Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Rádiós fronthaul forgalom támogatása kapcsolt Ethernet hálózatokban

2016. október 27.

Szerzők:

Nagy Árpád Péter, Bella Norbert

Konzulensek:

Dr. Simon Csaba (TMIT), Dr. Maliosz Markosz (TMIT), Dr. Farkas János (Ericsson)

Tartalomjegyzék

Kivonat

Jelenleg a kritikus, különösen pontosan és/vagy gyorsan célba juttatandó adatokat dedikált hálózaton, külön infrastruktúrán lehet csak küldeni. Az IEEE 802.1 munkacsoportja az általa standardizált Time-Sensitive Networking (TSN) megoldással egy jelentős lépést tett abba az irányba, hogy az Ethernet hálózatokon pontos időzítést lehessen megvalósítani és ehhez kapcsolódóan QoS (minőségbiztosítás) mechanizmusokat határoztak meg.

A jelenleg telepített hálózatokban a rádiós fronthaul forgalom különösen szigorú, legalább mikroszekundum, gyakran nanoszekundum pontosságot követel meg. A nyilvános mobil hálózatok rádiós fronthaul tartományában a CPRI (Common Public Radio Interface) protokollt használják erre a célra. A dolgozatban arra kerestük a választ, hogy milyen konkrét TSN mechanizmusok, illetve azok kombinációja képes hatékonyan támogatni a CPRI protokoll által generált forgalmat.

Abstract

Currently the time critical traffic with strict QoS (Quality of Service) guarantees is sent over dedicated networks deployed over purpose-built infrastructures. The solutions provided by the Time-Sensitive Networking (TSN) work group of IEEE 802.1 offer support for traffic requiring precise synchronization.

The fronthaul radio networks have one of the strictest requirements among the currently deployed networks with delay variations expressed in hundreds of nanoseconds. The fronthaul domain of the public mobile networks uses CPRI (Common Public Radio Interface) protocol for this goal. Our work investigates the applicability of several TSN mechanisms on supporting the radio fronthaul traffic.

1. Bevezetés

A klasszikus Ethernet ki tudta elégíteni azokat a piaci igényeket, ahol a best effort forgalom az elsődleges, viszont nem tud megfelelő garanciát nyújtani szigorú szolgáltatás minőségi (QoS, Quality of Service) elvárásokkal rendelkező forgalom számára. A felhasználók egyre nagyobb igénye a multimédiás folyamok iránt arra ösztönözte a kutató társadalmat, hogy kifejlesszenek olyan új mechanizmusokat, amik kiterjesztik az Ethernet szabvány QoS képességeit. Az IEEE 802.1 Time-Sensitive Networking (TSN) munkacsoportja azon dolgozik, hogy olyan mechanizmusokat szabványosítsanak, amelyek lehetővé teszik az idő szinkronizált, alacsony késleltetésű szolgáltatásokat az Ethernet hálózatokban. Eredetileg ezeket a mechanizmusokat a multimédia folyamok igényeit szem előtt tartva tervezték meg, de időközben új QoS igényű alkalmazások jelentek meg, elsősorban valós idejű ipari alkalmazások, ahol a vezérlő üzeneteket nagyon szigorú időzítési határok között kell tartani. A mobil rádió erőforrás menedzsment egy fontos részhalmaza az ilyen szigorú igényekkel rendelkező vezérlő alkalmazásoknak. A Common Public Radio Interface (CPRI) egy már a mindennapi gyakorlatban is alkalmazott ipari szabvány, amit arra terveztek, hogy ilyen vezérlő funkciókat valósítson meg. Jelenleg a hálózat, ami a CPRI folyamokat szállítja a rádiós egység és az alapsávi egység között (azaz fronthaul hálózat) dedikált pont-pont optikai technológiával van megvalósítva, ami egy elég költséges megoldás. Ez arra motiválja az üzemeltetőket, hogy alternatívákat keressenek, amivel csökkenteni lehet a költségeket, és csökkentsék az üzemeltetési és menedzsment többletet. Az Ethernet technológia, kiegészítve a TSN mechanizmusokkal, olyansokoldalú és alacsony üzemeltetési költségűmegoldást képes nyújtani, amelyik megfelelő jelölt erre a feladatra.

A TSN alkalmazhatósága a CPRIoE (CPRI over Ethernet) részeként nem egyszerű. Ezért a dolgozat célja a csomagokra bontott CPRI forgalomviselkedésének vizsgálata egy olyan Ethernet fronthaul hálózatban, amelyik alkalmazza az általunk kiválasztott TSN mechanizmus.. A vizsgálataink részeként olyan szcenáriókat választottunk ki, amelyek rámutatnak a CPRIoE jelenlegi problémáira. A kiválasztott szcenáriók elemzéséhez szükség volt egy megfelelő szimulációs környezetre, amiben könnyű a megadott körülmények között dolgozni és teljesíti a bemutatott követelményeket.

A következő fejezetekben bemutatjuk a Time-Sensitive Networking mechanizmusokat és a CPRI protokollt. Azután a szimulációs eszközökről írunk, köztük a kiválasztottat, amit a munkához használtunk, az alap hálózatot és a forgalom modelleket. A 5. fejezetben mutatjuk

be a szimulációs eseteket és az ezekből kapott eredményeket. Végül egy összefoglalással zárjuk a dolgozatot.

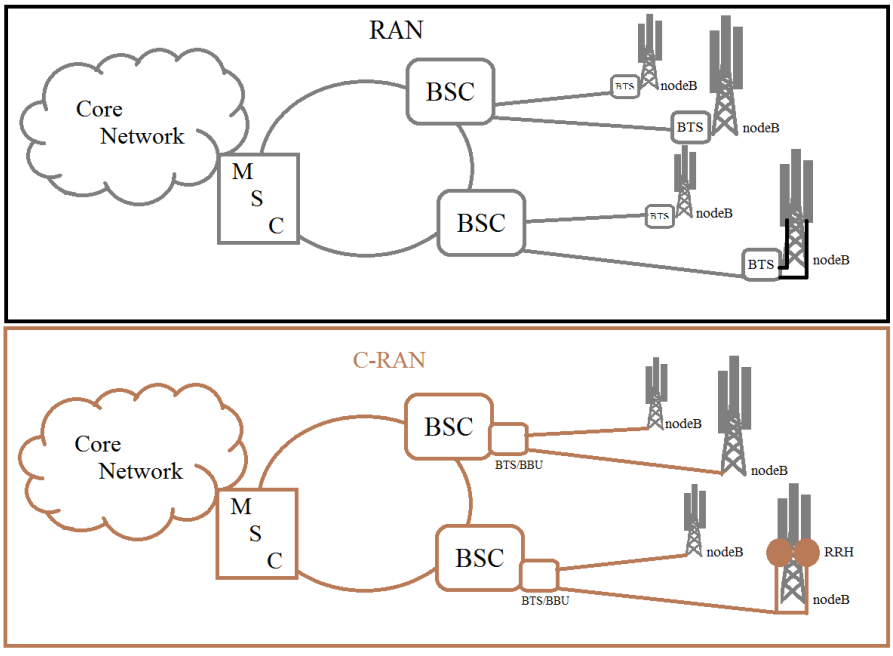
2. Time-Sensitive Networking

A Time-Sensitive Networking (TSN) egy munkacsoport az IEEE 802.1-n belül. TSN új szabványokat fejleszt alacsony késleltetésű stream szolgáltatásokhoz. A következő fejezetekben a TSN-t kiszólgáló hálózatokat tárgyaljuk. [1]

2.1. Rádiós hálózatok

A TSN-nek sok felhasználási lehetősége van az ipari, autóipari, kereskedelmi és a fronthaul hálózatokban. Ez a tanulmány a TSN eszközök fronthaul hálózati felhasználására koncentrál, a hálózat típus magas QoS követelményei miatt. Ez a rész egy rövid összefoglalása a tipikus fronthaul hálózatoknak és a QoS követelményeiknek.

Az egyik fő különbség a C-RAN (Cloud Radio Access Network) architektúra és a RAN architektúra között, hogy egy alapsávi CPRI átvitelre van szükség a központi alapsávi egységek (BBU, Baseband Unit) és a távoli rádióállomások (RRH, Remote Radio Head) között, ez az 1. ábrán látható. Erre az alapsávi hálózatra hivatkoznak fonthaul-ként a C-RAN architektúrában. A hálózatnak oly módon kell az alapsávi egységek és a rádiós állomások összeköttetését megoldania, hogy az a rádiós teljesítményre ne legyen negatív hatással. Ez azt jelenti, hogy a hálózat felelős, azért hogy a BBU és az RRH közötti CPRI forgalom teljesítse a CPRI által felállított követelményeket. A következő fejezet egy rövid összefoglalást nyújt a CPRI-ről.



