FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Monitorování provozu sítě pomocí Model-Driven Telemetry
Student:	Ladislav Loub
Vedoucí:	Ing. Tomáš Čejka, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Seznamte se s problematikou monitorování stavu a vytížení síťových prvků a jeho rozhraní.
Seznamte se síťovými protokoly NETCONF a gRPC a jazykem pro popis datových konfiguračních modelů YANG.
Prostudujte existující YANG modely k síťovým prvkům.
Navrhněte strukturu sběru dat ze sítě pomocí Model-Driven Telemetry.
Vytvořte prototyp kolektoru naměřených dat získaných z reálných síťových prvků s využitím navržené struktury dat.
Otestujte vytvořený kolektor a porovnejte ho s aktuálně používaným řešením (založeným na SNMP) s ohledem na množství informací, které lze ze zařízení získat.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 24. června 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Monitorování provozu sítě pomocí Model-Driven Telemetry

Ladislav Loub

Katedra číslicového návrhu
Vedoucí práce: Ing. Tomáš Čejka, Ph.D.

6. ledna 2021

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářské práce Ing. Tomáši Čejkovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Dále bych chtěl poděkovat sdružení CESNET z. s. p. o. za poskytnutí technických prostředků a možnosti testování práce na produkční síti, což bylo velkým přínosem. Mé díky patří i mým rodičům a bratrovi za jejich podporu a trpělivost v průběhu mého vzdělávání.

Nakonec jedno speciální díky kolegovi Ing. Josefu Verichovi, jehož podpora a zájem mi velkou měrou pomohly v průběhu psaní práce. Je mi velice líto, že se výsledků této práce již nedožil.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. ledna 2021 . . . . . .

Odkaz na tuto práci

Loub, Ladislav. *Monitorování provozu sítě pomocí Model-Driven Telemetry.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Práce je zaměřená na problematiku stavového monitoringu síťových zařízení. Hlavním cílem je zhodnocení novějšího způsobu monitorování pomocí Model-Driven Telemetry v porovnání se starším způsobem založeným na protokolu SNMP. Výsledkem práce je testovací kolektor pro sběr, uložení a zobrazení sledovaných dat. Primárním zdrojem dat jsou zařízení společnosti Cisco.

Klíčová slova Monitorování sítě, Telemetrie, YANG modely, kolektor síťových dat

Abstract

The thesis is focusing on the issue of state monitoring of network devices. The main goal is to evaluate the newer method of monitoring using Model-Driven Telemetry in comparison with the older method based on the SNMP protocol. The result is a testing collector for collecting, storing and displaying monitored data. Cisco devices are the primary source of data.

Keywords Network monitoring, Model-Driven Telemetry, YANG models, monitoring collector

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Monitorování síťových prvků bylo nedílnou součástí provozu každé rozsáhlejší sítě od vzniku samotného konceptu propojení zařízení do jedné sítě. V prvopočátku toto sledování sloužilo primárně pro jednodušší dohledání problémů a bylo výrazně založeno na protokolu ICMP spíše, než na jednotném monitoringu. Nicméně s konstantním rozšiřováním sítí, které se začaly dostávat mimo akademickou oblast i do sféry komerční, bylo zapotřebí k monitoringu přistoupit jinou cestou. Už nebylo možné, aby síť spravovalo pouze pár zasvěcených jedinců, kteří ji vybudovali, ale bylo potřeba vytvořit rozhraní pro ostatní, pomocí kterého budou schopní sledovat síť sami. Jedním z první protokolů, který byl takto vytvořen a zadefinován v RFC 1028, byl protokol SGMP. SGMP vznikl v roce 1987, nicméně vzhledem k tomu, že byl navržen pouze jako přechodný, byl již po roce nahrazen protokolem SNMP.[1]

Protokol SNMP už bude známý mnohem většímu počtu lidí, neboť i přes to, že se jeho vznik datuje k roku 1988, kdy fungoval pouze pro sítě TCP/IP (bližší informace jsou v RFC 1067 [2]), dochoval se dodnes v revidované a vylepšené verzi SNMPv3. Bohužel i přes to, že tento protokol prošel od svého vzniku nemalými změnami, jsou na něm v dnešní době znát nedostatky z období jeho vzniku, které nelze jednoduše opravit.

Některé tyto nedostatky, především dostatečnou rozšiřitelnost pro sledování tisíců a více síťových zařízení, se snaží vyřešit nový přístup k monitorování sítě nazývaný telemetrie, anglicky také známá jako Streaming Telemetry (terminologie společnosti Cisco) nebo Model-Driven Telemetry.

Cíl práce

Hlavním tématem práce je analýza monitorování sítě pomocí Model-Driven Telemetry. Práce je zpracována v součinnosti se sdružením CESNET z. s. p. o. Monitorování stavových informací v současnosti probíhá pomocí protokolu SNMP ve verzích 2 a 3. Bohužel tento protokol v případě sběru velkého množ-

ství dat nadměrně vytěžuje síťové zařízení i průchodnost linek. Cílem této práce je proto zjistit, zda je možné tento problém vyřešit pomocí nového přístupu tzv. Model-Driven Telemetry. Mezi hlavní přínosy této práce zároveň patří sestavení testovacího kolektoru a návod pro použití pracovníky síťového oddělení.

Práce je rozdělena do šesti částí: První kapitola dává základní náhled do problematiky monitorování síťových zařízení. Zároveň se snaží objasnit pojem telemetrie, který se používá v hojném množství. Nakonec obsahuje popis, jak k této metodě monitorování přistupují jednotlivý výrobci zařízení.

Kapitola druhá provádí analýzu protokolu SNMP, popisuje možnosti a základní principy Model-Driven Telemetry. Závěr druhé části je zaměřen na datové modely YANG z pohledu jejich vlastností.

Ve třetí kapitole je proveden průzkum systémů, které jsou provozované sdružením CESNET. Následně obsahuje popis jejich možných variant pro výsledné řešení. Důraz je kladen na výběr databáze pro uchování dat a kolektoru pro sběr dat ze síťových zařízení. Hlavním kritériem výběru je kompatibilita s ostatními nástroji.

Čtvrtá kapitola se zaměřuje na samotný návrh testovacího řešení pro sběr dat. Je vypracováno mapování nejpoužívanějsích dat získaných pomocí SNMP na YANG modely. Zároveň obsahuje seznam modelů s popisem, které jsou vhodné pro rozšíření monitorování.

V páté kapitole se nachází rozbor realizace navrženého řešení. Zobrazuje zajímavé části konfigurace jednotlivých nástrojů společně s jednoduchým návodem na instalaci a provozování. Obsahuje konfiguraci pro síťové zařízení na základě navržených modelů.

V závěrečné šesté kapitole je provedeno testování navrženého řešení na základě případné použitelnosti v produkci, jeho vytěžování zdrojů síťových prvků a případnou rozšiřitelnost.

Kapitola 1

Monitorování síťových prvků

1.1 Co a proč monitorujeme

Monitoring je v dnešní době nedílnou součástí kterékoliv větší sítě. Na rozdíl od začátku provozování počítačových sítí, kde byl monitoring nasazován spíše podle potřeby, dnes musíme přistoupit k centrálnímu řešení. Samotná myšlenka nahrazení či vylepšení protokolu SNMP přitom také není nejmladší. Některé nedostatky byly diskutovány už okolo roku 2001 a částečně shrnuty byly v RFC 3535 [3]. Diskuze k němu vedoucí proběhly hlavně na shromážděních NANOG-22 v květnu 2001, RIPE-40 v říjnu 2001, LISA-XV v prosinci 2001 a IETF-52 taktéž v prosinci 2001. Důraz byl sice kladen spíše na konfigurovatelnost zařízení, nicméně dokument následně hovoří o obecném pojmu „network management“, do kterého monitorování rovněž spadá. Výsledky těchto diskuzí se nakonec projevily v protokolu NETCONF, jehož první oficiální verze byla zadefinována v RFC 4741, které vyšlo na konci roku 2006. Pojem telemetrie z pohledu monitorování sítě nemůžeme pevně asociovat s protokolem NETCONF, už jen z pouhého faktu, že z pohledu IETF telemetrie jako jednotný pojem není popsán v žádném RFC. Protokol NETCONF by měl být vnímán spíše jako inspirace při návrhu telemetrie. Společné znaky nalezneme přinejmenším v datových modelech YANG.

Potřeba monitorování sítě tedy není založena pouze na předpokladu, že by bylo vhodné síť sledovat, ale i z důvodu ušetření času inženýrům, kteří síť vybudovali. Velice brzy se ukázalo, že není možné zároveň sítě konfigurovat, plánovat a k tomu ještě neustále sledovat stejnou skupinou lidí. Postupně se tedy začaly vyvíjet různé monitorovací systémy. První monitorovací systémy byly většinou dodávány přímo od výrobce zařízení a byly tedy závislé na konkrétním hardwaru. Velký zlom nastal s příchodem TCP/IP schématu a především rozmachem internetu. Vývoj monitorování se v tu chvíli vydal směrem protokolu SNMP a čím dál více začal být kladen důraz na uživatelské prostředí a webové rozhraní nástroje. S tím začaly vznikat nástroje typu all-in-one, které teoreticky stačilo nasadit na server v síti, zadefinovat

zařízení, které má systém monitorovat a bylo téměř hotovo. Systém měl předdefinované, na které informace se monitorovaného zařízení bude dotazovat. Díky protokolu SNMP byly tyto informace stejné napříč různými zařízeními. Velký efekt na jednotnost dat mělo RFC 2863 [4], které zadefinovalo strukturu dat pro síťová rozhraní společně s doporučeními pro výrobce zařízení. Jedním z příkladů tohoto systému je HP Network Node Manager, který je stále používán ve sdružení CESNET. Dalšími z řady podobných nástrojů byly například Cisco Network Assistant, Nagios nebo Zabbix.

Na samotném typu systémů all-in-one samozřejmě nemusí být nutně něco špatného a je pochopitelné, že pro mnoho sítí to může být naprosto přijatelné řešení. Bohužel v dnešní době software zastarává poměrně rychlým tempem a tyto systémy tím, že zároveň umožňovaly i konfiguraci zařízení, byly poměrně velké a složité. Tím pádem byly i hůře aktualizovatelné. S každou aktualizací navíc souvisí riziko spojené s omezením možností monitorování starších zařízení. Pokud se jedná o aktualizaci softwaru například v telefonu nebo v počítači, tak omezení podpory starších zařízení je sice nepříjemné, ale lze se s ním vyrovnat poměrně rychle. V případě rozsáhlé sítě ovšem výměna rackového routeru nepřipadá moc v úvahu, pouze z důvodu nefunkčnosti komunikace mezi routerem a monitorovacím serverem. Mnoho síťových zařízení je navíc navrhováno na životnost více než deset let. Z pohledu aktualizace softwaru se jedná o poměrně dlouhou dobu a zajištění kompatibility pro všechny tyto zařízení je velmi složitý úkol. Posledním negativním faktorem all-in-one řešení bývají většinou poměrně vyšší náklady na pořízení.

Z toho důvodu se dnes monitoring zařízení začíná pomalu přesouvat do menších samostaných částí, které se jako celek spojí do funkčního monitorovacího nástroje. Tato řešení jsou dnes známá například jako produkty Elastic-Search Stack nebo InfluxData Stack. Jednotlivé možnosti rozeberu podrobněji v Kapitole 3. Velká výhoda těchto řešení je v jejich možnostech rozšiřitelnosti a nahraditelnosti. Správce už není limitován pouze jedním systémem, ale může si ho přizpůsobit. Tímto přístupem se tedy dá zajistit mnohem jednodušší zpětná kompatibilita.

Je potřeba si odpovědět na otázku, co vlastně chceme sledovat. Situace bohužel není tak jednoduchá, jak by se mohlo na první pohled zdát. Velice totiž záleží na tom, co je cílem. Každá síť je svým způsobem unikátní a má jiné požadavky. Koncová síť totiž bude mít na monitoring jiné požadavky než velká transportní síť. Z podledu sítě CESNET2 jsem identifikoval, že bude například potřeba monitorovat kromě sledování stavů síťových rozhraní i informace týkající se směrování a IP protokolů. Tedy velkou měrou monitoring protokolů BGP, OSPF, IPv4 a IPv6. Požadavků je více, ale ty budou podrobněji rozebrány v Sekci 4.5.

1.2 Pull vs. Push přístup

Monitorování sítě pomocí protokolu SNMP by se dalo nazvat jako „pull“ metoda. V principu se tato metoda zakládá na tom, že monitorovací systém, v případě SNMP nazvaný jako Manager, posílá po určitém intervalu dotaz routeru, v terminologii protokolu SNMP nazývanému Agent, na sbíraná data. Funguje to tak, že na Manageru nakonfigurujeme, aby si například každých 30 sekund obnovil stav síťových rozhraní na routeru. SNMP Manager to zařídí tím způsobem, že každých 30 sekund vygeneruje dotaz, který odešle na tento router. Jakmile router dotaz obdrží, tak ho zpracuje. Nejprve proběhne vyhodnocení hlaviček příchozího paketu. Poté provede dotaz do svojí interní databáze. Tento krok je většinou nejnáročnější na výpočetní prostředky. V článku [5] společnosti Cisco lze nalézt jak probíhá vyhodnocení SNMP dotazu na jejich zařízení. Následně router výsledky z interní databáze opět přetransformuje do požadovaného formátu a odešle zpět na SNMP Manager. Celým tímto procesem vzniká lehké zpoždění. Větším problémem ovšem je, že tento proces zbytečně zatěžuje procesor routeru. [1] Zároveň tímto způsobem mohou vznikat situace, kdy se problémy v síti nemusí projevit okamžitě a kdy může vyhodnocení SNMP dotazu nastat těsně po tom co některé rozhraní změní stav z UP na DOWN. V tu chvíli si bude monitorovací systém stále myslet, že je vše v pořádku po dobu, než odešle další dotaz. Tento způsob je zároveň nevhodný i k tomu, aby odhalil případné „flapování“, tedy stav, kdy se informace mění ze stavu UP na DOWN a obráceně v krátkém časovém intervalu.

Lepším přístupem je „push“ metoda. Nejprve na monitorovaném zařízení nastavíme, které informace chceme přenášet a v jakém časovém intervalu na kolektor. Klient následně, v našem případě například router z předchozího příkladu, jakmile vyprší zadaný interval, provede dotaz do interní databáze [2] a požadovaná data odešle monitorovacímu systému. Na první pohled se sice může zdát, že jediný krok, který oproti předchozí metodě zmizel, je zpracování dotazu od serveru a naopak jsme routeru přidali požadavek na neustálou kolekci dat. Router si ovšem dané údaje stejně uchovává ve své interní databázi a plnění této databáze probíhá na nižší úrovni než zpracování dotazů z monitorovacího systému. Největším zjednoduššením pro router je tedy, že ví, která data a v jakém intervalu odešle už v průběhu sběru dat narozdíl od „pull“ modelu, kde musí pokaždé reagovat na nově příchozí požadavek.

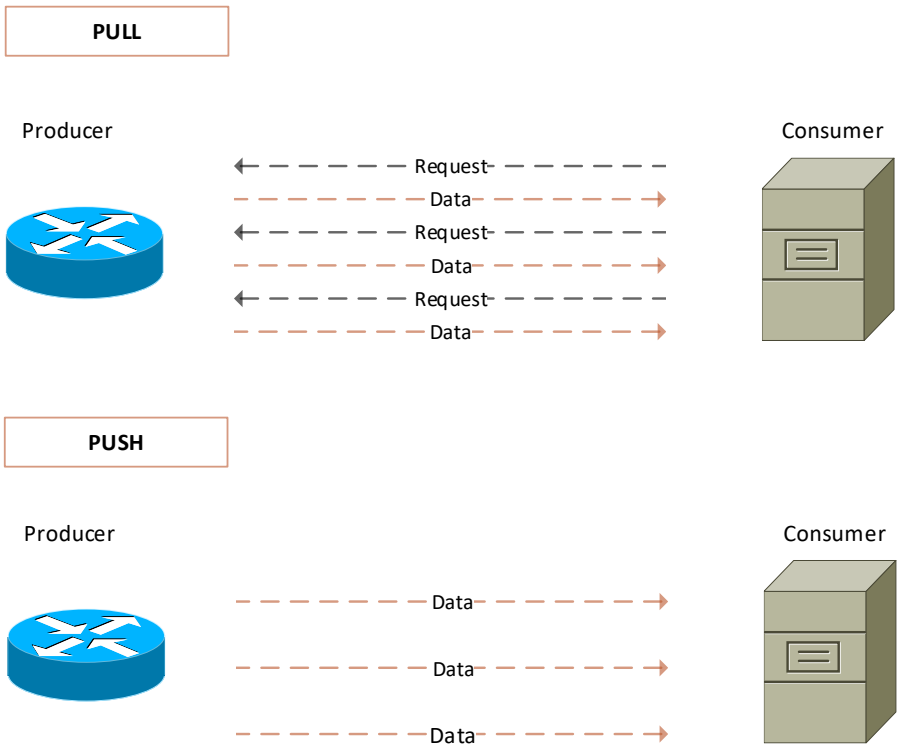
Protokol SNMP tuto metodu v omezené míře má také. Funkcionalita je známá jako SNMP Trap. Problémy s touto funkcionalitou jsou ovšem dva.

