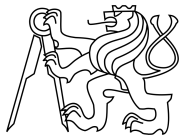
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Nástroj pro hromadnou správu síťových zařízení
Student:	Lukáš Merta
Vedoucí:	Ing. Jiří Mlejnek
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Seznamte se s existujícími nástroji pro hromadnou správu síťových zařízení využitelných poskytovatelem připojení.
Analyzujte požadavky na aplikaci, která umožní přehledně prohlížet a filtrovat jednotlivá nastavení z více zařízení současně a hromadně tato nastavení měnit. Zaměřte se především na oblast správy uživatelů a konfiguraci síťového nastavení.
Na základě analýzy proveďte návrh a implementaci této aplikace. Rozsah realizovaných požadavků konzultujte s vedoucím práce.
Aplikaci nasaďte a otestujte její připravenost pro produkční nasazení.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 27. září 2016

Poděkování

Rád bych poděkoval vedoucímu své práce Ing. Jiřímu Mlejnkovi za jeho čas a rady a své rodině za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“) a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Brně dne 8.2.2017

Odkaz na tuto práci

Merta, Lukáš. *Nástroj pro hromadnou správu síťových zařízení.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá možností konfigurace více síťových prvků s využitím responzivního webového uživatelského rozhraní, které je použitelné na všech zařízeních. Zaměřuje se na správu uživatelských účtů a konfiguraci síťového rozhraní na prvcích se systémem RouterOS, k nimž se aplikace připojuje pomocí API nebo protokolu SSH.

Klíčová slova

webová aplikace, NMS, hromadná správa zařízení, konfigurace síťových prvků, bezpečnost, SNMP, SSH, API

Abstract

This thesis deals with configuration of multiple network devices and includes responsive web interface to target many devices. Its main focus is on user and interface management of RouterOS powered systems. To each device connects via API or SSH protocol.

Keywords

web application, NMS, bulk device management, network device configuration, security, SNMP, SSH, API

Obsah

Úvod

Práce se zabývá možností konfigurace jednotlivých zařízení se zaměřením na správu více zařízení najednou. Zaměřuje se na správu uživatelů, kteří mohou k zařízení přistupovat a nastavení síťových rozhraní.

Pro správu jediného síťového zařízení nejsou potřeba žádné speciální nástroje, každé zařízení má vlastní podporu konfigurace. Většina běžných routerů (směrovačů), tiskáren, telefonů a podobných zařízení s možností připojení do sítě určených pro použití v domácnostech a menších firmách, se konfiguruje přes webové rozhraní, uživatel si tak vystačí s webovým prohlížečem. Pouze některá z těchto zařízení mají možnost připojení přes příkazový řádek nebo jiné rozhraní pro propojení s jinými systémy (tzv. API). U profesionálních zařízení naopak webové rozhraní zcela chybí nebo je chápáno jako doplněk a primárním způsobem připojení k nim je příkazový řádek. Práce se dále nebude zabývat možností konfigurace přes webové rozhraní, protože není primárně určeno pro napojení jiných systémů.

Pokud se ovšem množina spravovaných zařízení rozrůstá, je stále obtížnější udržet si přehled o nastavení jednotlivých zařízení a každá změna týkající se více z nich je také časově náročná na provedení. Znamená totiž manuální připojení ke všem zařízením a provedení změny konfigurace. Velmi podobná situace je se sledováním nejrůznějších statistik mnoha zařízení a hledání vzájemných souvislostí.

Práce se zaměřuje na nástroje pro hromadnou správu (označovány zkratkou NMS z anglického spojení „network management system“), které usnadňují a zpříjemňují práci všem, kteří mají na starosti více než jedno zařízení. Umožňují jednoduše prohlížet nastavení a statistiky a spravovat více zařízení najednou bez nutnosti se přihlašovat ke každému zvlášť. Kromě toho také mnohdy zvládají monitorování hodnot, zobrazení historie (graf) a zasílání upozornění na definované události (například překročení limitní hodnoty nebo výpadek). Práce se soustředí na bezplatné nástroje (nejlépe s otevřeným kódem pro možnost vlastních úprav).

1 Aktuální stav

V současnosti musí správci více zařízení (nepoužívající žádný nástroj pro hromadnou správu) přistupovat k jednotlivým zařízením zvlášť. Pokud chtějí například přidat uživatelský účet s přístupovými právy k zařízení, musí se postupně připojit ke všem zařízením a u všech tento nový účet vytvořit. Tento postup je nejen zdlouhavý a nudný, ale také náchylný k chybám, může se lehce stát, že do některého zařízení se správce nepřipojí vůbec (buďto jen zapomene, nebo v daný okamžik není zařízení dostupné a musí se konfigurace nahrát později, až přístupné bude) nebo nastaví hodnotu odlišně.

V referenční síti autora práce je aktuálně spravována necelá tisícovka síťových prvků využívajících převážně systém RouterOS (1) od firmy MIKROTIKLS SIA (dále jen „Mikrotik“ – velmi často se chybně používá i pro označení systému, pod značkou Mikrotik firma prodává své produkty – síťové prvky; v textu bude výraz Mikrotik používán pro označení výrobce i značky). Současný systém jednotlivá zařízení monitoruje a do přístupových bodů (nikoliv však do koncových zařízení uživatelů sloužících pro jejich přístup k síti) umožňuje nahrát připravený skript, ale už neřeší, zda proběhlo spuštění skriptu správně (pokud navíc skončí provádění skriptu chybou, nelze do zařízení nahrát novější skript, systém se vždy pokouší nejprve spustit původní chybný skript – v takovém případě je potřeba ze všech zařízení daný chybný skript ručně smazat). Protože chybí návratová hodnota skriptu, nelze tímto způsobem ani zjistit hodnoty, které systém nesleduje prostřednictvím SNMP (zkratka vysvětlena dále v textu v samotné sekci SNMP na straně 4) – např. uživatelské účty vytvořené v zařízení tímto způsobem ani sledovat nelze.

Z předchozího odstavce je zřejmý důvod, proč bylo vybráno právě dané téma práce. Cílem je najít nástroj, který pomůže s otázkami typu:

- Jaká je nejstarší verze systému a na kterém zařízení?
- Jaké je zastoupení jednotlivých verzí?
- Které balíčky jsou povoleny na kterých zařízeních?
- Kteří uživatelé mohou k jakému zařízení přistupovat?
- Má určitý uživatel právo přistupovat ke všem zařízením?
- Nezůstal na některém zařízení výchozí účet pro správu bez hesla?

Z těchto otázek dále vyplynou některé požadavky na systém.

2 Analýza problému

Hlavním cílem aplikace je zajištění přehledu o všech spravovaných zařízeních a možnost nejen jednoduše zobrazit požadovaný parametr napříč těmito zařízeními, ale také daný parametr změnit. Kromě tohoto ústředního cíle jsou od aplikace vyžadovány další funkčnosti.

2.1 Požadavky na aplikaci

Na aplikaci jsou kladeny požadavky s ohledem na interní použití správci sítě, kteří jsou nejen v kanceláři, ale také se často pohybují v terénu. Proto musí být dostupná odkudkoliv pro jakékoliv zařízení.

Souhrn obecných požadavků

- bezplatná licence k užívání, nejlépe i otevřený kód (open source) pro možnost případných úprav
- podpora zařízení se systémem RouterOS (konfigurace přes příkazovou řádku nebo API)
- webové rozhraní pro dostupnost na široké škále zařízení
- programové rozhraní (API) pro propojení s jinými aplikacemi
- dostupná z vnější sítě (internetu)
- zobrazování hodnot v reálném čase (pouze s minimální prodlevou nutnou na zpracování)
- přizpůsobené zobrazení i na menších obrazovkách mobilních telefonů
- vyhledávání a filtrování v hodnotách i názvech parametrů
- multiplatformní serverová část (tj. umožňující běh pod různými operačními systémy)
- zabezpečení uložených přihlašovacích údajů k zařízením

Nad rámec těchto povinných požadavků (náměty na případná rozšíření)

- zobrazení některých hodnot a historie i pro odpojené zařízení (bez připojení k síti, offline)
- nahrání nového nastavení pro odpojené zařízení automaticky po jeho opětovném připojení
- zobrazení grafů historických hodnot s možností vlastního měřítka a období
- unifikované rozhraní pro různá zařízení
- podpora dalších systémů jiných výrobců
- role jednotlivých uživatelů (budou k ní přistupovat pouze správci, kteří mají úplnou kontrolu nad celou sítí)

Souhrn funkčních požadavků

- evidence zařízení – přidání, zobrazení, editace, mazání
- správa uživatelů aplikace – přidání, zobrazení, editace, mazání
- vlastnosti a parametry zařízení (zobrazení, přidání, editace, mazání)
    - operační systém (nebo firmware) – název a verze – pouze zobrazení
    - aktuální doba běhu (od posledního startu) – pouze zobrazení
    - IP adresy

    - uživatelé
    - role uživatelů
    - informace o rozhraní
- vlastnosti a parametry rozhraní
    - typ rozhraní
        - fyzické nebo virtuální
        - kabelové (optické, metalické), bezdrátové (licencované nebo nelicencované pásmo) apod.
    - přiřazené IP adresy
    - běžící služby (DHCP server, klient)
    - stav spojení (připojeno nebo odpojeno)
- možnost (automatického) zjištění informací po zadání údajů pro připojení k zařízení (IP nebo MAC adresa, případně přihlašovací údaje)

2.2 Obecné možnosti konfigurace jednotlivých zařízení

Pro čtení údajů a konfiguraci jednotlivého zařízení lze využít několik způsobů, které si podrobněji projdeme. Standardem v tomto ohledu je protokol SNMP, který byl přesně pro tyto účely navržen. Dalším hojně využívaným přístupem je využití příkazové řádky (taktéž označován, i když je význam trochu jiný, jako konzola nebo terminál) – k tomu se využívají různé protokoly (nejběžnější použití je SSH nebo telnet), dále některá zařízení nabízejí vlastní rozhraní (tzv. API). Naopak u nejlevnějších zařízení pro domácí použití je nejběžnější webové rozhraní.

2.2.1 SNMP

Jak již význam zkratky napovídá (z anglického „Simple network management protocol“, neboli protokol pro správu sítě), jedná se o standard pro správu síťových prvků (2), proto také většina z nich tento protokol na různé úrovni možností podporuje. Jedná se o jeden z nejčastěji využívaných protokolů pro čtení stavových hodnot ze zařízení jako např. teplota, vytížení procesoru, využití paměti apod. Lze jej ale použít i pro konfiguraci.

