74.433/KOT

KIVONAT

Keret és protokoll minősítő eljárás és rendszer

Keret protokoll minősítő és feldolgozó rendszer és eljárás adatot feldolgozó rendszerben (például adatcsomagok vagy keretek kapcsolása illetve forgalomirányítása). Jelen találmányban előre meghatározott tesztek alapján analizáljuk a keretnek egy részét, majd a keret elkövetkezendő feldolgozásához szükséges főbb jellemzőket letároljuk. A keret (vagy bemeneti információs egység) főbb jellemzői a keretben használt 3. rétegbeli protokoll típusa, a 2. rétegbeli egységbezárási technika, a kezdő utasításcím, a látszólagos helyi hálózat használatát indikáló jelzőbitek, és annak az adatfolyamnak az azonosítója, amely adatfolyamhoz a keret tartozik. Az analízis legnagyobb részét hardver eszközök használatával valósítjuk meg, így az analízis gyorsan, uniform időben elvégezhető. Majd a keret letárolt jellemzőit a feldolgozó komplexumban felhasználjuk a keret feldolgozása során. A processzort prekondícionáljuk a kezdő utasítás címével, a 3. rétegbeli fejrész helyével és a keret típusának jelzőbitjeivel. Így, a processzor a felhasznált utasításcím vagy kód belépési pont alapján a megfelelő helyen, a keret típusa alapján kezdi meg a keret feldolgozását. A járulékos vizsgálatok és elágazási utasítások elkerülése végett az elágazásoknál további utasításcímeket helyezhetünk el a veremben. Ráadásul, az egy adatfolyamot alkotó kereteket az érkezésük sorrendjében továbbítjuk és dolgozzuk fel.

S. B. G. & K.
Szabadalmi Ügyvivői Iroda
H-1062 Budapest, Andrássy út 113.
Telefon: 461-1000, Fax: 461-1099

74.433/KOT

Keret és protokoll minősítő eljárás és rendszer

A találmány tárgya

Jelen találmány a kommunikációs hálózati berendezéseknek abba a csoportjába tartozik, amelyeket különböző típusú és képességekkel rendelkező infromációkezelő rendszerek vagy számítógépek összekapcsolására használunk, míg a berendezés komponenseinek és eljárásainak alkalmazásával adatot dolgozunk fel. Pontosabban, jelen találmány egy, adatátviteli hálózathoz kapcsolt feldolgozóegységek adatfolyamának irányításához szükséges eljárás és rendszer; amely eljárás és rendszer számos bemeneti információs egység (más néven „csomag" vagy „keret") kezelésére és - több független, párhuzamosan működő processzor segítségével - feldolgozására képes úgy, hogy ezen bemeneti információs egység protokolljára nincsen megkötés (egy protokoll a sok lehetséges közül).

A találmány háttere

Jelen találmány szorosan kapcsolódik az alábbi dokumentumokhoz, amelyeknek jogait átengedték a jelen találmány jogos birtokosa számára.

A Brian Bass és társai által 1999. augusztus 27-én iktatott, S.N. 09/384,691 szabadalmi számú, az „Összetett

Eljárásokat Feldolgozó Hálózati Processzor" („Network Processor Processing Complex and Methods") címet viselő szabadalom, amelyre jelen leírásban mint a Hálózati Feldolgozó Egység Szabadalomra - röviden NPU-ra (Network Processing Unit - hálózati feldolgozó egység) - fogunk hivatkozni.

Az 1998. március 3-án kiadott, 5,724,348 számú, „Hatékony Hardver/Szoftver Interfész Adatkapcsoláshoz" („Efficient Hardware/Software Interface for a Data Switch") címet viselő amerikai szabadalom, amelyre jelen leírásban mint az Interfész Szabadalomra fogunk hivatkozni.

Az 1999. június 11-én iktatott, S.N. 09/330,968 számú, „Nagysebességű Párhuzamos/Soros Adatkommunikációs Link" („High Speed Parallel/Serial Link for Data Communikcations") címet viselő szabadalom, amelyre jelen leírásban mint a Link Szabadalomra fogunk hivatkozni.

Számos, az IBM-nek jegyzett szabadalom és alkalmazás, amelyeknek témái a többprotokollos kapcsolási szolgáltatások - amelyekre MSS-ként hivatkoznak (MSS: multiprotocol switching services - többprotokollos kapcsolási szolgáltatások) -, amelyek közül néhánynak a feltalálója Cedric Alexander; ezen szabadalmakra jelen leírásban mint az MSS Szabadalmakra fogunk hivatkozni.

Jelen találmány most következő leírása során előzetesen feltételezzük, hogy az olvasó rendelkezik némi háttértudással a hálózati adatkommunikáció, és az ilyen hálózati kommunikációban részt vevő nagyon hasznos routerek és kapcsolók terén. Jelen találmány szempontjából különösen fontos, és ezért feltételezzük hogy az olvasó ismeri az ISO-

féle (International Standards Organisation - Nemzetközi Szabványügyi Szervezet) hálózati architektúra modellt, amely a hálózat funkcióit rétegekbe sorolja. Egy tipikus, ISO modell szerinti architektúra legalsó szintje az első réteg (Layer1) - utalásokban L1 - amely fizikai útvonalat biztosít a felsőbb rétegekbe tartó jelek számára, amely jelek rendre áthaladnak a második (Layer2 vagy L2), a harmadik (Layer3 vagy L3) rétegen egészen a hetedik rétegig (Layer7 vagy L7), amely a hálózathoz kapcsolt számítógépes rendszer állandó alkalmazás programozási rétege. Jelen dokumentumban az ilyen rétegekre való L1, L2, L3 hivatkozások a hálózati architektúra megfelelő rétegeire utalnak. Jelen leírásban feltételezzük a csomagként vagy keret néven ismert, a hálózati kommunikációban használt bitfüzérek alapvető ismeretét.

Napjainkban a hálózati műveletekről kialakított kép egyre meghatározóbb részévé válnak a sávszélességgel kapcsolatos megfontolások (vagyis, a rendszer által időegység alatt feldolgozott adat mennyisége). Az elmúlt években a hálózati forgalom drasztikusan megemelkedett, amely nagy részben az Internet (lazán összekapcsolt számítógépekből álló publikus hálózat, amelyre gyakran világhálóként (www: world wide web - világháló) hivatkozunk) robbanásszerű növekedésével, kisebb részben a magántulajdonú adatatátviteli hálózatok vagy intranetek népszerűségének növekedésével magyarázható. Az interneten és intraneteken keresztül zajló, távoli helyek közötti nagymennyiségű információávitel a távoli információhoz és alkalmazásokhoz való hozzáférés állandóan növekvő igényét elégíti ki. Az Internet nagy számú, földrajzilag elosztott

helyeken élő felhasználó számára biztosítja az egyre növekvő mennyiségű információ és számos új alkalmazás, például e-kereskedelem (e-commerce) elérését, és mindez folyamatosan tovább növeli a hálózatok terhelését. Más alkalmazások, mint például elektronikus levelezés (e-mail), fájl átvitel, adatbázis hozzáférések mind tovább terhelik a hálózatokat, amelyek egy része már így is túl nagy terhelésnek van kitéve a magas szintű hálózati forgalom miatt.

A hálózatokon áthaladó forgalom is egyre változatosabb. Régebben a hálózatok nagy részét elsősorban egy bizonyos típusú kommunikációs forgalom bonyolítására használtuk, például a távbeszélő hálózaton hangot, az adatátviteli hálózaton digitális adatot vittünk át. Természetesen a hang jele mellett a távbeszélő hálózaton korlátozott mennyiségű „adatot" is átvittünk (mint például a hívó és a hívott fél telefonszámát irányítási és számlázási okokból), de a hálózatokat azért elsősorban, alapjában véve homogén csomagok átvitelére használtuk.

Jelenleg azonban a hang- és adatforgalom egyre inkább ugyanazokba a hálózatokba konvergálnak. Az internet további terjedésével és a megbízhatóság és biztonság területén tovább fejlődő technológiával lehetőségünk nyílik arra, hogy számos különböző fajta információt vigyünk át (ide értendő a különböző típusú információk keveréke, úgy mint hang és adat) gyakorlatilag egyidőben.

Adatot jelenleg ingyen továbbíthatunk az interneten (az Internet Protokollon, azaz IP-n keresztül) , és általában a hang átvitele (voice traffic) is alacsony költségű. Az olyan

technológiák, mint a hang átvitele IP felett (VoIP - voice over IP), hang átvitele aszinkron átviteli mód (asynchronous transfer mode - ATM) felett (VoATM - voice over ATM), vagy hang átvitele kerettovábbítás (frame relay - FR) felett (VoFR - voice over FR) mind gazdaságos megoldások hang átvitelére a jelenlegi környezetben. Ahogy ezek a szolgáltatások migrálnak, az ipar megoldásokat fog adni a költség-struktúra megváltoztatására és a processzorok közti információátviteli szolgáltatás költsége és a szolgáltatás minősége között visszaesés kiküszöbölésére.

A szolgáltatás minőségébe beletartozik a kapacitás vagy sávszélesség, a válaszidő (mennyi ideig tart egy keret feldolgozása), és a feldolgozás rugalmassága (különböző protokollok és keret-konfigurációk kezelése, úgy mint különböző egységbezárási (encapsulation) vagy keret fejrész eljárások). Ezen erőforrások igénybe vevői figyelembe fogják venni a szolgáltatás minőségét és a szolgáltatás költségét egyaránt, a bemutatott helyzettől függő visszaesésekkel együtt.

Néhány korábbi adatcsomag irányító rendszerben a csomagok protokollja vagy formátuma megegyezett, esetleg korlátozott számú protokoll vagy formátum volt megengedve. Az ilyen rendszerek nagy előnye a gyorsaság és a rövid válaszidő, hiszen relatíve leegyszerűsödik a tervezés, miután a rendszerben csak egy bizonyos típusú (vagy korlátozott számú) protokoll csomagjai találhatóak, így a rendszer leszűkíthető az engedélyezett protokoll(ok)ra. Amikor a teljes adatátviteli rendszer egyetlen entitás felügyelete alá tartozott, a vezérlő

entitás könnyen rákényszerítette a felhasználókat az egyetlen standard átviteli protokoll használatára (a felhasználók vagy követték a megengedett protokoll(oka)t, vagy nem vették igénybe a rendszert; ugyanis a rendszer úgy volt programozva, hogy csak az előre specifikált protokoll(oka)t tudta kezelni, az ettől/ezektől eltérőeket pedig nem, akármilyen kicsiny is volt az eltérés).

Mégis, még egy olyan kommunikációs „standard", mint az Ethernet keretei is formattálhatóak a számos protokoll egyikének segítségével, továbbá ezek a keretek egységbe zárhatóak (üzenetté alakíthatóak) különböző egységbezárási technikákkal. Ezek a különböző protokollok és egységbezárási technikák változó mennyiségű adatot állítanak elő, tipikusan a keret elejében, még a kulcsfontosságú infromációk - például az L3 szintű üzenet - előtt. Ezért, az Ethernet keret által hordozott kulcsfontosságú információ a kereten belül az Ethernet L3 protokolltól vagy Ethernet formátumtól és az egységbezárási technikától függően - amennyiben ezeket egyáltalán alkalmaztuk - különböző helyeken található meg. Egy L3 üzeneteket feldolgozó rendszernek előbb ezt a helyet kell felderítenie, és ez komoly kihívást jelent többprotokollos rendszer esetében. Így például az Ethernet DIX (Digital, Intel and Xerox - Digital, Intel és Xerox) Version 2 különbözik az Ethernet 802.3-tól, az IPX Ethernet fölött (IPX over Ethernet, IPX - Internet Packet eXchange: internet csomagok váltása) különbözik az IPX Ethernet 802.3-tól, amelynek három különböző formátuma van (Novell Proprietary (Novell szabadalom), LLC (Logical Link Control - logikai kapcsolatvezérlés) és SNAP

(Standard Network Access Protocol - standard hálózat hozzáférési protokoll). Továbbá, az IPX minden verziója támogatja vagy nem támogatja az ún. IEEE (Institute of Electrical and Electronics Engineers - Villamos- és Elektronikai Mérnökök Egyesülete) 802.q szabvánnyal használható látszólagos LAN-t (Local Area Network - lokális hálózat), amelynek hatásaként megváltozik a keret formátuma és így az L3 üzenet helye is.

Az első rendszerekben, amelyek már többféle protokollú kereteket támogattak gyakran jelentős többletterhelésre (overhead) (azaz több száz soros, összehasonlító és elágazó utasításokat tartalmazó számítógépes programra) volt szükség ahhoz, hogy meghatározzák a protokollt és egyik protokollhoz tartozó keretet egy másikká alakítsanak át, továbbá, hogy a felesleges információt (például az egységbezárásra vonatkozó információt) kivonják a keretből. Az ilyen típusú többprotokollos feldolgozás sok időt vett igénybe, a protokoll meghatározásához szükséges idő állandóan változott. Tehát azért, mert az ilyen rendszereknek a protokoll meghatározásához és a szükséges feldolgozáshoz változó hosszúságú időre volt szükségük, a rendszereket a lehető leghosszabb szükséges időtartamra kellett konfigurálni (hogy a legrosszabb esetet is lekezeljék), ami által minden keret feldolgozása a legrosszabb eset idejére lassult; vagy meg kellett engedni azt, hogy néhány keret feldolgozására ne kerüljön sor az osztályozásra szánt idő során.

A legtöbb processzor a feldolgozást egy utasításkészlet általános kezdő utasításával (ugyanazon helyről minden adat

esetében) és a jelzőbitek beállításával kezdi, amelyeket a processzor olyan esetekben olvas ki szelektíven, amikor meg kell határoznia hogy merre menjen, és mely utasításokat hajtsa végre. Így, számos processzor a működése során tesztekkel állapítja meg azt, hogy éppen milyen jellegű adattal van dolga, és hogy hol kezdje meg a tényleges feldolgozást. Ezek a tesztek számos ciklusból állnak és nagy mennyiségű feldolgozást jelentenek.

A szakmában ismerni olyan korábbi, többprocesszoros adatkezelő rendszereket, amelyek a szigorú elsőnek-be elsőnek-ki (first-in first-out - FIFO) adatfeldolgozás elvén működnek. Amíg ez rutinszerű feldolgozás során remekül működik, egy ilyen rendszer zsúfolttá válik és megszűnik működni, ha valamelyik bemenetének a feldolgozása késleltetődik. Egy bemenet feldolgozásának késleltetése megállíthatja a többiek feldolgozását.

Ismertek más, korábbi rendszerek amelyek nyomon követik a bemeneti üzenetet a feldolgozás során. Ezeknek a rendszereknek megvan az a korlátjuk és hátrányuk, hogy a feldolgozási teljesítmény jelentős részét arra használják, hogy nyomon kövessenek a rendszerben található minden információs egységet, és néhányuk már nem tud újabb - például új adatfolyamból vagy rendszeren belül generált üzenetből származó - bemeneti információs egységeket fogadni.

Tehát a korábbi, adatcsomagokat kezelő rendszereknek nem kívánatos hátrányaik és korlátaik vannak, amelyek hatással vannak a rendszer rugalmasságára vagy a működési sebességére, vagy mindkettőre. A korábbi rendszerek további hátrányai és

korlátai válnak nyilvánvalóvá a szakmában jártasak számára jelen találmány további leírása során.

A találmány ismertetése

Jelen találmány kiküszöböli a korábbi rendszerek hátrányait és korlátait, és közben hatékony megoldást biztosít adatfolyam irányítására kereteket vagy csomagokat továbbító hálózatokban, amely kereteket vagy csomagokat a számos megengedett üzenetprotokoll valamelyikével állítjuk elő és amely keretek vagy csomagok tetszés szerint alkalmazhatnak látszólagos LAN rendszert. A csomagok vagy keretek gyors és hatékony analizálásával meghatározzuk a keret típusát és fő karakterisztikáit, majd ezeket eltároljuk hogy a keret feldolgozásánál, illetve jövőbeni referenciaként felhasználhassuk, mint például a korábban említett NPU Szabadalomban leírt típusú hálózati processzornál.

Jelen találmánynak előnye, hogy gyorsan és hatékonyan kezeli a különböző protokollokkal rendelkező csomagokat, így azok gyorsabban és könnyebben feldolgozhatók és az egész rendszer nagyobb sebességgel dolgozza fel a kereteket.

Jelen találmányban az egymást követő különböző formátumú csomagokat illetve kereteket egy router vagy kapcsoló dolgozza fel úgy, hogy nem rendelkezik előzetes információval arról, hogy adott csomagot vagy keretet milyen formátumban hoztak létre. Ebben a megvalósításban ez úgy történik, hogy meghatározzuk az üzenet vagy csomag második rétegbeli (L2) egységbezárási formátumát, majd letárolt szabályok alkalmazásával meghatározzuk az L2 egységbezárást, az L3

protokollt és a virtuális helyi hálózat (VLAN) jelenlétét. Egy ilyen meghatározás eredményeképpen a processzor készen áll a kezdő utasításcímről induló futásra; azaz a processzor előfeltételként rendelkezik a keret azonosításán alapuló utasítás kezdőcímével. A processzor így a kezdő utasítás címén kívül rendelkezik még egy pointerrel, amely a keret adatrészében található L3 fejrész elejére mutat; továbbá rendelkezik még a protokollt, a VLAN jelenlétét és az egységbezárási formulát mutató jelzőbitekkel.

Jelen találmánynak előnye, hogy a csomag kezdeti feldolgozása során összegyűjtjük és letároljuk a csomagra vonatkozó főbb információkat, így a feldolgozás során a csomagról vagy keretről tárolt információ később előnyösen felhasználható, hiszen a távoli lépésekben gyorsabb és hatékonyabb feldolgozást tesz lehetővé, mint például az NPU Szabadalomban ismertetett hálózati feldolgozó egységekben.

Jelen találmánynak előnye, hogy az egyetlen adatfolyamból érkező minden bemeneti csomagot vagy keretet a számos független processzor valamelyikéhez rendelhetjük, majd a kimeneti (feldolgozott) csomagokat vagy kereteket visszaállíthatjuk érkezési sorrendjükbe.

Jelen találmánynak előnye, hogy több adatfolyam együttes feldolgozására képes úgy, hogy azok nem befolyásolják egymást, és egyik adatfolyam sem blokkolja le a többit. Így abban az esetben, amikor az egyik adatfolyam feldolgozása megáll, mert egy részének feldolgozására várni kell, attól még a többi adatfolyam feldolgozása folytatódhat.

A találmány jelenlegi felépítésével szükség esetén „kiöblíthetjük" a rendszert (flushing of the system), azaz a sorrendre való tekintet nélkül azonnal továbbíthatjuk a csomagokat - ezzel semmibe véve a normális működést, amely során az adatfolyamokat érkezési sorrendjüknek megfelelően kezeljük.

Jelen találmánynak szintén előnye, hogy hatékonyan használja a buffereket és más tárolási eszközöket, továbbá gyorsan működik, így a feldolgozás sebessége nem csökken az adatfolyamok kezelésével járó többletterhelés miatt.

Jelen felépítés szemlélteti, hogy a találmány kialakítható ugyanazon a félvezető hordozón, mint amin a hálózati processzorok és a hozzájuk tartozó tárolási egységek, így lesz gyors a komponensek közötti adatátvitel.

Jelen találmányt inkább valósítjuk meg hardver, mint szoftver eszközökkel, és a szükséges formátum vizsgálatok uniform időben végrehajthatóak, függetlenül a formátumtól és a formátum illetve az egységbezárási technika meghatározásához szükséges összehasonlítások számától. Az ismertetett felépítés szerint a keret osztályozása két órajel cikluson belül végrehajtható - az adott keret típusát és látszólagos LAN támogatását jelző bitek beállításával, és a keretre vonatkozó főbb információk tárolásával együtt. A két azonos ciklus során a keretet a forgalomirányító egy üresjáratú (idle) hálózati feldolgozási egységbe küldheti (amint ezt a referenciaként használt NPU Szabadalom is ismerteti). A keret protokollját és egységbezárási eljárását meghatározó feldolgozás eredményeként meghatározzuk és továbbadjuk a processzornak a kezdőcímet,

amely kezdőcím alapján a processzor megkezdheti a kerethez kapcsolódó működést. Ilyenkor a processzorba előre betöltjük a kezdőcímet (pontosabban egy pointert, amely az ide vonatkozó utasítástárra mutat) és a feldolgozáshoz szükséges más, releváns információt. A processzor előzetes, a feldolgozáshoz szükséges kezdőcímmel való feltöltése - amit gyakran a processzor prekondícionálásának nevezünk - nagy processzor-hatékonyságot tesz lehetővé - ugyanis nincsen szükség arra,hogy a processzor a vizsgálat eredményei alapján számos vizsgálati és ugró utasítást hajtson végre, hanem egyszerűen az adott üzenet formátumához tartozó kezdőcímen kezdheti meg a működést.

Jelen találmányhoz tartozó rendszer előnye az is, hogy a keret minősítése és előfeldolgozása, továbbá a keret elosztása a hálózati feldolgozási egységek felé párhuzamosan történhet. Ennek a párhuzamos feldolgozásnak köszönhetően a keretek kezelése sokkal hatékonyabbá válik, és így a rendszer is gyorsabban működhet.

Jelen találmány alkalmazásával a számos feldolgozó egység egymástól független lehet, ugyanakkor egyazon adatfolyam feldolgozása során annak részei nem kerülnek más, nem kívánatos sorrendbe. Adott adatfolyamhoz tartozó feldolgozott keretek vagy csomagok a kimeneten ugyanabban a sorrendben jelennek meg, mint amilyen sorrrendben ezen keretek vagy csomagok a rendszerbe kerültek, ellenkező esetben kiöblítés (flush) parancsot hajtunk végre.

Végül, jelen találmányhoz tartozó rendszerhez új adatfolyamokat és csomag vagy keret előállítást kapcsolhatunk

anélkül, hogy megzavarnánk a hálózatból kapott adatfolyam sorrendjét megőrző feldolgozást.