1. ábra - RAN, C-RAN architektúra

2.2. CPRI

A CPRI (Common Public Radio Interface) ipari cégek együttműködésével jött létre. A rádiós egység (Radio Equipment, RE) és a rádiós egység vezérlő (Radio Equipment Controller, REC) közötti interfészként fejlesztették ki. Az összeköttetés optikai kábellel történik így nagyobb sebesség érhető el a hálózaton és nagyobb távolságra lehet helyezni a bázis állomásokat, így akár több antennát irányíthat egy vezérlő. Az összeköttetés dedikált pont-pont alapon történik, a cél pedig az lenne, hogy az állomások egy hálózatra csatlakozzanak az összes vezérlővel, így kevesebb eszköz kéne a felépítéshez és rugalmasabban lehetne változtatásokat eszközölni. Ezekre adhat megoldást az Ethernet.[3]

A CPRI folyamokra nagyon szigorú követelmények vonatkoznak. Ezek közül a tanulmány során a legfontosabb a maximális végpont-tól végpontig tartó késleltetésre (e2eD) vonatkozó megkötés, amely nem lehet több mint 100µs.[2]

2.3. Ethernet emlékeztető

Bár az Ethernet több mint 50 éves technológiájának bemutatása túlmutat ennek a dolgozatnak a mondanivalóján, mégis úgy érezzük néhány részletének fel elevenítésére szükség van a dolgozat mondandójának pontos megértéséhez, ez a néhány dolog következik ebben a fejezetben.

A címek, amik az eszközöket azonosítják, a fizikai (MAC - Media Access Control) címek. Ezek 48 bites címek, amiknek első 24 bitje a gyártó azonosító a másik 24 bitje az eszközt azonosítja. Az Etherneten az adatok kisebb egységekben más néven keretekben közlekednek. A keretek felépítése a 1. táblázat látható.

<table>
<tr><td>ETHERNET</td><td>Preamble</td><td>SFD</td><td>MAC DA</td><td>MAC SA</td><td>802.1Q tag</td><td>length</td><td>Payload</td><td>FCS</td><td>IFG</td></tr>
<tr><td>LAYER 2 FRAME</td><td></td><td></td><td rowspan="2">6oktet</td><td rowspan="2">6oktet</td><td rowspan="2">opcionális 4 oktet</td><td rowspan="2">2 oktet</td><td rowspan="2">46-1500 oktet</td><td rowspan="2">4 oktet</td><td rowspan="2">12 oktet</td></tr>
<tr><td>LAYER 1 PACKET</td><td colspan="2">7 oktet 1oktet</td></tr>
</table>

1. táblázat - Ethernet keret[4]

Ezek a részek kifejtve:

- Preamble szinkronizációs szerepet tölt be
- Keret kezdet határoló (SFD – Start of Frame Delimiter)
- Cél fizikai cím (MAC DA – MAC destination address)
- Forrás fizikai cím (MAC SA – MAC source address)
- Opcionális 802.1Q tag – ez akkor van használva, ha alkalmazzuk a 802.1Qban definiált szabványt (VLANok)

- Hosszúságot vagy Ethernet II esetén Ethernet típust jelző mező (length)
- Hasznos adat (Payload)
- Keret ellenőrző sorozat (FCS – Frame Check Sequence)
- Keretek közti szakadék (IFG – Inter Frame Gap)

3. Szimulátor

Az OMNeT++ egy objektum orientált moduláris diszkrét esemény szimulációs keretrendszer nyílt forráskódú komponens bázissal. Az OMNeT++ elsődleges területe a kommunikációs hálózatok, de az architektúrája használhatóvá teszi IT rendszerekhez, sorokat használó hálózatokban és egyéb területeken is.

Az OMNeT++ nem egy konkrét szimulátor, hanem infrastruktúrát és eszközöket ad szimulációk írásához. A keretrendszer alapja a könnyen összerakható komponens architektúra szimulációs modellekhez. Ezek a modellek újrahasznosítható komponensekből, azaz modulokból állíthatók össze.

A modulokat kapukon keresztül lehet összekötni és az összetett modulokat magas szintű nyelvvel (Network Description, NED) lehet összeállítani, ennek az egymásba ágyazásnak nincs konkrét határa. A NED tartalmazza a C++-ban fejlesztett modulok „kivezetéseit” és így az egyszerű moduloktól kezdve a komplex topológiákig minden objektumnak kell, hogy legyen NED-es reprezentációja a szimulátorban.

A modulok kapukon és összeköttetéseken keresztül üzenet átadással kommunikálnak. Ezek az üzenetek egy előre meghatározott útvonalon vagy közvetlenül a célnak továbbítatók. A modulok rendelkezhetnek olyan paraméterekkel, amik a viselkedésüket befolyásolják vagy a modell topológiát állítják be. Egyszerű modulok a legalsó szintjei a modell hierarchiának és C++ nyelven programozhatók.

Az OMNeT++ szimulációi futtathatók grafikus felhasználói felületről vagy parancssoros interfészről is, az első hasznos lehet prezentációnál és hibakeresésnél, a második pedig kötegelt végrehajtásnál.

A szimulációk fejlesztését az OMNeT++-ban egy integrált fejlesztő környezet (IDE, Integrated Development Environment) támogatja, aminek Eclipse platform az alapja, ami egy bővíthető Java alapú keretrendszer. Szükség esetén Java-ban írhatók modulok, amiket könnyen hozzá lehet adni az IDE-hez.[6]

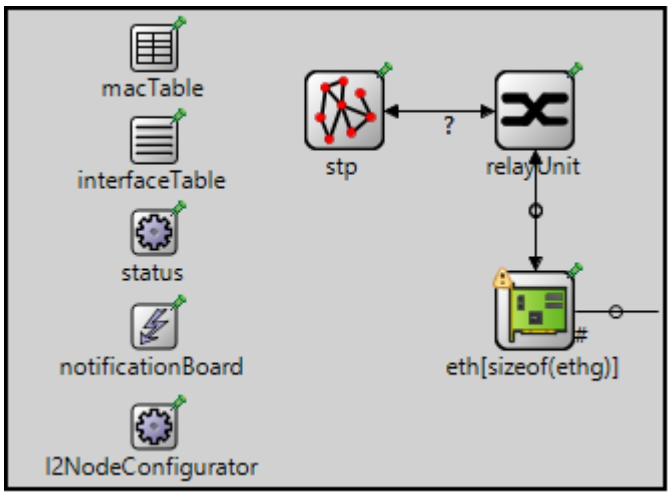
Az OMNeT++ egyik előnye még a hozzá kapcsolódó INET framework amelyben többek közt implementálva vannak a szimpla Ethernethez köthető modulok mind például:

- EthernetHost – egy forgalom generátor, amely képes második rétegbeli forgalmat generálni és fogadni.
- EthernetSwitch – egy kapcsoló, amely képes a portjai közötti csomagtovábbításra második rétegbeli információk alapján

- EthernetLink – Szabványos Ethernet vezetékek implementációi
- EthernetInterface – Ilyen interfész kártyákon keresztül kommunikálnak az egyes modulok, ha szigorúan akarjuk nézni akkor ezek a konkrét portok implementációi mivel kártyánként csak egy portot implementáltak.
- MAC – Az interfész kártyák modulja ez valósítja meg a fizikai rétegen keresztüli kommunikációt, számunkra ez a modul a legérdekesebb a dolgozat kapcsán.[7]

3.1. Kapcsolók a szimulátorban

A kapcsolók felépítése az 2. ábra látható.

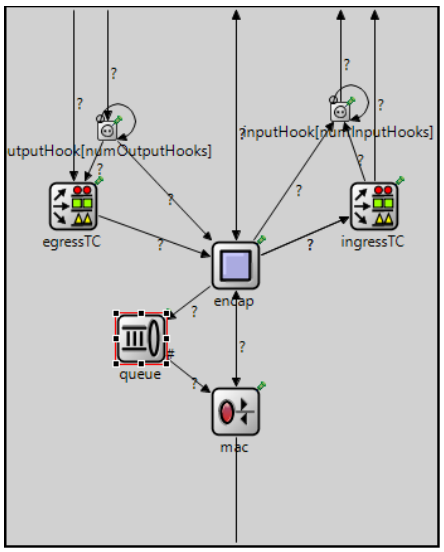


2. ábra - Kapcsoló felépítése[7]

A képen látható macTable bejegyzései szerint dönti el a relayUnit, hogy melyik eth<*> interfacere küldje a beérkezett csomagot (mi nem használjuk az STP-t). Az eth<*> mérete attól függ hány különböző kapcsolatot állítunk a kapcsolóra, így ezek az interfészek kvázi a portokat jelölik és ezekben a portokban lévő logikát módosítottuk a tanszéki projekt keretein belül.[7]

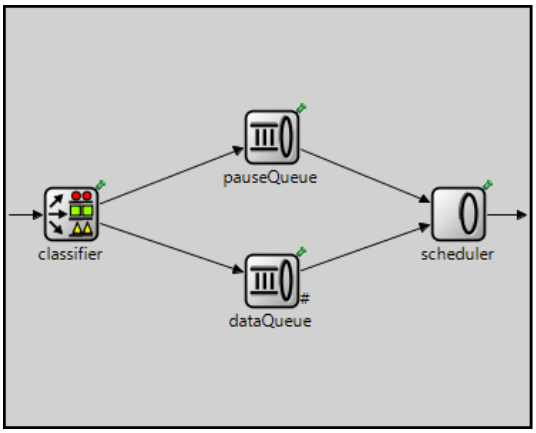
3.2. Szigorú prioritásos sorbanállás

Ezt a szimulátort az INET Frameworkkel kiegészítve használtuk egy tanszéki projekt kapcsán. Az INET kiegészítésben többek között implementálták a szabványos Ethernet funkciókat, például a számunkra fontos és az előző alfejezetből megismert Ethernet interfész implementációjának blokkvázlata látható az 3. ábra.



