Cite This: H. Hristov, A. Ignatov, „Automated Management of Network Infrastructure“, *Mathematics and informatics*, vol. 62, no. 6, pp. 602-615, 2019, стр., ISSN 1314-8532 (Online), ISSN 1310-2230 (Print).

AUTOMATED MANAGEMENT OF NETWORK INFRASTRUCTURE

Summary

This article discusses network management automation problem. Difficulties encountered while expanding network infrastructure are addressed. An analysis of the path to automation in network management is given.

A process classification for network automation is proposed. A brief survey of the most popular network management tools is presented. A comparison between these and a custom tool contributed by the authors is made. The custom tool is used in “Information Services” Plc for network management.

In conclusion we shared some thoughts, about application extension, popularization and integration.

Keywords: ***network management, network automation, software development for network management***

1. Въведение.

IT индустрията непрекъснато разработва и развива нови технологии, средства и методи за мрежово администриране. Много от тях се предлагат с конкурентни предимства, които системният администратор е необходимо да познава.

Разширяването на мрежовата архитектура в една компания води със себе си до увеличаване на работата по нейното администриране. Това от своя страна заставя компаниите да наемат нови специалисти или да търсят поддръжка от външни фирми, което повлиява негативно върху финансовите вложения на компаниите, а често и върху ефективността на администриране. Най-същественият проблем при разширяването на мрежова инфраструктура се оказва въпросът за нарастването на нейната сложност. При поддръжка на сложна мрежова среда често се случва и най-опитни специалисти да се затрудняват, като при разрешаването на еднотипни задачи от различни администратори се стига до прилагането на различни решения. Проблемът е още по-комплициран, тъй като при многократното изпълнение на рутинни, често срещани мрежови задачи системните администратори са склонни да допускат различни типове грешки. Например, при модернизация или частична подмяна на съществуваща мрежова инфраструктура, при оптимизиране на съществуващи мрежови решения, при интегриране на нова софтуерна компонента критичните ситуации, водещи до усложнения по поддръжката на мрежовата среда, нарастват значително. В практиката унифицираното решение на тези проблеми е известно като „автоматизиране на мрежови операции” (Hämäläinen, et al., 2012), (MIHĂILĂ et al., 2017). Автоматизираното конфигуриране и управление на мрежи позволява на системния администратор да извършва по-ефективно своята работа, като същевременно е по-продуктивен. В отделните компании този въпрос се разрешава от различни софтуерни приложения.

В настоящата статия посочваме някои специфики при управлението на процеси, подходящи за автоматизация, категоризираме тези процеси по сложност, разглеждаме популярните системи и средства за приложение и разработване на автоматизирано мрежово управление на устройства, като в заключение представяме авторско софтуерно решение за автоматизирано управление на мрежова инфраструктура, придружено с анализ на неговите силни и слаби страни.

2. Автоматизация на процеси в мрежовото администриране.

2.1. Анализ на мрежови процеси, подходящи за автоматизация.

Много от процесите в мрежите подлежат на автоматизация – от т. нар. „read only" процеси до такива, които зависят от динамично променящи се условия. (Burns et al., 2001), (El-Darieby & Bieszczad, 1999). „Read only" процесите отразяват, верифицират и архивират състоянието на мрежата. Такива например са: запазването на конфигурациите на устройствата в база от данни, генериране на списък и опис на хардуерните устройства, проверка на ключови стойности, сравнение на действителното състояние с шаблони и т.н. Характерното за тези процеси е, че при тях не настъпват промени в състоянието на активното оборудване. Поради тази своя черта те са идеалният кандидат за автоматизация.

Процесите, които предполагат извършването на тестове, и интеграцията на нови обекти в мрежовите устройства също са отлични кандидати за автоматизация. При тези процеси могат да се автоматизират определени проверки и действия, в които се проследяват и отчитат техни ключови параметри. Например, такива тестове са периодичните проверки за състоянието на отделни параметри на мрежовата инфраструктура и устройствата, за които при незадоволителни резултати се генерират уведомления. Такъв е подходът и при интеграцията на нови обекти в мрежата.

