Marek Bolanowski, Mateusz Wojtas

Możliwości bieżącej modyfikacji zachowania urządzeń sieciowych w procesie tworzenia własnych stanowisk naukowo-dydaktycznych

Edukacja - Technika - Informatyka nr 4(18), 345-351

2016

© Wydawnictwo UR 2016 „Edukacja – Technika – Informatyka” nr 4/18/2016
ISSN 2080-9069 www.eti.rzeszow.pl
ISSN 2450-9221 online DOI: 10.15584/eti.2016.4.45

MAREK BOLANOWSKI[1], **MATEUSZ WOJTAS**[2]

Możliwości bieżącej modyfikacji zachowania urządzeń sieciowych w procesie tworzenia własnych stanowisk naukowo-dydaktycznych

Possibilities of modification a running configuration of network device in the process of creating custom research and teaching stands

[1] Doktor inżynier, Politechnika Rzeszowska im. Ignacego Łukasiewicza, Wydział Elektrotechniki i Informatyki, Katedra Energoelektroniki, Elektroenergetyki i Systemów Złożonych, Polska
[2] Inżynier, Politechnika Rzeszowska im. Ignacego Łukasiewicza, Wydział Elektrotechniki i Informatyki, Katedra Energoelektroniki, Elektroenergetyki i Systemów Złożonych, Polska

Streszczenie

Artykuł prezentuje możliwości modyfikacji zachowania urządzeń sieciowych przez studentów i uczniów w procesie tworzenia własnych stanowisk naukowo-dydaktycznych. W artykule zaprezentowane zostały modele sterowania warstwą logiczną urządzenia, wskazano również najlepsze języki skryptowe, które mogą posłużyć do testowania własnych algorytmów.

Słowa kluczowe: sieci komputerowe, skrypty, urządzenia sieciowe.

Abstract

The article presents the possibility of modifying the behavior of network devices by students in the process of creating custom research and educational stands. In the article the models of control plane for logical device were presented, also the authors indicated the best scripting languages that can be used to test personal (custom) algorithms.

Key words: computer network, scripts, network devices.

Wstęp

Proces nauczania oraz realizacji prac badawczych w obszarze szeroko pojętej teleinformatyki jest niejednokrotnie związany z dużymi nakładami na budowę środowiska dydaktycznego czy też stanowiska badawczego. W obszarze edukacji możemy wyróżnić dwa podejścia do tworzenia stanowisk dydaktycznych:

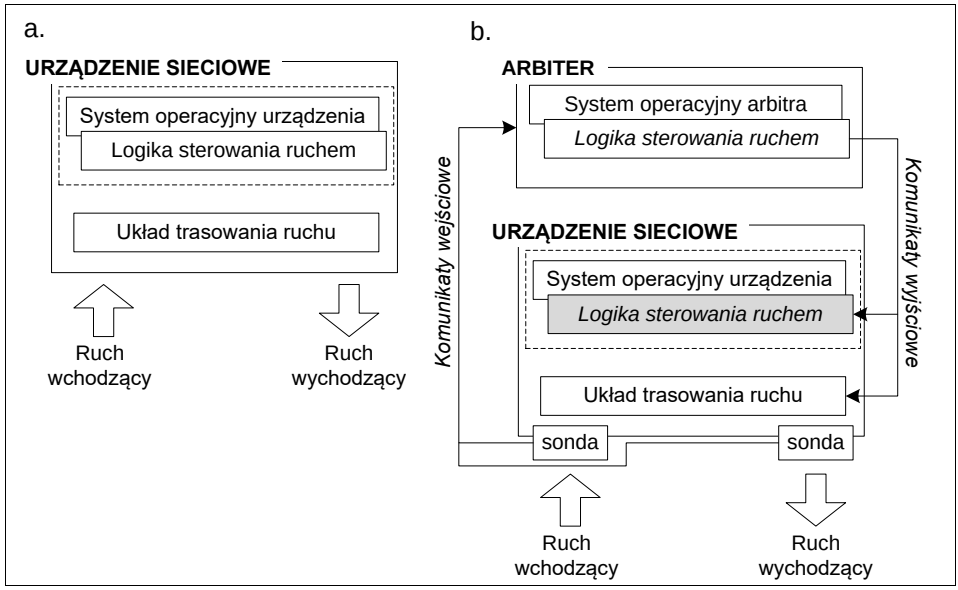
1. Zestaw dydaktyczny zbudowany w oparciu o środowisko programowe instalowane na standardowym komputerze klasy PC, wykorzystywane przy nauczaniu takich przedmiotów jak programowanie, grafika komputerowa, projektowanie CAD itp.
2. Zestaw dydaktyczny zbudowany w oparciu o stanowisko komputerowe oraz elementy innej infrastruktury, w tym fizyczne urządzenia sieciowe, obrabiarki numeryczne, urządzenia pomiarowe, roboty itp.

