**Budapesti Műszaki és Gazdaságtudományi Egyetem**
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

# Tudományos Diákköri Konferencia dolgozat

Maller Levente Márk

# Járműkommunikációs felhőtechnológiák modellezése és vizsgálata Cloud-in-the-Loop szimulációs környezetben

KONZULENSEK

Dr. Bokor László – BME-HIT

Suskovics Péter – Ericsson

BUDAPEST, 2021

# Tartalomjegyzék

# Összefoglaló

Manapság már számos jármű nyújt intelligens szolgáltatásokat, például képesek környezetérzékelésére (pl. objektumérzékelés és -felismerés), továbbá a gyűjtött információ feldolgozásával magas hozzáadott értékű, vezetést segítő alkalmazások megvalósítására (pl. ütközéselkerülés). Az okos és később az önvezető járműtechnológiákat azonban a járműkommunikációs (V2X) megoldások teszik teljessé. A járművek által gyűjtött információ ezáltal kiterjeszthetővé válik a többi, kapcsolódó forgalmi résztvevő számára. Ezzel lehetőség nyílik olyan alkalmazások használatára, mint a kooperatív koordináció járművek között vagy később a teljes és optimális forgalom-menedzsment. A V2X egyik ígéretes iránya a Cellular Vehicle-to-Everything (C-V2X) [1] kommunikáció, amelynek alapja, hogy a járművek a mobilhálózatok segítségével kapcsolódnak és kommunikálnak egymással. A koncepció megvalósítására az 5. generációs celluláris mobilhálózatok (5G) adhatnak megoldást, amelyek a későbbiekben képesek lesznek ultra-alacsony késleltetést és ultra-magas megbízhatóságot (URLLC) biztosítani és ezt ötvözni a küldetés-kritikus működéssel a kapcsolódó eszközök számára, kielégítve a szigorú járműipari követelményeket. Továbbá az 5G képes lesz elosztott felhő alapú (Edge Cloud) rendszerek hálózati integrációjára [2] is. Az ilyen rendszerek lényege, hogy a felhő erőforrások fizikailag elosztott módon, a felhasználókhoz közel helyezkednek el. A két technológia együttes alkalmazása nagymértékben felgyorsítja az információfeldolgozást, minimalizálva a késleltetést. Ezáltal olyan használati esetek válhatnak elérhetővé, mint a valósidejű HAD térkép rekonstrukció [3], amely folyamatos és dinamikus térképhasználatot biztosít a jövő autonóm járművei számára.

Ilyen új technológiákra épülő koncepciók és a kapcsolódó használati esetek vizsgálata és validálása, annak komplexitása réven sok kihívással jár, és valós hardverek és járművek alkalmazása esetén igen költséges és bonyolult folyamat. Erre nyújt megoldást a Cloud-in-the-Loop (CiL) keretrendszer [4], amelynek segítségével a vizsgálandó járműforgalom szimulálható, lehetőséget teremtve különféle használati esetek implementálására és a feldolgozott információ alapján egy szorosan integrált, valós elosztott felhő környezet vezérlésére. Az Edge Cloud-ot alkalmazó C-V2X használati esetek követelményeinek (pl.: késleltetés) teljesítése érdekében, a járművek

azokhoz az elosztott erőforrásokhoz (Edge szerver) kapcsolódnak, melyek jobb elérést biztosítanak. Ezekben a rendszerekben, az Edge szervereken, a szolgáltatásokat nyújtó alkalmazás komponensek futnak, amelyek a járműveken futó komponensekkel kommunikálnak, ennek megfelelően az alkalmazások relokációját biztosítani kell a járművek pozíciójának függvényében. A dolgozatom fókuszában ezen alkalmazások által az elosztott környezetre kifejtett hatások vizsgálata és teljesítmény-analízise áll. Az alkalmazások a járműszámnak megfelelően skálázódnak, ezáltal az alkalmazásszintű hálózati forgalom is megnövekszik. Ennek szignifikáns hatása van a hálózat QoS paramétereire. A célom, hogy a CiL rendszeren tervezett és kivitelezett mérések során, különféle forgalmi szituációkat modellezve ezeket a hatásokat vizsgáljam és értékeljem. Az alkalmazás skálázódása hatással van az elosztott felhő környezetet megvalósító Kubernetes [5] architektúra teljesítményére is. Ezáltal a felhőerőforrások terheltségének és az alkalmazásszintű QoS degradáció korrelációjának vizsgálata is kiemelet szerepet kap a dolgozatban. Továbbá célom, hogy a felhő környezet olyan lehetőségeit és funkcióit elemezzem, amelyek kritikusak lehetnek C-V2X alkalmazásoknál.

# Abstract

Nowadays, various vehicles provide intelligent services, such as discovering the environment (e.g., object detection and recognition) and implementing high value-added driving applications (e.g., collision avoidance) by processing the collected information. However, innovative and later self-driving vehicle technologies are completed by vehicular communication (V2X) solutions. This allows the data collected by the vehicles to be extended to other connected road users, enabling the use of applications such as cooperative coordination between vehicles or later complete and optimal traffic management. One promising direction of V2X is the Cellular Vehicle-to-Everything (C-V2X) [1] communication, which is based on vehicles connecting and communicating with each other using mobile cellular networks. To implement this concept, 5th generation cellular mobile networks (5G) can provide a solution that will later be able to provide ultra-low latency and ultra-high reliability (URLLC) and combine this with a mission-critical operation for the connected devices, satisfying the strict requirements of the automotive industry. In addition, the integration of distributed cloud-based (Edge Cloud) systems will also be possible with 5G networks [2]. The essence of such systems is that cloud resources are physically distributed and located close to the users. The combined use of the aforementioned technologies greatly speeds up information processing, thus minimizing latency. This will make advanced use cases available such as real-time HAD map reconstruction [3], which provides continuous and dynamic map use for the autonomous vehicles of the future.

Investigating and validating concepts based on such new technologies and the relating use cases, due to their complexity, involves many challenges and is a very costly and complicated process in the case of using real hardware and vehicles. The Cloud-in-the-Loop (CiL) simulation framework [4] provides a solution to this problem, which allows the vehicle traffic to be simulated, allowing the implementation of different use cases and the control of a closely integrated, real-life distributed cloud environment based on the processed information. To meet the requirements of C-V2X use cases (e.g., latency) using Edge Cloud technology, vehicles connect to the Edge server (distributed resource) that provides better access (e.g., better network conditions). In these systems, the Edge servers run the service-providing application components,

which communicate with the components running on the vehicles; consequently, the relocation of the applications must be ensured depending on the position of the vehicles. My TDK work focuses on investigating and analyzing the effects of these applications on the distributed environment. The applications are scaled according to the number of vehicles, which increases application-level network traffic. This has a significant impact on the QoS parameters of the network. I aim to investigate and evaluate these impacts by modeling various traffic scenarios during predefined measurements executed on the CiL framework. Application scaling also affects the performance of the Kubernetes [5], the architecture that implements the distributed cloud environment. Therefore, the correlation between the load on cloud resources and the application-level QoS degradation also plays a crucial role in my work. Furthermore, my goal is to analyze the capabilities and features of the cloud environment that may be critical for C-V2X applications.

# 1 Bevezetés

## 1.1 Motivációk és áttekintés

Napjainkban a járműkommunikációs (V2X) megoldások egyre gyakrabban kerülnek alkalmazásba, számos manapság is kapható jármű képes már V2V és V2I közvetlen kommunikáció megvalósítására. Az elsődleges balesetmegelőzési funkciók mellett számos új használati eset lát napvilágot, amelyek kiszolgálására egyre komplexebb és erősebb rendszerek válnak szükségessé. A telekommunikációs hálózatok fejlődésével és az 5G megjelenésével a járműkommunikációs modellek is fejlődtek, megszületett a V2X kommunikáció egyik legígéretesebb iránya a C-V2X (celluláris járműkommunikáció), amely már 3GPP szabványrendszer szerves része. Az 5G rengeteg újítást hozott celluláris mobilhálózat működésébe, a minden generációváltásnál megszokott adatátviteli sebességek növekedése mellett olyan funkciók és szolgáltatások jelentek meg, mint az ultra-alacsony késleltetésű és ultra-magas megbízhatóságú kapcsolat vagy a hálózati virtualizációs megoldások. Ezek mind olyan szolgáltatások, amelyek nagymértékben elősegítik a C-V2X kommunikációs modell megvalósítását, ezek kapcsán már kész követelményrendszereket is definiáltak, adott use case csoportoknak megfelelően. Az 5G-s hálózatok azonban olyan technológiák integrációját is lehetővé fogják tenni, mint az elosztott felhő alapú (Edge Cloud) rendszerek. Az Edge Cloud lényege, hogy a felhős erőforrások fizikailag elosztott módon, közel a felhasználókhoz helyezkednek el. Kiegészülve az 5G-s hálózatokkal az információfeldolgozás nagymértékben fel fog gyorsulni. Ezáltal számos új használati esetet és olyan területeket teremtve, mint az Edge Cloud alapú C-V2X járműkommunikáció. Azonban az elosztott felhő alapú rendszerek működtetése manapság már konténerizáció alapú, Cloud Native megoldások alapjain nyugszik. Konténerizált alkalmazások egy nagykiterjedésű, elosztott hálózaton való vezérlésére és orkesztrációjára a Kubernetes platform nyújt megoldást. Ebből kifolyólag az 5G hálózatok Edge Cloud integrációjára is ígéretes irány a Kubernetes alkalmazása, amely a C-V2X járműkommunikációs alkalmazások kiszolgálását is biztosíthatja. Ennek a motivációja indította el a munkát, amelynek az eredménye a jelen dolgozathoz is felhasznált Cloud-in-the-Loop szimulációs keretrendszer.

## 1.2 Előzmények és kitűzött célok

A dolgozatban alkalmazott Cloud-in-the-Loop szimulációs keretrendszer a BSc szakdolgozatom során végzett munka eredménye. A keretrendszer lényege, hogy egy SUMO nevű forgalom szimulátor alapján képes valós forgalmi térképek és C-V2X járművek szimulációjára és ezeket az információkat felhasználva egy szorosan integrált, valós, felhő alapú elosztott rendszer vezérlésére. Az elosztott rendszert egy Raspberry Pi eszközökre épülő Kubernetes cluster valósítja meg. A keretrendszer segítségével ezáltal adott járműipari use case-ek implementációjával azok átfogó vizsgálata lehetséges. A CiL-Simulator nagy előnye, hogy valós járművek és mobilhálózati eszközök nélkül vizsgálhatóak a különféle C-V2X használati esetek és maga az elosztott rendszer, ezáltal kiváltva a sokszor költséges valós tesztmódszereket. Az Edge Cloud alapú C-V2X járműkommunikáció esetében a szolgáltatásokat elosztott alkalmazások biztosítják. A járműveken futó alkalmazáskomponensek az Edge szervereken futó, távoli komponensekkel kommunikálnak a hálózaton keresztül. Ezáltal, hogy az adott use case-ek által előírt követelményeket teljesüljenek, a járműnek minden esetben ahhoz az Edge szerverhez kell kapcsolódnia, amely a legjobb hozzáférést és hálózati kapcsolatot biztosítja. Ennek az a következménye, hogy a távoli alkalmazáskomponenst minden esetben az adott járműhöz kapcsolódó Edge szerverre kell áthelyezni, amelynek szignifikáns hatása lehet a szolgáltatás QoS paramétereire (pl.: késleltetés). Ezért a BSc szakdolgozatom elkészítése során a keretrendszer segítségével korábban alkalmazás áthelyezési stratégiákat vizsgáltam az Edge szervereket reprezentáló Kubernetes node-ok között. A mérések során az alkalmazás létrehozási idők útján vizsgáltam a relokációs késleltetéseket. Továbbá, a Kubernetes cluster-t terheléses vizsgálatoknak is alávetettem.

A jelen dolgozatban a korábban elkészült keretrendszert felhasználva új megközelítésű méréseket fogok bemutatni. Járműipari felhasználásban, a szolgáltatásokat nyújtó alkalmazásoknak szigorú követelményeket teljesítő hálózati szolgáltatási minőséget kell biztosítania. Ezért azok alkalmazásszintű hálózati forgalmát jellemző QoS paramétereinek vizsgálata kulcsfontosságú kérdés. Ezeket a paramétereket számos hatás befolyásolhatja, többek között az alkalmazásokat futtató architektúra teljesítménye vagy az hálózati terheltség. Ezeknek a szempontoknak a vizsgálatára a felhő alapú elosztott hálózat működését, egy adott, implementált use case segítségével alkalmazásszintű hálózati forgalomgenerálással fogom vizsgálni. A

szimuláció során a járműveket kiszolgáló alkalmazások a növekvő járműszámnak megfelelően skálázódnak, az alkalmazások által generált adatforgalom mellett, a Kubernetes erőforrásainak terheltsége is növekszik. Ezek szignifikáns hatással lehetnek az alkalmazás relokációs műveletekre is. Ezen hatások elemzését adott alkalmazások hálózati forgalmának QoS degradációjának vizsgálatával fogom megvalósítani. A különböző tényezők hatásainak vizsgálata érdekében a méréseket különféle forgalmi szituációk modellezésével és adatsebességek konfigurálásával fogom lefolytatni. A kapott eredmények elemzésével a célom, hogy átfogóbb képet kapjak a Kubernetes platform járműipari környezetben való működéséről, továbbá, hogy elemezzem, hogy az milyen hatást gyakorolt az alkalmazások működésére.

