UNIVERSITETI I EJL

ЈИЕ УНИВЕРЗИТЕТ

SEE UNIVERSITY

FAKULTETI I SHKENCAVE DHE TEKNOLOGJIVE BASHKËKOHORE

ФАКУЛТЕТ ЗА СОВРЕМЕНИ НАУКИ И ТЕХНОЛОГИИ

FACULTY OF CONTEMPORARY SCIENCES AND TECHNOLOGIES

STUDIME PASDIPLOMIKE – CIKLI I DYTË

TEZA:

"Integrimi i LonTalk protokollit në IP me anë të middleware komponentës"

KANDIDATI:

Ardit Saliu

MENTORI:

Prof. dr.Azir Aliu

Tetovë 2017

Abstrakti

Ky punim jep një përshkrim të plotë të rrjetave e veçanërisht analizim të protokollove të ndryshme të kontrollit. Meqenë se këtu jepet bërthama softwerike e cila mund të shfrytëzohet e gatshme ose të zgjerohet, mund të përdoret drejtëpërdrejtë si një urë ndërmjet një kërkese të nivlit të lartë të që kërkon ndërlidhje ose vetëm qasje në një rrjet kontrolli i bazuar në LonTalk. Zgjidhja e dhënë mundëson përdorim të menjëhershëm në anën e rrjetit të kontrollit, pra e gatshme të komunikoj me paisje tjera LonWorks, ndërsa serveri mundëson që në anën e klientëve të ndërtohet interfejs sipas kërkesës dhe të lidhet pa problem me internetin.

Teoria e rrjetit është studimi i grafikëve si një përfaqësim i marrëdhënieve simetrike ose marrëdhënieve asimetrike ndërmjet objekteve të veçanta. Në shkencat kompjuterike dhe shkencën e rrjetit, teoria e rrjetit është pjesë e teorisë së grafikut ku një rrjet mund të definohet si një grafik në të cilin nyjet dhe skajet e të cilit kanë atribute.

Një rrjet kompjuterik ose rrjet i të dhënave, është një rrjet dixhital i telekomunikimeve që lejon nyjet e ndryshme të ndajnë burimet. Në rrjetet kompjuterike, pajisjet kompjuterike të rrjetit shkëmbejnë të dhëna me njëri tjetrin duke përdorur një lidhje të dhënash. Lidhjet midis nyjeve janë krijuar duke përdorur mediat kabllore (metale ose fibra optike) ose media pa tel.

Pajisjet kompjuterike të rrjetit që dërgojnë, rrugëtojnë apo pranojnë të dhënat quhen nyjet e rrjetit. Nyjet mund të përfshijnë hostat si kompjuterët personal, telefonat, serverat si dhe pajisje të tjera me aftësi të caktuar.

Rrjetet kompjuterike mbështesin një numër të madh aplikacionesh dhe shërbimesh si: qasja në World Wide Web, video dixhitale, audio dixhitale, përdorimi i përbashkët i serverave të aplikimit dhe ruajtjes, printera dhe makina faksi dhe përdorimi i aplikacioneve të postës elektronike dhe mesazheve të menjëhershme si dhe shumë të tjerë. Rrjetet kompjuterike ndryshojnë në mesazhin transmetues të përdorur për të kryer sinjalet e tyre, protokollet e komunikimit për të organizuar trafikun e rrjetit, madhësinë e rrjetit, topologjinë dhe qëllimin organizativ. Rrjeti kompjuterik më i njohur që është i shtrirë në gjithë globin njihet si Interneti.

Fjalët kyçe: rrjet, internet, TCP/IP, LonWork, LonTalk, Iot dhe Ilot.

Deklaratë origjinaliteti

Deklaroj që kjo tezë bazohet nga rezultatet e hulumtimit dhe studimeve të gjetura nga vet unë. Referencat nga studimet hulumtuese dhe raportet nga studiues të tjerë janë të përmendura në mënyrë të duhur. Kjo tezë, as tërësisht e as pjesërisht nuk është dorëzuar më parë për ndonjë gradë shkencore.

Ardit Saliu

Mirënjohje

Këtë punim ua dedikoj familjes time të dashur për mbështetjen e tyre gjatë gjithë viteve, për mësuesit dhe pedagogët e mi të shkëlqyer që më udhëzuan gjatë studimeve, si dhe dëshiroj të falënderoj lexuesin e këtijë punimi.

Përmbajtja

Tabela e figurave

Përmbajta e Tabelave

1. Hyrje

Teoria e rrjetit është studimi i grafikëve si një përfaqësim i marrëdhënieve simetrike ose marrëdhënieve asimetrike ndërmjet objekteve të veçanta. Në shkencat kompjuterike dhe shkencën e rrjetit, teoria e rrjetit është pjesë e teorisë së grafikut ku një rrjet mund të definohet si një grafik në të cilin nyjet dhe skajet e të cilit kanë atribute.

Një rrjet kompjuterik ose rrjet i të dhënave, është një rrjet dixhital i telekomunikimeve që lejon nyjet e ndryshme të ndajnë burimet. Në rrjetet kompjuterike, pajisjet kompjuterike të rrjetit shkëmbejnë të dhëna me njëri tjetrin duke përdorur një lidhje të dhënash. Lidhjet midis nyjeve janë krijuar duke përdorur mediat kabllore ose media pa tel.

Pajisjet kompjuterike të rrjetit që dërgojnë, rrugëtojnë apo pranojnë të dhënat quhen nyjet e rrjetit. Nyjet mund të përfshijnë hostat si kompjuterët personal, telefonat, serverat si dhe pajisje të tjera me aftësi të caktuar. Dy pajisje të tilla mund të thuhet se janë të lidhura së bashku kur një pajisje është në gjendje të shkëmbejë informacion me pajisjen tjetër, pavarësisht nëse kanë lidhje direkte me njëri-tjetrin. Në shumicën e rasteve, protokollet e komunikimit specifik për aplikacionet shtresohen (d.m.th. mbarten si ngarkesë) mbi protokollet e tjera më të përgjithshme të komunikimit. Kjo mbledhje e mrekullueshme e teknologjisë së informacionit kërkon menaxhimin e rrjetit të aftë për të mbajtur të gjitha duke ecur në mënyrë të besueshme.

Rrjetet kompjuterike mbështesin një numër të madh aplikacionesh dhe shërbimesh si qasja në World Wide Web, video dixhitale, audio dixhitale, përdorimi i përbashkët i serverave të aplikimit dhe ruajtjes, printera dhe makina faksi dhe përdorimi i aplikacioneve të postës elektronike dhe mesazheve të menjëhershme si dhe shumë të tjerë. Rrjetet kompjuterike ndryshojnë në mesazhin transmetues të përdorur për të kryer sinjalet e tyre, protokollet e komunikimit për të organizuar trafikun e rrjetit, madhësinë e rrjetit, topologjinë dhe qëllimin organizativ. Rrjeti kompjuterik më i njohur që është i shtrirë në gjithë globin njihet si Interneti.

Në kapitullin e dytë jepet sqarim i detajuar i rrejteve të kontrollit dhe ndërlidhja e tyre me rrjetet kompjutertike.

1.1 Përshkrimi i problemit

Qasja në paisje që kryejnë procese industriale apo monitorojnë ato është shumë e kufizuar. Kryesisht mjediset e tilla janë të menxhuara nga vet integruesit apo nga prodhuesit e paisjeve, ky kufizim është barier e vështirë për çdo zhivillues softweri, sepse paisjet e tilla kryesisht janë në përdorim dhe procese të caktuara qoftë edhe ato industriale janë të mvarura drejtëpërdrejtë nga paisjet e kontrollit. Nga kjo lind edhe kërkesa për qasje në rrjetet e kontrollit, por nga pamundësia për qasje direkte ose nga mungesa e hardwerit ky punim mund të përdoret për të tejkaluar këtë pengesë kryesisht teknkike. Në thellësi të këtijë punimi trajtohen në mënyrë sipërfaqësore edhe protokollet tjera që përdoren në rrjetat për kontrollë. Gjithsesi objekt hulutimi i këtij punimi është LonTalk-ut i cili ka kufizim përdorimi tek ne, ky punim është shumë i përshtatshëm për integrim ose ndërlidhje të rrjetave të kontrollit me rrjetat që neve sot na rrethojnë në jetën e përditshme si Interneti. Meqenëse ne rrethohemi nga paisje inteligjente të cilat në vete kanë të integruar një modul kontrolli, por nga pa mundësia për t'iu qasë atyre krijohet edhe një barierë për zhvilluesit në këtë fushë.

1.2 Metodologjia

Metodologjia e hulumtimit që do të apklikohet në këtë punim përfshinë metodologjinë kuantitative dhe kualitative.

Kualitative – analizimi i Controll Networks në building, factory and home automation.

Kuantitative – thjeshtësia dhe zgjerueshmëria e LonTalk protokollit në IP me anë të middleware komponentes si opsion më i lirë dhe më adekuat për Home Automation.

1.3 Hipotezat

1. Topologjia dhe linqet aktuale të TCP/IP-së mundësojnë dërgimin e mesazheve për kontroll në rrejt te bazurar në sistemet e hapura.

2. Krijimi i middleware-it (ndërjmetesuesi) do të mundesoj ndërlidhjen e Controll Network-ut me TCP/IP.

3. IP është protokoll më i përshtatshëm për bartjen e LonTalk mesazheve nga një Controll Network në Internet.

1.4 Kontributi

Ky punim jep një përshkrim të plotë të rrjetave e vaçanërisht analizim të protokollove të ndryshme të kontrollit. Me qenë se këtu jepet bërthama softwerike e cila mund të shfrytëzohet e gatshme ose edhe të zgjerohet, mund të përdoret drejtëpërdrejtë si një urë ndërmjet një kërkese të nivlit të lartë të që kërkon ndërlidhje ose vetëm qasje në një rrjet kontrolli i bazuar në LonTalk. Zgjidhja e dhënë mundëson përdorim të menjëhershëm në anën e rrjetit të kontrollit, pra e gatshme të komunikoj me paisje tjera LonWorks, ndërsa serveri mundëson që në anën e klientëve të ndërtohet interfejs sipas kërkesës dhe të lidhet pa problem me internetin.

1.5 Struktura e punimit

Në kapitullin e dytë jepet përshkrim i detajuar i rrjeteve, rrejteve për kontrollë, protokollet e rrjetit, topologjitë ekzistuese që mund të përdoren për një rrjet apo rrjet kontrolli. Po ashtu faktor kyç i një rrjeti janë edhe mediumet që mund të përdoren për ndërlidhje të paisjeve për të krijuar një rrjet. Theks të veçantë në ketë kapitull ka edhe kominikimi RS232/RS485, ku analizohen mënyra e fuksionimit për shkak se prej këtu lind edhe ideja për komunimin të standardizuar të paisjeve të ndryshme si dhe komunikim në distanacë të ndryshme, për shkak se barierat e mediumit nuk lejojnë distancë të madhe si dhe te zhurma nga rrethet ekstreme industriale lind nevoja për komunikim më kompleks i cili do të jetë në gjendje të ofertojë besueshmëri dhe qasshmëri të pranueshëme.

Në kapitullin e tretë përshkruhen katër standardet e rrjeteve të kontrollit duke u futur në detaje teknike edhe atë për ARC.net, BACnet, EIB.TB si dhe LonWorks i cili më vonë është objekt i hulumtimit më të detajuar.

Në kapitullin e katërt jepet përshkrim sipërfaqësor i standardit kryesor të LonTalk edhe atë të ISO/IEC 14908-4, i cili definon kornizat e ndërtimit të një sistemi kontrolli të bazuar në IP. Po ashtu në ketë kapitull bëhet edhe analizimi i produkteve aktuale të cilat janë kryesisht të dizajnuara dhe

prodhuara nga prodhues të cilët janë prezent drejtpërdrejtë në ketë fushë. Në përbërjen e këtij kapitulli janë prezent edhe “network interface” të cilët shfrytëzohen aktualisht si dhe analizim i detajuar i routerave që përdoren për ndërlidhje të rrjeteve të njëjta apo të ndryshme të kontrollit.

Në kapitullin e pestë jepet ideja bazë e këtijë punimi. Si rezulatat i kostos së lartë të paisejve për ndërlidhje të rrjetave të kontrollit lind edhe nevoja për zgjidhje teknike me kosto më të ulët e cila mundëson përdorimin e network interfacave të bashkëngjitur në një kompjuter dhe të bëhet kominikimi me një rrjet kontrolli nëpërmjet një IP rrjeti ku sot të gjitha pasijet janë direkt apo indiret të lidhuara ose me qasje në rrjetin global – Internet.

Kapitulli i gjashtë përmban temën aktuale të paisjeve të mençura dhe të lidhura në Internet. Në të bëhet përshkrim çka nënkupton një IoT dhe se si një IoT paisje funksionon apo lidhet. Po ashtu këtu përmendet edhte trendi global i lidhjeve të paisjeve industriale në Internet.

Në kapitullin e shtatë jepet përshkrim i detajuar i zgjedhjes softwerike të problematikës në fjalë. Këtu ndodhet përkrishimi dhe pamje nga aplikacioni i cili mundëson komunikim me një rrjet kontrolli nëpërmejt paisjeve të ndryshmë që kanë qasje deri te serveri i krijuar veçanërisht për këtë proces.

Në kapitullin e tetë bëhet përmbyllja e kësaj problematike të analizuar, po ashtu edhe konkluzionete që lindën dhe u analizuan gjatë programimit të këtij middleware i cili mundëson përdorim të mëtutjeshëm me funksionalitetet e kufizuara në anën e rrjetit të kontrollit.

Në kapitullin e nëntë jepet kodi i kësaj zgjidhje teknike për të dy anët i shkruar në C# dhe munëdisa e tijë që në të ardhmen ky kod të optimizohet dhe të zgjerohet me funksionalitete më komplekse.

2. Rrjetat për konroll

Një rrjet kontrolli është një rrjet nyjesh që së bashku monitorojnë, dallojnë dhe kontrollojnë ose lejojnë kontrollin e një procesi ose të një mjedisi. Një rrjet pajisjesh shtëpiake është një shembull i mirë i një rrjeti kontrollesh. Në fakt, ndërkohë mijëra kontrolle rrjetesh ekzistojnë në jetën e përditëshme tek automjetet, frigoriferët, tek semaforët, sistemi i ndriçimit të qytetit dhe në një kat fabrikë. Rrjetet e kontrolleve ndryshojnë jashtëzakonisht shumë në numrin e nyjeve (nga tre deri ne mijëra) në një rrjet edhe në përbërjen e tyre.

Rjetet, ndryshe nga njerëzit që komunikojnë me njeri-tjetrin, kanë tendencën të jenë të padukshëm. Në të ardhmen, rrjetet e kontrollit pritet të bëhen një aspekt i rëndesishëm i asaj që ndonjëherë quhet përdorimi i kudondodhur i kompiuterit.

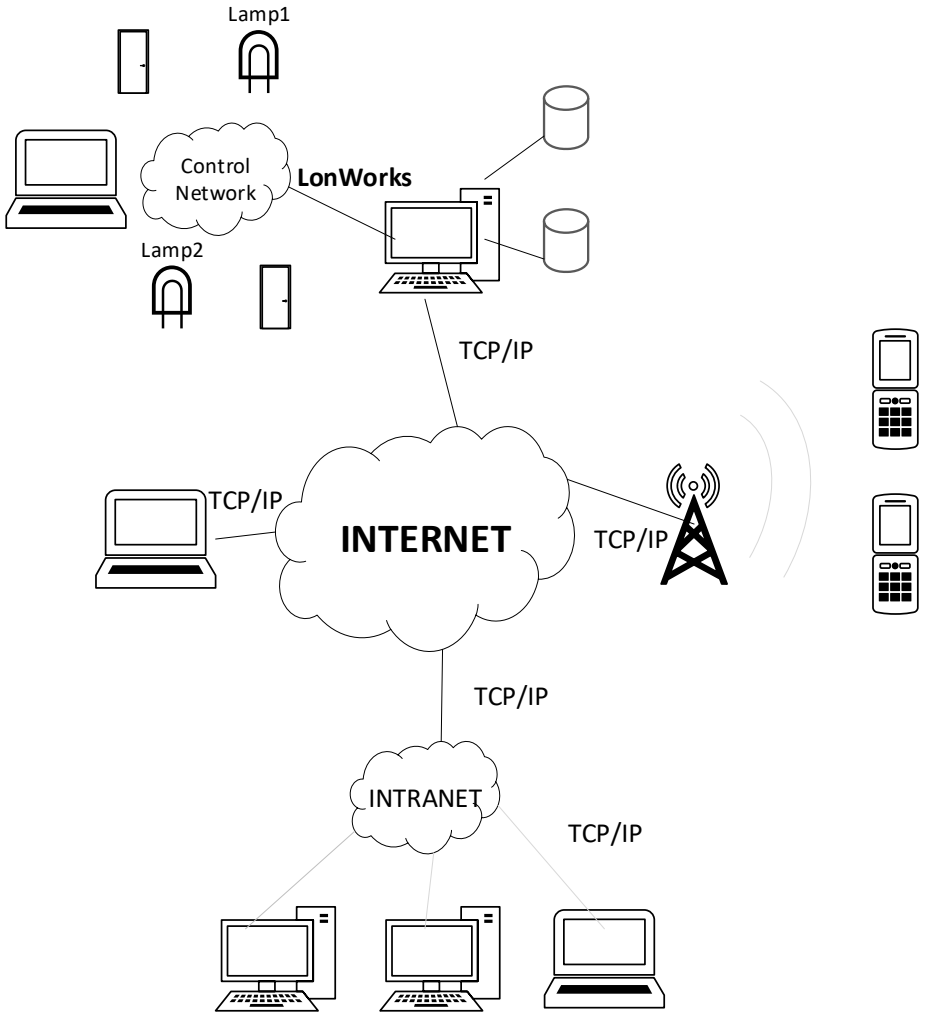

Figura 1 - Rrjete të ndryshme (rrjet kontrolli, Internet dhe Intranet)

Rrjeti i kontrollit është duke përhapur konceptin e shkëmbimit të të dhënave në një botë

kontrolli. Rjetet e kontrollit përhapin në mënyrë të dukshme mundësinë e shkëmbimit të të dhënave. Kur kombinohen me rrjete të dhënash, rrjetet e kontrollit sigurojnë informacion të menjëhershëm edhe jetësor në ndërmarrje, në vazhdimësi me mjetet për të vepruar menjëherë në të. Rrjetet e kontrollit ndërlidhin pajisjet duke zëndësuar mbikqyrësit qëndror edhe duke instaluar frerët e së djeshmes.

Komunikimi midis nyjeve në një rrjet kontrolli mund të jetë "peer-to-peer" ose zotërues-skllav. Nyjet në disa rrjete kontrolli përmbajnë tre procesore në një: dy nga këto kushtuar lëvizjes së të dhënave brenda rrjetit dhe tjetri për programin e specializuar të lidhur me atë nyje. Ky modalitet e bën atë jo të pa kushtueshme edhe të shpejtë për të ndërtuar procesore të rinj për rrjetin e kontrollit. Gjithmonë në rritje, rrjetet e kontrollit janë duke u bërë nga Hardware lehtësisht të gjindshëm dhe pjesë përbërëse të software -it.

Ekziston një standard për dizajnimin e rrjeteve të kontrollit të përgjithshem ose e thirrur si TC247/WG4 e cila ndanë në tre nivele edhe atë nivelin e menaxhimit (komunikimi nga një njësi pune tek një njësi tjetër pune), niveli i automatizimit (nga kontrollues fabrik në kontrollues fabrikash - njësi pune) dhe niveli i fushës (nga kontrollues i njësisë në veprues / sensor pajisjesh). Këto nivele karakterizohen edhe ndryshojnë nga mbeshtetje sasi/tipe të ndryshme pajisjesh, mbështetje grumbuj të ndryshëm informacioni që duhet dërguar, kërkesë për performanca të ndryshme komunikimi si dhe mbështetje të llojeve të ndryshme mediumesh transportues.

Sasi e madhe protokollesh të ndryshme ekzistojnë në botën e rrjeteve të kontrollit, inxhinierët e kontrollit kanë lirinë të zgjedhin cili protokoll duhet përdorur në secilin nivel. Si rezultat i mospërputhjes ato mund te përdorin edhe gateway. Disa protokolle mund të përdoren në menyrën e duhur dhe efektive në të tre nivele, ndërsa disa janë dizajnuar për të punuar vetëm në një nivel specifik.

Pajisjet inteligjente që përdorin protokolle të përbashkëta (nyjet) - Protokolli i komunikimit përcakton formatin e mesazhit që do të transmetohet midis pajisjeve dhe përcakton veprimet e pritura kur njëpajisje i dërgon një mesazh tek një pajisje tjetër.

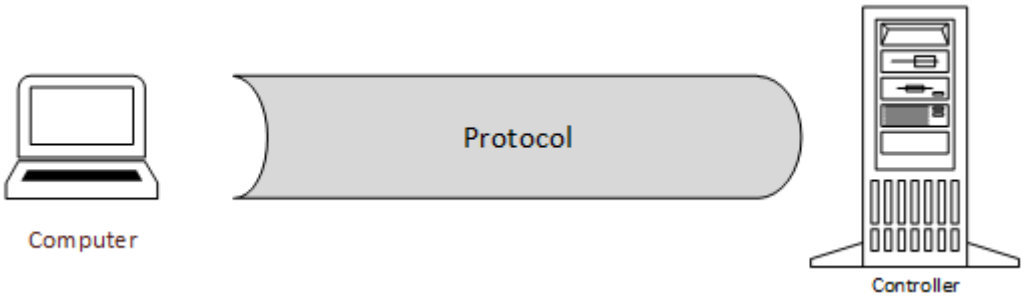

Figura 2 - Një kompjuter komunikon nëpërmjet një protokolli me një kontroller

Protokolli normalisht merr formën e një software të dedikuar për pajisje të veçant ose kodi i software-it i cili përmban instruksione se si të operohet me çdo pajisje në rrjet [1].

1.6 Protokollet e rrjetit

Protokolli është një strukturë rregullash të cilat lejojnë kompjuterët, kontrolluesit dhe pajisjet të komunikojnë nga njëra tek tjetra. Elementet kyçe të protokollit përcaktojnë formatin e të dhënave, informacionin e nevojshëm për konvertimin e të dhënave midis makinerive si dhe kohën për të përcaktuar shpejtësinë bashkë me rendin e transmetimit të të dhënave. Ekzistojnë tre grupe protokolli edhe atë: Pronësore (p.sh., N1 dhe N2), të hapura (p.sh., LonWorks, BACnet dhe EIB) dhe Standarte (p.sh. OSI).

Protokollet pronësore - Janë të zhvilluara nga prodhues sistemesh ose kompjuterësh për të komunikuar me hardware-in dhe sotware-in e tyre mbi një rrjet të rekomanduar. Edhe atë specifikisht të dizajnuara dhe tërësisht të testuara për sistem specifik tregtar dhe për rrjete specifike. Natyra pronare e sistemit siguron besueshmërinë dhe paprekshmërinë e tij.

Protokollet e hapura - Të zhbllokosh protokolle nënkupton të zbulosh procedura, struktura dhe kode duke lejuar zhvilluesit e sistemeve të tjera për të shkruar interfaces dhe të shpëndajnë të dhëna në rrjetin e tyre. Pranimi i një protokolli tv hapur varet nga cilësia e tij, tiparet dhe shërbimet provided. Nëse një protokoll i hapur përdoret gjërësisht nga dizajnuesit e sistemeve dhe integruesit e sistemeve, ajo do të bëhet e ashtuquajtur ‘de facto’ ose protokoll standard i industries.

Protokollet standarde 'de facto'

Janë të gjithë përdorura dhe kryesisht njihen me këto përparësi si performancë premtuese, të testuar me sukses në shumë aplikacione, e rafinuar nga shume përmirësime gjatë kohës si dhe shumë tregtues mund të "bashkëjetojnë" thjesht në një rrjet të vetëm. Gjithashtu ekzistojnë disa mangësi si p.sh. nevojë për përzgjedhje të kujdesshme nga shumë tregtues, veçanërisht nga mungesa e familjaritetit të klientit me atë që ofrohet. Është shumë e vështirë të unifikosh ose përcaktosh një "protokoll standard" për të gjithë aplikacionet. Organizata Ndërkombëtare e Standardeve (ISO) ka publikuar një model për ndërlidhje të sistemeve të hapura (OSI) [2].

2.1.1 OSI Modeli

Çdo shtresë ka nje bashkësi të përcaktuar funksionesh. Modeli siguron një referencë të përgjithshme të dobishme të protokollit të komunikimit. Shumica e protokolleve të komunikimit përfshirë edhe ato që përdoren në fushën tonë sot, përdorin ose të gjitha ose disa nga shtatë shtresat e modelit OSI [3].

OSI Model			
Layer		Protocol data unit (PDU)	Function
Host layers	7. Application	Data	High-level APIs, including resource sharing, remote file access
	6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4. Transport	Segment (TCP) /Datagram (UDP)	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1. Physical	Bit	Transmission and reception of raw bit streams over a physical medium

Tabela 1 - OSI modeli

Konceptet bazë:

1. Aplikacionet e afta për rrjet prodhojnë të dhëna

2. Çdo shtresë protokolli i shton një prijës të dhënave që merr nga shtresa poshtë tij. Kjo quhet enkapsulim. Të dhënat e enkapsuluara transmetohen në Njesinë e të Dhënave të Protokollit (PDUs). Ka PDU prezantimi, PDU seancë, PDU transporti, etj.
3. PDU-të kalojnë përmes grumbullit të shtresave derisa ato të mund të transmetohen mbi një shtresë fizike.
4. Çdo shtresë në një makinë flet të njëjtën gjuhë si shtresat e njëjta të një makinerie tjetër, prandaj mund të komunikojnë nëpërmjet shtresës fizike.
5. Të dënat e kaluara së larti janë de-enkapsuluar përpara se të kalohen edhe më lart.
6. Të gjitha informacionet kalojnë poshtë përmes të gjitha shtresave derisa arrin ne shtresën fizike.
7. Shtresa fizike ndyshon drejtimin e PDU-se dhe e transmeton PDU-në në kabllo. Shtresa fizike siguron lidhjen e vërtetë fizike ndërmjet makinerive tek të cilat ndodh i gjithë komunikimi.

Të dhënat nga një shtresë përballohen që të kalohen poshtë në shtresën nën të. Në 'botën reale', procesi i enkapsulimit (shtimi i prijësit) nuk ndodh gjithmonë në të gjitha shtresat e tij.

Shtresa fizike

Specifikon rrugën e komunikimit dhe mediumin fizikë të rrjetit, mediumet e transmetimit aktual (p.sh., TP, coax, fibra, RF, etj.), nivelet e sinjalit dhe kapaciteti drejtues i sinjalit elektrik, karakteristikat e medias (p.sh., gjatësia, shpejtësia, bandwidth).

Standardet e përdorura më gjërësisht në industrine e automatizimit janë standardet IEEE802 dhe RS232/RS485 për komunikimin serial.

Transmetimi dhe marrja e të dhënave nga mediumi fizik menaxhohet në këtë shtresë. Shtresa fizike merr të dhëna nga Data Link Layer dhe i transmeton ato në kabllo. Shtresa fizike kontrollon shpeshtësinë, amplitudën, fazën dhe modulimin e sinjalit të përdorur për transmetimin e të dhënave, performon demodulimin dhe dekodimin menjëherë sapo merr apo pranon. Që dy pajisje të komunikojnë, ato duhet të jenë të lidhur me të njëjtin tip të mediumit fizik (kabllot). Ethernet me ethernet, FDDI me FDDI etj. Dy stacionet përfundimtare që përdorin protokolle të ndryshme munden të komunikojnë vetëm përmes një ure multi-protokoll ose ndërlidhësi – gateway.

Shtresa fizike është përgjegjëse për dy pune: Komunikimi me Data Link Layer dhe Transmetimin dhe marrjen e të dhënave.

Data Link Layer

Kontrollon, rendit, dhe sinkronizon transmetimin e të dhënave (dërgimin edhe marrjen). Dallimi i gabimeve me nivelin më të ulët dhe rikuperimin e gabimeve, koha dhe funksione të tjera të lidhura me kontrollin e medias fizike.

Data Link Layer është shtresa e dytë e modelit OSI. Performon funksione të ndryshme varur nga protokolli i përdorur i hardwerit, por ka 4 funksione parësore:

KOMUNIKIMI me shtresën e rrjetit sipër;

SEGMENTIMI i datagramës të shtresës së sipërme (të quajtura paketa) në korniza me përmasa që mund të trajtohen nga hardware i komunikimit;

RENDITJA E BIT-eve. Organizimi i modeleve të bit-eve të të dhënave përpara transmetimit;

KOMUNIKIMI me shtresën fizike më vonë.

Kjo shtresë siguron kalim të besueshëm të të dhënave nëpër hallkën fizike. Datalink Layer lidhet me adresimin fizik, topologjinë e rrjetit, menaxhimin e hallkës fizike, njoftimet e gabimit, shpërndarjen e urdhëruar të kornizave dhe kontrollin e rrjedhjes.

Po ashtu ekzistojnë nën shtresa të Data Link, Instituti i Inxhinierisë Elektrike dhe Elektronike (IEEE) ka ndarë Data Link Layer në dy nënshtresa: Logical Link Control (LLC) dhe Media Access Control (MAC).

Kjo shtresë gjithashtu vendos cilat metoda të kontrollit të komunikimit (zotërues-skllav, peer-to-peer, hibride) janë përdorur në rrjet, siguron menaxhimin e gabimeve dhe siguron kontrollin e rrjedhjes së të dhënave të komunikimit.

LLC është i zbatuar në pjesën softwerike, kurse MAC zakonisht është zbatuar në hardware. Hardwari MAC siguron menaxhimin e bit-wve ashtu si edhe kodimin, dallimin e gabimeve dhe aftësinë e udhëtimit.

Shtresa e rrjetit

Vendos dhe përfundon lidhjet midis dërguesit dhe marrësit të informacionit në rrjet. Cakton adresat unike të çdo nyje në rrjet (p.sh. kompjuterit). Adresat identifikojnë fillimin dhe fundin e paketës së transmetimit të të dhënave. Të dhënat që dalin kalohen poshtë nga shtresa e transportit, enkapsulohet në protokollin e shtresës së rrjetit dhe më pas dërgohen në Data Link Layer për segmentim dhe transmetim. Të dhënat që hyjnë defragmentohen në një rend korrekt, Headers-at e IP-së hiqen dhe pastaj Datagramet e grumbulluara kalohen në shtresën e transportit.

Shtresa e rrjetit interesohet me funksionet parësore që vijojnë:

Komunikimi me shtresën e transportit më poshtë,

Menaxhimin e lidhjes dhe ndërlidhjen midis hosteve ose rrjeteve,

Komunikimi me shtresën Datalink më poshtë.

Shtresa e transportit

Mban besueshmërinë në rrjet dhe rrit integritetin e të dhënave nëpërmjet shpërndarjes pa gabime të të dhënave në pjesën e duhur. Është përgjegjesia e shtresës së transportit të shikojë dallimin e gabimeve dhe ritransmetimin e të dhënave për të rikuperuar ato gabime ose të dhëna të humbura.

Shtresa e transportit mund të përdor teknika të ndryshme si CYCLING REDUNDANCY CHECK, Windowing dhe njohjen. Nëse të dhënat humben ose dëmtohen është përgjegjësia e shtresës së transportit për ti rikuperuar nga ai gabim.

Funksionet e kësaj shtrese janë:

- Komunikimi me shtresën e sesioneve më poshtë,
- Dallon gabimet dhe te dhënat e humbura, transmeton të dhënat, rimbledh Datagramet në rrjedhje të dhënash,
- Komunikon me shtresën e rrjetit më poshtë.

Session Layer

Vendos, menaxhon dhe përfundon lidhjet e komunikimit, ndaj edhe quhet “session” layer, siguron prioritetin (e lartë, normal), siguron rikuperim të të dënave nga gabime shtesë të software-it, session Layer gjurmon lidhjet të quajtura “sessions”, për shembull: ndjek shkarkimin e disa dosjeve njëherësh të kërkuar nga një aplikacion FTP i veçantë ose lidhjet e shumta telnet nga një stacion i vetëm klient ose rikthimet e faqeve të web-it nga një web server. Në botën e TCP/IP kjo është e menaxhuar nga softweri duke adresuar një lidhje tek një makineri në distancë dhe përdor numër të ndryshëm portesh lokale për çdo lidhje.

Session performon funksionet që vijojnë:

- Komunikimi me shtresën e paraqitjes më sipër,
- Organizon edhe menaxhon një ose me shumë lidhje për aplikim midis hosteve,
- Komunikimi me shtresën e transportit më poshtë.

Presentation Layer

Merr, shpaketon, dekodon dhe përkthen të dhënat në formate dhe kode që mund të përdoren lehtësisht nga programi i aplikimit ose anasjelltas. Shtresa e paraqitjes trajton

konvertimin e formateve të të dhënave, kështu që makineritë mund të 'paraqesin' të dhëna që janë krijuar në një sistem tjetër. Për shembull, trajton konvertimin e të dhënave të formatit JPG/JPEG në formatin 'Sun Raster' kështu që një makineri 'Sun' mund të shfaq një imazh JPG/JPEG. Shtresa e paraqitjes performon këto funksione:

- Komunikim me shtresën e aplikimit më lartë,
- Përkthimin e formateve standarte të të dhënave në formate të kuptueshme nga makineritë lokale,
- Komunikim me Session Layer më poshtë.

Shtresa e aplikimit

Shtresa e aplikimit është aplikacioni në përdorim nga përdoruesi. Për shembull, një kërkues në web, një FTP, IRC, një klient Telnet dhe aplikacione të tjera me bazë TCP/IP si versioni i rrjetit "Doom", "Quake", ose "Unreal".

Shtresa e aplikimit siguron interface të përdoruesit dhe është përgjegjës për shfaqjen e të dhënave dhe imazheve tek përdoruesi në një format të pranueshëm. Detyra e shtresës së aplikimit është të organizoj dhe shfaq të dhëna në një format të përshtatshem me njerëzit dhe te interface me shtresën e paraqitjes.

