Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Mesterséges Intelligencia Tanszék

Buga Péter

TÖBBUTAS HÁLÓZATI FOLYAMOK FELÜGYELETI ADATAINAK FELDOLGOZÁSA ÉS GRAFIKUS MEGJELENÍTÉSE

KONZULENS

Dr. Maliosz Markosz

BUDAPEST, 2025

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Buga Péter**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem. A Függelékben egyértelműen megjelöltem, hogy a mesterséges intelligencia eszközeit alkalmaztam-e a dolgozat elkészítéséhez; amennyiben igen, annak módját és mértékét a táblázatban közöltem. Tudomásul veszem, hogy a mesterséges intelligenciával generált tartalomért – annak mértékétől függetlenül – teljes felelősséggel tartozom.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2025. 12. 11.

...…….

Buga Péter

Összefoglaló

A kritikus fontosságú hálózati alkalmazások, mint az ipari automatizálás vagy a járműipari kommunikáció, megkövetelik a kiemelkedő megbízhatóságot. A Time-Sensitive Networking (TSN) és Deterministic Networking (DetNet) szabványok ezt olyan redundancia-mechanizmusokkal (FRER, PREOF) biztosítják, amelyek többszörözött adatfolyamokat használnak. Bár ez növeli a rendelkezésre állást, a hálózat komplexitása miatt az üzemeltetés és a hibakeresés nehézkessé válik.

Szakdolgozatom célja egy olyan vizualizációs és előrejelző keretrendszer megvalósítása, amely valós időben dolgozza fel a BME Távközlési és Mesterséges Intelligencia Tanszék (TMIT) által fejlesztett hálózati szoftver telemetria adatait. A rendszer mikroszolgáltatás-alapú architektúrára épül és Docker konténerekben fut. A központi elem egy egyedi Python modul, amely a hálózati eszközök UDP üzeneteit gyűjti és továbbítja a Prometheus idősor-adatbázisba. A vizualizációt Grafana dashboard-ok (áttekintő felületek) biztosítják, amelyek átfogó képet adnak a hálózati topológiáról, a redundáns útvonalak állapotáról és a kritikus metrikákról.

A reaktív felügyelet mellett a rendszer támogatja a proaktív hálózatkezelést is egy ARIMA-alapú előrejelző modul segítségével. Ez a komponens a múltbeli forgalmi adatok alapján becsüli meg a jövőbeli tendenciákat. A Mininet környezetben végzett tesztek igazolták, hogy a rendszer stabilan működik, és az előrejelző modell dinamikus hálózati viszonyok között is megfelelő pontossággal képes támogatni az üzemeltetési döntéseket.

Abstract

Mission-critical network applications, such as industrial automation and automotive communication, demand high reliability. Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet) standards address this through redundancy mechanisms (FRER, PREOF) that utilize replicated data streams. While this increases availability, the resulting network complexity makes operation and troubleshooting challenging.

The objective of this thesis is to implement a visualization and forecasting framework that processes telemetry data from network software developed at the Department of Telecommunications and Artificial Intelligence at BME in real-time. The system is built on a microservices architecture running in Docker containers. A custom Python module collects UDP messages from network devices and forwards them to a Prometheus time-series database. Visualization is provided by Grafana dashboards, offering a comprehensive view of the network topology, the status of redundant paths, and critical metrics.

In addition to reactive monitoring, the system supports proactive network management through an ARIMA-based forecasting module. This component estimates future trends based on historical traffic data. Tests performed in a Mininet environment confirmed the system's stability and the forecasting model's ability to accurately support operational decisions even under dynamic network conditions.

1 Bevezetés

1.1 A feladat értelmezése

Az ipari, távközlési és egyéb kritikus alkalmazások hálózati átvitelének megbízhatósága kiemelt fontossággal bír. Az egyik jól bevált megoldás erre a probléma megoldására az adatfolyamok többszörözésén alapuló redundancia alkalmazása. Ennek elve: az adatfolyamot több redundáns útvonalon keresztül továbbítjuk, és ahol ezek az útvonalak egyesülnek, ott kiszűrjük a duplikátumokat és helyreállítjuk az eredeti sorrendet. Ez az elv alapja a Frame Replication and Elimination for Reliability (FRER) [1] mechanizmusnak, amely az IEEE 802.1CB szabvány része és a Time-Sensitive Networking (TSN) keretrendszer szerves részeként működik. Hasonló elvek alapján működik a Packet Replication, Elimination and Ordering (PREOF) [2] is, amely a Deterministic Networking (DetNet) technológia komponenseként működik.

A Budapesti Műszaki és Gazdaságtudományi Egyetem TMIT tanszék hálózati csoportja kifejlesztett egy szoftvert, amely képes a fent említett szabványosított technológiáknak megfelelően kezelni a többszörözött adatfolyamokat megadott hálózati csomópontokban. Ez a szoftver működése során az adatok továbbításán felül felügyeleti információkat is szolgáltat. Ezek az adatok betekintést adnak az adatátvitel belső folyamataiba, beleértve az átvitt adatfolyamok számlálóit, a többszörözés és eliminálás állapotát, valamint a csomóponton belüli felügyeleti pontok (Maintenance Intermediate Points – MIP) adatait.

Ahhoz, hogy a többutas hálózati átvitel különböző ágait és azok működésének részleteit könnyen lehessen vizsgálni és elemezni, szükséges egy olyan komplex vizualizációs és elemzési rendszer, amely képes:

- a hálózat topológiáját grafikusan ábrázolni,
- a felügyeleti adatokat strukturáltan tárolni és feldolgozni,
- a felügyeleti adatokat valós idejű és történeti grafikonok segítségével megjeleníteni,
- a kritikus hálózati eseményeket előre jelezni a korábbi adatok alapján.

1.2 A tervezés célja

A diplomaterv célja egy olyan vizualizációs és előrejelzési rendszer megtervezése és implementálása, amely a FRER/PREOF protokollokat megvalósító hálózati csomópontok felügyeleti adatait képes feldolgozni, tárolni és vizualizálni. A keretrendszer fő komponensei:

1. **Topológia-vizualizáció**: A Mininet [3] alapú hálózati szimulációs topológiák grafikus megjelenítése, amely segít a hálózat szerkezetének megértésében és az adatfolyamok útvonalainak nyomon követésében.
2. **Felügyeleti adatok tárolása és feldolgozása**: A szoftver által generált JSON-formátumú UDP-üzenetek fogadása, feldolgozása és egy adatbázisban történő tárolása, ahol az adatok hosszú távon megőrzésre kerülnek.
3. **Vizualizáció**: A felügyeleti adatok megjelenítése interaktív felületeken, amelyek valós idejű és történeti metrikai információkat mutatnak be.
4. **Előrejelzés**: A kritikus hálózati események (például forgalmi anomáliák) előrejelzése, amely lehetővé teszi a proaktív hálózatkezelést.

1.3 A feladat indokoltsága

A FRER/PREOF mechanizmusok alkalmazásával a hálózati megbízhatóság jelentősen javítható, azonban a hálózati továbbítás komplexitása megnő. Az ilyen redundanciákon alapuló rendszerek hatékony üzemeltetéséhez szükséges a belső működési folyamatok részletes nyomon követése, melyre a felügyeleti adatok lehetőséget nyújtanak.

Gyakorlati relevancia: A kritikus alkalmazásokban (autonóm járművek, orvosi eszközök, ipari automatizáció) egy hálózati hiba vagy késedelem súlyos következményekkel járhat. A szoftver által biztosított felügyeleti adatok és azok vizualizálása lehetővé teszik:

- a hálózat valós idejű egészségének felügyeletét,
- a megbízhatósági mechanizmusok működésének ellenőrzését,
- a rendszer teljesítményének optimalizálását,
- a potenciális hálózati problémák korai felismerését.

Tudományos relevancia: A hálózati adatokra vonatkozó előrejelzési módszerek alkalmazása lehetővé teszi az anomáliamintázatok automatikus felismerését és a kritikus eseményekre való felkészülést még azelőtt, hogy azok végzetes hatást gyakorolnának a rendszerre.

1.4 A diplomaterv felépítése

A dolgozat a következő szerkezetben épül fel:

A **2. fejezet** az irodalomkutatás, amely részletesen bemutatja a FRER és PREOF mechanizmusok működési elvét, illetve áttekinti a meglévő hálózati megfigyelési és vizualizációs megoldásokat és az előrejelzési technikákat.

A **3. fejezet** a rendszer tervezését tárgyalja. Részletezi a Mininet alapú tesztkörnyezet felépítését és a hálózati szoftver által szolgáltatott felügyeleti adatokat. Bemutatja a technológiaválasztás folyamatát és indoklását az adatbázis, a megjelenítő és az előrejelző modell tekintetében, valamint definiálja a rendszer adatáramlási architektúráját és a szükséges megjelenítő felületeket (dashboard-okat).

A **4. fejezet** a megvalósítás lépéseit mutatja be. Ismerteti a fejlesztési környezet kialakítását és a rendszer egyedi fejlesztésű komponenseinek implementációs részleteit.

Az **5. fejezet** az elkészült rendszer elemzését tartalmazza. Képernyőképekkel illusztrálva bemutatja a megvalósított megjelenítést és azok funkcióit, majd részletesen értékeli az előrejelző modul pontosságát a különböző hálózati forgalmi szituációkban, stabil és dinamikus környezetben egyaránt.

A **6. fejezet** összegzi a diplomamunka eredményeit, értékeli a kitűzött célok teljesülését, valamint javaslatokat tesz a rendszer jövőbeli továbbfejlesztési lehetőségeire.

2 Irodalomkutatás

A diplomatervben tervezett rendszer megvalósítása előtt szükséges áttekinteni a témakörhöz kapcsolódó elméleti hátteret és technológiai megoldásokat.

Elsőként a FRER (Frame Replication and Elimination for Reliability) mechanizmus kerül bemutatásra, amely a TSN keretrendszer részeként Ethernet hálózatokon biztosít redundanciát keretmásolás és eliminálás révén. Ezt követően a PREOF (Packet Replication, Elimination and Ordering) technológia kerül tárgyalásra, amely a DetNet architektúra részeként hasonló megbízhatóságot nyújt, de IP-alapú környezetre optimalizálva. A fejezet összehasonlítja a két megoldás működését és alkalmazási területeit.

Az adatbázisok szakasz a hálózati telemetria tárolására alkalmas idősor-alapú (time-series) megoldásokat, valamint a hatékony lekérdezési stratégiákat vizsgálja. A megjelenítő felületek rész a dashboard-alapú vizualizációt és a topológia-ábrázolás módszereit elemzi.

Az előrejelzési technikák fejezet a statisztikai és gépi tanulási alapú idősor-előrejelzési módszereket hasonlítja össze a hálózati metrikák kontextusában. Az adatfeldolgozási lehetőségek szakasz a telemetria-adatok gyűjtésének architektúráit mutatja be. Végül a hálózati veszteségek modellezése témakör az adatvesztés és késleltetés szimulációs módszereibe nyújt betekintést, ami elengedhetetlen a rendszer teszteléséhez.

2.1 Frame Replication and Elimination for Reliability

A Frame Replication and Elimination for Reliability (FRER) [1] egy hibadetektálási és helyreállítási mechanizmus, amelyet az IEEE 802.1CB szabvány definiál a Time-Sensitive Networking (TSN) keretrendszer részeként. A mechanizmus célja a kritikus hálózati alkalmazásokban az adatátvitel megbízhatóságának növelése redundáns átviteli utak alkalmazásával.

2.1.1 A FRER működési elve

A FRER alapelve az adatkeretek többszörözésén és többutas átvitelén alapul. A hálózati csomópontok konfigurációja alapján az eredeti adatfolyamot több példányban sokszorosítja, és ezeket a másolatokat különböző, egymástól független útvonalakon

keresztül továbbítja a célcsomópont felé. Az útvonalak találkozási pontjain a hálózati csomópontok felismerik és konfiguráció függően kiszűrik a duplikált kereteket, így a végponthoz már csak egyetlen, helyreállított adatfolyam érkezik. A mechanizmus működése során minden kerethez egy egyedi sorszámot rendelnek, amely lehetővé teszi a duplikátumok azonosítását. A hálózati csomópontok a sorszámok alapján eldöntik, hogy egy beérkező keret új információt hordoz-e, vagy egy korábban már feldolgozott keret másolata. A FRER két alapvető műveletet definiál:

- **Keretmásolás (replication):** Az eredeti adatfolyam szétválasztása több tagfolyamra, amelyek különböző útvonalakon haladnak tovább.
- **Kereteliminálás (elimination):** A duplikált keretek felismerése és eldobása a sorszámok alapján az útvonalak egyesülési pontjain.

2.1.2 A megbízhatóság biztosítása

A FRER mechanizmus hatékonysága abban rejlik, hogy az adatátvitel megbízhatóságát a redundancia révén növeli anélkül, hogy a végponti alkalmazásoknak tudniuk kellene a hálózat belső működéséről. A redundancia védelmet nyújt az esetleges hálózati hibák ellen, mivel, ha egy útvonalon elveszik a keret, az a redundáns útvonalon továbbra is eljuthat a célhoz.

A szabvány nem határoz meg konkrét algoritmusokat arra vonatkozóan, hogy mely forgalmat kell sokszorosítani, hány másolatot kell készíteni, vagy hogyan kell reagálni az állandó hibákra. Ezek a döntések a hálózat konfigurációjától és az alkalmazás követelményeitől függenek.

2.1.3 Alkalmazási területek

A FRER mechanizmus különösen alkalmas olyan kritikus alkalmazások számára, ahol az adatvesztés vagy késleltetés súlyos következményekkel járhat:

- **Ipari automatizálás**: Gyártósorok, robotvezérlés és folyamatirányítás, ahol a valós idejű kommunikáció elengedhetetlen.
- **Járműipari alkalmazások:** Autonóm járművek belső kommunikációja, ahol a biztonsági rendszerek között megbízható és időben determinisztikus adatátvitelre van szükség.

- **Energiaipar:** Intelligens hálózatok vezérlőrendszerei, ahol a kritikus vezérlőüzenetek elvesztése áramkimaradáshoz vagy eszközkárosodáshoz vezethet.
- **Orvosi eszközök:** Életfenntartó vagy műtéti robotikai rendszerek, amelyek esetében a kommunikációs hiba emberi életet veszélyeztethet.

2.2 Packet Replication, Elimination and Ordering

A Packet Replication, Elimination and Ordering Functions (PREOF) [2] a Deterministic Networking (DetNet) architektúra szolgáltatásvédelmi mechanizmusa, amely hasonló redundancia-alapú megbízhatósági megközelítést alkalmaz, mint a FRER, azonban IP-alapú hálózatokra optimalizálva. A PREOF funkciókat az Internet Engineering Task Force (IETF) DetNet munkacsoport szabványosította a determinisztikus hálózatok számára.

2.2.1 A PREOF működési elve

A PREOF három alapvető funkciót integrál egyetlen mechanizmusba:

- **Packet Replication Function (PRF):** A csomag többszörözési funkció felelős az eredeti adatfolyam több példányban történő sokszorosításáért. A másolatok különböző, egymástól független útvonalakon kerülnek továbbításra a célcsomópont felé.
- **Packet Elimination Function (PEF):** A csomageliminálási funkció a duplikált csomagok felismerését és eldobását végzi a sorszám-információk alapján. Ez biztosítja, hogy a célcsomóponthoz csak egy példány jusson el minden egyes csomagból.
- **Packet Ordering Function (POF):** A csomag-sorrendező funkció a sorszámok alapján helyreállítja az eredeti adatfolyam sorrendjét, még akkor is, ha a különböző útvonalakon haladó csomagok eltérő késleltetéssel érkeznek meg.

A PREOF működésének alapja a sorszám-információk hozzáadása az adatcsomagokhoz. Ezek a sorszámok lehetővé teszik, hogy a hálózat végpontjai és köztes csomópontjai azonosítsák a duplikált csomagokat és helyreállítsák az eredeti sorrendet. A mechanizmus egy összetett adatfolyamot hoz létre, amelynek több tagfolyama van, és ezek a tagfolyamok független útvonalakon haladnak.

2.2.2 A PREOF alkalmazási környezete

A PREOF elsősorban olyan kritikus alkalmazások számára készült, ahol az IP-alapú hálózati infrastruktúra determinisztikus viselkedése elengedhetetlen:

- **Ipari IoT (Internet of Things):** Elosztott szenzor rendszerek, ahol a csomópontok között IP-alapú kommunikáció történik.
- **Telekommunikációs infrastruktúra:** Mobil hálózatok szegmensei és törzs hálózatok, ahol a szolgáltatásminőség kritikus.
- **Kritikus infrastruktúrák:** Távoli vezérlőrendszerek, amelyek WAN hálózatokon keresztül kommunikálnak, például energetikai vagy vízkezelési rendszerek.
- **Autonóm járművek közötti kommunikáció:** Vehicle-to-Vehicle (V2V) és Vehicle-to-Infrastructure (V2I) alkalmazások, ahol az IP-alapú kommunikáció dominál.