Ze stránky s manuálem k SNMP pro RouterOS (3) lze stáhnout sadu objektů pro správu (v části „Management information base (MIB)“), v ní se ale nenachází informace o uživatelských účtech. Příkaz *print* v některých částech menu v konzoli (např. *interface*) obsahuje parametr *oid*, který zobrazí identifikátory objektů spravovaných pomocí SNMP. V části menu *user* (která slouží pro správu uživatelů) příkaz *print* parametr *oid* neobsahuje. Z toho vyplývá, že tento protokol v RouterOS nelze použít pro správu uživatelských účtů, a proto se jím práce dále zabývat nebude.

2.2.2 Webové rozhraní

Jedná se o uživatelsky nejpřívětivější možnost konfigurace, kdy pro připojení je potřeba pouze webový prohlížeč. V něm se pak zobrazí jednotlivé stránky s nastavením a statistikami. Toto ovládání je velmi intuitivní, uživatelé jsou na něj běžně zvyklí z internetových stránek.

Pro automatizaci není tato možnost příliš vhodná z několika důvodů. Předně většinou neexistuje kompletní popis celého rozhraní, k dispozici jsou pouze jednoduché manuály, které jsou psány spíš jako průvodci. Dále získání hodnot znamená čtení a zpracování jazyka HTML, který slouží k zobrazování dat, opačný postup (získání dat) je proto mnohem náročnější (navíc náchylný na změny v rozhraní – z toho vyplývá nutnost zkoumat změny při nefunkčnosti a patřičně upravit aplikaci). Další překážkou u některých zařízení může být fakt, že jsou stránky mnohdy generované až na straně webového prohlížeče, nestačí tak poslat požadavek na danou stránku a pak ji jednoduše zpracovat, ale je nezbytné mít zprovozněn virtuální webový prohlížeč a z něj po zpracování data získávat. Z uvedených důvodů se touto možností nebudu dále zabývat

2.2.3 Příkazová řádka – telnet, SSH

Většina profesionálních zařízení (ne-li všechna) disponuje možností zadávání příkazů přes místní nebo vzdálený terminál (též označován jako konzola). Uživateli se zobrazí textová příkazová řádka, kam může psát příkazy a zařízení na jejich základě vypisuje požadované informace nebo provádí nastavení.

Pro přístup k příkazové řádce se stále ještě používá (dnes již zastaralý a nezabezpečený) protokol telnet (4), který je ale nahrazen novějším a zabezpečeným protokolem SSH (5). Tyto protokoly nejsou jediné, ale jsou zdaleka nejrozšířenější. Zkratka SSH vznikla z anglického „secure shell“ a označuje zabezpečené rozhraní pro komunikaci. Z uživatelského hlediska je přístup přes telnet totožný s přístupem přes SSH. Oba protokoly jsou standardizovány a díky tomu implementovány v mnoha zařízeních.

Výhodou tohoto přístupu je dobrá dokumentace jednotlivých příkazů a jejich parametrů (6). Určitou nevýhodou je mnohdy textově orientovaný formátovaný výstup, který je potřeba dále zpracovávat pro strojové načtení a následné zobrazení. Některá zařízení ale disponují příkazy (nebo jejich parametry), které výstup formátují právě pro strojové čtení. Jako obecný příklad poslouží formát CSV (7) („comma separated values” – hodnoty oddělené čárkou; u nás je používanějším oddělovačem středník, protože čárka slouží k oddělení desetinných míst u čísel, k oddělení ale může být použit i jiný znak, např. tabulátor). Na prvním řádku (hlavička) mohou být volitelně vypsány názvy oddělené čárkou a na dalších řádcích jsou hodnoty vlastností s názvem z prvního řádku, oddělené taktéž čárkou. Pro takový formát existují knihovny (8) (9) pro načtení a zpracování – vývojář tak má přístup přímo ke zpracovaným údajům.

Konkrétně příkazová řádka systému RouterOS nabízí několik parametrů příkazu *print* (10) pro strojové zpracování (zkratky použité v ukázkách níže: *id* – identifikátor záznamu, *z* – záznam, *v* – vlastnost záznamu)

- *as-value* – všechny záznamy na jednom řádku jako pole parametrů a hodnot oddělených středníkem

  ```
  id=*1;z1v1=h1;z1v2=h2;id=*2;z2v1=h3;z2v2=h4
  ```

- *terse* – každý záznam na jednom číslovaném řádku (za číslem mohou být další symboly značící určité stavy nebo hodnoty), vlastnosti s hodnotou oddělené mezerou

  ```
  0 I z1v1=h1 z1v2=h2
  1   z2v1=h3 z2v2=h4
  ```

- *value-list* – každý záznam jako jeden sloupec, na řádku je vždy jedna vlastnost (název s dvojtečkou na začátku řádku, hodnoty oddělené mezerami – zarovnáno do sloupců)

  ```
  v1: z1h1 z2h2
  v2: z1h3 z2h4
  ```

2.2.4 API

Zkratka vychází z anglického „application programmable interface“ a označuje rozhraní vytvořené speciálně pro komunikaci mezi zařízeními nebo aplikacemi (není primárně určeno pro komunikaci s uživatelem). Obecně API umožňuje propojení různých systémů, dobře se z něj načítají data pro další zpracování a do něj zapisují. Dokumentace k API bývá podrobná (11).

2.3 Současné nástroje

Vzhledem k požadavku na bezplatnou licenci práce vynechává placené (např. ManageEngine Network Configuration Manager (12)) nebo v bezplatné verzi omezené (např. počtem spravovaných zařízení – SolarWinds Network Configuration Manager (13)) nástroje. Nezabývá se ani takovými, které nedisponují webovým rozhraním a jsou vázány na použití pouze na běžném počítači (nebo jen na některém z podporovaných operačních systémů).

Přímo tvůrce RouterOS, firma Mikrotik, na svých stránkách nabízí ke stažení několik nástrojů pro konfiguraci. Jedním z nich je The Dude (14), který slouží pro správu jakékoliv sítě (s množstvím pokročilých funkcí jakými jsou automatické prohledávání sítě, sledování statistik a vytváření grafů, zasílání upozornění při výpadku). Samozřejmostí je podpora pro zařízení se systémem RouterOS, pro jejichž správu je nástroj určen, proto se jím práce bude zabývat nejvíce (prozkoumá ale i další volně šiřitelné nástroje). Ostatní konfigurační nástroje od firmy Mikrotik slouží pro nastavení jednoho zařízení.

2.3.1 The Dude

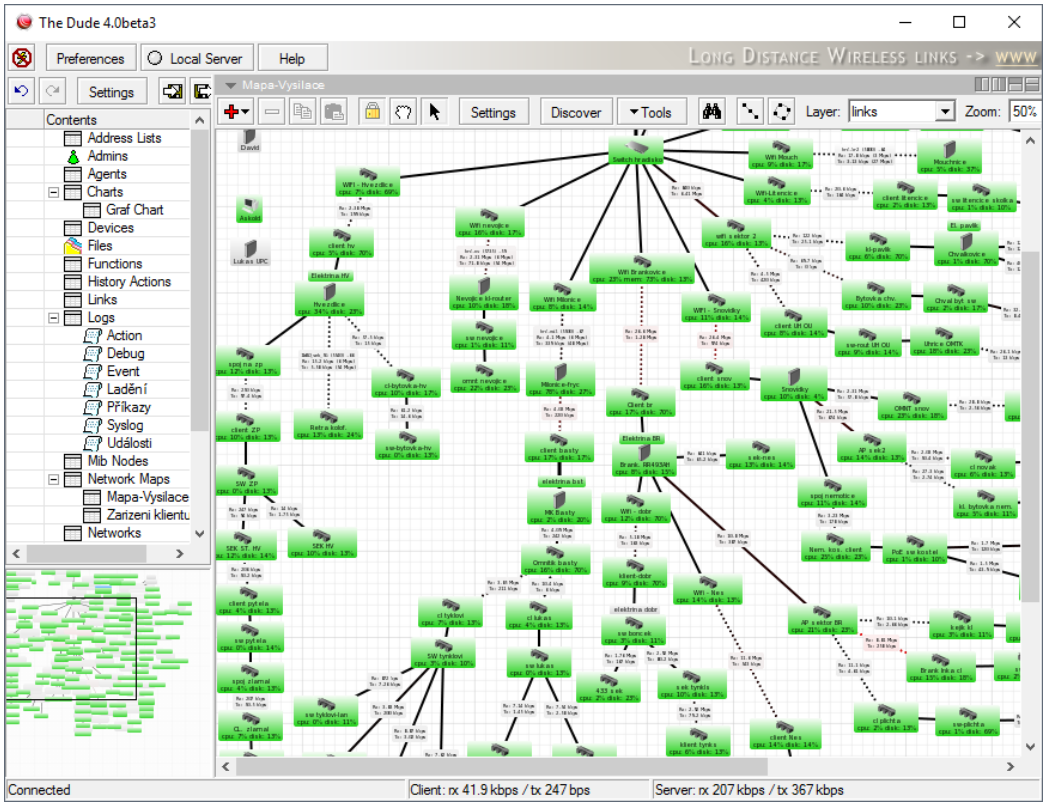
Pro správu sítě je používán tento nástroj v referenční síti již dlouhou dobu, obsahuje aplikaci pro Windows (s možností běhu pod Linuxem nebo MacOS prostřednictvím Wine, resp. Darwine) a k tomu (s mnohem omezenější funkčností) webové rozhraní. V používané starší verzi (4.0beta3) je ještě možné

v klientské části vytvořit tzv. lokální server, který může běžet i jako služba na pozadí, což je aktuálně v reálném provozu využíváno. Nástroj je dostupný volně k užívání, má ale uzavřený kód.

Správa zařízení

Nástroj poskytuje přehled nad všemi sledovanými zařízeními zobrazenými na mapách (Obrázek 1) nebo vypsanými v seznamu s indikací stavu (Obrázek 2) a umožňuje snadné spuštění konfiguračních (Winbox, telnet, SSH, …) a diagnostických (ping, traceroute, snmpwalk, …) nástrojů.

Pokud se jedná o zařízení se systémem RouterOS, lze v seznamu zjistit aktuální verzi systému, distribuovat novější a zařízení tak hromadně aktualizovat. Nejdříve je nutné danou verzi do systému uložit, ten pak při aktualizaci hlídá, zda se jedná o správnou platformu a není-li již v zařízení novější verze.