Процесите с една идея по-сложни от по-горе споменатите, които са свързани с автоматично разпознаване и конфигуриране на устройства, при което се интегрират и инсталират нови обекти, също подлежат на автоматизация. При тези процеси се изграждат нови обекти, които променят както състоянието на устройствата от пасивни на активни, така и поведението им, т.е. тези процеси зависят от променящите се параметри в мрежовата среда и условията в нея. Поради това автоматизацията в тези процеси е с по-висока степен на сложност. За тях в литературата се е приело понятието „автоматизиране на развоя", т.е. тестване и въвеждане в експлоатация на нови технологии, нови процеси, нови продукти и др. Успешното им интегриране се реализира на етапи, тъй като често то е свързано с миграция към нова мрежова инфраструктура. Освен това, софтуерът при тези процеси е по-сложен, защото е необходимо да проследява допълнителни показатели за оценка работата на мрежата и по-специално за услугите, които се предоставят [1].

Най-сложните за автоматизиране са процесите, които съдържат операции в отговор на събития по мрежата. Това може да включва аварии, отговор на атаки, отговор на критични стойности на показатели и др. От софтуерна гледна точка тези процеси са

трудни за автоматизация заради възникването на непредвидени ситуации, които изискват човешка намеса.

Въз основа на така направения анализ следва да представим *категоризация на мрежови процеси, подлежащи на автоматизация.*

2.2. Категоризиране на мрежови процеси, подлежащи на автоматизация.

В настоящата подточка категоризираме мрежовите процесите, подлежащи на автоматизация. Категоризацията започва от по-лесните за автоматизиране мрежови процеси и продължава с по-трудните.

Категоризация на мрежови процеси, подлежащи на автоматизация, според тяхната сложност:

1) Процеси, подлежащи на автоматизиране, при които единствено се подават команди към устройството, като например при архивиране на конфигурации, еднакви промени, инвентарна информация и т.н.;

2) Процеси, подлежащи на автоматизация, след предварително моделиране на състояние: изискват експертна намеса (проверка на условие, изпълнение на действия в зависимост от резултата, действия, изпълнени в определена последователност, синхронизиране на дейности/резултати);

3) Процеси, зависещи от отговор на динамично събитие (изпълнение на действия в отговор на събития). Изисква се предварителна подготовка като дефиниране на събития, дефиниране на сценарий от действия съобразно динамично настъпване на събитие, тестване, верификация и обработка на грешки;

4) Процеси, подлежащи на автоматизация, но изискващи човешка намеса. Извършва се набор от действия, след като бъдат одобрени/допълнени от оператор.

3. Системи и средства за управление на мрежова инфраструктура.

3.1. Водещи платформи в областта.

В областта на компютърните мрежи има разработени мощни системи за автоматизирано управление, които се прилагат, развиват и поддържат от големи общности. Тези системи позволяват да бъдат автоматизирани широк набор от дейности и процеси, извършвани от системните администратори. Както в големите корпоративни организации, така и при по-малките такива, в които се управляват до няколко сървъра, се използва поне една система за автоматизирана мрежова администрация. Сред най-популярните в областта и масово използвани от общността на системните администратори са системите Ansible, Chef и Puppet [2, 3, 4, 5, 6]. Тези софтуерни системи са се разпространили и наложили като водещи поради много и различни причини. Сред основните причини за тяхното масово внедряване са многото разработени разширения над основната им функционалност, предоставени чрез специализирани модули, добавки (plugins) и приложен програмен интерфейс (API). Съществена черта на стотиците разширения на системите Ansible, Chef и Puppet е, че предлагат хиляди безплатни решения. При това допълнителните възможности, освен за употреба във функциониращa мрежова среда, се използват и за извършване на симулативни и

експериментални системни тестване, което е особено важно за областта. Не на последно място, популярността на системите Ansible, Chef и Puppet и техните свободно достъпни разширения предоставят възможност за разработка и интеграция на собствени и специфични добавки (plugins) [7, 8, 9].

Работата на Chef и Puppet е базирана на технологията DSL (Domain Specific Languages) и по своята същност представлява „договори" със сървър или мрежово устройство. Чрез тях се предоставя възможност да се указват файлове или пакети, дефинират се състояния на конфигурацията и набора от услуги, които могат да бъдат активни или не, докато управлението и изпълнението на тези договори се извършва от инструментите на средата.