A PREOF lehetővé teszi, hogy a szolgáltatásvédelem end-to-end módon valósuljon meg, nem csak alhálózatokon belül, hanem a teljes domain-en keresztül. Ez különösen fontos olyan forgatókönyvekben, ahol a hálózat heterogén technológiákból áll és az IP réteg biztosítja az egységes kommunikációt.

2.2.3 A FRER és a PREOF összehasonlítása

Bár mindkét mechanizmus hasonló redundancia-alapú megbízhatósági elveken alapul, fontos különbségek vannak közöttük:

- **Hálózati réteg és protokoll:** A FRER az IEEE 802.1CB szabvány részeként Layer 2 (adatkapcsolati réteg) szinten működik Ethernet keretekkel, míg a PREOF az IETF DetNet keretében Layer 3 (hálózati réteg) szinten működik IP csomagokkal.
- **Alkalmazási környezet:** A FRER a Time-Sensitive Networking (TSN) keretrendszer része, amely elsősorban helyi és nagyvárosi hálózatokra (LAN/MAN) fókuszál. A PREOF a Deterministic Networking (DetNet) architektúra komponense, amely szélesebb körű hálózati környezetekre, beleértve WAN hálózatokat is, alkalmazható.

2.3 Előrejelzési technikák

Az előrejelzési technikák célja a múltbeli adatok alapján a jövőbeli értékek becslése. Hálózati metrikák esetében az előrejelzés különösen fontos szerepet tölt be a proaktív hálózatkezelésben és a potenciális hibák korai észlelésében. A FRER és PREOF

mechanizmusok esetében az előrejelzés lehetővé teszi a hálózati veszteségek, késleltetések és egyéb teljesítménymutatók jövőbeli alakulásának előrevetítését, ami hozzájárul a redundancia-igény optimális meghatározásához és a szolgáltatásminőség fenntartásához, illetve javításához. Az irodalomkutatás során két fő csoportba tartozó módszereket vizsgáltam, a klasszikus idősor-alapú előrejelzési technikákat és a gépi tanulást alkalmazó megközelítéseket. Mindkét kategória sajátos előnyökkel és hátrányokkal rendelkezik a hálózati adatok kontextusában.

2.3.1 Klasszikus idősor-alapú előrejelzés

A klasszikus idősor-alapú modellek az idősor egymást követő megfigyelési pontjai közötti statisztikai összefüggéseken alapulnak. Ezek a modellek lineáris összefüggéseket feltételeznek, és stacionárius idősorokon működnek optimálisan, ahol az átlag és a variancia időben állandó marad. Az Autoregressive (AR) modell a jövőbeli értékeket az idősor korábbi értékeinek lineáris kombinációjaként fejezi ki. A Moving Average (MA) modell a múltbeli előrejelzési hibák alapján modellezi a jövőbeli értékeket. Az Autoregressive Integrated Moving Average (ARIMA) [4] modell az AR és MA kombinációja, amely nemstacionárius idősorok kezelésére is képes, ám érdemes stacionáriussá konvertálni az adathalmazt, mivel így pontosabb eredményt mutat. Az ARIMA három paramétert használ:

- **p (AR rend):** A figyelembe vett késleltetett megfigyelések súlya.
- **d (differenciálási fok):** A stacionaritás eléréséhez szükséges differenciálási lépések száma, amely eliminálja a trendeket.
- **q (MA rend):** A modellbe bevont múltbeli reziduális hibák súlya.

A paraméterek meghatározása hiperparaméter-optimalizálással történik, amely során különböző p, d és q értékkombinációkat tesztelünk, és a modell teljesítménye validációs metrikák alapján értékelhető. Az ARIMA modellek széles körben alkalmazottak pénzügyi, közgazdasági és környezettudományi előrejelzésekben. Hálózati telemetria esetében eredményesen alkalmazhatóak forgalmi minták, késleltetések és csomagvesztési arányok előrejelzésére. A klasszikus idősor-modellek előnyei az alacsony számítási igény, az interpretálhatóság és az elméleti megalapozottság. Hátrányuk, hogy egyváltozós megközelítésük nem használja ki a metrikák közötti kölcsönhatásokat, és nehezen kezelik a nemlineáris összefüggéseket.

2.3.2 Gépi tanulást alkalmazó előrejelzés

A gépi tanulás-alapú előrejelzési módszerek napjainkban egyre népszerűbbek az idősor-elemzés területén. Ezek a technikák képesek olyan összetett, nemlineáris összefüggéseket is felismerni az adatokban, amelyeket a statisztikai modellek nem tudnak megragadni. A gépi tanulási megközelítések különösen alkalmasak többváltozós idősorok elemzésére, ahol több metrika együttes alakulását kell figyelembe venni. A Long Short-Term Memory (LSTM) [5] neuronhálózatok az idősor-előrejelzés egyik legígéretesebb gépi tanulási technikáját képviselik. Az LSTM egy speciális visszacsatolt neuronhálózat (RNN) architektúra, amelyet kifejezetten hosszú távú függőségek tanulására terveztek. A hagyományos RNN-ekkel ellentétben az LSTM rendelkezik egy belső memóriastruktúrával, amely lehetővé teszi a távoli időpontok közötti összefüggések megőrzését. Az LSTM hálózatok három fő komponense, a bemenet (input gate), az elfelejtési (forget gate) és a kimenet (output gate) kapu – együttműködve határozzák meg, hogy mely információkat őrizzük meg, melyeket felejtsünk el, és melyeket használjunk fel a kimeneti előrejelzéshez. Ez a mechanizmus különösen ígéretes hálózati metrikák esetében, ahol a múltbeli forgalmi minták vagy rendellenességek hosszú távú hatással lehetnek a jövőbeli viselkedésre. A gépi tanulás-alapú előrejelzés előnyei közé tartozik a nemlineáris összefüggések kezelésének képessége, a többváltozós elemzés természetes támogatása és a komplex minták felismerése. Ugyanakkor ezek a módszerek sokkal több számítási erőforrást igényelnek mind a tanítási, mind az előrejelzési fázisban. További kihívást jelent a megfelelő hiperparaméterek meghatározása, a túltanulás elkerülése, valamint az, hogy a neurális hálózatok „fekete doboz” jellege megnehezíti az eredmények interpretálását. Hálózati telemetria-adatok esetében a gépi tanulás-alapú megközelítések különösen akkor válhatnak szükségessé, ha az adatokban ciklikus minták, szezonalitás vagy komplex anomáliák jelennek meg, amelyeket a klasszikus modellek nem tudnak kellő pontossággal leírni. A klasszikus és a gépi tanulási módszerek közötti választás függ az alkalmazási kontextustól, a rendelkezésre álló adatok mennyiségétől és minőségétől, valamint a számítási kapacitás korlátaitól.

2.4 Adatbázisok

Az adatbázisok kiválasztásának kutatását a tárolandó adatok természetének elemzésével érdemes kezdeni. A FRER és PREOF mechanizmusok felügyeleti adatai alapvetően időfüggő metrikák, amelyek folyamatosan generálódnak a hálózati infrastruktúrában.

Ezeknek az adatoknak két alapvető jellemzője van. Minden mérés rendelkezik egy időbélyeggel, amely meghatározza a mérés időpontját, valamint szükség van a különböző hálózati csomópontok (host-ok) adatainak egyértelmű megkülönböztetésére. Ez a két követelmény, az időbélyeg-központúság és a forrásazonosítás, egyértelműen az idősor alapú (time-series) adatbázisok, mint legkézenfekvőbb megoldás irányába mutat.

2.4.1 Idősor alapú adatbázisok

Az idősor alapú adatbázis (TSDB) olyan rendszer, amelyet kifejezetten idősorok tárolására és lekérdezésére optimalizáltak. Az idősorok időpont-érték párok sorozatából állnak, ahol minden adat egy konkrét időponthoz (timestamp) és egy vagy több mérési értékhez kapcsolódik. A TSDB-k alapvető célja, hogy hatékonyan kezeljék az időben folyamatosan beérkező adatokat, és gyors lekérdezési lehetőséget biztosítsanak időalapú szűrések és aggregációk számára. Az idősor alapú adatbázisok több szempontból is különböznek a hagyományos relációs adatbázisoktól:

- **Írás-optimalizálás:** A TSDB-k jellemzően írás-intenzív környezetekre vannak tervezve, ahol folyamatosan nagy mennyiségű adat érkezik. Az adatok általában csak hozzáadásra kerülnek (append-only), a múltbeli adatok módosítása ritkán vagy egyáltalán nem történik meg.
- **Adatmegőrzési szabályok (retention policies):** A TSDB-k beépített mechanizmusokat kínálnak az adatok életciklusának kezelésére. Meghatározható, hogy mennyi ideig őrizzük meg a nyers adatokat, és hogyan aggregáljuk azokat hosszabb távú tárolásra (downsampling). Például a percenkénti mérések hetente aggregálhatók óránkénti átlagokká.
- **Címkézési rendszer:** A TSDB-k címkék (label) segítségével kategorizálják az adatokat, ami lehetővé teszi a rugalmas szűrést és aggregációt. Például egy hálózati metrika címkézése tartalmazhatja a host nevét, a szolgáltatás típusát vagy a hálózati interface-t (csatoló).
- **Időalapú lekérdezések optimalizálása:** A TSDB-k indexelési struktúrái az időbélyegekre vannak optimalizálva, így az időintervallum-alapú lekérdezések (például „az elmúlt 24 óra adatai”) rendkívül gyorsak.

Hálózati telemetria esetében az idősor alapú adatbázisok ideális megoldást nyújtanak, mivel a hálózati metrikák, mint a csomagvesztési arány, késleltetés, vagy a FRER/PREOF

duplikáció-statisztikák folyamatosan generálódnak, és az elemzések során elsősorban időbeli összefüggésekre vagyunk kíváncsiak.

2.4.2 Adatmodellezési paradigmák

Az idősor-alapú adatbázisok működésének egyik kulcsfontosságú eleme az adatgyűjtés módja és időzítése, amely közvetlenül befolyásolja az adatok gazdagságát és a rendszer teljesítményét. Ezek az adatbázisok két alapvető adatgyűjtési modellt támogatnak. A **„push”** modell esetében az adatforrások (például hálózati csomópontok) aktívan küldik az adatokat az adatbázisnak. Ez a megközelítés akkor előnyös, ha az adatforrások rövid életciklusúak, dinamikusan jönnek létre és szűnnek meg, vagy ha azonnali adatküldésre van szükség eseményvezérelt folyamatokban. A push modell hátránya, hogy az adatbázisnak nincs közvetlen kontrollja az adatgyűjtés ütemezése felett, és az adatforrásoknak maguknak kell kezelniük az adatbázis elérhetőségét és pl. a hálózati hibák esetén az újraküldési logikát. A **„pull”** (vagy scrape) modell esetében az adatbázis vagy egy központi gyűjtő komponens rendszeres időközönként lekéri az adatokat a forrásokból. Ez a megközelítés központi kontrollt biztosít az adatgyűjtés ütemezése felett, és lehetővé teszi, hogy az adatbázis maga figyelje az adatforrások elérhetőségét. A pull modell általában egyszerűbb konfigurációt tesz lehetővé, mivel az adatforrásoknak csak egy egyszerű API-t vagy metrika-exportot kell biztosítaniuk, anélkül, hogy maguknak kellene kezelniük az adatküldést.

A push vagy pull intervallum, azaz az az időköz, amelyen belül az adatgyűjtés történik, kritikus hatással van az adatok gazdagságára és így a későbbiekben az előrejelzési modellek pontosságára is. Rövidebb intervallumok (például 5-10 másodperc) részletesebb képet adnak a hálózati viselkedésről, lehetővé téve a gyors változások és anomáliák pontosabb észlelését. Azonban ez nagyobb terhelést jelent mind az adatforrásokra, mind az adatbázisra, és jelentősen megnöveli a tárolt adatok mennyiségét. Hosszabb intervallumok (például 30-60 másodperc vagy több perc) csökkentik az adatmennyiséget és a rendszer terhelését, de a részletesség elvesztése azt eredményezheti, hogy rövid ideig tartó események vagy kiugrások nem kerülnek rögzítésre. Ez különösen problémás lehet hálózati telemetria esetében, ahol a csomagvesztési csúcsok vagy késleltetési anomáliák akár másodpercek alatt is bekövetkezhetnek és elmúlhatnak. Az intervallumok meghatározása az alkalmazás követelményeitől függ. Kritikus hálózati infrastruktúrák esetében, ahol a FRER/PREOF mechanizmusok meghibásodása súlyos

következményekkel járhat, rövidebb intervallumok ajánlottak a valós idejű felügyelet érdekében. Ezzel szemben, hosszú távú kapacitástervezés esetén elegendő lehet hosszabb intervallum, különösen, ha az adatokat később aggregáljuk.

Az idősor alapú adatbázisok rugalmassága lehetővé teszi, hogy az adatgyűjtési és -megőrzési stratégiát az alkalmazás specifikus igényeihez igazítsuk, biztosítva a megfelelő egyensúlyt az adatgazdagság, a tárolási költségek és a rendszer teljesítménye között.

2.5 Megjelenítő felületek

A megjelenítő felület kiválasztása kritikus a hálózati felügyeleti rendszerek tervezésénél, mivel az adatok értelmezhetősége és a gyors döntéshozatal közvetlenül függ a vizualizáció minőségétől. A hálózati adatok idősor-alapú metrikák, amelyek időbeli alakulását leghatékonyabban grafikus eszközökkel lehet bemutatni.

2.5.1 Követelmények a megjelenítő felületekkel szemben

A megjelenítő felületeknek hatékonyan kell kezelniük az idősorok grafikus ábrázolását, mivel a hálózati metrikák folyamatos időbeli változásának megjelenítése a fő feladatuk. Különösen fontos a rugalmas adatforrás-integráció, mivel a rendszernek képesnek kell lennie különböző idősor-adatbázisokhoz való kapcsolódásra és a metrikákat kombinálni egyetlen nézetben. A kritikus hálózati infrastruktúrák felügyeletéhez elengedhetetlen a valós idejű frissítés, amely lehetővé teszi a rendszergazdák gyors reagálását. Az interaktív elemzési funkciók, mint az időintervallumok kiválasztása vagy a metrikák összehasonlítása, segítik a felhasználókat a problémás időszakok gyors azonosításában. A testreszabhatóság biztosítja, hogy különböző felhasználói szerepkörök és felügyeleti helyzetek eltérő vizualizációs igényeit kielégítsük.

2.5.2 Dashboard-alapú megjelenítési megoldások

A hálózati felügyeleti rendszerek jellemzően dashboard-alapú architektúrát követnek, ahol egy központi webes felületen több panel jeleníti meg a különböző metrikákat. A dashboard-ok tervezésénél az információs hierarchia kialakítása kritikus, ahol a legfontosabb metrikák előtérbe kerülnek, míg a részletesebb diagnosztikai adatok külön nézetekben jelennek meg. A többszintű szűrési lehetőségek révén a felhasználók specifikus hálózati csomópontokra, időintervallumokra vagy metrika-kategóriákra fókuszálhatnak.

2.5.3 Topológia vizualizáció

A redundáns útvonalak esetében különösen fontos a hálózati topológia vizualizációja, mivel így egyszerűbben meg lehet érteni a rendszer működését és áttekinteni az adatfolyamok útvonalait. A topológia vizualizáció gráfokra épül, ahol a csomópontok a hálózati eszközöket, az élek pedig a kapcsolatokat reprezentálják.

A megjelenítő felületek tervezése során a kulcs az egyszerűség és a funkcionalitás közötti egyensúly megtalálása. A jól megtervezett vizualizációs rendszer segíti a hálózati rendszergazdákat a hálózatkezelésben és a gyors probléma azonosításban.

2.6 Adatfeldolgozási lehetőségek

Az adatfeldolgozás architektúrájának kialakítása során három fő kihívást kell megoldani: a beérkező metrikák adatbázisba való eljuttatását, az adatbázisban tárolt adatok előrejelzési modellekhez való továbbítását, illetve eredmények tárolását.

2.6.1 Metrikák kapcsolata az adatbázissal

A metrikák adatbázisba való beérkezésekor a megfelelő címkézés (labeling) alkalmazása kritikus fontosságú. A címkék azok az elemek, amelyek kontextusba helyezik az információkat, például a forrás csomópont azonosítóját vagy a metrika típusát. A címkék kialakítása az adatmodellezés egyik legfontosabb lépése, mivel a későbbi lekérdezések hatékonysága nagymértékben függ a címkék megfelelő rendszerezettségétől.