Jelen találmány kiemelt változatában nem csak prekondícionáljuk a processzort (letároljuk az első utasítás címét), hanem további utasításcímeket tárolunk a későbbi feldolgozás céljából. Így a processzor nem csak az első utasítás címét, hanem későbbi elágazási pontoknál esedékes utasítások címét is tartalmazza, így ki tudja küszöbölni a felesleges vizsgálatokat (IF condition THEN GO TO instruction#1 ELSE GO TO instruction#2 - jelentése: a feltétel teljesülése esetén hajtsd végre az egyes számú utasítást, ellenkező esetben hajtsd végre a kettes számú utasítást) a kód végrehajtása során. Így sokkal hatékonyabban végrehajthatjuk a kódot.

A jelen találmánnyal kapcsolatos további feladatok és előnyök válnak nyilvánvalóvá a szakmában jártasak számára, amennyiben a találmány elkövetkezendő leírását összevetik a mellékelt rajzokkal és az igénypontokkal.

Az ábrák rövid ismertetése

Most, miután szóltunk a korábbi megoldások korlátairól és hátrányairól, miután bemutattuk jelen találmány számos feladatát és előnyét, további feladatok és előnyök válnak nyilvánvalóvá a szakmában jártasak számára, amennyiben figyelemmel kísérik a jelen találmány által tökéletesített forgalomirányítási rendszert és eljárást szemléltető ábrákat, amelyek a következők:

1. ábra blokk-diagram, amely szemlélteti az NPU szabadalomban ismertetett, beágyazott processzorokat tartalmazó interfész eszközt és amely hasznosnak bizonyulhat jelen találmány megértése szempontjából;

2. ábra blokk-diagram, amelyen az első ábrán levő beágyazott processzorhoz hasonló típusú processzor látható, amely kiegészül egy minősítő hardver támogatással, amit jelen találmányban is alkalmazunk;

3A - 3T. ábrák diagramok, amelyek a jelen találmányhoz tartozó hardver minősítőben használt számos Ethernet protokollt szemléltetik;

4. ábra folyamatábra, amely a jelen találmányhoz tartozó hardver támogatású minősítőt szemlélteti, benne a keretrészletek feldolgozásához jelen találmányban használt logikával;

5. ábra jelen találmány minősítőjét szemléltető funkcionális diagram;

6. ábra folyamatábra, amely a jelen találmányhoz tartozó hardver minősítő opcionálisan kiegészített változatát szemlélteti, amelyben a kezdő utasításon kívül további címeket tárolhatunk a veremben (stack);

7. ábra az egyes keretekhez tartozó várakozási sorok sematikus rajza;

8. ábra jelen találmány befejező egységének (completion unit) részletes rajza, amelyen az N processzor mindegyikéhez két címketároló (label store) tartozik;

9. ábra címketároló formátumának sematikus rajza, ahol a címketárolóval követjük nyomon az N processzor által kezelt adatfolyamokat;

10. ábra folyamatábra, amely a befejező egység által végrehajtott azon logikát szemlélteti, amelyet a befejező egység egy új keret valamely feldolgozó egységhez való továbbításakor kapott jelzés vétele és feldolgozása során végrehajt;

11. ábra folyamatábra, amely a befejező egység által abban az esetben végrehajtott logikai folyamatot szemlélteti, amikor az egy keret feldolgozásának befejezését indikáló jelet dolgoz fel;

12. ábra a 8. ábrán látható befejező egység rajza, amelyet - a befejező egység működését illusztrálandó - adatokkal töltöttünk fel.

A kiviteli példák ismertetése

Jelen találmány most következő leírásában a feltalálók által jelenleg ismert legjobb implementációt mutatjuk be eléggé részletesen. Mindazonáltal, ezen leírás a jelen találmánnyal kapcsolatos általános koncepciókat kívánja tanító célzattal bemutatni, a valóságban nem korlátozzuk jelen találmány felépítését az itt ismertetett felépítésre - főleg azért, mert a szakmában jártasaknak is számos lehetséges megoldást és esetleges változtatás juthat eszébe az itt bemutatott specifikus struktúra és műveletek láttán.

Az 1. ábrán egy adatátviteli rendszerhez kapcsolható, csomagok vagy információs egységek - más néven keretek - vételére, feldolgozására és az adatátviteli rendszerbe való visszaküldésére alkalmas feldolgozó rendszer blokk diagramja látható (a keret, csomag illetve információs egység kifejezéseket a leírásban felváltva használjuk). Ahogy az 1. ábrán is látható, az adatfeldolgozó rendszer számos részegységből áll, amiket egyetlen közös hordozóra integráltunk, ahogyan azt az NPU Szabadalom is említi. A teljes egység egyetlen, közös hordozóra való integrálásával a rendszer számos komponensét egymáshoz közel helyezhetjük el, ezzel lecsökkentve a komponensek közötti kommunikáció idejét és megnövelve a rendszer működési sebességét. A számos processzor, támogató logikák és memóriák egyetlen, közös hordozóra helyezésével csökkentjük a belső kapcsolatokban előforduló hibák gyakoriságát és megnöveljük a zajjal vagy más, idegen eredetű jelekkel szembeni ellenállást, amelyek elronthatják a hálózatbeli adatátvitelt.

A 10 hordozón a részegységeket egy felső és egy alsó konfigurációba rendezzük, ahol a „felső" konfiguráció (amelyet néha „belépésnek" nevezünk) azokra a komponensekre utal, amelyek az adatátviteli hálózatból a chip felé (a chip felé vagy chipbe) tartó adatokhoz kapcsolódnak; és ahol az „alsó" konfiguráció (amelyet néha „kilépésnek" nevezünk) azokra a komponensekre utal, amelyeknek feladata hogy adatot vigyenek át a chipből az adatátviteli hálózat felé (a chipből elfelé, a hálózat felé, a hálózatba). Az adatfolyamok követik a felső illetve alsó konfigurációk elhelyezkedését, így az 1. ábrán levő rendszerben egy felfelé és egy lefelé haladó adatfolyam található. A felső vagy belépési konfiguráció elemei között található egy Várakozási Sorba Beállító és Onnan Kivevő 16 Ütemező Felső logika (Enqueue-Dequeue-Scheduling Upward Logic - EDS-UP), egy többszörösen multiplexált Felső 14 MAC (Media Access Control - média hozzáférés vezérlő) (PMM-UP - P Multiple Multiplexed Upward), egy 18 Felső Adattovábbító Kapcsoló (SDM-UP - Switch Data Mover Upward), egy 20 rendszer interfész (SIF - System Interface), és egy Adat Sorbarendező 24 B Soros Link (DASL-B - Data Align Serial Link B). Az adat linkekről részletesen olvashatunk a korábban említett Link Szabadalomban, és rendszer ezen részének bemutatása során a jobb megértés végett hivatkozhatunk erre a dokumentumra. Fontos megérteni, hogy ugyan a találmány jelenlegi felépítésében a hivatkozott találmányban leírtakkal azonos adat linkeket használunk, számos, előnyösebb rendszert is alkalmazhatunk - különösen olyanokat, amelyek nagymértékű adatforgalmat és rendszerszintű követelményeket támogatnak - hiszen jelen találmány megvalósítási lehetőségeit nem

korlátozzák olyan kiegészítő eszközök, mint például a jelenlegi felépítésben alkalmazott adat linkek.

A rendszer alsó (vagy kilépési) oldalán az alábbi komponenseket jelöltük: 26 A és 28 B adat linkeket (DASL-a és DASL-B), 30 rendszer interfészt (SIF), a 32 alsó adattovábbító kapcsolót (SDM-DN), a 34 várakozási sorba beállító és onnan kivevő ütemező alsó logikát (EDS-DN), és a 36 többszörösen multiplexált alsó MAC-t (PMM-DN). A 10 hordozón található továbbá számos belső, statikus, véletlen hozzáférésű memória komponens (S-RAM-ok), egy 40 forgalomirányító ütemező (TRAFFIC MGT SCHEDULER, MGT - management), és egy - az NPU Szabadalomban sokkal részletesebben leírt - 12 beágyazott processzorokból álló felépítmény. A 38 interfész ezközt a megfelelő DMU (Data Manipulation Unit - adatmanipulációs egység) buszok kapcsolják a 14, 36 PMM-ekhez. Az 38 interfész eszköz lehet bármilyen szerkezet, amellyel az L1 áramkörhöz csatlakozni tudunk, például egy Ethernet Fizikai eszköz (ENET PHY - Ethernet Physical) vagy aszinkron átviteli mód keretező berendezés (Asynchronous Transfer Mode Framer - ATM FRAMER), amelyek mindketten jól ismertek és általánosan igénybevehetők ilyen célokra. Az interfész eszköz típusát és méretét legalább részben meghatározza a hálózati média, amelyhez a chip-et és a rendszert kapcsoljuk. A chip működéséhez számos külső, dinamikus, véletlen hozzáférésű memória eszköz (D-RAM) és S-RAM áll rendelkezésre.

Igaz ugyan, hogy jelen találmányt hálózati környezetben mutatjuk be, amely hálózatban a megfelelő kapcsoló és irányító berendezéseken kívül az adatfolyam legnagyobb részét az

épületeken belül kialakított elektromos vezetőkön, azaz vezetékeken és kábeleken visszük át; a találmány jól szemlélteti azt, hogy a hálózati kapcsolókat és komponenseket ugyanígy használhatjuk vezeték nélküli környezetben. Például, az előbb említett média hozzáférés vezérlő (Media Access Control - MAC) elemeket megfelelő rádiófrekvenciás eszközökre cserélhetjük le - például olyanokra, amelyek a szilikon-germánium technológiával készülnek - és ennek eredményeképpen a találmány közvetlenül egy vezetéknélküli hálózathoz csatlakoztatható. Ahol ezt a technológiát megfelelően alkalmazzák, a rádiófrekvenciás elemeket VLSI (Very Large Scale Integrated - nagyon nagy integráltságú) elemekbe lehet integrálni, amint ezt már a szakmában jártas olvasó felfedezhette. Egy másik megoldásban rádiófrekvenciás vagy más, vezeték nélküli válaszegységek (response device) helyett az itt említett többi elem mellé infrás (IR - infrared) válaszegységeket helyezhetünk el, hogy így vezeték nélküli hálózati berendezésben használatos kapcsolót kapcsoló berendezést kapjunk.

Az 1. ábrán látható nyilak az interfész rendszerben levő adatfolyam irányát mutatják. A 38 ENET PHY blokkból a DMU buszon a 14 Ethernet MAC-be érkező adatkereteket vagy üzeneteket az 16 EDS-UP eszköz a 16a belső adattároló bufferekbe helyezi. A keretek lehetnek normál keretek vagy vezetett keretek, utóbbiak kapcsolódnak a processzorok egyikében hamarosan megtörténő feldolgozás helyéhez és eljárásához.

A 2. ábrán látható a jelen találmányt előnyösen alkalmazható 100 feldolgozó rendszer blokk diagramja. Ezen a

2. ábrán számos 110 feldolgozó egységet helyeztünk el a 112 forgalomirányító egység és a 114 befejező egység között. Minden beérkező F keretet (a rendszerhez csatolt hálózatból érkezik, a hálózatot az ábra nem jelöli) egy 116 UP adattárba tárolunk le, amelyet az 117 UP DS i/f interfész - amely adatokat írhat és adatokat olvashat az adattárból - kapcsol össze a 110 feldolgozó egységekkel. A kereteket az adattárból a 112 forgalomirányító sorban a 110 feldolgozó egységek valamelyikéhez rendeli, a döntést a 112 forgalomirányító az alapján hozza meg, hogy a feldolgozó egység rendelkezésre áll-e a keret feldolgozásához. A rendelkezésre állást az a processzor, amelyikhez az F keretet hozzárendeltük, jelezheti például azzal, hogy egy jelet küld a 112 forgalomirányítónak, amely azt mutatja, hogy az a bizonyos feldolgozó egység éppen nem végez műveleteket és készen áll a munkára. A munka hozzárendelésének még számos más, eme rendszert előnyösen érintő változata van (például körbeforgó (round-robin) allokáció vagy „legrégebben használt" (LRU - least recently used) algoritmus). A 110 feldolgozó egységek struktúrájáról és funkcióiról részletes, a feldolgozó rendszerről általános leírást találhatunk a korábban hivatkozott NPU Szabadalomban. A 112 forgalomirányító és a számos 110 feldolgozó egység között egy 118 segéd hardver minősítő található, amelyről később ebben a dokumentumban részletesen szólunk, különösen a 4. és 5. ábrákkal kapcsolatban. A 122 utasítástárat, - amelyben több utasításkészletet tárolunk, hogy azokat a 110 feldolgozó egységek később visszakereshessék és végrehajthassák - szintén a 110 feldolgozó egységekhez kapcsoljuk. Ahogyan azt majd később részletesen leírjuk, a 122

utasítástárban levő kezdő utasítást az üzenet típusa - protokollja és egységbezárási eljárása - alapján a 118 segéd hardver minősítő által meghatározott címmel címezzük.

A 114 befejező egység a számos 110 feldolgozó egység, az alsó sorbaállító rendszer (a 34, DN Enqueue címkéjű elem az 1. ábrán) és a felső sorbaállító rendszer (a 16 elem az 1. ábrán) között helyezkedik el. A 34 DN Sorbaállító rendszer a feldolgozó együttesből kilépő feldolgozott kereteket leküldi abba a hálózatba vagy rendszerbe, amelyhez a feldolgozó együttest kapcsoltuk; a 16 UP Sorbaállító rendszer a feldolgozott kereteket a kapcsolószerkezethez küldi. A 112 forgalomirányítót úgy tervezzük, hogy azonosítási információt rendeljen minden kerethez és tárolja is le ezt az információt, továbbá rendeljen azonosítási információt az ilyen kereteket majdan feldolgozó feldolgozó egységhez is, és azt is tárolja le. Ezeket az azonosítási információkat használva a 114 befejező egység biztosítja azt, hogy az egy adatfolyamot alkotó feldolgozott kereteket az érkezésük sorrendjében továbbítsuk. Ebből a nézőpontból később, ebben a specifikációban, még részletesebben tárgyaljuk jelen találmányt.

A 3. ábrán (ami számos kisebb rajzból áll: 3A - 3T) azok az üzenetformátumok láthatók, amelyek fogadására és feldolgozására jelen feldolgozó rendszer fel van programozva; habár az üzenetek vagy keretek repertoárját a szakmában jártasak meg tudják változtatni annak érdekében, hogy azok belepasszoljanak adott rendszer környezetébe. Jelen rendszer átalakítható úgy, hogy más üzenet formátumokat is elfogadjon, ebbe beleértve azokat az üzenet formátumokat és

változtatásokat, amelyeket majd a jövőben fognak megtervezni. Így, a 3. ábrán látható üzenet formátumok csak azt illusztrálják, hogy mennyi különböző keret formátum létezik, különböző protokollokkal és egységbezárási típusokkal; és jelen találmány egy rugalmas rendszer, amelyet arra terveztek, hogy számos különböző protokollt és egységbezárási formátumot fogadjon el, és hogy segítse a keretek feldolgozását úgy, hogy biztosít egy pointert - amely az egységbezárás és a protokoll típusára mutat - és egy kezdőcímet annak a processzornak az utasítástárába, amely processzor az adott keretet kezeli.

A 3A. ábra az általános vagy alap Ethernet üzenet formátumot illusztrálja, amelyet néha Ethernet Version 2.0/DIX-nek is nevezünk. Ebben az üzenet formátumban az üzenetbe beletartozik a rendeltetési cím (Destination Address - DA), a forrás címe (Source Address - SA), egy blokk, amely az üzenet típusát mutatja (Type - Típus), az üzenet szövege vagy adatok és egy ciklikus redundancia ellenőrző vagyis CRC (Cyclical Redundancy Checking - ellenőrző összeg) mező az üzenet verifikációjához. A DA rendeltetési címet és az SA forrás címét 6 bájtban (48 bitben) határozzuk meg, a Típust jelölő blokk hossza 2 bájt, a CRC végződés 4 bájt hosszú. Általában az üzenet ezektől különböző része - az Adat - egészen 1500 bájt-ig bármilyen hosszú lehet, de azt amint majd később látni fogjuk az Ethernet néhány típusa más előnyök kiaknázása végett korlátozza ezt a rugalmasságot. Az SA forrás címe jelentheti azt, hogy az üzenet egy egyedi üzenet, amelyet a hálózat egyetlen csomópontjának egyetlen címmel címeztek, de jelentheti azt is, hogy ez egy többes küldéses (multicast) vagy adatszórásos (broadcast) üzenet. Egy többes küldéses

üzenet a hálózati csomópontok egy csoportjának, egy adatszórásos üzenet az összes állomásnak szól. A Típust jelző blokk 16 bitje azonosítja a használt magasabb szintű protokollt. Minden regisztrált Ethernet protokollhoz tartozik egy egyedi típuskód, egy számérték, amely mindig nagyobb, mint az Ethernet 802.3 hosszmezőjében előforduló legnagyobb érték, így ezen két mező együtt létezhet. Az adatmező tipikusan 46 - 1500 bájt hosszú, ezzel feltételezzük hogy a felsőbb rétegek majd biztosítják, hogy teljesüljön a 46 bájtnyi minimális hossz, mielőtt az adatot az MAC rétegbe továbbítják.

A 3B. ábra az általános Ethernet formátumnak egy változatát illusztrálja, amelyre gyakran IEEE 802.3 Ethernet formátumként hivatkozunk. Hasonlít a 3A. ábrán látható alap Ethernet üzenetformátumhoz, kivéve azt, hogy a Típus mező helyett itt egy Hossz hosszmező található (Length - LEN), melynek 16 bitje az őt követő adatmező hosszát mutatja, nem számolva a kitöltő elemeket (pad - töltelék). Ez a szabvány 64 bájtban határozza meg a csomagok minimális hosszát, így az Adat adatmező mérete legalább 64 bájt kell, hogy legyen. Amennyiben az Adat adatmezőben kevesebb, mint 64 bájt adat van, akkor az MAC réteg kitöltő karakterekkel tölti fel az LLC adatmezőt a minimális méretre, még mielőtt a csomagot a hálózatba elküldené. Mindazonáltal, a hosszmező a kitöltő karakterek nélküli hosszt tartalmazza, ezért az üzenetet fogadó rendszer könnyen azonosíthatja és figyelmen kívül hagyhatja a kitöltő karaktereket.

A 3C. ábra az Ethernet üzenetek, különösen az IEEE 802.1q szabványhoz tartozók Tag Vezérlési Információ Formátumát (Tag Control Information Format) illusztrálja. A formátumban 3 bit

jelzi a felhasználói prioritást, 1 bit a Kanonikus Formátum Indikátor (Canonical Format Indicator - CFI), és 12 bit a Látszólagos LAN Azonosító (Virtual LAN Identifier - VID). A látszólagos LAN vagy a lokális hálózat olyan hálózati csomópontok azonosítása, amelyeket látszólagos lokális hálózatnak definiáltunk úgy, hogy a címek VLAN-t alkossanak, továbbá megengedve azt, hogy a fizikailag nem összekötött csomópontok logikailag összetartozzanak és inkább legyenek csoportként, mint egyedileg megcímezhetőek.

A 3D. ábrán egy - az IEEE 802.q szabványt követő - Beágyazott RIF (Routing Information Field - forgalomirányítási információs mező) (Embedded RIF - E-RIF) formátum látható, amelyet néhány Ethernet protokoll üzenetformátumaként használunk. Ebben a formátumban az első 3 bit jelzi az RT útvonal típusát (route type - RT), a következő 5 bit mutatja a LTH hosszúságot (az E-RIF teljes hosszát bitekben, beleértve az E-RIF útvezérlést (route control) és az E-RIF útleírót (route descriptor)), majd 1 bit jelzi a D útleíró irányát (normális esetben a „0" jelzi, hogy az útleírón előrefelé haladunk át, de speciálisan irányított keretek esetén az értéke „1", ekkor az útleírón visszafelé irányban haladunk át). Az E-RIF formátumnak része továbbá egy 6 bites legnagyobb keret indikátor (Largest Frame indicator - LF) és egy 1 bites Nem Kanonikus Formátum Indikátor (Non Canonical Format Indicator - NCFI). Az RT útvonal típusa az alábbi értékeket veheti fel: 00x, 01x, 10x, 11x, amelyek jelentései: speciálisan irányított keret, átlátszó keret, minden utat bejáró keret (all route explore frame), fában szélességi bejárást végző keret (spanning tree explorer frame). Az LF

legnagyobb keret mező mérete 1470 bájt, vagy kevesebb, az Ethernet IEEE 802.3 szabványtól függően. Az NCFI értéke 0, ha az MAC címeket nem kanonikus formában határozták meg, értéke 1, ha kanonikus formában.

A 3E. ábrán látható E-RIF útleíró formátumban egy 12 bites LAN ID helyi hálózat azonosító (Local Area Network Identifier - LAN ID) és egy 4 bites hídszám (Híd#) található. Az E-RIF Útleíró Formátum mező is jól ismert a szakmában, használata a hasonló mezőkre vonatkozó szabvány szerint történik.

A 3F. és 3G. ábrákon Ethernet üzenetben használatos LLC formátumok komponensei láthatók, a 3F. ábrán egy 802.2 LPDU (Protocol Data Unit - protokoll adategység) formátum, a 3G. ábrán egy Generikus SNAP formátum. A 3F. ábrán látható LPDU formátumban található egy 1 bájtos (8 bites) DSAP Rendeltetési Hely Szolgálatelérési Pont (Destination Service Access Point - DSAP), egy 1 bájtos SSAP Forrás Szolgálatelérési Pont (Source Service Access Point - SSAP) és egy 1 - 2 bájtos Vezérlőmező (Control), amelyben parancs, válasz, sorozatszám illetve poll/final (lekérdezés/utolsó) bit(ek) található(k). Ebben a kontextusban a szolgálatelérési pont 6 bitből áll, továbbá egy U bitből és egy végső (final) bitből (egy külön I bit tartozik a rendeltetési hely szolgálatelérési pontjához és egy C bit mutatja, hogy parancs illetve válasz üzemmódban vagyunk-e). A 3G. ábrán az SNAP formátum látható, amelynek három bájtja a szerveződést (organization) (Organizationally Unique Identifier - Szerveződésileg Egyedi Azonosító), két bájtja a 0002 Internet szabványbeli formátumhoz rendelt típust mutatja. Például a Típus mező értéke IP esetén 0800, IPX esetén 8137,

ARP esetén 0806, RARP esetén 8035, 802.1q VLAN esetén 8100, IPv6 esetén 86DD, Appletalk 80DB, és Appletalk AARP esetén 80F3.

