УДК 621.391

Е.Б. ТКАЧЕВА, канд. техн. наук, ИССАМ СААД, МОХАММЕД ДЖАМАЛ САЛИМ

ПРИМЕНЕНИЕ ASCR ДЛЯ ВЕРИФИКАЦИИ ПРОТОКОЛОВ УПРАВЛЕНИЯ SOFTWARE-DEFINED NETWORK

Введение

В современных инфокоммуникационных системах значительное внимание уделено интерактивным приложениям и сервисам реального времени. К подобным сервисам относятся трансляция потокового видео, IP-телефония, видеоконференцсвязь, центры обработки данных, игровые сервера и другие. На качество интерактивных сервисов значительное влияние имеет среда передачи данных: условия функционирования сетевых элементов и сети в целом.

Концепция Software-Defined Network (SDN) позволяет максимально адаптировать условия среды передачи данных под требования сервисов с предоставлением заданного уровня QoS [1]. «Автоподстройка» сети под определенный сервис обеспечивается благодаря уровню управления, введенному в SDN. Однако эффективное функционирование сетей, построенных на основе концепции SDN, осложнено рядом факторов: некорректным распределением ресурсов, избыточностью при взаимодействии различных реализаций протоколов управления, возникновением противоречий при функционировании оборудования различных производителей.

Анализ решений [2 – 4] показал, что большинство ошибок возникает из-за отличий в функциональности протоколов, заложенной в процессе разработки, различных интерпретаций спецификаций, несовместимости требований и логических ошибок при проектировании сетей.

Одним из методов раннего обнаружения и устранения подобного рода ошибок является верификация. Основной целью верификации является доказательство того, что результат функционирования элементов сети полностью соответствует требованиям и задачам, определенным в спецификации [5]. В рамках парадигмы SDN наряду с тестированием применимы формальные методы верификации, которые позволяют исследовать поведенческие свойства протоколов с помощью теоретико-обоснованного похода.

Существующие методы верификации, такие как SPIN[6], PRISM [7], SAL [8] или Alloy [9], имеют ряд существенных ограничений. Ограничения накладывают базовые логики их работы (LTL, CTL, PCTL, FOL μ-calculus), которые не позволяют учитывать множество особенностей SDN.

Расширение математического аппарата и формальных логик, которые лежат в основе инструментов верификации, позволит адаптировать существующие методы под особенности функционирования SDN. Таким образом, актуальна задача развития и усовершенствования существующих методов формальной верификации сетей и протоколов, построенных на основе концепции SDN.

Обзор протоколов уровня управления в сетях, построенных на основе концепции SDN

Основным протоколом SDN, функционирующим на уровне управления, является протокол OpenFlow [10]. Он управляет процессом передачи и обработки данных, поступающих от оборудования уровня передачи данных (маршрутизаторов, коммутаторов, серверов, центров обработки данных и других). В процессе установления и управления соединением между OpenFlow коммутаторами и контроллером SDN значительную роль также играют протокол OF-CONFIG [11] и OpenFlow Discovery Protocol (OFDP) [12].

OF-CONFIG предоставляет возможность удаленного конфигурирования параметров SDN оборудования. Так, OF-CONFIG принимает участие в процессе формирования таблиц переадресации и принятии решений относительно действий протокола OpenFlow.

OpenFlow Discovery Protocol дает возможность исследовать топологию сети, осуществить сбор статистических данных и уведомлять об изменениях в структуре сети. Протокол OFDP использует протокол канального уровня Link Logical Discovery Protocol (LLDP).

На рис. 1 приведены основные этапы обмена сообщениями между контроллером и коммутатором на уровне управления (control plan).