2.6.2 Adatok kapcsolata az előrejelzési rendszerrel

Az előrejelzési modellek integrálása több lépésből áll. Először a releváns történeti adatokat kell lekérdezni az adatbázisból, meghatározva az időablakot és a szűrési feltételeket. Az adatok ezután előfeldolgozási lépéseken mennek keresztül, amely magában foglalja a hiányzó értékek kezelését és az adatok normalizálását. Az előkészített adatokat az előrejelzési modell feldolgozza és generálja a jövőbeli értékekre vonatkozó becsléseket. A generált előrejelzéseket megkülönböztető címkézéssel visszaírjuk az adatbázisba, így későbbi elemzésre és vizualizációra is rendelkezésre állnak. Ez lehetővé teszi a múltbeli előrejelzések pontosságának utólagos értékelését, a rendszer finomhangolását.

2.7 Hálózati veszteségek modellezése

A hálózati veszteségek modellezése kritikus fontosságú a FRER és PREOF mechanizmusok hatékonyságának értékeléséhez. Virtualizált környezetekben a valós hálózati hibák nem jelennek meg természetesen, ezért explicit modellezésre van szükség. A két legfontosabb jellemző a késleltetés (delay) és a csomagvesztés (packet loss).

2.7.1 Késleltetés és csomagvesztés modellezése

A késleltetés és csomagvesztés modellezésekor figyelembe kell venni a hálózat véletlenszerű viselkedését. A teljes késleltetés több tényezőből áll össze, a jel terjedési ideje, a feldolgozási idő és a várakozási idő. Ezek közül a várakozási idő a leginkább változó, különösen nagy forgalom esetén. Vezeték nélküli hálózatoknál a késleltetés ingadozása (jitter) még jelentősebb lehet a változó kapcsolatminőség és az újraküldések miatt. A csomagvesztés modellezhető statisztikai módszerekkel, ahol valószínűségi eloszlás alapján határozzuk meg egy csomag elvesztését.

2.7.2 Hálózati emuláció eszközei

Linux rendszereken a tc (traffic control), NetEm [6] eszközök lehetővé teszik a link paraméterek dinamikus módosítását. A NetEm segítségével késleltetést, jitter-t és csomagvesztést lehet beállítani, valamint ezeket időben változtatni. A hálózati veszteségek pontos modellezése lehetővé teszi a FRER és PREOF mechanizmusok alapos értékelését szimulált környezetekben, hozzájárulva a rendszerek optimalizálásához.

3 Tervezés

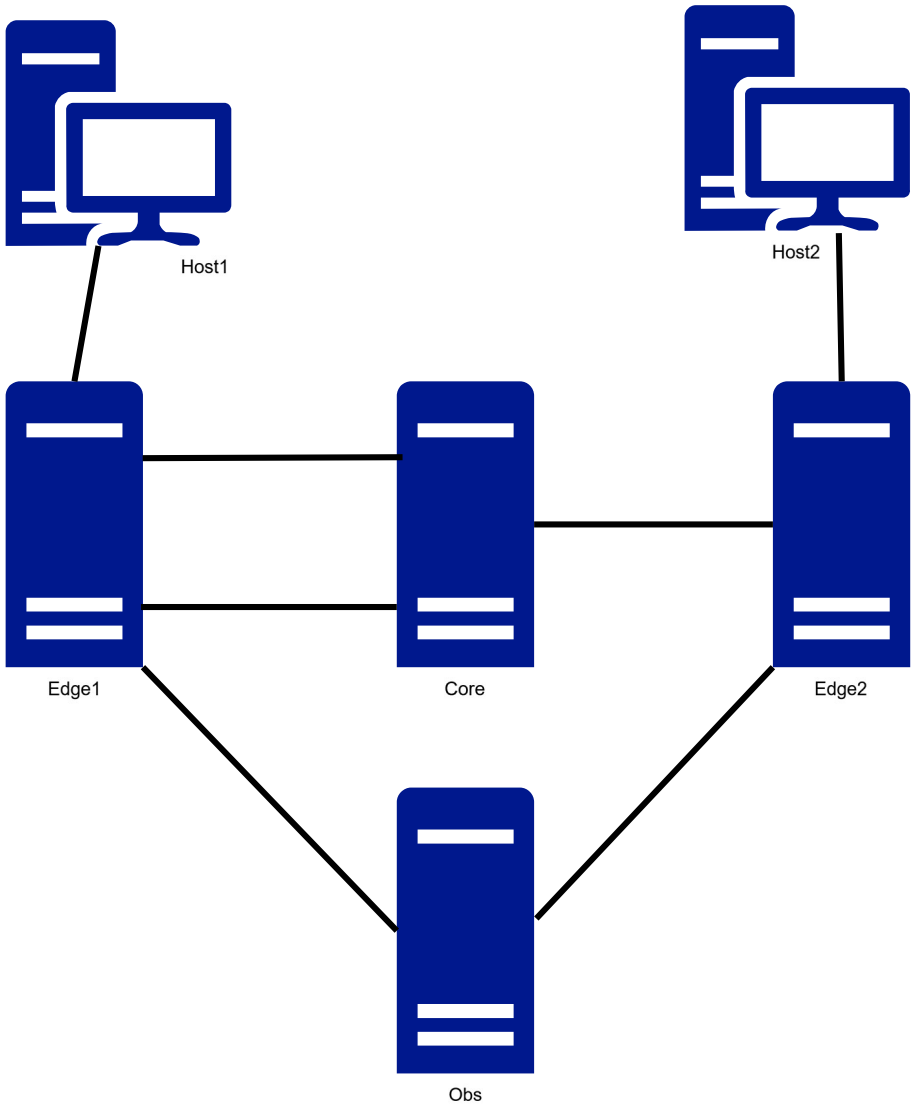
Az irodalomkutatás során áttekintett elméleti háttér és technológiai megoldások alapján ebben a fejezetben bemutatom a felügyeleti és előrejelző rendszerének tervezését. A cél egy olyan rendszer létrehozása, amely képes a redundancia-alapú mechanizmusok működési adatainak gyűjtésére, tárolására, vizualizációjára és előrejelzésére. A tervezett rendszer mikroszolgáltatás-alapú architektúrát követ, amely lehetővé teszi az egyes komponensek független fejlesztését és skálázását. A tervezési folyamat a hálózati teszt környezet kialakításával kezdődik, amely biztosítja a FRER/PREOF mechanizmusok szimulációját. Az adatgyűjtő komponens a hálózati csomópontokról származó metrikák összegyűjtéséért felelős, amelyek egy idősor-alapú adatbázisban kerülnek tárolásra. Az adatáramlási architektúra meghatározza, hogyan jutnak el az adatok a forrásoktól a feldolgozási és megjelenítési komponensekhez. Az előrejelzési modul a történeti adatok alapján generál előrejelzéseket a hálózati metrikák jövőbeli alakulására. A megjelenítő felület dashboard-okon keresztül biztosítja a metrikák és előrejelzések valós idejű vizualizációját. A következő alfejezetek részletesen bemutatják az egyes komponensek tervezését és az alkalmazott technológiai megoldásokat.

3.1 Teszt környezet

A rendszer tesztelésének és fejlesztésének alapját egy Mininet-alapú hálózati emuláció adja, amely lehetővé teszi a többutas továbbítási mechanizmusok működésének szimulálását virtualizált környezetben. A teszt környezet kiindulópontja egy előre elkészített Mininet topológia, amely tartalmazza a hálózati csomópontokat, linkeket és a redundancia-alapú adatfolyamok szimulációjához szükséges konfigurációkat.

3.1.1 Mininet topológia

A Mininet [3] egy hálózat emulátor, amely Linux network namespace-ek és virtual Ethernet párok segítségével hoz létre virtuális hálózati topológiákat egyetlen fizikai vagy virtuális gépen. A kiindulási topológia a következő ábrán látható:



3.1. ábra: Mininet topológia

A topológia (3.1. ábra) hat fő hálózati csomópontból áll:

- Host1: A forráscsomópont, amely forgalmat generál a Host2 felé.
- Edge1 (PRF): A tanszéki szoftvert futtató csomópont, amely packet replication funkciót lát el. Két redundáns linken keresztül továbbítja a csomagokat a Core felé.
- Core: Központi csomópont, amely összeköti a redundáns útvonalakat.
- Edge2 (PEF): A tanszéki szoftvert futtató csomópont, amely packet elimination funkciót lát el, eliminálva a duplikált csomagokat.
- Host2: A célcsomópont, amely fogadja az adatfolyamot.
- Obs (observability host): A felügyeleti csomópont, amely az exporter alkalmazást futtatja.

3.1.2 Futtatási környezet Windows platformon

A Mininet hálózatokat legegyszerűbben Linux rendszereken lehet futtatni, Windows platformon Windows Subsystem for Linux (WSL) [7] segítségével használható. A WSL2 egy teljes Linux kernel-t biztosít Windows alatt, amely lehetővé teszi a Mininet és a hozzá tartozó hálózati névterek működését. A WSL környezetben futó Mininet topológia alapértelmezetten elkülönített a Windows gazdarendszertől. Ahhoz, hogy a topológiában létrehozott hálózati csomópontok elérhetők legyenek a Windows rendszerből és a rajta futó alkalmazásokból, hálózati alagutat (tunnel) kell létrehozni a WSL és a Windows között. Ez az alagút biztosítja, hogy a Mininet topológia gyökér névterében található interface-ek elérhetők legyenek a többi alkalmazás számára.

3.1.3 Konténerizált alkalmazások

A rendszer többi komponense – beleértve az adatbázist, az előrejelzési modult, a megjelenítő felületet és a topológia ábrázolót – Docker konténerekben [8] fut. A Docker Desktop biztosítja a konténerizált alkalmazások futtatási környezetét, amely lehetővé teszi a mikroszolgáltatások egyszerű kezelését, skálázását és hálózati összekapcsolását. A Mininet topológia és a Docker konténerek közötti kommunikáció a következőképpen valósul meg:

- Hálózati alagút (tunnel) konfigurálása: A WSL és a Windows host között hálózati alagút kerül beállításra, amely lehetővé teszi a Mininet csomópontok elérését a host rendszer felől.
- Docker hálózati konfiguráció: A Docker konténerek egy közös hálózatban futnak, amely csatlakozik a Mininet topológiához vezető tunnel-hez.

Ez az architektúra biztosítja, hogy a teszt környezet rugalmasan konfigurálható és különböző hálózati szimulációs forgatókönyvek tesztelésére alkalmas legyen, miközben a fejlesztési és futtatási környezet platformfüggetlen marad a konténerizáció révén.

3.2 Program által nyújtott felügyeleti adatok

A tanszéki szoftver periodikusan generál értesítő (notification) üzeneteket, amelyek részletes információkat tartalmaznak a csomópont aktuális állapotáról és a FRER/PREOF mechanizmusok működéséről. Ezek az üzenetek JSON formátumban kerülnek továbbításra UDP protokollon keresztül a feldolgozó rendszer felé. Emellett a főbb

eseményekről a csomópontok küldenek „push” üzeneteket is, ezek a küldése az esemény hatására történik.

3.2.1 Notification üzenet struktúra

Minden üzenet négy fő komponensből áll:

- notif_seq: az értesítés sorszáma, amely lehetővé teszi az üzenetvesztés észlelését
- notif_hostname: a küldő csomópont egyedi azonosítója (pl. „edge1”, „edge2”)
- notif_tstamp: Unix timestamp formátumú időbélyeg, amely az üzenet generálásának pontos időpontját jelzi
- notif_msg: periodikus üzenet esetében tartalmazza a csomópont összes figyelt objektumának állapotát, amelyek különböző típusokba sorolhatók. Push üzenet esetében pedig magát az eseményt, ami kiváltotta (pl. telnet belépés)

A következő fejezetek a különböző periodikus noitification tartalmakat mutatja be.

3.2.2 Redundancia-kezelési objektumok

Packet Replication Function (PRF) adatok a csomagmásolási funkció állapotát írják le:

- Name: a PRF példány neve
- Octets_passed: az áthaladó oktettek összesített száma
- Packets_passed: az áthaladó csomagok száma
- Pipelines: az adat útvonalak listája, amely mindegyikhez tartalmazza:
    - Name: az útvonal azonosítóját
    - Action_count: az útvonalon található feldolgozási lépések számát
    - Mask_state: az útvonal állapotát írja le, „unmasked” = aktív, „masked” = inaktív, de a menedzsment csomagok mindenképp áthaladnak
- Type: típus jelölő jelen esetben „Replicate”, ezzel jelzi, hogy ez egy PRF objektum

Packet Elimination Function (PEF) adatok a duplikált csomagok eliminálásáért felelős funkció állapotát tartalmazzák:

- Name: a PEF példány neve
- Discarded_packets: az eldobott csomagok száma

- Passed_packets: a továbbított csomagok száma
- Reset_msec: az időtúllépési küszöb milliszekundumban (pl. 2000 ms)
- Latent_errors: az olyan esetek száma, amikor egy csomag várt időn belül nem érkezett meg
- Latent_error_paths: a problémás útvonalak száma
- Latent_error_resets: a hibás állapot hatására történő visszaállások száma
- Recovery_seq_num: az aktuális helyreállítási sorszám
- Seq_recovery_resets: a szekvencia-helyreállítási algoritmus újraindításainak száma
- History_length: a szekvenciaszámok tárolására fenntartott előzmények mérete
- Recovery_algorithm: a használt helyreállítási algoritmus típusa
- Use_init_flag: konfigurációs beállítás, hogy jelezzen-e, ha egy útvonal inicializálódik (Igaz/Hamis)
- Use_reset_flag: konfigurációs beállítás, hogy jelezzen-e, ha egy útvonal visszaállt (Igaz/Hamis)
- Type: típus jelölő, jelen esetben „Seqrec” , ezzel jelzi, hogy ez egy PEF objektum

3.2.3 Hálózati interface statisztikák

A hálózati interface-ek forgalmi adatai, ide tartoznak a virtuális és fizikai interface-ek is:

- recv_octets: fogadott oktettek száma
- recv_packets: fogadott csomagok száma
- send_octets: küldött oktettek száma
- send_packets: küldött csomagok száma

3.2.4 Parser statisztikák

A parser (elemző) objektumok a bejövő csomagok stream (folyam) azonosításának eredményét tartalmazzák:

- No_match octets: olyan oktettek száma, amelyek nem tartoznak egyetlen definiált stream-hez sem

- No_match packets: olyan csomagok száma, amelyeket nem sikerült stream-hez rendelni
- Stream_* octets: az egyes stream-ekhez rendelt oktettek száma, ahol a * a stream nevét jelöli
- Stream_* packets: az egyes stream-ekhez rendelt csomagok száma, ahol a * a stream nevét jelöli

3.2.5 Szekvenciagenerátor objektum

Ezek az objektumok a szekvenciaszám-generátor állapotát írják le:

- Name: a generátor azonosítója
- Type: típus jelölő, jelen esetben „Seqgen”
- Use_init_flag: konfigurációs beállítás, hogy jelezzen-e, ha egy útvonal inicializálódik (Igaz/Hamis)
- Use_reset_flag: konfigurációs beállítás, hogy jelezzen-e, ha egy útvonal visszaállt (Igaz/Hamis)

3.2.6 MIP pontok

MIP (Maintenance Intermediate Point) – azaz belső karbantartási pontok, melyek célja a különböző belső működések és a stream-ek feldolgozása közbeni belső objektumok állapotának megfigyelése.

A MIP mezői:

- Level: a MIP hierarchikus szintje
- Name: a MIP neve
- Recv: a fogadott csomagok száma
- Send: a küldött csomagok száma
- Stream_name: a stream neve, amihez tartozik a MIP
- Type: típus jelölő, jelen esetben „MIP”, ezzel jelzi, hogy ez egy karbantartási pont objektum

- Object: a megfigyelt objektum, ami lehet PRF vagy PEF, itt ugyanazok a mezők jelennek meg, mint az ezekhez tartozó objektumoknál

Ezek a felügyeleti adatok részletes képet adnak a FRER/PREOF mechanizmusok működéséről, a redundáns útvonalak terheléséről, a duplikált csomagok kezeléséről és a hálózati interface-ek forgalmáról, amelyek alapján a rendszer valós idejű betekintést adhat a hálózat állapotába.

3.3 UDP fogadó és metrika export

A rendszer központi eleme egy Python nyelven implementált UDP fogadó és metrika exportáló modul, amely a hálózati csomópontokból érkező értesítéseket összegyűjti, feldolgozza, majd az idősor-alapú adatbázis számára exportálja. A modul két fő komponensből épül fel: az UDP Receiver a beérkező UDP üzenetek fogadásáért és rekonstruálásáért felel, míg a Metrics Exporter az adatok feldolgozását és az exportált metrikák rendezését végzi.