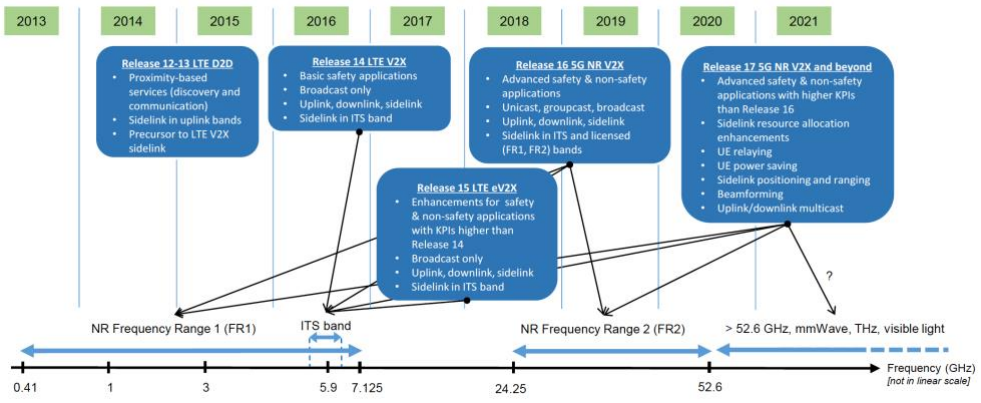
# 2 Cellular Vehicle-to-Everything (C-V2X) technológiák

A Vehicle-to-Everything (V2X) járműkommunikáció alapvető célja, hogy a járművek által gyűjtött információt (pl.: szenzoradatok, kameraképek, veszélyt jelző figyelmeztetések) megossza a többi forgalmi résztvevővel. Az adatokat felhasználva számos autóipari használati eset (use case) valósítható meg. Ezek segítségével megelőzhetőek a balesetek, elősegíthető a közlekedés dinamikájának biztonságos növekedése, és a megfelelő használati esetek és technológiák együttes használata mellett teljes forgalom-menedzsment is megvalósítható. A későbbiekben ez olyan technológiáknak fekteti le az alapot, mint a teljes autonóm vezetés. A V2X kommunikáció - ahogy az elnevezése is mutatja - a járművek és bármely más forgalmi résztvevő kommunikációját összefoglaló fogalom, amely négy fő kategóriába sorolható be [6]. Ezek közül a Vehicle-to-Vehicle (V2V) kommunikáció a járművek közötti információcserét megvalósító megoldásokat foglalja magába, amelyek egy része már több korszerű járműben is megtalálható [7]. A Vehicle-to-Infrastructure (V2I) kommunikáció az út menti, illetve okos városokban megtalálható infrastruktúra és a járművek közötti információcserét jelenti. Jelenleg legelterjedtebben a V2V és V2I fizikai és hozzáférési rétegekben az IEEE 802.11p szabvány [8] által definiált vezeték nélküli technológiát alkalmazza. A Vehicle-to-Pedestrian kommunikáció segítségével a járművek a gyalogosokkal képesek adatmegosztásra. Továbbá a Vehicle-to-Network (V2N) kommunikáció a járművek és a hálózat közötti kommunikációs formákat foglalja magába. A V2N egyik nagy előnye, hogy a hálózat segítségével közvetetten képes a V2V és V2I megvalósítására is. A járművek nagyfokú mobilitásából adódóan a V2N megvalósítására kézenfekvő a celluláris mobilhálózatok felhasználása, vagyis a C-V2X kommunikáció alkalmazása. A mobilhálózatok segítségével a kapcsolódó eszközök (járművek, infrastruktúra) folyamatos összeköttetésben állhatnak, továbbá azok internetes elérése is biztosított. A C-V2X kommunikációra építkezve számos új autóipari használati eset alkalmazására nyílik lehetőség, amelynek megvalósítására a direkt kommunikáció során megjelenő korlátok miatt nincs lehetőség. Ezekhez olyan funkciók szükségesek, mint például a távoli kapcsolat, nagy sűrűségű eszközkiszolgálás vagy nagy adatsebesség. Továbbá számos manapság már használatban lévő use case is

implementálható a C-V2X alkalmazásával. Az egyik legnagyobb kihívás, hogy ezek a rendszerek teljesítsék azokat a szigorú követelményeket, amelyeket az autóipari használati esetek megkövetelnek, különösen az alacsony késleltetést és nagy megbízhatóságot. Ebben kiemelt szerepe van a megfelelő rádiós hozzáférési technológia megválasztásának, továbbá azoknak a megoldásoknak és funkcióknak, melyek az elmúlt években megjelentek.

## 2.1 C-V2X rádiós hozzáférés fejlődése

A 3rd Generation Partnership Project (3GPP) [9], a mobilkommunikációs protokollok kidolgozásáért felelős szabványosító szervezet az utóbbi években aktívan foglalkozik a C-V2X kommunikációval. A legutóbbi szabvány kiadások mind tartalmaznak olyan mobilkommunikációs hálózatot érintő újításokat, amelyek a celluláris járműkommunikáció alapját biztosítják. A 3GPP a 14-es és 15-ös kiadásaiban (Release 14/15) eleinte az LTE rendszerekre dolgozta ki a szabványt (LTE V2X), de a generáció váltás során megjelent 16-os kiadás már az újonnan kifejlesztett 5G NR rádiós interfészre építkező C-V2X szabványt tartalmazza (2.1. ábra). Ma már az 5G-re épülő megoldások élveznek elsőbbséget, annak köszönhetően, hogy olyan teljesítményt és funkciókat nyújtanak, amelyek hatékonyabbá és megbízhatóbbá teszik a járműkommunikációt.




**2.1. ábra V2X újítások a rádiós hozzáférés vonatkozásában, a 3GPP kiadásaiban [10]**

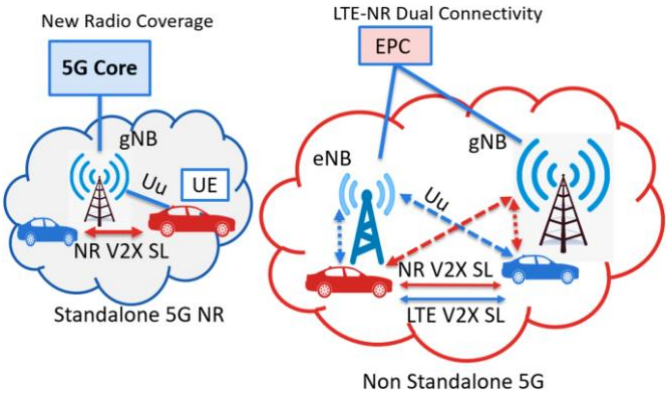
### 2.1.1 LTE V2X (3GPP Release 14/15) [10]

A 3GPP Release 14-ben megjelenő, majd Release 15-ben továbbfejlesztett LTE V2X szabványa az LTE rádiós interfészt alkalmazó járműkommunikációt definiálja. Az LTE V2X számára 5,9 GHz-es, ITS frekvenciasávot [11] írták elő működési

tartománynak. A járművek a hálózattal való kommunikációt megvalósító Uu interfész mellett, egy PC5 interfésszel is rendelkeznének, amellyel hálózat által támogatott direkt kommunikációra (sidelink) is képesek egymással. A szabványt úgy alakították ki, hogy a ráépülő rendszerek támogassák az alapvető járműipari használati eseteket. Ezek kiterjednek a forgalmi biztonságot támogató, forgalom menedzsment és információtechnikai alkalmazásokra. Az LTE V2X továbbá hasonló üzenetszórási technikákat is alkalmaz, amelyet ITS-G5 a CAM-mel (Cooperative Awareness Messages), vagy DSRC a BSM-mel (Basic Safety Messages) [12]. A módszer lényege, hogy a járművek periodikusan alapvető információkat osztanak meg magukról a többi kompatibilis eszközzel, például sebesség, irány vagy lokációs adatokat.

### 2.1.2 NR V2X

A 5. generációs celluláris mobilhálózatok eredményeképpen a C-V2X kommunikáció is fejlődött. A 3GPP 16-os kiadásában az LTE V2X szabványt kiegészítve, illetve arra építkezve definiálták az (5G) NR V2X járműkommunikációs modellt. A korábbi kiadásokban szabványosított interfészek (NR Uu, PC5) továbbfejlesztetett változata itt is megjelenik, amelyek kompatibilitása biztosított (2.2. ábra) a korábbi szabványban meghatározott megoldásokkal [10]. A sidelink (SL) kapcsolat működésében 2 típust (mode) különböztetnek meg, az elsőben (mode 1) a bázisállomás (BS) ütemezi az SL erőforrásokat felhasználói eszközök (UE) számára, a másodikban (mode 2) az UE-k ütemezik az átviteli erőforrásokat, a BS által meghatározott erőforrásokból. A Uu interfész legnagyobb újítása, hogy az LTE-ben definiált nagy megbízhatóságú, alacsony késleltetésű kommunikáció (HRLLC) továbbfejlesztéseként megjelenik az ultra-magas megbízhatóságú, alacsony késleltetésű kommunikáció (URLLC). Az 5G működési frekvenciatartománya 5 GHz környéke mellett milliméter hullámhosszú (mmW), 30 GHz - 300 GHz frekvenciatartományokra is kiterjed, ahol akár több 10 Gbps-os átvitelisebességek is elérhetőek. Az 5G hatékonyságának növelése érdekében több startégiát is alkalmaznak, ilyen a spektrum kihasználtság növelése, a meglévő és az új frekvenciasávokban egyaránt. Továbbá az erőforrások újrafelhasználása kisebb cellák létrehozásával, ezáltal lehetőséget teremtve a frekvencia újrafelhasználásra. Végül a spektrális hatékonyság növelése fejlett kódolástechnikával és modulációval, többcsatornás interferencia kezeléssel és továbbfejlesztett antennákkal. [7]

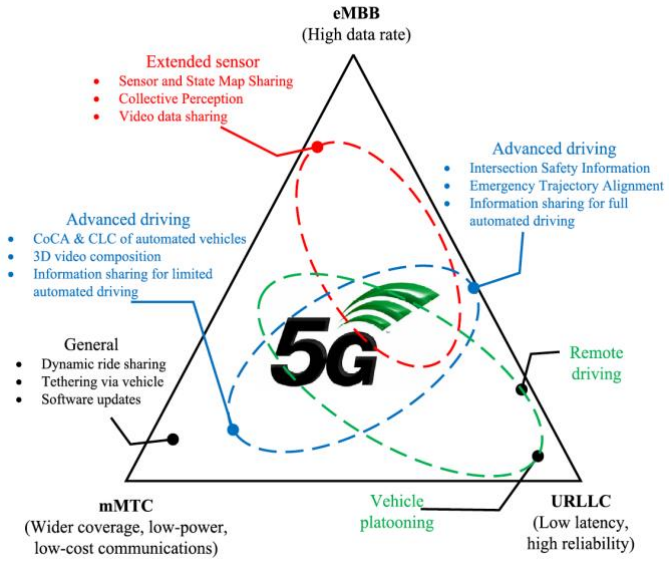



**2.2. ábra Az NR és LTE Uu és PC5 interfészek kapcsolata [14]**

A 5G által hozott újítások megteremtették a szolgáltatásnyújtás három kategóriáját [13] (2.3. ábra), amelyet a 3GPP és az ITU [15] is alkalmaz:

- enhanced Mobile Broadband (eMBB): Az eMBB szolgáltatások célja a nagy adatsebességű átvitel biztosítása, adott lefedési területen, olyan módon, hogy a csomag hibaarány $10^{-3}$ alatt maradjon. Ez olyan szolgáltatások használatára ad lehetőséget, mint a virtuális valóság (VR), vagy Ultra HD minőségű videófolyamok átvitele elhanyagolható hibaaránnyal. V2X tekintetében olyan használati esetekben lehet hasznos az eMBB, mint a multimédiás szolgáltatások, továbbá a járművek által gyűjtött szenzoradatok továbbítása (pl.: HAD térkép rekonstrukció).
- Ultra-Reliable Low Latency Communication (URLLC): Az URLLC szolgáltatások fő célja az olyan alkalmazások kiszolgálása, amelyeknél kiemelten fontos a késleltetésre vonatkozó követelmények kielégítése, nagyon nagy megbízhatósággal, kis hibaaránnyal ($10^{-5}$ alatti csomag hibaarány). Ezek az alkalmazások olyan V2X használati eseteket foglalnak magukban, mint például a távoli vezetés vagy a fejlett vezetést segítő alkalmazások, ahol olyan követelményeket kell teljesíteni, mint az 1 ms körüli késleltetés és 99,999 %-os megbízhatóság. A későbbiekben továbbá elengedhetetlen lesz az összekapcsolt autonóm vezetés megvalósítására is.
- massive Machine Type Communications (mMTC): Az mMTC szolgáltatások lényege, hogy hogy az 5G hálózatokkal nagy sűrűségű eszköz is kiszolgálható. Mindezt alacsony energiafelhasználású, alacsony komplexitású módon, kis átviteli sebesség és $10^{-5}$ alatti csomag hibaarány mellett. A V2X alkalmazások területén ez főleg olyan helyzetekben hasznos,

amikor adott, kis területen nagy mennyiségű jármű vagy útmenti eszköz (Road Side Unit) tartózkodik, és azok folyamatosan adatokat (pl.: szenzoradatok) osztanak meg a környezetükről. Fontos, hogy ezen a területen nem biztosítható az alacsony késleltetés, ezért késleltetés toleráns alkalmazásokra szükséges tervezni mMTC alkalmazásakor.




**2.3. ábra Az 5G hálózat alap szolgáltatástípusai és a hozzájuk tartozó V2X use case-ek [13]**

Az 5G által hozott rádiós hozzáférésben való fejlődés, továbbá a járművekben alkalmazott fejlettebb szenzorok komplexebb use case-ek megvalósításának lehetőségét teremtették meg. Ezáltal a 3GPP előrelátóan a 15-ös kiadásában négy fő use case csoportot határozott meg [6][9], amelyek különböző követelményeket is követelnek meg a celluláris hálózattól.

1. Konvojban haladás (Vehicle Platooning): Ez a kategória olyan use case-eket foglal magában, amelyeknek a lényege, hogy a járművek képesek koordináltan, csoportban haladni. Ilyenkor a járművek periodikusan osztanak meg információt a helyzetükről és mozgásukról, amellyel biztosítható az összehangolt haladás fenntartása.
2. Fejlett, vezetést segítő megoldások (Advanced Driving): Ebbe a csoportba tartozik minden olyan use case, amelyek a vezetés automatizálását segítik, illetve a későbbi teljes autonóm vezetést megvalósító megoldások. Technikailag ezek a használati esetek a járművek által gyűjtött információk (környezeti tényezők és objektumok) feldolgozására és - ami a hálózat

szempontjából fontos - azok megosztására építkeznek. Továbbá ezek az adatok kiegészülnek a járművek által továbbított helyváltoztatási szándékot jelző információkkal.

