т.е. корректного и безопасного изменения содержимого таблиц коммутации пакетов в сетевых коммутаторах. SDN — это иерархическая управляющая система, в которой контроль перемещений пакетов данных осуществляется опосредованно: контроллеры управляют поведением коммутаторов путем изменения наборов правил коммутации пакетов в таблицах коммутации, а коммутаторы управляют движением пакетов данных в сети в соответствии с правилами коммутации пакетов. Конфигурация SDN определяется топологией сети и распределением наборов правил коммутации пакетов по таблицам коммутаторов. Команды контроллера могут изменять конфигурацию (реконфигурировать) SDN путем добавления, удаления или модификации правил коммутации. Необходимость в проведении реконфигурирования SDN возникает во многих случаях: для оптимизации таблиц коммутации во избежание их переполнения, для поддержания и восстановления маршрутов движения пакетов при подключении и отключении аппаратуры, а также при истечении срока действия правил коммутации или миграции хостов на граничных точках сети, для балансировки нагрузки каналов связи, для мониторинга сетевого трафика.

В наиболее общем виде Проблему Реконфигурирования Сети (Network Update Problem, NUP) можно сформулировать так: для заданной конфигурации C и пары спецификаций — инварианта Φ и постусловия Ψ — сформировать такую последовательность α команд реконфигурирования, чтобы в результате ее применения к конфигурации C образовалась конфигурация C', удовлетворяющая требованию Ψ, и при этом каждая промежуточная конфигурация удовлетворяла бы условию Φ. В ранних работах [6, 7, 8, 9], посвященных изучению вопросов реконфигурирования традиционных телекоммуникационных сетей, исследователи рассматривали лишь частные случаи этой задачи для специальных сетевых протоколов с целью предотвращения различных аномалий поведения сети.

Более систематичное исследование NUP было предпринято в статьях [10, 11, 12]. В них изучалась задача корректного глобального реконфигурирования SDN: как преобразовать одну заданную конфигурацию SDN C в другую заданную конфигурацию C' так, чтобы в ходе этого преобразования любой пакет в сети либо перемещался по маршруту, прокладываемому только правилами конфигурации C, либо по маршруту, прокладываемому только правилами конфигурации C'. В статье [11] показано, что корректное глобальное реконфигурирование можно осуществить при помощи универсального трехфазного алгоритма с использованием пометок (тегов) в специально выделенных для этой цели полях заголовков пакетов (например, в полях VLAN). Исследование других вариантов NUP проводилось в работах [13] (синтез сетевых конфигураций) и в [14, 15] (оптимизация таблиц коммутации).

Хотя алгоритм реконфигурирования SDN, предложенный в статье [11], дает решение одного из вариантов NUP, для его реализации необходимы дополнительные ресурсы — выделенные поля в заголовках пакетов. Поэтому важно выяснить, какие варианты NUP можно решить без использования техники тегирования пакетов. В данной статье мы исследовали один из вариантов NUP, в котором новая сетевая конфигурация C' образуется из начальной конфигурации C за счет добавления некоторого множества правил коммутации пакетов в таблицы коммутаторов. Этот вариант NUP можно рассматривать как задачу восстановления конфигурации SDN, в которой некоторые правила коммутации были удалены по истечении срока их активности. Наш вклад в ее изучение представлен тремя основными результатами. Мы предложили усовершенствованную абстрактную модель SDN, пригодную для формализации многочисленных вариантов NUP, которые рассматривались в статьях [10, 11, 12, 13, 14, 15]. В рамках предложенной модели SDN мы сформулировали задачу восстановления конфигураций и установили достаточные условия, при которых ее можно решить, не прибегая к тегированию пакетов. Мы также показали, что в некоторых случаях задача восстановления конфигураций не может быть решена без использования средств, выходящих за рамки предложенной модели.

2. Модель программно-конфигурируемой сети

Для построения формальной модели SDN в рамках протокола OpenFlow [5] мы повысили уровень абстракции в определении элементов сети. Заголовки пакетов и порты коммутаторов являются атомарными сущностями, правила коммутации пакетов определяются произвольными предикатами на множестве состояний пакетов.

Пусть H — это множество заголовков пакетов данных, W — множество коммутаторов сети и P — множество портов отдельного коммутатора. Пары из множества $V = P \times W$ называются *точками сети*, пары из множества $L = H \times P$ — *локальными состояниями пакетов*, а тройки из множества $S = H \times P \times W$ — *состояниями пакетов*. Каждый коммутатор имеет два специальных порта $Drop$ и $Contr$. Пакеты, попавшие в порт $Drop$ (порт $Contr$), должны быть сброшены (отправлены контроллеру). Положим $L_0 = H \times (P \cup \{Drop, Contr\})$.

Топологическое устройство сети определяется *отношением коммуникации точек сети* $T = \{(v', v'') :$ существует канал связи между точками v' и $v''\}$. Так как в современных сетях каналы связи являются дуплексными, отношение коммуникации симметрично. Точка v называется *граничной*, если ни один канал связи не соединяет v с какой-либо другой точкой сети. Граничные точки служат для связи сети с окружающим миром. Запись E_T служит для обозначения множества всех *граничных состояний пакетов* $\langle h, p, w\rangle$, где $\langle p, w\rangle$ — граничная точка сети.