Работата на Ansible е аналогична на Chef и Puppet, като съществено различие е употребата на технологията SSH. Благодарение на нея не се налага инсталация на агент на устройството. Заради това улеснение Ansible е предпочитан и технически по-добър избор за автоматизация на мрежови устройства. Друго съществено предимство на Ansible пред Chef и Puppet е възможността за групиране на устройствата според тяхната функционалност или роля, както и задаването на съответствия между конфигурацията и ролите. Чрез такова групиране се постига намаляване на броя на уникалните за управление устройства, вследствие на което драстично се намалява работата на системния администратор.

3.2. Средства за разработка на собствени решения.

Универсален инструмент, от софтуерна гледна точка, за разработване на автоматизирани решения в мрежовото администриране са скриптовите езици за програмиране. Например, (Stoitsov & Rangelov, 2014) коментират PHP имплементация на API интерфейс за RouterOS при интегриране с външни системи. За скриптовия език при мрежовата автоматизация може да се мисли като за „канап", който свързва всички технологии. Повечето популярни и силно развити скриптови езици за програмиране като Python, PHP, Ruby, Perl, Expect и т.н. се възприемат като подходящи средства за разработване на софтуерни мрежови решения главно заради силната си програмна съвместимост с други софтуерни технологии и интерпретативния характер на изходния код, за който не се налага компилация преди изпълнение. Към настоящият момент, поради масовата си използваемост от софтуерната общност, Python е с няколко крачки по-напред в развитието пред останалите скриптови езици за програмиране. Той е разработен в началото на 90-те години като надежден скриптов език с общо предназначение в програмирането (Donaldson, 2013), (Guttag, 2016). Според доклад на TIOBE Index за юни 2019 г., Python е третият най-широко разпространен и масово използван език за програмиране, като прогнозата на анализаторите е, че „до 3-4 години той ще се превърне в най-употребявания език"[10]. Python има хиляди готови модули, които предлагат решаването на множество специфични проблеми с имплементация на минимален обем изходен код. В частност за Python могат да се посочат и някои функционални предимства, какъвто например е случаят с библиотеката Virtualenv, чрез която: се постига управление на средата на Python за всеки проект; не се налага

администраторски достъп за инсталиране или ъпгрейд на модули; постига се изолация на промените между отделните проекти и т.н. Поради тези и много други причини Python се счита, че е най-подходящият програмен инструмент за разработване на автоматизирани софтуерни мрежови решения.

Други широко разпространени и използвани средства за намиране на решения за автоматизирано управление на мрежови устройства са т. нар. шаблонни езици (templating language). Чрез тях се постига дефиниране на съдържание като, например, модел за конфигуриране на външен и вътрешен интерфейс, контрол на достъпа, политика за качество на услугата (QoS – Quality of Service) и т.н. В инструментариума на шаблоните езици са включени стандартни езикови средства за работа като вложени изрази, употреба на променливи, условни оператори, логически оператори, цикли и др. По-известните шаблонни формати са Embedded Ruby(ERB) и Jinja2. Първият позволява да се използват възможностите на езика Ruby, докато вторият е по-широко използван заради тясната си връзка с Python и Ansible.

Към популярните и ефективни средства за автоматизирано управление на мрежова инфраструктура спадат платформите на големите производители на мрежово оборудване, от които най-голям пазарен дял имат HP, Cisco и Juniper. Тези компании, притежаващи мултимилионен бизнес, са разработили собствени софтуерни платформи, каквито са HP BTO (Hewlett-Packard Business Technology Optimisation), Cisco Prime Infrastructure и Juniper Network management and operations [11, 12, 13]. Макар и тези платформи да са много добри средства за автоматизирано управление на мрежова инфраструктура, тяхното приложение в хетерогенна среда, съставена от устройства на различни производители, е с ограничени възможности. Поради тази причина в големи по мащаб мрежови инфраструктури споменатите платформи са със слабо приложение.

4. Разработване на авторско софтуерно решение за управление и администриране на мрежови устройства.