3. Kiterjesztett szenzorok (Extended Sensors): Ahogy a csoport elnevezése is mutatja, az ebben található use case-ek feladata, hogy a járművek és a további V2X eszközök által gyűjtött szenzor- és kameraadatot, vagy egyéb információt a rendszer megossza a többi kompatibilis eszközzel. Így, a járművek érzékelése nagymértékben kiterjeszthető, ezáltal olyan információk feldolgozására is lehetőséget teremtve, amelyet a jármű alapvetően nem lenne képes érzékelni.
4. Távoli vezetés (Remote Driving): Minden olyan use case-t ebbe a kategóriába tartozik, amely segítségével a járművek egy távoli operátor beavatkozásával, valós időben irányíthatók.

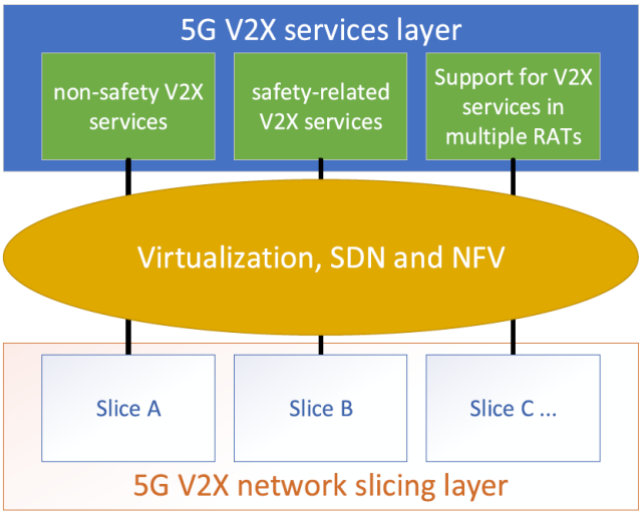
A különböző use case csoportok, különféle QoS követelményekkel is rendelkeznek (2.1. táblázat). Jól látható, hogy a jövőben az 5G-ben biztosítható három kategóriába besorolt szolgáltatások (eMBB, URLLC, mMTC) hivatottak ezeket követelményeket kiszolgálni.

| Use case csoport | Csomagméret/payload [Byte] | Üzenet/sec | Adatsebesség (Mbps) | Késleltetetés (ms) | Megbízhatóság (%) |
|---|---|---|---|---|---|
| Vehicle Platooning | 50-6000 | 2-50 | 50-65 | 10-500 | 90-99,99 |
| Advanced Driving | 300-12000 | 10-100 | 10-53 | 3-100 | 90-99,999 |
| Extended Sensors | 1600 | 10 | 10-1000 | 3-100 | 90-99,999 |
| Remote Driving | - | - | UL:25<br>DL:15 | 5 | 99,999 |

**2.1. táblázat 3GPP use case csoportok és követelményeik [16]**

## 2.2 NFV/SDN

Az 5G hálózatok egyik legnagyobb technológiai újítása a Network Function Virtualization (NFV) bevezetése. Lényege, hogy a hálózat fizikai eszközei elkülöníthetővé válnak azok funkcióitól. Ezáltal a fizikai hálózat felett számos másik virtuális hálózat is létrehozható, elkülönítve azok funkcióit, kontroll síkját (control plane) és adat síkjait (data plane) egymástól. Így, felhasználási szinten úgy működnek a virtuális hálózatok, mintha különálló fizikai hálózatokat alkalmaznánk [17]. A Software-Defined Networking (SDN) hasonlóan az NFV-hez, hálózati absztrakciós eljárás. Az SDN segítségével a hálózati alsóbb rétegei ismerete és konfigurálása nélkül teszi menedzselhetővé a hálózatot. Mindezt a hálózati eszközök (router, switch) virtualizációjával valósítja meg, ezáltal a döntéshozás a virtuális hálózat kontroll síkján valósul meg, míg az adatok fizikai továbbítását a hardver valósítja meg. Ennek eredményeképpen a hálózat konfigurációja és menedzsmentje sokkal flexibilisebbé és dinamikusabbá válik. Az NFV és SDN és együttes használata lehetővé teszi az 5G rendszerek hálózati felosztását. Ezáltal a két módszer együttes használata során a NFV elkülöníti a hálózat funkcióit, majd az SDN menedzseli azokat különféle felhasználási eseteknek megfelelően [18]. Ezekre a virtuális hálózatokra különféle követelményeknek megfelelő szolgáltatásokat lehet definiálni a felhasználási területnek megfelelően (5G Network Slicing). A V2X alkalmazásoknál is nagy előnyt jelent, mivel a V2X szolgáltatási területeknek megfelelően felosztható a hálózat [13] (2.4. ábra). Az NFV és SDN együttes használata mellett a hálózat terhelése optimalizálható, ezáltal javítható a megbízhatóság és csökkenthető a késleltetés.




**2.4. ábra Példa az 5G V2X hálózati felosztásra [13]**

## 2.3 Multi-access Edge Computing (MEC)

A V2X use case-ek fejlődésével és komplexitásuk növekedésével, egyre nagyobb az igény a nagyobb számításigényű adatfeldolgozásra. A járművek fedélzetén elhelyezkedő hardverek számítási kapacitása azonban sok használati eset kiszolgálására nem elegendő, és ezeknek az eszközöknek a fejlesztése igen költséges lehet. Kézenfekvő megoldást adhatnak viszont a felhő alapú erőforrások alkalmazása, olyan módon, hogy járműkommunikációban releváns adatok feldolgozása kiszervezésre kerülnek a felhőbe. A centrális felhő alapú feldolgozás azonban a nagy távolságok miatt fellépő késleltetés és instabil kapcsolat miatt nem képes biztosítani azokat a szigorú követelményeket, amelyeket a V2X use case-ek megkövetelnek [13]. Erre nyújtanak megoldást az Multi-Access Edge Computing [19] rendszerek. A megközelítés alapja az, hogy a felhős erőforrások fizikailag elosztott módon helyezkednek el, közel a felhasználókhoz (V2X esetén a járművekhez). Ezáltal az Edge szervereket (például közel az adótornyokhoz) integrálva az 5G hálózatba az alacsony késleltetésű, nagy adatsebességű kapcsolat útján a V2X használati esetek számításai kiszervezhetőek a felhőbe. Ezzel a rendszer tehermentesíti járművek hardvereit, miközben a legtöbb használati esetre definiált küldetés-kritikus működést is képes megvalósítani alacsony késleltetés és nagy megbízhatóság mellett. A MEC rendszerekbe, azok jellegéből adódóan olyan járműipari use case-eket érdemes implementálni, amelyek kihasználják az architektúra adottságait (2.2. táblázat). Az ilyen use case-ek erősen építkeznek az MEC által kezelt, elosztott, gyorsan hozzáférhető információra. Továbbá a szervereken lokálisan feldolgozott információ (pl.: szenzoradat) a hálózatot is tehermentesíti, mivel a többi jármű számára már csak a processzált adatot kell eljuttatni.

| Use case | Leírás | MEC revelancia |
|---|---|---|
| Kereszteződés áthaladás támogatás | Áthaladó jármű ütközésre való figyelmeztetése | Magas |
| Szoftverfrissítés | A járművek szoftverfrissítéseinek átvitele | Közepes |
| Valós idejű környezetérzékelés és nagy felbontású térképek | Járművek figyelmeztetése veszélyes útviszonyokra (pl.: jeges útfelület) | Magas |

| | | |
|---|---|---|
| „Keresztüllátás” (See-through) | Videófolyam továbbítása az előző jármű számára a megelőzendőtől | Magas |
| Kooperatív sávváltás autonóm járművek esetében | A sávváltást kezdeményező jármű jelzi a további jármű(vek) számára szándékát | Magas |
| Gyalogos forgalmi résztvevő detektálása | A jármű jelzi az érzékelt gyalogost a többi jármű számára | Magas |

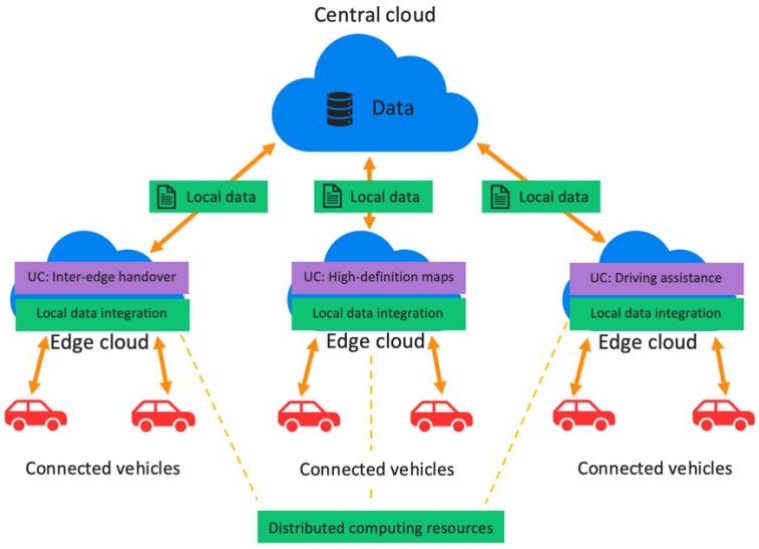
**2.2. táblázat MEC releváns autóipari use case-ek [20]**

Edge Computing rendszerek V2X kommunikációs alkalmazásaira az egyik legjobb példa a 2.2.-es táblázatban is feltűntetett „nagy felbontású térképek” use case. Az Edge szerverek nagy előnye, hogy a több jármű vagy eszköz által továbbított adatokat képes aggregálni és feldolgozni. Ezt kihasználva ezek az Edge szervereken elhelyezkedő, nagy felbontású, ún. HAD térképek [3] a járművek által továbbított koordinációs (pl.: lokáció, sebesség), szenzor és kamera adatokat dinamikusan, valós időben frissülve képesek tárolni [3]. Ezeket a térképeket a többi kapcsolódó jármű is folyamatosan eléri, amely a rendszer tulajdonságaiból adódóan közel valós időben képes információt biztosítani egy adott terület összes feldolgozott eseményéről és objektumáról. A HAD térképek rengeteg új használati eset lehetőségét teremtik meg, továbbá olyan jövőbeli megoldásoknak adhatnak alapot, mint az optimális forgalom-menedzsment vagy teljes autonóm vezetés.

A Edge Computing-ra építkező járműkommunikációs megoldásokkal több szervezet is aktívan foglalkozik. Többek között, az ETSI is készített már tanulmányt MEC-re építkező use case-ekről [21], de létrejött egy olyan konzorcium is, amely kifejezetten az autóipari fókuszú Edge Computing megoldásokkal foglalkozik, az Automotive Edge Computing Consortium (AECC). Az AECC is General Principles and Vision [22] kiadványában három olyan szempontot definiál, amely kiemelt jelentőséggel bír a felhő alapú járműkommunikációs megoldások megvalósításában. Ezeket hivatottak a járműipari Edge Cloud architektúrák realizálni a jövőben (2.5. ábra):

- Lokalizált hálózat: Helyi hálózatok kialakítása, amelyekhez az adott területen mozgó járművek csatlakoznak. Ezáltal a járművek és felhő között továbbított nagy adatmennyiségek is eloszlanak a helyi hálózatoknak megfelelően.

- Elosztott számítási erőforrás: Az erőforrások fizikai elosztása, a helyi hálózatoknak megfelelően. Ezáltal a számítási erőforrások koncentrációja kisebb lesz, amely alacsonyabb feldolgozási időket eredményez a kapcsolódó járművek által igényelt feladatok elvégzésére.
- Lokális adatok integrációja: Helyi adatok feldolgozása és egyesítése az elosztott erőforrás és a lokális hálózat segítségével. Ezáltal adott területen mozgó járművek számára a releváns információt kommunikálja a rendszer, ezzel felgyorsítva az adatfeldolgozást és valós idejű kommunikációt eredményezve.




**2.5. ábra Az Edge Cloud koncepció alapja**

## 2.4 Cloud Native és konténerizált szolgáltatási platformok

Manapság a felhő alapú számítástechnikának az egyik legfontosabb ágazata a Cloud Native computing, amely telekommunikációs szektorban is egyre nagyobb szerepet tölt be [23]. A koncepció lényege, hogy a szolgáltatásokat biztosító applikációk kialakítása és futtatása olyan módon valósul meg, hogy azok a lehető legjobban kihasználják a felhő alapú környezetek adottságait. Ezáltal flexibilisek és könnyedén skálázhatóak lesznek. A Cloud Native alapját a konténerizáció adja, amelynek lényege, hogy az alkalmazások komponenseikre bonthatók, amelyek jól definiált feladatot ellátó, uniform konténerekbe enkapszulálhatóak. Ezek a konténerizált alkalmazások tartalmaznak minden komponenst, amely segítségével a felhő infrastruktúra bármely

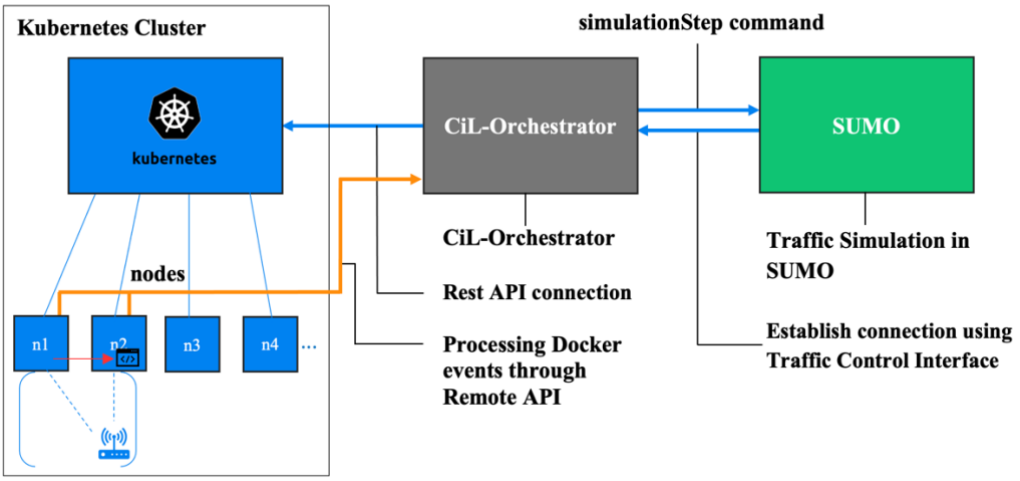
szerverére könnyen mozgathatóak, majd futtathathatóak. A konténerizált alkalmazások és szolgáltatások menedzsmentjére és orkesztrációjára elterjedten használt nyílt forráskódú platform a Kubernetes [5]. Segítségével elosztott, több szerverből álló felhő alapú hálózatokon hatékonyan kezelhetők az alkalmazások, amelyeket olyan funkciók segítenek, mint például működés közbeni skálázhatóság. Az 5. generációs celluláris mobilhálózatok új funkciói, például a virtualizációs megoldások következtében a Cloud Native megoldások egyre nagyobb szerepet kapnak az 5G hálózatokban. Ennek következtében a Kubernetes 5G rendszerekben való alkalmazása is egyre elterjedtebbé válik. A Kubernetes segítségével konténer alapú, felhő rádiós hozzáférési hálózat (Cloud Radio Access Network [24]) kiépítésére és menedzsmentjére is van lehetőség [25]. Azonban az egyik legizgalmasabb terület a Kubernetes, 5G-s MEC rendszerekben való felhasználása [26]. Egy 5G ökoszisztémával együttműködő Cloud Native orkesztrációs platform nagymértékben segítené az olyan elosztott megoldások hatékony működését, mint a MEC. Főleg olyan területeken, mint az Edge Cloud alapú C-V2X kommunikáció, kiemelten fontos a megbízható, jól vezérelt működés. Ezáltal egy olyan jól bevált rendszer alkalmazása, mint a Kubernetes potenciális platformját adhatja a jövőben ezeknek a járműipari megoldásoknak.