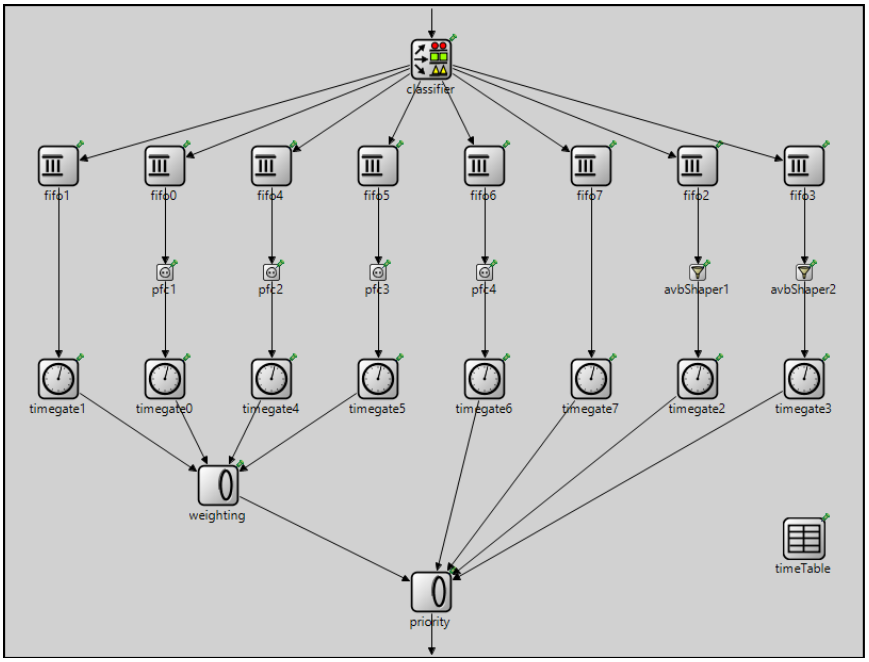
3. ábra - Ethernet interfész blokkvázlata[7]

A projektben belépésünkkor már elkezdtek fejleszteni a Ethernet TSN kiegészítéseit és a feladatom az volt, hogy ezeknek az új funkcióknak a validációs és integrációs tesztelését végezzem kezdetekben. Ide kapcsolódik a prioritásos sorbanállás implementációja is amely lehetővé teszi, hogy MAC cím alapján különböző első be-első ki (FIFO, First in-First Out) sorokba irányítsuk az Ethernet kereteket. Az Interfészkártya ábráján látható priossal bekeretezve a kimeneti sorok helye a blokkvázlaton ennek a belső felépítése látható az 4. ábra.



4. ábra - Kimeneti sorok belső felépítése[7]

Ezen felül pedig a dataQueue blokkban implementált 802.1p szabványnak megfelelő prioritásos FIFO sorok láthatóak az 5. ábra.



5. ábra - Prioritásos FIFO sorok

Minden sor között megállapítható prioritásbeli különbség. A szabvány szerint jobbról balra haladva a legmagasabb prioritása a 3-as míg a legalacsonyabb az 1-es FIFO sornak van. Látható továbbá, hogy a belépési ponton van egy rendező modul (classifier), ez a modul felelős azért, hogy a megfelelő mintára illeszkedő, esetünkben ez a cél fizikai cím, kereteket a megfelelő sorba juttassa. A sorokból opcionálisan pfc és avbShaper modulokon át mehetünk az időkapukhoz ezeket az opcionális köztes modulokat azonban nem implementáltuk mivel nem is állt szándékunkban használni őket. Az időkapuk implementálva vannak, azonban mindegyik úgy van felkonfigurálva, hogy folyamatosan átengedje a kereteket abból a szempontból viszont fontosak, hogy a MAC modultól ezek a kapuk kérhetik egy keret adásának megszakítását.

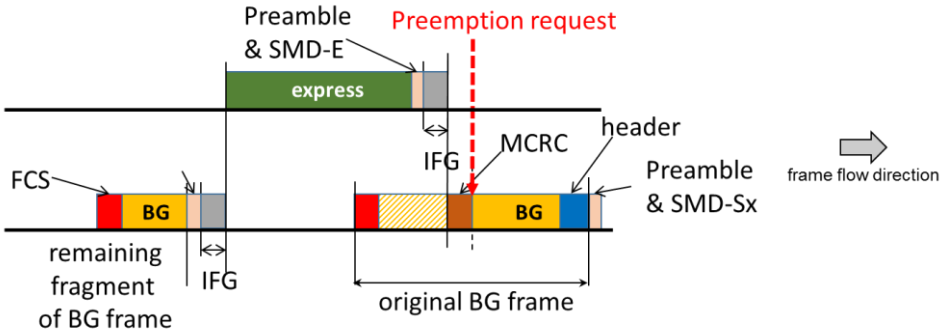
3.3. Keretmegszakítás

Ez a folyamat, ahol egy alacsonyabb prioritású forgalom keretre megszakítást kér egy beérkező magasabb prioritású beérkező forgalom kerete (azaz expressz keret) és azután folytathatja a küldést miután az expressz keret kiment. Már ismerős lehet a magasabb hálózati rétegekben illetve azok számára is, akik már foglalkoztak bármilyen ütemező működésével, az Ethernet világában azonban ez a módszer eddig egyáltalán nem volt jelen.

Most azonban, hogy bevezették érdemes megvizsgálni milyen előnyökkel járhat az Ethernet világában, ha nem csak prioritást tudunk felállítani egyes folyamok közt, de a prioritásos folyam kereteit előnyben is tudjuk részesíteni.

Az Ethernet keret megszakítás lényege, hogy a MAC modul több bemeneti FIFO sor közül tudja kiválasztani melyikből kerüljön kiszolgálás alá keret. Ezeknek a soroknak a prioritása alapján a megszakításra képes MAC modul mindig a legmagasabb prioritással rendelkező és éppen keretet várakoztató sorból választ keretet a továbbításra. Ha az adás alatt lévő keretnél magasabb prioritási sorba keret érkezik (expressz), akkor ez egy megszakítási kérést küld a MAC modulnak. A MAC modul erre az aktuálisan adás alatt lévő keret állapota szerint tud válaszolni különböző módokon.

Mielőtt azonban ezekre az esetekre kitérünk fontos megnézni, hogy a miben különbözik a fejléce egy szimpla Ethernet keretnek, egy expressz keretnek és egy megszakított keretnek. A szimpla Ethernet keret mezői kis mértékben meg lettek változtatva annak érdekében, hogy a megszakított keretek megfelelő kiszolgálását biztosítsák. Egy ilyen módosított és már feldarabolt keret látható az 6. ábra.[8]



6. ábra - Keret megszakítás[9]

A preamble változatlan maradt. A keret kezdet határoló (Start of Frame Delimiter, SMD) meg lett változtatva SMD-E és SMD-Sx-re, de a mezők funkcionalitása megmaradt a MAC értesítése egy keret elejének érkezéséről. Az SMD-E mondja meg a MAC-nek, hogy egy nem megszakítható expressz keretről van szó és az SMD-Sx jelzi a megszakítható háttér keret kezdetét. Ezután jönnek az forrás és cél fizikai címek (MAC DA/MAC SA) és az Ethernet típus (EtherType) mező. Aztán jön a hasznos adat (payload) és a záró keret ellenőrző sorozat (Frame Check Sequence, FCS).[8]

A feldarabolt keret töredékekben a preamble 6 byte hosszú, hogy legyen elég hely a darabolás számláló (Fragment Count) mezőnek, ami a MAC modulnak a töredék számát mondja meg. Az SMD-Cx hasonló a többi SMD-hez és a folytatható töredék kezdetét jelzi. Az MCRC egy nem végső rész CRC-je, a fogadónak jelzi, hogy a keret meg lett szakítva.

Ha eltekintünk a legrosszabb esettől, amikor az alacsonyabb prioritású folyam kerete nem tartalmaz annyi hasznos adatot, hogy megszakítható legyen, három esetet különböztethetünk meg.

