FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Detekce a rozpoznávání periodické komunikace v síťovém provozu
Student:	Bc. Josef Koumar
Vedoucí:	Ing. Tomáš Čejka, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Nastudujte problematiku monitorování síťového provozu pomocí rozšířených síťových toků tzv. IP flows a seznamte se s metodami analýzy periodického průběhu časových řad. Zaměřte se na známé algoritmy pro zkoumání periodicity (periodogramy, Lomb-Scargle algoritmus, FFT a podobně).
S využitím existujících algoritmů navrhněte metodu analýzy časových řad vytvořených ze síťových toků pro detekci periodického chování komunikace včetně odhadu parametrů periodického chování a případně i rozpoznávání periodického typického chování známých aplikací a operačních systémů.
Vytvořte softwarový prototyp navržené metody analýzy provozu jako modul do open source systému NEMEA [1].
Otestujte vyvinutý prototyp a vyhodnoťte jeho použitelnost pro reálné nasazení a to zejména z pohledu potřebných výpočetních zdrojů a přesnosti výsledků.

[1] T. Cejka, et al., "NEMEA: A Framework for Network Traffic Analysis," in 12th International Conference on Network and Service Management, 2016.

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 14. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Detekce a rozpoznávání periodické komunikace v síťovém provozu

Bc. Josef Koumar

Katedra informační bezpečnosti
Vedoucí práce: Ing. Tomáš Čejka, Ph.D.

24. dubna 2022

Poděkování

Děkuji svému vedoucímu Ing. Tomášovi Čejkovi, Ph.D. a Ing. Karlovi Hynkovi za rady a vstřícnost. Dále děkuji sdružení CESNET, z. s. p. o. za podporu při výzkumu této práce. Nakonec bych rád poděkoval rodině a přátelům za podporu při psaní této práce. Zejména pak Tereze Ausficírové za rady ohledně gramatiky a stylizace českého jazyka a Bc. Marcelu Poláčkovi za pomoc při vytváření datových sad.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů 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 předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 24. dubna 2022 . . . . . .

Odkaz na tuto práci

Koumar, Josef. *Detekce a rozpoznávání periodické komunikace v síťovém provozu.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

V této práci se zaměřujeme na vytvoření metody detekce periodických chování v časových řadách ze síťového provozu, zejména šifrovaného, reprezentovaném pomocí síťových toků. Dále diskutujeme využití detekované periodicity k určení aplikace, služby či operačního systému, který daný provoz generoval.

Klíčová slova NEMEA, periodická komunikace, detekce periodicity, Lomb-Scargle periodogram, klasifikace dle periodicity, analýza síťového provozu, šifrovaný síťový provoz, CESNET, python

Abstract

In this thesis we focus on development of a method of detection periodic behaviour in time series from network traffic, especially encrypted, represented by network flows. We also discuss utilization of detected periodicity behaviour to get information about application, service or operating system that generate network traffic.

Keywords NEMEA, periodic communication, periodicity detection, Lomb-Scargle periodogram, classification by periodicity, network traffic analyze, encrypted network traffic, CESNET, python

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Analýza komunikace na síti je v dnešní době jedna z podstatných úloh každého monitorovacího systému a lze ji provádět různými způsoby podmíněnými konkrétním úkolem analýzy. V konkrétním případě bezpečnostní analýzy je cílem odhalit potencionálně škodlivou komunikaci, jako jsou neoprávněné přístupy k zařízením v síti. Dále může analýza podávat statistické informace o síti a provozu na ní nebo monitorování funkčnosti a vyhodnocování chyb způsobených konfigurací.

Monitorování sítě a následná analýza dat, stejně jako bezpečnost obecně, jsou prostředky pro minimalizaci rizik a hrozeb. Ve většině případů firmu motivuje k implementování monitorování sítě či bezpečnosti případná ztráta při bezpečnostních událostech nebo ztráty způsobené výpadky sítě. Dále díky monitorování sítě může firma sledovat přístupy k jednotlivým částem v síti a tím kontrolovat, zda fungují nasazené bezpečnostní politiky. To vše lze vyčíst z monitorování sítě.

Komunikace na síti můžeme rozdělit do skupin dle důležitosti dané komunikace. Obecně je to datová, řídící a správní rovina (angl. data, control and management plane). Datová rovina reprezentuje komunikaci vyvolanou uživatelem, systémem nebo zařízením za nějakým účelem. Do řídící roviny patří komunikace provádějící akce, které umožní úspěšný průběh komunikace z datové roviny. A nakonec komunikace, která má za účel změnu nastavení nebo monitorování síťových zařízení v síti, náleží do správní roviny. V této práci se věnujeme periodické komunikaci, která může náležet do jakékoliv z těchto tří skupin a v těchto skupinách má významnou roli.

Znalost periodické komunikace může být přínosem pro řadu aplikací, příkladem je predikce provozu nebo detekce anomálií. Detekce periodické komunikace je však velmi náročný úkol. Mezi příklady periodické komunikace na počítačové síti patří například zjišťování stavu, zda nedošlo ke změně v komunikačních aplikací. Takovou periodickou komunikaci využívají například Facebook Messenger či MS Teams. Dále mezi periodické komunikace patří síťové

služby, které přiřazují prostředky, monitorovací sondy odesílající nasbíraná síťová data a také mnohdy komunikace Command and Control (dále jen C&C) malware.

Jelikož spousta aplikací, služeb či dokonce malwaru má v sobě zakomponované procesy, které automaticky generují periodický provoz za nějakým účelem, tak je možné, že parametry těchto periodických chování na sebe prozrazují dost, aby byly dle těchto parametrů identifikovatelné. Tato myšlenka je v dnešní době velice zajímavá vzhledem k vysokému procentu komunikace, která probíhá šifrovaně. Tzn. identita procesu se schová za šifrování, ale periodou nám o sobě může vypovědět tolik, že dokážeme identitu získat, aniž bychom komunikaci dešifrovali. To se dá přirovnat k tzv. postrannímu kanálu, kterým ze systému nechtěně unikají informace.

V práci se budeme zabývat návrhem a implementací modulu detekce periodicity pro open-source monitorovací systém NEMEA [1] a tento modul následně otestujeme na reálné síti.

Kapitola 1

Cíl práce

Cílem této práce je vytvořit analyzér rozšířených síťových toků zvaných IP flows (popsaných např. v [2]), jehož cílem bude vytvoření a následná analýza časových řad za účelem detekování periodických komunikací v síti. Výstupem bude seznam komunikací, které jsou periodické s detekovanou periodou, a jejich vlastností vzhledem k periodicitě. Pro tento účel vytvoříme skupinu modulů pro již existující open-source monitorovací systém NEMEA [1].

Detekce periodické komunikace na časových řadách ze síťových toků musí při detekci pracovat s více parametry síťového toku současně (například počet paketů a počet bajtů). Tyto parametry dále nazýváme metrikami detekce periodicity. Zároveň by metoda měla být založena na známých matematických řešeních detekce periodicity na časových řadách, které v našem konkrétním případě budou dosahovat nejlepších možných výsledků.

Moduly budou schopny ze vstupních IP flows vytvořit pro každou komunikaci časovou řadu s více metrikami, o které rozhodnou, pomocí naší metody pro detekci periodicity, zda je periodická či nikoliv, a pro každou periodickou komunikaci získají informace o jejích vlastnostech. Mezi takové vlastnosti patří perioda síťových toků, časová perioda a hodnoty metrik v periodě.

Dalším cílem práce je zjištění, zda lze získané periodické vlastnosti použít pro detekování aplikace, či služby, která danou periodickou komunikaci generuje. Pokud bychom s vysokou pravděpodobností mohli pomocí periodického chování zjistit původce komunikace, tak bychom získali tuto informaci pouze na základě znalosti časů a hodnot metrik síťových toků, to znamená bez nutnosti analýzy aplikační vrstvy TCP/IP modelu. Takový druh analýzy je proto vhodný pro aplikaci na šifrované komunikaci.

Kapitola 2

Existující relevantní práce

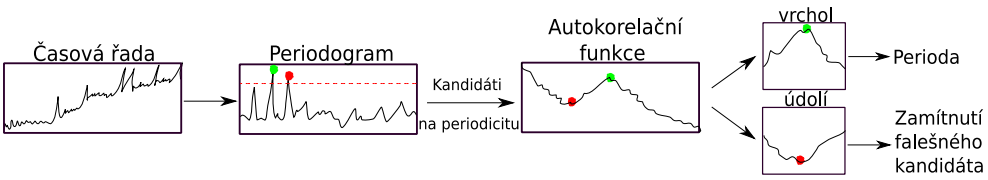
2.1 Detekce periodického chování

2.1.1 Autoperiod

Metoda Autoperiod představená v [3] je první metodou, která využívá informace v periodogramu i autokorelační funkci k poskytování přesných periodických odhadů bez převzorkování. Klíčovou myšlenkou je vytvořit klasický periodogram pro vstupní signál a zároveň kruhovou autokorelační funkci (oba termíny popsány v kapitole 4). Tomuto způsobu, kde se při detekci periodického chování pracuje s frekvenční i časovou doménou, se říká *hybridní přístup detekce periodicity*.

Pomocí statistických testů se nejdříve určí statisticky významné vrcholy periodogramu. Autoři pro tyto vrcholy předpokládají existenci možnosti, že vrcholy jsou falešnými příznaky (kvůli spektrálnímu úniku) nebo poskytují pouze hrubý odhad periody. Proto používají verifikaci významných vrcholů pomocí autokorelační funkce, díky které získají více jemnozrnný odhad potencionálních period.

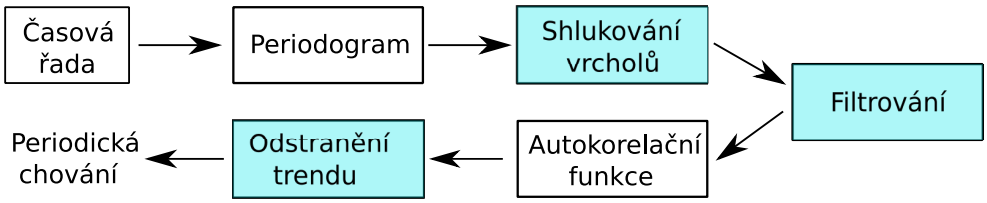
Při verifikaci vrcholu se kontroluje, zda perioda spočtená z jeho frekvence leží v autokorelační funkci na vrcholu. Pokud ano, je perioda označena za validní, jinak je považována za falešného kandidáta. Tato metoda je vyznačena na obrázku 2.1.



Obrázek 2.1: Diagram metody Autoperiod ([3], přeloženo)

2.1.2 CFD-Autoperiod

Metoda CFD-Autoperiod [4] je vylepšením metody Autoperiod z publikace [3]. Také přistupuje k detekci periodicity pomocí hybridního přístupu, ale navíc vylepšuje výsledky na silně zašuměných časových řadách a na časových řadách, obsahujících více periodických chování současně. K tomu se používají tyto techniky: shlukování (angl. clustering) významných vrcholů, filtrování časové řady a odstranění trendu z časové řady. Průběh metody CFD-Autoperiod je vyobrazena na obrázku 2.2.



Obrázek 2.2: Diagram metody CFD-Autoperiod ([4], přeloženo)

Shlukování statisticky významných vrcholů periodogramu počítá s tím, že nejvíce spektrálních úniků periodogramu se nachází kolem skutečných vrcholů způsobených přítomností periodického chování v časové řadě. Proto se všechny tyto vrcholy nekontrolují v Autokorelační funkci, kde by mohlo s nezanedbatelnou pravděpodobností dojít k propuštění spektrálního úniku jako další periodu k té skutečné. Místo toho se vytvoří *centroid* těchto vrcholů a předloží se jako kandidát na periodické chování.

Kandidáti na periodicitu jsou do Autokorelační funkce posílány v rostoucím pořadí a pokaždé, když je nějaký kandidát přijat Autokorelační funkcí, je pomocí filtrování odstraněn z časové řady. Po každém filtrování je vytvořena Autokorelační funkce nově vzniklé časové řady. Tímto způsobem lze odstranit nedostatek Autokorelační funkce, která je vlivem přítomnosti více periodických chování zašuměná a s vysokou pravděpodobností by nešlo detekovat všechny přítomné periody.

Odstraněním trendu v Autokorelační funkci autoři vyřešili problém, kdy není příliš velký rozdíl mezi vrcholem a údolím. Jakmile je trend odstraněn, tak jsou rozdíly mezi vrcholy a údolími výraznější a díky tomu se snáze detekují.

2.1.3 Sumarizační technika pro detekci periodického chování

Metoda sumarizace síťových toků [5] používá sumarizační techniky s cílem nalezení periodických chování v komunikaci.

Klíčovou myšlenkou je fakt, že periodická komunikace vykazuje velmi nízký rozptyl a standardní odchylku rozdílů časů sousedících síťových toků. A zároveň běžná komunikace generovaná náhodným procesem (například uživatelem) vykazuje velmi velký rozptyl.

Metoda nejdříve rozdělí provoz na komunikace mezi unikátními IP adresami (dvojice zdrojová a cílová IP adresa). Následně se vytvoří posloupnost rozdílů časů síťových toků pro sledovaný náhodný proces X (dvojici IP adres), tedy $dt_i = t_i - t_{i-1}$ pro $i := 2, 3, ..., n$. A spočte se standardní odchylka pomocí:

$$SD = \sqrt{\left(\frac{1}{m-1}\sum_{i=1}^{m}(dt_i - \mu)\right)}$$

kde $m = n-1$ je počet prvků vytvořené posloupnosti rozdílů časů a μ je jejich průměr.

S touto jednoduchou metodou dosahovali autoři v experimentech poměrně vysokého počtu správných vyhodnocení.

2.1.4 Segmentační metoda

Segmentační metoda byla vytvořena v diplomové práci [6] na Matematickém institutu univerzity v Leidenu. Tato metoda na rozdíl od předchozích počítá s tím, že časová řada je složena z datových bodů, které mají více dimenzí. To znamená, že jeden datový bod obsahuje kromě časové informace více než jednu hodnotu. Příkladem může být počet paketů a počet bajtů jako dvě dimenze takového datového bodu.

Metoda rozděluje časovou řadu do částí stejné velikosti, tzv. segmentů. Tyto segmenty poté mohou být porovnávány, a pokud jsou si podobné (s nějakou prahovou hodnotou), tak je časová řada považována za periodickou s periodou rovnou velikosti segmentu. K porovnávání používají vlastní navržené skóre periodicity:

$$S(V,p) = \sqrt{\frac{1}{k}\sum_{l=0}^{k-1}\sum_{j=0}^{p-1}\frac{1}{m_j - 1}\sum_{i=0}^{m_j-1}\frac{v_{ip+j,l} - \overline{v}_{j,l}}{\overline{v}_{,l}}}$$

kde V je časová řada obsahující n datových bodů $v_{i,j}$, které mají k dimenzí, p je testovaná perioda (velikost segmentu), $\overline{v}_{j,l}$ je průměr $v_{ip+j,l}$ přes $i := 0, ..., m_j - 1$ a $\overline{v}_{,l}$ je průměr $v_{i,j}$ přes $i := 0, ..., n-1$

Největší slabinou metody je to, že musíme rozdělit časovou řadu do segmentů nějaké konstantní velikosti, tzn. musíme dopředu odhadnout, jakou periodu chceme testovat. Autor popisuje možnost, jak jsou nejčastější periody (1 až 5) otestovány. Jako možnost uvádí použít Autokorelační funkci k odhadu kandidátů na periodicitu, kteří pak definují velikost segmentů k otestování.

2.2 Detekce aplikací, služeb a operačních systémů pomocí periodických chování

V práci [7], která se předně zabývá detekci anomálií pomocí detekovaných periodických chování, se popisuje vznik periodických chování na síťové komunikaci. Veškerá periodická komunikace se děje nezávisle na uživateli, to znamená, že je vygenerovaná nějakým předdefinovaným procesem. Příklady takových legitimních procesů může být kontrola zda neexistuje nová verze aplikace, kontrola, zda se nezměnil nějaký sledovaný stav ovlivnitelný z více stran (nejvíce u komunikačních aplikacích) a například synchronizace databáze s hlavním serverem. Zároveň existují i nelegitimní procesy, které vytvářejí periodickou komunikaci, jako jsou například keyloggers, spyware, C&C a adware. Proto je pro administrátora velice důležité, když jsou tyto periodické komunikace detekovány.

Autorům se podařilo odhalit periodu u BitTorrentu, která je způsobená běžnými kontrolními zprávami, u RSS zpráv, u aktualizací operačních systémů, u keyloggerů, u nejrůznějších aplikací (například počasí), u Peer-to-peer protokolů, u adware, u C&C provozu botnetu, a dokonce i na některých webech detekovali periodicitu způsobenou počítadly, které počítají, jak dlouho se uživatel na stránce zdržel.

Práce [8] se zaměřuje na detekci škodlivých zařízení pomocí periodické komunikace. Jejich experimenty ukázaly, že periodicita velice souvisí se škodlivou činností a lze jí snadno integrovat do detekčních systémů. Cílem práce nebylo detekovat botnet, ale přímo aktivity, které škodlivá zařízení provádí, nehledě na to, o jaké činnosti jde, a vytvořit tak „graylist“ externích zařízení, u kterých je vysoká pravděpodobnost, že se jedná o škodlivé zařízení. K vytváření „graylistu“ se používají hlášení z NIDS (Network Intrusion Detection Systems), která jsou označena jako škodlivá. Díky tomu jsou schopni získat škodlivá periodická chování, která reprezentují pomocí shluků (angl. clusters) sjednocujících podobná periodická chování.

Kapitola 3

Časové řady ze síťového provozu

Časová řada je chronologicky uspořádané pozorování hodnot nějaké náhodné veličiny. To znamená, že časová řada je posloupnost hodnot vygenerovaných nějakým procesem, pro které platí, že mají přiřazen nějaký čas na časové ose. Pokud platí, že časová informace hodnot má tvar $t := 1, 2, ..., n$, pak mluvíme o *rovnoměrně rozložené časové řadě*. Jinak mluvíme o *nerovnoměrně rozložené časové řadě* a platí, že $t_i \leq t_{i+1}$ pro $i := 1, 2, ..., n$.

3.1 Tvorba časových řad ze síťového provozu

Při tvorbě časové řady ze síťového provozu (resp. ze síťových toků) musíme nejdříve provoz rozdělit na jednotlivé komunikace generované nějakými procesy, aplikacemi či službami. Vhodné je proto rozdělení na *síťové závislosti*.

Síťová závislost je vztah mezi IP adresou (klient) a službou na jiné IP adrese (server), tedy dlouhodobý provoz mezi dvěma IP adresami pod nějakým (většinou registrovaným) portem transportní vrstvy. Příklady takových závislostí můžeme vidět v tabulce 3.1.

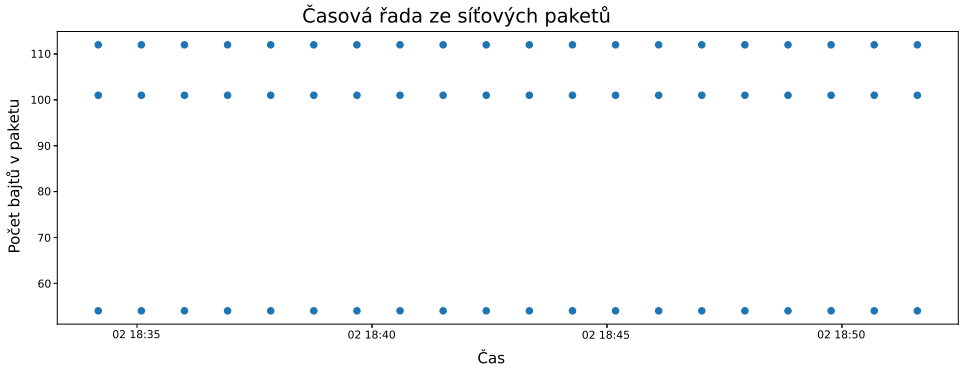
Tabulka 3.1: Příklady síťových závislostí

Zdrojová IP adresa	Cílová IP adresa	Port transportní vrstvy
192.168.0.10	192.168.0.1	53
192.168.0.10	93.25.14.36	443
192.168.0.11	192.168.0.1	53
192.168.0.11	192.168.0.2	21
192.168.0.11	25.142.154.2	50124 – 49012

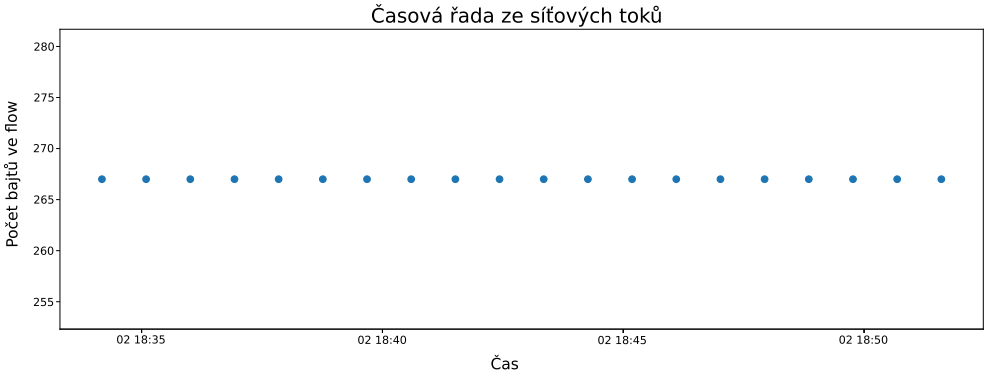
Rozdělení pomocí síťových závislostí nemusí vždy nutně rozdělit provoz na komunikace generované nějakými procesy. Může se stát, že jedna závislost je využívaná více procesy zároveň. I přes tento fakt je to nejlepší způsob jak komunikaci triviálně rozdělit na jednotlivé časové řady.

Rozdíl mezi časovými řadami vytvořené přímo ze síťového provozu, tzn. datové body, které jsou reprezentovány pakety, a ze síťových toků, tzn. datové body, které jsou reprezentovány flow záznamy, je v tom, že více bodů časové řady je reprezentováno jedním bodem. Periodicita tak bude reprezentována menším počtem bodů a její detekce bude vyžadovat méně výpočetního času. V časové řadě ze síťových toků je navíc v datovém bodu důležitá metrika počtu paketů ve flow záznamu.

Příklad časové řady vytvořené z paketů je na obrázku 3.1. Její perioda je zřetelná, opakuje se pravidelně 112, 101 a 54 bajtů s časovou periodou 55 sekund. Stejný provoz převedený na síťové toky je použit k vytvoření časové řady na obrázku 3.2. Perioda je opět zřetelná, opakuje se v ní pravidelně 267 bajtů s časovou periodou 55 sekund. Zatímco hodnoty a délka opakující se sekvence se převedením na síťové toky změní, časová perioda zůstává stejná.



Obrázek 3.1: Příklad časové řady ze síťových paketů



Obrázek 3.2: Příklad časové řady ze síťových toků

3.2 Volba typu časové řady

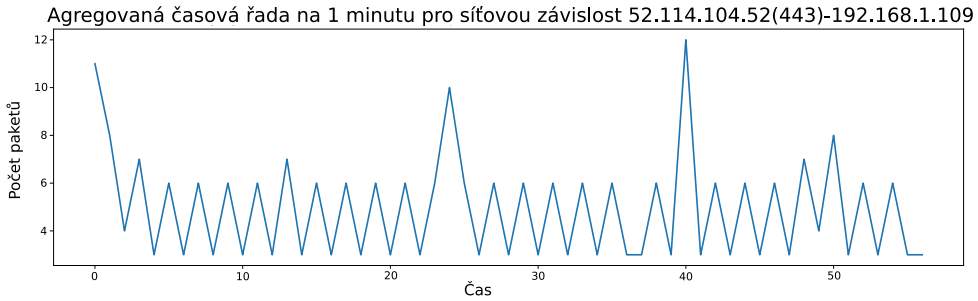
Po rozdělení provozu na síťové závislosti máme dvě možnosti jak postupovat. Buď zvolíme časový interval (například 5 minut) a podle něj agregujeme všechny flow záznamy v časovém intervalu do jednoho datového bodu časové řady. Tím vytvoříme *rovnoměrně rozloženou časovou řadu*. Nebo každý flow záznam vložíme do časové řady jako její datový bod s časovou informací rovnou času přenosu flow záznamu (prvního nebo posledního paketu z něj). Tím vytvoříme (ve většině případech) *nerovnoměrně rozloženou časovou řadu*.

Rozdíly v těchto dvou možnostech jsou pro detekci periodického chování zásadní. V tabulce 3.2 jsme sepsali výhody a nevýhody časových řad vytvořených agregací na nějaký časový interval. Metodami pro detekci periodicity na takových časových řadách jsou například Periodogram, Autokorelační funkce či Power spektrum.

Tabulka 3.2: Výhody a nevýhody agregovaných časových řad ze síťového provozu

Agregované časové řady ze síťového provozu	
Výhody	**Nevýhody**
• Většinou méně hodnot v časové řadě • Všechny časové řady naměřené ve stejném období mají stejný počet hodnot • Metody detekce periodicity nemusí pracovat s časem hodnot časové řady • Lze použít větší škálu metod, jelikož většina metod pro detekci periodicity je vytvořená pro rovnoměrně rozložené časové řady	• Agregací na časové intervaly může snadno dojít ke ztrátě informace o skutečné periodě • Reprezentace prázdných časových intervalů je 0, což může mít negativní vliv na metody detekce periodicity • Volba agregačního intervalu (nemožné, aby byl pro všechny časové řady ideální)

Na obrázku 3.3 je vyobrazen příklad agregované časové řady, která je agregovaná na jednu minutu. Jelikož se jedná o rovnoměrně rozloženou časovou řadu, tak je vyobrazena jako spojnicový graf. Všimněte si, že se až na výjimky střídají v počtu paketů dvě hodnoty, a to šest a nula, kde nula reprezentuje prázdný interval (v dané minutě nebyl přenesen žádný flow záznam).



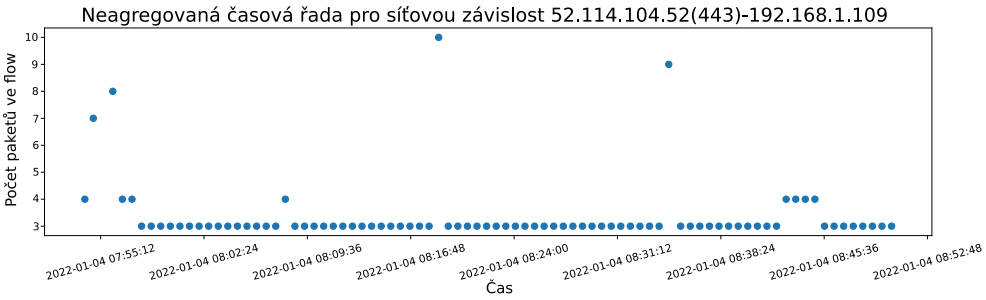
Obrázek 3.3: Příklad agregované časové řady

V tabulce 3.3 jsme sepsali výhody a nevýhody časových řad vytvořených vkládáním flow záznamů do časové řady spolu s časem přenosu daného flow. Metodou pro detekci periodicity na takových časových řadách je například Lomb-Scargle periodogram.

Tabulka 3.3: Výhody a nevýhody neagregovaných časových řad ze síťového provozu

Neagregované časové řady ze síťového provozu	
Výhody	**Nevýhody**
• Nedochází k žádné ztrátě informací • Výstupem metod je skutečná perioda flow záznamů a časová perioda, nikoliv perioda agregovaných intervalů • Hodnoty na časové řadě reprezentují pouze flow záznamy • Není třeba volit agregační interval	• Metody detekce periodicity musí pracovat s časem hodnot časové řady • Existuje menší škála metod pro detekci periodicity na nerovnoměrně rozložených časových řadách • Většinou více hodnot na časové řadě

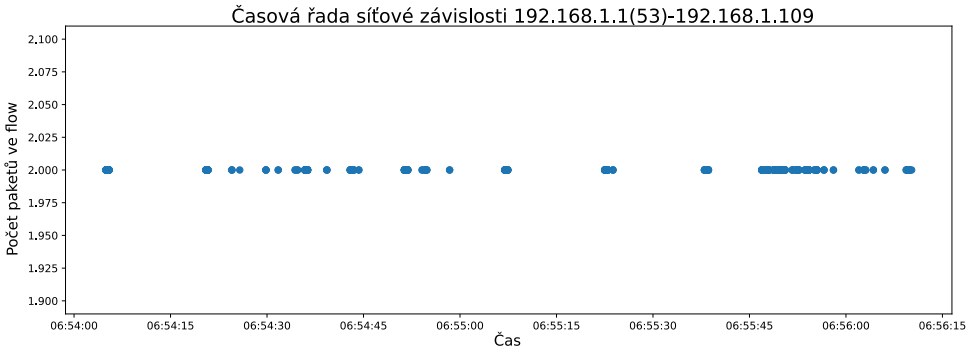
Na obrázku 3.4 je vyobrazen příklad neagregované časové řady. Jelikož se jedná o nerovnoměrně rozloženou časovou řadu, tak je vyobrazena jako bodový graf. Zde stojí za povšimnutí, že až na výjimky je počet paketů ve flow roven třem, takže při detekci na agregované časové řadě z předchozího příkladu by byla detekována chybná perioda a chybné hodnoty periodicity.