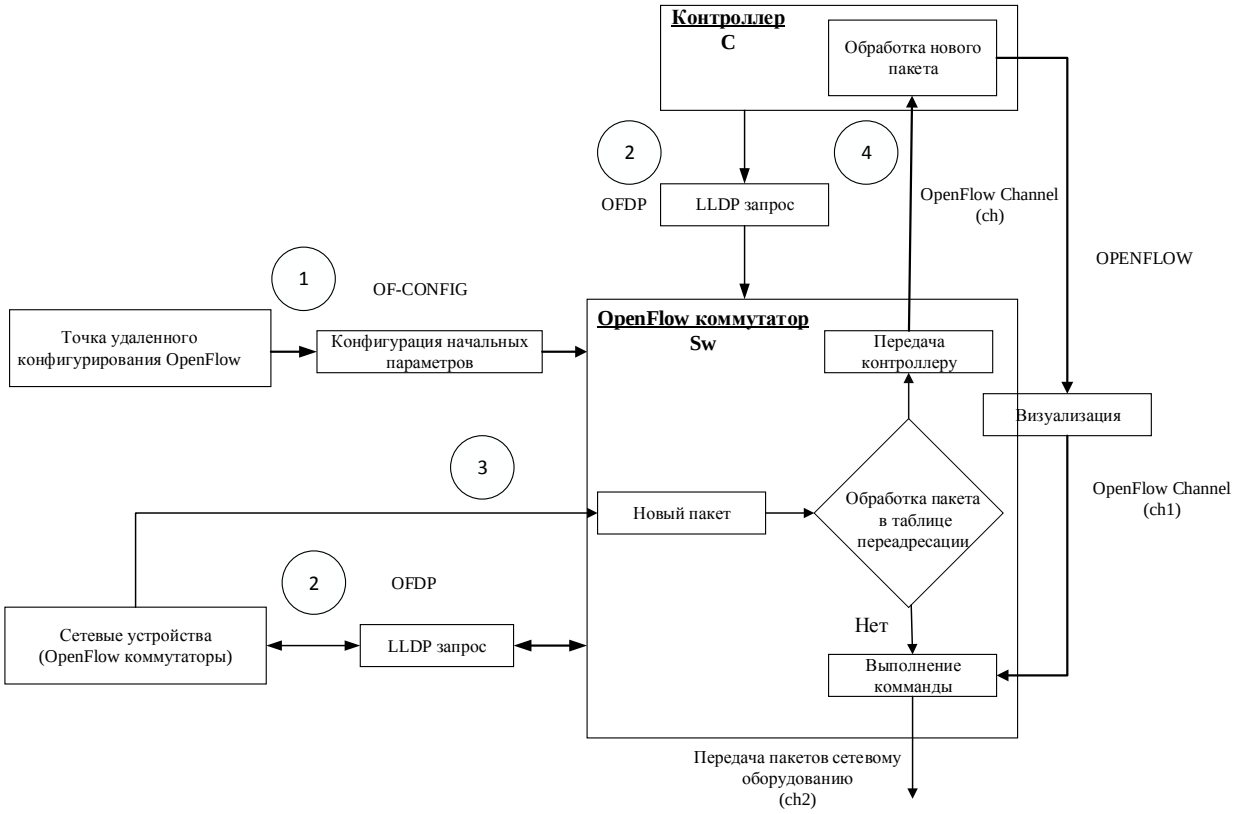


Рис. 1. Взаимодействие между элементами SDN архитектуры

1.OF-CONFIG является первичным протоколом, который необходим для полноценного функционирования SDN. Протокол OF-CONFIG позволяет конфигурировать статические параметры OpenFlow коммутатора, выделяет диапазоны виртуальных и физических портов, устанавливает приоритетность передаваемых данных, тип QoS и другие. В качестве точки удаленного конфигурирования может выступать как сетевой сервис (приложение), так и администратор сети. Спецификация протокола OF-CONFIG не дает рекомендаций относительно процесса конфигурирования и последующей передачи данных OpenFlow коммутатору.