Правило коммутации пакетов задается тройкой $r = (G_r, A_r, m_r)$, где $G_r \subseteq L$, $A_r \subseteq L \times L_0$, а m_r — натуральное число. Предикат G_r (*предохранитель*) представляет собой абстракцию шаблона правила, двухместный предикат A_r (*действие*) является абстракцией последовательности действий (инструкции) правила, а число m_r — это приоритет правила. Семантика правила r определяется отношением $F_r \subseteq L \times L_0$ следующего вида: $(\ell, \ell_0) \in F_r \iff \ell \in G_r \wedge (\ell, \ell_0) \in A_r$.

Таблица коммутации tab коммутатора w — это конечное множество правил коммутации $\{r_1, r_2, \ldots, r_N\}$. Семантика таблицы tab определяется отношением коммутации пакетов R_{tab} следующим образом. Пусть k — это наивысший приоритет правил коммутации таблицы tab. Для каждого i, $1 \le i \le k$, обозначим записью tab^i множество всех правил с приоритетом i: $tab^i = \{(G, A, i) : (G, A, i) \in tab\}$. Определим рекурсивно по убыванию индекса i от k до 1 пары предикатов R^i_{tab} и B^i_{tab}:
$R^k_{tab} = \bigcup\limits_{r \in tab^k} F_r$, $B^k_{tab} = \bigcup\limits_{r \in tab^k} G_r$;
$R^i_{tab} = \{(\ell, \ell_0) \ : \ \exists\, r\ (r \in tab^i \wedge \ell \notin B^{i+1}_{tab} \wedge (\ell, \ell_0) \in F_r\})$, $B^i_{tab} = B^{i+1}_{tab} \cup \bigcup\limits_{r \in tab^i} G_r$.

Предполагая по умолчанию, что все пакеты, для которых не находится подходящего правила коммутации, отправляются контроллеру, введем предикат
$R^0_{tab} = \{(\langle h, p\rangle, \langle h, Contr\rangle) \ : \ \langle h, p\rangle \notin B^1_{tab}\}$
и определим отношение $R_{tab} = \bigcup\limits_{i=0}^{k} R^i_{tab}$. Оно означает, что каждый пакет, поступивший в некоторый порт коммутатора w, либо обрабатывается правилом с наивысшим приоритетом, применимым к этому пакету, либо отправляется контроллеру.

Таблицы коммутации SDN должны быть однозначными: никакой пакет не должен подпадать под два разных правила коммутации одинакового приоритета в одной таблице, т. е. для каждой пары правил $r_1 = (G_1, A_1, m_1)$ и $r_2 = (G_2, A_2, m_2)$ в случае $m_1 = m_2$ должно выполняться равенство $G_1 \cap G_2 = \emptyset$. Обозначим записью Tab множество всевозможных однозначных таблиц коммутации пакетов.

Сетевой конфигурацией C называется пара (T, I), где T — отношение коммуникации на множестве точек сети V, а отображение $I : W \to Tab$ — *функция загрузки коммутаторов*, приписывающая каждому коммутатору w таблицу коммутации $tab_w = I(w)$. Для сетевой конфигурации $C = (T, I)$ мы можем определить *отношение одношаговой коммутации пакетов* R_C на множестве глобальных состояний пакетов S следующим образом: включение $(\langle h, p, w\rangle, \langle h', p', w'\rangle) \in R_C$ выполняется тогда и только тогда, когда выполняется одно из следующих четырех условий:

1) пересылка пакета в следующий коммутатор: существует порт p'' коммутатора w, для которого верны включения $(\langle h, p\rangle, \langle h', p''\rangle) \in R_{I(w)}$ и $(\langle p'', w\rangle, \langle p', w'\rangle) \in T$;
2) пересылка пакета за пределы сети: $w' = w$, $(\langle h, p\rangle, \langle h', p'\rangle) \in R_{I(w)}$ и p' — граничный порт;
3) сброс пакета: $w' = w$, $(\langle h, p\rangle, \langle h', p'\rangle) \in R_{I(w)}$ и $p' = Drop$;
4) пересылка пакета контроллеру: $w' = w$, $(\langle h, p\rangle, \langle h', p'\rangle) \in R_{I(w)}$ и $p' = Contr$.

В нашей модели SDN отношение одношаговой коммутации пакетов R_C составляет семантику сетевой конфигурации C и полностью определяет поведение пакетов в сети. Но главное внимание уделяется не самому отношению R_C, а маршрутам передачи данных между граничными точками сети. Последовательность состояний пакетов $path = s_0, s_1, \ldots, s_i, s_{i+1}, \ldots$ называется *маршрутом* в конфигурации C в том случае, если $s_0 \in E_T$, и включение $(s_i, s_{i+1}) \in R_C$ выполняется для каждого i, $i \ge 0$. Маршрут называется *полным*, если он оканчивается в таком состоянии пакетов $s = \langle h, p, w\rangle$, для которого верно $s \in E_T$ или $p \in \{Drop, Contr\}$. Обозначим множество всех полных маршрутов в конфигурации C (маршрутов, исходящих из граничного состояния пакетов s) записью $Path(C)$ (соответственно $Path(C, s)$).

Изменение сетевых конфигураций происходит вследствие истечения сроков активности правил коммутации пакетов, при отключении или выходе из строя телекоммуникационной аппаратуры, а также в результате выполнения команд, полученных от контроллера. В протоколе OpenFlow [5] для реконфигурирования сетей можно использовать (наряду с другими) команды следующих типов:

$add(w, r)$ для добавления правила коммутации r в таблицу коммутатора w;
$del(w, G_0, m)$ для изъятия правил из таблицы коммутатора w: правило $r = (G, A, m)$ удаляется из таблицы в том и только том случае, когда предикат G_0 является логическим следствием предиката-предохранителя G, т. е. $G \subseteq G_0$;
$mod(w, G_0, A, m)$ для модификации правил таблицы коммутатора w: во всех правилах $r = (G, A', m)$ действие A' заменяется действием A в том и только том случае, когда предикат G_0 является логическим следствием предиката-предохранителя G.