A 3H. ábrán egy Ethernet feletti IPX-beli üzenet formátumát látjuk, amely formátumban egy Ethernet MAC fejrész és egy IPX fejrész található, ahol az Ethernet MAC fejrészben található a 6 bájtos SA forráscím, a szintén 6 bájtos DA rendeltetési cím, és egy két bájtos Típus, amelynek értéke 8137, ami azt mutatja, hogy a keret IPX formátumú. Ezután az IPX fejrészben a következő részek találhatók: 2 bájtos ellenőrző összeg, 2 bájtos csomaghossz, 1 bájt TC (Tag Control - címke vezérlés), 1 bájt PT (Payload Type - hasznos adat típusa), 4 bájt jelzi a rendeltetési helyhez tartozó hálózatot, 6 bájt jelzi a rendeltetési helyhez tartozó hálózati csomópontot, 2 bájt jelzi a rendeltetési hely socket-jét, 4 bájt a forrás hálózatát, 6 bájt a forrás hálózati csomópontját és végül 2 bájt jelzi a forrás socket-jét.

A 3I. ábrán egy Ethernet 802.3 verzió feletti IPX üzenetformátumot (más néven Novell formátum) látunk, amelyhez tartozó Ethernet 802.3 MAC fejrészben az üzenet hosszát a harmadik mezőben specifikáljuk (ellentétben a 3H. ábrán látható Ethernet feletti IPX Típus mezejével). Ebben a formátumban az ellenőrző összeget a protokollnak megfelelően „FFFF"-re állítjuk.

A 3J. ábrán 802.2-vel Ethernet 802.3 feletti IPX látható, amelyben az üzenetben levő MAC és IPX fejrészeket (hasonlóan a 3H. ábrán látottakhoz) a 802.2 LLC LPDU mezői választják el.

A 3K. ábrán SNAP-s 802.3 feletti IPX keret formátuma látható, - hasonlóan a 3J. ábrával kapcsolatban ismertetett

formátumhoz - amelyben a 802.3 MAC fejrész után az LLC LPDU mező, majd az IPX fejrész következik. Az LLC LPDU rész és az IPX fejrész között található SNAP mező mutatja az OUI-t és a 8137 értékű ETípust (EType).

A 3L. ábrán 802.1q VLAN támogatású, Ethernet feletti IPX formátum látható, amelynél a Típus mezőben a 8100 érték szerepel, továbbá a VLAN csomag az Ethernet MAC fejrész és az IPX fejrész között helyezkedik el (az IPX fejrész formátuma megegyezik a 3H, 3J. és 3K. ábráknál fentebb ismertetett formátummal). A VLAN csomagban található egy 2 bájtos TCI (Tag Control Information - címke vezérlés információ) mező, egy 2 bájtos LEN hosszúság vagy e-típus mező, majd következik egy e-rif vezérlőmező és változó számú e-rif leíró mező, ahol az e-rif mezők számát a (LEN - 2) / 2 formula alapján határozzuk meg.

A 3M. ábrán 802.1q VLAN támogatású, Ethernet 802.3 feletti IPX formátum látható. A Típus mező értéke 8100, a VLAN csomag megegyezik az előző VLAN példában, a 3L. ábrán bemutatott csomaggal. Az IPX fejrész megegyezik a 3I. ábrán már korábban bemutatott megoldással, amelyben az ellenőrző összeg értékét „FFFF"-re állítjuk.

A 3N. ábrán látható keret-elrendezés a VLAN támogatással igénybe vett Ethernet 802.3 feletti IPX keret felépítését mutatja be. A felépítésben megtalálható a 802.3 MAC fejrész, amelyhez tartozó Típus mező 8100-as értéke VLAN csomag jelenlétére utal (mint a 3M. ábrán), a VLAN csomag (szintén a 3M. ábrához hasonlóan), az LLC LPDU (megegyezik a 3J. ábrán szereplő és avval kapcsolatban bemutatott LLC LPDU-val), és az IPX fejrész (ahogyan azt a 3H. ábrán is láthatjuk).

A 3O. ábrán 802.1q-t használó VLAN és SNAP támogatású Ethernet 802.3 feletti IPX üzenet konfigurációját illetve formátumát láthatjuk. Felépítése megegyezik a 3N ábrán bemutatottal, azzal a különbséggel, hogy az SNAP mező az LLC LPDU és az IPX fejrész között helyezkedik el.

A 3P. ábrán Ethernet feletti IPv4 formátum látható, amelyben az üzenetnek részei az Ethernet MAC fejrész és az IPv4 fejrész. Az ábra pontosan mutatja az egyes fejrészek mezőinek hosszát.

A 3Q. ábrán 802.2-vel 802.3-as Ethernet feletti IPv4 üzenetformátum látható, amelyben az MAC fejrészt az LLC LPDU majd az IPv4 fejrész követi.

A 3R. ábrán SNAP-vel Ethernet 802.3 feletti IPv4 keret üzenet formátuma látható, amelyben a 802.3 MAC fejrészt az LLC LPDU, majd az IPv4 fejrész követi (UDP vagy TCP esetén opcionálisan csatolható farokrésszel).

A 3S. ábrán 802.1q VLAN támogatású, Ethernet feletti IPv4 üzenetformátum látható. Ebben a formátumban fellelhetőek úgy az IPv4, mint más, 802.1q VLAN támogatású rendszereknél látott VLAN Csomag tulajdonságai.

A 3T. ábrán 802.1q VLAN támogatású, 802.3 (és 802.2) Ethernet feletti IPv4 üzenetformátum látható, amely ötvözi a 802.2-vel 802.3 feletti IPv4 attribútumait a VLAN Csomag üzenetének karakterisztikájával.

A 3H - 3T. ábrákon levő legalsó sor a keret harmadik rétegbeli (Layer 3 vagyis L3) részét mutatja, és az üzenetnek ezen L3 rétegbeli része - az L3 részt megelőző üzenetrész méretének változásai miatt - más-más helyen kezdődik adott üzenet típusától - a protokolltól és az egységbezárási

technikától - függően. Habár az L3 rész feldolgozása szükségszerű (az egységbezárást figyelmen kívül hagyva), egy többprotokollos, több egységbezárási technikát alkalmazó rendszerben nehézkessé válhat ezen L3 üzenetrész kezdetének megtalálása. Továbbá, mivel a számos 110 processzor bármelyike által a kereten végrehajtott utasítások függnek a keret protokolljának és egységbezárásának típusától, szükségünk van egy eszközre (esetünkben a 118 segéd hardver minősítőre), amely a processzor 122 utasítástárában levő megfelelő kezdő utasításra mutató pointert előállítja.

A 4. ábrán látható blokk-diagram a 2. ábrán 118 elemként feltüntetett segéd hardver minősítő felépítését mutatja, a 122 utasításkészlet megfelelő részeivel és a számos 110 feldolgozó egység valamelyikével együtt. A 118 segéd hardver minősítő a bemeneti információs egységeknek (vagy kereteknek) megfelelő 128 bites szegmenseken végez műveleteket, amely 128 bites szegmenseket - más néven „FISH"-eket (FI Segment H) - a 118 segéd hardver minősítő (és a számos 110 feldolgozó egység valamelyike) a 112 forgalomirányító egységen keresztül kapja meg. A minősítő funkciót az első 3 FISH alapján hajtjuk végre (azaz a kerethez tartozó első 384 bit alapján, amit a könnyebb megkülönböztetés végett FISH1, FISH2 és FISH3-nak nevezünk). Az első FISH (FISH1) még nem az aktuálisan kapott keret, hanem a keretre vonatkozó információk halmaza, úgy mint a port száma, amelyen keresztül a keret érkezett, 291 default kód belépési pont és a 292 indikátor (értéke igen vagy nem), amelynek értéke alapján engedélyezhetjük a keret minősítését jelen találmány hardver minősítőjének segítségével.

A 210 blokkban az Ethernet típusát a keret különböző részeinél hasonlítjuk össze, így tudjuk megállapítani, hogy a mezők megfelelnek-e a jelenleg konfigurált protokollnak, például az első (pl. IPX) vagy második (pl. IPv4) Ethernet verziónak. A 220 blokkban megállapítjuk hogy az SAP mező (Service Access Point - szolgálatelérési pont) megfelel-e a jelenleg konfigurált protokollnak, ismét a regiszter specifikációjának megfelelően (azaz a regiszterben egy specifikus értéket tárolunk, amely a protokoll típusát jellemzi). Továbbá a rendszerben az SNAP mező értéke alapján meghatározzuk az esetleg eltérő típusú egységbezárási technikát (a 240 blokkban egy külön mezőnek például „AAAA03" értéke alapján) és a 250 blokk segítségével kimutatjuk a látszólagos helyi hálózat (VLAN) jelenlétét. A 260 blokk a minősítés vezérlő, amelynek felelőssége a kerettel kapcsolatos paraméterek tárolása és a protokoll típusa, a 3 rétegbeli részre mutató pointer és a minősítő jelzőbitek kimenetek előállítása a 270, 272 és 274 kimeneti vonalakon.

Az összes definiált formátumhoz - az összes üzenet számára - előre meghatározhatjuk és a 280 táblában letárolhatjuk a vezérlési belépési pontot (a feldolgozás kezdetén, a 122 utasítástárban levő első utasítás címe). Így, ha ETYPE = 0 és nincs VLAN, akkor a vezérlés belépési pont (kezdőcíme) az utasítástár 122a címén található, amennyiben ETYPE = 1 és nincs VLAN, akkor a vezérlés belépési pont a 122b címen található. Hasonlóan, ETYPE = 0 és VLAN illetve ETYPE = 1 és VLAN esetén a megfelelő vezérlés belépési pontok (az a hely, ahol az aktuális üzenet feldolgozása megkezdődik) a 122c illetve 122d címeken találhatóak. A 122e utasítással kezdődik

meg az ERIF mezőt tartalmazó keretek feldolgozása, továbbá a 122f utasítással kezdünk default esetben, amikor nem sikerült azonosítani a protokollt és az egységbezárási metódust.

Az üzenet FISH1-ében minden esetben megtalálható és a 290 blokkban kiolvasható a default vezérlés belépési pont. Azután a 295 blokkban meghatározzuk, hogy a default vezérlés belépési pontot használjuk-e - amennyiben a 295 vonalon engedélyezett a hardver minősítés, és a 280 blokk nem határozott meg más vezérlés belépési pontot, akkor a default belépést alkalmazzuk; ellenkező esetben a vezérlés belépési pontot a 280 tábla alapján kapjuk meg.

A 118 hardver minősítőből kimenő 270 és 272 vonalakat (amelyek rendre a 118 hardver minősítő által előállított minősítő jelzőbiteket és az L3 báziscímet szállítják) a 110 különálló processzorhoz vezetjük - amelynek feladata a keret feldolgozása - , majd letároljuk őket a 110a általános célú regiszterekben, amely regiszterek szintén a keretet feldolgozó és azt a 110b adattárban letároló feldolgozási egységhez tartoznak. A 295 eszközből kimenő 276 kimeneti vonalon kapjuk meg adott típusú kerethez a 122 utasítástárbeli kezdőcímet, amely adatot a 110c utasítás vezérlő logikában tároljuk. A 110 feldolgozó egységnek az ALU (Arithmetic/Logic Unit - aritmetikus/logikai egység) is része. A 110 processzor a 110c utasítás vezérlő logikában levő utasításszámlálóval éri el a 122 utasítástár utasításait. Így tehát, a 118 hardver minősítő által meghatározott protokoll és egységbezárási módszer alapján, a 110 feldolgozó egységnek birtokában van a keret feldolgozásához megfelelő utasításkészlet kezdőcíme, és a keret típusát mutató, megfelelően beállított jelzőbitek

segítségével a 110 processzor a korrekt utasításokat használva megkezdheti a keret feldolgozását.

Az 5. ábrán az üzenetformátum kategóriájának meghatározásához használt logika látható. A műveletsor a 310 blokknál kezdődik, ahol kiválasztjuk a FISH2-t, majd a 320 blokknál folytatódik, ahol a keret 13 - 14 bájtjait (ez a két bájt tartalmazza a típusra vonatkozó információt egy olyan keretben, ahol a 6 bájtos DA rendeltetési hely címet a 6 bájtos SA forrás címe, majd a Típus mező követi) teszteljük. Ha ezen bájtok megegyeznek ETYPE0 vagy ETYPE1 tartalmával, akkor az eljárással meghatározzuk a protokoll információt és a 323 blokkban beállítjuk a megfelelő jelzőbiteket, majd a 325 blokkban befejezzük a feldolgozást. Máskülönben, ha a típus blokk (type block) értéke kevesebb, mint 0600H (hexadecimális érték), akkor a keret nem Ethernet V2.0DIX, hanem Ethernet 802.3 formátumú, továbbá a szóban forgó mező nem típus mező hanem inkább hossz mező, ebben az esetben a feldolgozást az 5. ábra diagramjának bal felén követhetjük végig. Amennyiben a típus blokk értéke 8100, akkor a keret 802.1q VLAN támogatású (példaként lásd a 3L, 3M, 3N, 3O, 3S. és 3T. ábrákat); ebben az esetben a feldolgozást az 5. ábra diagramjának jobb felén követhetjük végig. Amennyiben a típus mezőben más érték található, akkor a vezérlés a 325 blokkra kerül, ahol a minősítést elvégezettnek tekintjük anélkül hogy bármilyen, a protokollal kapcsolatos információt rögzítettünk volna - hiszen láthatóan nem ismerjük a keret protokollját.

Amennyiben a 320 blokkban vizsgált 13 - 14 bájtok értéke kisebb 0600H-nál, akkor a 322 blokkban a 15 - 17 bájtokat vizsgálva próbáljuk megállapítani, hogy megfelelnek-e

valamilyen típus SAP vagy LLC (Logical Link Control) mezőjének (például a 3K. ábrán használt AAAA03 érték). Ha a mezőt beazonosítjuk valamely ismert SAP mezővel, akkor beállítjuk az SAP mezőt és a 323 blokkban letároljuk a protokoll információt, majd a 325 blokkhoz érve befejezettnek tekintjük a minősítést. Ha a mező egy SNAP mező, akkor a vezérlés a 324 blokkra kerül, ahol a FISH3 2 - 6 bájtjait vizsgáljuk ismert ETYPE-ot keresve bennük. Amennyiben az ETYPE-ot sikerül felismerni, a protokoll információkat a 323 blokkban letároljuk, majd a 325 blokknál kilépünk a rendszerből.

Amennyiben a 320 blokkban vizsgált 13 - 14 bájtok értéke 8100 - amely érték a 802.1q IEEE szabványban specifikáltak szerint látszólagos helyi hálózat (VLAN - virtual local area network) jelenlétére utal -, akkor a VLAN létezését a 330 blokkban rögzítjük, majd a 340 blokkban ellenőrizzük a CFI mező meglétét. Amennyiben találunk CFI mezőt, a minősítés befejeződik és a vezérlés a 325 blokkra kerül. Amennyiben nem, akkor a 350 blokkban a FISH3 1 - 2 bájtjait vizsgáljuk ismert ETYPE-ot (hasonlóan a 320 blokkbeli vizsgálathoz) vagy hosszúságot (0600H-nál kisebbet) keresve. Amennyiben egy ETYPE-ot sikerül felismerni, a protokoll információkat a 323 blokkban letároljuk, majd a vezérlés a 325 blokkra kerül, ahol befejezettnek tekintjük a minősítést. Amennyiben a 350 blokkban vizsgált mező nem bizonyul ETYPE-nak, a 325 blokkban befejezettnek tekintjük a minősítést. Ha a 350 blokkban végrehajtott vizsgálat eredményeként hosszértéket kapunk (0600H-nál kisebbet), akkor a 360 blokkban ismert SAP-t keresünk a 3 - 5 bájtokban. Amennyiben a AAAA03 értéket

kapjuk, a vezérlés a 370 blokkra kerül, amelyben ismert ETYPE-ot keresünk a 6 -10 bájtokban.

A 6. ábrán a 4. ábrán bemutatott hardver minősítő továbbfejlesztett változata látható. A 6. ábrán megtalálható a 4. ábra összes eleme, azzal az újítással, hogy a 110c utasítás logikában található 110d utasítás veremben (instruction stack) most - egyetlen kezdőcím helyett - számos cím tárolható. A veremben szerepel a kezdeti vezérlési utasítás, amit elágazások esetén (fork (villa) vagy branch (elágazás)) használt címek követnek, melyek segítségével a processzor el tudja kerülni a későbbi ágaknál aktuális tesztek elvégzését illetve feltételes állítások vizsgálatát. A kezdőcímeket tehat megfelelő sorrendben tároljuk a veremben, amelyből elágazó utasítás esetén kiemelhetjük a következő címet.

A különböző protokollokat és változatos egységbezárási technikákat alkalmazó Ethernet üzenetek definíciószerű tartalmáról az olvasó további információt talál az Ethernet keret felépítését leíró szabványban vagy referencia kézikönyvben. Néhány mindenki számára elérhető dokumentum, amelynek ismerete hasznosnak bizonyulhat az Ethernet protokollok és egységbezárási technikák és az ezekhez kapcsolódó szabványok és opciók megértéséhez: ISO/IEC Final CD 15802-3, IEEE P802.1D/D15, 1997 november 24., Annex C; IEEE Draft Standard 802.1Q/D9, 1998 február 20; RFC 1700 - hozzárendelt számozást készítette J. Reynolds és J. Postel, 1994 október (a dokumentum elérhető a http://www/isi.edu/rfc-editor/rfc.html oldalon is); IBM Token Gyűrű Hálózati Architektúra Referencia (Token Ring Network Architecture Reference); és IBM LAN Hidak és Kapcsolók Összefoglaló (LAN

Bridge and Switch Summary). A kiadási szám SG24 - 5000 - 00, 1.3 verzió, 1996 január. Az 1.1.1 fejezet különösen ajánlott.

Hardver minősítőt tervezhetünk számos módon, például a rendelkezésre álló számos, logikai tervű hardver konfiguráció (vagy, mint jelen esetben: szilikon hordozó) tervezésére és előállítására alkalmas szoftver eszközök valamelyikével; de terveztethetjük logikai tervezővel akár hagyományos módon, kézzel is. Példánkban a szükséges vizsgálatokat egy VLSI hardver definíciós nyelv (VLSI Hardver Definition Language - VHDL) nevű szoftver nyelven programoztuk, majd a programot átküldtük egy jól ismert szoftver elemen (olyanon, amelyet az IBM illetve a Synopsis is forgalmaz), így a terv alapján megkaptuk a vizsgálatok elvégzéséhez szükséges hardver eszköz kapuit és logikáit. Más hasonló rendszereket is használhatunk, hiszen a logika tervezőjének nem szükséges ismernie a kapuk struktúráját illetve a kapuk helyét, elég ha ismeri azok logikai funkcióit és adott bemenetre illetve tesztekre adott válaszaikat.

Ahogyan azt már fentebb említettük, jelen találmány feldolgozó rendszerét magában foglaló rendszerekben szükség lehet arra, hogy képesek legyenek az adatfolyam feldolgozott kereteit érkezési sorrendben továbbítani, függetlenül attól, hogy a kereteket melyik processzorok dolgozták fel. Egy ilyen rendszerben a 112 forgalomirányító, miután beazonosít egy rendelkezésre álló feldolgozási egységet és a kapott keretet ehhez a feldolgozási egységhez irányítja, létrehozza és letárolja a kerethez és a kerethez rendelt feldolgozó egységhez tartozó azonosítási információt.

A keretek általában azonosítási információval, például az üzenetszámmal (nevezzük MAC-nek is), a keret forrásának címével (más néven SA-val) és a keret rendeltetési helyének címével (más néven DA-val) érkeznek a rendszerbe. Az ilyen jellegű információk helye és tartalma az üzenet formátumától illetve az egységbezárási technikától függően változó, de ez az az információ, amelynek segítségével lehetőség nyílik arra, hogy a keretet a rendszeren, kapcsolókon és routereken keresztül épségben eljuttassuk rendeltetési helyére, és ott a kereteket megfelelő sorrendbe rakva megkapjuk a teljes üzenetet, még akkor is, ha az üzenet hosszabb, mint egyetlen keret. Az üzenet részeit általában adatfolyamnak nevezzük, így az adatfolyam minden részének ugyanavval az azonosítási információval kell rendelkeznie (ugyanavval az MAC-vel, SA-val és DA-val). A 112 forgalomirányító egység által a bemenő kerethez rendelt aktuális címkét (vagy azonosítási információt) számos módon előállíthatjuk, például MAC - SA + DA művelettel vagy más üzenetformátumokban az LID és az MID mezők logikai XOR kapcsolatával.

Ahogyan az a 7. ábrán is látszik, minden keret számára kialakíthatunk három listát vagy várakozási sort. Először is, definiáljuk a 400 várakozási sort, amelyben az elvégzett munkát (a kimeneti, illetve feldolgozott kereteket, amelyek visszaérkeztek a hozzájuk rendelt processzortól) tároljuk; a várakozási sor buffer vagy memóriaigénye legalább egy keretnyi, processzoronként - az ábrán keret-0 és keret-N jelölik ezeket a helyeket, míg NPU-0 és NPU-N jelölik a keretekhez rendelt processzorokat. Amikor a 112 forgalomirányító a keretet egy feldolgozó egységhez küldi, a

keret azonosítóját továbbítja a 410 második memóriához illetve várakozási sorhoz, amelyben az NPU-0 és NPU-N processzoroknak megfelelő tárolóhelyeket 0-tól N-ig számozzuk. Amennyiben egy m azonosítójú vagy címkéjű keretet az NPU-0-ba küldünk, az NPU-0-nak megfelelő 0 memóriaegységbe m kerül, amely azt jelzi az NPU-0 egy m azonosítójú bemeneti információs egységet dolgoz fel. Később még megemlítjük, hogy egy későbbi m azonosítójú keret még ugyanahhoz az adatfolyamhoz tartozik, míg egy újabb címkét vagy azonosítót tartalmazó keret már egy újabb adatfolyamhoz. Így, ha egy 0 címkéjű bemeneti információs egység érkezik NPU-1-be, akkor az NPU-1-nek megfelelő 1 tárhelybe a 0 érték kerül. Aztán, ha később ugyanennek az adatfolyamnak egy második (szintén 0 címkéjű) bemeneti információs egységét fogadja a 112 forgalomirányító, amely ezt a bemeneti információs egységet az NPU-N-hez rendeli, akkor a N tárhelybe szintén a 0 érték kerül, ezzel mutatva, hogy az információs egységet a N. processzor dolgozza fel.