Работата на старшия системен администратор в компанията „Информационно обслужване” АД, която поддържа десетки хиляди активни мрежови устройства, е свързана с възникването на множество непредвидени мрежови конфликти. Част от работа на администратора е да разрешава ситуации, при които се изисква конфигуриране или промяна в състоянието на няколко стотин устройства във възможно най-кратък срок. Извършвайки задълбочен анализ над мрежовата инфраструктурата на „Информационно обслужване“ и съществуващите софтуерни системи за автоматизирано управление на мрежи в едно със средствата за разработване на такива системи, през 2018 г. взехме решение да разработим собствено софтуерно приложение, по-късно наречено RouterCfg.

Логично е да се постави въпросът защо при наличието на толкова и различни платформи и системи за автоматизирано управление на мрежи се стигна до избор за разработването на собствено приложение. Анализът над функционирането на мрежовата инфраструктурата на „Информационно обслужване“ и съществуващите софтуерни системи за автоматизирано управление на мрежи освен ефективността на

управление на инфраструктурата включваше и много други въпроси, по-съществените от които са свързани със следните казуси:

Какъв е инвестиционният разход за закупуване или разработване на софтуер и респективно за обучаване за работа с него, както и дългосрочното ангажиране с външна компания за поддръжката при покупка на софтуер?;

Водещите софтуерни продукти на пазара на големите хардуерни производители на мрежови устройства като HP, Cisco и Juniper не могат да обслужват хетерогенната среда от устройства, каквато е при „Информационно обслужване“;

Водещите продукти на пазара за инвентаризация и мониторинг на мрежи като LogicMonitor, Paessler PRTG Network Monitor, SpiceWorks, Nagios и др. [14], които предлагат корпоративно и хардуерно независими решения имат дълъг обучителен период, а същевременно предоставят много ненужна функционалност за нашата компанията, която се калкулира в общата цена. Същевременно тези приложения предоставят единствено резултати от наблюдение над състоянието на мрежовата инфраструктура, без да е възможно да се управлява или променя наблюдаваното състояние;

Разширяване на функционалността на съществуваща софтуерна система, която да се синхронизирана с нуждите на мрежите, поддържани от „Информационно обслужване“, на практика е по-сложна задача за разрешаване от разработване на отделно софтуерно приложение (Първото предполага специална поръчка, ако собственикът на системата предлага такава възможност, или при ползване на продукт с отворен код (open source) да се наемат софтуерни разработчици, които в тясно сътрудничество с мрежовите специалисти от компанията да разработят необходимото функционално разширение на системата);

Избирането на съществуваща софтуерна платформа неминуемо ще трябва да се адаптира към специфичните условия на инфраструктурата на „Информационно обслужване“ – процес, който е сложен и трудно предвидим като време за осъществяване;

Финансовият и човешкият ресурс, необходими за разработването на собствено софтуерно решение, според направени от нас прогнози, са в пъти по-малки от плащането на лицензионни услуги за употреба и разширение на съществуваща софтуерна платформа;

Разработката на собствен продукт е съобразен с конкретните нужди на компанията, нейната инфраструктура, клиенти и пр.

Важността на споменатите казуси, както и някои по-субективни въпроси, натежаха в полза на разработването на собствено софтуерно приложение, подходящо за целите на компанията, което следва да представим.

5. Кратко представяне на възможностите и функционалността на софтуерното приложение RouterCfg.

5.1. Структура на RouterCfg.

Със софтуерното приложение RouterCfg могат да се управляват мрежови процеси от първите три категории на направената по-горе „*Категоризация на мрежови процеси подлежащи на автоматизация според тяхната сложност*“. Приложението успешно е интегрирано и се използва от системните администратори на „Информационно обслужване” АД. Чрез RouterCfg се автоматизират рутинни дейности като инвентаризация, проверка на съответствие, еднотипни изменения на конфигурация, динамични команди/промени, свързани с проверка на условия и автоматизиран отговор на събития. При работата с RouterCfg се разрешават типични задачи като четене и запис на информационен поток с цел вземане на динамични решения съобразно състоянието на устройството и/или средата, конекцията към мрежовото устройство в отделен клас, паралелно изпълнение на команди в множество независими устройства, визуализиран изход от мрежови устройства към потребителя, извършване на тестови настройки и корекции, валидиране на входна информация, обработка на грешки и др.