Воспользуемся записью $com(C)$ для обозначения той сетевой конфигурации, которая образуется из конфигурации C после выполнения команды com. Для последовательности команд $\alpha = com_1, \ldots, com_k$ будем полагать $\alpha(C) = com_k(\ldots, com_1(C)\ldots)$.

Поскольку SDN является вполне асинхронной распределенной системой, любые две команды реконфигурирования, отправленные контроллером подряд друг за другом, могут быть выполнены в произвольном порядке даже тогда, когда они адресованы одному и тому же коммутатору. В протоколе OpenFlow есть средства синхронизации, позволяющие регулировать порядок выполнения команд. Не вдаваясь в подробности их функционирования, будем предполагать, что конечное множество команд реконфигурирования (*реконфигурационный блок*) Com снабжено отношением строгого частичного порядка $\prec$: если $com' \prec com''$, то при любом выполнении блока Com команда com'' выполняется после команды com'. Для пары реконфигурационных блоков $(Com_1, \prec_1)$ и $(Com_2, \prec_2)$ запись $(Com_1, \prec_1); (Com_2, \prec_2)$ будет обозначать их последовательную композицию, которая представляет собой блок $(Com_1 \cup Com_2, \prec)$, в котором для любой пары команд com', com'' соблюдается порядок $com' \prec com''$ в том и только том случае, если com' и com'' принадлежат одному и тому же множеству Com_i, $i = 1, 2$, и при этом $com' \prec_i com''$, или если $com' \in Com_1$ и $com'' \in Com_2$.

3. Задача восстановления конфигурации

Известно немало способов формализации политик маршрутизации пакетов. Например, конфигурацию (S, R_C) можно рассматривать как размеченную систему переходов и использовать регулярные выражения, темпоральные логики, μ-исчисление или фрагменты логики предикатов первого порядка (см. [10, 11, 13, 16, 17]) для формальной спецификации политик маршрутизации. Независимо от способа спецификации воспользуемся записью $C \models \varphi$ для обозначения утверждения о том, что конфигурация C удовлетворяет спецификации φ.

В общем виде задачу реконфигурирования сети (NUP) можно сформулировать так. Для заданных формальных спецификаций политик маршрутизации Φ (инвариант) и Ψ (постусловие), а также конфигурации C построить такой реконфигурационного блока $(Com, \prec)$, чтобы для любой его линеаризации α выполнялись два

требования: 1) $\alpha(C) \models \Psi$, и 2) $\beta(C) \models \Phi$ для каждого префикса β последовательности α.

Задача восстановления конфигурации (NRP) — это частный случай NUP. Сетевая конфигурация C' может самопроизвольно измениться, превратившись в конфигурацию C после удаления из некоторых таблиц $w_1, w_2, \ldots, w_n$ отдельных правил коммутации пакетов $r_1, r_2, \ldots, r_n$, например, ввиду истечения срока их активности, ошибочного выполнения реконфигурационных команд, неисправности функционирования сетевого оборудования и пр. Задача контроллера состоит в том, чтобы «плавно» восстановить конфигурацию C' из конфигурации C. Требование «плавности» подразумевает, что в ходе восстановления должны быть разрушены только те полные маршруты, которые содержатся в множестве $Path(C)$, но отсутствуют в множестве $Path(C')$, и должны быть проложены только те полные маршруты, которые содержатся в множестве $Path(C')$, но отсутствуют в множестве $Path(C)$. Формально эту задачу можно рассматривать как вариант NUP, в котором постусловие $\Psi(X)$ задается равенством $X = C'$, а инвариант $\Phi(X)$ — формулой

$$Path(C) \cap Path(C') \subseteq Path(X) \ \wedge Path(X) \subseteq Path(C) \cup Path(C').$$

Для обозначения этой задачи воспользуемся записью (C, C')-NRP .

Вначале исследуем NRP, когда все правила коммутации в конфигурациях C и C' имеют равный приоритет. Иногда конфигурацию C' нельзя плавно восстановить простым добавлением утраченных правил в каком-либо порядке. Рассмотрим конфигурацию C', изображенную на Рис. 1(а). Здесь $H = \{g, h\}$, и правила r_1 и r_2 обрабатывают пакеты обоих типов g и h. Записи $r_1 : g, h$ и $r_3 : g$ означают, что правило r_1 применимо к пакетам обоих типов g и h, а правило r_3 применимо только к пакетам типа g. Очевидно, что в конфигурации C' имеются четыре полных пути:
$p'_1 = (h, v_{11}), (h, v_{31}), (h, v_{21}), (h, v_{41}), (h, v_{43})$; $p'_2 = (g, v_{11}), (g, v_{31}), (g, v_{32})$;
$p'_3 = (g, v_{22}), (g, v_{41}), (g, v_{12}), (g, v_{31}), (g, v_{32})$; $p'_4 = (h, v_{22}), (h, v_{41}), (h, v_{43})$.
После изъятия правил коммутации r_1 и r_2 конфигурация C' превращается в конфигурацию C, в которой есть четыре полных маршрута:
$p_1 = (h, v_{11}), (h, Contr)$; $p_2 = (g, v_{11}), (g, Contr)$;
$p_3 = (h, v_{22}), (h, Contr)$; $p_4 = (g, v_{22}), (g, Contr)$.
Но попытка восстановить конфигурацию C' путем добавления на первом шаге правила коммутации r_1 в таблицу коммутатора w_1 (см. Рис. 1(б)) приводит к появлению полного маршрута $p_0 = (h, v_{11}), (h, v_{31}), (h, v_{21}), (h, Contr)$, которого нет в множествах маршрутов $Path(C)$ и $Path(C')$. Подобная же ситуация возникает и тогда, когда реконфигурация начинается с восстановления правила коммутации r_2.