[1] V moderních routerech už sice procesor neobstarává klasické forwardování paketů (to probíhá na ASIC koprocesorech), ale jeho nadměrné vytížení ze strany monitorovacího systému stejně není žádoucí. Procesor se primárně stará o výměnu routovacích tabulek, což ho v případě globálního routování pomocí protokolu BGP může velice zatížit. Zároveň tím pádem může nastat situace, kdy je procesor natolik vytížený, že monitorovacímu systému na dotaz neodpoví.

[2] Krok záleží na konkrétní implementaci. Například v případě implementace telemetrie jsou data ukládána v průběhu intervalu a dotaz tedy neprobíhá.

Trap je nakonfigurován tak, aby reagoval na změnu, je tedy naprosto nevhodný pro některé účely. Například na sběr dat ohledně vytížení síťového rozhraní. Druhým problémem je, že pro transport používá protokol UDP, je tedy možnost, že se paket ztratí cestou a monitoring se nedozví [3], že bylo například síťové rozhraní ve stavu DOWN, dokud mu nepřijde zpráva, že je znovu UP, což může být poměrně zásadní problém. [6]

Způsob fungování modelů je znázorněn na diagramu 1.1.



Obrázek 1.1: Rozdíl modelů Pull a Push

1.3 Rozbor pojmu telemetrie

Tato práce se zabývá primárně objasněním pojmu telemetrie, v anglickém překladu známém jako Network Telemetry, Streaming Telemetry nebo také

[3]Tento nedostatek byl částečně nahrazen ve verzi SNMPv2, kdy byla přidána operace Inform, která se chová téměř stejně jako Trap, ale posílá zároveň potvrzení o doručení.

Model-Driven Telemetry. Termín telemetrie se v posledních letech poměrně rozšířil a používá se v hojném měřítku jako označení pro sledování dat na dálku. Tento častý výskyt způsobuje nepřesnosti a obtížnější vyhledávání. Situaci navíc nepomáhá ani skutečnost, že telemetrie referovaná v tomto dokumentu není zatím nijak standardizována. Z tohoto pohledu by mohlo stát za zmínku RFC 7923 [7], které vyšlo roce 2016. Pojednává o možnostech využití datových modelů YANG pro sběr dat, ale jedná se spíše o obecné doporučení, jakým způsobem mají být služby založené na tomto principu použité. V dokumentu se nevyskytují žádné implementační požadavky a není tam ani žádná konkrétní zmínka o telemetrii. Vzhledem k dataci RFC a jeho informačnímu charakteru by se dalo očekávat, že se jednalo o dokument, na jehož základě byla postupně vybudována implementace telemetrie, konkrétně využití datových modelů YANG.

Ve směru určité standardizace se prosazuje spíše pracovní skupina OpenConfig, která se zabývá různými existujícími YANG modely a zároveň vyvíjí unifikované rozhraní gNMI. Jedná se o skupinu poskytovatelů služeb, kteří se společnými silami snaží sjednotit koncept telemetrie. V této práci bohužel nebylo možné rozhraní gNMI použít kvůli chybějící podpoře v testovaných dostupných zařízeních, avšak je zmíněno pro kompletnost tohoto přehledu.

Nejblíže k definování pojmu telemetrie má v tuto chvíli asi draft „Network Telemetry Framework“ [8], který prošel pátou revizí v říjnu 2020. Kromě Model-Driven Telemetry, která je hlavním tématem této práce se ovšem zabývá i dalšími principy monitorování sítě.

Pod pojmem telemetrie (Model-Driven Telemetry) je tedy asi nejjednodušší si představit sadu nástrojů a protokolů sloužících pro sběr a zpracování dat, dodržujících určité principy. Těmito principy jsou především využití „Push“ modelu, popsaného v předchozí Kapitole 1.2, a YANG modely pro popis struktury odesílaných dat. Nasazení telemetrie se udává v několika krocích, kdy posledním krokem by měla být plná automatizace, při které si monitorovací systém sám konfiguruje síťová zařízení, na základě nasbíraných dat.

1.4 Odlišné přístupy výrobců zařízení

Vzhledem k tomu, že se stále jedná o poměrně novou technologii, mají výrobci zařízení relativní volnost ve svém přístupu k ní. Dobrou zprávou pro správce sítí je fakt, že se výrobci při implementaci telemetrie do svých zařízení, drží principů ve výše zmíněných dokumentech. Zároveň vznikají YANG modely přímo od IETF, které ve většině zařízení bývají implementovány a jsou případně rozšířeny o modely přímo od výrobce. Drobné rozdíly lze pozorovat například v použití transportního protokolu, kdy lze vidět protokoly gRPC, UDP nebo TCP. V čem se ovšem rozdíly mezi výrobci projevují mnohem zásadnějším vlivem, je množství a způsob implementovaných YANG modelů. Zatímco základní modely od IETF nebo OpenConfig má naimplementované

většina firem, podpora rozšiřujících modelů je v některých případech omezená. Zároveň je potřeba si pohlídat schopnosti daného hardware, kdy některé starší architektury telemetrii nepodporují ani po aktualizaci softwaru.

Kapitola 2

Analýza

2.1 Protokol SNMP

SNMP je protokol sloužící pro sběr dat ze zařízení v síti. Spadá do aplikační vrstvy podle modelu TCP/IP. První verze tohoto protokolu byla oficiálně zadefinována v RFC 1155.

Základními komponentami jsou:

- Spravované zařízení
- Agent
- Správcovské zařízení

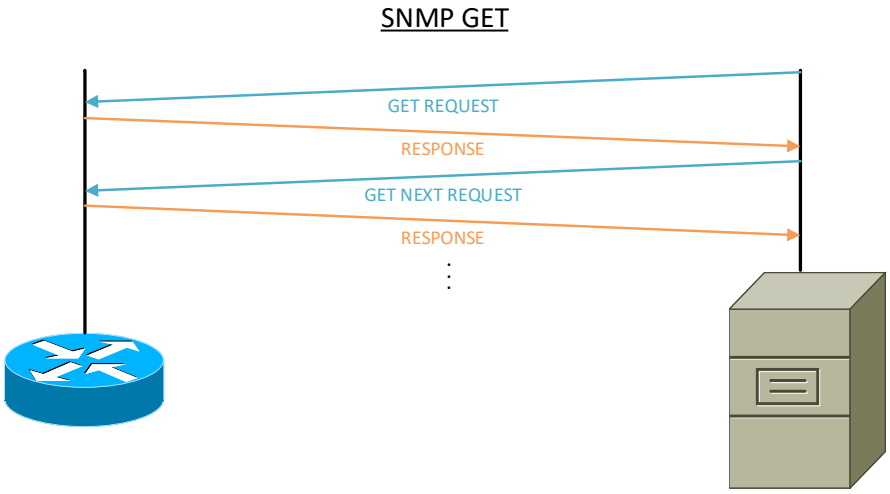
Spravované zařízení je zařízení, ze kterého chceme získat data. V případě monitorování sítě se tedy jedná primárně o switche a routery.

Agent je software, který je spuštěný na spravovaném zařízení. Jeho úkolem je vyhodnocení a reakce na požadavky ze správcovského zařízení.

Správcovské zařízení odesílá požadavky na spravované zařízení a získává z nich tímto způsobem potřebná data. V anglické terminologii označované také jako SNMP Manager nebo Network Management System.

Důležitou součástí protkolu SNMP je Management information base (MIB). Každý SNMP Agent si udržuje databázi popisující sledovaná data zařízení. SNMP Manager následně tuto databázi využije k překladu požadavků pro konkrétní zařízení. Díky tomu jsou data získávaná pomocí SNMP strukturovaná a stejná přes mnoho zařízení.

Správcovské zařízení využívá pro komunikaci s Agentem několik jednoduchých operací. Jedná se o operace GET, GET NEXT, GET BULK, SET, TRAP, INFORM a RESPONSE. Jednoduchost těchto operací přispěla k oblibě tohoto protokolu, nicméně ona jednoduchost začíná dnes působit problémy. Například hojně využívaná operace GET BULK způsobuje větší výpočetní náročnost pro síťové zařízení. Systém periodicky posílá dotaz a router mu odpovídá. Průběh operace je znázorněn na obrázku 2.1.

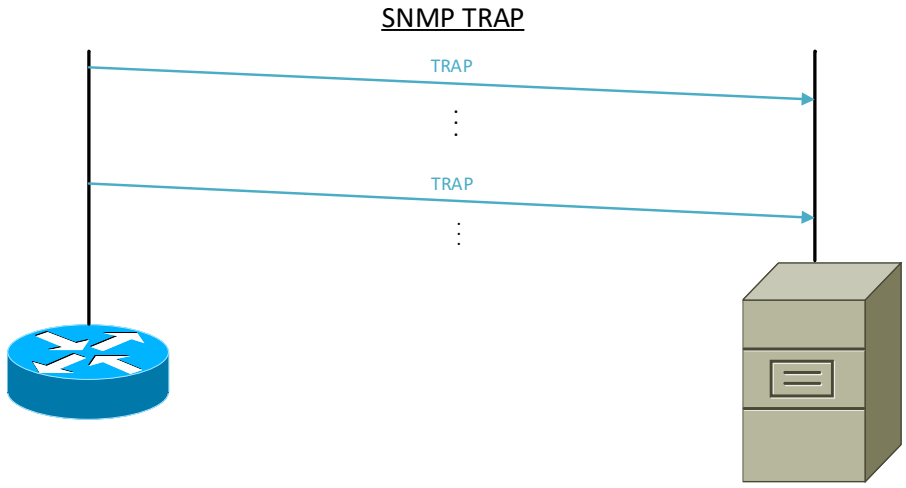


Obrázek 2.1: Operace SNMP GET

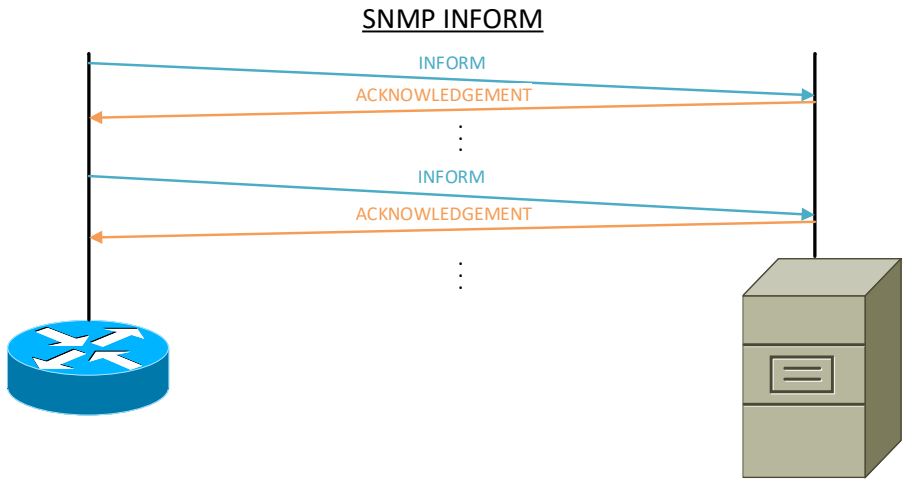
Operace TRAP by se dala zařadit do modelu „Push“ podle Sekce 1.2. Jakmile na zařízení nastane událost, která je pomocí SNMP sledována, proces SNMP na routeru zareaguje odesláním zprávy TRAP na monitorovací systém. Podstatnou nevýhodou ovšem je, že data jsou transportována pomocí protokolu UDP na portu 162 a je tedy riziko, že zpráva nemusí dorazit. Diagram operace TRAP je na obrázku 2.2.

INFORM je vylepšením operace TRAP. Přidává ověření přijetí paketu, není ovšem vhodná pro data, která se velmi často mění. Komunikace v průběhu operace INFORM je zobrazena v obrázku 2.3. [9]

Protokol má v dnešní době z pohledu monitorování sítě stále dominantní postavení, především ve své poslední verzi SNMPv3. Rozdíl mezi první a druhou verzí byl méně zásadní a přinesl spíše restrukturalizaci některých částí a přidání operace INFORM. Třetí verze přinesla změnu mnohem zásadnější, když umožnila provádět autentizaci zařízení a zároveň šifrování zpráv posílaných mezi Agentem a Managerem. [10]



Obrázek 2.2: Operace SNMP TRAP



Obrázek 2.3: Operace SNMP INFORM

2.2 Rozdílný přístup telemetrie

Nejzásadnější rozdíl telemetrie oproti protokolu SNMP se nachází v samotném způsobu sběru dat. Telemetrie je postavená na modelu „Push“, který jí dává nemalou výhodu oproti SNMP. Druhým rozdílem je struktura dat, která jsou posílána. Zde je situace taková, že nelze jednoznačně říci, který přístup je lepší. Jak datové modely YANG, tak MIB v SNMP jsou strukturované a základní struktura je zadefinovaná pomocí různých RFC [4]. Zde by možná situace mohla být i lehce nakloněná k SNMP z důvodu delší historie a vyladění drobností. Na druhou stranu MIB občas postrádají implementaci některých dat, která jsou už v rámci YANG modelů zadefinovaná a implementovaná v síťových zařízeních. Jedním takovým případem jsou optická data z transceiverů, která lze získat pouze pomocí telemetrie[5].

Mezi další rozdíly je potřeba uvést i kódování dat, které může hrát také nemalou roli. Protokol SNMP byl navržený velice jednoduše. Samotná data jako taková jsou tedy sebrána a jako plain text odeslána. Tento způsob je jednoduchý a vhodný pro zpracování a zároveň pro řešení případných problémů při přenosu, ale má značnou nevýhodu pokud je posílaných dat mnoho. Telemetrie s touto variantou počítá a je tedy možné zvolit různé kódovaní dat, podle aktuální potřeby. Podrobnější informace ke kódování lze nalézt v Sekci 2.6.

V tabulce 2.1 lze nalézt porovnání různých způsobů monitorování.

Tabulka 2.1: Porovnání přístupů monitorování (Zdroj [11])

Typ	Metoda	Přesnost	Struktura dat
Telemetrie	Push	milisekundy	pomocí YANG modelů
SNMP Trap	Push	sekundy	pomocí MIB
SNMP Get	Pull	milisekundy	pomocí MIB
Syslog	Push	sekundy	nestrukturované
CLI	Pull	minuty	nestrukturované

2.3 Událostmi řízená telemetrie

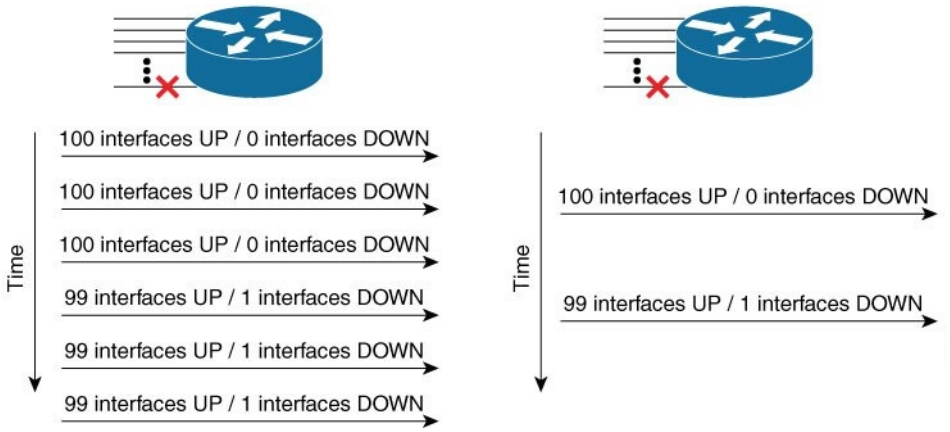
Událostmi řízená telemetrie neboli „Event-Driven Telemetry“ je podkategorií pojmu Model-Driven Telemetry. Tímto způsobem je označována konfigurace telemetrie, která se chová podobným způsobem jako operace SNMP Trap a Inform. Jedná se o způsob sběru dat, která se odesílájí pouze při změně místo

[4]Informační databáze pro SNMP je standardizována v mnoha RFC a prakticky kopíruje aktualizace samotného protkolu SNMP. Prvním RFC, které se MIB týkalo, bylo RFC 1155. Nejnovější aktualizací je RFC 4293. Podrobný seznam RFC dokumentů, které se týkají MIB, lze nalézt na adrese `https://en.wikipedia.org/wiki/Management_information_base`.

[5]Bohužel pouze z některých zařízení. Při testování se vyskytl problém, kdy router sice YANG model rozpoznal a operační systém by data poslal, ale neměl žádná data z hardwaru, která by mohl poslat.

periodického přenášení. Toto nastavení se hodí především u dat, kde je změna méně častá, ale rychlost jakou se k nám data dostanou je zásadní. Typickým příkladem je stav síťového rozhraní na routeru. Není potřeba, aby nám router posílal například každou minutu údaj, že všechny rozhraní jsou ve stavu OK, ale potřebujeme, aby nám dal vědět okamžitě, jakmile jeden přejde do stavu DOWN.