Obrázek 3.4: Příklad neagregované časové řady

Graf na obrázku 3.4 je dobrý příklad toho, jak vypadají časové řady ze síťového provozu. Z povahy síťových toků jsou hodnoty na časových řadách pro používané metriky, jako je počet paketů a počet bajtů ve flow záznamu, přirozená čísla. Pokud budeme definovat periodickou komunikaci jako periodu opakování flow záznamů, kde například perioda s opakující se sekvencí dvou datových bodů znamená, že se opakují dva rozdílné flow záznamy, tak úkol detekce periodicity je poměrně odlišný od časových řad z jiných oblastí. U většiny řad ze síťových toků, které lze považovat za periodické, se v drtivé většině opakuje od 1 do 3 flow záznamů na časové řadě, což znamená, že nejčastější periody u těchto časových řad jsou 1, 2 a 3. Je nutné poznamenat, že nejvíce se objevuje perioda jednoho flow záznamu.

Navíc šum je zde definovaný jako změna hodnoty nějaké metriky ve flow záznamu, například změna počtu paketů nebo bajtů, nebo změna času přenosu flow záznamu o nějaký čas, například vlivem zahlcení linky. Příklad šumu v čase na časové řadě ze síťového provozu můžeme vidět na obrázku 3.5.



Obrázek 3.5: Příklad mezer v neagregované časové řadě

Agregování časové řady ze síťového provozu by dokázalo při správně nastaveném agregačním intervalu zmenšit dopady šumu v časové ose, ale zároveň by to rozšiřovalo vliv šumu v hodnotách metrik flow záznamu. Příklad takového dopadu můžeme vidět i na časové řadě vyobrazené na obrázku 3.3, kde některé vrcholy jsou vyšší kvůli šumu v původní neagregované časové řadě 3.4. Tím

je myšleno, že pokud je přítomen nepravidelný šum, který například zvětšuje počet bajtů byť o jeden, tak agregací s tímto šumem ovlivníme všechny ostatní flow záznamy v agregovaném intervalu a již nebude platit tvrzení, že perioda p je perioda opakování p různých flow záznamů a problém odhalení periody bude složitější.

Na základě těchto zjištění jsme zvolili při detekci periodického chování na síťové komunikaci reprezentované pomocí síťových toků pracovat s nerovnoměrně rozloženými časovými řadami.

3.3 Analýza časových řad

Časová řada je množina pozorování x_i, pro které platí, že každé z nich bylo pozorováno v nějakém specifickém čase $t \in T$. Je přirozené předpokládat, že pozorování x_t je realizace nějaké náhodné veličiny X_t. Časová řada $\{x_t|t \in T\}$ je potom realizace náhodných veličin $\{X_t|t \in T\}$. To vede k úvaze, že se jedná o realizaci stochastického procesu $\{X_t|t \in T\}$ [9].

Stochastický proces jsou náhodné veličiny $\{X_t|t \in T\}$ definované jako pravděpodobnostní prostor $(\Omega, \mathcal{F}, \mathcal{P})$, kde Ω je prostor elementárních jevů (neprázdná množina možných výsledků), $\mathcal{F}$ je množina náhodných jevů (výsledky náhodného pokusu) a funkce $\mathcal{P}$ přiřazuje každému náhodnému jevu jeho pravděpodobnost.

Časová řada může být potom reprezentována jako proces definovaný jako:

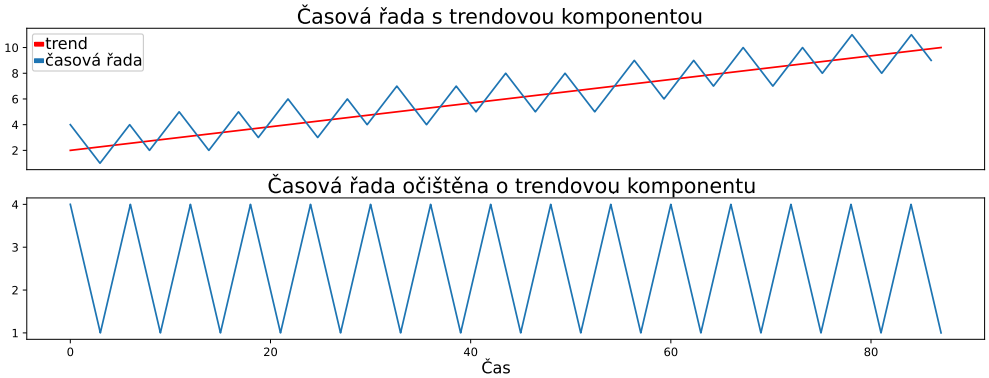
$$X_t = T_t + S_t + Y_t$$

kde T_t je pomalu měnící se funkce známá jako komponenta trendu, S_t je funkce se známou periodou p nazývanou komponenta sezónnosti a Y_t je komponenta náhodného šumu, která je stacionární [9].

Cílem většiny analýz časových řad je vytvoření modelu, který nejlépe reprezentuje časovou řadu. Tzn. odhalení T_t a S_t, které jsou deterministické, a popsání náhodného šumu Y_t tak, aby jeho vliv mohl být zahrnut do předpovědí modelu. Cílem této práce je zisk periody p z komponenty S_t.

Analýza časových řad vždy začíná odhalením a následném odstranění komponenty trendu T_t, jelikož pokud je trendová komponenta přítomna, není možné získat s vysokou spolehlivostí periodu p sezónní komponenty S_t. Vliv trendu na sezónní komponentu můžeme vidět na příkladu časové řady vyobrazeném na obrázku 3.6, kde první graf znázorňuje časovou řadu, ve které je přítomná komponenta trendu T_t, a druhý graf znázorňuje časovou řadu po odstranění trendové komponenty.

Všimněte si, že periodu p je po odstranění trendové komponenty snadné odhalit. Proto všechny techniky detekce periodického chování na časových řadách předpokládají, že časová řada je nejdříve očištěna od trendové komponenty.



Obrázek 3.6: Příklad časové řady $X_t = T_t + S_t + Y_t$

Hodnoty na časové řadě ze síťového provozu jsou vždy generovány deterministickým procesem. To znamená, že deterministický proces vytvoří komunikaci, kterou reprezentuje flow záznam. Časová řada je tedy soubor deterministických spojení generovaných procesem, který je spuštěn buď přímo uživatelem nebo programem. Periodická časová řada je soubor opakovaných deterministických spojení, které vygeneroval proces v drtivé většině případů nezávislý na uživateli.

Pokud se tedy proces, který generuje periodickou časovou řadu ze síťového provozu, chová deterministicky, potom lze předpokládat, že tato časová řada může být reprezentována procesem definovaným jako:

$$X_t = \sum_{i=1}^{n} S_{t_i} + Y_t$$

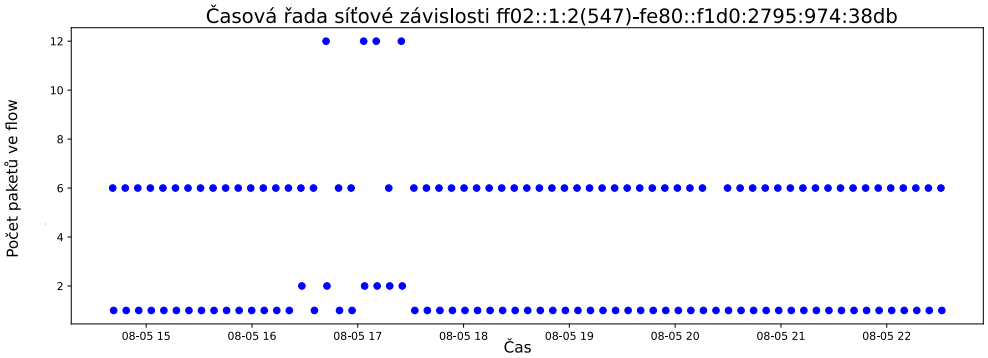
kde n je počet periodických procesů v časové řadě, $i \in \{1, ..., n\}$, S_{t_i} je i-tá komponenta sezónnosti, která má nějakou periodu p_i, Y_t je komponenta šumu, který je stacionární, a $+$ je operátor slučující datové body jednotlivých komponent do jedné časové řady. Přítomnost více sezónních komponent je způsobena tím, že rozdělení provozu na síťové závislosti nemusí vždy rozdělit provoz na části generované jediným procesem. Což značí, že existuje možnost, že dva či více různých procesů komunikuje pod stejným portem transportní vrstvy se stejnou IP adresou. Absence trendové komponenty je způsobena povahou periodických časových řad ze síťového provozu, ve kterých nikdy není trendová komponenta přítomna.

Pokud by nastal podobný trend, jako je vyobrazen na obrázku 3.6, tak ve spojeních dochází k trvalému zvyšování počtu paketů, respektive počtu bajtů. Jenže to by už nebylo periodické chování. Navíc pokud by byl předpoklad, že je v takové časové řadě přítomná trendová komponenta a ta by byla odstraněna, tak by jako výsledek detekce periodicity vyšla smyšlená perioda. Z toho plyne, že veškeré výkyvy v časových řadách ze síťového provozu, které se na první pohled mohou zdát jako trend, jsou ve skutečnosti šumem.

Šum v periodických časových řadách ze síťového provozu může mít tři různé důvody. Může být způsobený tím, že přenášená periodická událost v sobě obsahuje nějakou mírně se měnící informaci, která se projeví v počtu bajtů, nebo dokonce v počtu paketů, pokud změna v paketu přesáhne maximální velikost bajtů (obvykle 1500 bajtů). Dalším důvodem je časové posunutí jinak periodických datových bodů. A posledním důvodem jsou datové body, které generuje jiný neperiodický proces komunikující pod stejným portem transportní vrstvy se stejnou IP adresou.

Z toho vyplývá, že pokud je rozdělení na síťové závislosti dokonalé, tak v časové řadě je provoz jednoho procesu. Pokud je provoz periodický, tak je časová řada definovaná jako $X_t = S_t + Y_t$, pokud je neperiodický, pak je definována jako $X_t = Y_t$.

Na obrázku 3.7 můžeme vidět časovou řadu ze síťového provozu reprezentující provoz na specifické síťové závislosti. Všimněme si, že nemůžeme z dostupných informací rozhodnout, zda časovou řadu generuje jeden proces či dva či dokonce více než dva procesy. Pokud by ji generoval jeden proces, pak je perioda opakování dvou flow záznamů a opakuje se počet paketů 1 a 6. Pokud by byly procesy dva, tak jsou přítomny dvě nezávislé periody. Obě jsou periody opakování jednoho flow záznamu. První má opakující se počet paketů roven 1 a druhá 6. Potenciálně se vyskytuje i proces generující náhodný šum.



Obrázek 3.7: Příklad časové řady $X_t = \sum_{i=1}^{n} S_{t_i} + Y_t$

Dalším krokem analýzy časových řad je odhalení periody p. Nejznámější metodou je *Periodogram* [10] nebo varianty z něj odvozené (*Laplaceův*, *Bartlettův*, *Welchův*, či *Lomb-Scargleův periodogram*).

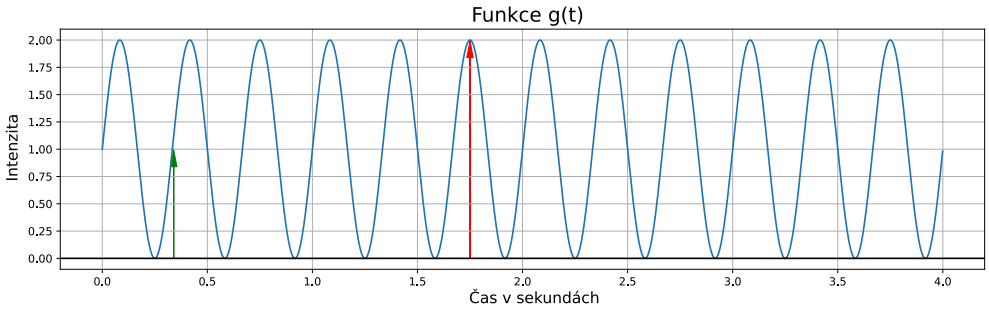
Po získání periody p, respektive period p_i pro $i \in \{1, ..., n\}$, lze odvodit celkový model X_t, pomocí kterého je možno například předpovídat vývoj časové řady v budoucnu, detekovat anomálie či hledat podobnosti mezi modely různých časových řad.

Kapitola 4

Matematický základ

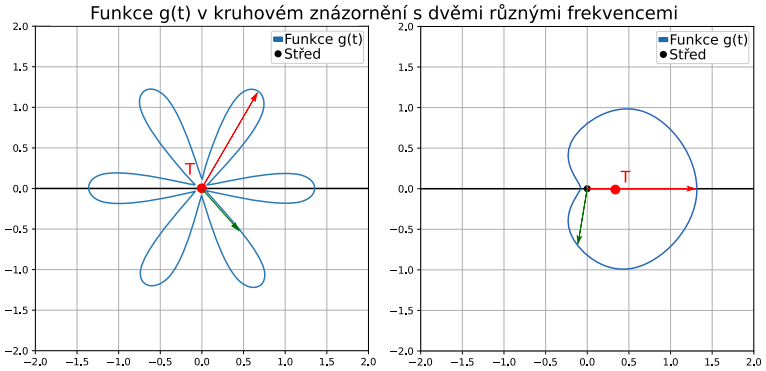
4.1 Fourierova transformace

Mějme kontinuální signál reprezentovaný funkcí $g(t)$, kde t je čas. Tato funkce je vyobrazená na obrázku 4.1.



Obrázek 4.1: Funkce g(t)

Klíčovou myšlenkou Fourierovy transformace je vzít graf funkce $g(t)$ a obtočit jej kolem bodu do kruhu jak je znázorněno na obrázku 4.2.



Obrázek 4.2: Funkce g(t) obtočená kolem bodu do kruhu pro dvě frekvence

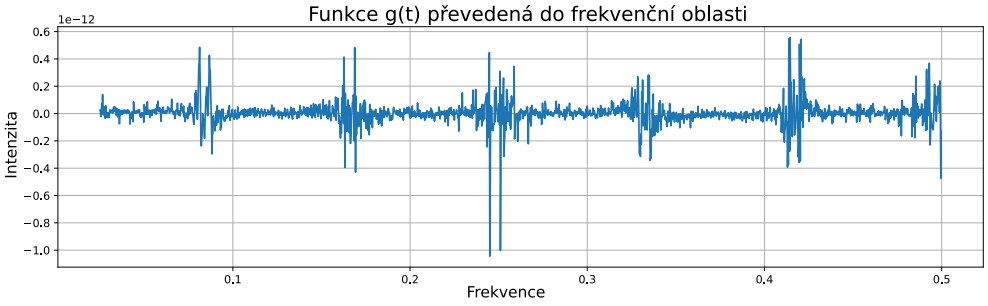
Tento bod nazveme středem a označíme ho S. Funkce je obtočená kolem bodu s nějakou frekvencí, která definuje, jak veliký časový úsek funkce $g(t)$ se vejde do jednoho otočení o 360°.

Poté si představíme vektor $v(t)$, jehož výška je rovna intenzitě funkce $g(t)$ v čase t. Na obrázku 4.1 jsou dva příklady vektorů. Jeden zelený v čase cca 0.34 sekund a jeden červený v čase cca 1.75 sekund. Stejné vektory jsou vyobrazené i v kruhovém znázornění funkce $g(t)$ na obrázku 4.2. Přičemž na kruhovém znázornění funkce $g(t)$ délky největších vektorů $v(t)$ korespondují s největšími vzdálenostmi od středu kruhu S. Takže když si vezmeme vektor $v(t)$ a pohybujeme v čase po funkci $g(t)$, tak se na obrázku 4.2 neustále točíme v kruzích.

V této chvíli máme tedy dvě frekvence. První je frekvence funkce $g(t)$ a druhá je frekvence otočení o 360° v grafu zobrazeném na obrázku 4.2. A platí, že první frekvence je pevně daná podle funkce $g(t)$, zatímco druhou frekvenci si můžeme zvolit dle libosti. V závislosti na volbě druhé frekvence se bude měnit podoba funkce v grafu.

Nyní do kruhového znázornění funkce $g(t)$ přidáme bod T reprezentující těžiště. V levém grafu na obrázku 4.2 je těžiště T téměř shodné se středem S a v pravém je nejdál od středu jak může být. Se změnou frekvence otáčení se těžiště mění. Platí, že pro většinu frekvencí se těžiště pohybuje poblíž středu S. Ale jakmile se frekvence otáčení přibližuje frekvenci funkce $g(t)$, tak se těžiště začne vychylovat od středu, přičemž jakmile se frekvence rovnají, je vzdálenost mezi T a S maximální.

Pokud budeme sledovat vývoj vzdálenosti mezi T a S pro různé frekvence otáčení, tak získáme graf zobrazený na obrázku 4.3. A výsledkem je, že jsme převedli $g(t)$ z časové oblasti do frekvenční oblasti. Tento jev se dá stejným způsobem otočit a získat z grafu na obrázku 4.3 opět funkci $g(t)$ znázorněnou na obrázku 4.1.



Obrázek 4.3: Funkce g(t) převedená do frekvenční oblasti

Když tento proces zapíšeme matematickým zápisem, tak na diskrétním čase definujeme Fourierovu transformaci pro posloupnost hodnot definovanou funkcí $g(t)$, kde $t := t_0, t_1, ..., t_n$, jako:

$$\mathcal{F}\{g\} = \sum_{k=1}^{n} g(t_k) e^{-2\pi i f t}$$

Pomocí druhé mocniny *Fourierovi transformace* je známá jako *výkonová spektrální hustota* (angl. power spectral density, zkráceně PSD) nebo *výkonové spektrum* (angl. power spectrum):

$$\mathcal{P}_g = |\mathcal{F}\{g\}|^2$$

PSD je funkce reálných hodnot frekvencí f, která kvalifikuje příspěvek každé frekvence f do celkového signálu. Takže pomocí *PSD* lze získat dominantní frekvence v signálu.

4.2 Autokorelační funkce

Autokorelační funkce (angl. Autocorrelation function, zkráceně *ACF*) je matematická reprezentace stupně podobnosti mezi časovou řadou a její verzí posunutou o nějaký časový interval. Koncept je shodný s konceptem korelační funkce mezi dvěma časovými řadami, ale s rozdílem, že druhá časová řada je první posunutá o nějaký čas.

Analýzu pomocí Autokorelační funkce provádíme za účelem získání periodických vzorů v časové řadě nebo za účelem identifikování fundamentálních frekvencí v signálu implikovaným jeho harmonickými frekvencemi.

Diskrétní Autokorelační funkce funkce $g(t)$ pro posunutí o zpoždění l definujeme jako:

$$R(l) = \frac{\sum_{k=l}^{n} (g(t_k) - \overline{g(t)})(g(t_{k-l}) - \overline{g(t)})}{\sum_{k=1}^{n} (g(t_k) - \overline{g(t)})}$$

kde $\overline{g(t)}$ je průměr hodnot funkce $g(t)$. Lze definovat i jako kovariance časové řady a jejího posunutí děleno variance časové řady:

$$R(l) = \frac{Cov(g(t_k), g(t_{k-l}))}{Var(g(t_k))}$$

Na obrázku 4.4 je vyobrazena funkce $g(t)$, která je vstupem do autokorelační funkce přes všechna možná zpoždění jak je znázorněno na obrázku 4.5.

Lze si všimnout, že funkce $g(t)$ je zašuměná sinusoida a autokorelační funkce tuto sinusoidu odhaluje.

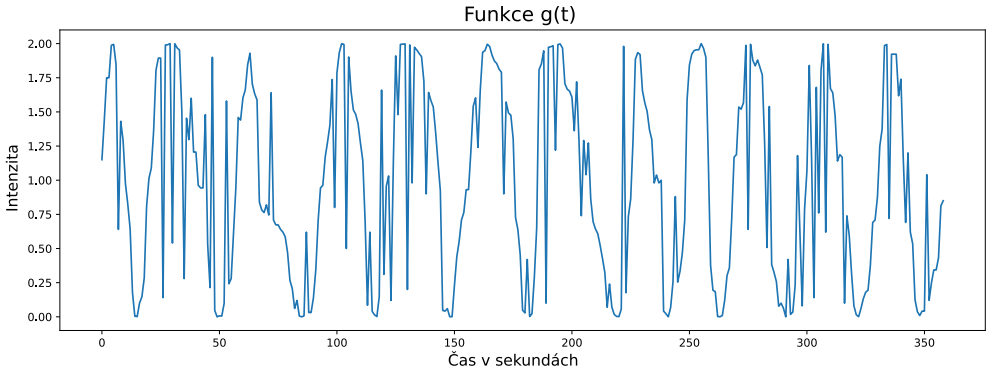
Kruhová autokorelační funkce se oproti klasické liší v tom, že o zpoždění l uděláme cyklický posun. Běžná autokorelace má s narůstajícím zpožděním

čím dál tím menší počet prvků, které vzájemně koreluje (tento jev je dobře viditelný i na obrázku 4.5), zatímco kruhová autokorelace koreluje pro každé zpoždění stejný počet prvků.

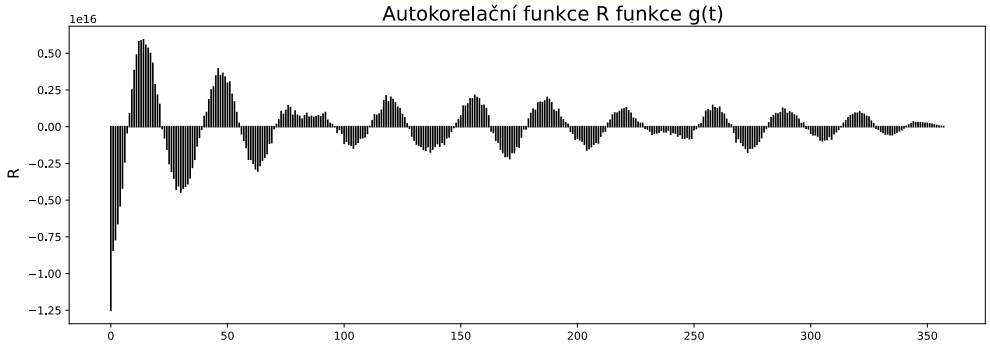
$$R(l) = \frac{\sum_{k=1}^{n}(g(t_k) - \overline{g(t)})(g'(t_k, l) - \overline{g(t)})}{\sum_{k=1}^{n}(g(t_k) - \overline{g(t)})}$$

kde $g'(t, l)$ je funkce, jejíž návratová hodnota vrací hodnoty $g(t)$ cyklicky posunuté o zpoždění l. Funkci, která vrací posloupnost hodnot kruhové autokorelační funkce pro zpoždění $l_i := i$, kde $i \in \{1, ..., n-1\}$, označíme ACF.

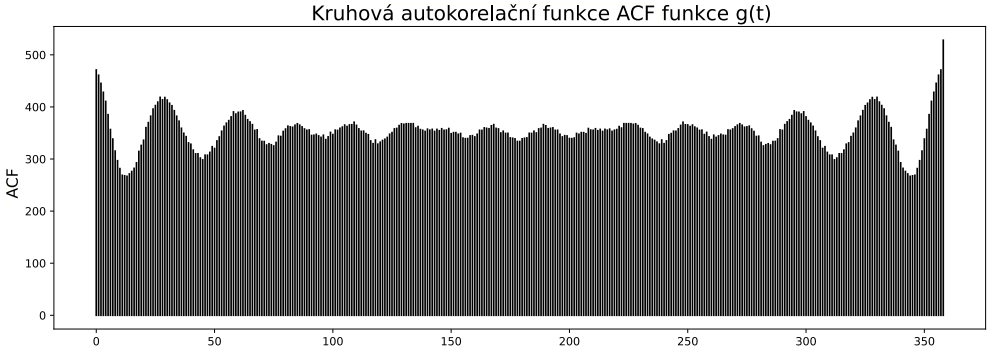
Kruhová autokorelační funkce aplikovaná na funkci $g(t)$ vyobrazenou na obrázku 4.4 je na obrázku 4.6.



Obrázek 4.4: Funkce g(t)



Obrázek 4.5: Autokorelační funkce aplikovaná na funkce g(t)



Obrázek 4.6: Kruhová autokorelační funkce aplikovaná na funkci g(t))

Efektivního způsobu výpočtu ACF lze dosáhnout vhodným použitím Fourierovy transformace, respektive její efektivní implementace nazývané *Fast Fourier Transformation* (zkráceně FFT):

$$ACF(g(t)) = \mathcal{F}^{-1}\left\{\mathcal{F}\left\{g(t)\right\}\mathcal{F}'\left\{g(t)\right\}\right\}$$

kde $\mathcal{F}'\left\{g(t)\right\} = \mathcal{F}\left\{g'(t)\right\}$, a $g'(t)$ je posloupnost hodnot funkce $g(t)$ v opačném pořadí.

4.3 Periodogram

Periodogram je metoda pro odhad výkonnostního spektra náhodného signálu z jeho vzorků. Termín periodogram byl poprvé definován Arturem Schusterem v publikaci [10].

Klíčovou myšlenkou je, že každá časová řada může být vyjádřená pomocí kombinace kosinusových a sinusových vln s různými periodami a amplitudami. Tento fakt může být použit pro analýzu periodických chování v časové řadě. Časovou řadu tedy můžeme vyjádřit ve tvaru:

$$X_t = \mu + \sum_{j=1}^{p} (\alpha_j cos(\omega_j t) + \beta_j sin(\omega_j t)) + \epsilon_t$$

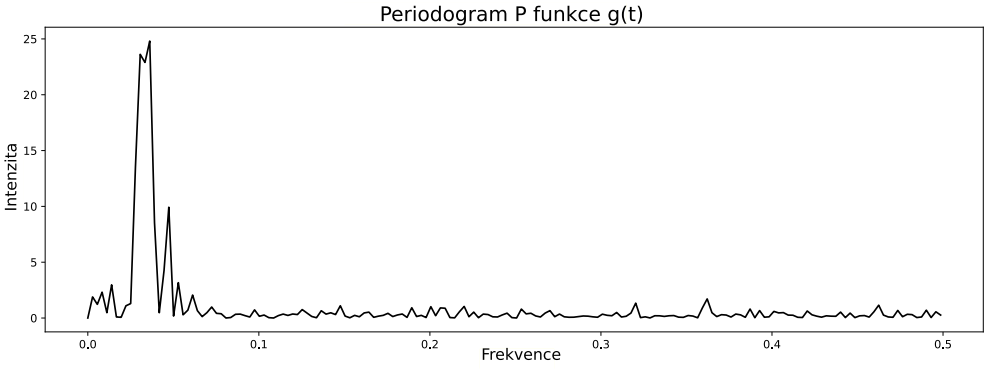
kde X_t je náhodná veličina, kterou analyzujeme, $t := 1, 2, ..., n$, $\omega_1, \omega_2, ..., \omega_p$ jsou od sebe různé frekvence, ϵ_t je bílý šum a $\mu, \alpha_j, \beta_j \in \mathbb{R}$ jsou koeficienty. Všimněte si, že model nepředpokládá žádný trend. Pokud by časová řada obsahovala trendovou složku, je nutné ji nejdříve odstranit. Hodnota p je volena jako $\lfloor \frac{n}{2} \rfloor$, protože delší cyklus není možné v časové řadě pozorovat. Tato nejvyšší možná pozorovatelná frekvence se nazývá *Nyquistova frekvence*.

Cílem analýzy je nalézt hodnoty ω_j, které jsou v časové řadě skutečně významné. A právě k tomuto účelu slouží periodogram, který můžeme zapsat jako:

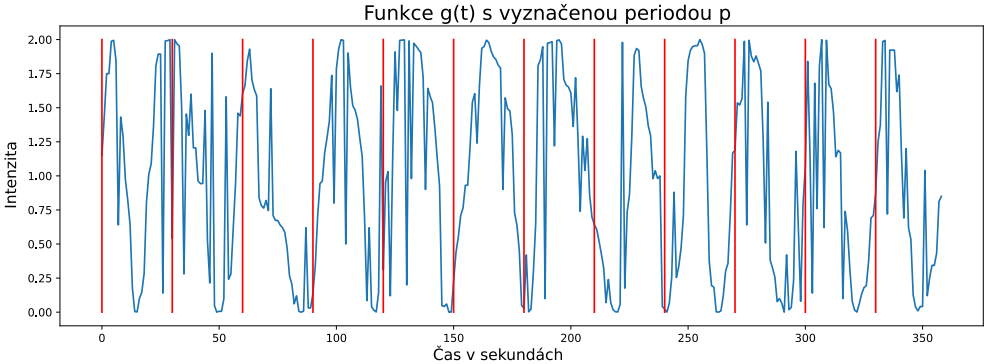
$$P(\omega) = \frac{1}{2\pi n}\left(\left(\sum_{t=1}^{n} x_t cos(\omega t)\right)^2 + \left(\sum_{t=1}^{n} x_t sin(\omega t)\right)^2\right) \text{ pro } -\pi \leq \omega \leq \pi$$

kde x_t jsou hodnoty na časové řadě. Periodogram časové řady jsou hodnoty pro každé ω v definovaném rozsahu frekvencí (například mezi *0* a *0.5*). Příklad takového periodogramu pro časovou řadu definovanou funkcí $g(t)$ z grafu znázorněném na obrázku 4.4, která je zašuměná sinusoida, je vyobrazen na obrázku 4.7.