Этот эффект обусловлен взаимной зависимостью правил коммутации r_1 и r_2: каждое из них применимо к тем пакетам, которые были коммутированы другим правилом. Чтобы разорвать эту зависимость, введем вспомогательные предикаты на множестве локальных состояний пакетов L.

Рассмотрим произвольную сетевую конфигурацию Z и пару правил коммутации пакетов $r' = (G', A', m')$, $r'' = (G'', A'', m'')$ в таблицах коммутаторов w' и w'' соответственно. Обозначим записью R_Z^+ транзитивное замыкание отношения одношаговой коммутации пакетов R_Z. Тогда *предикатом зависимости* $\theta_{r'r''}$ правил r' и

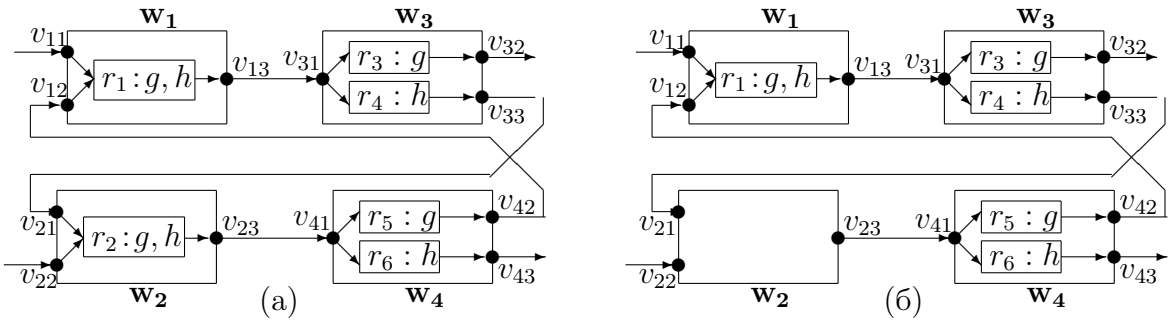

Рис. 1.

r'' назовем отношение на множестве L, которое задается следующей формулой:

$$\theta_{r'r''}(x) = \exists s \in S, \ell \in L_0 \ [E_T(s) \wedge R_Z^+(s, \langle x, w' \rangle) \wedge G'(x) \wedge R_Z^+(\langle x, w' \rangle, \langle \ell, w'' \rangle) \wedge G''(\ell)] \ .$$

Она выполняется тогда и только тогда, когда существует полный маршрут

$$s \stackrel{r_0}{\to} s_1 \to \cdots s_i \stackrel{r'}{\to} s_{i+1} \to \cdots s_j \stackrel{r''}{\to} s_{j+1} \to \cdots \ ,$$

в котором правило r' коммутирует пакеты, пребывающие в состоянии $s_i = \langle x, w' \rangle$, а правило r'' применяется к тем же пакетам, но *после* применения правила r'. Предикаты зависимости позволяют ввести двухместное отношение $\sqsubseteq_Z$ на множестве правил коммутации пакетов в конфигурации Z: правило r'' *зависит* от правила r' (обозначается $r'' \sqsubseteq_Z r'$) тогда и только тогда, когда $\theta_{r'r''} \not\equiv false$. Отношение $\sqsubseteq_Z$, в свою очередь индуцирует двухместное отношение $\prec_Z$ на множестве реконфигурационных команд: $add(w'', r'') \prec_Z add(w', r')$ тогда и только тогда, когда правило коммутации r'' в таблице коммутатора w'' зависит от правила r' в таблице коммутатора w' в сетевой конфигурации Z. Самый простой случай (C, C')-NRP — это тот, в котором отношение зависимости является отношением частичного порядка.

Теорема 1. *Предположим, что конфигурация C' образовалась из конфигурации C, после удаления правил коммутации пакетов $r_1, r_2, \ldots, r_n$ из таблиц коммутаторов $w_1, w_2, \ldots, w_n$ и при этом $\sqsubseteq^+_{C'}$ является отношением частичного порядка на множестве правил $U = \{r_1, r_2, \ldots, r_n\}$. Тогда реконфигурационный блок $(Com, \prec_{C'})$, где $Com = \{add(w_i, r_i) : 1 \le i \le n\}$, является решением (C, C')-NRP.*

Доказательство. Проводится индукцией по n. Если $U = \{r_1\}$, то утверждение теоремы очевидно. Предположим, что утверждение теоремы верно для любого множества U, содержащего не более $n-1$ правил коммутации. Пусть $U' = U \cup \{r_n\}$ и правило r_n является минимальным по отношению зависимости $\sqsubseteq^+_{C'}$. Будем считать, что команда $com_n = add(w_n, r_n)$ выполняется первой в пакете $(Com, \prec^+_{C'})$. Допустим, что в конфигурации $Z = com_n(C)$ возник новый полный маршрут $path$, которого не было среди маршрутов конфигурации C. Если бы этот маршрут также отсутствовал среди маршрутов конфигурации C', то он имел бы вид

$$s_0 \to s_1 \to \cdots \to s_i \stackrel{r_n}{\to} s_{i+1} \to \cdots \to s_j \stackrel{r}{\to} s_{j+1} = (h, Contr),$$

где s_j — состояние пакета, к которому применимо одно из правил r множества U. Тогда $r \sqsubseteq^+_{C'} r_n$, что противоречило бы минимальности правила r_n. Следовательно,

$path \in Path(C')$. Значит, $Path(C) \cap Path(C') \subseteq Path(Z) \subseteq Path(C) \cup Path(C')$, и по индуктивному предположению $(Com, \prec_{C'})$ — решение задачи (C, C')-NRP. □

Если отношение зависимости $\sqsubseteq^+_{C'}$ не является частичным порядком на множестве удаленных правил, то механизм приоритетов приобретает решающее значение для плавного восстановления конфигурации C'.