# 3 A Cloud-in-the-Loop szimulációs keretrendszer

A Cloud-in-the-Loop szimulációs keretrendszer a BSc szakdolgozatom elkészítése során végbement munka legfontosabb eredménye [4][27]. Implementáláskor a célkitűzés egy olyan többkomponensű rendszer megalkotása volt, amely segítségével a Edge Cloud alapú C-V2X járműkommunikációs modell vizsgálható mélyrehatóan. A rendszer feladata, hogy C-V2X járműkommunikációra képes járművek mozgását, illetve a szimulált forgalmi térképen elhelyezkedő Edge szerverek hozzáférési tulajdonságait modellezve vezéreljen egy valós elosztott felhő alapú hálózatot. Ezáltal adott járműipari use case által specifikált szimulált környezettel szorosan integrált elosztott felhő alapú rendszer és járműipari felhőalkalmazások működése vizsgálható valós időben. A keretrendszer több potenciális alkalmazási lehetőséget kínál Edge Cloud alapú C-V2X rendszerek vizsgálatára, használható a felhő környezet dimenzionálására, avagy adott use case és a hozzá tartozó követelmények alapján a hálózat megfelelő kialakítására. Továbbá a keretrendszer lehetőséget nyújt implementált C-V2X use case-ek, adott cloud környezetben való validálására. A keretrendszer három fő komponensből építkezik (3.1. ábra), amelyek lehetővé teszik az átfogó vizsgálatokat:

- Simulation of Urban MObility (SUMO) [28] forgalom szimulátor: A SUMO segítségével a keretrendszer képes nagy kiterjedésű úthálózatokon mozgó járműforgalom szimulálására. A szoftver számos lehetőséget nyújt a szimulált objektumok konfigurálására és manipulációjára, továbbá képes valós forgalmi térképek importálására is, minden forgalmi szabályozással kiegészítve.
- Cloud-in-the-Loop Orchestrator: A CiL-Orchestrator a keretrendszer központi eleme. Az egyik legfontosabb feladata, hogy kapcsolatot teremtsen SUMO-val, továbbá az elosztott felhő alapú hálózattal. Ez a rendszerkomponens adja a koncepció legnagyobb előnyét, itt kapcsolódik össze szimulált környezet a valós környezettel. Ebből kifolyólag az Orchestrator-ban kerülnek implementálásra azok a funkciók, algoritmusok, amelyekkel a szimulált járművek mozgásának megfelelő hálózati működés előidézhető.

- Elosztott felhő alapú hálózat (Kubernetes): Az elosztott felhő alapú hálózat szerepét egy Kubernetes orkesztrációs platformra építkező valós hardverekből felépülő cluster tölti be. A cluster egyes elemei (node-jai) valósítják meg a C-V2X járműkommunikációs modell Edge szervereit. A Kubernetes alkalmazásával lehetőség nyílik egy valós, potenciális járműipari elosztott hálózat mellett az egyes szolgáltatásokat biztosító alkalmazások vizsgálatára is.




**3.1. ábra A Cloud-in-the-Loop Szimulációs keretrendszer architektúrája**

A keretrendszerbe továbbá számos olyan funkció implementálásra is került, amely különböző, a rendszer teljesítményét leíró metrikákat képesek gyűjteni. Majd gyűjtött adatokat előre specifikált módon tárolni későbbi feldolgozásra.

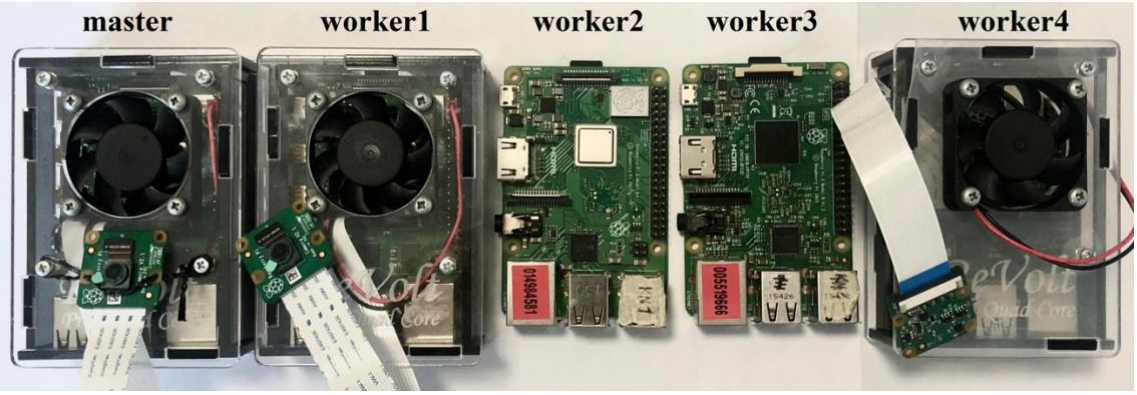
## 3.1 Kubernetes a CiL szimulációs keretrendszer kontextusában

Az Egde Cloud alapú C-V2X járműkommunikációs backend megvalósításáról a keretrendszerben egy Kubernetes orkesztrációs platform gondoskodik, amelynek alapját egy Raspberry Pi egylapkás számítógépekből álló hardveres cluster adja. A Kubernetes rendszer node-jai reprezentálják a felhő alapú C-V2X koncepció egyes Edge szervereit. A járműipari C-V2X use case-eket kiszolgáló alkalmazások és szolgáltatások vizsgálatát az elosztott felhő alapú hálózaton futtatható konténerizált alkalmazások segítik. A Kubernetes cluster-ek felépítésüket tekintve rendelkeznek egy úgynevezett master node-dal, amelyek a teljes, több hardverből álló rendszeren futó alkalmazások, szolgáltatások menedzsmentjéért és orkesztrációjáért felelős. A master által vezérelt,

kiszolgálást biztosító node-okat worker-nek nevezik. Ezeken futnak a konténerizált alkalmazások, amelyek reprezentálják azokat az autóipari use case-eket biztosító szolgáltatásokat, amiknek a vizsgálatára a CiL keresztrendszer teremt lehetőséget

A Kubernetes cluster hardveres alapját adó 5db Raspberry Pi eszközök (3.2. ábra) elnevezései (leírják az adott node szerepét is rendszerben) a következők:

- master node: A Kubernetes cluster menedzsmentjéért felel, továbbá az CiL Orchestrator a szimuláció alapján történő vezérlést érvényesítő kérései mind a master-en keresztül jutnak érvényre.
- worker1 node: A jelenlegi mérési összeállításban az egyik Edge szerver szerepét tölti be, biztosítja a járműipari alkalmazásokat modellező konténerizált alkalmazások futtatását.
- worker2 node: A méréseket egyéb módon segítő funkciókat tölt be jelenleg.
- worker3 node: Hasonlóan a woker2 node-hoz, a jelenlegi mérések során nem reprezentál Edge szervert, de mérések során alkalmazott node.
- worker4 node: Hasonlóan a worker1 node-hoz az egyik Edge szervet tölti be,




**3.2. ábra A Kubernetes cluster hardveres elemei**

A keretrendszerben az egyes node-ok Wi-Fi kapcsolaton keresztül csatlakoznak egy külön alhálózatot biztosító router-hez, amely a mérésekhez szükséges eszközökön kívül minden egyéb hálózati eszközt izolál, ezáltal biztosítva mérési eredmények pontosságának növelését.

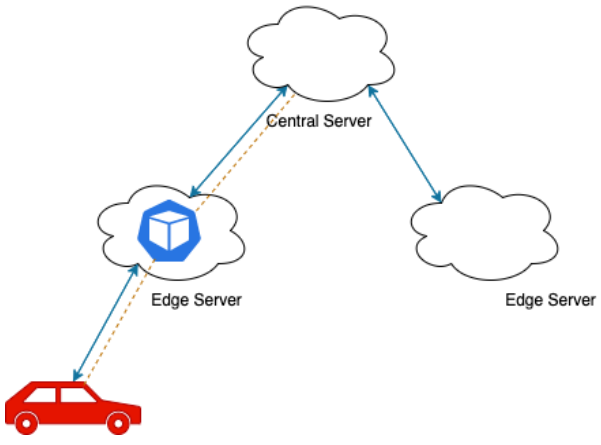
# 4 Vizsgálati szempontok, implementáció

Az Edge Cloud alapú C-V2X járműkommunikációs modell esetén a különféle szolgáltatásokat elosztott alkalmazások biztosítják. Az alkalmazások lokális komponense a jármű hardverén fut, a távoli „backend” komponense viszont a jármű számára legjobb hálózati kapcsolatot és hozzáférést kínáló, közeli Edge szerveren. Az alkalmazások által biztosított szolgáltatások célja, hogy valamely autóipari use case működését biztosítsa. Ebből kifolyólag az alkalmazásoknak teljesítenie kell azokat követelményeket, amelyet az adott használati esetek előírnak többek között a késleltetésre, a megbízhatóságra és az adatsebességre vonatkozóan. Amennyiben a jármű mozgása révén az adott Edge szerverrel való hálózati kapcsolatában QoS degradáció lép fel és nem képes teljesíteni az előírt követelményeket, a rendszernek relokálnia kell a backend alkalmazáskomponenst, egy olyan Edge szerverre, amely képes biztosítani azt a szolgáltatási minőséget, amely megfelel a követelményeknek. Az alkalmazás áthelyezés (handover) esetében fontos szempont, hogy a rendszer biztosítsa a szolgáltatás folytonosságát (service continuity), úgy, hogy közben az elosztott erőforrások terheltsége se legyen indokolatlanul magas. A CiL szimulációs keretrendszerrel folytatott első vizsgálataim során is ezen szempontok szerint vizsgáltam két különböző alkalmazás áthelyezési stratégiát [27]. Az Edge Cloud rendszerek architektúrális felépítésük révén, számos C-V2X használati eset alkalmazásánál biztosítani kell a megfelelő alkalmazás handover-t, beleértve az aktuális munkafolyamatok átvitelét is. Azonban, hogy a szolgáltatási követelményeknek eleget tegyen a rendszer, az alkalmazások működése során is folyamatosan kell biztosítani az előírt szolgáltatási minőséget. Ezért ebben a dolgozatban az alkalmazásszintű hálózati forgalom QoS vizsgálata és a forgalomnak az elosztott rendszerre gyakorolt hatásának elemzése került középpontba. A handover események ezáltal az alkalmazásszintű QoS degradáció útján is vizsgálhatóak. A hálózati forgalmat generáló alkalmazások implementálásával több mérési szempont is terítékre kerül. A járműveket kiszolgáló alkalmazások skálázódása a hálózati és alkalmazásszintű QoS paraméterekre való hatása mellett a Kubernetes cluster számítási erőforrásaira is hatással van. Az megvalósított mérések során az alkalmazásszám növekedésének hatásait és a generált forgalmi paraméterek változtatásának hatásait vizsgáltam az alkalmazásszintű QoS

paraméterekre. Mindezt olyan szimulált forgalmi szituációkban vizsgálva, ahol a járműveket kiszolgáló backend alkalmazások több esetben is áthelyeződnek az Edge szerverek között.

## 4.1 Implementáció

Ahhoz, hogy alkalmazásszintű hálózati forgalmat generáljak, szükség volt egy olyan modell felépítésére, amely visszaadja egy járműipari use case-et kiszolgáló több komponensű alkalmazás viselkedését. Az alap koncepció szerint a járműben futó alkalmazáskomponens a hálózat (valós környezetben 5G hálózat) segítségével kommunikál az Edge szerveren futó backend alkalmazáskomponenssel. Jelenleg a Kubernetes rendszerekben munkamenetet migrációt is megvalósító, stateful alkalmazás relokáció megvalósítására még csak PoC megoldások képesek [29]. Ezért ahhoz, hogy az alkalmazások közötti hálózati kapcsolat QoS szempontjából is vizsgálható legyen, egy olyan use case implementálását valósítottam meg, amelyben az Edge alkalmazást egy stateless konténerizált alkalmazás valósítja meg. A use case szerint a járművön futó alkalmazáskomponens folyamatosan adatcsomagokat továbbít a Edge alkalmazás részére, amelyeket az továbbküld egy központi szerveren futó alkalmazás felé. A jármű mozgásából adódóan minden esetben a rendszer arra az Edge szerverre helyezi át a backend alkalmazást, amely a legjobb hálózati QoS paramétereket biztosítja a jármű számára (4.1. ábra).