Konfigurace se téměř neliší od klasické konfigurace telemetrie. Jediný rozdíl je v nastavení časového okna, ve kterém má zařízení údaje sbírat na 0 sekund. V tu chvíli zařízení přejde do stavu, kdy posílá pouze změny. Graficky je tento proces znázorněn na obrázku 2.4. Diagram vpravo zobrazuje posílání dat při změně. Vlevo je vidět případ, kdy jsou data posílána neustále po určitém intervalu. Tato varianta je označovaná jako „Streaming Telemetry“ nebo také „Cadence Telemetry“ podle článku [12] na stránkách společnosti Cisco.



Obrázek 2.4: Event-Driven telemetry (Zdroj [13])

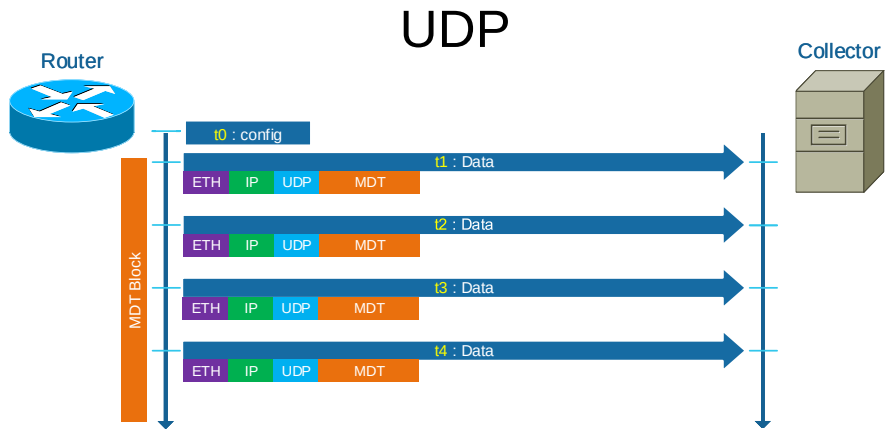
2.4 Transportní vrstva

Získaná data lze ze síťových zařízení na server přenést mnoha způsoby a v zásadě se situace může lišit výrobce od výrobce. V průběhu vývoje telemetrie lze pozorovat neustálý posun v transportních protokolech. Na začátku byl implementován transport pomocí UDP a TCP. Poté došlo k využití frameworku gRPC vyvíjeného společností Google. V současné chvíli se pravděpodobně nejvíce používají možnosti transportu pomocí TCP a gRPC. Prozatimní vývoj naznačuje, že další progres by mohl být ve směru rozhraní gNMI. [6]

[6]Rozhraní gNMI je jakýmsi sjednocením v použití frameworku gRPC. Použití gRPC se může u různých výrobců lehce lišit a může vést ke komplikacím. Řešením by tedy mělo být jednotné rozhraní nazvané gNMI vyvíjené pracovní skupinou OpenConfig. Více o gNMI se lze dozvědět v dokumentaci na GitHubu [14] a v komunitním článku [15].

2.4.1 UDP

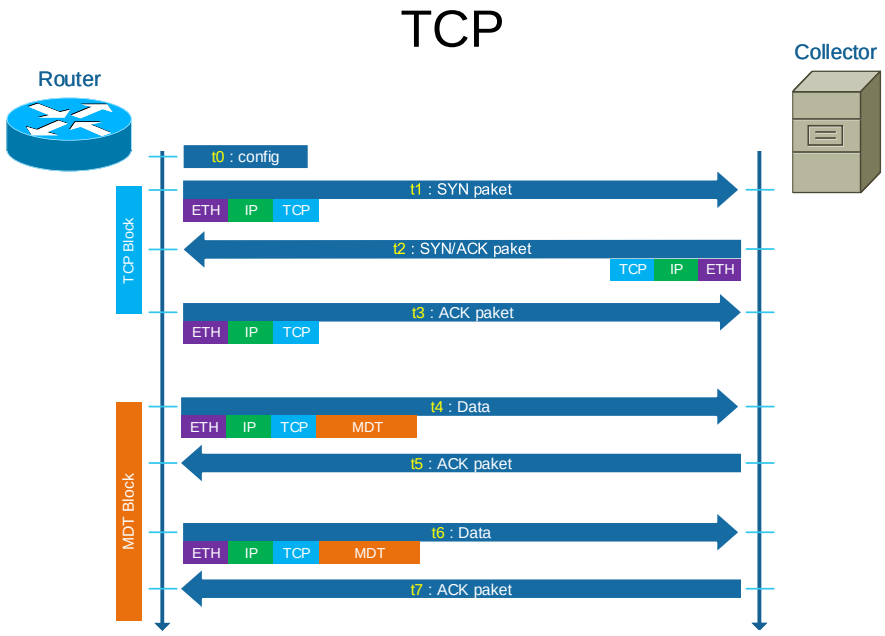
Nejjednodušší přenos dat je určitě pomocí protokolu UDP. Router začne data posílat v určitých intervalech okamžitě po konfiguraci zařízení. Schéma přenosu lze vidět na obrázku 2.5. Toto řešení samozřejmě trpí stejnými problémy jako operace SNMP Trap, kdy není jisté, jestli informace k cíli dorazila. Protokol UDP se tedy téměř nepoužívá.



Obrázek 2.5: Transport pomocí protokolu UDP

2.4.2 TCP

O něco lepším řešením je využít protokol TCP. Průběh komunikace je zobrazen na obrázku 2.6. Po úvodním navázání spojení (tzv. „TCP handshake“) se začínají data posílat v předem nakonfigurovaném intervalu. Ke každému přijatému paketu na straně kolektoru je vygenerována odpověď ACK. Komunikace nasbíraná pomocí programu Wireshark je vidět na obrázku 2.7. Ve žlutě označených paketech se posílají samotná aplikační data ve formátu JSON. Přednastavený interval posílání dat byl 30 sekund. Použití čistého TCP protokolu je spolehlivější varianta než pouze UDP, ale stále nalezneme některé nedostatky. Hlavním nedostatkem je absence bezpečnostních mechanismů pro dosažení důvěrnosti a integrity dat a ověření odesílatele a příjemce. Data nejsou šifrovaná a nemůže tedy dojít ani k autentizaci jednotlivých zařízení.



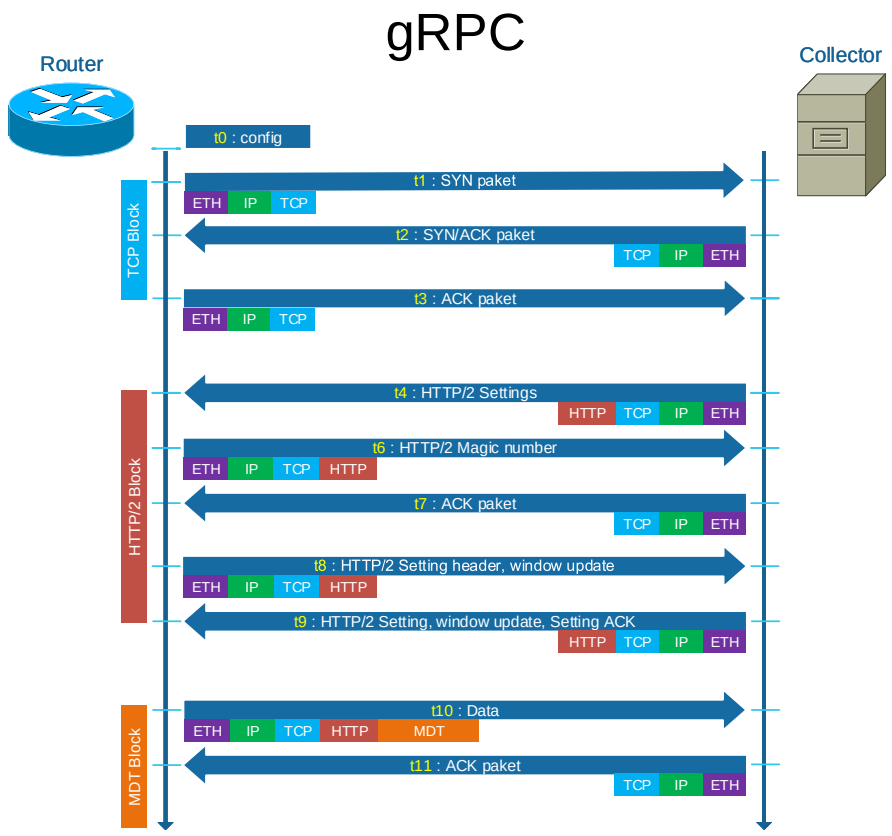
Obrázek 2.6: Transport pomocí protokolu TCP

Time	Source	Destination	Protocol	Info
0.000000	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [SYN] Seq=0
0.000112	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [SYN, ACK]
0.000492	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=1
0.622135	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=1
0.622201	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
0.622624	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=1
0.622632	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
0.622641	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=2
0.622648	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
0.628164	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [PSH, ACK]
0.628206	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
31.261140	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=6
31.261192	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
31.261215	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=6
31.261223	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
31.261565	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=6
31.261574	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
31.261683	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=6
31.261690	78.128.211.188	195.113.[redacted]	TCP	57000 → 59365 [ACK] Seq=1
31.261696	195.113.[redacted]	78.128.211.188	TCP	59365 → 57000 [ACK] Seq=6

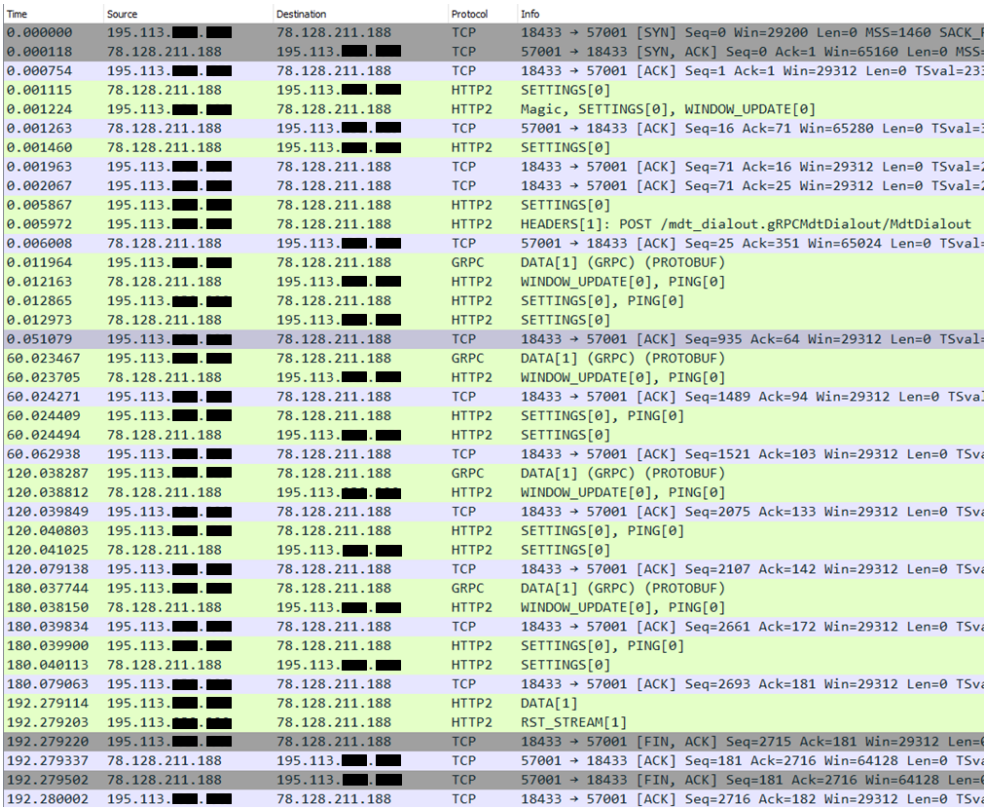
Obrázek 2.7: Transport pomocí protokolu TCP - Wireshark

2.4.3 gRPC

Framework gRPC je open source nadstavbou protokolu RPC (Remote Procedure Calls), která je vyvíjena společností Google. Na čtvrté vrstvě ISO/OSI modelu se jedná o TCP spojení. Nad ním ovšem probíhá protokol HTTP/2, který nám pomáhá výše zmíněné problémy TCP vyřešit. Případné použití protokolu TLS nám povoluje autentizovat zdroje dat, čímž můžeme potlačit případné nebezbečí zasílání dat do kolektoru útočníkem. Průběh komunikace lze vidět v diagramu 2.8. Nejprve proběhne navázání TCP spojení. Následně proběhne nastavení protokolu HTTP/2 a nakonec již jsou posílána samotná data. Průběh nasbíraný z reálných prvků a analyzovaný nástrojem Wireshark lze pozorovat na obrázku 2.9.



Obrázek 2.8: Transport pomocí protokolu gRPC



Time	Source	Destination	Protocol	Info
0.000000	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_
0.000118	78.128.211.188	195.113.[redacted]	TCP	57001 → 18433 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=
0.000754	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=23
0.001115	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
0.001224	195.113.[redacted]	78.128.211.188	HTTP2	Magic, SETTINGS[0], WINDOW_UPDATE[0]
0.001263	78.128.211.188	195.113.[redacted]	TCP	57001 → 18433 [ACK] Seq=16 Ack=71 Win=65280 Len=0 TSval=
0.001460	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
0.001963	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=71 Ack=16 Win=29312 Len=0 TSval=
0.002067	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=71 Ack=25 Win=29312 Len=0 TSval=
0.005867	195.113.[redacted]	78.128.211.188	HTTP2	SETTINGS[0]
0.005972	195.113.[redacted]	78.128.211.188	HTTP2	HEADERS[1]: POST /mdt_dialout.gRPCMdtDialout/MdtDialout
0.006008	78.128.211.188	195.113.[redacted]	TCP	57001 → 18433 [ACK] Seq=25 Ack=351 Win=65024 Len=0 TSval=
0.011964	195.113.[redacted]	78.128.211.188	GRPC	DATA[1] (GRPC) (PROTOBUF)
0.012163	78.128.211.188	195.113.[redacted]	HTTP2	WINDOW_UPDATE[0], PING[0]
0.012865	195.113.[redacted]	78.128.211.188	HTTP2	SETTINGS[0], PING[0]
0.012973	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
0.051079	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=935 Ack=64 Win=29312 Len=0 TSval=
60.023467	195.113.[redacted]	78.128.211.188	GRPC	DATA[1] (GRPC) (PROTOBUF)
60.023705	78.128.211.188	195.113.[redacted]	HTTP2	WINDOW_UPDATE[0], PING[0]
60.024271	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=1489 Ack=94 Win=29312 Len=0 TSval
60.024409	195.113.[redacted]	78.128.211.188	HTTP2	SETTINGS[0], PING[0]
60.024494	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
60.062938	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=1521 Ack=103 Win=29312 Len=0 TSva
120.038287	195.113.[redacted]	78.128.211.188	GRPC	DATA[1] (GRPC) (PROTOBUF)
120.038812	78.128.211.188	195.113.[redacted]	HTTP2	WINDOW_UPDATE[0], PING[0]
120.039849	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=2075 Ack=133 Win=29312 Len=0 TSva
120.040803	195.113.[redacted]	78.128.211.188	HTTP2	SETTINGS[0], PING[0]
120.041025	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
120.079138	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=2107 Ack=142 Win=29312 Len=0 TSva
180.037744	195.113.[redacted]	78.128.211.188	GRPC	DATA[1] (GRPC) (PROTOBUF)
180.038150	78.128.211.188	195.113.[redacted]	HTTP2	WINDOW_UPDATE[0], PING[0]
180.039834	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=2661 Ack=172 Win=29312 Len=0 TSva
180.039900	195.113.[redacted]	78.128.211.188	HTTP2	SETTINGS[0], PING[0]
180.040113	78.128.211.188	195.113.[redacted]	HTTP2	SETTINGS[0]
180.079063	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=2693 Ack=181 Win=29312 Len=0 TSva
192.279114	195.113.[redacted]	78.128.211.188	HTTP2	DATA[1]
192.279203	195.113.[redacted]	78.128.211.188	HTTP2	RST_STREAM[1]
192.279220	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [FIN, ACK] Seq=2715 Ack=181 Win=29312 Len=
192.279337	78.128.211.188	195.113.[redacted]	TCP	57001 → 18433 [ACK] Seq=181 Ack=2716 Win=64128 Len=0 TSva
192.279502	78.128.211.188	195.113.[redacted]	TCP	57001 → 18433 [FIN, ACK] Seq=181 Ack=2716 Win=64128 Len=
192.280002	195.113.[redacted]	78.128.211.188	TCP	18433 → 57001 [ACK] Seq=2716 Ack=182 Win=29312 Len=0 TSva

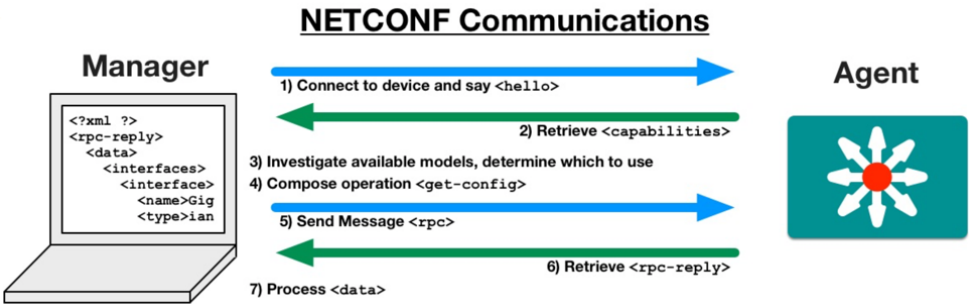
Obrázek 2.9: Transport pomocí protokolu gRPC - Wireshark