Oczywiście budowa stanowisk laboratoryjnych w oparciu o rzeczywiste elementy peryferyjne (podeście drugie) jest stosunkowo droga. W bardzo wielu przypadkach fizyczne elementy peryferyjne zastępowane są często przez ich modele w środowisku programowym. Często takie modele nie odpowiadają w pełni ich rzeczywistym wzorcom [Bolanowski, Krutys 2015]. Z tego powodu instytucje ponoszą duże wydatki na zakup rzeczywistych urządzeń do realizacji zadań i ćwiczeń dydaktycznych. Zakup takich elementów wyposażenia jest często dotowany w części lub w całości przez danego producenta. Dzięki temu instytucja dydaktyczna rozbudowuje swoją bazę laboratoryjną, a producent zyskuje kadrę wykwalifikowanych inżynierów, którzy potrafią obsługiwać jego urządzenia. Takie podejście jest wysoce pożądane na etapie kształcenia zawodowego i w procesie pozyskiwania umiejętności z zakresu wykorzystania standardów i technologii przemysłowych. Podkreślić należy, że jest ono charakterystyczne nie tylko w obszarze kształcenia na szczeblu szkół średnich, ale również w szkołach wyższych zarówno na etapie studiów inżynierskich, jak i magisterskich.

Współczesny rynek pracy potrzebuje jednak fachowców, którzy potrafią kreatywnie modyfikować działanie urządzeń, protokołów i technologii kilkukrotnie w trakcie cyklu życia danego urządzenia. Bardzo niewielka liczba producentów pozwala na ingerencję w oprogramowanie (system operacyjny) danego urządzenia, a przez to utrudnia lub wręcz uniemożliwia modyfikowanie ich standardowego zachowania. Jest to zrozumiałe ze względu na niezawodność tych urządzeń oraz bezpieczeństwo ich eksploatacji. Niemniej jednak takie podejście nie stwarza możliwości rozwoju uczniów i studentów w zakresie kreatywnego rozwiązywania problemów, tworzenia nowych rozwiązań i testowania ich w środowisku produkcyjnym. W niniejszym artykule autorzy skupili się na możliwości bieżącej modyfikacji zachowania urządzeń na przykładzie urządzeń sieciowych. W pracy oprócz modelu sterowania zaprezentowane zostały również wykorzystane protokoły oraz topologie stanowisk.

Model sterowania urządzeniem

Sterowanie urządzeniem sieciowym może być realizowane na dwa sposoby: z wykorzystaniem wyłącznie wewnętrznego systemu operacyjnego, który monitoruje prace urządzenia oraz realizuje cały algorytm sterowania, oraz z wykorzystaniem zewnętrznego arbitra. Oba przypadki zostały schematycznie przedstawione na rys. 1.