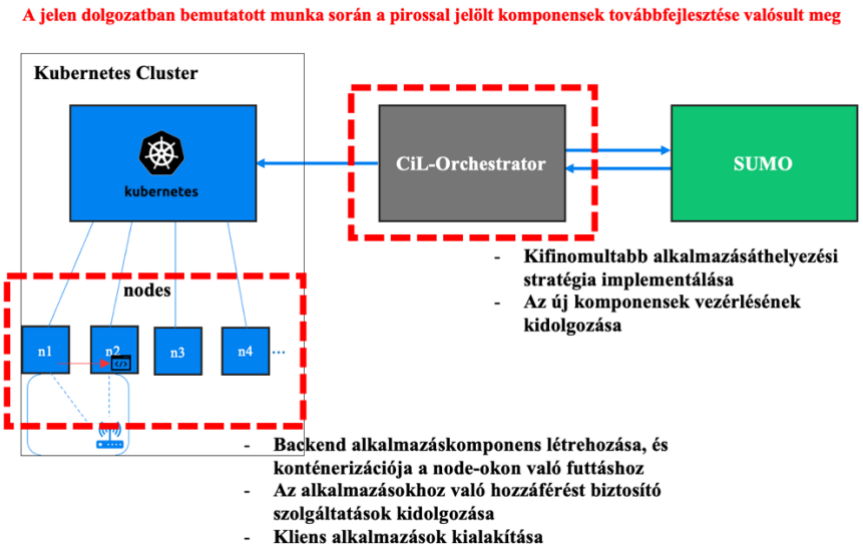



**4.1. ábra Az implementált use case alap koncepciója**

A use case keretrendszerbe való implementálásához három komponens megtervezésére és kialakítására volt szükség:

- 1. A járművön futó alkalmazáskomponens: Az alkalmazásszintű hálózati forgalom generálására egy iperf [30] nevű, IP hálózatok analízisére kialakított szoftvert alkalmaztam. A program képes TCP, UDP és SCTP protokollok alapján adatcsomagokat generálni és továbbítani, az adatátviteli sebesség, csomagméret és számos egy paraméter konfigurálásával. Az átvitt csomagok alapján képes metrikákat generálni, amelyekkel leírhatók az alkalmazásszintű hálózati forgalom QoS paraméterei.
- 2. Az Edge szerveren futó, backend alkalmazáskomponens: A Kubernetes node-okon futó konténerizált alkalmazás feladata, hogy a beérkező csomagok továbbítsa egy „központi szerver” felé. Bővebben a 4.4.1-es fejezetben.
- 3. A központi szerver alkalmazás: Az alkalmazásszintű hálózati forgalom fogadásáért az iperf szoftver szerver üzemmódú futtatása felel. A fogadott csomagokat megvizsgálva információt küld vissza kliens részére, ami alapján képes metrikákat generálni az átvitelről.

Mivel a keretrendszerben minden járművön futó alkalmazáskomponenseknek nem lehet külön hardvert dedikálni, ezért ezek futtatása egy olyan Kubernetes cluster részeként szereplő worker node-on valósul meg, amely a modellezett járműipari használati eset által specifikált, szimulált forgalmi környezetben nem valósít meg Edge szervert. Fontos, hogy az iperf alkalmazás a Kubernetes környezetén kívül fut, hogy az a platform teljesítményét közvetlenül ne befolyásolja, továbbá a lehető legjobban megközelítve modellezze a valós működést. Az Edge szerveren futó, backend alkalmazáskomponenst megvalósító konténerizált alkalmazás elhelyezkedése a jármű szimulációban pozíciója minden esetben arra Edge szervert reprezentáló worker node-ra helyezi át a rendszer, amely a szimulált környezetben a legjobb hozzáférést biztosít. Továbbá a központi szerver alkalmazás elhelyezkedése a keretrendszer orkesztrációt és a további mérési adatokat gyűjtését megvalósító laptopon valósul meg, ezáltal az adatgyűjtés mellett valós időben is monitorozhatóak az adatküldési folyamatok.



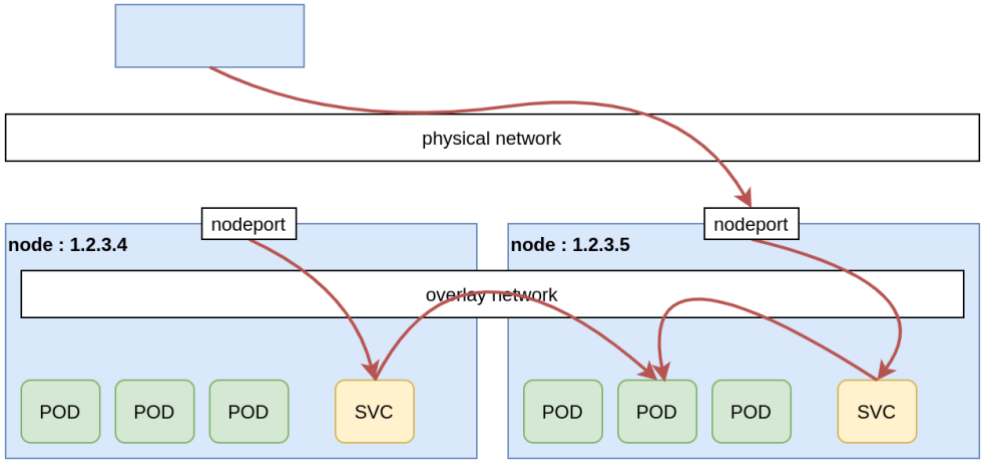

**4.2. ábra A keretrendszerben eszközölt fejlesztések**

Az implementáció során számos fejlesztés valósult meg a keretrendszerben, ezekből a legfontosabbak a 4.2 ábrán láthatóak. Az ábrázolt fejlesztéseket a következő alfejezetek részletezik. A backend alkalmazás fejlesztését a 4.1.2 alfejezet mutatja be, az alkalmazáshoz kapcsolódó szolgáltatáskidolgozást a 4.1.1 alfejezet részletezi. A kliens alkalmazás kialakításáról a teljes 4.1 fejezetben esik szó, továbbá a Cil-Orchestrator-t érintő fejlesztésekről nagyrészt a 4.1.3 alfejezet értekezik.

### 4.1.1 Alkalmazásszintű hálózati folyamatok

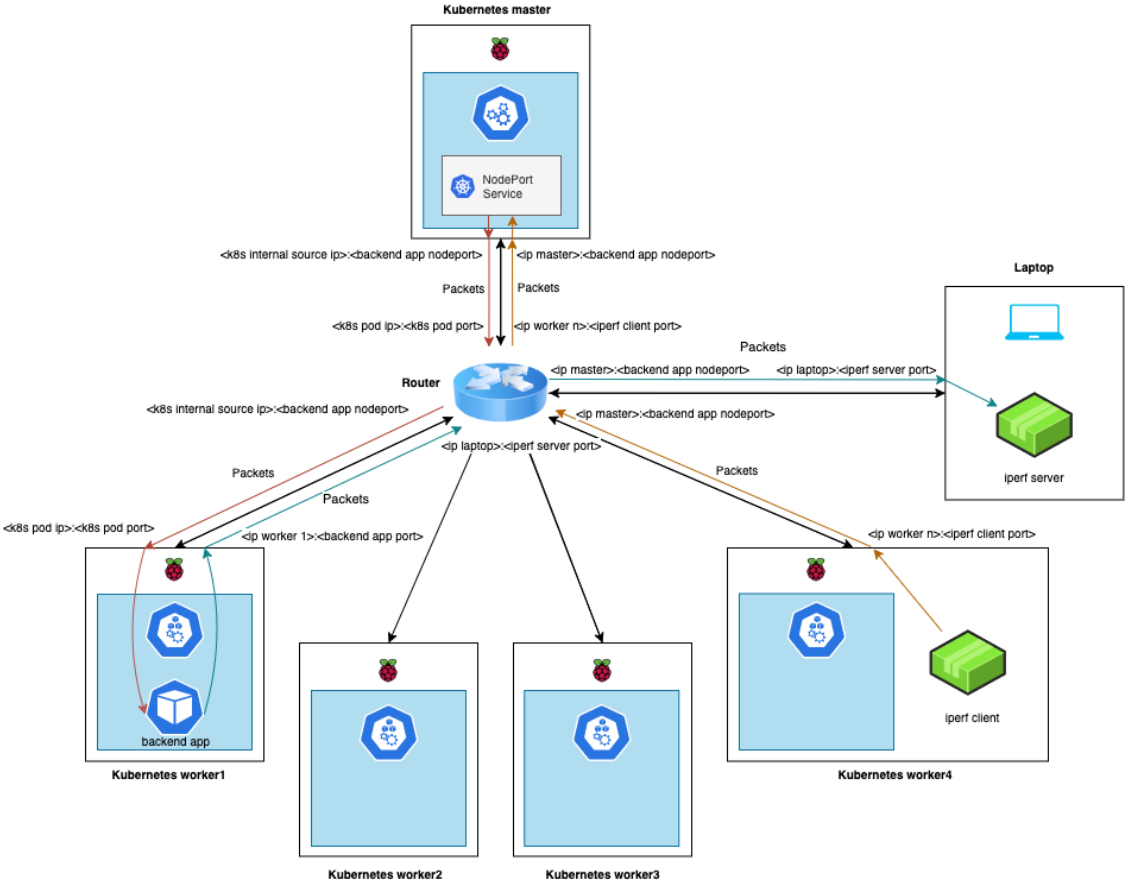
Az ismertetett use case implementálása során szükségessé vált az alkalmazáskomponensek kommunikációjának a keretrendszerben való kialakítása. Egy valós celluláris rendszerben a mobilhálózat (mobil IP) végzi a csomagok megfelelő irányú továbbítását, a keretrendszerben azonban a Kubernetes által biztosított hálózati funkciókat használtam ki a csomagok célba juttatásához. Alapvetően a járművön futó alkalmazáskomponens által továbbított csomagokat minden esetben a járműhöz kapcsolódó Edge szerveren futó backend alkalmazáskomponens felé kell továbbítania a rendszernek. A Kubernetes cluster-en futó konténerizált alkalmazások úgynevezett pod-okban futnak [31], a pod-ok hálózati eléréséhez úgynevezett service-eket [32] kell definiálni. A Kubernetes NodePort (4.3. ábra) elnézésű service típusa azt a célt szolgálja, hogy egy adott pod-hoz definiált statikus port-ot minden cluster node IP címén elérhetővé teszi, majd a bejövő csomagokat a Kubernetes overlay hálózatán [33] továbbítja annak a worker-nek, ahol a szolgáltatást biztosító pod fut. Ezt követően a C-

V2X járművek által generált, folyamatos hálózati forgalmat modellezve, a Kubernetes képes biztosítani az aktuális kiszolgáló Edge szerver-en futó backend alkalmazáskomponens felé a csomagtovábbítást.




**4.3. ábra Kubernetes NodePort service [34]**

Ezekre építkezve az implementált mérések során, a keretrendszer elindítja a szerver alkalmazást megvalósító iperf szervert, amely majd a küldött csomagok és a klienssel való kommunikáció alapján létrehozott metrikákat ábrázolja és menti későbbi feldolgozásra. Ezt követően minden szimulációba belépő jármű számára létrehoz egy backend alkalmazást a járműpozíciója alapján megfelelően meghatározott Edge szervert reprezentáló worker node-on. Fontos, hogy a SUMO-ból kinyert jármű azonosítók alapján meghatározott port számmal a Kubernetes minden jármű számára létrehoz egy service-t, amelyen keresztül elérhetők az azokat kiszolgáló egyes backend komponensek. Ezzel közel párhuzamosan a rendszer minden szimulált jármű számára létrehoz egy iperf klienst (az erre dedikált worker node-on), amely a C-V2X használati esetek járműoldali alkalmazáskomponenseit reprezentálja. Ezek a kliensek úgy kerülnek konfigurálásra, hogy azok a master node IP címére továbbítsák az adatokat, arra port-ra, amelyre korábban az egyes backend komponensek konfigurálva lettek. Ezáltal a master node a beérkező csomagokat a NodePort service-ek az egyes járművekhez definiált cél port-ok alapján továbbítja a megfelelő backend komponenseknek, amelyeket azok továbbküldenek a központi szerver alkalmazást megvalósító iperf szerver részére. Az alkalmazásszintű hálózati kommunikációt a 4.4. ábra szemlélteti.




**4.4. ábra Példa az implementált use case hálózati kommunikációjára**

### 4.1.2 Az Edge szerveren futó backend alkalmazáskomponens

A CiL keretrendszerben az Edge szerveren futó backend alkalmazáskomponenst reprezentáló alkalmazásnak az általam vizsgált use case esetében azt a funkciót szükséges ellátnia, hogy a beérkező csomagokat továbbítja egy előre konfigurált IP cím adott portjára. A jelenlegi mérések során UDP csomagok forgalma alapján vizsgáltam a use case működését, illetve az elosztott hálózat teljesítményét. Ennek megfelelően a backend alkalmazásnak is UDP csomagok kezelését kellett megvalósítania. Ehhez egy Python nyelven írt alkalmazást használtam fel (4.1. kódrészlet, Függelék 1.), amely a beérkező csomagok egy előre definiált IP cím adott port-jára továbbítja (a fogadó IP címet a kódot futtató hoszt határozza meg). A Python szkript a <fogadó port>:<cél IP cím>:<cél port> formátumban várja a konfigurációs paramétereket.

```
knownClient = None
knownServer = (remoteHost, remotePort)
...
    data, addr = s.recvfrom(32768)
    ...
    if addr == knownClient:
        if debug:
            print("\tforwording tO " + str(knownServer))

        s.sendto(data, knownServer)
    else:
        if debug:
            print("\tforwarding to " + str(knownClient))
        s.sendto(data, knownClient)
```

**4.1. kódrészlet Részlet a backend alkalmazást megvalósító Python kódból**

Ahhoz, hogy a Kubernetes környezetben futtatható legyen az alkalmazás, konténerizálni kell. A Kubernetes a konténerizált alkalmazások használatához Docker [35] konténerizációt és futtatókörnyezetet alkalmaz. A Docker segítségével úgynevezett docker image-ek készíthetőek, amelyek a Kubernetes a pod-jaiban képes futtatni. Az image elkészítéséhez a Docker buildx nevű plugin-jét alkalmaztam, amely képes az ARM architektúrára épülő Raspberry Pi-okon futtatható alkalmazásokat konstruálni. Az alkalmazást a konfigurálhatóság érdekében úgy hoztam létre, hogy a konténeren kívülről is át lehessen adni a hálózati címeket. A konténerizáció során a forráskód mellett a Docker az összes komponenst is enkapszulálja, amelyek az alkalmazás futtatásához szükségesek.