3.3.1 UDP Receiver

A fejlesztés kezdeti állapotában a komponens feladata mindössze az volt, hogy a megadott UDP port-on fogadja az érkező üzeneteket, majd a JSON formátumú „notification” objektumokat visszaállítva megjelenítse a konzolon.

Ezt tovább kell fejleszteni a következő funkciókkal:

- Párhuzamos feldolgozás: a fogadó résznek és a feldolgozó egységnek párhuzamosan kell működniük egymás mellett,
- Naplózási funkció: minden beérkező és feldolgozott üzenetet naplózni kell, ezzel biztosítva a visszakövethetőséget,
- Adatátadási interfész: a sikeresen feldolgozott üzeneteket át kell adni feldolgozásra az exporter-nek.

Ezzel a módosítással a komponens már nem csupán a nyers adatok megjelenítésére, hanem megbízható és folyamatos adatátvitelre is alkalmas.

3.3.2 Metrika exporter

A metrika exporter feladata, hogy a beérkező, JSON formátumú „notification” üzenetekből kinyert információkat rendezett, azonosítható formában továbbítsa az idősor-

alapú adatbázis felé. A feldolgozás során a rendszer minden objektumhoz címkéket rendel, amelyek lehetővé teszik a metrikák pontos azonosítását és szűrését.

Az alábbiakban (3.3.2.1) az egyes komponensekhez tartozó címkék kerülnek bemutatásra:

Komponens Neve	Címkék
PEF	hostname, component_name, component_type, object_name, object_type
PRF	hostname, component_name, component_type, pipeline_name, state
Sequence Generator	hostname, component_name, component_type
MIP	hostname, component_name, component_type, object_name, stream_name
Interface	hostname, interface_name
StreamParser	hostname, stream_name

3.3.2.1 Táblázat: Komponens Címkék

3.4 Adatbázis

A rendszernek olyan idősor-adatbázisra van szükség, amely képes hatékonyan kezelni a nagy mennyiségű hálózati metrikát. A kiválasztás során három elterjedt megoldást vizsgáltam: Prometheus, InfluxDB és VictoriaMetrics. A választás fő szempontjai a következők voltak: az adatok lekérdezésének egyszerűsége, a „pull” alapú adatáramlás – az adatgyűjtés ütemezésének kontrollja érdekében –, valamint a rendszer stabilitása, érettsége és a rendelkezésre álló dokumentáció minősége.

3.4.1 Prometheus

A Prometheus [9] egy nyílt forráskódú adatbázis és riasztási rendszer, amelyet eredetileg a SoundCloud fejlesztett ki 2012-ben, majd 2016-ban a Cloud Native Computing Foundation (CNCF [10]) második projektje lett a Kubernetes [11] után. Mára az idősor-adatbázisok egyik legszélesebb körben használt megoldásává vált.

3.4.1.1 Architektúra és adatmodell

A Prometheus egy pull-alapú architektúrát követ, ahol a központi szerver rendszeres időközönként lekérdezi a felügyelt célpontoktól a metrikákat HTTP végpontokon keresztül. Az adatmodell több dimenzióra épül, minden metrika egy egyedi név és opcionálisan kulcs-érték párok kombinációjával azonosítható.

3.4.1.2 PromQL lekérdező nyelv

A Prometheus Query Language (PromQL) egy funkcionális lekérdező nyelv, amely kifejezetten idősor-adatok elemzésére lett tervezve. Támogatja az alapvető aritmetikai műveleteket, aggregációs függvényeket (sum, avg, max, min), időablakos lekérdezéseket (rate, increase) és összetett matematikai műveleteket. Például egy interfész másodpercenkénti átviteli sebességének számítása 5 perces ablakkal: rate(network_bytes_total[5m]). A PromQL viszonylag egyszerű szintaxissal rendelkezik, de mégis kifejező, ami gyors tanulást tesz lehetővé.

3.4.1.3 Közösségi támogatás és dokumentáció

A Prometheus kiemelkedő minőségű dokumentációval rendelkezik, amely részletes útmutatókat és példákat tartalmaz. A CNCF projekt státusza biztosítja a hosszú távú támogatást és a folyamatos fejlesztést. Jelenleg ez a legelterjedtebb idősor alapú adatbázis, ezért nagy mennyiségű információ található róla a közösségi fórumokon.

3.4.1.4 Előnyök és korlátok

A Prometheus fő előnyei közé tartozik az egyszerű telepítés és üzemeltetés, a robusztus lekérdező nyelv, valamint a széles körű integráció más eszközökkel. Azonban vannak korlátai is: a beépített magas rendelkezésre állás hiánya, a hosszú távú adatmegőrzés sajátosságai, általában hetekre vagy hónapokra optimalizált. A nyílt forráskódú verzió nem támogat klaszterezést, bár léteznek kerülő megoldások, mint például a Thanos [12] projekt.

3.4.2 InfluxDB

Az InfluxDB [13] egy speciálisan idősor-adatok tárolására tervezett adatbázis, amelyet az InfluxData cég fejleszt 2013 óta. A legújabb generáció, az InfluxDB 3.x, – amely 2023-ban jelent meg először felhőalapú változatban, majd 2025 áprilisában nyílt forráskódú

(Core) verzióban –, alapvető változásokat hozott a platform architektúrájában és képességeiben.

3.4.2.1 Architektúrális tervezés

Az InfluxDB 3 generációban a korábbi Go alapú implementációt felváltotta egy Rust-ban írt új architektúra. Ez három fő felhasználói problémát old meg, amelyek a korábbi verziókban jelentős korlátozást jelentettek.

- Korlátlan kardinalitás: A rendszer képes korlátlan számú egyedi idősor-kombinációt kezelni teljesítményvesztés és költségnövekedés nélkül, ami lehetővé teszi a metaadatok gazdagságának maximalizálását.
- Natív SQL támogatás: Teljes értékű SQL lekérdezési képességet biztosít az InfluxQL mellett, ami jelentősen megkönnyíti a meglévő SQL-alapú eszközök integrációját.
- Elválasztott számítás és tárolás (Object Storage): Az új architektúra lehetővé teszi költséghatékony object store-ok (S3, Azure Blob, GCS) használatát hosszú távú adatmegőrzéshez, emellett kiváló lekérdezési teljesítményt nyújtva.

3.4.2.2 Adatgyűjtési modell

Az InfluxDB push-alapú modellt alkalmaz, ahol a kliensek aktívan küldik az adatokat az adatbázisba HTTP API-n vagy különböző protokollokon keresztül.

3.4.2.3 Lekérdező nyelvek:

Az InfluxDB 3.x két fő lekérdező nyelvet támogat:

- SQL: Teljes értékű SQL támogatás standard szintaxissal. Ez a legtermészetesebb választás a relációs adatbázisokból érkező fejlesztők számára.
- InfluxQL: A korábbi verziókból ismert SQL-szerű nyelv, amely visszafelé biztosítja a kompatibilitást.

3.4.2.4 Licencelés

Az InfluxDB 3 Core nyílt forráskódú és ingyenesen használható, ám bizonyos enterprise funkciók (pl. klaszterezés, kereskedelmi támogatás) csak az InfluxDB Cloud vagy Enterprise kereskedelmi változatokban érhetők el.

3.4.2.5 Közösségi támogatás

Az InfluxDB dokumentációja átfogó és jól strukturált. A közösség nagy, bár a 3.x verzió viszonylagos újdonsága miatt jelenleg kevesebb tapasztalat és információ áll rendelkezésre.

3.4.2.6 Előnyök és hátrányok

Az InfluxDB 3.x előnyei közé tartozik a korlátlan számú metrika kezelése, a könnyen tanulható SQL lekérdező nyelv, az olcsó hosszú távú adattárolás, valamint a gyors lekérdezések nagy adatmennyiség esetén. A hátrányok közé sorolható, hogy viszonylag új rendszer kevés tapasztalattal, kisebb közösségi támogatással és a fejlett funkciók csak a fizetett verziókban érhetőek el.

3.4.3 VictoriaMetrics

A VictoriaMetrics [14] egy nagy teljesítményű, költséghatékony és skálázható, nyílt forráskódú idősor-adatbázis, amelyet kifejezetten nagy mennyiségű felügyeleti adat kezelésére terveztek. A projektet 2018-ban indították és gyorsan népszerűvé vált az erőforrás hatékonysága miatt.

3.4.3.1 Architektúra és adatmodell

A VictoriaMetrics architektúrája az egyszerűségre és a teljesítményre összpontosít. Az egy csomópontú verzió egyetlen bináris fájlként futtatható, ami rendkívül egyszerűvé teszi a telepítést és az üzemeltetést. Az adatmodellje megegyezik a Prometheus-éval, metrikanevekből és kulcs-érték párokból álló címkékből (label) épül fel. A rendszer egyik legfontosabb erőssége a saját fejlesztésű tárolómotorja, amely rendkívül hatékony adattömörítést tesz lehetővé.

3.4.3.2 Adatgyűjtési modell

A VictoriaMetrics rugalmas adatgyűjtési modellt kínál, amely mind a pull, mind a push alapú megközelítést támogatja. Képes Prometheus-kompatibilis célpontokról adatokat lekérdezni (pull), emellett fogadni tud adatokat az InfluxDB line protocoll és OpenTSDB protokollokon keresztül is (push). Ez a kettős képesség rendkívül sokoldalúvá teszi, mivel könnyen integrálható már létező környezetekbe.

3.4.3.3 MetricsQL lekérdező nyelv

A VictoriaMetrics a MetricsQL lekérdező nyelvet használja, amely a Prometheus PromQL nyelvének egy visszafelé kompatibilis kiterjesztése. A MetricsQL megtartja a PromQL szintaxisát és funkcióit, de számos fejlesztést és kényelmi funkciót is bevezet, amelyek megkönnyítik az összetett lekérdezések írását.

3.4.3.4 Licencelés és közösségi támogatás

A VictoriaMetrics alap verziója Apache 2.0 licenccel rendelkezik, így teljesen nyílt forráskódú és ingyenesen használható. A klaszterezett, magas rendelkezésre állású verzió kereskedelmi licenc alatt érhető el, amely vállalati támogatást is nyújt. A dokumentációja gyakorlatias, a közössége pedig aktív, bár méretét tekintve még elmarad a versenytársaitól.

3.4.3.5 Előnyök és hátrányok

A VictoriaMetrics legfőbb előnyei a kiemelkedő teljesítmény, az alacsony erőforrás-igény (CPU, RAM, tárhely), a hatékony adattömörítés és a rugalmas adatgyűjtési modell. A MetricsQL lekérdező nyelv fejlesztései és a Prometheus-szal való kompatibilitás vonzóvá teszik a meglévő felügyeleti rendszerek kiváltására. Hátránya, hogy a klaszterezett verzió beállítása és üzemeltetése bonyolultabb lehet, és a közössége még nem olyan nagy, mint más megoldásoké.

3.4.4 Összehasonlítás

A három adatbázis összevetése alapján a projekt számára a legmegfelelőbb megoldás kiválasztása a specifikus követelmények mentén történt. Az alábbi táblázat (3.4.4.1) foglalja össze a legfontosabb szempontokat:

Szempont	Prometheus	InfluxDB 3.x	VictoriaMetrics
Adatgyűjtés	Pull (elsődleges)	Push	Push & Pull
Lekérdező nyelv	PromQL	SQL, InfluxQL	MetricsQL
Érettség	Nagyon magas (CNCF Graduated)	Új (3.x verzió)	Közepes
Közösségi támogatás	Kiemelkedő, legnagyobb	Növekvő, de a 3.x-re korlátozott	Jó, de kisebb

Erőforrásigény	Közepes	Magas (Rust/Arrow alapok)	Alacsony
Licencelés (alap)	Apache 2.0	Apache 2.0 / MIT	Apache 2.0

3.4.4.1 Táblázat: Adatbázisok összehasonlítása

Bár az InfluxDB 3.x natív SQL támogatás képessége vonzó, a platform viszonylagos újdonsága miatt a közösségi tudásbázis és a bevált gyakorlatok még nem olyan kiforrottak, mint a versenytársak esetében. A VictoriaMetrics kiemelkedő teljesítményt és erőforrás-hatékonyságot kínál, de a projekt jelenlegi állásában ezek az előnyök nem bírnak akkora súllyal, mint a széles körű ismertség és a könnyen érthető dokumentáció.

A fentiek mérlegelése után a Prometheus mellett döntöttem. A választás legfőbb indoka a rendszer rendkívüli beváltsága és a mögötte álló hatalmas közösségi támogatás. Mivel a Prometheus a cloud-native ökoszisztéma szabványává vált, a dokumentációja rendkívül részletes, és szinte bármilyen felmerülő problémára található megoldás a közösségi fórumokon. Ez a projektfejlesztés során felbecsülhetetlen értékű, mivel jelentősen felgyorsítja a hibaelhárítást és az integrációt.

3.5 Megjelenítő

A rendszer vizualizációs rétegének kiválasztása kiemelet jelentőségű, mivel ez a komponens teszi lehetővé a komplex hálózati metrikák és előrejelzések ember által értelmezhető formában történő bemutatását. A megfelelő eszköznek képesnek kell lennie a Prometheus adatbázisból származó idősoros adatok hatékony lekérdezésére és interaktív, testre szabható dashboard-okon való megjelenítésére. A piacon elérhető számos megoldás közül kettő emelkedik ki, mint a legelterjedtebb választás: a Grafana és a Kibana.

3.5.1 Grafana

A Grafana [15] egy nyílt forráskódú analitikai és vizualizációs platform, amelyet kifejezetten idősoros adatok megjelenítésére és elemzésére terveztek. Egyik legnagyobb erőssége a rugalmas adatforrás-kezelés. Natívan támogatja a legnépszerűbb idősor-adatbázisokat, köztük a Prometheust, az InfluxDB-t és a VictoriaMetrics-et, de képes csatlakozni relációs adatbázisokhoz (pl. MySQL, PostgreSQL) és egyéb rendszerekhez is.

A Grafana lehetővé teszi komplex, interaktív dashboard-ok létrehozását, amelyek különböző panelekből (pl. grafikonok, táblázatok, hőtérképek, egyedi értékek) épülnek fel. Minden panel egyedi lekérdezést futtat a kiválasztott adatforráson, így a felhasználók egyetlen felületen láthatják a rendszer különböző aspektusait. A platform széles körben elterjedt a DevOps és IT infrastruktúra felügyeleti területeken, és a Prometheus-szal párosítva gyakorlatilag iparági szabványnak tekinthető a metrikaalapú megfigyelésben.

3.5.2 Kibana

A Kibana [16] az Elastic Stack (ELK Stack) vizualizációs komponense, amely szorosan integrálódik az Elasticsearch kereső- és analitikai motorral. Elsődleges célja a nagy mennyiségű, strukturálatlan vagy félig strukturált naplóbejegyzések (log) adatok keresése, elemzése és vizualizációja. A Kibana rendkívül erős az adatok feltárásában, lehetővé téve a felhasználók számára, hogy interaktívan szűrjék és aggregálják a naplóbejegyzéseket, majd az eredményeket különböző diagramokon (pl. oszlopdiagramok, térképek) jelenítsék meg.

Míg a Kibana képes metrikák megjelenítésére is, ha azok Elasticsearch-ben vannak tárolva, az elsődleges fókusza egyértelműen a naplóbejegyzések elemzésén van. Adatforrás támogatása korlátozottabb, mint a Grafanáé, mivel szinte kizárólag az Elasticsearch-re épül.

3.5.3 Összehasonlítás

A két platform összehasonlításakor a legfontosabb különbség az alapvető felhasználási területben rejlik. A Grafana az idősoros metrikák vizualizációjára specializálódott, míg a Kibana a naplóbejegyzések elemzésére.