2.Контроллер SDN осуществляет сбор информации о начальной топологии сети. Сбор сведений о сетевой топологии осуществляется при помощи OpenFlow Discovery Protocol (рассылки пакетов протокола LLDP). В этом случае контроллер отправляет на все коммутаторы своей зоны пакет LLDP, который должен быть разослан на все порты коммутатора. Коммутатор может пересылать пакет соседним коммутаторам или послать ответный пакет контроллеру.

3.OpenFlow коммутатор на один из портов получает сообщение, содержащее служебную информацию о типе пересылаемых данных. Взаимосвязь между OpenFlow коммутато-

ром и другим сетевым оборудованием может осуществляться при помощи любого протокола передачи данных. Обработка пакета происходит в соответствии с алгоритмом, который приведен на рис. 1.

4.Перенаправление пакета контроллеру осуществляется в том случае, если в таблице переадресации коммутатора не найдено ни одного совпадения для поля *Match* [10, 13]. В поле *Match* пересылаемого пакета содержится следующая информация: номер входящего порта, Ethernet адреса источника и получателя, тип Ethernet пакета, идентификатор и приоритет VLAN, IP адреса источника и получателя, тип протокола IP, Type of Service (ToS) биты протокола IP и TCP/UDP порты источника и получателя временные метки.

Стоит отметить, что для протокола OFDP сегодня нет официальных стандартов, таким образом, каждый производитель оборудования закладывает свои требования при его разработке, версии спецификаций протокола OF-CONFIG содержат ограниченный набор требований и алгоритмов (UML диаграммы), а основным недостатком OpenFlow v.1.0 отсутствие поддержки IPv6, multicast запросов. Для наглядности в таблице приведены основные отличия спецификаций протокола OpenFlow.

Функциональность	Версия 1.1	Версия 1.2	Версия 1.3
Поддержка нескольких таблиц переадресации	+	+	+
Виртуальные порты	+	+	+
Мониторинг отказов подключения к контроллеру	+	+	+
Multicast трансляция		+	+
Базовая поддержка IPv6		+	+
Механизм мониторинга смены ролей		+	+
Расширенная поддержка IPv6			+
Туннелирование ID метаданных			

Таким образом, основные проблемы, возникающие в процессе функционирования сетей, построенных на основе концепции SDN, связаны с отсутствием стандартизированных протоколов взаимодействия, наличием неопределенностей и расхождений в существующих версиях спецификаций протоколов (таблица). Отсутствие единых стандартов существенно влияет на логику функционирования оборудования: производителям необходимо учитывать, формализировать и внедрять в свои решения требования, заложенные в различных версиях OpenFlow протоколов, при этом логика работы сетевых элементов может значительно различаться.

Для выявления критических ошибок функционирования протоколов и их устранения без существенных временных и материальных затрат необходимо проведение анализа и верификации.

Обзор существующих подходов к верификации SDN

На сегодняшний день существуют два основных подхода для оценки функционирования и верификации протоколов SDN [13].

Первый подход базируется на принципах имитационного моделирования и тестовой проверки моделей протоколов и элементов SDN. Например, аппарат NICE разработан для проверки корректности функционирования SDN контроллера. NICE позволяет проверить такие свойства контроллера, как зацикливание и образование тупиковых ситуации при обработке запросов. Однако недостатком NICE является отсутствие возможности проверки других процессов (взаимодействие с коммутатором). Данная особенность вносит существенные ограничения и не позволяет выполнить верификацию всей сети.

NDB отладчик [14] позволяет выполнять как автоматическую верификацию, так и анализ фиксированных состояний протоколов управления. Недостаткам NDB является одновре-

менная проверка лишь небольшого сегмента сети, как правило, одной зоны SDN контроллера.

Преимуществом программ-эмуляторов, как Mininet, GENY [15] является анализ поведения протоколов управления, который позволяет учитывать реальные временные характеристики и параметры сети. Применение Mininet, GENY, как и любого инструмента имитационного моделирования не позволяет определить все граничные состояния протоколов.