Můžeme si všimnout, že maximální vrchol je okolo frekvence *0,0334*. Z té si můžeme odvodit periodu opakování pomocí rovnice $p = \frac{1}{f}$, kde p je perioda a f je frekvence. Výsledná perioda se rovná přibližně *30*. Na obrázku 4.8 je zobrazena funkce $g(t)$ s naivně vyznačenými hranicemi dle výsledné periody.



Obrázek 4.7: Periodogram P funkce g(t)



Obrázek 4.8: Funkce g(t) s vyznačenou periodou

Problém, před kterým stojíme, a v tomto příkladu byl snadno řešitelný lidským okem, je ten, že potřebujeme automaticky určit, jaké vrcholy periodogramu P jsou významné a jaké nikoliv. Pro tento problém existuje celá řada statistických testů. Například *test R. A. Fishera* nebo *Siegelův test*.

Periodogram lze také vyjádřit pomocí PSD, respektive pomocí Fourierovy transformace pomocí vzorce:

$$P(f) = \frac{1}{N}\left|\sum_{n=1}^{N} g(t_n)e^{-2\pi i f k n N}\right|^2 = \frac{1}{n}\left|\mathcal{F}\{g\}\right|^2 = \frac{1}{n}\mathcal{P}_g$$

takto definoval klasický periodogram Artur Schuster v publikaci [10].

4.4 Lomb-Scargle periodogram

V předchozí části u periodogramu počítáme pouze s hodnotami na časové řadě a nikoliv s časem těchto hodnot. Předpokládá se totiž, že časová řada je rovnoměrně rozložená. Pokud bychom měli nerovnoměrně rozloženou časovou řadu a použili k její analýze klasický periodogram, tak by výsledek byl s vysokou pravděpodobností chybný.

Problémem, u kterého byla zjištěna potřeba periodogramu umožňujícího pracovat s nerovnoměrně rozloženými daty, byla analýza dat z vesmírných sond. V astronomii se objevily časové řady, které obsahovaly mezery a heteroskedastické nejistoty. Právě pro tyto účely byl vyvinut Lomb-Scargle periodogram (dále LS periodogram), jehož základ je položen v článcích od N. R. Lomba [11] a J. Scargla [12].

Můžeme přepsat klasický periodogram, jak jej definoval Schuster do následující podoby:

$$P(f) = \frac{1}{N}\sum_{n=1}^{N} g_n e^{-2\pi i f t_n}$$

kde t_n koresponduje pozorovaným časům. Takže nyní namísto rovnoměrného intervalu počítáme se skutečným časem jednotlivých hodnot g_n. Tento periodogram ztrácí určité statistické vlastnosti Diskrétní Fourierovy transformace. Nicméně Scargle ve článku [12] vyřešil tento problém pomocí zobecněné formy periodogramu.

Skutečnost je taková, že Scargleův modifikovaný periodogram je totožný s výsledkem, který lze získat přizpůsobením sinusového modelu datům na každé frekvenci f a vytvořením periodogramu z odpovídajících χ^2 hodnot na každé frekvenci f. [11]

Používáme model:

$$y(t; f) = A_f sin\left(2\pi f\left(t - \phi_f\right)\right)$$

kde A_f je amplituda a ϕ_f je fáze frekvence f. Můžeme spočítat χ^2 pro každou frekvenci f pomocí:

$$\chi^2 = \sum_n \left(y_n - y\left(t; f\right)\right)^2$$

Nejlepší model je takový, pro který výběr A_f a ϕ_f minimalizuje χ^2, který nazveme $\widehat{\chi}^2$. To znamená, že můžeme získat odhad periodogramu pouhým spočtením χ^2 na každé frekvenci, kterou bychom chtěli otestovat. Scargle [12] dokázal, že je možné LS periodogram zapsat jako:

$$P_{LS}(f) = \frac{1}{2}\left[\widehat{\chi}_0^2 - \widehat{\chi}^2(f)\right]$$

kde $\widehat{\chi}_0^2$ je hodnota pro neměnný referenční model. [13]

Toto vede k umožnění generalizování periodogramu skrz známou úpravu:

$$\chi^2 = \sum_n \left(\frac{y_n - y(t;f)}{\sigma_n}\right)^2$$

kde σ_n je nejistota každého měření y_n. Což nám normalizuje $y_n - y(t;f)$.

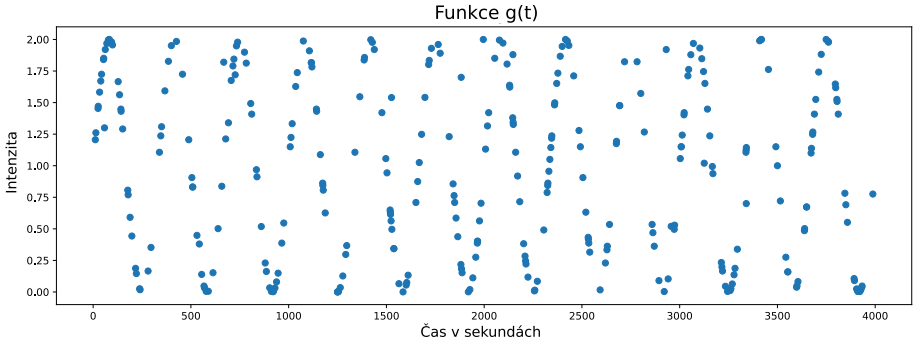
LS periodogram lze zapsat i pomocí vzorce [14]:

$$P_{LS}(\omega) = \frac{1}{\sum_i y_i^2}\left[\frac{(\sum_i y_i cos\omega(t_i - \widehat{\tau}))^2}{\sum_i cos^2\omega(t_i - \widehat{\tau})} + \frac{(\sum_i y_i sin\omega(t_i - \widehat{\tau}))^2}{\sum_i sin^2\omega(t_i - \widehat{\tau})}\right]$$

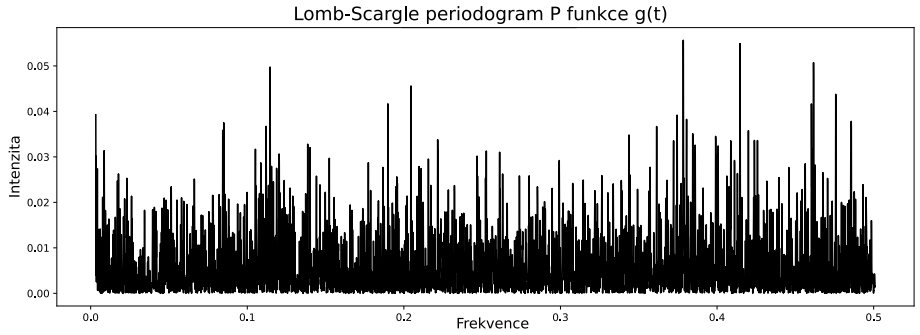
kde časová řada je definována dvojicí (t_i, y_i) pro $i := 1, 2, ..., n$, ω je frekvence a parametr $\widehat{\tau}$ je vypočten pomocí:

$$tan2\omega\widehat{\tau} = \frac{\sum_i sin2\omega t_i}{\sum_i cos2\omega t_i}$$

Na obrázku 4.9 je vyobrazena nerovnoměrná časová řada sinusového signálu s určitým procentem šumu. LS periodogram této časové řady je znázorněn na obrázku 4.10. Všimněte si, že se objevuje značné množství vrcholů, u kterých je intenzita téměř stejná. Získat periodu i u LS periodogramu nemusí být přímočaré. Stejně jako u klasického periodogramu potřebujeme pomocí statistiky odhadnou, jaké vrcholy jsou významně velké a jaké nikoliv. Pro tyto účely existuje řada funkcí a testů. Například *Funkce falešného poplachu*, *Kumulativní distribuční funkce* a *Scarglova kumulativní distribuční funkce*.



Obrázek 4.9: Funkce g(t) reprezentovaná nerovnoměrně rozloženou časovou řadou



Obrázek 4.10: LS periodogram funkce g(t)

4.4.1 Vlastnosti LS periodogramu

LS periodogram má dva hlavní benefity oproti klasickému periodogramu. Za prvé distribuce šumu na každé jednotlivé frekvenci je chí-kvadrát distribuována podle nulové hypotézy a za druhé výsledek je ekvivalentní periodogramu odvozenému z analýzy nejmenších čtverců.

V tabulce 4.1 jsou vypsány mýty a skutečnosti ohledně LS periodogramu.

Tabulka 4.1: Mýty ohledně LS periodogramu

Mýty ohledně LS periodogramu	
Mýtus	**Fakt**
LS periodogram je výpočetně efektivnější než klasický periodogram	Výpočetně jsou na tom oba podobně
LS periodogram je rychlejší než periodogram nejmenších čtverců, protože se vyhýbá explicitnímu řešení modelovaných koeficientů	Modelované koeficienty lze dopočítat, přičemž výpočetní rozdíl je zanedbatelný
LS periodogram umožňuje analytický výpočet statistik pro vrcholy periodogramu	I když to platí pro jednotlivé frekvence, neplatí to pro relevantnější otázku maximálních výšek vrcholů na více frekvencích, které je nutné buď aproximovat nebo vypočítat převzorkováním
Bayesovská analýza dokazuje, že LS periodogram je optimální statistika pro detekci periodického signálu v datech	Bayesovská analýza dokazuje, že LS periodogram je optimální statistika pro vložení sinusoidy do dat

4.5 Statistické testy významnosti na LS periodogramu

4.5.1 Pravděpodobnost falešného poplachu

Pravděpodobnost falešného poplachu (angl. False Alarm Probability, zkráceně FAP) je typickým přístupem ke kvantifikaci významnosti vrcholu. FAP měří pravděpodobnost, že soubor dat bez signálu by v důsledku náhodného zarovnání mezi náhodnými chybami vedl k vrcholu podobné velikosti.

Hodnoty na LS periodogramu následují χ^2 rozdělení se dvěma stupněmi volnosti, tzn. pokud $Z = P_{LS}(f_0)$ je hodnota intenzity na LS periodogramu pro frekvenci f_0, pak

$$P_{single}(Z) = 1 - exp(-Z)$$

je kumulativní pravděpodobnost, že intenzita na LS periodogramu menší než Z je výsledkem Gaussovského čistého šumu.

4.5.2 Kumulativní distribuční funkce

Kumulativní distribuční funkce (angl. Cumulative Distributive Function, zkráceně CDF) je ve statistice funkce, která určuje pravděpodobnost, že X bude mít hodnotu menší nebo rovnou x ($P[X \leq x]$). Pokud CDF použijeme na LS periodogram, tak máme na ose x intenzitu z periodogramu a na y ose je hodnota vyjadřující, kolik procent hodnot z periodogramu je menší než tato hodnota.

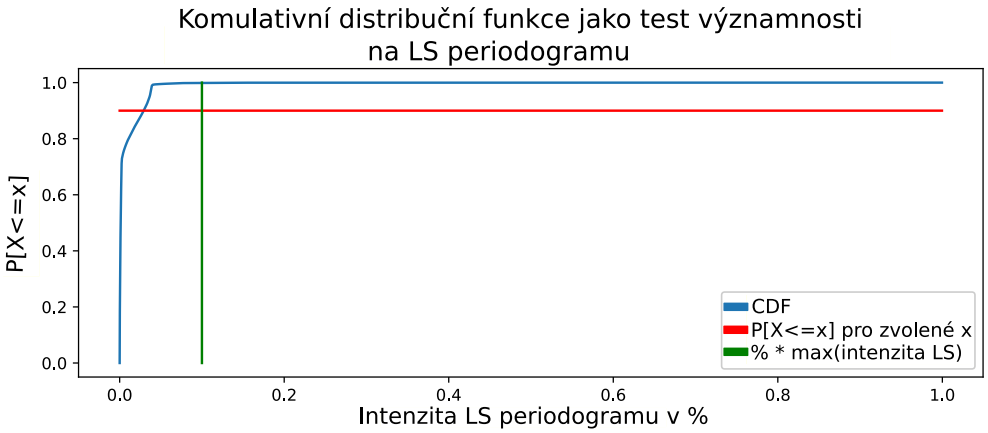
Test významnosti pro LS periodogram vypadá tak, že určíme „rychlost“ růstu CDF. To znamená, že určíme prahovou hodnotu $P[X \leq x]$ a hodnotu osy x, kterou definujeme jako nastavené procento z nejvyšší hodnoty osy x. Test významnosti pak spočívá v tom, zda místo, kde hodnota CDF překročí mezní hodnotu $P[X \leq x]$, má na ose x větší hodnotu než je nastavené procento z nejvyšší hodnoty osy x. Pokud toto platí a zároveň kontrolovaný vrchol má větší nebo rovnou hodnotu než má první hodnota CDF testu, tak je vrchol přijat jako statisticky významný.

Ekvivalentně by šel tento test vysvětlit tak, že pokud je výsledná CDF hodnota pro určený bod na ose x větší než nastavené procento. tak přijímáme. Přičemž bod definujeme jako nastavené procento z nejvyšší hodnoty na ose x.

Na obrázku 4.11 vidíme případ, kdy CDF test významnosti přijme pro všechny hodnoty, které přesáhnou stanovenou hranici ($P[X \leq x]$).

4.5.3 Scarglova kumulativní distribuční funkce

Scarglova kumulativní distribuční funkce (angl. Scargle's Cumulative Distributive Function, zkráceně SCDF) je založena na předpokladu, že data časové řady, která nejsou součástí signálu, tj. nejsou periodická, jsou Gaussovský čistý šum [15]. Pak hodnota periodogramu $Z = P_{LS}(\omega)$ na jakékoliv frekvenci ω je



Obrázek 4.11: Kumulativní distribuční funkce jako test významnosti na LS periodogramu funkce g(t)

exponenciálně distribuovaná s pravděpodobnostní funkcí hustoty, která byla definovaná Scarglem v roce 1982 v publikaci [12].

$$p_Z(z)dz = P\left[z < Z < z + dz\right] = \frac{1}{\sigma_X^2} e^{\frac{-z}{\sigma_X^2}}$$

Kumulativní distribuční funkce je pak daná pomocí rovnice:

$$P_Z(z) = P[Z < z] = \int_{\zeta=0}^{z} p_z(\zeta)d\zeta = 1 - e^{\frac{-z}{\sigma_X^2}}$$

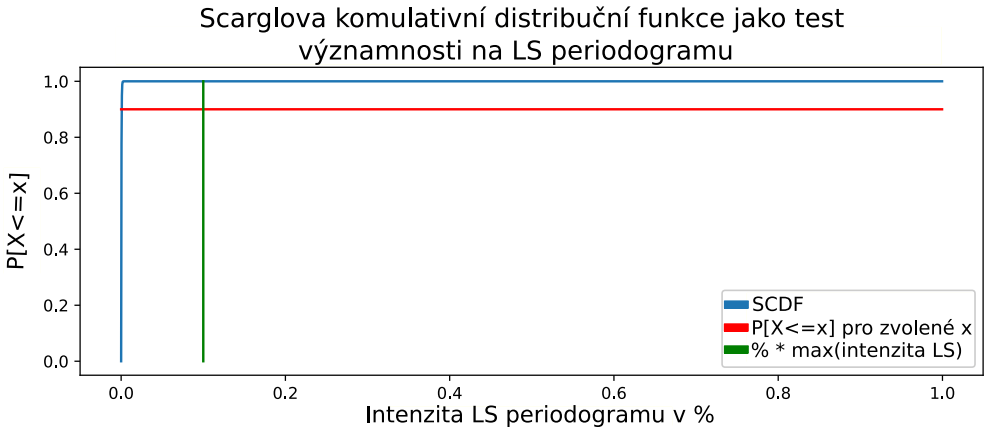
Přičemž nás zajímá pravděpodobnost, že hodnota periodogramu na určité frekvenci je větší než specifikovaná prahová hodnota z. To je dáno v rovnici:

$$P[Z > z] = 1 - P_Z(z) = e^{\frac{-z}{\sigma_X^2}}$$

Jak se pozorovaná hodnota z na periodogramu zvětšuje, stává se exponenciálně méně pravděpodobné, že by tak vysoká (nebo vyšší) úroveň hodnoty na periodogramu mohla být produkována pouze čistým šumem, a podle toho je pravděpodobnější, že pozorovaná úroveň je způsobena skutečným deterministickým jevem v měřeném signálu.

Stojí za zmínku, že argument exponenciály v kumulativní distribuční funkci není pouze pozorovaný výkon z, ale poměr $\frac{-z}{\sigma_X^2}$, což je poměr hodnoty na periodogramu k celkovému rozptylu dat.

Dále můžeme postupovat jako u CDF, tedy vezmeme místo klasické kumulativní distribuční funkce SCFD. Na obrázku 4.12 vidíme aplikování SCDF testu na stejný LS periodogram (resp. časovou řadu) jako příklad u CDF testu. Všimněte si, že křivka SCDF je více pravidelná než křivka CDF, která má nepravidelné kolísání v průběhu.



Obrázek 4.12: Scarglova kumulativní distribuční funkce jako test významnosti na LS periodogramu funkce g(t)

Pokud bychom chtěli SCDF testu podrobit pouze jednu hledanou hodnotu z (hodnota intenzity testovaného vrcholu), stačí nám vypočítat dvě hodnoty:

$$P_Z(z) = P[Z < z] = \int_{\zeta=0}^{z} p_z(\zeta)d\zeta = 1 - e^{\frac{-z}{\sigma_X^2}}$$

$$P_Z(s) = P[Z < s] = \int_{\zeta=0}^{s} p_s(\zeta)d\zeta = 1 - e^{\frac{-s}{\sigma_X^2}}$$

kde s je hodnota na ose X definovaná jako určité procento z maximální hodnoty periodogramu.

Test významnosti pomocí těchto dvou výsledných hodnot pak probíhá následovně:

1. Pokud je $P_Z(s)$ menší než nastavený práh osy y, pak na periodogramu nejsou žádné statisticky významné vrcholy.

2. Pokud je $P_Z(z)$ menší než nastavený práh osy y, pak testovaná hodnota z na periodogramu není statisticky významná.

3. Jinak je testovaná hodnota z významná a je detekovaná perioda na frekvenci f_z testované hodnoty periodogramu.

Kapitola 5

Návrh metody detekce periodicity pro časové řady ze síťového provozu

Cílem této práce je automatická detekce periodického chování na časových řadách vytvořených ze síťového provozu. Taková časová řada vznikne ze síťových toků (flow záznamů, resp. z obousměrných flow záznamů zvaných biflow) tak, že se vezme nějaká metrika z flow záznamu, např. počet paketů, a spojí se dohromady s časem flow záznamu a takto vzniklý datový bod se vloží do časové řady. Navržená metoda dokáže pracovat i na více metrikách, tzn. vytvoří se více časových řad z těch samých flow záznamů pro různé metriky (počet paketů, počet bajtů, ...) a výsledná detekovaná perioda se musí objevit na všech metrikách.

Metoda pro automatickou detekci periodického chování používá známé matematické postupy, které kombinuje v algoritmu speciálně navrženém pro detekci periodicity na nerovnoměrně rozložených časových řadách vytvořených ze síťového provozu.

Cílem algoritmu je popsat časovou řadu pomocí sady atributů a štítků, které vyjadřují, zda je provoz náhodný, či periodický, a v případě periody dále popisují jaký vzor se opakuje a jak často.

5.1 Klíčová myšlenka algoritmu

Hlavní myšlenka je volba LS periodogramu jako komponenty pro detekci periodicity na nerovnoměrně rozložených datech. Všimli jsme si totiž, že časové řady ze síťového provozu vykazují stejné jevy jako časové řady z vesmírných sond, kvůli kterým byl LS periodogram vytvořen. Samozřejmě jsme museli zakomponovat do metody statistický test významnosti na LS periodogramu, z nichž nejlepší výsledky jsme zaznamenaly s SCDF testem [15].

Metoda navíc kombinuje hybridní přístupy k detekci periodicity [3], [4], [6] k získání lepších výsledků. Postup metody je následující:

1. Pomocí kruhové autokorelační funkce (ACF) jsou získáni kandidáti na periodu.
2. Vytvoření LS periodogramu.
3. Provedení SCDF testu pro všechny frekvence kandidátů na periodicitu a získání periodických chování.

Metoda může snadno pracovat s více metrikami, stačí stejný postup zopakovat pro všechny metriky a pak pomocí „průniku“ odhalit společné periody, pro které platí, že jsou periodami pro celou časovou řadu.

5.2 Algoritmus detekce periodicity

Algoritmus pro detekování periodických chování na časové řadě vytvořené ze síťového provozu je sepsán v algoritmech 1 a 2 , přičemž podrobný popis jednotlivých kroků se nachází v podkapitolách. Popis algoritmu je vyobrazen na obrázku 5.1.

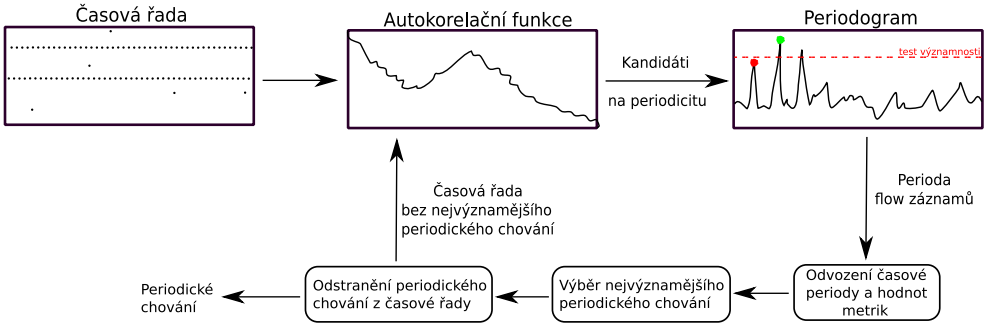
Pseudokód algoritmu 2 obsahuje funkci provádějící hlavní části metody a pseudokód algoritmu 1 obsahuje volání této funkce spolu s dalšími kroky, kde kroky označené pomocí * jsou volitelné.

Algoritmus 1: Algoritmus detekce periodicity

```
1 Vytvoření časové řady ze síťového provozu
2 PeriodicityDetection(Časová řada, ...)
3 Klasifikace zbývajícího provozu, který zůstal v časové řadě po
    průběhu funkce PeriodicityDetection (podkapitola 5.9)
4 Odvození hodnoty důvěry ve výsledek (podkapitola 5.10)
5 * Detekce významně větších mezer, rozdělení časové řady na části dle
    mezer a aplikování funkce PeriodicityDetection na tyto části
    (podkapitola 5.11)
6 * Klasifikace dle mezer časové řady (podkapitola 5.12)
7 * Odvození celkové hodnoty důvěry k nalezeným periodickým
    chováním na celé časové řadě vzhledem k výsledkům na částech
    rozdělených dle významně větších mezer (podkapitola 5.13)
```

Algoritmus 2: Algoritmus detekce periodicity

```
1  Function PeriodicityDetection(Časová řada, ...):
2      if Počet bodů nebo procentuální velikost časové řady vzhledem k
         originální časové řadě klesla pod stanovenou hranici then
3          return
4      Hledání kandidátů na periodicitu (podkapitola 5.4)
5      if Časová řada je konstantní (podkapitola 5.5) then
6          Přidání Periodic-constant pro každého kandidáta na
             periodicitu do výstupních štítků
7          return
8      Vytvoření LS periodogramu pro každou metriku (podkapitola 5.6)
9      for Pro každého kandidáta na periodicitu do
10         for Pro každou metriku do
11             Spočtení frekvenčního intervalu pro kandidáta p
                 (podkapitola 5.6)
12             Provedení statistického testu významnosti nejvýznamnější
                 frekvence z frekvenčního intervalu kandidáta p na LS
                 periodogramu (podkapitola 4.5) a výpočet spolehlivosti
                 testu významnosti (podkapitola 5.7)
13         if Pokud je kandidát na periodu p na všech metrikách
             statisticky významný then
14             Přidání dočasného štítku Periodic s periodou p
                 (podkapitola 5.3)
15         else if Pokud je kandidát na periodu p na některé metrice
             statisticky významný then
16             Přidání dočasného štítku Periodic-< metric > s periodou p
                 pro každou metriku kde je kandidát významný
                 (podkapitola 5.3)
17         else
18             Přidání dočasného štítku Non-periodic (podkapitola 5.3)
19     if V dočasných štítcích není žádný štítek then
20         return
21     Výběr nejvýznamnějšího z dočasných štítků vzhledem ke
         spolehlivosti a počtu opakování periodického chování
22     Dopočítání časové periody k periodě datových bodů p a dohledání
         opakující se hodnoty všech metrik (podkapitola 5.8)
23     Přidání štítku mezi výstupní štítky a odstranění bodů z časové
         řady, které reprezentuje
24     PeriodicityDetection(Zbytek časové řady, ...)
```



Obrázek 5.1: Diagram metody detekce periodických chování na časových řadách ze síťového provozu

5.3 Přiřazované štítky a jejich atributy

Na základě výsledku algoritmů 1 a 2 mohou být přiřazeny síťové závislosti tyto štítky:

- *Periodic* je přiřazen tehdy, pokud je detekováno periodické chování s nějakou periodou p na všech metrikách.
- *Periodic-*$<$ *metric* $>$ je přiřazen tehdy, pokud je detekováno periodické chování s nějakou periodou p na konkrétní metrice $<$ *metric* $>$, přičemž perioda p není detekována na všech metrikách.
- *Periodic-constant* je přiřazen tehdy, pokud není detekován štítek *Periodic* a zároveň je časová řada konstantní s nějakou prahovou hodnotou, poté jsou přiřazeni kandidáti na periodicitu jako štítek *Periodic-constant*.
- *Non-periodic* je přiřazen tehdy, pokud je nalezen kandidát na periodicitu c, který není potvrzen LS periodogramem.
- *Outlier-in-flow* je přiřazen jako doplňující štítek ke štítkům *Periodic* a *Periodic-*$<$ *metric* $>$, pokud zbývající datové body v časové řadě (bez *Outliers-in-time*) mají pouze jednu hodnotu v každé metrice. Potom jsou považovány za speciální druh šumu, který není náhodný.
- *Random-noise* je přiřazen jako doplňující štítek ke štítkům *Periodic* a *Periodic-*$<$ *metric* $>$, pokud zbývající datové body v časové řadě jsou náhodným šumem.

Obecně může být přiřazeno více štítků zároveň, jelikož se v časové řadě může objevovat více periodických chování a naše metoda je dokáže detekovat.

Štítek *Periodic* (resp. *Periodic-*$<$ *metric* $>$, resp. *Periodic-constant*) nese řadu atributů, které definují, o jaké periodické chování se jedná. Nejdůležitějšími

atributy jsou *flow perioda* (perioda datových bodů), tzn. kolik flow se opakuje v periodě, *časová perioda*, *hodnoty metrik*, které definují periodu, *spolehlivost* testu významnosti, procentuální *délka* periodického chování vzhledem k celé časové řadě, *Outliers-in-time* popisující podposloupnosti časové řady, které mají stejnou délku a hodnoty metrik jako periodické chování jen s jinou časovou periodou, a *důvěra* v celkový výsledek. Dále mohou být volitelně přidruženy atributy vyplývající z detekce významně větších mezer, která je popsána v podkapitole 5.11. Popis všech atributů přiřazovaných k periodickému chování je k dispozici k nahlédnutí v příloze B.

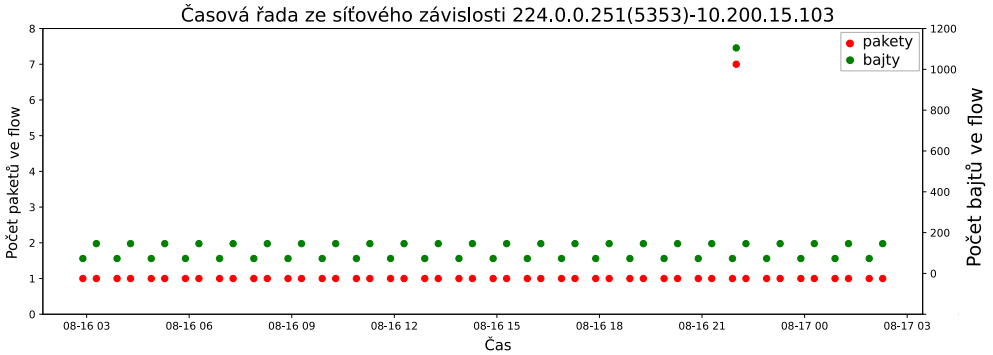
5.4 Hledání kandidátů na periodicitu

Pro odhadnutí, jaké periody se v časové řadě objevují, používáme kruhovou autokorelační funkci (ACF), která je definována v podkapitole 4.2. ACF funkce hledá kandidáty na periodu pouze na základě hodnot datových bodů dané metriky a zanedbává čas datových bodů. Proto je vhodné přidat metriku rozdílů mezilehlých časů flow záznamů.