Szempont	Grafana	Kibana
Elsődleges felhasználás	Idősoros metrikák felügyelete	Naplóbejegyzések analízise és feltárása
Adatforrás támogatás	Széleskörű (Prometheus, InfluxDB stb.)	Elsősorban Elasticsearch
Lekérdező nyelv	Az adatforrás saját nyelve (pl. PromQL)	Elasticsearch DSL

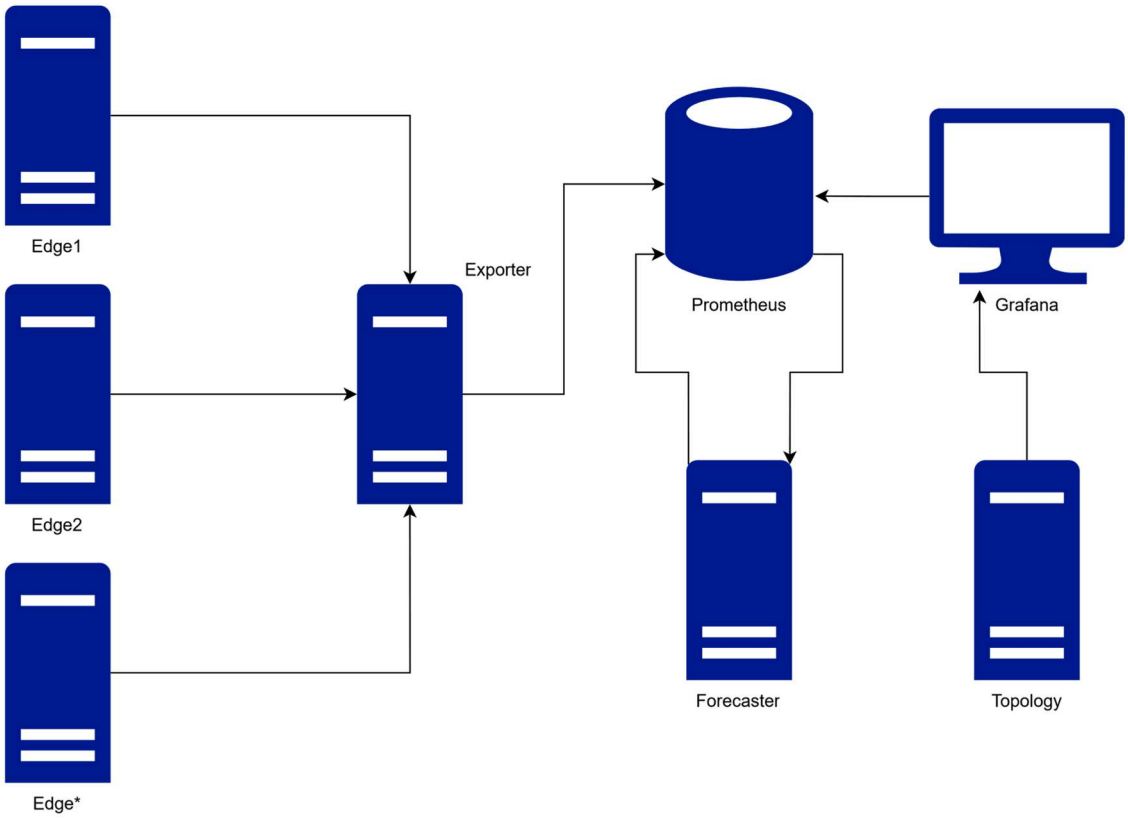
Ökoszisztéma	Gyakran használt Prometheus-szal	Az Elastic Stack (ELK) része

3.5.3.1 Táblázat: Megjelenítők összehasonlítása

A projekt követelményei egyértelműen az idősoros metrikák feldolgozására és megjelenítésére fókuszálnak, amelyet a Prometheus adatbázis szolgáltat. Ebben a kontextusban (3.5.3.1) a Grafana a természetes és hatékonyabb választás.

3.6 Adatáramlás

A rendszeren belüli adatáramlás egy több lépcsős folyamatot követ, amely biztosítja a metrikák megbízható gyűjtését, tárolását, vizualizációját és az előrejelzések generálását. Az alábbi ábra mutatja az adatfolyamokat a komponensek között:



3.2. ábra: Adatáramlás

A folyamat öt fő szakaszra bontható, a következő fejezetekben ezek kerülnek részletes bemutatásra.

3.6.1 Metrikák gyűjtése és exportálása

A folyamat a hálózati csomópontokon (pl. „edge1”, „edge2”) futó tanszéki programmal kezdődik. Ezek a csomópontok periodikusan, UDP üzenetek formájában küldik el a működésükről szóló, JSON formátumú felügyeleti adatokat (notification-öket) egy központi feldolgozó alkalmazásnak (UDP Receiver & Metrics Exporter) (3.2. ábra). Ez a központi komponens fogadja, feldolgozza és az adatbázisnak megfelelő formátumra alakítja a beérkezett adatokat. A feldolgozott metrikákat egy HTTP API végponton teszi elérhetővé, amelyet az adatbázis rendszeres időközönként le tud kérdezni.

3.6.2 Adatok tárolása

A Prometheus (3.2. ábra) szerver a pull alapú adatgyűjtési modelljének megfelelően, előre beállított időközönként lekérdezi a metrikákat a központi feldolgozó által biztosított HTTP végpontról. Az így begyűjtött adatokat a hozzájuk tartozó címkékkel (label) együtt a saját idősor-adatbázisában tárolja, lehetővé téve a hatékony, időalapú lekérdezéseket.

3.6.3 Adatok vizualizációja

A megjelenítő felület (3.2. ábra) közvetlenül a Prometheus adatbázishoz, mint adatforráshoz csatlakozik. A felhasználói felületen található grafikonok és panelek a háttérben lekérdezéseket futtatnak az adatbázison, hogy megjelenítsék a valós idejű és múltbeli metrikákat. A felhasználó a felületen keresztül interaktívan szűrhet, nagyíthat és elemezhet különböző időintervallumokat.

3.6.4 Előrejelzési folyamat

Az előrejelző modul (3.2. ábra) egy különálló, automatizált komponensként működik. A modul automatikusan felderíti a Prometheusban tárolt releváns metrika-címke kombinációkat. Környezeti változókban megadott konfiguráció alapján kiválasztja az előrejelezni kívánt metrikákat, majd lekérdezi azok múltbeli adatait egy meghatározott időablakra visszamenőleg. Amennyiben elegendő adat áll rendelkezésre, a modul illeszt egy előrejelzési modellt, majd ezt felhasználva előrejelzéseket generál a jövőbeli értékekre. A kapott előrejelzéseket a rendszer előrejelzés specifikus metrika neveken visszatárolja a Prometheus adatbázisba, így azok a megjelenítő felületen a tényleges adatokkal együtt ábrázolhatók.

3.6.5 Topológia megjelenítés

A folyamat egy különálló szolgáltatással kezdődik (3.2. ábra), amely a hálózati topológia vizualizációjáért felel. A felhasználó egy API végponton keresztül feltölti a hálózatot leíró Mininet Python szkriptet. A szolgáltatás feldolgozza ezt a fájlt, kinyeri belőle a csomópontokat, az összeköttetéseket és a statikus IP-címeket. Ezen adatok alapján egy grafikus ábrát generál a hálózatról, amit egy másik API végponton keresztül szolgáltat. Ezt a képet a központi megjelenítő felület lekérdezi és beágyazza a dashboard-ba, kontextust biztosítva a metrikákhoz.

3.7 Előrejelzés

A rendszer követelménye a hálózati metrikák jövőbeli alakulásának előrejelzése, amely lehetővé teszi a proaktív hálózatkezelést és a potenciális problémák korai felismerését. Az irodalomkutatás során vizsgált klasszikus statisztikai és modern gépi tanulási módszerek közül a projekt igényeinek és kereteinek leginkább megfelelő technika kiválasztása volt a cél.

A gépi tanulás-alapú modellek, mint például az LSTM hálózatok, rendkívül hatékonyak lehetnek komplex, nemlineáris mintázatok felismerésében, azonban jelentős hátrányokkal is rendelkeznek. Magas számítási erőforrás-igényük, a tanításhoz szükséges nagy mennyiségű adat, valamint a modellek „fekete doboz” jellege, ami megnehezíti az eredmények értelmezését, mind olyan tényezők, amelyek a projekt jelenlegi fázisában plusz kihívást jelentenének.

Ezzel szemben a klasszikus idősor-alapú modellek, mint az ARIMA, alacsonyabb számítási igénnyel rendelkeznek, könnyebben implementálhatók és az eredményeik jobban interpretálhatók. Bár az ARIMA modellek alapvetően egyváltozósak és lineáris összefüggéseket feltételeznek, a hálózati metrikák (pl. csomagvesztés, késleltetés) előrejelzésére a gyakorlatban elegendő és megbízható alapot nyújtanak a cél eléréséhez.

A fentiek mérlegelése alapján a projekt előrejelzési komponensének megvalósításához az ARIMA (Autoregressive Integrated Moving Average) modellt választottam. A döntés fő indokai:

- Egyszerűség és alacsony erőforrásigény: Az ARIMA modell implementációja és futtatása nem igényel speciális hardvert (pl. GPU-t), így könnyen integrálható a meglévő mikroszolgáltatási architektúrába.

- Interpretálhatóság: A modell statisztikai alapjai és paraméterei (p, d, q) lehetővé teszik az előrejelzési logika megértését.
- Megfelelő a projekt céljaira: A diplomaterv keretein belül az ARIMA modell egy jól bevált megoldást kínál az idősor-előrejelzés alapelveinek demonstrálására.

Az előrejelző modul egy különálló Python szolgáltatásként valósul meg, amely periodikusan lekérdezi a releváns múltbeli adatokat a Prometheus adatbázisból, stacionáriussá alakítja azokat, majd hiperparaméter-optimalizálással megkeresi a legjobb ARIMA modellt. Az elkészült előrejelzéseket a szolgáltatás visszatárolja az adatbázisba, ahol azok a dashboard-okon a valós adatokkal együtt megjeleníthetők.

3.8 Dashboard-ok

A rendszer vizuális rétegének központi elemei a Grafana dashboard-ok. A dashboard-okkal szembeni alapvető igény, hogy a rendelkezésre álló, nagy mennyiségű adatot úgy jelenítsék meg, hogy azok könnyen értelmezhetőek legyenek, kiemeljék a legfontosabb információkat, és egy átfogó áttekintés mellett részletesebb képet is tudjanak adni a hálózat állapotáról. Ezen célok elérésére négy fő dashboard kerül kialakításra, amelyek különböző szinteken nyújtanak betekintést a rendszer működésébe.

3.8.1 Általános áttekintő

Ez a dashboard a hálózat teljes állapotának magas szintű áttekintésére szolgál. Célja, hogy a felhasználó egyetlen pillantással felmérhesse a rendszer általános „egészségét”. A felületen minden egyes felügyelt csomópont (host) legfontosabb adatai jelennek meg.

A megjelenített kulcsmetrikák közé tartoznak a küldött és kapott csomagok és oktettek száma, illetve a MIP pontok száma. Ezen felül a dashboard tartalmaz egy panelt, amely listázza a csomópontok által küldött eseményvezérelt push üzeneteket, így a fontos események (pl. telnet bejelentkezés) is azonnal láthatóvá válnak.

3.8.2 Csomópont specifikus

Míg az általános dashboard egy madártávlati áttekintő képet ad a hálózatról, ez a felület egyetlen kiválasztott csomópont részletes áttekintését teszi lehetővé. A felhasználó egy legördülő menüből választhatja ki az elemezni kívánt host-ot, és a dashboard összes panelje dinamikusan frissül a kiválasztott csomópont adataival.

Itt minden, a csomópont által szolgáltatott metrika részletes, idősoros grafikonon jelenik meg, többek között:

- a PRF és PEF objektumok összes elérhető metrikája,
- a hálózati interfészek bejövő és kimenő forgalma (csomagok és oktettek),
- a parser és stream specifikus statisztikák,
- a belső működést felügyelő MIP pontok adatai.

Ez a nézet elengedhetetlen a hibakereséshez és egy-egy csomópont viselkedésének elemzéséhez.

3.8.3 Interfész összehasonlító

Ez a speciális dashboard a redundáns útvonalak működésének ellenőrzésére szolgál. A felület lehetővé teszi a felhasználó számára, hogy kiválasszon egy „A” és egy „B” csomópontot, majd ezeken belül egy-egy interfészt. A dashboard ezután egymás mellett ábrázolja az „A” pontból küldött és a „B” pontba beérkezett csomagok és oktettek számát.

A grafikonok vizuálisan megjelenítik a két interfész forgalma közötti esetleges eltéréseket. Ideális esetben a küldött és a fogadott adatok száma megegyezik. Ha a fogadott oldalon kevesebb csomag látható, az csomagvesztésre utal a két pont közötti útvonalon.

3.8.4 Topológia

Ez a dashboard egy statikus, de informatív képet ad a hálózat felépítéséről. A felületen a topológia-generátor szolgáltatás által készített ábra jelenik meg, amely bemutatja a hálózati csomópontokat és az azokat összekötő linkeket. Az ábra segít kontextusba helyezni a többi dashboard-on látott metrikákat, vizuálisan is megjelenítve, hogy melyik adat melyik hálózati eszközhöz vagy útvonalhoz tartozik. Ez különösen a redundáns útvonalak azonosításában és a forgalom útjának megértésében nyújt támogatást.

4 Megvalósítás

Ez a fejezet a projekt és a tervekben leírtak alapján való implementációt fogja részletezni. A rendszer egy mikroszolgáltatás-alapú architektúrára épül, amelyet a Docker konténerizációs technológia támogat. A konténerek összehangolt kezelését a Docker Compose [17] biztosítja, amely lehetővé teszi a több komponensből álló alkalmazás definiálását és futtatását egyetlen YAML konfigurációs fájl segítségével.

4.1 Tesztkörnyezet telepítése

A fejlesztés és a futtatás alapfeltétele egy megfelelően konfigurált tesztkörnyezet, amely magában foglalja a szükséges operációs rendszer-szintű eszközöket és a konténerizációs platformot.

4.1.1 WSL (Windows Subsystem for Linux)

A fejlesztés Windows operációs rendszeren történt, de a Linux-alapú eszközök és a konténerizáció zökkenőmentes használata érdekében a Windows Subsystem for Linux (WSL) 2-es verziója került telepítésre. A „wsl --install” parancs futtatásával egy alapértelmezett Ubuntu disztribúció települ, amely teljes értékű Linux környezetet biztosít a Windows-on belül.

4.1.2 Docker Desktop

A konténerizált alkalmazások futtatásához a Docker Desktop for Windows szoftver lett telepítve. Ez az alkalmazás integrálódik a WSL 2-vel, lehetővé téve a Linux konténerek natív sebességű futtatását. A projektben szereplő összes szolgáltatás (Prometheus, Grafana, és az egyedi fejlesztésű Python alkalmazások) Docker konténerként fut. Egy docker compose yaml fájl definiálja ezeket a szolgáltatásokat, a közöttük lévő kapcsolatokat, a port-átirányításokat és a maradandó adattároláshoz szükséges köteteket.

4.1.3 Prometheus és Grafana

A rendszer két központi eleme, a Prometheus és a Grafana, szintén a Docker Compose konfiguráció részeként indul.

4.1.3.1 Prometheus

A hivatalos „prom/prometheus:latest” Docker képet használja. A konfigurációja egy yaml fájlban található, amely a docker compose file-ban megadottak szerint a konténerhez van csatolva. Ez a fájl definiálja a lekérdezési célpontokat (scrape targets), mint például a metrikákat szolgáltató Exporter-t.

4.1.3.2 Grafana

A hivatalos „grafana/grafana:latest” konténer képet használja. A perzisztens tárolás és az automatikus konfiguráció (provisioning) a docker compose fileban definiált köteteken keresztül valósul meg.

A teljes környezet indítását és a különböző komponensek közötti kapcsolat tesztelését egy shell szkript végzi, amely elindítja a Docker Compose szolgáltatásokat, és „curl” segítségével ellenőrzi a Prometheus és az Exporter végpontok elérhetőségét.

4.2 Implementáció

A következő alfejezetek a rendszer egyedi fejlesztésű komponenseinek implementációját ismertetik. A kódbázis [18] verziókezelését a Git [19] rendszer biztosítja, amely lehetővé teszi a fejlesztési folyamat következetes nyomon követését.

4.2.1 Exporter implementáció

A tanszéki szoftver által küldött UDP csomagokban lévő JSON adatok Prometheus metrikákká alakításáért egy Python alkalmazás felel. Ez az alkalmazás a hivatalos „prometheus-client” [20] Python könyvtárat használja.

A szkript definiálja a szükséges metrika típusokat (Gauge, Counter) a megfelelő címkékkel (labels). Amikor egy új JSON üzenet érkezik, a szkript feldolgozza azt, és a benne található adatok alapján frissíti a megfelelő metrikák értékét. Például a „seqrec_discarded_packets_counter” egy Counter, amely a „hostname”, „component_name” és egyéb címkék alapján számolja az eldobott csomagokat. Az alkalmazás egy beépített HTTP szervert indít, alapértelmezetten a 9100-as port-on, amely a „/metrics” végponton szolgáltatja az aktuális metrika értékeket a Prometheus számára.

4.2.2 Topológia implementáció

A hálózati topológia vizualizációjáért egy különálló, Flask [21] alapú Python mikroszolgáltatás felel. Ez a szolgáltatás két fő API végpontot biztosít:

- /upload: Ezen a POST végponton keresztül tölthető fel egy Mininet topológiát leíró python fájl. A szolgáltatás a feltöltött fájlt dolgozza fel, kinyerve belőle a hálózati elemeket (host-ok, switch-ek) és a kapcsolatokat.
- /topologyImage: Ez a GET végpont egy SVG vagy PNG képet generál a feltöltött topológiából a „graphviz" [22] könyvtár segítségével, majd visszaadja azt a kliensnek.