RouterCfg е структуриран в три каса с цел модулност и слаба свързаност на компонентите. Първият клас (RtrCfg.py) се грижи за обработката на информационния поток от потребителя. Вторият клас (Router.py) осигурява връзката между обработената информация и мрежовата инфраструктура включително и мрежовите устройства, а третият (Interpreter.py) предоставя програмната логиката на приложението. Чрез обособяването на интерпретатора на приложението в самостоятелен клас е възможно да се създаде напълно нов интерпретатор за специфичен тип устройство, който да има различни контролни структури и/или функционалност. Независимостта на конекциите към мрежата, благодарение на софтуерното решение за отделеност на класовете, се постига лесно с надграждане на допълнителни протоколи, без да се нарушава работата на готовите скриптове. Възможностите за надграждане са многобройни, при това в условие на хетерогенна среда. Това е специално заложено изискване към софтуера, тъй като такива са случаите в голяма част от съществуващата мрежова инфраструктура. Мрежовата инфраструктура, заради големите си начални инвестиции и сравнително дълъг експлоатационен живот, често е сбор от устройства на различни производители с различна експлоатационна възраст и различни изисквания за конфигуриране и работа. Погледнато от по-общ план, на RouterCfg може да се гледа като на модул, който успешно да се развие до системен софтуер за администриране и поддръжка на мрежи.

5.2. Работа с RouterCfg.

Системният администратор, който си служи с RouterCfg, може да попадне на множество от различни сценарии. Въпреки това, обичайно се изпълняват някои рутинни действия. Софтуерът се стартира от команден ред, като изисква да се въведат няколко задължителни параметъра:

Файл, който съдържа IP адресите на устройствата;

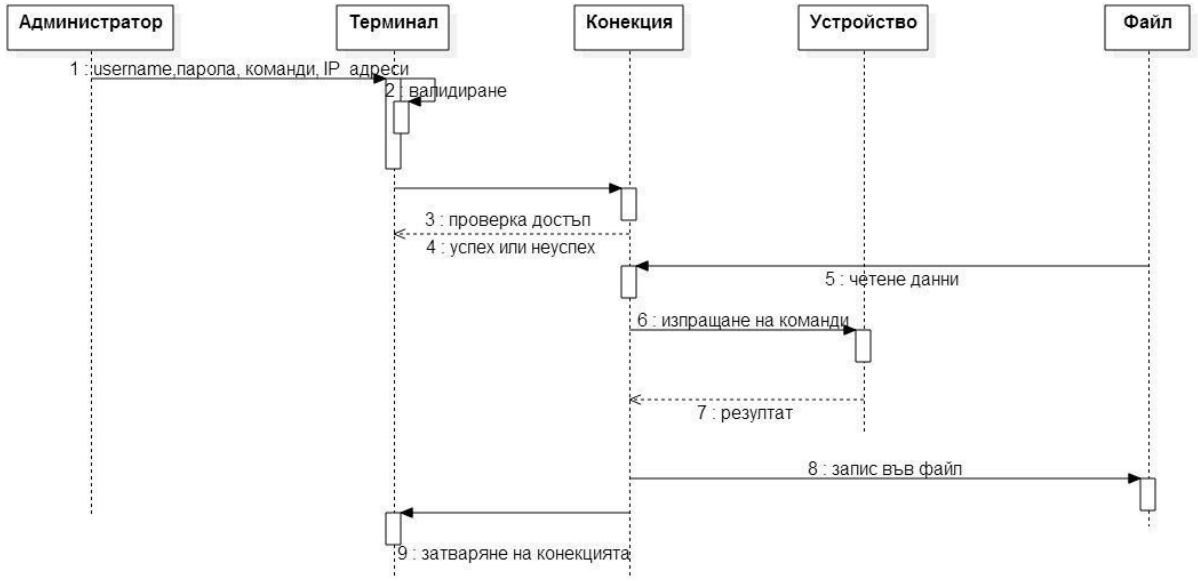
Файл, който съдържа конфигурационните команди;

Потребител и парола за достъп до устройствата (или цифров сертификат);

Опционални ключове за извеждане на информация.