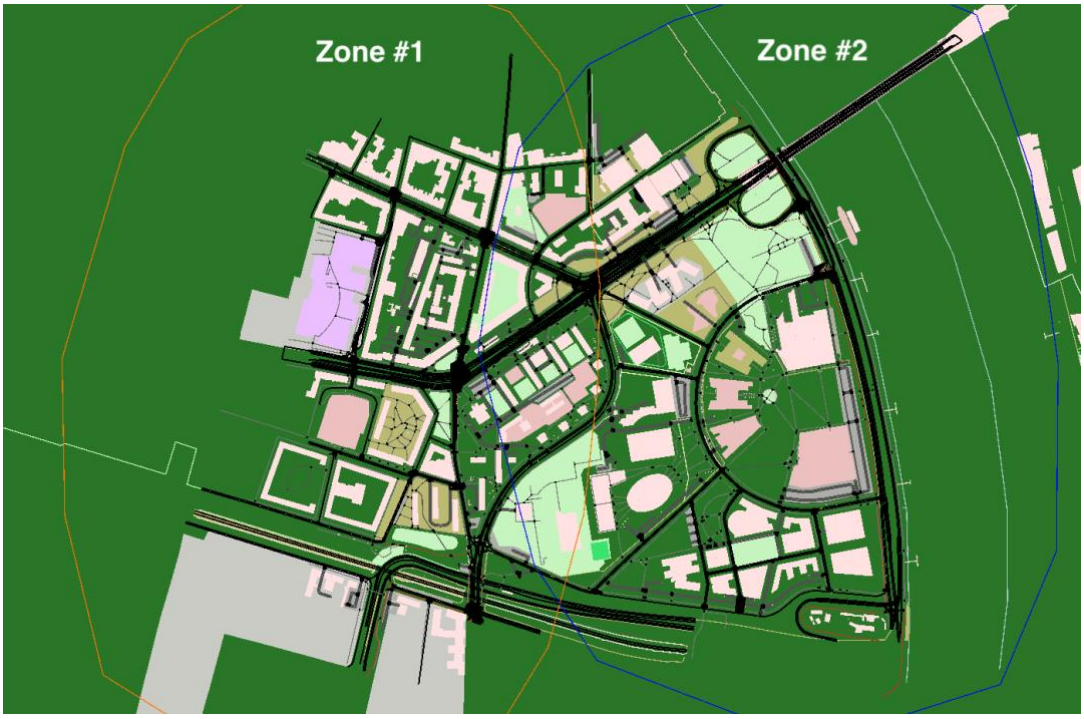
### 4.1.3 Az implementált use case szimulációs környezete

A Cloud-in-the-Loop keretrendszer egyik legnagyobb előnye, hogy bármely use case által specifikált forgalmi helyzetet képes modellezni, és a szimulált információk alapján egy valós, elosztott hálózatot vezérelni. Ezáltal átfogóan vizsgálhatók és validálhatóak architektúrák, koncepciók mielőtt költséges valós hardvereket és rendszereket (járművek, celluláris hálózatok) alkalmaznánk. Ezáltal a jelenleg implementált use case-nek megfelelően is ki kellett alakítani azt a szimulációs környezetet, amely segítségével modellezhető a valós működés.

A szimuláció alapjául szolgáló forgalmi térképet a Budapest XI. kerületi Infopark környékén elhelyezkedő városi környezet adja (4.5 ábra). Az Edge Cloud alapú C-V2X járműkommunikáció modellezésére két Edge szervert valósítottam meg, amelyekhez késleltetési zónákat definiáltam. Ezek zónák reprezentálják azt a határt, amelyeken belül a hozzájuk tartozó Edge szerver még ki tudja szolgálni az adott késleltetésre vonatkozó követelményeket, megfelelő megbízhatóság mellett. A zónák mérete jelenleg még nem specifikált egyetlen követelményrendszerre sem, pusztán egy lehetséges topológiának a megvalósítása, adott lefedettséggel, azonban a mérésben vizsgált esetek vizsgálatára tökéletesen megfelelnek. Továbbá a keretrendszer feladata jelenleg az elosztott rendszer és az általa futtatott alkalmazások működésének vizsgálata, ezáltal celluláris hálózat és annak felépítéséből adódó tényezők sem vizsgáltak. A két zónának megfelelően a Kubernetes két worker node-ja tölti be az Edge szerverek szerepét, ezáltal adott zónákban mozgó járművek számára az adott zónához tartozó worker node futtatja a járműveket kiszolgáló backend alkalmazásokat. A szimuláció során a járművek előre meghatározott útvonalakat követnek, ezzel jól modellezze egy városi környezetben való forgalmat. Ennek hatása a mérési eredményeken is megjelenik, mivel szignifikáns lesz a zóna váltások által előidézett alkalmazás áthelyezések hatása.

A SUMO által generált szimulációs adatokat a CiL-Orchestrator dolgozza fel, majd ezek alapján vezérli a Kubernetes cluster-en futó pod-okat, service-eket, továbbá járművön futó alkalmazáskomponenst reprezentáló iperf kliens alkalmazásokat is. Azokban az esetekben, mikor járművek belépnek a zónák átlapolódó tartományába, megindul az alkalmazás relokációs folyamat. A jelen dolgozat elkészítése során ennek a folyamatlogikának a továbbfejlesztése is megtörtént. Az CiL-Orchestrator-ba implementált algoritmus szerint, az átlapolódó területre belépő járműhöz tartozó

backend alkalmazáskomponens áthelyezése megvalósul. Amennyiben nem halad tovább az új zóna irányába és kilép az átfedett tartományból, vissza korábbi zónába, a rendszer visszahelyezi az alkalmazást. Az alkalmazás áthelyezési módszer megvalósítására Cold Migration [27] stratégiát alkalmaztam.




**4.5. ábra Az implementált use case szimulációs térképe**

# 5 Mérési eredmények és értékelés

A mérések során az alkalmazásszintű hálózati forgalmat az iperf szoftver által generált UDP csomagok segítségével generáltam. Az UDP protokoll legfontosabb tulajdonsága, hogy a továbbított csomagok célba jutásáról a címzett nem ad visszajelzést a feladónak, így azok újraküldése sem valósul. Ezáltal a hálózati hibák miatt a csomagok elveszhetnek. A mérések során, a csomagtovábbítás alapján generált iperf metrikák (5.1. ábra) segítségével vizsgáltam az alkalmazásszintű QoS degradációt. Ez különösen olyan esetekben jelentős, mikor a szimulált járművek által indikált alkalmazás relokáció valósul meg, ilyenkor az UDP csomagvesztés jelentős lehet, amelynek mértéke sok információt hordoz a vizsgálandó use case működését illetően.

```
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  5] 0.00-30.00 sec   611 KBytes   167 Kbits/sec   2.789 ms 971/5860 (17%)
```

**5.1. ábra iperf szoftver által gyűjtött metrikák**

A méréseket két megközelítésből végeztem. A szimuláció során olyan, use case által specifikált körülményeket vizsgáltam, amelyek modellezik azt a működést, ami valós körülmények során is megjelenik. Ezek kiterjednek azokra a szituációkra, amelyek során a járművek számának növekedésének hatására romlik a hálózati kapcsolat minősége, vagy azokra, amelyek megnövekedő alkalmazásszintű hálózati forgalom hatását vizsgálják.

## 5.1 A C-V2X járműforgalom növekedésének hatása az alkalmazásszintű QoS degradációra

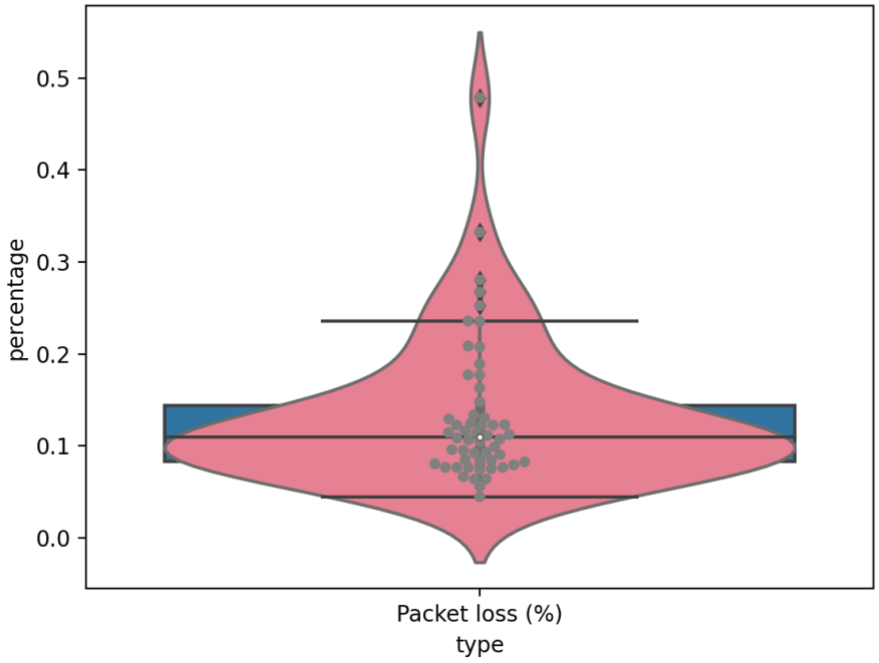
A járműszám növekedésének megfelelően a Kubernetes által biztosított backend alkalmazások skálázódnak, ez terheli az elosztott rendszer számítási erőforrásait. Továbbá a járműveknek megfelelően létrejönnek a járműveken futó alkalmazáskomponenseket reprezentáló iperf kliens instanciák is, amelyek hálózati, UDP forgalmat generálnak. Ezek együttes hatása szignifikánsan befolyásolja az alkalmazásszintű hálózati forgalom QoS paramétereit. A mérések során, arra kerestem a választ, hogy a járműszám növekedése milyen hatással van egy adott, mért jármű alkalmazásszintű hálózati csomagátvitelére.

Az iperf kliens konfigurációja során a csomagátviteli sebességre 50 Kbyte/s-ot állítottam be, amelyeket a kliens 3 másodpercenként 30 másodperces csomagküldés formájában valósít meg. Ennek hatása még nem terheli túlságosan a Raspberry Pi eszközökre építkező elosztott hálózat hardveres erőforrásait, de szignifikáns hatása van a QoS paraméterekre a járműszámmal való skálázódás esetén. A méréseket 25, 50, 100, 200 jármű városbeli szimulációjával végeztem.

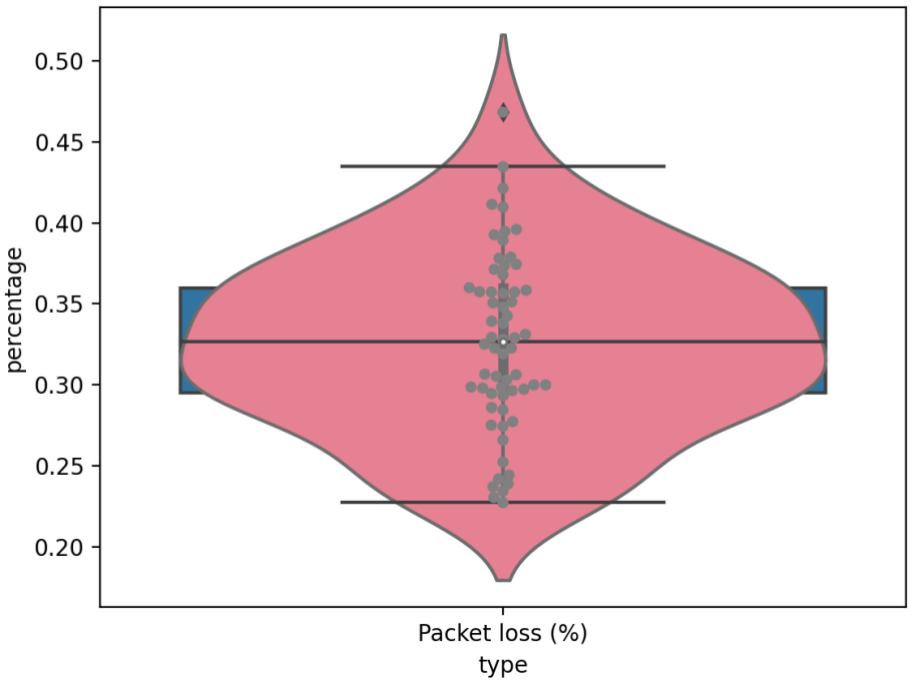
| Járművek száma (db) | Adatátviteli sebesség a csomagvesztés eredményeképpen (KByte/s) | Összes csomag (db) | Elveszett csomagok (db) | Csomagvesztés (%) |
|---|---|---|---|---|
| 25 | 44,609 | 1500 | 199.389 | 13,29 |
| 50 | 34,316 | 1500 | 490.29 | 32,671 |
| 100 | 30,938 | 1500 | 590.062 | 39,317 |
| 200 | 25,144 | 1500 | 769.1 | 51,246 |