2.4.4 NETCONF

Posledním zmíněným a uvažovaným transportním protokolem je protokol NETCONF a jeho rozšíření založené na REST API — RESTCONF. Protokol byl vytvořen za účelem zjednodušení automatické konfigurace síťových zařízení. Využívá YANG modely pro určení jednoznačné struktury dat. Protokol ovšem neslouží primárně ke sledování zařízení, nýbrž k jejich konfiguraci. Proč se tedy tímto protokolem vůbec zabývat? Odpověď je jednoduchá. Přestože cílem je monitorování pomocí telemetrie,[7] je potřeba se podívat na společné vlastnosti. Nejdůležitějším společným prvkem jsou YANG modely. Pokud se podíváme na jednotlivé modely, které používáme na zařízeních, tak zjistíme, že velká část z nich je zdvojená. Nalezneme tedy jednu variantu, která má v názvu „rw“ (read-write) a slouží tedy ke konfiguraci zařízení. Druhá varianta „ro“ (read-only) pak slouží právě k monitorování. Pokud bychom si označili

[7]Prokol NETCONF teoreticky lze použít i pro získání stavových informací ze zařízení. Nepodařilo se mi ovšem dohledat žádného výrobce, který by měl funkcionalitu telemetrie naimplementovanou tímto způsobem. NETCONF má mnohem častější použití v případě telemetrie, pro zjištění jednotlivých dostupných YANG modelů na zařízení.

konfiguraci pomocí YANG modelů aplikovaných protokolem NETCONF jako „Model-Driven Configuration“, tak pojem, kterým se zabývám v této práci „Model-Driven Telemetry“, bude jeho opakem. Protokol NETCONF se společně s telemetrií navzájem doplňují a je možné s jejich pomocí nich tvořit automatizované systémy a mechanizmy v síti mnohem jednodušeji. Komunikace protokolu NETCONF je zachycena na obrázku 2.10. [8]



Obrázek 2.10: Komunikace protokolu NETCONF (Zdroj [16])

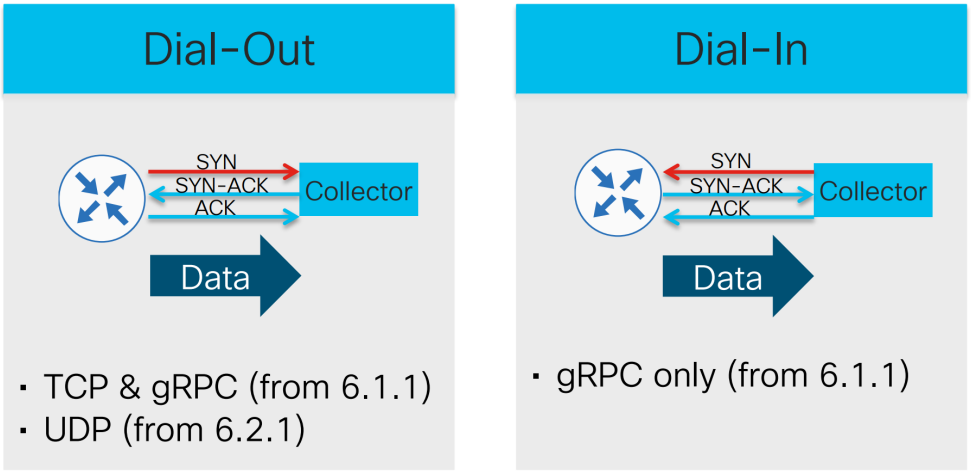
2.5 Způsob navázání spojení

Tato sekce se týka především zařízení Cisco. Komunikace mezi kolektorem a síťovým zařízením může být iniciována dvojím způsobem.

Prvním způsobem je navázání spojení ze strany routeru na kolektor. Toto řešení se označuje jako „Dial-Out“. Nejprve aplikujeme konfiguraci na routeru. V rámci této konfigurace musíme zadat cílový kolektor. Router následně navazuje spojení na kolektor, podle zadané adresy.

Druhý způsob funguje opačně. Spojení se navazuje ze strany kolektoru na router. Tato varianta funguje pouze při použití gRPC, jak lze vidět na obrázku 2.11. Varianta se nazývá „Dial-In“. Průběh nastavení probíhá obdobně jako u první varianty. Na zařízení musíme nastavit modely, které budeme používat. Změnou ovšem je, že na router nezadáváme adresu kolektoru, ale pouze zpřístupňujeme nějaký nevyužitý port pro navázání spojení ze strany kolektoru.

[8] Na obrázku lze teoreticky zpozorovat i jednu nevýhodu protokolu. Protokol data kóduje do formátu XML, který nemusí být úplně vhodný pro konstatní zasílání dat po síti.



Obrázek 2.11: Rozdíl mezi „Dial-Out“ a „Dial-In“ (Zdroj [17])

2.6 Kódování dat

Výběr kódování samotných dat, která se sbírají, také hraje nemalou roli. V rámci analýzy se mi podařilo dohledat použití pouze dvou kódování, která se používají k přenosu po médiu. Jedná se o Google Protocol Buffers (GPB) a JavaScript Object Notation (JSON). Kódování dat hraje roli především v rámci vytížení samotných linek. Z tohoto pohledu je při srovnání formát JSON méně úsporný a velikost poslaných dat je větší. To je na druhou stranu kompenzováno snazší čitelností, může se tedy hodit při úvodním řešení problémů. Telemetrie ovšem není navržená tak, aby byla vhodná pro člověka, nýbrž pro stroj. Proto je mnohem vhodnější využít Google Protocol Buffers.

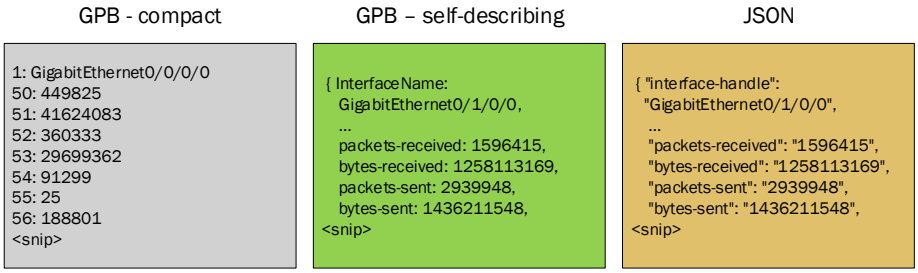
GPB lze rozdělit na dvě kategorie. Compact a Key-Value.

- Key-Value formát se skládá z dvojice klíč a hodnota. Klíč má textovou formu. Hodnota bývá většinou číselná, ale může být i textová. Tato varianta není o tolik výhodnější než formát JSON. Označuje se také jako „Self-Describing“, tedy popisující sama sebe. Označení vychází z toho, že klíče v názvu obsahují informaci, čeho se týkají a není tedy potřeba překlad.

- Compact formát se také skládá z klíče a hodnoty, nicméně klíč není typu string, ale jedná se o datový typ integer. Zde už je tedy potřeba překlad číselné hodnoty, aby bylo možné rozeznat, o kterou hodnotu se jedná. Kompaktní formát také používá kódování nazvané Varints. V zásadě se jedná o způsob, jak datový typ integer, který má například délku 32 bitů, poslat po fyzickém médiu v kratší bitové délce. Díky tomu jsou odeslaná data výrazně kratší (téměř 8 krát kratší TCP stream v průběhu

testování), ale na druhou stranu je tento formát nevhodný pro hledání případných problémů. [9]

Rozdíly mezi kódováním lze vidět na obrázku 2.12.



Obrázek 2.12: Rozdíly kódování

Výběr vhodného kódování je důležitá část. Na úvodní nastavení a testování by se dalo doporučit použít kódování JSON. Jakmile bude testovací fáze hotová, změnit na Compact GPB. Zásadní ovšem je, udělat si i rozbor vhodných kolektorů, protože ne všechny podporují například kompaktní GPB kódování.

2.7 Datové modely YANG

Yet Another Next Generation (YANG) je jazyk datových modelů určených pro konfiguraci a monitorování síťových zařízení. Data jsou organizována do stromové struktury. YANG je vyvíjen společně s protokolem NETCONF. YANG modely byly zadefinovány v RFC 6020. Na každém zařízení, kde chceme provozovat telemetrii potřebujeme soubory, které obsahují popis dat nazývané YANG moduly. Jedním příkladem může být YANG modul Cisco-IOS-XR-pfi-im-cmd-oper na ukázce 2.1. Moduly nemusí mít jednotnou strukturu napříč výrobci zařízení. O sjednocení modelů se snaží jak IETF, tak OpenConfig. V příkladu níže se ovšem jedná o modul, který je vydaný společností Cisco a nelze ho tedy použít na zařízení jiného výrobce.

Modul obsahuje jak informace pro uživatele těchto modulů (sekce ohledně organizace, kontakt nebo revize), tak pro síťové zařízení, které podle nich určí data pro zaslání (sekce import, include a container).

[9] Kódování pomocí Varints se používá i ve variantě Key-Value, ale pouze u číselných hodnot. Podrobnější informace ke kódování lze nalézt v dokumentaci Google Proto bufferů [18].

2.1: YANG modul Cisco-IOS-XR-pfi-im-cmd-oper

```
module Cisco-IOS-XR-pfi-im-cmd-oper {
  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-pfi-im-cmd-
      oper";
  prefix pfi-im-cmd-oper;

  import Cisco-IOS-XR-types {
    prefix xr;
  }
  include Cisco-IOS-XR-pfi-im-cmd-oper-sub2 {
    revision-date 2017-06-26;
  }
  ...
  organization
    "Cisco Systems, Inc.";
  contact
    ...
  description
   ...
  revision 2017-06-26 {
    description
      "Change identifiers to be more readable.";

   ...

  container interfaces {
    config false;
    description
      "Interface operational data";
    container interface-xr {
      list interface {
        key "interface-name";
        leaf interface-name {
          type xr:Interface-name;
        }
      }
    }
    ...
  }
}
```

Kapitola 3

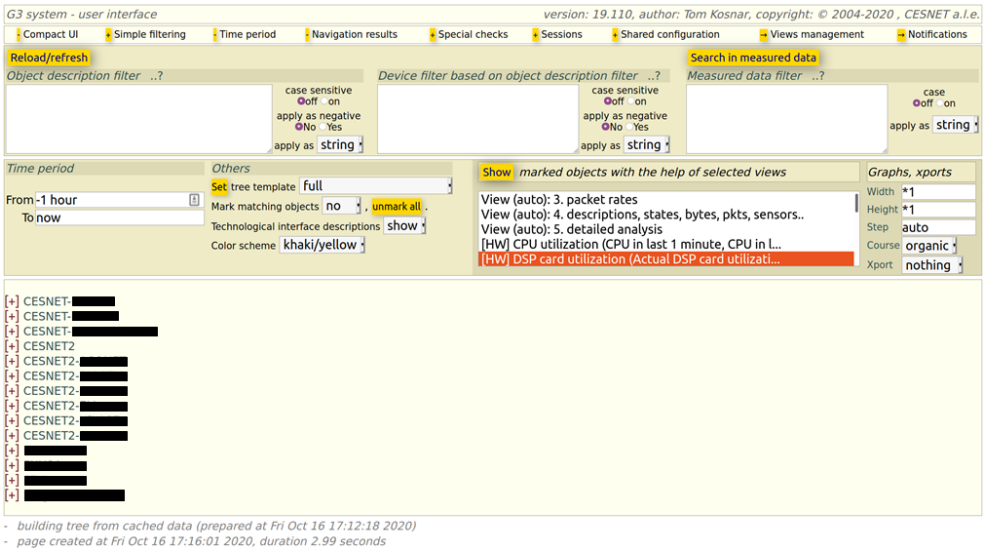
Existující nástroje

3.1 Současné řešení používané sdružením CESNET

Pro stavové monitorování se v rámci sítě CESNET2 používají dva nástroje G3 a HP Network Node Manager. Oba tyto nástroje využívají protokol SNMP pro sběr dat.

G3 je software vyvíjený interně v rámci sdružení CESNET. Funguje na protokolu SNMPv3. Primární využití je zpětné dohledávání problémů a dodatečných informací k nim. Systém není vhodný k soustavnému monitorování sítě pro účely řešení výpadků. Nástroj G3 data získává pouze metodou pull každých 40 až 600 sekund. Využívá se dotazování na stroje v tomto náhodném intervalu, aby se rozprostřelo vytížení monitorovacího systému i síťových zařízení. Systém by tedy mohl v případě nasazení telemetrie benefitovat především kratší dobou sběru dat a nebyla by nutnost implementovat složité náhodné dotazování. Dalším faktorem by mohla být menší zátěž síťových prvků a širší spektrum sbíraných dat. Ukázky webového rozhraní G3 lze vidět na obrázcích 3.1 a 3.2.

HP Network Node Manager je monitorovací systém vyvinutý společností Hewlett Packard. V CESNETu slouží jako nástroj pro pracoviště stálé služby, které je díky němu schopné zaznamenat téměř okamžitě výpadky v síti. V současné době se využívá protokol SNMPv2 s možností přechodu na SNMPv3. Jedná se o poměrně robustní systém, který sbírá mnoho dat ze síťových prvků, ale využívá pouze metodu pull, stejně jako G3. Výhodou je schopnost vykreslit částečně topologii sítě na základě získaných dat. Mezi nevýhody by se určitě zařadilo zbytečně velké množství dat, která nejsou tolik podstatná. Navíc pro hlubší analýzu problémů slouží systém G3. Dále vzhledem k tomu, že se jedná o kompletní řešení od společnosti HP, není možné v softwaru provádět jakékoliv úpravy. Další nevýhodou tohoto řešení je komerční licence, kterou je nutné nakoupit. Ukázku uživatelského rozhraní ze systému lze vidět na obrázku 3.3.



Obrázek 3.1: G3 úvodní pohled

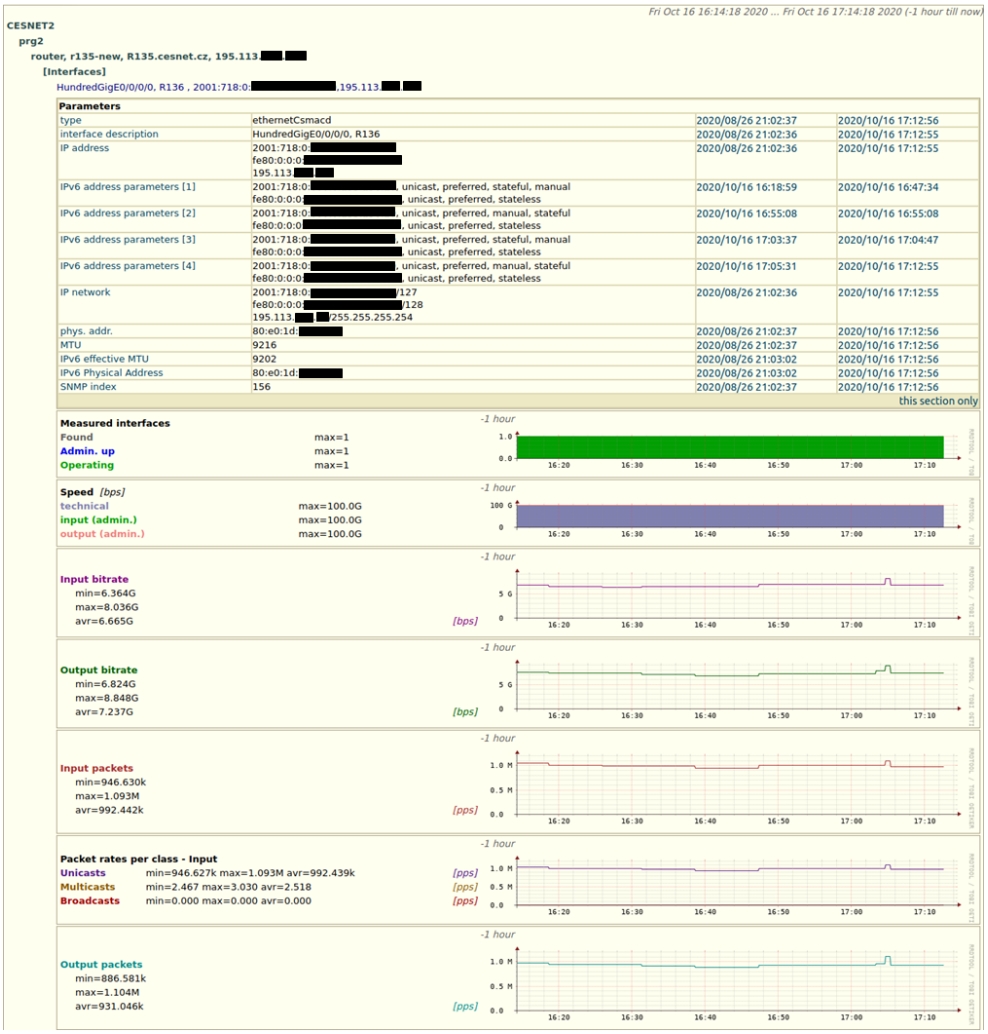
Idea monitorování pomocí telemetrie vzešla z vyšší zátěže některých páteřních síťových zařízení a drobných problémů s protokolem SNMP. Každou možnost, která tyto problémy může mitigovat, je potřeba prozkoumat. Systém HP Network Node Manager bohužel nelze překonfigurovat, aby byl schopen pracovat s daty získanými pomocí telemetrie a je tedy vysoká pravěpodobnost jeho náhrady v případě potvrzení výhod telemetrie oproti SNMP. Systém G3 tím, že je vyvíjen sdružením, umožňuje úpravy a je tedy v plánu případně systém přepsat, kvůli přidání podpory sběru dat i pomocí telemetrie.