Rys. 1. Modele sterowania ruchem: a. klasyczny, z wykorzystaniem wewnętrznych mechanizmów urządzenia; b. z zastosowaniem arbitra

O ile model przedstawiony na rys 1a. jest modelem klasycznym i nie wymaga komentarza, o tyle model przedstawiony na rys. 1b wymaga wyjaśnienia. W modelu z arbitrem istnieje możliwość przeniesienia modułu logiki sterowania ruchem z urządzenia do zewnętrznego arbitra. Można wyróżnić dwa podejścia:

1. Z całkowitym przeniesieniem logiki sterowania do arbitra przy wykorzystaniu architektury SDN [Hyojoon, Feamster 2013]. Zastosowanie takiego podejścia oddaje w ręce studenta całkowitą kontrolę nad przepływami, ale wymaga od niego dużej wiedzy nie tylko z zakresu urządzeń sieciowych, ale również z obszaru programowania.
2. Z zastosowaniem podejścia hybrydowego – arbiter na bieżąco z wykorzystaniem skryptów analizuje przesyłany ruch i w momencie wystąpienia wcześniej zdefiniowanego zdarzenia wywoływany jest skrypt konfiguracyjny wysyłany do urządzenia sieciowego, który modyfikuje konfiguracje logiki sterowania ruchem w urządzeniu sieciowym. Takie podejście wydaje się odpowiednie w większości przypadków. Student nie musi zajmować się całym ruchem, ale steruje wybranymi przepływami z wykorzystaniem skryptów i wcześniej zdefiniowanych warunków granicznych dla zadanego parametru sieciowego. Od studenta wymaga się wiedzy z zakresu sieci komputerowych oraz podstaw obsługi skryptów.

Kluczowe dla obydwu przypadków jest ścisłe określenie zasad współpracy między arbitrem a urządzeniem sieciowym w szczególności określenie formatu

i sposobu przesyłania komunikatów wejściowych i wyjściowych pomiędzy urządzeniem a arbitrem oraz roli sond, które próbkują przesyłany ruch na urządzeniu sieciowym.

W chwili obecnej na rynku pojawiła się jednak spora grupa urządzeń, która posiada system operacyjny oparty na dobrze znanym systemie Linux lub też umożliwia uruchamiania własnych skryptów sieciowych. Dzięki temu idea arbitra została w pewien okrojony sposób zaszyta w samym urządzeniu. Uprościło to znacznie budowę stanowiska laboratoryjnego umożliwiającego automatyczne reagowanie na zdarzenia i bieżącą modyfikację zachowania urządzeń. W dalszej części autorzy skupią się właśnie na prezentacji tych technik. W prowadzonych pracach brano pod uwagę szereg urządzeń i systemów operacyjnych; ich zestawienie oraz obsługiwane języki skryptowe zostały pokazane w tabeli 1. Zgodnie z informacjami zawartymi w tabeli można zauważyć, że wszystkie zaprezentowane systemy pozwalają na automatyzację zadań za pomocą skryptów. Różnią się one jednak możliwościami, udostępniając użytkownikowi odmienne mechanizmy i środki do automatyzacji procesów. Zdecydowanie najczęściej urządzenia umożliwiają interpretowanie skryptów napisanych w Pythonie [Internet 3; Internet 4]. Dotyczy to systemów Cisco IOS [Internet 5], EXOS [Internet 6] oraz AOS [Internet 7]. Jest to zapewne uwarunkowane faktem, że ten język skryptowy jest uniwersalny, a jego stosowanie wygodne dla użytkowników.

Tabela 1. Porównanie systemów pracujących na urządzeniach sieciowych

Cechy systemu	Cisco IOS	ExtremeXOS	AOS	Junos OS	Netgear ProSafe OS
Automatyzacja procesów	TCL, Python	TCL, Python	Bash, Python	XSLT, SLAX	Skrypty konfiguracyjne (tylko komendy)
Zarządzanie	CLI, GUI, SNMP	CLI, GUI, SNMP	CLI, GUI, SNMP	CLI, GUI, SNMP	CLI, GUI, SNMP
Wsparcie REST API	Pełne	Odczyt	Pełne	Odczyt	Brak

Stosunkowo często występują również interpretery języka TCL, które zaimplementowano w Cisco IOS i EXOS. Warto zwrócić uwagę również na AOS potrafiący wykonywać skrypty napisane w bashu. Dzięki temu użytkownik wykorzystujący na co dzień system Linuxie może tworzyć programy automatyzujące procesy bez konieczności specjalnego przygotowania. Junos OS wspiera przede wszystkim skrypty oparte na składni XML (XSLT) bądź napisane przy użyciu autorskiego języka Juniper – SLAX. Przełączniki Netgear pozwalają jedynie na tworzenie skryptów konfiguracyjnych.