OBRÁZEK 1

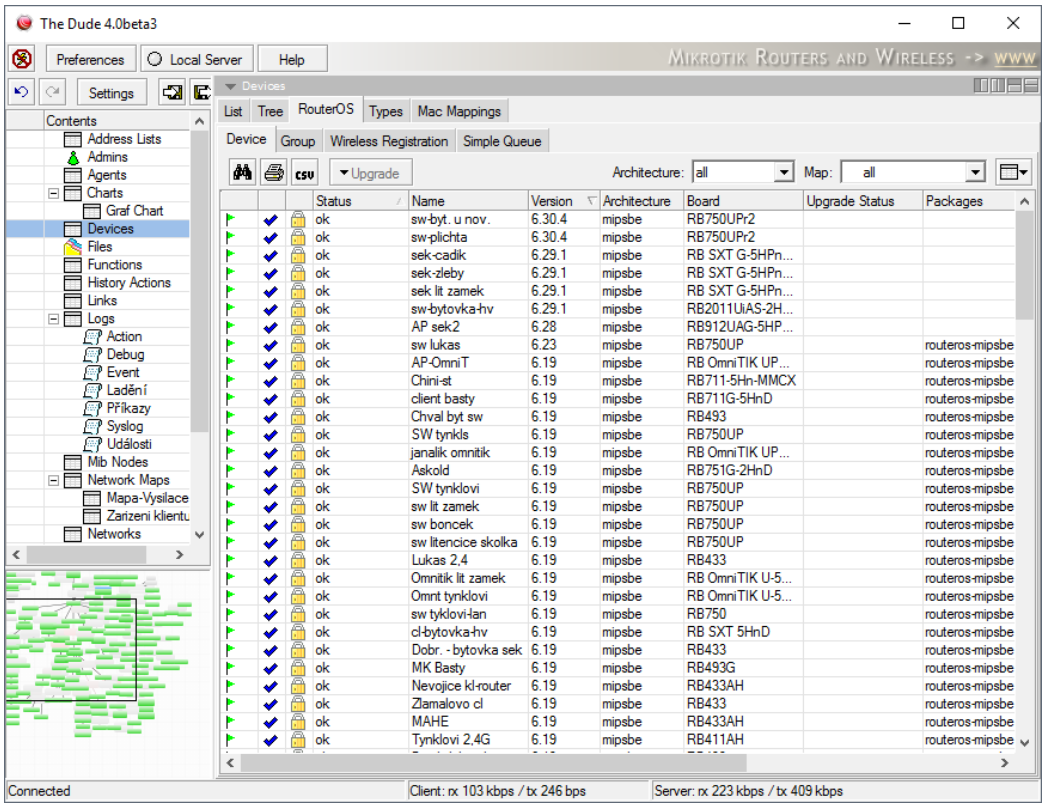
Bohužel ale už nekontroluje závislosti mezi zařízeními, může se tak při nevhodném výběru zařízení stát, že zatímco se novější verze nahrála do jednoho zařízení, které se právě restartuje, aby se zaktualizovalo, nahrávání do jiného (závislého) se přeruší (a již nenaváže). Je proto doporučeno vytvořit si skupiny, které obsahují pouze vzájemně nezávislá zařízení. Bohužel při změnách topologie sítě je potřeba myslet také na úpravu jednotlivých skupin.

V případě, že některé zařízení není přímo dostupné, lze využít v síti více agentů (tím se rozumí běžící instance systému), kteří spolu komunikují. Lze tak zvenčí zpřístupnit pouze tohoto agenta bez nutnosti

povolování přístupu přímo k zařízením. Správce pak vidí v přehledu všechna zařízení (může si je členit do map nebo skupin) ze všech agentů.

Monitoring služeb běžících na sledovaném zařízení

Pro každé zařízení lze definovat služby, které se mají sledovat, ke každé je přiřazena odpovídající sonda (15) (což je definice, jak se má zjistit, zda služba běží nebo ne – jaký protokol použít, na jaký port přistoupit, jaká data odeslat a jaká je očekávaná odpověď). Systém pak u služeb sleduje dostupnost



OBRÁZEK 2

(s nastavením jak často se má zjišťovat a jak dlouho čekat na odpověď), zaznamenává výpadky (je možné nastavit, kolik po sobě jdoucích neúspěšných pokusů se vyhodnotí jako výpadek) a umožňuje nastavit různé formy upozornění (od zapípání nebo zablikání přes zaznamenání do místního nebo vzdáleného logu až po odeslání emailu). Lze tak velmi granulárně rozlišovat důležitost služby a v případě potřeby informovat zodpovědnou osobu. Kromě notifikací se také výpadek služby projeví na mapě různou změnou barvy při výpadku části sledovaných služeb nebo (další změnou barvy) při výpadku všech. Pokud se někdo daným výpadkem již zabývá, může ho potvrdit, což se opět projeví změnou barvy a ostatní tak jsou o této skutečnosti informováni. Všechny barvy lze měnit, nicméně výchozí nastavení (zelená – vše v pořádku, oranžová – částečný výpadek, červená – úplný výpadek všech služeb, modrá – potvrzení řešení výpadku) je zvoleno vhodně.

Součástí monitoringu jsou i grafy sledovaných služeb (např. doba odezvy), u nichž lze nastavit, jak dlouho se mají a s jakou granularitou ukládat (data pro starší období se stále více agregují – např. pro poslední 2 hodiny se uchovávají všechny hodnoty, za poslední den se vytváří průměr za 5 minut, za

poslední týden se průměruje každá hodina atd.). Mezi grafy najdeme i datové toky sledovaných spojů mezi zařízeními. Obecně lze vytvářet graf s jakoukoliv přesností z jakéhokoliv zdroje (které lze libovolně uživatelsky definovat – např. dělat průměr odezvy různých zařízení, sčítat toky v různých částech sítě).

Webové rozhraní

Webové rozhraní na rozdíl od desktopového má velmi omezené možnosti. V podstatě v něm lze pouze pracovat se seznamy (prohlížet a přidávat položky, např. zařízení, sledované služby) a prohlížet mapu (která se obnovuje místo po sekundě každých 30 s, nelze kliknutím na zařízení zobrazit jeho podrobnosti nebo měnit jeho pozici). Je možné také zobrazovat grafy sledovaných hodnot pro jednotlivá zařízení (např. odezva) nebo spoje mezi nimi (příchozí/odchozí provoz).

Aktuální verze

Novější verzi (serverovou část) lze instalovat pouze jako balíček do systému RouterOS (obsahuje balíčky např. pro bezdrátové sítě, směrování, zabezpečení). V testovaném staženém obrazu již je balíček dude obsažen, stačí jej pouze dle návodu (16) příkazem `/dude set enabled=yes` povolit a poté router restartovat. Bohužel poslední verze už neobsahuje webové rozhraní, které bylo odstraněno.

2.3.2 Další nástroje

Při důkladnějším zkoumání dalších nástrojů pro správu sítí bylo vždy zjištěno, že se jedná pouze o podporu monitorování sítě (byť mnohdy na vysoké úrovni) s žádnou nebo minimální podporou pro nastavení (většinou pomocí protokolu SNMP) nebo dostupnou jako placený doplněk. Nemalé množství aplikací bylo úplně přeskočeno bez podrobnějšího zkoumání, protože disponovaly pouze desktopovým rozhraním.

OpenNMS

Jedním z hojně používaných nástrojů splňujícím kritéria na bezplatnou licenci a otevřený kód je OpenNMS (17). Po bližším prozkoumání bylo bohužel zjištěno, že je zaměřen na monitorování. Umožňuje sledovat různé systémy a služby pomocí široké škály podporovaných protokolů a údaje poté vykreslovat do nastavitelných grafů. Podporuje události a na jejich základě generuje notifikace nebo po propojení se systémy pro správu úkolů může dle nastavených pravidel vytvořit nový s popisem chyby.

Network Management Information System

Dalším zkoumaným nástrojem se stal NMIS (18). Z názvu lze usoudit, že by se mohlo jednat o nástroj pro správu sítě, nikoliv jenom sledování. Zatímco základní aplikace (sloužící jako monitoring) je vyvíjena jako open source, její součást pro správu opConfig (19) je již placená (pro větší množství než 20 zařízení, což je příliš limitující faktor).

2.4 Shrnutí

Nejdříve byly stanoveny požadavky a očekávání od nástroje, který by pomohl se správou síťových zařízení. U současných nástrojů byla zkoumána zejména správa uživatelů a konfigurace síťového nastavení. Protože referenční síť je z největší části provozována na nařízeních se systémem RouterOS, byla hlavní podmínkou spolupráce s tímto systémem. Nejpokročilejší možnosti nabízí aplikace The Dude od samotného výrobce RouterOS (bohužel nebyla dlouho aktivně vyvíjena), ale v současnosti nepodporuje správu uživatelů. Kromě hromadné aktualizace systému nemá další pokročilé možnosti hromadné správy více zařízení, umožňuje pouze pohodlně z jednoho prostředí měnit nastavení konkrétního zařízení. Ostatní zkoumané systémy se zaměřují hlavně na monitoring a pokud umožňují nastavovat některé vlastnosti, děje se tak pomocí protokolu SNMP (který v RouterOS nelze použít pro správu uživatelů).

3 Testovací prostředí

Pro testování funkčnosti různých aplikací není vhodné používat reálnou síť s uživateli, a to hned z několika důvodů. Předně uživatel nezná kvalitu testované aplikace, případné chyby mohou způsobit chybné provedení příkazů. Dále může obsahovat automatické procesy, které probíhají na pozadí – např. sama prozkoumává síťové okolí a snaží se zjistit běžící služby nebo se k některým připojit a získat z nich údaje. Podobné obtíže může způsobit neznalost uživatele chování dané aplikace (např. jestli je potřeba změny uložit nebo se provádějí automaticky). Rozhodně není žádoucí stav, kdy z nějakého obdobného důvodu testování aplikace dojde k výpadku služby koncovému uživateli. Pro vytvoření testovacího prostředí byl zvolen nástroj GNS3 (20), který umožňuje vytvářet virtuální zařízení (ať už routery, switche, huby nebo také počítače) a různě je mezi sebou (nebo i s hostitelským počítačem) propojovat. Díky tomu, že tento program splňoval nároky pro testování, nebyly zkoumány jiné alternativy.