Второй подход базируется на формальных методах верификации: SPIN[6], PRISM [7], SAL [8] и Alloy [9]. При применении данных методов осуществляется систематизированный поиск и анализ множества пространства состояний, достижимых в процессе функционирования модели сети или сетевых протоколов. Преимуществом подобных формальных методов является их выразительная мощность: с их помощью может быть сформирована любая система взаимодействующих процессов.

Однако применение основных из них для верификации сетей, построенных на основе концепции SDN, затруднительно, так как базовые формальные логики (LTL, μ-calculus, PCTL) не позволяют полноценно формализовать приоритетность процессов, время их функционирования, доступность ресурсов и другие. Таким образом, необходима разработка метода формализации и верификации SDN, который позволяет полноценно и в то же время универсально описать все характерные для SDN свойства

Применение аппарата алгебры распределенных коммуникационных ресурсов для верификации протоколов управления в SDN

В качестве математического аппарата формализации и последующей верификации протоколов уровня управления SDN предлагается использовать алгебру распределенных коммуникационных ресурсов (ACSR) [16]. Аппарат ACSR был разработан для формальной верификации сложных интегрированных систем, которые функционируют в режиме реального времени и чувствительны к временным задержкам и срокам выполнения процессов. Математический аппарат алгебры распределенных коммуникационных ресурсов позволяет задать приоритет выполнения процессов, формализовать параллельное выполнение нескольких процессов, распределение ресурсов и последовательность их смены во времени.

ACSR определяет два типа действий:

1. Длительные процессы: учитывается длительность выполнения процессов, имеющих место при функционировании протоколов, и их зависимость от потребляемых ресурсов (например, поведение SDN контроллеров);

2. Мгновенные события: определяется детерминированная смена состояний протоколов (например, синхронизацию между процесами).

Каждый процесс в соответствии с ACSR может быть определен множеством

$$P ::= \Theta \mid A^u : P \mid e.P \mid P + Q \mid P \| Q \mid P \Delta_u Q \mid P \perp Q \mid [P]_U \mid P \setminus F \mid recX.P \mid X\,, \qquad (1)$$

где оператор Θ указывает на то, что элемент (процесс) не выполняет никаких действий (например, определяет начальную конфигурацию сети или возникновение тупика в процессе функционирования).

Оператор $A^u : P$ соответствует занятости ресурсов сети на протяжении определенного времени u в результате выполнения процесса A, которые по истечению времени u переходят к процессу P. Оператор $e.P$ соответствует выполнению мгновенного события e (применим, когда нет необходимости учета времени) и переходу к процессу P.

Оператор вида $P + Q$ описывает недетерминированный процесс, в котором любой из процессов P или Q может быть выбран с равной вероятностью. Оператор $P \| Q$ соответствует параллельному выполнению двух процессов P и Q. Оператор $P \Delta_u Q$ определяет условие, при котором процесс P выполняется в течение времени u включительно. По истечению

этого времени процесс P останавливается и начинается выполнение процесса Q. Оператор $P \perp Q$ указывает на то, что выполнение процесса P может быть прекращено в любое время в пользу выполнения процесса Q. $[P]_U$ указывает на то, что в течение времени u будут выполняться процессы, результатом которых является процесс P. $P \setminus F$ является оператором ограничения. $recX.P$ описывает бесконечную рекурсию.

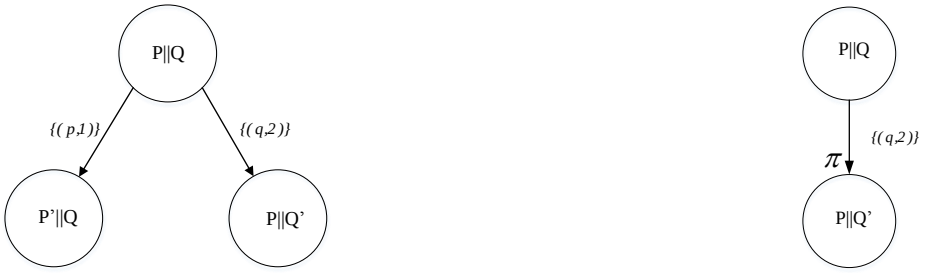
С помощью ACSR взаимосвязь между событиями может быть описана следующим типом событий:

- мгновенные события;
- связь типа «точка-точка» по различным каналам связи;
- приоритетный доступ к каналам;
- входные и выходные события.