Лемма 1. *Пусть C' — конфигурация, изображенная на Рис. 1(а), и пусть конфигурация C получена из C' за счет удаления правил в таблицах коммутаторов w_1 и w_2. Предположим, что $(Com, \prec)$ — произвольный реконфигурационный блок, состоящий из команд, работающих с правилами коммутации одного и того же приоритета. Тогда $(Com, \prec)$ не является решением задачи (C, C')-NRP.*

Доказательство. Предположим, что блок $(Com, \prec)$ является кратчайшим решением (C, C')-NRP, и пусть α — произвольная линеаризация этого блока. Поскольку все правила коммутации, с которыми работают команды последовательности α, имеют равный приоритет, каждая команда изменяет отношение одношаговой коммутации. Рассмотрим кратчайший префикс $\beta = \beta', com$ последовательности α, который активизирует все правила коммутации, присутствующие в конфигурации C'. Тогда команда com должна активизировать одно из правил конфигурации C'. Так как все правила коммутации имеют одинаковый приоритет, команда del не может активизировать правила коммутации. Значит, com — это команда вида add или mod. Поскольку все правила имеют одинаковый приоритет, команды реконфигурирования, относящиеся к одним правилам, не изменяют активности других правил. Значит, команда com относилась к одному из правил коммутации пакетов, содержащихся в конфигурации C'. Рассмотрим по отдельности каждое из шести правил коммутации, присутствующих в конфигурации C' , и оба типа команд add и mod, которые могли относиться к этим правилам. Ограничимся анализом двух типичных случаев применения команд add и mod к правилу таблицы w_1.

Допустим, что $com = add(w_1, r_1)$. Поскольку таблица коммутатора w_1 должна быть однозначной, ни одно из правил этой таблицы в конфигурации $\beta'(C)$ неприменимо к пакетам типа h и g. Значит, в конфигурации $\beta'(C)$ пакеты этого типа направляются коммутатором w_1 по умолчанию в контроллер сети. А поскольку все остальные правила r_2–r_6 уже были активизированы, в конфигурации $\beta'(C)$ есть полный маршрут $p' = (g, v_{21}), (g, v_{41}), (g, v_{11}), (g, Contr)$, который отсутствует в множествах $Path(C)$ и $Path(C')$. Следовательно, $com \neq add(w_1, r_1)$.

Допустим, что $com = mod(w_1, G, A)$, где G — предохранитель правила коммутации r_1, а A — действие этого правила. Значит, в таблице коммутатора w_1 в конфигурации $\beta'(C)$ уже есть правило коммутации с предохранителем G. В силу минимальности β действие A' этого правила отлично от действия правила r_1. Тогда в конфигурации $\beta'(C)$ должен быть полный маршрут $p'' = (g, v_{21}), (g, v_{41}), (g, v_{12}), (g', v')$, где $(g', v') \neq (g, v_{31})$. Очевидно, что p'' не является началом ни одного из полных маршрутов множества $Path(C) \cup Path(C')$. Следовательно, $com \neq mod(w_1, G, A)$.

Таким образом, команда com не могла активизировать правило r_1. Аналогичные рассуждения, примененные к правилам $r_2 - r_6$, приводят к заключению о том, что команда реконфигурирования com не активизирует ни одно из этих правил, вопреки

выбору $\beta = \beta', com$ как кратчайшего префикса последовательности α, восстанавливающего активность всех правил r_1–r_6. Поэтому предположение о существовании реконфигурационного блока, оперирующего только с правилами одного и того же приоритета и дающего решение задачи (C, C')-NRP, приводит к противоречию. □

Рассуждения, примененные для доказательства леммы 1, могут быть использованы и для обоснования более общего утверждения.

Теорема 2. *Предположим, что сетевая конфигурация C' образовалась из конфигурации C, после удаления множества правила коммутации пакетов U, и отношение $\sqsubseteq^+_{C'}$ не является частичным порядком на множестве U. Тогда никакой реконфигурационный блок $(Com, \prec)$, составленный из команд, оперирующих с равноприоритетными правилами коммутации, не является решением (C, C')-NRP.*

Чтобы найти решение (C, C')-NRP для произвольного отношения зависимости правил $\sqsubseteq_{C'}$, воспользуемся вспомогательными правилами коммутации с более высоким приоритетом. Для каждого восстанавливаемого правила коммутации вида $r_i = (G_i, A_i, 1),\ 1 \le i \le n$, расщепим его относительно правил множества U следующим образом: для каждого двоичного набора $\sigma = (\sigma_1, \ldots, \sigma_n)$ введем предохранитель $G_{i,\sigma} = G_i \wedge \bigwedge\limits_{j=1}^{n} \theta_{ij}^{\sigma_j}$, где $\theta_{ij}^1 = \theta_{ij}$ и $\theta_{ij}^0 = \neg\theta_{ij}$, а затем сформируем множество правил коммутации $Rules(r_i) = \{r_{i,\sigma} = (G_{i,\sigma}, A_i, 2) : \sigma \in \{0,1\}^n, G_{i,\sigma} \not\equiv false\}$. Обозначим символом Z конфигурацию, полученную из конфигурации C за счет добавления всех правил множества $Rules(r_i)$ в таблицу коммутатора w_i для каждого $i,\ 1 \le i \le n$. Ясно, что Z — однозначная сетевая конфигурация, задающая то же самое отношение одношаговой коммутации, что и конфигурация C'.