**5.1. táblázat A járműszám növekedésének átlagos hatása a QoS paraméterekre**

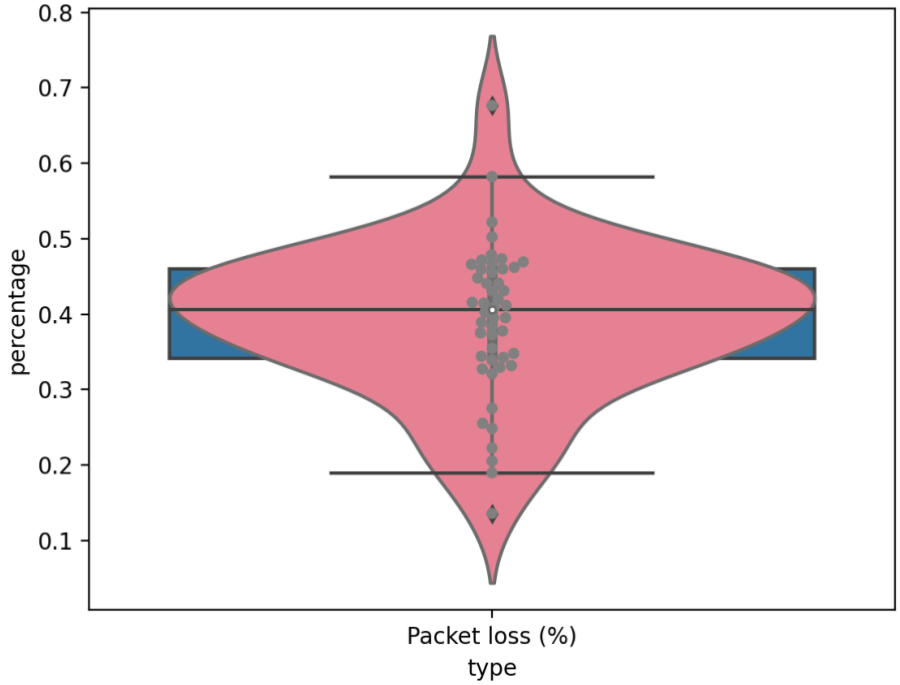
A 5.1. táblázat bemutatja, hogy adott az járműszám növekedése átlagosan milyen hatással van az átvitel minőségére. Ezeket az átlagos eredményeket a 30 másodperc alatt elküldött 1500 db UDP csomagra vonatkoztatott átviteli érték adják. A járműszám növekedésének hatása jól látható degradációt eredményez az átvitelben. Ennek okai, hogy a hálózat terheltsége mellett a Kubernetes cluster hardveres terheltsége is megnövekszik, főleg azáltal, hogy egyre több alkalmazás relokációt kell megvalósítania. A csökkenő számítási kapacitás hatására az egyes alkalmazás áthelyezési folyamatok ideje is megnövekszik, amelyek további csomagvesztést okoznak. A csomagvesztések eloszlását szemléltető ábrákon (5.2-5.5 ábrák) ennek hatásai jól vizsgálhatóak. 25 jármű esetében (5.2 ábra), sokkal kevesebb alkalommal fordul elő zónaváltás (a többi jármű által), ezáltal az átlagos 13% körüli csomagvesztés körül mozognak az veszteségek, a kiemelkedő értékeket is azok az események adhatták, amikor a vizsgált jármű zóna váltása során, több jármű is éppen új zónába lépett. 50 és 100 (5.3 és 5.4 ábra) jármű esetében a csomagvesztések már sokkal nagyobb szórás mutatnak. Ez arra enged következtetni, hogy a nagyobb valószínűséggel előforduló azonos idejű zónaváltások szignifikáns hatással vannak a csomagvesztésre. 200 (5.5 ábra) jármű esetén viszont már a hálózati terheltség hatása jelenik meg erőteljesebben, amely következményeképpen átlagosan a csomagok fele elveszik.



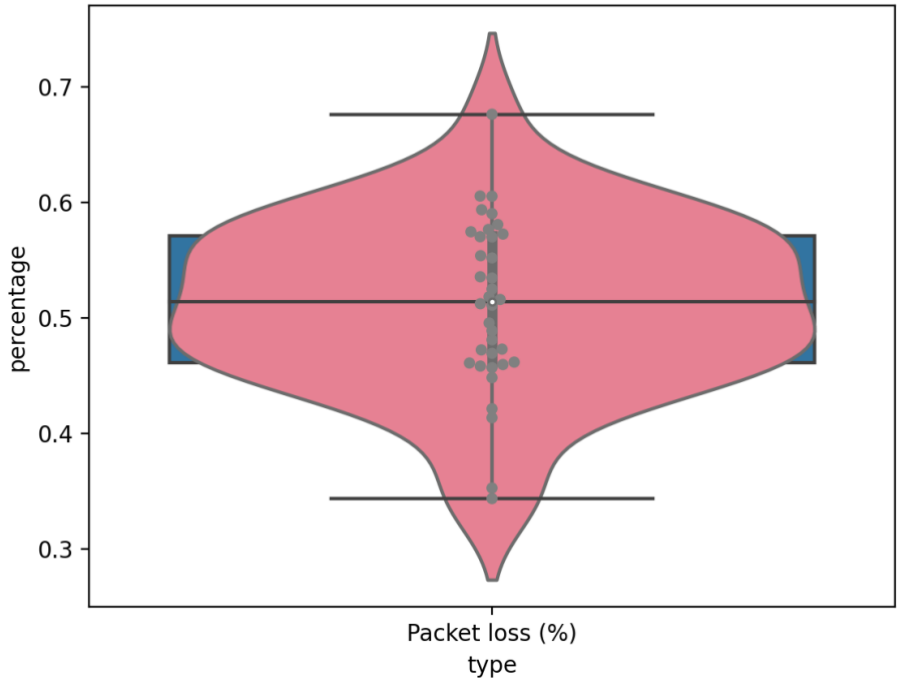

**5.2. ábra A csomagvesztés eloszlása 25 jármű esetén**




**5.3. ábra A csomagvesztés eloszlása 50 jármű esetén**




**5.4. ábra A csomagvesztés eloszlása 100 jármű esetén**




**5.5. ábra A csomagvesztés eloszlása 200 jármű esetén**

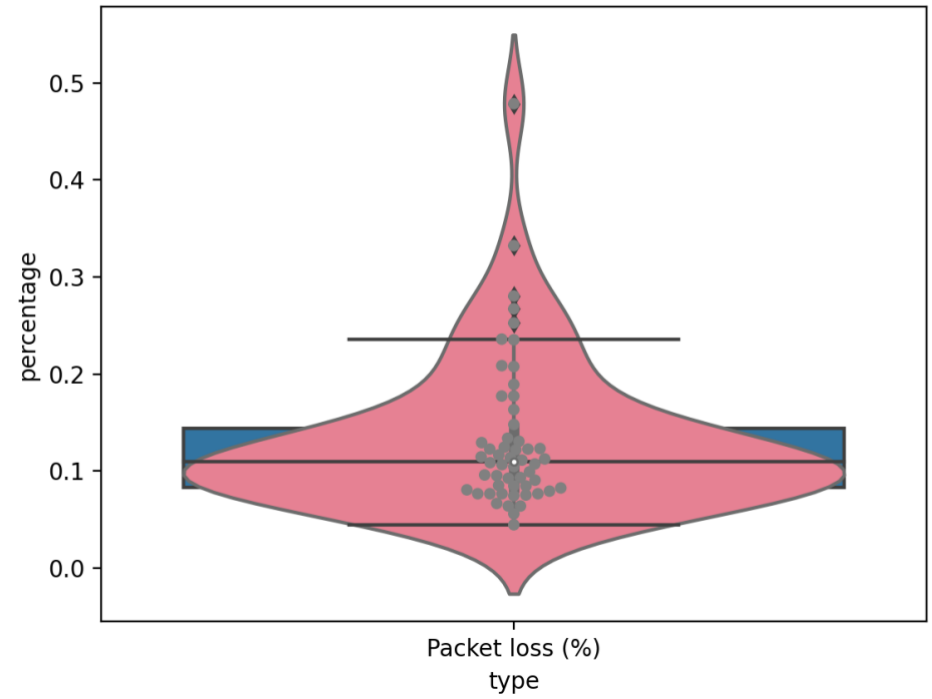
## 5.2 Az adatátviteli sebesség növekedésének hatása az alkalmazásszintű QoS degradációra

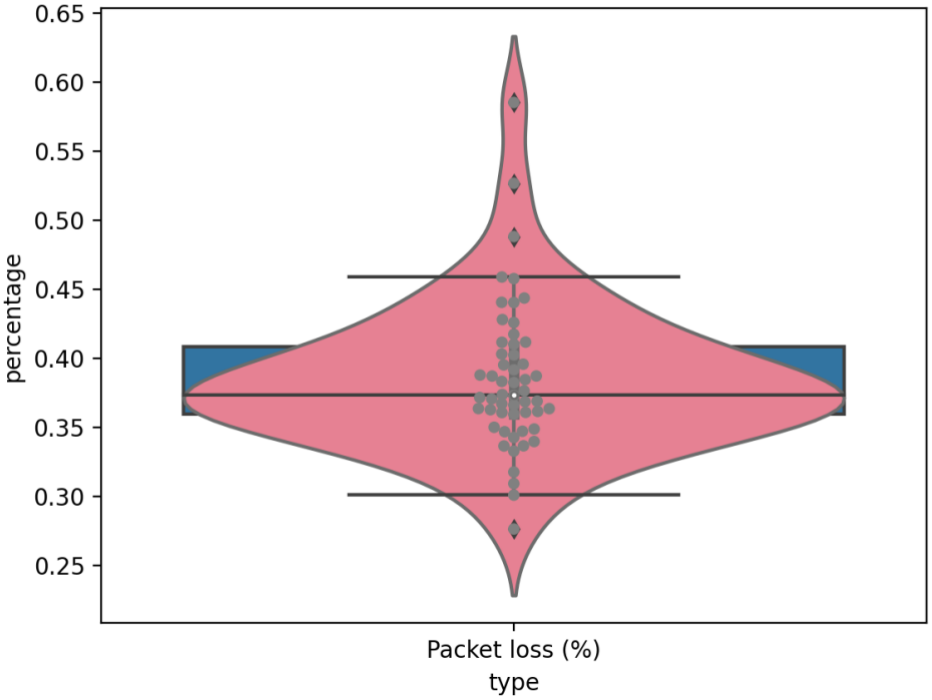
Az átviteli sebesség növekedésének a hálózati forgalomra gyakorolt hatásait a korábbi mérési adatok összevetésével valósítottam meg. Az összehasonlítás alapjául azt mérési szituációt vettem, amelyben 25 jármű mozog szimulációban, mivel ebben az esetben még kisebb hatása volt az elosztott rendszer hardveres terheltsége okozta QoS degradációnak. Ezáltal sokkal kiemeltebben jelenik meg a hálózati terhelés hatása, amely jó összehasonlítási alapot ad vizsgálatok során. A mérések során a vizsgált jármű adatátviteli sebességét 300 KByte/s-ra növeltem, a többi, háttérterhelést megvalósító jármű által továbbított adatok mennyiségét is dupláztam, ezzel 100 KByte/s-ra növelve az átviteli sebességet. Az adatátvitel az 5.1 alfejezetben bemutatott méréseknél alkalmazott módszerhez hasonlóan, 30 másodperces csomagátviteli blokkokban valósult meg.

| Járművek száma (db) | Célzott adatátviteli sebesség (KByte/s) | Adatátviteli sebesség a csomagvesztés eredményeképpen (KByte/s) | Csomagvesztés (%) |
|---|---|---|---|
| 25 | 50 | 44,609 | 13,29 |
| 25 | 300 | 125,346 | 31,75 |

**5.2. táblázat A járműszám növekedésének átlagos hatása a QoS paraméterekre**

Az 5.2 táblázatban látható átlagos átviteli értékek alapján jól látható, hogy az egyes kliensek által továbbított adatok átviteli sebességének növelése hatására a csomagvesztés is több mint a duplájára nőtt, az 5.6 ábrán látható csomagvesztések eloszlását és szórását vizsgálva azonban látható, hogy hasonlóak a két esetben. Ez azt mutatja, hogy a hálózati forgalom növekedése ugyan okozott QoS degradációt, azonban a zóna váltások által indikált, node-ok közötti alkalmazás áthelyezések száma nem változott, és ezáltal a Kubernetes hardveres terheltségének hatása sem jelentkezett olyan mértékben, amelyet az 5.1 fejezetben mutatott például az 50 járművet bevezető mérés eloszlása. Az 5.1 és 5.2 fejezetekben bemutatott eredmények újabb, későbbi tervezési és tesztelési irányokat jelöltek ki, az egyik legfontosabb eredmény, hogy az alkalmazás relokációs feladatokat az erős QoS függőségek miatt az aktuális járműforgalmak alapján érdemes optimalizálni az Edge Cloud alapú C-V2X rendszerekben.




**5.6. ábra A csomagvesztés eloszlása az adatátviteli sebességek növekedésének hatására**

# 6 Összegzés

A TDK dolgozatomban bemutattam a V2X járműkommunikáció egy éppen kiforró típusát, a C-V2X járműkommunikációs modellt, amely a közeli jövő 5. generációs celluláris mobilhálózata adta lehetőségeket kihasználva képes egy új szintre emelni járműkommunikációs megoldásokat. Ehhez kapcsolódóan egy rövid áttekintést adtam a 3GPP szabványosítási törekvéseiről, amelyek nagymértékben hozzájárulnak a C-V2X rendszerek későbbi bevezetéséhez. A korábbi LTE hálózatokra építkező V2X megoldások mellett bemutattam az 5G-s NR V2X funkcióit és megoldásait, és a kapcsolódó járműipari használati esetek alapján meghatározott követelményrendszert. Továbbá ehhez kapcsolódóan ismertettem azokat celluláris járműkommunikáció szempontjából fontos, 5G-s hálózatok hozta újításokat, mint az URLLC kommunikáció, a hálózat virtualizációs megoldások vagy az Edge Computing integráció lehetősége. Ezt követően részleteztem a dolgozatom központi témáját adó Edge Computing alapú C-V2X járműkommunikációt. Ezzel kapcsolatban bemutattam a potenciális autóipari használati eseteket, illetve a Cloud Native computing modellre építkező Kubernetes platformot, amely a jövő elosztott felhő alapú járműkommunikáció kiszolgáló platformja lehet. Ezután rövid betekintést nyújtottam a jelenlegi munka alapjául is szolgáló Cloud-in-the-Loop szimulációs keretrendszer felépítésébe, amely még a BSc szakdolgozatom alatt végzett munka eredményeként született.