A feltöltés tesztelésére és egyszerűsítésére egy szkript is készült, amely egy adott topológia fájlt küld el a szolgáltatásnak.

4.2.3 Előrejelző modul implementáció

Az előrejelző modul egy Python szolgáltatás, amelynek központi eleme az „ARIMAForecaster" osztály. Ez az osztály felel az ARIMA modellek illesztésért és az előrejelzések készítésért.

4.2.3.1 Konfiguráció

A modul működése környezeti változókon keresztül finomhangolható, ezek az alábbiak:

- PROMETHEUS_BASE_URL: Beállítja a Prometheus adatbázis elérési útját, ahonnan a metrikák lekérdezésre kerülnek.
- FORECAST_API_PORT: Meghatározza a port-ot, amelyen az előrejelző szolgáltatás API-ja elérhető lesz.
- FORECAST_MODEL_DIR: Kijelöli a könyvtárat, ahová a rendszer az illesztett előrejelzési modelleket menti.
- FORECAST_CADENCE_SECONDS: Szabályozza, hogy a szolgáltatás milyen időközönként (másodpercben) futtasson új előrejelzéseket.
- FORECAST_MAX_CONCURRENCY: Korlátozza az egy időben, párhuzamosan futtatható előrejelzési folyamatok maximális számát.
- FORECAST_RETRAIN_DAYS: Megadja, hogy a meglévő modelleket hány naponta kell automatikusan újra tanítani a friss adatok alapján.

- FORECAST_FILTER_CONFIG: Beállítja az elérési útját annak a konfigurációs fájlnak, amely alapján a rendszer szűri az előrejelzésbe bevont metrikákat.
- FORECAST_LOG_LEVEL: Meghatározza a naplózás részletességének szintjét (pl. INFO, DEBUG).
- FORECAST_LOOKBACK_MINUTES: Beállítja, hogy a modell az illesztéséhez hány percnyi múltbeli adatot vegyen figyelembe.
- FORECAST_HORIZON_MINUTES: Kijelöli az előrejelzés időhorizontját, azaz, hogy hány percre előre jósoljon a modell.
- FORECAST_ARIMA_ORDER: (Opcionális) Lehetővé teszi az ARIMA modell (p,d,q) paramétereinek manuális beállítását.

4.2.3.2 Működés

Az előrejelző szolgáltatás egy automatizált ciklusban működik. Periodikusan lekérdezi a Prometheus adatbázisból a releváns metrikák múltbeli adatait, amelyeket átad az "ARIMAForecaster" komponensnek.

Ha egy metrikához nincs elegendő múltbeli adat, a folyamat nem folytatódik és a kimenetre ír egy üzenetet. Amennyiben rendelkezésre áll elegendő adat, de még nem létezik modell az adott metrikához, a szolgáltatás előfeldolgozza az adatokat majd illeszt egy új ARIMA modellt, amit elment a későbbi használathoz. Ezt követően a meglévő vagy frissen tanított modellt használja a jövőbeli értékek előrejelzésére.

A rendszer a konfigurációban megadott időközönként automatikusan újra is illeszti a modelleket, hogy azok naprakészek maradjanak. Végül az előrejelzések eredményeit új Prometheus metrikákként teszi közzé, így azok megjeleníthetők és elemezhetők lesznek.

4.2.4 Dashboard-ok implementálása

A Grafana dashboard-ok JSON formátumban vannak definiálva. A konfiguráció ezt a mappát a Grafana konténer megfelelő könyvtárába csatolja. A Grafana automatikus „provisioning" funkciója biztosítja, hogy a szolgáltatás indulásakor a fájlban definiált beállítások alapján a Grafana automatikusan betöltse az összes csatolt mappában található JSON fájlt és létrehozza belőlük a dashboard-okat. Ez a megközelítés lehetővé teszi, hogy a dashboard-ok a kóddal együtt legyenek verzió kezelve, és a rendszer telepítésekor azonnal, manuális beavatkozás nélkül rendelkezésre álljanak.

5 Eredmények elemzése

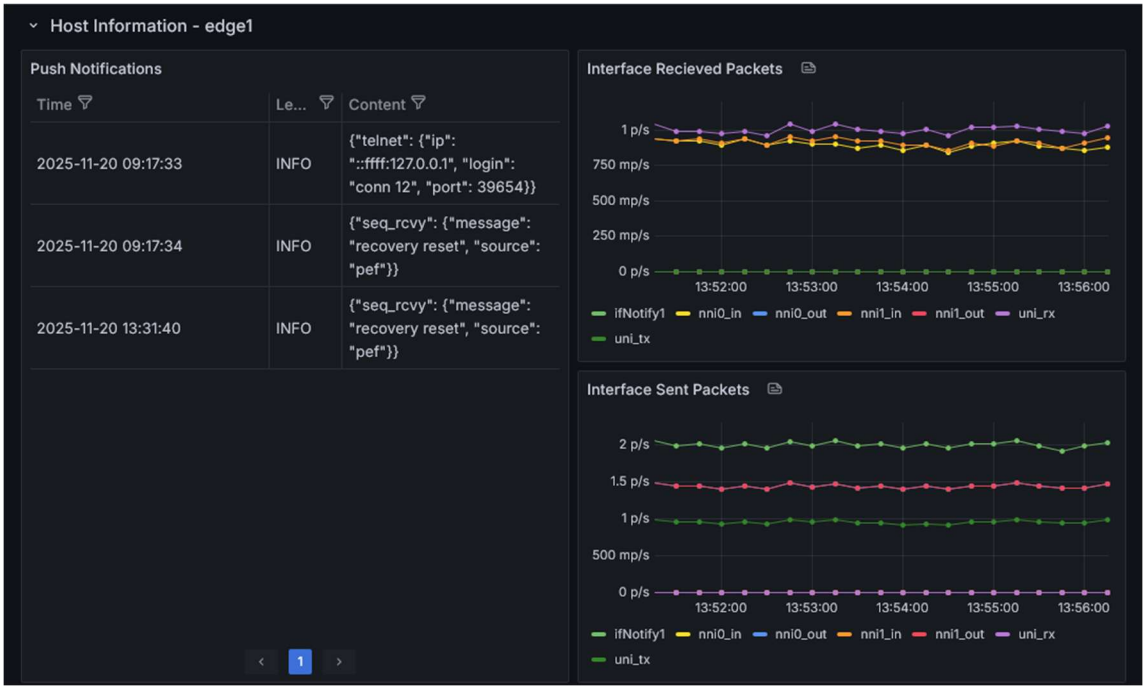
Ebben a fejezetben a feladat gyakorlati megvalósítása kerül bemutatásra a vizualizációs felületeken (dashboard-okon) keresztül, melyek értelmezését részletes magyarázatok segítik. A fejezet második részében megvizsgálom az előrejelzések pontosságát a megjelenített adatok elemzésével.

5.1 Dashboard-ok bemutatása

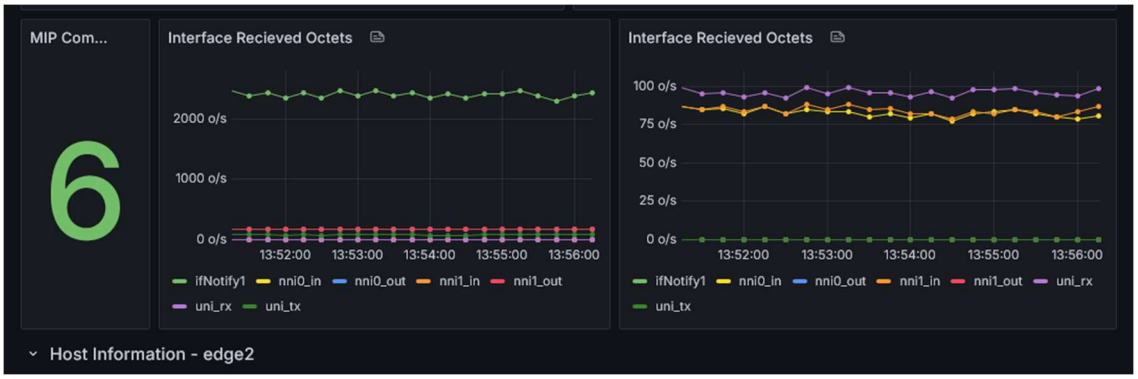
Az alábbiakban a 4.2.4. fejezetben ismertetett dashboard-ok gyakorlati megvalósítását mutatom be képernyőképek segítségével.

5.1.1 Általános áttekintés

Ahogy az ábrákon (5.1. ábra, 5.2. ábra) is látható, a felület host-onként szegmentálva mutatja be a telemetriai adatokat. A bal oldali táblázat a push értesítéseket tartalmazza, míg a jobb oldali diagramok az interfészek átviteli teljesítményét (csomag és oktett) ábrázolják. A grafikonokon látható, hogy a forgalom sebessége közel konstans, azonban az egyes interfészeken különböző sebességek mérhetők. A bal alsó sarokban található, hogy az adott csomóponton hány darab MIP pont van konfigurálva.



5.1. ábra: Push notification-ök és csomagok



5.2. ábra: MIP komponensek és oktettek

5.1.2 Csomópont specifikus

A következőkben a csomópont-specifikus dashboard részletes bemutatására kerül sor.

5.1.2.1 Interface rész

A 5.3. ábra és az 5.4. ábra a kiválasztott hálózati csomópont részletes forgalmi statisztikáit szemlélteti. A felületen grafikus formában követhető nyomon az eszköz összes interfészének adatforgalma, külön bontva a küldött és fogadott csomagok (packets), illetve oktettek (octets) számát.

5.3. ábra: Fogadott forgalom grafikonok

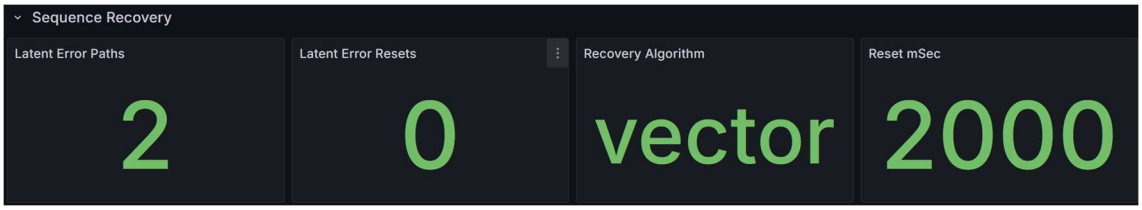
5.4. ábra: Küldött forgalom grafikonok

5.1.2.2 Sequence Recovery

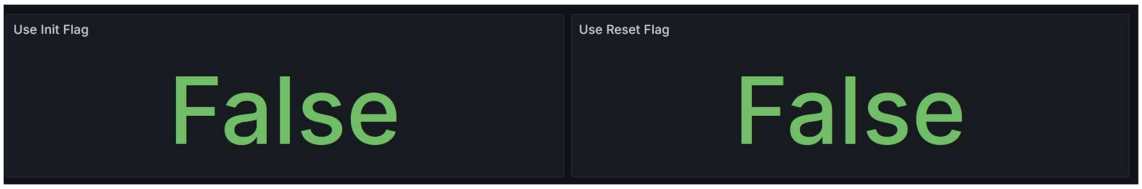
A Sequence Recovery funkció működését az 5.5 – 5.8. ábrák mutatják be részletesen. A 5.5. ábra a továbbított és az eldobott csomagok számának időbeli alakulását mutatja. A 5.6. ábra a helyreállítási folyamat belső állapotait követi nyomon: a helyreállítási szekvenciaszámot, a visszaállítások számát, az előzménytároló hosszát és a látens hibákat. A 5.7. ábra és a 5.8. ábra a funkció legfontosabb konfigurációs paramétereit összesíti.

5.5. ábra: Átengedett és eldobott csomagok grafikonok

5.6. ábra: Folyam visszaállítás grafikonjai



5.7. ábra: Folyam visszaállítás paraméterei



5.8. ábra: Konfigurációs beállítások

5.1.2.3 Sequence Recovery – MIP

A Sequence Recovery objektum MIP adatait az 5.9 – 5.12. ábrák mutatják be részletesen. A 5.9. ábra a menedzsmenthez szükséges küldött és fogadott csomagokat, a helyreállítási visszaállításokat, az időzítési beállításokat, valamint a szekvenciaszámok alakulását és az előzmény méretét mutatja. A 5.10. ábra a csomagfeldolgozás statisztikáit, a látens hibák és visszaállítások számát, valamint a MIP szinteket és a hibás útvonalak számát jeleníti meg. A 5.11. ábra az egyes MIP pontokon konfigurált helyreállítási algoritmust mutatja. A 5.12. ábra pedig a vezérlő flag-ek aktuális logikai állapotát ábrázolja.

5.9. ábra: MIP információk

5.10. ábra: További MIP információk

5.11. ábra: Visszaállási algoritmus

5.12. ábra: Konfigurációs beállítások

5.1.2.4 Replicate

A Replicate funkció működését az 5.13. és 5.14 ábrák mutatják be. A 5.13. ábra felső részén a továbbított csomagok száma látható a grafikonon, míg az alsó részen a replikációs pipeline-ban lévő műveletek száma látható. A 5.14. ábra felső grafikonja a továbbított adatmennyiséget oktettekben ábrázolja. Az alsó szekció a kimeneti ágak maszkolási állapotát mutatja.

5.13. ábra: Áthaladt csomagok és művelet számok

5.14. ábra: Áthaladt oktettek és maszkolás

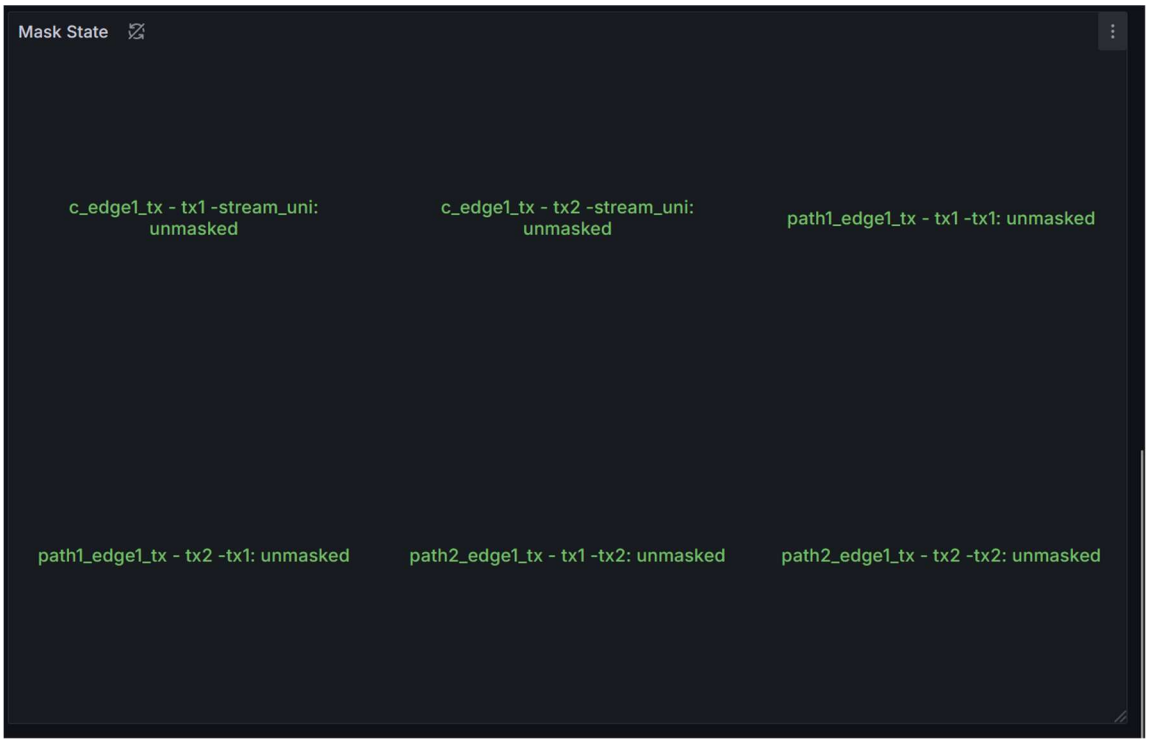
5.1.2.5 Replicate – MIP

Replicate objektum MIP részletes adatait az 5.15- 5.18. ábrák mutatják be. A 5.15. ábra a küldött menedzsment csomagokat és a továbbított csomagok számát ábrázolja. A 5.16. ábra a fogadott csomagok és a továbbított oktettek mennyiségét jeleníti meg. A 5.17. ábra a MIP szinteket és a műveletek számát mutatja az egyes replikációs ágakon. A 5.18. ábra pedig a maszkolási állapotokat szemlélteti, jelezve, hogy az egyes útvonalak engedélyezve vannak-e.