Лемма 2. *Если в конфигурации C' отсутствуют топологические циклы (т.е. ни один маршрут не проходит дважды через один и тот же коммутатор), то транзитивное замыкание отношения зависимости $\sqsubseteq^+_Z$ является частичным порядком на множестве правил коммутации $\bigcup\limits_{i=1}^{n} Rules_i$.*

Доказательство. Прежде всего заметим, что в конфигурации Z отношение зависимости $\sqsubseteq^+_Z$ на множестве правил $Rules$ транзитивно замкнуто, т.е. $(Rules, \sqsubseteq_Z) = (Rules, \sqsubseteq^+_Z)$. Для обоснования этого утверждения достаточно рассмотреть произвольную цепочку правил коммутации из множества $Rules$, последовательно сравнимых друг с другом по отношению зависимости

$$r_{i_k,\sigma_k} \sqsubseteq_Z r_{i_{k-1},\sigma_{k-1}} \sqsubseteq_Z \cdots \sqsubseteq_Z r_{i_2,\sigma_2} \sqsubseteq_Z r_{i_1,\sigma_1}\ ,$$

и показать индукцией по длине k этой цепочки, что $r_{i_k,\sigma_k} \sqsubseteq_Z r_{i_1,\sigma_1}$. Здесь используются особенности определенных выше расщепленных правил коммутации пакетов.

Далее, допустим, что $\sqsubseteq^+_Z$ не является частичным порядком и есть последовательность правил, находящихся в отношении зависимости и образующая цикл

$$r_{i_1,\sigma_1} \sqsubseteq_Z r_{i_(k-1),\sigma_(k-1)} \sqsubseteq_Z \cdots \sqsubseteq_Z r_{i_2,\sigma_2} \sqsubseteq_Z r_{i_1,\sigma_1}\ .$$

Тогда верно соотношение $r_{i_1} \sqsubseteq_Z r_{i_1}$, означающее, что в конфигурации Z имеется топологический цикл, проходящий через коммутатор, в таблице которого располагается правило r_{i_1} . Поскольку, $R_Z = R_{C'}$, такой же топологический цикл имеется и в конфигурации C', что противоречит условию леммы. Источником обнаруженного противоречия является предположение о взаимной зависимости некоторой пары правил коммутации в конфигурации Z. □

Можно построить следующее решение (C, C')-NRP. Пусть $Com_1 = \{add(r_{i,\sigma}, w_i) : r_{i,\sigma} \in Rules_i\}$, $Com_2 = \{add(r_i, w_i) : 1 \le i \le n\}$, и $Com_3 = \{del(w_i, G_i, 2) : 1 \le i \le n\}$.

Теорема 3. *Предположим, что конфигурация C' не содержит топологических циклов, а конфигурация C образовалась из C' удалением множества правил коммутации U. Тогда реконфигурационный блок $(Com_1, \prec_Z); (Com_2, \emptyset); (Com_3, \emptyset)$ является решением задачи (C, C')-NRP.*

Доказательство. Рассмотрим конфигурацию Z, которая образуется из конфигурации C, после того как все правила из каждого множеств $Rules_i$ будут вставлены в таблицы соответствующих коммутаторов w_i, $1 \le i \le n$. Из леммы 2 и теоремы 1 следует, что блок $(Com_1, \prec_Z)$ является решением задачи (C, Z)–NRP, причем $Path(C') = Path(Z)$. Таким образом, блок $(Com_1, \prec_Z)$ восстанавливает все утраченные полные маршруты конфигурации C'. Далее блок $(Com_2, \emptyset)$ добавляет в соответствующие таблицы коммутации низкоприоритетные правила коммутации $r_1, r_2, \ldots, r_n$. Обозначим конфигурацию, образующуюся в результате применения этого блока записью Z'. Поскольку для каждого такого правила r_i выполняется равенство $G_{r_i} = \bigvee_{r_{i,\sigma} \in Rules_i} G_{r_{i,\sigma}}$, каждое низкоприоритетное правило r_i, будучи вставленным в таблицу коммутатора w_i в конфигурации Z, оказывается блокированным совокупностью высокоприоритетных правил множества $Rules_i$. Поэтому ни отношение одношаговой коммутации, ни множество полных путей в процессе выполнения блока $(Com_2, \emptyset)$ не изменяется, т. е. $R_Z = R_{Z'}$ и $Path(Z) = Path(Z')$. На третьем этапе все высокоприоритетные правила коммутации вида $r_{i,\sigma}$ удаляются в произвольном порядке. Но, как следует из определения правил этого вида, отношение коммутации каждого правила $r_{i,\sigma}$ является подмножеством отношения коммутации правила r_i. Поэтому при удалении правила $r_{i,\sigma}$ немедленно разблокируется та «часть» низкоприоритетного правила r_i, которая полностью возмещает функциональность удаленного высокоприоритетного правила $r_{i,\sigma}$. Таким образом, в процессе выполнения блока $(Com_3, \emptyset)$ сохраняется отношение одношаговой коммутации и множество полных маршрутов конфигурации Z', т. е. $Path(Z') = Path(C')$. Нетрудно видеть, что после удаления всех высокоприоритетных правил коммутации конфигурация Z' преобразуется в конфигурацию C'. Значит, реконфигурационный блок $(Com_1, \prec_Z); (Com_2, \emptyset); (Com_3, \emptyset)$ является решением задачи (C, C')-NRP. □

Решение, предложенное в теореме 3, не является оптимальным; более избирательное расщепление правил множества U позволяет значительно сократить размер реконфигурационного блока. Но если конфигурации C и C' содержат правила коммутации с разными приоритетами, гарантировать решение (C, C')-NRP уже нельзя.