Autokorelační funkce nám tedy dává heuristický odhad periodického chování. Potvrzení odhadu musí přijít od metody, která pracuje zároveň s hodnotami metrik i s časem datových bodů.

Kandidáty na periodicitu vybíráme tak, že ponecháme ve výsledku ACF pouze takové body x_i, pro které platí $x_{i-1} < x_i < x_{i+1}$ (tj. lokální maxima). Poté spočítáme rozdíly indexů mezi těmito body a dostáváme tzv. lagy. Pokud se nějaký lag objevuje častěji, než je nastavené procento všech lagů, tak je lag předán jako kandidát na periodu.

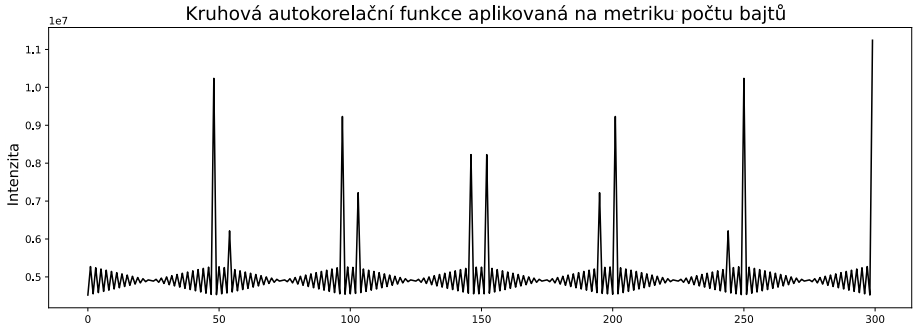
Časová řada reprezentovaná metrikami počtu paketů a počtu bajtů ve flow záznamu, na kterou aplikujeme ACF s cílem získání kandidátů na periodicitu, je vyobrazena na obrázku 5.2.



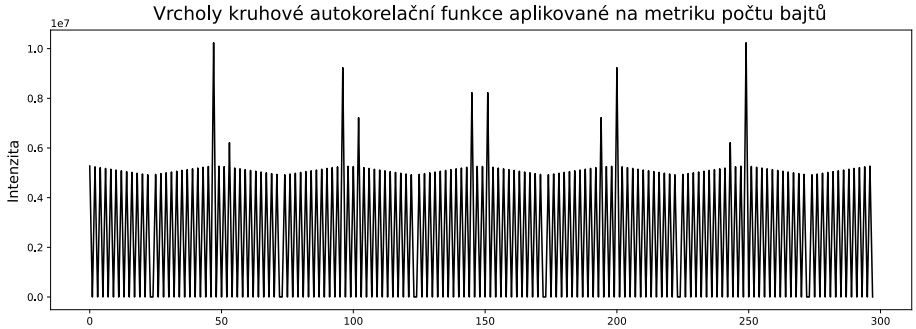
Obrázek 5.2: Příklad časové řady, na níž bude aplikováno hledání kandidátů

Na obrázku 5.3 je výsledek autokorelační funkce aplikované na metriku počtu bajtů ve flow časové řady zobrazené na 5.2 (pro počet paketů je graf podobný). Dále jsou na obrázku 5.4 zobrazené vrcholy kruhové autokorelační

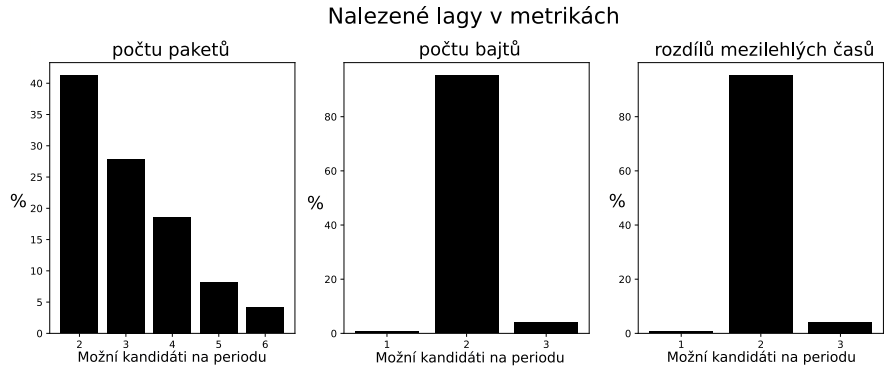
funkce počtu bajtů, znázorňující vybrání lokálních maxim. A nakonec na obrázku 5.5 jsou histogramy všech nalezených kandidátů na periodicitu pro testované metriky, z nichž je pomocí 10% prahové hodnoty vybrána *2*. Při výběru se kandidát (nebo jeho dělitel) musí objevit ve všech testovaných metrikách.



Obrázek 5.3: Kruhová autokorelační funkce aplikovaná na metriku počtu bajtů ve flow



Obrázek 5.4: Vrcholy kruhové autokorelační funkce aplikované na metriku počtu bajtů ve flow



Obrázek 5.5: Histogramy lagů pro počet paketů, počet bajtů a rozdíl mezilehlých časů

Stojí za povšimnutí, že lagy na metrice počtu paketů jsou rozděleny s konstantním klesáním. To je způsobeno tím, že se v metrice paketů opakují stejné hodnoty. Zatímco na metrice počtu bajtů se opakují dvě rozdílené hodnoty, a proto jsou lagy z větší části shodné.

5.5 Test konstantnosti časové řady

Za konstantní časovou řadu považujeme tu, ve které jsou všechny nebo téměř všechny hodnoty určité metriky stejné.

Testování, zda není časová řada konstantní, probíhá po získání kandidátů na periodicitu, a to kvůli drobné nevýhodě LS periodogramu, který ne vždy funguje správně na konstantních časových řadách (patrné z testování na syntetických časový řadách viz podkapitola 7.1.1) a mnohdy nedokáže periodogram správně vytvořit, a tak je detekce periodicity nemožná.

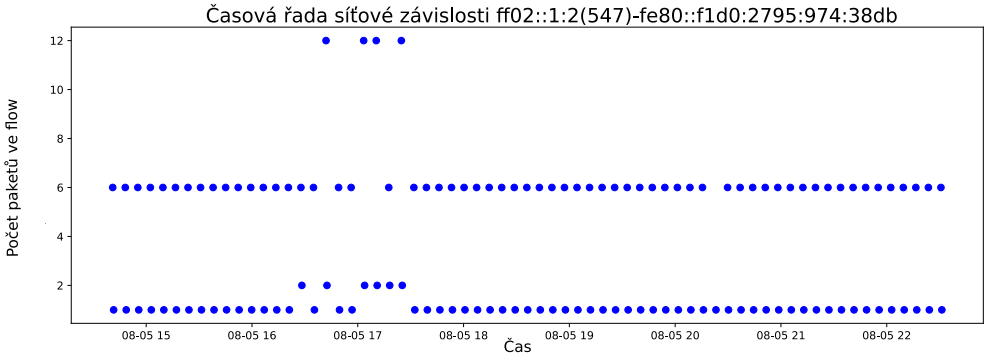
Test probíhá pomocí histogramu, u kterého hledáme, zda se nějaká hodnota opakuje ze zadaného procenta všech hodnot. Pokud kontrolovaná časová řada nějaké metriky toto splňuje, tak je daná metrika označena jako konstantní. Pokud jsou časové řady všech metrik označeny jako konstantní, pak je celá závislost označena konstantní.

V případě, že je časová řada označena za konstantní (tj. konstantní na všech metrikách), označíme za periody kandidáty z Autokorelační funkce (ACF). Závislosti pak přiřadíme štítek Periodic-constant s flow periodou rovnou 1 (pro konstantní časovou řadu se opakuje vždy jedna hodnota).

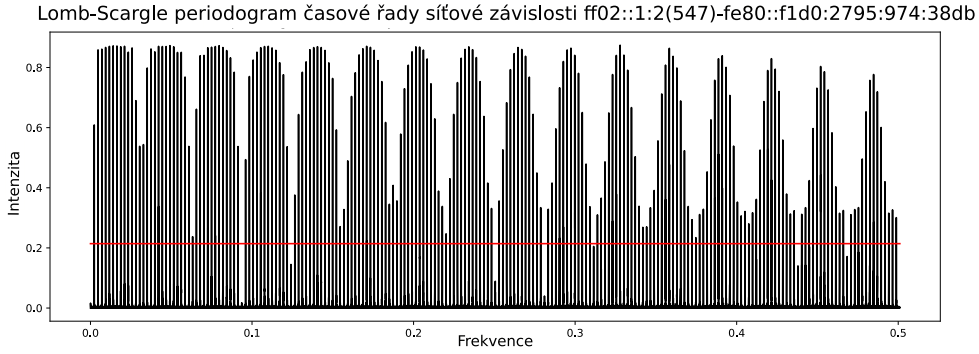
5.6 LS periodogram

V časových řadách vytvořených ze síťového provozu nemáme většinou zajištěné, že by data byla rovnoměrně vzorkovaná a rozmístěná, ale u některých závislostí časové řady tyto vlastnosti mají. Jelikož obecně může docházet k oběma případům, je pro časové řady ze síťového provozu LS periodogram, který je definován v kapitole 4.4, vhodnou volbou pro detekci periodicity. Pro příklad budeme uvažovat časovou řadu vyobrazenou na obrázku 5.6

Periodogram se používá k identifikaci dominantních period (respektive frekvencí) časové řady. K tomu, abychom poznali, jaké hodnoty na periodogramu nám ukazují detekovanou periodu, musíme použít statistický test významnosti, který nám řekne, zda je konkrétní hodnota statisticky významná vzhledem k ostatním hodnotám. Na grafu LS periodogramu vyobrazeném na obrázku 5.7 jsou statisticky významné hodnoty na vrcholech periodogramu nad červenou hranicí spočtenou pomocí statistického testu významnosti.

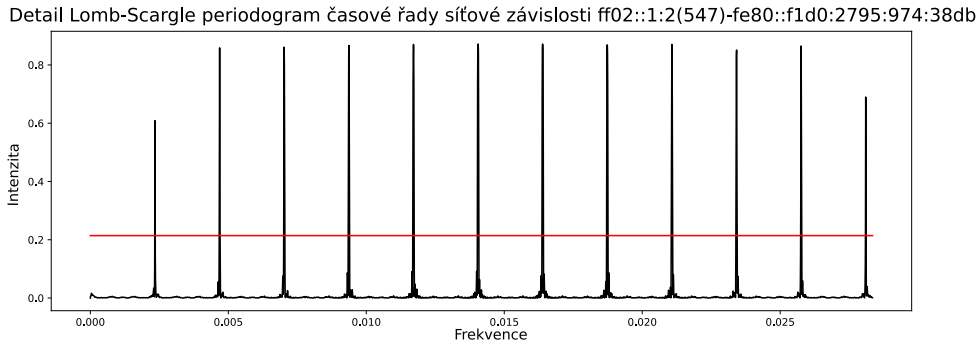


Obrázek 5.6: Příklad časové řady, na níž bude aplikován LS periodogram



Obrázek 5.7: LS periodogram aplikován na časovou řadu ze síťového provozu

Na grafu LS periodogramu vidíme, že existuje více period, což je stejné pro každou časovou řadu. Je to způsobeno tím, že pokud má časová řada periodu p, tak bude mít i periodu $2p$, atd., obecně kp. Příklad tohoto jevu je lépe vidět z detailu LS periodogramu vyobrazeném na obrázku 5.8. Zde stojí za povšimnutí, že vzdálenost významných vrcholů od sebe je stejná.



Obrázek 5.8: Detail LS periodogram aplikován na časovou řadu ze síťového provozu

Proto je vhodné použít ACF pro odhad, jaké periody se v časové řadě vyskytují, a pak se zaměřit na konkrétní frekvence periodogramu k nim vztažené. Z periody p lze získat frekvenci pomocí vzorce $p = \frac{1}{f}$, frekvenci lze získat pomocí $f = \frac{1}{p}$. Poté, co získáme potřebnou frekvenci, hledáme blízko ní (v nějakém menším intervalu kolem ní) nejvyšší vrchol, pro který provedeme statistický test významnosti. Interval, který má smysl vytvořit pro časové řady ze síťového provozu, kde periody mohou být jen celá čísla, lze spočítat následovně:

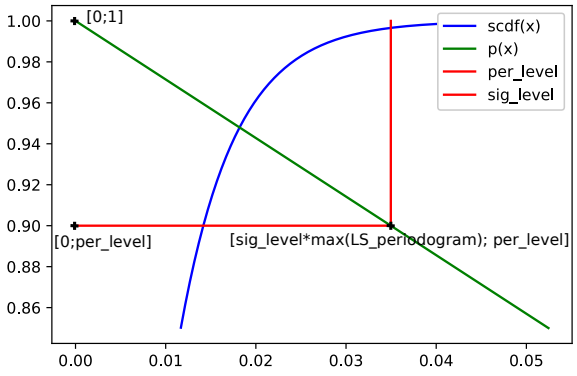
$$I_f = \left(\frac{1}{p+0.5}; \frac{1}{p-0.5}\right)$$

Důležité je poznamenat, že tvorba LS periodogramu silně závisí na počtu datových bodů a na časovém intervalu, ve kterém jsou datové body rozprostřeny. Pro časové řady s málo datovými body nebo pro časové řady s příliš krátkým časovým intervalem může LS periodogram selhat v hledání periody a výsledek může být nesprávný. Na datech ze síťového provozu doporučujeme časový interval alespoň v řádu hodin a časové řady, které obsahují alespoň několik desítek flow záznamů.

5.7 Spolehlivost statistických testů významnosti

Spolehlivost je podstatným ukazatelem pro určité aplikace, které by následně výsledky detekce periodicity používaly. Postup výpočtu spolehlivosti CDF a SCDF testů významnosti je ekvivalentní, proto si ho vysvětlíme jen pro SCDF test.

Jak již bylo řečeno v definici SCDF testu v kapitole 4.5.3, je potřeba dvou prahových hodnot. První prahová hodnota určuje hodnotu na ose x (označme si jí *sig_level*) a druhá na ose y (označme si jí *per_level*). Nejprve si vyznačíme známé body a funkce v SCDF testu na obrázku 5.9.



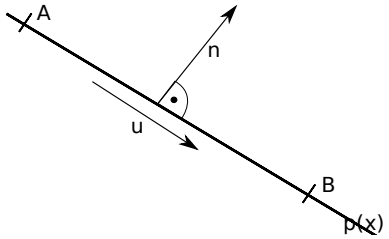
Obrázek 5.9: Známé body na SCDF testu

Pro účely výpočtu spolehlivosti si označíme:

$$t_{pl} = per\_level$$

$$t_{sl} = sig\_level * max(LS\_peridoogram)$$

Spolehlivost získáme tak, že spočítáme průnik přímky $p(x)$ a funkce $SCDF(x)$ a pak odvodíme, v kolika procentech přímky $p(x)$ se průnik nachází. Nejdříve si tedy musíme odvodit rovnici přímky $p(x)$ zobrazenou na obrázku 5.10.



Obrázek 5.10: Přímka p(x) spolehlivosti SCDF testu

Body na přímce $p(x)$ označíme jako $A = [0; 1]$ a $B = [t_{sl}; t_{pl}]$. Chceme z nich získat rovnici přímky $p(x)$:

$$p(x) : ax + by + c = 0$$

Definujeme normálový vektor $\widehat{n} = (a, b)$ a směrový vektor:

$$\widehat{u} = \overline{AB} = B - A = (t_{sl}, t_{pl} - 1)$$

Normálový vektor $\widehat{n}$ lze odvodit ze směrového vektoru $\widehat{u} = (u_1, u_2)$ pomocí předpisu $\widehat{n} = (u_2, -u_1)$, takže platí $\widehat{n} = (t_{pl}-1, -t_{sl})$. Dosadíme tedy $a = t_{pl} - 1$ a $b = -t_{sl}$ do rovnice $p(x)$:

$$p(x) : (t_{pl} - 1)x - t_{sl}y + c = 0$$

Zbytek odvodíme dosazením bodu A a dopočítáním c:

$$c = -(t_{pl} - 1)x + t_{sl}y = -(t_{pl} - 1) * 0 + t_{sl} * 1 = t_{sl}$$

Dostáváme tedy rovnici $p(x) : (t_{pl} - 1)x - t_{sl}y + t_{sl} = 0$, ze které vyjádříme y:

$$y = \frac{(t_{pl} - 1)x + t_s l}{t_s l}$$

Při odvozování hodnoty spolehlivosti SCDF testu hledáme průsečíky funkce $SCDF(x)$ a přímky $p(x)$. Hledáme tedy takové x, že pro něj platí $SCDF(x) = p(x)$. To znamená, že platí:

$$1 - e^{\frac{-x}{\sigma_X^2}} = \frac{(t_{pl}+1)x + t_{sl}}{t_{sl}}$$

Tato rovnice má následující řešení:

$$x = \sigma_X^2 \mathcal{W}_n \left(\frac{t_{sl}}{\sigma_X^2 (t_{pl} - 1)} \right)$$

kde $\mathcal{W}_n$ () pro nějaké $n \in N$ je *Lambertova W funkce* [16].

Jakmile vlastníme výsledné x, je spolehlivost spočtena jako procentuální hodnota na ose x (což je ekvivalentní k procentuální hodnotě na přímce $p(x)$) mezi 0 a t_{sl}. Tedy spolehlivost $C = 1 - \frac{x}{t_{sl}}$, kde $C \in < 0, 1 >$, tzn. pro procenta stačí vynásobit stem. Na obrázku 5.9 takto spočtená spolehlivost vychází na cca 45 %.

5.8 Získání časové periody a opakujících se hodnot

5.8.1 Opakující se hodnoty

Z LS periodogramu získáme potvrzení kandidáta na periodu p, jehož hodnotu nazýváme flow perioda. Ta nám říká, kolik datových bodů (flow záznamů) na časové řadě se periodicky opakuje. Takže nám zbývá nalézt, jaké přesně hodnoty to jsou. Projdeme tedy časovou řadu každé metriky a hledáme nejčastěji se vyskytující posloupnosti délky p. Taková posloupnost potom definuje opakující se hodnoty dané detekované periodicity s periodou p.

Ve většině případů je taková posloupnost nalezena s opravdu významně větším počtem opakování, než s jakým se objevuje druhá nejčastější posloupnost.

5.8.2 Časová perioda

Detekce periodicity se provádí na základě opakujících se datových bodů (flow periody). Po detekci tedy potřebujeme odvodit, jak dlouhý je interval (v jednotkách času), po kterém se vzor opakuje – časová perioda. Odvození časové periody nemusí být vždy jednoznačné, proto k výslednému štítku přiřazujeme takovou časovou periodu, která se vyskytovala nejčastěji. Zbylé časové periody jsou později přiřazeny ke stejnému štítku jako tzv. *anomálie v čase* na daném štítku (atribut *Outliers-in-time*), pokud nejsou detekovány jako významné s nějakou periodou po odstranění této periodicity, tj. v další iteraci rekurzivního procházení časové řady. A je spočtena průměrná časová perioda, která je také přidružena ke štítku.

Odvození časových period, z nichž vybereme tu nejčastější, je snadné. Stačí projít časovou řadu a hledat podposloupnost, která pasuje na detekovanou posloupnost délky periody p pro daný štítek a z času na časové řadě odvodit nejčastěji se vyskytující časovou periodu.

Existují případy, u kterých se opakují v periodickém chování stejné hodnoty každé metriky. To potom znamená, že všechny hodnoty u dané metriky mají v periodickém chování stejnou hodnotu. V takovém případě algoritmus zvažuje, zda se flow perioda nedá zjednodušit na periodu rovnou jedné. To nastane pouze tehdy, pokud platí, že nová periodicita s flow periodou rovnou jedné klasifikuje více nebo stejně datových bodů časové řady jako původní periodicita.

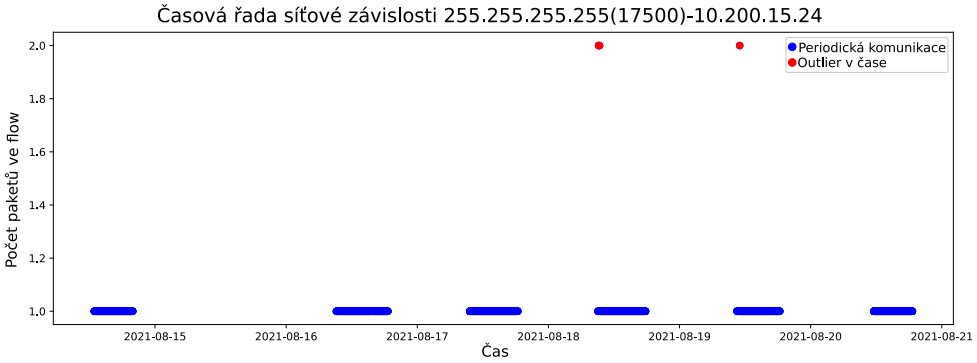
Nejčastější časová perioda je řešení, které v některých případech nefunguje příliš dobře. Například se objevují časové řady, které obsahují více časových period s téměř shodným počtem opakování. U takového případu není nejčastější časová perioda reprezentativní, a co je horší, detekovaná perioda se bude v rekurzivním algoritmu detekovat znovu do té doby, dokud jsou ostatní časové periody statisticky významné (protože rekurzivní algoritmus vždy odstraní body opakující se s nejčastější časovou periodou). Pro tyto případy jsme implementovali možnost odstraňování *anomálií v čase* spolu s periodickými datovými body.

5.9 Klasifikace zbývajícího provozu

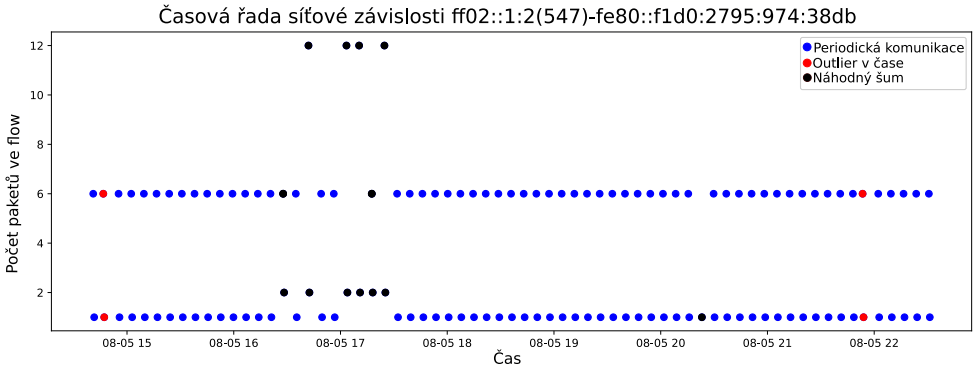
Poté, co rekurzivní algoritmus skončí, nám vrátí nalezená periodická chování, popsaná vždy štítkem a dalšími parametry a zbytkovou časovou řadou, na které už nebyl schopen nic detekovat (tj. body neodpovídající žádné z period). Jako další krok danou časovou řadu projdeme a odstraníme z ní všechny hodnoty, které mohou být anomálie v čase, pokud již nejsou odstraněny spolu s periodickým chováním v rekurzivním algoritmu. Zbývající body jsou považovány za náhodný šum nebo tzv. *anomálie ve flow*.

Anomálie ve flow je speciální případ náhodného šumu a je pouze jeden případ, kdy náhodný šum považujeme za *anomálii ve flow*. A to tehdy, když všechny zbývající body mají stejnou hodnotu. Na obrázku 5.11 můžeme vidět označené body, které algoritmus označí jako *anomálie ve flow*.

V grafu časové řady znázorněné na obrázku 5.12 jsou červeně zbarvené body, které jsou *anomálie v čase* pro periodické chování detekované na této časové řadě (s opakovanými hodnotami 1 a 6). Procentuální poměr *anomálií v čase* si uložíme pro pozdější počítání důvěry v detekované periodické chování. Body, které nejsou podezřelé, že by mohly být *anomálií v čase* pro nějaké detekované periodické chování a zároveň nejsou *anomálie ve flow*, jsou označeny za náhodný šum, jehož procentuální poměr také sledujeme.



Obrázek 5.11: Příklad outlierů ve flow



Obrázek 5.12: Příklad outlierů v čase a náhodného šumu

5.10 Hodnota důvěry ve výsledek

Důležitou součástí každého přiřazeného štítku označujícího periodicitu je hodnota důvěry ve správnost výsledku (od 0 do 100 %). Výpočet důvěry jsme stanovili na základě čtyř dostupných metrik, které nejvíce vypovídají o spolehlivosti informací o nalezené periodicitě:

- hodnota spolehlivosti SCDF testu,
- počet datových bodů, které popisují dané periodické chování (v procentech z celkového počtu datových bodů časové řady),
- kolik provozu je pro toto periodické chování identifikováno jako outlier v čase (v procentech) a
- úspěšnost detekované periody vzhledem k teoretické úspěšnosti periody se shodnou délkou v teoretické časové řadě shodné délky.

Hodnotu důvěry ve štítek spočítáme pomocí vzorce:

$$r_{st} = \frac{\frac{cl}{100-o} + \frac{c+l+(100-o)}{3} + p_r}{3}$$

kde c je spolehlivost SCDF testu, která byla definovaná v podkapitole 5.7, l je procentuální počet datových bodů klasifikovaného provozu, o je procentuální poměr *anomálií v čase*, a p_r je úspěšnost periodického chování. Dále je nutné poznamenat, že l a o jsou na sobě závislé a platí, že $o \in < 0, 100 - l >$.

Úspěšnost periodického chování p_r se spočítá pomocí vzorce:

$$p_r = \frac{p_l + p_f}{2}$$

kde p_l určuje, jak moc je dané periodické chování úspěšné vzhledem k tomu, jak úspěšné by mohlo být v časové řadě dané velikosti a vypočítá se pomocí:

$$p_l = \frac{100p_0}{\left\lfloor \frac{n}{p} \right\rfloor}$$

kde p_0 je počet opakování periodického chování v časové řadě, pro které platí $p_0 \in < 2, n >$, p je perioda (počtu datových bodů, které se v periodickém chování opakují) a n je velikost časové řady (počtu datových bodů časové řady).

Což značí, že zatímco $l + o$ představuje procentuální zastoupení periodického chování (nezávisle na konkrétní časové periodě) ze všech bodů, tak p_l představuje procentuální zastoupení periodického chování na časové řadě vzhledem k dané periodě p. Pokud je p_l rovno 100, pak je časová řada tvořena p_0 opakováními periodického chování (ne nutně se stejnou časovou periodou) a maximálně $p-1$ náhodnými datovými body. Pokud je $l+o$ rovno 100, tak je celá časová řada tvořena periodickým chováním (ne nutně se stejnou časovou periodou).

Dále p_f ve vzorci $p_r = \frac{p_l+p_f}{2}$ představuje procentuální zastoupení periodického chování vzhledem k určení, že perioda rovná 1 je nejvýznamnější a perioda $\lfloor \frac{n}{2} \rfloor$ je nejméně významná. Perioda $\lfloor \frac{n}{2} \rfloor$ je největší možná perioda. Potom platí:

$$p_f = \frac{100p_0}{n}$$

Tato část vzorce má za cíl zvýhodnit v p_r menší periody, které mají zákonitě i větší možnost opakovatelnosti v časové řadě. Tzn. pokud mají dvě rozdílné periody p_l blížící se 100 (jsou úspěšné), tak se díky p_f zvýhodní perioda, která je menší a tím pádem se častěji opakuje.

Pro p_0 platí, že lze vypočíst pomocí l a o pomocí vzorce:

$$p_0 = \frac{(l+o)}{100}\frac{n}{p}$$

Nejlepší možný výsledek nastává, když se c blíží 100, l se také blíží 100, tím pádem o se blíží 0 a p_r se také blíží 100. V takovém případě se části jmenovatele $\frac{cl}{100-o}$, $\frac{cl(100-o)}{3}$ a p_r blíží 100 a celkový výsledek se tedy také blíží hodnotě spolehlivosti 100 %.

V případě, že výsledek nebude ideální, nám první část vzorce $\frac{cl}{100-o}$ zajišťuje rychlý pokles, pokud má chování nízkou spolehlivost SCDF testu c nebo je jako periodický označen jen malý úsek řady (nízké l). V případě, že l bude nízké, ale bude větší počet *anomálií v čase* o, tak se pokles této části vzorce zmenší. To samozřejmě nemusí vždy být výhodou. Například když najdeme periodické chování, které pokrývá relativně velkou část časové řady, ale zároveň má z nějakého důvodu nízkou spolehlivost SCDF, tak bude hodnota důvěry nejspíše výrazně menší, než bychom chtěli, proto máme ve vzorci onu druhou část. Tato druhá část vzorce chce, aby se všechny hodnoty blížily k ideální hodnotě ($c = 100$; $l = 100$; $o = 0$) a je „jedno“ jaká část se zmenšila (resp. u o zvětšila), protože reakce je stejná. Pokud nastane příklad nevýhody první části vzorce, tak pokles v této části nebude tak výrazný. Poslední část vzorce p_r je částečně závislá na l, takže pokud se l bude blížit 100, tak i p_l. Zároveň díky p_f dosáhneme zvýšení důvěry na základě počtu opakování, protože čím je p větší, tím je blíže k *Nyquistově frekvenci* a tak je p_f menší. Všechny tyto části vzorce poté zprůměrujeme, čímž se snažíme snížit nevýhody každé části vzorce a dosáhnout nejlepšího možného odhadu důvěry.