Математический аппарат ACSR также позволяет формализовать:

- конфликты, возникающие между сетевыми ресурсами: если $P = \{(p,1)\}:P'$ и $Q = \{(p,1)\}:Q'$, то $P \| Q \sim NIL$,
- первоочередность выполнения одного из нескольких процессов: $P = \{(p,1)\}:P' + \Theta:P$ – первоочередным является выполнение процесса, соответствующего первому слагаемому,
- приоритетность выполнения процессов: $P \| Q \Rightarrow P' \| Q:\{(p,1)\} \prec P \| Q':\{(q,2)\}$ может быть выражена как $P \| Q \rightarrow_{\pi} P \| Q':\{(q,2)\}$.

Графическая интерпретация формул ACSR выполняется с помощью вероятностно временных графов. Так, визуализация приоритета выполнения того или иного процесса может быть представлена следующим образом (рис. 2):



а *б*

Рис. 2. Графическая интерпретация ACSR формализмов $P \| Q \Rightarrow P' \| Q:\{(p,1)\} \prec P \| Q':\{(q,2)\}$ и $P \| Q \rightarrow_{\pi} P \| Q':\{(q,2)\}$: отсутствие приоритета выполнения между процессами (*а*), приоритетное выполнение процесса (*б*)

Пример применения ACSR. Формализация обмена сообщениями между OpenFlow коммутатором и контроллером

В качестве примера применения приведем начальный обмен сообщениями между коммутатором OpenFlow и контроллером, схема которого представлена на рис. 3.

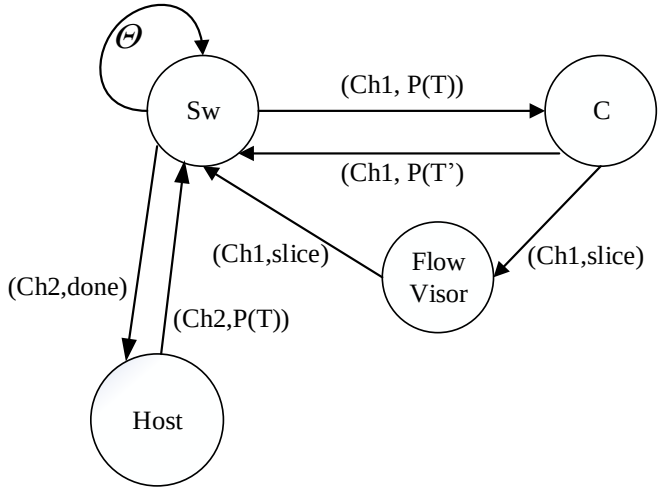


Рис. 3. Обмен сообщениями между OpenFlow коммутатором и контроллером

Пакет $P(T)$, который передается от коммутатора к контроллеру по каналу (*ch1*) имеет определенный набор атрибутов, которые задают правила последующей его обработки ($MatchP(T)$). Мощность аппарата ACSR позволяет формализовать набор атрибутов пакета $P(T)$ с разной степенью детализации.

Обобщенное представление передачи сообщения может быть заданно как

$$Sw \xrightarrow{\Theta} Sw \xrightarrow{ch2,P(T)} C \xrightarrow{ch1,P(T')} Sw \xrightarrow{ch2,done} H,$$

$$Sw \xrightarrow{\Theta} Sw \xrightarrow{ch2,P(T)} C \xrightarrow{ch1,slice} FL \xrightarrow{ch1,slice} Sw \xrightarrow{ch2,done} H.$$

Входные события определяются следующим формализмом: $(e?, P_i)$, выходные – $(e!, P_i)$.