Рассмотрим конфигурацию C', изображенную на Рис. 2(а), в которой есть высокоприоритетные правила (например, $r_1 : f : 2$) и низкоприоритетные правила

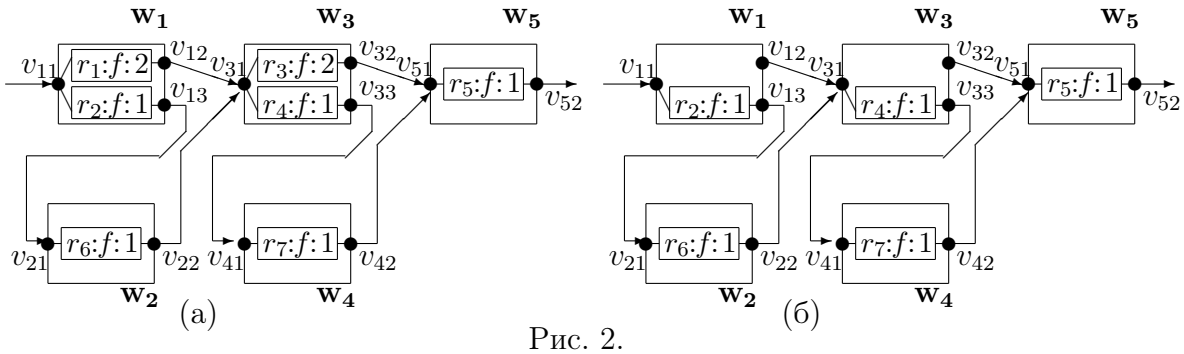

Рис. 2.

(например, $r_2 : f : 1$). Будем считать, что $H = \{f\}$. Так как правила с высоким приоритетом подавляют правила с низким приоритетом, в конфигурации C' есть только один маршрут $p_1 = (f, v_{11}), (f, v_{31}), (f, v_{51}), (f, v_{52})$. Однако стоит удалить высокоприоритетные правила r_1 и r_3, как конфигурация C' превращается в конфигурацию C, изображенную на Рис. 2(б). В ней активизируются низкоприоритетные правила в коммутаторах w_1 и w_3, и они прокладывают маршрут $p_2 = (f, v_{11}), (f, v_{21}), (f, v_{31}), (f, v_{41}), (f, v_{51}), (f, v_{52})$. Это единственный маршрут в конфигурации C. Но теперь плавно восстановить конфигурацию C' из конфигурации C при помощи обычных реконфигурационных команд уже невозможно.

Теорема 4. *Для сетевых конфигураций C' и C, изображенных на рис. 2, (C, C')-NRP не имеет решения.*

Доказательство. Обозначим записью $path'$ маршрут пакетов с заголовком f в конфигурации C', а записью $path$ маршрут пакетов с заголовком f в конфигурации C. Как видно из устройства правил коммутации пакетов в таблицах коммутаторов, эти маршруты отличаются друг от друга переходами из состояний пакетов в коммутаторах w_1 и w_3. Предположим, что существует реконфигурационный блок, являющийся решением (C, C')-NRP. Рассмотрим произвольную линеаризацию α этого блока. В последовательности команд α выделим кратчайший префикс β, удовлетворяющий требованию $path' \in \beta(C)$. Поскольку $path' \notin Path(C)$, последовательность команд β непуста, и, значит, имеет вид $\beta = \beta', com$, где com — некоторая команда реконфигурирования сети. Как следует из выбора префикса β, верно $path' \notin \beta'(C)$. Согласно определению решения (C, C')-NRP, должно выполняться включение $Path(\beta'(C)) \subseteq Path(C) \cup Path(C')$. Значит, $path \in Path(\beta'(C))$. Таким образом, команда com должна изменять конфигурацию $\beta'(C)$ так, чтобы маршрут $path$ преобразовался в маршрут $path'$. Но команда com не может этого сделать, т. к. маршруты $path$ и $path'$ имеют разные переходы в двух коммутаторах, а команда реконфигурирования может изменить отношение одношаговой коммутации только для одного коммутатора. Следовательно, реконфигурационного блока, являющегося решением (C, C')-NRP, не существует. □

4. Заключение

Мы рассмотрели один из простейших вариантов задачи реконфигурирования SDN — задачу восстановления конфигураций (NRP). Но большой интерес представляют и

другие варианты NUP (помимо задачи глобального реконфигурирования, которую мы обсудили во введении).

1). *Синтез сетевых конфигураций.* Для заданного постусловия Ψ построить конфигурацию C', удовлетворяющую требованию $C' \models \Psi$. Начальная конфигурация C несущественна: чтобы достичь C', достаточно удалить все исходные правила коммутации пакетов и вставить на их место правила коммутации конфигурации C'. Исследование этого варианта NUP было предпринято в статье [13].