5.15. ábra: Küldött menedzsment csomagok és áthaladt csomagok

5.16. ábra: Fogadott menedzsment csomagok és áthaladt oktettek

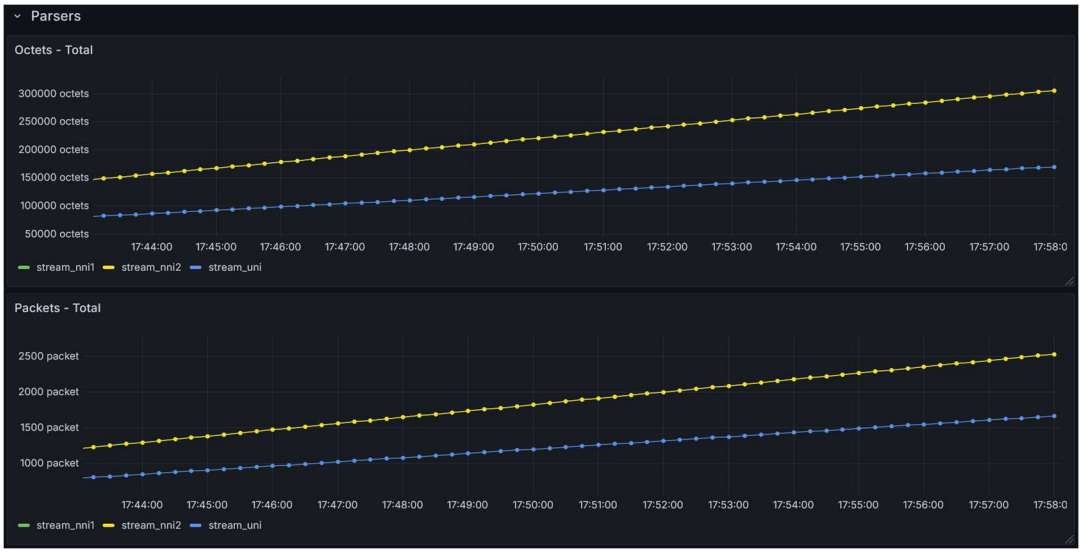
5.17. ábra: MIP szint és művelet számok



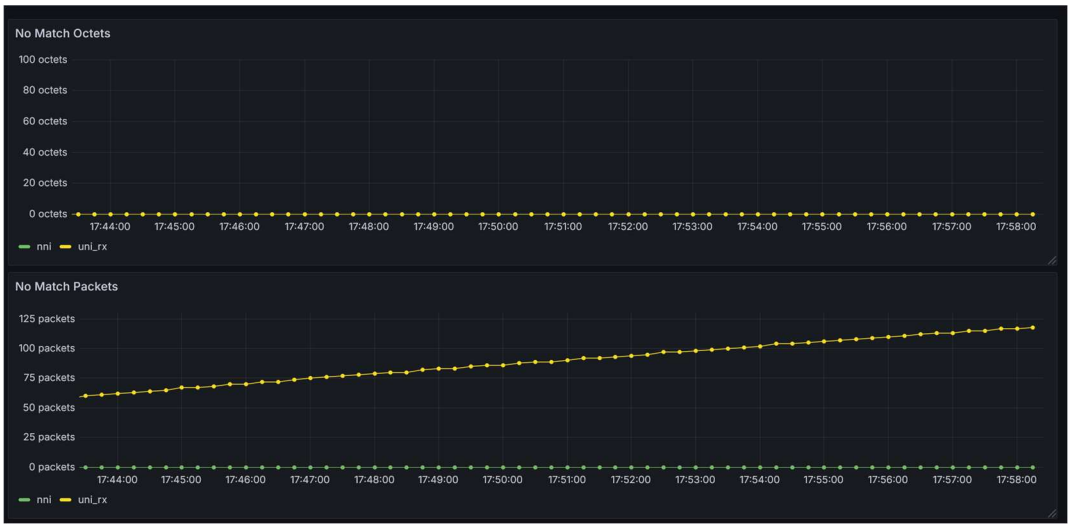
5.18. ábra: Maszkolás státusza

5.1.2.6 Parser

Az 5.19. és 5.20. ábra a Parser objektumok statisztikáit jeleníti meg. A 5.19. ábra az egyes azonosított stream-ek forgalmát mutatja. A 5.20. ábra a "No Match" adatokat ábrázolja, amelyek azokat a csomagokat és oktetteket jelölik, amelyeket nem sikerült egyetlen stream-hez sem hozzárendelni.



5.19. ábra: Folyamonkénti csomagok és oktettek



5.20. ábra: Egyetlen folyamhoz se tartozó csomagok és oktettek

5.1.2.7 Sequence Generator

A 5.21. ábra a Sequence Generator objektum vezérlőparamétereit szemlélteti. A panelen a konfigurációs jelzőbitek (flag-ek), a Reset és az Init funkciók aktuális logikai állapota olvasható le.

5.21. ábra: Konfigurációs beállítások

5.1.3 Interfész összehasonlító

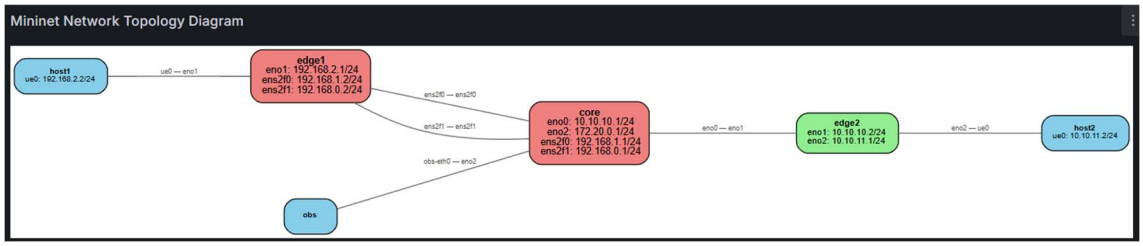
A 5.22. ábra és a 5.23. ábra a hálózati adatfolyamok összehasonlító elemzésére szolgáló felületet mutatja be. A dashboard lehetővé teszi két kiválasztott végpont (Host A és Host B) közötti forgalom közvetlen megfeleltetését. Az ábrákon a felső grafikonok a forrásoldali küldött és a céloldali fogadott adatmennyiséget ábrázolják, míg az alsó panelek a két érték közötti eltérést, azaz a számított adatvesztést vizualizálják. A bemutatott példában az „A” host szerepét az edge1 tölti be a két redundáns kimenő útvonallal, míg a „B” host az edge2 a bejövő interface-el. Az adatvesztés NetEm segítségével került előidézésre.

5.22. ábra: Küldött és elvesztett oktettek

5.23. ábra: Fogadott és elveszett csomagok

5.1.4 Topológia

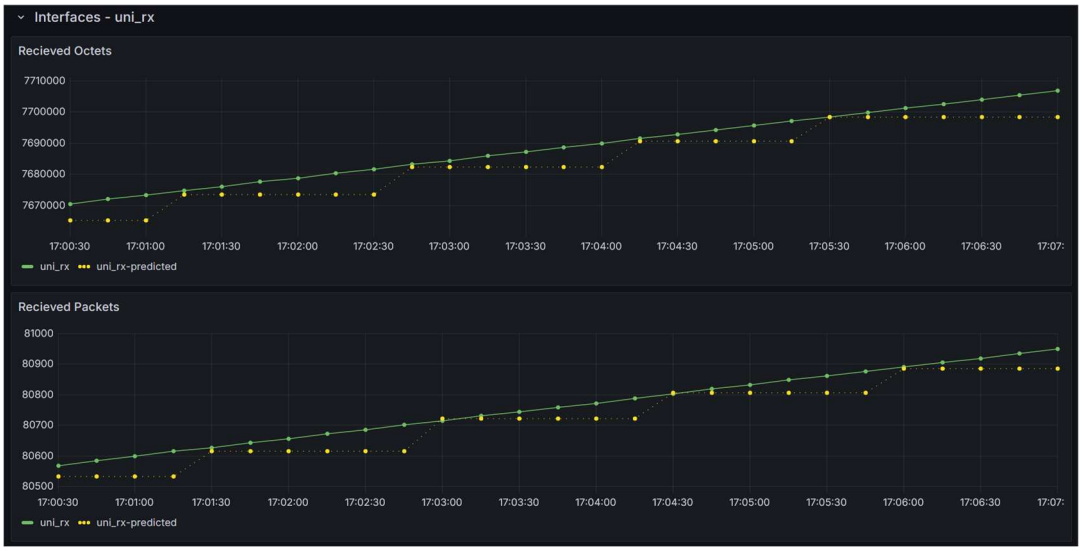
A 5.24. ábra a rendszer topológia-vizualizációs modulját szemlélteti. A dashboard a Mininet szimulációs környezet aktuális állapotát térképezi fel, gráf alapú nézetben megjelenítve a hálózati csomópontokat és a köztük lévő fizikai összeköttetéseket. A diagram nemcsak a struktúrát ábrázolja, hanem megjeleníti a csomópontok konfigurációs adatait is beleértve az interfész-neveket és IP-címeket, ezzel segítve a hálózati architektúra és az adatfolyam-útvonalak gyors áttekintését.



5.24. ábra: Topológia vizualziáció

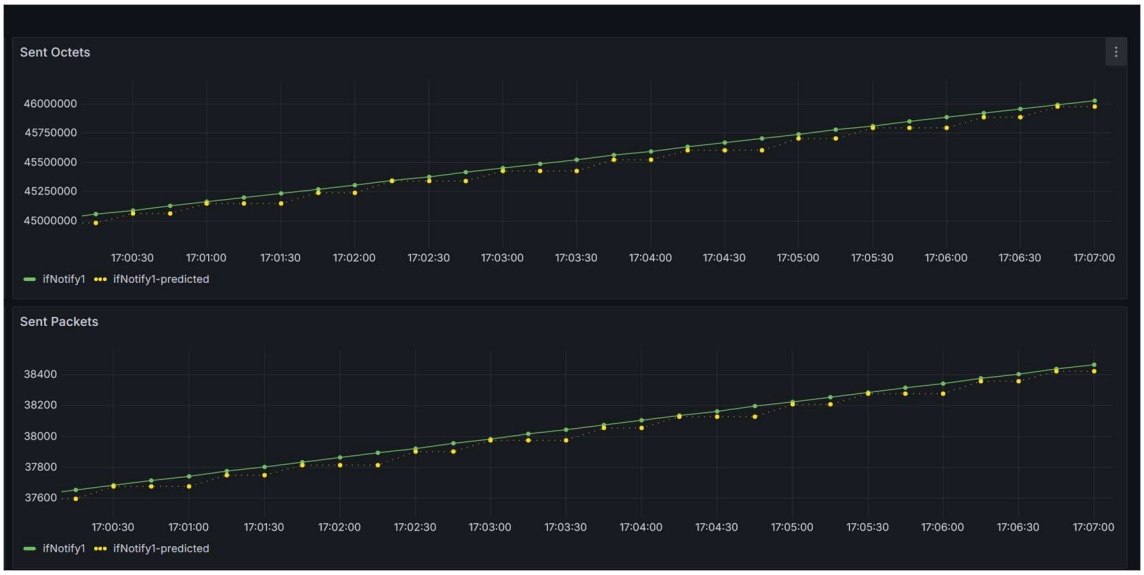
5.1.5 Előrejelzés

Az 5.25. ábra és az 5.26. ábra az előrejelzési dashboard felépítését mutatja be. A felületen a kiválasztott host összes hálózati interfésze megjelenik, lehetővé téve a valós idejű forgalmi adatok és a modell által generált előrejelzés összehasonlítását. A grafikonokon a zöld vonal a ténylegesen mért értékeket (küldött/fogadott csomagok és oktettek), míg a sárga pontok az előrejelzett adatpontokat jelölik, vizuális visszajelzést adva az ARIMA modell pontosságáról.

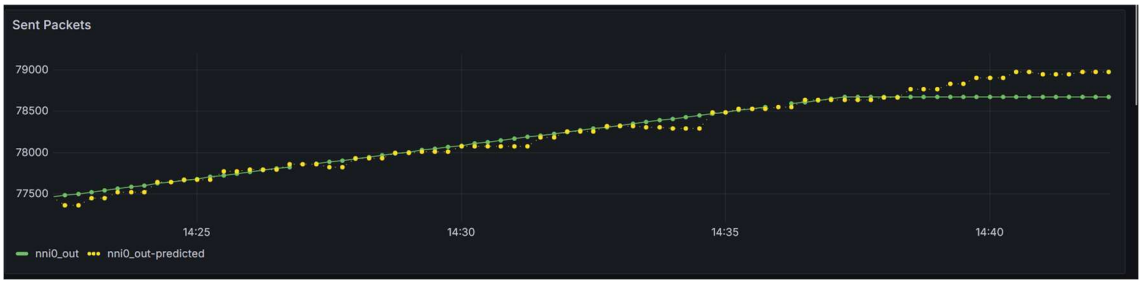


5.25. ábra: Fogadott oktett és csomag előrejelzése

A 5.27. ábra az előrejelzési modell működését szemlélteti egy konkrét interfész példáján keresztül. A grafikonon megfigyelhető, hogy a rendszer 5 perces időablakra készít előrejelzést (sárga pontok). Mivel a jövőbeli időpontokra vonatkozóan még nem áll rendelkezésre valós mérési adat, a tényleges forgalmat jelző zöld vonal az utolsó ismert értéknél konstanssá válik, míg az előrejelzés tovább folytatódik, előrevetítve a várható forgalmi tendenciát.



5.26. ábra: Küldött oktett és csomagok előrejelzése



5.27. ábra: Küldött csomagok előrejelzés

5.2 Előrejelzés pontossága

Az implementált előrejelző rendszer teljesítményének validálásához az éles működés közben illesztett modellek metaadatainak és pontossági mutatóinak elemzésére került sor. A vizsgálat célja annak meghatározása volt, hogy a választott ARIMA algoritmus mennyire képes adaptálódni a különböző hálózati forgalmi mintázatokhoz.

5.2.1 Modellkonfiguráció és paraméterek

A rendszer egységesen az ARIMA (AutoRegressive Integrated Moving Average) algoritmust alkalmazza, (2, 0, 2) hiperparaméter-beállítással. Ez a konfiguráció egy másodrendű autoregresszív és mozgóátlag komponenst jelöl differenciálás nélkül (d=0), így a modell a bemeneti idősorokat stacionáriusként kezeli, mivel a rendszer előfeldolgozása biztosítja ezt a tulajdonságot. Az illesztési ablak 180 percet ölel fel, amely alapján a rendszer 5 perces időtávra (horizon) készít előrejelzést.

5.2.2 Eredmények a forgalom dinamikájának függvényében

A modellek pontosságának értékelése az átlagos négyzetes hiba (RMSE – Root Mean Squared Error) mérőszám alapján történt. Az eredmények azt mutatják, hogy a rendszer teljesítménye szoros összefüggést mutat a vizsgált interfész forgalmi karakterisztikájával.

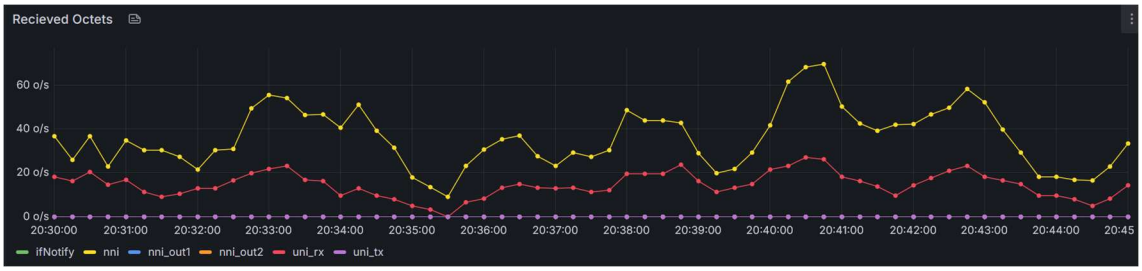
5.2.2.1 Stabil forgalmi környezet

Azokon az interfészeken, ahol az adatforgalom konstans vagy minimális ingadozást mutat, például forgalommentes időszakokban, a modellek kiváló, akár 0.0-s RMSE értéket érnek el. Az elemzés rávilágított, hogy ilyen esetekben az algoritmus adaptív módon a „bázis” -stratégiát részesíti előnyben. Ez azt jelenti, hogy a rendszer sikeresen felismeri a statikus viselkedést, és erőforrás-hatékony módon az utolsó mért értéket vetíti előre, gyakorlatilag hiba nélkül.