2.1.2 TCP/IP Modeli

Ashtu si modeli OSI, modeli TCP / IP shtresohet dhe përdoret në të njëjtën mënyrë si modeli OSI, por me më pak shtresa. Ndërsa interneti modern dhe shumica e komunikimeve përdorin Protokollin e Internetit (IP), modeli TCP/IP është teknikisht më në linjë me implementimet e rrjetit modern. Siç u tha më parë, shtresat brenda modelit TCP / IP konsiderohen më pak të ngurtë se ato të modelit OSI, që në thelb do të thotë se shumë protokolle të zbatuara mund të konsiderohen në zona të ngurtë midis një zone dhe një tjetre [4]. Komponenti i protokollit TCP/IP (shpesh i referuar si protokoll TCP/IP) përmban protokollet e njëjta të referuara në seksionet e mëparshme të modelit OSI. Figura më poshtë tregon shtresat e modelit TCP/IP:

Application	Layer 4
Transport	Layer 3
Internet	Layer 2
Link	Layer 1

Figura 3 - TCP/IP Modeli

The Link Layer

Shtresa e lidhjes është shtresa më e ulët e modelit TCP/IP, gjithashtu referohet në disa burime si shtresa e ndërfaqes së rrjetit. Shtresa e lidhjes kombinon funksionet e shtresës fizike dhe të lidhjes së të dhënave në një shtresë të vetme. Kjo përfshin funksione të rrjetit fizik të kornizës si modulimi, kodimi i linjës dhe sinkronizimi i biteve, sinkronizimi i kornizave dhe zbulimi i gabimeve, dhe funksionet e nënsistemit LLC dhe MAC. Protokollet e përbashkëta përfshijnë Address Resolution Protocol (ARP), Neighbor Discovery Protocol (NDP), IEEE 802.3 and IEEE 802.11.

The Internet Layer

Shtresa e Internetit është shtresa tjetër deri nga shtresa e lidhjes dhe shoqërohet me shtresën e rrjetit të modelit OSI. Funksionet përfshijnë routimin e trafikut, kontrollin e trafikut, fragmentimin dhe adresimin logjik. Protokollet e zakonshme përfshijnë IP, ICMP and IGMP.

The Transport Layer

Shtresa e transportit është shtresa e ardhshme dhe zakonisht lidhet drejtpërdrejt me të njëjtën shtresë me emrin në modelin OSI. Funksionet përfshijnë segmentimin e mesazhit, njohjen, kontrollin e trafikut, multipleksimin e sesionit, zbulimin dhe korrigjimin e gabimeve (ri-resendime), dhe riorganizimin e mesazhit. Protokollet e zakonshme përfshijnë Transport Control Protocol (TCP) [5] and User Datagram Protocol (UDP) [6].

The Application Layer

Shtresa e aplikimit është shtresa më e lartë në modelin TCP/IP dhe lidhet me sesionin, prezantimin dhe shtresat e aplikimit të modelit OSI. Shtresa e aplikimit të modelit TCP/IP përdoret për të trajtuar të gjitha funksionet e komunikimit në proces, këto funksione kryhen nga shtresa të ndryshme, kur referohen në modelin OSI. Ekzistojnë një numër funksionesh të ndryshme që kryhen nga kjo shtresë, duke përfshirë krijimin e sesioneve, mirëmbajtjen dhe përfundimin, përkthimin e kodeve të karaktereve, konvertimin e të dhënave, kompresimin dhe enkriptimin, qasjen në distancë, menaxhimin e rrjetit dhe mesazhet elektronike. Protokollet e përbashkëta përfshijnë Named Pipes, NetBIOS, MIME, TLS, SSL, FTP, DNS, HTTP, SMTP dhe shumë të tjerë.

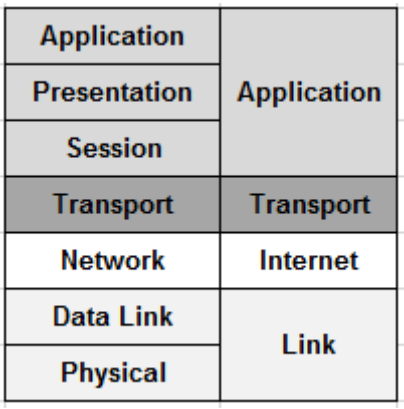

Figura 4 - OSI dhe TCP/Modeli

Figura më lartë tregon krahasimin në mes OSI dhe TCP/IP modelit. Shpesh herë ekziston mosqartësi midis këtyre dy modeleve të ndryshme si dhe kjo është e zakonshme për inxhinierët e rinj të rrjetit, pasi shumë prej tyre kanë të paktën disa njohuri për TCP/IP, por nuk kanë dëgjuar kurrë për OSI. Duhet të jetë e qartë se këto janë modele rreptësisht dhe duhet të konsiderohen entitete të ndara nga njëri-tjetri kur mësohen.

1.7 Topologjia e rrjetit

Lidhja fizike e pajisjeve në rrjet

Në një lidhje Point-to-Point, dy pajisje monopolizojnë një medium komunikimi. Për arsye se një medium nuk është shpërndarë, nuk është e nevojshme që një mekanizëm të identifikojë kompjuterat. Prandaj dy pajisje “pikë me pikë” nuk kanë nevojë për adresim.

Figura 5 - Topologjia Point to Point

Lidhjet pikë më pikë mund të jenë 'simplex', 'half-duplex', ose 'full-duplex'. Kur pajisjet duhet të angazhohen në komunikime dy-anësore në një half-duplex, disa mekanizma kthesë duhet të jenë në vend për të ndryshuar rolet e pajisjeve të dërgimit dhe marrjes.

Simplex: sinjali rrjedh në një drejtim, vetëm një stacion transmeton dhe tjetri merr;

Half-duplex: çdo stacion mund ti bëjë të dyja, të transmetojë dhe të marrë në të njëjtën kohë;

Full-duplex: Të dy stacionet transmetojnë dhe marrin njëherësh. Kapaciteti i linkut shpërndahet

ndërmjet të dyja pajisjeve ose nga dy rrugë të ndara transmetimi. Kapaciteti i kanalit është i ndarë për të transmetuar dhe për të marrë.

Shumë-pikësh (Multi-point):

Lidh tre ose më shumë pajisje së bashku përmes një mediumi të vetëm komunikimi. Për ndarjen e një kanali të vetëm, çdo pajisje ka nevojë për një mënyrë për të identifikuar vetveten si dhe pajisjen tek e cila dëshiron që të dërgojë informacion. Metoda e përdorur për të identifikuar dërguesit dhe marrësit është e quajtur adresim.

Topologjia magjistralë

Një topologji magjistrale lidh kompjuterët përgjatë një ose më shumë kabllosh për të lidhur në mënyrë lineare si në figurën 4. Një rrjet që përdor topologji magjistrale referohet si një "rrjet magjistral" i cili është forma origjinale e rrjeteve të Ethernetit. Ethernet 10Base2 (i njohur gjithashtu si 'thinnet') përdoret për topologjine magjistrale. Topologjia magjistrale është mënyra më e lirë e lidhjes së kompjuterëve për të formuar një grup pune ose LAN sektorial, por kjo ka edhe disavantazhin që një lidhje e vetme e lirë ose prishje kablloje mund të rrëzojë të gjithë LAN-in.

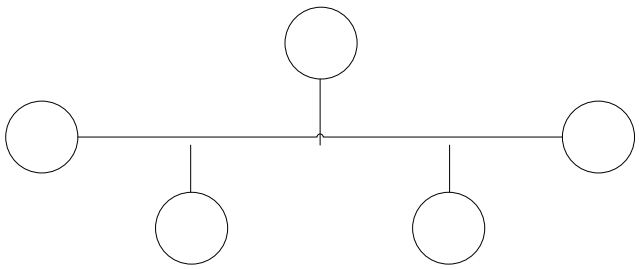

Figura 6 - Topologjia Bus

Përfundimi është çështje me rëndesi në rrjetet magjistrale. Sinjali elektrik nga një kompjuter që transmeton është i lirë të udhëtojë në të gjithë gjatësin e kabllos. Pa përfundimin, kur sinjali arrin fundin e telit, hidhet mbrapsht dhe udhëton mbrapsht në kabllo. Kur një sinjal kthen jehonën dhe përparon drejt një magjistraleje të papërfunduar, quhet tringëllimë. Përfunduesit thithin energjinë elektrike dhe ndalojnë reflektimin.

Topologjia yll

Një topologji yll lidh kompjuterat me kabllo individuale tek një njësi qendrore, zakonisht një HUB si në figurën 5. Kur një kompjuter ose pjesët përbërëse të rrjetit transmetojnë një sinjal në rrjet, sinjali udhëton në HUB. Më pas, ai e dërgon sinjalin njëkohësisht tek të gjithë përbërësit e

tjerë të lidhur me HUB. Ethernet 10BaseT është një rrjet i bazuar në topologjinë yll. Topologjia yll është mënyra më popullore për të lidhur kompjuterat në një grup pune ose në rrjet sektorial.

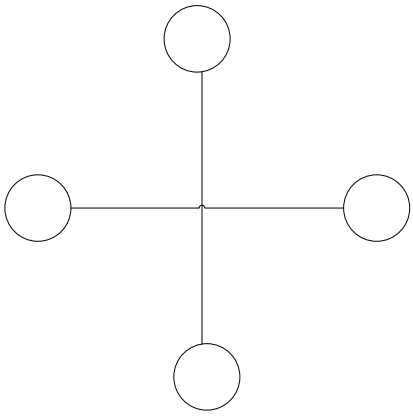

Figura 7 -Topologjia Star

Topologjia unazë

Një topologji unaze lidh kompjuterat përgjatë një rruge të vetme, fundet e së cilës janë të bashkuar për të formuar një rreth si në figuren 6. Rrethi mund të jetë vetëm logjik, por marrëveshja fizike e instalimit mund të jetë ngjashme me topologjinë yll, me një HUB ose përqëndrues në qendër. Topologjia unazë është përgjithsisht e përdorur në rrjetet token unazë dhe së bashku unaza fizike dhe logjike zakonisht lëviz përreth një kampusi ose grumbulli ndërtesash për të formuar një rrjet shtylle me shpejtësi të lartë.

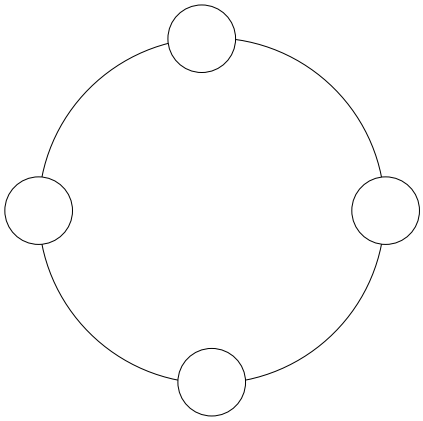

Figura 8 - Topologjia Ring

Hierarkia (Pema)Tree

Është topologji më e zakonshme që gjendet sot në korporatat e mëdha. Qëllimi i kësaj topologjie është që to ofertojë funksionalitet rrjet edhe në rast të dështimit si dhe zgjerueshmëri më të lehtë.

Topologjia Hibride

Kjo topologji është kombinim i topologjive të ndryshme, mundëson ndërlidhje të teknologjive, protokolleve dhe mediumeve të ndryshme, të gjitha mund të bashkëveprojnë si një rrjet.

E lirë

Ashtu si topologjia hibride, por pa qenë i nevojshëm një hardware shtesë për paketat e ndryshme të të dhënave midis topologjive. “LonWork” përdor këtë lloj topologjie.

1.8 Mediat për bartje të informacioneve

Llojet e lidhjeve që transmetojnë të dhëna midis nyjeve (pajisjeve) dallohen nga llojet e sinjaleve që bartin, ato mund të barten nepërmjet valëve të radios, kabllove - sinjaleve elektrike, si pulse të tingujve zanorë, si tufa drite përmes ajrit si dhe si tufa drite nëpërmjet një fibre optike. Mediumet e transmetimit mund të ndahen po ashtu ne dy grupe bazuar në ndërthueshmërine edhe atë medium i ndërthurur:

1. Çift telash të ndërthurur – përbëhet nga çifte të izoluara telash bakri të përdredhura rreth njeri tjetrit për të formuar një kabllo.
2. Kabllo coaxial – bëhet nga një tel qendror bakri, shtresë mburojash të izoluara, të gërshetuara metali dhe mburoje të jashtme.
3. Kabllo fiber optike – përbëhet nga një grumbull fijesh të holla xhami si dhe medium i pandërthurur:

1. Rrjetet Wireless – që përdorin valë radioje me frekuencë të caktuar për të transmetuar të dhëna.

Kryesisht mediumet transmetuese ballafaqohen me degradim sinjali për shkak të aftësive specifike të vet mediumit. Aftësitë e mediumit janë drejtëpërdrejtë të lidhura me llojin edhe nëse kemi të bejmë me çifte telash të ndërthurur apo kabllo coaxiale, po këto dy lloje mediumi bazën e ka në metalin e përdorur si përcues elektrikë në një frekuencë të caktuar, d.m.th. drejtëpërdrejtë lloji i metalit përçues veçohet me përçushmëri dhe rezistencë të caktuar, kryesisht të dy llojet përdorin bakrin si metal përçues por jo gjithmonë. Tek çift telash të përdredhur kryesisht një tel (fije metali) përdoret për të dërgu Tx Transmit dhe teli

tjetër për Rx – Receive. Ndërsa tek kabllot coaxiale meqenëse kemi nje fije primare, të përqendëndruar në qendër të kabllos, këtu po e njëjta fije përdoret për të dy proceset pra për dërgim dhe për pranim, kjo arrihet duke përdorë frekuencë të caktuar për secilin nga proceset e ndryshme me njëra-tjetrën. Këto lloje kabllosh gjejnë përdorim të madh në aplikimet që kanë të bëjnë me ndërlidhje të nyjesh në distanca jo shumë të mëdha. Kostoja e prodhimit dhe ndërlidhjes së këtijë lloji kabllosh është relativisht e ulët, procesi është jo shumë i komplikuar dhe shpejtë i mundshëm.

Sa u përket fibrave optike, si medium për bartje përdoret xham, tek i cili lëshohet sinjal drite dhe po i njëjti përcjell atë në anën tjetër. Prodhimi i kabllove të tilla është proces industrial i komplikuar dhe po i njëjti është i kushtueshëm gjithashtu ndërlidhja e dy fijeve optike është proces i komplikuar dhe po i njëti nuk mund të bëhet pa paisje të specialuzuara. Përparësi e këtij lloji kabllosh është se janë shumë pak të ndikuara nga ambienti dhe janë të pamvarura nga interferenca elektromagnetike, përparesi kjo që e bën të favorizueshëm në aplikime ku interferenca elekromagnetike është prezente. Ky lloj mediumi po ashtu gjen përdorim të madh në aplikimet, ku distanca ndërmejt dy nyjeve që janë të ndërlidhura është e madhe shpesh herë edhe në disa dhjetra apo qindra kilometra. Kabllot optike mund të jenë të dy llojeve të ndryshme edhe atë single mode ose multi mode.

Tek rrjetat pa tela ose wireless, mediumi përdorë ajrin si medium për barjte të të dhënave, ato mund të jenë si valë infra të kuqe IR ose si radio frekuncë RF brenda një rrjeti të lokalizuar. Fleksibiliteti dhe lëvizshmëria e kanë bërë teknologjinë wireless së fundmi të bëhet shumë e njohur, në rrjetet kompjuterike dhe rrjetet e kontrollit. Një rrjet wireless vepron ekzaktësisht njëjtë si një rrjet i lidhur me kabllo. Pajisjet janë thjesht të lidhura me një transmetues-marrës që bën të gjitha përkthimet dhe komunikimin e nevojshëm për të konvertuar sinjalet elektrike në sinjale radio. Pjesa më e madhe e rrjeteve të automatizimit kanë module wireless. Komisioni Federal i Komunikimit (FCC) i ka kushtuar tre gjatësi frekuencash për përdorim komercial: 900MHz, 2.4GHz, dhe 5.7GHz sepse në këto frekuenca ka shumë pak zhurmë industriale elektrike.

1.9 Komunikimi RS232/RS485

Të dhënat ndërmjet një pranuesi dhe një dërguesi mund të transmetohen në dy mënyra

kryesore: seriale dhe paralele.

Komuikimi serial është mënyra e transferimit të një biti nëpërmjet një mediumi të caktuar.

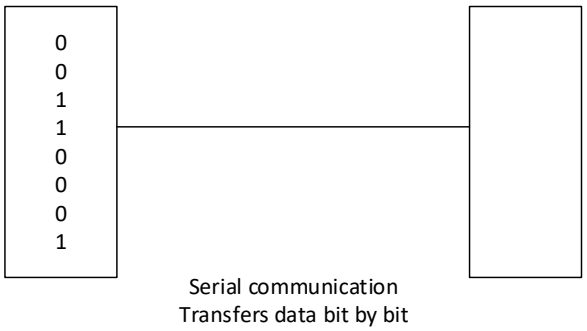

Figura 9 - Komunikimi serial

Komunikimi paralel është mënyra e transferimit të bloqeve p.sh. BYTE në të njëjtën kohë nëpërmjet një mediumi të caktuar.

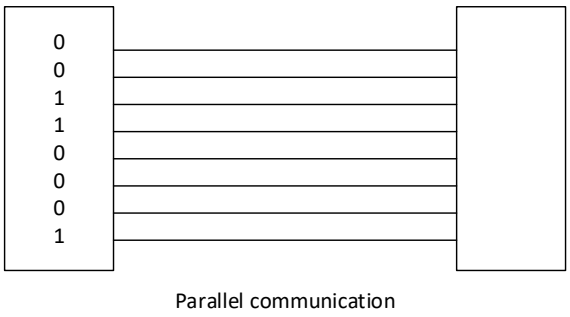

Figura 10 - Komunikimi paralel

Komunikimi serial

Komunikimi serial, si çdo transmetues të dhënash, kërkon koordinim midis dërguesit dhe marrësit. Për shembull, kur të niset një transmetim dhe kur të përfundohet ai, kur një bit ose byte i veçantë përfundon dhe një tjetër fillon, kur kapaciteti i marrësit është tejkaluar, edhe kështu në vazhdim. Një protokoll përcakton metodat specifike të koordinimit të transmetimit midis një dërguesi dhe një marrësi. Qëllimi i protokolleve të transmetimit serial të të dhënave është i gjërë dhe kompleks, përfshirë gjithçka nga lidhjet elektrike tek metodat e kodimit të të dhënave. Kjo pjesë përmbledh protokollet dhe standartet më të rëndsishme të lidhura me përdorimin e driverit serial.

Komunikimet sinkrone dhe asinkrone

Transferimi serial i të dhënave varet në minutazhin e saktë me kusht që të diferencojë bit-et në rrjedhjen e të dhënave. Ky minutazh mund të trajtohet në një nga dy mënyrat: sinkrone dhe asinkrone. Në komunikimin asinkron, qëllimi i minutazhit është një bytë i vetëm. Në komuinikimin sinkron, qëllimi i minutazhit përfshin një ose me shumë blloqe me byte. Termat asinkron dhe sinkron janë duke çorientua lehtësisht, sepse të dyja llojet e komunikimit kanë nevojë për sinkronizim midis dërguesit dhe marrësit.

Komunikimi asinkron është standardi mbizotërues në industrinë e kompjuterëve personal, sepse është më e lehtë për tu zbatuar dhe sepse ka avantazhin unik që byte-t mund të dërgohen kurdo që janë gati, në krahasim me pritjen e grumbullimit të blloqeve të të dhënave.

Asinkron nënkupton "jo sinkronizim" dhe kështu nuk kërkon dërgimin dhe marrjen e karaktereve në çdo lëvizje. Gjithsesi, fillimi dhe përfundimi i çdo byte të dhënash duhet të identifikohet nga bit-et e fillimit dhe ndalimit. Biti i fillimit tregon kur byte i të dhënave është gati për të filluar dhe bit-i i ndalimit sinjalizon kur ai mbaron. Kërkesa për të dërguar këto dy bite shtesë shkakton që komunikim asinkron të jetë lehtësisht më i ngadaltë sesa ai sinkron, gjithsesi ai ka avantazhin që procesori nuk ka për tu marrë me karaktere të tjerë që nuk lëvizin.

Komunikimi dy-drejtimesh

Porta seriale në PC tuaj është një pajise full-duplex që nënkupton se ajo mund të dërgojë dhe të marrë të dhëna në të njëjtën kohë. Me kusht që të jetë e aftë ta bëjë këtë, ajo përdor linja të ndara për transmetimin dhe marrjen e të dhënave. Disa lloje të pajisjeve seriale mbështesin vetëm komunikim një anësh dhe prandaj përdor vetëm dy tela në kabllo – linjat e transmetimit dhe terenit të sinjalit.

Komunikimi sinkron

Të dhënat sinkrone dhe ora

Kur modemi përdoret për të dërguar të dhëna, ora është e kombinuar me të dhënat për të

rregulluar një sinjal analog. Ora përfton sinjalin analog që është marrë dhe përdorur për të dekoduar rrymën e të dhënave sinkrone. Nëse modemi ndodhet në mënyrën sinkrone, ora është shpesh prodhim për portën seriale të modemit. Nësë modemi është në mënyrën sinkrone, ora që është prejardhje nga marrja e sinjalit analog përdoret për të dekoduar të dhënat, por nuk është prodhim në portën seriale të modemit.

Komunikimi asinkron

Që kur modelet e marrësit në qendër të çdo kohe të biteve, ajo mund të jetë sa ½ e kohës pushim të biteve (shumë herët ose shumë vonë) dhe ende të lexojë bitin në mënyrë korrekte. Që nga momenti kur ora e dërguesit dhe marrësit mund të jetë gabim, secili mund të jetë deri në ¼ kohë pushim e bitit (duke lejuar në rastin më të keq, në të cilin një është ¼ kohë biti më shpejt dhe tjetra është ¼ kohë biti më ngadalë). Që kur orët risinkronizohen në fillim të çdo karakteri (nga kalimi në bitin e fillimit), orët duhet të përshtaten se vetëm kështu ata mund të lëvizin më pak se ¼ kohë biti në (përafërsisht) 10 bit për karakter. Kjo kërkon një përpikshmëri prej ¼ kohë biti në 10 bite, e cila është (¼/10)=2.5%, e cila përmbushet lehtësisht.

Për arsye të besimit në komunikimin asinkron të të dhënave në shtimin e një biti të fillimit edhe të ndalimit në çdo karakter të të dhënave, është zakonisht më pak i efektshëm sesa komunikimi sinkron i të dhënave (për çdo 10 bit të dërguar, 2 janë sipër). 20% e kapacitetit të transmetimit është çuar dëm në bitin e fillimit dhe ndalimit, kështu që një linje 9,600-bits/s siguron vetëm 7,680bits/s të shpejtësissë së transmetimit. Disa njerëz e quajnë komunikim midis të dhënave fillim/ndalim (start/stop) në kontrast me komunikimin asinkron të të dhënave që janë zakonisht të përdorura.

Pajisjet e komunikimit asinkron të të dhënave kanë nevojë të konfigurohen për sa vijon:

- Shkallë e bitit (si 9,600 bits/s);
- Numrin e të dhënave të biteve për karakter (si 8);
- Tipin e barazimit të përdorur (si asnjë);
- Numrin e biteve të ndalimit (zakonisht 1, edhe pse kohëzgjatja e bitit të ndalimit prej 1,5 dhe 2 kohë-biti janë zakonisht të rregullueshme, por këto duheshin vetëm për

teleshkruesit e vjetër mekanik që kanë nevojë për më shumë se një kohë biti ndalimi midis karaktereve.

Kontrolli i rrjedhjes

Për arsye se një dërgues dhe një marrës nuk munden gjithmonë të procesojnë të dhënat në të njëjtën shkallë, kërkohen disa mënyra negocimi kur të fillohet dhe ndalohet transmetimi. Driveri serial mbështet dy metoda tv kontrollimit të rrjedhjes seriale të të dhënave. Një metodë mbështetet në hardware me portë seriale; tjetri është i përfshirë në software.

Kontrolli i rrjedhjes në hardware përdor dy nga linjat e sinjalit të portës seriale për të kontrolluar transmetimin e të dhënave. Kur driveri serial është gati për të pranuar të dhëna nga një pajisje e jashtme, ai pranon sinjalin e Data Terminal Ready (DTR) në pinin 1 të portit serial, të cilin e merr pajisja e jashtme përmes hyrjes së Data Set Ready (DSR) të saj.

Kontrolli i rrjedhjes gjithashtu mund të trajtohet në software duke përdorur një dakordësi e vendosur me karakteret si sinjale fillimi dhe ndalimi. Driverat serial mbështetin kontrollin e rrjedhjes XON/XOFF, i cili zakonisht cakton karakteret ASCII DC1 (të njohura gjithashtu si kontrolli-Q) si sinjali i fillimit dhe karakteri DC3 (Kodi I Pajisjes 3(kontrolli-S)) si sinjali ndalimit, edhe pse ju mund të zgjidhni karaktere të ndryshme.

Dukshëm, kontrolli i rrjedhjes është bërë si një pjesë e derdhjes së të dhënave; ju duhet të fusni gjëra speciale në të dhënat e juaja për ta bërë atë ose të ndalojë. Kjo teknikë quhet sinjalizimi brenda - kanalit. Dy disavantazhet më të mëdha të sinjalizimit Brenda-Kanalit janë që vazhdimisht kanë nevojë të monitorojnë derdhjen e të dhënave, me shpejtësi të lartë, për të gjetur sinjalet dhe nëse një nga DTE ndodh që të dërgojë një sinjal Brenda-Kanalit si pjesë e gjërave, gjëra të paparashikueshme mund të ndodhin! Shikoni pjesën e rrjedhjeve Brenda-Kanalit për më shumë informacione.

Norma Baud

Norma Baud është një sistemi komunikimi të dhënash, transferimi i një numri simbolesh në sekondë. Një simbol mund të ketë më shumë se një gjendje, kështu që mund të përfaqesojë më shumë se një bit binar (një bit binar gjithmonë përfaqëson ekzaktësisht dy gjendje). Edhe pse norma Baud nuk mund të barazojë shkallën e biteve, veçanërisht në rastin

e modemeve të kohëve të fundit, të cilët mund të kenë (për shembull) deri në nëntë bite për simbol.

Për shembull, një modem Bell 212A përdor modulimin Phase Shift Keying (PSK) [7] dhe ka një nga katër SHIFTET me katër faza (te 0(deg), 90(deg), 180(deg), ose270(deg)). Që nga momenti që kërkon dy bite për të përfaqësuar këtë gjendje (00, 01, 10, dhe 11), modemi transmeton 1.200bits/s informacion, duke përdoruru një normë simbol prej 600 baud. Zakonisht norma baud e një modemi nuk do të barazojë shkallën e biteve dhe është pa asnjë interes te përdoruesi fundor – vetëm shkalla e të dhënave, në bite për sekondë. Edhe pse është duke u bazuar tek shkalla e të dhënave të modemit, përdoren bits/s (ose kbits/s, etj.), jo normën baud. Baud nënkupton "ndryshimet e gjendjes të linjës për sekond".

Porta seriale RS232

Nivelet e voltazhit me respekt ndaj terrenit përfaqësojnë sinjalet RS 232. Gjendet një tel për çdo sinjal, së bashku me sinjalin terren (referencë ndaj niveleve te voltazhit). Kjo INTERFACE është e përdorshme për komunikimin point-to-point në shpejtësi të ulët. Për shembull, porta COM1 në një PC mundë përdoret për një mouse, porta COM2 për një modem, etj. Ky është një shembull i komunikimit point-to-point: një portë, një pajisje. Falë mënyrës si sinjalet janë të lidhura, kërkohet një terren i përbashkët. Kjo nënkupton gjatësi të limituar të kabllove – rreth 30-60 metra maksimumi. (Problemet kryesore janë ndërhyrja dhe rezistenca e kabllove.) Shkurtimisht, RS 232 është dizajnuar për komunikimin e pajisjeve lokale dhe mbështet një transmetues dhe një marrës.

Një nga pjesët me universale të PC është porta seriale. Ju mund të lidhni një mouse, një modem, një printer, një kompjuter tjetër, etj. Por përdorimi i tij (bashkë software edhe hardware) është një nga sekretet më të mirëmbajtura nga shumica e përdoruesve, përveç kësaj nuk është e vështirë për të kuptuar si lidhen pajisjet me të dhe si ta programojmë.

Këtu ndodhet lista e të gjitha sinjaleve të specifikuara në standartin RS232C. Secila shkronjë identifikon secilin sinjal. Numri Pin në një lidhje DB-25 dhe DB-9 dhe emrin e tij të sinjalit. Shkronjat e qarkut të lidhura me çdo sinjal janë të ndara si më poshtë:

- Nëse shkronja e parë është A, ky është një qark i zakonshëm;
- Nëse shkronja e dytë është B, ky është nje qark sinjal;

- Nëse shkronja e parë është C, ky është nje qark kontrolli;
- Nëse shkronja e parë është D, ky është një qark i sinkronizuar;
- Nëse shkronjat janë të paraprira nga një S, ky është një kanal dytësor.

Porta seriale RS- 485

RS- 485 është e përdorur nga komunikimet shumëpikëshe: disa pajisje mund të lidhen me një kabllo të vetme sinjali – e ngjashme me p.sh. Rrjetet Ethernet, të cilat përdorin kabllo coaxial. Shumica e sistemeve RS 485 përdorin arkitekturën Zotërues/Skllav, ku secila njësi skllave ka adresën e saj unike edhe i përgjigjet vetëm paketave të adresuara tek kjo njësi. Këto paketa janë të gjeneruara nga Zotëruesi (p.sh. PC), i cili në mënyrë periodike bën sondazhin e të gjitha njësive skllave të lidhura. Ky përshkrim do të mbulojë kryesisht arkitekturën Zotërues/Skllav sepse ai është i mjaftueshëm për 95% të aplikacioneve. Në raste speciale (sistemet e sigurisë), është përdorur një version i përmirësuar i komunikimit të multiprocesorëve. Ky sistem përdor një linjë të vetme për komunikimin dy-drejtimesh; gjithsesi nuk ka zotërues. Të gjitha njësitë prezantojnë transmetimin e paketave të një gjatësia të përcaktuar dhe në të njëjtën kohë dëgjojnë edhe se kur të dhënat janë transmetuar me sukses. Nëse nuk është rasti, ato ndalojnë së komunikuari dhe dëgjojnë se çfarë ka ndodhur. Në të njëjtën kohë, paketat urgjente mund të transmetohen mbi linjën. Sistemi është ideal për pajisjet që kanë nevojë të transferojnë menjëherë disa të dhëna shumë të rëndësishme dhe të dhëna të përditësuara, pa pritur për zotëruesin që ti japin atyre një mundësi për ta bërë. Në anën tjetër, transmferimi i dobishëm i të dhënave është më pak efektiv (rreth 30% më pak efektiv sesa sistemi i parë). Në arkitekturën Zotërues/Skllav, skllavi nuk fillon kurrë komunikimin. Është vendimtare për Zotëruesin të dërgojë adresa të sakta.

RS 485 gjendet në dy versione: 1 çift i përdredhur ose 2 çifte të përdredhura

Çifti i përdredhur i vetëm RS 485

Në këtë version, të gjitha pajisjet janë te lidhura me një çift të përdredhur të vetëm. Në këtë mënyrë, të gjitha ato duhet të kenë drivera me dalje tri-gjendjesh (duke përfshirë Zotëruesin). Komunikimi kalon mbi një linjë të vetme në të dyja drejtimet. Është e rëndësishme të parandalohet transmetimi i shumë pajisjeve njëherësh (probleme software).

Çifti i përdredhur dysh RS 485

Këtu, Zotëruesi nuk duhet të ketë dalje tri-gjendjesh, që nga momenti që pajisjet Skllave transmetojnë mbi një çift të përdredhur të dytë, i cili përdoret për dërgimin e të dhënave nga Skllavi te Zotëruesi. Kjo zgjidhje zakonisht lejon përdorimin shumëpikësh në sistem, i cili ishte që në origjinë i dizajnuar (HW ashtu si edhe SW) për RS232. Sigurisht, softwari Zotërues ka nevojë të modifikohet, kështu që Zotëruesi në mënyrë periodike dërgon paketa pyetjesh të gjitha pajisjeve skllave. Rritja e lëvizjes së të dhënave është e dallueshme në volume të mëdha.

Krahasimi midis RS-232 dhe 485

Pra, cili është ndryshimi kryesor midis RS 232 dhe 485? Nivelet e voltazhit që respektojnë terrenin përfaqsojnë sinjalet RS 232. Ka një tel për çdo sinjal, bashkë me sinjalin tokësor (referencë ndaj niveleve të voltazhit). Kjo INTERFACE është e përdorshme për komunikimin point-to-point në shpejtësi të ulta. Për shembull, porta COM1 në një PC mund të përdoret për një mouse, porta COM2 për një modem, etj. Ky është një shembull i komunikimit point-to-point: një portë, një pajisje. Falë mënyrës se si janë lidhur pajisjet, kërkohet një terren i përbashkët. Kjo përfshin gjatësi kabllosh të limituar- rreth 30 deri ne 60 metra maksimum. (Problemet kryesore janë ndërhyrja dhe rezistenca e kabllove.) Shkurtimisht, RS 232 është dizajnuar për komunikimin e pajisjeve lokale dhe mbështet një transmetues dhe një marrës.

RS 485 përdorin një parim të ndryshëm: çdo sinjal përdor një linjë me çift të përdredhur (TP)- dy kabllo të përdredhur rreth vetes. Po flasim për 'Transmetimin e Balancuar të të Dhënave', ose 'Transmetimin e Voltazheve të Ndryshme'. Thjeshtë, le të etiketojmë një nga kabllot TP me 'A' dhe tjetrin me 'B'. Atëherë sinjali është joaktiv kur voltazhi tek A është negativ dhe voltazhi tek B është pozitiv. Përndryshe, sinjali është aktiv, A është pozitive dhe B është negative. Sigurisht, ndryshimi midis kabllos A dhe B ka rëndësi. Për 485 kabllo mund të jetë deri në 1200 metra i gjatë dhe qarqet që gjenden zakonisht punojnë me shkallë transfer 2.5 MB/s.

3. Standardet e rrjetit të kontrollit

Ndërveprimi i sistemeve apo standardeve të ndryshme në një rrjet kontrolli përcaktohet nga aftësia e pajisjes e përcaktuar nga prodhuesi për të kuptuar dhe përdorur të dhënat nga pajisja e një prodhuesi tjetër, pavarësisht nga tipet e nënsistemit ose qëllimi origjinal pa ndërhyrjen e gateway apo konvertuesve të protokollit.