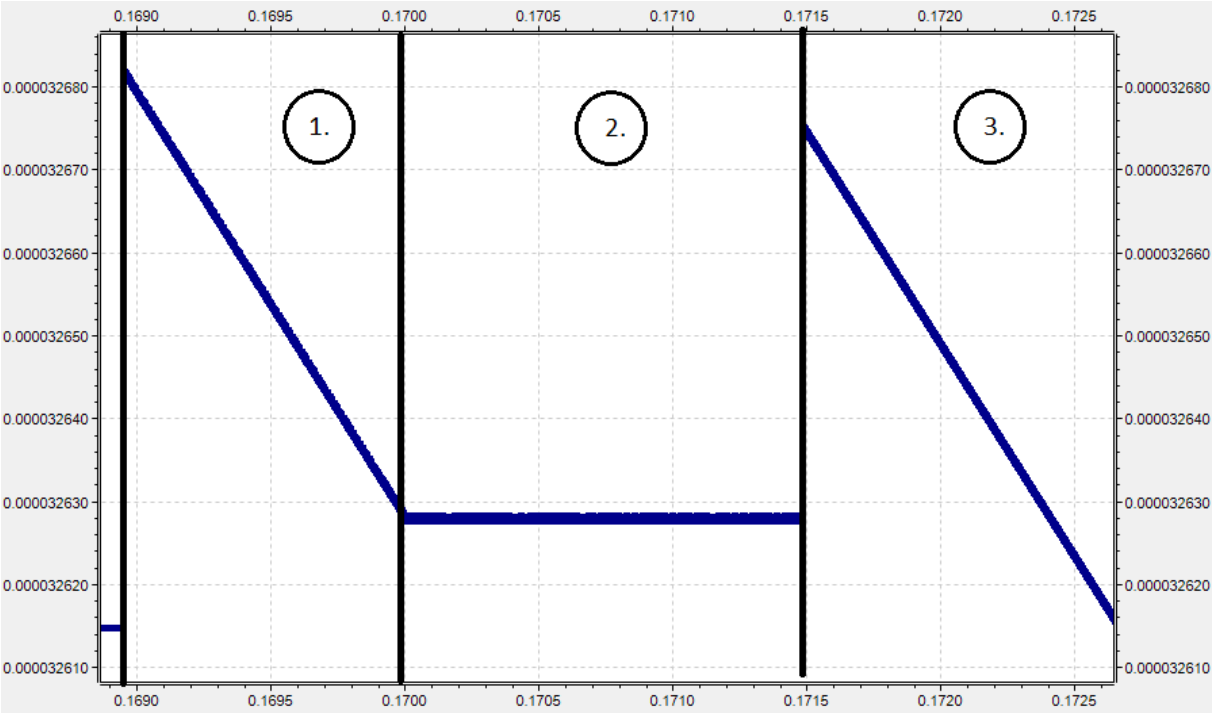
Ideális esetben azonnal elkezdhető a megszakítási folyamat, ezáltal az elszenvedett késleltetés a lehető legkisebb. A(z) 6. ábra - Keret megszakítás[9]n látható erre egy példa. Ezen az ábrán az alul látható a háttér (BG) keret idővonala, míg felül az expressz keret. A háttér keret idővonalán az látható, hogy miután továbbította a Preamble-t, az SMD-Sx-et és egy általános Ethernet fejlécet megkezdődik a hasznos adat továbbítása. Nagyjából a küldés felénél beérkezik egy expressz keret és egy megszakítási jelet küld a MAC modulnak, ami épp a háttér keretet továbbítja. A jel hatására a MAC modul ellenőrzi, hogy a jelenlegi keret megszakítható-e, azaz először ellenőrzi, hogy megszakítható típusú-e, amennyiben igen, akkor ellenőrzi, hogy van-e elég hasznos adat még a megszakítás végrehajtásához. Ha igen, akkor azonnal megkezdődik a megszakítás, vagyis a jelenleg kiküldés alatt álló oktett befejezése után a MAC modul hozzáfűz egy MCRC-t. A megszakítási folyamat befejezése után és az expressz keret kiküldése előtt ki kell várni egy kis időt, amit keretek közti szakadéknak (Inter-Frame Gap, IFG) hívnak, az IFG egy előre meghatározott 12 oktettnyi holt idő a keretek között a vonalon. Ezután már ki lehet küldeni az expressz keretet. A megszakított háttér keret küldése azután folytatható, miután az expressz keretet elküldték. Megjegyzendő, hogy két egymás után küldött keretet egy IFG-nek kell elválasztania, emiatt mielőtt a háttér keret küldését folytatni lehetne, egy IFG időt várni kell. Ezután egy preamble és SMD-Cx kerül az adat elé, azután küldhető. Ha egy újabb megszakítási jel érkezne, akkor a folyamat megismétlődne, de a mi esetünkben a keret küldése befejezhető és egy FCS-sel záródik.[8]

Amikor egy keretet megszakítunk, az első töredéknek legalább 64 byte hosszúnak kell lennie. Ez azt jelenti, ha egy háttér keretet épp elkezdtek kiküldeni és a megszakítási jel beérkezik, a feldarabolás folyamata nem tudja azonnal megállítani a továbbítást. Kénytelen várni, amíg a keretből 64 byte kiíródik a vonalra.

Egy másik eset, amikor nem maradt elég a háttér keretből, ahhoz hogy fel lehessen darabolni. Az expressz forgalomnak meg kell várni a küldés végét. A legrosszabb eset, amikor a MAC réteg épp elkezdte továbbítani a háttér keret utolsó 64 byte-ját (hasznos adat + FCS). Az expressz keretnek ilyenkor meg kell várni az IFG-t is, ami így 1 bit híján 76 byte (607 bit).

Az összefoglalása a keret megszakításnak egy 10 Gbps vonalon a 7. ábra látható. A 2. táblázatban megfigyelhető, hogy az első fázisban a késleltetés maximuma 67,2 ns, ez amiatt

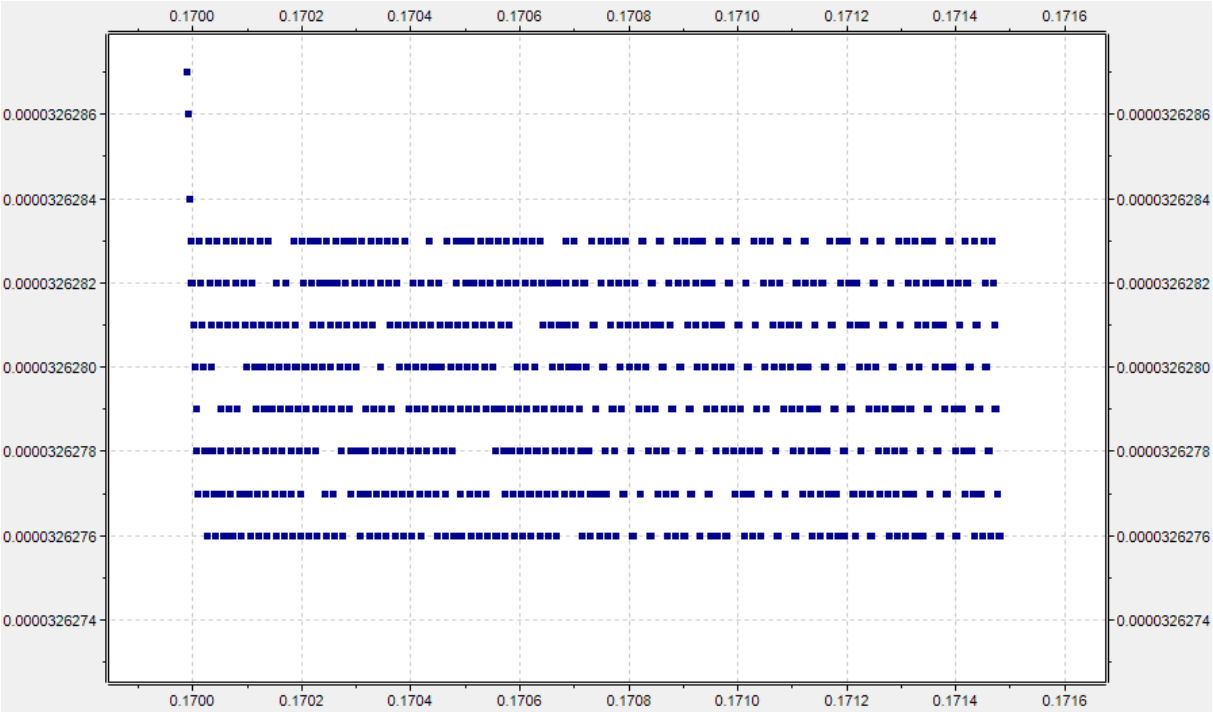
van, mert ennyi időbe telik egy 10 Gbps linken egy keretet összeállítani, ami kerettel együtt 84 byte és így már megszakítható. Ezután látható az azonnali megszakítás esete, amikor van elég hasznos adat a megszakítási kérés beérkezésekor. Ilyenkor a jel beérkezése és az expressz keret küldése közötti idő 12,8 ns és 13,5 ns tartományban mozog erről a fázisról látható egy ráközelített mérés a 8. ábra, attól függően mennyi van még a jelenleg kiküldés alatt álló oktettből, látható a 8 bitnek megfelelő 8 szint. Az utolsó eset, amikor nincs elég adat a megszakítható keretben. Ilyenkor az expressz keretnek meg kell várnia, amíg a háttér keret kiküldése befejeződik ez látható a harmadik fázisban. Itt a késletetés maximuma 60.7ns esetünkben. Ez azért van, mert ha nincs már 60 byte-nyi hasznos adat nem tudjuk elvágni a keret ezért meg kell várnunk, amíg az adása befejeződik.



7. ábra – Keretmegszakítás fázisai

Express	Min	Phase1	Phase2	Phase3
EndToEndDelay in us	32,6148	32,6820	32,6283	32,6755
FDV in ns		**67,20**	**13,50**	**60,70**

2. táblázat - Keretmegszakítás fázisainak eredményei



8. ábra - Keretmegszakítás 2. fázis ráközelítve

Belátható, hogy ez a mechanizmus hasznos lehet, ha egy hálózat üzemeltető szeretné teljes mértékben kihasználni a hálózatot azáltal, hogy engedélyezi a hálózat használatát best effort jellegű forgalom számára. Ezért érdemes lehet a keret megszakításos adástovábbítás további felhasználási lehetőségeit is tanulmányozni.