3.1 Instalace GNS3 VM, zprovoznění, vytvoření prostředí

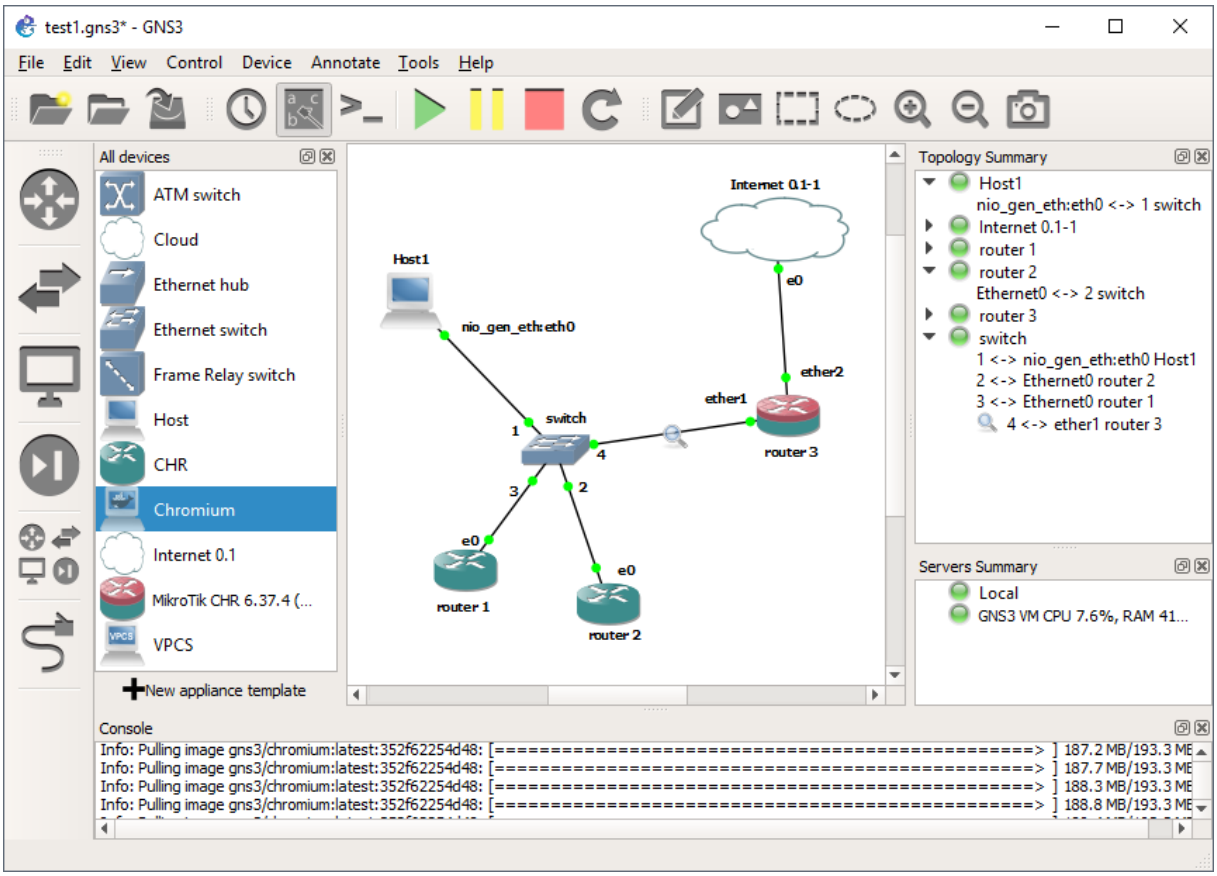
První překážkou při zprovozňování testovacího prostředí byla nutnost podpory virtualizační technologie procesorem (u procesorů Intel označována jako VT (21)). Přestože ji procesor podporuje, virtualizační (ale i identifikační) nástroje mohou zahlásit, že podporována není. Při vyhledávání problému na internetu jsou k dispozici zejména návody, jak povolit virtualizaci v BIOSu (několikanásobný vstup do něj, ověření nastavení a změna kombinace podporovaných technologií nemusí vést k úspěchu). Nakonec problém může vyřešit zkoumání chyby při spouštění některého dalšího virtualizačního nástroje. Odkaz na vyřešení opět pomocí restartu se vstupem do BIOSu již vyzkoušen byl, ale součástí návodu bylo také odinstalování Hyper-V, což se nakonec ukázalo jako průvodce celého problému.

Samotná instalace GNS3 je vcelku snadná. Nejdříve je nutné nainstalovat některý z virtualizačních nástrojů. Dle doporučení na stránkách GNS3 (rychlejší běh) byl nainstalován VMware Workstation Player (22), který je k nekomerčním účelům k dispozici zdarma. Pro něj lze pak ze stránek GNS3 stáhnout virtuální obraz (23). Po instalaci a spuštění GNS3 se zobrazí průvodce, ve kterém stačí vybrat daný virtualizační nástroj a obraz. Všechna testovací zařízení se pak spouští uvnitř virtuálního počítače z tohoto obrazu. Obrázek 3 ukazuje spuštěný program s několik běžícími a propojenými routery.

Spolu s programem lze volitelně nainstalovat další užitečné nástroje. Za zmínku stojí především program Wireshark (24), který umožňuje sledování síťového provozu. Stačí kliknout pravým tlačítkem na propojení dvou zařízení a vybrat možnost „Start capture“ (spustit zachytávání provozu). Zatím není možné sledovat provoz mezi dvěma routery, pouze na portech rozbočovačů. Stačí ale mezi routery umístit rozbočovač a zachytávat síťový provoz na něm. K čemu je tato schopnost dobrá, není potřeba příliš vysvětlovat – hodí se vždy pro kontrolu nebo ověření, jaká data mezi zařízeními putují a odladění vlastního programu nebo prozkoumání určitého protokolu.

Velkou výhodou GNS3 je možnost připojení k lokální konzoli každého zařízení. Není tak nutné mít zařízení vůbec propojená, nemusí mít ani nakonfigurovány síťová rozhraní, přesto je lze nastavovat a až poté propojit s připravenou konfigurací. Pokud nějaké zařízení přestane odpovídat po změně nastavení, hodí se tato možnost pro zjišťování příčin.

Součástí GNS3 je i Marketplace (25), v němž se nachází mnoho připravených zařízení různých výrobců, která nejsou standardní součástí programu. Lze v něm např. najít i „MikroTik Cloud Hosted Router“, který lze stáhnout a přidat do seznamu zařízení v GNS3.



OBRÁZEK 3

3.2 Licencování RouterOS pro testování

Pro testování RouterOS se nabízí několik možností licencí. První možností je si zakoupit licenci pro každý testovací router, což vzhledem k účelu není vhodná varianta. Mikrotik na svých stránkách nabízí ke stažení obrazy k instalaci systému, který běží bez omezení 24 hodin a poté je potřeba licenci zakoupit (nebo router smazat a vytvořit nový), do tohoto časového omezení se počítá pouze doba běhu, nikoliv čas, kdy byl vypnutý. V případě potřeby delšího testování je možné po zaregistrování na stránkách získat časově neomezenou verzi, ale s omezenou funkcionalitou.

Nejlepší možností pro daný účel (testování konfigurování zařízení) je tzv. verze Cloud Hosted Router, neboli varianta přímo určená pro běh ve virtualizovaném prostředí (předešlé varianty jsou určené pro

běh fyzických zařízeních, přestože je virtualizace možná). Opět je možné využít několik možností. Jednou z nich je časově omezená 60denní plně funkční zkušební verze (licenci lze získat pro zaregistrované uživatele). Během této doby lze produkt zakoupit (stálá licence včetně aktualizací) nebo po jejím uplynutí zůstane systém plně funkční, ale nepůjde jej aktualizovat. Poslední možností je bez časového omezení a bez nutnosti registrace (a je proto vhodná pro dlouhodobé testování, aniž by se musely routery mazat a znovu vytvářet), zato s omezením propustnosti, která je snížená na 1 Mb/s pro jednotlivé rozhraní, což vzhledem k zamýšlenému užívání není omezující. Z tohoto důvodu je využívána poslední možnost (lze neustále testovat aktualizace, aniž by bylo nutné do testovacího prostředí zasahovat).

3.3 Vytvoření zařízení/routerů

Do takto připraveného prostředí bylo vytvořeno několik routerů, které byly přes virtuální switch (přepínač) zapojeny k hostitelskému (virtuálnímu) rozhraní, které si vytvořil virtualizační nástroj. Na tomto rozhraní běží DHCP server, naproti tomu ve výchozím stavu na routerech běží na každém rozhraní DHCP klient, takže všechny routery získají IP adresu. Samozřejmě je možné toto chování změnit, nicméně pro rychlé počáteční testování nástrojů je takové chování žádoucí. Pro dlouhodobější testování je lepší varianta s ručním nastavením, u níž se nemohou měnit adresy jednotlivých routerů. Server DHCP může při každé žádosti o adresu přiřadit jinou, což by komplikovalo testování a vývoj aplikace.

4 Výběr vývojového prostředí a nástrojů

Celá aplikaci je rozčleněna na dvě logické části. První (nazvěme ji serverová neboli backend) je zodpovědná za komunikaci s jednotlivými zařízeními a uchovávání dat (vkládaných uživatelem nebo načtených ze zařízení). Obsahuje také rozhraní (API) pro komunikaci s jinými systémy (nebo napojení uživatelského rozhraní). Druhá (klientská neboli frontend) slouží k zobrazení dat a obecně interakci s uživatelem. Toto rozdělení nutně neznamená dvě různé samostatné aplikace, přestože k rozdělení nakonec došlo. Naproti tomu celá aplikace (obě části) může běžet jako jeden program na zařízení, z něhož se používá (stará se jak o přístup k síti a ukládání dat, tak o vykreslení uživatelského rozhraní).

4.1 Klientská aplikace

Aby byla klientská část aplikace dostupná z co nejširší škály zařízení (což z dnešního pohledu znamená stolní počítače s operačními systémy Windows, macOS a Linux, mobilní platformy se systémy Android, iOS a Windows Mobile – další platformy, které mají nízké zastoupení nebudou uvažovány), je možné využít několika možností vývoje.

- Vytvoření nativních aplikací pro každé prostředí, tato možnost je díky různorodosti nejnáročnější, protože vyžaduje vytvoření mnoha aplikací, na druhou stranu přináší uživatelům nejvyšší komfort; kvůli své náročnosti jak pro počáteční vývoj, tak pro další údržbu nebude o této variantě vůbec uvažováno.
- Další možností je využít nástrojů pro multiplatformní vývoj, například Xamarin pro .NET (s podporou jazyků C# a F#) umožňuje vytvoření jedné aplikace (včetně uživatelského rozhraní), která běží na Windows (od verze 8.1), Windows Mobile, Android (od verze 4.0.3) a iOS a s možností sdílet kód kromě uživatelského rozhraní také pro macOS (uživatelské rozhraní musí být vytvořené zvlášť).
- Poslední z variant je vytvoření webového rozhraní, které je přístupné ze všech platforem díky tomu, že v dnešní době každý běžný klientský počítač a mobil obsahuje webový prohlížeč.