Me anë të ndërveprimit në rrjetat e kontrollit mund të arrihet që pajisjet të shpërndahen ndërmjet nën sistemeve të ndryshme, të ulen shpenzimet, të shkurtohet koha e instalimit, të ulet kompleksitetin meqë edhe pjeset janë duke u reduktuar, pajisjet në nënsisteme të ndryshme mund të ndërveprojnë me njeri tjetrin, prandaj, new-breed e aplikacioneve mund te krijohet thjeshtë, pronarët mund të zgjedhin produkte nga prodhues të ndryshëm, eliminimi i varësisë së gateway, veçanërisht gjatë përmirësimit të sistemit si dhe mundësia për lëvizje-shtim-ndryshim relativisht e thjeshtë.

Në industrinë aktuale standardet më të përdorua janë ARC.net, BACnet, EIB.Tb dhe më e përdorua LONWORKS. Teknologjia e rrjeteve të kontrollit LONWORKS adreson shumë sfida teknike dhe biznesi. Ajo e përmbush këtë duke shkuar përtej të qenurit thjeshtë një protokoll komunikimi dhe duke siguruar një platformë të plotë në të cilën të ndërtohen sistemet e kontrollit.

3.1 ARC.net

ARCNET është klasifikuar që në origjinë si një rrjet lokal ose LAN. Një LAN është i definuar si një grup nyjesh që komunikon me një tjetër mbi një zonë gjeografikisht të limituar, zakonisht brenda një ndërtese ose brenda një grupi ndërtesash. Ky ishte qëllimi i ARCNET kur u paraqit në origjinë si një automatizim zyre LAN nga DATAPOINT CORPORATION në vitet 1970. DATAPOINT parashikoi një rrjet me fuqi informatike të shpërndarë që operon si një kompjuter i madh. Ky system është referuar si ARC (kompjuter me burim të atashuar [Attached Resource Computer]) dhe rrjeti [NET], që lidh këto burime është quajtur ARCNET.

Përdorimi i ARCNET si një rrjet automatizimi zyre është zvogëluar. Gjithsesi ARCNET vazhdon të ketë sukses në industrinë e automatizimit industrial, sepse karakteristikat e tij të performancës janë të përshtatshme për kontroll. ARCNET e ka provuar veten që është i fuqishëm. ARCNET është gjithashtu i shpejtë, siguron performancë të përcaktuar dhe mund

të përballojë distancë të largët duke e berë atë të përshtatshme për teknologjinë FIELDBUS. Termi FIELDBUS është i përdorur në industrinë e automatizmit industrial për të nënkuptuar një rrjet që përbëhet nga kompjutera, kontrollues dhe pajisje të kaluara në ''fushë''.

ARCNET është një FIELDBUS ideal. Sikundër rrjetet e automatizimit në zyre, një FIELDBUS duhet të shpërndajë mesazhe në një mënyrë me kohe të parashikueshme. Protokolli i kalimit me TOKEN të ARCNET siguron këtë komoditet. Mesazhet FIELDBUS janë përgjithësisht të shkurtër. Gjatësitë e paketës ARCNET variojnë nga 0 ne 507 byte me një mbikalim të vogël dhe çiftohet me shkallën e lartë të të dhënave ARCNET, zakonisht 2.5 Mbps, japin reagim të shpejtë ndaj mesazheve të shkurtra. ARCNET ka ndërtim CRC-16 (kontrolli i tepricës ciklike) për kontroll gabimesh dhe mbështetë skema kabllimesh të ndryshme fizike duke përfshirë edhe fibrat optike. Protokolli i lidhjes së të dhënave ARCNET është i vetëpërmbajtur në CHIP-in kontrollues ARCNET. Funksionet e rrjetit si kontrolli i gabimeve, kontrolli i rrjedhës dhe konfigurimet e rrjetit bëhen në mënyrë automatike pa ndërhyrje të software-it.

Model	**Description**
90C26	First generation controller
90C65	XT Bus interface
90C98A	XT Bus interface
90C126	XT Bus interface
90C165	XT Bus interface
90C66	AT Bus interface
90C198	AT Bus interface
20010	Microcontroller interface
20019	Microcontroller interface
20020	Microcontroller interface
20022	Microcontroller interface
20051	Integral microcontroller
20051+	Integral microcontroller

Tabela 2 - ARCnet Kotroleret dhe pershkitmi i tyre

Në termat e modelit referencë të Organizatës Ndërkombëtare të Standardeve OSI (Open

System Interconnect), ARCNET siguron shtresat fizike dhe të lidhjes së të dhënave të këtij modeli. Me fjalë të tjera ARCNET siguron një transmetim dhe marrje të suksesshme të një pakete të dhënash midis dy nyjesh rrjeti. Një nyje i referon CHIP-it konrollues të ARCNET dhe kabllos transmetues-marrës të lidhur me rrjetin. Nyjet janë adresa të caktuara të quajtura ID MAC (medium access control) dhe një rrjet ARCNET mund të ketë deri në 255 nyje të caktuara në menyrë unike [8].

Çelësi për performancë ARCNET dhe tërheqja e tij si një rrjet kontrolli është protokolli kalimit të tij me TOKEN. Në një rrjet kalimi me token, një nyje mund të dërgojë mesazh vetëm kur merr ‘’token’’. Kur një nyje merr tokenin bëhen mjeshtri momental i rrjetit; gjithsesi, mjeshtëria e tij është jetëshkurtër. Gjatësia e mesazhit që mund të dërgohet është e limituar, asnjë nga nyjet nuk dominon rrjetin që nga momenti kur duhet të heq dorë nga kontrolli i tokenit. Sapo mesazhi të dërgohet, tokeni kalohet tek një nyje tjetër duke e lejuar atë të bëhet mjeshtri momental. Duke përdorur kalimin me token si një mekanizëm për ndërmjetësimin e hyrjes së rrjetit nga çdo nyje teke, koha e performancës së rrjetit bëhet e parashikueshme ose determinuese. Në fakt, në rastin më të keq koha që një nyje merr për të dërguar një mesazh tek një nyje tjetër mund të llogaritet. Rrjetet industriale kërkojnë performancë të parashikueshme për të siguruar që ngjarjet e kontrolluara ndodhin kur duhet, ARCNET e siguron këtë parashikueshmëri.

Një Token (ITT- Ftesë për të transmetuar) është një sekuencë sinjalizuese unike që kalohen në menyrë të rregullt përgjatë të gjitha nyjeve aktive në rrjet. Kur një nyje specifike e merr tokenin, ka të drejtën të fillojë një sekuencë transmetimi ose duhet të kaloje tokenin tek fqinji i tij llogjik. Ky fqinj, i cili mund të ndodhet fizikisht kudo në rrjet, ka adresën tjetër më të lartë tek nyja me token. Në momentin që tokeni kalohet, marrësi ka të drejtën të fillojë një transmetim. Kjo sekuencë kalimi tokeni vazhdon në mënyrë unaze llogjike duke u shërbyer të gjitha nyjeve në mënyrë të barabartë. Adresat e nyjeve duhet të jenë unike dhe mund të variojnë nga 0 deri në 255 me 0 të rezervuar për mesazhet BROADCAST. Për shembull, marrim një rrjet që përbëhet nga katër nyje të adresuara 6, 109, 122 dhe 255. Përcaktimet e nyjeve janë të pavarura nga vendndodhja fizike e tyre në rrjet. Sapo rrjeti të konfigurohet, tokeni kalon nga një nyje tek tjetra me adresë nyjesh më të lartë edhe pse një nyje tjetër mund të jetë fizikisht më e afërt. Të gjitha nyjet kanë një fqinjë llogjik dhe do të vazhdojë të kalojë tokenin tek fqinji sipas një unaze logjike pa marrë parasysh topologjinë fizike të rrjetit.

Në një sekuencë transmetimi, nyja me token bëhet nyja burim dhe çdo nyje tjetër e

përzgjedhur nga nyja burim për komunikim bëhet nyja destinacion. Së pari nyja burim verifikon nëse nyja destinacion është në pozicion për të pranuar një transmetim duke dërguar një FBE (Free Buffer Enquiry). Nyja destinacion përgjigjet duke kthyer ACK (Acknowledgement) që nënkupton së një BUFFER është i gjindshëm ose duke kthyer një NAK (Negative Acknowledgement) që nënkupton se asnjë BUFFER nuk është i gjindshëm. Sapo merr një ACK, nyja burim dërgon një transmetim të dhënash (PAC) me 0 deri në 507 byte të dhëna. Nëse të dhënat janë marrë siç duhet nga nyja destinacion siç faktohet nga një test CRC i suksesshëm, nyja destinacion dërgon një tjetër ACK. Nëse transmetimi është i pasuksesshëm, nyja destinacion nuk bën asgjë, duke shkaktuar mbarimin e kohës së nyjes burim. Nyja burim do të dëshmojë që transmetimi dështoi dhe do të riprovojë përsëri pasi të marrë tokenin gjatë kalimit tjetër të tokenit. Sekuenca e transmetimit përfundon dhe tokeni i kalohet nyjes tjetër. Nëse mesazhi i dëshiruar tejkalon 507 bytes, mesazhi dërgohet si një seri paketash, d.m.th. një paketë për çdo kalim tokeni. Ky quhet një mesazh i fragmentuar. Paketat janë të kombinuara tek fundi i destinacionit për të formuar të gjithë mesazhin.

ARCNET suporton një mesazh BROADCAST, i cili është një mesazh i pavërtetuar për të gjitha nyjet. Në vend që të dërgoj të njëjtin mesazh në nyjet individuale një mesazh pas tjetrit, ky mesazh mund të dërgohet tek të gjitha nyjet gjatë një transmetimi. Nyjet që janë aktivizuar për të marrë mesazhet BROADCAST do të marrin një mesazh që specifikon nyjen 0 si destinacion. Nyja 0 nuk ekziston në rrjet dhe është e rezervuar për këtë funksion të BROADCAST. Nuk dërgohen ACK apo NACK përgjatë një mesazhi BROADCAST duke e bërë këtë lloj mesazhi të shpejtë.

Një tipar tjetër i ARCNET është aftësia e tij për të konfiguruar rrjetin automatikisht nëse një nyje është e shtuar ose e fshirë nga rrjeti. Nëse një nyje bashkohet në rrjet, nuk merr pjesë automatikisht në sekuencën me kalim tokeni. Sapo një nyje vëren që nuk i është dhëne kurrë një token, ajo do të bllokojë rrjetin me një shpërthim ri-konfigurimesh që shkatërron sekuencën me kalim tokeni. Nëse tokeni humbet, të gjitha nyjet do të pushojnë transmetimin dhe fillojnë një sekuencë kohe boshe bazuar në adresat e veta si nyje. Nyja me adresën më të lartë do të fillojë e para kohën boshe dhe do të fillojë një sekuence kalimi tokenësh tek nyja me adresën tjetër më të lartë. Nëse kjo nyje nuk përgjigjet, merret sikur ajo nuk ekziston. Adresa e nyjes destinacion rritet në numër dhe tokeni ridërgohet. Kjo sekuence përsëritet derisa një nyje të përgjigjet. Në atë moment, tokeni i lëshohet nyjes që përgjigjet dhe adresa e kësaj nyje shënohet si fqinji logjik i nyjes së origjinës.

Sekuenca përsëritet nga të gjitha nyjet derisa secila nyje të mësoje fqinjin e saj logjik. Në kohën kur tokeni kalohet nga një fqinj tek tjetri pa humbur kohë në adresa që mungojnë. Nëse një nyje lë rrjetin sekuenca e rikonfigurimit është lehtësisht ndryshe. Kur një nyje lëshon tokenin tek fqinji i saj logjik, ajo fillon të monitorojë aktivitetin e rrjetit për të siguruar që fqinji logjik të përgjigjet me një kalim tokeni ose fillimin e një sekuence transmetimi. Nëse asnjë aktivitet nuk dërgohet, nyja që kaloi tokenin dëshmon që fqinji i saj logjik ka lënë rrjetin dhe fillon menjëherë kërkimin për një fqinj të ri logjik duke rritur në numër adresën e nyjes së fqinjit të saj logjik dhe fillon një kalim tokeni. Aktiviteti i rrjetit është përsëri i monitoruar dhe procesi i rritjes së numrit dhe ridërgimi i tokenit vazhdon deri sa një fqinj i ri logjik të gjendet. Sapo të gjendet, rrjeti rikthehet në rutinën e tij normale të unazës logjike me kalimin e tokenit tek fqinji logjik. Me ARCNET, rikonfigurimi i rrjetit është automatik dhe i shpejt pa ndonjë ndërhyrje të software-it.

ARCNETi është rrjeti me kabllo më fleksibel, ai suporton topologjitë bus, yll dhe yll të shpërndarë. Në një topologji BUS, të gjitha nyjet janë të lidhura me të njëjtin kabllo. Topologjia yll kërkon një pajisje të quajtur HUB (passive ose active) e cila përdoret për të përqëndruar kabllot nga secila nga nyjet. Ylli i shpërndarë (të gjitha nyjet lidhen me një HUB aktiv me të gjitha HUB-et së bashku) ofron fleksibilitetin më të madh dhe lejon rrjetin të përhapet më shumë se katër milje (6.7km) pa përdorimin e kohëve të zgjatura boshe. Mediumi mbeshtetës përfshin kablot koaksiale, çiftin e përdredhur të kabllove dhe fibrat optike.

Çdo nyje ARCNET kërkon një chip kontrollues ARCNET dhe një kabllo transmetues-marrës që zakonisht qëndron në një NIM (network interface module) [9]. NIM-s përmbajnë gjithashtu logjikën BUS INTERFACE që përputhet me strukturën bus që ata kanë. Këta përshtatës rrjeti janë të lëvizshëm dhe prandaj të njohur me termin ‘module’. NIM-et ARCNET janë të gjendshëm për të gjitha strukturat e përhapura komerciale bus. NIM-s ndryshojnë në termat e kontrolluesve ARCNET që inkorporojnë dhe nga kabllo transmetues-marrës që mbështesin.

Kontrolluesit ARCNET

Zemra e çdo NIM është një chip kontrollues ARCNET që formon bazën e një nyje ARCNET. Korporata DATAPOINT zhvilloi nyjen origjinale ARCNET si një implementim diskret elektronik, duke i referuar atij si një modul INTERFACE burim ose RIM. Korporata e mikrosistemeve

standade (SMSC) siguroi implementimin e parë dhe të madh të teknologjis me integrim shkallë (LSI). Që nga ajo kohë, prodhues të tjerë chipesh siguruan licensa për të prodhuar chipe RIM. Sot SMSC dhe ndihmësi Toyo Microsystems Corporation (TMC) sigurojnë udhëheqjen në dizajnimin e ri të chipave ARCNET.

Përdorimi i HUB-eve

HUB-et e thjeshtësojnë kabllimin duke ndërlidhur NIM-e të shumta dhe në shumicën e rasteve ato nuk ushtrojnë kontroll mbi rrjetin. Funksioni parësor i një HUB-i është të sigurojë një metodë të leverdisshëm për përhapjen e rrjetit. Ka dy lloje HUB-esh që mund ta përformojnë këtë detyrë – një hub pasiv ose një hub aktiv.

HUB-et passive

- HUB-et pasive janë jo të kushtueshme, nuk kërkojnë energji dhe qëllimi i tyre i vetëm është të përshtatin rezistencën e linjës, të cilën e bëjnë përmes rezistorëve. Këto HUB-e zakonisht kanë katër porta për të lidhur katër transmetues-marrësa coaxial yll. Një nga disavantazhet e këtyre HUB-eve është që kufizojnë rrjetin deri në 60metra (200feet) dhe secilin segment të rrjetit me 30 metra (100feet) . Gjithashtu, portat e papërdorura duhet të ndërpriten me një resistor 93ohm për funksionimin e duhur. HUB-et pasive përdoren në rrjete të vogla coaxial yll (katër nyje ose me pak).

- HUB-et aktive janë në thelb përsëritës elektronik. Edhe pse ato kërkojnë energji, HUB-et aktive përballojnë të gjitha mundësite e kabllimit, përballojnë distanca më të gjata se HUB-et pasive, sigurojnë izolim edhe ruajtje kundër defekteve të kabllimit dhe reflektimeve. Këta janë HUB-et të cilët përdoren për të kablluar rrjetet e shpërndara yll.

3.2 BACNet

BACneti (Automatizimi i ndërtimit dhe Rrjetet e kontrollit) është zhvilluar nga Shoqëria Amerikane e Inxhinierëve të ngrohjes, ftohjes dhe ajrit të kondicionuar (ASHRAE). BACneti është një standard global ISO. Standardi kombëtar amerikan, një parastandart europian dhe është i përdorur në më shumë sesa 30 shtete [10]. BACnet është një protokoll

komunikimi të dhënash ose bashkësi komunikimi rregullash, që ASHRAE krijoi me qëllim që të standartizojë komunikimin midis përbërësve të sistemit të automatizimit të të dhënave. BACnet lejon sisteme nga tregtues të ndryshëm, siç është HVAC, ndriçimi, sistemet e sigurisë dhe zjarrit, për të komunikuar me njëri-tjetrin duke siguruar metoda të standartizuara për paraqitjen, kërkimin, interpretimin dhe transportimin e të dhënave.

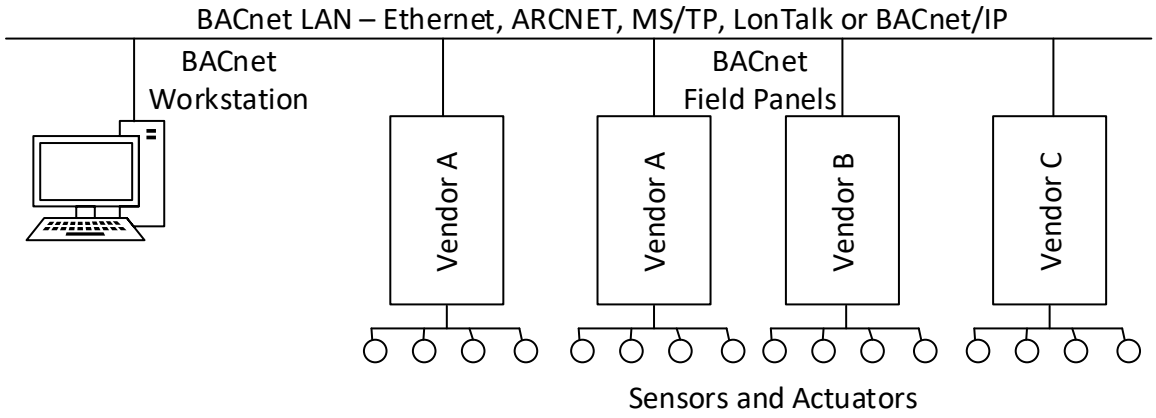

Figura 11 - BACnet dhe bashkëveprimi me rrjetet tjera për kontroll

Specifikimet e BACnetit përcaktojnë të gjitha aspektet e protokollit BACnet. Ai mund të përdoret për të shfaqur aspekte të ndryshme të një sistemi kontrolli. Shembujt janë:

- Një pajisje fizike (pjesët e pajisjes);
- Hyrje temperature (hyrje analoge);
- Ndërprerës (dalje binare);

Shërbimet – Shkëmbimi i informacioneve midis objekteve sigurohet nga shërbimet. Shërbimet janë të përdorura për të kryer leximet, shkrimet dhe I/O. Objekti që siguron shërbimin është një server dhe objekti që kërkon shërbimin është një klient. Shumica e objekteve mund të jenë së bashku një server dhe një klient, duke u varur nga nevojat e sistemit.

Karakteristikat – Një karakteristikë përmban informacion rreth një objekti. Objektet mund të përmbajnë një përmbledhje karakteristikash, disa nga të cilat mund të kërkohen nga lloje specifike objektesh. çdo objekt në BACnet duhet të ketë të paktën tre karakteristakat që vijojnë:

- identifikuesi objektit;
- emri objektit;
- lloji objektit.

Objektet BACnet

Protokolli BACnet me orientim ndaj objekteve. Pikat I/O, skedulimet dhe pajisjet janë shembuj objektesh. Një objekt mund të jetë një pikë e vetme ose një grup pikash që kryejnë një funksion të veçantë. Shumë mikroblloqe -VuR janë objekte BACnet. Një objekt BACnet paketon të gjitha informacionet që një sistem tjetër mund të ketë nevojë të kuptojë dhe punojë me objektin.

Çdo objekt ka një grup karakteristikash, siç është emri i objektit, lloji dhe vlera e momentit, që përshkruan sjelljen e tij ose drejton funksionimin e tij. Çdo karakteristik ka një emër dhe një vlerë. Të listuara me poshtë janë shembujt e karakteristikave që mund të ekzistojnë në një objekt analog hyrës BACnet:

Emri | Vlera

Emri i objektit | Temperatura e hapësirës

Ljoji i objektit | Hyrja analoge

Vlera aktuale | 74.1

Limiti i lartë | 78

Limiti i ulët | 68

Objektet standarde BACnet

BACnet përcakton llojet e objekteve që vijojnë si objekte standarte dhe përcakton sjelljen minimale të çdo objekti të kërkuar (karakteristikat që përmban dhe shërbimet që siguron).

OBJECT	EXAMPLE OF USE
Analog Input	Sensor input
Analog Output	Control output
Analog Value	Setpoint or other analog control system parameter
Binary Input	Switch input
Binary Output	Relay output
Binary Value	Binary (digital) control system parameter
Calendar	Defines a list of dates, such as holidays or special events, for scheduling.
Command	Writes multiple values to multiple objects in multiple devices to accomplish a specific purpose, such as day-mode to night-mode, or emergency mode.
Device	Properties tell what objects and services the device supports, and other device-specific information such as vendor, firmëare revision, etc.
Event Enrollment	Describes an event that might be an error condition (e.g., "Input out of range") or an alarm that other devices to know about. It can directly tell one device or use a Notification Class object to tell multiple devices.
File	Allows read and write access to data files supported by the device.

Group	Provides access to multiple properties of multiple objects in a read single operation.
Loop	Provides standardized access to a "control loop."
Multi-state Input	Represents the status of a multiple-state process, such as a refrigerator's On, Off, and Defrost cycles.
Multi-state Output	Represents the desired state of a multiple-state process (such as It's Time to Cool, It's Cold Enough and it's Time to Defrost).
Notification Class	Contains a list of devices to be informed if an Event Enrollment object determines that a warning or alarm message needs to be sent.
Program	Allows a program running in the device to be started, stopped, loaded and unloaded, and reports the present status of the program.
Schedule	Defines a weekly schedule of operations (performed by writing to specified list of objects ëith exceptions such as holidays. Can use a Calendar object for the exceptions.

Tabela 3 - Objektet standarde të BACnet-it

Objektet e prodhuesve BACnet

Standardet BACnet lejojnë objektet e zakonshme të krijohen nga tregtuesit. Transportuesi përdor objektet e zakonshme për mbështetje të ndërlidhjes dhe funksione të tjera në pronësi.

Objektet e pajisjes BACnet

Një objekt pajisje BACnet është një përfaqësim llogjik i një pjese të hardware-it të kontrollit, siç është kontrolluesi. Objektet e pajisjes janë një nga llojet më të rëndesishme të objekteve në protokollin BACnet, që nga momenti që ato përfaqësojnë kontrolluesit dhe përmbajnë një listë të objekteve pika të lidhura me pajisjen, siç është I/O I pajisjes. Objektet e pajisjes përmbajnë karakteristika dhe kryejnë shërbime siç është përgjigja ndaj kërkesave lexo-shkruaj për të siguruar informacion rreth hardware-it që ato përfaqësojnë. Një pajisje BACnet mundet të përmbajë gjithashtu një objekt program, i cili është logjika e kontrollit që pajisja zbaton.

Karakteristakat BACnet

Çdo objekt duhet të ketë një grumbull karakteristakash që përshkruajnë objektin. Shembuj janë:

- vlera momentale e një objekti hyrës (temperatura e hapësirës);
- njësitë e objektit hyrës (shkallet Fahrenheit);
- statusi i komunikimit të objektit hyrës (“mirë”).

Routerat BACnet

Një router BACnet transmeton mesazhet BACnet midis dy rrjeteve BACnet. Rrjetet mund të jenë të ndryshme (IP ne MS/TP) ose te njëjtën (IP ne IP). Routeri dërgon mesazhet e përshtatshme midis rrjeteve në të dyja drejtimet.

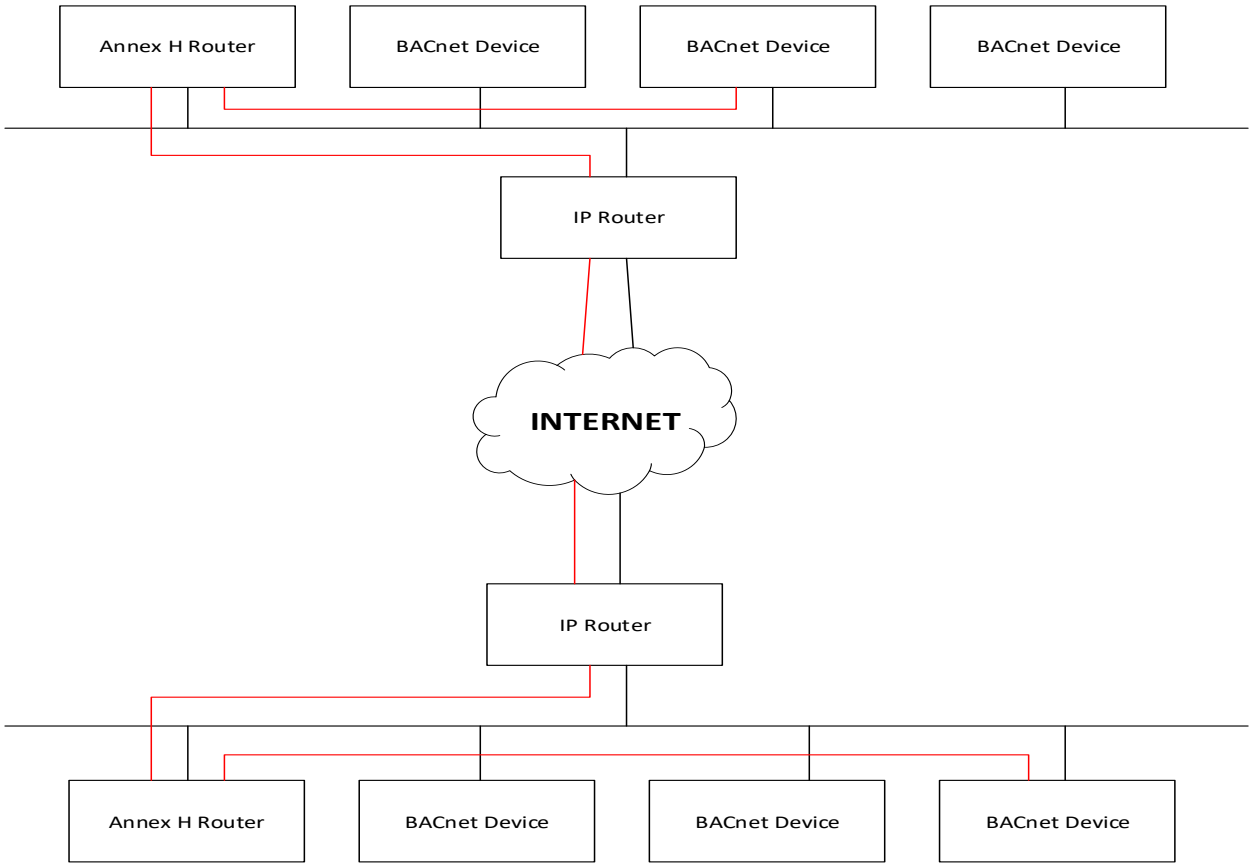

Figura 12 - BACnet - IP rrjeti

Teknologjitë e ndryshme rrjetesh sigurojnë nivele të ndryshme të shpejtësisë së transportit. Ljojet e rrjetit BACnet përfshijnë:

- Rrjete me shpejtësi të lartë për komunikime të nivelit të lartë (BACnet/IP);
- Nënrrjetet ekonomike MS/TP për kontrolluesit me specifikë aplikimi;

BACnet / IP

Rrjeti BACnet lejon mesazhet BACnet për tu komunikuar mbi një IP lokale dhe rrjetet me zonë të gjërë. BACnet përdor transmetimet IP për të lokalizuar dhe komunikuar me pajisje të tjera BACnet. Këto transmetime janë zakonisht të bllokuara nga routerat IP. Specifikimet BACnet përshkruajnë një metodë përdorimi të një pajisje për menaxhimin e transmetimeve

BACnet (BBMD) që lejon komunikimet BACnet/IP përmes routerave IP. BACnet/IP mbështetet shumë në transmetimet IP. Për të adresuar këtë çeshtje, BBMD duhet të përdoret.

MS/TP

Numri i rrjeteve MS/TP janë të përcaktuara nga routeri tek rrjeti MS/TP. Të gjeneruara automatikisht numrat e rrjetit BACnet MS/TP marrin forma si në vijim:

161XX

16 - ID e tregtuesit te BACnetit te transportuesit 1 - Rrjeti MS/TP XX - vlera e ROTARY SWITCHES të routerit.

SHËNIM: Në sistemet që qëndrojnë me vete, kur nuk përdoret router, XX - 01

Adresimet BACnet

Adresat BACnet gjenerohen automatikisht, bazuar në ndërpreresat rrotullues. Adresimi manual është i nevojshëm vetëm për kërkesa specifike të instalimit ose në rast konflikti me sisteme të tjera BACnet. Një adresë komplete BACnet duhet të identifikoje objektin dhe pajisjen që e referon atë. Një objekt ID është kombinimi i llojit të objektit dhe numrit të tij të instancës.

Rrjetet

Numri i rrjetit identifikon në mënyrë unike një rrjet në një sistem BACnet. Aplikimi i-VuR automatikisht gjeneron të gjitha numrat e rrjetit.

Sistem kontrolli tipik i-Vu(R) kanë të paktën një rrjet MS/TP me një shtyllë IP. Të gjitha sistemet e hapura i-Vu(R) duhet të kenë të paktën një rrjet MS/TP. Kur përdoren më shumë se një rrjet MS/TP, BACnet/IP mund të përdoren me routerat e hapur i-Vu(R) për tu lidhur me rrjetet MS/TP bashkë me një shtyllë kryesore të vetme BACnet/IP.

3.3 EIB.TB

European Installation Bus (EIB) është një sistem i hapur, gjithpërfshirës i cili mbulon të gjitha aspektet e automatizimit të ndërtimit. Menaxhohet nga një shoqëri neutrale EIB. Megjithëse blloqet e ndërtimit te standartizuara Bus Access Unit (BAU) janë të gjendshme nga shumë tregtarë, EIB-ja është specifikimi më i përafërt, jo një zbatim (siç është chipi ose një transmetues-marrës). Kjo nënkupton që EIB është e hapur: EIB mund të zbatohet nga

kushdo, në çdo platformë chip ose procesor të zgjedhur – të dyja si zbatim i pronësuar për produkte individuale, ashtu siç është edhe OEM BAU [11].

Topologjia e rrjetit

EIB është një rrjet tërësisht peer-to-peer, i cili akomodon deri në 65,536 pajisje. Topologjia logjike lejon deri në 256 pajisje në një linje. Ashtu siç është shfaqur në figure, 15 linjat mund të grupohen së bashku në një linjë të vetme në një zonë. Një fushë e tërë formohet nga zona bashkë me një linje kurrizore. Një medium i hapur, fushat e të cilës janë të ndara llogjikisht me një SystemID prej 16-bitesh. Pa adresat e përcaktuara për bashkuesit, (255 x 16) x 15 + 255 / 61,455 pajisje fundore mund të bashkangjiten tek një rrjet EIB. Kufizimet e instalimit mund të varen nga faktorët në zbatim (medium, llojet transmetues-marrës, furnizimi me energji, kapaciteti) dhe mjedisor (zhurma elektromagnetike...). Udhëzimet e produktit dhe instalimit duhet të merren parasysh.

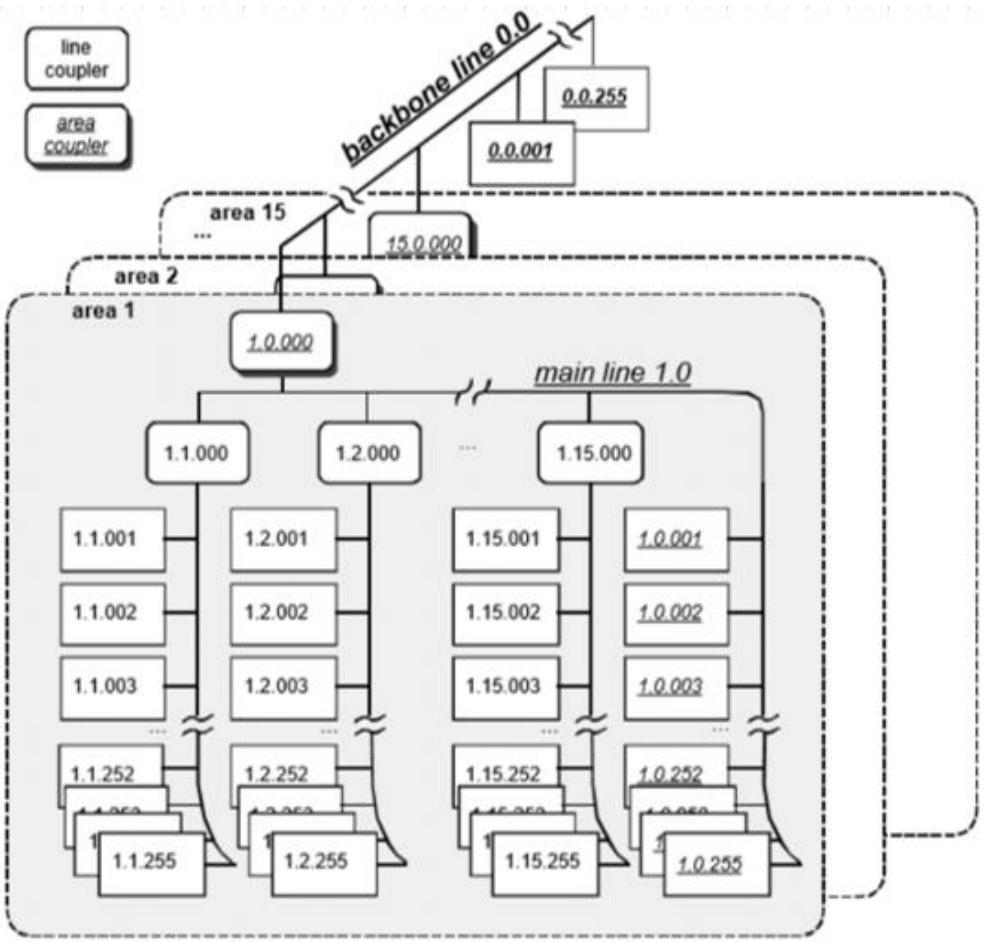

Figura 13 - Topologjia logjike e EIB

Bashkuesit lidhin linja dhe segmentet, p.sh. brenda mediumit TP (çifte te përdredhura) ose mediume të ndryshme; funksionaliteti i tyre mund të jetë (një kombinim) repeater, ure, router, filtrimi i paketave (për optimizimin e trafikut), mbrojtja firewall etj. EIB përcakton profile bashkuesish me standarde të ndryshme.