Funkcjonalność wszystkich omawianych systemów dopełnia wsparcie dla REST API [Zhou 2014]. Jest to narzędzie zyskujące coraz większą popularność,

także wśród rozwiązań sieciowych. Warto tutaj zwrócić uwagę, że zarówno Cisco IOS, jak i Alcatel-Lucent OS udostępniają zestaw metod umożliwiających odczyt i konfigurację urządzeń. Dzięki temu administratorzy wykorzystujący REST API nie będą mieć problemu ze zintegrowaniem tych urządzeń ze swoją siecią. Z kolei ExtremeXOS oraz Junos OS umożliwiają wykorzystanie REST API jedynie do odczytu konfiguracji i danych zapisanych na urządzeniu.

Możliwości bieżącej modyfikacji urządzeń

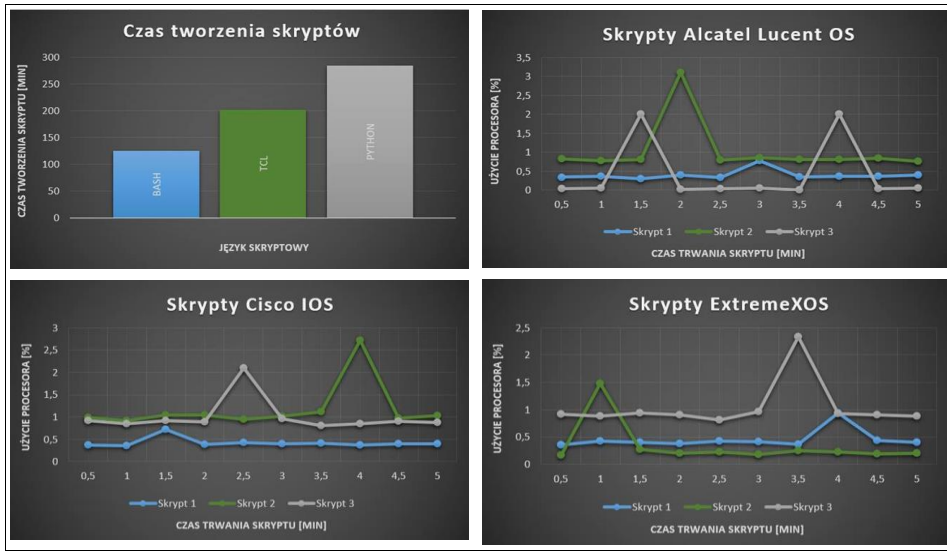
Rozważmy przykład zastosowania bieżącej możliwości modyfikowania konfiguracji urządzenia i jego zastosowanie w celu sprawdzenia działania metryk w protokole OSPF. Jeżeli posiadamy sieć działającą w oparciu o protokół OSPF, protokół ten samodzielnie nie zmienia tras (np. poprzez modyfikacje metryk) rutingu na podstawie bieżącej analizy obciążenia interfejsów czy też opóźnienia. Studenci mogą samodzielnie podjąć próbę modyfikacji metryk OSPF w celu sprawdzenia funkcjonowania autorskich algorytmów równoważenia obciążenia pomiędzy wieloma trasami o różnych kosztach. Podczas prac badawczych sprawdzono, jak do tego celu nadają się trzy przykładowe systemy operacyjne: Cisco IOS, Alcatel-Lucent (AOS), EXOS. W ramach prowadzonych badań sprawdzono te same zestawy czynności konfiguracyjnych realizowanych w oparciu od CLI, plik konfiguracyjny, zarządzenia zdalne i skrypty. Podczas pracy oceniano, jak szybko z wykorzystaniem danej metody można modyfikować bieżące zachowanie urządzenia, wygodę użycia, czas reakcji na zdarzenie oraz szybkość wykonania czynności konfiguracyjnych. Zestawienie wyników zostało przedstawione dla poszczególnych systemów operacyjnych w tabeli 2.