3.2 Možné databázové systémy

V rámci testovacího systému je potřeba využít databázi pro uložení nasbíraných dat. Pro účely uložení těchto dat lze využít téměř libovolnou databázi. Nicméně vzhledem k tomu, že se jedná o časové údaje sbírané periodicky, bude vhodné použít některou databázi optimalizovanou pro tyto účely (Time series database — TSDB).

Časovou databázi si lze asi nejlépe představit jako sérii stejných dat, měřených v určitém časovém intervalu, uložených v časové posloupnosti. Vlastnosti těchto databází staví na 3 předpokladech, vztahujících se k případům užití:

- Příchozí data téměř vždy vytvoří nový záznam (tzn. do databáze se přidávají nová data, ale existující data se nemění).
- Data přichází v časovém sledu.
- Čas je primární osa.

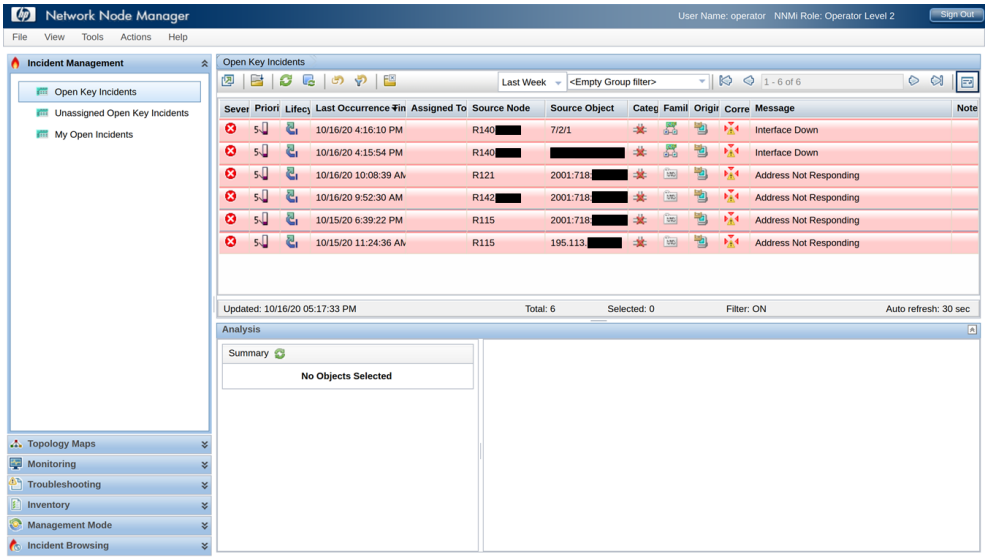


Obrázek 3.2: G3 detailní pohled na rozhraní

Podrobnější popis časových databází lze nalézt v článku [19].

V kategorii časových databází nalezneme poměrně mnoho příkladů. Jejich popularita zároveň stále roste, neboť sbíráme čím dál více dat. Mezi nejznámější databáze patří InfluxDB, Prometheus, Graphite a OpenTSDB. Vždy je ovšem nutné posoudit konkrétní databázi a její vlastnosti, jestli je vhodná na daný problém a jeho analýzu.

Pro kolektor byla nakonec zvolena databáze InfluxDB na základě porovnání z [20] a tabulky 3.1. Jedná se o jednu z nejznámějších a podle statistik i nejpopulárnějších databází v tomto sektoru, lze tedy očekávat větší komunitní podporu. Databáze je vyvíjena jako open-source pod licencí MIT. Je možné ji sdružovat do clusterů, ale pouze v placené verzi.



Obrázek 3.3: Ukázka nástroje HP Network Node Manager

Tabulka 3.1: Porovnání databází (Zdroj [17])

Typ	InfluxDB	Prometheus	Elasticsearch
Podporované datové typy	int, float, bool, string	float	string, int, float, bool, null
Přesnost	nanosekundy	milisekundy	milisekundy
Rychlost zápisu	470k metrik / sek	800k metrik / sek	30k metrik / sek

3.3 Existující nástroje pro sběr dat

Kolektor bude další součástí testovacího systému. Jeho primárním účelem je vytvořit rozhraní, na které bude možné posílat data ze sítě. Výběr konkrétního kolektoru už víceméně není tak důležitý jako výběr databáze. Z pohledu výkonu jednotlivého softwaru není rozdíl téměř znát. Kolektor si sám o sobě žádná data neuchovává. Jeho jediným úkolem je pouze navázat spojení s monitorovaným zařízením a předat přijatá data do databáze. Výkonnostní nároky samozřejmě mohou narůst, například pokud chceme s příchozími daty provádět transformaci, třeba změnu datového typu. V souhrnu lze ovšem říci, že využívané kolektory jsou co nejjednodušší, aby právě odebíraly co nejméně systémových prostředků. Zároveň jsou poměrně často vyvíjeny společně s databázemi.

Telegraf je jeden z kolektorů, který si zaslouží zmínku. Jedná se o software z dílny společnosti InfluxData Inc., tedy stejného vývojáře jako databáze InfluxDB. Stejně jako InfluxDB se jedná o open-source software. Velkou výho-

dou tohoto kolektoru je nepřeberné množství vstupních a výstupních pluginů, sloužících pro navázání kolektoru na další systémy. Jedním z příkladů by mohl být plugin pro protokol SNMP, díky kterému je možné zajistit hladší přechod z SNMP na telemetrii. V případě nutnosti zajištění monitorování i starších zařízení je možné provozovat SNMP a telemetrii zároveň pro zachování zpětné kompatibility. Další výhodou tohoto kolektoru je možnost provádět transformaci dat, která skrz něj procházejí. Jeho hlavní nevýhodou z pohledu této práce ovšem je nemožnost sběru telemetrických dat zakódovaných pomocí Compact GPB.

Pipeline je další možností pro kolekci telemetrických dat. Tento software je vyvíjený společností Cisco a má tedy větší provázanost s jejich síťovými zařízeními. Z tohoto pohledu tedy odstraňuje jednu z nevýhod Telegrafu, protože umožňuje využití kódování dat do Compact GPB formátu. Na druhou stranu ale postrádá široké množství ostatních podporovaných systémů. Zároveň na rozdíl od Telegrafu není možné pomocí Pipeline provádět i monitoring s využitím protokolu SNMP, protože tato funkcionalita není implementovaná.

Použitelných kolektorů je samozřejmě mnoho, nicméně tyto dva jsou nejpoužívanější a nabízejí největší možnosti. Z pohledu testování je jednodušší si vybrat jeden kolektor a na něm si vyzkoušet principy telemetrie. Pokud se bude jednat o produkční řešení, pravděpodobně bude vhodné použít kombinaci kolektorů a z každého využít jeho dobré vlastnosti a pokusit se eliminovat špatné.

Za zmínku by ještě stál nástroj Advanced NETCONF Explorer. Nejedná se o kolektor ve stejném smyslu jako dva výše zmíněné. Jedná se o nástroj vyvíjený opět společností Cisco, který slouží pro navázání spojení pomocí NETCONF protokolu a prohledávání možných YANG modelů na zařízení.

3.4 Existující vizualizační nástroje

Poslední částí je analýza nasbíraných dat. Nejjednodušší je provést analýzu v grafické podobě. Variant je i zde mnoho a principielně lze využít jakýkoliv software, který umí zobrazit data z databáze. V rámci této je využit program Grafana. Možné je ale i využití nástrojů jako Kibana nebo Icinga.

Příkladem, jak může vypadat monitoring jednoho rozhraní na jednom routeru lze vidět na obrázku 3.4.

Obrázek 3.4: Ukázka dashboardu v Grafaně

Kapitola 4

Návrh

4.1 Požadavky na testovací provoz

Primárním cílem práce není sestavit plně fungující monitorovací systém, který by bylo možné nasadit do produkce, nýbrž sestavit monitorovací systém, jenž je vhodný na otestování výkonu telemetrie a snadný na instalaci. Testovací kolektor bude následně otestován pro různé konfigurace popsané v Kapitole 2.

Cíl pro jednoduchou instalaci je vyřešen pomocí instalačního skriptu. Celé testovací řešení se bude sestávat z databáze InfluxDB, kolektorů Telegraf a Pipeline podle výběru uživatele v průběhu skriptu a grafického nástroje Grafana.

4.2 Struktura skriptu

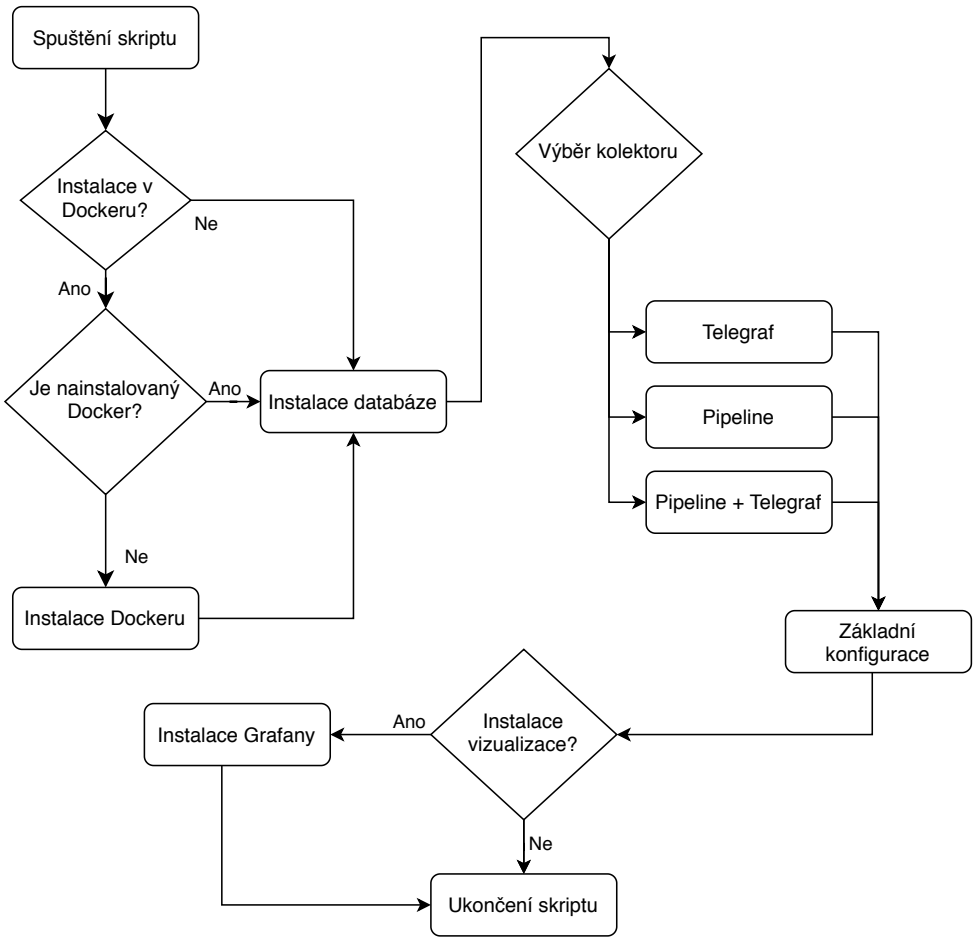
Instalační skript je vytovřen a otestován pro systémy založené na distibuci Debian 8 a vyšší. Zároveň je pro bezproblémové spuštění vyžadován démon pro správu systému *systemd*. Většina nových distribucí už v dnešní době tento démon používá narozdíl od staršího *initd*. Pro zjednodušení instalace byl použit program whiptail, který by měl být součástí většiny Linuxových distribucí založených na systému Debian. Jedná se o grafický program, který usnadňuje proces instalace. [10]

Grafickou strukturu logiky skriptu lze vidět na obrázku 4.1. Podle diagramu lze pozorovat, že dochází k několika větvením. První je založené na dotazu ohledně instalace celého řešení jako kontejnerů v programu Docker. V případě této volby proběhne kontrola, zda jsou již nainstalovány potřebné programy. Poté proběhne instalace databáze InfluxDB. Základní konfigurace databáze je jednoduchá a vyžaduje pouze vytvoření uživatele, který bude moci k databázi přistupovat. Následuje výběr softwaru pro sběr dat. Zde se nabízí tři různé varianty. Jednou je instalace pouze nástroje Telegraf, druhou je instalace

[10] Dokumentace k programu whiptail. `https://en.wikibooks.org/wiki/Bash_Shell_Scripting/Whiptail`

nástroje Pipeline a poslední možností je instalace obou dohromady. Následně proběhne zkopírování konfiguračních souborů z instalační složky a spuštění kolektoru. Posledním větvením ve skriptu už je pouze dotaz na instalaci grafického nástroje Grafana. Jeho instalace je volitelná pro případ, že by cílem instalace bylo pouze ladění chyb samotného sběru dat.

Vlastní konfiguraci libovolného použitého nástroje lze provést v souborech v instalační složce. Konkrétní možnosti konfigurace jsou uvedeny v Sekcích 5.2 a 5.3. Je ovšem potřeba počítat s tím, že některé úpravy konfigurace mohou způsobit nefunkčnost instalačního skriptu.



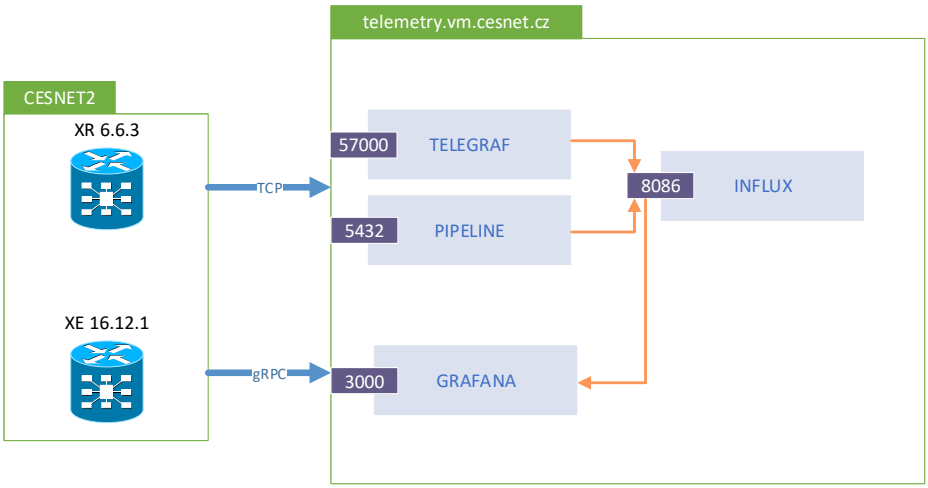
Obrázek 4.1: Logika instalačního skriptu

4.3 Kolektor nasazený přímo na serveru

Instalace testovacího kolektoru přímo na server probíhá pomocí nástrojů wget a dpkg. Nástroj wget stáhne jednotlivé nástroje a poté jsou nainstalovány

pomocí dpkg.

V rámci instalace databáze proběhne automaticky vystavení portu 8086 na vnější rozhraní serveru. Pokud není na serveru aplikovaný firewall, je potřeba počítat s tím, že rozhraní databáze je přístupné. Nástroje Telegraf a Pipeline v rámci instalace zaberou port 57000 resp. 5432. Na těchto portech očekávají příchozí spojení od síťových zařízení. Zároveň navazují spojení s databází na portu 8086. Posledním softwarem je Grafana, která je přístupná pomocí webového rozhraní na portu 3000. Všechny porty je možné téměř libovolně měnit, je ovšem důležité nepoužít port, který už je používán jiným procesem v systému. Výsledné řešení je zobrazeno na obrázku 4.2

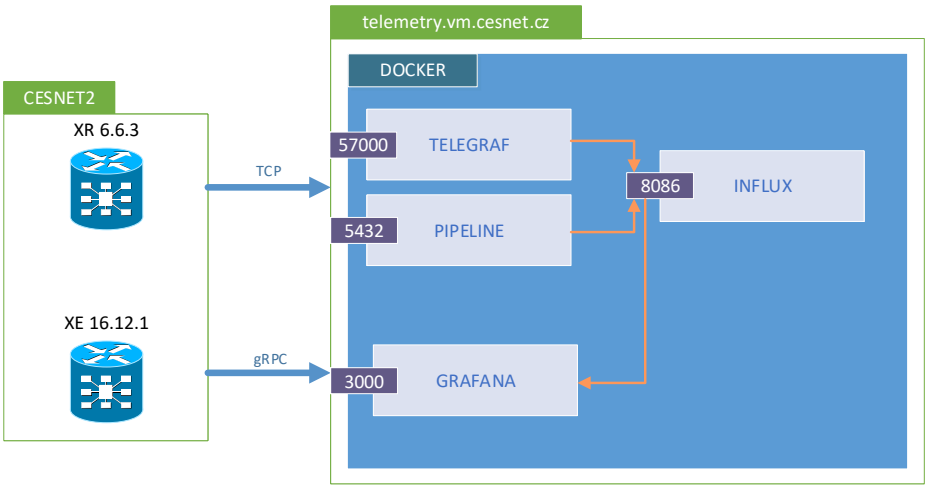


Obrázek 4.2: Struktura navrženého kolektoru na serveru

4.4 Kolektor nasazený v Dockeru

Druhou variantou instalace je nasazení navrženého kolektoru pomocí kontejnerů v Dockeru. Řešení je téměř identické s řešením v Sekci 4.3. Největším rozdílem je nasazení databáze. Díky architektuře Dockeru už InfluxDB nemá port 8086 vystavený přímo na vnějším rozhrazí serveru, což nám nabízí další vrstvu zabezpečení. [11] Konečné schéma nasazení v Dockeru lze vidět na obrázku 4.3.