Všimněte si, že k nejvyšší možné hodnotě důvěry (100 %) se může blížit pouze perioda opakování jednoho datového bodu, což jsme stanovili, protože cílem metody je detekce periodických spojeních a jedno spojení je reprezentované jedním flow záznamem, který je reprezentován jedním datovým bodem v časové řadě.

5.11 Detekce významně větších mezer

Jak je vidět i na již vyobrazených příkladech časových řad na obrázcích 5.11 a 3.5, komunikace často probíhá v různě dlouhých intervalech, mezi kterými jsou různě dlouhá období bez komunikace (mezery). Cílem této analýzy je rozdělit časovou řadu na tyto intervaly komunikace a aplikovat algoritmus detekce periodicity na každý z nich samostatně.

Nejprve je nutné detekovat takové mezery mezi jednotlivými body, které jsou oproti ostatním významně větší, a zároveň tak, aby časovou řadu nerozdělily na příliš mnoho částí, které by pak stejně LS periodogram nedokázal analyzovat. To znamená, že hledáme kompromis mezi počtem mezer, které chceme maximálně detekovat a velikostí mezer.

Nejdříve zkontrolujeme možnost výskytu statisticky významných mezer pomocí podmínky, že maximální mezera je více než t_{ss}-krát větší než průměrná mezera:

$$\frac{max(\mathcal{S})}{mean(\mathcal{S})} \leq t_{ss}$$

kde $\mathcal{S}$ je posloupnost velikostí mezer mezi body časové řady (v sekundách) a t_{ss} je nastavená prahová hodnota. Pokud je podmínka platná, tak se v časové řadě nacházejí významně velké mezery. Jinými slovy, pokud logaritmický rozdíl maximální a průměrné mezery není větší než zadaná prahová hodnota, není nutné vyhledávat v časové řadě významně větší mezery.

Detekce významných mezer se pak provádí pomocí matematického předpisu:

$$\{s_i | s_i > mean(\mathcal{S}) * (1 + t) \ \& \ s_i > stdev(\mathcal{S}) * (1 + t), s_i \in \mathcal{S}\}$$

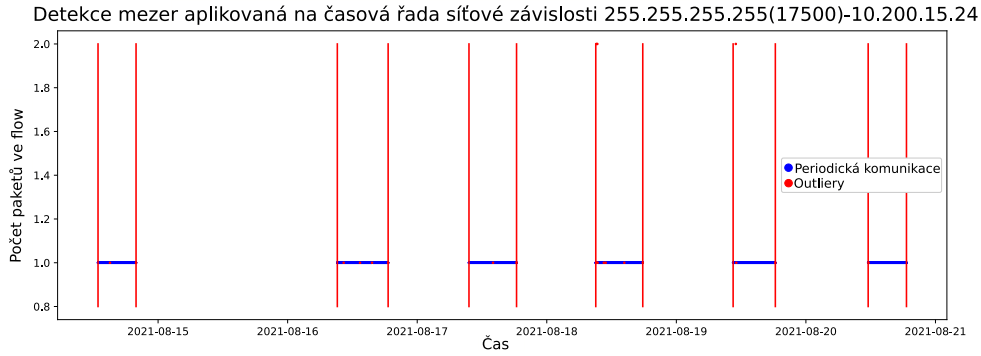
kde $\mathcal{S}$ je množina všech mezer, s_i je i-tá mezera, $mean(\mathcal{S})$ je průměr všech mezer, $stdev(\mathcal{S})$ je směrodatná odchylka všech mezer a t je prahová hodnota, zpočátku nastavená na 1. Pokud tento předpis vrátí více mezer než je nastavitelné procento l z počtu všech mezer N, pak se prahová hodnota t inkrementuje o 1 a detekce se opakuje. V pseudokódu je tento algoritmus vyobrazen v algoritmu 3.

Algoritmus 3: Algoritmus detekce významně větších mezer

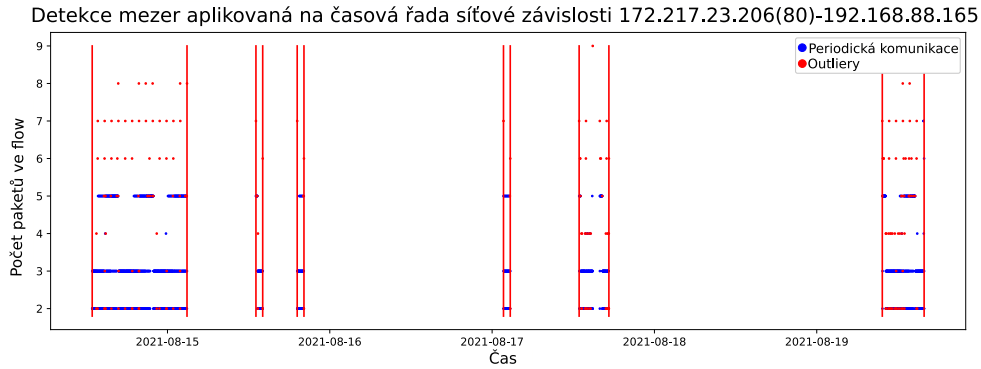
```
if max(S)/mean(S) ≤ t_ss then
    return {}
t = 1; l; S_sig = S; N = počet prvků S
while Počet prvků S_sig / N > l do
    S_sig = {s_i | s_i > mean(S)*(1+t) & s_i > stdev(S)*(1+t), s_i ∈ S_sig}
    t+ = 1
return S_sig
```

Při použití tohoto algoritmu jsme získali mezery ohraničené červenými hranicemi v časových řadách zobrazených na obrázcích 5.13 a 5.14.

Dle detekovaných mezer je následně časová řada rozdělena na části a na každou z nich se zavolá rekurzivní algoritmus (již bez detekce mezer a jejich klasifikace). Výsledky jsou následně používány pro zpřesnění hodnoty důvěry v detekované periodické chování a také pro popsání jejich vývoje na částech časové řady, tj. zda je periodicita ve všech částech stejná, či se nějak mění.



Obrázek 5.13: Příklad detekce významně větších mezer



Obrázek 5.14: Příklad detekce významně větších mezer

5.12 Klasifikace dle mezer časové řady

5.12.1 Klasifikace mezer

Při klasifikaci mezer postupujeme podle počtu mezer:

- Pokud není detekována ani jedna mezera, tak je přiřazen štítek *Continuous-communication*, který znamená, že se komunikuje nepřerušovaně.
- Pokud je detekována právě jedna mezera, tak zkoumáme, jak je velká v porovnání s délkou celé komunikace. Když je větší než nastavené procento, pak je přiřazen štítek *With-one-large-space*, jinak je přiřazen štítek *With-one-small-space*.
- Pokud je detekováno více mezer, pak zkoumáme jejich velikosti. Pokud jsou přibližně stejné, pak je přiřazen štítek *Similarly-large-spaces*, jinak *Randomly-large-spaces*. Přičemž existují prahové hodnoty, a také se zanedbávají některé příliš odlehlé délky mezer.

Po provedení klasifikace mezer je k časové řadě přiřazen právě jeden štítek popisující mezery. Musíme poznamenat, že pokud použijeme jako vstup časové řady, které jsou vytvořeny ze síťového provozu relativně krátkého časového úseku, tak pro většinu periodických časových řad bude přiřazen štítek *Continuous-communication*. A to z důvodu, že mezery v periodických komunikací nastávají většinou po nějakých velkých časových úsecích nebo vůbec.

5.12.2 Klasifikace rozdělení provozu

V této analýze sledujeme časové rozpětí a počet datových bodů časových řad, které byly vytvořeny z původní časové řady rozdělením dle statisticky významných mezer. Pokud mají přibližně stejnou délku, pak je přiřazen štítek *Similarly-large-parts*, jinak je přiřazen štítek *Randomly-large-parts*.

Na příkladu časové řady zobrazené na obrázku 5.13 jsou přiřazeny štítky *Similarly-large-spaces* a *Similarly-large-parts*. Můžeme vidět, že první mezera, která je výrazně delší, byla zanedbána.

5.12.3 Klasifikace vývoje detekce na rozděleném provozu

Jakmile je časová řada rozdělena významnými mezerami, je na jednotlivé části aplikován rekurzivní algoritmus, jehož výsledkem je pole štítků (detekovaných periodických chování). Následně zkoumáme, jak se vyvíjejí jednotlivá periodická chování detekovaná na původní časové řadě.

Na základě analýzy přiřazujeme ke každému periodickému chování detekovanému na původní časové ose následující informace:

1. Informace o procentuálním počtu částí, na kterých bylo detekováno dané periodické chování.

2. Informace o procentuálním počtu bodů původní časové řady, které patří do části, ve které je detekován daný štítek. To znamená, že každá část reprezentuje nějaké procento původní časové řady a výsledek je součet těchto procent u částí, na kterých bylo detekováno toto chování.

3. Informace o tom, jak se vyvíjí detekce na částech.

Nastávat může více vývojů, které klasifikujeme slovními štítky. Například když se nemění je přiřazen štítek *no-change*, a pokud je periodické chování detekováno do nějaké části a pak už ne, tak přiřazujeme štítek *disappearing*. Celkový popis štítků přiřazovaných na základě detekce periodického chování na částech je popsán v přílohách.

5.13 Odvození celkové hodnoty důvěry

Celková důvěra v nalezené periodické chování je spočítána na základě důvěry ve výsledek z celé časové řady a na základě důvěry ve výsledek z analýzy částí časové řady. Vzorec celkové důvěry ve výsledek je:

$$r = \frac{r_{st} + r_{sp}}{2}$$

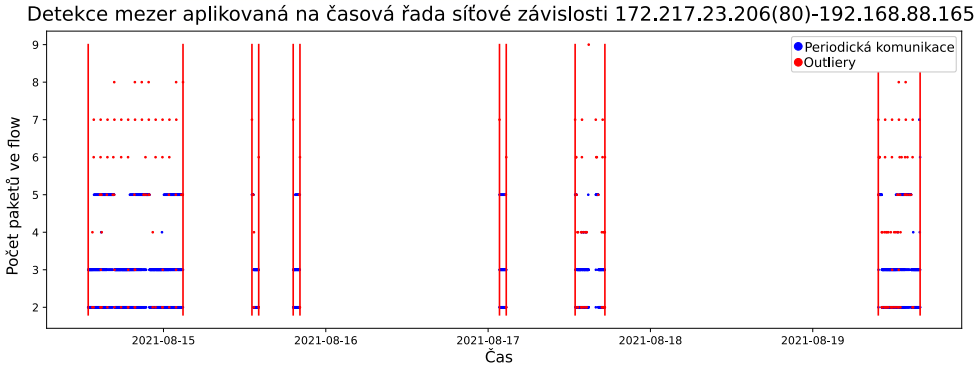
kde r_{st} je důvěra ve výsledek aplikovaného na celou časovou řadu a r_{sp} je důvěra ve výsledek z analýzy částí časové řady. Platí, že $r_{st}, r_{st} \in < 0, 100 >$. A r_{sp} se vypočítá pomocí vzorce:

$$r_{sp} = \frac{1}{n} \sum_{i=1}^{n} r_{ts_i} \frac{len(t_i)}{len(t)}$$

kde n je počet nalezených částí v časové řadě, t_i je i-tá část časové řady, t je celá časová řada, $len()$ je délka (resp. počet datových bodů) časové řady a r_{ts_i} je hodnota důvěry v daný štítek v části t_i, přičemž pokud v části t_i nebyl tento štítek nalezen, definujeme $r_{ts_i} = 0$.

Vzorec pro r_{ts} zohledňuje pomocí $\frac{len(t_i)}{len(t)}$ délku části časové řady t_i, takže čím je daná část větší, tím má důvěra jejího výsledku větší vliv na celkovou důvěru. Sumu zároveň normalizujeme pomocí $\frac{1}{n}$, aby platilo $r_{sp} \in < 0, 100 >$.

Na obrázku 5.15 je vidět, že mezery rozdělily časovou řadu do různě dlouhých úseků, na nichž se s různou důvěrou ve výsledek podařilo detekovat shodné periodické chování. Důvěra ve výsledek na celé časové řadě (r_{ts}) v tomto případě vychází přibližně 83 %. Důvěra ve výsledek z analýzy částí (r_{sp}) je rovna přibližně 67 %. Celková důvěra r je tedy spočtena na přibližně 75 %. Můžete si povšimnout, že r_{sp} je mírně nižší, což je způsobeno tím, že se v časové řadě vyskytují části, které jsou příliš malé na to, aby na nich metoda přes LS periodogram mohla být úspěšná (v těchto částech tedy není detekována žádná periodicita). Ale i přes tyto nedostatky LS periodogramu je výsledná důvěra ve štítek poměrně vysoká.



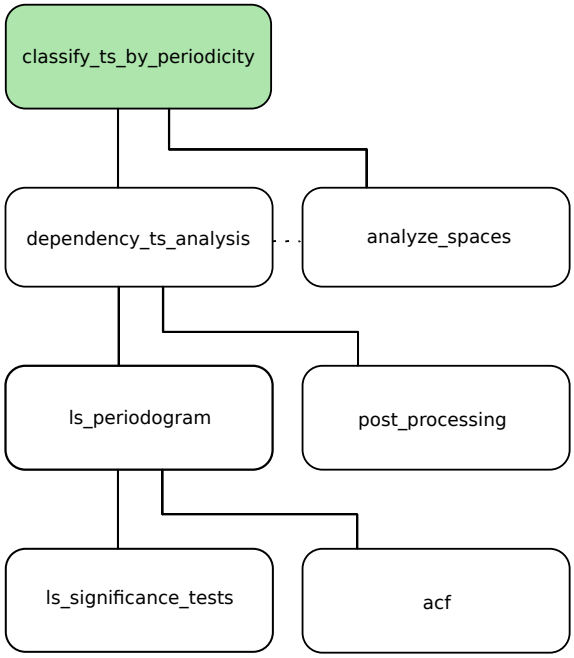
Obrázek 5.15: Příklad časové řady pro odvození celkové důvěry

Kapitola 6

Návrh a implementace modulů metody detekce periodicity

6.1 Návrh metody detekce periodicity

Metodu, která bude vykonávat algoritmus jak je popsán v algoritmech 1 a 2, můžeme rozdělit na jednotlivé moduly zajišťující jednotlivé části algoritmu. Tyto moduly spolu s vazbami, které mezi sebou mají, jsou vyobrazeny na obrázku 6.1.



Obrázek 6.1: Diagram modulů metody detekce periodického chování

Jediný modul, jehož funkce bude volána na časovou řadu, je nazvaný *classify_ts_by_periodicity*. Tento modul zaštiťuje všechny ostatní moduly, předává jim parametry zvolené uživatelem a nakonec vrací nalezená periodická chování spolu se všemi jejich atributy. Moduly a jejich účel jsme sepsali do tabulky 6.1.

Tabulka 6.1: Popis modulů detekce periodicity

Modul	**Funkcionalita**
classify_ts_by_periodicity	zaštiťuje ostatní moduly, předává parametry modulům a sbírá výsledky, které pak vrací
dependency_ts_analysis	zajišťuje provedení rekurzivního volání *ls_peridoogram* modulu s cílem detekce všech periodických chování v časové řadě
analyze_spaces	detekuje statisticky významné mezery a provádí klasifikaci časové řady na základě nich
post_processing	dopočítává časovou periodu a hodnoty metrik pro nalezená periodická chování
ls_periodogram	provádí detekci periodicity pomocí LS periodogramu
ls_significance_tests	aplikuje statistické testy významnosti na LS periodogramu
acf	provádí autokorelační funkci za účelem získání kandidátů na periodicitu

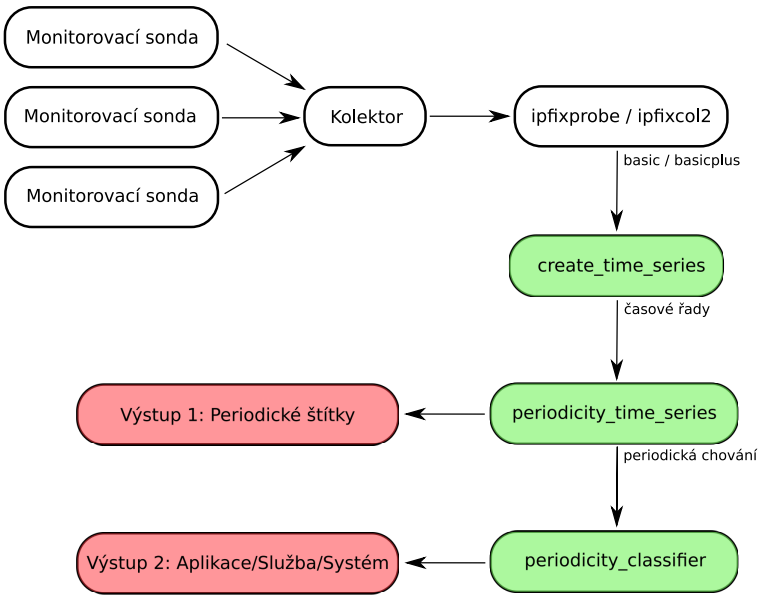
6.2 Návrh modulů pro open source systém NEMEA

Network Measurements Analysis (NEMEA) [1] je navržen s ohledem na streamwise koncept, tím je myšleno, že data jsou průběžně analyzována v paměti s minimálním ukládáním dat. Je vyvíjen jako open-source projekt veřejně přístupný pro celosvětovou komunitu a navržen jak pro experimentální, tak i provozní použití.

Systém NEMEA je navržen jako heterogenní modulární systém. Moduly jsou nezávislé procesy propojené jednosměrnými rozhraními pro komunikaci zvané TRAP Communication Interfaces (IFC). Přičemž každý IFC je vstup nebo výstup modulu a každý modul může mít libovolný počet vstupních a výstupních IFC. Data, která se posílají na IFC, jsou formátovaná do zpráv maximální velikosti 64kB. Rozhraní přenášejí libovolná data ve formě toků zpráv — flow záznamy, výsledky analýzy a další.

Na základě vlastností NEMEA systému a modulů pro něj vytvořených jsme se rozhodli rozdělit naší metodu do tří modulů. První z nich se jmenuje *cre-*

ate_time_series a přijímá flow záznamy od některého z již implementovaného modulu NEMEA systému (*ipfixprobe* nebo *ipfixcol2*), rozděluje je na síťové závislosti a vytváří časové řady, které distribuuje modulu *periodicity_time_series*. Tento modul provede metodu detekce periodicity popsanou v kapitole 5 pomocí knihovny *ls_periodogram_method*.. Výsledná periodická chování pošle na výstup jako seznam štítků a zároveň je distribuuje modulu *periodicity_classification* provádějící klasifikaci dle periodicity popsanou v kapitole 8. Tento modul provede klasifikaci síťové závislosti a výsledek vrátí na výstup. Toto zapojení modulů můžete přehledně vidět na obrázku 6.2



Obrázek 6.2: Diagram zapojení detekce periodického chování do NEMEA sytému

6.3 Implementace

Pro implementaci jsme zvolili programovací jazyk Python verze 3.9 [1] kvůli rychlosti implementace, existujícím knihovnám pro LS periodogram a kompatibilitě s NEMEA systémem. Struktura python modulů je členěna dle návrhů znázorněných na obrázcích 6.1 a 6.2.

Nejdůležitější používané knihovny jsou *pytrap* [2] knihovna NEMEA systému, *numpy* [3] pro efektivní práci s poli, *pandas* [4] pro práci se soubory ve formátu CSV ve fázi klasifikace, *astropy.timeseries.LombScargle* [5] pro efektivní vy-

[1] python.org
[2] github.com/CESNET/Nemea-Framework/tree/master/pytrap
[3] numpy.org
[4] pandas.pydata.org
[5] docs.astropy.org/en/stable/timeseries/lombscargle.html

tvoření LS periodogramu časové řady a *scipy.special.lambertw* [6] pro spočtení Lambertovy $\mathcal{W}_n$ funkce při výpočtu spolehlivosti testu významnosti na LS periodogramu.

6.3.1 Modul create_time_series

Tento modul vytváří časové řady ve formátu *Python dictionary*, kde klíče metriky detekce periodicity a hodnoty jsou pole hodnot.

Po každém intervalu nastaveném v parametru *-S* odesílá všechny časové řady na výstupní NEMEA rozhraní nebo uloží do CSV nebo JSON souboru.

6.3.2 Knihovna ls_periodogram_method

Tato knihovna obsahuje funkci *perform_classification_by_periodicity*, která přijímá časovou řadu ve formátu *Python dictionary*. Pro danou časovou řadu se provede algoritmus metody detekce periodicity na časových řadách ze síťového provozu popsaný v kapitole 5. Výstupem je seznam periodických chování a seznam štítků přiřazených dle statisticky významných mezer.

6.3.3 Modul periodicity_time_series

Tento modul načítá z CSV nebo JSON souboru časové řady ve formátu *Python dictionary*, v němž klíče jsou ID závislostí (*[IP adresa]:[port]-[IP adresa]*) a hodnoty jsou časové řady. Pro každou časovou řadu následně provede algoritmus detekce periodicity pomocí balíčku *ls_peridoogram_method*. Výstupem je seznam periodických chování pro každé ID závislosti.

Při konečném nasazení doporučujeme přidat paralelismus na provádění algoritmu 2, aby se prováděla detekce periodicity na více časových řad současně.

6.3.4 Pomocné skripty

V rámci testování byl vytvořeny pomocné scripty, které mohou být použity pro práci s metodou detekování periodických chování v časových řadách ze síťového provozu bez nutnosti nasazení NEMEA systému do reálné sítě. Tyto pomocné skripty zaštiťuje skript *pcap_to_ts.sh*, který na vstupu přijímá PCAP soubor [7]. Pomocí NEMEA modulu *ipfixprobe* jsou pakety v PCAP souboru převedeny na flow záznamy, které jsou odeslány na NEMEA rozhraní z nějž čte *create_time_series* modul. Ten uloží výsledné časové řady do CSV nebo JSON souboru, který načítá modul *periodicity_time_series*. Výsledky detekce periodicity následně uloží do CSV souboru [17].

[6] docs.scipy.org/doc/scipy/reference/generated/scipy.special.lambertw.html
[7] tcpdump.org/manpages/pcap.3pcap.html

Kapitola 7

Testování metody detekce periodicity

7.1 Experimentální vyhodnocení nastavení parametrů metody

Nejdříve musíme ustanovit vhodné parametry metody detekce periodického chování, které by byly nejvíce univerzální. Souhrn všech parametrů a jejich význam je sepsán v příloze C.

K tomuto účelu jsme provedli experimenty na syntetických i reálných časových řadách. Výsledné parametry jsme nastavili v modulu *classification_ts_by_periodicity* jako výchozí, jak je vyobrazeno v příloze C.

7.1.1 Experimenty na syntetických časových řadách

Pro experimentální vyhodnocení parametrů (především parametrů SCDF testu významnosti na LS periodogramu, *sig_level* a *per_level*, definovaných v podkapitole 5.7) jsme nejdříve uvažovali testy na syntetických časových řadách, které se budou časovým řadám ze síťového provozu podobat co nejvíce. Tedy musí platit, že perioda p značí výskyt p sousedících hodnot, které se opakují.

Ke generování takových časových řad jsme vhodně využili *Python* knihovnu *timesynth* [8]. Díky této knihovně jsme snadno nagenerovali pět datových sad, které jsme nazvali *Clear periodic*, *Noisy periodic*, *Clear constant*, *Noisy constant* a *Random*.

První dvě datové sady, *Clear periodic* a *Noisy periodic*, jsou časové řady s nějakou periodou, které vypadají jako časové řady ze síťového provozu. Při experimentech jsme si všimli, že pokud má časová řada periodu 1, tzn. dá se o ní říct, že má konstantní hodnotu metrik, tak LS periodogram ne vždy funguje správně. Tuto domněnku jsme se rozhodli otestovat dvěma datovými

[8] github.com/TimeSynth/TimeSynth

sadami *Clear constant* a *Noisy constant*. A nakonec jsme vygenerovali největší datovou sadu *Random* obsahující „náhodné“ časové řady.

V tabulce 7.1 jsou výsledky (v procentech detekování periodicity) na těchto časových řadách při změnách nastavení SCDF testu významnosti.

Tabulka 7.1: Testování na syntetických časových řadách bez testu konstantnosti

		per_level					
Typ řad	**sig_level**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**0.975**
Clear periodic	**0.1**	99.8	99.8	99.8	99.8	83.6	56.0
	0.05	99.8	99.8	99.8	99.8	99.8	99.8
	0.01	98.0	95.8	92.4	83.6	72.2	47.4
	0.005	88.0	80.4	70.6	56.0	37.2	18.4
Noisy periodic	**0.1**	100.0	100.0	99.6	98.8	94.6	85.4
	0.05	99.4	97.8	94.2	89.2	78.8	65.8
	0.01	68.2	58.6	48.0	34.0	23.8	14.4
	0.005	41.6	30.8	22.2	17.2	12.0	6.8
Clear constant	**0.1**	0.0	0.0	0.0	0.0	0.0	0.0
	0.05	0.0	0.0	0.0	0.0	0.0	0.0
	0.01	0.0	0.0	0.0	0.0	0.0	0.0
	0.005	0.0	0.0	0.0	0.0	0.0	0.0
Noisy constant	**0.1**	87.4	87.4	87.0	86.0	82.4	75.6
	0.05	86.6	84.2	81.8	78.2	71.6	57.4
	0.01	60.6	52.6	42.8	35.6	23.4	14.8
	0.005	39.6	30.4	22.4	17.0	11.4	6.4
Random	**0.1**	55.15	54.55	52.7	51.1	47.35	37.15
	0.05	52.55	50.6	47.9	42.25	33.1	19.95
	0.01	22.65	15.5	9.75	5.75	3.15	2.35
	0.005	7.45	4.25	3.1	2.7	2.5	2.35

Všimněte si, že konstantní časové řady jsou skutečně problematické. Proto jsme přidali do algoritmu část popsanou v podkapitole 5.5, která zajišťuje nápravu při selhání. S tímto vylepšením jsme provedli testy na datových sadách znovu a výsledky vynesli do tabulky 7.2

Tabulka 7.2: Testování na syntetických časových řadách s testem konstantnosti

		per_level					
Typ řad	**sig_level**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**0.975**
Clear periodic	**0.1**	100.0	100.0	100.0	100.0	100.0	100.0
	0.05	100.0	100.0	100.0	100.0	100.0	98.4
	0.01	98.4	96.2	92.8	84.0	72.6	48.0
	0.005	88.4	80.8	71.0	56.6	37.8	19.0
Noisy periodic	**0.1**	100.0	100.0	99.6	98.8	94.6	85.4
	0.05	99.4	97.8	94.2	89.2	78.8	65.8
	0.01	68.2	58.6	48.0	34.0	23.8	14.4
	0.005	41.6	30.8	22.2	17.2	12.0	6.8
Clear constant	**0.1**	100.0	100.0	100.0	100.0	100.0	100.0
	0.05	100.0	100.0	100.0	100.0	100.0	100.0
	0.01	100.0	100.0	100.0	100.0	100.0	100.0
	0.005	100.0	100.0	100.0	100.0	100.0	100.0
Noisy constant	**0.1**	99.4	99.4	99.2	98.2	95.2	88.6
	0.05	98.8	96.8	94.8	91.2	85.0	72.2
	0.01	75.4	67.4	58.0	50.8	38.8	30.6
	0.005	54.8	45.6	38.0	32.8	27.2	22.4
Random	**0.1**	55.15	54.55	52.7	51.1	47.35	37.15
	0.05	52.55	50.6	47.9	42.25	33.1	19.95
	0.01	22.65	15.5	9.75	5.75	3.15	2.35
	0.005	7.45	4.25	3.1	2.7	2.5	2.35

Lze si všimnout, že test konstantnosti pomohl nejen u čistě konstantních řad, ale také u periodických. To znamená, že generátor u datové sady *Periodic* taktéž vygeneroval periodické řady s periodou rovné 1.

Jak je asi zřejmé, cílem je maximalizovat výsledné procento detekování periodicity na *Clear periodic*, *Noisy periodic*, *Clear constant* a *Noisy constant* datových sadách. A zároveň minimalizovat výsledné procento na *Random* datové sadě. Musíme také poznamenat, že nelze předpokládat dokonalost těchto datových sad v podobnosti k síťovému provozu. Proto je nutné nastavení parametrů testovat i na reálných časových řadách ze síťového provozu a odvodit finální nastavení v závislosti na těchto testech.