4. Szimulációs beállítások

A 3. fejezetben tárgyalt szimulációs eszközben végeztük a TSN mechanizmusok CPRI forgalmakra gyakorolt hatásának mérését. Az általános fronthaul hálózatot egy kapcsolt Ethernet hálózattal modellezzük, amit több CPRI forrással és háttér forgalommal töltöttük fel, amik egymással kölcsönhatásba kerülnek. Ebben a fejezetben a hálózati topológiát, a csomópontok architektúráját, a forgalom forrásainak és a kapcsolók modelljeit mutatjuk be.

4.1. Csomópontok

4.1.1. Forgalom generátor

A hosztok felépítése roppant egyszerű, mindössze csomagokat küldő forgalom generátorokra és csomagokat fogadó nyelőkre bontható szét.

A csomagokat küldő forgalom generátorok egy egyszerű második rétegbeli csomag generálásra képes alkalmazást tartalmaznak melyeknek a cél cím „címkéje”, a küldendő Ethernet csomag mérete és a csomagküldés gyakoriságának megadása szükséges a forgalomgeneráláshoz. Opcionálisan megadható nekik csomaggenerálási hiba, börsztös csomagküldés illetve a csomagküldési gyakoriság bizonytalanná tétele (időben ingadoztatása). Az esetünkben érdemes beállítani a fizikai címeket is, hogy a megfelelő sorokba kerülhessenek az egyes generált folyamok keretei, ugyanis az osztályozás ez alapján történik.

A csomagokat fogadó forgalom generátorokat nyelőknek nevezzük és felépítésükben megegyeznek a forgalom generátorokkal csak máshogyan vannak beállítva. Elég csupán egyetlen csomagot küldetni velük a szimuláció elején, hogy a kapcsolók megtanulják a topológiát és ne broadcast üzemmódban működjenek.

A szimulátorban egy forgalom generátor küldésre történő beállítása tipikusan a következő képpen történik:

```
**.forgalom_generátor_neve.app.startTime = <csomaggenerálás_kezdete>
**.forgalom_generátor_neve.sendInterval = <csomaggenerálás_gyakorisága>
**.forgalom_generátor_neve.app.numPacketsPerBurst = <hány_csomag/börszt>
**.forgalom_generátor_neve.app.packetLength = <payload_mérete*>
```

*A headereket a szimulátor automatikusan hozzászámolja a keretekhez és csomagokhoz.

4.1.2. Kapcsolók

Miután megvizsgáltunk különböző kapcsolókat a nagy sávszélességű környezet miatt úgy döntöttünk, minden kapcsoló store-and-forward architektúra szerint működik, ami azt jelenti, hogy megvárja míg egy keret teljesen be nem érkezik és csak azután adja ki a vonalra.

Az alternatíva a cut-through keret továbbítás lenne [12], amikor csak néhány byte-ot értelmeznek a fejlécből, annak alapján meghatározzák a kimeneti portot, azután pedig az éppen beolvasott keret tartalmát byte-onként továbbítják is a megfelelő kimenetre.

A store-and-forward megoldás nagyobb késleltetést okoz, de a cut-through hátránya, hogy a hibás keretet nem képes kiszűrni (hiszen már továbbküldték azt, mielőtt az ellenőrző kódot kiszámolhatnák). A jelenlegi megoldások jelentős része a klasszikus store-and-forward eljárást alkalmazza, mi is ezt használtuk. Továbbá a tanszéki kollégákkal történt konzultálások után úgy döntöttünk minden kapcsoló dolgozzon 1.5µs kapcsolási késleltetéssel aminek ±5ns legyen az ingadozása. [10]

A kapcsoló paramétereit a szimulátorban a következőképpen állíthatjuk be:

```
**.switch?.eth[*].numInputHooks = <belépő_modulok_száma>
**.inputHook[0].typename = <belépő_modul_típusa(“delayer”)>
**.inputHook[0].delay = <késleltetés mértéke>
**.switch?.*.queueType = <sorok_típusa>
```

4.2. Topológiák

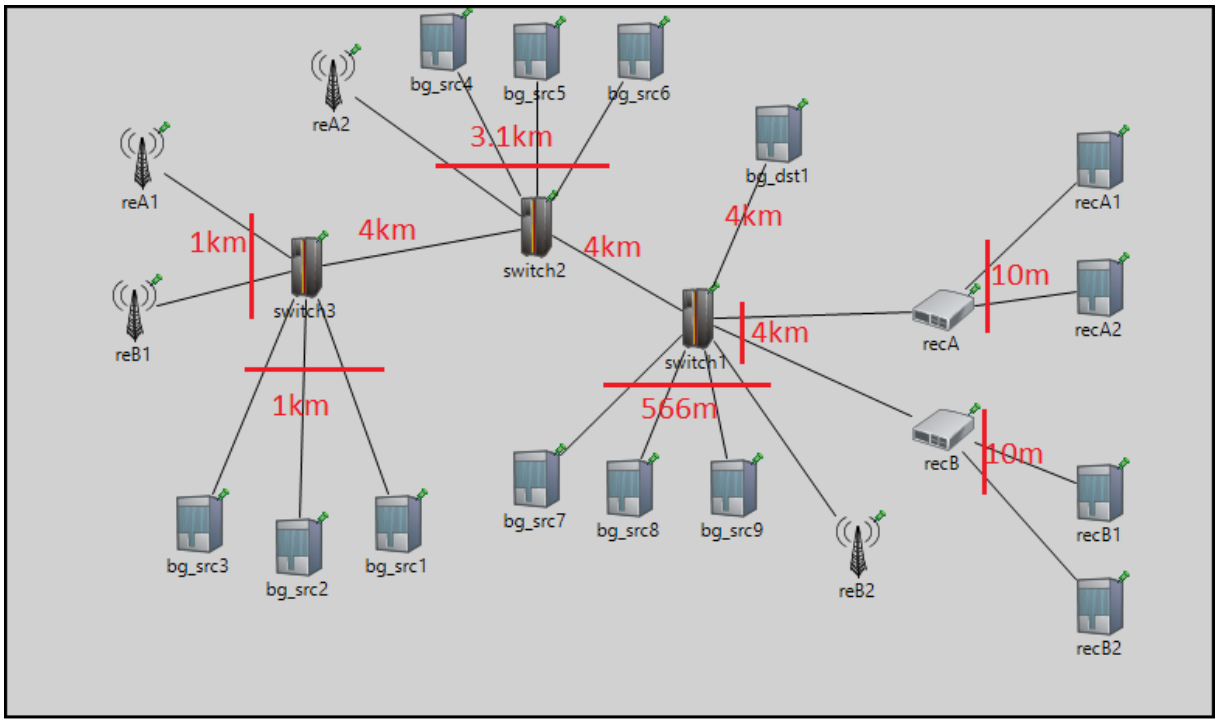
A fronthaul hálózat topológia modellje négy rádiós eszközből, négy rádiós eszköz vezérlőből, kilenc háttérforgalom generátorból és egy háttérforgalom végződésből áll. A különböző topológiáknál megpróbáltunk főleg városi környezetben előforduló elrendezéseket és méretezéseket használni. Lehet, hogy a négy rádiós eszköz kevésnek tűnik, de a mi célunk az volt, hogy a szimulációs eredményeket magyarázhassuk és bemutathassuk. Az eredmények értelmezése során (debugging) a folyamok közti bonyolultabb interakció sorozatot már három folyam és két ugrás után is sok nehézséget okoz. Ugyanakkor a sorozatos, kapcsolóként növekvő potenciális ütközési helyzetek képesek a gyakorlatban felmerülő hibákat reprodukálni.

Minden összeköttetés 10 Gbps Ethernet kapcsolat, amiken a terjedési késleltetés 5000 ns kilométerenként és 1 bit sorosítás ideje 0,1 ns. Ezekkel az összeköttetés hosszakkal egy sűrű városi környezetet kívánunk modellezni. A rádiós eszközök és az ezekhez csatlakozó kapcsolókhoz tartozó vezetékeinek minimális hossza attól függ hogyan lesz pontosan annyi a terjedésük miatti késleltetésük, hogy ütközzenek hasonló prioritású kerettel amikor belépnek a gerinchálózatra. Az belépési vezetékek hosszát minden topológiánál jelezzük.