[11] V případě použití více různých kontejnerů v Dockeru je vhodné i celý kolektor zaizolovat do vlastní sítě. Toto lze provést pomocí `https://docs.docker.com/network/`.



Obrázek 4.3: Struktura navrženého kolektoru v Dockeru

4.5 Výběr datových modelů

Datové modely YANG byly vybrány podle aktuálně nejpoužívanějších identifikátorů SNMP. Jedním z nejvíce používaných OID v systémech G3 a HP Network Node Manager je *1.3.6.1.2.1.2.2.1.1*, které reprezentuje informace rozhraní na síťových zařízeních. Tento identifikátor je zadefinovaný pomocí RFC 2863. Alternativy k tomuto SNMP OID by se na platformě Cisco IOS XR verze 6.6.3 nabízely tři. Modely od Openconfig *openconfig-interfaces.yang* a od IETF *ietf-interfaces.yang* by měly být více obecné a jsou všeobecně lepší pro nasazení na více různých zařízení. Situace je ovšem bohužel stále taková, že se uživatel nemůže úplně spolehnout, že tyto modely budou na všech zařízeních. Navíc nabízejí podstatně méně dat. Z toho důvodu jsem použil spíše modely přímo od společnosti Cisco, v tomto případě se jednalo o *Cisco-IOS-XR-pfi-im-cmd-oper.yang*. Zkrácená ukázka zpracovaná nástrojem pyang do stromové struktury je vidět na ukázce 4.1.

4.1: Stromová struktura YANG modulu

```
module: Cisco-IOS-XR-pfi-im-cmd-oper
  +--ro interfaces
     +--ro interface-xr
        +--ro interface* [interface-name]
           +--ro interface-name xr:Interface-name
           +--ro mac-address
           | ...
           +--ro arp-information
           | ...
           +--ro ip-information
           | ...
           +--ro encapsulation-information
           | ...
           +--ro interface-type-information
           | ...
           +--ro data-rates
           | ...
           +--ro interface-statistics
           | ...
           +--ro l2-interface-statistics
           | ...
           +--ro interface-handle? string
           +--ro interface-type? string
           +--ro state? Im-state-enum
           +--ro line-state? Im-state-enum
           +--ro encapsulation? string
           +--ro encapsulation-type-string? string
           +--ro mtu? uint32
           +--ro last-state-transition-time? uint32
           +--ro speed? uint32
           +--ro crc-length? uint32
           +--ro duplexity? Im-attr-duplex
           +--ro bandwidth? uint32
           +--ro max-bandwidth? uint32
           +--ro keepalive? uint32
           +--ro parent-interface-name? xr:Interface-name
           +--ro loopback-configuration? Im-cmd-loopback-enum
           +--ro description? string
           +--ro transport-mode? Im-attr-transport-mode
           +--ro fast-shutdown? boolean
           +--ro if-index? uint32
```

Modely použité v routeru s IOS XR lze nalézt v tabulce 4.1 a modely použité na operačním systému IOS XE v tabulce 4.2.

Tabulka 4.1: Vybrané YANG modely na IOS XR

Model	Popis
Cisco-IOS-XR-pfi-im-cmd-oper	Stavové informace síťových rozhraní
Cisco-IOS-XR-infra-statsd-oper	Statistiky síťových rozhraní
Cisco-IOS-XR-wdsysmon-fd-oper	Vytížení CPU
Cisco-IOS-XR-nto-misc-shmem-oper	Zatížení paměti RAM
Cisco-IOS-XR-ipv4-io-oper	IPv4 infomace k jednotlivým rozhraním
Cisco-IOS-XR-ipv6-ma-oper	IPv6 infomace k jednotlivým rozhraním
Cisco-IOS-XR-telemetry-model-driven-oper	Informace o stavu nakonfigurované telemetrie
Cisco-IOS-XR-ipv4-ospf-oper	Informace protokolu OSPF
Cisco-IOS-XR-ipv4-bgp-oper	Informace protokolu BGP

Tabulka 4.2: Vybrané YANG modely na IOS XE

Model	Popis
Cisco-IOS-XE-interfaces-oper	Informace ohledně vytížení CPU
Cisco-IOS-XE-mdt-oper	Informace o stavu nakonfigurované telemetrie
Cisco-IOS-XE-memory-oper	Zatížení paměti RAM
Cisco-IOS-XE-process-cpu-oper	Vytížení CPU
Cisco-IOS-XE-ospf-oper	Informace protokolu OSPF
Cisco-IOS-XE-bgp-oper	Informace protokolu BGP

Kapitola 5

Realizace

5.1 Konfigurace databáze

Pro základní použití si vystačíme s minimem změn v konfiguraci databáze. Skript v průběhu provede pouze jednu konfigurační změnu a tou je přidání nového uživatele pomocí SQL příkazu. Tento příkaz lze provést pouze v průběhu instalace a následně je tato možnost zakázána z důvodu zabezpečení databáze. Jméno a heslo do databáze je nastaveno pomocí proměnných *INFLUX_USER* a *INFLUX_PASSWD*. Oba údaje lze pro vyšší zabezpečení změnit v přiloženém souboru *.env*. I pro testovací kolektor je velice doporučeno tyto položky změnit, neboť v základu jsou v souboru velice jednoduché. Jméno a heslo k databázi jsou poměrně často používané údaje v průběhu instalace a byly tedy pro zjednoduššení navázány do jednoho souboru jako proměnné.

5.2 Konfigurace kolektoru

Nejprve se zaměřím na konfiguraci nástroje Telegraf. Jakožto program, který neslouží pouze pro sběr dat pomocí telemetrie, nabízí velké množství vstupních a výstupních pluginů. Nejdůležitějšími pluginy pro tuto práci jsou výstupní `[[outputs.influxdb]]` a vstupní `[[inputs.cisco_telemetry_mdt]]`.

Jednodušším z nich je `[[inputs.cisco_telemetry_mdt]]`, který lze vidět na ukázce 5.1. V pluginu jsou z pohledu konfigurace nejvíce zajímajímavé položky `transport` a `service_address`. Pomocí `transport` volíme transportní protokol probíraný v Sekci 2.4, kdy v tomto pluginu máme možnost pouze „tcp“ nebo „grpc“. Volbou `service_address` můžeme určit, na kterém portu bude server očekávat příchozí spojení ze síťových zařízení. V případě použití tranportu pomocí „grpc“ jsou zajímavé volby `tls_cert`, `tls_key` a `tls_allowed_cacerts`, díky kterým můžeme celý provoz zašifrovat a případně omezit spojení pouze na naše zařízení pomocí autorizace jejich klíčů. Základní cesty jednotlivých modelů jsou poměrně dlouhé a nemusí být úplně

pochopitelné. Pro jednodušší pracování s daty v databázi tedy můžeme použít modul pro vytváření aliasů `[inputs.cisco_telemetry_mdt.aliases]`.

5.1: Vstupní plugin nástroje Telegraf

```
[[inputs.cisco_telemetry_mdt]]
  transport = "tcp"
  service_address = ":57000"

 # tls_cert = "/etc/telegraf/cert.pem"
 # tls_key = "/etc/telegraf/key.pem"
 # tls_allowed_cacerts = ["/etc/telegraf/clientca.pem"]

 [inputs.cisco_telemetry_mdt.aliases]
  ifstats = "ietf-interfaces:interfaces-state/interface/
      statistics"
```

Druhým důležitým pluginem Telegrafu je výstupní `[[outputs.influxdb]]`, který slouží pro propojení s databází InfluxDB. Zde je podstané nastavit položku `urls`, tak aby směřovala na otevřený port databáze. V tomto případě se databáze nachází na stejném stroji a můžeme tedy použít jako adresu *localhost*. Dále musíme určit, do které datábaze se budou data z tohoto kolektoru zapisovat. K tomu slouží `database`. Pokud bychom navíc chtěli přímo identifikovat tuto konkrétní instanci Telegrafu, můžeme nastavit položku `database_tag`, čímž se všechna data, která se budou zapisovat do databáze taggovat příslušným identifikátorem. [12] Následují položky `username` a `password` slouží k přihlášení do databáze popsané v Sekci 5.1. Posledními položkami jsou opět možnosti šifrování komunikace mezi Telegrafem a InfluxDB. Toto řešení je vhodné použít, pokud se databáze a kolektor nacházejí na různých serverech.

[12] Stejně jako název databáze, tak i její případný tag se udává ve formátu string.

5.2: Výstupní plugin nástroje Telegraf

```
[[outputs.influxdb]]
   urls = ["http://127.0.0.1:8086"]

  database = "mdt"
  # database_tag = ""

  username = $INFLUX_USER
  password = $INFLUX_PASSWD

  # tls_ca = "/etc/ssl/trusted_ca/DigiCertHighAssuranceEVRootCA.
     pem"
  # tls_cert = "/etc/ssl/tcs/cert.pem"
  # tls_key = "/etc/ssl/tcs/telegrafkey.pem"
```

Konfigurace kolektoru Pipeline je v mnoha ohledech velice podobná konfiguraci Telegrafu. Musíme nakonfigurovat vstupní a výstupní pluginy. Jedná se o plugin gRPCDialout, který slouží pro navázání spojení s routery. Zde jsou ovšem oproti Telegrafu dvě podstatné změny. Plugin umožňuje spojení pomocí protokolu grpc a zároveň je možné posílaná data kódovat pomocí kompaktních GPB. Část konfigurace je v ukázce 5.3.

5.3: Vstupní plugin nástroje Pipeline

```
[gRPCDialout]
 stage = xport_input
 type = grpc

 encap = gpb
 listen = :5432
```

Výstupní plugin pro navázání spojení s databází lze vidět v ukázce 5.4.

5.4: Výstupní plugin nástroje Pipeline

```
[mymetrics]
 stage = xport_output
 type = metrics
 file = /data/metrics.json
 output = influx

 influx = $INFLUX_URL
 database = $INFLUX_DATABASE
```

Lehce složitější částí konfigurace je nutnost vytvoření modulu pro automatické spouštění nástroje Pipeline pomocí *systemd*, který lze najít v souboru *./src/install/pipeline.service*.

5.3 Konfigurace vizualizace

Konfigurace nástroje Grafana probíhá primárně ve dvou souborech. Jedná se o soubory grafana.ini a sample.yaml. Konfiguraci souboru grafana.ini je možné vidět na ukázce 5.5. V rámci skriptu bylo použito nastavení, které lze vidět na ukázce. Z důvodu zjednodušení instalace byl použit pouze protokol http místo https. Pokud ovšem je možnost vygenerovat certifikát, tak je vhodné použít spíše šifrovaný protokol https. Stroj, na kterém probíhalo testování, měl vygenerovaný certifikát a webové rozhraní tedy bylo vystavené pouze na protokolu https.

5.5: Soubor grafana.ini

```
[server]
    protocol = http
    http_addr = 127.0.0.1
    http_port = 3000

    ;cert_file = /etc/ssl/tcs/cert.pem
    ;cert_key = /etc/ssl/tcs/grafanakey.pem
```

V ukázce 5.6 lze vidět obsah souboru sample.yaml, který slouží pro konfiguraci databáze. Tu je možné v Grafaně nakonfigurovat i přes webové rozhraní. Pro skript byla ovšem zvolena možnost databázi nakonfigurovat pomocí souboru z důvodu jednodušší instalace. Soubor tedy ve velké míře využívá systémových proměnných, které jsou zadefinované v rámci instalačního skriptu v souboru *.env*.

5.6: Soubor sample.yaml

```
datasources:
 - name: InfluxDB
   type: influxdb
   url: $INFLUX_URL
   password: '$INFLUX_PASSWD'
   user: '$INFLUX_USER'
   database: $INFLUX_DATABASE
```

Poslední podstatnou částí nástroje Grafana jsou samostatné dashboardy, neboli okna, která vidí uživatel. Ukázka již vytvořeného dashboardu byla vidět

již na obrázku 3.4. Tyto dashboardy lze jednoduše exportovat a importovat do Grafany. V adresáři instalačního skriptu se nachází i tento export ve formátu JSON. Je možné ho použít nakopírováním souboru do složky */etc/grafana-/provisioning/datasources/* po provedení instalace. Tuto možnost je ovšem potřeba udělat ručně, neboť skript soubor sám nekopíruje. Důvod pro toto chování je prostý. Dashboardy je možné přizpůsobit velkou měrou a je potřeba, aby si je každý uživatel přizpůsobil dashboard tomu, která data chce sledovat a v jaké podobě.

5.4 Konfigurace síťových zařízení

V průběhu psaní práce jsem měl k dispozici testovací Cisco routery, na kterých běží operační systém IOS XR 6.6.3 a IOS XE 16.12.1. Většina konfigurace a testování probíhala na stroji se systémem IOS XR, neboť se jedná o systém, jenž používá většina páteřních routerů. Samotná konfigurace telemetrie probíhá na platformě XR následujícím způsobem.

Je potřeba zadefinovat cílový server, který bude data přijímat. K tomu slouží následující konfigurace (ukázka 5.7). Pomocí příkazu `address-family` nastavíme adresu cílového kolektoru a port, na kterém server poslouchá. [13] Následně nastavíme transportní protokol. Bohužel stroj, na kterém běží systém IOS XR v současnosti v CESNETu, je pouze 32 bitová architektura a tedy podle Cisco specifikace [21] v tabulce na straně 11, umožňuje transport pouze pomocí TCP a UDP. Poslední volbu, kterou je potřeba zadat je informace ohledně kódování dat. Zde jsou na výběr možnosti tři: JSON, KV-gPB a Compact-gPB. Jak už bylo zmíněno v Sekci 2.6, tak kompaktní GPB jsou sice nejmenší na přenos, ale je potřeba k nim mít překladový soubor *.proto* na kolektoru. Z toho důvodu byla využita jednodušší varianta KV-gPB.

5.7: Konfigurace cíle na IOS XR

```
telemetry model-driven
 destination-group DGroup1
  address-family ipv4 78.128.211.188 port 57000
   encoding self-describing-gpb
   protocol tcp
  !
 !
!
```

Další část konfigurace (ukázka 5.8) definuje, která data konkrétně budou posílána. Začátek celé cesty k datům je definovám pomocí souboru a následně je dvojtečkou oddělen od konkrétní cesty v modelu. V tomto případě se jedná

[13] Je samozřejmě možné použít i IPv6 adresu.

o celý model `telemetry-model-driven`, ale v jiném případě může cesta vypadat například takto: `memory-summary/nodes/node/detail`. Nejjednodušší způsob určení konkrétní cesty je využít vylistování modelu pomocí stromové struktury, jak bylo ukázáno v Sekci 4.5.

5.8: Konfigurace zdrojových dat na IOS XR

```
telemetry model-driven
 !
  sensor-group SGroup1
  sensor-path Cisco-IOS-XR-telemetry-model-driven-oper:telemetry
      -model-driven
 !
!
```

Poslední částí konfigurace na IOS XR je propojení předchozích dvou konfigurací pomocí položky `subscription`. V rámci ní musíme nastavit zdroj `sensor-group-id` [14] a cíl `destination-id`, podle předchozí konfigurace. Následně můžeme nakonfigurovat i `source-interface`, z něhož budou data posílána. Obecně je vhodné tuto položku nastavit, ideálně na rozhraní, které se nemění. Pokud tuto položku nenastavíme, tak router bude posílat data na kolektor se zdrojovou adresou rozhraní, který má podle směrovací tabulky zaregistrovaný jako „nejlepší“ cestu ke kolektoru. V případě problémů na síti se tedy může stát, že na kolektor přichází data z různých adres, i když se jedná o jeden zdroj. V rámci databáze to žádný problém není, ale může to způsobit zmatení administrátora při řešení problémů.