Pro první dvě možnosti se jedná o samostatnou aplikaci, která se připojuje k API serverové části, třetí může být její součástí. Díky největšímu zastoupení webových prohlížečů a nejsnadnějšímu vývoji byla zvolena třetí variantu, pro níž lze rozhraní vytvořit dvěma rozdílnými způsoby.

- Webový prohlížeč slouží pouze pro zobrazení rozhraní, ale stránky se generují na straně serveru, s každou interakcí se posílá na klientský prohlížeč nová stránka (nebo její část). Takto funguje dnes většina webů a součástí .NET platformy je přesně pro tyto účely vytvořené ASP.NET (26) (standardní klasické nebo nové ASP.NET Core).

- Celá aplikace běží ve webovém prohlížeči, tento způsob se označuje jako „Single page application“ (často zkracováno jako „SPA“ (27)). Ze serveru se načte při přístupu na stránku celá aplikace, která pak interaguje se serverovou části přes API obdobně jako klasická desktopová aplikace. Serverová část tak slouží pouze jako úložiště načítaného programu a dat.

Volba padla na druhou možnost, protože poskytuje uživateli mnohem lepší zážitek z používání (rychlejší odezva), navíc disponuje dalšími pokročilými schopnostmi. Celou aplikaci lze např. používat i offline (po patřičné úpravě), může si lokálně ukládat data nebo z ní lze vytvořit desktopovou verzi.

4.1.1 Jednostránkové aplikace (SPA)

Existuje celá řada různých knihoven a frameworků pro vytváření SPA (27). Mezi hojně rozšířené patří jQuery (28), Angular (29), Ember (30) a React (31), existuje ale mnoho dalších. Zatímco jQuery je knihovna pro snadnou práci s DOM (vysvětleno dále v textu v následujícím odstavci), událostmi a AJAX (32), Angular a Ember jsou MVC (33) nebo MVVM (34) (dle použití (35)) frameworky, které poskytují kompletní řešení pro celou aplikaci (od práce s daty, modelem až po použití šablon pro vykreslení uživatelského rozhraní), React naproti tomu je knihovna pro tvorbu uživatelských rozhraní.

DOM (36) neboli „document object model“ je rozhraní pro reprezentaci HTML ve webovém prohlížeči. Každý prvek stránky (od zadávacího pole přes tlačítka až po textové odstavce) je reprezentován objektem, jednotlivé objekty se vzájemně zanořují (dle vlastních pravidel) a vytváří tak stromovou strukturu. Na začátku při načtení stránky prohlížeč z HTML vytvoří DOM, který následně vykreslí. Pro manipulaci s DOM objekty lze využít javascript (změna jednotlivých uzlů, přidávání, mazání). Tímto způsobem lze např. na základě výběru možnosti ve formuláři patřičně zobrazit další zadávací pole nebo měnit barvy atd. Pro usnadnění těchto operací (a zjednodušení pro různé verze různých prohlížečů) vznikla knihovna jQuery, kterou je vhodné použít hlavně pro oživení statických webových stránek interakcemi a animacemi (a pomoc s komunikací se serverem), nehodí se příliš pro komplexní webové aplikace. Pracuje přímo s prvky webové stránky (DOM), díky čemuž je mnohem pomalejší než moderní frameworky. Ve své době ale znamenala obrovský pokrok (vývojář nemusel řešit rozdílnosti jednotlivých prohlížečů) a dodnes se jedná o nejpoužívanější knihovnu na webových stránkách.

React popularizoval myšlenku tzv. „virtualdomu“. Zatímco prvky DOMu obsahují velké množství vlastností a navázaných událostí, prvky virtualdomu jsou jejich mnohem odlehčenou variantou (jedná se o objekty Javascriptu). Aplikace místo náročných a pomalých úprav DOMu celou stránku rychle vykreslí do virtualdomu, poté provede rozdíl původního virtualdomu (který je nyní převeden na DOM a zobrazován v prohlížeči) s novým (tato operace je taktéž rychlá díky použití vhodných algoritmů) a do pomalého DOMu provede pouze nezbytně nutné úpravy tak, aby odpovídal aktuálnímu virtualdomu. Tato technika umožňuje celou stránku vždy znovu rychle překreslit, autor se tak nemusí starat o složité inter-

akce a aktualizace částí webu (např. pro chatovací aplikaci, která kromě chatu zobrazuje i počet nepřečtených zpráv není nutné řešit, kde všude se počet nepřečtených zpráv zobrazuje, jednoduše se překreslí celá).

Nové myšlenky mezi frameworky a knihovny pro tvorbu SPA přináší práce „Elm: Concurrent FRP for Functional GUIs“ (37), jejíž součástí je i jazyk Elm (který se od doby práce dále rozvíjí), ale hlavně Elm architektura. Ta znamená další zjednodušení – model celé aplikace je uložen na jednom místě, pro zobrazení se použije funkce, která převede model na HTML (stejný model se vždy převede na stejné HTML), odpadá tak problém se synchronizací. Změny v modelu jsou vyvolány zprávami, které přijímá aktualizační funkce spolu s původním modelem a vytváří model úplně nový, který opět vstupuje do funkce pro zobrazení HTML. Kromě virtualdomu využívá Elm i další techniky (např. requestAnimationFrame (38)) pro efektivní a velmi rychlé renderování HTML.

Protože pro serverovou část je použit jazyk F#, nabízí se možnost ho využít i pro vývoj webového rozhraní SPA. Toto je možné díky projektu Fable (39), který převádí F# do Javascriptu. Pro tento projekt je navíc dostupná knihovna Elmish (40), která je postavená na základech Elm architektury a umožňuje využít libovolnou knihovnu nejen pro renderování HTML, ale i nativních aplikací (propojením s React Native (41)). Díky ohromné popularitě Reactu pro něj existuje velké množství užitečných nástrojů a knihoven, podporuje ho i Elmish pro renderování.

Pro vizuální stránku tvorby uživatelského rozhraní slouží další knihovny obsahující množství komponent pro zobrazení různých prvků na stránce (např. tlačítka, formulářová zadávací pole nebo navigaci) včetně designu (rozvržení) stránky samotné. Jednou z takových je Bootstrap (42), který je v projektu využíván. Je navržen od základů pro tvorbu responzivního (43) designu (stránka a jednotlivé prvky se přizpůsobují podle velikosti obrazovky).

Díky rozšířenosti Reactu jsou pro něj k dispozici i pokročilé ladící nástroje přímo v prohlížeči (např. „React Developer Tools“ (44), které zobrazí přímo v prohlížeči komponenty Reactu, přestože jsou ve výsledku renderování jako běžné DOM objekty). Dále je pro vývoj používán Webpack (45) (nejpoužívanější v komunitě Reactu), který obsahuje server pro vývoj, jež umožňuje tzv. „hot reload“ (při změně v kódu se automaticky synchronizuje i zobrazovaná stránka), což umožňuje rychlejší a snadnější vývoj. Knihovna Elmish navíc obsahuje podporu pro vývojářské nástroje, které jednotlivé zprávy zachytává a umožňuje se vracet ke kterýmkoliv dřívějším modelům (před zasláním zprávy), lze se tak vrátit k jakémukoliv stavu v aplikaci nebo naopak jednotlivé zprávy znovu „přehrát“.

4.2 Serverová část

Aby mohla aplikace (serverová část) komunikovat se zařízeními v síti, musí je mít přístupná. Toho lze dosáhnout dvěma způsoby. Buďto aplikace běží přímo uvnitř dané sítě nebo je k ní propojena pomocí VPN („virtual private network“ – slouží k připojení ke vzdálené síti odkudkoliv – koncové zařízení se

připojí k bodu, který má přístupnou požadovanou síť a tváří se, jako by bylo součástí vzdálené sítě). Z pohledu aplikace jsou obě možnosti rovnocenné (VPN se pro ni tváří transparentně).

Předpokladem je běh na některém z běžně používaných serverových systémů (Windows nebo Linux), resp. možnost běhu na kterémkoliv z nich. Existuje množství prostředí, která lze pro vytvoření této části využít. Z programovacích jazyků a prostředí bylo na výběr z několika možností:

- C++ – nativní aplikace pro obě platformy – standardně se zdrojové kódy programu kompilují na každé platformě zvlášť, ale je možné je zkompilovat i pouze na jedné z nich pro obě,
- Java – aplikace běží ve virtuálním stroji (JVM), který je dostupný pro obě platformy a velmi rozšířený, kompiluje se pouze jednou pro obě platformy,
- C# nebo F# – součástí .NET platformy – velmi podobné z obecného pohledu Javě – aplikace taktéž běží ve virtuálním stroji, který je součástí téměř každé instalace Windows, naopak na Linuxu není tak rozšířený jako JVM,
- Javascript – původně jazyk pro webové stránky s podporou ve webových prohlížečích, dnes (zejména díky Node.js) lze psát i serverovou část (dříve bylo taktéž možné, ale ne zdaleka tak rozšířené), díky čemuž velmi rychle roste na popularitě, umožňuje sdílet kód mezi serverovou a klientskou částí; díky rozšířenému frameworku Electron (46) je vhodný i pro vytváření multiplatformních desktopových aplikací, které vypadají na všech platformách totožně. Vývojáři používají stejný jazyk se stejnými nástroji pro vše – tvorbu webových, desktopových aplikací i serverové části

Vzhledem k dlouhé zkušenosti s prostředím .NET padla volba na něj a na jazyk F#.

4.2.1 Webový server

Kromě komunikace se síťovými zařízeními se serverová část musí postarat také o komunikaci s klientskou částí a vzhledem k tomu, že se jedná o webovou aplikaci, musí se také odněkud načítat. Ke splnění obou požadavků slouží právě webový server. Webový prohlížeč se k němu připojí, načte a zobrazí patřičnou aplikaci. V případě SPA potom dále se serverem komunikuje – posílá na něj a načítá z něj potřebná data (např. údaje o zařízení), která poté zobrazí uživateli.

4.2.1.1 Protokol HTTP

Komunikace s webovým serverem probíhá pomocí protokolu HTTP (47) (48) nebo jeho šifrované varianty HTTPS. Prohlížeč pošle požadavek (na dokument) na server a ten pošle zpět dokument (spolu se stavovým kódem o úspěchu). Pokud není z nějakého důvodu dokument vrácen, posílají se jiné stavové kódy – např. s informací, že byl dokument přemístěn (a kam) nebo že neexistuje. Stavové kódy pro různé situace jsou přesně definovány v RFC 7231 (49). Přes HTTP protokol probíhá standardně přenos všech součástí dokumentu (texty, obrázky, videa apod.).