На основе правил ACSR могут быть построены следующие базовые формализмы, которые соответствуют требованиям спецификации протокола OpenFlow [17]:

- коммутатор (*Sw*) находится в неактивном сосотоянии – новые пакеты не поступают:

$$Sw := \{\Theta\} : Sw \tag{3}$$

- коммутатор (*Sw*) перенаправляет новый пакет $P(T)$, контроллеру по каналу *ch1*:

$$Sw := ch1!C \tag{4}$$

- коммутатор (*Sw*) получает новый пакет $P(T)$ от любого физического устройства по каналу *ch2*:

$$Sw := ch2?Host \tag{5}$$

- коммутатор (*Sw*) одновременно может получать пакеты от контроллера или других физических устройств сети по каналу *ch1* и *ch2* соответственно:

$$Sw := ch1?P(T).C(T) + ch2?P(T).Host(T); \tag{6}$$

- перенаправление пакета контроллеру $P(T)$ осуществляется в том случае, если в таблице переадресации коммутатора не найдено ни одного совпадения для поля $Match_j$:

$$Sw(T) := match_j SrcIP(T,P_0) \rightarrow ch2!Host(T) + \sim match_j SrcIP(T,P_0) \rightarrow ch1!C(T) + e.R(T); \tag{7}$$

- коммутатор (*Sw*) выполняет проверку IP-адреса пересылаемого пакета $P(T)$, в случае, если IP-адрес не принадлежит диапазону сети, отбрасывает пакет:

$$Sw(x) := matchSrcIP(T,P_i) \rightarrow \{\} : Drop(T). \tag{8}$$

Кроме того, для корректной оценки функционирования реализации протоколов управления SDN крайне важен анализ таких свойств, как:

• отсутствие петель и зацикливаний в процессе функционирования протокола: любой пакет должен быть доставлен непосредственно получателю, который идентифицирован в поле destination sourse таблицы переадресации. С помощью ACSR данное свойство может быть представлено следующим образом:

$$Swi := matchIP(T(dest), P_0) \to chi!(T) \setminus Swi := matchIP(T(dest), P_0) \to chi?(T); \quad (9)$$

• живость сети: все элементы сети, участвующие в передачи данных должны быть доступны и активны:

$$SDN := P_0 \xrightarrow{ch_i} P', [P']_U, P' \notin \Theta; \quad (10)$$

• достижение заключительного состояния: все сетевые состояния, гарантирующие корректную работу протокола должны быть достигнуты;

$$SDN := (Sw1 || Sw2 || Swi || Host)/\{ch1,ch2,ch3,chi\}; \quad (11)$$

• непротиворечивость существующих версий протоколов управления и их эффективное совместное функционирование (в случае использования оборудования OpenFlow различных производителей):

$$SDN := \{R,Conf\}: Sw1 \setminus \{R,Conf\}: Sw2 \setminus \{R,Conf\}: Swi. \quad (12)$$

Таким образом, математический аппарат алгебры распределенных коммуникационных ресурсов обеспечивает детальное представление процессов, имеющих место на уровне управления SDN, в частности позволяет формализовать процессы передачи и обработки пакетов OpenFlow коммутаторами. Наряду с этим ASCR позволяет описать свойства, характерные для всей сети в целом.

Заключение

Концепция Software-Defined Network призвана обеспечить требуемый уровень качества и гарантированную доступность сервисов, чувствительных к временным характеристикам сети, за счет разделение функций управления и передачи данных. Однако на пути развития сети, построенные на основе концепции SDN, сталкиваются с рядом трудностей, которые приводят к возникновению ряда критических ошибок.