A 420 harmadik memóriában megjelenik minden címke, amelyet az n feldolgozó egység jelenleg dolgoz fel. Minden címkéhez hozzárendeljük a hozzá tartozó processzor azonosítóját, és mivel a listázás folyamatos, adott üzenetfolyamhoz elsőnek rendelt processzor elsőként jelenik meg a memóriában. Ebben az esetben, az m címkét tekintve, a 422 memóriaegységben levő 0 azt jelenti, hogy NPU-0 dolgozza fel annak az adatfolyamnak az egyik információs egységét; a 424 memóriaegységben levő n azt mutatja, hogy a 0 címkéhez tartozó első egységet az NPU-N, a második egységet - lásd a 426 memóriaegységet - az NPU-1 dolgozza fel. Adott adatfolyam

esetén a forgalomirányítóhoz érkező bemeneti információs egységek sorrendjét úgy kell megőrizni, hogy ugyanezt az adatfolyamot legközelebb is az érkezési sorrendnek megfelelően tudjuk továbbítani; így látható, hogy a 424 és 426 címkéjű memóriaegységekben az NPU-k vagy feldolgozó egységek ugyanabban a sorrendben szerepelnek, mint amilyen sorrendben a bemeneti keretek megérkeztek a hálózatból és lettek továbbítva az N darab processzor felé.

A 8. ábrán az ismertetett adat menedzsment technikákat alkalmazó és bemeneti kereteket feldolgozó 114 befejező egység részletes felépítését látjuk. Ebben a felépítésben a 114 befejező egység a feldolgozó egységek kimenetének (például feldolgozott információs egységek) szétosztása érdekében számos - a 4. ábrán nem látható - körbenjáró (round robin) működésű eszközzel kommunikál. A számos round robin eszköz között található egy 450 előre körbenjáró (up round robin) és két hátrafelé körbenjáró (down round robin) eszköz, amelyek közül a 460 számút cél portok esetén (target ports) (kis számú, specifikusan címzett, gyakran használt port), a 470 számút általános szétosztásra (general distribution) (a specifikusan címzett cél portokon kívüli más portokhoz címzett feldolgozott információ) használjuk.

A 452, 462 és 472 logikai ÉS kapuk kapuznak a megfelelő 450, 460 és 470 round robin eszközök számára. A 450 up round robin kereteit előállító 452 ÉS kapu bemenetei: a keret UP keret (az 510 Kész FCB Oldalhoz (Ready FCB Page) tartozó UP blokkból való), a keret érvényes keret (VF - valid frame) (VF jelzi, hogy a keret érvényes, átvitelre készen áll), a címke mező érvényes a hozzá tartozó keret címke mezőben (M01-től

M92-ig), a címkéhez az adatfolyam legkorábbi (head frame) kerete tartozik.

Amikor a keretet továbbítjuk valamelyik processzornak, a 112 forgalomirányító kétfajta információt juttat el a 480 címke besorolóhoz (label enqueue) - egy címkét a keret számára a 482 vonalon és a kerethez rendelt processzor azonosítóját a 484 vonalon. A keret címkéje azonosítja az adatfolyamot amelyhez a keret tartozik. Jelen felépítés esetén a címkét az MAC meg forrás címe mínusz rendeltetési hely címe (MAC + SA - DA) összefüggés alapján kapjuk meg, így minden adatfolyamhoz egyedi azonosítót kapunk és az azonos adatfolyamhoz tartozó kereteket ugyanavval a címkével, míg a különböző adatfolyamokhoz tartozó kereteket különböző címkével illetve azonosítóval látjuk el.

A 9. ábrán látható az N darab processzorhoz tartozó információk tárolására használt 500 címke mező elem (label field element) formátuma. Az N processzor mindegyikéhez két ilyen címke mezőt rendelünk, az egyik az éppen feldolgozott kerethez, a másik a már feldolgozott, a feldolgozó komplexumból való távozásra váró kerethez tartozik. A továbbításra készen álló, már feldolgozott keretet az 510 memóriában vagy tárban tároljuk le - korábban már hivatkozunk rá Kész FCB Oldal (Ready FCB Page) néven -, ilyen tárolóval az N processzor mindegyike rendelkezik.

Az 500 címke mező elemben található egy L címke, egy H fejrész mező, egy V érvényesség mező, egy T farokrész mező, és egy N következő mező. A L címke az üzenetrészből származik és minden adatfolyam számára egyedi azonosítóként használható. A H fejrész mező egy adatfolyam kezdetét vagy az N feldolgozó

egység által jelenleg kezelt - akár jelenleg feldolgozás alatt álló, akár már feldolgozott és a feldolgozó komplexum elhagyására váró - összefüggő keretek láncát jelzi. Az N processzorból álló feldolgozó komplexumban feldolgozott minden adatfolyamnak az N processzor valamelyikében megtalálható a fejrésze vagy kezdete (vagy adott adatfolyamhoz tartozó, elsőként vett kerete), és ezt a kezdetet „fejként" azonosítva a megfelelő címke mező elem H mezőjébe ilyenkor 1-et írunk. Hasonlóan, minden adatfolyamnak megtalálható az utolsó kerete az N processzor valamelyikében, és ezt az utolsó keretet „farokként" azonosítva a T farokrész mezőjébe ilyenkor 1-et írunk.

A V érvényesség mező azt mutatja meg, hogy a processzor valós (a feldolgozás után biztosan valóssá váló) adatot tartalmaz vagy nem, az előbbit a V mezőbe írt 1, az utóbbit a V mezőbe írt 0 mutatja. Amikor a feldolgozás először megkezdődik még nincsen a rendszerben igazi vagy érvényes adat, így a V érvényesség mező a rendszer kezdeti beállításai során automatikusan 0-ra állítódik be. Később, amikor adott processzor 510 Kész FCB Oldaláról olvasunk ki adatokat, akkor a processzor FCB Oldalához tartozó V érvényesség mezőt 0-ra állítjuk, ezzel jelölve hogy a processzor nem rendelkezik többé érvényes információval adott címkével kapcsolatban (mivel az FCB Oldal információit már továbbadtuk a round robinoknak; ugyanakkor a processzor még mindig rendelkezhet más, a processzorhoz tartozó címke mezőben tárolt érvényes információval, mert a processzor egy másik kereten is dolgozhat). A N következő mező ugyanazon adatfolyam következő keretéhez tartozó címke mezőt jelöli - a N processzorhoz

tartozó 2N címke mezők valamelyikét. A 480 címke besoroló a forgalomirányítótól minden bemeneti információs egységről vagy keretről üzenetet kap, amelyben benne van az adatfolyamba továbbított keret azonosítója és a processzor azonosítója, amelyhez a keretet továbbítottuk.

A 10. ábra a 8. ábrán látható 480 címke besoroló működését mutatja be. Amikor egy keretet a 112 forgalomirányító továbbít az n processzor valamelyikéhez, a 600 blokkban a bemeneti információs egységnek vagy keretnek címkéjét a 482 vonalon keresztül, a keretet feldolgozó processzor azonosítóját a 484 vonalon keresztül továbbítjuk a 480 címke besorolóba. A 480 címke besoroló első 602 lépésében meghatározza, hogy az első címke mező által kijelölt tároló (storage) V érvényességi mezőjében az 1 érték szerepel-e. Amennyiben a V érvényességi mezőbeli érték 1, a kijelölt tároló foglalt és ekkor az adatot más tárolóban kell elhelyezni, ezt jelzi a 606 blokk; különben a kijelölt tárolót használjuk, ezt jelzi a 604 blokk. Következő lépésben, a 650 blokkban a megfelelő tároló V érvényességi mezőjét 1-re állítjuk, ezzel jelölve, hogy abban a tárolóban érvényes adatot tároltunk le, majd a 640 blokkban beállítjuk a tároló T farokrész mezőjét, ezzel jelölve hogy az aktuális adatfolyamnak ez az utolsó darabja (egészen addig, míg ugyanennek az adatfolyamnak a következő keretét nem fogadjuk, ekkor a T farokrész mezőt kezdeti állapotbeli értékére visszaállítjuk (reset)). Következő lépésben a 610 blokkban a címkét összehasonlítjuk a processzorok által jelenleg kezelt címkékkel (természetesen ezen címkék V mezőjének 1-nek kell lennie, hiszen ez jelenti azt, hogy érvényes az adott keret).

Az összehasonlítás eredménye vagy az, hogy címke megegyezik valamelyik jelenleg feldolgozás alatt álló címkével, ekkor a vezérlés a 670 blokkra kerül, vagy az, hogy a címke nem egyezik meg a jelenleg feldolgozás alatt álló címkék egyikével sem, ekkor a vezérlés a 630 blokkra kerül. Amennyiben tehát a címke megegyezik valamelyik aktuális címkével, akkor a valamelyik már létező adatfolyam része, így a 670 blokkban az adatfolyam eddigi végének T farokrész mezőjét reseteljük (azaz T = 0) és a megfelelő címke mező következő mezőre mutató pointerét beállítjuk úgy, hogy a jelenlegi keret helyére mutasson. Aztán a 680 blokkban a H mezőt 0-ra állíjuk, ezzel jelezve hogy az aktuális keret nem az adatfolyam első kerete. Amennyiben a címke nem egyezett meg egyik aktuális címkével sem, úgy a jelenlegi keret egy új adatfolyam és egyben ennek első kerete, ezért a 630 blokkban a H fejrész mezőt 1-re állítjuk. Miután a 630 vagy 680 blokkok végrehajtása megtörtént, azaz lezajlott a megfelelő jelzőbitek - különösen a H fejrész mező - beállítása, az újonnan érkezett keretek már meglévő adatfolyamokhoz kapcsolását és a mezők illetve jelzőbitek beállítását befejezettnek tekintjük.

A 11. ábrán a processzorok által feldolgozott keretek kézbesítését vagy feldolgozását illusztrálja. Először, a 710 blokkban az első mező indikátorát megcseréljük úgy, hogy a pointer egy, az első illetve következő mezők által mutatottól különböző tárolóra mutasson. Majd a 720 blokkban a V érvényességi mezőt 0-ra állítjuk, ezzel jelezve hogy az adat nem érvényes többé (a keretet kifelé irányítjuk, az adat már nem jelent feldolgozás alatt álló keretet). A 725 blokkban ellenőrizzük a T farokrész mezőt, hogy a szóban forgó keret

egy adatfolyam utolsó kerete-e (ha T = 1). Amennyiben igen, a vezérlés a 740 blokkra kerül, ezzel jelezve az eljárás befejeződését. Amennyiben nem, úgy a 730 blokkban beazonosítjuk a következő keretet (a Következő mező pointerének alapján) és annak H fejrész mezőjét vagy jelzőbitjét beállítjuk, ezzel jelölve azt, hogy a jelenleg a processzorokban levő adatfolyamnak ez az első kerete. Aztán, a 730 blokkból a jelzőbitek beállítása után a befejező 740 blokkba lépünk.

A 12. ábrán jelen találmány rendszerét egy példán keresztül mutatjuk be, azt illusztrálva hogy milyen sok adatfolyamot tudunk elhelyezni a 10. ábra kapcsán ismertetett befejező egység és logikáinak segítségével. Az N processzor és a 112 forgalomirányító már dolgozik egy ideje, ezért a 12. ábra egy pillanatfelvétel a befejező egység megfelelő részében tárolt adatokról, különösen tekintettel azok kapcsolatára a címketárolóval. Az ábrán jól látható, a címke besorolót a számos címke memóriához kapcsoltuk, az N processzor mindegyikéhez kettő tartozik. Továbbá minden processzor rendelkezik egy ehhez kapcsolt kimeneti bufferrel (amire Kész FCB Oldal néven is hivatkozunk), amelyben a feldolgozott keretek a három round robinhoz való továbbjutásra várnak. A címke memória párok mindegyikéhez tartozik egy első címke memória, amely azt mutatja meg, hogy melyik címke memória érkezett elsőként (és, amikor mindkét címke memória érvényes, akkor az első címke jelöli azt a keretet amelyik a Kész FCB Oldalon van, míg a második azaz később-kapott címke jelöli a keretet amelyet éppen akkor dolgoz fel a megfelelő processzor). Az ábrán öt különálló adatfolyam látható, habár a

feldolgozásban levő adatfolyamok mindenkori száma függ a rendszertől (különösen annak méretétől és a hálózati forgalomtól) és változhat az idők során. Jelen példában a tíz processzort 0-tól 9-ig a megfelelő sorszámmal jelöljük, és a címke memóriák indexe a 0 processzorhoz tartozó M01 és M02 memóriáktól kezdve a 9 processzorhoz tartozó M91 és M92 memóriákig tart, habár a processzorok száma tervezői döntés és szükség esetén megváltoztatható. Az A azonosítójú első adatfolyam az M01 címke memóriában kezdődik, jól látható a jelzésből (H = 1) hogy az M01 memória (ezzel utalunk arra, hogy a 0 processzorról van szó) az adatfolyamnak vagy láncnak az első elemét jelenti. Az M01 címke memória N Következő mezőjének pointere az M21 címke memóriára mutat, ezzel jelezve, hogy a 2 processzor kezeli ennek az adatfolyamnak a következő információs egységét. Az M21 címke memória Következő mezőjének pointere az M52 címke memóriára mutat, ami azt jelzi, hogy az 5 processzoré az adatfolyam következő része. Az M52 címke memória beállított farokrész mezője azt mutatja, hogy ennek az adatfolyamnak jelenleg ez az utolsó, az N processzor valamelyikénél feldolgozás alatt álló darabja. Ennek a példának az adatfolyam szekvenciáját az M01 címke memóriából az M21 címke memóriába mutató A1, és az M21 címke memóriából az M52 címke memóriába mutató A2 nyilak mutatják, amelyeknek célja az adatfolyam elemek közti logikai kapcsolat illusztrálása (a nyilak logikailag a Következő mező pointereit reprezentálják, az aktuális fizikai megvalósításban nem léteznek). Hasonlóan, az M02 címke memóriából az M11 címke memóriára mutató A3 nyíl ugyanazon adatfolyam további sorrendjét mutatja (habár ez az adatfolyam különbözik az M01,

M21, M52 címke memóriákkal kapcsolatban leírt adatfolyamtól). A harmadik adatfolyamot az M31 és M42 címke memóriákhoz kapcsolódó A4 nyíl, a negyedik adatfolyamot az M71 és M72 címke memóriákhoz kapcsolódó A5 nyíl jelzi. Végül, az ötödik adatfolyamot az M41 címke memória jelöli, amelyhez nem tartozik nyíl, hiszen ez egy jelenleg csak egy címke memóriát foglaló adatfolyam. Ez az M41 címke memória az adatfolyam feje és farokrésze is egyben, és Következő mezője nincsen, hiszen nem kapcsolódik hozzá másik címke memória.

Emlékezzünk arra, hogy abban az esetben, mikor egyetlen processzorhoz két címke memória tartozik, az egyik címke memória az egy bufferben letárolt, már befejezett illetve feldolgozott információs egységeket jelenti - erre Kész FCB Oldalként is szoktunk hivatkozni - amelyek készen állnak arra, hogy a feldolgozó komplexumból a megfelelő round robin eszközhöz továbbítsuk őket, előre vagy hátrafelé, attól függ; előre továbbítás esetén az információs egységet az interfész eszközhöz, hátrafelé való továbbítás esetén az információs egységet visszafelé, az adatátviteli hálózat irányába továbbítjuk. Ebben az esetben a 0, 4 és 7 processzorokhoz tartozó címke memória párokban adat található. Minden Kész FCB Oldalhoz tartozik továbbá egy UP mező (azt mutatja, hogy előre vagy hátrafelé továbbítandó oldalról van szó), és egy indikátor, amely - amennyiben hátrafelé továbbítandó oldalról van szó - jelzi azt, hogy a keretet egy cél porthoz vagy egy általános porthoz címeztük; ennek alapján továbbítjuk a lapot a cél portokhoz vagy az általános portokhoz használt round robinhoz. Amennyiben a 0 processzor által korábban megkapott keret címke memóriája M02, a keret az adatfolyam feje és a

processzor komplexumból és annak bufferéből a következő átvitelkor a keretet az up round robin eszköz döntése alapján felfelé visszük át, akkor az FCB Oldalt és a hozzá tartozó adatmezőket elmozdítjuk a Kész FCB Oldalról és az információt az up round robinba visszük át. Majd az első címke indikátort átbillentjük, ezzel jelezve hogy most már a 0 processzorhoz tartozó M01 címke memória az első, továbbá az M02 címke memória V érvényesség mezőjét 0-ra állítjuk, ezzel jelezve, hogy ez a címke memória nem aktív illetve nem érvényes többé, végül visszaállítjuk az alapértékére a megfelelő FCB Oldal VF érvényes mezőjét (0-ra állítjuk).

Jelen találmány a már meglevő adatfolyamok zavarása nélkül támogatja az új adatfolyamokat úgy, hogy nincsen szükség előzetes információra az új adatfolyammal kapcsolatban. Egy új adatfolyamot jelentő csomagot (például az egyik feldolgozó egység státuszát leíró üzenetet) egyszerűen az azonosító adataival tárolunk le, nem hivatkozunk más adatfolyamra. Az azonosító hiánya miatt nem fog megegyezni egyik - már meglévő - adatfolyam azonosítóval sem, beállítjuk a „nincs címke" mezőt (no label field) és az üzenet bármikor továbbítható.

Jelen találmányban lehetőség van a flush parancs végrehajtására is, amivel felülírhatjuk azt a tulajdonságot hogy a rendszer az adatfolyamokat a megfelelő sorrendben dolgozza fel, úgy, hogy megengedjük, hogy a rendszer a befejezett kereteket érkezési sorrendben dolgozza fel, nem törődve a címke mezők láncolásával (a Következő pointerekkel és azzal a feltétellel, hogy egy adott keret - mielőtt az őt továbbító round robinokhoz ér - az adatfolyam feje kell, hogy

legyen). Mindez megvalósítható az FCB Oldal „nincs címke" mezőjének segítségével.

Egy különálló adatfolyam egészen addig blokkolódik, míg fel nem dolgoztuk az adatfolyam fejét, hiszen a normál működés során (flush nélkül) a befejező egység csak az üzenetfolyam fejét alkotó kereteket tudja a round robinok felé irányítani. Ugyanakkor minden adatfolyamnak megvan a saját feje, így az egyik blokkolódása esetén megszakítás és zavarás nélkül folytatódhat a többi adatfolyam feldolgozása és a befejezett információs egységek továbbítása a round robin eszközök felé. Különösen hasznos, hogy amennyiben egyetlen adatfolyamot állítunk meg (például mert egy processzor meghibásodik vagy nem tudja feldolgozni az adatfolyam egy elemét), attól még a többi adatfolyamot nem állítjuk meg. Máskülönben a teljes feldolgozást le kellene állítani amíg azt az egy adatfolyam blokkolódást megszüntetjük.

Természetesen, - a találmány előbbi leírását összevetve a kísérő rajzokkal - a szakmában jártas olvasó számára a találmánnyal kapcsolatos számos módosítási lehetőség válhat nyilvánvalóvá. Például, a minősítő aktuális hardver megvalósítása számos tervezői döntés tárgya, az ismertetett döntések függnek az üzenet tartalmától, az egységbezárási eljárástól és az elvégzendő feldolgozástól. A rendszer implementáción és az üzenet konfiguráción számos - a rendszer által kezelhető - módosítást hajthatunk végre úgy, hogy nem térünk el jelen találmány szellemiségétől. A letárolt címkéket nem csak az üzenet tartalma, hanem számos más módszer alapján is létrehozhatjuk, akár egyszerűen a forgalomirányító által azonosított adatfolyamok folyamatos számozásával is. Ezen

kívül számos más módon adaptálhatjuk és módosíthatjuk előnyösen a rendszert anélkül hogy jelen találmány szellemiségétől eltérnénk. Ezért a kiviteli példa előbbi leírása csupán illusztrálja jelen találmány vezérelveit, és nem korlátozza azokat.

Szabadalmi igénypontok

1. Berendezés, amelyben megtalálható:

egy félvezető hordozó;

a hordozón kialakított N darab feldolgozó egység, ahol N>1;

a hordozón kialakított első belső adat memória, amely adat memóriát az N darab feldolgozó egység által elérhető információ tárolására használjuk;

az N darab feldolgozó egységhez művi úton csatolt forgalomirányító, amelynek segítségével bemeneti információs egységet veszünk és továbbítunk az N darab feldolgozó egység valamelyikéhez;

egy, a forgalomirányítóhoz kapcsolt minősítő, amely minősítőben található egy összehasonlító egység, amely összehasonlító egységgel meghatározzuk a bemeneti információs egység adatformátumát és amellyel kimeneti indikátorokat generálunk a bemeneti információs egység számára, kimeneti indikátorokat generálunk amelyek a bemeneti információs egység adatformátumát mutatják, továbbá kezdőcímet generálunk a bemeneti információs egységhez, és a kimeneti indikátorokat és a kezdőcímet a belső adat memóriában tároljuk; az indikátorok és a kezdőcím az N darab processzor valamelyikének rendelkezésére állnak és a processzor fel is használja azokat a bemeneti információs egység feldolgozása során; továbbá

egy befejező egység, amelyet a félvezető hordozón alakítunk ki és amelyet művi úton az N darab feldolgozó egységhez csatolunk hogy fogadja az N darab feldolgozó egység valamelyike által feldolgozott információs egységet.