4.2.1.2 Komunikace webového serveru s klientem

Protože HTTP požadavek iniciuje vždy klient a server na něj pouze odpovídá, nemůže server při změně právě zobrazovaných dat (např. obdržení odpovědi od spravovaného zařízení) poslat tuto zprávu klientovi. Pokud tedy probíhá nějaká úloha na serveru na pozadí a chceme její dílčí výsledky zobrazovat v prohlížeči, musí se prohlížeč neustále dotazovat na aktuální stav v určitém intervalu (např. 1 s), uživateli se tak zobrazují aktuální data a nemusí sám neustále obnovovat stránku. Tato možnost (označována jako „polling“) ovšem znamená zbytečné plýtvání prostředky (klient se neustále dotazuje na totéž a mnohdy dostává neustále stejnou odpověď).

Existují však techniky, jak dosáhnout možnosti zaslání dat ze serveru, aniž by se klient neustále dotazoval. Protokol HTTP nebyl navržen pro obousměrnou komunikaci, i proto starší metody „HTTP long polling“ a „HTTP streaming“ (50) s sebou přinášely mnohá omezení a neefektivní využívání prostředků. HTTP long polling funguje tak, že klient pošle požadavek, ale server spojení neuzavírá a čeká, až bude mít data k dispozici, poté je pošle a spojení uzavírá. Klient stejným způsobem opět navazuje nové spojení (buďto okamžitě nebo s krátkou prodlevou) a celý proces se s každými daty opakuje. HTTP streaming navazuje jedno spojení, přes které server posílá po celou dobu potřebná data.

Později vznikly další novější možnosti komunikace serveru s klientem. Na základech HTTP streamingu je založena specifikace Server-sent events (51) (klient naváže spojení a server mu může kdykoliv poslat zprávu, která se u klienta projeví ve formě události s daty). Bohužel tuto metodu nepodporují stále nezanedbatelně zastoupené prohlížeče Microsoftu (Internet Explorer a Edge) (52).

Relativně novou možností je protokol Websockets (53) (54), který umožňuje obousměrné zasílání zpráv. V případě, že se na serveru nějaká data změní, může být klientská aplikace ihned informována a uživateli tak zobrazit aktuální data s minimálním zpožděním, navíc nedochází k plýtvání prostředků – klientská aplikace

4.2.1.3 Autentizace

Jedná se o proces ověření identity uživatele, tj. zda je opravdu tím, za koho se vydává. Samotný protokol HTTP přímo podporuje a popisuje možnosti autentizace (55).

„Basic“ autentizace (56) spočívá v zasílání přihlašovacích údajů (uživatelské jméno a heslo) s každým požadavkem. Samotný prohlížeč vyzve uživatele k jejich zadání, běžně si údaje zapamatuje, aby je nemusel uživatel s každým načtením stránky znovu zadávat, a pak je při další komunikaci se serverem (při vstupu na určité stránky) automaticky posílá. Nevýhodou tohoto přístupu je, že se uživatel prakticky nemůže odhlásit a přihlašovací údaje se posílají s každým požadavkem (o tom, jak dlouho se údaje uchovají, rozhoduje prohlížeč – typicky se údaje mažou po zavření okna). Nelze tak např. vynutit automatické odhlášení po delší době nečinnosti. Z pohledu uživatelského prostředí aplikace není možné mít vlastní vzhled obrazovky pro přihlášení, opět se o vše stará prohlížeč (což na druhou stranu přináší

výhodu pro vývojáře, že nemusí formulář pro přihlášení vytvářet). Bezpečnější varianta autentizace je pomocí „HTTP Digest Access Authentication“ (57).

Další hojně využívanou možností autentizace je pomocí standardního HTML formuláře (58). Pro přihlášení je speciálně vytvořená stránka (nebo část) s formulářem pro zadání přihlašovacích údajů. Pokud uživatel přistupuje ke chráněné části webu, server jej přesměruje na přihlašovací stránku, uživatel zadá přihlašovací údaje a po zpracování odeslaných údajů (úspěšné autentizaci a autorizaci) získá k této části přístup. Vlastní řešení bývá často pomocí cookies (59) nebo předáváním vygenerovaného identifikátoru v každé adrese jako její parametr (všechny odkazy v rámci webu jej obsahují). Tento přístup ovšem přináší nevýhodu, pokud aplikace (webové stránky) přistupuje ke chráněnému zdroji z jiné domény (běží na jiné adrese, než odkud se načítají některá data). Je tak potřeba řešit bezpečnost z pohledu CORS (60) a CSRF (61). Nepraktické může být také připojení k takovému zdroji odjinud než z webového prohlížeče (např. mobilní nebo desktopová aplikace – vyžaduje práci s cookies kontejnery (62) pro uchování a odesílání cookies).

V aplikaci je pro své výhody používán *JSON web token* (63) se schématem „Bearer“ (64). Po odeslání přihlašovacích údajů (uživatelské jméno a heslo) je na serveru vytvořen token, který je vrácen do klientské aplikace, ta jej s dalšími požadavky přidává do HTTP hlavičky („Authorization: Bearer *token*“) a na serveru proběhne jeho ověření. Namísto přidání do hlavičky může být předán i jiným způsobem, např. přímo mezi daty požadavku. Token může také mj. obsahovat údaje o délce své platnosti. Pomocí tokenu lze přistupovat k jakémukoliv zdroji na serveru nebo i jiných serverech, bez ohledu na původní adresu, odkud se požadavek posílá, případně se pomocí něj přihlašovat i do jiných služeb. Protože není potřeba řešit, kde běží aplikace a k jakému zdroji se připojuje, lze snadno použít CDN (65) (síť serverů po celém světě s cílem „být co nejblíže uživateli“) pro rychlé načítání aplikace a poté se připojí k datům z jiného zdroje. Při použití tokenu na rozdíl od formulářů není potřeba pro relaci ukládat údaje na serveru (a s každým požadavkem je z úložiště načítat), token v sobě obsahuje všechny potřebné údaje a ověření probíhá pouze kontrolou podpisu.

Pokud se používá standardní nešifrovaný protokol HTTP, putují i údaje pro autentizaci nešifrovaně a jsou náchylné na různé druhy útoků, proto je vhodné komunikaci zabezpečit (např. použitím HTTPS).

5 Analýza a návrh aplikace

Základem celé aplikace jsou dvě části, serverová a klientská. Serverová běží jako webový server, který poskytuje rozhraní klientské části a zároveň obstarává komunikaci se zařízeními a ukládání dat. Klientská část je vytvořena webovou aplikací, která se načítá z webového serveru a dále s ním komunikuje přes definované rozhraní.

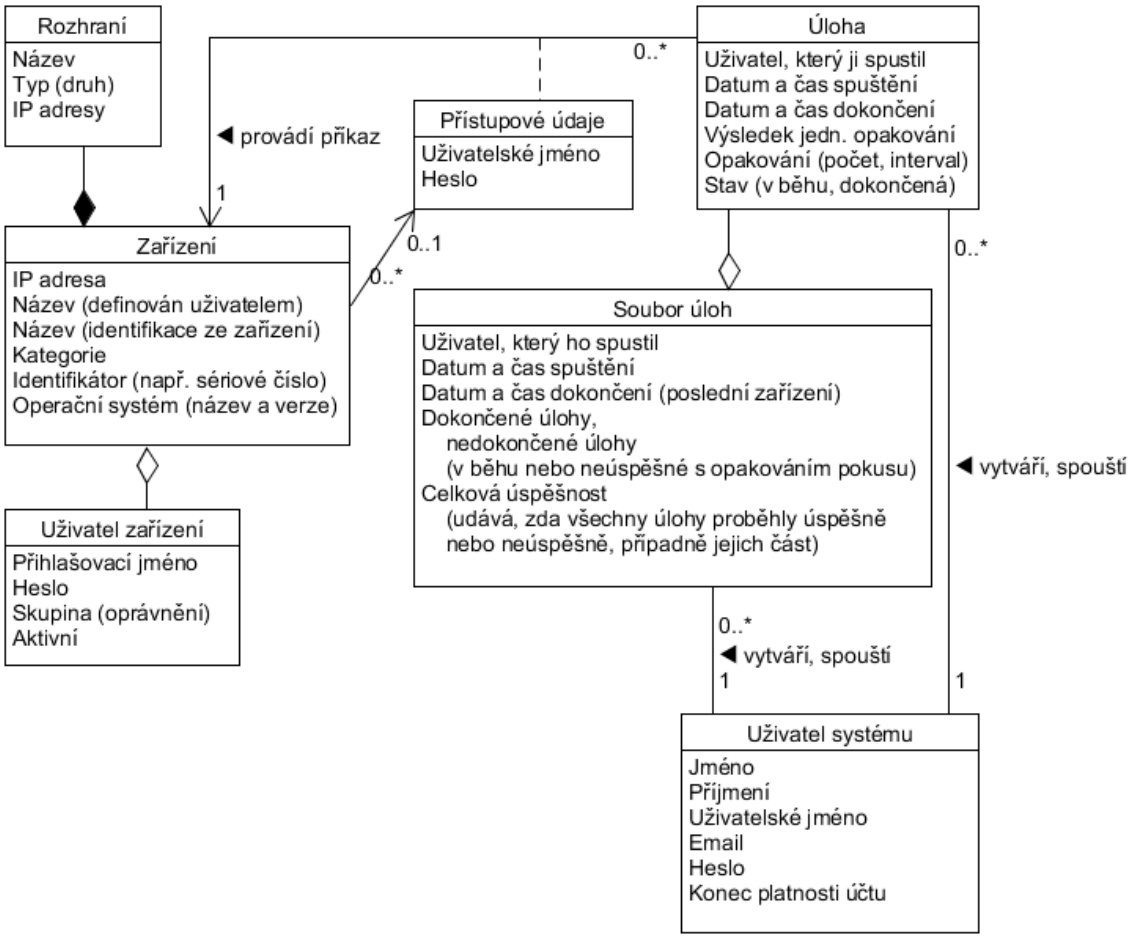
5.1 Ukládání dat

Aplikace potřebuje ke svému běhu ukládat různá data. Pokud mají mít uživatelé možnost se přihlašovat, musí vést jejich seznam včetně (hashovaných) hesel. Stejně tak informace o spravovaných zařízeních musí být uložena pro další načítání, zobrazení a práci s nimi.