4.2.1. Topológia1

Ennél a hálózati topológiánál a motiváció egy külvárosi fejhálózat modellezése volt emiatt az össz terjesztési távolság ~20km-re lett választva. A pontos adatok pedig a következők lettek:

- A gerinchálózat három ugrás széles, ami három kapcsolóból áll.
- A topológián további két csomópont látható (recA és recB), ezek a kapcsolók 0 ns kapcsolási késleltetésre vannak beállítva, a forgalmak egymás zavarása nélkül mennek át rajtuk és rádiós eszköz vezérlők modellezésére használjuk, amik általában több rádiós eszközt szolgálnak ki. A recA1, recA2, recB1, recB2 csomópontoknak csak technikai szerepük van, arra használjuk, hogy az expressz forgalmak kereteinek a statisztikáit külön tudjuk vizsgálni.
- A gerinc kapcsolók és a vezérlőt modellező kapcsolók közti linkek 4km hosszúra lettek választva. Ezt nevezzük gerinchálózatnak.
- A hálózatban kilenc háttérforgalom generátor található gerinc kapcsolónként 3-3.
- Négy expressz forgalom generátort helyeztünk el a hálózatban. Ezek a reA1, reA2, reB1 és reB2 ezeknek az 9. ábra látszik a belépési pontja
- További információ az egyes vezetékek hossza, ezeket is a 9. ábra lehet megtudni minden vezetéknél egyértelműen jelezzük.
- Ezt a topológiát csak 1500 byte-os expressz forgalommal használtuk.



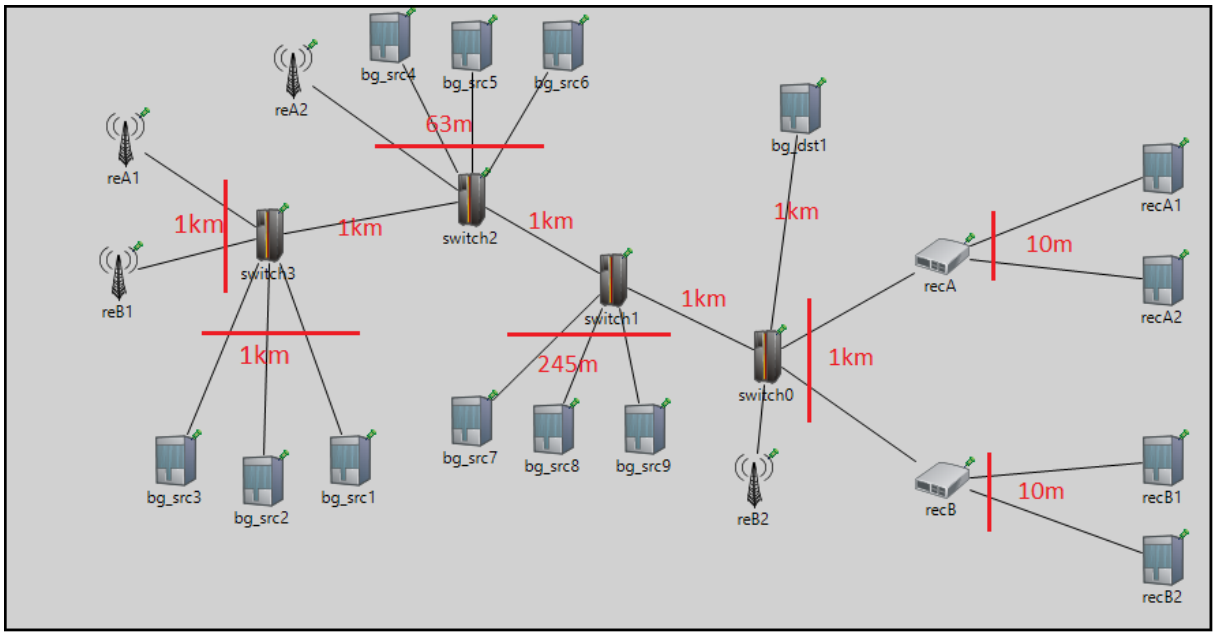
9. ábra - Első topológia

4.2.2. Topológia2

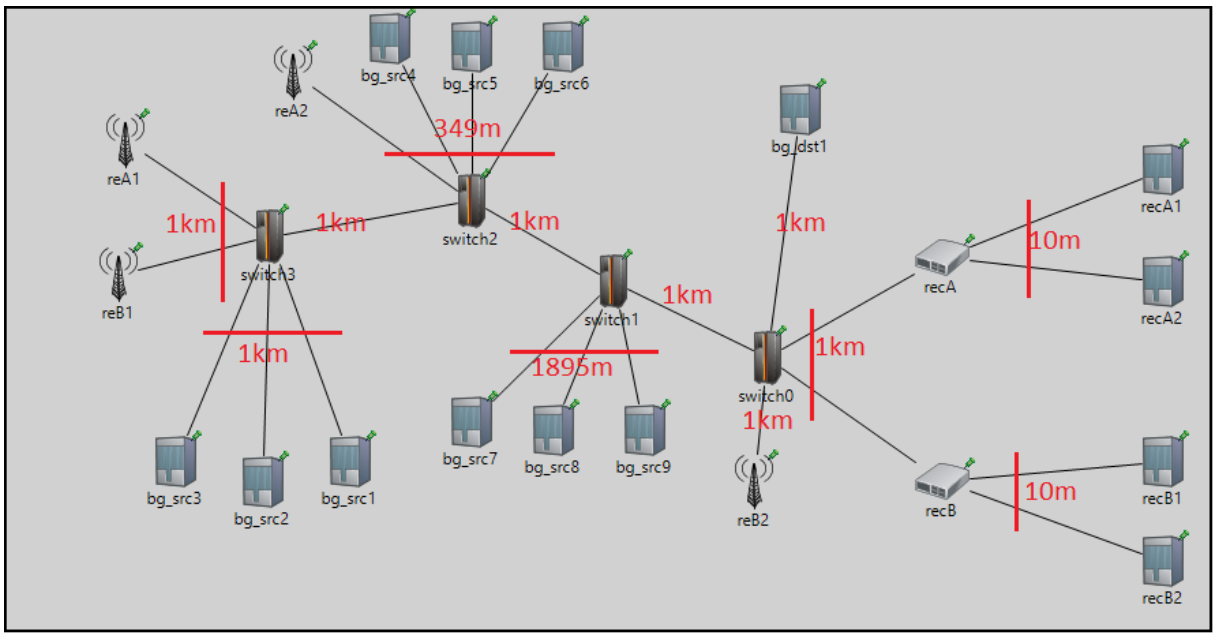
Ennél a hálózatnál egy frekventáltabb belvárosi környezet modellezése volt a cél ezért az előzőhöz képest lényegi eltérések a következők:

- A vezetékek rövidebbek

- Az egyes topológia „switch1” gerinckapcsolóját felbontottuk két gerinckapcsolóra ezek lettek a „switch0” és „switch1”.
- A gerinchálózat ebben az esetben a switch3-switch2-switch1-switch0-recX vonalat jelöli ezen a vonalon minden vezeték 1km hosszú.
- A gerinckapcsoló szétszedése után a háttérfolyamok az 1-esben míg az expressz folyam a 0-ásban lép be a hálózatra. Emiatt több az ütközési lehetőség a reA1, reA2 és reB1 által generált folyamok és az 1-esben belépő zavaró folyamok közt mint az előző topológiában.
- Ebben a topológiában már használtuk a 300 és az 1500 byteos expressz folyamokat is. Emiatt két verziója van a topológiáknak, hogy mindkét esetben legyen az expressz keretek közt ütközés.
- Az 10. ábra látható a 300 byteos verzió, míg az 11. ábra az 1500 byte-os verzió.



10. ábra - Topológia 300 byteos verziója



11. ábra - Topológia 1500 byteos verziója

4.3. Forgalom karakterisztikák

Minden egyes szimulációs kísérletnél a kapcsolóknak meg kell tudniuk a helyi fizikai címeket. Minden szimuláció kezdetén a források egy keretet küldenek ehhez a tanulási folyamathoz. Ezeket jóval az expressz forgalmak és a háttérforgalmak indítása előtt küldik, hogy ne zavarják meg a kísérletet. Ez az oka, hogy soha nem küldünk szimulációs idő szerint 0,1 másodperc előtt se expressz, se háttér keretet.

4.3.1. Háttérforgalom

A háttérforgalmakat a források generálják (bg_src*), és a célállomás a bg_dst fogadja. A háttérforgalom különböző sebességű lehet, amit a szcenárióknál jelzünk.

A 3. táblázatban a kísérletekben használt általános háttérforgalmi paraméterek láthatók.

Tulajdonság	Érték
Kezdési idő	0.105s
Keret méret	1500Byte
Sorosítás késleltetés	1230.4 ns
Átlag sávszélesség	1.1278Gbps

3. táblázat - Háttérforgalmi paraméterek

4.3.2. Express forgalom

Az expressz forgalom jelen esetben a már korábban említett CPRI folyamokat hivatott modellezni. Jelenleg ezek dedikált pont-pont összeköttetéseken továbbítódnak és nem egy elosztott hálózaton. Mi viszont arra törekszünk, hogy a második megvalósuljon, és ehhez szükség volt arra, hogy megtaláljuk a megfelelő keretméretet, amivel ki lehet küldeni egy Ethernet alapú hálózatra. Mivel erre nincs pontos követelmény vagy konkrét irányelv, ezért

egy ipari partnerrel konzultáltunk ezzel kapcsolatban. Arra jutottunk, hogy két különböző keretméretű forgalmat érdemes vizsgálni, a 300 byte-os és az 1500 byte-os méretűt.

A motiváció a különböző méretek között a következő:

- 300 byte még elegendő méretű adatot tud szállítani elfogadható mértékű overhead mellett és a kis keretméret miatt kevésbé érzékeny a keretvesztésre.
- 1500 byte maximális méret, itt az elsődleges cél a minél több adat szállítása egy keretben.