2. Az 1. igénypont szerinti berendezés, azzal jellemezve, hogy az összehasonlító egységben végrehajtunk egy vizsgálatot amely megmutatja hogy a bemeneti információs egység tartalmaz-e látszólagos helyi hálózat mezőt, továbbá azzal jellemezve, hogy a generált kimeneti indikátorokban található egy indikátor, amely megmutatja, hogy a bemeneti információs egység tartalmaz-e látszólagos helyi hálózat mezőt.

3. Az 1. vagy 2. igénypont szerinti berendezés, azzal jellemezve, hogy a minősítő számos, a hordozón kialakított hardver eszközt tartalmaz.

4. Az 1., 2. vagy 3. igénypont szerinti berendezés, azzal jellemezve, hogy a kimeneti indikátorok tartalmaznak egy indikátort, amely azonosítja a bemeneti információs egység típusát, továbbá tartalmaznak egy indikátort, amely azonosítja a bemeneti információs egység 2 rétegbeli egységbezárási technikáját.

5. Az 1., 2., 3. vagy 4. igénypont szerinti berendezés, azzal jellemezve, hogy az indikátorok tartalmaznak egy default kód belépési pontot.

6. Az 1 - 5. igénypontok bármelyike szerinti berendezés, azzal jellemezve, hogy a minősítőben található egy rendszer, amellyel meghatározhatjuk a kód belépési pontot a bemeneti információs egységnek a minősítő által meghatározott típusa és egységbezárási technikája alapján.

7. Az 1 - 6. igénypontok bármelyike szerinti berendezés, azzal jellemezve, hogy a minősítőben található egy rendszer, amellyel a bemeneti információs egység típusa alapján meghatározhatjuk a default kód belépési pontot és a kód belépési pontot.

8. A 7. igénypont szerinti berendezés, azzal jellemezve, hogy a berendezés tartalmaz egy kiválasztót, amely a bemeneti információs egység típusa alapján választ a default kód belépési pont és a kód belépési pont közül.

9. A 8. igénypont szerinti berendezés, azzal jellemezve, hogy tartalmaz egy rendszert, amely meghatározza hogy a bemeneti információs egység tartalmaz-e látszólagos helyi hálózat információt.

10. Az 1 - 9. igénypontok bármelyike szerinti berendezés, azzal jellemezve, hogy az indikátorok tartalmazzák az N darab feldolgozó egységből annak az azonosítóját, amelyhez a bemeneti információs egységet rendeltük.

11. A 10. igénypont szerinti berendezés, azzal jellemezve, hogy tartalmaz egy jelzőbitet, amely azt mutatja hogy a feldolgozott információs egységeket a hordozóról olyan sorrendben továbbítjuk, amilyen sorrendben fogadtuk az információs egységeket; a befejező egység a jelzőbit értékének megfelelően továbbítja a feldolgozott információs egységeket amint az N darab feldolgozó egység a feldolgozást befejezte.

12. A 11. igénypont szerinti berendezés, azzal jellemezve, hogy a minősítőben található egy rendszer, amely minden bemeneti információs egységnek generál egy azonosítót - az azonosítót a belső adat memóriában tároljuk - , amely azonosító jelzi az információs egység adatfolyamát és összekapcsolja ugyanazon adatfolyam későbbi információs egységeit ugyanazon adatfolyam korábbi információs egységeivel, ahol a korábbi információs egységet a processzorokban egy adott adatfolyam első információs egységeként azonosítjuk, és a feldolgozó egységekből átvitt

információs egységek azokra az információs egységekre korlátozódnak, amelyeket egy adott adatfolyam első információs egységeként azonosítottunk.

13. Az 1 - 12. igénypontok bármelyike szerinti berendezés, azzal jellemezve, hogy a forgalomirányító egység egy sorban folytonosan tárolja az információs egységek azonosítóit és a feldolgozó egység azonosítóját, amelyhez az információs egységet feldolgozás céljából továbbítottuk; és ahol

a befejező egységet összekapcsoljuk a folytonos sorral, és a befejező egység felhasználja a forgalomirányító által az információs egységekhez rendelt azonosítót és a feldolgozó egység azonosítóját amelyhez az információs egységet továbbítottuk, annak érdekében, hogy a feldolgozott információs egységeket ugyanabban a sorrendben állítsa össze mint amilyen sorrendben a bemeneti információsegységeket fogadtuk.

14. A 13. igénypont szerinti berendezés, azzal jellemezve, hogy a berendezéshez tartozik egy jel, amivel a sorrend felülírható és a feldolgozott információs egységeket abban a sorrendben továbbítjuk a hálózatba, amilyen sorrendben az információs egységek elkészülnek.

15. Eljárás bemeneti információs egységek feldolgozására, az alábbi lépésekkel:

a forgalomirányító fogadja a bemeneti információs egységet;

a forgalomirányítóból az információs egységet a számos processzor valamelyikéhez küldjük feldolgozásra;

amíg az információs egységet a forgalomirányítóból a számos processzor valamelyikéhez küldjük, kiolvassuk az információs egység néhány meghatározott bitjét;

az információs egység kiolvasott bitjeit összevetjük az előre meghatározott típusú információs egységeket azonosító ismert indikátorokkal, így azonosítjuk a bemeneti információs egység típusát és protokollját, vagy megállapítjuk azt, hogy a bemeneti információs egység nem felel meg egyik előre meghatározott bemeneti információs egység típusának sem; majd

a bemeneti információs egység bitjeinek vizsgálati eredményei alapján letároljuk a bemeneti információs egység típusának indikátorát és azzal a bemeneti információs egységgel kapcsolatos más információt; majd

a számos feldolgozó egység valamelyikében a letárolt indikátorokat és a bemeneti információs egységgel kapcsolatos egyéb információt felhasználjuk a bemeneti információs egység feldolgozása során.

16. A 15. igénypont szerinti eljárás, azzal jellemezve, hogy az indikátorok generálása és letárolása akkor történik meg, amikor a bemeneti információs egységet a számos feldolgozó egység valamelyikéhez továbbítjuk, úgy, hogy amikor a számos processzor valamelyike feldolgozza az információs egységet, akkor már előállítottuk és letároltuk az indikátorokat és az egyéb információt, és a számos feldolgozó egység valamelyike adott bemeneti információs egység feldolgozás során felhasználja az indikátorokat és az egyéb információt.

17. A 15. vagy 16. igénypont szerinti eljárás, azzal jellemezve, hogy kiegészül egy lépéssel, amelyben a bemeneti

információs egység tartalma alapján a bemeneti információs egység további feldolgozásához kezdőcímet generálunk, továbbá kiegészül egy lépéssel, amelyben a számos feldolgozó egység valamelyikénél a letárolt indikátorok felhasználásába beletartozik a kezdőcím felhasználása is.

18. A 15., 16. vagy 17. igénypont szerinti eljárás, azzal jellemezve, hogy a bitek kiolvasását és vizsgálatát hardver eszközökkel valósítjuk meg, miáltal a feldolgozást kevesebb feldolgozási ciklussal valósítjuk meg, mintha a bitek kiolvasását és vizsgálatát letárolt utasítások sorozatával valósítanánk meg.

19. A 18. igénypont szerinti eljárás, azzal jellemezve, hogy azt a lépést, amelyben a bemeneti információs egység típusának azonosítását és az indikátorok letárolását hardver eszközökkel valósítjuk meg, két gépi ciklus során hajtjuk végre, miáltal a számos feldolgozó egység valamelyike által végrehajtott, az indikátorokat felhasználó lépés hamarabb végrehajtódik, mintha a vizsgálatot programozott utasítások sorozatos végrehajtásával valósítottuk volna meg.

20. A 19. igénypont szerinti, információs egységeket feldolgozó eljárás, azzal jellemezve, hogy a lépés - amelyben a bemeneti információs egységet vizsgáljuk és indikátorait letároljuk, és a lépés - amelyben a forgalomirányítóból érkező bemeneti információs egységet fogadja a számos feldolgozó egység valamelyike, átlapolódnak.

21. A 15 - 20. igénypontok bármelyike szerinti eljárás, azzal jellemezve, hogy kiegészül az alábbi lépésekkel:

a bemeneti információs egység számára azonosítót generálunk és letároljuk azt;

az azonosító mellett letároljuk annak a proceszornak az azonosítóját, amelyik processzorhoz az információs egységet rendeltük; továbbá

az azonosítót és az információs egységhez rendelt feldolgozó egység azonosítóját használva a feldolgozott információs egységeket érkezésük sorrendjében továbbítjuk.

22. A 21. igénypont szerinti eljárás, azzal jellemezve, hogy kiegészül azzal a lépéssel, hogy a keretben „nincs címke" jelzőbit használata esetén válaszolni tudunk a feldolgozó egység által generált keretre.

23. A 22. igénypont szerinti, információs egységeket feldolgozó eljárás, azzal jellemezve, hogy a rendszer úgy válaszol egy „nincs címke" jelzőbites keretre, hogy a keretet további tárolás nélkül továbbítja a hálózat felé.

24. Eljárás bemeneti keret azonosítására és a kerethez tartozó indikátorok előállítására a keret további feldolgozásához, amely eljárás lépései a következők:

a bemeneti keret alapján meghatározzuk az egységbezárás típusát és a protokollt, úgy, hogy a bemeneti keret megfelelő részeit összevetjük az egységbezárás típusára és a protokoll típusára utaló előre meghatározott bitsorozatokkal;

minden bemeneti kerethez egységbezárás típus és protokoll típus indikátorokat generálunk és tárolunk le;

meghatározzuk és letároljuk a bemeneti keret 3. rétegbeli fejrészének a helyét (location); továbbá

a meghatározott protokoll típus és az egységbezárási eljárás alapján meghatározzuk és letároljuk a bemeneti keret további feldolgozásához szükséges kezdőpontot, miáltal a hely

(location) és a kezdőpont a bemeneti keret további feldolgozása során felhasználható.

25. A 24. igénypont szerinti, a bemeneti keret jellemzőit meghatározó eljárás, azzal jellemezve, hogy a további feldolgozáshoz szükséges kezdőpont meghatározása kiegészül azzal a lépéssel, hogy a bemeneti keret alapján meghatározzuk a default kód belépési pontot, majd a bemeneti keret protokollja és egységbezárási eljárása alapján meghatározzuk, hogy a letárolt vezérlés belépési pont ahhoz az egységbezárási és protokoll kombinációhoz lett-e letárolva; így amennyiben létezik ez a kombináció, akkor a letárolt vezérlés belépési pontot használjuk a további feldolgozás kezdőpontjaként, ellenkező esetben a default kód belépési pontot használjuk a további feldolgozás kezdőpontjaként.

26. Szerkezet hálózatból érkező adatcsomagok vételére és különböző formátumú adatcsomagok feldolgozására, amelynek részei:

számos, egymástól függetlenül működő processzor, amelyekkel adatcsomagokat dolgozunk fel és amelyek által előállított kimeneti adatcsomag a bemeneti adatcsomagon alapul;

a processzorokhoz csatlakoztatott forgalomirányító egység, amely fogadja a hálózatból érkező csomagot és a számos független processzor valamelyikéhez továbbítja azt;

a forgalomirányító egységhez csatlakoztatott minősítő eszköz, amely fogadja a csomagot, meghatározza annak protokollját és egységbezárási technikáját, valamint a keret további feldolgozásához szükséges kezdőcímet, a feldolgozást a feldolgozó egységek végzik; a minősítő eszköz részei:

a keretnek egy része alapján az egységbezárási technikát meghatározó logika;

a keretben levő látszólagos helyi hálózat információ meglétét meghatározó logika; továbbá

a keretek mindegyikéhez tartozó kimenet, amely tartalmazza az egységbezárás típusát és a további feldolgozáshoz szükséges kezdőcímet.

27. A 26. igénypont szerinti szerkezet, azzal jellemezve, hogy a minősítő hardver elemekből áll, letárolt program nélkül.

28. A 26. igénypont szerinti szerkezet, azzal jellemezve, hogy a minősítést két ciklusban hajtjuk végre úgy, hogy a keret további feldolgozása a feldolgozó egységek valamelyikénél a keret forgalomirányító egységből való továbbítása után két cikluson belül megkezdődhet.

29. A 26. igénypont szerinti szerkezet, azzal jellemezve, hogy a kezdőcímet úgy határozzuk meg, hogy a keret alapján egy default kezdőcímet generálunk és a keret feldolgozása során kezdőcímként ezt a default kezdőcímet használjuk, kivéve azt az esetet, amikor a minősítő rendszer által meghatározott egységbezárási eljáráshoz és protokollhoz a fentitől különböző kezdőcímet tároltunk le.

30. A 26. igénypont szerinti szerkezet, azzal jellemezve, hogy a szerkezet tartalmaz egy, nem a hálózatból érkezett új információs egységeket feldolgozó kezelő rendszert, amely kezelő rendszer megjelöli az új információs egységeket, ezzel mutatva azt, hogy ezek az új információs egységek nem a hálózatból érkeztek.

31. Szerkezet változó protokollal és egységbezárással rendelkező információs keret analizálására, továbbá a keret feldolgozásához szükséges kezdő hely (starting location) és a keret feldolgozásához szükséges kezdő utasításra mutató pointer előállítására, amely szerkezetnek részei:

komparátor, amely a keret előre meghatározott bájtjai alapján meghatározza, hogy azok a bájtok hosszúságot vagy protokollt jelentenek-e;

a keret protokollját és egységbezárási rendszerét meghatározó logika;

és amely szerkezet a protokoll és az egységbezárási rendszer alapján meghatározza a keret feldolgozásához szükséges kezdő helyet, továbbá a pointer alapján meghatározza a keret feldolgozásához szükséges kezdő utasítást.

A meghatalmazott

Dr. Köteles Zoltán
szabadalmi ügyvivő
az S.B.G. & K. Szabadalmi Ügyvivői Iroda
tagja
H-1062 Budapest, Andrássy út 113.
Telefon: 461-1000 Fax: 461-1099