Protože možností, jak ukládat data je mnoho (strukturovaný soubor, databáze SQL, NoSQL, grafová, objektová), a ještě více konkrétních implementací, poskytuje aplikace obecné rozhraní pro přístup k datům a přes něj se může napojit jakýkoliv zdroj dat (klidně se může jednat pouze o data uložená v operační paměti nebo souboru ve formátu JSON, XML, nebo jakémkoliv jiném). Samozřejmě je nutné dané napojení naprogramovat a namapovat konkrétní zdroj dat do aplikace.

5.2 Doménový model, jednotlivé významné entity

Každá entita má evidován jednoznačný identifikátor (id), který není dále jednotlivě uváděn. Seznam obsahuje pouze základní nosné entity bez pomocných vzniklých jako implementační detail.



OBRÁZEK 4

Uživatel systému

Jedná se o základní entitu nutnou pro přihlášení do aplikace – validuje se podle ní přihlašovací jméno a heslo uživatele. Nelze ji smazat (z důvodu návaznosti jiných entit – pro uchování historie), pouze ukončit platnost (a to pouze k aktuálnímu nebo budoucímu datu a času), kterou nelze znovu obnovit.

Všichni uživatelé v aplikaci mají stejná práva, neřeší se role nebo přístupová práva k jednotlivým entitám nebo částem aplikace.

Evidované údaje:

- Jméno
- Příjmení
- Uživatelské jméno (unikátní)
- Email
- Heslo (hashovaná podoba)
- Platnost účtu (do kdy platí) – zjednodušeno na možnost přihlášení v daném období (nekontroluje se při provádění akcí)

Zařízení

K zařízení se aplikace připojuje pomocí IP adresy nebo jeho názvu (který je ve výsledku přeložen na IP adresu, ale o tuto část se již nestará aplikace, ale některá z nižších vrstev, nad níž běží – framework nebo operační systém).

Evidované údaje zadávané uživatelem

- IP adresa / název hostitele (slouží k připojení k zařízení)
- Vlastní (uživatelský) název
- Přihlašovací údaje (jméno a heslo)
- Kategorie (dle použití daného zařízení – klientské, přístupové apod.) – slouží zejména pro hromadnou konfiguraci (např. servisní technik má vytvořen účet ve všech klientských zařízeních)

Další údaje, které jsou načítané ze zařízení a nelze je uživatelem v systému přímo měnit

- Unikátní identifikátor zařízení (pokud existuje, může jím být např. sériové číslo, slouží k odhalení případných duplicit v seznamu zařízení – pod 2 různými IP adresami/názvy hostitele se může skrývat totéž zařízení)
- Název (identifikace) zařízení
- Operační systém (název a verze)
- Rozhraní (seznam)
    - Název
    - Typ (druh) – metalické, optické, bezdrátové, virtuální apod.

- Přiřazené IP adresy (vč. masky)

Úloha (příkaz) pro jednotlivé zařízení

- Uživatel může spustit novou úlohu, zastavit nebo změnit interval opakování probíhající
- Uživatel, který ji spustil
- Datum a čas spuštění
- Datum a čas dokončení
- Výsledek jednotlivého opakování (úspěch, neúspěch – důvod)
- Počet opakování a interval mezi jednotlivými pokusy

Typy úloh (jednotlivé příkazy)

- Otestování dostupnosti zařízení (pomocí přístupu na TCP port, na němž protokol běží, nebo odezvy na ICMP echo)
- Otestování vyplněných přístupových údajů k zařízení
- Načtení údajů ze zařízení
- Vytvoření nového uživatele v zařízeních
- Smazání uživatele ze zařízení
- Přidání IP adresy k rozhraní
- Odebrání IP adresy z rozhraní
- Spuštění uživatelem napsaného příkazu

Soubor úloh

Pokud v jeden okamžik spouští uživatel stejnou úlohu pro více zařízení, jedná se o jejich soubor.

- Uživatel, který ho spustil
- Datum a čas spuštění
- Datum a čas dokončení (poslední zařízení)
- Dokončené úlohy, nedokončené úlohy (v běhu nebo neúspěšné s opakováním pokusu)
- Celková úspěšnost (udává, zda všechny úlohy proběhly úspěšně nebo neúspěšně, případně jejich část)

5.2.1 Definice rozhraní přístupu k datům (databáze)

Pro univerzální přístup k datům (s možností ukládání v různé podobě) je nezbytná definice rozhraní, které musí jednotlivá implementace přístupu k datům splnit.

Uživatel

- Načtení seznamu uživatelů
- Vyhledání uživatelů dle libovolných údajů (např. dle přihlašovacího jména a hesla pro přihlášení)

- Úprava uživatele (změna údajů)
- Vytvoření uživatele

Zařízení

- Načtení seznamu zařízení (s možností filtrace dle zadaných údajů)
- Přidání nového zařízení
- Úprava stávajícího zařízení
- Smazání stávajícího zařízení

Úloha

- Vytvoření nové úlohy
- Zápis výsledku úlohy

5.3 Správa uživatelů aplikace

Slouží pro přidávání, úpravy nebo odebírání (deaktivaci) uživatelů, kteří se mohou do aplikace přihlásit a pracovat s ní. Aplikace neřeší uživatelské role (všichni mohou provádět stejné operace). Každý má správu nad ostatními uživateli (mohou s nimi libovolně pracovat).

5.4 Správa zařízení

Kromě standardních možností přidávání, editace a mazání zařízení je možné také provádět

- Test dostupnosti
  - pomocí ICMP echo (66) (ping) nebo
  - zadáním protokolu (TCP nebo UDP) a portu (číslo)
- Test přihlášení – s (před)definovanými jmény a hesly

5.5 Správa síťových rozhraní

Rozhraní jsou načtena ze zařízení automaticky a není možné je přidávat nebo mazat. Lze jim pouze konfigurovat síťová nastavení (přidělit, upravit nebo odebrat IP adresu).

5.6 Správa uživatelů zařízení

Kromě možnosti úpravy uživatele (přidání, odebrání, úprava) pro jedno zařízení lze provádět tyto úpravy hromadně pro vybraná (skupinu) zařízení. Zařízení mohou být vybírána např. na základě typu (všechna daného typu). Do takto vybraných zařízení je možné

- Hromadně přidat uživatele
- Hromadně smazat určitého uživatele

5.7 Komunikace se zařízeními

S každým zařízením je navázáno spojení nezávislé na jiných zařízeních a na jiných úlohách, můžou tak běžet současně různé požadavky na jedno zařízení. Každý příkaz se na zařízení buď úspěšně provede, nebo skončí s chybou (např. zařízení daný příkaz nezpracuje korektně, není navázáno spojení, přístupové údaje jsou chybné). Výsledek komunikace a úspěšnost provedení příkazu je vrácen aplikaci k dalšímu zpracování (typicky zobrazení uživateli s podrobnostmi o chybě). Všechny příkazy k zařízení jsou ukládány včetně výsledků, je tak možné v jejich historii zjistit, jaké operace a kdy byly s kterým zařízením prováděny.

5.8 Instalace, spuštění

Výslednou aplikaci lze přímo spustit a začít používat, jedná se o samostatný program, je vyžadována minimální konfigurace. Ve výchozím stavu webový server naslouchá na standardním portu 80 (který je možné konfigurovat) a data ukládá do SQL databáze. Zvolený ovladač SQLProvider (67) je univerzální a podporuje velké množství různých databází – např. MySQL, PostgreSQL, Oracle, SQLite. Testováno bylo s MS SQL Server (68). Databázi je potřeba před prvním použitím iniciovat pomocí skriptu (součást zdrojových kódů) pro vytvoření databázové struktury a naplnění počátečních dat. Aby mohla aplikace s databází komunikovat, je potřeba nastavit před spuštěním řetězec pro připojení k databázi.

5.8.1 Vývojářská verze

Pro sestavení aplikace ze zdrojových kódů stačí spustit skript „build“ s patřičným parametrem (popsány ve skriptu), podle něj dojde automaticky k sestavení a spuštění serverové a (nebo) klientské části v režimu naslouchání na změny ve zdrojových kódech s automatickým novým sestavením. Po něm je aplikace restartována, aby reflektovala provedené změny, případně zobrazovaná stránka obnovena. Všechny požadované součásti (knihovny) jsou automaticky staženy z internetu, především se jedná o balíčky z repozitáře Nuget (69).

5.8.2 Produkční verze

Na rozdíl od vývojářské verze obsahuje již všechny potřebné součásti pro běh ve zkompilované podobě, tzn. není potřeba žádných nástrojů na sestavení. Tato verze neobsahuje žádné ladící informace, je optimalizována pro rychlý běh a je určena k použití v reálném prostředí.

Budoucí rozšíření

Jednou z oblastí, kterou lze dále rozšiřovat je zobrazování různých druhů grafů. Kromě standardních sloupcových nebo spojnicových lze použít další kombinace (koláčový, paprskový) nebo vytvořit vlastní, stejně tak lze prozkoumat možnosti použití jiných měřítek, např. logaritmického. Zobrazování grafů samotné nabízí mnoho dalších příležitostí k vylepšování, které by vydaly na samostatnou práci, příkladem může být pokročilé ovládání myší (posunování po osách přetahováním, přiblížení nebo oddálení

kolečkem, zobrazení podrobností po najetí kurzorem nad hodnotu) nebo skládání a porovnávání různých hodnot různých zařízení do jediného grafu.

Rozšíření je také možné v oblasti notifikací na různé události – překročení hodnoty, dlouhodobé vytížení sítě apod.

Vzhledem k možnosti propojení na jiné systémy se taktéž nabízí možnost integrace nebo vytvoření dalších klientů kromě webového. Jako nejužitečnější (pro nejvíce potencionálních uživatelů) se jeví klasická desktopová nebo mobilní aplikace.

Závěr

Práce se zabývá možnostmi správy sítě s mnoha zařízeními. Protože referenční síť sestává z většiny ze zařízení s operačním systémem RouterOS, zaměřuje se právě na tento systém. Nejdříve prozkoumává současné nástroje se zjištěním, že většina z nich se zaměřuje na monitoring a nastavení umožňují pomocí SNMP, které je bohužel vzhledem k požadavkům nevhodné, protože neposkytuje v zařízeních se systémem RouterOS veškeré možnosti nastavení. Pro připojení k jednotlivým zařízením byly využity protokoly telnet a SSH.

Uživatelské rozhraní je vytvořeno s cílem maximální jednoduchosti a intuitivního ovládání tak, aby nebyl potřeba žádný manuál k použití.

Reference

1. **MIKROTIK SIA.** MikroTik Routers and Wireless: Software. [Online] [Citace: 3. leden 2017.] https://mikrotik.com/software.

2. **aj., Jeffrey Case.** RFC 1157 - Simple Network Management Protocol (SNMP). [Online] [Citace: 28. leden 2018.] https://tools.ietf.org/html/rfc1157.