4.3.3. Hálózati profilok

A szabványosítási folyamat során két osztályba sorolták a hálózati profilokat, ezek a Profil A és Profil B elnevezéssel hivatkoznak, mi is ezt a konvenciót használjuk a dolgozatban. A fő különbség a kettő között az alkalmazott TSN mechanizmusokban van. A keretek méretei a korábban említettek szerint alakulnak,

A Profilok jellemzőinek összefoglalását tartalmazza a 4. táblázat.

	Profil A	**Profil B**
Priorításos sorok	Van	Van
TSN funkciók	Nem specifikált	Keretmegszakítás

4. táblázat - Profilok jellemzőinek összefoglalása

Ebből következik, hogy a Profil A esetén akár egy 1500 Byte méretű zavaró keretet miatti várakozás is előfordulhat, ami az abszolút legrosszabb esetet jelenti, míg a Profil B esetén legrosszabb esetben a legkisebb megszakíthatatlan keretet kell végigvárni, ami 123 Byte. [11]

5. Szimulációs eredmények

A következőkben azokat a szimulációs eredményeket kívánjuk bemutatni melyek érdekesnek bizonyultak azzal kapcsolatban, hogy milyen konklúziók vonhatóak le a CPRI forgalomhoz választott keretmérettel illetve az általa támasztott követelmények kielégítéséhez szükséges hálózati elemekről.

A mérések során három szcenáriót állítottunk fel ezek beállításairól az 5. táblázatban található több információ.

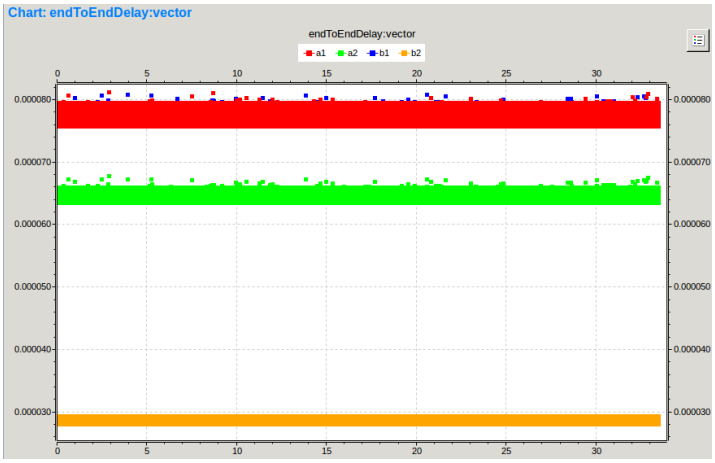
	Szcenário 1	Szcenário 2	Szcenário 3
Topológia	1	2	2
Express Keretméret	1500	1500	300
Profil A	igen	igen	igen
Profil B	igen	igen	igen

5. táblázat - Szcenáriók beállítása

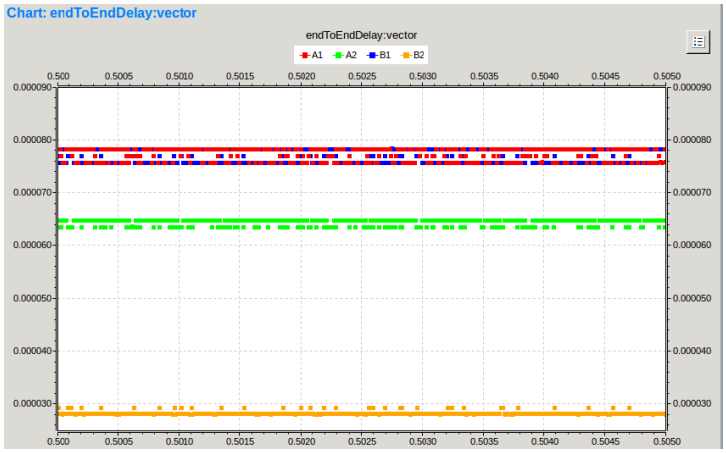
5.1. Teszt1: Szcenárió 1

Ennél a szkenriónál az egyes topológiát illetve az 1500 byteos expressz keretméretet használtuk, a motiváció az volt, hogy megnézzük hosszú linkeknél milyen értékeket mérhetünk nagy keretméret esetén, teljesíteni tudja-e ez a konfiguráció a CPRI követelményeket. Ezt a vizsgálatot azért is mutatjuk be, mert viszonyítási alapnak használjuk a következő topológiához.

Az 12. ábra látható a Profil A és az 13. ábra a Profil B futásának eredményeként kirajzolt end-to-end késleltetést mutató diagramm. Valamint az 6. táblázatban látható az end-to-end késleltetésre és a késletetés ingadozásra kimért számértékek összehasonlítása.



12. ábra – Teszt 1 Profil A eredmény



13. ábra – Teszt1 Profil B eredmény

	Profil A		Profil B		
Flow	**Max késleltetés [ns]**	**FDV [ns]**	**Max késleltetés [ns]**	**FDV [ns]**	**Késleltetés különbség**
A1	81148,9	5509,6	78289,5	2649,4	2859,4
B1	80716,9	5076,8	78291,5	2651,4	2425,4
A2	67691,2	4252,3	64789,3	1350,4	2901,9
B2	29329,6	1291	29329,6	1291	0

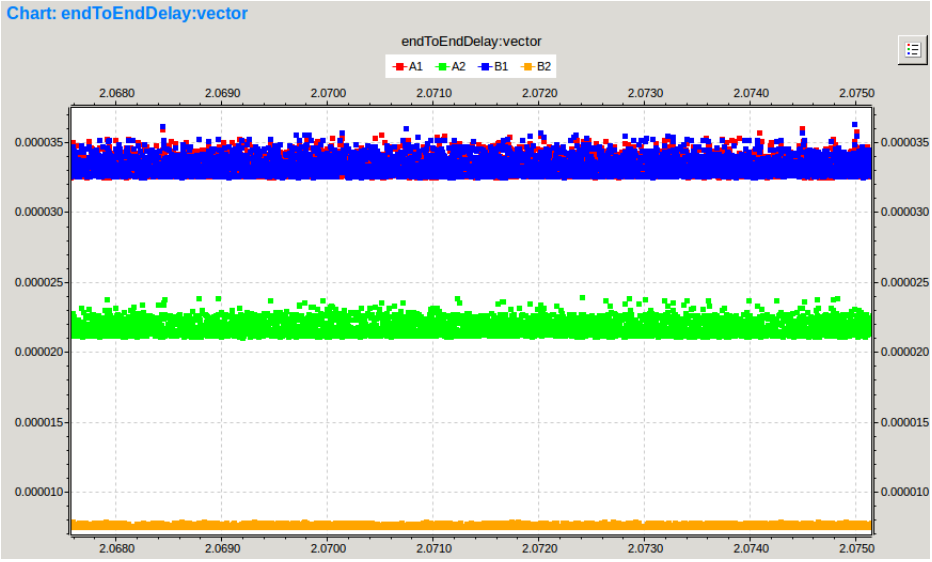
6. táblázat – Teszt1 Profil A és B eredményei

Egyből látható a keret megszakítás előnye a táblázatból. Ezzel a topológiával a legnagyobb nyereség mértéke ezzel a méréssel ~2900ns ami 580m 10Gbps Ethernet vezetéknek terjedési késleltetésnek megfelelő érték. Továbbá az ábrákon az is látszódik, hogy bár az FDV mindössze átlagosan 55,6%-al csökkent, de a forgalom karakterisztikája sokkal kiszámíthatóbb a B profil esetén.

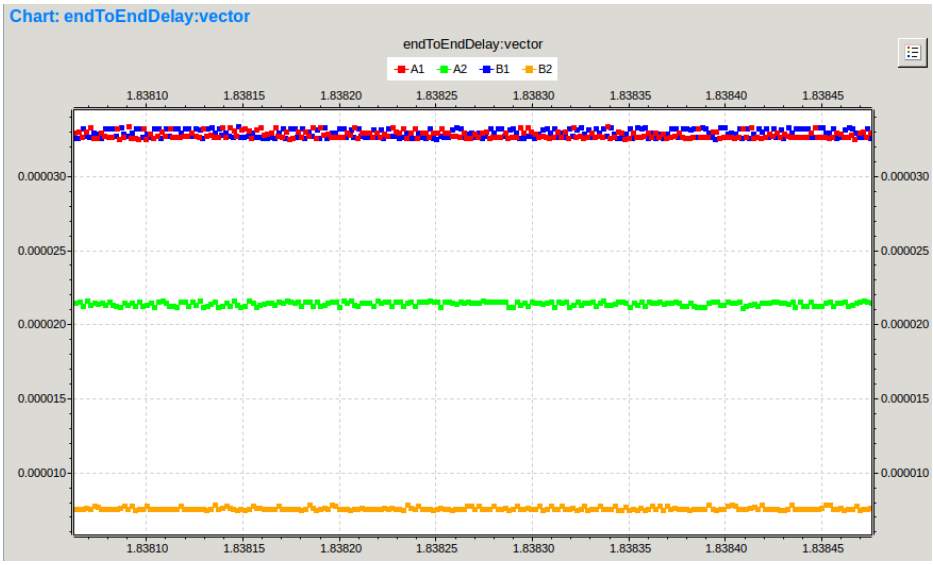
5.2. Teszt2: Szcenário 3

Erre a tesztre azért van szükség, hogy össze tudjuk hasonlítani a topológiabeli különbségeket. Míg az egyes topológia egy egymástól távolabb elhelyezkedő cellákat feltételez, ez egy sűrűbb elrendezés ezért érdekes lehet a terjedés illetve a keretméret miatt bekövetkező késleltetés arányának változása. Az itt használt expressz keretméret 300 byte.