След проверка за коректността на въведените данни в отделни нишки се изграждат конекции до всяко устройство и се изпълняват командите. Потребителят бива уведомяван след приключване изпълнението за всяко устройство. Добавена е опция, чрез която може да се извежда съобщение за всяка изпълнена команда и отговор на устройство.

Следва да разглеждаме типичен сценарий *Конфигуриране на допълнителен loopback интерфейс,* който представяме във вид на UML диаграма и описателно.



UML диаграма на последователност на сценария: *Конфигуриране на допълнителен loopback интерфейс.*

Сценарий: ***Конфигуриране на допълнителен loopback интерфейс при работа с RouterCfg:***

1) Потребителят стартира скрипта и въвежда потребителско име, парола, име на файл с команди и списък с IP адреси на устройства (файл).

2) Валидиране на входни данни.

3) Проверка за достъп до устройство и изграждане на контролна конекция.

4) Уведомяване на потребителя за статуса на конекцията.

5) Четене на данни от текстов файл (опция).

6) Четене и изпращане на команди към устройство.

7) Получаване на резултат от изпълнението на командата.

8) Визуализиране на резултата (опция).

9) Запис в текстов файл (опция).

10) Затваряне на конекцията.

Описателно представяне на сценарий *Конфигуриране на допълнителен loopback интерфейс:*

Една често срещана задача при администрирането на мрежи е настъпване на промени в конфигурацията на мрежовите устройства. Промените могат да бъдат от най-различно естество. При някои от тях конфигурацията е обща за всички устройства. Такъв процес е идеалният кандидат за автоматизация, тъй като едни и същи команди се изпълняват на различните устройства. По-честата ситуация обаче е разрешаване на задача, която изисква отделни параметри на различните устройства да бъдат уникални. Например, IP адресите да са уникални, лицата за контакт могат да са различни, access-list-те да се съобразяват с локалните мрежи и т.н. Тези случаи също могат да бъдат автоматизирани.

В разглеждания сценарий се конфигурира допълнителен loopback адрес на маршрутизаторите в мрежата. IP адресите се четат от текстов файл (*lbs.txt*), в който редовете представляват двойка IP адреси, разделени със специален знак. В нашия случай за разделител се използва знака за равенство „=". Първият адрес служи като идентификатор на маршрутизатор, а вторият адрес е кореспондиращ IP адрес, който трябва да бъде конфигуриран. При стартиране на RouterCfg се прочита файла с IP адресите на устройствата за конфигуриране, като първоначално се изгражда контролна конекция към тях. Администраторът получава съобщение с успешните и неуспешните свързвания към устройствата. След изпълнението на командите конекцията се затваря и администраторът отново се уведомява. Процесът на конфигуриране започва след въвеждане на потребителско име и парола за достъп от администратора, като за работата му са необходими входни данни от текстов файл, подобни на:

Команди за конфигуриране на loopback интерфейс:

conf t
int Loo10
<cmd "ip address " + <imp lbs.txt > + " 255.255.255.0">
end

Първата команда стартира глобален конфигурационен режим, втората създава интерфейс Loopback10, а третата съдържа няколко стъпки. Чрез конструкцията *<imp lbs.txt>* се прочита IP адрес от файла *lbs.txt*. IP адресът се конкатенира отдясно с низа „ip address", а отляво с низа „255.255.255.0", представляващ мрежовата маска. В резултат от изпълнението се получава нова конфигурационна команда „ip x.x.x.x 255.255.255.0", която задава IP адрес на интерфейса Loopback10. Запазената дума <cmd > указва, че това е команда за изпълнение, която бива изпратена към маршрутизатора. Съдържанието на файла *lbs.txt* е съставено от последователности:

IP на марщрутизатор=IP за Loopback10

С последната команда се излиза от глобален конфигурационен режим. Ако скриптът се стартира с опцията „-d", на терминала се отпечатва командата и отговорът на маршрутизатора. В конкретно разглеждания случай нито една от командите не връща отговор. Ако изпълнението е приключило без грешка, се отпечатва съобщение със зелен цвят. При наличие на грешка се отпечатва върнатото съобщение в червен цвят.

6. Ползи и перспективи.