Tabela 2. Porównanie metod modyfikacji zachowania urządzeń dla Cisco IOS (Catalyst 3560 v2), EXOS (X440), AOS(6860, 6900).

Nazwa	Stopień trudności	Wygoda	Reagowanie na zdarzenia	Szybkość konfiguracji
CLI	Niski	Niska	Wolno	Wolno
Pliki konfiguracyjne	Średni	Wysoka (przy wstępnej konfiguracji)	Brak	Szybko
Zdalna konfiguracja	Średni	Średnia	Szybko	Wolno
Skrypty	Wysoki	Wysoka	Natychmiast	Szybko

Przeprowadzone badanie jednoznacznie pokazały, że najlepsze możliwości bieżącej modyfikacji urządzeń uzyskiwano za pomocą skryptów. Generalnie wyniki uzyskane dla każdego z urządzeń są podobne. W wyniku przeprowadzonych badań wytypowano też języki skryptowe, które zostały poddane dalszej analizie: Pyton, TCL, Bash. W celu sprawdzenia, który z języków skryptowych jest najlepszy do automatyzacji czynności administracyjnych, w każdym z języków napisano 3 identyczne skrypty administracyjne: backup konfiguracji, wy-

słanie komunikatu po wystąpieniu błędu, wysłanie komunikatu po przekroczeniu wartości progowej. Rysunek 2 przedstawia średni czas tworzenia skryptu przez studenta ze znajomością programowania, ale bez znajomości danych języków skryptowych, w zależności od wykorzystanego języka skryptowego, oraz czas wykonania skryptów na poszczególnych urządzeniach sieciowych.



Rys. 2. Czas przygotowania skryptów oraz procentowy wzrost obciążenia procesora podczas wykonywania skryptu

Wnioski

W pracy zaprezentowano możliwości modyfikacji konfiguracji urządzeń sieciowych przy wykorzystaniu skryptów. Dotychczasowe stanowiska realizowane były głównie w oparciu o środowisko z arbitrem. W pracy dokonano analizę możliwości wykorzystania mechanizmów wewnętrznych skryptów oraz wskazano zestaw języków skryptowych, które mogą zostać użyte do samodzielnego przygotowania stanowisk badawczego lub dydaktycznego. Dzięki takiemu podejściu studenci zyskują możliwość testowania własnych algorytmów i mechanizmów w środowisku produkcyjnych urządzeń sieciowych. Wyniki pracy mogą być również wykorzystane przy planowaniu architektury platformy sprzętowej własnego środowiska laboratoryjnego.

Literatura

Bolanowski M., Krutys P. (2015), *Metody i środki zarządzania infrastrukturą sieciową w złożonym środowisku laboratoryjnym*, „Edukacja – Technika – Informatyka" nr 3(13).

Hyojoon K., Feamster N. (2013), *Improving Network Management With Software Defined Networking*, „Communications Magazine, IEEE" vol. 51, issue 2.

Internet 1: https://www.python.org/.
Internet 2: http://pypl.github.io/PYPL.html.
Internet 3: http://www.cisco.com/c/en/us/support/index.html.
Internet 4: http://www.extremenetworks.com/support/documentation/.
Internet 5: http://enterprise.alcatel-lucent.com/countrysite/pl/?content=ResourceLibrary&page=overview.
Zhou W., Li L., Luo M., Chou M. (2014), REST API Design Patterns for SDN Northbound API, Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on, IEEE.