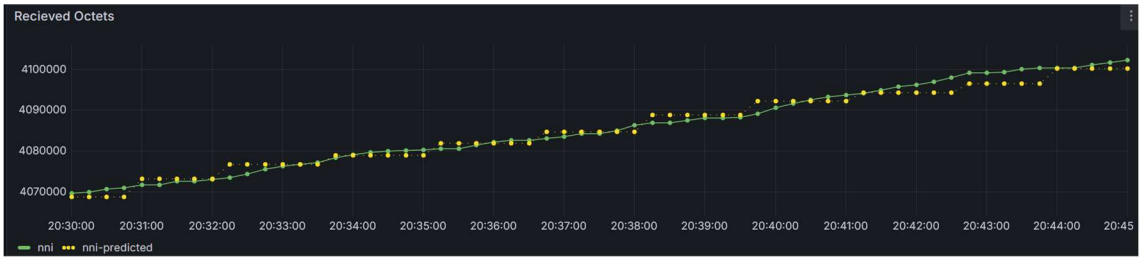
5.2.2.2 Dinamikus forgalmi környezet

A változó terhelésű, sztochasztikusabb adatfolyamok esetében, ahol tényleges, dinamikus forgalom mérhető változhat a modellek pontossága. A tesztkörnyezetben a hálózati interfészeken, NetEm segítségével szimulált hálózati hibák kerültek beállításra a Core csomópont interface-ein, amelyek 50%-os valószínűséggel váltanak „rossz” állapotba, ahol 100%-os a csomagvesztést, és 10% eséllyel váltanak „jó” állapotba, amelyben nincs veszteség.

A 5.28. ábra mutatja a veszteségi beállítások mellett a beérkező oktettek alakulását, a 5.29. ábra pedig az ehhez kapcsolódó predikciókat.



5.28. ábra: Szimulált fogadott oktettek



5.29. ábra: Előrejelzett fogadott oktettek

A modellek átlagos négyzetes hibája (RMSE) oktettek tekintetében 3415, míg csomagok esetében mindössze 6 körüli értéket mutatott. Ilyen körülmények között a rendszer automatikusan adaptálódik: csökkenti a bázis-stratégia súlyozását (jellemzően 0,75-re vagy akár 0,0-ra), és az előrejelzés során dominánsan az ARIMA algoritmus által kalkulált trendekre támaszkodik.

5.2.3 Összegzés

A kapott eredmények alapján megállapítható, hogy az 5 perces előrejelzési ablakban a ~6 csomagnyi átlagos hiba a hálózati forgalom nagyságrendjéhez viszonyítva elhanyagolható, így a modell alkalmas a tendenciák megbízható követésére. A rendszer sikeresen demonstrálta adaptivitását: statikus környezetben egyszerűsített heurisztikával, míg dinamikus terhelés mellett robusztus statisztikai modellezéssel biztosítja az előrejelzések pontosságát.

6 Összefoglalás

A szakdolgozat egy komplex hálózati felügyeleti és előrejelző rendszer tervezését és megvalósítását mutatja be, amely kifejezetten a FRER (Frame Replication and Elimination for Reliability) és PREOF (Packet Replication, Elimination and Ordering Functions) mechanizmusokat támogató hálózati eszközök felügyeletére készült.

6.1 A feladat célkitűzése

A dolgozat célja egy olyan integrált vizualizációs és analitikai keretrendszer létrehozása volt, amely képes a BME TMIT tanszék által használt, redundáns adatátvitelt megvalósító szoftver működését nyomon követni. A kritikus rendszerekben (pl. ipari automatizálás, járműipar) elengedhetetlen a megbízhatóság, ezért a rendszernek nemcsak megjelenítenie kell a valós idejű adatokat, hanem előre is kell jeleznie a potenciális hálózati problémákat.

6.2 Tervezés és technológiaválasztás

A rendszertervezés során a mikroszolgáltatás-alapú architektúra került kiválasztásra, amely biztosítja a modularitást és a skálázhatóságot. Az irodalomkutatás alapján a következő technológiák kerültek alkalmazásra:

- Adattárolás: A Prometheus idősor-adatbázis lett a választott megoldás a széleskörű támogatottsága és a feladathoz illeszkedő adatmodellje miatt.
- Vizualizáció: A Grafana került bevezetésre a rugalmas dashboard-kezelése és a Prometheus-szal való kiváló integrálhatósága miatt.
- Előrejelzés: A gépi tanulási módszerekkel szemben az ARIMA (Autoregressive Integrated Moving Average) statisztikai modell került kiválasztásra, mivel alacsony erőforrásigény mellett is megfelelő pontosságot és interpretálhatóságot biztosít a hálózati metrikák (csomagvesztés, forgalom) előrejelzésére.

6.3 Megvalósítás

A rendszer implementációja Docker konténerek és Mininet segítségével valósult meg, amely az alábbiakat foglalja magába:

- Tesztkörnyezet: Mininet alapú hálózati emuláció, amely a redundáns útvonalakat és a hálózati hibákat szimulálja.

- Adattárolás és megjelenítés: Docker konténerekben futó Prometheus és Grafana segítségével.
- Adatgyűjtés: Egy egyedi Python alapú UDP Receiver és Metrics Exporter modul, amely fogadja a hálózati csomópontok JSON üzeneteit, és a Prometheus számára feldolgozható formátumba konvertálja azokat.
- Előrejelző modul: Egy önálló szolgáltatás, amely periodikusan betanítja az ARIMA modelleket a múltbeli adatokon, és visszatölti az előrejelzéseket az adatbázisba.
- Topológia vizualizáció: Egy külön szolgáltatás, amely a Mininet szkriptek alapján grafikusan ábrázolja a hálózat felépítését.

6.4 Eredmények és elemzés

A projekt során öt fő dashboard készült el, amelyek különböző szinteken (áttekintő, csomópont-specifikus, interfész-összehasonlító, topológia, előrejelzés) nyújtanak betekintést a rendszer működésébe.

Az előrejelző rendszer teljesítményének elemzése azt mutatta, hogy az ARIMA modell sikeresen alkalmazkodik a hálózati viszonyokhoz:

- Stabil környezetben: Gyakorlatilag hiba nélkül (0.0 RMSE) működik.
- Dinamikus környezetben: A szimulált csomagvesztések és változó terhelés mellett is alacsony hibahatárral (átlagosan ~6 csomag eltérés) képes előrejelezni a tendenciákat 5 perces időtávon.

Összességében a szakdolgozatomban sikeresen demonstráltam, hogy a modern nyílt forráskódúeszközök és a klasszikus statisztikai módszerek kombinációjával hatékony felügyeleti rendszer építhető a jövő kritikus hálózati technológiái számára.

Irodalomjegyzék

[1] *„IEEE Standard for Local and metropolitan area networks--Frame Replication and Elimination for Reliability,”* in IEEE Std 802.1CB-2017 , vol., no., pp.1-102, 27 Oct. 2017, doi: 10.1109/IEEESTD.2017.8091139. (Elérés dátuma: 2025. november 9.)

[2] *Finn N., Thubert P., Varga B., Farkas J., “Deterministic Networking Architecture*”, RFC 8655, October 2019, Internet Engineering Task Force. URL: https://www.rfc-editor.org/info/rfc8655 (Elérés dátuma: 2025. november 9.)

[3] *Mininet: An instant virtual network on your laptop (or other PC*). from https://mininet.org/ (Elérés dátuma: 2025. november 15.)

[4] *A. A. Ariyo, A. O. Adewumi and C. K. Ayo, "Stock Price Prediction Using the ARIMA Model," 2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, Cambridge, UK, 2014, pp. 106-112, doi: 10.1109/UKSim.2014.67. , (Elérés dátuma: 2025. november 9.)

[5] *Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory*. Neural Comput. 9, 8 (November 15, 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
(Elérés dátuma: 2025. november 15.)

[6] *NetEm — Network Emulator. Linux man pages*. From: https://www.linux.org/docs/man8/tc-netem.html (Elérés dátuma: 2025. november 29.)

[7] *Microsoft. Windows Subsystem for Linux Documentation*. Microsoft Learn from https://learn.microsoft.com/en-us/windows/wsl/ (Elérés dátuma: 2025. november 15.)

[8] *Docker. What is a container?* from https://www.docker.com/resources/what-container/ (Elérés dátuma: 2025. november 15.)

[9] *„Overview | Prometheus,” in Prometheus Documentation — Introduction*, https://prometheus.io/docs/introduction/overview/ (Elérés dátuma: 2025. november 16.)

[10] *Cloud Native Computing Foundation*. From: https://www.cncf.io/ (Elérés dátuma: 2025. november 29.)

[11] *Kubernetes Documentation*. From: https://kubernetes.io/docs/home/ (Elérés dátuma: 2025. november 29.)

[12] *Thanos — Getting Started guide*. From: https://thanos.io/tip/thanos/getting-started.md/ (Elérés dátuma: 2025. november 29.)

[13] *„InfluxData Documentation,” InfluxData, Inc*., https://docs.influxdata.com/ (Elérés dátuma: 2025. november 16.)

[14] *„Welcome to VictoriaMetrics Docs,” VictoriaMetrics Documentation, VictoriaMetrics.,* https://docs.victoriametrics.com/ (Elérés dátuma: 2025. november 16.)

[15] *„Grafana OSS and Enterprise Documentation,” Grafana Documentation — Grafana Labs.* grafana.com/docs/grafana/latest/ (Elérés dátuma: 2025. november 16.)

[16] *„Getting Started with Kibana,” Elastic*. https://www.elastic.co/virtual-events/getting-started-kibana (Elérés dátuma: 2025. november 16.)

[17] *Docker Compose — Documentation*. From: https://docs.docker.com/compose/ (Elérés dátuma: 2025. november 29.)

[18] *Forráskód:* https://github.com/GriffManoue/networking-dashboard.git (Elérés dátuma: 2025. december 08.)

[19] *Git – the official Git website*. From https://git-scm.com/ (Elérés dátuma: 2025. november 17.)

[20] *client_python – Prometheus Python client library*. From: https://prometheus.github.io/client_python/ (Elérés dátuma: 2025. november 17.)

[21] *Flask Documentation*. From: https://flask.palletsprojects.com/en/stable/ (Elérés dátuma: 2025. november 29.)

[22] *Graphviz – open source graph visualization software*. From: https://graphviz.org/ (Elérés dátuma: 2025. november 17.)

Függelék

I. Függelék Nyilatkozat generatív mesterséges intelligencia alkalmazásáról

☐ **Nem használtam** semmilyen generatív MI segédeszközt.

☒ **Használtam** generatív MI segédeszközt. Az MI-vel generált tartalmakat ellenőriztem, a generált kimenetek valóságtartalmáról meggyőződtem, az alábbi táblázatban megfelelően jelöltem minden használatot.

<table>
<tr><th>Felhasználási módok</th><th>Generatív MI eszköz(ök) neve</th><th>Érintett részek (fejezet, oldalszám, hivatkozás)</th><th>Használat becsült aránya (felhasználási módonként)</th></tr>
<tr><td>Irodalomkutatás</td><td>ChatGPT 5</td><td>Irodalomjegyzék
62 – 63. oldal</td><td>5%</td></tr>
<tr><td>Prompt lényegi része</td><td colspan="3">Irodalomjegyzék formázása, prompt:
Formázd az alábbi forrásokat egy magyarnyelvű szakdolgozat irodalomjegyzékébe. Követsd ezt a formátumot:
- Az elérés dátumát zárójelben: (Elérés dátuma: YYYY. hónap nap.)
- A cím után szerző majd tárgy információk (év, szervezet, DOI, url stb.) végül elérés dátuma</td></tr>
<tr><td>Programkód generálása</td><td>Claude Sonnet 4.5</td><td>Előrejelző modul:
Adatok előfeldolgozása, Matematikai segédfunkciók az adatok skálázásához és az időbélyeg előállítása (2.6.2, 20.oldal, 4.2.3.2, 44. oldal)
Topológia beolvasása:
Szrktipt feldolgozásához használt regex kifejezések és hibakezelés (4.2.2, 43.oldal)</td><td>15%</td></tr>
<tr><td>Prompt lényegi része</td><td colspan="3">Adat előfeldolgozás:
Írj Python függvényt, amely egy idősort tisztít:
- Duplikátumok eltávolítása
- Üres értékek eltávolítása
- Kiugró értékek eltávolítása

Matematikai segédfunkciók:
Írj Python függvényt az idősor skálázása és transzformációjához:
- Stacionaritási teszt: ADF és KPSS teszteket végez
- Differenciálás: 1. vagy 2. rendű differenciálással sztatcionárissá alakít
- Differenciált értékeket visszakonvertálja eredeti skálára
- Időbélyeg számítás: az előrejelzés időpontját határozza meg</td></tr>
</table>

	Topológia beolvasás: Írj regex patterneket a Mininet scriptekhez: - Host: net.addHost('hostname', ...) - Link: net.addLink(src, dst, intfName1='eth0', intfName2='eth1') - IP: hostname.cmd ("ip a a 192.168.1.1/24 dev eth0") Topológia hibakezelés: Írj hibakezelési logikát a topológia parserhez: - File olvasási hibák kezelése - Duplikátumok eltávolítása - Hiányzó vagy üres értékek fallback kezelése - Link validáció (szereplő hosts meglétének ellenőrzése)		
Új ötletek, megoldási javaslatok generálása	-	-	-
Prompt lényegi része	-		
Vázlat létrehozása (szövegstruktúra, vázlatpontok)	-	-	-
Prompt lényegi része	-		
Szövegblokkok létrehozása	Gemini 3 Pro	Összefoglaló/Abstract: Magyar összefoglalót angolra fordítani (7.oldal) Bevetetés: Későbbi fejezetek tartalmi bemutatását megfogalmazni (1.4, 10.oldal) Megvalósítás: Az endpointok összegzése (4.2.1, 42.oldal, 4.2.2, 43.oldal) Összefoglalás: Összefoglalás megfogalmazása (6, 59-60.oldal)	10%
Prompt lényegi része	Összefoglaló/Abstract: Fordítsd angolra az alábbi magyar összefoglalót, megtartva a szakmai terminológiát és a szerkezetet. Az angol szöveg legyen akadémikus stílusú és pontosan tükrözze az eredeti tartalmat Bevezetés: Fogalmazd meg a bevezetésben a szakdolgozat fejezeteinek összefoglalását. Sorolj fel röviden, hogy az egyes fejezetek mit tartalmaznak, milyen kérdéseket válaszolnak meg, és hogyan kapcsolódnak egymáshoz Megvalósítás: Összegezd az alkalmazás API végpontjait strukturált formában. Csoportosítsd őket funkciók szerint, és röviden írd le minden csoport célját és főbb végpontjait Összefoglalás: Fogalmazd meg az összefoglalást a szakdolgozat főbb eredményeiről. Mutasd be röviden:		

	- Az implementált megoldások fő jellemzőit - Az elért teljesítményt/pontosságot - Az alkalmazott technológiákat - Lehetséges továbbfejlesztési irányokat		
Képek generálása illusztrációs célból	-	-	-
Prompt lényegi része	-		
Adatvizualizáció, grafikonok generálása adatpontok alapján	-	-	-
Prompt lényegi része	-		
Prezentáció készítése	-	-	-
Prompt lényegi része	-		
Egyéb (nevezze meg)	-	-	-
Prompt lényegi része	-		
Összesített százalékos érték (a feladat érdemi részére nézve)			**10%**

Összesített érték rövid, szöveges indoklása:
A generatív MI által készített tartalmak a szakdolgozat feladatainak érdemi részét kis mértékben érintették.

A szakirodalmi áttekintés saját munka, a ChatGPT 5 kizárólag a hivatkozások megfelelő formátumra hozását végezte. A teljes irodalomkutatáshoz viszonyított becsült felhasználási arány 5%.

A feladat megoldása során az adatok előfeldolgozásához, matematikai segédfunkciók (skálázás, időbélyeg-kezelés) megírására, valamint a topológia beolvasását végző szkriptek (regex illesztés, hibakezelés) elkészítésére került sor generatív módon a Claude Sonnet 4.5 modell segítségével. A rendszer központi logikáját és az előrejelző modellek architektúráját nem érintette. A generált kód aránya a teljes kódbázishoz képest 15%.

A generált szövegblokkok az absztrakt angolra fordításában, a bevezető fejezet szerkezeti felépítésében, valamint a megvalósítás és az eredmények összefoglaló részeinek megfogalmazásában játszottak szerepet a Gemini 3 Pro modell segítségével. A szakmai tartalom, az elemzések és a következtetések levonása saját eredmény. A generált szövegrészek aránya a teljes dolgozathoz viszonyítva 10%.

Összességében a mesterséges intelligencia a formai követelmények teljesítését, a kódolási részfeladatok gyorsítását és a szövegezés gördülékenységét segítette, a szakdolgozat érdemi, szakmai hozzáadott értékét nem helyettesítette. Mindezek alapján a generatív MI által készített tartalom becsült összesített aránya a feladat érdemi részére nézve: 10%.