FIG. 1

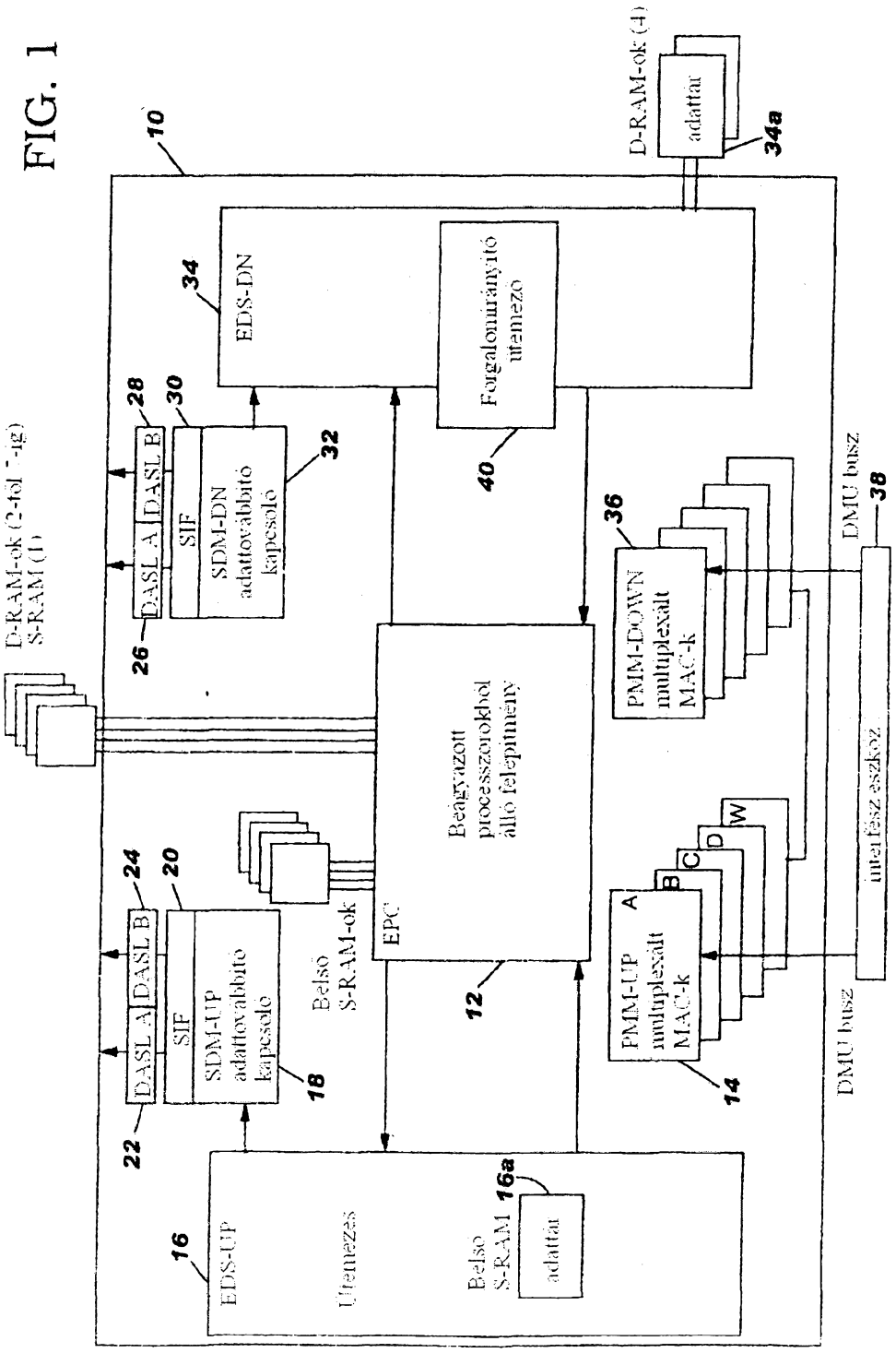

FIG. 2
AZ EPC BLOKKDIAGRAMJA

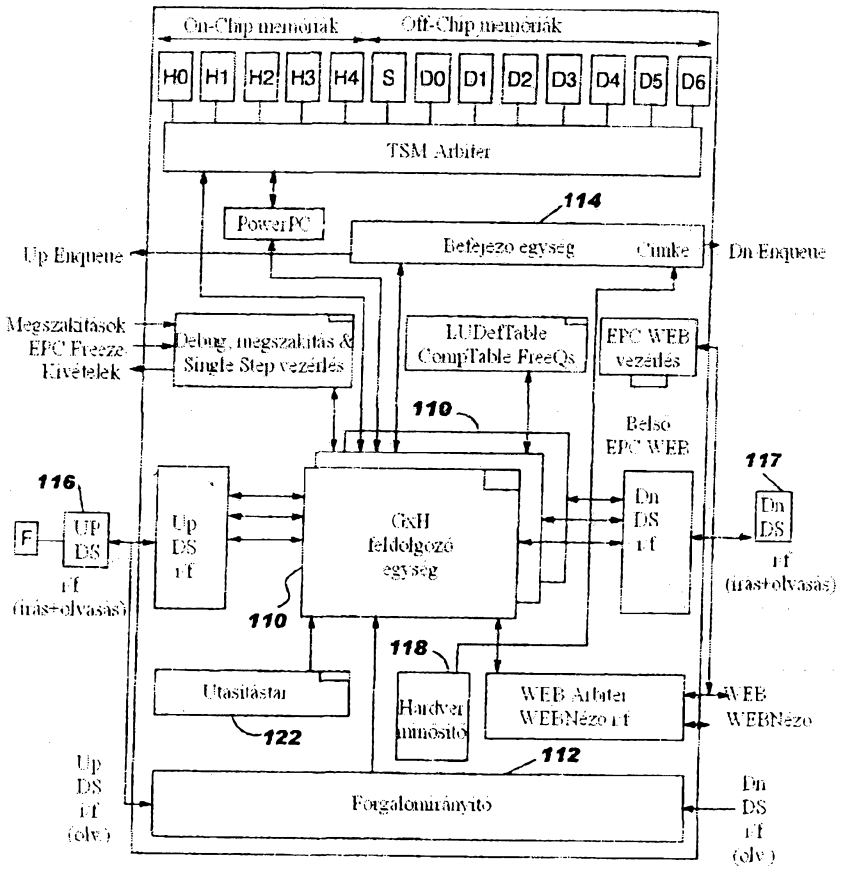

FIG. 3A

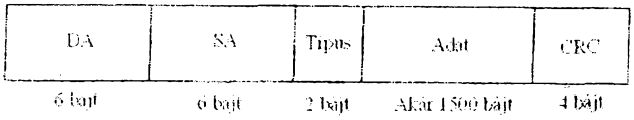

FIG. 3B

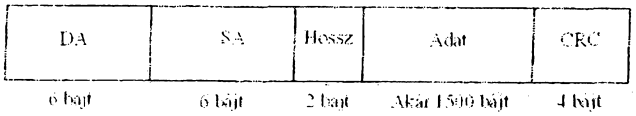

FIG. 3C

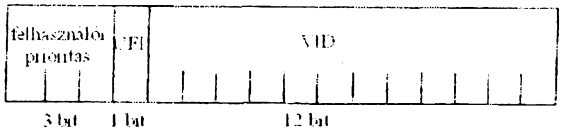

FIG. 3D

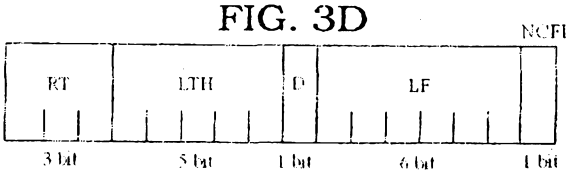

FIG. 3E

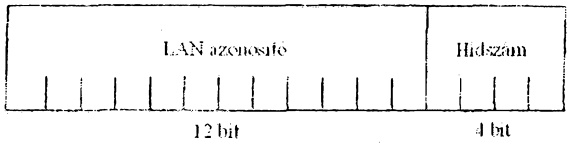

FIG. 3F

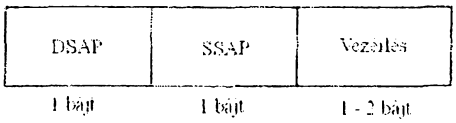

FIG. 3G

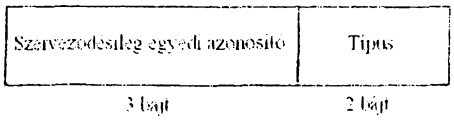

FIG. 3H

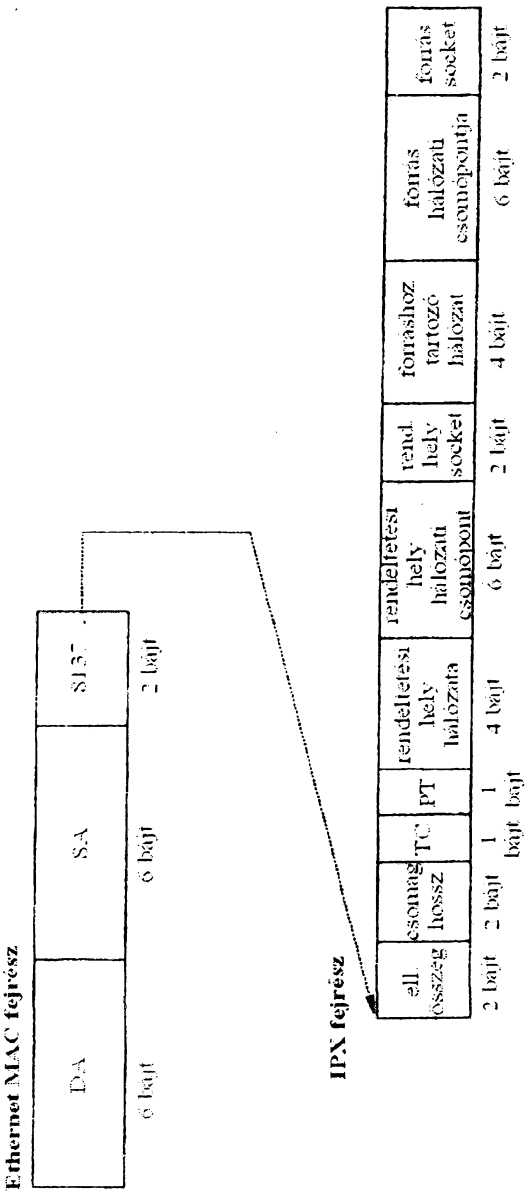

FIG. 3I

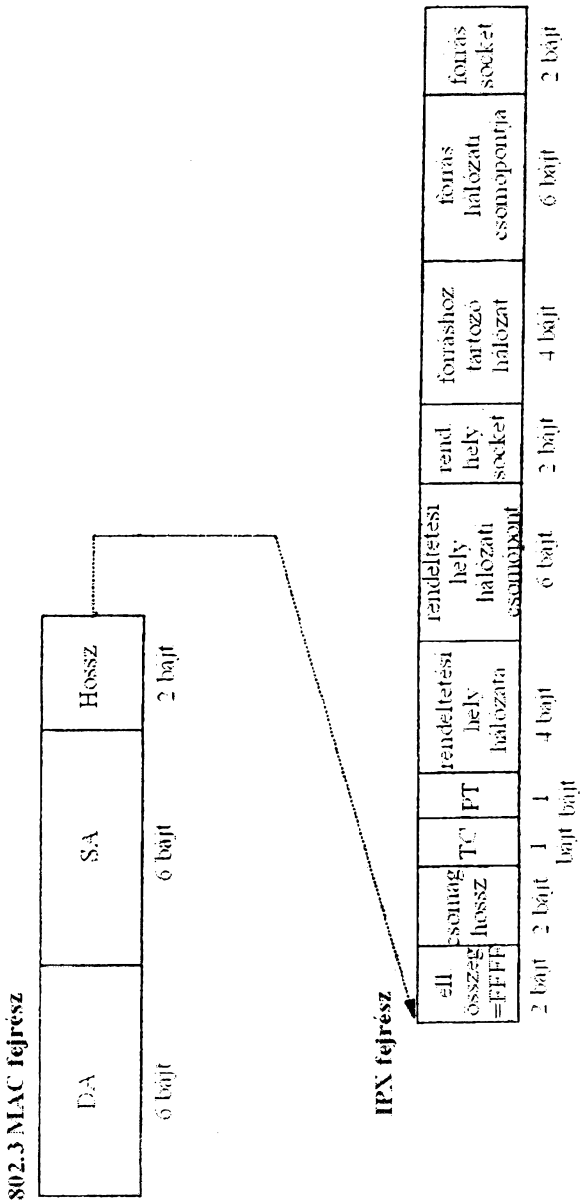
802.3 MAC fejrész
DA
SA
Hossz
6 bájt
6 bájt
2 bájt
IPX fejrész
ell összeg =FFFF
csomag hossz
TC
PT
rendeltetési hely hálózata
rendeltetési hely hálózati csomópont
rend. hely socket
forráshoz tartozó hálózat
forrás hálózati csomópontja
forrás socket
2 bájt
2 bájt
1 bájt
1 bájt
4 bájt
6 bájt
2 bájt
4 bájt
6 bájt
2 bájt

FIG. 3J

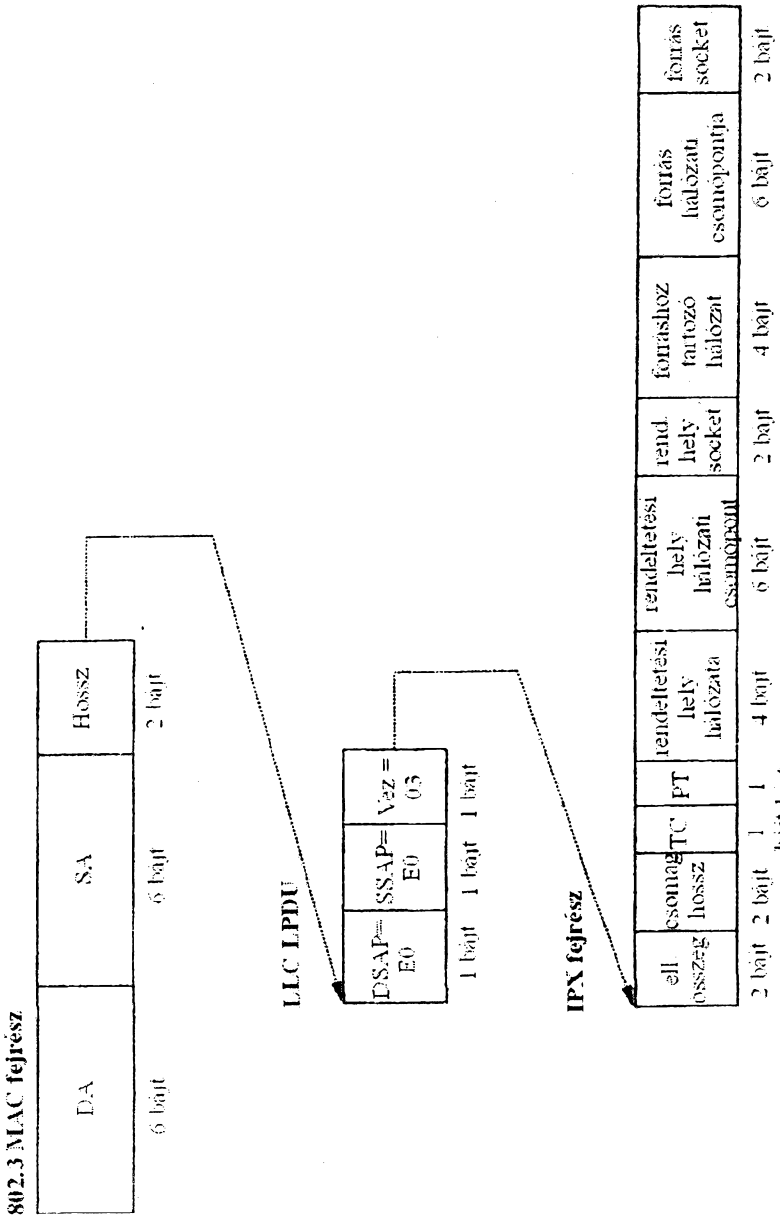
802.3 MAC fejrész
DA
SA
Hossz
6 bájt
6 bájt
2 bájt
LLC LPDU
DSAP= E0
SSAP= E0
Vez = 03
1 bájt
1 bájt
1 bájt
IPX fejrész
ell összeg
csomag hossz
TC
PT
rendeltetési hely hálózata
rendeltetési hely hálózati csomópont
rend. hely socket
forráshoz tartozó hálózat
forrás hálózati csomópontja
forrás socket
2 bájt
2 bájt
1 bájt
1 bájt
4 bájt
6 bájt
2 bájt
4 bájt
6 bájt
2 bájt

FIG. 3K

FIG. 3L

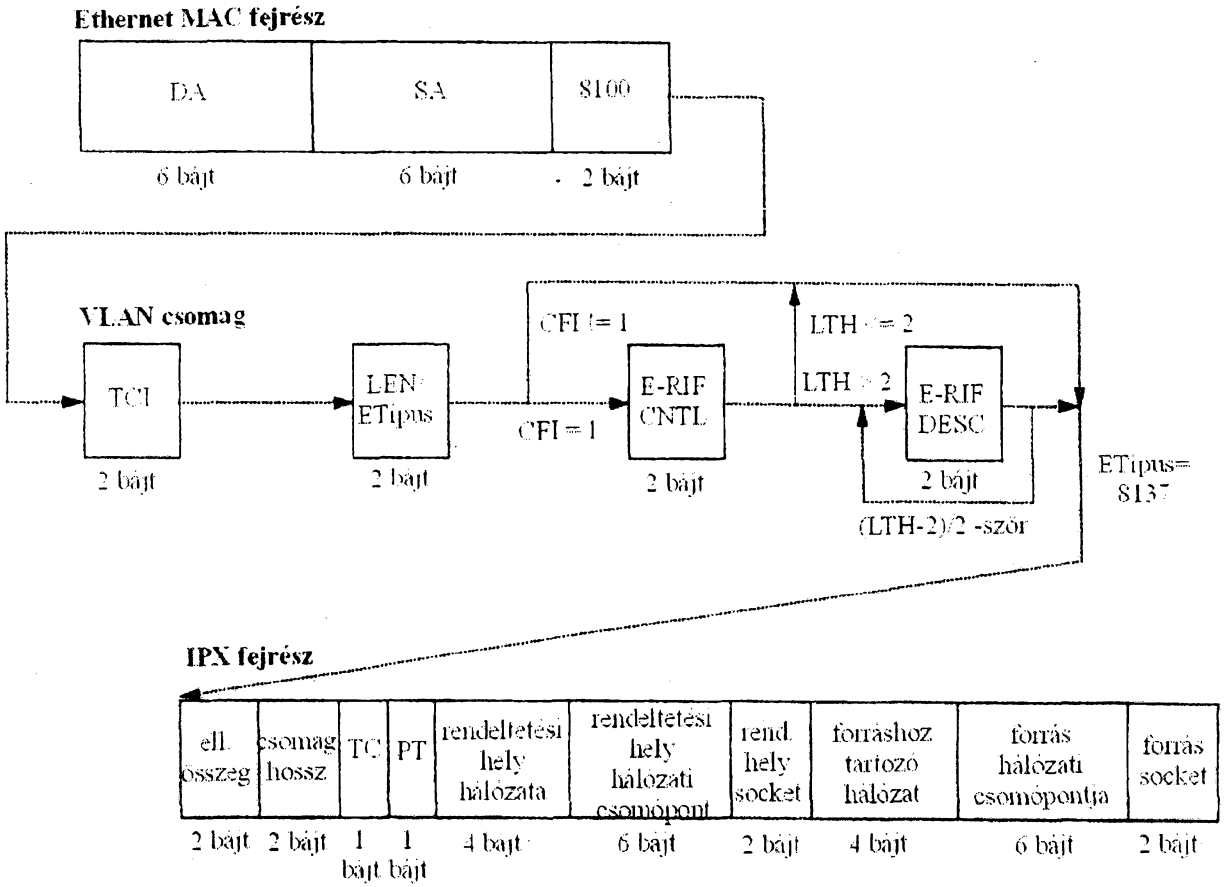

FIG. 3M

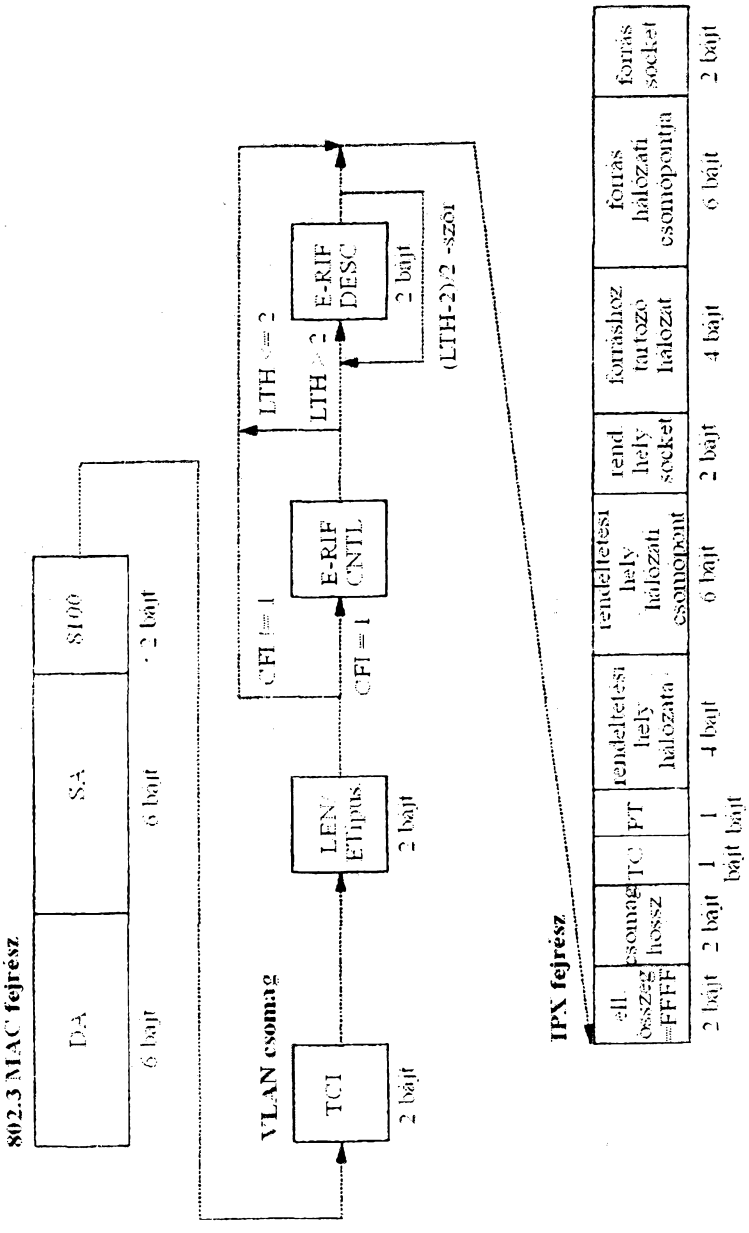
802.3 MAC fejrész
DA
SA
8100
6 bájt
6 bájt
2 bájt
VLAN csomag
TCI
2 bájt
LEN/ ETípus
2 bájt
CFI = 1
E-RIF CNTL
LTH <= 2
LTH > 2
E-RIF DESC
2 bájt
(LTH-2)/2 -szer
IPX fejrész
áll. összeg =FFFF
csomag hossz
TC
PT
rendeltetési hely hálózata
4 bájt
rendeltetési hely hálózati csomópont
6 bájt
rend hely socket
2 bájt
forráshoz tartozó hálózat
4 bájt
forrás hálózati csomópontja
6 bájt
forrás socket
2 bájt
2 bájt
2 bájt
1 bájt
1 bájt

FIG. 3N

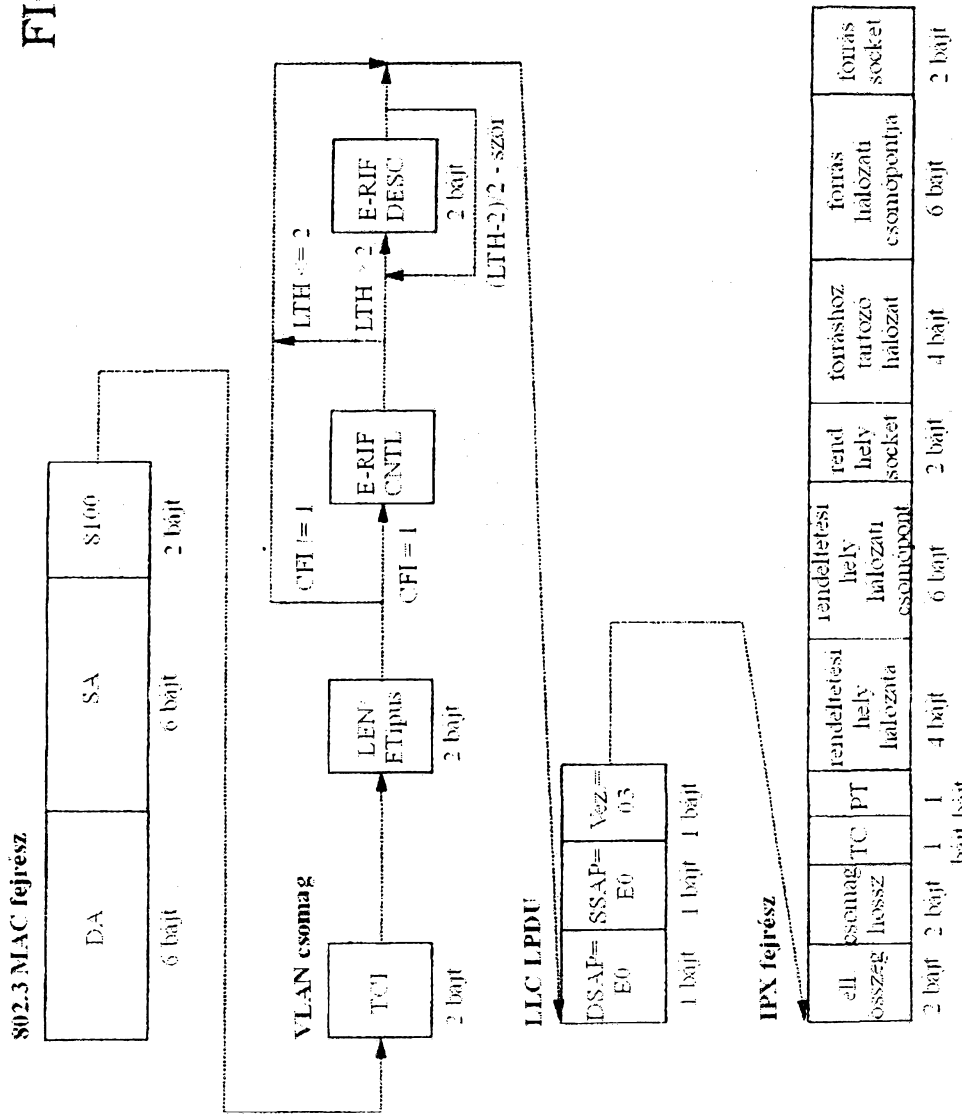
802.3 MAC fejrész
DA
SA
8100
6 bájt
6 bájt
2 bájt
VLAN csomag
TCI
LEN/ ETípus
2 bájt
2 bájt
CFI = 1
CFI = 1
E-RIF CNTL
LTH = 2
LTH > 2
E-RIF DESC
2 bájt
(LTH-2)/2 - szer
LLC LPDU
DSAP= E0
SSAP= E0
Vez= 03
1 bájt
1 bájt
1 bájt
IPX fejrész
ell. összeg
csomag hossz
TC
PT
rendeltetési hely hálózata
rendeltetési hely hálózati csomópont
rend. hely socket
forráshoz tartozó hálózat
forrás hálózati csomópontja
forrás socket
2 bájt
2 bájt
1 bájt
1 bájt
4 bájt
6 bájt
2 bájt
4 bájt
6 bájt
2 bájt

FIG. 30

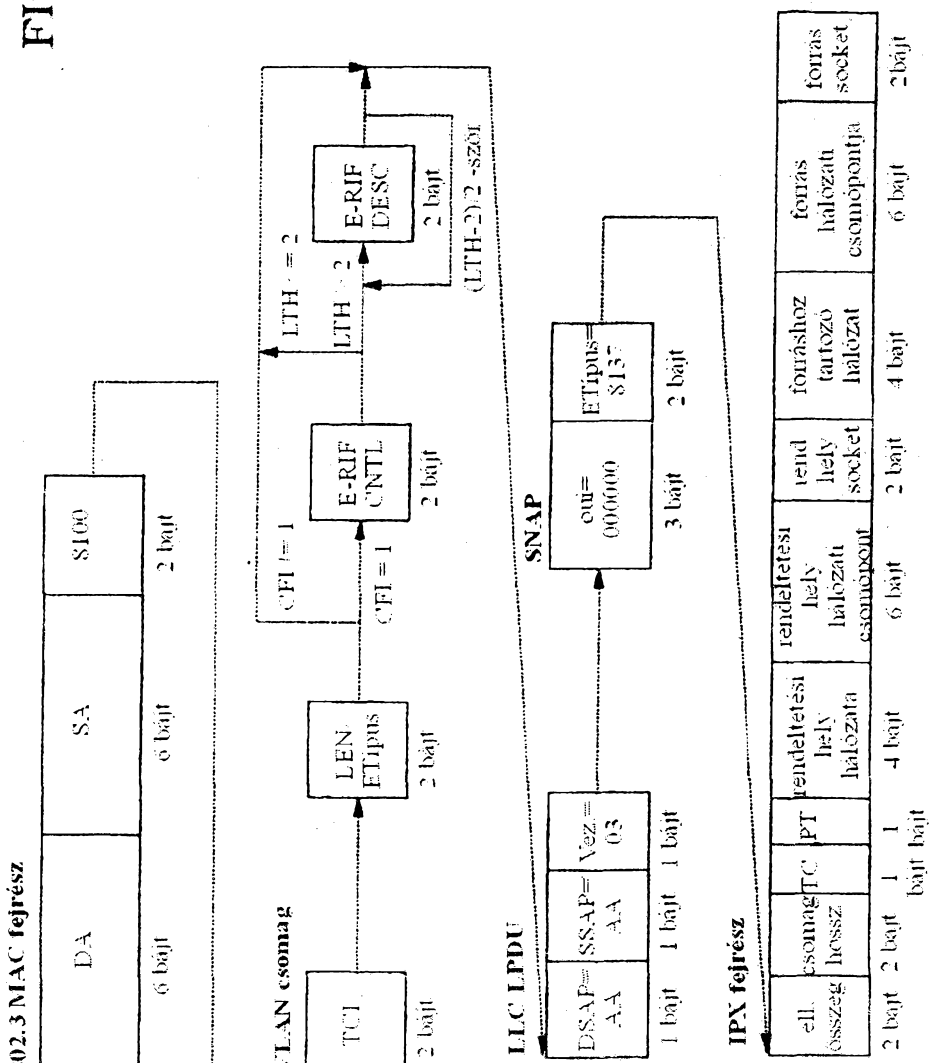
802.3 MAC fejrész
DA
SA
8100
6 bájt
6 bájt
2 bájt
VLAN csomag
TCI
LEN ETípus
E-RIF CNTL
E-RIF DESC
2 bájt
2 bájt
2 bájt
2 bájt
CFI = 1
CFI = 1
LTH = 2
LTH = 2
(LTH-2):2 -szer
LLC LPDU
DSAP= AA
SSAP= AA
Vez.= 03
1 bájt
1 bájt
1 bájt
SNAP
oui= 000000
ETípus= 8137
3 bájt
2 bájt
IPX fejrész
ell. összeg
csomag hossz
TC
PT
rendeltetési hely hálózata
rendeltetési hely hálózati csomópont
rend. hely socket
forráshoz tartozó hálózat
forrás hálózati csomópontja
forrás socket
2 bájt
2 bájt
1 bájt
1 bájt
4 bájt
6 bájt
2 bájt
4 bájt
6 bájt
2bájt