2). *Локальное корректное реконфигурирование сетей.* Задача состоит в том, чтобы преобразовать конфигурацию C в новую конфигурацию C' за счет открытия множества новых полных маршрутов P_{add} и закрытия множества старых маршрутов P_{del}. Требование корректности предусматривает, что в ходе реконфигурации все остальные маршруты конфигурации C сохраняются без изменений. Задача локального реконфигурирования рассматривалась в работах [6, 7, 8, 9].

3). *Оптимизация сетевых конфигураций.* Для заданной конфигурации C и некоторой заданной числовой характеристики конфигураций $f(X)$ (ею может быть общее число правил коммутации пакетов или максимальное число правил в отдельной таблице коммутации) построить оптимальную конфигурацию C', которая реализует то же самое отношение одношаговой коммутации. Некоторые предварительные результаты исследования указанной задачи представлены в статье [15].

Список литературы

1. McKeown N., Anderson T., Balakrishnan H., et al. Openflow: Enabling Innovation in Campus Networks // *SIGCOMM Computer Communication Review.* 2008. V. 38, N 2. P. 69–74.

2. Foster N., Guha A., Reitblatt M., et al., Languages for Software-Defined Networks // *IEEE Communications Magazine.* February 2013. P. 128–134.

3. T. S. E. N. Zheng Cai, A. L. Cox. Maestro: A System for Scalable OpenFlow Control // Technical Report TR10-08. Rice University, 2010.

4. Voellmy A., Kim H., Feamster N. Procera: A Language for High-Level Reactive Network Control // *Proceedings of the 1-st Workshop on Hot Topics in Software Defined Networks.* 2012. P. 43–48.

5. OpenFlow Switch Specification. Version 1.4.0, October 14, 2013. https://www.opennetworking.org.

6. Francois P., Shand M., Bonaventure O. Disruption-free topology reconfiguration in OSPF networks // *IEEE INFOCOM.* May 2007.

7. Francois P., Coste P.-A., Decraene B., Bonaventure O. Avoiding disruptions during maintenance operations on BGP sessions // *IEEE Transactions on Network and Service Management.* 2007. V. 4. N 7. P. 1–11.

8. S. Raza, Y. Zhu, C.-N. Chuah. Graceful network state migrations // *IEEE/ACM Transactions on Networking.* 2011. V. 19, N 4. P. 1097–1110.

9. Vanbever L., Vissicchio S., Pelsser C., Francois P., Bonaventure O. Seamless network-wide IGP migration // *ACM SIGCOMM Computer Communication Review - SIGCOMM '11*. 2011. V. 41, N 4. P. 314–325.

10. Reitblatt M., Foster N., Rexford J., Walker D. Consistent updates for software-defined networks: change you can believe in! // *HotNets*. 2011. V. 7.

11. Reitblatt M., Foster N., Rexford J., Schlesinger C., Walker D. Abstractions for Network Update // *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 2012. P. 323–334.

12. Katta N.P., Rexford J., Walker D. Incremental Consistent Updates // *Proceedings of the 2-nd Workshop on Hot Topics in Software Defined Networks*. 2013. P. 49–54.

13. Noyes A., Warszawski T., Cernyand P., Foster N. Toward Synthesis of Network Updates // *Proceedings of the 2-nd Workshop on Synthesis*. July 13–14. 2013. Saint Petersburg, Russia, 2013.

14. A. X. Liu, C. R. Meiners, and E. Torng. TCAM Razor: A systematic approach towards minimizing packet classifiers in TCAMs // *IEEE/ACM Transactions on Networking*. 2010. V. 18. P. 490–500.

15. Kogan K., Nikolenko S.I., Culhane W., Eugster P., Ruan E. Towards efficient implementation of packet classifiers // *Proceedings of the 2-nd Workshop on Hot Topics in Software Defined Networks*, 2013.

16. Захаров В.А., Смелянский Р.Л., Чемерицкий Е.В. Формальная модель и задачи верификации программно-конфигурируемых сетей // Моделирование и анализ информационных систем. 2013. Т. 20, №6. С. 36–51. [Zakharov V.A., Smelyansky R.L., Chemeritsky E.V. A Formal Model and Verification Problems for Software Defined Networks // Modeling and analysis of information systems. 2013. V. 20, No. 6. P. 36–51 (in Russian)].

17. Kazemian P., Chang M., Zeng H., Varghese G., McKeown N., Whyte S. Real Time Network Policy Checking using Header Space Analysis // *Proceedings of the 10-th USENIX Symposium on Networked Systems Design and Implementation*. 2013.

On the Update Problems for Software Defined Networks

Zakharov V.A., Chemeritsky E.V.

Lomonosov Moscow State University
Leninskiye Gory, 1-52, Moscow, GSP-1, 119991, Russia
Applied Research Center for Computer Networks

Keywords: software defined network, switch, controller, forwarding rule, packet, network update, route, post-condition, invariant

The designing of network update algorithms is urgent for the development of SDN control software. A particular case of Network Update Problem is that of restoring seamlessly a given network configuration after some packet forwarding rules have been disabled (say, at the expiry of their time-outs). We study this problem in the framework of a formal model of SDN, develop correct and safe network recovering algorithms, and show that in general case there is no way to restore network configuration seamlessly without referring to priorities of packet forwarding rules.

Сведения об авторах:
Захаров Владимир Анатольевич,
Московский государственный университет им. М.В. Ломоносова,
профессор;
Чемерицкий Евгений Викторович,
Московский государственный университет им. М.В. Ломоносова,
аспирант факультета ВМК.