Одним из методов проверки с последующим устранением ошибок в работе протоколов управления SDN является формальная верификация. Для формализации процессов предложено применение математического аппарата ACSR, который позволяет полноценно описать как мгновенные, так и длительные процессы, определить приоритет и очередность обработки пакетов, формализовать свойства которыми должны обладать контроллер и коммутатор OpenFlow.

Применение ACSR, в отличие от существующих методов, позволяет формализовать свойства каждого элемента взаимодействия как в отдельности, так и в рамках целостной системы. Так, формулы (3) – (8) определяют базовый обмен сообщениями между контроллером и коммутатором с помощью протокола OpenFlow. А формулы (9) – (12) определяют общие свойства протоколов управления и могут служить шаблоном для последующей проверки новых реализаций протоколов OpenFlow и OF-CONFIG.

Список литературы 1.Software-Defined Networking: The New Norm for Net works [Electronic resource] // Open Networking Foundation. – [2012]. – Режим доступа: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf 2.SDN Testing & Validation: SDN Solutions Showcase [Electronic resource] // Open Networking Foundation.- [2014]. – Режим доступа: https://www.opennetworking.org 3. *Benson T., Anand A., Akella A., and Zhang M.* Microte: fine grained traffic engineering for data centers. In Proceedings of the Seventh // Conference on emerging Networking EX-

periments and Technologies, ACM, 2011, 8 p. 4. Отчёт о НИР «создание прототипа отечественной пкс платформы управления сетевыми ресурсами и потоками с помощью сетевой операционной системы (сос) на основе анализа и оценки существующих сетевых операционных систем для пкс сетей и выбора одной из них для последующего» (Промежуточный) – [2013] – Режим доступа: cs.msu.ru/sites/cmc/files/scproj/4047_otchet_nir1_0.pdf 5. *Карпов Ю.Г.* Model Checking. Верификация параллельных и распределенных программных систем / Ю.Г. Карпов. – СПб.: БХВ-Петербург, 2010. – 552 с. 6. *Holzmann G.* The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, 2005. 7. *Kwiatkowska M., Norman G., and Parker D.* PRISM 4.0: Verification of Probabilistic Real-time Systems. In 23rd International Conference on Computer Aided Verification, pages 585–591. Springer, 2011. 8. *L. De Moura, S. Owre* In Computer Aided Verification pages 496–500. Springer, 2004. 9. *Jackson D.* Alloy: a lightweight object modelling notation. ACM Transactions on Software Engineering and Methodology, 11(2): p.256–290, Apr. 2002. 10. Openflow tutorial. [Electronic resource] – [2012]. – : http://www.openflow.org/wk/index.php/OpenFlow_Tutorial 11. OpenFlow Management and Configuration Protocol (OF-CONFIG1.1.1) [Electronic resource] // Open Networking Foundation.– [2013] – Режим доступа: https://www.opennetworking.org/stand 12. OpenFlow Discovery Protocol and Link Layer Discovery Protocol Electronic resource] – [2011] – : http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol 13. *Skowyra R., Lapets A., Bestavros A.* A Verification Platform for SDN-Enabled Applications // Technical Report BUCS-TR-2012-016, 2012, Boston University 14. *N. Handigol, B. Heller, V. Jeyakumar, D. Mazi`eres, and N. McKeown*. Where is the debugger for my software-defined network? In Proceedings of the first workshop on Hot topics in software defined networks, p. 55–60. ACM, 2012. 15. *Kwiatkowska M., Norman G., and Parker D.* PRISM 4.0: Verification of Probabilistic Real-time Systems. In 23rd International Conference on Computer Aided Verification, pages 585–591. Springer, 2011 16. *Choi J., Lee I., and Xie H.*, "The specification and schedulability analysis of real-time systems using ACSR," in Proc. IEEE Real- Time Systems Symp., Dec. 1995. 17. *OpenFlow Switch* Specification (Series) [Electronic resource] // Open Networking Foundation. – [2014]. – Режим доступа: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow

Харьковский национальный
университет радиоэлектроники

Поступила в редколлегию 15.01.2015