7.1.2 Experimenty na reálných časových řadách

Pro odvození „univerzálního“ nastavení pro časové řady ze síťového provozu jsme testovali na reálných časových řadách vytvořených z provozu Laboratoře monitorování síťového provozu FIT ČVUT v Praze. A na jejich základě jsme ručně vyhodnotili, jaké časové řady by měli být označeny jako periodické a jaké nikoliv, a dle toho jsem rozdělili časové řady do dvou datových sad *Periodic* a *Random*. Experimentální vyhodnocení nastavení parametrů testu významnosti jsou vyobrazeny v tabulce 7.3 (výsledky v procentech detekování periodicity).

Tabulka 7.3: Testování na časových řadách z NETMONLAB

		per_level					
Typ řad	**sig_level**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**0.975**
Periodic	**0.1**	100.0	100.0	100.0	100.0	100.0	100.0
	0.05	100.0	100.0	100.0	100.0	97.9	95.2
	0.01	99.1	97.3	97.1	96.9	95.8	93.6
	0.005	98.4	91.1	81.6	76.3	69.3	58.6
Random	**0.1**	100.0	98.1	89.9	79.1	69.1	59.2
	0.05	69.4	51.8	34.2	19.2	5.8	3.8
	0.01	45.6	32.9	23.8	11.2	3.1	1.2
	0.005	22.5	11.3	2.6	0.4	0.01	0.001

7.1.3 Závěrečné nastavení parametrů

Na základě výsledků experimentů jsme se rozhodli nastavit parametry SCDF testu významnosti na:

$$per\_level = 0.95$$

$$sig\_level = 0.05$$

Pro případ, že by uživatel chtěl být při testování přísnější, doporučujeme zvýšit prahovou hodnotu *per_level* například na 0.975.

7.2 Testování metody na síťovém provozu

Poté, co jsme odvodili „univerzální“ nastavení parametrů, jsme vytvořili dvě datové sady z provozu na síti Laboratoře monitorování síťového provozu FIT ČVUT. První zahrnuje všechny závislosti proběhlé během deseti dnů a druhá během osmnácti dnů. Pro všechny závislosti byly vytvořeny časové řady, na které jsme použili metodu detekce periodicity.

Z nahlédnutí do detekovaných periodicit jsme odvodili, že metoda má minimální poměr falešně ohlášených periodicit (angl. False Positive Rate). A je zřejmé, že čím „delší“ je časová řada (větší počet datových bodů, flow záznamů a delší skutečný čas), tím metoda dosahuje lepších výsledků. Naopak pokud metoda dostane „malé“ časové řady (malý počet datových bodů a krátký skutečný čas), tak poměr falešně ohlášených periodicit prudce vzroste. Doporučujeme tedy časové řady alespoň délky v řádech hodin a alespoň s několika desítkami datových bodů.

Po vyfiltrování „malých“ časových řad metoda vykazuje velice kladné výsledky a její časová náročnost je poměrově nesrovnatelně menší než je čas měření. V číslech pro časové řady z měření probíhající deset dnů, se metoda detekce periodicity pro všechny časové řady (bez paralelního provádění více časových řad současně) ukončila po přibližně hodině a půl běhu. Pro časové řady z měření na osmnácti dnech to pak bylo přibližně čtyři hodiny. Metoda i přes implementaci v jazyku *Python* dosahuje rychlosti, při které by byla nasaditelná do reálného provozu středně velké sítě. Pro nasazení do větších sítí či páteřních linek je zapotřebí metodu implementovat v jazyce *C++* s nejrychlejší implementací *FFT* a v této chvíli nejrychlejší známé implementaci LS periodogramu přes *NFFT* (*Nonequispaced FFT*).

7.3 Vyhodnocení náročnosti na výpočetní zdroje

7.3.1 Výpočetní složitost algoritmu

Nejdříve odvodíme přibližnou výpočetní složitost algoritmu. U použitých matematických funkcí je složitost známá a soupis je vyobrazen v tabulce 7.4, kde n je počet datových bodů, m je počet frekvencí a N je přesnost Lambertovy $\mathcal{W}_N$ funkce.

Tabulka 7.4: Výpočetní složitost použitých matematických operací

Operace	**Složitost**
LS periodogram	$\mathcal{O}(n \log m)$
SCDF	$\mathcal{O}(2n)$
Fourierova transformace	$\mathcal{O}(n \log n)$
ACF	$\mathcal{O}(3n \log n)$
Lambertova $\mathcal{W}_N$ funkce	$\mathcal{O}(N^3)$

Jeden běh rekurzivního algoritmu má potom složitost:

$$\mathcal{O}\left(n \log m + 3n \log n + N^3 + (2k+3)n\right)$$

kde n je počet datových bodů, m je počet frekvencí LS periodogramu, k je počet kandidátů na periodu v daném zanoření rekurze, $k \in \{0, ..., 5\}$ a N je přesnost Lambertovy $\mathcal{W}_N$ funkce.

Bez výpočtu spolehlivosti SCDF testu významnosti má jeden běh rekurzivního algoritmu složitost:

$$\mathcal{O}\left(n \log m + 3n \log n + (2k+2)n\right)$$

Obecně platí, že nejvíce výpočetně náročné je vytvoření LS periodogramu.

7.3.2 Časová náročnost algoritmu

Dalším důležitým faktorem je časová náročnost algoritmu v závislosti na podobě časové řady. Největší vliv má počet datových bodů a délka časové řady. Pro zjištění vlivu počtu datových bodů a délky na časovou náročnost algoritmu jsme použili datovou sadu *Campus DNS traffic* publikovanou v [18]. Základní údaje o této datové sadě jsou vyobrazené v tabulce 7.5.

Tabulka 7.5: Vlastnosti datové sady *Campus DNS traffic*

Vlastnost	**Hodnota**
Počet flow záznamů	31 383 381
Počet síťových závislostí	21 726
Maximální počet datových bodů v závislosti	6 495 738
Směrodatná odchylka počtu datových bodů v závislostech	47 779
Průměrný počet datových bodů v závislosti	1 443
Medián počtu datových bodů v závislostech	22
Maximální délka závislosti v sekundách	169 205
Směrodatná odchylka délky závislostí v sekundách	52 907
Průměrná délka závislosti v sekundách	40 243
Medián délky závislostí v sekundách	8 284

V tabulce 7.6 je vyobrazen čas výpočtu v závislosti na počtu datových bodů. Červeně podbarvené řádky označují důležité počty bodů z tabulky 7.5, tedy medián, průměr, směrodatnou odchylku a maximum. Jelikož průměrný čas je 5,6 sekund, tak celý dataset by odhadem mohl trvat něco kolem 121 665 sekund, což je asi 33 hodin. Reálný výpočetní čas celého datasetu je 30 hodin. Připomínám, že dataset reprezentuje DNS provoz ze dvou dnů, takže můžeme předpokládat, že pokud by záznam obsahoval veškerou komunikaci na síti, a ne pouze DNS provoz, tak by implementace metody v jazyce *Python* jistě výrazně překročila dobu měření.

Tabulka 7.6: Testování závislosti počtu datových bodů na čase výpočtu

Počet datových bodů	Čas v sekundách
22	0.3
946	5.6
981	4.1
1 443	5.7
10 000	4.5
20 352	11.2
32 661	10.6
47 779	9.1
811 379	69.9
6 495 738	477.4

Z tabulky 7.6 vyplývá, že výpočetní čas je skutečně značně závislý na počtu datových bodů v závislosti, ale také, že výpočetní čas závisí na více faktorech než jen počtu datových bodů. Pokud si zobrazíme v grafech závislost s počtem datových bodů 43 106 (na obrázku 7.1) a závislost s 32 661 datovými body (na obrázku 7.2), tak si můžeme okamžitě povšimnout, že graf na obrázku 7.1 se od grafu na obrázku 7.2 liší v podstatné vlastnosti. Tou je délka časové řady.

Délka časové řady z obrázku 7.2 je více než trojnásobná oproti časové řadě z obrázku 7.1. Proto jsme provedli další sadu měření, ve které jsme se zaměřili na délku časové řady. Výsledky jsou vyobrazeny v tabulce 7.7, kde jsme opět podbarvili červeně řádky označují důležité délky z tabulky 7.5, tedy medián, průměr, směrodatnou odchylku a maximum.

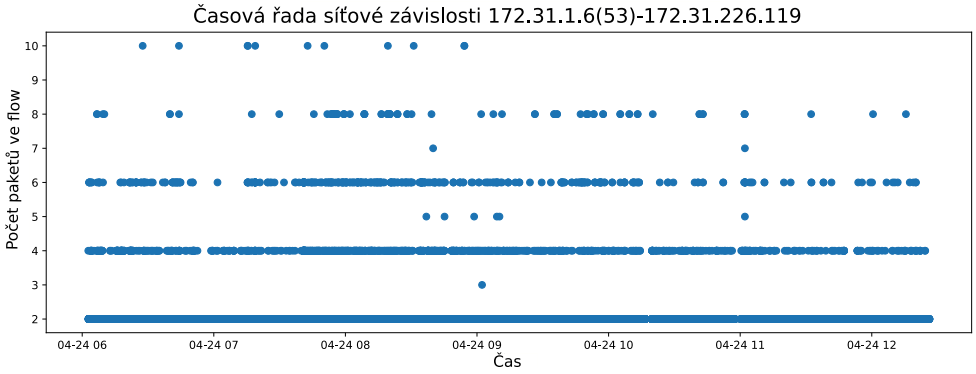
Tabulka 7.7: Testování závislosti délky časové řady na čase výpočtu

Délka časové řady v sekundách	Počet datových bodů	Čas v sekundách
2 582	665	0.09
8 284	1 585	0.70
25 000	4 213	2.19
40 243	3 664	3.03
52 907	5 325	5.02
75 059	5 731	6.45
100 018	4 007	4.36
169 203	6 495 738	477.40

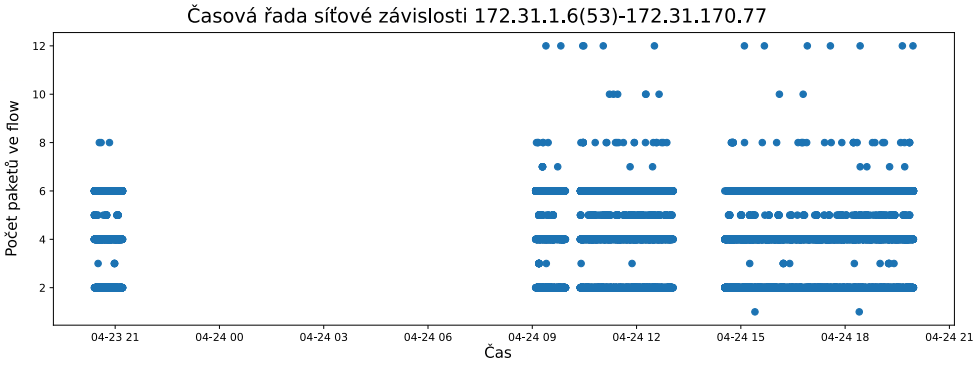
Stojí za zmínku, že jsme vybírali takové časové řady, které obsahují počet bodů úměrně rostoucí s délkou časové řady. Výjimkou je časová řada s maximální délkou, která má počet bodů mnohonásobně větší, což se výrazně projevilo na čase výpočtu.

Pro urychlení zpracovávání časových řad s velkým množstvím datových bodů jsme implementovali heuristiku, která předpokládá, že pokud je časová řada periodická, tak se perioda projeví v prvních q datových bodech. Pokud

metodu použijeme, tak nejpomalejší části algoritmu (LS periodogram a ACF) pracují pouze s prvními q body. Při testování metody jsme volili $q = 500$ a datovou sadu *Campus DNS traffic* algoritmus zpracoval do jedné a půl hodiny při zachování velice podobných výsledků (liší se v drtivé většině případů pouze mírně procenta atributů délky, anomálií v čase, atd.).



Obrázek 7.1: Časová řada s 43 106 datovými body a výpočetním časem 6 sekund



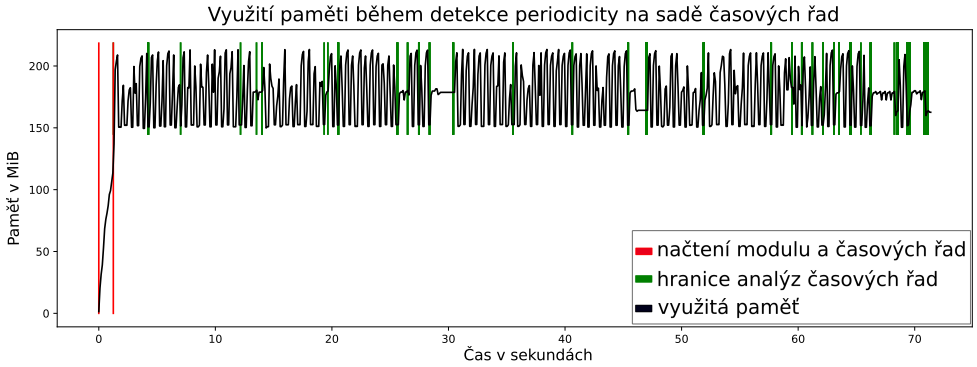
Obrázek 7.2: Časová řada s 32 661 datovými body a výpočetním časem 11 sekund

7.3.3 Paměťová náročnost algoritmu

Paměťová náročnost algoritmu silně závisí na délce analyzované časové řady. Platí, že každý datový bod časové řady obsahuje x-krát prvek, který je typu *int* nebo *float*. Datové typy *int* a *float* zabírají v jazyce *Python* 24 bajtů paměti. Uložení jedné časové řady do paměti stojí $24{*}xn$ bajtů, kde n je počet datových bodů. Při použití metrik počet paketů a bajtů ve flow záznamu bude s časovou informací potřeba $24 * 3n = 72n$ bajtů paměti.

Implementovali jsme dva přístupy k práci s časovými řadami z datových sad. Za datovou sadu považujeme záznam síťového provozu. První pracuje v paměti s celou datovou sadou ve formátu *JSON*, což potřebuje přibližně $72mN$ bajtů, kde N je počet časových řad (počet závislostí) a m je průměrný

počet datových bodů v časových řadách z dané datové sady. Tento způsob je vhodný pro datové sady, u kterých víme, že jejich paměťové nároky nepřesáhnou dostupné prostředky. To jsou většinou datové sady velikosti malého počtu gigabajtů. Vytížení paměti RAM při vykonávání programu na malé datové sadě je vyobrazené na obrázku 7.3. Všimněte si, že na začátku programu se nejdříve načtou knihovny jazyku *Python* a hned poté se načte do paměti celá datová sada ve formátu *JSON*. Tento jev je na grafu označen červenými hranicemi. Poté algoritmus zpracovává jednotlivé časové řady (zelené hranice označují začátek a konec analýzy jedné časové řady). Z grafu lze vyčíst, že vzdálenost mezi začátkem a koncem analýzy je rozdílná pro každou časovou řadu, což je způsobeno velikostí časové řady a počtem přítomných period.



Obrázek 7.3: Využití paměti v průběhu analýzy detekce časových řad na datové sadě

Druhý přístup k práci s časovými řadami je vytvořen s cílem co nejvíce šetřit paměť. Její vytížení je tedy $72n$ bajtů, kde n je počet datových bodů v analyzované časové řadě. Tento způsob pracuje s časovými řadami ve formátu *CSV*, což umožňuje zpracování jednotlivých časových řad a tím minimalizovat náročnost na paměť. Tento způsob je vhodný pro velké datové sady, které se během analýzy nevejdou do paměti RAM.

Kapitola 8

Klasifikace aplikací a služeb

Cílem je využití periodických chování v časových řadách ze síťového provozu k přiřazení (šifrovaného) provozu ke konkrétní aplikaci, službě či operačnímu systému. A to vše pouze na základě času flow záznamu, počtu paketů a počtu bajtů v něm.

8.1 Použité datové sady

NETMONLAB
Čtrnáctidenní záznam sítě laboratoře monitorování síťového provozu na FIT ČVUT v Praze. Síť obsahuje 85 zařízení (46 reprezentovaných pomocí IPv4 adresy a 39 pomocí IPv6 adresy), která vygenerovala 531 328 flow záznamů tvořených 29 281 320 pakety rozdělených mezi 2 731 síťových závislostí. Velikostně se tato síť řadí mezi malé sítě.

COLLECTOR_NEMEA
Čtyřhodinový záznam páteřních propojů akademických sítí univerzit v České republice spravovaných sdružením CESNET. Záznam obsahuje 2 787 956 síťových závislostí, jejichž časové řady jsou omezeny maximálním počtem datových bodů rovným 2 000.

KOUMAJOS_HOME
Datová sada obsahující několik desítek více hodinových experimentálních měření na domácí síti autora práce, které mají za cíl získat záznam síťového provozu konkrétních aplikací. Například MS Teams, FB Messenger, Discord, Slack, Telegram, Skype, Cisco Webex a další.

KOUMAJOS_MOBILE
Datová sada obsahující několik experimentálních měření provozu mobilního telefonu se systémem Android autora práce, která se zaměřují na získání záznamu síťového provozu aplikací a služeb na systému Android.

DNS_CAMPUS_TRAFFIC **[18]**
Datová sada obsahující DNS provoz na síti typu CAN naměřených během jednoho dne. Na síti bylo v době měření aktivních 4000 uživatelů.

8.2 Anotace periodických chování

Pro natrénování nasaditelného klasifikátoru aplikací a služeb bylo nutné anotovat periodická chování na základě aplikace, služby či procesu, který dané periodické chování vygeneroval. K tomu jsme použili poloautomatickou metodu, která využívá doménová jména získaná z *TLS* provozu, logy prohlížečů *Firefox* a *Google Chrome*, majitele veřejné IP adresy a logy využití sítě aplikacemi operačního systému (na linuxu pomocí příkazu *lsof* [9] a na windows pomocí aplikace *tcplogview* [10]).

Při anotaci datové sady bylo velice důležité kontrolovat, zda perioda není generovaná nějakou běžně používanou službou namísto přímo aplikací. Kontrola musela probíhat pomocí aplikace *Wireshark* [11] spolu se zalogovanými *SSL* klíči, tzn. prováděli jsme *Deep Packet Inspection* [19]. Pokud bychom anotovali periodické chování generované nějakou takovou službou dle aplikace, tak bychom zanesli do datové sady značnou chybu. Služby, které byly v našich datových sadách hojně aplikacemi používané, je možné vidět i s hodnotami jejich metrik v tabulce 8.1.

Tabulka 8.1: Běžně používané služby

Proces	Počet paketů	Počet bajtů
TCP Reset Flag / TCP Window Update	1	40
TCP Retransmission / TCP Out-of-Order	1	44
TCP Keep-Alive	1 2	52 80 – 144
TLS Encrypted Alert	2 3 4	111 151 191
TCP Connection Termination	3	120
WebSocket Ping-Pong	3 5	198 – 266 316 – 370
HTTP2 Ping	3 4	234 226 – 300
HTTP2 Goaway	6 7	200 – 500 300 – 900

8.3 Periodicita aplikací a služeb

Po aplikaci metody detekce periodicity na datové sady máme lepší představu o tom, jaké aplikace a služby se chovají periodicky. Z experimentů je jisté, že periodicky se chovají všechny aplikace, ve kterých dochází ke komunikaci

[9] linux.die.net/man/8/lsof
[10] nirsoft.net/utils/tcp_log_view.html
[11] wireshark.org

mezi uživateli. To jsou například všechny sociální sítě, emailoví klienti a herní platformy. Další aplikace většinou pravidelně sledují, zda nedošlo ke změně v externím zdroji. Příkladem je Google Drive, Dropbox, Github a další.

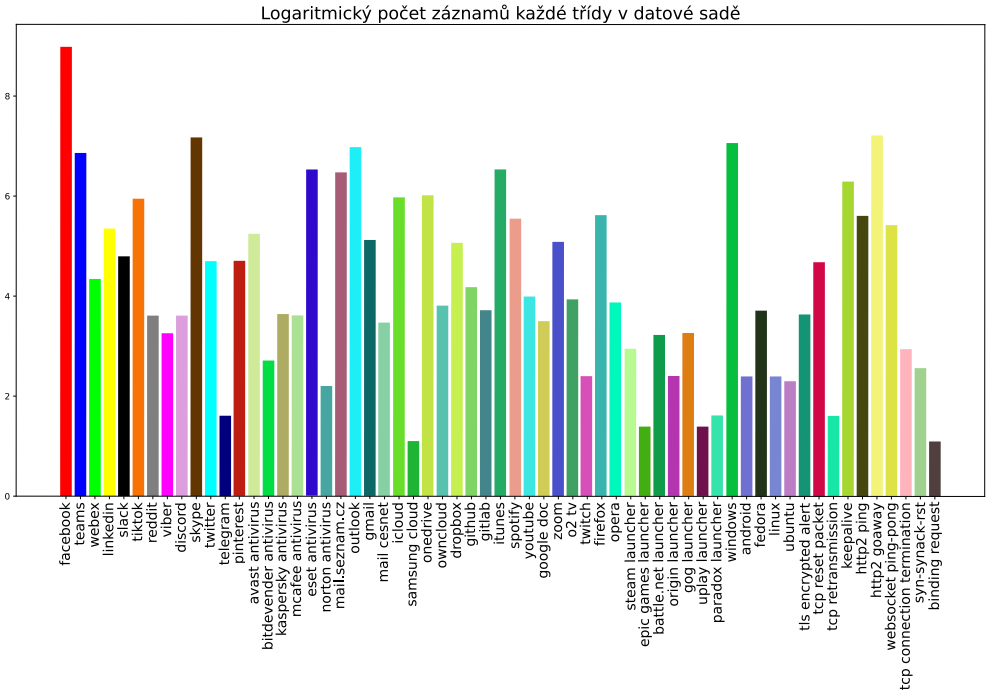
Dalšími zajímavými příklady, které můžeme zmínit, jsou antivirové programy, online přehrávače hudby, aktualizace a další služby operačních systémů, a v neposlední řadě například manažeři hesel s databází uloženou v cloudu.

Mezi služby, které se typicky chovají periodicky, patří *DNS*, *NTP*, *SSDP* či *DHCP*.

8.4 Trénování klasifikátoru aplikací a služeb

Pro natrénování klasifikátoru jsme vytvořili datovou sadu obsahující periodická chování spolu se štítkem reprezentující konkrétní aplikaci, službu či operační systém. Datová sada je reprezentovaná souborem ve formátu CSV, kde řádky jsou periodická chování a sloupce jsou atributy periodického chování nebo atributy časové řady, na které bylo chování detekováno.

Na obrázku 8.1 je vyobrazen logaritmický počet záznamů v datové sadě pro každou klasifikační třídu. Lze podotknout, že datová sada je silně nevyvážená. V datové sadě převládá klasifikační třída *Facebook*, která zahrnuje všechny známé sociální sítě společnosti *Meta*, tzn. *Facebook*, *Messenger*, *WhatsApp* a *Instagram*. Datová sada je vytvořena z reálného provozu, kde aplikace této společnosti jsou uživateli hojně využívané.



Obrázek 8.1: Vizualizace počtu záznamů klasifikačních tříd v datové sadě

Přirozenými parametry pro periodické chování je velikost periody (atribut *flow_period*), která v časových řadách ze síťových toků je v drtivé většině rovna 1, časová perioda (atributy *time_period* a *avg_time_period*) a konkrétní opakující se hodnoty v metrikách (atributy *val_bytes_int* a *val_packets_int*). Statistiky těchto parametrů v datové sadě jsou vypsány v tabulce 8.2.

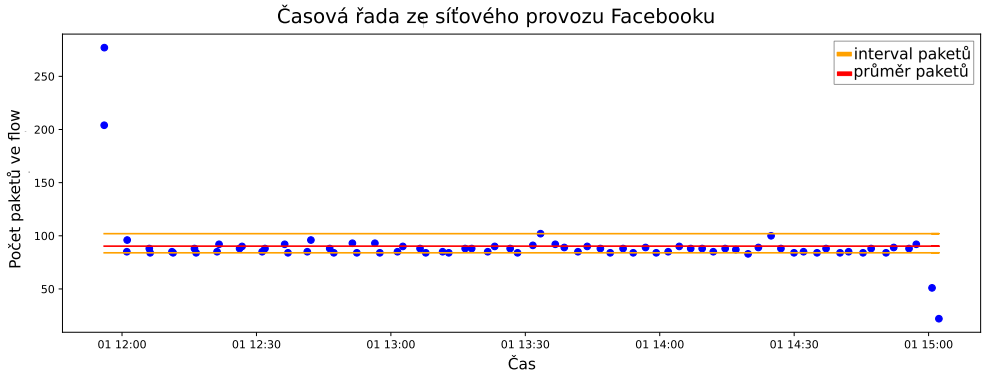
Tabulka 8.2: Statistika hlavních parametrů periodického chování v datové sadě

	Časová perioda	**Průměrná časová perioda**	**Počet paketů**	**Počet bajtů**
Průměr	444.26	709.8	19.4	4 696.8
Směrodatná odchylka	12 265.0	7 874.5	101.5	61 619.4
Minimum	0.0	0.0	1	40
25%	0.0	15.9	8	1 111
50%	0.0	40.5	12	2 862
75%	0.0	102.8	17	4 980
Maximum	710 089.0	255 476.5	11 129	5 294 595

Za upozornění stojí, že atribut *time_period*, který představuje nejvýznamnější časovou periodu, je ve více než 75 % případech roven 0 sekundám. Můžeme tedy odvodit, že tento parametr nám v odhalení aplikací moc nepomůže. Kvůli tomu máme k dispozici průměrnou časovou periodu, která je více rozprostřená a mohla by ve strojovém učení hrát podstatnou roli.

Dalšími atributy v datové sadě jsou takové, které jsme dopočítali v průběhu detekce. Mezi ně patří například atribut určující, kolik procent datových bodů z časové řady je reprezentováno daným periodickým chováním (atribut *length*), dále atribut určující, kolik procent bodů časové řady by mohlo být také reprezentováno daným chováním, ale pouze za předpokladu, že je zde nějaká anomálie v časové periodě (atribut *outliers_in_time*) a atribut znázorňující důvěru v periodické chování. Tyto atributy se v datové sadě pohybují od 0 do 100 %.

Poslední skupinou atributů jsou takové, které jsou dopočítány z časové řady. Mezi ně patří průměry metrik (atributy *mean_packets* a *mean_bytes*), a rozmezí metrik, na kterých se nachází 90 % datových bodů dané metriky (atributy *packets_x* a *packets_y*, a atributy *bytes_x* a *bytes_y*). Tyto atributy jsou pro periodickou časovou řadu vyobrazeny na obrázku 8.2. Tato časová řada je komunikace sociální sítě *Facebook*, která je označena za periodickou, tzn. do datových bodů je LS periodogramem vložena sinusoida. Jak vidíte, v této časové řadě je periodické chování silně závislé na přenášených datech a tak se hodnoty metrik pohybují v určitém intervalu. Takže pro strojové učení nejsou atributy *val_packets_int* a *val_bytes_int* tolik reprezentativní, proto jsme přidali atributy definující interval v každé metrice. Statistiky těchto atributů v datové sadě jsou vypsány v tabulce 8.3.



Obrázek 8.2: Odvozené atributy z časové řady

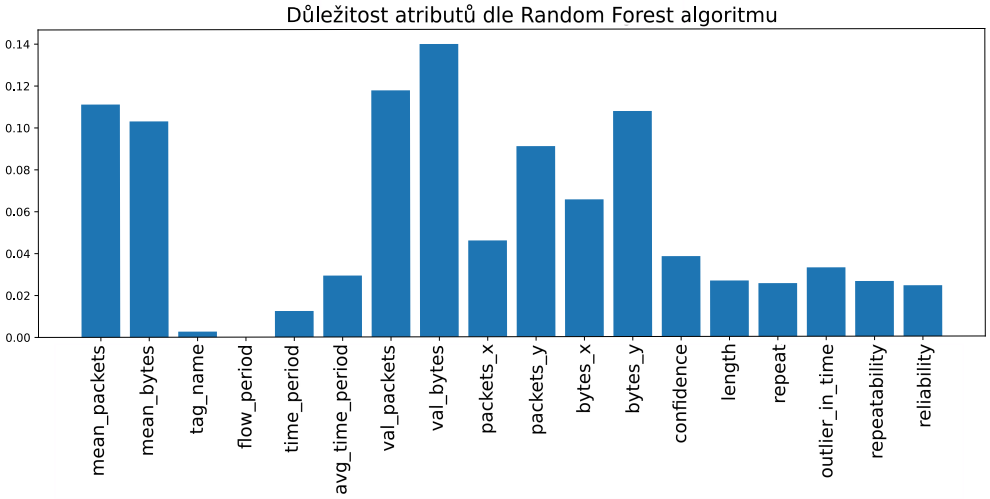
Tabulka 8.3: Statistika odvozených parametrů periodického chování v datové sadě

	Průměr paketů	**Průměr bajtů**	**Rozmezí paketů**	**Rozmezí bajtů**
Průměr	90.15	76955.76	8.66 – 176.34	1 299.45 – 143 625.8
Směrodatná odchylka	692.12	896750.9	17.48 – 1 977.59	13 706.88 – 2 546 945.0
Minimum	1.0	80.85	1.0 – 1.0	40.0 – 52.0
25%	12.94	4 210.61	4.0 – 19.0	291.0 – 6 533.2
50%	25.35	7 030.78	6.0 – 41.0	698.0 – 10 394.0
75%	53.43	18 373.0	9.0 – 83.0	1 319.0 – 25 308.0
Maximum	32 829.39	44 253 310.0	786.0 – 100 959.0	1 135 673.0 – 150 620 600.0

Zde stojí za zmínku, že tyto atributy jsou velice pěkně rozprostřené a proto předpokládáme, že by spolu s hlavními atributy mohly hrát důležitou roli.