5.9: Konfigurace intervalu posílání dat na IOS XR

```
telemetry model-driven
 !
  subscription Sub1
  sensor-group-id SGroup1 sample-interval 30000
  destination-id DGroup1
  source-interface Loopback0
 !
!
```

Konfigurace na zařízení s IOS XE je o něco kratší. Co se důležitosti z pohledu síťové infrastruktury CESNET týče se jedná o mnohem menší počet zařízení. Navíc jde spíše o přístupové switche a narozdíl od platformy XR nebyl test telemetrie prováděn na těchto produkčních zařízeních. Na této platformě

[14] Na ukázce lze vidět nastavení takzvané Streaming Telemetry, kde data se odesílají po 30000 milisekundách neboli každých 30 sekund. V případě, že bychom chtěli data posílat pouze při změně, nastavili bychom *sample-interval* na 0.

probíhá konfigurace všech části dohromady. Opět nalezneme stejné možnosti kódování jako na IOS XR. Adresu kolektoru konfigurujeme pomocí příkazu `receiver`. Položka `source-ip` nám nahrazuje `source-interface` ze systému IOS XR. Zde ovšem narozdíl od systému IOS XR není tato položka volitelná, nýbrž povinná. Bez ní stroj data neposílá. Cestu k datům konfigurujeme pomocí příkazů `filter xpath`. Za zmínku ještě stojí klasické nastavení intervalu posílání dat pomocí příkazu `update-policy`. Zde je změna oproti systému IOS XR, kdy pokud chceme posílat data při změně, tak použijeme místo možnosti *periodic* možnost *on-change*.

5.10: Konfigurace telemetrie na IOS XE

```
telemetry ietf subscription 101
  encoding encode-kvgpb
  filter xpath /memory-ios-xe-oper:memory-statistics/memory-
      statistic
  stream yang-push
  update-policy periodic 6000
  source-ip 195.113.x.x
  receiver ip address 78.128.211.188 5432 protocol grpc-tcp
```

Kapitola 6

Testování

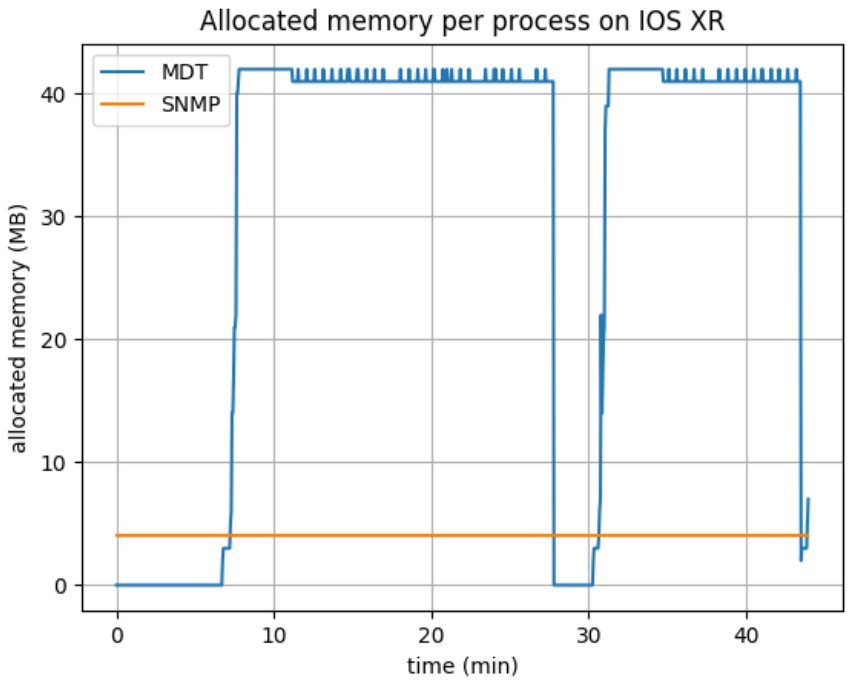
6.1 Vytížení síťových prvků

Testování kolektoru a síťových prvků probíhalo nejprve na jednom testovacím routeru. Na základě tohoto testování byl poté sestaven větší měřený test. Ten proběhl sběrem dat z 11 páteřních routerů v síti CESNET2. Datové modely byly zvoleny podle Sekce 4.5. Konkrétní nasazené konfigurace jsou k nalezení ve složce v části *./src/config/*. Test primárně sbíral údaje z rozhraní routerů a vytížení hardwaru samotného routeru. Testování probíhalo pouze na platformě s operačním systémem IOS XR. Vzheledem k tomu, že se jednalo o test na produkčních strojích, probíhalo na nich v průběhu testu i monitorování pomocí SNMP. Bylo tedy možné porovnat rozdíly telemetrie a SNMP souběžně. Test proběhl ve dvou fázích.

V první fázi byl použit model neustálého streamování dat z routeru na kolektor v intervalu každých 30 sekund. Tato data byla sbírána přibližně 15 minut. Poté proběhlo odkonfigurování a vyčkání pár minut, aby se uvolnily prostředky. Následovala druhá fáze, kde se pro posílání části dat použila Event-Driven Telemetry.

Odchylky mezi sběrem dat pomocí SNMP a telemetrie se projevily nejvíce v rozdílu využití operační paměti RAM routeru, jak bylo očekáváno. V grafu 6.1 lze vidět rozdíl mezi vytížením procesu *snmpd* a *mdtd*. Je vidět, že telemetrie si v průběhu testu alokovala přibližně 8x více více paměti než SNMP. Důvod tohoto rozdílu bude v samotném přístupu. Jak bylo popsáno v Sekci 1.2, router si data v případě telemetrie pravděpodobně ukládá do operační paměti a po uběhnutí intervalu je odesílá pryč. Zatímco u SNMP reaguje okamžitě při zpracování příslušného paketu, není tedy potřeba data v průběhu ukládat na další místo. Pomocí grafu 6.2 můžeme vidět, jaké byly hodnoty volné paměti v průběhu testu.

Druhou stranou testování zátěže je vytížení CPU. Na grafu 6.3 vidíme vytížení CPU procesy *snmpd* a *mdtd* na jednom routeru, který zpracovává externí peering pomocí protokolu BGP. Tento stroj se běžně pohybuje mezi



Obrázek 6.1: Paměť alokovaná procesy SNMP a MDT

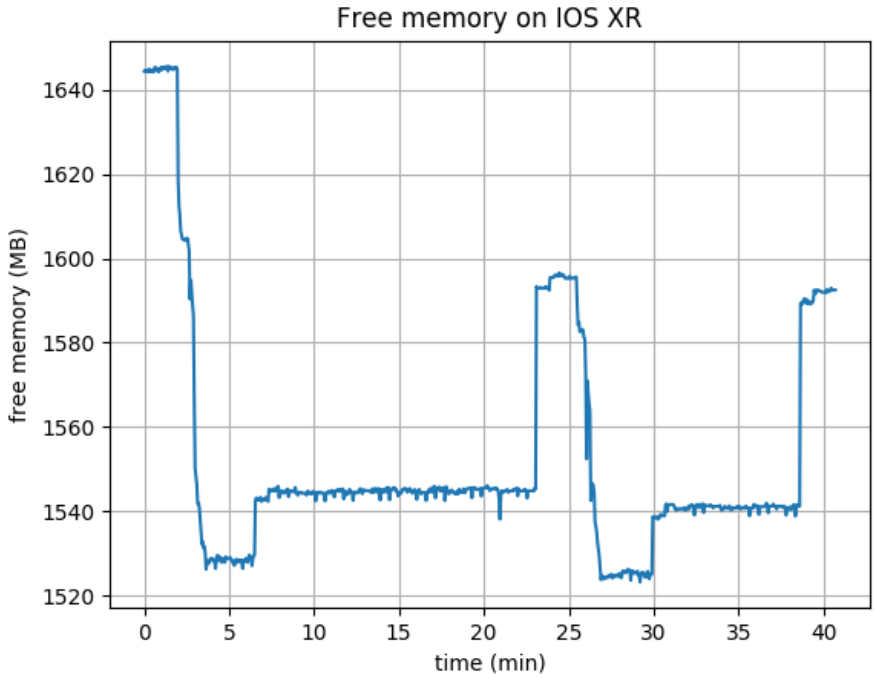
90 % - 100 % vytížení CPU a jakákoliv zátěž tedy může být problém. V grafu jsou vidět výkyvy, které lze přisoudit příchodu požadavku na stavová data a jejich zpracování. Největší vytížení protokolem SNMP se pohybovalo okolo 2 %. Naproti tomu proces telemetrie CPU téměř nevytěžuje. Naměřené hodnoty byly minimální a hodnotu 0,01 % překročily ve velice malém měřítku.

Z testu tedy vyplynula skutečnost, že telemetrie je z pohledu zátěže síťového zařízení vhodná ve chvíli, kdy se nám nedostává procesorového času a máme vetší množství operační paměti RAM. Protokol SNMP je naopak vhodné použít, pokud máme malé množství operační paměti, ale nemáme problém se zatížením procesoru.

6.2 Vytížení kolektoru

V rámci testování proběhl i záznam chování navrženého kolektoru. Byly zaznamenány statistiky z nástrojů InfluxDB a Telegraf. Kolektor byl otestován na virtuálním serveru, který měl 4 jádra CPU, 16 GB RAM paměti a připojení do sítě rychlostí 1 Gb/s. Na serveru byl operační systém Debian 10.

Vytížení procesoru bylo velmi nízké v případě obou nástrojů. Graf z průběhu testu lze vidět na obrázku 6.4.

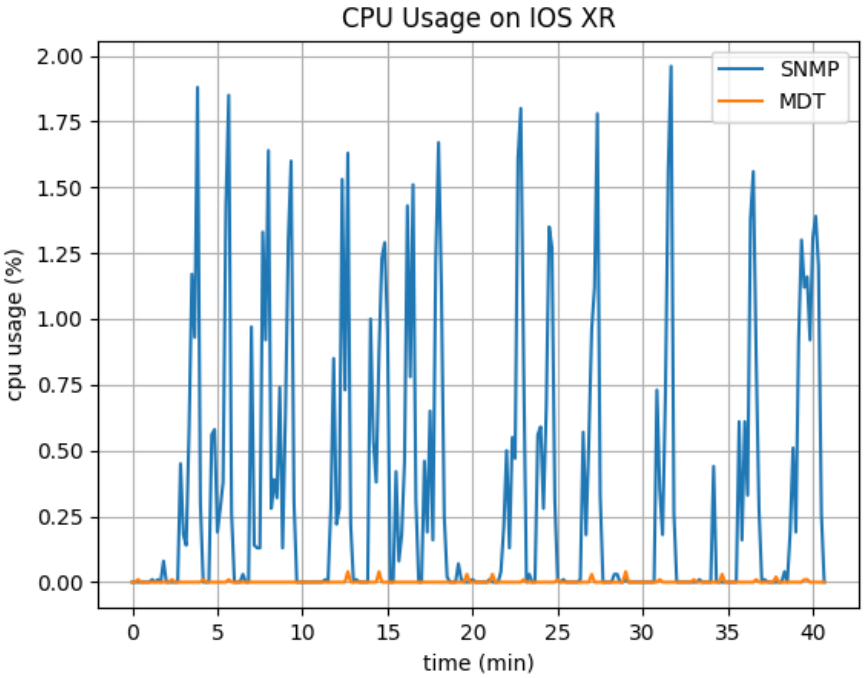


Obrázek 6.2: Volná paměť RAM na IOS XR

Druhým sledovaným paramatrem bylo vytížení operační paměti. Vytížení jednotlivých procesů lze vidět na obrázku 6.5. Je zřejmé, že samotné procesy paměť nijak moc nezabírají. V případě databáze InfluxDB je ovšem potřeba dávat pozor na hodnotu cachované paměti. Z obrázku 6.6 lze vyčíst jak položka „Buff/Cache“ pomalu roste. Databáze totiž paměť pro svoje fungování nealokuje skrz úvodní proces, ale využívá buffrované paměti. Pro dlouhodobější fungování databáze je tedy dobré nastavit agregaci dat po určitém intervalu. V InfluxDB toto nastavení lze provést pod názvem „Retention policy“.

Součástí sledování kolektoru bylo i odchytávání paketů přicházejících na server a výpočet datového toku. Bylo zpracováno porovnání mezi neustále posílanou telemetrií a telemetrií posílanou pouze při změně. Podle testování se ukázalo, že při posílání stejných dat každých 30 s po dobu 15 minut je rozdíl skoro čtyřnásobný. Porovnání lze nalézt na obrázku 6.7.

V rámci porovnání datového toku proběhlo i otestování velikosti TCP streamu při použití různého kódování. Výsledky lze vidět na obrázku 6.8. Jak bylo předpokládáno v Sekci 2.6, kompaktní kódování GPB se ukázalo být nejúspornější z pohledu objemu dat posílaných po fyzickém médiu a to skoro sedmkrát.



Obrázek 6.3: Zatížení CPU na IOS XR

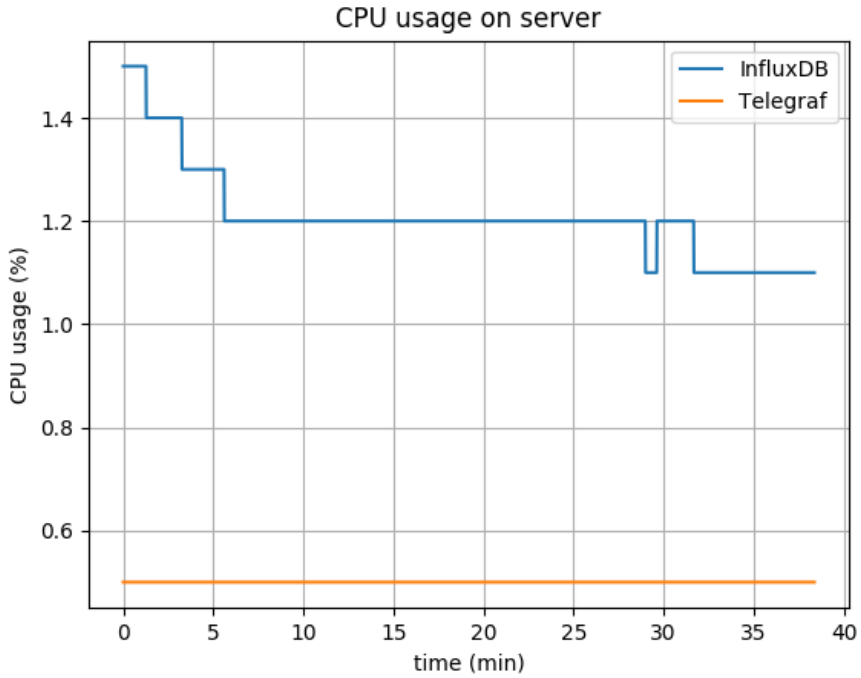
6.3 Možnosti rozšíření navrženého řešení

Celý kolektor byl navrhován s možností snadného rozšíření hned od začátku. Testovací řešení je tedy sestaveno velice modulárně z jednotlivých částí, kde každá část plní pouze jednu povinost. Jednotlivé části jsou mezi sebou propojené pomocí protokolu http. V případě produkčního nasazení výsledného řešení je možné a dokonce i doporučené použít zabezpečený protokol https.

Samotné nástroje pro kolekci dat Telegraf a Pipeline je možné nasazovat téměř neomezeně.

Podobné platí i pro vizualizační nástroj Grafana, který může mít mnoho instancí připojených k databázi. Ačkoliv to nedává takový smysl, jako například u kolektorů, je to i v tomto případě možné.

Posledním použitým nástrojem byla databáze InfluxDB. Tu je také možné rozšířit a spojit do klusterů. Problémem tohoto řešení však je, že není zdarma, ale musí se zaplatit. Na druhou stranu, jak bylo vidět v Sekci 6.2, ani při posílání dat z větší části sítě nebyl virtuální server vytížen natolik, aby bylo potřeba databázi spojovat do klusteru.

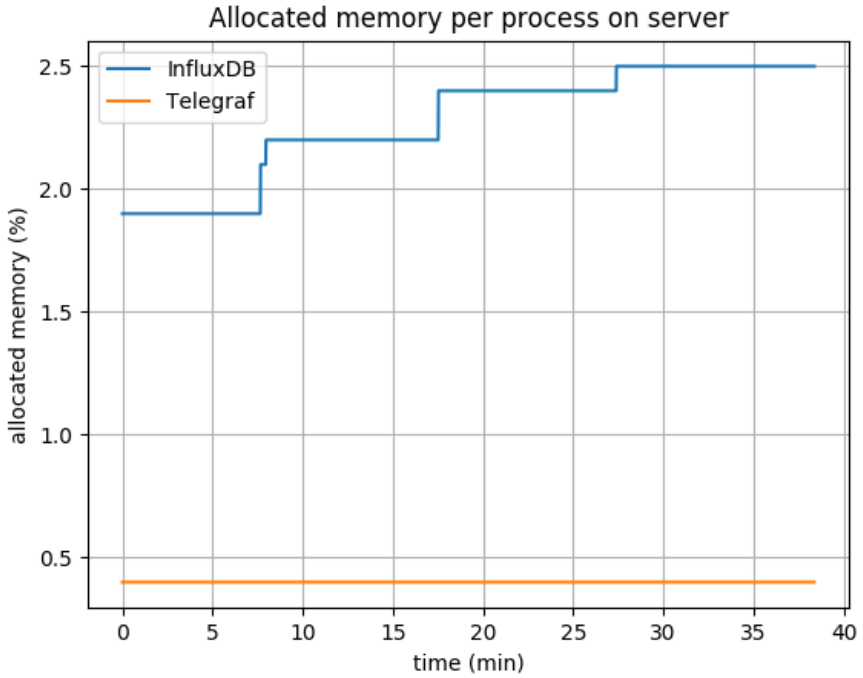


Obrázek 6.4: Zatížení CPU na serveru

6.4 Vyhodnocení testování

Částečné výhody telemetrie byly znát už v průběhu analýzy. Nepopiratelně se jedná o novou technologii, která bude výrobci čím dál více využívána. V průběhu návrhu už se ovšem začaly projevovat některé problémy, které částečně pramenily z chybějící standardizace, ale i z nedostatečné implementace některých vlastností na straně jak vývojářů síťových zařízení, tak vývojářů softwaru použitého v testovacím systému. To je ovšem pochopitelné, vzhledem k tomu, že se využití telemetrie ve stavovém sledování sítě používá pouze pár let a v produkci ho používá pouze menší část společností. Myslím, že se dá očekávat následný vývoj, který bude pravděpodobně směřovat na využití více jednotných YANG modelů, buď vytvořených přímo IETF nebo mnohem pravděpodobněji z pracovní skupiny OpenConfig. Zároveň lze očekávat mnohem větší nástup transportního protokolu gNMI, oproti prostému transportu pomocí samotného TCP nebo UDP.