FIG. 3P

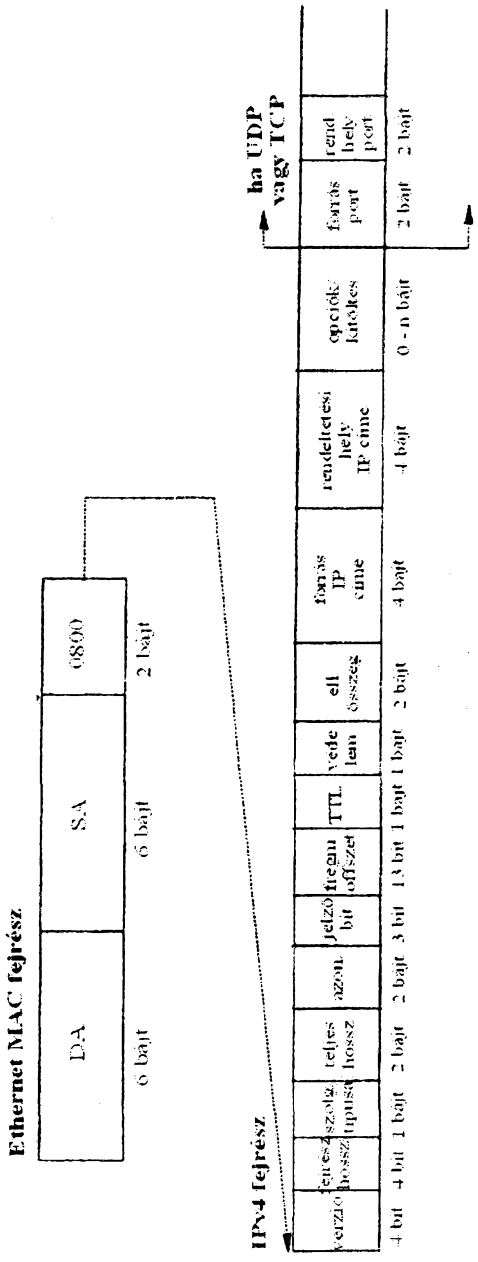
Ethernet MAC fejrész
DA
SA
0800
6 bajt
6 bajt
2 bajt
IPv4 fejrész
verzió
fejrész hossz
szolg. típusa
teljes hossz
azon.
jelző bit
tiegm. offszet
TTL
véde lem
ell összeg
forrás IP címe
rendeltetési hely IP címe
opciók kitöltés
ha UDP vagy TCP
forrás port
rend. hely port
4 bit
4 bit
1 bajt
2 bajt
2 bajt
3 bit
13 bit
1 bajt
1 bajt
2 bajt
4 bajt
4 bajt
0 - n bajt
2 bajt
2 bajt

FIG. 39

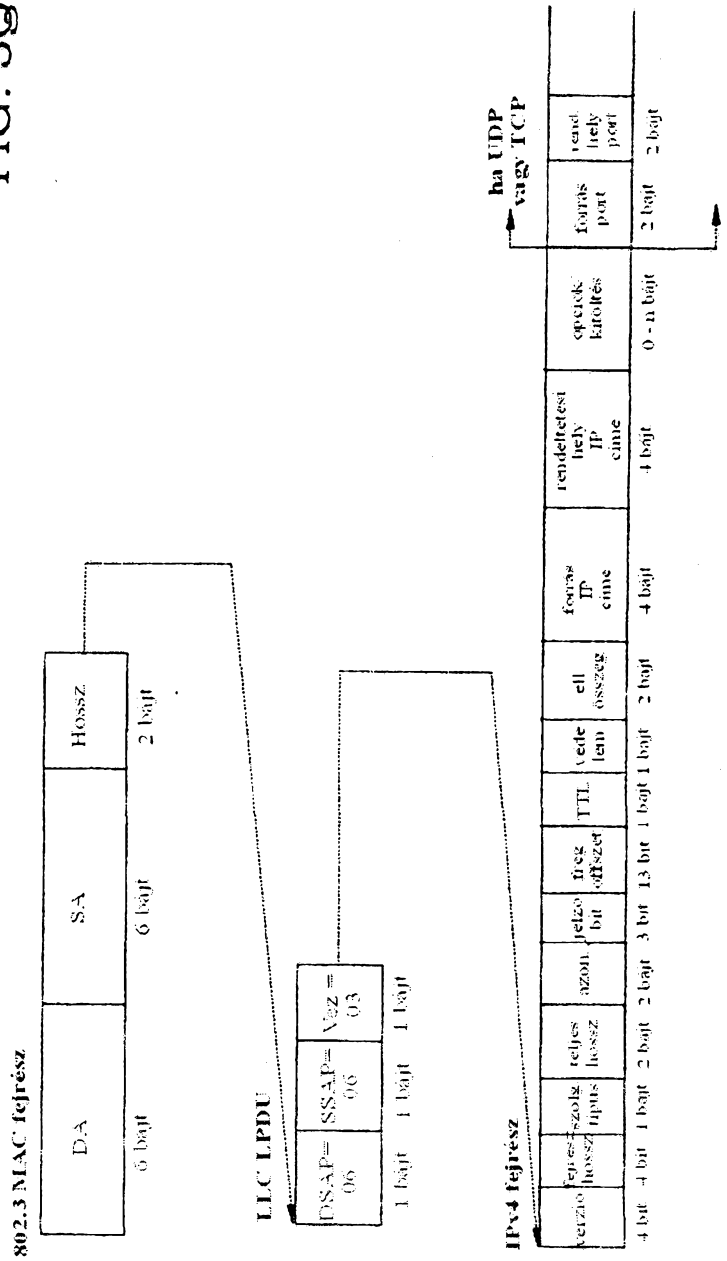
802.3 MAC fejrész
DA
SA
Hossz
6 bajt
6 bajt
2 bajt
LLC LPDU
DSAP= 06
SSAP= 06
Vez = 03
1 bajt
1 bajt
1 bajt
IPv4 fejrész
verzió
fejrész hossz
szolg típus
teljes hossz
azon
jelző bit
frag offszet
TTL
vezérlen
ell összeg
forrás IP címe
rendeltetési hely IP címe
opciók kitöltés
ha UDP vagy TCP
forrás port
rend. hely port
4 bit
4 bit
1 bajt
2 bajt
2 bajt
3 bit
13 bit
1 bajt
1 bajt
2 bajt
4 bajt
4 bajt
0 - n bajt
2 bajt
2 bajt

FIG. 3R

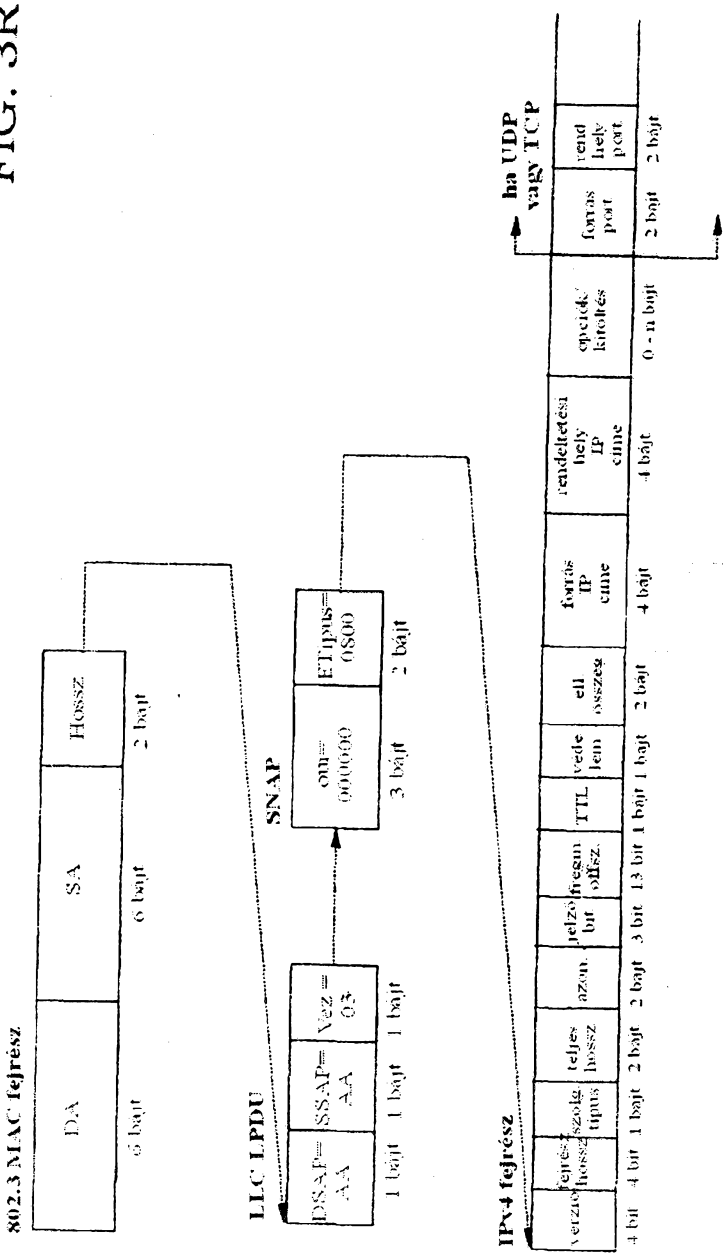
802.3 MAC fejrész
DA
SA
Hossz
6 bajt
6 bajt
2 bajt
LLC LPDU
DSAP= AA
SSAP= AA
Vez = 03
1 bajt
1 bajt
1 bajt
SNAP
om= 000000
ETípus= 0800
3 bajt
2 bajt
IPv4 fejrész
verzió
fejrész hossza
szolg. típus
teljes hossz
azon.
jelző bit
feloszlás
TTL
védelem
ell összeg
forrás IP címe
rendeltetési hely IP címe
opciók/ kitöltés
ha UDP vagy TCP
forrás port
rend. hely port
4 bit
4 bit
1 bajt
2 bajt
2 bajt
3 bit
13 bit
1 bajt
1 bajt
2 bajt
4 bajt
4 bajt
0 - n bajt
2 bajt
2 bajt

FIG. 3S

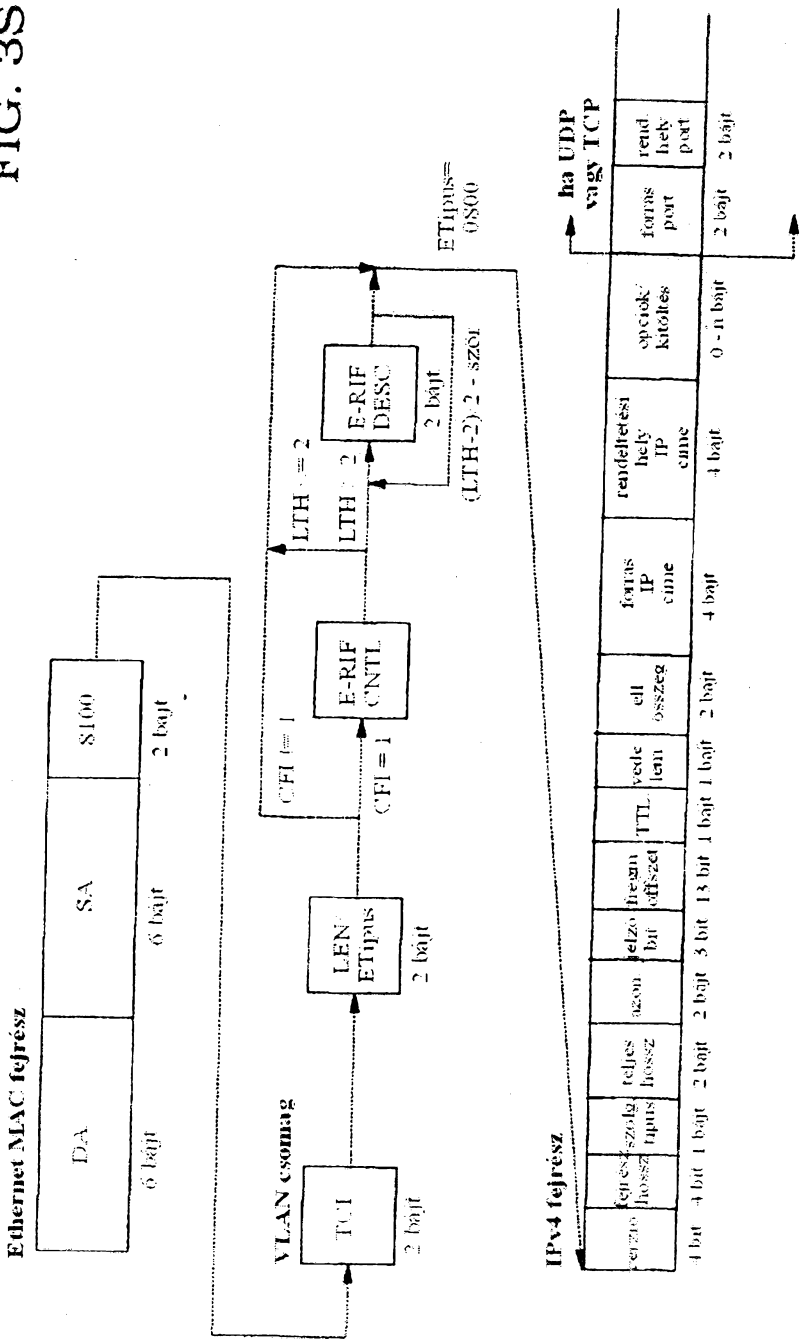
Ethernet MAC fejrész
DA
SA
S100
6 bajt
6 bajt
2 bajt
VLAN csomag
TCI
2 bajt
LEN/ETipus
2 bajt
CFI = 1
E-RIF CNTL
CFI = 1
LTH = 2
LTH = 2
E-RIF DESC
2 bajt
(LTH-2) 2 - szer
ETipus= 0800
IPv4 fejrész
4 bit
4 bit
1 bajt
2 bajt
2 bajt
3 bit
13 bit
1 bajt
1 bajt
2 bajt
4 bajt
4 bajt
0 - n bajt
2 bajt
2 bajt
ha UDP vagy TCP
forras port
rend. hely port

FIG. 3T

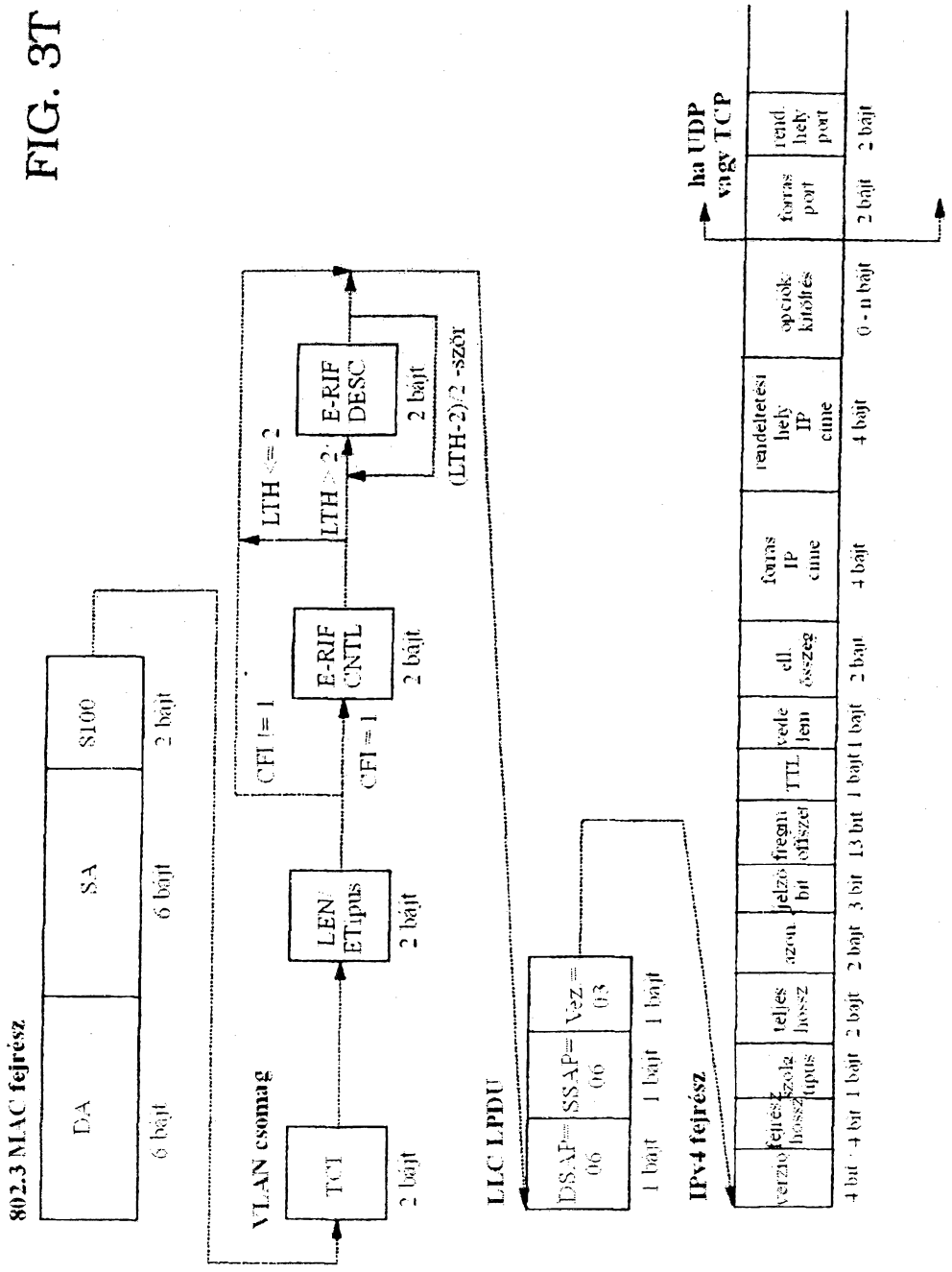
802.3 MAC fejrész
DA
SA
S100
6 bájt
6 bájt
2 bájt
VLAN csomag
TCI
LEN/ ETípus
E-RIF CNTL
E-RIF DESC
2 bájt
2 bájt
2 bájt
CFI != 1
CFI = 1
LTH <= 2
LTH > 2
(LTH-2)/2 -szer
LLC LPDU
DSAP= 06
SSAP= 06
Vez = 03
1 bájt
1 bájt
1 bájt
IPv4 fejrész
ha UDP vagy TCP
verzió
fejlész hossz
szolg típus
teljes hossz
azon
jelző bit
fragm eltolás
TTL
véde lem
ell. összeg
forrás IP címe
rendeltetési hely IP címe
opciók kitöltés
forrás port
rend. hely port
4 bit
4 bit
1 bájt
2 bájt
2 bájt
3 bit
13 bit
1 bájt
1 bájt
2 bájt
4 bájt
4 bájt
0 - n bájt
2 bájt
2 bájt