3. **MIKROTIKLS SIA.** Manual:SNMP - MikroTik Wiki. [Online] [Citace: 28. leden 2018.]

4. **Postel, J. a Reynolds, J.** RFC 854 - Telnet Protocol Specification. [Online] [Citace: 3. únor 2017.] https://tools.ietf.org/html/rfc854.

5. **Ylonen, T. a Lonvick, C.** The Secure Shell (SSH) Protocol Architecture. [Online] [Citace: 12. leden 2017.] https://www.ietf.org/rfc/rfc4251.txt.

6. **MIKROTIKLS SIA.** Manual:TOC - MikroTik Wiki. [Online] [Cited: listopad 5, 2017.] https://wiki.mikrotik.com/wiki/Manual:TOC.

7. **Shafranovich, Yakov.** RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files. [Online] [Cited: 11 5, 2017.] https://tools.ietf.org/html/rfc4180.

8. **Close, Josh.** CsvHelper. [Online] [Cited: 11 05, 2017.] http://joshclose.github.io/CsvHelper/.

9. **Petricek, Tomas and Guerra, Gustavo.** F# Data: CSV Parser. [Online] [Cited: 11 05, 2017.] http://fsharp.github.io/FSharp.Data/library/CsvFile.html.

10. **MIKROTIKLS SIA.** Manual:Scripting - MikroTik Wiki. [Online] [Cited: listopad 5, 2017.] https://wiki.mikrotik.com/wiki/Manual:Scripting.

11. —. Manual:API - MikroTik Wiki. [Online] [Cited: leden 28, 2018.] https://wiki.mikrotik.com/wiki/Manual:API.

12. **Zoho Corporation Pvt. Ltd.** Network Configuration Management. [Online] [Citace: 22. květen 2016.] https://www.manageengine.com/network-configuration-manager/index1.html.

13. **SolarWinds Worldwide, LLC.** Automated network configuration and change management software. [Online] [Citace: 26. květen 2016.] http://www.solarwinds.com/network-configuration-manager.

14. **MIKROTIKLS SIA.** The Dude. [Online] [Citace: 3. leden 2017.] https://mikrotik.com/thedude.

15. —. Manual:The Dude v6/Probes - MikroTik Wiki. [Online] [Citace: 3. ledna 2017.] https://wiki.mikrotik.com/wiki/Manual:The_Dude_v6/Probes.

16. —. Manual:The Dude v6/Installation - MikroTik Wiki. [Online] [Citace: 3. leden 2017.] http://wiki.mikrotik.com/wiki/Manual:The_Dude_v6/Installation.

17. **OpenNMS Group, Inc.** *Enterprise Grade Network Management.* [Online] OpenNMS Group, Inc. [Citace: 22. květen 2016.] http://www.opennms.org/.

18. **Opmantek Pty Ltd.** Network Management System. Free Software Tools. NMIS. [Online] [Citace: 17. leden 2017.] https://opmantek.com/network-management-system-nmis/.

19. —. Configuration Management | Network. [Online] [Citace: 17. leden 2017.] https://opmantek.com/network-configuration-management-opconfig/.

20. **GNS3 Technologies Inc.** GNS3 | The software that empowers network professionals. [Online] [Citace: 11. prosinec 2016.] https://gns3.com/.

21. **Burger, Thomas Wolfgang.** Intel® Virtualization Technology for Directed I/O (VT-d): Enhancing Intel platforms for efficient virtualization of I/O devices | Intel® Software. [Online] [Cited: listopad 26, 2017.] https://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices.

22. **VMware, Inc.** VMware Workstation Player - VMware Products. [Online] [Citace: 11. leden 2017.] http://www.vmware.com/products/player/playerpro-evaluation.html.

23. **GNS3 Technologies Inc.** Software - VM Download - GNS3. [Online] [Citace: 8. únor 2017.] https://gns3.com/software/download-vm.

24. **Wireshark Foundation.** Wireshark · Go Deep. [Online] [Citace: 11. leden 2017.] https://www.wireshark.org/.

25. **GNS3 Technologies Inc.** Marketplace - Appliances - GNS3. [Online] [Citace: 11. leden 2017.] https://gns3.com/marketplace/appliances.

26. **Microsoft Corporation.** *ASP.NET | The ASP.NET Site.* [Online] [Citace: 3. únor 2017.] https://www.asp.net/.

27. **Code School LLC.** Single-page Applications | Code School. [Online] [Citace: 10. prosinec 2017.] https://www.codeschool.com/beginners-guide-to-web-development/single-page-applications.

28. **The jQuery Foundation.** *jQuery.* [Online] [Citace: 3. únor 2017.] http://jquery.com/.

29. **Google Inc.** *One framework. - Angular.* [Online] [Citace: 3. únor 2017.] https://angular.io/.

30. **TILDE INC.** *Ember.js - A framework for creating ambitious web applications.* [Online] [Citace: 3. únor 2017.] http://emberjs.com/.

31. **Facebook Inc.** *A JavaScript library for building user interfaces - React.* [Online] [Citace: 3. únor 2017.] https://facebook.github.io/react/.

32. **Dria.** Ajax - Web developer guides | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

33. **Mills, Chris.** MVC architecture - App Center | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.

34. **Microsoft Corporation.** The MVVM Pattern. [Online] [Citace: 9. prosinec 2017.] https://msdn.microsoft.com/en-us/library/hh848246.aspx#sec4.

35. **Google Inc.** Angular Docs. [Online] [Cited: prosinec 9, 2017.] https://angular.io/guide/template-syntax#template-syntax.

36. **Mozilla Corporation.** Introduction to the DOM - Web APIs | MDN. [Online] [Citace: 9. prosinec 2017.] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.

37. **Czaplicki, Evan.** Elm: Concurrent FRP for Functional GUIs. 2012.

38. **Shepherd, Eric.** window.requestAnimationFrame() - Web APIs | MDN. [Online] [Cited: prosinec 9, 2017.] https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame.

39. **Garcia-Caro, Alfonso.** Fable: JavaScript you can be proud of! [Online] [Citace: 10. prosinec 2017.] http://fable.io/.

40. **Tolmachev, Eugene.** Elmish. [Online] [Citace: 11. prosinec 2017.] https://fable-elmish.github.io/elmish/.

41. **Facebook Inc.** React Native · A framework for building native apps using React. [Online] [Citace: 10. prosinec 2017.] https://facebook.github.io/react-native/.

42. **Twitter, Inc.** [Online] [Citace: 10. prosinec 2017.] http://getbootstrap.com/.

43. **Michálek, Martin.** *Vzhůru do (responzivního) webdesignu.* s.l. : Michálek Martin - Vzhůru dolů, 2017. 978-80-88253-00-6.

44. **Facebook Inc.** GitHub - facebook/react-devtools: An extension that allows inspection of React component hierarchy in the Chrome and Firefox Developer Tools. [Online] [Citace: 10. prosinec 2017.] https://github.com/facebook/react-devtools.

45. **Koppers, Tobias.** webpack. [Online] [Citace: 10. prosinec 2017.] https://webpack.js.org/.

46. **GitHub, Inc.** Electron - Build cross platform desktop apps with JavaScript, HTML, and CSS. [Online] [Citace: 3. únor 2017.] http://electron.atom.io/.

47. **Roy Fielding, Julian Reschke.** Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. [Online] [Citace: 13. prosinec 2017.] https://tools.ietf.org/html/rfc7230.

48. **další, Mike Belshe a.** RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2). [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc7540.

49. **Fielding, Roy a Reschke, Julian.** RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. [Online] [Citace: 16. prosinec 2017.] https://tools.ietf.org/html/rfc7231#section-6.

50. **Loreto, Salvatore, a další.** RFC 6202 - Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc6202.

51. **WHATWG.** HTML Standard. [Online] [Citace: 17. prosinec 2017.] https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events.

52. **Deveria, Alexis.** Can I use... Support tables for HTML5, CSS3, etc. [Online] [Citace: 17. prosinec 2017.] https://caniuse.com/#search=Server-sent%20events.

53. **Fette, Ian a Melnikov, Alexey.** RFC 6455 - The WebSocket Protocol. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc6455.

54. **WHATWG.** HTML Standard. [Online] [Citace: 17. prosinec 2017.] https://html.spec.whatwg.org/multipage/web-sockets.html.

55. **Fielding, Roy a Reschke, Julian.** RFC 7235 - Hypertext Transfer Protocol (HTTP/1.1): Authentication. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7235.

56. **Reschke, Julian.** RFC 7617 - The 'Basic' HTTP Authentication Scheme. [Online] [Citace: 17. prosinec 2017.] https://tools.ietf.org/html/rfc7617.

57. **Shekh-Yusef, Rifaat, Ahrens, David a Bremer, Sophie.** RFC 7616 - HTTP Digest Access Authentication. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7616.

58. **aj., Yutaka Oiwa.** RFC 8053 - HTTP Authentication Extensions for Interactive Clients. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc8053.

59. **Barth, Adam.** RFC 6265 - HTTP State Management Mechanism. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc6265.

60. **Kesteren, Anne van.** Cross-Origin Resource Sharing. [Online] [Citace: 29. leden 2018.] https://www.w3.org/TR/cors/.

61. **The Open Web Application Security Project.** Cross-Site Request Forgery (CSRF) - OWASP. [Online] [Citace: 29. leden 2018.] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

62. **Microsoft.** CookieContainer Class (System.Net). [Online] [Citace: 31. leden 2018.] https://msdn.microsoft.com/en-us/library/system.net.cookiecontainer%28v=vs.110%29.aspx?f=255&MSPPError=-2147217396.

63. **Jones, Michael, Bradley, John a Sakimura, Nat.** RFC 7519 - JSON Web Token (JWT). [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc7519.

64. **Jones, Michael a Hardt, Dick.** RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc6750.

65. **Jahoda, Bohumil.** K čemu slouží CDN? [Online] [Citace: 31. leden 2018.] http://jecas.cz/cdn.

66. **Postel, J.** RFC 792 - Internet Control Message Protocol. [Online] [Citace: 22. prosinec 2017.] https://tools.ietf.org/html/rfc792.

67. **McKinlay, Ross.** SQLProvider. [Online] [Citace: 24. leden 2018.] http://fsprojects.github.io/SQLProvider/index.html.

68. **Microsoft.** SQL Server 2016. [Online] [Citace: 24. leden 2018.] https://www.microsoft.com/cs-cz/sql-server/sql-server-2016.

69. —. NuGet Gallery | Home. [Online] [Citace: 22. prosinec 2017.] https://www.nuget.org/.