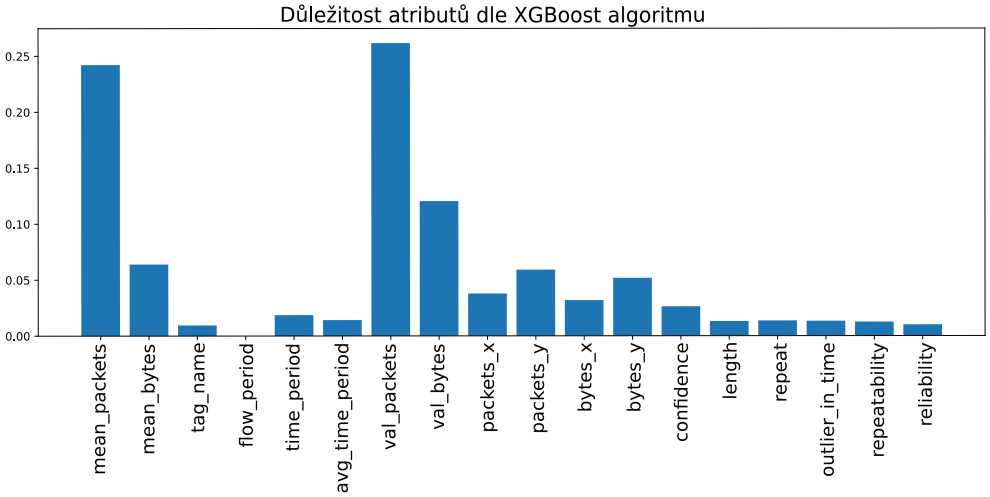
Abychom zjistili důležitost atributů pro náš klasifikační problém, využijeme metodu *feature_importances_* z knihovny *sklearn*. Tato metoda používá natrénovaný model k výpočtu důležitosti jednotlivých atributů ke klasifikaci. Výsledky se většinou mírně odlišují dle zvoleného klasifikačního algoritmu. Graf důležitosti atributů pro *Random Forest* klasifikační algoritmus je vyobrazen na obrázku 8.3.

Lze si povšimnout, že předpoklad, že budou hlavní atributy *val_packets_int* a *val_bytes_int* významné, je pravdivý. Stejně tak se potvrdil předpoklad, že atributy *flow_period* a *time_period* budou málo významné. Dále je nutné upozornit na to, že dalšími podstatnými atributy jsou všechny atributy ze skupiny dopočítané dle časové řady.



Obrázek 8.3: Důležitost atributů dle Random Forest klasifikátoru

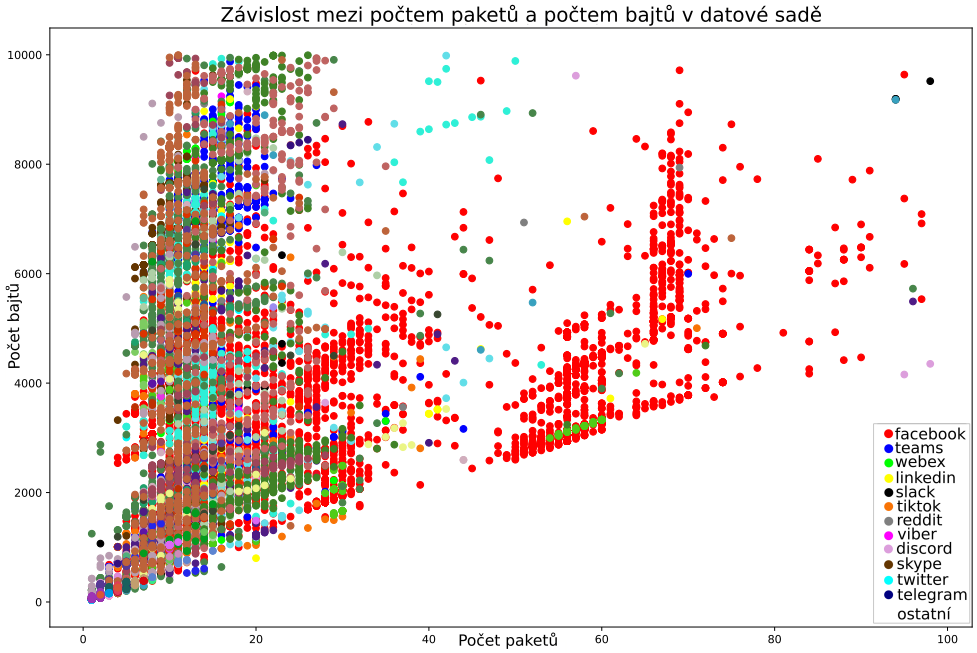
Obdobný výsledek získáme i pro klasifikační algoritmus *XGBoost*, který je znázorněn na obrázku 8.4. Jeden výrazný rozdíly je, že jsou téměř stejně významné argumenty *time_period* a *avg_time_period*. Druhý je, že *flow_period* je mírně významným atributem.



Obrázek 8.4: Důležitost atributů dle Random Forest klasifikátoru

Z obou výsledků je zřejmé, že klíčovou roli hrají atributy *val_packets_int* a *val_bytes_int*. Můžeme si závislost těchto dvou parametrů vizualizovat v závislosti na klasifikační třídě, ke které dané periodické chování náleží. Vizualizace je zobrazená na obrázku 8.5. Pro další nahlédnutí do datové sady můžeme převést významné atributy na procenta a zobrazit v radarovém grafu, jak je znázorněno na obrázku 8.6. Z obou dvou náhledů je zřejmé, že dato-

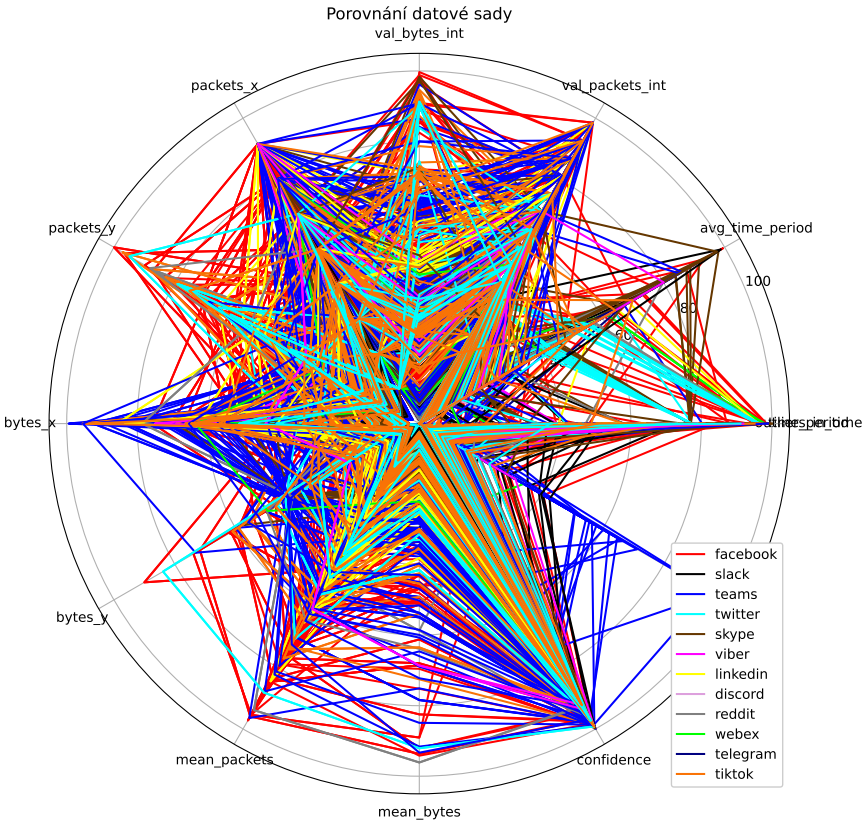
vou sadu bychom mohli nazvat zašuměnou. Nevidíme zde nenarušené shluky jednotlivých klasifikačních tříd a když nahlédneme přímo do dat zjistíme, že bude záležet na absolutních hodnotách každé metriky. Například když uvažujeme dvojici metrik *val_packets_int* a *val_bytes_int*, tak jejich body se příliš nepřekrývají s ostatními třídami. Proto některé klasifikační algoritmy založené na rozhodovacích stromech by mohli být úspěšné, zatímco algoritmy založené na shlucích bodů nejspíš nebudou fungovat s příliš velkou přesností.



Obrázek 8.5: Závislost mezi počtem paketů a počtem bajtů

Na tuto datovou sadu můžeme aplikovat principy předzpracování dat pro zlepšení klasifikace. Datovou sadu náhodně rozdělíme na trénovací a testovací datovou sadu s poměrem 70:30, přičemž zajistíme, aby v trénovací i testovací sadě byly obsaženy všechny klasifikační třídy. Trénovací sadu použijeme k natrénování klasifikátorů. Testovací sadu použijeme k ohodnocení modelu s cílem vybrat ten nejlepší, který prověříme validací.

Mezi testované klasifikátory patří *Random Forest*, *Extra Tree*, *Decision Tree*, *kNN*, *Naive Bayes*, *Logical Regression* a *XGBoost*. Přičemž jsme použili knihovny *sklearn* [20] a *xgboost* [21].



Obrázek 8.6: Porovnání datové sady přes všechny významné atributy

Kapitola 9

Vyhodnocení klasifikace aplikací a služeb

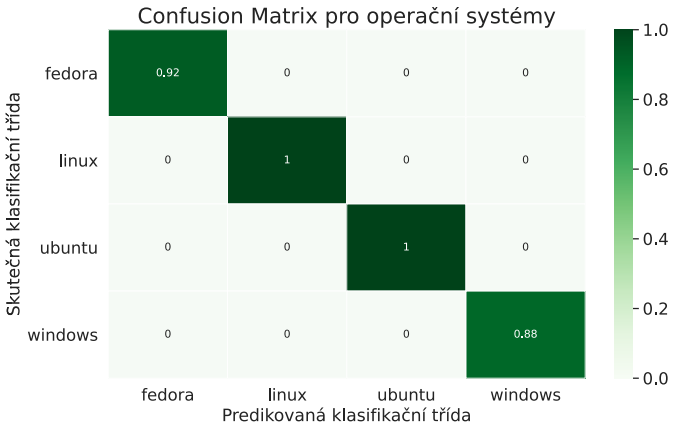
Cílem je klasifikovat aplikace, služby a operační systémy s minimálním *False Positive* a *False Negative*, přičemž nezáleží na jejich poměru. Ideální metrikou pro odhadnutí úspěšnosti klasifikačního modelu pro nás bude *F1-score*. Nicméně pro úplný přehled o úspěšnosti modelu budeme uvažovat také metriky *Precision*, *Recall* a prostou celkovou přesnost modelu (*Accuracy*), tzn. poměr správně klasifikovaných ke špatně klasifikovaným. V tabulce 9.1 jsou vyobrazeny výsledky pro různé klasifikační algoritmy.

Tabulka 9.1: Hodnocení natrénovaných modelů různých klasifikačních algoritmů v procentech

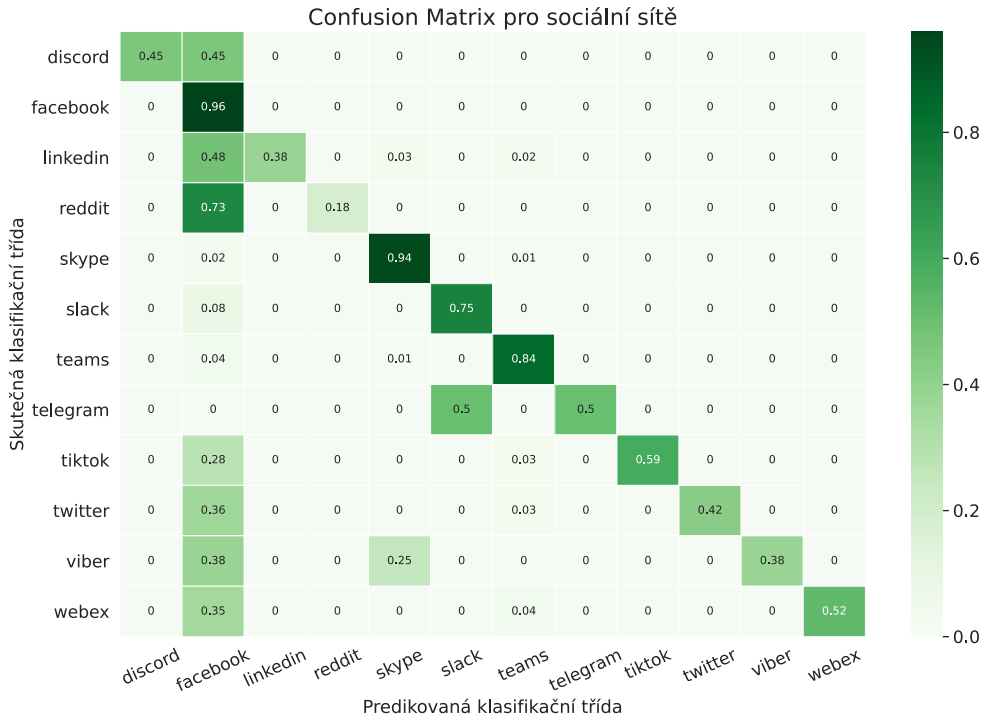
		Accuracy	**Precision**	**Recall**	**F1-score**
Naive Bayes	*macro avg*	8	16	20	10
	weighted avg	8	25	8	4
Logistic Regression	*macro avg*	42	1	2	1
	weighted avg	42	19	42	26
kNN	*macro avg*	62	50	40	41
	weighted avg	62	62	62	61
Extra Tree	*macro avg*	65	61	59	58
	weighted avg	65	66	66	66
Decision Tree	*macro avg*	77	45	46	45
	weighted avg	77	77	77	77
Random Forest	*macro avg*	89	87	64	72
	weighted avg	89	90	89	88
XGBoost	*macro avg*	89	82	66	71
	weighted avg	89	89	89	89

Z tabulky 9.1 je zřejmé, že klasifikační algoritmy založené na rozhodovacích stromech (*Extra Tree*, *Decision Tree*, *Random Forest* a *XGBoost*) mají výrazně lepší výsledky. Srovnatelné se na první pohled zdají modely algoritmů *Random Forest* a *XGBoost*. Pokud pro ně zobrazíme *Confusion Matrix*,

zjistíme taktéž velice obdobné výsledky. Na obrázku 9.1 je znázorněna *Confusion Matrix* omezena na operační systémy v datové sadě. A na obrázku 9.2 je vyobrazena pro sociální sítě. Je na místě si všimnout, že klasifikační třída *Facebook* způsobuje většinu špatných predikcí.



Obrázek 9.1: Vizualizace Confusion Matrix operačních systémů



Obrázek 9.2: Vizualizace Confusion Matrix sociálních sítí

Pro odhalení, jaký z těchto dvou úspěšných modelů je vhodnější, jsme použili validaci zvanou *K-násobná křížová validace* (angl. *K-Fold Cross-Validation*). Při nastavení $K = 10$ je výsledek pro *Random Forest* rovný $0.844 + / - 0.006$ a pro *XGBoost* rovný $0.857 + / - 0.007$.

Dále použijeme metodu obou klasifikátorů *predict_proba*, která pro testovanou instanci vrací pravděpodobnost pro každou klasifikační třídou, s jakou je instance dle modelu danou třídou. V tabulce 9.2 jsou znázorněny výsledky tohoto pokusu, když za konečnou predikci označíme takovou třídu, které je přiřazena pravděpodobnost 90 % nebo vyšší.

Tabulka 9.2: Predikování modelů na základě pravděpodobnosti 90 % nebo vyšší

	Accuracy	**Procento klasifikovaných instancí**
XGBoost	97.05	76.63
Random Forest	99.79	39.53

Z tabulky je vidět, že model pomocí algoritmu *Random Forest* předpověděl klasifikační třídu téměř vždy správně, ale více než 60 % instancí označil jako neznámé (speciální třída *Unknown*). Zatímco model používající *XGBoost* klasifikoval se stále vysokou pravděpodobností mnohem větší procento instancí a jako neznámé označil kolem 24 % instancí.

Závěrem můžeme vzít v potaz i obecné vlastnosti těchto dvou algoritmů. O *XGBoost* se dá prohlásit, že je vhodnou volbou pro nevyvážené datové sady, zatímco *Random Forest* s větší pravděpodobnostní upřednostní třídu, která má více účastí v datové sadě. Naše datová sada je silně nevyvážená, takže celkově je pro nás vhodnější volbou model používající algoritmus *XGBoost*.

Závěr

V práci jsme měli za cíl vytvořit metodu detekce periodicity, která by byla vhodná pro použití na časových řadách vytvořených ze síťových toků. Tento cíl se nám podařilo splnit a navíc je vytvořená metoda vhodná pro použití na jakýchkoliv časových řadách, ve kterých je perioda definovaná jako opakování stejných diskrétních hodnot. Použít tak lze metodu na časové řady vytvořené ze síťových paketů, u kterých může být v některých specifických případech vhodné jejich použití místo časových řad vytvořených ze síťových toků. Metodu jsme otestovali a zvolili jsme vhodné univerzální nastavení pro použití na jakýchkoliv časových řadách ze síťových toků.

Dále jsme dokázali, že detekování periodicity v síťovém provozu má smysl, protože se objevuje u klíčových aplikací, služeb či dokonce malware. Označení komunikace, která se pravidelně opakuje, může být použito administrátory například pro lepší nastavení sítě (například nastavení QoS či nastavení firewallů). I proto, že se periodicky opakuje i malware, je důležité rozlišit jaký proces, aplikace či služba je jejím původcem.

V poslední části práce jsme se proto zaměřili na možnost klasifikace dle detekované periodicity pomocí strojového učení. K tomu jsme vytvořili několik datových sad obsahujících periodicitu známých aplikací, služeb a operačních systémů, a zároveň jsme detekovali periodicitu na veřejně přístupných datových sadách obsahujících komunikaci známých aplikací, služeb, operačních systémů a malware. Výsledkem je datová sada periodických chování, kterou strojové učení využívá ke klasifikaci.

Pomocí datové sady jsme vytvořili modely strojového učení, které klasifikují aplikace, služby a operační systémy dle detekované periodicity. Z modelů jsme vybrali ten nejlepší, který používá algoritmus *XGBoost*. Ten dosahuje *F1-score* kolem 86 % a dále jsme jej validovali pomocí *K-Fold Cross-Validation*. Takto vytvořený model jsme dále testovali na provozu reálné sítě.

Výsledný prototyp detekce periodicity a klasifikace aplikací, služeb a operačních systémů je nasaditelný do malých a středně velkých sítích.

Tato diplomová práce vznikla v rámci projektu FIT ČVUT v Praze, FIT VUT v Brně a sdružení CESNET *„Analýza šifrovaného provozu pomocí síťových toků“* (*FETA*) v rámci výzvy bezpečnostního výzkumu *IMPAKT 1*, Ministerstva vnitra ČR. Obsah této práce bude v rámci tohoto projektu dále vylepšován, aby byl nasaditelný do vysokorychlostních sítí. Detekce periodického chování byla implementována do vlastní knihovny, kterou budou moci používat další aplikace a experimenty tohoto projektu.

Literatura

[1] Čejka, T.; Bartoš, V.; Švepeš, M.; aj.: NEMEA: A framework for network traffic analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*, 2016, s. 195–201, doi:10.1109/CNSM.2016.7818417.

[2] Hofstede, R.; Čeleda, P.; Trammell, B.; aj.: Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*, ročník 16, č. 4, 2014: s. 2037–2064, doi:10.1109/COMST.2014.2321898.

[3] Vlachos, M.; Yu, P.; Castelli, V.: On Periodicity Detection and Structural Periodic Similarity. 04 2005, doi:10.1137/1.9781611972757.40.

[4] Puech, T.; Boussard, M.; D'Amato, A.; aj.: A Fully Automated Periodicity Detection in Time Series. In *Advanced Analytics and Learning on Temporal Data*, editace V. Lemaire; S. Malinowski; A. Bagnall; A. Bondu; T. Guyet; R. Tavenard, Cham: Springer International Publishing, 2020, ISBN 978-3-030-39098-3, s. 43–54.

[5] Hubballi, N.; Goyal, D.: FlowSummary: Summarizing Network Flows for Communication Periodicity Detection. 12 2013, s. 695–700, doi:10.1007/978-3-642-45062-4_98.

[6] Splunder, J. v.: Periodicity detection in Network Trac. Aug 2015. Dostupné z: `https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/mastervansplunder.pdf`

[7] Bartlett, G.; Heidemann, J.; Papadopoulos, C.: Using Low-Rate Flow Periodicities for Anomaly Detection: Extended. Technická Zpráva ISI-TR-2009-661, USC/Information Sciences Institute, Srpen 2009. Dostupné z: `https://www.isi.edu/~johnh/PAPERS/Bartlett09a.html`

[8] Apruzzese, G.; Marchetti, M.; Colajanni, M.; aj.: Identifying malicious hosts involved in periodic communications. 10 2017, s. 1–8, doi:10.1109/NCA.2017.8171326.

[9] Brockwell, P. J.; Davis, R. A.: *Stationary Time Series*. New York, NY: Springer New York, 1991, ISBN 978-1-4419-0320-4, s. 1–41, doi:10.1007/978-1-4419-0320-4_1. Dostupné z: https://doi.org/10.1007/978-1-4419-0320-4_1

[10] Schuster, A.; Moore, H. L.; Douglass, A. E.: *Periodogram analysis*. 1898.

[11] Lomb, N.: Least-squares frequency analysis of unequally spaced data. 1976, doi:10.1007/BF00648343.

[12] Scargle, J.: Studies in astronomical time series analysis. II - Statistical aspects of spectral analysis of unevenly spaced data. *The Astrophysical Journal*, ročník 263, December 1982, doi:10.1086/160554.

[13] VanderPlas, J. T.: Understanding the Lomb–Scargle Periodogram. *The Astrophysical Journal Supplement Series*, ročník 236, č. 1, may 2018: str. 16, doi:10.3847/1538-4365/aab766. Dostupné z: https://doi.org/10.3847/1538-4365/aab766

[14] Zechmeister, M.; Kürster, M.: The generalised Lomb-Scargle periodogram. 2009, doi:10.1051/0004-6361:200811296.

[15] Frescura, F.; Engelbrecht, C.; Frank, B.: Significance Tests for Periodogram Peaks. 2007. Dostupné z: https://www.researchgate.net/publication/1894545_Significance_Tests_for_Periodogram_Peaks

[16] Weisstein, E. W.: Lambert W-Function. Dostupné z: https://mathworld.wolfram.com/LambertW-Function.html

[17] Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC 4180, Říjen 2005, doi:10.17487/RFC4180. Dostupné z: https://rfc-editor.org/rfc/rfc4180.txt

[18] Singh, M.; Singh, M.; Kaur, S.: *10 Days DNS Network Traffic from April-May, 2016*. Mendeley Data, V2, 2019, doi:10.17632/zh3wnddzxy.2.

[19] Bhatt, B.: Understanding Deep Packet Inspection. 2021. Dostupné z: https://www.linkedin.com/pulse/understanding-deep-packet-inspection-babul-bhatt/

[20] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; aj.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, ročník 12, 2011: s. 2825–2830.

[21] Chen, T.; Guestrin, C.: XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4232-2, s. 785–794, doi: 10.1145/2939672.2939785. Dostupné z: http://doi.acm.org/10.1145/2939672.2939785

Příloha **A**

Seznam použitých zkratek

C&C
Command and Control provoz malware

NIDS
Network Intrusion Detection Systems

PSD
Výkonová spektrální hustota (angl. Power spectral density) či Výkonové spektrum (angl. Power spectrum)

ACF
Autokorelační funkce

FFT
Fast Fourier Transform

LS periodogram
Lomb-Scargle periodogram

NFFT
Nonequispaced Fast Fourier Transform

FAP
False Alarm Probability

CDF
Commulative Distributive Function

SCDF
Scargle's Commulative Distributive Function

NEMEA
Network Measurements Analysis

RAM
Random Access Memory

IFC TRAP Communication Interfaces

JSON
JavaScript Object Notation

PCAP
Packet CAPture

CSV
Comma-Separated Values

FIT ČVUT
Fakulta informačních technologií Českého vysokého učení technického v Praze

NETMONLAB
Laboratoř monitorování síťového provozu FIT ČVUT

CAN
Campus Area Network

Příloha B

Popis atributů periodického chování

Výstupem algoritmu detekce periodicity jsou štítky reprezentující detekovaná periodická chování. Příklad takového výstupu a popis atributů můžete vidět v následujícím příkladu:

```
'name': 'Periodic',
'flow_period': 1,
'time_period': 30,
'avg_time_period': 30.0012115483155,
'val_metrics': {
    'packets': [2]
    'bytes': [104],
  },
'confidence': 98.45391413031575,
'length': 96.89265536723164,
'repeat': 921,
'outlier_in_time': 0.3631961259079903,
'repeatability': 97.87460148777896,
'reliability': 97.03506766135021,
'reliability_spaces': 96.580191615275,
'final_reliability': 96.8076296383126,
'in-parts': 100.0,
'in-parts-traffic': 100.0,
'parts-change': 'not-change',
```

name:
název popisující detekované chování.
Může nabývat hodnot: *Periodic*, *Periodic-< metric >*, *Periodic-constant*, *Non-periodic*, *Outlier-in-flow* a *Random-noise*

flow_period:
perioda opakování, tj. jak dlouhý je opakující se vzor v počtu flow (obecně jde o počet datových bodů)

time_period:
nejvýznamnější časová perioda

avg_time_period:
průměrná časová perioda spočtená ze všech výskytů *val_metrics* v časové řadě

val_metrics:
hodnoty všech metrik, které se opakují

confidence:
hodnota vyčíslující jak významné bylo potvrzení/zamítnutí v testu významnosti, například SCDF testu

length:
kolik procent datových bodů z časové řady (flow) vytvořené ze závislosti je tímto štítkem klasifikováno

repeat:
počet opakování detekovaného periodického chování

outlier_in_flow:
kolik procent bodů časové řady by mohlo být také klasifikováno tímto štítkem, ale pouze za předpokladu, že je zde nějaká anomálie v časové periodě (tzn. není jisté, zda je to náhodná komunikace nebo anomálie v čase); tj. jde o množství bodů, které nebyly zahrnuty mezi periodické jen kvůli (drobné) odchylce v čase, ale hodnoty metrik odpovídají.

repeatability:
úspěšnost periodického chování vzhledem k potenciálu na dané délce časové řady

reliability:
spolehlivost tohoto štítku v rámci celé závislosti

reliability_spaces:
spolehlivost tohoto štítku z výsledků na částech rozdělených významně většími mezerami

final_reliability:
celková spolehlivost štítku (podrobný popis jednotlivých hodnot spolehlivosti je uveden v kapitolách níže)

in-parts:
v kolika procentech částí rozdělených významně většími mezerami byl nalezen tento štítek

in-parts-traffic:
v kolika procentech datových bodů časové řady (flow) byl nalezen tento štítek, pokud jej budeme hledat v částech

parts-change:
klasifikace změn nalezení tohoto štítku v částech

Příloha C

Parametry knihovní funkce perform_periodicity_detection

Na základě postupů popsaných v kapitole 5 byla vytvořena Python knihovna *ls_periodogram_method* pomocí, které lze detekovat periodicitu. Tuto knihovnu zaštiťuje funkce *perform_classification_by_periodicity*, která na základě zvolených parametrů provede detekci periodicity na předané časové řadě. Popis této funkce spolu s popisem jednotlivých parametrů je popsán níže.

---

ls_peridoogram_method.perform_periodicity_detection(
data, min_dur=0, min_datapoints=10, len_recurse_threshold=0.05,
sig_test="scdf", per_level=0.95, sig_level=0.05, acf_level=0.2,
max_recurse_depth=5, time_threshold=0.05, length_threshold=None,
space_min_length=0.025, sig_space_threshold=2.5,
metrics=["packets", "bytes"], time_information="t1",
constant_level=0.95, delete_outlier_in_time=False,
enable_part_analysis=True, one_space_threshold=0.1,
constant_compare=0.3, even_deviation=0.5,
traffic_deviation=0.1, enable_space_classification=True,
enable_time_period_simplification=True, enable_plot=False,
enable_confidence=True,delete_candidates=True,
qheuristic=0,min_part_length=0.01,
)

Funkci je předána časová řada vytvořená ze síťového provozu ve specifikovaném formátu (Python dictionary s klíči z parametrů *metric* a *time_information*) spolu s parametry, které nastavují detekci periodicity. Výstupem je dvojice hodnot, kde první obsahuje výsledky detekce periodicity a druhá obsahuje klasifikaci časové řady dle statisticky významných mezer.