FIG. 4

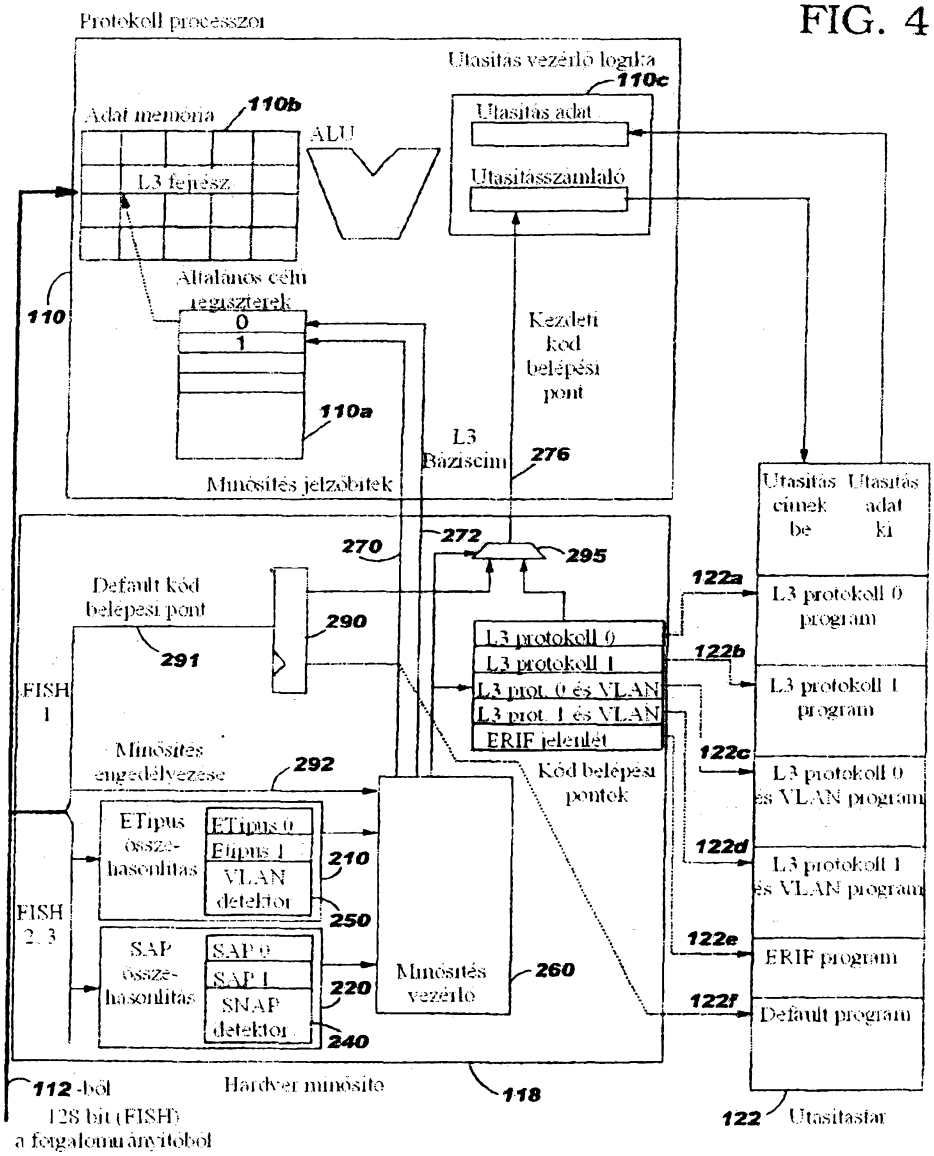
Protokoll processzor
Utasítás vezérlő logika
110c
Adat memória
110b
ALU
Utasítás adat
L3 fejrész
Utasításszámláló
110
Általános célú regiszterek
0
1
110a
Kezdeti kód belépési pont
L3 Bázischim
276
Minősítés jelzőbitek
270
272
295
Utasítás címek be
Utasítás adat ki
Default kód belépési pont
290
291
122a
L3 protokoll 0 program
L3 protokoll 0
L3 protokoll 1
122b
L3 protokoll 1 program
FISH 1
L3 prot. 0 és VLAN
L3 prot. 1 és VLAN
ERIF jelenlét
122c
Minősítés engedélyezése
292
Kód belépési pontok
L3 protokoll 0 és VLAN program
122d
ETípus összehasonlítás
ETípus 0
ETípus 1
VLAN detektor
210
250
L3 protokoll 1 és VLAN program
FISH 2..3
SAP összehasonlítás
SAP 0
SAP 1
SNAP detektor
220
240
Minősítés vezérlő
260
122e
ERIF program
122f
Default program
112-ből 128 bit (FISH) a fogalomnyílvitából
Hardver minősítő
118
122
Utasításlar

FIG. 5

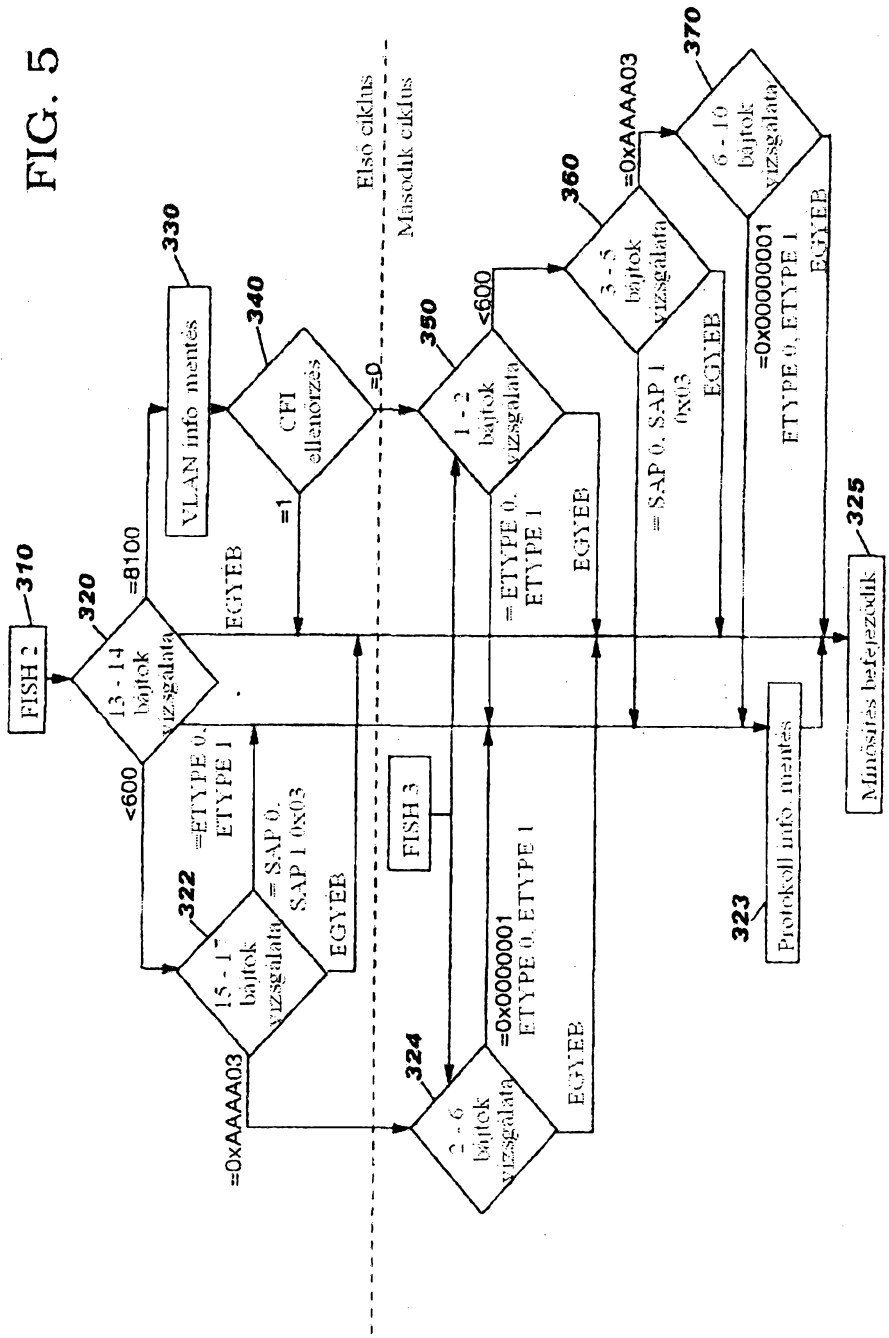
FISH 2
310
320
13 - 14 bajtok vizsgálata
=8100
<600
EGYÉB
330
VLAN infó mentés
340
CFI ellenőrzés
=1
=0
Első ciklus
Második ciklus
350
1 - 2 bajtok vizsgálata
<600
= ETYPE 0, ETYPE 1
EGYÉB
360
3 - 5 bajtok vizsgálata
=0xAAAA03
= SAP 0, SAP 1 0x03
EGYÉB
370
6 - 10 bajtok vizsgálata
=0x00000001 ETYPE 0, ETYPE 1
EGYÉB
325
Minősítés befejeződik
322
15 - 17 bajtok vizsgálata
=ETYPE 0, ETYPE 1
= SAP 0, SAP 1 0x03
EGYÉB
FISH 3
323
Protokoll infó mentés
=0xAAAA03
324
2 - 6 bajtok vizsgálata
=0x0000001 ETYPE 0, ETYPE 1
EGYÉB

FIG. 6

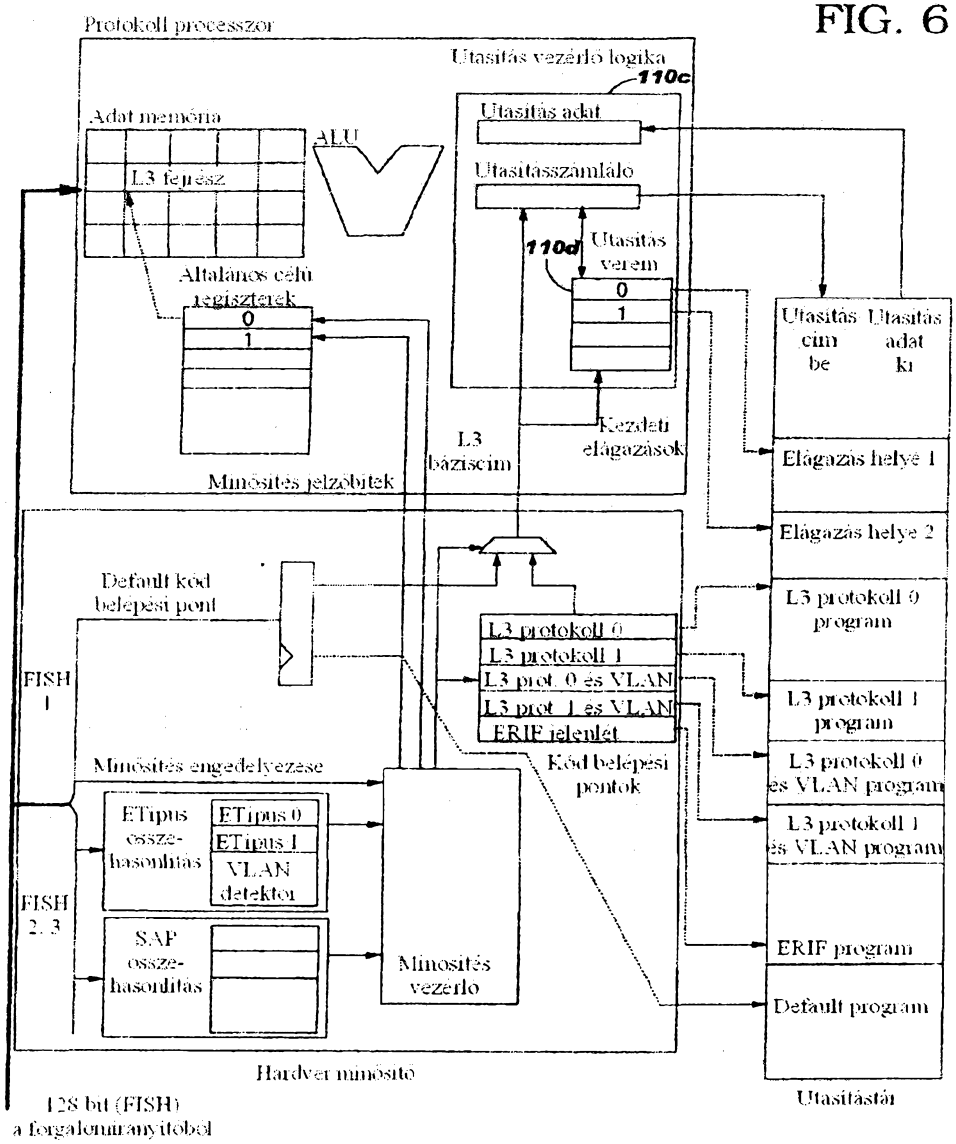
Protokoll processzor
Utasítás vezérlő logika
110c
Adat memória
ALU
Utasítás adat
Utasításszámláló
110d
Utasítás verem
L3 fejrész
Általános célú regiszterek
0
1
Utasítás cím be
Utasítás adat ki
L3 bázíscím
Kezdeti elágazások
Minősítés jelzőbyték
Elágazás helye 1
Elágazás helye 2
Default kód belépési pont
L3 protokoll 0
L3 protokoll 1
L3 prot. 0 és VLAN
L3 prot. 1 és VLAN
ERIF jelenlét
L3 protokoll 0 program
L3 protokoll 1 program
L3 protokoll 0 és VLAN program
L3 protokoll 1 és VLAN program
FISH 1
Minősítés engedélyezése
Kód belépési pontok
ETípus összehasonlítás
ETípus 0
ETípus 1
VLAN detektor
FISH 2..3
SAP összehasonlítás
Minősítés vezérlő
ERIF program
Default program
Hardver minősítő
128 bit (FISH) a forgalomirányítóból
Utasításai

FIG. 7

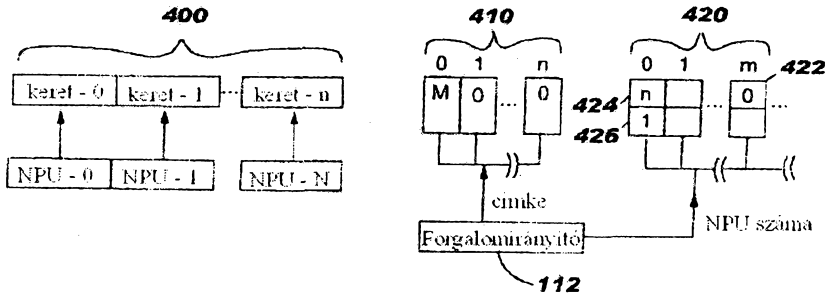
400
keret - 0
keret - 1
keret - n
NPU - 0
NPU - 1
NPU - N
410
0 1 n
M 0 ... 0
420
0 1 m
422
424
426
n
1
0 ...
címke
Forgalomirányító
NPU száma
112

FIG. 9

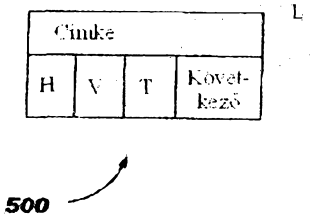
Címke
H
V
T
Követ-kező
L
500

FIG. 8

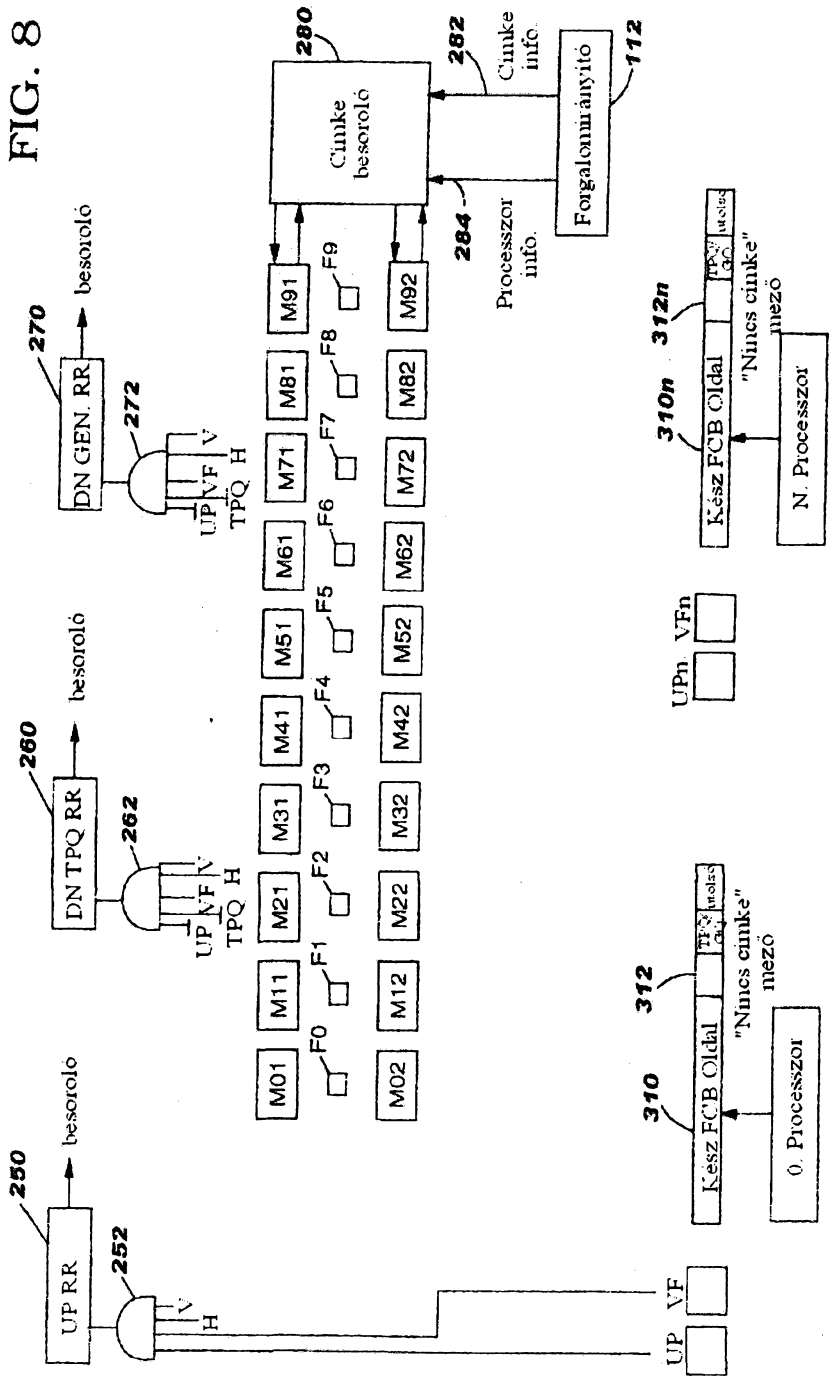
250
besoroló
UP RR
252
V
H
UP VF
260
besoroló
DN TPQ RR
262
UP VF V
TPQ H
270
besoroló
DN GEN. RR
272
UP VF V
TPQ H
M01
M11
M21
M31
M41
M51
M61
M71
M81
M91
M02
M12
M22
M32
M42
M52
M62
M72
M82
M92
F0
F1
F2
F3
F4
F5
F6
F7
F8
F9
280
Cimke besoroló
282
Cimke info.
284
Processzor info.
112
Forgalomirányító
UPn VFn
310
Kész FCB Oldal
"Nincs cimke" mező
0. Processzor
312
310n
Kész FCB Oldal
"Nincs cimke" mező
N. Processzor
312n

FIG. 10

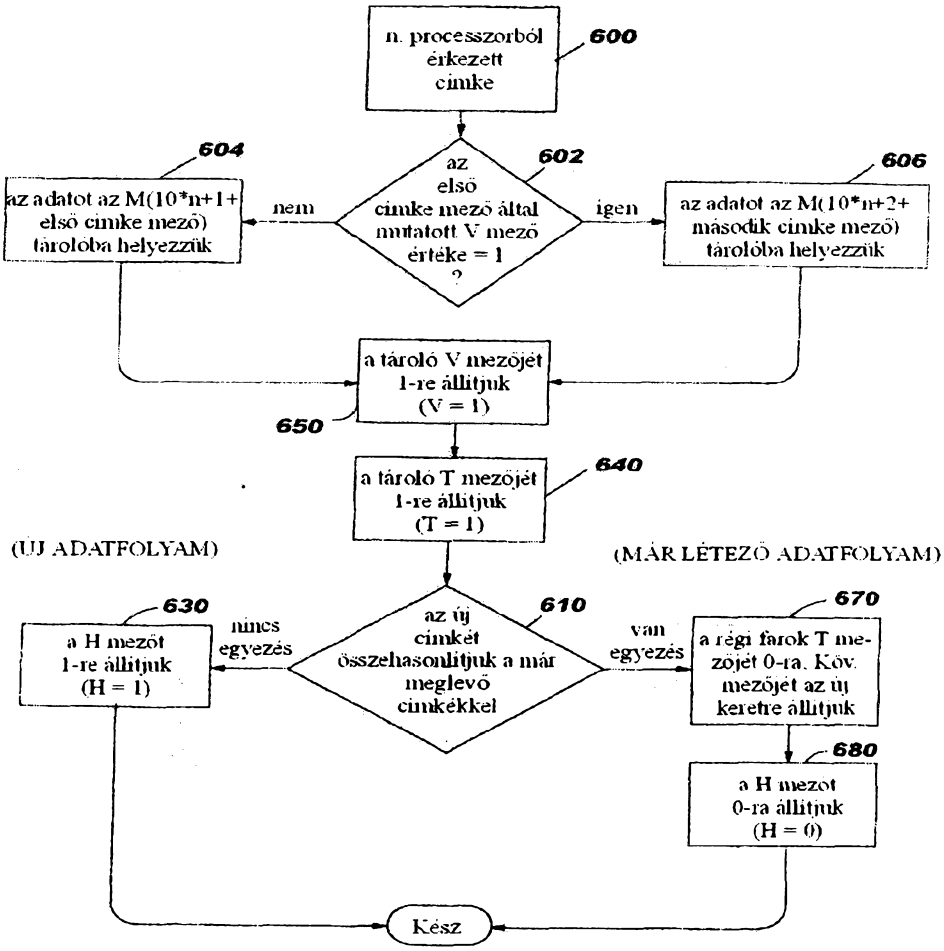

FIG. 11

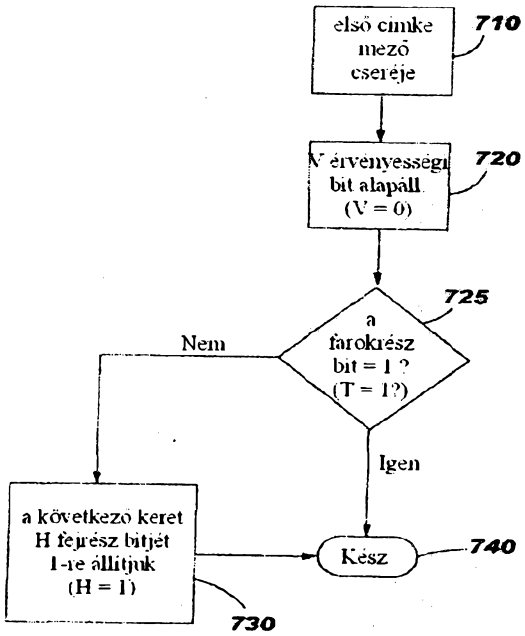
első címke mező cseréje
710
V érvényességi bit alapáll. (V = 0)
720
a farokrész bit = 1 ? (T = 1?)
725
Nem
Igen
a következő keret H fejrész bitjét 1-re állítjuk (H = 1)
730
Kész
740

FIG. 12