Основната характеристика на ползите за системния администратор при употреба на RouterCfg са свързани с автоматизирането на процесите по управление на мрежовите устройства. Това автоматизирано управление носи многостранни преки ползи по отношение на надеждност на операциите, минимизиране на сривовете в мрежовите устройства при извършване на промени, бързодействие и ефективност на работа на администратора, общата стабилност на мрежата, нейната мащабируемост и др. Автоматизацията на управление на мрежовата инфраструктура води и до косвени ползи, изразени в намаляване на необходимостта от допълнителен персонал от системни администратори, намаляване на имиджови загуби за компанията поради сривове на мрежата, породени от човешка грешка или неточност, пестене на време от рутинни дейности, възможност за повишаване квалификацията на персонала, възможност за тестване на нови и по-перспективни технологии, намаляване времето за реакция, гарантиране качеството на услугата за потребителя (SLA Service Level Agreement) и др. Понастоящем RouterCfg е с разработена версия 1.1, която е публикувана в GitHub със свободен достъп за ползване, с отворен изходен код [15]. Приложението се използва активно от системни администратори на „Информационно обслужване“ за администрирането и управлението на хиляди мрежови устройства, но поради различни условия и обстоятелства от финансов, правен и социален характер не е публикувано под определен лиценз. При първоначалното проектиране бяха планирани модули и класове, които да осигурят: графичен потребителски интерфейс, достъпен чрез уеб среда; модул за филтриране на потоците от информацията към и от отделните устройства; модул, който надгражда нивото на абстракция над хардуера така, че да е допустимо използване на конфигурационни команди за различни типове оборудване; модул, чрез които се постига синхронизация на работата с отделните устройства и др. Към настоящия момент тези модули не са част от експлоатацията на приложението, като са развити до ниво на тестово интегриране. От тази гледна точка може да говорим за две перспективи на развитие. Софтуерна перспектива, т.е. разширение на приложението с допълнителна функционалност. Потребителска перспектива, т.е. популяризирането на приложението и предоставянето му за използване от други компании, проявяващи интерес.

7. Заключение.

В настоящата статия дискутирахме въпроси за автоматизираното управление на мрежова инфраструктура. Разгледахме трудностите, които системните администратори срещат при разширяването на мрежовата среда. Анализирахме мрежовите процеси, за които е приложимо автоматизирано управление. Направихме преглед на водещите в областта платформи и системи, чрез които се разрешават въпроси по управление на мрежите и на средствата за разработване на собствени софтуерни приложения, като анализирахме техни силни и слаби страни. Представихме авторско софтуерно приложение RouterCfg за автоматизирано управление и администриране на мрежова

инфраструктура, чрез което се обслужват хиляди мрежови устройства, собственост на „Информационно обслужване" АД. Разкрихме ползите от разработването на RouterCfg за системните администратори. Посочихме софтуерната и потребителската перспектива от разработването и интегрирането на приложението в работна среда.

В заключение, въз основа на изложеното в статията, опитът ни в областта на мрежовото администриране, разработването на RouterCfg и интеграцията на приложението в работеща мрежова инфраструктура стигаме до следните изводи:

1) Чрез избора, решенията, реализацията и интеграцията на собствено приложение, което да обслужва мрежата на „Информационно обслужване" АД, компанията, в тази си дейност, запазва своята финансова, корпоративна и техническа независимост. Същевременно ефективното обслужване на хиляди мрежови устройства, измерено като финансови вложения и човешкия ресурс, е в пъти по-ефикасно от закупуването на готов софтуерен продукт.

2) Качеството и ефективността при управлението и администрацията на мрежови устройства чрез RouterCfg не само не отстъпва на големите софтуерни приложения, предлагани от Ansible, Chef и Puppet, HP, Cisco и Juniper, но конкретно за инфраструктурата на „Информационно обслужване", ги превъзхожда поради това, че е разработено съобразно специфичните нужди на компанията.