Příklad spuštění funkce s přednastavenými volitelnými parametry a výpis relevantních výstupů pro tuto situaci:

```
from ls_periodogram_method import \
            perform\_periodicity\_detection

data = {
       "t1": [...],  # list of floats (unix time)
       "packets": [...],  # list of ints
       "bytes": [...],  # list of ints
}

results, results_by_spaces = \
            perform\_periodicity\_detection(data)

print(results)
```

Výstupem tohoto spuštění jsou detekovaná periodická chování (v tomto případě pouze jedno) jako *Python dictionary*, kde klíče jsou jednotlivé atributy periodického chování:

```
[{
'name': 'Periodic',
'flow_period': 1,
'time_period': 30,
'val': [1],
'val_metrics': {
    'packets': [1],
    'bytes': [233]
},
'confidence': 99.95,
'length': 99.62,
'repeat': 150,
'outliers_in_time': 0.32,
'repeatability': 100,
'reliability': 99.85,
'reliability_spaces': 99.88,
'final_reliability': 99.87,
'in_parts': 100.0,
'in_parts_traffic': 99.99999999999999,
'parts_change': 'no_change',
},]
```

data : python dictionary
Časová řada uložená ve formátu *Python dictionary*, kde klíče jsou jednotlivé metriky určené v parametru *metrics* a jeden klíč reprezentující časovou informaci určený v parametru *time_information* a kde hodnoty jsou pole obsahující hodnotu metriky na pozici i pro časovou informaci uloženou v hodnotě metriky *time_information* na pozici i. Tzn. datový bod časové řady je popsán i-tými prvky těchto polí.

min_dur : int, volitelný parametr, default = 0, $\in \{0, ..., n\}$
Minimální doba trvání časové řady (rozdíl časové informace prvního bodu a posledního bodu) v sekundách, nutná pro provedení klasifikace periodicity. Pokud bude doba trvání menší než hodnota tohoto parametru, pak detekce periodicity nebude provedena a návratová hodnota je prázdné pole.

Příklad použití: V časových řadách ze síťového provozu (závislosti) se často objevují časové řady, které reprezentují jednorázovou komunikaci nahuštěnou na malém časovém úseku. U takových časových řad se periodicita neobjevuje, tímto parametrem můžeme časové řady tohoto typu snadno odfiltrovat.

min_datapoints : int, volitelný parametr, default = 25, $\in \{0, ..., n\}$
Minimální počet bodů v časové řadě nutný pro provedení klasifikace. Pokud bude počet bodů menší než tato hodnota, pak detekce periodicity nebude provedena.

Pro časové řady, které obsahují příliš málo datových bodů, často dochází ke špatnému vyhodnocení Autokorelační funkcí a LS periodogramem a dojde k nesprávnému označení časové řady za periodickou. Tímto parametrem tomu můžeme snadno zabránit.

len_recurse_threshold : float, volitelný parametr, default = 0.05, $\in < 0, 1 >$
Procentuální maximální počet bodů z původní časové řady, který je ještě přípustný pro analýzu periodicity v rekurzivním algoritmu. Pokud je zbytek časové řady menší než tento procentuální počet bodů, pak rekurzivní algoritmus skončí.

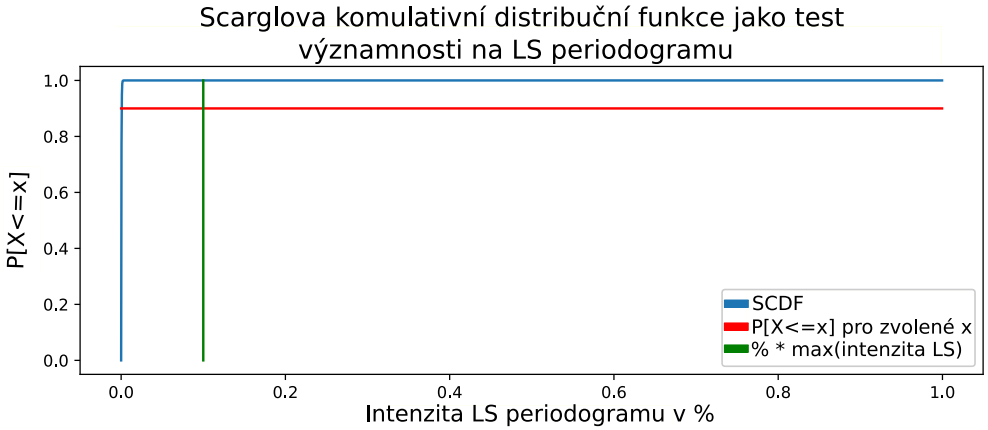
sig_test : str, volitelný parametr, default = "scdf",$\in \{"scdf", "cdf", "fap"\}$
Volba testu významnosti pro kandidáty na periodu na LS periodogramu. Na volbě testu silně závisí parametry *per_level* a *sig_level* (pro různé testy se vhodná hranice může výrazně lišit). Doporučený test významnosti je *SCDF* popsaný v podkapitole 4.5.3. Pro test *FAP* popsaný v podkapitole 4.5.1 se používá pouze jedna prahová hodnota nastavitelná pomocí *sig_level*.

per_level : float, volitelný parametr, default = 0.95, $\in < 0, 1 >$
Prahová hodnota pro test významnosti na LS periodogramu. Tato hodnota nastavuje požadovanou pravděpodobnost, že pro testované z budou ostatní hodnoty menší než toto z (tedy kolik procent hodnot je menší než tato hodnota).

Matematicky se vyjadřuje jako:

$$P[Z < z] = \text{per\_level}$$

Na přiloženém grafu C.1 je vyobrazen SCDF test, u kterého je *per_level* nastaven na 90 % (osa y).



Obrázek C.1: Scarglova kumulativní distribuční funkce

sig_level : float, volitelný parametr, default = 0.05, $\in < 0, 1 >$
Prahová hodnota pro test významnosti na LS periodogramu. Hodnota *sig_level* určuje, kolik procent z největší hodnoty na LS periodogramu se stane hranicí pro podmínku významnosti na periodogramu (zelená čára na grafu C.1). Tuto hranici vypočteme následovně:

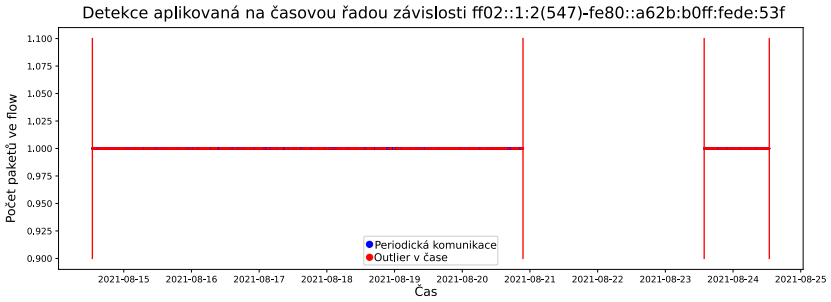
$$s = max(LS\_periodogram) * sig\_level$$

Pokud potom některá z hodnot LS periodogramu menších než je s splní podmínku $P[Z < z] \geq per\_level$ pro $z \leq s$, pak na časové řadě je nějaká významná perioda.

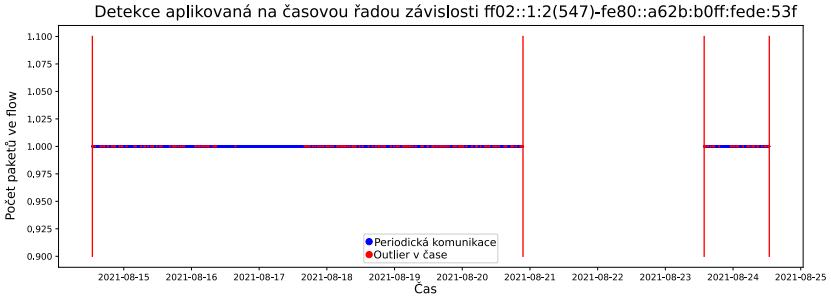
acf_level : float, volitelný parametr, default = 0.2, $\in < 0, 1 >$
Prahová hodnota pro test významnosti na Autokorelační funkci. Určuje, kolik procent ze všech nalezených „lagů“ pomocí Autokorelační funkce musí tvořit určitý lag, aby byl považován za významný a byl přijat jako kandidát na periodu.

max_recurse_depth : int, volitelný parametr, default = 5, $\in \{\textbf{None}, 0, ..., n\}$
Maximální hloubka zanoření rekurze. Znamená také maximální počet periodických chování, které může být přiřazeno jedné časové řadě. Lze vypnout nastavením na hodnotu None.

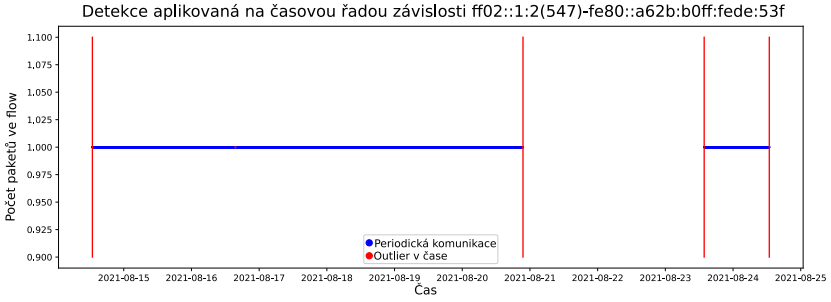
time_threshold : float, volitelný parametr, default = 0.05, $\in <0,1>$
Maximální dovolená odchylka v časové periodě (tj. pokud je bod v čase oproti očekávání posunutý o méně než tuto hodnotu, je ještě považován za součást periodického chování, jinak je označen jako outlier v čase). Určuje se v procentech dané časové periody. Pro ukázku jsou na grafech C.2, C.3 a C.4 vyobrazeny výsledky detekce periodicity při nastavování parametru time_threshold na hodnoty 0.1, 0.2 a 0.3. Lze si povšimnout, že čím větší nastavíme parametr, tím více ubývá v této konkrétní časové řadě anomálií v čase.



Obrázek C.2: Porovnání výsledků při nastavení time_threshold na 0.1



Obrázek C.3: Porovnání výsledků při nastavení time_threshold na 0.2



Obrázek C.4: Porovnání výsledků při nastavení time_threshold na 0.3

length_threshold : float, volitelný parametr, default = None, $\in \{None, < 0, 1 >\}$
Určuje procento z původního počtu datových bodů časové řady, které musí každé detekované periodické chování zahrnovat. Pokud je počet bodů odpovídajících detekovanému periodickému chování menší, je štítek zahozen a rekurzivní algoritmus je ukončen. Výchozí nastavení je vypnuté pomocí nastavení na hodnotu None.

space_min_length : float, volitelný parametr, default = 0.025, $\in < 0, 1 >$
Maximální procento z počtu všech mezer, které může být přijato jako statisticky významné. V kapitole 5.11 ve vzorci a pseudokódu označeno jako l.

sig_space_threshold : float, volitelný parametr, default = 300, ≥ 0
Prahová hodnota pro test, zda se v časové řadě nacházejí statisticky významné mezery.

metrics : list, volitelný parametr, default = ["packets", "bytes"]
Pole metrik, které jsou použity pro analýzu periodicity. Řetězce v poli jsou použity jako klíče do slovníku předaného v parametru *data*.

time_information : str, volitelný parametr, default = "t1"
Určuje, pod jakým klíčem se ve slovníku *data* nachází pole obsahující časovou informaci bodů časové řady.

delete_outlier_in_time : bool, volitelný parametr, default = False
Při detekci periodicity jsou v rekurzivním algoritmu odstraněny body, které detekované chování popisuje. Defaultně se body, které algoritmus popsal jako *Outliers-in-time*, neodstraňují. Tímto parametrem lze zařídit, že jsou z časové řady odstraněny i tyto body. Cílem je, aby nepředstavovaly zbytečný šum pro další detekci periodického chování. Jelikož nikdy není jisté, že body označeny jako *Outliers-in-time* jimi skutečně jsou, tak může nastat situace, kdy periodické chování označí za *Outliers-in-time* body, které jsou generované jiným periodickým chováním. Tento parametr je ve výchozím nastavení vypnutý.

enable_part_analysis : bool, volitelný parametr, default = True
Tímto parametrem se dá vypnout detekce statisticky významných mezer, rozdělení časové řady dle těchto mezer a následná klasifikace každého úseku rekurzivním algoritmem. Při vypnutí se nebude počítat *final_reliability*, jelikož bez analýzy mezer je *final_reliability* rovno *reliability*. Návratová hodnota funkce při vypnutí je definována jako:

```
return results, None
```

one_space_threshold : float, volitelný parametr, default = 0.1, $\in < 0, 1 >$
Tento parametr je prahovou hodnotou pro případ, kdy se v časové řadě objeví pouze jedna statisticky významná mezera. Prahová hodnota určuje procentuální hodnotu k časové délce celé časové řady. Výsledná hodnota se porovná s mezerou a v případě, že je mezera větší, je časová řada oklasifikována štítkem *With-one-large-space*. Jinak se použije štítek *With-one-small-space*.

constant_compare : float, volitelný parametr, default = 0.3, $\in < 0, 1 >$
Parametr definuje odchylku pro určení kolik statisticky významných mezer je stejně velkých, větších a menších než je průměrná statisticky významná mezera. Toto porovnání probíhá kvůli případnému zanedbání anomálně velkých mezer při určení, zda má časová řada statisticky významné mezery podobně nebo náhodně velké.

even_deviation : float, volitelný parametr, default = 0.5, $\in < 0, 1 >$
Tento parametr definuje odchylku pro porovnání, zda jsou všechny statisticky významné mezery stejně velké.

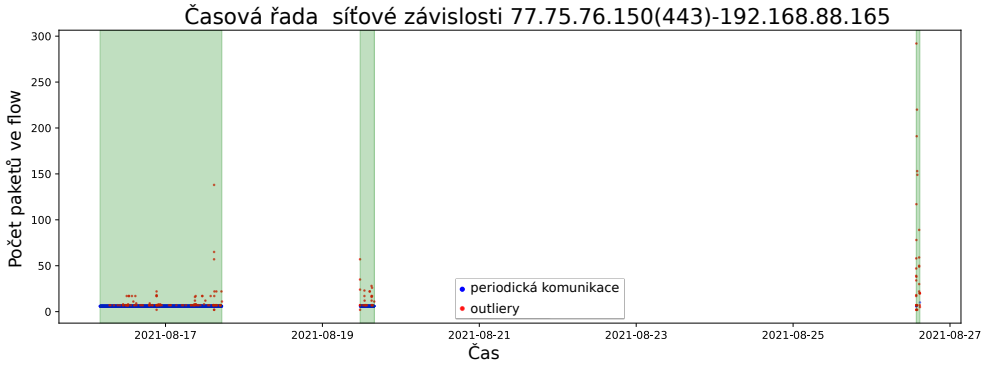
traffic_deviation : float, volitelný parametr, default = 0.1, $\in < 0, 1 >$
Tento parametr definuje odchylku pro porovnání, zda jsou všechny části časové řady rozdělené statisticky významnými mezerami stejně velké.

enable_space_classification : bool, volitelný parametr, default = True
Tímto parametrem lze vypnout klasifikaci dle statisticky významných mezer. Takže se sníží počet atributů periodického chování a návratová hodnota funkce při vypnutí je definována jako:

```
return results, None
```

enable_time_period_simplification : bool, volitelný parametr, default = True
Tímto parametrem lze vypnout funkci zjednodušování časové a flow periody. Ta funguje tak, že pokud se v detekované flow periodě opakují stejné hodnoty (stejné flow záznamy), je flow perioda zjednodušena na opakování jednoho flow záznamu. Tento parametr je defaultně zapnutý, ale doporučujeme ho vypnout v případě, že je funkce použita pro detekci periodicity na jiných datech než na časových řadách vytvořených z komunikace na síťové závislosti. V takových případech totiž nemůžeme zaručit, že jiné informace, než je flow perioda, jsou korektní.

enable_plot : bool, volitelný parametr, default = False
V rámci běhu funkce je vytvořen graf a zobrazen uživateli. V grafu se zobrazuje časová řada (její metrika, jenž je zapsána jako první v *metrics*), spolu se všemi datovými body jenž jsou „outliery“ (*Outliers-in-time*, *Outliers_in_flows*, *Random-noise*) a spolu s rozdělením na statisticky významné mezery. Příklad takového výstupu můžete vidět na grafu C.5.



Obrázek C.5: Příklad grafového výstupu modulu detekce periodicity

enable_confidence : bool, volitelný parametr, default = True
Tímto parametrem lze vypnout výpočet spolehlivosti statistického testu významnosti popsané v podkapitole 5.7.

delete_candidates : bool, volitelný parametr, default = True
Tímto parametrem lze vypnout mazaní kandidátů na periodicitu, kteří jsou dělitelní jinými kandidáty.

qheuristic : int, volitelný parametr, default = 0, $\in \{0, ..., n\}$
Tento parametr slouží ke zrychlení detekce periodického chování na časové řadě tím, že do detekce vloží prvních q datových bodů časové řady. Jedná se tedy o heuristický odhad výsledku, který počítá s tím, že pokud je časová řada periodická s periodou p, pak se s vysokou pravděpodobností projevila perioda i na prvních q bodech. Ve výchozím nastavení je heuristika vypnutá nastavením hodnoty parametru na 0.

min_part_length : float, volitelný parametr, default = 0.01, $\in < 0, 1 >$
Tímto parametrem lze nastavit minimální velikost části časové řady, kterou lze oddělit od časové řady pomocí statisticky významných mezer. Velikost se určuje procentuálním poměrem k původní časové řadě.

Příloha D

Instalace a příklady spouštění modulů

D.1 Instalace modulů

V této části si ukážeme, jak vzniklé moduly nainstalovat pro běh. Veškeré *Python* moduly a *Jupyter* notebooky předpokládají *Python* verze *3.9* nebo vyšší. Kód byl vyvíjen a testován na operačním systému *Ubuntu 21.10*.

D.1.1 Knihovna ls_periodogram_method

```
pip3 install setuptools

cd ls_periodogram_method/

python3 setup.py bdist_wheel

cd ..

pip3 install —e ls_periodogram_method
```

D.1.2 Jupyter notebooky

```
pip3 install Jupyter

cd classifications

pip3 install —r requirements.txt

jupyter notebook xgboost.ipynb
```

D.1.3 NEMEA moduly

Pro spuštění modulů je třeba mít nainstalovaný *NEMEA* systém [12] pomocí:

```
apt-get install -y gawk bc autoconf automake gcc g++ \
            libtool libxml2-dev make pkg-config libpcap-dev \
            libidn11-dev bison flex

git clone --recursive https://github.com/CESNET/nemea

./bootstrap.sh

./configure --enable-repobuild --prefix=/usr \
            --bindir=/usr/bin/nemea --sysconfdir=/etc/nemea \
            --libdir=/usr/lib64

make

make install
```

A taktéž je třeba nainstalovat flow exportér *ipfixprobe* [13] pomocí:

```
git clone --recurse-submodules \
            https://github.com/CESNET/ipfixprobe

cd ipfixprobe

autoreconf -i

./configure --with-nemea --with-pcap

make

sudo make install
```

Nakonec je třeba nainstalovat *Python* knihovny potřebné pro moduly.

```
pip3 install -r requirements.txt
```

[12] github.com/CESNET/Nemea
[13] github.com/CESNET/ipfixprobe

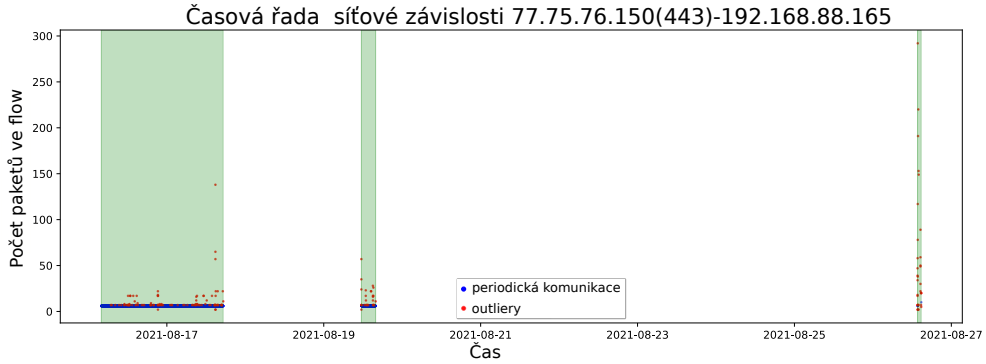
D.2 Příklad použití knihovny ls_periodogram_method

Po instalaci je možné importovat knihovnu *ls_periodogram_method*, resp. knihovní funkci *perform_periodicity_detection*, do *Python* skriptu nebo *Jupyter* sešitu pomocí:

```
from ls_periodogram_method import perform_periodicity_detection
```

Použití knihovní funkce pro detekci periodicity na časové řadě s časem datových bodů v metrice *time* a o dvou metrikách *packets* a *bytes*:

```
time_serie = {
    "packets": [...],
    "bytes": [...],
    "time": [...],
}
results = perform_periodicity_detection(
    time_serie,
    metrics=["packets","bytes"],
    time_information="time",
    enable_plot=True
)
print(results)
```



Obrázek D.1: Výstup spuštěného příkladu

(['name': 'Periodic-constant', 'flow_period': 1, 'time_period': 35, 'avg_time_period': 34.9, 'val': [6], 'val_metrics': 'bytes': [2810], 'packets': [6], 'confidence': 100.0, 'length': 95.5, 'repeat': 4171, 'outliers_in_time': 0.2, 'repeatability': 95.8, 'reliability': 96.7, 'reliability_spaces': 75.68819672930746, 'final_reliability': 86.1, 'in-parts': 33.3, 'in-parts-traffic': 88.3, 'parts-change': 'change': 'disappearing', 'val': 1629213342.312, 'name': 'Random-noise', 'flow_period': None, 'time_period': None, 'val': None, 'confidence': None, 'length': 4.1, 'outliers_in_time': None], ['Randomly-large-spaces', 'Randomly-large-parts'])

D.3 Příklady použití modulů

D.3.1 create_time_series modul

Tento modul očekává na vstupu IFC rozhraní, které je výstupem modulu *ipfixprobe* [14] pracujícím s pluginem *basicplus*. Výstupem modulu jsou časové řady sestavené dle síťových závislostí.

Časové řady lze ukládat do souborů ve formátu *CSV* (parametrem *-c*) nebo ve formátu *JSON* (parametrem *-j*). Parametrem *-S* určujeme, po jaké době jsou časové řady uloženy do jednoho souboru. Ve výchozím nastavení je parametr nastaven na 0 což značí, že bude časové řady vytvářet, dokud bude flow exportér běžet nebo nebude modul ručně vypnut. Pokud používáme jako výstup *CSV* soubor, můžeme specifikovat maximální objem dat v časových řadách uchovávaný v RAM paměti. Modul pak průběžně časové řady ukládá do souboru, čímž je umožněno vytváření velkého množství časových řad s velikostí, která nepřesáhne prostředky počítače (velikost paměti RAM).

Nejdříve musíme spustit exportér síťových toků *ipfixprobe* s nějakým nastaveným vstupem (síťové rozhraní, *PCAP* soubor, ...) a s výstupem v podobě NEMEA rozhraní, které je nastaveno na plugin *basicplus*.

```
sudo ipfixprobe —i 'pcap;file=file.pcap' \
         —o 'unirec;i=u:biflows:timeout=WAIT;e' —p basicplus
```

Spuštění samotného modulu na tvorbu časových řad vypadá následovně.

```
./create_time_series.py —i u:biflows —c timeseries.csv \
         —p Ports.csv —S 0 —q 0 ——size=100000000
```

Časové řady jsou poté uloženy v *CSV* souboru *timeseries.csv*, který má pro každou časovou řadu 8 políček obsahujících ID závislosti, celkový počet flow, celkový počet paketů, celkový počet bajtů, pole počtu paketů ve flow, pole počtu bajtů ve flow, pole počátečního času flow a pole koncového času flow.

```
224.0.0.251(5353)—169.254.252.4;20;40;3440;[...];[...];[...];[...]
```

D.3.2 periodicity_time_series modul

Tento modul načítá časové řady z *CSV* nebo *JSON* souboru a následně na ně volá knihovní funkci *perform_periodicity_detection* vhodně nastavenou pro klasifikaci periodických chování.

Modul očekává určení vstupního souboru pomocí parametru *–csv* nebo *–json* a v parametru *-p* zadání výstupního souboru pro detekovaná periodická chování. Pokud modul chceme použít k anotovaní, tak pomocí parametru *-c*

[14] github.com/CESNET/ipfixprobe

zadáme cestu k souboru, ve kterém se nachází anotace spolu s ID závislosti. Také můžeme všechna periodická chování anotovat stejným štítkem pomocí parametru *-F*. Pomocí parametrů *-s* a *-n* můžeme k periodickým chováním přidat informaci o zdroji (například označení sítě nebo jméno *PCAP* souboru).

Příklad spuštění modulu na časové řady uložené v *CSV* souboru:

```
./periodicity_time_series.py -p periodicity.csv \
    --csv timeseries.csv -n "office network" -s "firm A"
```

Po provedení příkazu jsou v *CSV* souboru *periodicity.csv* uložená periodická chování, která obsahují 29 políček z nichž 22 nějakým způsobem klasifikuje dané periodické chování či časovou řadu, na kterém bylo chování detekováno. Příkladem políček mohou být *flow_period*, *time_period*, *val_packets*, *val_bytes*, *tag_by_spaces* a *tag_by_traffic*.

D.3.3 pcap_to_periodicity skript

Tento skript zajišťuje snadnou práci s *PCAP* soubory, s jejichž pomocí jsme vytvářeli datovou sadu pro detekci aplikací, služeb a operačních systémů. Skript spustí flow exportér *ipfixprobe* s cílem převést pakety z *PCAP* souboru na flow záznamy, které odešle prostřednictvím NEMEA rozhraní. Dále skript zajistí, že z daného NEMEA rozhraní čte flow záznamy *create_time_series* modul, který zpracovává flow záznamy, dokud exportér nenačte poslední paket z *PCAP* souboru. Nakonec skript spustí *periodicity_time_series* modul, který pracuje s časovými řadami uloženými v souboru.

Ukázka použití skriptu:

```
./periodicity_time_series.py -r file.pcap -P periodicity.csv \
    -c timeseries.csv -n "office network" -s "firm A" -O
```

Tento příklad má stejný efekt, jako kdyby byly zadány příklady spuštění *create_time_series* a *periodicity_time_series* modulů.

Navíc můžeme si práci ještě více ulehčit a zadat pouze:

```
./periodicity_time_series.py -r file.pcap -s "firm A" -O
```

Skript automaticky uloží časové řady do souboru *file.pcap.timeseries.csv* a periodická chování do souboru *file.pcap.results.csv*. Navíc nastaví sám parametr *-n* na jméno *PCAP* souboru.

D.3.4 periodicity_classifier modul

Tento modul používá natrénovaný model strojového učení ke klasifikování periodických chování na aplikaci, službu nebo operační systém. Na vstupu očekává *CSV* soubor, který je výstupem z modulu *periodicity_time_series*. Modul umožňuje klasifikaci pomocí dvou různých modelů strojového učení a to *XGBoost* a *Random Forest*.

Příklad spuštění modulu:

```
./periodicity_classifier.py -p file.pcap.results.csv -t 0.9 \
    --model=xgboost --model_file=model/xgboost.json \
    --classes=model/xgboost_classifications.json
```

Výstupem jsou první tři nejvíce pravděpodobné štítky pro každou síťovou závislost, která má přiřazený štítek s pravděpodobností alespoň 90 % (nastavené v parametru -*t*):

```
142.251.36.74(443)-192.168.1.102
    keepalive     0.96
    http2 ping    0.02
    facebook      0.01
```

Příloha **E**

Obsah přiloženého DVD

- `readme.txt` ..... stručný popis obsahu DVD
- `src`
  - `classifications` ..... jupyter notebooky
  - `csv_databanks` ..... datové sady
  - `ls_periodogram_method` ..... knihovna metody detekce periodicity
    - `src` ..... zdrojové kódy implementace
      - `ls_periodogram_method` ..... knihovní moduly
    - `test` ..... testování knihovny detekce periodicity
  - `model` ..... modely strojového učení
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `create_time_series.py` .....
  - `pcap_to_periodicity.sh` .....
  - `periodicity_classifier.py` .....
  - `periodicity_time_series.py` .....
  - `Ports.csv` .....
  - `requirements.txt` .....
- `text` ..... text práce
  - `DP_Koumar_Josef_2022.pdf` .....