A dolgozat másik felében az elmúlt félévben végzett munkát mutattam be, amely során a CiL keretrendszerbe további komponenseket implementálva, és a rendszert továbbfejlesztve újabb, felhő alapú C-V2X járműkommunikációt modellező méréseket végeztem. A mérések célja a C-V2X járműkommunikációs megoldásokat kiszolgáló alkalmazások hálózati forgalmának és a vonatkozó QoS degradáció vizsgálata volt. Ehhez szükség volt egy új járműipari use case implementálására, amely során a járműveken futó alkalmazáskomponensek, az azokat kiszolgáló Edge szervereken futó backend alkalmazáson keresztül kommunikálnak egy távoli szerverrel. Ennek kapcsán bemutattam a Kubernetes azon hálózati funkcióit, amelyek lehetővé tették az implementációt és részleteztem a backend alkalmazást megvalósító komponenst, továbbá a méréshez megvalósított szimulált városi és hálózati környezetet. A méréseket két szempontból végeztem el. Az első méréscsoport során azt vizsgáltam, milyen

hatással van az Edge szerverek által kiszolgált járműszám növekedése a járművek által generált alkalmazásszintű hálózati forgalomra. A további mérések során, pedig azt vizsgáltam, hogy a járművek által generált forgalom növekedése milyen hatással van az alkalmazásszintű hálózati forgalomra. Az eredmények azt mutatták, hogy a járművek általi Edge szerver közötti váltás okozta alkalmazás áthelyezések szignifikáns hatással vannak az elosztott rendszer erőforrásaira, amelyek az alkalmazások által generált hálózati forgalomban erőteljesen megmutatkoznak. Következtetésképp a keretrendszert alkalmazva további vizsgálatok során olyan módszereket (alkalmazás áthelyezési stratégiákat, algoritmusokat) kell kialakítani, amelyek segítségével biztosíthatjuk a járműkommunikációs használat által támasztott követelményeket. Továbbá, a későbbi terveimben átfogóbb méréseket szeretnék végezni, ezekhez azonban egy nagyobb, számítási kapacitással rendelkező, valós rendszereket pontosabban modellező cluster-ba szeretném átköltöztetni a keretrendszert. Így a méréseket már olyan reprezentatív adatforgalmakkal és járműszámokkal tudnám elvégezni, amely valós forgalmi szituációkat és cluster-működést reprezentál.

# Irodalomjegyzék


[1] R. Weber, J. Misener, and V. Park, ‘C-V2X - A Communication Technology for Cooperative, Connected and Automated Mobility’, in *Mobile Communication - Technologies and Applications; 24. ITG-Symposium*, 0 2019, pp. 1–6.

[2] ‘3GPP SA6 initiatives to enable new vertical applications’, *https://www.3gpp.org/news-events/conferences/2082-sa6_verticals*.

[3] H. Ma, S. Li, E. Zhang, Z. Lv, J. Hu, and X. Wei, ‘Cooperative Autonomous Driving Oriented MEC-aided 5G-V2X: Prototype System Design, Field Tests and AI-based Optimization Tools’, *IEEE Access*, vol. PP, pp. 1–1, Mar. 2020, doi: 10.1109/ACCESS.2020.2981463.

[4] L. Maller, L. Bokor, and P. Suskovics, ‘Implementation of a Cloud-in-the-Loop Simulation Framework for Evaluating Automotive Edge Cloud Solutions’, *Pervasive Mob. Comput.*, Aug. 2021.

[5] ‘Kubernetes’. https://kubernetes.io/

[6] X. Wang, S. Mao, and M. Gong, ‘An Overview of 3GPP Cellular Vehicle-to-Everything Standards’, *GetMobile Mob. Comput. Commun.*, vol. 21, pp. 19–25, Nov. 2017, doi: 10.1145/3161587.3161593.

[7] Volkswagen, ‘Car2X in the new Golf: A “technological milestone”’. https://www.volkswagen-newsroom.com/en/stories/car2x-in-the-new-golf-a-technological-milestone-5919

[8] D. Jiang and L. Delgrossi, ‘IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments’, in *VTC Spring 2008 - IEEE Vehicular Technology Conference*, 2008, pp. 2036–2040. doi: 10.1109/VETECS.2008.458.

[9] ‘3GPP’, *https://www.3gpp.org/*.

[10] M. H. C. Garcia *et al.*, ‘A Tutorial on 5G NR V2X Communications’, *IEEE Commun. Surv. Tutor.*, vol. 23, no. 3, pp. 1972–2026, 2021, doi: 10.1109/comst.2021.3057017.

[11] ‘Harmonisation of the 5.9 GHz spectrum band for real-time information exchange will improve road and urban rail transport safety’. https://digital-strategy.ec.europa.eu/en/news/harmonisation-59-ghz-spectrum-band-real-time-information-exchange-will-improve-road-and-urban-rail

[12] I. Kim, H. Yoo, E. Young Hyun, S. Cho, and B. Jeon, ‘An Integrated Communication Message Framework of Inter-Vehicles for Connected Vehicles using Mobile Virtual Fence(MVF)’, *Int. J. Eng. Technol.*, vol. 7, pp. 102–105, Aug. 2018, doi: 10.14419/ijet.v7i3.33.18584.

[13] A. Alalewi, I. Dayoub, and S. Cherkaoui, ‘On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey’, *IEEE Access*, vol. 9, pp. 107710–107737, 2021, doi: 10.1109/ACCESS.2021.3100472.

[14] S. A. Abdel Hakeem, A. A. Hady, and H. Kim, ‘5G-V2X: standardization, architecture, use cases, network-slicing, and edge-computing’, *Wirel. Netw.*, vol. 26, no. 8, pp. 6015–6041, 0 2020, doi: 10.1007/s11276-020-02419-8.

[15] ITU, ‘Work Item: G.QoE-5G’. https://www.itu.int/itu-t/workprog/wp_item.aspx?isn=14680

[16] *5G; Service requirements for enhanced V2X scenarios*, Release 15. 3GPP, 2018.

[17] SDxCentral Studios, ‘How 5G NFV Will Enable the 5G Future’. https://www.sdxcentral.com/5g/definitions/5g-nfv/

[18] E. Tittel, ‘SDN vs. NFV: What’s the difference?’ https://www.cisco.com/c/en/us/solutions/software-defined-networking/sdn-vs-nfv.html
[19] ETSI, ‘Multi-access Edge Computing (MEC)’. https://www.etsi.org/technologies/multi-access-edge-computing
[20] F. Giust *et al.*, ‘Multi-Access Edge Computing: The Driver Behind the Wheel of 5G-Connected Cars’, *IEEE Commun. Stand. Mag.*, vol. 2, no. 3, pp. 66–73, Sep. 2018, doi: 10.1109/MCOMSTD.2018.1800013.
[21] ‘Multi-access Edge Computing (MEC); Study on MEC Support for V2X Use Cases’, Sep. 2018.
[22] ‘AECC: General Principle and Vision (White Paper) Version 3.0’, Jan. 2020.
[23] Ericsson, ‘Cloud native is transforming the telcom industry’. https://www.ericsson.com/en/cloud-native?gclid=CjwKCAjwwsmLBhACEiwANq-tXEubl9djVBh_QIr1NxiSyKkmmnAZyTe9QTJNXHnwdO4tBWu6Fff4JhoCcFkQAvD_BwE&gclsrc=aw.ds
[24] Sean Hsu, ‘What is C-RAN? The Evolution From D-RAN to C-RAN’. https://www.ufispace.com/company/blog/what-is-cran-the-evolution-from-dran-to-cran
[25] C. Novaes, C. Nahum, I. Trindade, D. Cederholm, G. Patra, and A. Klautau, ‘Virtualized C-RAN Orchestration with Docker, Kubernetes and OpenAirInterface’. 2020.
[26] KubeEdge Maintainers, ‘KubeEdge@MEC: Combining the Kubernetes ecosystem with 5G’. https://www.cncf.io/blog/2021/07/20/kubeedgemec-combining-the-kubernetes-ecosystem-with-5g/
[27] L. Maller, ‘Cloud-in-the-Loop szimulátor fejlesztése és alkalmazása járműipari használati esetek Edge Cloud környezetben történő vizsgálatához’, Budapesti Műszaki és Gazdaságtudományi Egyetem, Budapest, 2020.
[28] ‘Simulation of Urban MObility (SUMO)’. https://www.eclipse.org/sumo/
[29] ‘Kubernetes Pod Migration - PoC’. [Online]. Available: https://docs.google.com/document/d/1E5p_FOHDGAp5YEQ23dCi9I8wPnMzd4aOazxI4uO_AMo/edit
[30] ‘iPerf - The ultimate speed test tool for TCP, UDP and SCTP’. https://iperf.fr/iperf-doc.php
[31] ‘Kubernetes Pods’. https://kubernetes.io/docs/concepts/workloads/pods/
[32] ‘Kubernetes Service’. https://kubernetes.io/docs/concepts/services-networking/service/
[33] ‘Kuberenetes - Cluster Networking’. https://kubernetes.io/docs/concepts/cluster-administration/networking/
[34] Thibault Debatty, ‘Exposing a Kubernetes application : Service, HostPort, NodePort, LoadBalancer or IngressController?’ https://cylab.be/blog/154/exposing-a-kubernetes-application-service-hostport-nodeport-loadbalancer-or-ingresscontroller
[35] ‘Docker’. https://www.docker.com/

# Ábra- és táblázatjegyzék

# Rövidítések jegyzéke

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| C-V2X | Cellular-Vehicle-to-Everything |
| CAM | Cooperative Awareness Message |
| CiL | Cloud-in-the-Loop |
| eMBB | enhanced Mobile Broadband |
| LTE | Long Term Evolution |
| MEC | Multi-access Edge Computing |
| mMTC | massive Machine Type Communications |
| NFV | Network Function Virtualization |
| PoC | Proof of Concept |
| QoS | Quality of Service |
| RSU | Road-Side Unit |
| SDN | Software-Defined Networking |
| UDP | User Datagram Protocol |
| URLLC | Ultra-Reliable Low Latency Communication |
| V2I | Vehicle-to-Infrastructure |
| V2P | Vehicle-to-Pedestrian |
| V2V | Vehicle-to-Vehicle |
| V2X | Vehicle-to-Everything |
| VR | Virtual Reality |

# Függelék

## Függelék 1. A backend alkalmazást megvalósító Python kód

```
import sys, socket

# Whether or not to print the IP address and port of each packet received
debug = True


def fail(reason):
    sys.stderr.write(reason + '\n')
    sys.exit(1)


if len(sys.argv) != 2 or len(sys.argv[1].split(':')) != 3:
    fail('Usage: udp-relay.py localPort:remoteHost:remotePort')

localPort, remoteHost, remotePort = sys.argv[1].split(':')

try:
    localPort = int(localPort)
except:
    fail('Invalid port number: ' + str(localPort))
try:
    remotePort = int(remotePort)
except:
    fail('Invalid port number: ' + str(remotePort))

try:
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind(('', localPort))
except:

    fail('Failed to bind on port ' + str(localPort))

knownClient = None
knownServer = (remoteHost, remotePort)
sys.stdout.write('All set, listening on ' + str(localPort) + '.\n')
while True:
    data, addr = s.recvfrom(32768)
    if knownClient is None or addr != knownServer:
        if debug:
            print("")
        knownClient = addr

    if debug:
        print("Packet received from " + str(addr))

    if addr == knownClient:
        if debug:
            print("\tforwording tO " + str(knownServer))

        s.sendto(data, knownServer)
    else:
```

```
        if debug:
            print("\tforwarding to " + str(knownClient))
        s.sendto(data, knownClient)
```

# Függelék 2. Az adatfeldolgozást és ábrázolást megvalósító Python kód

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def read_create_data(path):
    interval = []  # [None for x in range(N)]
    transfer = []
    bandwidth = []
    jitter = []
    lost = []
    total = []
    type = []
    percentage = []

    with open(path, 'r') as f:
        for line in f:
            line = line.split(" ")
            line = [elem for elem in line if not (elem == " " or elem ==
"" or elem == "[")]
            #if len(line) >= 9:
            #    print(len(line), line[8], line)
            print (line)
            if len(line) >= 9 and "sec" in line:
                interval.append(line[1])
                transfer.append(line[3])
                bandwidth.append(float(line[5]))
                jitter.append(float(line[7]))
                lost_total = line[9].split("/")
                #if int(lost_total[1]) >= 5900:
                lost.append(int(lost_total[0]))
                #if int(lost_total[1]) >= 5900:
                total.append(int(lost_total[1]))
                #if int(lost_total[0]) >= 5500:
                    #print(line)
                type.append("Packet loss (%)")
                percentage.append(int(lost_total[0])/int(lost_total[1]))
                print(int(lost_total[0])/int(lost_total[1]))

    return type, interval, transfer, bandwidth, jitter, lost, total,
percentage


type, interval, transfer, bandwidth, jitter, lost, total, percentage =
read_create_data("/Users/leventemaller/PycharmProjects"

"/Docker_post-process/QoS_logs"

"/load_test_1.txt")
```

```
data = {"type": type, "interval": interval, "transfer": transfer,
"bandwidth": bandwidth, "jitter": jitter, "lost": lost,
        "total": total, "percentage": percentage}
df_data = pd.DataFrame(data)
df_data.to_csv("./asd.csv")

packet_loss_mean = df_data["percentage"].mean()
jitter_mean = df_data["jitter"].mean()
total_packets = df_data["total"].mean()
lost_packets = df_data["lost"].mean()
bandwidth_mean = df_data["bandwidth"].mean()

print(packet_loss_mean)
print(jitter_mean)
print(total_packets)
print(lost_packets)
print(bandwidth_mean)

seaborn_df = pd.DataFrame(df_data, columns=['type', 'percentage'])
print(seaborn_df)
df1 = pd.read_csv("./test1.csv")
print(df1.head())

ax = sns.boxplot(x='type', y='percentage', data=seaborn_df)
# Add jitter with the swarmplot function.
ax = sns.swarmplot(x='type', y='percentage', data=seaborn_df,
color="grey")
ax = sns.violinplot(x='type', y='percentage', data=seaborn_df,
palette="husl")

plt.show()
```