NOTES/БЕЛЕЖКИ

1. Official site of Juniper Network, https://www.juniper.net/us/en/solutions/automation/the-automation-journey/, [last visited 05.08.2019].
2. Bravo, M. Top 5 configuration management tools, Specialized site - opensource.com, https://opensource.com/article/18/12/configuration-management-tools/, [last visited 05.08.2019].
3. Julian, J., Lowe, S. Choosing Network Automation Tools, Specialized site - Network Computing, Informa PLC, https://www.networkcomputing.com/data-centers/choosing-network-automation-tools/, [last visited 11.08.2019].
4. Official site of Ansible , https://www.ansible.com/, [last visited 07.08.2019].
5. Official site of CHEF, https://www.chef.io/, [last visited 12.08.2019].
6. Official site of Puppet, https://puppet.com/, [last visited 16.08.2019].
7. Specialized GitHub site, https://github.com/jedelman8/nxos-ansible, [last visited 20.08.2019].
8. Specialized GitHub site, https://github.com/Juniper, [last visited 20.08.2019].
9. Specialized GitHub site, https://github.com/cisco/cisco-network-puppet-module, [last visited 20.08.2019].
10. TIOBE Index for August 2019 , https://www.tiobe.com/tiobe-index/, [last visited 05.08.2019]
11. HP BTO Software Applications Portfolio, Hewlett-Packard Development Company, 2010,

http://www.hp.com/hpinfo/newsroom/press_kits/2010/HPSoftwareUniverseBarcelona2010/HP_Applications_Portfolio_brochure.pdf, [last visited 08.09.2019].
12. Official site of Cisco Prime Infrastructure, https://www.cisco.com/c/en/us/products/cloud-systems-management/prime-infrastructure/index.html, [last visited 08.09.2019].
13. Official site of Juniper Networks, https://www.juniper.net/uk/en/, [last visited 08.09.2019].
14. Ferrill, P. The Best Network Monitoring Software for 2019, PCMag Digital Group, December 2018, https://www.pcmag.com/roundup/339630/the-best-network-monitoring-software, [last visited 11.09.2019].
15. Specialized GitHub site, hhttps://github.com/a-ignat/RouterCfg, [last visited 20.09.2019].

REFERENCES/ЛИТЕРАТУРА

Burns, J., Cheng, A., Gurung, P., Rajagopalan, S., Rao, P., Rosenbluth, D., Surendran, A. & Martin, D. (2001). Automatic Management of Network Security Policy. *Proceedings DARPA Information Survivability Conference and Exposition II*. DISCEX'01, vol. 2, June, pp. 12-26, ISBN: 0-7695-1212-7, DOI: 10.1109/DISCEX.2001.932156.

Donaldson, T. (2013). *Python: Visual QuickStart Guide*. 3rd Edition, Kindle Edition, Peachpit Press, ISBN-13: 978-0321929556.

El-Darieby, M. & Bieszczad, A. (1999). Intelligent Mobile Agents: Towards Network Fault Management Automation. *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*. pp. 611-622, ISBN: 0-7803-5748-5, DOI: 10.1109/INM.1999.770711.

Guttag, J. (2016). *Introduction to Computation and Programming Using Python: With Application to Understanding Data*. Second edition,The MIT Press, ISBN 978-0-262-52962-4.

Hämäläinen, S., Sanneck, H. & Sartori, C. (2012). *LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency*, John Wiley & Sons, January, ISBN: 978-1-119-97067-5.

Stoitsov, G. & Rangelov, V. (2014). One implementation of API interface for RouterOS, *TEM Journal*,Vol.3, No.2, May, 148-152, ISSN: 2217-8309.

MIHĂILĂ, P., BĂLAN, T., CURPEN, R. & SANDU, F. (2017). Network Automation and Abstraction using Python Programming Methods. *MACRo 2017 - Proceedings of the 6th International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics*. Volume 2: Issue 1, Oct, p.95–103, DOI: https://doi.org/10.1515/macro-2017-0011

Hristo Hristov,
Assistant Professor, PhD,
University of Plovdiv „Paisii Hilendarski”,
Faculty of Mathematics and Informatics, Department of Computer Science
236, „Bulgaria” blvd, 4003 Plovdiv, Bulgaria
E-mail: hth@uni-plovdiv.bg

Angel Ignatov
Senior Network Administrator, „Information Services”
Plc, Plovdiv, blvd. „Sankt Peterburg” 59,
E-mail: a.ignatov@is-bg.net