A tesztek eredményei az A hálózati profil esetén a 14. ábran láthatóak, míg a B hálózati profil esetén a 15. ábra tartalmazza azt.



14. ábra –Teszt 2 Profil A eredmény



15. ábra – Teszt 2 Profil B eredmények

	Profil A		Profil B		
Flow	**Max késleltetés [ns]**	**FDV [ns]**	**Max késleltetés [ns]**	**FDV [ns]**	**Késletetési különbség**
A1	36573,1	4144,3	33478,6	1049,2	3094,5
B1	36603,4	4181,3	33491,3	1062,1	3112,1
A2	24022,4	2995,8	21747,9	713	2274,5
B2	7944,9	482,6	7870,5	408,3	74,4

7. táblázat – Teszt 2 Profil A és B eredmények

Az eredményeken látszik, hogy rövidebb hálózatnál a nagy háttér folyam keretekkel való ütközés behatása sokkal nagyobb. Az A profil esetén a késleltetés akár több mint 10%-ért

felelős ezért itt sokkal nagyobb hatása van a keretmegszakításnak, mint az egyes topológia hosszabb kevesebb ugrást tartalmazó hálózatban.

5.3. Teszt3: Scenario2 Vs. Scenario3

Erre a tesztre azért volt szükség, hogy össze tudjuk hasonlítani a kisebb és nagyobb expressz keretméret hatását a hálózaton. A tesztek eredménye és a késleltetés értékek összehasonlítása az 8. táblázatban látható.

		SZCENÁRIÓ 2(1500B)		SZCENÁRIÓ 3(300B)		
PROFIL A	**Flow**	**Max késleltetés [ns]**	**FDV [ns]**	**Max késleltetés [ns]**	**FDV [ns]**	**Különbség [ns]**
	A1	44535,4	6177,4	36475,2	3853,5	**2323,9**
	A2	31135,7	3738,6	23922	2733	**1005,6**
PROFIL B	**Flow**	**Max késleltetés [ns]**	**FDV [ns]**	**Max késleltetés [ns]**	**FDV [ns]**	
	A1	41035,8	2673,4	33329,1	716,2	**1957,2**
	A2	28795,1	1397,9	21592,5	423,3	**974,6**

8. táblázat – Teszt 3 Szcenárió 2 és 3 eredményei

Látható az eredményeken, hogy a kisebb keretmérettel sokkal alacsonyabb maximális késleltetés érhető el mint abban az esetben ha maximálisan kihasználjuk a hasznos adatra vonatkozó felső korlátot. Ennek az az oka, hogy a store-and-forward művelet miatt ugrásonként a keret méretével arányos értékkel növekszik a teljes késleltetés (end-to-end delay).

5.4. Konklúzió

Az eredmények alapján az expresszfolyamra a következő konklúziókat vonhatjuk le.

A CPRI-t szállító Ethernet kereteknek mindenképpen a legmagasabb prioritású sorba kell kerülniük és így már a szimpla Ethernet technológiákkal is csökkenthető a zavaró háttérforgalmak által okozott késleltetés növekedés, mert maximum egy háttérforgalomhoz tartozó keret adásának a megvárása szükséges ugrásonként.

A keretmegszakítás alkalmazása tovább csökkentheti a késleltetést illetve annak ingadozását és ezáltal a tervezett csomagkapcsolt hálózat átmérője nagyobb lehet mint alapesetben.

Láttuk továbbá azt is, hogy a kisebb 300 byteos Ethernet keret választása preferált a CPRI keretek szállítására, mivel így minden ugrásnál kevesebb a keret vonalra írásának költsége és így tárol-továbbít (Store-and-forward, SF) architektúra esetén szintén nagyobb lehet a hálózat átmérője. Valamint az egymással a kapcsolókban a gerinchálózatra történő belépésnél ütköző CPRI keretek miatti késleltetés is csökken így emiatt is nagyobbra lehet

méretezni a hálózatot mint abban az esetben, ha a CPRI forgalmat 1500 bytos Ethernet keretek szállítják.

6. Összefoglalás

Egyre jobban érezhető a hálózatok sávszélességének és átmérőinek növekedésével az igény, hogy a kerettovábbítás késleltetését és interferenciáját csökkenteni tudják csomagkapcsolt hálózatokon. A szimpla Ethernetről belátható hogy ezeket az időérzékeny igényeket nem tudja kiszolgálni.

A TSN munkacsoport munkásságának egyik már láthatóan működő eredménye a keretmegszakítás egy hatékony eszköz arra, hogy a már létező és elterjedt Ethernet technológia ezzel kiegészítve egy alternatívát tudjon biztosítani a dedikált pont-pont kapcsolatokkal szemben, és hogy megoldást nyújtson a különböző ipari, autó ipari és fronthaul hálózatok multiplexálási problémáira.

A tanszéki munkacsoport folytatja munkáját a továbbiakban a TSN munkacsoport támogatásával, az általuk fejlesztett technológiák kutatásával, implementálásával és tesztelésével. Ezek az új technológiák még tovább javíthatják az Ethernet hálózatok QoS szolgáltatási képességeit.

A jelen dolgozat eredményeiből látható, hogy a szimpla Ethernet karakterisztikáinak odafigyelő megválasztása is elengedhetetlen a nemkívánatos többlet késleltetés elkerülésének érdekében. Az is látszott, hogy elengedhetetlen az Ethernet hálózatok működésével kapcsolatos átfogó ismeretek megléte a hálózatok tervezésekor.

De a legfontosabb üzenet, hogy ezek gondos megválasztásával nyert késleltetés is mind kevés még ahhoz, hogy teljesítsük a CPRI követelményeke. A keretmegszakítással kiegészített Ethernet azonban már képes a CPRI követelmények tartományain belül tartani a forgalmat.

7. Köszönetnyilvánítás

Ezúton szeretnénk megköszönni a Budapesti Műszaki és Gazdaságtudományi Egyetem Távközlési és Médiainformatikai Tanszék és az Ericcson Magyarország Kft. TSN kutatási projekten dolgozó szakembereinek. Névszerint: Dr. Bíró József, Moldován István, Máté Miklós, Dr. Farkas János, Dr. Varga Balázs. Az ő közreműködésük és munkájuk nélkül, a munkánk nem készülhetett volna el.

Szeretnénk kiemelni Máté Miklós munkáját a TSN szabványok szimulációs környezetben való megvalósításáért, a szimulátor kiegészítése és a szimulációs vizsgálataink megvalósításához szükséges hibaelhárítási folyamatok alatt nyújtott tanácsaiért.

Irodalomjegyzék

[1] Time-Sensitive Networking Task Group description, http://www.ieee802.org/1/pages/tsn.html

[2] CPRI Specification V7.0 (2015-10), Common Public Radio Interface (CPRI), Interface Specification, http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf (2016-10-09)

[3] CPRI over Ethernet, Nigel Bragg (2016-01), http://www.ieee802.org/1/files/public/docs2016/cm-bragg-CPRI-over-switched-Ethernet-0116-v01.pdf, 2016-10-24

[4] Ethernet frame expansion, Glenn Parsons, July 2006, http://www.ieee802.org/3/as/public/0607/802.3as_overview.pdf (2016-09-24)

[5] 802.1Qbu Frame Preemption, http://www.ieee802.org/1/pages/802.1bu.html (lastmodified: 2015-10-7)

[6] OMNeT++ simulation manual (version 5.0), https://omnetpp.org/doc/omnetpp/manual/

[7] INET reference (version 3.4.0), https://omnetpp.org/doc/inet/api-current/neddoc/r.html

[8] 802.1Qbu Frame Preemption, http://www.ieee802.org/1/pages/802.1bu.html (lastmodified: 2015-10-7)

[9] P802.1CM Time-Sensitive Networking for Fronthaul, Farkas János, 2015-11, http://www.ieee802.org/1/files/public/docs2015/cm-farkas-intro-1115-v03.pdf, 2016-10

[10] Measurement of Switching Latency in High Data Rate Ethernet Networks, T. Hehr, M. Voznak, M. Kozak, L. Bohac, (2015-03), http://www.eejournal.ktu.lt/index.php/elt/article/view/10445/6873 - 2016-10

[11] Profiles A and B, Farkas János, 2016-03, http://www.ieee802.org/1/files/public/docs2016/cm-farkas-profiles-A-and-B-0316-v01.pdf, 2016-10

[12] Felderman, R. E., Kulawik, A. E., Seitz, C. L., Seizovic, J., & Su, W. K. (1995). Myrinet: A gigabit-per-second local area network. IEEE micro, (February), 29-36.

Ábrajegyzék

Táblázatjegyzék