Mediumi

Ashtu siç është dhënë, EIB është shumë i optimizuar për çdo medium individualisht. Zbatimet e mundshme e përmirësojnë më tej për një kombinim të performancës transmetues-marrës dhe kostos. EIB.IR (infra te kuqe) është momentalisht duke u zhvilluar. EIB.MMS zgjat EIB me shërbime MultiMedia të dedikuara.

EIB.TP – Twisted Pair në EIB TP, zbulim përplasje në nivelin e bitëve me logjikë 0 dominuese siguron që në rast përplasje, transmetimi gjithmonë të dalë me sukses për një nga partnerët e komunikimit. Eliminimi që vjen si rezultat i ritransmetimit rrit më tej performancën e EIB TP. Bashkë me adresimin e fuqishëm në grup të EIB. Shmangia e përplasjes EIB TP1 furnizon për efikasitet ekstrem me kohën e reagimit 100ms për dy transmetime njëkohësisht. Votimi i shpejtë lejon deri në 14 pajisje për tu votuar për informacion statusi prej 1 byte brenda 50ns. Një segment fizik TP mund të jetë deri në 1000m i gjatë.

EIB.PL – Linjë energjie EIB PL (Powerline) përdor një teknike të re modulimi SPREAD FREQUENCY SHIFT KEYING. Më një filtër me korrespondim numerik të përshtatur, BAU që garanton komunikim të përshtatshëm që komunikimet për adresimet në grup të punojnë në mënyrë të besueshme në PL. Hyrja në Medium është e kontrolluar nëpërmjet një sekuence që bën hyrjen, me vende të caktuara rastësore për ritransmetim. Distanca maksimale midis dy pajisjeve (pa repeater): 600m. (Komunikimi ndikohet nga kushtet e ndotjes elektromagnetike në instalim).

EIB.RF – Radio Frekuenca EIB RF (Radio Frequency). Linjat janë fizikisht të ndara nga një transmetues i ndryshëm i frekuencës. Në kushtet e fushës së lirë, distanca e transmetimit është rreth 300m. Ritransmetimi siguron që volumet e mëdha mund të mbulohen brenda ndërtesës. Funksionaliteti i ritransmetuesit shpërndahet në mënyrë të përmirsuar ndërmjet pajisjeve të instaluara nga vetë sistemi.

EIB.net – Automation Networking. Specifikat EIB.net zbaton EIB në të gjitha mediumet me një shtresë lidhjesh logjike duke u bazuar në ISO/IEC 802-2, duke përfshirë Ethernet 10 Mbit.

Shtylla kryesore me shpejtësi të lartë jo të kufizuara, EIB.net lejon gjithashtu pajisjet me nivel menaxhimi ose automatizimi që të lidhen në mënyrë të drejtpërdrejtë.
Me EIB.net "i", EIB.net bëhet i drejtueshëm në distancë përmes rrjeteve ekzistuese në zyrë ose në ndërtesë – ose edhe të komandueshëm në distancë përmes Internetit – duke zbatuar protokollin e internetit (IP).

Protokolli i komunikimit EIB OSI

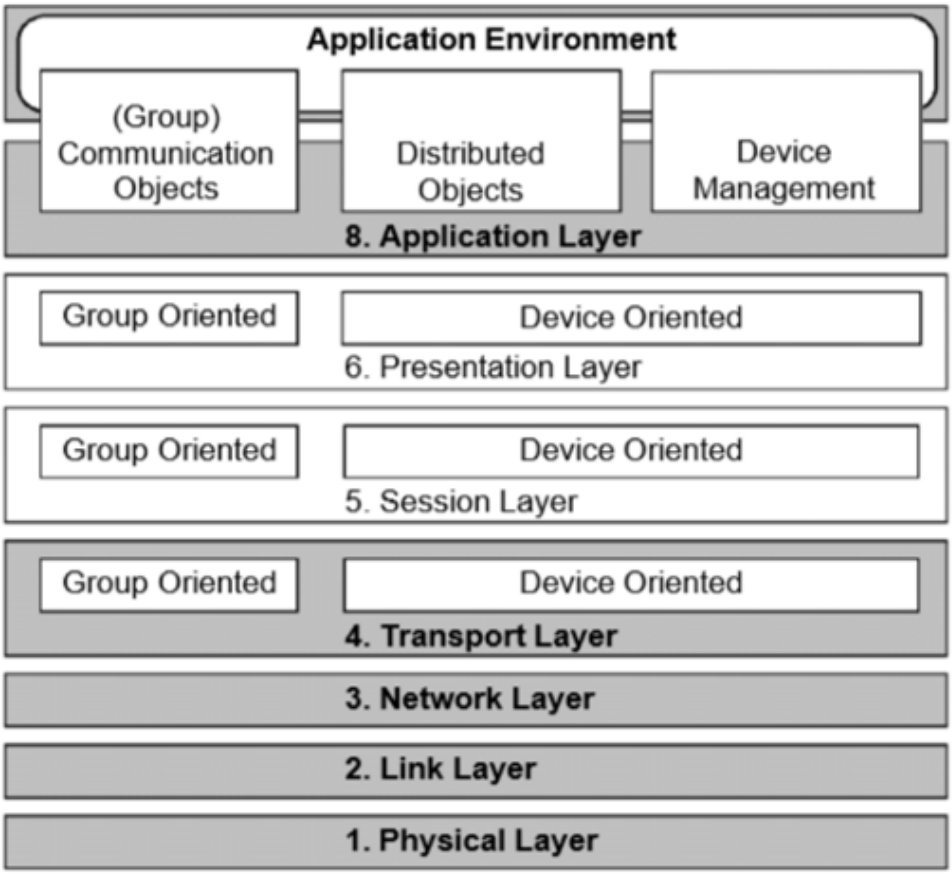

Figura 14 - EIB komunikimi sipas OSI modelit

Figura 14 tregon se si pirgu i komunikimit EIB është strukturuar sipas modelit me 7 shtresa OSI. Kjo është gjithashtu e reflektuar në strukturën kornizë të treguar në figurën 3. Shtresa fizike dhe shtresa lidhëse varen dukshëm në karakteristikat e mediumit fizik. Për MEDIUM ACCESS CONTROL, EIB përshkruan CSMA (carrier sense multiple access) me shmangie përplasje të optimizuar. Si është shpjeguar në seksionin 2, EIB mund të quhet "oportuniste" këtu, në të mekanizmi preciz mund të përmirësohet shumë për mediumin e veçantë. DAF

(the destination adress flag) dallon midis Grupit dhe telegramet me orientim ndaj pajisjes. Përmes Protokollit për Kontroll Informacioni të Rrjetit (NPCI), shtresa e rrjetit kontrollon numërimin e hopeve; për pajisje të tjera nga routersat ose urat janë të pa rëndësishme. Shtresa e transportit menaxhon marrëdhëniet e komunikimit logjik, të cilat mund të jenë:

1. një në shumë pa lidhje midis tyre (grup multicast);
2. një në shumë pa lidhje (broadcast);
3. një në një pa lidhje;
4. një në një me orintim lidhës, siguron vënien në hartë midis adresave dhe një përfaqësuesi të brendshëm abstrakt, Communication_Reference_ID (cr_id).

Të gjitha shërbimet janë të vëna në hartë në mënyrë transparente përmes shtresës së prezantimit dhe sesionit, të cilat janë të rezervuara. Shtresa e aplikimit zbaton për menaxhimit klient-server të rrjeteve EIB (shiko seksionin 6.1). Shtresa e aplikimit në grup merret me përcaktimin e një grupi cr_id në një instancë lokale të Objektit të Komunikimin në Grup (ose variables se ndare), për marrjen (one-to-one) dhe për dërgesën (one-to-one). Për lehtësi, të dyja grupet e komunikimit në grup dhe objektet e shpërndara janë të veshura nga shtresa e përdoruesit EIB, e cila merr përsipër shtresën e aplikimit nga thelbi i aplikimit. Shtresa e përdoruesit sillet si menaxhuesi i parazgjedhur i aplikimit të serverit.

octet 0	**1**	**2**	**3**	**4**	**5**	**6**		**7**	**8**	**..**	**N-1**	**N<22**
Controll Field	Source Address		Destination Address		DAF; NPCI; length	TP CI	APCI	data / APCI	Data			Check Octet

Tabela 4 - EIB PDU Struktura e kornizës

Kornizat PDU (EIB Protocol Data Unit) mund të përmbajnë forma aplikimi të dhënash deri në 14 byte (që zgjatet në 230 është momentalisht nën shqyrtim). Në seksionin tjetër, do të zbulojmë rëndësinë qëndrore të lehtësirave me orientim grupi të sistemit operativ EIB.

Menaxhimi i rrjetit EIB dhe adresimi

Menaxhimi i rrjetit – Për të menaxhuar burimet e rrjetit (p.sh. Kur konfiguron një instalim), EIB përdor një kombinim të broadcast dhe komunikimit point-to-point. Me anë të broadcast

(mundësisht duke përdorur numrin serial unik të një pajisje), çdo pajisje në instalim i jepet një adrese fizike unike e cila përdoret nga ai moment e tutje për komunikimet point-to-point. Një lidhje (mundësisht me autorizim hyrje) mund të ndërtohet, për shembull të shkarkohet një “applet” i plotë, imazh binar të një program aplikimi. Hyrja pa lidhje është e mundur në objektet e shpërndara EIB përmes adresimit <device>,<property>, si një mekanizëm autokton me nivle menaxhimi EIB për vizualizim të statusit dhe kontrollit. Një mënyrë e shpejtë votimi zotërues-skllav siguron kontrollin direkt dhe të statusit të nënsistemeve kritike.

Adresimi në grup Run-time Efficiency EIB suporton adresim multicast të plotë (grup). Nënkupton që:

1. EIB nuk është i limituar ndaj pajisjeve grupuese, çdo pajisje mund të publikojë variabla të ndryshëm (njihet si Objektet e Komunikimit (grup)) në mënyrë individuale, të cilat mund të grupohen pavarësisht nga njeri-tjetri në rrjetin e gjërë të variablave të shpërndarë. Si bonus, vetitë e objekteve të shpërndara mund të publikohen gjithashtu si variabël i shpërndarë.
2. Siç është shpjeguar më sipër në pirgjet e komunikimit EIB me orientim grupi, një variabel i shpërndarë mund të jetë plotësisht i lexuar/shkruar dydrejtimesh. Në këtë mënyrë, pajisjet mund të dërgojnë korniza të dhëna multicast.
3. EIB bën të mundshëm një hapësire 16 bit për këto variabla të shpërndarë. Edhe me kufizimin e disa zbatimeve në 15 bit, kjo nënkupton që një instalim mund të ketë deri në 32k variabla të shpërndara (për adresa grupi), secila me çdo numër të instancave lokale. Qëllimi dhe efikasiteti që rezulton e bën komunikimin e adresave në grup, e bën mënyrën “runtime” më të preferuarën për komunikimin me nivel fushë autonome të EIB. Në këtë mënyrë lehtësisht të papritur, EIB shkon drejt uljes së automatizimit të panevojshëm të niveleve hierarkike përmes adresimit të duhur dhe skemave modeluese të pajisjes.

Multi-klient Menaxhimi multi-server i rrjetit OO EIB. Një instalim EIB mund të shikohet si një grumbullim i burimeve të shpërndara, të cilat mund të menaxhohen përmes rrjetit. Në këtë fund çdo pajisje EIB zbaton një server të cilin siguron kontroll mbi burimet lokale (duke përfshirë shërbimet pritëse për CPU të jashtme ose burimet e memories ku hyhet përmes Interface-it fizik të jashtëm (PEI). Një seri e APCI i bën këto shërbime të përdorshme nga klientë në largësi. Përmes prezantimit të objekteve të shpërndara EIB, burimet e rrjetit zakonisht bëhen OO (object oriented). Klientët e menaxhimit tipikisht kanë hyrje në rrjet për kontroll apo për shërbime konfigurimi. EIBA zbaton një përcjellje totale të tregtuesve neutral, veglat standarte të konfigurimit bazuar në PC, që menaxhojnë “appletet” të shkarkueshme,

siç përshkruhet në 8. Pajisjet që mbahen në dorë janë gjithashtu të gjendshme. Klientët e bazuar në rrjet (kryesisht të montuara në DIN-rail) lejon vetë konfigurimin interaktiv (instalim i shpejtë) të nënsistemeve.

3.4 LonWorks

Nyjet LonWorks komunikojnë me njëri-tjetrin nëpërmjet protokollit LonWork që është i zbatuar në FIRMWARE në chip-in Neuron. Për të takuar objektivin e suportimit të aplikacioneve në një shkallë të gjërë industrish, protokolli LonTalk prezantohet si një grumbullim të shërbimeve nga të cilat dizajnuesi merr dhe zgjedh ashtu siç dëshiron. Udhëzimet e ndërveprimit LonWorks i sigurojnë drejtime dizajnueseve mbi të cilat bëhen zgjedhjet dhe çfarë vlerash të përdorin për të krijuar mundësinë produkteve të tyre të ndërveprojnë me produkte të tjera LonWorks [12].
Qëllimi i këtyre udhëzimeve është të sigurojnë një grupim standartesh që lejojnë produktet LonWorks të jenë të dizajnuara në mënyrë të pavarur dhe te integruara me nyje të tjera LonWorks dhe sisteme nga tregtues të tjerë pa nevojën për të zhvilluar kode aplikacioni, hardware ose vegla me porosi.

Segmenti

Termi segment i referohet një pjese teli të vetme të pandërprerë. Një segment suporton deri në 64 paisje.

Topologjitë e kablimit

Ka dy kategori të topologjive të kablimit të mbështetura nga FTT-10: topologjia free dhe dyfish të terminuara.

Gjatësia e kombinuar e kabllit që lidh paisjet e shumta në rrjetin LonWorks me një qasje topologjie free është e limituar deri në 500metra (1640feet) dhe kërkon një modul të vetëm përfundues (termination) të instaluar kudo në segment.

Në qasjen dyfishë terminuar, të gjitha kabllot çifte të përdredhura bëhen një bus linear ose zinxhir nuk mbështet T-ndërlidhese të asnjë lloji. çdo fund fizik i segmentit duhet të mbaroj duke përdorur një modul përfundimi dhe maksimumi i gjatësisë bus varet nga përmasa e telit (1400 metra ose 4590 feet duke përdoru 0.65mm2).

Kanalet, Paisjet përsëritëse dhe Routersat

Termi kanal “channel” është një term Echelon i përdorur për të përshkruar mediumin fizik tek i cili një nyje është e lidhur dhe llojin e trasmetues-marrësit që përdor për të komunikuar. Megjithatë protokolli LonTalk mbështet rrjete me segmente që përdorin mediume te ndryshme (kanale), të gjitha paisjet në një kanal të dhënë duhet të komunikojnë duke përdorur të njëjtin lloj trasmetues-marrës. Çdo nyje LonWorks është e lidhur fizikisht me një kanal. Forma fizike e kanalit varet nga mediumi: një kanal çifti të përdredhur është një çift telash i përdredhur; një kanal RF është një frekuencë radioje specifike; një kanal linje energjie është një seksion i vazhdueshëm i kabllimit energjie AC dhe kështu me rradhë.

Një repeater është i përdorur për të amplifkuar fuqinë e sinjaleve hyrëse dhe dërgimin e tij-ai nuk bën routim të përzgjedhur. Një segmet kabllimi i vetëm mbështet deri në 64 nyje (FTT-10 transmetues-marrës), por një kanal mund të shtrihet në segmente të shumta dhe që përdorin repaters shtrese fizike për të zgjeruar të dyja numrin e nyjeve dhe gjatësinë e rrjetit te kontrollit. Duke u varur nga aplikimi dhe performansa e pritshme e rrjetit (në kushte të kohës së përgjigjes), numrin maksimal të nyjeve të mbështetura nga një rrjet LonWorks është 32385.

Routersat përdoren për të menaxhuar trafikun e mesazheve të rrjetit, zgjerojnë përmasat fizike të një kanali (së bashku në terma të gjatësisë dhe numrit të paisjeve të lidhura) dhe për të lidhur kanalet që përdorin mediume të ndryshme (llojet e transmetues-marrësit). Routersat përbëhen nga një çift Neuronesh dhe transmetues-marrësish. Secili shikon një kanal të ndarë - ata lidhin dy kanale dhe bëjnë routim selektiv të paketave midis tyre. Routersat mund të konfigurohen duke përdorur një nga katër algoritmet e routimit që përdorin shkallë të ndryshme të intelegjencës duke përfshirë repeaters, bridges (urat), routerat e konfigurar dhe routerat që mësojnë.

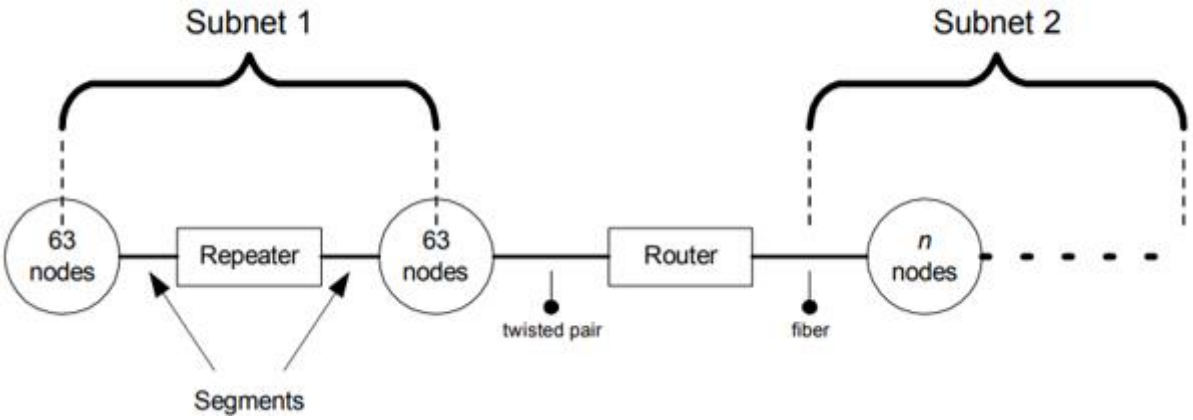

Figura 15 - Topologjia rrjet

Repaetersat:

- përdoren për të zgjeruar kapacitetin e rrjetit përtej 63 nyjes.

Routers:

- përdoren kur nyjet numërojnë përtej 127;
- përdoren kur gjatësia e telit tejkalon distancën maksimale 500 metra ose 1400 metra;
- ndajnë trafikun midis seksioneve të një rrjeti, në këtë rast reduktimin e trafikut;
- konverton nga një lloj mediumi në një tjetër. Psh; çift i përdredhur në Fiber.

Subnet:

- një nënrrjet është një grupim logjik deri në 127 paisjesh.

User interface:

- Një stacion pune (PC) mund të përdoret për interface në një rrjet paisjesh LonWorks. Ai duhet të përmbajë një adaptues komunikimi LonWorks dhe software. Lidhja mund të bëhet nga një lidhje seriale nëpërmjet një karte SLTA LonWorks (adaptues serial LonTalk) me modem ose nga karta PCC-10, të cilat sigurojnë lidhje direkte me një FieldBus LomWorks. Stacioni i punës llogaritet si një nyje në rrjet.

Adresimi logjik

Për të thjeshtësuar routimin, protokolli LonTalk përcakton një formë hierarkike të adresimit për të adresuar në mënyrë logjike çdo nyje. Në procesin e njohur si menaxhim rrjeti, softwari përdoret për të shoqëruar ID neuron unike 48 bitëshe të nyjes në një fushë logjike, subnet, dhe numër nyjesh (DSN). Mediumi i transmetimit (kanali) nuk ndikon mënyrën si një nyje

adresohet logjikisht. Fushat mund të përmbajnë kanale të ndryshme, ashtu siç munden subnetet dhe grupet. Të gjitha komunikimet përbëhen nga një ose më shumë pakete që shkëmbehen midis paisjeve. Çdo nyje në kanal shikon tek çdo paketë e transmetuar në kanal për të përcaktuar nëse është e dedikuar për atë nyje. Nëse një nyje gjen që një paketë është adresuar në atë nyje, ajo copton mesazhin për të përcaktuar nëse përmban të dhëna për një aplikacion që ekzekutohet në nyje ose nëse përmban informacione menaxhimi rrjeti. Varur nga përmbajtja e masazhit nyjet marrëse përgjijgjen me një "konfirmim" ACK, një përgjigje ose një mesazh autentikimi.

Variablat e rrjetit

Echelon përcakton një variabël rrjeti si një e dhënë që programi i aplikimit të një paisje të vaçantë pret të marrë nga paisjet e tjera në rrjet (nje variabel hyrës rrejti ose NVI) ose pret të bëj të qasshme paisjet e tjera në rrjet (një variabel dalës rrejti ose NVO). Shembujt e variablave të rrjetit përfshijnë temperaturën e zonës, temperatura e shkarkimit të ajrit, lagështinë relative, ndryshimi i pozicionit dhe gjendjen e zënë dhe të pa zënë. Lidhësi i variablave të rrjetit lejon paisjet të shpërndajnë të dhëna.

SNVT

Me kusht që aplikacionet nga prodhues të ndryshëm të ndërveprojnë në mënyrë të thjeshtë, variablat e tyre të rrjetit duhet të interpretohen në të njëjtën mënyrë. Organizata LonMark ka përcaktuar dhe publikuar variabla të përbashkëta sistemi, të quajtura lloje standarde variablash rrejti (SNVTs shqiptohet "sniv-its"). Çdo SNVT ka veti të standardizuara që janë përdorur për të përcaktuar variablat e shpërndarë në rrjet. Këto veti përfshijnë konteskstin e variablës (temperatura, adresa, etj.), njësia e matjes (grade, volt, etj.) dhe përmasat (litra, gallon, GPM, KGPH, etj.).

Bindings (lidhje)

Me kusht që paisjet LonWorks të shpërndajnë të dhëna (ndërveprojnë), variablat e rrjetit kanë aftësinë për tu lidhur njëri me tjetrin në rrjet. Ka dy rregulla të lidhura me lidhjen e variablave të rrjetit:

- Për të lidhur dy variabla rrjeti, njëra duhet të jetë dalëse ((output) (NVO)) dhe tjetra duhet të jetë një hyrës-input (NVI). Një NVO mund të lidhet me shumë NVI.

- Për të lidhur dy variabla rrjeti, SNVT-ja e variablave duhet të përputhet. Kjo siguron përputhshmëri të informacionit.

Një shembull i lidhjes së variablave të rrjetit mund të përfshijë daljen e temperaturës së hapsirës së matur nga një termostat i lidhur me hyrjen e temperaturës së hapsirës ose më shumë kontrolluesëve VAV që kontrollojnë këtë variabël.
SNVT-të e lidhura lejojnë komunikimet peer-to-peer midis paisjeve LonWorks - pa nevojë të kontrolluesit të hapësirës.

Parametrat e konfigurimit

Parametrat e konfigurimit janë pjesë specifike e informacionit që janë përdorur për të përcaktuar sjelljen e nyjes. Për shembull, vetitë e konfigurimit janë shpesh të përdorura për të përcaktuar limitet e alarmit dhe limitet e operimit.
Ekzistojnë dy tipe të parametrave të konfigurimit.

SCPT

Standard Configuration Parameter Types - Llojet standarde të parametrave të konfigurimit – shqiptuar skip-its. Këto janë të përcaktuara nga shoqëria LonMark dhe janë përdorur nga prodhuesit për të përcaktuar të dhënat e përbashkëta të konfigurimit. Shembujt e SCPT përfshijnë pikë vendosjen, vlerat e parazgjedhura, limitet minimale dhe maksimale, veçoritë e fitimit dhe koha e vonesës.

UCPT

User-definet Configuration Parameter Types – Llojet e parametrave të konfigurimit të definuara nga përdoruesi. Sipas nevojës prodhuesit mund të definojnë parametrat e tyre të konfigurimit. Nëse kjo përdoret, prodhuesit duhet të dokumentojnë UCPT-tv me burimet e të dhënave të asocuara me paisjen.

Profilet funksionale

Aty ku tipet standarde të variablave të rrjetit përcaktojnë të dhëna brenda variablave të rrjetit, një nivel më i lartë i standardizimit përcakton funksionalitetin për paisjet. Ky nivel standardizimi është referuar tek një profil funksional. Prodhuesit e paisjeve përdorin profile

për të bërë publik variablat e rrjetit, vetitë e konfigurimit, sjelljen e parazgjedhur dhe sjelljen power-up për paisjet që ato prodhojnë. Profilet standardizojnë sjelljen funksionale, jo produktet dhe nuk ka rëndësi kush prodhon një paisje LonMark gratis, të dyja paisjet do të jenë të afta të shpërndajnë të dhëna. Për më shumë, interfejsi në paisje bëhet më i parashikueshëm dhe konsistent.

ID e programit

Një ID programi është një identifikues unik për funksionalitetin e paisjes që është përfshirë në një paisje LonWorks. Paisjet që janë të çertifikara nga shoqëria LonMark përmbajnë ID programi në një format standard, që përfshinë prodhuesit, funksionalitetin e paisjes, llojin e përdorur të transmetues-marrësit dhe dokumentimin lidhur me qëllimin e përdorimit të paisjes. ID e programit janë përdorur nga veglat e menaxhimit të rrjetit për të identifikuar nyjet në rrjetin LonWorks.

4. Lon Talk

Chipi Neuron zbaton protokoll të plotë rrjeti duke përdorur të tre CPU-të në chip. Ky protokoll rrjeti ndjek modelin e referencës ISO - OSI për protolollet e rrjetit, i cili mbështet adresimin fleksibil dhe kanalet e shumta të komunikimit në një rrjet të vetëm. Programet e aplikimit që janë duke punuar në nyjet LONWORKS përdorin këtë protokoll për të komunikuar me aplikacionet që punojnë në nyjet e tjera LONWORKS diku në të njëjtin rrjet. Procesorët në chipin Neuron janë përdorur për të kryer software-in protokollar njëkohësisht programin e aplikimit [12].

Tiparet kryesore të protokollit LonTalk janë:

Suporti i shumëfishtë i mediumeve

Protokolli që përpunon në chipin Neuron është i pavarur nga mediumet. Kjo lejon që chipi Neuron të mbështesë një varietet të gjërë të mediumeve të komunikimit, duke përfshirë çiftin e kabllove të përdredhura, linjat e energjisë, frekuencat radio, rrezet infra të kuqe, kabllot koaxial dhe fibrat optike.

Mbështetje për kanalet e shumëfishta të komunikimit

Një kanal është një medium transporti fizik për paketat. Një rrjet mund të përbëhet nga një ose më shumë kanale. Me kusht që paketat të transferohen nga një kanal tek tjetri, një pajisje e quajtur router ose përsëritës përdoret për të lidhur të dy kanalet. Një router përfshin dy transmetues-marrësa që komunikojnë midis dy kanaleve. Protokolli LonTalk mbështet një pajisje të tillë kështu që rrjetet multi - mediale të mund të ndërtohen dhe ngarkimi i rrjetit mund të optimizohet duke lokalizuar trafikun.

Normat e komunikimit

Kanalet mund të konfigurohen për shkallë të ndryshme bit-ësh për të lejuar shkëmbimin e distancave, xhiron dhe konsumin e energjisë. Shkallët e lejuara të bit-eve janë 0.6, 1.2, 2.4, 4.9, 9.8, 19.5, 39.1, 78.1, 156.3, 312.5, 625 dhe 1,250 kbit-s. Xhiroja e kanalit varet nga shkalla e bit, valët e oscilatorit dhe saktësia, karakteristikat e transmetues-marrësit, përmasa mesatare e një pakete dhe pëdorimi i konfirmimit, prioritetit dhe autentifikimit. Një paketë

mesatare është në shtrirjen e 10 deri në 16 byte të gjatë, duke u varur në gjatësinë e identifikuesit të fushës, mënyrën e adresimit dhe madhesinë e fushës së të dhënave për përditësimin e variables së rrjetit ose një mesazh eksplicit. Përmasa më e madhe e paketës është 255 bajt duke përfshirë të dhënat, adresimin, kalimin e protokollit.

Kufinjtë e adresimit LonTalk

Niveli më i lartë i hierarkisë së adresimit është një fushë. Për shembull, nëse aplikime të ndryshme të rrjetit janë zbatuar në një medium të përbashkët komunikimesh siç është RF, identifikues të ndryshëm të fushës mund të përdoren për ti mbajtur aplikimet plotësisht të ndarë. Identifikuesi i fushës është i zgjedhshëm për të qenë 0, 1, 3, ose 6 bytes i gjatë. Një nyje e vetme mund të jetë një pjesëtar deri në dy fusha. Një nënrrjet është një grupim logjik nyjesh nga një ose më shumë kanale. Një router inteligjent operon në nivel nënrrjeti. Përcakton cila nënrrjet shtrihet në cilët anë të saj dhe nis paketat në perputhje me rrethanat. Niveli i tretë i adresimit është nyja. Mund të gjenden deri në 127 nyje për nënrrjet. Kështu një maksimum prej 255 x 127 = 32,385 nyje mund të gjenden në një fushë të vetme. Secila nyje mund të jetë nje anëtar i një ose dy fushave, duke lejuar një nyje të shërbejë si një inter-domain gateway. Kjo gjithashtu lejon, për shembull, një nyje të vetme sensor të transmetojë daljet e tij në dy fusha të ndryshme. Nyjet mund të grupohen gjithashtu. Grupet e nyjeve mund të zgjërohen në nënrrjeta të ndryshme brenda një fushe. Grupet mund të zgjerohen në mediume të ndryshme transmetimi ashtu si edhe kanale të ndryshme. Deri në 256 grupe mund të përcaktohen brenda një fushe dhe deri në 64 nyje mund të jene në një grup për shërbim konfirmimi ose kërkesë-përgjigje. Për shërbimet e pakonfirmuara brenda një fushe, një numër i palimituar nyjesh mund ti përkasin një grupi. Një nyje e vetme mund të jetë anëtar i deri në 15 grupeve për marrje mesazhesh. Adresimi në grup ulë numrin e byteve të informacionit të adreses së transmetuar me do mesazh dhe gjithashtu lejon shumë nyje të marrin një pjeseë të informacionit duke përdorur një mesazh të vetëm në rrjet [13].

Shërbimet e mesazhit

Protokolli LonTalk ofron katër tipe baze të shërbimit të mesazhit. Dy llojet e para të shërbimit janë të konfirmuara end-to-end dhe përfshin të mëposhtmet:

Të konfirmuar (ACKD), ku një mesazh është dërguar tek një nyje ose një grup nyjesh dhe

konfirmime individuale priten nga çdo marrës. Nëse konfirmimet nuk janë të gjitha të marra, dërguesit i mbaron koha dhe përsërit transfertën. Numri i përpjekjeve dhe koha e mbaruar (time-out) janë të dyja të zgjedhshme. Konfirmimet gjenerohen nga rrjeti i CPU-së pa ndërhyrjen e aplikimit. ID-të e transfertës bëjnë gjurmimin e mesazheve dhe konfirmimeve, kështu që një aplikim nuk merr mesazh të dyfishtë.
Kërkesë-përgjigje (REQUEST), ku një mesazh i dërgohet një nyjeje ose grupi nyjesh dhe përgjigjet individuale priten nga çdo marrës. Mesazhi në ardhje procesohet nga aplikimi në anën marrëse përpara se të gjenerohet një përgjigje. I njëjti opsion përpjekje dhe mbarim kohor janë të mundshëm siç janë edhe me shërbimin ACKD. Përgjigjet mund të përfshijne të dhëna, kështu që ky shërbim është veçanërisht i përshtatshëm për thirrjet me procedurë në distancë ose aplikimet klient/server.

Dy llojet e mbetura të shërbimit janë të pakonfirmuara. Ato janë si më poshtë:
Të përsëritura (UNACKD_RPT), ku një mesazh i dërgohet një nyje ose grupi nyjesh shumë herë dhe nuk pritet asnjë përgjigje. Ky shërbim është tipikisht i përdorur MULTICASTING në grupet e mëdha të nyjeve, një situate në të cilën trafiku i gjeneruar nga të gjitha përgjigjet do të mbingarkonte rrjetin.

Të pakonfirmuara (UNACKD), ku një mesazh i dërgohet njëherë një nyje ose një grupi nyjesh dhe asnjë përgjigje nuk pritet. Kjo përdoret tipikisht kur shkalla më e lartë e arritshme e transmetimit kërkohet ose kur sasi të mëdha të dhënash janë për tu transferuar. Kur përdoret ky shërbim, aplikimi nuk duhet të jetë i ndjeshëm ndaj humbjes rastësore të një mesazhi.

Protokolli LonTalk siguron zbulimin e mesazheve të dyfishta dhe normalisht dërgon vetëm një herë një mesazh tek aplikimi destinacion edhe kur mesazhi ka qenë i dyfishtë. Paketat e dyfishta mund të ndodhin kur konfirmimet dhe përgjigjet janë humbur, kur paketat janë padashur sipër në mediume të hapura si RF dhe linja e energjisë edhe kur përdoret shërbimi i përsëritur i pakonfirmuar. Vetëm në rastin e shërbimit kërkesë-përgjigje, ku përgjigja përfshin të dhëna ndryshe nga kodi i mesazhit, dërgohet një mesazh më shumë se një herë te aplikimi destinacion. Aftësia e zbulimit të kopjeve është e siguruar nga një database transaksionale në çdo nyje.

Autentikimi

Protokolli LonTalk mbështet mesazhet e verifikuara, që lejojnë marrësit e një mesazhi të përcaktojnë nëse dërguesi është i autorizuar të dërgojë atë mesazh. Kjo përdoret për të parandaluar hyrjen e paautorizuar në të ose kontrollin e nyjeve dhe aplikimet e tyre. Verifikimi zbatohet nga shpërndarja e çelësave 48-bitesh, një për secilën fushë, tek nyjet në momentin e instalimit. Që nje mesazh i verifikuar të pranohet nga marrësi, së bashku dërguesi dhe marrësi duhet të zotërojnë të njëjtin çelës. Ky çelës është i ndryshëm nga ai i nyjes Neuron ID.

Prioriteti

Protokolli LonTalk ofron një mekanizëm prioritar opsional për të përmirësuar kohën e përgjigjes për paketat kritike. Protokolli lejon përdoruesin të specifikojë PRIORITY TIME SLOTS në një kanal, i dedikuar nyjeve prioritare. Çdo PRIORITY TIME SLOT në një kanal shton kohë në transmetimin e çdo mesazhi, por kapaciteti i kushtuar atyre është i gjindshëm në fund të çdo pakete për hyrje prioritare pa asnjë problem për kanalin.

Shmangia e përplasjeve

Protokolli LonTalk përdor një algoritëm unik për shmangien e përplasjeve i cili ka aftësinë që nën kushtet e mbingarkimit, kanali mundet ende të mbajë kapacitetin e tij maksimal, përveçse kur xhiroja e tij degradon për shkak të përplasjes së tepërt.

Zbulimiil përplasjeve

Me transmetues-marrësin e komunikimeve që mbështesin zbulimin e përplasjeve të hardware-it, protokolli LonTalk mbështet zbulimin e përplasjes dhe ritransmetimin automatik. Kjo lejon një nyje që të ritransmetojë paketën shumë më herët se nëse një do ishte mbështetur vetëm ne mbarimet e kohës në shtresën e sipërme. Ritransmetimi i paketës është subjekt në vonesat hyrëse të mediumeve normale. Detajet që mbesin nga veprimi i zbulimit të përplasjes variojnë si një funksion i PORT MODE të komunikimit.

Interpretimi i të dhënave

Një varg i veçantë i kodeve të të dhënave ruhet për transmetim të kornizave të huaja. Deri në 288 byte të dhëna mund të jetë ngulitur në një paketë mesazhi dhe transmetohet si çdo

mesazh tjetër. Protokolli LonTalk nuk aplikon procesim të veçantë ndaj kornizave të huaja, ato trajtohen si një grup i thjeshtë bytesh. Programi i aplikimit mund të interpretojë të dhënat në çdo mënyrë që dëshiron.

Menaxhimi i rrjetit dhe shërbimet e diagnostifikimit

Menaxhimi i rrjetit dhe shërbimet e diagnostifikimit sigurohen nga dy kategori të mesazheve që janë procesuar në çdo nyje LONWORKS.

Adresimi Lontalk

Rrjetet LONWORKS janë të strukturuara në mënyrë hierarkike, duke përdorur adresim logjik i cili përfshinë fushat, nënrrjetat dhe paisjet individuale (të ashtuquajtura nyje) [14].

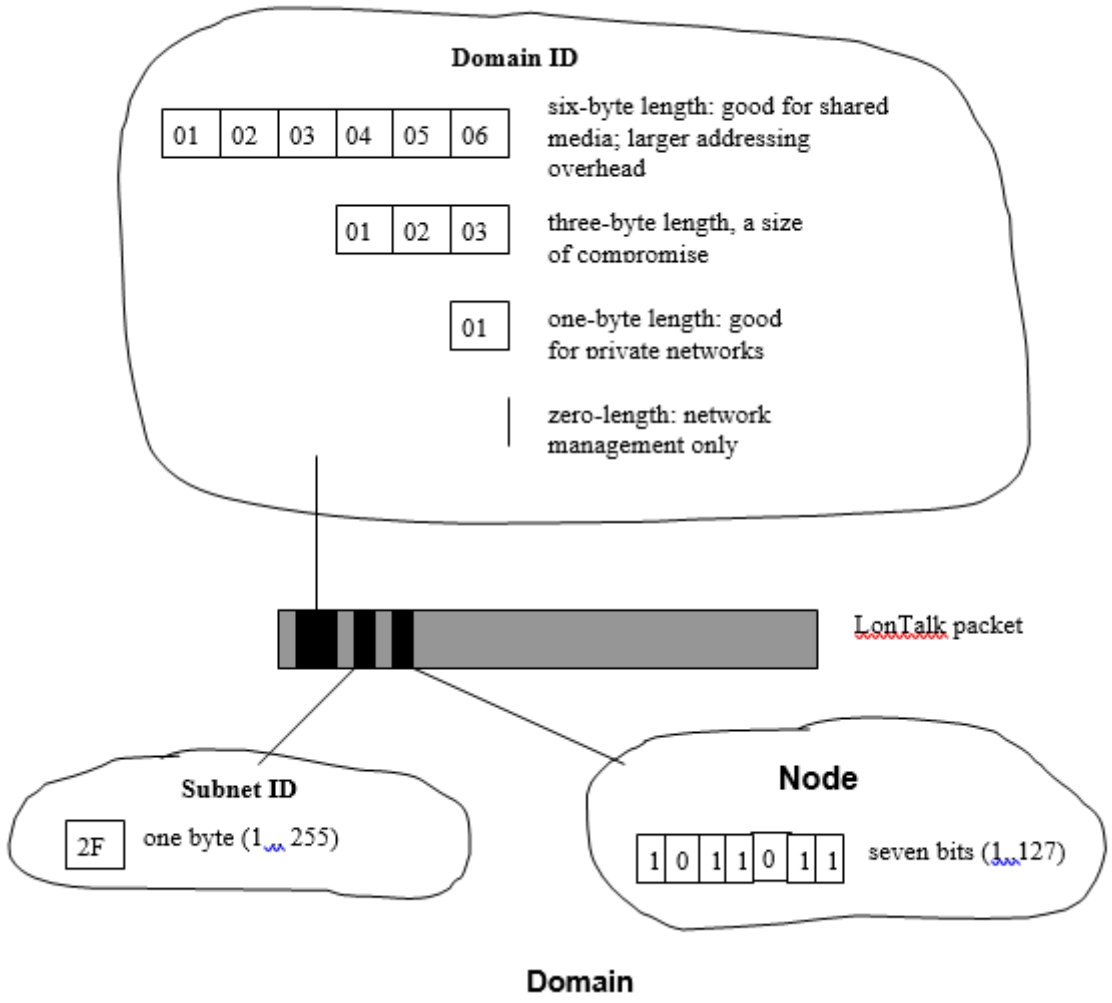

Figura 16 - Adresimi i Domain/Subnet/Node

Domain(Fushat)

Një fushë LonWorks është një bashkësi paisjesh që mund të komunikojnë me njëri tjetrin. Një fushë Lonworks ka një identifikues prej 0,1,3 ose 6 byte të gjatë dhe është zakonisht i dhënë në HEX - sistem hexadecimal. Shembuj:

01 paraqet një fushë me indetifikues 1-byte të vlerës 0x01;
010203 paraqet një fushë me indetifikues 3-byte të vlerës 0x010203;
0c03cb2e010a paraqet një fushë me identifikues 6-byte të vlerës 0x0c03cb2e010a;
Fusha me identifikues zero-byte është kryesisht e referuar si fushë me gjatësi zero, e cila duhet të rezervohet për qëllime të menaxhimit të rrjetit.
Që nga momenti kur identifikuesi i fushës është pjesë e çdo pakete në rrjet, gjatësia e ID-së së fushës ka një ndikim në performansën e rrjetit. Një ID fushe 6-byte kërkon 40bit më shumë për të transmetuar se sa ID-të 1-byte për këtë arsye, një rrjet privat do të mund të përdorë zakonisht identifikues fushe 1-byte.
Rrjetet e mediumit të hapur siç janë ato që përdorin linjën e energjisë ose kanalet e radio frekuencave duhet të përdorin identifikues fushe më të gjatë edhe nëse performansa e kanalit është e kufizuar nga kufizimet e transmetues marrësit. Kjo për të siguruar që një paisje që nuk është pjesë e të njëjtit rrjet nuk mundet kurrë në mënyrë aksidentale të marrë një paketë.

Subnet ID – Nënrrjeti ID

Çdo fushë LonWorks mund të akomodojë deri në 255 nënrrjete. Një nënrrjet (mundësisht boshe) është një grup paisjesh kryesisht të ndërlidhura me një kanal LonWorks. Çdo nënfushë ka një identifikues numerik 1 deri 255, të cilin një mjet menaxhimi rrjeti ia cakton një paisje LonWorks si pjesë e procesit të përdorimit. Që nga momenti që një nënrrjet është i lidhur me një kanal, ID-ja e nënrrjetit në mënyrë automatike mbart informacionin për vendndodhjen e paisjes. Një mënyrë adresimi duke përdorur nënrrjetin ID si një pjesë të adresës destinacion lejon për routerat LonWorks të kufizojnë ekspozimin e paketave në rrjet. Paketat do të mund të ekspozohen në kanale vetëm siç kërkohen: kanali (nënrrjeta) nga e cila vjen paketa, kanali (nënrrjeta) tek e cila është targetuar dhe të gjitha kanalet midis burimit dhe destinacionit.

ID e Nyjes

Çdo nënrrjet LonTalk mund të strehojë deri në 127 pajisje, ku secila nga këto pajisje do të ketë një identifikues numerik nyje në vargun nga 1 në 127. Kjo ID është unike brenda nënrrjetit të pajisjes. I vetmi qëllim i ID së nyjes është të identifikojë në mënyrë unike një pajisje në një nënrrjet të dhënë. ID e nyjes në vetvete nuk ka asnjë kuptim ose ndikim në routera.

Adresimi ID Nyje/Nënrrjet

ID e nyjes përdoret vetëm së bashku me ID e nënrrjetit. Mënyra e afërt e adresimit LonTalk është quajtur adresimi ID Nënrrjet/Nyje (shkurt: adresimi S/N). ID e fushës është gjithmonë e transmetuar si një pjesë e çdo pakete LonTalk.

Mënyra e adresimit LonTalk

Kur një pakete LonTalk nevojitet të udhëtojë nga një burim tek një ose më shumë nyje destinacion, paketa ka nevojë të mbartë informacionin e adresimit për nyjen destinacion ashtu siç edhe për nyjen nismëtare (kështu që marrësit mund të përgjigjen në përputhje me rrethanat).

Më herët diskutuam një mekanizëm adresimi që përdor Neuron ID të nyjes nismëtare (adresimi Neuron ID) ose ID Nyje/Nënrrjet të pajisjes përfundimtare (adresimi ID Nënrrjet/Nyje). Disavantazhi i këtyre mënyrave adresimi është që ato janë të përshtatshme vetëm për të adresuar një pajisje të vetme përfundimtare. Kjo është ajo të cilës i referohemi si unicast: një paketë pritet të udhëtojë nga një pajisje në një pajisje tjetër. Për një paketë që të arrijë destinacione të shumëfishta brenda të njëjtës kohë, të cilës zakonisht i referohemi si multicast, mekanizma të ndryshëm adresimi janë të nevojshëm.

Adresimi në grup

Protokolli LonTalk mbështet adresimin në grup, i cili lejon deri në 64 pajisje të jenë pjesë e deri në 127 grupeve në fushën LONWORKS. Shumica e pajisjeve mund të jenë pjesë e vetëm deri në 15 grupeve të ndryshme në të njëjtën kohë. Anëtarësimi i grupit te çdo pajisje LONWORKS mbahet në tabelën e adresës të chipit Neuron, një burim i rrallë që përdoret gjithashtu për qëllime të tjera. Megjithatë, një pajisje do të jetë e aftë të bashkohet në më pak se 15 grupe, subjekt i konfigurimeve të pajisjes dhe rrjetit. Grupet mund të jenë në

numër të palimituar. Kjo është ajo të cilës i referohemi si një grup i hapur dhe është kufizuar te llojet e shërbimit të pakonfirmuara. Me fjalë të tjera, ju mund të përdorni grupe të hapura vetëm nëse pajisja përfundimtare nuk ka nevojë ti përgjigjet dërguesit.

Adresimi Broadcast

Alternative për adresimin në grup është adresimi Broadcast. Një paketë që përdor adresimin broadcast është dërguar tek të gjitha pajisjet në fushë veprimi, ku fusha e veprimit broadcast mund të jetë një nënrrjet, ose e gjithë fusha. Këtyre mekanizmave adresimi i referohemi zakonisht si broadcast fushë dhe broadcast nënrrjeti përkatësisht. Si grupet e hapura, adresimi broadcast nuk lejon përdorimin e llojeve të shërbimit të konfirmuar. Në këtë drejtim, të dyja teknikat janë shumë të ngjashme. Ndryshimi kryesor është se të bëhesh një pjesëtar i grupit kërkon një nga pak hyrjet e tabelës së adresës, ndërkohe që të bëhesh objektiv për një paketë duke përdorur adresimin broadcast nuk kërkon këtë burim të çmuar të pajisjes.

Kur të përdorim adresimin në grup VS. Adresimin broadcast

Së pari, nuk duhet harruar që vetëm adresimi në grup lejon "acknowleged multicast" multicastet e konfirmuara dhe llojet e shërbimit kërkesë përgjigje. Gjithashtu, ju nuk duhet të përdorni asnjëherë adresimin broadcast me shërbime të pakonfirmuar, të përsëritura unacknowledged repeated (UCKD-RPT), por vetëm shërbimet UCKD-RPT të pakonfirmuara/ papërsëritura.

Së dyti, mesazhet e adresuara në grup mund të drejtohen dhe përdoren edhe nëse paisja përfundimtare nuk është në nënrrjet. Paketat broadcast të nënrrjetit mund të drejtohen në nënrrjetin përfundimtare sipas tabelës së forwardimit të routerave (duke pasur parasysh routerat e para-konfiguruar). Routerat gjithashtu mirëmbajnë informacionin e routimit për 127 grupe të mundshme. Mesazhet broadcast të fushës do të mund të shkojnë kudo pavarsisht vërshimit të rrjetit.

4.1 ISO/IEC 14908-4

Ky Standard Ndërkombëtar specifikon transportimin e paketave të Protokollit të

Rrjetit të Kontrollit (CNP - Control Network Protocol) për rrjetet komerciale të kontrollit të zonës lokale mbi rrjetet e Protokollit të Internetit (IP) duke përdorur një tunel mekanizëm ku pako CNP është e enkapsuluar brenda paketave të IP-së. Kjo vlen për të dy CNP nyjet dhe CNP routerat.

Qëllimi i këtij Standardi Ndërkombëtar është të sigurojë ndërveprimin midis pajisjeve të ndryshme CNP që dëshirojnë të përdorin rrjetet IP për të komunikuar duke përdorur protokollin CNP. Trupi kryesor i këtij Standardi Ndërkombëtar është i pavarur nga protokolli CNP që transportohet mbi rrjetin IP [15].

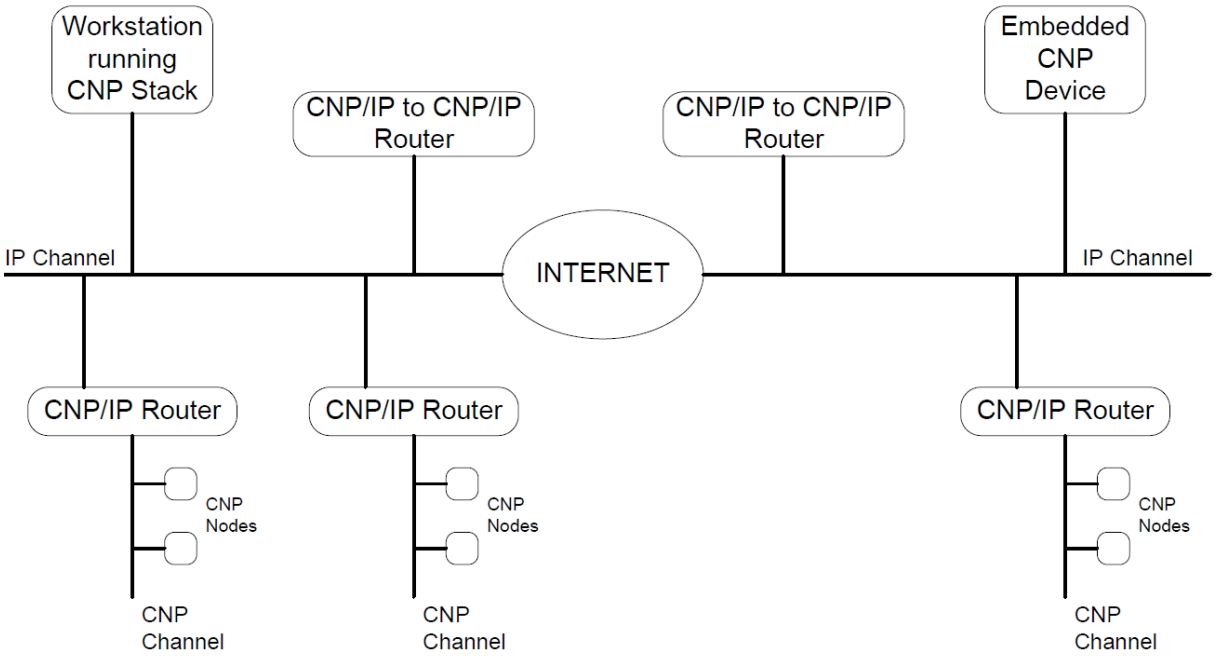

Figura 17 - CNP/IP rrjeti

Figura 17 përshkruan dy lloje të pajisjeve CNP: nyjet CNP dhe routers CNP. Duhet të theksohet se routerët e treguar mund të kanalizojnë pako ndërmjet kanaleve tipike CNP (të tilla si kabllot e çiftëzuar ose rryma elektrike) dhe një kanal IP ose mund të kanalizoj pako CNP midis dy kanaleve IP. Në këtë Standard Ndërkombëtar kanali IP do të përcaktohet në mënyrë të tillë që ta lejojë atë të përdoret si çdo kanal tjetër CNP. Në diagramin e mësipërm, rrjeti IP mund të konsiderohet të jetë një ose më shumë kanale IP. Ky Standardi Ndërkombëtar mbulon vetëm se si pako CNP transportohen mbi kanalet IP, nuk përfshinë se si paketat CNP janë të shpërndara midis kanaleve CNP standarde dhe kanaleve IP, po ashtu ky specifikim nuk synon të mbulojë shtresat më të ulëta (fizike, MAC dhe shtresat e lidhjes) të secilës prej tyre CNP standarde ose kanale IP.

4.2 Paisjet aktuale në treg

Teknologjia LonWork, e zhvilluar nga Echelon, lejon zhvillimin e ndërveprimit të vërtetë të paisjeve dhe sistemeve. Gjithsesi, që nga koha kur teknologjia është e pavarur nga mediumi i komunikimit dhe nuk përshkruan se si programet e aplikimit të paisjes do të strukturohen, thjeshtë duke përdorur teknologjinë Lonworks nuk garanton që paisjet LonWorks nga prodhues të ndryshëm mund të ndërveprojnë në të njëjtin sistem. Pavarësisht, teknologjia LonWorks është e përdorur gjërësisht në sistemet pronësore siç janë: sistemet e kontrollit të automjeteve, sistemet transportuese dhe sistemet e monitorimit të zyrës qendrore të telefonisë. Ka një ndryshim të rëndësishëm midis një grumbulli të paisjeve të ndërveprueshme dhe një sistemi të hapur. Është e rëndësishme të ketë një sistem të hapur pa paisje të ndërveprueshme, por mjaftueshëm e mundshme të ketë një grumbull paisjesh të ndërveprueshme në një sistem të mbyllur. Me fjalë të tjera, paisjet e ndërveprueshme janë të nevojshme, por jo të mjaftueshme për të arritur sistemet e hapura. Dizajni i duhur i rrjetit është kërkesa shtesë për të zbatuar një sistem të hapur të vërtetë. Paisjet e ndërveprueshme janë përbërësit më bazik në zhvillimin e sistemeve të hapura. Megjithatë, shoqëria LONMARK është krijuar dhe mbështetur te ata prodhues që prodhojnë paisje të ndërveprueshme.

Shoqëria LONMARK i ka ndarë paisjet në grupet e më poshtme sipas përdorimit të tyre:

- Qasje-futje-monirotimi;
- Menaxhimit të energjisë;
- Paisje zjarri dhe tymi;
- Gateways;
- Lëvizës(shtyrës) të përgjithshëm;
- Kontrollues të përgjithshëm,
- Interfejs i përgjithshëm njeri-makineri HMI;
- HVAC;
- Hyrje dalje I/O;
- Industriale;
- Ndriçimi;
- Infrastrukturë rrjeti;
- Ftohës dhe sensorë.

4.2.1 Network Interface

Interfacei i rrjetit USB është paisje me kosto të ulët dhe performansë të lartë në rrjetet LONWORKS për paisjet që kanë USB portë (PC dhe kontrolerë të ndryshëm). E dizajnuar për rrjetet e kontrollit LONWORKS për të cilët nevojitet një PC për monitorim, menaxhim dhe diagnostifikim të rrjetit, interfacet e rrjetit USB janë ideal për kontrollë industrial, automatizim të ndërtesave, procesim dhe kontrollë të proceseve të ndryshme. Tiparet e interface-it së rrjetit USB janë të thjeshtë për tu instaluar, drajvera me konfigurim automatik për Microsoft Windows 2000, 2003 dhe XP dhe janë të përputhshëm me të gjitha aplikacionet LNS dhe OpenLDV, ashtu siç edhe me Analizuesin e protokollit LonScanner.

Interfejsi i rrjetit USB U10

Interface i rrjetit USB U10 lidhet me TP/FT-10 (ANSI/CEA-709.3) nëpërmjet topologjisë së lirë LONWORKS, kanalizohet drejtëpërdrejtë përmes një konektori të shkëputshëm, cilësor dhe i përfshirë me paisjen. Interface U10 është totalisht i përputhshëm me kanalet TP/FT-10 me ose pa fuqinë/energjinë lidhëse.

Figura 18 - USB U10 TP/FT-10

Figura 18 tregon interface-in U10. Konektori i rrjetit ndodhet në anën e djathtë të figurës dhe USB plug A ndodhet në anën e majt.

Interface i rrjetit USB U20

Interface i rrjetit USB U20 lidhet me kanalet LONWORKS PL-20 C-Band (ANSI/CEA-709.2) përmes një plug-in konektori me qarkun dhe rrymen të përfshirë në pasije. Interface-i i rrjetit USB U20 mundet gjithashtu të lidhet drejteperdrejtë me sistemet e energjisë 10.8-18VDC (siç është në automjete, kamiona dhe autobus) pa ndërlidhje me qarkun, apo me çdo linjë energjie virtuale përmes një furnizuesi qarku/energjie të furnizuar nga konsumatori, qarku bashkues me linjë energjisë alternative.

Figura 19 - U20 USB Network Interface

Figura 19 tregon interface-in U20. Lidhja e furnizuesit bashkues qark/energji është në anën e djathtë te figurës 20 dhe plug “A” i USB është në të majtë.

LED-et e Interface të rrjetit USB

Secila Interface USB ka tre LED që mund të përdoren për të përcaktuar statusin momental të Interface-it. Këto LED-e futen në punë menjëherë sapo aplikohet energjia në plug “A” USB të Interface-it.

- SVC LED Kur Interface i USB nuk është konfiguruar akoma, SVC LED (LEDi i shërbimit) do të vezullojë në mënyrë pulsuese, me një periudhë prej 2 sekondash dhe 50% cikël pune. Pasi një aplikacion hapet dhe konfiguron Interface-in, SVC LED do të fiket.
- TX LED (LEDi i Transmetimit) ndriçon kur një mesazh dërgohet përmes USB Interface në rrjet. LEDi do të ndriçojë për të paktën 40ms.
- RX LED (LEDi I marrjes) ndriçon ku Interfacei USB merr një mesazh nga rrjeti. Ky LED do të ndriçojë për të paktën 40ms.

4.2.2 Routerat

Routerat janë pajisje infrastrukture që lidhin dy kanale dhe drejtojnë paketat midis tyre. Routerat mund të instalohen për të përdorur një nga katër algoritmet e routimit:

- Përsëritës. Një përsëritës është forma më e thjeshtë e routerit, thjeshtë duke forwarduar të gjitha paketat e vlefshme midis dy kanaleve. Duke përdorur një përsëritës, një nënrrjet mund të ekzistojë përmes segmenteve të shumëfishta të kanalit.
- Routerat që mësojnë. Një router që mëson monitoron trafikun e rrjetit dhe mëson topologjinë e rrjetit në nivelin fushë/nënrrjet. Routeri që mëson pastaj përdor njohuritë e tij për të routuar në mënyrë selektive/përzgjedhëse paketat midis kanaleve. Routeri që mëson nuk mund të mësojë topologjinë grup, kështu që të gjithë paketat që përdorin adresimin në grup forwardohen.
- Routeri i konfiguruar. Është si edhe një router që mëson, edhe një router i konfiguruar ruton/drejton në mënyrë përzgjedhëse paketat midis kanaleve duke konsultuar tabelat e routimit të brendshëm. Ndryshe nga routeri që mëson, përbërja e tabelave të routimit të brendshëm janë të përcaktuar nga një vegël menaxhimi rrjeti. Një vegël menaxhimi rrjeti mund të optimizoj trafikun e rrjetit duke përcaktuar tabelat e routimit për të dyja nënrrjetet dhe routimin e grup adresimit. Routerat e konfiguruar nuk e kryejn algoritmin e mësimit ashtu siç bejnë routerat që mësojnë. Në vend të kësaj, vegla e manaxhimit të rrjetit parakonfiguron tabelat e forwardimit të routerit në momentin e instalimit, bazuar në njohurinë e veglës së topologjisë së rrjetit.

Routerat e konfiguruar dhe routerat që mësojnë janë një group routerash të njohur si routera inteligjent. Të zgjedhësh midis routerave që mësojne dhe të konfiguruar fillimisht, çdo router që mëson vendos tabelat e routimit të brendsëm për të treguar që të gjithë nënrrjetat mund të shtrihen në anën tjetër të routerit. Figura 20 ilustron një rrjet me dy routera që mësojne. Nëse paisja 5 gjeneron një mesazh të adresuar te paisja 2, kanali 2 do të mbaj mesazhin tek routeri që mëson 1 dhe 2. Duke shqyrtuar burimin e fushës së nënrrjetit të mesazhit, routeri që mëson 1 shënon në tabelat e brendshme të routimit që nënrrjeti 2 shtrihet poshtë tij. Routeri pastaj krahason ID e nënrrjetes burim dhe destinacion. Për arsye se ato janë të ndryshme, mesazhi kalohet.

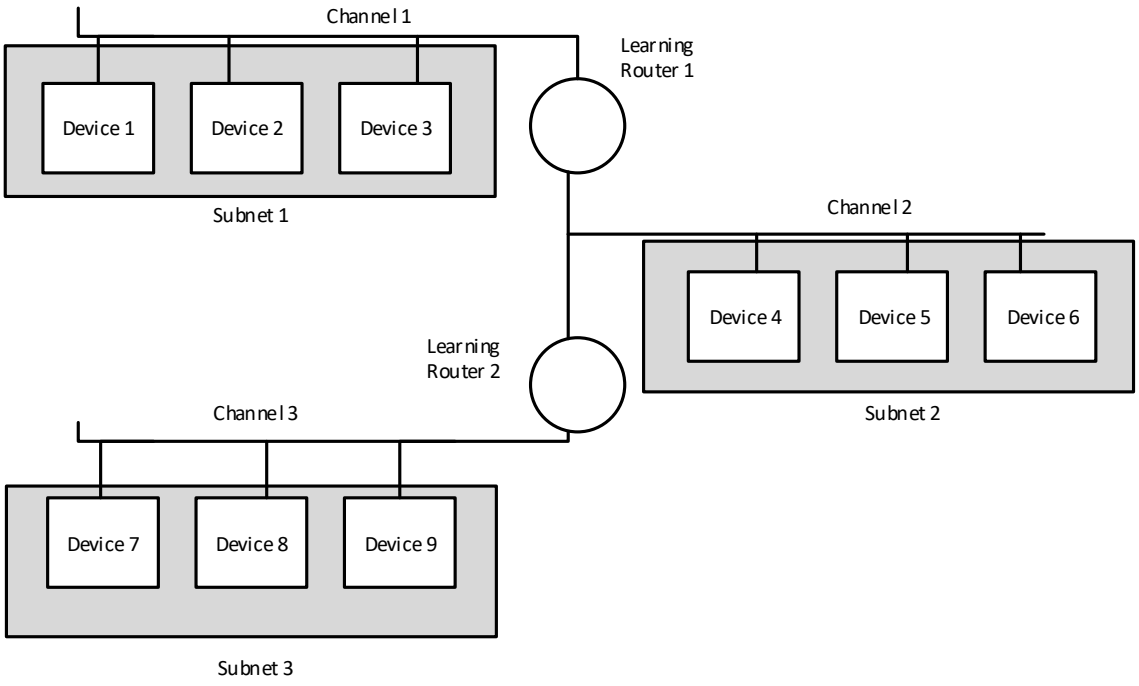

Figura 20 - Rrjeti LON me dy learning Routers

Ndërkohë, routeri që mëson 2 gjithashtu e ka kaluar mesazhin, duke bërë shënim të duhur në tabelat e tij të routimit të brendshëm për sa i përket vendndodhjes. Nëse paisja 2 gjeneron një konfirmim, konfirmimi zgjidhet nga një router që mëson 1, i cili tani shënon vendndodhjen e nënrrjetit 1, routeri që mëson 1 provon tabelat e brendshme të routimit dhe zbulon që nënrrjeti 2 shtrihet më poshtë ndër të, kalon mesazhin. Kur mesazhi shfaqet në nënrrjetin 2, vërehet nga të dyja, paisja 5 (paisja destinacion) dhe routeri që mëson 2, i cili nuk e kalon atë por vetem shënon që nënrrjeti 1, ashtu si nënrrjeti 2, shtrihet diku më sipër. Routeri që mëson 2 nuk do të mësoj për ekzistencën ose vendndodhjen e nënrrjetit 3 deri sa një mesazh të origjinohet (të niset) nga aty.

Routerat e konfiguruar duhet të përdoren kur të jetë e mundur për arsyet e më poshtme:

- rrjedhja parësore e trafikut që ndodh derisa një router që mëson është duke mësuar topologjinë e rrjetit mund të shkaktoj probleme të përplasjes.
- Topologjia e rrjetit mund të ketë një “unazë” të padëshiruar, të cilët janë të zakonshme në linjat e energjisë dhe rrjetet RF, që mund ti shkaktojnë një routeri që mëson një imazh të pasaktë rrjeti.
- Routerat që mësojnë nuk mësojnë rreth grupeve, por routerat e konfigurar mund të

jenë të konfigurar për të forwarduar në mënyrë përzgjedhëse paketat e adresuara në grup.

Nënrrjetat nuk mund të kryqëzojnë routerat inteligjent. Ndërkohë që bridges and repeaters lejojnë nënrrjetat të zgjerojnë kanale të shumëfishta, të dy anët e një routeri inteligjent mund ti përkasin nënrrjetave të ndara. Fakti që routerat inteligjent jane selektues për paketat që forwardojnë në çdo kanal mund të përdoren për të ritur kapacitetin total të një sistemi në kushte të paisjes dhe lidhjes. Në përgjithesi, është gjithmonë një ide e mirë të ndahet trafiku midis “grupe të interesit” nëse është e mundur.

5. Integrimi i LonTalk në IP

Meqë LonWorks është i ndryshëm nga IP protokolli, i njëjti nuk mund të integrohet në mënyrë automatike dhe për atë arsye duhet që të bëhet bashkimi i TCP/IP protokollit dhe LonWorks protokollit për të ndërtuar një rrjet funksional për kontroll.

Implementimi i LonWorks përmes IP rrjetit përfshijnë një kompjuter i cili posedon kartelë rrjeti dhe nga ana tjetër një pajisje e LonWorks adapteri i cili do e luaj rolin e LonWorks përmes IP routerit. Kartela e rrjetit iu mundëson shfrytëzuesit për tu qasur në IP rrjetin, me çka përdoruesit e kanë të mundur edhe qasjes së rrjetit për kontroll LonWorks përmes IP rrjetit, edhe atë aplikacionet të cilat funksionojnë në LonWorks mund të qasen nga çdo kompjuter apo stacion pune i cili ka kyçje në IP rrjet.

Ky implementim mundëson që çdo kompjuter të monitoroj dhe kontrolloj LonWorks rrjetat, qofshin ato LAN, WAN, rrjet korporate apo rrjet global botëror.

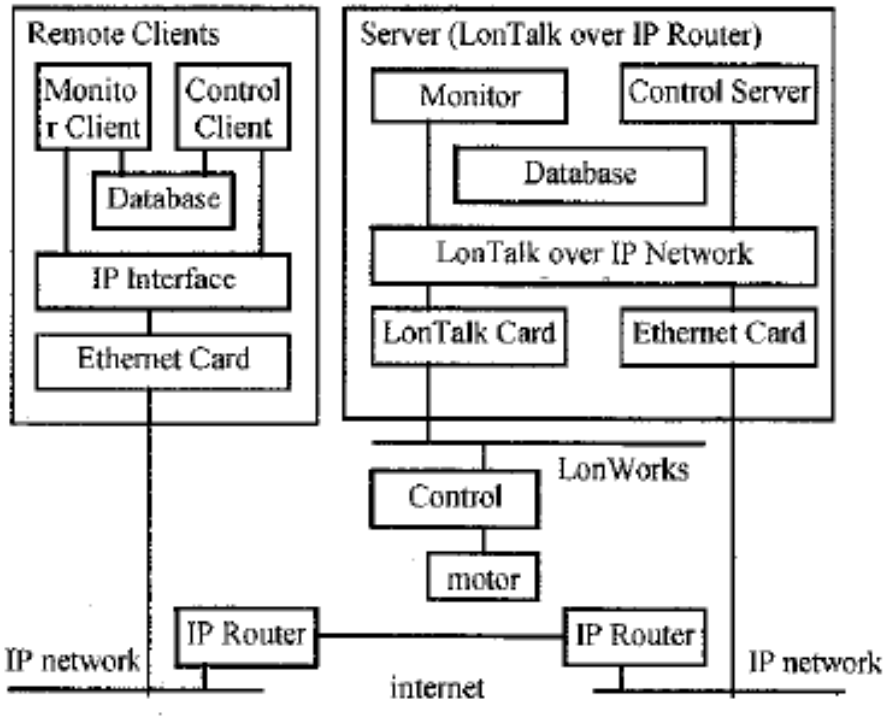

Figura 21 - LONTalk përmes IP

Figura më lartë tregon arkitekturën e LonWorks përmes TCP/IP protokollit. Në modelin tonë klient/server, serveri do të kontrolloj dhe monitoroj LonWorks rrjetin lokalisht, ndërsa klientët do të monitorojn apo kontrollojnë LonWorks rrjetin nga distanca. Serveri pranon LonWorks variablën nga LonTalk Network Interface apo nga LonTalk routeri dhe të njëjtën e dërgon përmes kartelës së rrjetit në IP rrjetin, ndërsa nga ana tjetër klientët e ndryshëm pranojnë dhe dërgojnë komandat e duhura vetëm përmes kartelës së rrjetit.

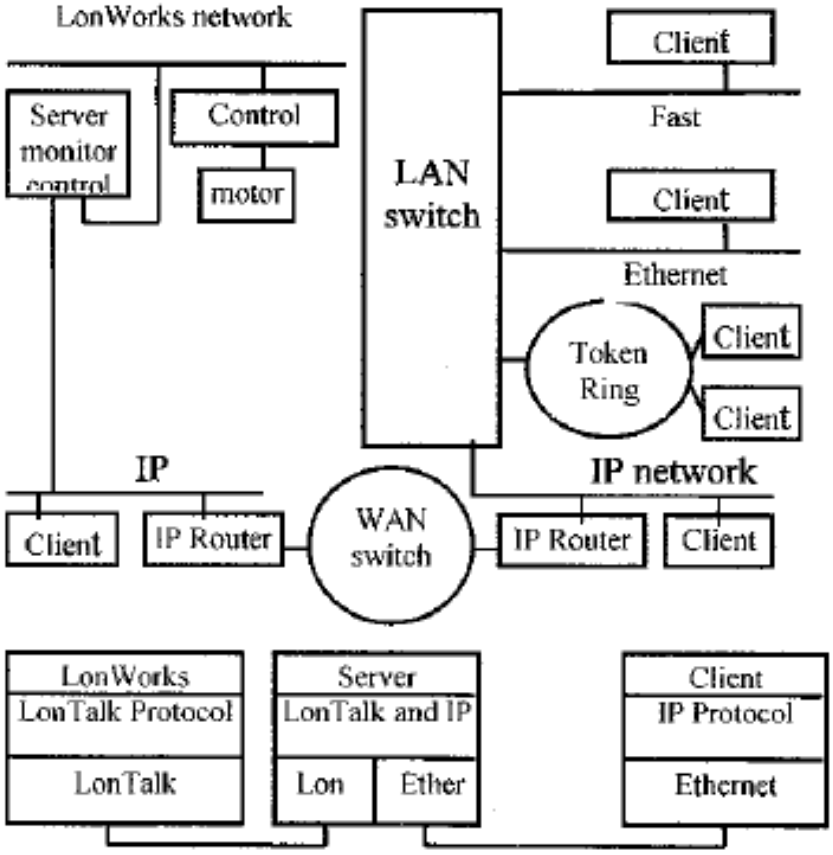

Figura 22 - Linku fizik i LonTalk në IP

Figura më lartë përshkruan linkun fizik dhe protokollet e implementuara në integrimin e LonWorks përmes TCP/IP rrjetit.

5.1 Propozimi i dhënë softuerik

Aplikacionin që do ta zhvillojmë do të jetë Client - Server solucion, ku tipi i mesazheve që do të dërgohet do të jetë e natyrës REQUEST - RESPONSE. Pra Klienti do të dërgoj një REQUEST deri te serveri për monitorim, konfigurim apo kontroll të pajisjeve dhe nga ana tjetër serveri do ti përgjigjet me RESPONSE komandën që do të konfirmoj që komanda është ekzekutuar me sukses, apo ndryshimi i bërë është i suksesshëm.

Komunikimi në nivel të rrjetit do të jetë me anë të Socket-wve të rrjetit, ku në server do të ngritet një sesion Socket- i cili do të jetë në vazhdimësi në pritje të kërkesave nga ana e klientëve dhe nga ana tjetër klientët do ti qasen, me atë që do të jetë e mundur që më tepër klientë të mund njëkohësisht të shfaqin kërkesa ndaj serverit, me çka edhe i tërë ekzekutimi i solucionit do të jetë asinkron, ku poashtu edhe aplikacioni i klientit do të jetë asinkron.

Për sa i përket arkitekturës në nivel softuerik, në anën e serverit do të kemi :

- implementimin e logjikës së kontrollit dhe monitorimit të LonWorks pajisjeve ku si ndërmjetësues do të përdoret Network Interface,
- krijimin e një servisi i cili do të pranoj dhe dërgoj variablat kundrejt klientëve të ndryshëm.

Ndërsa në nivelin e klientit do të kemi:

- implementimi i logjikës së qasjes së serverit për tu ndërlidhur në LonWorks rrjetin lokal;
- ridërgimin e mesazheve që duhet përcjellur nga aplikacionet e ndryshme, qoftë ato për kontroll të derivateve, ndryçimit dhe automatizimit shtëpiak, apo çfarëdolloj mesazh që ka për synim të kontrollit dhe monitorimit të pajisjeve të fundit të cilat janë pjesë e LonWorks rrjetit për kontroll.

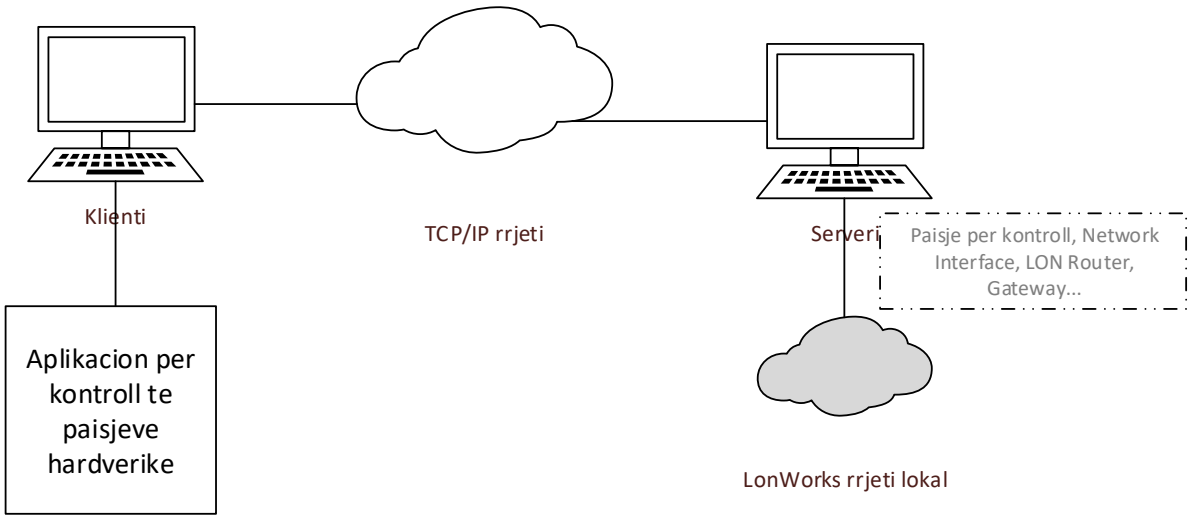

Figura 23 - Propozimi i dhënë softuerik për integrimin e LonTalk në IP përmes middleware komponentës

5.2 Gjuhët programore

Për sa i përket programimit dhe shkruarjes së aplikacioneve për nevoja të këtij punimi do të përdorim platformën Net të Windows duke përdorur Multithread dhe thirrjen e kodit të pamenaxhueshëm për komunikim me LonWorks pajisjet. Ndërsa vetë Nyjet kanë softuer të impelentuar në Neuron C gjuhën programuese. Neuron C është gjuhë programuese e dizajnuar për Neuron Chips dhe është e bazuar në ANSI C. Ajo përfshinë zgjerim të ANSI C dhe në mënyrë të drejtpërdrejtë përkrah Neuron Chip firmware. Neuron C është gjuhë e fuqishme për zhvillim të LonWorks, aplikacioneve të nyjeve dhe në përgjithsi çdo pajisje që përkrah LonWorks ka në procesorët e tyre të implementuar të gjithë algoritmin e kontrollit të pjesës harduerike, qofshin ato motor elektrik, senzorë të ndyshëm, etj.

Threadi është një rrjedhë e pavarur e instrukcioneve në një program. Çdo program i shkruar është një paradigmë e programimit sekuencial në të cilën ka fillim, mbarim, një sekuencë dhe një pikë të vetme të ekzekutimit. Threadi është i ngjajshëm me programin sekuencial. Sidoqoftë, threadi nuk është program në vete, ai nuk mund të ekzekutohet në vete, por ekzekutohet brenda programit. Mekanizmi ku më tepër threada ekzekutohen në të njëjtin program njihet si Multithreading [16].

Meqë platforma e komunikimit me pajisjet kërkon dërgimin dhe pranimin e mesazheve, si dhe procesimin e tyre në mënyrë simultante, në implementimin e LonWorks duhet të kemi më së paku dy threada edhe atë një për pranim të mesazheve dhe një për dërgim të mesazheve. Pra e njëjta gjë vlen edhe për pjesën e kodit të shkruar për aplikacion e serverit si dhe për pjesën e shkruar për aplikacionin e klientit.

Pjesa e komunikimit me pajisjet ndërmjetësuese (LonWorks Network Interface) për ritransmetim të variablave do të bëhet duke përdorur LonWorks API i shkruar në kod të pamenaxhueshëm. E nëjta është e shkruar në gjuhën C në kod të pamenaxhueshëm dhe thirrja e drejtpërdrejtë e disa metodave do të bëhet duke përdorur Platform Invoke. Shkrimi i Platform Invoke deklaratave për API të thjeshta është i drejtpërdrejtë. CLR do të kujdeset për ngarkimin e DLL dhe të gjitha parameter marshaling.

Libraria kryesore e cila komunikon me drajverin e instaluar për LonWorks Interface është ldv32.lib . E njëjta është e shkruar në gjuhën C dhe çdo funksion në këtë librari përdorin extern “C” dhe “stdcall calling conventions”.

Funksionet të cilat do ti përdorim nga kjo libari janë:

ldv_close() – bën mbylljen e sesionit të hapur;

ldv_open() – hap network interface paisjen;

ldv_read() – bën leximin e mesazheve nga sesioni i hapur;

ldv_write() – shkruan mesazh në sesionin e hapur;

ldv_register_event() – regjistron objekt në Windows Event për të pranuar notifikime rreth mesazheve të reja.

I gjithë ky modul do të shkruhet për futjen e varibalave për monitorim apo komandim të pajisjeve në LonWorks rrjetin.

Thelbi i sistemit të krijuar është LAN aplikacion për shkëmbim të mesazheve në TCP/IP socket teknologjinë e programimit të shkruar në C#. Aplikacioni është multithread aplikacion rrjeti dhe funksionon në mënyrën pa blokime.

Socket programming është thelbi i programimit të rrjetit në Windows dhe Linux dhe sot platforma. NET e zbaton atë në një mënyrë të fuqishme. Në këtë punim, ne do të shohim bazat e programimit socket në C #. Për të qenë i saktë, kam krijuar një klient komandë dhe një server komandues, për të komunikuar në mes të një serveri të largët dhe klientit për të dërguar komandat e specifikuara tek ata.

Në programimin e rrjetit me bazë socket, ju nuk keni qasje direkt në kartelën e rrjetit për të dërguar dhe marrë paketa. Në vend të kësaj, krijohet një lidhje ndërmjetëse për të trajtuar ndërfaqen e programimit në rrjet. Supozoni se një socet është një lidhje që lidh aplikacionin tuaj me një ndërfaqe rrjeti të kompjuterit tuaj. Për dërgimin dhe marrjen e të dhënave nga rrjeti duhet të thirrni metodat e socket-it.

Hapësira e sistemit 'System.Net.Sockets' përmban klasa që ofrojnë ndërfaqen aktuale të NET-it në API-të e nivelit të ulët të Winsock. Në programimin e rrjetit, pavarësisht prej të

cilave gjuhë programimi për të përdorur, ekzistojnë disa koncepte të përbashkëta si adresa IP dhe port. IP adresa është një identifikues unik i një kompjuteri në një rrjet dhe porti është si një portë përmes së cilës aplikimet komunikojnë me njëri-tjetrin. Shkurtimisht, kur duam të komunikojmë me një kompjuter të largët ose një pajisje mbi rrjetin, duhet ta dimë adresën e saj IP. Pastaj, ne duhet të hapim një port (Port) në atë IP dhe pastaj dërgojmë dhe marrim të dhënat e kërkuara [17].

Bota e lidhjes IP sillet rreth dy llojeve të komunikimit: me lidhje të orientuara dhe pa lidhje. Në një socket të orientuar në lidhje, protokolli TCP përdoret për të krijuar një sesion (lidhje) midis dy pikave të adresave IP. Ekziston një sasi mjaft e madhe e sipërme që përfshihet në krijimin e lidhjes, por sapo të krijohet, të dhënat mund të transferohen në mënyrë të besueshme mes pajisjeve.

Socket pa lidhje përdorin protokollin UDP. Për shkak të kësaj, nuk kërkohet asnjë lidhje informacioni mes pajisjeve të rrjetit dhe shpesh është e vështirë të përcaktohet se cila pajisje vepron si "server" dhe që vepron si "klient". Ne do të përqendrohemi në llojin e parë të programimit të socket në këtë punim.

Krijimi i connection-orineted sockets

Në NET Framework, ju mund të krijoni komunikime të orientuara drejtpërdrejtë me hostët e largët në një rrjet. Për të krijuar një socket të orientuar në lidhje, sekuenca të veçanta të funksioneve duhet të përdoren për programet e serverit dhe programet e klientëve:

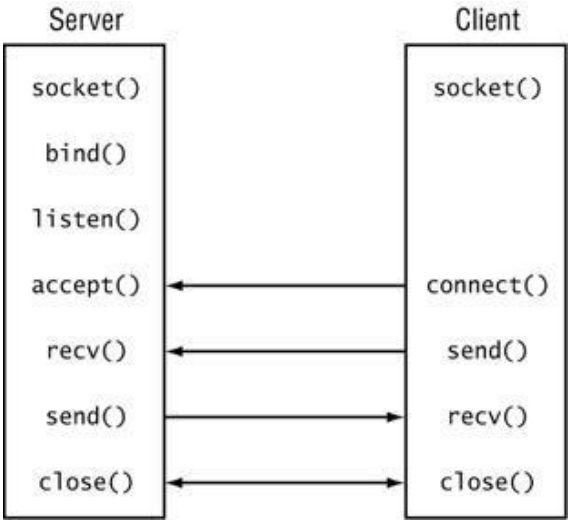

Figura 24 - Socket Client Server komunikimi

SERVERI

Ka katër detyra për tu kryer përpara se një server të mund të transferojë të dhëna me një lidhje të klientit:

- Krijimi i një socke;.
- Bind socket në një IPEndPoint lokale;
- Vendoseni socket-in në regjimin e dëgjimit;
- Pranoni një lidhje hyrëse në socket.

Hapi i parë për të ndërtuar një server TCP është krijimi i një instance të objektit Socket. Tre funksionet e tjera të nevojshme për operacionet e suksesshme të serverëve arrihen pastaj duke përdorur metodat e objektit Socket. Kodi i mëposhtëm C # përmban këto hapa [18-19]:

```
IPEndPoint localEndPoint = new IPEndPoint(IPAddress.Any, 8000);
Socket newsock = Socket(AddressFamily.InterNetwork ;
SocketType.Stream, ProtocolType.Tcp);
newsock.Bind(localEndPoint);
newsock.Listen(10);
Socket client = newsock.Accept().
```

Objekti Socket i krijuar nga metoda Accept () tani mund të përdoret për të transmetuar të dhëna në drejtim në mes të serverit dhe klientit të largët.

KLIENTI

Tani që ju keni një server TCP që punoni, mund të krijoni një program klient të thjeshtë TCP për të ndërvepruar me të. Ka dy hapa të nevojshme për të lidhur një program klient me një server TCP:

- Krijo një socket,
- Lidheni socketin në adresën e serverit të largët.

Siç ishte për programin e serverit, hapi i parë për krijimin e programit të klientit është krijimi i një objekti Socket. Socket objekt është përdorur nga socket Connect () metodë për të lidhur socket me një host të largët:

```
IPEndPoint ipep = new IPEndPoint(Ipaddress.Parse("127.0.0.1"), 8000);
Socket server = new Socket(AddressFamily.InterNetwor;
```

SocketType.Stream, ProtocolType.Tcp);
server.Connect(ipep).

Ky shembull përpiqet të lidhë socket-in me serverin e vendosur në adresën 127.0.0.1. Kjo është adresa IP e hostit lokal (kompjuteri aktual) dhe është një IP loopback për testimin e një aplikacioni rrjeti pa një rrjet. Sigurisht, gjithashtu mund të përdorni hostname së bashku me metodën Dns.Resolve () në një rrjet të vërtetë. (Dns është në hapësirën e System.Net). Pasi programi i serverit të largët TCP pranon kërkesën e lidhjes, programi i klientit është i gatshëm të transmetojë të dhëna me serverin duke përdorur metodat standarde Send () dhe Receive ().

6. IoT dhe IIoT

6.1 Internet of Things

Interneti i gjërave (IoT) definohet si një paradigmë në të cilën objektet e paisura me sensorë, **actuators** dhe procesorë komunikojnë me njëri-tjetrin për ti shërbyer një qëllimi kuptimplotë.

Sot Interneti është bërë i kudo gjendur, ka prekur pothuajse çdo cep të globit dhe është duke ndikuar jetën njerëzore në mënyra të paimagjinueshme. Gjithsesi rrugëtimi është larg mbarimit. Ne jemi duke u futur në një periudhë më të përhapur ndërlidhshmërie ku një lloj shumë i gjërë i paisjesh do të lidhen me Web-in. Ne po futemi në periudhen e "internetit të gjërave". Ky definim është përcaktuar nga autorë të ndryshëm në mënyra të ndryshme. Le të shikojmë dy nga definimet më popullore. Vermesan [20] përcakton internetin e gjërave thjesht si një ndërveprim midis botës fizike dhe digjitale. Bota digjitale ndërvepron me botën fizike duke përdorur me sasi të madhe të sensorëve dhe **actuators.** Një tjetër definim Pena-Lopez [21] definon internetin e gjërave si një paradigmë në të cilën aftësitë e procesimit dhe ndërlidhjes janë ngulitur në çdo lloj objekti të përfytyrueshëm. Ne i përdorim këto aftësi pët të pyetur rreth gjendjes së objektit dhe për të ndryshuar gjendjen e tij nëse është e mundshme. Në gjuhën e përditshme, interneti i gjërave i referohet një lloji të ri të botës ku pothuajse të paisjet që përdorim janë të lidhura në nji rrjet. Në mund ti përdorim ato në mënyrë bashkëpunuese për të përmbushur detyra komplekse që kërkojnë një shkallë të lartë inteligjence.

Për këtë inteligjence dhe ndërlidhje paisjet IoT janë të paisura me sensorë, actuators, procesorë dhe transmetues-marrës të integruar. IoT nuk është një teknologji teke, por është grumbull i teknologjive të ndryshme që punojnë së bashku në varg.

Sensorët dhe actuators janë paisje që ndihmojnë në ndërveprimin me ambientin fizik. Të dhënat që grumbullohen nga senzorët duhet të ruhen dhe përpunohen në mënyre inteligjente me kusht që të nxjerrin përfundime të dobishme nga ai. Shënimi që definon gjërësisht kushtin senzorë; një telefon mobil ose edhe një furrë me mikrovalë mund të quhet senzorë për aq kohë sa ai siguron të dhëna hyrëse rreth gjendjes së momentit (gjendje e

brendshme + ambienti). Një actuator është një paisje që përdoret për të kryer një ndryshim në ambient siç është kontrolluesi i temperaturës në një kondicionues ajri.

Ruajta dhe përpunimi i të dhënave mund të bëhet në skaj të vet rrjetit ose në një remote server. Nëse ndonjë para përpunim i të dhënave është i mundshëm, atëherë është kryesisht i bëshëm tek sensori ose tek disa paisje të tjera të afërta. Të dhënat e përpunuara dërgohen atëherë tek një remote server. Aftësitë e ruajtjes dhe përpunimit të një objekti IoT janë gjithashtu të kufizuara nga burimet e mundshme, të cilat janë shpesh shumë të sforcuara prej kufizimeve të përmasës energjisë, forcës dhe aftësive kompjuterike. Si rezultat sfida kryesore e kërkimit është të sigurohet që ne marrim llojin e duhur të të dhënave në nivelin e dëshiruar të saktësisë. Përgjatë me sfidat e grumbullimit të të dhënave dhe trajtimin, ka sfida në komunikim gjithashtu. Komunikimi midis paisjeve IoT është kryesisht wireless, sepse janë kryesisht të instaluara në vendndodhje të shpërndara gjeografikisht. Kanalet wireless kanë shpesh shkallë të lartë deformimi dhe janë jo të besueshme. Në këtë situatë kominikimi i besueshëm i të dhënave pa shumë ritransmetime është një problem i rëndësishëm ashtu dhe teknologjitë e kominikimit janë tërësisht pjesë e studimit të paisjeve IoT.

Tani, pas përpunimit të të dhënave të marra, disa verprime duhet të merren në bazë të ndërhyrjeve që burojnë. Natyra e veprimeve mund të jetë e ndryshme. Ne mund të modifikojmë në mënyrë të drejtëpërdrejtë botën fizike përmes **actuators**. Ose mund të bëjmë diçka virtualisht. Për shembull, ne mund të dërgojmë disa informacione tek gjërat tjera të mençura.

Procesi i ndikimit të ndryshimit në botën fizike është shpëshherë i varur në gjendjen e tijë në atë moment kohorë. Kjo quhet ndërgjegjësim konteksti. Çdo veprim është i marrë duke pas konsideratë kontekstin, sepse një aplicaion mund sillet në mënyrë të ndryshme në kontekste të ndryshme. Për shembull; një person mund të mos i pëlqejë mesazhet që i vijnë nga zyra dhe i ndërpret ato kur është në pushime.

Sensorët, actuators, serverat kompjuterik dhe rrjeti i kominikimit krijojnë infrastrukturën bërthamë të “IoT framework”. Gjithsesi, ka shumë aspekte softweri që është e nevojshme të konsiderohen. Së pari neve na nevojitet një middleware që mund të përdoret për lidhjen dhe

menaxhimin e të gjithë përbërësve heterogjen. Na nevojitet shumë standardizim për të lidhur shumë paisje të ndryshme.

Interneti i gjërave gjen aplikime të shumta në kujdesin për shëndetin, fitness, edukim, zbavitje, jetë sociale, ruajtje e energjisë, monitorimi i ambientit, automatizimin e shtëpise dhe sistemet e transportit. *Ne duhet të fokusohemi në këto zona të përdorimit në seksionin 9.* Mund të konstatojmë që në të gjitha këto fusha aplikimi, teknologjitë IoT kanë qenë të afta në mënyrë të konsiderueshme të ulin përpjekjen njerzore dhe të përmirësojnë cilësinë e rrjetit.

6.1.1 Arkitektura e IoT

Nuk ka asnjë konsensus të vetëm në arkitekturë për IoT, për të cilën është rënë dakord ne mënyrë universale. Arkitekturat e ndryshme janë proposuar nga kërkues të ndryshëm.

Arkitektura me tre dhe pesë shtresa

Arkitektura më bazike është arkitekture me tre shtresa [22, 23, 24] siç është edhe e treguar në figurën 26. Është prezantuar në fazat e para të kërkimit në këtë fushë.

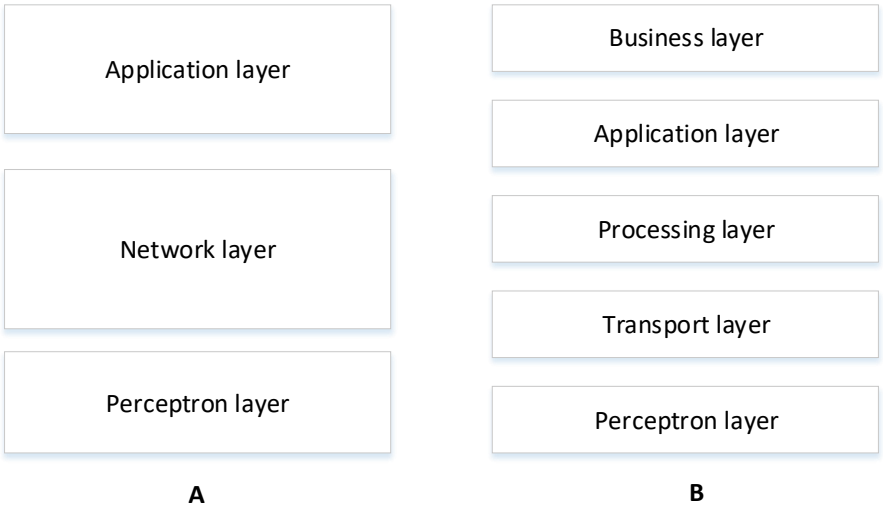

Figura 25 - Arkitektura e IoT (A: tre shtresa) (B: pesë shtresa)

Ka tre shtresa: *perceptimi, rrjeti* dhe *shtresa e aplikacionit*.

Shtresa e perceptimit është shtresa fizike, e cila ka senzorë për të ndjerë dhe grumbulluar informacion rreth ambientit. Ajo ndjen disa parametra fizik ose identifikon objektet e tjera të mençura në ambient.

Shtresa e rrjetit është e përgjegjshme për lidhjen me gjëra të tjera të mençura, pajisje rrjeti, dhe servera. Tiparet e saj janë gjithashtu të përdorur për transmetimin dhe procesimin e të dhënave sensoriale.

Shtresa e aplikacionit është e përgjegjshme për shpërndarjen e shërbimeve specifike të aplikacionit tek përdoruesi. Kjo definon aplikacionet e ndryshme në të cilët interneti i gjërave mund të vendoset, për shembull, shtëpitë e mençura, qytetet e mençura dhe shëndetin e zgjuar.

Arkitekturat e "resw" "cloud and fog"

Në veçanti, kemi qenë paksa të paqartë për natyrën e të dhënave të gjeneruara nga pajisjet IoT, dhe natyra e procesimit të të dhënave. Në disa arkitektura sistemi, procesimi i të dhënave bëhet në mënyrë të centralizuar nga kompiuterat RE. Një arkitekturë RE qendrore e tillë e mbanë renë në qendër, aplikacionet sipër saj dhe rrjetin me gjërat e meçura nën të [25]. Cloud computing i është dhënë përparësi sepse siguron fleksibilitet dhe përshkallëzueshmëri te madhe. Ofron shërbime si infrastruktura bërthame, platformë, program dhe ruajtje. Zhvilluesit mund të sigurojnë mjetet e tyre të ruajtjes, mjete programi, nxjerrjen e të dhënave, mjetet e mësimit të aparaturave dhe mjete vizualizimi përmes resë.

Së fundmi, kanë një lëvizje drejt një arkitekture tjetër sistemi, d.m.th. "fog computing" [26, 27, 28], ku senzorët dhe gateway-it e rrjetit bëjnë një pjesë të procesimit dhe analizim të të dhënave. Një "fog" arkitekture [29] paraqet një trajtim të shtresuar siç tregohet në figurë, i cili fut monitorimin, parapërpunimin, ruajtjen dhe shtresat e sigurisë midis shtresës fizike dhe transportit. Shtresa e monitorimit monitoron fuqinë, burimet, përgjigjet dhe shërbimet. Shtresa e parapërpunimit performon filtrimin, përpunimin dhe analizën e të dhënave sensoriale. Shtresa e ruajtjes së përkohshme siguron funksionalitete të ruajtjes siç janë: replikimi i të dhënave, shpërndarja dhe ruajtja e të dhënave. Së fundmi shtresa e sigurisë performon kodimi/dekodimi dhe siguron integritet të të dhënave dhe privatësi. Monitorimi

dhe parapërpunimi bëhen në skaj të rrjetit përpara dërgimit të të dhënave në cloud.

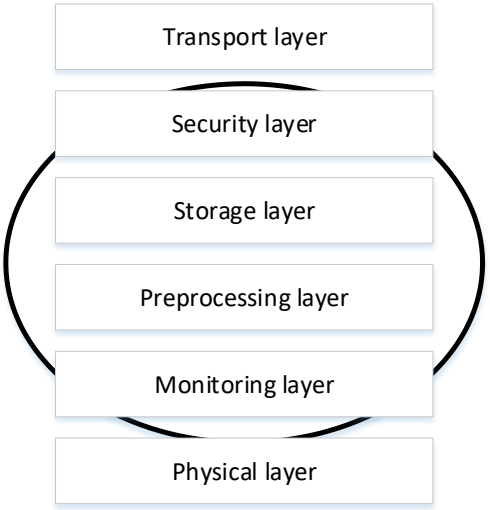

Figura 26 - Arkitektura FOG e një gateway intelegjent

Shpeshherë termat “fog computing” dhe “edge computing” këmbehen. Termi më i fundit është më i vjetër se sa i mëparshmi dhe interpretohet të jetë më i përgjithshëm. “fog computing” fillimisht i emruar nga Cisco i referohet gateway-save dhe sensorëve të mençur, kurse “edge computing” është paksa më deportues në natyrë. Kjo paradigmë parashikon shtimin e aftësive të parapërpunimit të të dhënave tek paisjet fizike, siç janë: motorët, pompat dhe dritat. Qëllimi është të bëhet sa më shumë të jetë e mundur nga parapërpunimi i të dhënave në këto paisje, të cilat janë quajtur të jenë në skaj të çdo rrjeti. Në kushtet e arkitekturës së sistemit, diagrami arkitekturor nuk është dukshëm i ndryshëm nga figura. Si rezultat, ne nuk e përshkruajmë “edge computing” veçmas.

Si përfundim dallimi midis artiketurës protokoll dhe arkitekturës së sistemit nuk është shumë i prerë. Shpeshherë protokollet dhe sistemi janë të koduara. Ne duhet të përdorim pirgun e protokollit të përgjitshëm 5 shtresorë (diagrama arkitekturore prezantohet në figurë) për arkitekturat “fog” dhe “cloud”.

Social IoT

Le të diskutojmë tani një paradigmë të re: Social IoT (SIoT). Këtu ne konsiderojmë marrëdhëniet sociale midis dy objekteve njësoj siç njerëzit formojnë marrëdhënie social [30]. Këtu janë tre aspektet kryesore të një sistemi SIoT:

SIoT është e drejtueshme. Ne mund të fillojmë me një paisje dhe të drejtojmë

përmes gjitha paisjeve që janë të lidhura në të. Është e thjeshtë të zbulohen paisje dhe shërbime të reja që përdorin një rrjet të tillë social të paisjeve IoT.

Një nevojë për besueshmëri (forca e marrëdhënies) është prezente midis paisjeve (e ngjashme me “friends-at në facebook).

Ne mund të përdorim modele të ngjashme të studimit të rrjetit shoqëror njerzorë për të studiuar poashtu rrjetin shoqëror të paisjeve IoT.

Komunikimi

Ndërkohë që interneti i gjërave po ritet në mënyrë shumë të shpejtë gjendet një numër i madh i paisjeve të mençura heterogjene që lidhen me internetin. Paisjet IoT janë të furnizuara me bateri, me llogaritshmëri minimale dhe burime për ruajtje. Për shkak të natyrës së tyre të sforcuara, janë të përfshira shumë sfida kominikimi, të cilat janë [31]:

1. Adresimi dhe identifikimi, meqenë se miliona gjëra do të lidhen me internetin, do të duhet që ato të identifikohen përmes një adrese unike, në bazë të cilës ato do të komunikojnë me njëri-tjetrin. Për këtë na duhet një hapsirë e madhe adresimi dhe një adresë unike për çdo objekt të mençur.

2. Kominikimi me fuqi të ulët - kominikimi i të dhënave midis paisjeve është një detyrë që konsumon energji, veçanërisht kominikimi wireless. Megjithatë na nevojitet një zgjidhje që e lehtëson kominikimin me konsumim të ulët energjie.

3. Protokollet e routimit me kërkesë memorje të ulët dhe modele kominikimi efikas.

4. Komunikimi me shpejtësi të lartë dhe pa humbje.

5. Lëvizshmëria e gjërave të mençura.

Paisjet IoT lidhen kryesisht përmes Internet protokollit. Ky pirg është shumë kompleks dhe kërkon një sasi të madhe energjie dhe memorje nga paisjet që lidhen. Paisjet IoT mundet gjithashtu të lidhen në mënyrë lokale përmes rrjeteve jo IP, të cilat konsumojnë më pak energji dhe lidhen me interntin përmes një gateway të mençur. Kanalet e komunikimit jo IP siç janë: **BT, RFID, NCF** janë mjaft të njohura, por janë të limituara në rrethin e tyre (deri në disa metra). Prandaj, aplikacionet e tyre janë të kufizuara në rrjete personale të vogla “PAN”, PAN-sat janë duke u përdorur gjerësisht në aplikacionet IoT si paisjet e veshura të lidhura me telefonët e mençur. Për të ritur fushën e llojeve të tilla të rrjetit lokal, ishte e nevojshme të modifikohej pirgu IP për të lehtësuar kominikimin me fuqi të ulët që përdor pirgun IP. Një nga

zgjidhjet është 6LoWPAN, e cila përmban IPv6 me rrjete personale me energji të ulët. Fusha e PAN me 6LoWPAN është e ngjashme me rrjetet me zonë lokale dhe konsumi i tyre i energjisë është shumë i vogël.

Teknologjitë e kominikimit që kryesojnë të përdorura në botën IoT janë IEEE 802.15.4, WiFi me energji të ulët, 6LoWPAN, RFID, NFC, Sigfox, LoraWAN dhe protokolle pronësorë të tjera për rrjetet Wireless.

6.2 Industrial Internet of Things

IIoT ose Industrial Internet of Things - Është grumbullimi i sensorëve, komunikimit, ruajtjes së të dhënave dhe analitikës në një mjedis industrial. Përdoret në disa industri të tilla si prodhimi, logjistika, nafta dhe gazi, transporti, energjia - shërbimet, minierat dhe metalet, aviacioni dhe sektorë të tjerë industrialë.

Shpeshherë është diskutuar në lidhje me Industrinë 4.0 dhe Digitalizimin e Fabrikës. Ndonjëherë, termat janë përdorur edhe në mënyrë të ndërsjellë edhe pse kjo nuk është 100% e saktë. Digjitalizimi është një element i IIoT-së duke lëvizur nga metodat manuale në ato dixhitale. Një shembull i kësaj është lëvizja nga kontrollet e butonave në Human Machine Interface. IIoT është një krah i Industrisë 4.0, përparësitë e tjera që përfshin industria 4.0, simulimi, binjakëzimi digjital, prodhimi shtesë, të dhënat e mëdha, realiteti i shtuar, cloud-i, robotët autonome dhe natyrisht siguria kibernetike janë të gjitha të lidhura me IIoT. Industria 4.0 nganjëherë quhet Revolucioni i katërt industrial. Është një valë përparimesh që së bashku po ndryshojnë mënyrën se si funksionon industria. Revolucioni i parë industrial është epoka në të cilën ne mendojmë në fillim të viteve 1800, të cilat prezantuan makinat. I dyti u zhvillua në fillim të viteve 1900 dhe paraqiti energji elektrike dhe linjën e montimit. I treti filloi me futjen e PLC duke futur automatizimin dhe IT. Ndërsa tre revolucionet e para ishin të shpërndarë afërsisht një shekull larg, ky revolucion i katërt po vjen rreth 20 vjet më vonë praktikisht në këmbë të fundit! Disa akademikë konsiderojnë valët e tretë dhe të katërt si pjesë e të njëjtit prirje të përgjithshme.

"Industrie 4.0" kombinon metodat e prodhimit me teknologjinë më të fundit të informacionit dhe komunikimit. Forca lëvizëse prapa këtij zhvillimi është digjitalizimi me

shpejtësi në rritje i ekonomisë dhe shoqërisë. Ajo po ndryshon të ardhmen e prodhimit dhe punës në Gjermani: Në traditën e motorit me avull, linja prodhimi, elektronikë dhe IT, fabrikat e zgjuara po përcaktojnë tani revolucionin e katërt industrial. Baza teknologjike ofrohet nga sisteme inteligjente, të rrjetëzuara në mënyrë digjitale, të cilat do të mundësojnë në masë të madhe proceset e prodhimit të vetë-menaxhimit: Në botën e Industrie 4.0, njerëzit, makineritë, pajisjet, sistemet dhe produktet e logjistikës komunikojnë dhe bashkëpunojnë me njëri-tjetrin drejtpërdrejt. Proceset e prodhimit dhe të logjistikës janë të integruara inteligjente në të gjithë kufijtë e kompanisë për ta bërë prodhimin më efikas dhe fleksibil. Kjo lehtëson zinxhirët e krijimit të vlerave të mençura të cilat përfshijnë të gjitha fazat e ciklit jetësor të produktit - nga ideja fillestare e produktit, zhvillimi, prodhimi, përdorimi dhe mirëmbajtja deri tek riciklimi. Në këtë mënyrë, dëshirat e konsumatorëve për çdo gjë nga ideja e prodhimit deri tek riciklimi mund të merren parasysh, si dhe shërbimet përkatëse. Kjo i mundëson kompanive të prodhojnë produkte që përshtaten sipas kërkesave individuale të konsumatorëve më lehtë se më parë. Prodhimi individual dhe mirëmbajtja e produkteve mund të bëhet normë e re. Në të njëjtën kohë, shpenzimet e prodhimit mund të zvogëlohen pavarësisht nga prodhimi i individualizuar. Rrjetëzimi i kompanive në zinxhirin e furnizimit bën të mundur që të zgjedh jo vetëm hapat individual të prodhimit, por të gjithë zinxhirin e vlerës. Informacioni gjithëpërfshirës në kohë reale u mundëson kompanive të reagojnë në disponueshmërinë e disa lëndëve të para në fillim, për shembull. Proceset e prodhimit mund të kontrollohen në të gjithë kufijtë e kompanisë për të kursyer burimet dhe energjinë.

IIoT shpesh paraqitet si një revolucion që po ndryshon fytyrën e industrisë në një mënyrë të thellë. Në realitet, kjo është një evolucion që ka origjinën në teknologjitë dhe funksionet e zhvilluara nga furnizuesit e automatizimit vizionar më shumë se 15 vjet më parë, ndoshta do të nevoiten edhe 15 vite të tjera për të realizuar dhe përdorur potencialin e plotë të IIoT-së.

7. Krijimi i një sistemi demonstrues virtual

Sistemi demonstrues virtual përbëhet nga dy Windows aplikacione, edhe atë LonClient aplikacionin, dhe LonServer Aplikacionin, të shkurara në gjuhën C# të bazuara në NET framework 4.5.

Aplikacionet e shkruara përdorin mënyrën askinkrone për ekzekutim, që nënkupton që kur klienti dërgon një komandë, serveri e ekzekuton në mënyrë asinkrone dhe ekzekutimi i proceseve nuk pezullohet përderisa ai pret konekcione nga klienti. Në po të njëjtën mënyrë është zhvilluar edhe klient aplikacioni, pra procesimi i tij nuk është i pezulluar për pritje të përgjigjes nga serveri.

Të dy aplikacionet përmbajnë klasën e përbashkët StateObject e cila përdoret për pranimin e të dhënave nga pajisja remote (në distancë).

Kjo klasë përbëhet nga:

```
public class StateObject - Client  socket;
public Socket workSocket = null;
size of receive buffer;
public const int BufferSize = 1024;
receive buffer;
public byte[] buffer = new byte[BufferSize];
received data string;
public StringBuilder sb = new StringBuilder();
```

Socket – që tregon Socketin e klientit;

Buffer Size – madhësia e vargut që do të pranojmë;

buffer – vargu i të dhënave që do të pranojmë;

sb – që është stringbuilder, në të cilën do të grumbullojmë të dhënat e pranuara në formatin string.

Për dallim nga LonClient aplikacioni, LonServer aplikacioni përmban klasën LonWorksModule për manovrim me pajisjet LON për kontroll në rrjet dhe metodat e saj janë të shkruara për komunikim me pjesën e drajverit të pajisjeve të shkruara në kod të

pamenaxhueshëm të ldv32.dll librarisë. Pra kjo klasë është ura lidhëse ndërmjet rrjetit lokal LON dhe TCP/IP rrjetit duke përdorur pajisjen kompjuterike PC të thjeshtë.

```
public class LonWorksModule

    #region LDV32_DLL
    public static short m_OpenLDVhandle = 0;
    [DllImport("ldv32.dll")]
    public static extern short ldv_open(string id, ref short handle);
    [DllImport("ldv32.dll")]
    public static extern short ldv_open_cap(string id, ref short handle, LdvDeviceCaps lvdcaps, string ss, long
tag);
    [DllImport("ldv32.dll")]
    public static extern short ldv_close(short handle);
    [DllImport("ldv32.dll")]
    public static extern short ldv_read(short handle, IntPtr data, short len);
    [DllImport("ldv32.dll")]
    public static extern short ldv_ërite(short handle, byte[] data, short len);
    [DllImport("ldv32.dll")]
    public static extern short ldv_register_event(short handle, IntPtr @event);
    [DllImport("ldv32.dll")]
    public static extern short ldv_get_device_info(string id, IntPtr data);
    #endregion

    #region WIN_EVENTS
    [DllImport("kernel32.dll")]
    public static extern IntPtr CreateEvent(IntPtr lpEventAttributes, bool bManualReset, bool bInitialState,
string lpName);
    [DllImport("kernel32.dll", SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    public static extern bool CloseHandle(IntPtr hObject);
    #endregion

    #region LONEnumerations
    public enum LdvCode
    public enum LdvDeviceCaps
    #endregion
```

Përshkrimi i LonServer aplikaconit

Për qëllimet e kësaj teze të magjistraturës kemi zhvilluar apostafat aplikacionin Windows që do të instalohet në serverin e menduar, i cili përbëhet nga një WindowsForm, në të cilën përzgjedhim LON Interfejsin në të cilin do të ndërlidhemi dhe e cila paraprakisht kërkon që te ajo maqinë pune të kemi të instaluar drajverin për pajisjen apostafate, në shembullin tonë të testuar në U10 USB Network Interface. Drajveri i instaluar në maqinën testuese është 153-0411-01A_OpenLDV400.

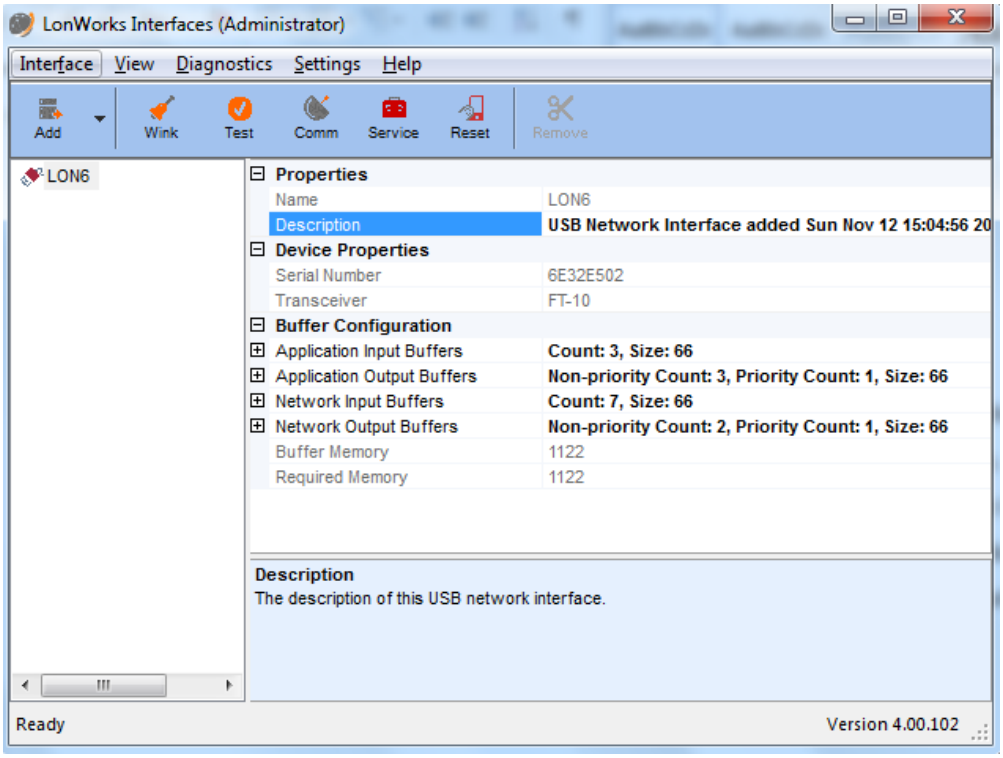

Figura 27 - LonWorks Driver Configurator nga Control Panel-i

Pajisja e përdrorur USB LON Interface U10

Figura 28 - U10 Netëork Interface e konektuar në star rrjet

Apliakcioni LonServer është si vijon:

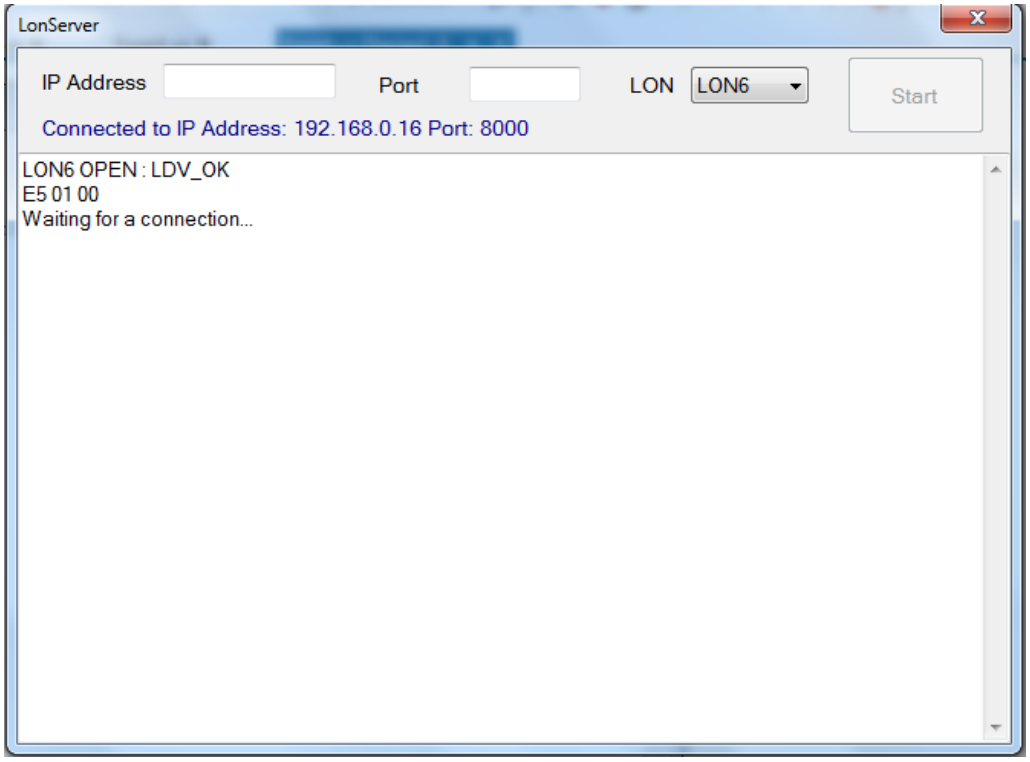

Figura 29 - LonServer GUI i zhvilluar si pjesë e komponentës middleware

IP Address pjesa është e rezervuar për të vendosur informatat e IPAdreses së maqinës në të cilën do të instalojmë aplikacionin, poashtu edhe Portën, për të krijuar Socketin për komunikim me pajisjet e klientëve. Në rast se informatat e njëjta nuk jepen nga ana e shfrytëzuesit, aplikacioni në mënyrë automatike detekton IP adresën e maqinës dhe hap komunikimin përmes asaj IP adrese dhe portës 8000.

LON komboboksi është i menduar që shfrytëzuesi të identifikoj LON pajisjes për të hapur komunikimin me pjesën e rrjetit LON.

Në moment kur shtypim butonin START kemi fillimin e dy proceseve:

1. Procesin e krijimit të SOCKETIT i cili gjithë kohën bën ndëgjimin për arritjen e mesazhve;
2. Procesin e hapjes së LON komunikimit të rrjetit Lokal për kontroll dhe pritjen për ekzekutim të mesazheve që do të arrijnë nga klienti (ana e TCP/IP rrjetit), apo nga pajisjet e ndryshme industriale ose shtëpiake (ana e LON rrjetit).

Çdo mesazh i dërguar apo pranuar në vete përmban "<EOF>" sinjalizuesin, i cili edhe njëherit tregon që mesazhi i transmetuar mbaron në këtë sinjalizues.

Në prapavi të formës janë katër metodat kryesore të cilët mundësojnë hapjen e komunikimit dhe dërgimin e përgjigjeve tek klienti, edhe atë:

- private void StartListening() e cila është pjesë e DoWorkEventHandler(StartToListen) BackgroundWorker-it që do të bëj resolvimin e IP adresës IPEndPoint localEndPoint = new IPEndPoint(ipAddress, portNr) dhe hapjen e Socket-it për komunikim Socket listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp)
- private void AcceptCallback(IAsyncResult ar) e cila e sinjalizon threadin kryesorë për komunikim dhe krijon një StateObject të shpjeguar më lartë
- private void ReadCallback(IAsyncResult ar) bën leximin e informatave nga socketi i klientit dhe njëherit dërgon përgjigjen e kërkuar përmes Send metodës
- private void Send(Socket handler, String data) dërgon përgjigjen te klienti.
- private void SendCallback(IAsyncResult ar) mbaron komunikimin me klientin për sesionin e nisur.

Pjesët e metodave për komunikim me rrjetin LON përbëhen nga :

- private bool startLon(string ControllerID) përmes të cilës e thirrim ldv_open() metodën të përshkruar më lartë;
- private short writeLDV(string txt) e cila dërgon komandën duke përdorur metodën ldv_write();
- private byte[] readlLDV() bën leximin e informatave nga LON rrjeti për ti përgatitur të njëjtat më pas për ridërgim përmes TCP/IP pakove në rrjetin global deri te klienti;
- në mbyllje të formës kemi thirrje të ldv_close() metodës për mbyllje të komunikimit me rrjetin LON.

Ajo që vlen të përmendet është edhe metoda që kemi krijuar për rekordimin e komunikimit të të dy rrjeve, përmes dy metodave:

- public static void CommLog(string msg) e cila shkruan informatat në formatin e veçantë për komunikimin i cili bëhet me pajisjet e rrjetit për kontroll LON

```
16:25:08.673- *************************
16:25:08.674- LON6 SUCESSFULLY STARTED
16:25:08.681- READ :E5 01 00  -LDV_OK
16:25:38.379- WRITE:50 00  -LDV_OK
16:25:38.480- READ :50 00  -LDV_OK
16:26:11.543- WRITE:22 10 65 00 02 00 00 00 00 00 00 00 00 00 00 6A 00  -LDV_OK
16:26:11.645- READ :16 1E 65 01 10 00 00 00 00 00 00 00 00 00 00 00 2A 01 00 00 00 00 00 02 81 01 FF FF FF FF FF FF  -LDV_OK
16:26:14.407- WRITE:50 00  -LDV_OK
16:26:14.510- READ :50 00  -LDV_OK
16:26:15.541- WRITE:50 00  -LDV_OK
16:26:15.642- READ :50 00  -LDV_OK
```

Figura 30 - Logu i LON komunikimit

- public static void SocketLog(string msg) e cila shkruan informatat e komunikimit të TCP/IP rrjetit dhe poashtu madhësinë e pakove të dërguara të shprehura në bajtë.

```
16:25:08.697- Connected to IP Address: 192.168.0.16 Port: 8000
16:25:38.376- Read 10 bytes from socket:192.168.0.16:8000
16:25:38.481- Sending 14 bytes to socket: 192.168.0.19:49169

16:26:11.542- Read 58 bytes from socket:192.168.0.16:8000
16:26:11.647- Sending 104 bytes to socket: 192.168.0.19:49170

16:26:14.405- Read 10 bytes from socket:192.168.0.16:8000
16:26:14.511- Sending 14 bytes to socket: 192.168.0.19:49172

16:26:15.538- Read 10 bytes from socket:192.168.0.16:8000
16:26:15.643- Sending 14 bytes to socket: 192.168.0.19:49173
```

Figura 31 - Logu i SOCKET komunikimit në TCP/IP

Përshkrimi i LonClient aplikaconit

Ky aplikacion poashtu përbëhet vetëm nga një formë, në të cilinë kemi përmbledhur disa funksione bazike për monitorim dhe konfigurim së vetë Network Interface pajisjes. Ajo poashtu në vete përmban edhe metodat e duhura për krijimin e një Socket lidhjes për komunikim me LonServerin.

Vlen të ceket që LonClient për çdo komandë është programuar të hap një sesion Socket, dhe të ndërpres veprimin, pra nuk është si pjesa e LonServer, e cila bën ndëgjim të vazhdueshëm, sepse në propozimin e dhënë softuerik kemi cekur që është e mundur që më tepër se një klient të mund të komunikoj në kohën e njëjtë me aplikacionin LonServer.

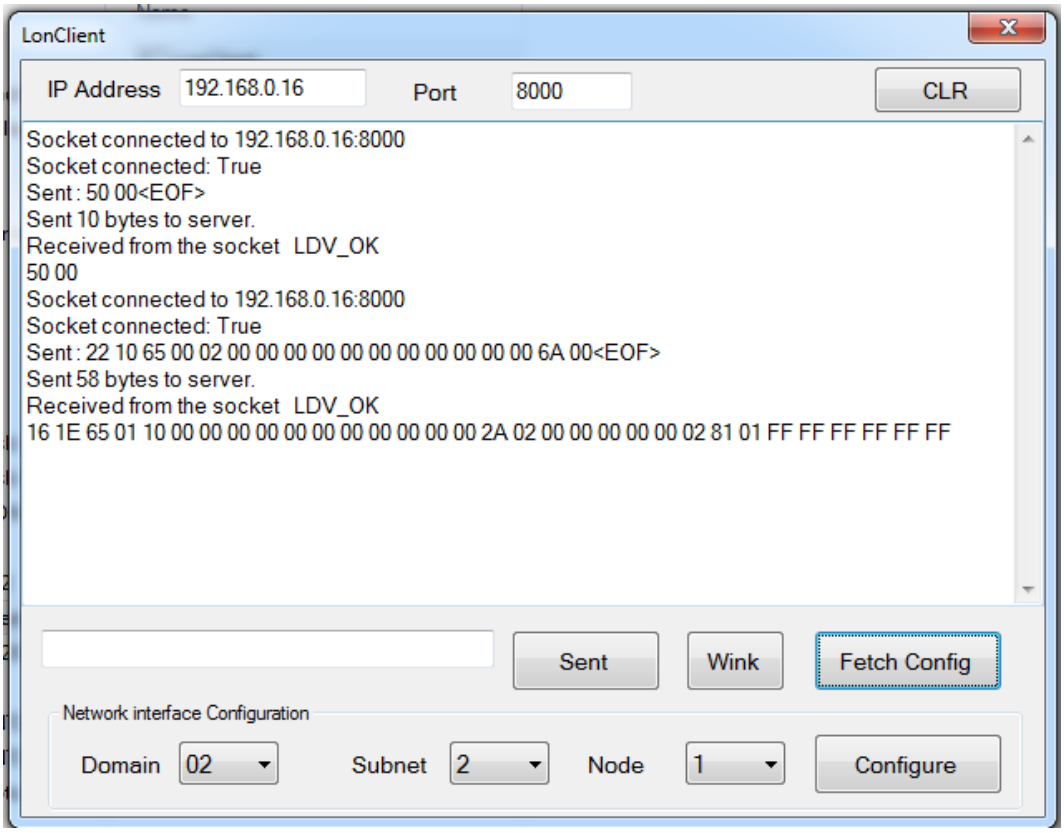

Figura 32 - LonClient GUI i zhvilluar si pjesë e komponentës middleware

LonClient aplikacion përbëhet nga katër pjesë:

- Pjesës së lartë në të cilën definojmë IP adresën dhe Portin e Serverit në të cilin kemi të instaluar LonServer aplikacionin.
- Sent butonit me tekstboksin para tij, të cilin mund ta përdorim si një terminal komunikimi në qoftë se kemi njohuri për formatin e komandave të cilat duhet të dërgohen.
- Wink dhe Fetch Configuration butonat, përmes të cilit me Wink komandën e testojmë lidhjen dhe në pajisjen U10 kemi ndezje të Wink Ledit dhe përmes Fetch Configuration komandës lexojmë konfiguracion e partitetit Domain – Subnet-Node e cila është tejet e rëndësishme për komunikim në vetë LON rrjetin.
- Network Interface Configuration pjesën, në të cilën paraqesim pamje vizuale të vlerave të konfiguracionit të lexuara nga Fetch Configuration butoni, si dhe të zgjedhim vlerat të cilat ne duam për të krijuar konfigurim të ri të pajisjes së Network Interface-it duke shtypur në fund Configure butonin.

Metodat kryesore të cilat mundësojnë TCP/IP komunikimin janë:

- private void StartClient(string textToSent) kjo metodë bën hapjen e një Socket Sesioni dhe dërgon mesazhin në pjesën textToSent, i cili paraprakisht është i thirrur nga dy metoda tjera private void PrepDatatoSent(string command) dhe private void StartToSentData(object sender, DoWorkEventArgs e) edhe kjo e bërë që çdo komunikim i ri të jetë në thread të veçantë që të mos ketë interference me GUI pjesën e aplikacionit.
- private void ConnectCallback(IAsyncResult ar) e cila e thirrën Socketin nga ObjectState dhe për të sinjalizuar fillimin e komunikimit.
- private void Send(Socket client, String data) bën dërgimin e pakove të cilat njëherit përmbajnë komandat që duhet të ekzekutohen në LON rrjetin.
- private void SendCallback(IAsyncResult ar) sinjalizon dërgimin e suksesshëm të mesazhit në LonServer.
- private void Receive(Socket client) e cila fillon leximin e mesazheve të pranuara.
- private void ReceiveCallback(IAsyncResult ar) mbaron leximin e mesazheve të pranuara dhe e mbyll Socket-in.

Regjistrimi i komunikimit në pjesën e LonClient bëhet në mënyrë të miksuar, duke loguar edhe pjesën e komandave të dërguara edhe komunikimin socket.

```
18:03:28.802- Socket connected to 192.168.0.16:8000
18:03:28.834- Socket connected: True
18:03:28.849- Sent : 50 00<EOF>
18:03:28.849- Sent 10 bytes to server.
18:03:29.052- Received from the socket   LDV_OK
50 00
18:03:31.969- Socket connected to 192.168.0.16:8000
18:03:31.985- Socket connected: True
18:03:32.000- Sent : 22 10 65 00 02 00 00 00 00 00 00 00 00 00 00 6A 00<EOF>
18:03:32.000- Sent 58 bytes to server.
18:03:32.110- Received from the socket   LDV_OK
16 1E 65 01 10 00 00 00 00 00 00 00 00 00 00 2A 02 00 00 00 00 00 02 81 01 FF FF FF FF FF FF
18:04:29.720- Socket connected to 192.168.0.16:8000
18:04:29.752- Socket connected: True
18:04:29.767- Sent : 22 1F 61 00 11 00 00 00 00 00 00 00 00 00 00 63 00 01 00 00 00 00 00 03 85 01 FF FF FF FF FF FF<EOF>
18:04:29.783- Sent 103 bytes to server.
18:04:29.954- Received from the socket   LDV_OK
16 0F 61 01 01 00 00 00 00 00 00 00 00 00 00 23
```

Figura 33 - Logu i përzier LONTalk mesazhe dhe SOCKET komunikimi

Procesimi i LON pakove poashtu bëhet në anën e klientit dhe për qëllimet e këtij punimi bëjmë procesimin e vlerave të konfiguraciont të Network Interface përmes private void processResponse(string str).

Testimi i komunikimit

Në adresën 192.168.0.16 porta 8000 do të startojmë aplikacionin server LonServer dhe në adresën 191.168.0.19 do të startojmë aplikacionin LonClient nga i cili do të testojmë:

- Wink Komandën (LON mesazh 50 00);
- Kërkimin e informatave të konfigurimit (LON mesazhi 22 10 65 00 02 00 00 00 00 00 00 00 00 00 00 00 00 6A 00);
- Konfigurimin e Network Interface (Domain 01, Subnet 3, Node 5).

Screenshot nga aplikacionet në të cilat kemi testuar komandat, ku në foton 1 dhe 2 kemi screenshot nga komandat që kemi dërguar, edhe atë në figurën 1 nga kërkimi i konfigurimit ekzistues do të shohim nga combobox që vlerat janë për Domain = 02 , Subnet = 2 , Node = 1. Ndërsa pas dërgimit të komandës për konfigurim vlerat e të njëjtës do të ndryshojnë në Domain =01, Subnet = 3, Node = 5.

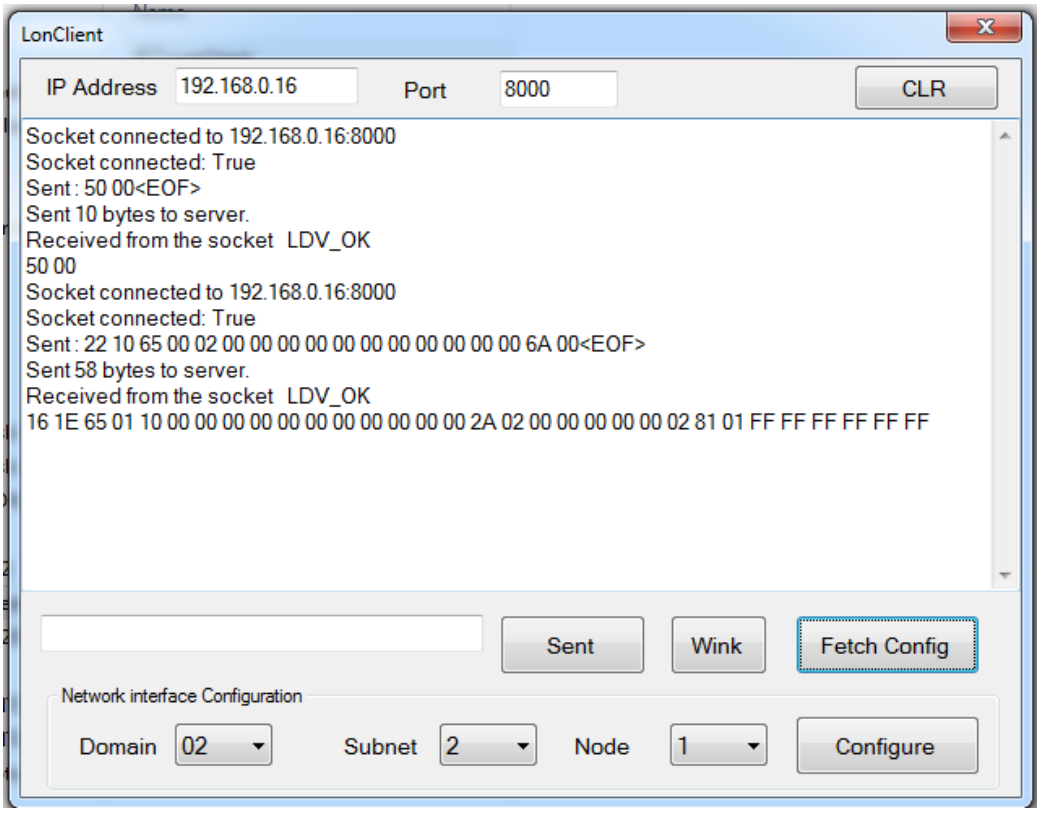

Figura 34 - LONClient Fetch Configuraton

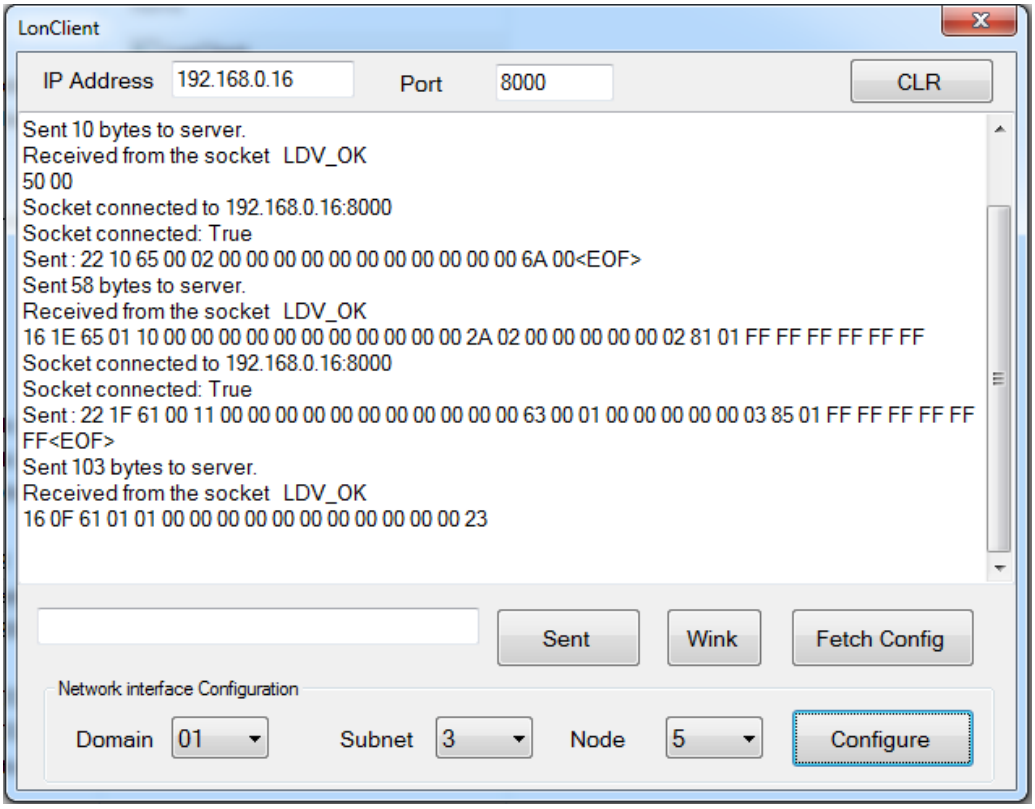

Figura 35 - LonClient Configure mesazhi Domain 01, Subnet 3, Node 5

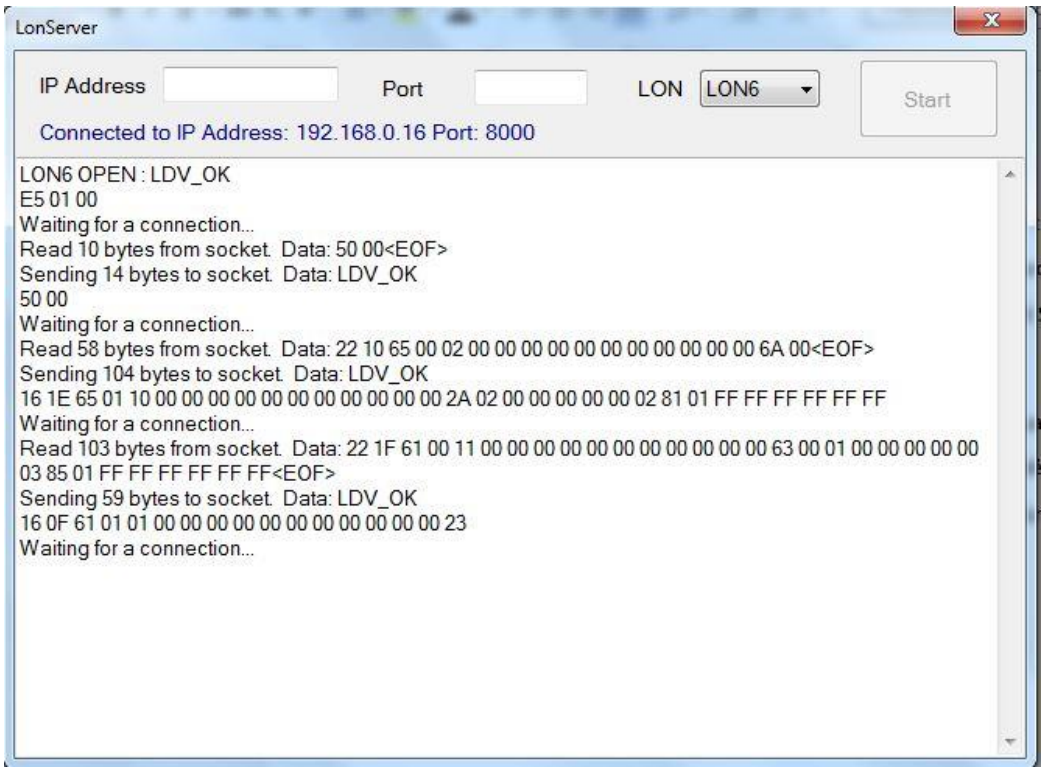

Figura 36 - LonServer komunikimi i Fetch Config dhe Configure mesazheve

8. Përfundimi

Trendi momental i ndërlidhjes së paisjeve të ndryshme të mençura (duke filluar nga paisjet elektro-shtëpiake e deri te paisjet që kryejnë procese industriale të ndërlikuara) synon që çdo paisje të jetë disi e lidhur në rrjetin globale - Internet. Ndërlidhja e paisjeve të cilat nuk kanë automatikisht mbështetje për TCP/IP krijon një kufizim i cili kërkon një zgjidhje dytësore për ndërlidhje. Kjo ndërlidhje për paisjet që kanë mbështetje për një protokoll rrjeti të kontrollit është e lehtëarritshme me përdorim të një paisje shtesë e cila luan rolin e ndërlidhësit edhe atë ndërmjet paisjes dhe një IP rrjeti. Këto lloje paisjesh aktualisht ekzistojnë në treg, por secila nga këto është e krijuar dhe ndërtuar nga prodhues të ndryshëm të cilët përmbushin interesat e po ashtu edhe qëllimin e krijimit të varshmërisë ndaj prodhuesit, vlen të theksohet se këto paisje karakterizohen me kosto shumë të lartë.

Koncepti i dhënë në këtë punim mundëson që me propozimin e dhënë softuerik, me përdorim të adapterëve që janë të lehtë gjinshëm në treg si dhe me paisjeve që kanë Chip të integruar, në rastin konkret mund të përdoret çfardo paisje SoC (System on a Chip) që është e bazuar në arkitekturën e procesorëve x86, por edhe me anë të optimizimit mund të ekzekutohet lehtësisht në çfardo ARM procesori, të krijohet një paisje ndërlidhëse ndërmjet rrjetave të kontrollit dhe IP rrjeteve.

Duke pasur parasysh edhe zhvillimet aktuale në fushën e IoT jo të gjitha paisjet që na rrethojnë mund të zëvendëosohen me pasije të mençura, duhet që edhe në të ardhmen të vazhdojmë përdorimin e paisjeve që kryejnë procese të ndryshme komplekse si dhe ti ndërlidhim, ndërsa informacionet që ato i gjenerojnë të mund të këmbehen në rrjetin global. Nga kjo lind edhe problematika që shumë zhvillues e intstitucione janë fokusuar për përsosje të standardeve të sigurisë në rrjeta dhe parandalimin e rrjedhjes së informacioneve të gjeneruara nga këto rrjeta kontrolli të arrijnë vetëm deri tek anët kompetente e të mos rrjedhin në mënyrë të pakontrolluar në Internet.

9. Shtojcat

Kodi i solucionit LonClient

```
using System;
using System.Windows.Forms;

namespace LonServer
{
   static class Program
   {
      ///<summary>
     /// The main entry point for the application.
      /// <summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new MainForm());
      }
   }
}


using System.Net.Sockets;
using System.Text;

namespace LonServer
{
  public class StateObject
   {
      // Client  socket.
      public Socket workSocket = null;
      // Size of receive buffer.
      public const int BufferSize = 1024;
      // Receive buffer.
      public byte[] buffer = new byte[BufferSize];
      // Received data string.
      public StringBuilder sb = new StringBuilder();
   }
}
namespace LonServer
{
   public class LonWorksModule
   {
      #region LDV32_DLL
      public static short m_OpenLDVhandle = 0;
      [DllImport("ldv32.dll")]
      public static extern short ldv_open(string id, ref short handle);
      [DllImport("ldv32.dll")]
      public static extern short ldv_open_cap(string id, ref short handle, LdvDeviceCaps lvdcaps, string ss, long tag);
      [DllImport("ldv32.dll")]
      public static extern short ldv_close(short handle);
      [DllImport("ldv32.dll")]
      public static extern short ldv_read(short handle, IntPtr data, short len);
      [DllImport("ldv32.dll")]
```

```
        public static extern short ldv_write(short handle, byte[] data, short len);
        [DllImport("ldv32.dll")]
        public static extern short ldv_register_event(short handle, IntPtr @event);
        [DllImport("ldv32.dll")]
        public static extern short ldv_get_device_info(string id, IntPtr data);
        #endregion

        #region WIN_EVENTS
        [DllImport("kernel32.dll")]
        public static extern IntPtr CreateEvent(IntPtr lpEventAttributes, bool bManualReset, bool bInitialState, string
lpName);
        [DllImport("kernel32.dll", SetLastError = true)]
        [return: MarshalAs(UnmanagedType.Bool)]
        public static extern bool CloseHandle(IntPtr hObject);
        #endregion

        #region LONEnumerations
        public enum LdvCode
        {
            LDV_OK = 0,
            LDV_NOT_FOUND = 1,
            LDV_ALREADY_OPEN = 2,
            LDV_NOT_OPEN = 3,
            LDV_DEVICE_ERR = 4,
            LDV_INVALID_DEVICE_ID = 5,
            LDV_NO_MSG_AVAIL = 6,
            LDV_NO_BUFF_AVAIL = 7,
            LDV_NO_RESOURCES = 8,
            LDV_INVALID_BUF_LEN = 9,
            LDV_NOT_ENABLED = 10,

            /* added in OpenLDV 1.0 */
            LDVX_INITIALIZATION_FAILED = 11,
            LDVX_OPEN_FAILED = 12,
            LDVX_CLOSE_FAILED = 13,
            LDVX_READ_FAILED = 14,
            LDVX_WRITE_FAILED = 15,
            LDVX_REGISTER_FAILED = 16,
            LDVX_INVALID_XDRIVER = 17,
            LDVX_DEBUG_FAILED = 18,
            LDVX_ACCESS_DENIED = 19,

            /* added in OpenLDV 2.0 */
            LDV_CAPABLE_DEVICE_NOT_FOUND = 20,
            LDV_NO_MORE_CAPABLE_DEVICES = 21,
            LDV_CAPABILITY_NOT_SUPPORTED = 22,
            LDV_INVALID_DRIVER_INFO = 23,
            LDV_INVALID_DEVICE_INFO = 24,
            LDV_DEVICE_IN_USE = 25,
            LDV_NOT_IMPLEMENTED = 26,
            LDV_INVALID_PARAMETER = 27,
            LDV_INVALID_DRIVER_ID = 28,
            LDV_INVALID_DATA_FORMAT = 29,
            LDV_INTERNAL_ERROR = 30,
            LDV_EXCEPTION = 31,
            LDV_DRIVER_UPDATE_FAILED = 32,
            LDV_DEVICE_UPDATE_FAILED = 33,
            LDV_STD_DRIVER_TYPE_READ_ONLY = 34,

            /* added in OpenLDV 4.0 */
```

```
        LDV_OUTPUT_BUFFER_SIZE_MISMATCH = 40,  // priority and non-priority output buffer sizes must be the
same
        LDV_INVALID_BUFFER_PARAMETER = 41,  // invalid buffer parameter (e.g. too large)
        LDV_INVALID_BUFFER_COUNT = 42,  // invalid buffer count (e.g. need at least one buffer of each type)
        LDV_PRIORITY_BUFFER_COUNT_MISMATCH = 43,  // if one of the priority output buffer counts is zero, then
both must be zero
        LDV_BUFFER_SIZE_TOO_SMALL = 44,  // buffer size is too small to support subsequent buffer configuration
changes
        LDV_BUFFER_CONFIGURATION_TOO_LARGE = 45,  // requested buffer configuration is too large to fit in
available space
        LDV_WARNING_APP_BUFFER_SIZE_MISMATCH = 46,  // application buffer input-output size mismatch may
cause problems (warning only)
    };
    public enum LdvDeviceCaps
    {
        LDV_DEVCAP_UNKNOWN = 0x00000000,
        LDV_DEVCAP_NOCHANGE = 0x00000000,

        // static capabilities
        LDV_DEVCAP_L5 = 0x00000001,
        LDV_DEVCAP_L2 = 0x00000002,

        LDV_DEVCAP_LWIP = 0x00000010,
        LDV_DEVCAP_PA = 0x00000020,
        LDV_DEVCAP_XDRIVER = 0x00000040,

        LDV_DEVCAP_SICB = 0x00000100,
        LDV_DEVCAP_LDVEX = 0x00000200,
        LDV_DEVCAP_NOSTATUS = 0x00000800,

        LDV_DEVCAP_SWITCHABLE = 0x00001000,
        LDV_DEVCAP_ATTACHABLE = 0x00002000,
        LDV_DEVCAP_ALL_STATIC = 0x0000FFFF,

        LDV_DEVCAP_ANY = (LDV_DEVCAP_L5 | LDV_DEVCAP_L2 | LDV_DEVCAP_PA),

        // dynamic capabilities
        LDV_DEVCAP_CURRENTLY_L5 = 0x00010000,
        LDV_DEVCAP_CURRENTLY_L2 = 0x00020000,

        LDV_DEVCAP_CURRENTLY_ATTACHED = 0x20000000,
        LDV_DEVCAP_CURRENTLY_AVAILABLE = 0x40000000,

        // LDV_DEVCAP_ALL_DYNAMIC = 0xFFFF0000,
    };
    #endregion
  }

}

using System;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;
using System.Threading;
using System.Runtime.InteropServices;
using Microsoft.Win32.SafeHandles;
using static LonServer.LonWorksModule;
```

```
using System.IO;

namespace LonServer
{
   public partial class MainForm : Form
   {
      #region GlobalVariablesAndEvents
      // Thread signal.
      public static ManualResetEvent allDone = new ManualResetEvent(false);
      private BackgroundWorker bwListener;
      short LDVResult;
      IntPtr myEventHandle = LonWorksModule.CreateEvent(IntPtr.Zero, false, true, "LONEvent");
      AutoResetEvent autoResetEvent = new AutoResetEvent(false);
      IntPtr pdata = Marshal.UnsafeAddrOfPinnedArrayElement(data, 0);
      public static byte[] data = new byte[257];
      #endregion

      #region WinFormEvents
      public MainForm()
      {
         InitializeComponent();
      }
      private void btnConnect_Click(object sender, EventArgs e)
      {
         if (cbLon.SelectedItem == null)
         {
            MessageBox.Show("Please choose LON");
            return;
         }
         if (startLon(cbLon.SelectedItem.ToString()))
         {
            this.bwListener = new BackgroundWorker();
            this.bwListener.WorkerSupportsCancellation = true;
            this.bwListener.DoWork += new DoWorkEventHandler(StartToListen);
            this.bwListener.RunWorkerAsync();
            btnConnect.Enabled = false;
         }
      }
      private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
      {
         if (this.bwListener != null)
         {
            this.bwListener.CancelAsync();
            this.bwListener.Dispose();
         }
         LDVResult = ldv_close(m_OpenLDVhandle);
      }
      #endregion

      #region SockeEvents
      private void StartToListen(object sender, DoWorkEventArgs e)
      {
         StartListening();
      }
      private void StartListening()
      {
         // Data buffer for incoming data.
         byte[] bytes = new Byte[1024];

         // Establish the local endpoint for the socket.
```

```
        // The DNS name of the computer
        IPHostEntry ipHostInfo;
        int portNr = 8000;
        if (string.IsNullOrEmpty(txtIPAddress.Text))
        {
            ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
        }
        else
        {
            ipHostInfo = Dns.GetHostEntry(txtIPAddress.Text);
        }

        if (!string.IsNullOrEmpty(txtPort.Text))
        {
            portNr = Convert.ToInt16(txtPort.Text);
        }
        IPAddress ipAddress = ipHostInfo.AddressList[1];
        IPEndPoint localEndPoint = new IPEndPoint(ipAddress, portNr);
        lblConnectionResult.Invoke((MethodInvoker)(() => lblConnectionResult.Text = "Connected to IP Address: " +
ipHostInfo.AddressList[1].ToString() + " Port: " + portNr.ToString()));
        SocketLog("Connected to IP Address: " + ipHostInfo.AddressList[1].ToString() + " Port: " + portNr.ToString());
        // Create a TCP/IP socket.
        Socket listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        // Bind the socket to the local endpoint and listen for incoming connections.
        listener.Bind(localEndPoint);
        listener.Listen(100);
        AppendResultText("Waiting for a connection...\n");
        while (true)
        {
            // Set the event to nonsignaled state.
            allDone.Reset();
            // Start an asynchronous socket to listen for connections.
            listener.BeginAccept(new AsyncCallback(AcceptCallback), listener);
            // Wait until a connection is made before continuing.
            allDone.WaitOne();
        }
    }
    private void AcceptCallback(IAsyncResult ar)
    {
        // Signal the main thread to continue.
        allDone.Set();

        // Get the socket that handles the client request.
        Socket listener = (Socket)ar.AsyncState;
        Socket handler = listener.EndAccept(ar);

        // Create the state object.
        StateObject state = new StateObject();
        state.workSocket = handler;
        handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new AsyncCallback(ReadCallback), state);
    }
    private void ReadCallback(IAsyncResult ar)
    {
        String content = String.Empty;
        // Retrieve the state object and the handler socket
        // from the asynchronous state object.
        StateObject state = (StateObject)ar.AsyncState;
        Socket handler = state.workSocket;

        // Read data from the client socket.
```

```
        int bytesRead = handler.EndReceive(ar);

        if (bytesRead > 0)
        {
          // There  might be more data, so store the data received so far.
          state.sb.Append(Encoding.ASCII.GetString(
            state.buffer, 0, bytesRead));
          // Check for end-of-file tag. If it is not there, read
          // more data.
          content = state.sb.ToString();
          if (content.IndexOf("<EOF>") > -1)
          {
            AppendResultText("Read " + content.Length.ToString() + " bytes from socket.  Data: " + content);
            SocketLog("Read " + content.Length.ToString() + " bytes from socket:" +
handler.LocalEndPoint.ToString());
            short ldvResult = writeLDV(content);

            byte[] inresult = readlLDV();
            // try 3 times to read data from LON Interface
            for (int i = 0; i < 2; i++)
            {
              if (inresult[0] == 0x00)
              {
                Thread.Sleep(100);
                inresult = readlLDV();
              }
            }
            Send(handler, ((LdvCode)LDVResult).ToString() + Environment.NewLine + ByteArrayToHexString(inresult));
          }
          else
          {
            // Not all data received. Get more.
            handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
            new AsyncCallback(ReadCallback), state);
          }
        }
      }
      private void Send(Socket handler, String data)
      {
        // Convert the string data to byte data using ASCII encoding.
        byte[] byteData = Encoding.ASCII.GetBytes(data);
        AppendResultText("Sending " + data.Length.ToString() + " bytes to socket.  Data: " + data);
        SocketLog("Sending " + data.Length.ToString() + " bytes to socket: " +
handler.RemoteEndPoint.ToString()+Environment.NewLine);
        // Begin sending the data to the remote device.
        handler.BeginSend(byteData, 0, byteData.Length, 0,
          new AsyncCallback(SendCallback), handler);
      }
      private void SendCallback(IAsyncResult ar)
      {
        // Retrieve the socket from the state object.
        Socket handler = (Socket)ar.AsyncState;
        // Complete sending the data to the remote device.
        int bytesSent = handler.EndSend(ar);
        handler.Shutdown(SocketShutdown.Both);
        handler.Close();
        AppendResultText("Waiting for a connection...\n");
      }
      #endregion
```

```
#region LonMethods
private bool startLon(string ControllerID)
{
   bool result = false;
   GC.ReRegisterForFinalize(autoResetEvent);
   autoResetEvent.SafeWaitHandle = new SafeWaitHandle(myEventHandle, true);
   LDVResult = LonWorksModule.ldv_open(ControllerID, ref LonWorksModule.m_OpenLDVhandle);
   LonWorksModule.ldv_register_event(LonWorksModule.m_OpenLDVhandle, myEventHandle);
   CommLog("**************************");
   if (LDVResult != 0)
   {
      CommLog(ControllerID.ToString() + " NOT STARTED");
      AppendResultText(ControllerID.ToString() + " NOT OPEN : " + ((LdvCode)LDVResult).ToString());
      result = false;
   }
   else
   {
      CommLog(ControllerID.ToString() + " SUCESSFULLY STARTED");
      AppendResultText(ControllerID.ToString() + " OPEN : " + ((LdvCode)LDVResult).ToString());
      // btnConnect.Invoke((MethodInvoker)(() => btnConnect.Enabled = false));
      result = true;
   }
   byte[] inresult = readlLDV();
   AppendResultText(ByteArrayToHexString(inresult).ToString());
   return result;
}
private short writeLDV(string txt)
{
   byte[] bytWrite = new byte[] { };
   bytWrite = HexStringToByteArray(txt.Substring(0, txt.IndexOf("<")));
   LDVResult = ldv_write(m_OpenLDVhandle, bytWrite, Convert.ToInt16(bytWrite.Count()));
   CommLog("RITE:" + ByteArrayToHexString(bytWrite) + " -" + ((LdvCode)LDVResult).ToString());
   return LDVResult;
}
private byte[] readlLDV()
{
   byte[] dataR = new byte[257];
   short Readresult;
   IntPtr pdataR = Marshal.UnsafeAddrOfPinnedArrayElement(dataR, 0);

   Readresult = ldv_read(m_OpenLDVhandle, pdataR, 257);
   byte[] result = RemoveZeros(dataR);
   if (result[0] != 0x00)
   {
      CommLog("READ :" + ByteArrayToHexString(result) + " -" + ((LdvCode)LDVResult).ToString());
   }
   return result;
}
#endregion

#region GeneralMethods
private byte[] RemoveZeros(byte[] data)
{
   int leng = Convert.ToInt32(data[1]) + 2;
   if (data.Length < leng)
   {
      leng = data.Length;
   }
   byte[] resizedarr = new byte[leng];
```

```
            for (int i = 0; i < leng; i++)
            {
                resizedarr[i] = data[i];
            }
            return resizedarr;
        }
        private byte[] HexStringToByteArray(string s)
        {
            s = s.Replace(" ", "");
            byte[] buffer = new byte[s.Length / 2];
            for (int i = 0; i < s.Length; i += 2)
                buffer[i / 2] = (byte)Convert.ToByte(s.Substring(i, 2), 16);
            return buffer;
        }
        private string ByteArrayToHexString(byte[] data)
        {
            StringBuilder sb = new StringBuilder(data.Length * 3);
            foreach (byte b in data)
                sb.Append(Convert.ToString(b, 16).PadLeft(2, '0').PadRight(3, ' '));
            return sb.ToString().ToUpper();
        }
        #endregion

        #region LogMethodsAndCrossThreadChanges
        private void AppendResultText(string text)
        {
            if (this.InvokeRequired)
            {
                this.BeginInvoke((Action)delegate { this.AppendResultText(text); });
                return;
            }
            txtResult.AppendText(text + Environment.NewLine);
        }
        public static void CommLog(string msg)
        {
            string filepath = Environment.CurrentDirectory.ToString() + "\\CommLogg" + DateTime.Now.ToString("dd-
MM-yyyy") + ".txt";
            using (StreamWriter writer = new StreamWriter(filepath, true))
            {
                writer.WriteLine(DateTime.Now.ToString("HH:mm:ss.fff") + "" + "- " + msg);
                writer.Flush();
            }
        }
        public static void SocketLog(string msg)
        {
            try
            {
                string filepath = Environment.CurrentDirectory.ToString() + "\\IPSocetLogg" + DateTime.Now.ToString("dd-
MM-yyyy") + ".txt";
                using (StreamWriter writer = new StreamWriter(filepath, true))
                {
                    Writer.WriteLine(DateTime.Now.ToString("HH:mm:ss.fff") + "" + "- " + msg);
                    writer.Flush();
                }
            }
            catch (IOException)
            {
                return;
            }
        }
```

```
        #endregion

    }
}
```

Kodi i solucionit LonClient

```
using System;
using System.Windows.Forms;

namespace LonClient
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LonClient());
        }
    }
}
```

```
using System.Net.Sockets;
using System.Text;
namespace LonClient
{
    public class StateObject
    {
        // Client  socket.
        public Socket workSocket = null;
        // Size of receive buffer.
        public const int BufferSize = 1024;
        // Receive buffer.
        public byte[] buffer = new byte[BufferSize];
        // Received data string.
        public StringBuilder sb = new StringBuilder();
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace LonClient
{
   public partial class LonClient : Form
   {
      #region GlobalVariables
      // ManualResetEvent instances signal completion.
      private static ManualResetEvent connectDone = new ManualResetEvent(false);
      private static ManualResetEvent sendDone = new ManualResetEvent(false);
      private static ManualResetEvent receiveDone = new ManualResetEvent(false);
      private BackgroundWorker bwListener;
      private static String response = String.Empty;
      //the command string that will be passed to the networks Socket as data part
      private string commandPass = string.Empty;
      #endregion

      public LonClient()
      {
         InitializeComponent();

         // populate comboboxes with allowed values
         for (int i = 1; i <= 255; i++)
         {
            cbNode.Items.Add(i.ToString());
         }
         for (int i = 1; i <= 127; i++)
         {
            cbSubnet.Items.Add(i.ToString());
         }
      }

      #region SocketMethods
      private void StartClient(string textToSent)
      {
         // Connect to a remote device.
         try
         {
            // Establish the remote endpoint for the socket.
            IPHostEntry ipHostInfo;
            int portNr = 8000;
            if (string.IsNullOrEmpty(txtIPAddress.Text))
            {
               ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
            }
            else
            {
               ipHostInfo = Dns.GetHostEntry(txtIPAddress.Text);
            }
            if (!string.IsNullOrEmpty(txtPort.Text))
            {
               portNr = Convert.ToInt16(txtPort.Text);
            }
            IPAddress ipAddress = ipHostInfo.AddressList[1];
            IPEndPoint remoteEP = new IPEndPoint(ipAddress, portNr);
```

```
        // Create a TCP/IP socket.
        Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        client.ReceiveTimeout = 100;
        // Connect to the remote endpoint.
        client.BeginConnect(remoteEP, new AsyncCallback(ConnectCallback), client);
        connectDone.WaitOne();
        AppendResultText("Socket connected: " + client.Connected.ToString());

        // Send test data to the remote device.
        Send(client, textToSent + "<EOF>");
        sendDone.WaitOne();

        // Receive the response from the remote device.
        Receive(client);
        receiveDone.WaitOne();

        // Release the socket.
        client.Shutdown(SocketShutdown.Both);
        client.Close();
    }
    catch (Exception e)
    {
        AppendResultText(e.Message.ToString());
    }
}
private void ConnectCallback(IAsyncResult ar)
{
    try
    {
        // Retrieve the socket from the state object.
        Socket client = (Socket)ar.AsyncState;
        // Complete the connection.
        client.EndConnect(ar);
        AppendResultText(String.Format("Socket connected to {0}", client.RemoteEndPoint.ToString()));
        // Signal that the connection has been made.
        connectDone.Set();
    }
    catch (Exception e)
    {
        AppendResultText(e.Message.ToString());
        this.bwListener.CancelAsync();
        this.bwListener.Dispose();
    }
}
private void Receive(Socket client)
{
    try
    {
        // Create the state object.
        StateObject state = new StateObject();
        state.workSocket = client;
        // Begin receiving the data from the remote device.
        client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new AsyncCallback(ReceiveCallback), state);
    }
    catch (Exception e)
    {
        AppendResultText(e.Message.ToString());
    }
}
private void ReceiveCallback(IAsyncResult ar)
```

```
{
   try
   {
      // Retrieve the state object and the client socket
      // from the asynchronous state object.
      StateObject state = (StateObject)ar.AsyncState;
      Socket client = state.workSocket;

      // Read data from the remote device.
      int bytesRead = client.EndReceive(ar);

      if (bytesRead > 0)
      {
         // There might be more data, so store the data received so far.
         state.sb.Append(Encoding.ASCII.GetString(state.buffer, 0, bytesRead));

         // Get the rest of the data.
         client.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0, new AsyncCallback(ReceiveCallback), state);
      }
      else
      {
         // All the data has arrived; put it in response.
         if (state.sb.Length > 1)
         {
            response = state.sb.ToString();
            processResponse(response);
            AppendResultText("Received from the socket   " + response);
         }
         // Signal that all bytes have been received.
         receiveDone.Set();
      }
   }
   catch (Exception e)
   {
      AppendResultText(e.Message.ToString());
   }
}
private void Send(Socket client, String data)
{
   // Convert the string data to byte data using ASCII encoding.
   byte[] byteData = Encoding.ASCII.GetBytes(data);
   AppendResultText(String.Format("Sent : " + data));
   // Begin sending the data to the remote device.
   client.BeginSend(byteData, 0, byteData.Length, 0, new AsyncCallback(SendCallback), client);
}
private void SendCallback(IAsyncResult ar)
{
   try
   {
      // Retrieve the socket from the state object.
      Socket client = (Socket)ar.AsyncState;

      // Complete sending the data to the remote device.
      int bytesSent = client.EndSend(ar);
      AppendResultText(String.Format("Sent {0} bytes to server.", bytesSent));

      // Signal that all bytes have been sent.
      sendDone.Set();
   }
   catch (Exception e)
```

```
        {
            AppendResultText(e.Message.ToString());
        }
    }
    #endregion

    #region BackgroundWorkerMethdods
    private void PrepDatatoSent(string command)
    {
        this.bwListener = new BackgroundWorker();
        commandPass = command;
        this.bwListener.WorkerSupportsCancellation = true;
        this.bwListener.DoWork += new DoWorkEventHandler(StartToSentData);
        this.bwListener.RunWorkerAsync();
    }
    private void StartToSentData(object sender, DoWorkEventArgs e)
    {
        if (!string.IsNullOrEmpty(commandPass))
        {
            StartClient(commandPass);
            commandPass = string.Empty;
            this.bwListener.CancelAsync();
            this.bwListener.Dispose();
            connectDone = new ManualResetEvent(false);
            sendDone = new ManualResetEvent(false);
            receiveDone = new ManualResetEvent(false);
        }
        else
        {
            this.bwListener.CancelAsync();
            this.bwListener.Dispose();
        }
    }
    #endregion

    #region WindowsEvents
    private void btnSent_Click(object sender, EventArgs e)
    {
        PrepDatatoSent(txtDataToSent.Text);
    }
    private void btnWink_Click(object sender, EventArgs e)
    {
        PrepDatatoSent("50 00");
    }
    private void btnFetchConfig_Click(object sender, EventArgs e)
    {
        PrepDatatoSent("22 10 65 00 02 00 00 00 00 00 00 00 00 00 00 00 6A 00");
    }
    private void btnClear_Click(object sender, EventArgs e)
    {
        txtResult.Text = "";
    }
    private void btnConfigure_Click(object sender, EventArgs e)
    {
        PrepDatatoSent("22 1F 61 00 11 00 00 00 00 00 00 00 00 00 00 00 63 00 " + cbDomain.SelectedItem.ToString()
+ " 00 00 00 00 00 0" + cbSubnet.SelectedItem.ToString() + " 8" + cbNode.SelectedItem.ToString() + " 01 FF FF FF FF
FF FF");
    }
    #endregion
```

```
    #region ApplicativeProccessingMethods
    private void processResponse(string str)
    {
        string result = str.Substring(str.IndexOf(Environment.NewLine), str.Length -
str.IndexOf(Environment.NewLine)).TrimStart('\r', '\n');

        byte[] processingRslt = HexStringToByteArray(result);

        if (processingRslt[0] == 0x16 && processingRslt.Length > 30)
        {
            //populate comboboxes
            cbDomain.Invoke((MethodInvoker)(() => cbDomain.SelectedItem = Convert.ToString(processingRslt[17],
16).PadLeft(2, '0')));
            cbSubnet.Invoke((MethodInvoker)(() => cbSubnet.SelectedItem = Convert.ToString(processingRslt[23], 16)));
            cbNode.Invoke((MethodInvoker)(() => cbNode.SelectedItem = Convert.ToString(processingRslt[24],
16).Substring(1, 1)));
        }

    }

    private void AppendResultText(string text)
    {
        if (this.InvokeRequired)
        {
            this.BeginInvoke((Action)delegate { this.AppendResultText(text); });
            return;
        }
        CommLog(text);
        txtResult.AppendText(text + Environment.NewLine);
    }
    private byte[] RemoveZeros(byte[] data)
    {
            int leng = Convert.ToInt32(data[1]) + 2;
            if (data.Length < leng)
            {
                leng = data.Length;
            }
            byte[] resizedarr = new byte[leng];

            for (int i = 0; i < leng; i++)
            {
                resizedarr[i] = data[i];

            }
            return resizedarr;
    }
    private byte[] HexStringToByteArray(string s)
    {
        s = s.Replace(" ", "");
        byte[] buffer = new byte[s.Length / 2];
        for (int i = 0; i < s.Length; i += 2)
            buffer[i / 2] = (byte)Convert.ToByte(s.Substring(i, 2), 16);
        return buffer;
    }
    private string ByteArrayToHexString(byte[] data)
    {
        StringBuilder sb = new StringBuilder(data.Length * 3);
        foreach (byte b in data)
            sb.Append(Convert.ToString(b, 16).PadLeft(2, '0').PadRight(3, ' '));
        return sb.ToString().ToUpper();
```

```
        }
        public static void CommLog(string msg)
        {
            try
            {
                string filepath = Environment.CurrentDirectory.ToString() + "\\CommunicationLogg" +
DateTime.Now.ToString("dd-MM-yyyy") + ".txt";
                using (StreamWriter writer = new StreamWriter(filepath, true))
                {
                    writer.WriteLine(DateTime.Now.ToString("HH:mm:ss.fff") + "" + "- " + msg);
                    writer.Flush();
                }
            }
            catch (IOException)
            {
                return;
            }
        }
        #endregion
    }
}
```

10. Referencat

[1] Spider Systems, Ltd., "Packets and Protocols", 1990.

[2] ISO, "Open Systems Interconnection--Basic Reference Model", ISO 7498.

[3] ISO, "Internal Organization of the Network Layer", ISO 8648.

[4] https://tools.ietf.org/html/rfc1180

[5] https://tools.ietf.org/html/rfc793

[6[https://www.ietf.org/rfc/rfc768.txt

[7] https://en.wikipedia.org/wiki/Phase-shift_keying

[8] https://tools.ietf.org/html/rfc1051

[9] https://tools.ietf.org/html/rfc1201

[10] http://www.bacnet.org/DL-Docs/SPC135-Nashville-Minutes-1987-06.pdf

[11] http://www.eiba.ru/download/eib02_system.pdf

[12] https://www.echelon.com/assets/bltf61daa59060f0150/005-0193-01C_PL_Data_Book.pdf

[13] CTA-852.1: Enhanced Protocol for Tunneling Component Network Protocols over Internet Protocol Channels

[14] http://www.lonmark.org/membership/directory/ Directory of LonMark International Member Companies

[15] CTA-709.1: Control Network Protocol Specification

[16] Manning Publications, ".NET Multithreading Paperback", Alan Dennis, 2003

[17] Addison-Wesley, "TCP/IP Illustrated, Volume 2: The Implementation", Wright G. and W. Stevens,1994.

[18] Sybex, "C# Network Programming", Richard Blum, 2002

[19] Morgan Kaufmann. "TCP/IP Sockets in C#: Practical Guide for Programmers (The Practical Guides)", David Makofske, Michael J. Donahoo, Kenneth L. Calvert, 2004

[20] “Internet of things strategic research roadmap,” in Internet of Things: Global Technological and Societal Trends, vol. 1, O. Vermesan, P. Friess, P. Guillemin, 2011.

[21] Itu Internet Report 2005: The Internet of Things, I. Peña-López, 2005

[22] “Choices for interaction with things on Internet and underlying issues,” Ad Hoc Networks, vol. 28, I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, 2015

[23] “Towards internet of things: survey and future vision,” International Journal of Computer Networks, vol. 5, no. 1, O. Said and M. Masud, 2013

[24] “Research on the architecture of internet of things,” in Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10), vol. 5, M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, IEEE, Chengdu, China, August 2010.

[25] “The fog computing paradigm: scenarios and security issues,” in Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '14), I. Stojmenovic and S. Wen, IEEE, Warsaw, Poland, 2014

[26] “Fog computing and smart gateway based communication for cloud of things,” in Proceedings of the 2nd IEEE International Conference on Future Internet of Things and Cloud (FiCloud '14), M. Aazam and E.-N. Huh, Barcelona, Spain, August 2014

[27] “SIoT: giving a social structure to the internet of things,” IEEE Communications Letters, vol. 15, no. 11, L. Atzori, A. Iera, and G. Morabito, 2011

[28] “Sensor mania! The internet of things, wearable computing, objective metrics, and the quantified self 2.0,” Journal of Sensor and Actuator Networks, vol. 1, no. 3, M. Swan, 2012

[29] “A survey of mobile phone sensing,” IEEE Communications Magazine, vol. 48, no. 9, N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, 2010

[30] “The Internet of Things: a survey,” Computer Networks, vol. 54, no. 15, L. Atzori, A. Iera, and G. Morabito, 2010

[31] “The web of things: a survey,” Journal of Communications, vol. 6, no. 6, D. Zeng, S. Guo, and Z. Cheng, 2011