Odpověď na otázku, jestli nasadit monitorování pomocí telemetrie, tedy není úplně jednoznačná. Ano, telemetrie má nesporné výhody oproti SNMP, jako například nižší využití CPU na síťových zařízeních. Na druhou stranu tento úbytek je částečně vykompenzován vyšším využitím paměti. Je tedy potřeba zvážit, kterých prostředků je méně. Dalším faktorem je samozřejmě



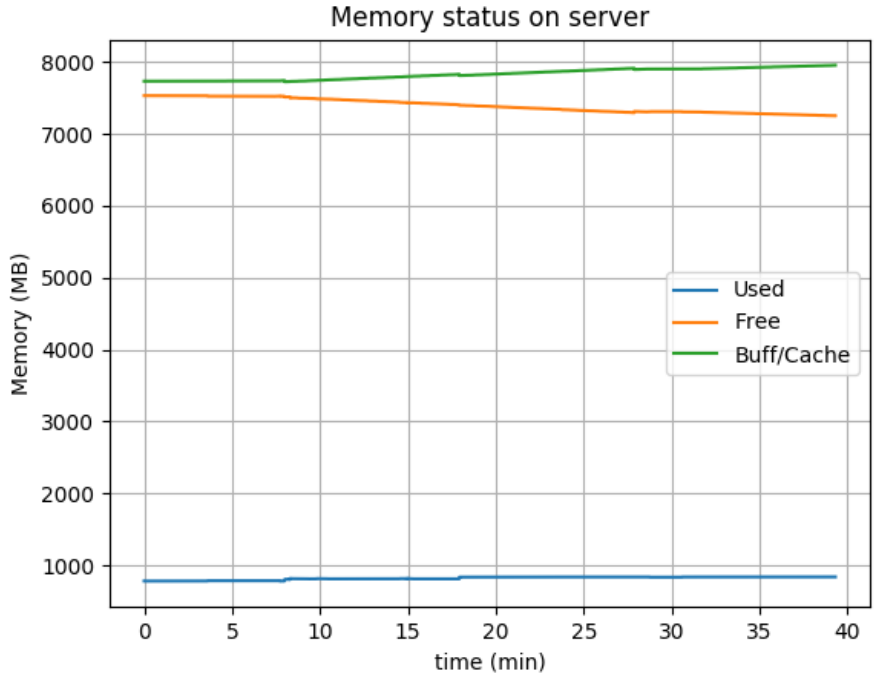
Obrázek 6.5: Zatížení paměti RAM programy Telegraf a InfluxDB

vytížení linek, kde se ukazuje být telemetrie silně dominantní, především díky využití formátu Proto Bufferů.

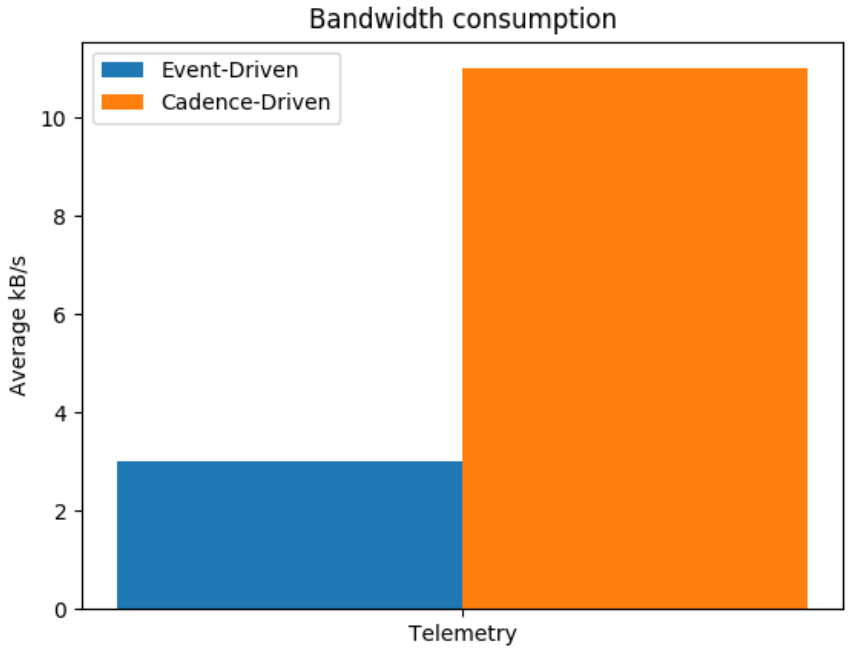
Víceméně by šlo říci, že pokud není potřeba sbírat spousty hodnot a zároveň existuje v síti nasazený monitorovací systém založený na protokolu SNMP, není nutné se ho okamžitě zbavovat a snažit se ho nahradit pomocí systému založeném na Streaming Telemetry. Protokol SNMP určitě v dohledné době hned tak nezmizí. Na druhou stranu pokud je v plánu provést upgrade síťových zařízení za novější, je vhodné určitě na telemetrii pamatovat a poohlédnout se po zařízeních, která ji budou podporovat. Lze nejspíše očekávat, že monitorovací systémy budou budovány s použitím jak SNMP, tak telemetrie, z důvodů podpory co největší škály zařízení.

Výhoda YANG modelů jednoznačná v případě používání optických vláken v infrastruktuře. Zatímco pomocí SNMP nelze sledovat údaje z těchto rozhraní, YANG modely už to umožňují. V souhrnu při správném výběru modelů lze získat mnoho informací o síti, které s využitím SNMP získat nelze.

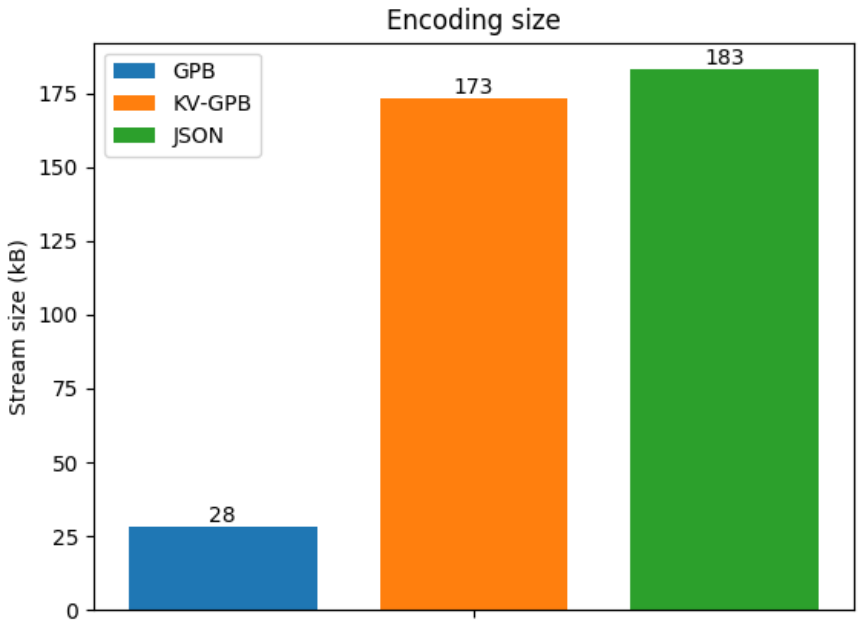
Posledním, ale velice důležitým argumentem pro nasazení telemetrie, je případ, kdy je v plánu provádět jednotnou konfiguraci z nějakého centrálního serveru. V tom případě je nasazení telemetrie už pouze krůček od automatizace celé síťové infrastuktury s potenciálním využitím strojového účení.



Obrázek 6.6: Stav paměti na serveru



Obrázek 6.7: Rozdíl ve vytížení linky mezi „Streaming telemetry“ a „Event telemetry“



Obrázek 6.8: Velikost kódování

Závěr

Monitorování síťové infrastruktury je kriticky důležité pro udržení funkcionality a provozuschopnosti, pro plánování rozvoje a v neposlední řadě i kvůli bezpečnostním důvodům. Proto je nezbytné se touto problematikou zabývat a hledat případně experimentálně vyhodnocovat nové použitelné způsoby pro efektivní sběr a vyhodnocování dat o provozu síťových prvků.

Cílem práce byl návrh a vytvoření testovacího experimentálního kolektoru pro sběr telemetrických dat. Tato telemetrická data získaná pomocí tzv. Model-Driven Telemetry by měla být nástupcem stávajících technologií jako je SNMP. Důvodem by měla být efektivita přenosu dat a optimální zatížení síťových prvků poskytujících data.

V rámci této závěrečné práce se podařilo vytvořit testovací prostředí využívající reálné síťové prvky a snadno replikovatelný kolektor dat, což celé umožnilo sledovat a vyhodnotit chování technologie Model-Driven Telemetry. Díky připraveným skriptům a konfiguracím je možné toto vytvořené prostředí snadno přenést do produkčního prostředí a navíc je možné infrastrukturu potenciálně duplikovat kvůli dosažení větší škálovatelnosti. Pomocí vytvořeného prostředí byla telemetrie porovnána ve spolupráci se sdružením CESNET vůči aktuálně používanému způsobu monitorování v produkčních podmínkách národní akademické infrastruktury CESNET2.

V kapitole 1 byly rozebrány principy proč by telemetrie měla být výhodnější než doposud používaná řešení. Zároveň byl definován pojem telemtrie pro účely této práce. Následně v rámci Kapitoly 2 proběhla hlubší analýza možností telemetrie, především na zařízeních Cisco, podstatná část principů je však u většiny výrobců síťových zařízení stejná. Došlo na konkrétní porovnání starších principů představovaných pomocí SNMP s novějšími reprezentovaných v tomto případě telemetrií. Toto porovnání poté velice dobře posloužilo v Kapitolách 4 a 5 pro návrh a nasazení testovacího kolektoru. Nejpodstatnější zjištění z pohledu práce ovšem přinesla Kapitola 6 a především sekce 6.4. V této sekci byly zhodnoceny získané poznatky ohledně telemetrie a vyslovena doporučení ohledně nasazování telemetrie v rámci sítě.

Výsledky práce se ukázaly být prospěšné a začaly být prováděny postupné kroky pro nasazení telemetrie do produkce v síti CESNET2. Aby bylo možné telemetrii nasadit do produkce, bude ovšem potřeba provést úpravy v nástroji G3 popsaného v Sekci 3.1. Tyto úpravy a celkové nasazení telemetrie do produkce ve sdružení CESNET se jeví být vhodným pokračováním této práce do budoucna.

Literatura

[1] Kay, R.: Simple Network Management Protocol. *Computerworld [online]*, září 2002, [cit. 2020-08-22]. Dostupné z: https://www.computerworld.com/article/2578217/simple-network-management-protocol.html

[2] Case, J.; Fedor, M.; Schoffstall, M.; aj.: A Simple Network Management Protocol. *RFC1067 [online]*, srpen 1988, [cit. 2020-08-22]. Dostupné z: https://tools.ietf.org/html/rfc1067

[3] Schoenwaelder, J.: Overview of the 2002 IAB Network Management Workshop. *RFC3535 [online]*, květen 2003, [cit. 2020-08-22]. Dostupné z: https://tools.ietf.org/html/rfc3535

[4] McCloghrie, K.; Kastenholz, F.: The Interfaces Group MIB. *RFC2863 [online]*, červen 2000, [cit. 2020-08-22]. Dostupné z: https://tools.ietf.org/html/rfc2863

[5] Cadora, S.: The limits of SNMP. *Cisco [online]*, červen 2016, [cit. 2020-08-22]. Dostupné z: https://blogs.cisco.com/sp/the-limits-of-snmp

[6] Gervasi, P.: Why streaming telemetry tops SNMP in tracking network performance. *Techtarget [online]*, červenec 2019, [cit. 2020-08-22]. Dostupné z: https://searchnetworking.techtarget.com/tip/Why-streaming-telemetry-tops-SNMP-in-tracking-network-performance

[7] Voit, E.; Clemm, A.; Gonzalez Prieto, A.: Requirements for Subscription to YANG Datastores. *RFC7923 [online]*, červen 2016, ISSN 2070-1721, [cit. 2020-08-22]. Dostupné z: https://tools.ietf.org/html/rfc7923

[8] Song, H.; Qin, F.; Martinez-Julia, P.; aj.: Network Telemetry Framework. *[online]*, říjen 2020, [cit. 2020-11-07]. Dostupné z: https://tools.ietf.org/html/draft-ietf-opsawg-ntf-05

[9] SNMP tutorial. [online], [cit. 2020-11-07]. Dostupné z: https://www.manageengine.com/network-monitoring/what-is-snmp.html

[10] What is SNMPv1, SNMPv2c, and SNMPv3? [online], [cit. 2020-11-07]. Dostupné z: https://www.dpstele.com/snmp/v1-v2c-v3-difference.php

[11] NE40E V800R011C10 Configuration Guide - System Monitor 02. [online], [cit. 2020-11-07]. Dostupné z: https://support.huawei.com/enterprise/en/doc/EDOC1100125844/3ea8d77c/telemetry-configuration

[12] Sunil, P.; Adesh, K.: ASR9K Model Driven Telemetry Whitepaper. *Cisco [online]*, březen 2020, [cit. 2020-11-07]. Dostupné z: https://www.cisco.com/c/en/us/support/docs/routers/asr-9000-series-aggregation-services-routers/215321-asr9k-model-driven-telemetry-whitepaper.html

[13] Telemetry Configuration Guide for Cisco NCS 6000 Series Routers, IOS XR Release 7.1.x. [online], srpen 2020, [cit. 2020-11-07]. Dostupné z: https://content.cisco.com/chapter.sjs?uri=/searchable/chapter/content/en/us/td/docs/routers/ncs6000/software/ncs6k-r7-1/telemetry/configuration/guide/b-telemetry-cg-ncs6000-71x/b-telemetry-cg-ncs6000-71x_chapter_01.html.xml

[14] Paul, B.; Marcus, H.; Carl, L.; aj.: gRPC Network Management Interface (gNMI). *[online]*, leden 2018, [cit. 2020-11-07]. Dostupné z: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md

[15] De Oliveira Novais, B.: Understanding gNMI on IOS-XR with Python. *[online]*, leden 2020, [cit. 2020-11-07]. Dostupné z: https://community.cisco.com/t5/service-providers-documents/understanding-gnmi-on-ios-xr-with-python/ta-p/4014205

[16] Okasha, K.: Network Automation and the Rise of NETCONF. *[online]*, květen 2017, [cit. 2020-11-07]. Dostupné z: https://medium.com/@k.okasha/network-automation-and-the-rise-of-netconf-e96cc33fe28

[17] Wallace, B.: Advanced Topics in XR Telemetry. *Cisco [online]*, leden 2020, [cit. 2020-11-07]. Dostupné z: https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2020/pdf/BRKSPG-2503.pdf

[18] Encoding. [online], [cit. 2020-11-07]. Dostupné z: https://developers.google.com/protocol-buffers/docs/encoding/

[19] Kulkarni, A.: What the heck is time-series data (and why do I need a time-series database)? *[online]*, listopad 2018, [cit. 2020-11-07]. Dostupné z: `https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/`

[20] DB-Engines Ranking of Time Series DBMS. [online], [cit. 2020-11-07]. Dostupné z: `https://db-engines.com/en/ranking/time+series+dbms`

[21] Barth, S.; Precup, C.: Model-Driven Telemetry and Analytics. *Cisco [online]*, 2019, [cit. 2020-11-07]. Dostupné z: `https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2019/pdf/BRKNMS-3537.pdf`

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

ASIC Application Specific Integrated Circuit

BGP Border Gateway Protocol

CESNET Czech Education and Scientific NETwork

CLI Command-line interface

CPU Central processing unit

gNMI gRPC Network Management Interface

gRPC Google Remote Procedure Calls

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISO/OSI International Organization for Standardization/Open Systems Interconnection model

JSON JavaScript Object Notation

KV-GPB Key-Value Google Protocol Buffers

MDT Model-Driven Telemetry

MIB Management information base

NETCONF Network Configuration Protocol

OID Object identifier

OSPF Open Shortest Path First

RAM Random-access memory

REST Representational state transfer

RFC Request for Comments

RIPE Réseaux IP Européens

RPC Remote procedure call

SGNP Simple Gateway Monitoring Protocol

SNMP Simple network management protocol

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

XML Extensible markup language

YANG Yet another next generation

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `config` ..... adresář obsahující configurace síťových zařízení
  - `install` ..... adresář obsahující soubory k instalaci
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF