SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

INTELIGENTNI SUSTAV ZA UNOS PODATAKA O TELEKOMUNIKACIJSKOJ TRANSPORTNOJ MREŽI

Magistarski rad

Zagreb, 2001.

Magistarski rad izrađen je na Zavodu za telekomunikacije
Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Mentor: prof. dr. sc. Xxxxx Xxxxx

Magistarski rad ima 111 stranica.

Rad br.

Sadržaj

1. Uvod

U današnje vrijeme sve je brži razvoj novih tehnologija. Kompleksnost transportne telekomunikacijske mreže je sve veća, uz sve veći broj korisnika i novih usluga. Ulažu se velika sredstva za modernizaciju telekomunikacijske opreme, uvođenje novih usluga, i sve je veća važnost softverskih rješenja i programskih alata. Ovakav razvoj otvara tržišnu utakmicu sve većem broju proizvođača i dobavljača telekomunikacijske opreme i postavlja sve veće zahtjeve, posebno u području upravljanja mrežom. Samo kvalitetnim upravljanjem informacijskim resursima i stalnim unapređenjem telekomunikacijske mreže može se držati korak u ovom složenom i propulzivnom području.

Da bi mogli ostvariti efikasan nadzor i upravljanje telekomunikacijskom mrežom nužan je informacijski sustav. Temelj takvog informacijskog sustava je baza podataka koja sadrži sve relevantne podatke o transportnoj mreži. Metodologija informacijskog opisa transportne mreže i sama baza podataka transportne mreže razvijena je i opisana kao jedan od rezultata istraživačkog projekta na Zavodu za telekomunikacije Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu [3].

Vrlo kompleksni procesi i odnosi u realnim mrežama koje treba opisati i velika količina podataka koju treba unijeti za takav informacijski opis, te potrebno vrijeme i velika mogućnost pogreške pri unosu složenih podataka su problemi čije rješenje zahtijeva inteligentni informacijski sustav. Upravo takav sustav koji bi uz pomoć ugrađenog znanja i intuitivnog grafičkog sučelja omogućio efikasno i brzo punjenje baze podataka transportne mreže relevantnim i točnim podacima predmet je ovog rada. Ekspertni sustav, nazvan ***T**ransport **N**etwork **D**esigner* (TND) sastoji se iz posebne baze podataka nazvane repozitorij predložaka i triju aplikacija: NED (***N**etwork **E**lement **D**esigner*), EQD (*Network **Eq**uipment **D**esigner*) i NTD (***N**etwork **T**opology **D**esigner*) koje će biti opisane u ovom radu.

Poglavlje 2. sadrži temeljne definicije transportne mreže koje služe kao početna uporišta za definiranje čitavog modela. Bitno je reći da se koncepti objašnjeni u ovom poglavlju temelje na ITU preporukama, te na rezultatima prijašnjih istraživanja i studija izrađenih na Zavodu za telekomunikacije [9], [10], [11].

U trećem poglavlju opisan je konceptualni model baze podataka transportne mreže TRNET koji je ujedno i implementacijski.

U četvrtom poglavlju dan je kratak pregled značajnijih tehnologija za pristup klijentske aplikacije prema bazi podataka. Prilikom odabira tehnologije za pristup bazi podataka koja će se upotrijebiti u izradi ekspertnog sustava važno je upoznati se s mogućnostima i restrikcijama koje pojedine tehnologije pružaju i naći kompromis između brzine i upravljivosti koje pružaju sučelja niske razine naspram efikasnosti koda i portabilnosti koju pružaju sučelja visoke razine, vodeći se postavljenim zahtjevima.

U petom poglavlju opisan je ekspertni sustav TND sa detaljnim opisom njegovih komponenata: repozitorija predložaka i aplikacija NED, EQD i NTD, njihovih međusobnih odnosa i načina na koji oni djeluju kao sustav koji omogućuje vrlo

efikasan unos podataka u bazu podataka transportne mreže TRNET i grafički prikaz unesenih podataka.

U šestom poglavlju prikazana je verifikacija ekspertnog sustava TND na konkretnom modelu transportne mreže. Pokazano je na koji način je sustav korišten da bi se kreirali predlošci mrežnih elemenata i uređaja i u vrlo kratkom vremenu unijeli svi podaci potrebni za opis modela. Provjera unesenih podataka izvršena je pregledom grafičkog prikaza unesenih podataka aplikacijom NTD i raznim upitima nad bazom podataka koji su pokazali da uneseni podaci u potpunosti odgovaraju zadanom modelu.

2. Model transportne mreže

2.1 Osnovne postavke

Telekomunikacijska mreža se može funkcionalno prikazati kao hijerarhija apstrakcija, u kojoj je jedan od temeljnih principa slojevitost mreže. U sklopu takvog koncepta *transportna mreža* obavlja logičke funkcije koje se odnose na prijenos i usmjeravanje informacije. Ona osigurava pouzdan prijenos, ne ulazeći u sadržaj onoga što se prenosi. Logička veza u transportnoj mreži je ustanovljena između dva korisnička terminala preko pristupnih i tranzitnih čvorova. Prema tome, terminali, pristupni čvorovi i tranzitni čvorovi predstavljaju funkcijsku (logičku) sliku događanja u stvarnim fizičkim pristupnim i tranzitnim čvorovima transmisijske mreže. Upravljivi resursi (objekti) su funkcionalni prikazi fizičkih resursa implementiranih u transmisijskoj mreži.

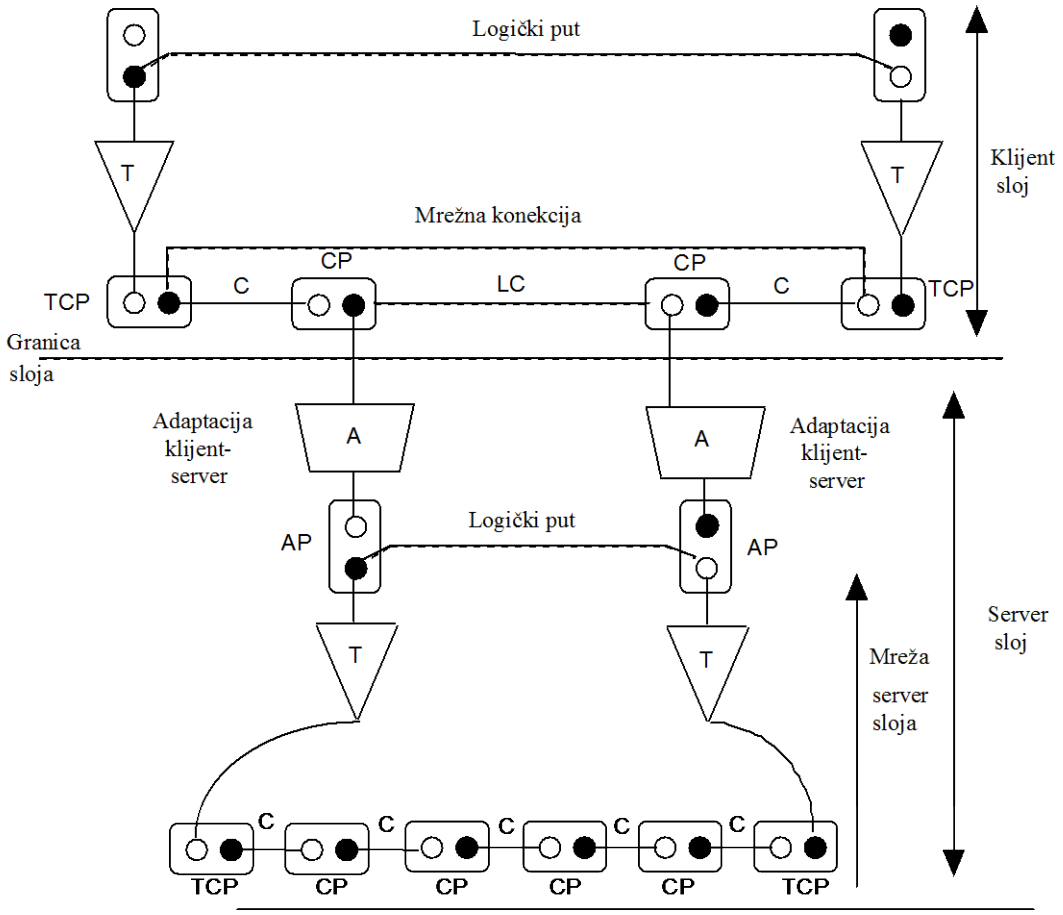
Budući da je transportna mreže velika i složena, sazdana često od različitih komponenata, potrebno je definirati mrežni model s komponentama, funkcijama i entitetima koji će omogućiti efikasno upravljanje. Generički model transportne mreže (*Generic Network Information Model*) je definiran tako da je funkcionalno neovisan o implementacijskoj tehnologiji. Drugim riječima, definiraju se funkcije, komponente i entiteti koji govore što treba učiniti, ali ne i kako. Generički model opisuje funkcioniranje i arhitekturu transportne mreže na apstraktni način, pomoću manjeg broja definiranih elemenata arhitekture [3], [7], [8], [9].

2.2 GENERIČKI MODEL TRANSPORTNE MREŽE

Generički model transportne mreže se temelji na konceptima *slojevitosti* i *podjele* koji omogućavaju visok stupanj rekurzivnosti.

Svaki transportni sloj osigurava transport za viši sloj, a koristi transport ponuđen od nižeg sloja. Sloj koji nudi transport naziva se server, a sloj koji koristi transport, klijent. Za takve slojeve se kaže da sudjeluju u klijent/server vezi. Da ne bi došlo do zabune, potrebno je dati dodatno objašnjenje. U klijent/server vezi je klijent viši sloj, a server niži. To je obrnuto u odnosu na brzine prijenosa u pojedinom sloju. Naime, brzina server sloja je obično viša nego brzina klijent sloja. Na slici 2.1 dan je primjer tog odnosa zajedno s prikazom ostalih elemenata transportne mreže koji će se u nastavku detaljnije opisati.

Kako je često pojedini sloj mreže kompliciran jer pokriva veliko geografsko područje, uobičajena je podjela sloja mreže na odgovarajući broj podmreža. Primjeri takve podjele mogu biti internacionalna, nacionalna, tranzitna, regionalna i lokalna podmreža.



Slika 2.1 – Prikaz elemenata transportne mreže i klijent/server veze

Transportna mreža se može dekomponirati u više nezavisnih slojeva s klijent/server pridruživanjem između susjednih slojeva. Svaki sloj mreže može se posebno podijeliti na način koji najbolje odražava internu strukturu sloja.

Koncept podjele je značajan kao okvir za definiranje:

- strukture mreže unutar sloja mreže,
- administrativnih granica između mrežnih operatora koji zajedno osiguravaju puteve od-kraja-do-kraja unutar pojedinog sloja,
- granica domena unutar sloja mreže pojedinog operatora, s pogledom na performanse elemenata podmreža od kojih je mreža sastavljena,
- nezavisnih granica domena rutinga u odnosu na proces upravljanja puteva.

Koncept slojevitosti transportne mreže se temelji na sljedećim pretpostavkama:

- svaki sloj mreže moguće je definirati nezavisno od ostalih slojeva,
- za svaki sloj mreže su definirane jednostavne funkcije,
- jednostavnije je projektirati i definirati rad svakog sloja posebno nego čitave transportne mreže,
- model slojevite mreže može biti korisniji u definiranju upravljanih objekata u *TMN*-u,

- svaki sloj mreže posjeduje sposobnost samostalnog rada i održavanja kao što su alternativno usmjeravanje i automatski oporavak nakon otkrivanje grešaka. Ove mogućnosti minimiziraju akcije održavanja i utjecaj na druge slojeve,
- sa stanovišta arhitekture mreže moguće je dodati ili promijeniti sloj bez utjecaja na druge slojeve.

2.2.1 Koncept podjele

Koncept podjele može se primijeniti u dva srodna područja:

- podjela podmreža kojima se opisuje topologija,
- podjela mrežnih konekcija kojima se opisuje povezanost.

Podmreža jednostavno opisuje sposobnost pridruživanja brojnih referentnih točaka. Ona ne opisuje direktno topologiju arhitekturnih komponenata koje su se koristile u sastavljanju podmreže. Općenito, svaka podmreža može se podijeliti u manje podmreže povezane linkovima.

Logički put je transportni entitet koji se formira povezivanjem zaključenja logičkog puta s mrežnom konekcijom i koji predstavlja pojedinačnu instanciju kapaciteta sloja mreže. Mrežna konekcija je instancija kapaciteta najveće podmreže definirane u sloju mreže. Na isti način, kao što je moguće dijeliti podmrežu, moguće je dijeliti i mrežnu konekciju. Općenito, mrežnu konekciju moguće je dijeliti u serijsku kombinaciju podmrežnih konekcija i link konekcija. Svaka podmrežna konekcija može biti dalje podijeljena u serijsku kombinaciju podmrežnih konekcija i link konekcija. Podjela mrežnih konekcija i podmrežnih konekcija odražavat će i podjelu podmreža. Kad se mrežna konekcija u potpunosti dekomponira u sastavne link konekcije i podmrežne konekcije, svaka se link konekcija na klijent sloju može smatrati apstraktnim transportnim entitetom sastavljenim od adaptacijske funkcije i logičkog puta na server sloju.

2.2.2 Koncept slojevitosti

Koncept slojevitosti se temelji na odnosu klijent/server između dva sloja mreže. Link konekcija u klijent sloju mreže se opskrbljuje s logičkim putem u server sloju mreže (slika 2.1). Pri tome svaki sloj ima vlastite mogućnosti rada i održavanja. U osnovnoj klasifikaciji transportna mreža se sastoji od tri sloja, a podjela se temelji na njihovim funkcijama:

- *Sloj kanala* (krugova, vodova) je mreža kanala (logičke veze između čvorova, tj. veze s kraja na kraj) koja zadovoljava tražene usluge od pojedinih korisnika. Različite mreže sloja kanala su definirane u skladu s ponuđenim vrstama usluga (npr. mreže s komutacijom kanala, mreže s komutacijom paketa, mreže iznajmljenih linija).
- *Sloj puteva* je mreža sastavljena od više kanala koji se uobičajeno nazivaju putem. Namjena ovog sloja je da podrži različite vrste mreža sloja kanala, dakle da osigura transport za sloj kanala. Uspostava puta se obavlja na zahtjev iz mreže kanala. U slučaju *SDH* razlikujemo dva sloja mreže puteva i to: niži red puteva *LO* (*LO* - *Lower Order*) i viši red puteva *HO* (*HO* – *Higher Order*).
- *Sloj transmisijskih medija* je mreža koja stavlja na raspolaganje transmisijski kapacitet između čvorova u mreži puteva, i to na njihov zahtjev. Ovaj sloj se dalje

može dijeliti na mrežu sloja sekcija i mrežu sloja fizičkih medija. Nadalje, sloj sekcije se može podijeliti na sloj mutipleksnih sekcija i sloj regeneratorskih sekcija.

Transportne funkcije i transportni entiteti opisuju transport informacija u transportnoj mreži. Svaka transportna funkcija, odnosno transportni entitet, su odvojeni referentnim točkama. Funkcije transportne mreže djeluju na prisutne informacije na ulazu ili ulazima i prikazuju procesirane informacije na izlazu ili izlazima. One su definirane i karakterizirane informacijskim procesom između ulaza i izlaza. Dvije transportne funkcije (ili entiteti) koje slijede jedna iza druge su spojene referentnom točkom. Pri tome referentna točka spaja izlaz jedne transportne funkcije (ili entiteta) i ulaz druge transportne funkcije (ili entiteta).

2.3 ARHITEKTURA TRANSPORTNE MREŽE

Osnovne komponente arhitekture transportne mreže su:

- *Komponente topologije* koje osiguravaju apstraktni opis mreže u smislu topološkog odnosa između skupova sličnih referentnih točaka,
- *Transportni entiteti* koji osiguravaju transparentni prijenos informacija kroz sloj mreže,
- *Transportne funkcije*
- *Referentne točke*.

Topološke komponente omogućavaju opis logičke topologije mreže. Mrežu je moguće u potpunosti opisati sa slojem mreže, podmrežom i linkom.

- *Sloj mreže* se definira kao potpuni skup sličnih *pristupnih točaka sloja AP* (*AP – Access Point*, tip referentne točke) koje mogu biti pridružene zbog prijenosa informacije. Svaki sloj mreže nosi jedan tip karakteristične informacije (brzina, kodiranje, veličina okvira). Proces upravljanja sloja mreže uspostavlja ili raskida vezu između pristupnih točaka i na taj način se mijenja njihova povezanost.

- *Podmreža* je određena konačnim skupom sličnih *konekcijskih točaka CP* (*CP – Connection Point*, tip referentne točke) koje mogu biti pridružene radi prijenosa karakteristične informacije. Upravljanje sloja može uspostaviti ili raskinuti vezu između konekcijskih točaka pridruženih podmreži. Na taj se način mijenja povezanost podmreže. Podmreže se obično sastavljaju od nižih razina (manjih) podmreža i linkova između njih. Najniža razina ove rekurzije, koja je interesantna u mrežnoj arhitekturi, je matrica sadržana u pojedinom mrežnom elementu (u fizičkom pogledu to su npr. centrala ili prospojnik kao fleksibilni elementi, ili digitalni razdjelnici kao fiksni elementi). Pod pojmom *mrežni element* (NE – *Network element*) podrazumijevamo grupu ili dio telekomunikacijske opreme koja može komunicirati sa upravljačkim sustavom.

- *Link* je određen podskupom konekcijskih točaka u jednoj podmreži, koji je pridružen podskupu konekcijskih točaka druge podmreže zbog prijenosa karakterističnih informacija između podmreža. Upravljanje sloja ne može uspostaviti ili raskinuti skup konekcijskih točaka pridruženih određenom linku. Link predstavlja topološku vezu između para podmreža. Općenito, link se koristi

da bi se opisalo pridruživanja između konekcijskih točaka sadržanih u jednom mrežnom elementu s konekcijskim točkama u drugome. Najnižu razinu linka u postupku rekurzije predstavlja transmisijski medij.

Transportni entiteti osiguravaju transparentan prijenos informacija između referentnih točaka sloja mreže. To znači da nema promjene informacija između ulaza i izlaza, osim onih koje su uzrokovane smetnjama i degradacijom u postupku prijenosa. U ovisnosti da li se nadgleda integritet prenesene informacije, razlikujemo dva osnovna entiteta: *logički put* (trail) i *konekciju C* (*C – Connection*). Konekcija se, s obzirom na topološke komponente kojima pripada, dalje dijeli na *mrežnu konekciju NC* (*NC – Network Connection*), *podmrežnu konekciju SNC* (*SNC – Sub-Network Connection*) i *link konekciju LC* (*LC – Link Connection*).

- *Logički put* predstavlja prijenos valjanih karakterističnih informacija između pristupnih točaka, zajedno s dodanom informacijom koja osigurava integritet prijenosa informacije. Logički put se formira od mrežne konekcije dodavanjem funkcije zaključenja logičkog puta između *TCP*-ova i pristupnih točaka.

- *Mrežna konekcija* je sposobna transparentno prenositi informaciju kroz sloj mreže. Ona je omeđena *zaključnim točkama konekcije TCP* (*TCP* – Termination Connection Point). To je najviša razina apstrakcije u sloju mreže i ona može biti podijeljena u lanac podmrežnih konekcija i link konekcija. Ne postoji informacija o integritetu prenesenih informacija, već se informacija o integritetu same mrežne konekcije može doznati iz drugih izvora.

- *Podmrežna konekcija* je sposobna prenositi informaciju kroz podmrežu. Ona je omeđena točkama konekcije (CP – Connection Point). Podmrežne konekcije su, općenito, sastavljene od lanca podmrežnih konekcija i link konekcija niže razine te se mogu prikazati kao njihova apstrakcija. Najniža razina ove rekurzije je matrična konekcija koja prikazuje npr. prospajanje u odgovarajućem mrežnom elementu.

- *Link konekcija* (*LC*) je sposobna transparentno prenositi informaciju kroz link između dvije podmreže. Ona je omeđena *CP*-ovima na granici linka i podmreže te prikazuje pridruživanje između takvog para *CP*-ova.

Adaptacija i zaključenje logičkog puta su dvije osnovne transportne procesne funkcije u arhitekturi transportne mreže. One se pojavljuju na granici slojeva i procesiraju informacije između odgovarajućih ulaza i izlaza.

- *Funkcija zaključenja logičkog puta T* (*T* - Trail Termination) osigurava informaciju koja omogućuje očuvanje integriteta logičkog puta prilikom prijenosa. To se postiže umetanjem dodatne informacije na predajnoj strani (Source, slanje signala), zaključenjem logičkog puta i nadgledanjem na prijemnoj strani (Sink, primanje signala).

- *Funkcija adaptacije A* (*A* - Adaptation) predajnog signala (source) je proces u kojem se karakteristična informacija klijent sloja mreže adaptira u formu prikladnu za prijenos u serverskom sloju mreže. Komplementarna funkcija obnavljanja informacije funkcije adaptacije je na prijemnoj strani (sink). Specifične funkcije adaptacije ovise o karakterističnim informacijama dva sloja

između kojih se adaptacija izvodi. Neki od mogućih primjera funkcija adaptacije su: kodiranje, promjena brzine prijenosa, multipleksiranje.

Referentna točka je komponenta arhitekture transportne mreže koja se formira povezivanjem izlaza (sink) jedne transportne funkcije ili transportnog entiteta s ulazom (source) druge transportne funkcije ili transportnog entiteta. Ona je dvosmjerna ako su odgovarajući pridruženi izlazi i ulazi transportnih funkcija i transportnih entiteta upareni. Ova veza se nikada ne proteže izvan mrežnog elementa. Primjeri referentnih točaka su zaključna točka konekcije *TCP*, točka konekcije *CP* i pristupna točka *AP*.

2.4 UPRAVLJANJE TRANSPORTNOM MREŽOM (TMN)

Svrha TMN-a (eng. Telecomunications Management Network) je podrška administraciji telekomunikacijske mreže u upravljanju svim elementima mreže. TMN je konceptualno odvojena mreža koja sučeljava telekomunikacijsku mrežu radi prijema informacija i kontrole operacija u mreži. Međutim, TMN često koristi dijelove telekomunikacijske mreže za svoju komunikaciju.

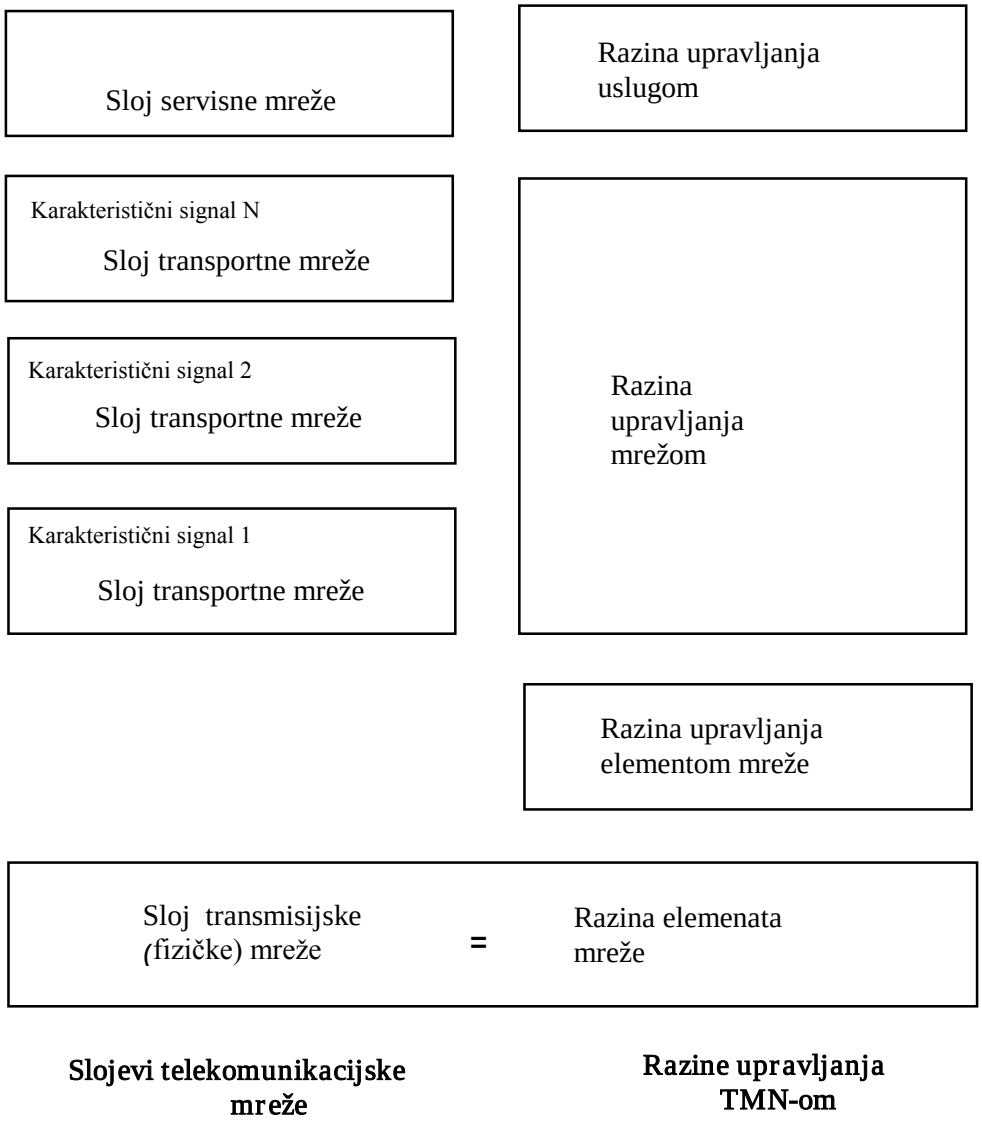
Razina upravljanja mrežom (TMN) odnosi se u biti na upravljanje transportnom mrežom. I ovdje je bitan uvjet upravljiva transportna mreža. Ne može se govoriti o upravljanju transportnom mrežom ako uređaji transmisijske mreže nisu upravljivi. Prema tome, upravljanje transportnom mrežom ovisi o tome da li su u transmisijskoj mreži implementirani uređaji i oprema koji omogućavaju upravljanje. *SDH* mreža, *ATM* mreža i neki elementi *PDH* mreže su primjeri upravljivih transportnih mreža. Sama transportna mreža može imati više slojeva (definiranih različitim karakterističnim signalima prijenosa) i sukladno tome više razina upravljanja.
Razina upravljanja uslugom bavi se ugovorenim aspektima usluga koje su ponuđene korisniku ili su dostupne potencijalnim novim korisnicima.

Na slici 2.2 prikazan je odnos slojeva telekomunikacijske mreže i razina upravljanja u TMN-u. Ovaj rad opisuje upravljanje funkcionalnošću transportnog sloja, koja se u TMN-u definira kao razina upravljanja mrežom i razina upravljanja elementom mreže.

TMN je distribuirani informacijski sustav, što znači da postoji mreža računala u čijim su čvorovima smještene odgovarajuće baze podataka s opisima upravljanih objekata. Upravljani objekti su informacijski prikazi fizičkih ili logičkih resursa telekomunikacijske mreže nad kojima se izvodi proces upravljanja. Nad upravljanim objektima se izvršavaju određene operacije koje se moraju odraziti i na stvarne resurse. Objekti se u TMN-u definiraju pod pojmom *managed objects (MO)*, ili *network element (NE)*. Pojam mrežnog elementa (eng. network element ili NE) će biti osnov za opis svih elemenata mreže, te će se u ovom radu veoma često koristiti. Njegova definicija biti će specificirana s obzirom na ograničenja i ciljeve rada u kasnijim poglavljima.

Dakle, u bazama podataka mora u svakom trenutku biti prikazana realna slika stanja telekomunikacijske mreže. Često akcije nad određenim upravljanim objektima uzrokuju akcije nad drugim upravljanim objektima koji su smješteni u drugom čvoru, dakle i drugom računalu. Zbog toga se mora osigurati i efikasna komunikacija za

potrebe samog TMN-a, pri čemu se koriste resursi mreže kojom se i upravlja. Treba naglasiti da proces upravljanja telekomunikacijskom mrežom, promatran sa stanovišta te mreže, mora biti neprimjetan i ne smije narušavati njen rad.



Slika 2.2 – Odnos između slojeva telekomunikacijske mreže i razina upravljanja u TMN-u

2.5 ANALIZA OBJEKTNOG SUSTAVA TRANSPORTNE MREŽE

2.5.1 UVOD

Analiza objektnog sustava obuhvaća informacijsku i funkcijsku analizu procesa i elemenata u transportnoj mreži. Ciljani informacijski sustav obuhvaća opis upravljanja konfiguracijom SDH i PDH mreže.

Odgovorne osobe zadužene za stanje telekomunikacijske mreže žele u svakom trenutku imati relevantne podatke o transportnoj mreži kako bi mogli poduzimati neophodne upravljačke akcije. Analiza objektnog sustava uključuje pronalaženje

entiteta (objekata) koji opisuju zadane procese, a koji moraju biti usklađeni s međunarodnim standardima i preporukama i treba rezultirati u objektima koji će mogu osigurati upravljanje konfiguracijom. Pritom treba imati na umu da sustav mora biti tako koncipiran da omogući modularno dodavanje elemenata koji će osigurati i ostale kategorije upravljanja (npr. upravljanje pogreškama, performansama, sigurnošću) kada se za to steknu potrebe i uvjeti.

2.5.2 POLAZIŠTA, CILJEVI I OGRANIČENJA

Osnovni cilj ovog rada je razvoj informacijskog sustava za opis transportne mreže, koji obuhvaća nadzor i upravljanje statičkom konfiguracijom digitalne mreže sa SDH i PDH elementima. Pri tome će se opisivati samo procesi u transportnoj mreži koji se obavljaju iza komutacije. S obzirom na stanje današnje tehnologije, pri opisu transportne mreže moraju se obuhvatiti električka i optička te elementi sveoptičke mreže.

Polazne postavke za razvoj informacijskog sustava su TMN te Generički model transportne mreže koji su obuhvaćeni odgovarajućim preporukama.. Polazni elementi koji proizlaze iz Generičkog modela su:

- Komponente topologije (mreža, podmreža, link)
- Transportni entiteti (logički put, konekcija)
- Transportne funkcije (adaptacija, terminacija)
- Referentne točke
- Principi slojevitosti i podjele

Polazni element koji proizlazi iz definicija i preporuka TMN-a je *mrežni element.* Mrežni element je osnovna jedinica opisa upravljanja transportne mreže. Opis mrežnog elementa mora osim upravljanja transportnom mrežom omogućiti opis funkcionalnosti mreže. Precizna analiza i definicija mrežnog elementa nije dana ni u samom TMN-u. Stoga će detaljna analiza objektnog sustava dati odgovor na pitanje kako najpreciznije i najsvrsishodnije definirati mrežni element za potrebe opisa konfiguracije i upravljanja transportne mreže.

Upravljanje statičkom konfiguracijom mreže je podijeljeno na:

- upravljanje temeljnom nefleksibilnom statičkom konfiguracijom
- upravljanje temeljnom fleksibilnom statičkom konfiguracijom.

Temeljna nefleksibilna statička konfiguracija definirana je mrežnim elementima i njihovim međusobnim povezivanjem kod početnog konfiguriranja (izgradnje) mreže. Ona je nepromjenjiva do onog trenutka dok u mrežu ne instaliramo novi uređaj ili zamjenjujemo postojeći. Prema tome, temeljna statička konfiguracija je nefleksibilna u smislu da uspostavljene temeljne konekcije ne možemo mijenjati do trenutka kad zbog novog uređaja ne vršimo određenu rekonfiguraciju mreže.

Zahvaljujući novim rješenjima u današnjoj transmisijskoj tehnici temeljna fleksibilna statička konfiguracija dozvoljava određeni stupanj fleksibilnosti. Fleksibilnost osiguravaju matrične konekcije koje možemo prospojiti u odgovarajućim prospojnicima. U slučaju potrebe moguće je promjenom matričnih

konekcija rekonfigurirati mrežu. S obzirom da se taj postupak ne provodi često, odnosno vrijeme odziva sustava kod uspostavljanja takvih konekcija je relativno dugo, ta se konfiguracija također može smatrati statičkom.

Koristeći navedene principe i elemente iz TMN-a i Generičkog modela, uz ograničenja kojima su definirane granice budućeg informacijskog sustava, mogu se definirati ciljevi informacijskog sustava u cjelini. Razvijeni informacijski sustav korisniku mora omogućiti:

- opis mreže po svakom sloju (koristeći princip slojevitosti mreže),
- opis mreže po administrativnim područjima (koristeći princip podjele mreže),
- opis prospojenosti kroz mrežu za svaki logički put, odnosno mrežnu konekciju (koristeći postavke i definicije logičkog puta kao transportnog entiteta iz Generičkog modela),
- opis svih logičkih putova kroz neki mrežni element (koristeći definicije transportnih entiteta iz Generičkog modela i mrežnog elementa koji proizlazi iz TMN-a),
- prikaz strukture svakog mrežnog elementa,
- prikaz strukture pojedinog uređaja i transportnog sloja,
- prikaz kapaciteta mreže (slobodnih, iskorištenih, instaliranih),
- prikaz odgovarajućeg transmisijskog sloja i uređaja u njemu.

U postupku oblikovanja baze podataka korištena je metodologija formalnog opisivanja elemenata, strukture i topologije telekomunikacijske mreže koja je razvijena na Zavodu za telekomunikacije FER-a [9].

Njene su bitne karakteristike sljedeće:

- opisi transmisijske i transportne mreže predstavljaju u matematičkom smislu povezani graf (multigraf) u kojem su objekti prikazani čvorovima, granama i njihovom kombinacijom,
- topologija mreže sadržana je u identifikatoru pojedinog objekta (entiteta),
- odnos između pojedinih objekata i njihovih svojstava izražavaju se funkcijskim i višeznačnim ovisnostima, koji se formalno prikazuju u odgovarajućim konceptualnim modelima podataka,
- u analizi odnosa između pojedinih objekata koristi se objektni pristup,
- postupak implementacije obuhvaća prevođenje konceptualnog modela u relacijski logički model podataka,
- integracija dodanog znanja u bazi podataka ostvaruje se raspoloživim alatima nad kreiranom bazom podataka.

Konačni rezultat je inteligentna baza u kojoj su podaci formirani na potpun, prirodan i neredundantan način. Integracija dodanog znanja osigurava konzistentnost i integritet baze podataka pri prvom punjenju i kasnijim ažuriranjima.

Postizanje ovih ciljeva ovisi o pravilnom i preciznom definiranju entiteta koji su bitni za opis transportne mreže. Prvi korak u tom pravcu je definiranje osnovnih objekata sustava transportne mreže.

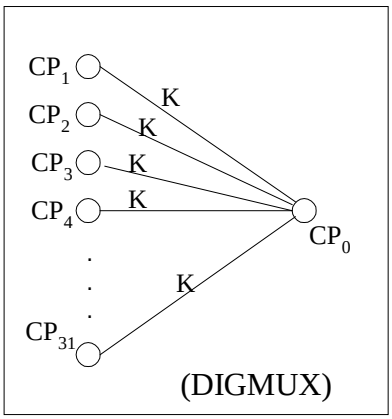
2.5.3 OSNOVNI ELEMENTI OBJEKTNOG SUSTAVA TRANSPORTNE MREŽE

Transportni sloj mreže se naziva i logičkim slojem jer opisuje povezanost mreže na logičkom nivou. Cilj je transportne mreže prenijeti informaciju bez obzira na sadržaj, lokaciju korisnika koji komuniciraju i upotrijebljene resurse. No, kako bi na efikasniji način mogli upravljati transportnom mrežom, nužno je definirati elemente koji će povezivati (logički) transportni nivo s fizički stvarnim elementima u mreži. To znači da se moraju definirati korisniku razumljivi elementi topologije mreže. Oni se moraju prilagoditi opisima i definicijama koji služe kao polazište za definiranje sustava. Pri tome treba stalno imati na umu da su transportne funkcije zapravo slika funkcija koje obavlja neki fizički uređaj u telekomunikacijskoj mreži.

U ovom poglavlju bit će ukratko opisani osnovni elementi transportne mreže koji proizlaze iz početnih postavki, prošireni za elemente opisa topologije mreže. Detaljna analiza mreže sa SDH i PDH elementima dati će detaljan opis svih ovdje navedenih elemenata. Pri analizi odnosa elemenata korišten je “bottom-up” princip jer je ocijenjeno da se hijerarhijske ovisnosti između pojedinih elemenata mreže mogu na takav način efikasnije uočiti i opisati.

Kao osnovni element transportne mreže definiran je ***mrežni element.*** Mrežni element je osnovna jedinica koja koja je specificirana odgovarajućom transportnom funkcijom koju obavlja. Njegovi unutarnji aspekti i način ostvarivanja u potpunosti su u rukama proizvođača. Svaki mrežni element transportnog sloja se na fizičkom sloju predstavlja kao konkretni uređaj ili dio nekog uređaja. Statički konfigurirati mrežni element u transportnu mrežu znači smjestiti ga zajedno s uređajem kojem pripada u odgovarajući transportni čvor i povezati ga s ostalim elementima mreže (npr. drugim mrežnim elementima, uređajima, transportnim čvorovima) po jednoznačno definiranom postupku. Mrežni element se prikazuje konekcijama i referentnim točkama (slika 2.3).

Konekcija je temeljni element za opis povezanosti i funkcionalnosti transportne mreže. Funkcija transportne mreže se prikazuje konekcijom, a obavlja se u mrežnom elementu. Konekcije se grubo mogu podijeliti na link konekcije i podmrežne konekcije. Prikaz funkcije koja se obavlja u mrežnom elementu je **temeljna podmrežna konekcija**. Ona povezuje ulaznu temeljnu referentnu točku s izlaznom temeljnom referentnom točkom unutar jednog mrežnog elementa. ***Temeljne referentne točke*** se definiraju unutar mrežnog elementa. U ITU preporukama referentne točke se definiraju kao točke kojima se zaključuje konekcija. Prema tome, mrežni element se može prikazati temeljnim referentnim točkama i temeljnim podmrežnim konekcijama. **Temeljne link konekcije** opisuju povezanost mrežnih elemenata, uređaja, transportnih čvorova i na kraju čitave mreže. One povezuju izlaznu referentnu točku jednog mrežnog elementa sa ulaznom referentnom točkom sljedećeg mrežnog elementa.

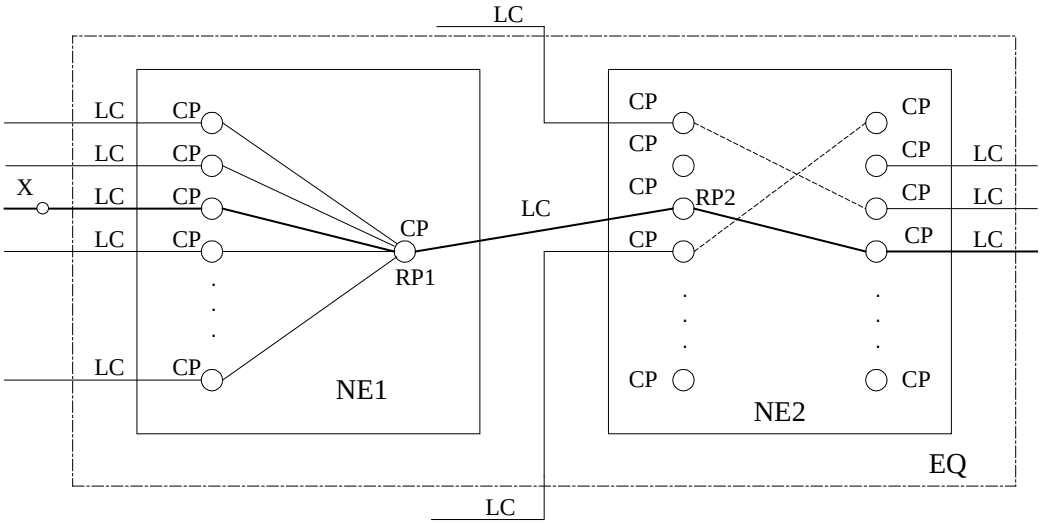


Slika 2.3 – Prikaz mrežnog elementa

Na slici 2.3 grafički je prikazan mrežni element digitalni multipleksor. Prikazane su 31 ulazna i jedna izlazna referentna točka (CP) i temeljne podmrežne konekcije (K) kojima se opisuje funkcija multipleksiranja.

Uređaj je element koji u topološkoj hijerarhiji prikaza transportne mreže predstavlja viši hijerarhijski nivo od mrežnog elementa, konekcija i referentnih točki. Sastoji se od mrežnih elemenata (opisanih temeljnim referentnim točkama i temeljnim podmrežnim konekcijama) povezanih temeljnim link konekcijama. Uređaj se fizički može identificirati putem inventarnog broja ili sl. Uređaj dakle povezuje logički transportni nivo prikaza sa stvarnim fizičkim uređajem telekomunikacijske mreže. Smješta se u transportni čvor. Povezivanje dva uređaja obavlja se također temeljnim link konekcijama. Na slici 2.4 prikazana su dva mrežna elementa povezana u jedan uređaj. Uređaj je označen sa oznakom EQ a mrežni elementi sa NE1 i NE2. Referentne točke su označene sa CP.

Na slici 2.4 je prikazana i temeljna link konekcija koja spaja dva mrežna elementa (tj. spaja referentne točke označene sa RP1 i RP2). Ta je temeljna link konekcija označena sa LC. Vidljivo je da su i ostale konekcije koje se nalaze van mrežnih elemenata označene sa LC, jer su to također temeljne link konekcije. Te temeljne link konekcije povezuju prikazane mrežne elemente sa drugim mrežnim elementima koji se ne nalaze unutar uređaja na slici.



Slika 2.4 – Prikaz uređaja sa dva mrežna elementa

Transportni čvor se sastoji od uređaja povezanih temeljnim link konekcijama. Kao topološki entitet na višem hijerarhijskom nivou prikaza nalazi se na određenoj lokaciji. Pridjeljuje se određenoj mreži. Transportni čvorovi se također povezuju temeljnim link konekcijama.

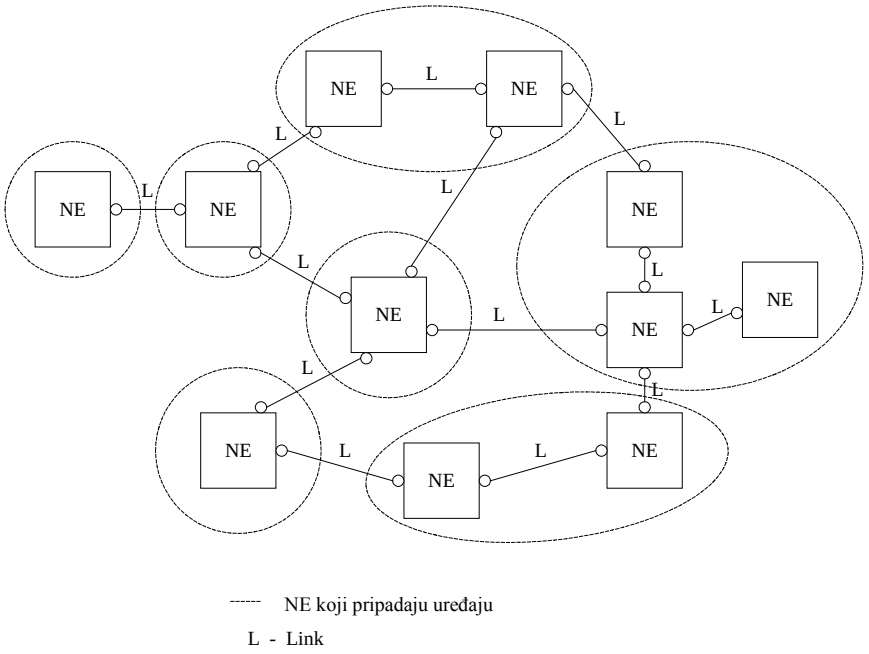
Mreža se sastoji od jednog ili više transportnih čvorova. Kao element na višem hijerarhijskom stupnju nasljeđuje vlasništvo nad svim elementima transportne mreže koji se nalaze unutar transportnih čvorova. Dvije mreže povezuju se temeljnim link konekcijama. Mreže se nalaze unutar mreže višeg hijerarhijskog nivoa. Na taj način možemo definirati po volji velik broj hijerarhijskih nivoa mreža. Mreže se povezuju temeljnim link konekcijama. Temeljna link konekcija može povezivati mreže na različitim hijerarhijskim nivoima pa je potrebno definirati hijerarhijski nivo temeljne link konekcije koja povezuje dvije mreže. To će biti učinjeno u detaljnoj analizi mreže i konceptualnom modeliranju.

Logički put predstavlja komunikacijski put kroz kojim se razmjenjuju informacije između dva korisnika. Na nivou najvišeg detalja logički put možemo prikazati kao slijed temeljnih referentnih točki ili slijed temeljnih podmrežnih i temeljnih link konekcija.

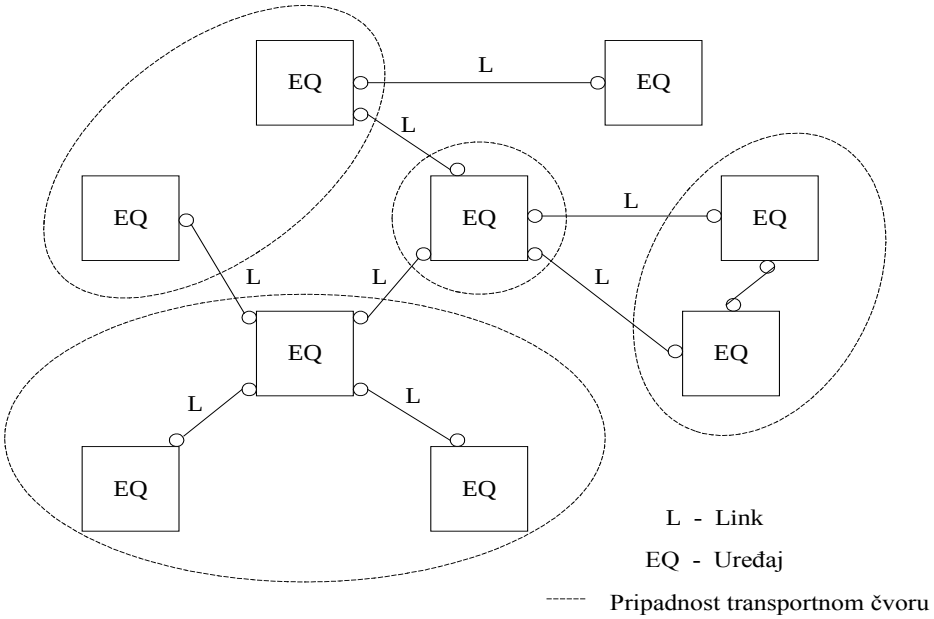
Na slikama 2.5 do 2.7 prikazana je sinteza mreže od razine najnižeg detalja do razine mreže. Slika 2.5 prikazuje povezivanje mrežnih elemenata u uređaje. Slika 2.6 prikazuje povezivanje uređaja u transportne čvorove, slika 2.7 prikazuje povezivanje transportnih čvorova u podmreže, a povezivanje različitih nivoa mreže prikazano je na slici 2.8.

Na slikama 2.5 do 2.8 zorno su prikazana topologija mreže na raznim hijerarhijskim razinama. Pri tom je postignuto da se za svaki hijerarhijski nivo prikaza mreža može prikazati kao skup grana i čvorova. Bitno je napomenuti da se opisom hijerarhijskih odnosa u topologiji mreže korisniku može omogućiti da sam definira po

volji velik broj razina mreže i to prema vlastitoj definiciji tih hijerarhijskih nivoa. To znači da se npr. operater u Hrvatskoj može odlučiti na kreiranje županijskih mreža, regionalnih i nacionalnih mreža, dok se npr. operater u Švicarskoj može odlučiti za generiranje kantonalnih i nacionalnih podmreža.



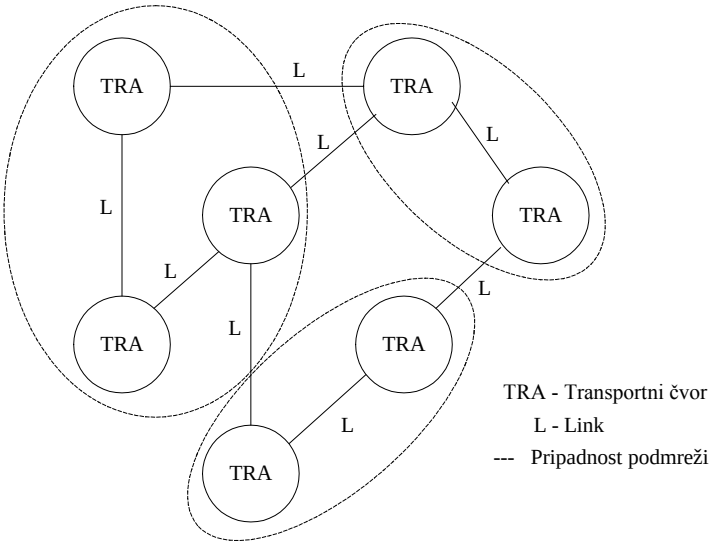
Slika 2.5 – Povezivanje mrežnih elemenata



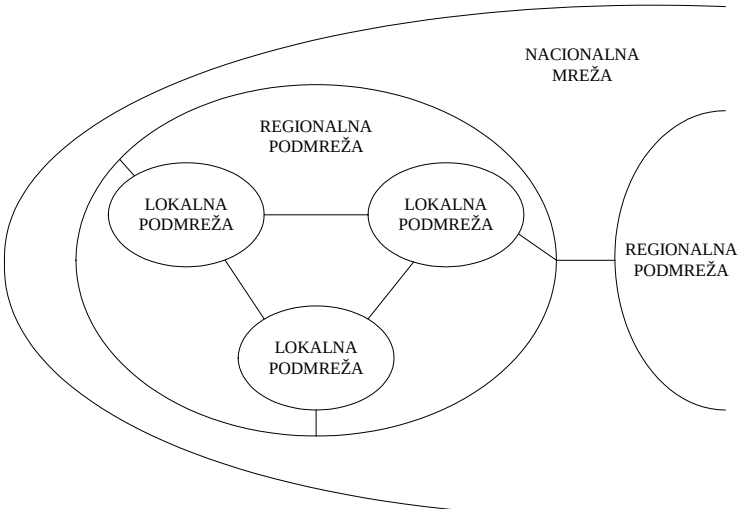
Slika 2.6 – Povezivanje uređaja

Iz slika 2.5 do 2.8 vidi se da je osnovna metodologija opisa sačuvana. Objekti mreže su čvorovi (mrežni elementi, uređaji, transportni čvorovi) i grane (temeljne link i podmrežne konekcije). U danom trenutku mreža se može top-down metodom

prikazati sa najviše hijerarhijske razine (prikaz mreža i podmreža) do najniže hijerarhijske razine temeljne referentne točke, link i podmrežne konekcije. Detaljna analiza mora dati entitete i atribute kojima će se na jednoznačan način opisati prikazi mreže na raznim hijerarhijskim nivoima.



Slika 2.7 – Povezivanje transportnih čvorova



Slika 2.8 – Povezivanje različitih nivoa mreža

3. TRNET (*Transport Network Database*)

3.1 Konceptualni i implementacijski model

3.1.1 DEFINIRANJE ENTITETA

Temeljni cilj konceptualnog modela je da na temelju rezultata iz procesa analize definira entitete i veze između entiteta. Rezultat konceptualnog modela je cjelovit, neredundantan prikaz informacijskog sustava. Za razvoj konceptualnog modela baze podataka informacijskog sustava transportne mreže korišten je Model entiteti-veze (ER).

Na temelju dosad izvršene analize, definiranih slojeva i funkcija mreže u skladu sa ITU preporukama identificirani su niže navedeni osnovni entiteti transportne mreže. Osim naziva entiteta i funkcije koju obavljaju, dan je i engleski naziv za entitet, jer je čitav informacijski sustav razvijan tako da bude razumljiv široj znanstvenoj zajednici [3], [7]:

a. **MREŽNI ELEMENT (**NETWORK ELEMENT, NE**),**
b. **BROADCAST PROSPAJANJE** (MPXC, multipoint cross-connect definira broadcast prospajanje);
c. **PROSPAJANJE GRUPE** (GTP, group termination point, definira prospajanje grupe);
d. **REFERENTNA TOČKA** (REFERENCE POINT, identificira i definira referentnu točku);
e. **TEMELJNA PODMREŽNA KONEKCIJA** (BASIC SUBNETWORK CONNECTION, definira temeljne podmrežne konekcije unutar mrežnog elementa);
f. **TRANSPORTNI SLOJ** (LAYER, definira slojeve u mreži);
g. **RP_WAVELENGHT (**SKUP VALNIH DULJINA, definira skupove valnih duljina u mreži);
h. **UREĐAJ** (EQUIPMENT, definira uređaj);
i. **TEMELJNA LINK KONEKCIJA** (BASIC LINK CONNECTION, definira temeljne link konekcije, osim tipa LC);
j. **TRANSPORTNI ČVOR** (NODE, definira čvor);
k. **LOKACIJA** (LOCATION, definira lokaciju u kojoj se čvor nalazi);
l. **MREŽA** (NETWORK, definira mrežu);
m. **LINK KONEKCIJA (**LINK CONNECTION, definira temeljnu link konekciju tipa LC);
n. **HIJERARHIJSKI NIVO (**HIERARCHY LEVEL, definira dozvoljene hijerarhijske razine u mreži);
o. **LOGIČKI PUT** (TRAIL, definira logički put);
p. **TRANSPORTNI PUT** (PATHWAY, definira transportni put);
q. **ELEMENT TRANSPORTNOG PUTA** (PATHWAY ELEMENT, definira element transportnog puta).

Definirano je da se konekcije i referentne točke nalaze unutar mrežnog elementa. To znači da egzistencija referentnih točki i konekcija ovise o mrežnom elementu. Stoga se odnosi mrežnog elementa, referentne točke i konekcije kao najvažnijih elemenata za opis transportne mreže mogu definirati na sljedeći način:

Mrežni element *kao osnovni element za opis transportne mreže smatra se* ***egzistencijalno i identifikacijski jakim entitetom***. ***Temeljne referentne točke i temeljne podmrežne konekcije*** *definiraju kao* ***egzistencijalno i identifikacijski slabi entiteti***.

Od navedenih entiteta, za opis transportne mreže kao egzistencijalno i identifikacijski jaki entiteti su definirani MREŽA, TRANSPORTNI ČVOR, UREĐAJ, MREŽNI ELEMENT, LINK KONEKCIJA. Između pojavnosti ovih entiteta ipak postoje stanovita ograničenja koja osiguravaju jednoznačnost postupka konfiguriranja mreže. Iako su u našem modelu transportne mreže entiteti REFERENTNA TOČKA, TEMELJNA PODMREŽNA KONEKCIJA i TEMELJNA LINK KONEKCIJA izuzetno značajni u smislu prikaza najdetaljnijeg opisa transportne mreže i prikaza najfinije topologije, njihovu pojavnost smo opisali kao egzistencijalno i identifikacijsko slabe entitete.

Da bismo mogli prikazati topologiju mreže na svim hijerarhijskim razinama mreže (nacionalna, regionalna, lokalna…), izabrali smo entitete TRANSPORTNI ČVOR i MREŽA. Entitet MREŽA opisan je u generičkom modelu kao osnovni entitet za prikaz nivoa mreže. U našem je modelu takav prikaz proširen pa je entitet MREŽA povezan i sa topološkim karakteristikama mreže. Entitet TRANSPORTNI ČVOR ima višestruku ulogu. S jedne strane povezivanje transportnih čvorova putem LINK KONEKCIJA opisuje topologiju mreže, dok s druge strane vezom tipa "sadrži" s entitetom UREĐAJ omogućuje se i kontinuitet prikaza odnosa u mreži na nižoj hijerarhijskoj razini. Konačno, preko entiteta LOKACIJA bit će moguće izvršiti povezivanje sa GIS informacijskim sustavima budućeg složenog prikaza mreže. Prema tome, za početak opisa konfiguracije mreže potrebna je pojava barem jednog entiteta TRANSPORTNA MREŽA na najnižem hijerarhijskom nivou jer mora biti poznata mreža za koju se opisuje konfiguracija (naravno, ukoliko postoje, mogu se opisati i pojave svih ostalih mreža za koje želimo opisati konfiguraciju, kao i njihovi hijerarhijski odnosi). Pojava entiteta TRANSPORTNI ČVOR zahtijeva da bude poznata mreža kojoj pripada.

Neka od svojstava MREŽNIH ELEMENATA su zajednička za sve NE, dok su neka svojstva specifična samo za pojedini tip mrežnog elementa. Zajednička svojstava svih mrežnih elemenata opisana su atributima u entitetu MREŽNI ELEMENT. Svojstva pojedinog tipa mrežnog elementa koja su bitna za modeliranje transportne mreže, a o kojima ćemo pohranjivati podatke u bazu podataka, definiramo odgovarajućim atributima za svaki tip mrežnog elementa. Entiteti koji rezultiraju iz ovakvog načina opisa mrežnih elemenata (prema tipovima) su:

- **MULTIPLEKSOR** (MULTIPLEKSOR, **NE_MUX,** funkcija adaptacije na server sloj),
- **DEMULTIPLEKSOR** (DEMULTIPLEKSOR, **NE_DEMUX,** funkcija adaptacije sa server sloja)
- **OHI** (OVERHEAD INSERTION, **NE_OHI,** funkcija terminacije - dodavanja zaglavlja),
- **OHE (**OVERHEAD EXTRACTION, **NE_OHE,** funkcija terminacije - izdvajanja zaglavlja),
- **ELEKTRIČKI PROSPOJNIK** (ELECTRICAL CROSS-CONNECT, **NE_XC,** funkcija prospajanja električkih signala),
- **ELEKTRIČKI DROP/INSERT** (ELECTRICAL DROP/INSERT, **NE_ED/I,** funkcija izdvajanja ili dodavanja električkog signala),
- **ELEKTRIČKI LINIJSKI TERMINAL** (ELECTRICAL LINE TERMINAL, **NE_ELT,** funkcija adaptacije signala na liniju),
- **REGENERATOR** (REGENERATOR, **NE_REG,** funkcija obnavljanja električkog signala),
- **PRETVORNIK SIGNALA** (CONVERTER, **NE_CONV,** funkcija adaptacije - pretvorbe električkog signala u optički, i obrnuto),
- **SPREŽNIK** (COMBINER, **NE_COMB,** optički multipleksor, funkcija adaptacije),
- **RASPREŽNIK** (SPLITTER, **NE_SP,** optički demultipleksor, funkcija adaptacije),
- **OPTIČKI PROSPOJNIK** (OPTICAL CROSS-CONNECT, **NE_OXC,** funkcija prospajanja optičkih signala),

- **OPTIČKI LINIJSKI TERMINAL** (OPTICAL LINE TERMINAL, **NE_OLT,** funkcija adaptacije),
- **OPTIČKI DROP/INSERT** (OPTICAL DROP/INSERT, **NE_OD/I,** funkcija izdvajanja ili dodavanja optičkog signala),
- **OPTIČKO POJAČALO** (OPTICAL FIBER AMPLIFIER, **NE_OFA,** funkcija pojačavanja optičkog signala),

Rezultat provedenog postupka je pokazao da je struktura transportne mreže izuzetno složena, što se u konačnici manifestira kompleksnim odnosima između identificiranih entiteta. Identificirani skup atributa i entiteta osigurava sve potrebne informacije za efikasan postupak nadzora i upravljanja konfiguracijom transportne mreže. Međutim, zbog relativno siromašnih semantičkih osobina relacijskog modela u kojem će se implementirati baza podataka, potrebno je ekspertno znanje o transportnoj mreži efikasno ugraditi u navedeni relacijski model putem odgovarajućih procedura, trigera i pravila, a algoritmima u razvijenim aplikacijama postići da postupak unosa, ažuriranja i pretraživanja podataka osigura integritet i točnost pohranjenih i prikazanih informacija. Baza podataka će zajedno sa ugrađenim znanjem i razvijenim aplikacijama omogućiti da Informacijski sustav o transportnoj mreži postane ekspertni sustav za konfiguriranje, održavanje i praćenje rada transportne mreže.

3.1.2 MODEL ENTITETI-VEZE (E-R MODEL)

Odgovornost projektanta baze podataka je da definirajući entitete realno prikaže odnose iz vanjskog svijeta koji se preslikavaju u bazu podataka. Osnovni entiteti koji proizlaze iz detaljne analize transportne mreže opisani su u prethodnom paragrafu. Također su definiranjem nekih međusobnih odnosa naznačene i veze koje postoje između pojedinih entiteta. Dobivene informacije o strukturi podataka uobličene su u E-R dijagram transportne mreže. Model entiteti–veze (eng. Entity-Relationship diagram, skraćeno E-R dijagram) dat će potpunu sliku svih entiteta te veza između njih. Nazivi entiteta i atributa dani su na engleskom jeziku, jer su i u modelu radi univerzalnosti svi entiteti (buduće relacije) i atributi imenovani na engleskom jeziku.

Model entiteti–veze koristi entitet i vezu kao dva osnovna objekta. Velika odlika modela entiteti-veze je mogućnost dobrog grafičkog prikaza. Nacrtani dijagram entiteti-veze (eng. Entity-Relationship diagram, skraćeno E-R dijagram) omogućuje bolje razumijevanje podataka. U praksi se najviše koriste Chenov i Martinov grafički prikaz dijagrama entiteti-veze. Ja ću u daljnjem opisu koristiti Chenov dijagram [12].

Zbog kompleksnosti prikaza E-R dijagram je podijeljen na tri dijela. Oni prikazuju pojedine funkcionalne dijelove transportne mreže: topološku statičku konfiguraciju, fleksibilnu statičku konfiguraciju i logičke putove, te raščlambu mrežnih elemenata. Svaki od navedenih dijelova je specifičan i bit će detaljno opisan. Potpuna slika transportne mreže dobiva se povezivanjem sva tri E-R dijagrama. Tri E-R dijagrama su prikazani odvojeno isključivo zbog preglednosti i čine jednu cjelinu.

Dijagram na slici 3.1 prikazuje entitete kojima se opisuje temeljna statička karakteristika mreže. Entitet mreža (NETWORK) može sadržavati ili biti unutar (“nested” odnos) druge mreže, što znači da svaka mreža može sadržavati podmrežu (u hijerarhijskom smislu). Svaka se mreža nalazi na određenom hijerarhijskom nivou. Svi hijerarhijski nivoi moraju se definirati prije definiranja mreža i podmreža. Na taj

način mogu se uspostaviti jasni hijerarhijski odnosi između mreža, definirajući unaprijed po volji velik broj nivoa mreže (međunarodna, nacionalna, regionalna, lokalna, itd.).

Unutar mreže najniže hijerarhijske razine nalaze se transportni čvorovi (NODE). Smatramo da svi transportni čvorovi koji se nalaze u mreži niže hijerarhijske razine pripadaju i nadređenim mrežama više hijerarhijske razine. Npr., ako se čvor TCZagreb1 nalazi u mreži GradZagreb, tada se istovremeno nalazi i u mreži RHrvatska, ukoliko je mreža RHrvatska nadređena mreži GradZagreb. Mreža na najvišoj hijerarhijskoj razini ne nalazi se unutar neke druge mreže, pa je odnos mreža-mreža tipa 0,1:1,M.

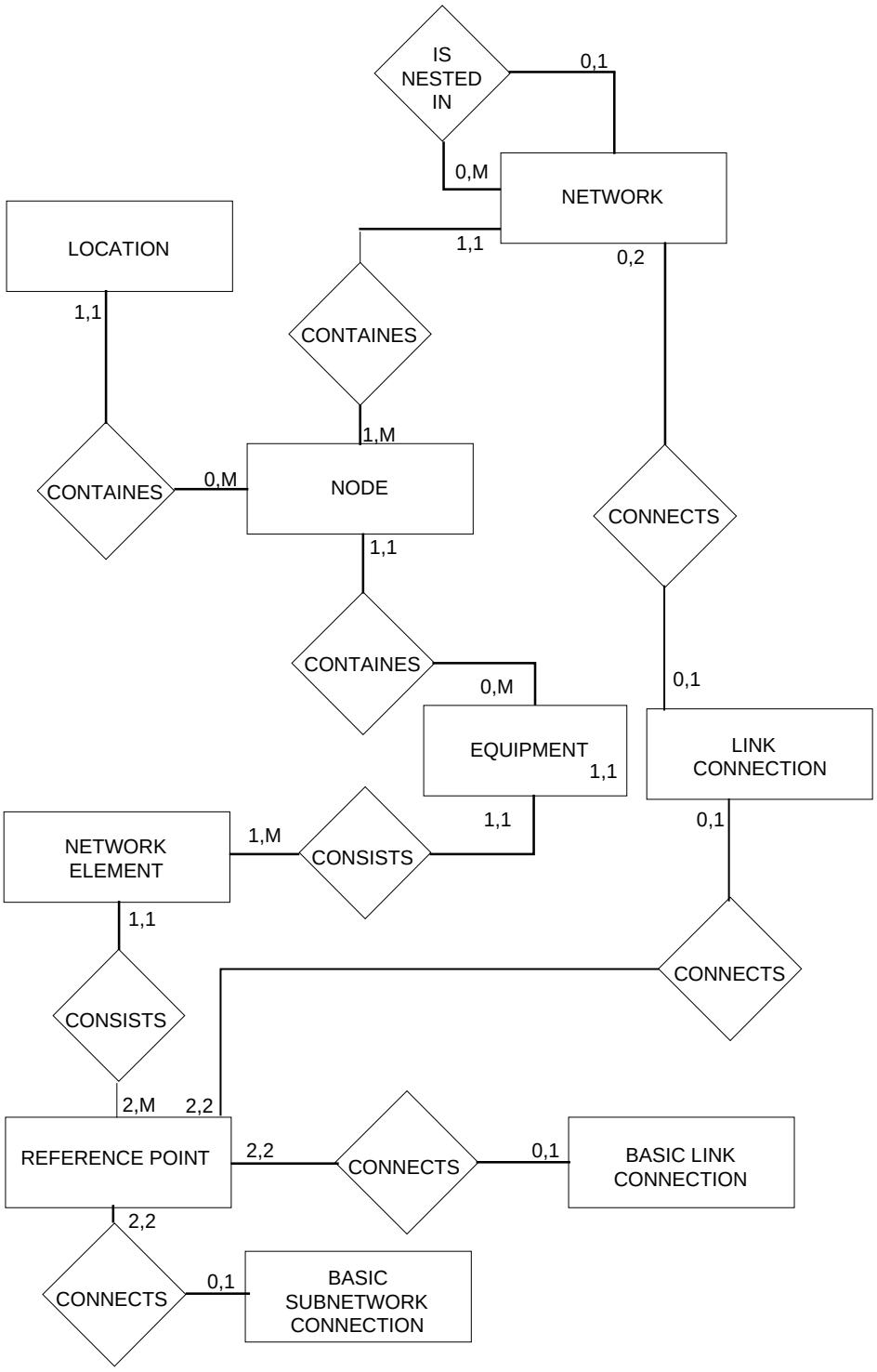
Transportni čvor je definiran unutar mreže. Povezan je s entitetom lokacija (LOCATION) koji određuje zemljopisni pojam. Transportni čvor se nalazi samo na jednoj lokaciji, dok lokacija može sadržavati više čvorova. Lokacija je entitet koji definira sam korisnik.

Oprema, tj. uređaj (EQUIPMENT), nalazi se u određenom transportnom čvoru (a time i u određenoj lokaciji).

Uređaj ima značajnu ulogu pri opisu topologije mreže i funkcionalnosti transportnog sloja mreže. Mrežni element (NETWORK ELEMENT), kao osnova opisa transportne mreže, je smješten u uređaju.

Referentna točka (REFERENCE POINT) je definirana kao ulazna ili izlazna referentna točka za pojedini mrežni element. Svaki mrežni element mora imati dvije ili više referentnih točki, dok se svaka referentna točka mora nalaziti samo u jednom mrežnom elementu.

Temeljna podmrežna konekcija (BASIC SUBNETWORK CONNECTION) opisuje vezu između ulazne i izlazne referentne točke unutar mrežnog elementa. Ovim se entitetom opisuju temeljne podmrežne konekcije unutar mrežnih elemenata tipa NE_OXC, NE_XC, NE_ED/I te NE_OD/I. U ovim se mrežnim elementima ne može jednoznačno odrediti veza između ulaznih i izlaznih referentnih točaka pa je te konekcije potrebno opisati posebnim entitetom. Temeljnim podmrežnim konekcijama tipa MC (koje se nalaze u NE_XC i NE_OXC) definira se statička fleksibilna konfiguracija mreže. Specifičnost opisa NE_OD/I i NE_ED/I je u tome što se osim Source i Sink referentnih točaka jedini od svih tipova mrežnih elemenata moraju opisivati dodatnim tipovima referentnih točki. To su referentne točke u kojima se signal "vadi" iz dolaznih signala za lokalni promet (***Drop*** tip referentne točke), te referentne točke u kojima se signal "umeće" na nastala ispražnjena mjesta (***Insert*** tip referentne točke). Temeljnim podmrežnim konekcijama tipa KS-SS (konekcija spajanja *Source* i *Sink* referentne točke), KS-IS (konekcija spajanja *Insert* i *Sink* referentne točke) te KS-SD (konekcija spajanja *Source* i *Drop* referentne točke) spajaju se referentne točke unutar NE_ED/I i NE_OD/I.



Slika 3.1 – E-R dijagram entiteta za opis statičkog dijela transportne mreže

Temeljna link konekcija (BASIC LINK CONNECTION) opisuje veze između dviju referentnih točki i to između referentnih točki koje se nalaze u dva različita mrežna elementa. Na taj način dobijamo potpunu "sliku" ponude trasnportne mreže. Npr., povezivanjem uređaja unutar transportnog čvora konekcijama tipa LLC (Local Link Connection – Lokalna Link Konekcija) dobijamo strukturu transportnog čvora.

Referentna točka omeđuje dvije konekcije, dok se jednom konekcijom uvijek omeđuju dvije i samo dvije referentne točke.

Zbog mogućnosti postavljanja fizičkog medija u realnu telekomunikacijsku mrežu prije kreiranja transportnih čvorova, pa čak i prije kreiranja samih mreža i podmreža, definiran je entitet LINK_CONNECTION kao egzistencijalno i identifikacijski jak entitet. Link konekcija povezuje dvije mreže, a hijerarhijski nivo konekcije definiran je hijerarhijski najnižom mrežom koja je zajednička dvjema mrežama koje link konekcija spaja. Link konekcija može spajati dvije referentne koji se nalaze unutar jedne mreže. Tada je hijerarhijska razina link konekcije jednaka hijerarhijskoj razini te mreže. Na taj način može se opisati povezanost mreže na bilo kojoj hijerahijskoj razini i željenom nivou detalja.

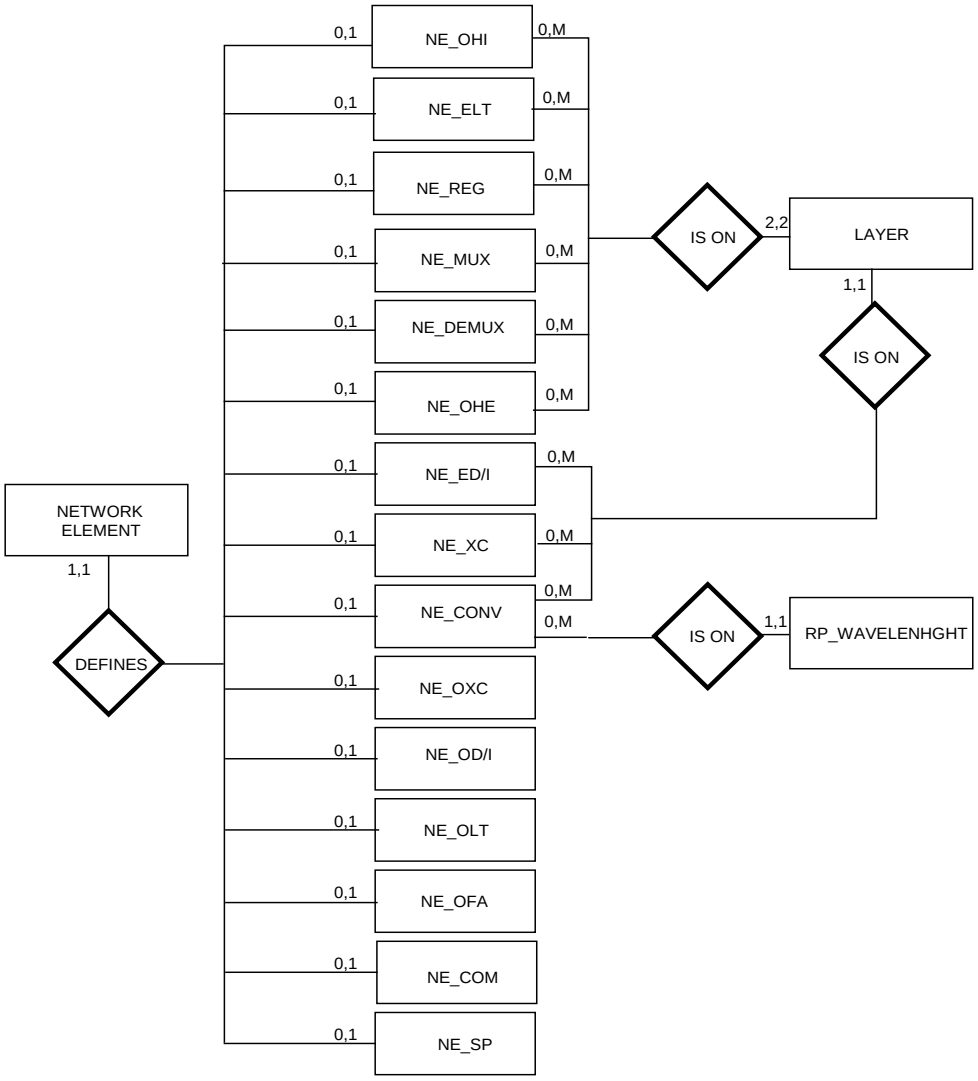
Sliku 3.1 treba promatrati samo kao dio cjelovitog E-R dijagrama za opis konceptualnog modela baze podataka transportne mreže. Slika 3.2 proširuje sliku 3.1 opisujući sve podtipove mrežnog elementa. Odnos između nadtipa entiteta NETWORK_ELEMENT i odgovarajućih podtipova spada u kategoriju ekskluzivne neobvezatne specijalizacije. Svi podtipovi entiteta NETWORK_ELEMENT su prikazani na slici 3.2. Oni predstavljaju proširenje općih karakteristika mrežnih elemenata opisanih u tipu entiteta NETWORK_ELEMENT na svaki od podtipova mrežnih elemenata. Npr. zajednička svojstva za svaki mrežni element opisana su u tipu entiteta NETWORK_ELEMENT, no ako se želi dobiti određena specifična svojstva za npr. multipleksor, onda se ona nalaze zapisana u podtipu entiteta NE_MUX.

Kao što je vidljivo na slici 3.2, dijagram daje definiran odnos između mrežnog elementa kao nadtipa i 15 podtipova mrežnih elemenata. Odnos između mrežnog elementa i svakog od podtipa tog entiteta je isti: svaki podtip entiteta mora biti opisan jednim entitetom mrežni element, dok svaki mrežni element opisuje jedan podtip mrežnog elementa.

Entitet LAYER definira slojeve mreže. Slojevi mreže su definirani prema brzini prijenosa, te se može precizno odrediti njihova karakteristika. Stoga su električki mrežni elementi definirani na točno određenim slojevima.

Mrežni elementi sa slike 3.2 mogu se prema tipu podijeliti na električke i optičke, s konvertorom (NE_CONV) koji vrši prilagodbu s optičkog na električki signal i obrnuto, a kojeg prema dogovoru smatramo optičkim elementom.

Podtipovi mrežnih elemenata koje smo do sada definirali ne moraju predstavljati konačan skup podtipova, no pri eventualnom proširenju novim podtipovima iz dijagrama na slici 3.2 postaje jasan princip njihova dodavanja. Dodavanje novog podtipa mrežnog elementa je jednostavno zato što su sve opće karakteristike već opisane u entitetu NETWORK_ELEMENT. Opisuju se samo specifične karakteristike za novi podtip. Novi se podtip entiteta s entitetom NETWORK_ELEMENT povezuje istim načinom kao i svi ostali već postojeći podtipovi mrežnih elemenata. Skup podtipova mrežnih elemenata koje smo do sada definirali u potpunosti zadovoljava potrebe nadzora i upravljanja konfiguracijom mreže.



Slika 3.2 – E-R dijagram entiteta i pod-tipova entiteta mrežnog elementa

Karakteristike koje dodatno opisuju podtipove entiteta mrežni element su:

a) Električki podtipovi mrežnih elemenata:

1. Multipleksor (**NE_MUX**) : tip multipleksora (PDH, SDH), slojevi (ulazni i izlazni)
2. Demultipleksor (**NE_DEMUX**): tip demultipleksora (PDH, SDH), slojevi (ulazni i izlazni)
3. Električki Drop/Insert (**NE_ED/I**): sloj mreže, broj *Source*, *Sink* te *Drop* i *Insert* referentnih točaka
4. Električko pojačalo – regenerator (**NE_REG**): visina pojačanja, ulazni i izlazni sloj mreže, karakteristične vrijednosti pojačanja
5. Električki linijski terminal (**NE_ELT**): ulazni i izlazni sloj mreže, funkcija (da li prilagođava signal s linije, odnosno na liniju, tj. da li je ulazni ili izlazni mrežni element transportnog čvora), karakteristične vrijednosti ELT
6. Električki prospojnik (**NE_XC**): sloj, kapacitet prospajanja, broj ulaznih referentnih točki, da li prospojnik može vršiti funkcije prospajanja tipa point-to-

point, point-to-multipoint i group-to-group, te da li prospojnik služi za zaštitno prospajanje signala
7. Dodavanje zaglavlja (overhead insertion- **NE_OHI**): slojevi (ulazni i izlazni)
8. Izdvajanje zaglavlja (overhead extraction- **NE_OHE**): slojevi (ulazni i izlazni)

b) Optički tipovi mrežnih elemenata:
8. Optički prospojnik (**NE_OXC**): kapacitet prospajanja, broj ulaznih referentnih točki, da li prospojnik može vršiti funkcije prospajanja tipa point-to-point, point-to-multipoint i group-to-group, te da li prospojnik služi za zaštitno prospajanje signala
10. Optički linijski terminal (**NE_OLT**): funkcija (da li prilagođava signal s linije, odnosno na liniju, tj. da li je ulazni ili izlazni mrežni element transportnog čvora), karakteristične vrijednosti
11. Optičko pojačalo (**NE_OFA**): visina pojačanja, karakteristične vrijednosti
12. Optički Drop/Insert (**NE_OD/I**): broj *Source* i *Sink*, te broj *Drop* i *Insert* referentnih točaka
13. Rasprežnik (**NE_SP**): broj referentnih točaka
14. Sprežnik (**NE_COM**): broj referentnih točaka
1. Pretvornik (**NE_CONV**): sloj električkog signala, valna duljina optičkog, tip pretvorbe (optičko-električka ili obrnuto).

Na slici 3.3 opisan je statički fleksibilan dio transportne mreže s prikazom entiteta za opis logičkog puta.

Dijagram na slici 3.3 se može podijeliti na dva dijela: na dio koji uključuje entitete i odnose između entiteta potrebnih za opis logičkog puta, te na dio kojim se opisuje fleksibilna statička konfiguracija mreže. Entitet referentna točka je prikazan i na slici 3.1, jer se referentnim točkama definira i statička konfiguracija tj. ponuda mreže. Kako je već navedeno i opisano, opisujući statičku fleksibilnu konfiguraciju mreže pomoću referentnih točaka koristimo entitete koji su već prethodno definirani (referentne točke). Time smanjujemo broj zapisa u bazi podataka.

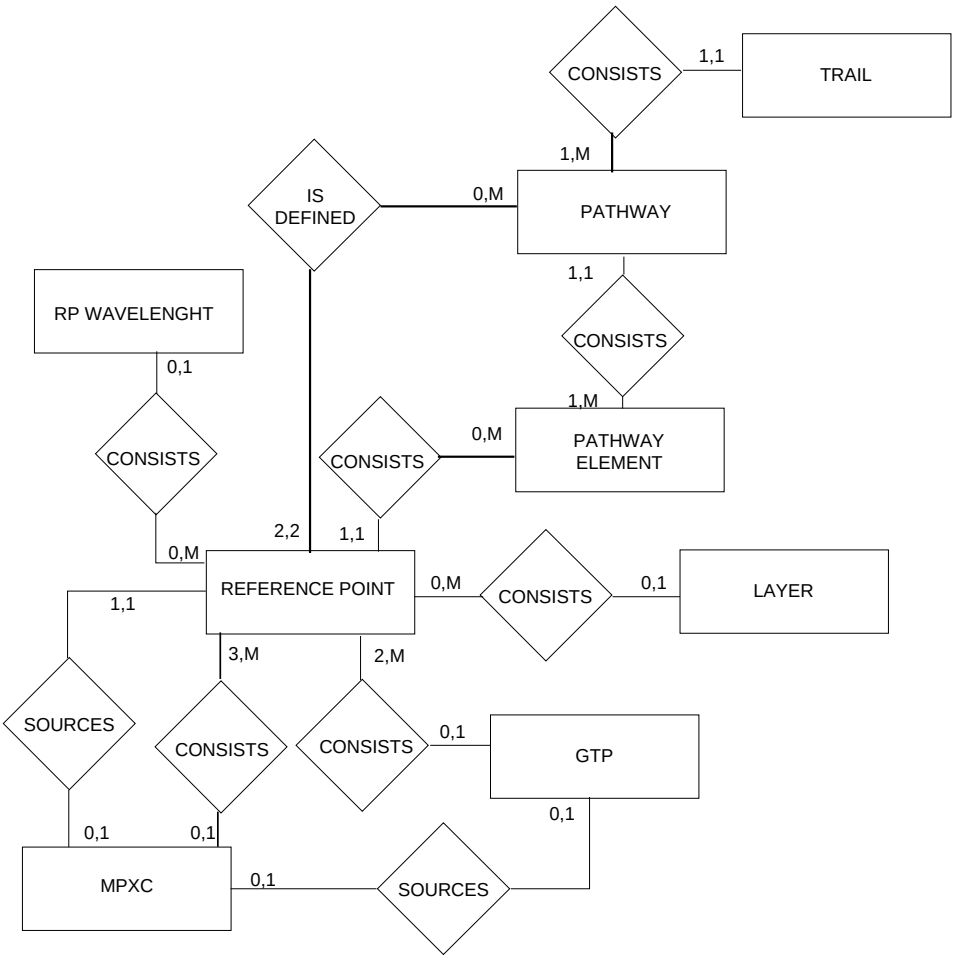
Referentne točke su npr. temelj za opis MPXC entiteta. Zajednička Source referentna točka MPXC entiteta je bitna za opis point-to-multipoint prospajanja. Mogu se dobiti sve referentne točke koje su spojene na Source referentnu točku. Na taj način su indirektno određeni svi entiteti BASIC_SUBNETWORK_CONNECTION koji pripadaju dotičnom MPXC-u. Naime, veza između entiteta MPXC i entiteta BASIC_SUBNETWORK_CONNECTION postoji samo preko entiteta REFERENCE_POINT.

Referentna točka se može nalaziti unutar nekog GTP, ili može biti Source za neki MPXC. Referentna točka u električnom podtipu mrežnog elementa nalazi se na točno definiranom sloju transportne mreže.

GTP mora sadržavati barem dvije referentne točke, a MPXC mora imati definiranu Source referentnu točku. Ukoliko se radi o broadcast prospajanju grupe referentnih točaka, MPXC mora imati definiran Source GTP. Veza između GTP i MPXC entiteta je redundantna u smislu da se njihova ovisnost može izvesti preko referentnih točaka, tj. preko veze MPXC i REFERENCE POINT te preko veze između REFERENCE POINT i GTP. Redudantna veza je eliminirana u daljem postupku oblikovanja Modela entiteti-veze korištenjem Designer2000.

Entitet RP_WAVELENGHT definira skup valnih duljina koje prolaze referentnom točkom.

Logički put (TRAIL) se definira pomoću slijeda referentnih točki. Entiteti PATHWAY i PATHWAY_ELEMENT služe da bi se što bolje opisao logički put na transportnom sloju mreže. Oni su prilagođeni za efikasno dobijanje informacija o logičkom putu.



Slika 3.3 – E-R dijagram entiteta za opis fleksibilne statičke konfiguracije mreže

Logički put opisujemo s jednim ili više PATHWAY-a. Jedan PATHWAY pripada samo jednom logičkom putu, a definira se između dvije referentne točke.

Entitet PATHWAY_ELEMENT opisuje svaku od referentnih točaka kroz koje prolazi PATHWAY. Svaka pojava entiteta PATHWAY_ELEMENT opisuje samo jednu od referentnih točaka kroz koju prolazi PATHWAY. Isto tako, kroz jednu referentnu točku može prolaziti više PATHWAY-a. Svaki PATHWAY_ELEMENT pripada samo jednom PATHWAY-u.

Bitno je naglasiti da se za opis PATHWAY-a može koristiti i PATHWAY na server sloju kojim se služi PATHWAY na klijent sloju.

3.1.3 RELACIJSKE SHEME BAZE PODATAKA TRANSPORTNE MREŽE

ER model prikazan na slikama 3.1 do 3.3 pretvoren je u logički relacijski model baze podataka. Dobiveni rezultati u obliku relacijskih shema prikazani su na slijedeći način:

1. Najprije je neveden naziv entiteta. U zagradi pored naziva entiteta dan je hrvatski naziv.
2. Uz nazive atributa su navedene i dozvoljene vrijednosti za pojedine entitete (domenski integritet), kao i tip podataka (integer, string, varchar …).
3. Na kraju je označena relacijska shema za svaki pojedini tip entiteta. Masno otisnutim slovima označen je primarni ključ relacijske sheme, dok je strani ključ podcrtan.
4. Za atribut koji je ujedno i strani ključ naznačena je relacijska shema i atribut koji su izvor ovog stranog ključa, i to u obliku "RELACIJSKA_SHEMA.ATRIBUT".
5. Tip podataka naznačen kao *sequence1, sequence2,…* zapravo je ORACLE prilagođeni inkrementirajući cijeli broj koji će sustav nakon fizičke implementacije automatski dodjeljivati svakoj sljedećoj relaciji, a oznaka 1,2 .. uz naziv sequence označava da se radi o različitim sequencama, tj. da se koriste različite sequence za atribute u različitim relacijskim shemama.

Slijedi popis relacijskih shema:

1. NETWORK (MREŽA)

a. ***NETWORK_ID*** - identifikator mreže; sequence1
b. ***NETWORK_HIERARCHY_LEVEL*** – hijerarhijska razina mreže; string(20)
c.
d. ***SUPERIOR_NETWORK_ID*** – identifikator mreže više hijerarhijske razine kojoj mreža pripada; number
e. ***ADMINISTRATION*** – naziv administracije mreže; string (20)
f. ***NETWORK_LABEL*** – korisnički naziv mreže; string (20)

NETWORK ***(NETWORK_ID,*** *NETWORK_HIERARCHY_LEVEL,* *SUPERIOR_NETWORK_ID, ADMINISTRATION, NETWORK_LABEL*)

2. NODE (TRANSPORTNI ČVOR)

a. ***NODE_ID*** – identifikator čvora; sequence2
b. ***NETWORK_ID*** – identifikator mreže kojoj transportni čvor pripada (NETWORK.NETWORK_ID); number
c. ***LOCATION***– naziv lokacije u kojoj se čvor nalazi (LOCATION.LOCATION); string(15)
d. ***NODE_LABEL*** – korisnički naziv transportnog čvora; string (20)

NODE ***(NODE_ID,*** *NODE_LABEL, NETWORK_ID, LOCATION*)

3. LOCATION (LOKACIJA)

b. ***LOCATION*** – korisnički određen naziv lokacije, ujedno i identifikator, string(15)

c. ***X_COORDINATE*** **-** koordinata lokacije; number
d. ***Y_COORDINATE*** - koordinata lokacije; number
e. ***Z_COORDINATE*** - koordinata lokacije; number

LOCATION ***(LOCATION,*** *X_COORDINATE****,*** *Y_COORDINATE, Z_COORDINATE*)

4. EQUIPMENT (OPREMA, UREĐAJ)

a. ***EQ_ID*** - identifikator uređaja; sequence3
b. ***OPERATIONAL_ STATE*** - vrijednosti: "enabled" ili "disabled", govori da li je uređaj ispravan; string(8)
c. ***ADMINISTRATIVE_STATUS*** - vrijednosti mogu biti: "in use" ili "not in use", govori da li se uređaj koristi ili ne; string(10)
d. ***EQ_TYPE*** – naziv tipa uređaja; string (15)
e. ***LABEL –*** naziv uređaja, najčešće prema nazivu kako ga deklarira proizvođač; string(20)
f. ***NODE_ID*** **-** definira transportni čvor u kojem se uređaj nalazi (NODE.NODE_ID); string(15)

EQUIPMENT ***(EQ_ID,*** *OPERATIVE_STATE, ADMINISTRATIVE_STATUS, LABEL, EQ_TYPE, NODE_ID*)

5. NETWORK ELEMENT (MREŽNI ELEMENT)

a. ***NE_ID*** - identifikator mrežnog elementa; sequence4
b. ***ADMINISTRATIVE_STATUS*** **-** vrijednosti mogu biti: "in use" ili "not in use", govori da li se mrežni element koristi ili ne; string(10)
c. ***AVAILABILITY_STATUS*** - vrijednosti: "available" ili "not available", govori da li je NE na raspolaganju ili ne; string(13)
d. ***OPERATIONAL_STATE*** - vrijednosti: "enabled" ili "disabled", govori da li je mrežnom elementu omogućen rad ili ne; string(8)
e. ***TYPE*** - vrijednosti: "MUX", "DEMUX", "SP", "COM", "OFA", "OHI", "OHE", "ED/I", "OD/I", "XC", "OXC", "LT", "OLT", "REG", "CONV"; string(5)
f. ***EQ_ID*** - identifikator uređaja u kojem se NE nalazi (EQUIPMENT.EQ_ID); number
g. ***SIGNAL_TYPE*** - vrijednost može biti "optical", "electrical", "optical-electrical" ili "electrical-optical". Vrijednosti "optical-electrical" ili "electrical-optical" će atribut poprimiti ako se radi o NE_CONV; string(18)

NETWORK_ELEMENT ***(NE_ID,*** *SIGNAL_TYPE, AVAILABILITY_STATUS, OPERATIONAL_STATE, ADMINISTRATIVE_STATUS, TYPE, EQ_ID)*

6. MULTIPLEXER (MULTIPLEKSOR)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number

b. ***HIGHER_LAYER_ID*** – identifikator klijent sloja signala (ulaz u NE_MUX), (LAYER.LAYER_ID); number
c. ***LOWER_LAYER_ID*** – identifikator server sloja signala (izlaz iz NE_MUX), (LAYER.LAYER_ID); number
d. ***NUMBER_OF_RP*** - broj referentnih točaka, uključujući AP; integer
e. ***MUX_TYPE*** – vrijednosti mogu biti "PDH" ili "SDH"; string(3)

NE_MUX ***(NE_ID,*** *HIGHER_LAYER_ID, LOWER_LAYER_ID, NUMBER_OF_RP, MUX_TYPE*)

7. DEMULTIPLEXER (DEMULTIPLEKSOR)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***HIGHER_LAYER_ID*** - identifikator klijent sloja signala (izlaz iz NE_DEMUX), (LAYER.LAYER_ID); number
c. ***LOWER_LAYER_ID*** – identifikator server sloja signala (ulaz u NE_DEMUX), (LAYER.LAYER_ID); number
d. ***NUMBER_OF_RP*** - broj referentnih točaka, uključujući AP; integer
e. ***DEMUX_TYPE*** – vrijednosti mogu biti "PDH" ili "SDH"; string(3)

NE_DEMUX ***(NE_ID,*** *HIGHER_LAYER_ID, LOWER_LAYER_ID, NUMBER_OF_RP, DEMUX_TYPE*)

8. OHI (IZDVAJANJE ZAGLAVLJA)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***INPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na ulazu u NE_OHI, (LAYER.LAYER_ID); number
c. ***OUTPUT_SIGNAL_LAYER_ID*** - identifikator sloja signala transportne mreže na izlazu iz NE_OHI, (LAYER.LAYER_ID); number

NE_OHI ***(NE_ID,*** *INPUT_SIGNAL_LAYER_ID, OUTPUT_SIGNAL_LAYER_ID)*

9. OHE (UMETANJE ZAGLAVLJA)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***INPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na ulazu u NE_OHE, (LAYER.LAYER_ID); number
c. ***OUTPUT_SIGNAL_LAYER_ID*** - identifikator sloja signala transportne mreže na izlazu iz NE_OHE, (LAYER.LAYER_ID); number

NE_OHE ***(NE_ID,*** *INPUT_SIGNAL_LAYER_ID, OUTPUT_SIGNAL_LAYER_ID)*

10. XC (ELEKTRIČKI PROSPOJNIK)

- ***a. NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
- ***b. XC_LAYER_ID***- identifikator sloja na kojem se obavljaju prospajanja, (LAYER.LAYER_ID); number
- ***c. NUMBER_OF_RP*** - broj referentnih točaka - definira kapacitet; integer
- ***d. NO_OF_INPUT_RP*** - definira broj ulaznih referentnih točaka, može biti promjenjiv; integer
- ***e. PROTECTION_SWITCHING*** – definira da li se radi o NE_XC za zaštitno prospajanje ili ne vrijednosti Yes/No; string(3)
- ***f. POINT_TO_POINT*** – definira da li NE_XC može obavljati prospajanja ovog tipa vrijednosti Yes/No; string(3)
- ***g. GTP*** – definira da li NE_XC može obavljati prospajanja group-to-group, vrijednosti Yes/No; string(3)
- ***h. BROADCAST*** – definira da li NE_XC može obavljati prospajanja tipa broadcast vrijednosti Yes/No; string(3)

NE_XC ***(NE_ID, XC_LAYER_ID, NUMBER_OF_RP, NO_OF_INPUT_RP, PROTECTION-SWITHCHING, POINT_TO_POINT, GTP, BROADCAST)***

11. ELECTRICAL DROP/INSERT (ELEKTRIČKI DROP/INSERT)

- ***a. NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
- ***b. ED/I_LAYER_ID*** – identifikator sloja na kojem ED/I vrši svoju funkciju, (LAYER.LAYER_ID); number
- ***c. NUMBER_OF_RP*** - broj referentnih točki; integer
- ***d. NO_OF_DROP_RP*** - broj referentnih točki funkcije *Drop* (istovremeno je to i broj referentnih točki funkcije *Insert*); integer

NE_ED/I ***(NE_ID, ED/I_LAYER_ID, NUMBER_OF_RP, NO_OF_DROP_RP)***

12. ELECTRICAL LINE TERMINAL (LINIJSKI TERMINAL)

- ***a. NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
- ***b. INPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na ulazu u NE_ELT, (LAYER.LAYER_ID); number
- ***c. OUTPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na izlazu iz NE_ELT, (LAYER.LAYER_ID); number
- ***d. LT_FUNCTION*** – definira da li je funkcija linijskog terminala prilagodba sa linije ili prilagodba na liniju, vrijednosti “FROM LINE” ili “ON LINE”; string(9)
- ***e. ELT_CHARACTERISTIC*** – opisuje karakteristične vrijednosti za ELT; string(9)

NE_ELT ***(NE_ID, LT_FUNCTION, INPUT_SIGNAL_LAYER_ID, OUTPUT_SIGNAL_LAYER_ID, ELT_CHARACTERISTIC)***

13. ELECTRICAL AMPLIFIER - REGENERATOR (ELEKTRIČKO POJAČALO – REGENERATOR)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira, number
b. ***AMPLIFICATION*** - visina i pojačanja karakteristika, string (15)
c. ***INPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na ulazu u NE_REG, (LAYER.LAYER_ID); number
d. ***OUTPUT_SIGNAL_LAYER_ID*** – identifikator sloja signala transportne mreže na izlazu iz NE_REG, (LAYER.LAYER_ID); number
e. ***AMP_CHARACTERISTIC*** – opisuje karakteristične vrijednosti; string(9)

NE_REG ***(NE_ID,*** *AMPLIFICATION, AMP_CHARACTERISTIC, INPUT_SIGNAL_LAYER_ID, OUTPUT_SIGNAL_LAYER_ID,)*

14. CONVERTER (PRETVORNIK)

a. ***NE_ID*** – identifikator mrežnog elementa, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***TYPE*** - govori da li se radi o optičko-električkoj ili električko-optičkoj pretvorbi: vrijednosti: “optical-electrical” i “electrical-optical”; string(17)
c. ***ELECTRICAL_LAYER_ID*** - identifikator sloja transportne mreže električkog signala, (LAYER.LAYER_ID); number
d. ***WAVELENGTH_ID –*** identifikator skupa valnih duljina optičkog signala u koji se vrši pretvorba (ili iz kojeg se vrši pretvorba), (WAVELENGTH.WAVELENGTH_ID); number

NE_CONV ***(NE_ID,*** *TYPE, WAVELENGTH_ID, ELECTRICAL_LAYER_ID)*

15. COMBINER (SPREŽNIK)

a. ***NE_ID*** - identifikator, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira, number
b. ***NUMBER_OF_RP*** - broj referentnih točaka, uključujući AP; integer

NE_COM ***(NE_ID,*** *NUMBER_OF_RP)*

16. SPLITTER (RASPREŽNIK)

a. ***NE_ID*** – identifikator, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira, number
b. ***NUMBER_OF_RP*** - broj referentnih točaka, uključujući AP; integer

NE_SP ***(NE_ID,*** *NUMBER_OF_RP)*

17. OPTICAL XC (OPTIČKI PROSPOJNIK)

a. ***NE_ID*** – identifikator ; strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***NUMBER_OF_RP*** - broj referentnih točaka - definira kapacitet; integer
c. ***NO_OF_INPUT_RP*** - definira broj ulaznih referentnih točaka, može biti promjenjiv, ako se referentne točke mogu fleksibilno pridjeljivati ulazu odnosno izlazu; integer
d. ***PROTECTION_SWITCHING*** – definira da li se radi o NE_OXC za zaštitno prospajanje ili ne, vrijednosti Yes/No; string(3)

e. ***POINT_TO_POINT*** – definira da li NE_OXC može obavljati prospajanja ovog tipa, vrijednosti Yes/No; string(3)
f. ***GTP*** – definira da li prospojnik može obavljati NE_OXC group-to-group, vrijednosti Yes/No; string(3)
g. ***BROADCAST*** – definira da li prospojnik može obavljati NE_OXC tipa broadcast, vrijednosti Yes/No; string(3)

NE_OXC ***(NE_ID,*** *NUMBER_OF_RP, NO_OF_INPUT_RP, PROTECTION-SWITHCHING, POINT_TO_POINT, GTP, BROADCAST)*

18. OPTICAL LINE TERMINAL (OPTIČKI LINIJSKI TERMINAL)

a. ***NE_ID*** – identifikator, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira, number
b. ***LT_FUNCTION*** – definira da li je funkcija linijskog terminala prilagodba sa linije ili prilagodba na liniju vrijednosti "FROM LINE" ili "ON LINE", string(9)
c. ***OLT_CHARACTERISTIC*** – opisuje karakteristične informacije vezane za kvalitetu NE_OLT; string(9)

NE_OLT *(**NE_ID,** LT_FUNCTION, OLT_CHARACTERISTIC)*

19. OPTICAL DROP/INSERT (OPTIČKI DROP/INSERT)

a. ***NE_ID*** – identifikator, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira, number
b. ***NUMBER_OF_RP*** - broj referentnih točki; integer
c. ***NO_OF_DROP_RP*** - broj referentnih točki funkcije *Drop* (istovremeno je to i broj referentnih točki funkcije *Insert)*; integer

NE_OD/I *(**NE_ID,** NUMBER_OF_RP, NO_OF_DROP_RP)*

20. OPTICAL FIBRE AMPLIFIER (OPTIČKO POJAČALO)

a. ***NE_ID*** – identifikator, strani ključ (NETWORK_ELEMENT.NE_ID) prema kojem se entitet referencira; number
b. ***AMPLIFICATION*** - visina i karakteristika pojačanja; string (15)
c. ***AMP_CHARACTERISTIC*** – može biti pohranjen dijagram karakteristike pojačanja; string(9)

NE_OFA *(**NE_ID,** AMPLIFICATION, AMP_CHARACTERISTIC)*

21. REFERENCE POINT (REFERENTNA TOČKA)

a. ***RP_NUMBER*** - definira redni broj referentne točke unutar NE (atribut NE_ID), dio identifikatora referentne točke; integer
b. ***NE_ID*** - definira NE u kojem se referentna točka nalazi, strani ključ (NETWORK_ELEMENT.NE_ID); number

c. ***RP_LAYER_ID*** – za električki signal identificira sloj PDH ili SDH mreže na kojem se referentna točka nalazi, (LAYER.LAYER_ID); number
d. ***RP_FUNCTION*** - definira da li se radi o *"Sink"*, *"Source"*, *"Drop"* ili *"Insert"*referentnoj točki (za NE u kojem se nalazi); string(6)
e. ***GTP_ID*** - identifikator GTPa ako je referentna točka unutar nekog GTPa, inače null, (GTP.GTP_ID); number
f. ***MPXC_ID*** - identifikator MPXCa ako je referentna točka unutar nekog MPXCa, inače null; number
g. ***RP_ROLE*** – definira hijerarhijsko mjesto referentne točke u mreži (vrijednosti mogu biti: "internal", "equipment", "node", "network-access"); string(9)
h. ***RP_STATUS*** – govori o stanju referentne točke, vrijednosti mogu biti: "active", "not active" ili "disabled", string(10)
i. ***RP_SIGNAL_TYPE*** – može biti: "optical" ili "electrical", string(10)
j. ***RP_TYPE*** – definira da li se radi o "AP", "ADP", "CP", "TCP" referentnoj točki; string(3)
k. ***WAVELENGTH_ID –*** identifikator skupa valnih duljina koje prolaze ovom točkom, null vrijednost ako točkom prolaze električki signali, (WAVELENGTH.WAVELENGTH_ID); number
l. ***TUNABLE –*** govori da li je valna duljina signala u optičkoj referentnoj točki promjenjiva ili ne, string(3)

REFERENCE_POINT ***(NE_ID, RP_NUMBER,*** *RP_LAYER, RP_FUNCTION, GTP_ID, MPXC_ID, RP_ROLE, RP_STATUS, RP_SIGNAL_TYPE, RP_TYPE, WAVELENGTH_ID, TUNABLE****)***

22. BASIC LINK CONNECTION (TEMELJNA LINK KONEKCIJA)

a. ***PREVIOUS_NE_ID*** - identifikator NE u kojem se nalazi prethodna referentna točka, dio složenog identifikatora prethodne referentne točke, dio identifikatora TEMELJNE LINK KONEKCIJE, strani ključ (REFERENCE_POINT.NE_ID); number
b. ***PREVIOUS_RP_NUMBER*** - dio identifikatora prve referentne točke koju konekcija spaja, dio identifikatora TEMELJNE LINK KONEKCIJE, strani ključ (REFERENCE_POINT.RP_NUMBER); number
c. ***NEXT_NE_ID –*** identifikator NE u kojem se nalazi druga (sljedeća) referentna točka, dio složenog identifikatora sljedeće referentne točke, dio identifikatora TEMELJNE LINK KONEKCIJE, strani ključ (REFERENCE_POINT.NE_ID); number
d. ***NEXT_RP_NUMBER*** - dio identifikatora druge referentne točke koju konekcija spaja, dio identifikatora TEMELJNE LINK KONEKCIJE, strani ključ (REFERENCE_POINT.RP_NUMBER); number
e. ***TYPE*** - može biti: "LILC","FILC","LLC","LC"; string(4)

BASIC_LINK_CONNECTION ***(PREVIOUS_NE_ID, PREVIOUS_RP_NUMBER, NEXT_NE_ID, NEXT_RP_NUMBER,*** *TYPE****)***

23. BASIC SUBNETWORK CONNECTION (TEMELJNA PODMREŽNA KONEKCIJA)

b. ***PREVIOUS_RP_NUMBER*** - identifikator prve referentne točke koju konekcija spaja, dio identifikatora TEMELJNE PODMREŽNE KONEKCIJE, strani ključ (REFERENCE_POINT.RP_NUMBER); number

c. ***NE_ID –*** identifikator mrežnog elementa u kojem se konekcija nalazi, dio identifikatora TEMELJNE PODMREŽNE KONEKCIJE, strani ključ (REFERENCE_POINT.NE_ID); number
d. ***NEXT_RP_NUMBER*** - identifikator druge referentne točke koju konekcija spaja, dio identifikatora TEMELJNE PODMREŽNE KONEKCIJE, strani ključ (REFERENCE_POINT.RP_NUMBER); number
e. ***TYPE*** - vrijednost može biti : "PP" (point_to_point), "PM" (point_to_multipoint), "GG" (group_to_group), "GM" (group_to_multipoint), "PS" (protection switching), "IS", "SS", "SD", (konekcije spajanja: IS=KS-IS, SS=KS-SS, SD=KS-SD); string(2)
f. ***ACTIVITY_STATUS*** - može biti: "active", "not active"; string(10)

BASIC_SUBNETWORK_CONNECTION ***(NE_ID, PREVIOUS_RP_NUMBER, NEXT_RP_NUMBER,*** *TYPE, ACTIVITY_STATUS****)***

24. LINK CONNECTION (LINK KONEKCIJA)

a. **LC_ID** – identifikator konekcije tipa LC; sequence5
b. **LC_HIERARCHY_LEVEL** – hijerarhijska razina konekcije LC; string(20)
c. ***PREVIOUS_NE_ID*** - identifikator mrežnog elementa prve referentne točke, (REFERENCE_POINT.NE_ID); number
d. ***PREVIOUS_RP_NUMBER*** - dio identifikatora prve referentne točke koju konekcija spaja, (REFERENCE_POINT.RP_NUMBER); number
e. ***NEXT_NE_ID –*** identifikator mrežnog elementa druge referentne točke, (REFERENCE_POINT.NE_ID); number
f. ***NEXT_RP_NUMBER*** - dio identifikatora druge referentne točke koju konekcija spaja, (REFERENCE_POINT.RP_NUMBER); number
g. ***PREVIOUS_NETWORK_ID –*** identificira mrežu iz koje promet dolazi, (NETWORK.NETWORK_ID); number
h. ***NEXT_NETWORK_ID –*** identificira mrežu u koju se promet šalje, (NETWORK.NETWORK_ID); number

LINK_CONNECTION ***(LC_ID,*** *LC_HIERARCHY_LEVEL, PREVIOUS_NE_ID, PREVIOUS_RP_NUMBER, NEXT_NE_ID, NEXT_RP_NUMBER, NEXT_NETWORK_ID, PREVIOUS_NETWORK_ID)*

25. GTP (GRUPA REFERENTNIH TOČAKA)

a. ***GTP_ID*** –identifikator, sequence6
b. ***NE_ID –*** identifikator mrežnog elementa u kojem se GTP nalazi, strani ključ (REFERENCE_POINT.NE_ID); number
c. ***NO_OF_RP –*** broj referentnih točaka koje GTP sadrži; integer
d. ***ACTIVITY_STATUS –*** stanje aktivnosti: "active" ili "not active"; string(10)

GTP ***(GTP_ID****, NE_ID, NO_OF_RP, ACTIVITY_STATUS)*

26. MPXC (MULTIPOINT VEZA)

a. ***MPXC_ID*** – identifikator MPXC-a, sequence7
b. ***SOURCE_RP_NUMBER*** - dio identifikatora referentne točke koja je Source za taj broadcast unutar NE_XC (NE_OXC), (REFERENCE_POINT.RP_NUMBER); number
c. ***NE_ID*** - identifikator NE_XC (NE_OXC) u kojem se MPXC nalazi, istovremeno je i to i dio složenog identifikatora referentne točke koja je Source za taj broadcast unutar NE XC, strani ključ (REFERENCE_POINT.NE_ID); number
d. ***NO_OF_RP*** - broj točaka koje su vezane ovom broadcast konekcijom uključujući ***SOURCE_RP_NUMBER***; integer
e. ***ACTIVITY_STATUS –*** stanje aktivnosti, može biti: active ili not active; string(10)

MPXC ***(MPXC_ID***, ***NE_ID***, *SOURCE_RP_NUMBER*, *NO_OF_RP*, *ACTIVITY_STATUS)*

27. TRAIL (LOGIČKI PUT)

a. ***TRAIL_ID*** – identifikator; sequence8
b. ***CAPACITY***- kapacitet veze između dva korisnika, u Mb; number
c. ***NO_OF_PATHWAYs*** - broj PATHWAY-a na koje se logički put razbija; integer
d. ***ACTIVITY_STATUS*** - može biti: “active”, “not active”, “not active protection”, “active protection”; string(21)
e. ***ALTERNATIVE TRAIL*** – identifikator alternativnog logičkog puta(TRAIL.TRAIL_ID); integer
f. ***TYPE*** - može biti: “unidirectional”, “bidirectional”, “broadcast”, “group_unidirectional”, “group_bidirectional”, “group_broadcast”; string(20)

TRAIL ***(TRAIL_ID,*** *CAPACITY, NO_OF_PATHWAYS, ACTIVITY_STATUS, ALTERNATIVE_TRAIL, TYPE)*

28. PATHWAY (TRANSPORTNI PUT)

a. ***PATHWAY _ID*** – dio složenog identifikatora PATHWAY-a, redni broj pathway-a unutar logičkog puta; integer
b. ***TRAIL_ID*** - identifikator logičkog puta kojem PATHWAY pripada, dio složenog identifikatora PATHWAY-a strani ključ (TRAIL.TRAIL_ID); number
c. ***QOS*** - atribut u kojem se mogu navesti neki temeljni podaci o kvaliteti tog dijela logičkog puta (pathway-a); string(20)
d. ***AP_SOURCE_RP_NUMBER*** – početna točka PATHWAY-a, dio složenog identifikatora referentne točke, (REFERENCE_POINT.RP_NUMBER); number
e. ***AP_SOURCE_NE_ID*** – identifikator NE u kojem se nalazi početna referentna točka koja pripada dotičnom PATHWAY-a, dio složenog identifikatora referentne točke, (REFERENCE_POINT.NE_ID); number
f. ***AP_SINK_RP_NUMBER*** – krajnja točka PATHWAY-a, dio složenog identifikatora referentne točke, (REFERENCE_POINT.RP_NUMBER); number
g. ***AP_SINK_NE_ID*** – identifikator NE u kojem se nalazi krajnja referentna točka koja pripada dotičnom PATHWAY-a, (REFERENCE_POINT.NE_ID); number

PATHWAY ***(PATHWAY_ID,*** ***TRAIL_ID***, *QOS, AP_SOURCE_RP_NUMBER, AP_SOURCE_NE_ID, AP_SINK_RP_NUMBER, AP_SINK_NE_ID)*

29. PATHWAY ELEMENT (ELEMENT TRANSPORTNOG PUTA)

a. ***TRAIL _ID*** – identifikator logičkog puta, dio složenog identifikatora PATHWAY_ELEMENT-a, strani ključ (TRAIL.TRAIL_ID); number
b. ***PATHWAY_ID*** – identifikator PATHWAY--a kojem je ovaj element pripadajući, dio složenog identifikatora PATHWAY_ELEMENT-a, strani ključ (PATHWAY. PATHWAY_ID); integer
c. ***SORT*** - redni broj PATHWAY_ELEMENT-a u PATHWAY- u, dio složenog identifikatora PATHWAY_ELEMENT-a; integer
d. ***RP_NUMBER*** - dio složenog identifikatora referentne točke koja pripada dotičnom PATHWAY_ELEMENT-u, (REFERENCE_POINT.RP_NUMBER); number
e. ***NE_ID*** – identifikator NE u kojem se nalazi referentna točka koja pripada dotičnom PATHWAY_ELEMENT-u, dio složenog identifikatora referentne točke, (REFERENCE_POINT.NE_ID); number
f. ***TYPE_OF_RP*** – definira klijent-server odnos, može biti: "TCP", "ADP", "CP", "AP", "BRP"; string(3)

PATHWAY ELEMENT ***(PATHWAY_ID, TRAIL_ID, SORT****, RP_NUMBER, NE_ID, TYPE_OF_RP)*

Osim ovih relacija, prema potrebama definiranim u 5. poglavlju, valja definirati i dodatne relacije koje služe kao šifrarnici, tj. definiraju domene pojedinih atributa ostalih relacija.

30. LAYER

a. LAYER_ID – identifikator sloja PDH i SDH mreže, sequence9
b. LAYER_LABEL – naziv sloja PDH i SDH mreže, string(6)

LAYER ***(LAYER_ID,*** *LAYER_LABEL)*

31. RP_WAVELENGTH (SKUP VALNIH DULJINA)

a. WAVELENGTH_ID – identifikator grupe valnih duljina, sequence10
b. WAVELENGTH – valna duljina, u nanometrima; single

RP_WAVELENGTH ***(WAVELENGTH_ID,*** *WAVELENGTH)*

32. HIERARHY_LEVEL

a. NETWORK_HIERARHY_LEVEL - hijerarhijska razina; string(20)

HIERARHY_LEVEL ***(NETWORK_HIERARHY_LEVEL)***

Budući da ORACLE 8 Server ne prihvaća znak "/" u nazivima atributa i tablica, svi atributi i tablici koji su sadržavali navedeni znak su u danjem tijeku modeliranja modificirani. To se odnosi na mrežne elemente NE_ED/I i NE_OD/I čiji su nazivi promijenjeni u NE_EDI i NE_ODI. Atribut ED/I_LAYER_ID je promijenjen u EDI_LAYER_ID.

3.2 Analiza problema unosa podataka - potreba za ekspertnim sustavom

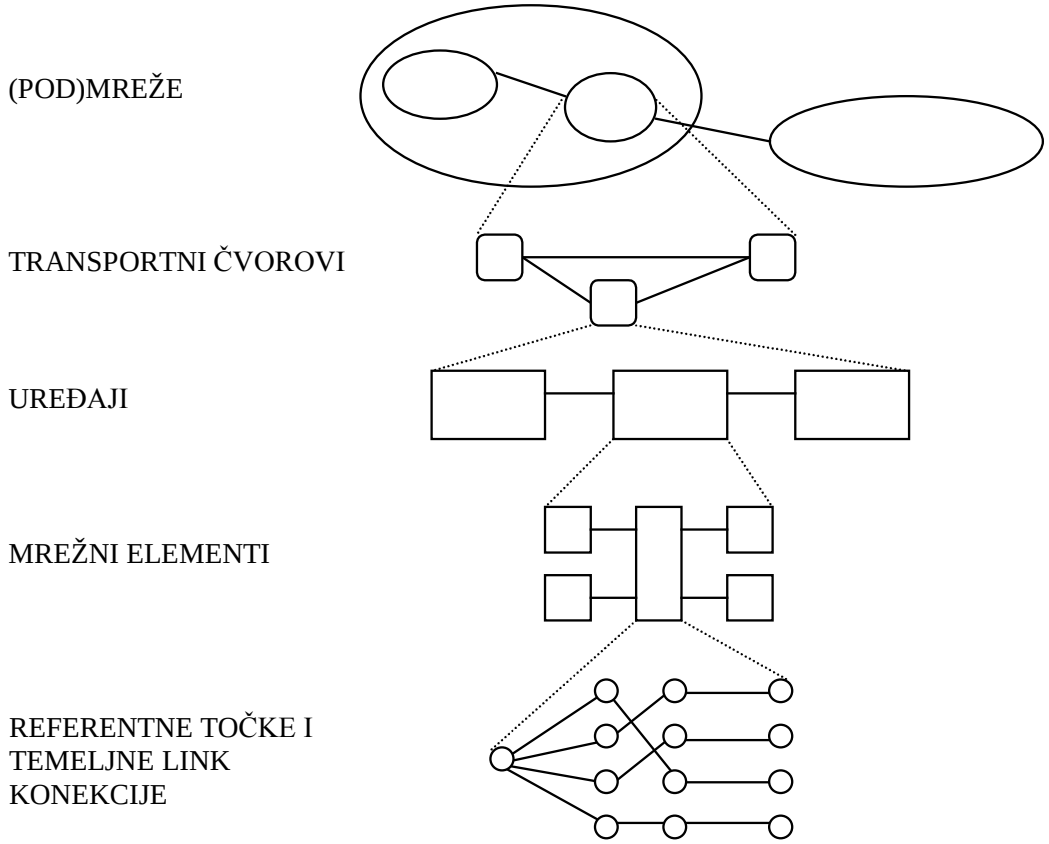
Kompleksan zadatak informacijske analize transportne mreže rezultirao je informacijskim modelom transportne mreže [3]. Takav informacijski model transportne mreže detaljno opisuje funkcionalnost i topologiju transportne mreže na različitim nivoima detaljnosti, različitim slojevima, SDH i PDH elementima, uključuje optiku i elektriku, a opis elemenata neovisan je o proizvođaču. Ovaj vrlo kompleksan i sveobuhvatan informacijski model preslikan je u relacijsku bazu podataka koristeći ORACLE-ov sustav za upravljanje bazom podataka (RDBMS).

Primjene rezultata ovog istraživanja početno smo ograničili na nadzor, imajući na umu da nam uz informacijski opis transportne mreže realiziran u bazi podataka razvoj odgovarajućih aplikacija nad takvom bazom podataka otvara široke mogućnosti primjene kao što su nadzor, upravljanje, analiza pouzdanosti itd.

3.2.1 Problem punjenja baze podataka transportne mreže

Baza podataka transportne mreže kao jedna kompleksna struktura sadrži ogromne količine podataka zbog detaljnosti opisa kojim je obuhvaćena transportna mreža. Osim na najdetaljnijem nivou, mreža je opisana i na svim višim nivoima.

Primjenu skrivanja kompleksnosti (enkapsulacije), kao jednog od glavnih principa objektnog pristupa, objasniti ćemo na primjeru uređaja: svaki uređaj može sadržavati desetke mrežnih elemenata (zapravo, taj broj nije ograničen u ovom modelu), svaki mrežni element može sadržavati stotine i tisuće referentnih točaka koje su međusobno povezane sa približno istim brojem konekcija.



Slika 3.4 – Prikaz transportne mreže na različitim hijerarhijskim nivoima

Dakle, da bi unijeli podatke o jednoj instanciji uređaja u bazu transportne mreže, osim same instancije entiteta uređaja moramo unijeti i sve entitete koje on 'skriva' kao opis na detaljnijem nivou i tako rekurzivno sve do najdetaljnijeg nivoa. Na najnižem nivou, unos jednog entiteta uređaja, ovisno o njegovoj kompleksnosti, može zahtijevati unos desetaka tisuća referentnih točaka i približno istog ili nešto manjeg broja temeljnih podmrežnih konekcija i temeljnih link konekcija sadržanih unutar uređaja.

Nadalje, u jednom transportnom čvoru može biti neograničen broj uređaja, a cijela transportna mreža sastoji se od čitavog niza transportnih čvorova, koji su opet dijelovi podmreža, mreža itd. Iz ovog primjera možete uočiti jasan hijerarhijski pristup temeljen na principu podjele. Rezultat takvog pristupa možete vidjeti na slici 3.4.

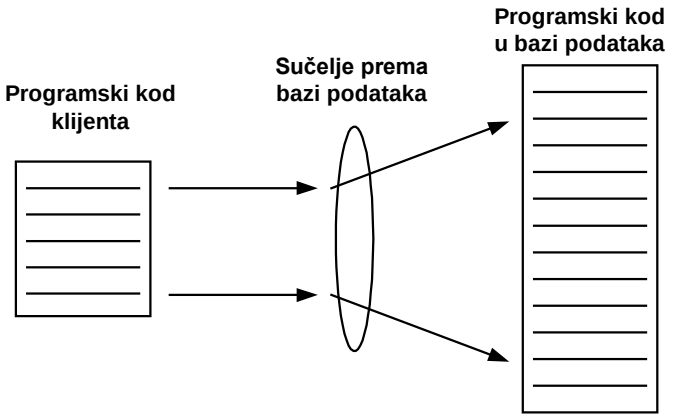
Zbog ogromne količine podataka koje treba unijeti da bi se čitava mreža detaljno opisala prvo punjenje baze podataka transportne mreže očito predstavlja ozbiljan i kompleksan problem. Taj problem, ukoliko se adekvatno ne riješi, može onemogućiti praktičnu realizaciju ovakvog sustava, ne samo zbog potrebnog čovjek-vremena za unos, već i zbog velike mogućnosti pogreške pri 'ručnom' unosu tako velikog broja podataka. Upravo taj problem rješava ekspertni sustav ***T**ransport **N**etwork **D**esigner* (TND) koji je okosnica ovoga magistarskog rada i biti će opisan u sljedećim poglavljima.

4. Tehnologije za pristup klijentske aplikacije prema bazi podataka

4.1 Osnovne postavke

Tehnologije za pristup bazama podataka se vrlo brzo razvijaju posljednjih godina. Javljaju se nove tehnologije i neprestano se usavršavaju. Tim tehnologijama želi se postići nekoliko stvari:

- Predstaviti klijentu programsko sučelje koje će pojednostavniti upravljanje i komunikaciju s bazom podataka. Relativno kompleksne operacije nad bazom podataka koje zahtijevaju veću količinu programskog koda mogu se izvesti na puno jednostavniji način preko takvih sučelja, sa puno manje linija koda. Takvo sučelje stoga možemo slikovito predočiti kao "povećalo koda" (slika 4.1). Svaka operacija proširuje se u kompleksniji kod potreban za njeno izvođenje. Time se olakšava rad programeru koji se tako može više usredotočiti na zadatak koji je potrebno obaviti nad bazom podataka, a ne na detalje tehnika izvođenja zadatka koje su u samim bazama podataka prilično kompleksne.



Slika 4.1 – Sučelje prema bazi podataka kao "povećalo koda"

- Napraviti sučelje koje će na univerzalan način moći pristupati što više različitih izvora podataka. Većina baza podataka pruža i svoje izvorno (*native*) sučelje koje je specifično i ograničeno na pojedini sustav za upravljanje bazom podataka, a programski kod koji koristi takvo specifično sučelje ne može se na jednostavan način upotrijebiti za druge DBMS-ove. Ako aplikacija koristi dobro dizajnirano univerzalno sučelje, takva aplikacija može pristupati bazi podataka bilo kojeg DBMS-a, bez promjene koda. Dakle, moguć je jednostavan prelazak sa jednog DBMS-a na drugi. Isto tako, jedna aplikacija može istovremeno koristiti više različitih DBMS-ova primjenom istog koda. To se najčešće postiže dodatnim programskim slojevima koji specifičnosti pojedinog DBMS-a prilagođavaju sloju iznad, univerzalnom sučelju.

Pri dizajniranju, ali i odabiru sučelja potrebno je posebnu pažnju obratiti na sljedeći problem: Visoka upravljivost, konfigurabilnost i brzina izvođenja u pravilu je

vezana i uz veliki broj linija koda koje upravljaju detaljima procesa. S malim brojem linija koda postižemo jednostavnost, efikasnost koda, veću univerzalnost i prenosivost, ali u pravilo gubimo utjecaj na detaljnije procese a tako i brzina izvršavanja koda više nije onakva kakvu bismo inače mogli postići. Tu je potrebno pronaći određen kompromis i ravnotežu, uzimajući u obzir postavljene zahtjeve.

4.2 Značajnija sučelja baza podataka za Windows platforme

U bližoj prošlosti razvijeno je nekoliko sučelja prema bazama podataka. Oni se razlikuju po svojoj namjeni, mogućnostima i po načinu realizacije svoje zadaće.

Značajnija sučelja na Windows platformama su:

- ODBC (***O**pen **D**ata**b**ase **C**onnectivity*)
- MFC (***M**icrosoft **F**oundation **C**lasses*) ODBC klase
- DAO (***D**ata **A**ccess **O**bjects*)
- RDO (***R**emote **D**ata **O**bjects*)
- OLE DB (***O**bject-**L**inking and **E**mbedding **D**ata**b**ase* – iako naziv ne odražava pravu bit ovog sučelja, zadržan je iz povijesnih razloga što će biti razjašnjeno u ovom poglavlju)
- ADO (***A**ctiveX **D**ata **O**bjects*)

4.2.1 ODBC

Prvu verziju ODBC-a napravio je Microsoft krajem 80-tih i početkom 90-tih godina 20. stoljeća. Njegova osnovna namjena je da kao univerzalno sučelje prema relacijskim bazama podataka aplikaciji omogućuje pristup bilo kojoj relacijskoj bazi podataka, neovisno o DBMS-u. Prilično je raširen i popularan i generalno je prihvaćen kao standard kad je riječ o sučeljima prema relacijskim bazama podataka.

ODBC je ograničen na relacijske baze podataka. Zbog svoje relacijske prirode i orijentiranosti prema relacijskim bazama vrlo je teško koristiti ODBC za komunikaciju s nerelacijskim izvorima podataka (objektne baze podataka, e-mail spremišta, mrežni direktoriji itd).

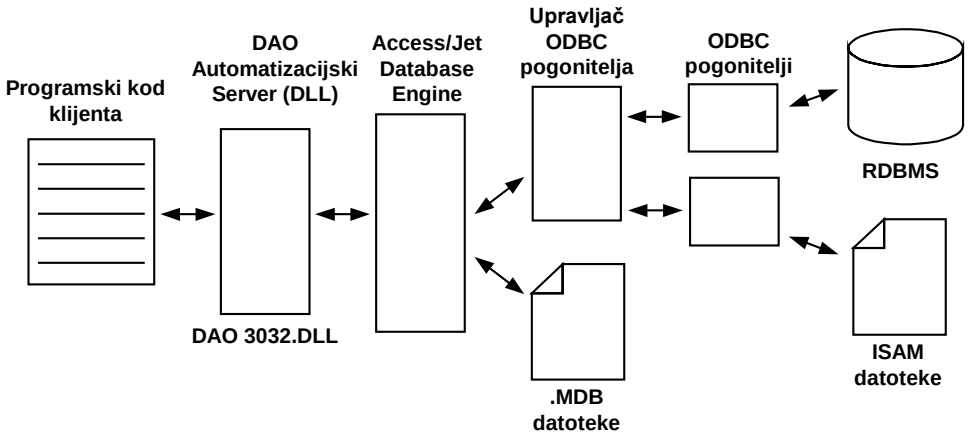
4.2.2 MFC ODBC klase

Iako ODBC pruža univerzalno sučelje prema relacijskim bazama podataka, ODBC kao programsko sučelje (API - *Application Programming Interface*) nije jednostavan. Postoje MFC (*Microsoft Foundation Classes*) C++ klase koje enkapsuliraju ODBC API što uvelike pojednostavljuje njegovu upotrebu i povećava efikasnost koda. Time se naravno gubi ona kontrola niske razine koju sam ODBC pruža, pa MFC ODBC možemo smatrati programskim sučeljem visoke razine.

Detaljniji opis ODBC-a i MFC ODBC klasa bit će dan u poglavlju 4.3.

4.2.3 DAO

DAO (***D**ata **A**ccess **O**bjects*) je skup COM (***C**omponent **O**bject **M**odel*) automatizacijskih sučelja (*COM Automation Interfaces*) za *Jet database engine* od *Microsoft Access* baza podataka. DAO komunicira direktno s *Access/Jet* bazama podataka. *Access/Jet database engine* može komunicirati i sa ostalim bazama podataka, ali indirektno, preko ODBC-a, kako je prikazano na slici 4.2.



Slika 4.2 – DAO arhitektura

DAO nije tek funkcijski orijentiran API (*Application Programming Interface*), već pruža objektni model za programiranje baza podataka. Dakle, prilagođen je i lakše ga je implementirati u objektno orijentiranim programskim jezicima.

DAO pruža objekte za spajanje na bazu podataka, manipuliranje podacima, pa čak i strukturom *MS Access* baze podataka. To znači da je pomoću DAO objektnog modela moguće direktno mijenjati ili kreirati relacijske sheme, indekse, poglede i slično u *MS Access* bazi podataka bez SQL DDL (*Data Definition Language*) instrukcija.

DAO pruža koristan objektni model za programiranje baza podataka i dobar je izbor ukoliko ga koristimo za *MS Access* bazu podataka. Ako ga koristimo za neku drugu bazu podataka, pozivi se preusmjeravaju preko ODBC-a, kao što se vidi na slici 4.2. Zbog tog prolaska kroz više softverskih slojeva dok se ne dopre do ODBC-a i same baze podataka, što znači neizbježan gubitak na brzini.

DAO je lakše koristiti nego ODBC API jer je jednostavniji i objektno orijentiran, ali ne pruža tako nisku razinu kontrole kao što ga daje ODBC. Zato DAO možemo klasificirati kao sučelje visoku razinu za pristup bazama podataka. Postoji skup MFC klasa koje daljnje simplificiraju upotrebu DAO automatizacijskog sučelja. Prefiks njihova naziva je "CDAO". Detaljnije o njima može se pročitati u dokumentaciji *Visual C++* [2].

4.2.4 RDO

RDO je kratica od *Remote Data Objects*. RDO je razvijen kao apstrakcija ODBC API-ja za *Visual Basic* programere. Zbog toga je RDO usko vezan uz ODBC i *Visual*

Basic. RDO je lakše koristiti nego ODBC API, ali ne pruža nisku razinu kontrole kao što ga daje ODBC. Zato DAO možemo klasificirati kao sučelje visoke razine za pristup bazama podataka.

Zbog toga što RDO direktno komunicira s ODBC API sučeljem (za razliku od DAO-a, koji ODBC API uvijek poziva kroz *Jet engine*) daje dobre performanse za aplikacije koje koriste relacijske baze podataka.

RDO mogu upotrebljavati *Visual C++* aplikacije ubacivanjem `RemoteData` komponente. `RemoteData` je OLE komponenta koja se može vezati na komponente korisničkog sučelja aplikacije (*User Interface*). RDO funkcije zatim se mogu pozivati kroz metode `RemoteData` komponente.

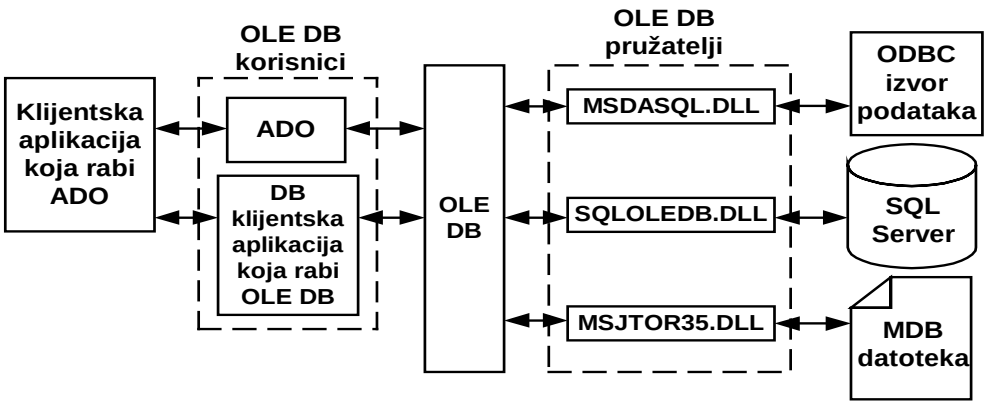
4.2.5 OLE DB

OLE DB je skraćenica za *Object-Linking and Embedding Database*. Ovaj naziv ostao je iz povijesnih razloga i ne opisuje stvarni karakter pojma koji predstavlja. OLE DB je širi od ODBC-a na dva načina. Prvo, pruža OLE, zapravo COM sučelje za programiranje baza podataka. Drugo, OLE DB pruža sučelje prema relacijskim i nerelacijskim izvorima podataka.

Dakle, OLE DB pruža OLE (COM) sučelje. OLE je bio prvobitan naziv za COM. Kad je OLE DB nastajao, naziv OLE se još uvijek koristio za COM. Od tada, COM se počeo koristiti kao naziv za temeljnu Microsoftovu komponentnu tehnologiju. Koristeći ga kao akronim, OLE DB naziv asocira na OLE/COM i baze podataka. Ipak, značenje kratice nema smisla jer OLE DB nema puno toga zajedničko s komponentama korisničkog sučelja kao što su OLE komponente.

Realizacija OLE DB-a kao COM sučelja za programiranje baza podataka je važno jer COM sučelje može biti puno robusnije od tradicionalnog sučelja na razini poziva, kao što je ODBC. Ova fleksibilnost može rezultirati u boljim performansama i robusnijim mehanizmom upravljanja pogreškama i može omogućiti interakciju s nerelacijskim izvorima podataka. Kao i ODBC, OLE DB možemo klasificirati kao sučelje niske razine prema bazama podataka. OLE DB uključuje funkcionalnost ODBC-a prema relacijskim bazama podataka i proširuje je pristupom nerelacijskim izvorima podataka.

Postoje dvije kategorije OLE DB softvera: OLE DB korisnici i OLE DB pružatelji. Slika 4.3 ilustrira odnos između OLE DB korisnika i OLE DB pružatelja.



Slika 4.3 – OLE DB korisnici i pružatelji

OLE DB korisnik je svaka aplikacija koja koristi ili upotrebljava OLE DB sučelja. OLE DB pružatelji su DLL-ovi (*Dynamic Link Libraries*) koji implementiraju OLE DB sučelja i ostvaruju direktnu komunikaciju s izvorom podataka. OLE DB pružatelji imaju sličnu funkcionalnost kao ODBC pogonitelji, samo što OLE DB pružatelji implementiraju COM sučelja umjesto API funkcija.

OLE DB omogućuje pristup svakom izvoru podataka za kojeg postoji OLE DB pružatelj. Ovi izvori podataka mogu biti npr. spremišta e-mailova, objektne baze podataka, mrežni direktoriji, i druga nerelacijska spremišta podataka. Na slici 4.3 vidimo da OLE DB pružatelj MSDASQL.DLL omogućuje komunikaciju s ODBC izvorima podataka. To je korisno za one izvore podataka koji imaju ODBC pogonitelj, ali još nemaju OLE DB pružatelj.

OLE DB korisniku eksponira skup COM sučelja koja mogu biti pozivana iz npr. C++ programa. OLE DB ne pruža automatizacijsko sučelje. Microsoft-ovi napori usmjereni su na daljnji razvoj ove tehnologije koja će postepeno zamijeniti ODBC.

4.2.6 ADO

ADO je skraćenica za *ActiveX Data Objects*. ADO je sučelje sagrađeno nad OLE DB, tj. ADO je OLE DB korisnik. Aplikacije koje koriste ADO zapravo indirektno koriste OLE DB sučelja. ADO pruža objektni model za programiranje baza podataka koji je sličan, ali fleksibilniji objektnom modelu DAO. Npr. koristeći ADO moguće je kreirati `Recordset` objekte bez prethodno kreiranog `Connection` objekta, što se ako koristimo DAO ne može učiniti.

ADO pojednostavljuje OLE DB. OLE DB je prilično kompleksan i velik, i programi koji ga koriste moraju implementirati neka kompleksna COM sučelja. ADO je puno jednostavniji za upotrebu na način da enkapsulira OLE DB i čini jedan sloj iznad, te ga možemo klasificirati kao programsko sučelje visoke razine.

ADO je moguće koristiti sa više programskih jezika nego OLE DB jer implementira tzv. automatizacijsko sučelje (*Automation interface*). To omogućuje korištenje ADO-a u skriptnim jezicima kao što su VBScript i JavaScript. OLE DB se ne može koristiti u skriptnim jezicima jer oni ne podržavaju pokazivače i zbog toga ne mogu koristiti COM sučelja.

4.3 ODBC

4.3.1 Definicija ODBC-a

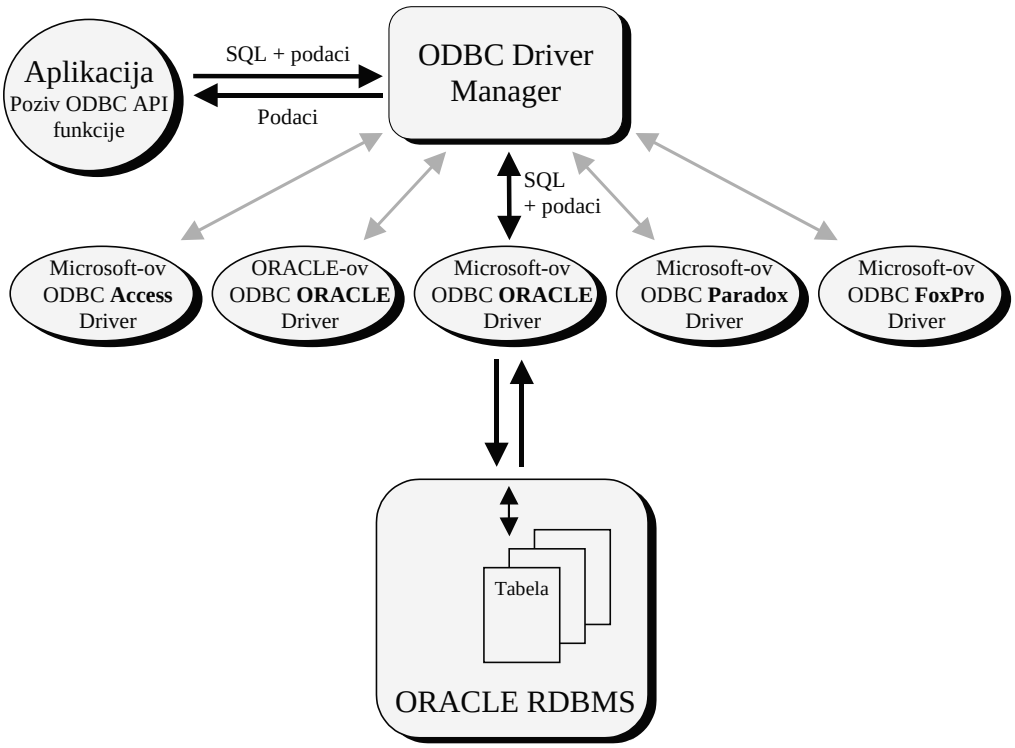
ODBC (*Open Database Connectivity*) je naziv za skup standarda kojima se definira jedinstveno logičko sučelje prema različitim DBMS-ima (*Database Management System*) za koje postoji odgovarajući ODBC pogonitelj. Preko standardnih funkcija definiran je pristup, spajanje na bazu, operacije nad podacima i relacijskim shemama. ODBC je implementiran kao programsko sučelje (API – *Application Programming Interface*) koje nam omogućava razvoj aplikacija neovisnih o specifičnostima pojedinih DBMS-a.

4.3.2 Kako radi ODBC?

Jedno od bitnih svojstava ODBC-a je da omogućava pristup različitim bazama podataka (koje mogu biti na istom ili udaljenom računalu) preko ODBC API (***A**pplication **P**rogramming **I**nterface*) programskog sučelja.

ODBC nam nudi aplikacijsko programsko sučelje (API - možemo ga shvatiti kao skup funkcija i klasa) koje implementiraju razni proizvođači DBMS-a u ODBC pogonitelju specifičnom za njihov DBMS (npr. Microsoft-ov pogonitelj za ORACLE). U vlastitom programu pozivamo funkcije ODBC API sučelja koje konstruiraju SQL izraz i zovu *ODBC Driver Manager*, a ovaj će proslijediti taj poziv odgovarajućem pogonitelju (*driver*). Pogonitelj dalje komunicira sa DBMS-om kako bi izvršio zadanu akciju.

Dakle, pozivi svih ODBC API funkcija na opisani način pretvaraju se u SQL izraze koji se dalje prosljeđuju prema bazi. Moguće je i direktno izvršenje SQL izraza preko funkcije `CDatabase::ExecuteSQL` ili izravno preko ODBC API funkcije `SQLExecDirect`.



Slika 4.4 – Mehanizam interakcije sa DBMS-om preko ODBC-a

4.3.3 IZVOR PODATAKA (*Data Source*)

Preko ODBC-a možemo pristupati samo onim bazama podataka za koje imamo odgovarajući ODBC pogonitelj i koje smo registrirali u *registry* listi računala kao izvor podataka pomoću *ODBC Data Source Administrator*-a (ne treba miješati *ODBC Driver Manager* i *ODBC Data Source Administrator*).

ODBC Data Source Administrator nam dolazi u paketu Windows operativnog sustava i možemo ga pokrenuti iz *Start→Control Panel*. Za ORACLE su za sada dostupna dva pogonitelja; *Oracle ODBC Driver for Rdb* verzija 2.00.1800 (SQOCI32.DLL) i *Microsoft ODBC Driver for Oracle* verzija 2.00.006325 (MSORCL10.DLL). Smatram da je Microsoft-ov pogonitelj pogodniji za korištenje jer ORACLE-ov pogonitelj, barem u ovoj verziji, ne podržava neke funkcije vrlo bitne za brzinu izvođenja određenih zadataka, kao što je npr. **SQLSetPos** koja se vrlo često koristi. Koje funkcije pogonitelj podržava i druge informacije koje želite saznati o pogonitelju koji imate, možete saznati ako pokrenete primjer **odbcinfo** iz *MS VC++ 6.0 Enterprise* paketa.

Izvor podataka sadrži informaciju o tome kako se spojiti na bazu podataka. Potrebno je specificirati pogonitelj, ime izvora podataka (DSN - ***D**ata **S**ource **N**ame*), i zatim izabrati bazu na koju se želimo spojiti (baza može biti na istom ili udaljenom računalu). Ime izvora podataka (DSN) je proizvoljno i služi nam za identifikaciju izvora podataka.

Postoji uska veza između `CDatabase` objekta i izvora podataka. Svaki `CDatabase` objekt predstavlja vezu prema izvoru podataka. Moguće je imati i više

simultanih konekcija nad istim izvorom podataka, gdje svaka konekcija mora biti predstavljena svojim `CDatabase` objektom.

Ne moramo uvijek konfigurirati izvor podataka preko ODBC administratora, već to možemo učiniti i programski, korištenjem ODBC API funkcije `::SQLConfigDataSource`. ODBC administrator stvara nove izvore podataka tako da ih upiše u *Windows registry*. *ODBC Driver Manager* pretražuje tu datoteku. Nadalje, možemo programski kreirati nove tablice i mijenjati relacijske sheme.

4.3.4 ODBC MFC KLASE

4.3.4.1 Uvod

Pri programiranju aplikacija u *Visual C++*-u možemo koristiti gotove klase koje programera oslobađaju kompleksnosti direktnog kodiranja pomoću ODBC funkcija. To je grupa klasa u sklopu *Microsoft*-ove biblioteke klasa MFC (***M****icrosoft* ***F****oundation* ***C****lass Library*). Te klase uvelike olakšavaju rad programeru jer ODBC API koji je inače funkcijski orijentiran enkapsulira u objekte. Program koji koristi ODBC mora imati barem jedan objekt klase `CDatabase` (predstavlja konekciju prema bazi podataka) i jedan objekt klase `CRecordset` (predstavlja podatke iz tablice ili pogleda).

Slijedi kratki opis svake klase:

`CDatabase` - Sadrži vezu prema izvoru podataka (*Data Source*) kroz koju možemo manipulirati podacima.

`CRecordset` - Sadrži skup redaka (najčešće 1) iz izvora podataka, tj. iz tablice i čitav niz funkcija za kretanje od retka do retka tablice (*scrolling*) i upis podataka (*updating*), tj. izmjenu, brisanje i dodavanje novih redaka. Također omogućava i filtriranje, sortiranje i parametriziranje izabranog skupa podataka sa informacijama dobivenim za vrijeme izvođenja programa. Pomoću RFX (*Record Field Exchange*) mehanizma vrši prijenos podataka iz varijabli `CRecordset` objekta u izvor podataka i obratno.

`CRecordView` - Klasa indirektno izvedena iz klase `CView`. Omogućuje prikaz podataka iz `CRecordset` objekta u obliku forme. Pomoću DDX (*Dialog Data Exchange*) mehanizma vrši prijenos podataka iz kontrola dijaloga u varijable `CRecordset` objekta i obratno.

`CDBException` - `exception` klasa za obradu pogrešaka vezanih uz podatke. Ponaša se isto kao i druge `exception` klase.

`CFieldExchange` - Sadrži kontekstne informacije za RFX (*Record Field Exchange*) mehanizam, kao što je npr. smjer izmjene podataka. RFX, slično kao i DDX služi za razmjenu podataka, ali ne između dijaloga i varijabli, nego između varijabli `CRecordset` objekta (tzv. atributnih varijabli) i izvora podataka.

4.3.4.2 RECORDSET

Što je recordset*?*

Objekt klase `CRecordset` predstavlja skup zapisa selektiranih iz izvora podataka. Skup zapisa može biti iz tablice, upita (*query*) ili iz procedure na strani baze podataka (*stored procedure, predefined query*) koja pristupa jednoj ili više tablica.

Tipovi recordset-a

Postoje dvije osnovne vrste *recordset*-a. To su `dynaset` i `snapshot`. `Dynaset` tip uzima zapis iz izvora u trenutku kada se pozicioniramo na taj zapis. Na taj način vidimo eventualne promjene koje naprave drugi korisnici. `Snapshot` uzima sve zapise u trenutku otvaranja *recordset*-a, tako da ne možemo vidjeti kasnije promjene koje rade drugi korisnici. Da bi vidjeli dodavanja i brisanja redaka koje rade drugi korisnici treba pozvati funkciju `CRecordset::Requery`.

Nadalje, postoje još dva tipa *recordset*-a koje ODBC podržava. To su `dynamic` i `forward only`. `Dynamic` tip *recordset*-a je sličan `dynaset`-u, s tom razlikom da ne treba eksplicitno zvati `CRecordset::Requery` da bi vidjeli dodane i izbrisane zapise. Zbog toga `dynamic` košta nešto više procesnog vremena nego `dynaset` i mnogi ODBC pogonitelji ga ne podržavaju. Naprotiv, `forward only` tip *recordset*-a nam daje najefikasniji način pristupa podacima ako ne radimo nikakve izmjene u tablici i ne trebamo ići unazad unutar *recordset*-a. To je korisno primjerice kada trebamo samo pročitati neki *recordset* ili prenijeti zapise iz jedne tablice u drugu. Da bi koristili `forward only` tip *recordset*-a moramo pri pozivu metode `CRecordset::Open` navesti `CRecordset::forwardOnly` za `nOpenType` parametar, i `CRecordset::readOnly` za `dwOptions` parametar metode.

Prijenos podataka između recordset*-a i izvora podataka*

Iako nam pruža pristup do svih izabranih zapisa, `CRecordset` objekt u nekom trenutku sadrži samo jedan zapis (ako ne radimo sa *Bulk Row Fetching*-om), tzv. tekući zapis (*current record*). Tekući zapis je pohranjen u obliku varijabli unutar objekta klase koju izvodimo iz klase `CRecordset`. U svakoj varijabli pohranjen je jedan atribut tekućeg zapisa. Varijable *recordset* objekta koje čuvaju vrijednosti atributa jednog zapisa možemo nazivati “atributne varijable”. Evo primjera *recordset* klase za tablicu CROSSCONNECT:

```
class CXCSet : public CRecordset
{
public:
   CString  m_TYPE;
   CString  m_MANUFACTURER;
   long     m_CROSSCONNECT_LEVEL;
   long     m_NO_OF_REFERENCE_POINTS;
   long     m_NO_OF_SOURCE_POINTS;
```

```
   // Definicija ostalih varijabli
   ...
   // Definicija metoda
   ...
};
```

Imena atributnih varijabli mogu biti proizvoljno izabrana, ali je poželjno da budu izvedena iz stvarnih naziva atributa u tablici. Veza pojedine varijable sa odgovarajućim atributom iz tablice definira se u funkciji `CRecordset::DoFieldExchange`. To nije gotova funkcija, već je mora implementirati sam programer (deklarirana je kao `virtual`). Tu funkciju koriste metode klase `CRecordset` u radi prenošenja podataka iz tablice u atributne varijable i obratno. Nadalje, potrebno je voditi računa o tome koji C++ tip podatka odgovara nekom SQL tipu. Ako se vlastita *recordset* klasa izvodi iz `CRecordset` klase pomoću *Class Wizard*-a, *Class Wizard* će automatski kreirati funkciju `DoFieldExchange` sa odgovarajućim pozivima RFX funkcija.

***R**ecord **F**ield Ex**c**hange* mehanizam (RFX)

RFX mehanizam služi prijenosu podataka između izvora podataka i varijabli u *recordset* objektu, i to u oba smjera. Postoje globalne funkcije nazvane RFX funkcije koje služe u tu svrhu, kao npr. `RFX_Text, RFX_Int, RFX_Date` i druge, svaka za određeni tip podataka.

Tijelo funkcije `DoFieldExchange` (uglavnom) se sastoji od poziva RFX funkcija. To su globalne funkcije koje služe izmjeni podataka između izvora podataka (tablice) i atributnih varijabli. Smjer prijenosa određen je sadržajem objekta klase `CFieldExchange`. Ponovno napominjem da funkciju `DoFieldExchange` vjerojatno nećete direktno zvati. Nju pozivaju neke funkcije (metode) klase `CRecordset`.

Primjer:

```
void CXCSet::DoFieldExchange(CFieldExchange* pFX)
{
   //{{AFX_FIELD_MAP(CXCSet)
   // Naznačimo da slijede atributi
   pFX->SetFieldType(CFieldExchange::outputColumn);
   // Izmjena podataka pomoću RFX funkcija
   RFX_Text(pFX, _T("[TYPE]"), m_TYPE);
   RFX_Text(pFX, _T("[MANUFACTURER]"), m_MANUFACTURER);
   RFX_Long(pFX, _T("[CROSSCONNECT_LEVEL]"),
       m_CROSSCONNECT_LEVEL);
   ...
//}}AFX_FIELD_MAP
}
```

4.3.4.2.1 Sortiranje, filtriranje i parametriziranje *recordset*-a

Sortiranje recordset-*a*

Možemo imati i sortirani *recordset* tako da postavimo varijablu `CRecordset::m_strFilter` tipa `CString` prije nego što pozovemo `Open` ili `Requery`. Pri konstrukciji SQL izraza, `m_strFilter` će biti dodan iza `ORDER BY` klauzule.

Primjer:

```
// Konstrukcija recordset-a
CXCSet rsXCSet(NULL);
// Postavi kriterij sortiranja
rsXCSet.m_strSort = "MANUFACTURER ASC, TYPE DESC";
// Pokreni upit
rsXCSet.Open();
```

Navedeni primjer konstruirati će SQL izraz:

SELECT * FROM CROSSCONNECT ORDER BY MANUFACTURER ASC, TYPE DESC

i otvoriti *recordset* sa svim prospojnicima iz tablice CROSSCONNECT, sortiran prema proizvođaču, a za istog proizvođača prema tipu u padajućem redoslijedu.

Filtriranje recordset-*a*

Prije nego što pozovemo `Open` ili `Requery`, možemo postaviti varijablu `CRecordset::m_strFilter` tipa `CString` koja se dodaje nakon `WHERE` pri konstrukciji SQL izraza.

Primjer:

```
// Konstrukcija recordset-a
CXCSet rsXCSet(NULL);
// Postavi filter
rsXCSet.m_strFilter = "MANUFACTURER = 'SIEMENS'";
// Pokreni filtrirani upit
rsXCSet.Open(CRecordset::dynaset, "CROSSCONNECT");
```

Navedeni primjer konstruirati će SQL izraz:

SELECT * FROM CROSSCONNECT WHERE MANUFACTURER = 'SIEMENS'

i otvoriti *recordset* sa svim prospojnicima iz tablice CROSSCONNECT kojima je proizvođač SIEMENS.

Parametrizacija recordset-*a*

Ako želimo izabrati samo neke zapise iz tablice, i to prema svojstvima koja nisu unaprijed poznata, ili konkretno, ako želimo iz tablice CROSSCONNECT izabrati prospojnike istog proizvođača i to onog kojeg korisnik aplikacije izabere, to možemo učiniti na dva načina:

1) Nakon što korisnik izabere proizvođača, postavimo filter koji izabire tražene zapise i zovemo `Open` koji konstruira SQL `SELECT` izraz, npr.

SELECT * FROM CROSSCONNECT WHERE MANUFACTURER='SIEMENS'

Ako želimo prospojnike nekog drugog proizvođača, potrebno je konstruirati novi filter i osvježiti *recordset* pozvavši `Requery`.

2) ili upotrebom parametriziranog filtera:

```
SELECT * FROM CROSSCONNECT WHERE MANUFACTURER = ?
```

gdje znak '?' predstavlja parametar na koji će biti vezane konkretne vrijednosti - u ovom slučaju 'SIEMENS'. Ako sada želimo prospojnike nekog drugog proizvođača, dovoljno je promijeniti vrijednost parametra i pozvati `Requery`.

Upotreba parametara je puno efikasnija nego upotreba neparametriziranih filtera. Za parametrizirani *recordset*, baza podataka mora procesirati SQL SELECT izraz samo jednom. Kod filtriranih *recordset*-a bez parametara baza procesira SQL SELECT izraz svaki put kada pozovete `Requery` sa novom vrijednošću filtera.

Parametrizaciju koristimo i da bi poslali parametre spremljenoj proceduri na strani baze. U tom slučaju moramo konstruirati cijeli ODBC CALL izraz sa parametrima koji šaljemo funkciji `Open`.

Parametrizirati možemo svaki *recordset*.

Primjer:

```
// Postavimo parametrizirani filter
rsXCSet.m_strFilter = "MANUFACTURER = ?";
// Postavite CString varijablu
//  koju ste namijenili za parametar
rsXCSet.m_strMANUFACTURERParam = "SIEMENS";
// Osvježite recordset
if(!rsXCSet.Requery())
   return FALSE;
```

Potrebno je postaviti i varijablu `m_nParams = 1` i u funkciji `CXCSet:: DoFieldExchange` dodati retke koji kvalificiraju varijablu `m_strMANUFACTURERParam` kao parametar:

```
// Dojavimo CFieldExchange klasi da slijede
```

```
//  parametarske varijable
pFX->SetFieldType(CFieldExchange::param);

// RFX pozivi za parametarske varijable
//  i to u redosljedu pojavljivanja '?'
RFX_Text(pFX, "parametar1", m_strMANUFACTURERParam);
```

4.3.4.2.2 Izmjena, dodavanje novih, i brisanje zapisa

Da bi saznali da li se u neki *recordset* uopće može upisivati (ili se može samo čitati), možete pozvati funkciju `CRecordset::CanUpdate`.

Kod dodavanja i izmjene, vrši se spremanje atributnih varijabli *recordset*-a u za to određeni memorijski prostor, tzv. *buffer*, kako bi bilo moguće poništiti promjenu i vratiti stare vrijednosti. To kopiranje vrši se pri pozivu funkcija `AddNew` i `Edit`.

Pri pozivu tih funkcija vrši se i označavanje svakog atributa kao promijenjenog ili nepromijenjenog (*dirty / clean*). Pri kasnijem pozivu `Update` funkcije samo atributi označeni kao promijenjeni (*dirty*) šalju se prema izvoru podataka. Time se smanjuje promet između aplikacije i izvora podataka.

OPREZ! Ako `Update` ne uspije, ostajemo u `edit` ili `addnew` modu.

Izmjena

Da li je moguće vršiti izmjene na *recordset*-u provjeravamo pozivom funkcije `CRecordset::CanUpdate` koja vraća `TRUE` ako su izmjene moguće. Nadalje, moramo biti postavljeni na neki postojeći zapis, tj. `IsBOF, IsEOF` i `IsDeleted` moraju vratiti `FALSE`.

Nakon što smo se pozicionirali na zapis koji želimo mijenjati, pozivom funkcije `Edit` ulazimo u tzv. *edit* mod. `Edit` kopira atributne varijable u *buffer* kako bi bilo moguće poništiti izmjenu i vratiti prijašnje vrijednosti atributnih varijabli. *Buffer* se koristi i da bi se odredilo koji atributi su promijenjeni. Svi atributi označuju se kao nepromijenjeni (*clean*).

Zatim promijenimo atributne varijable koje želimo i pozovemo funkciju `Update` čime se izmjena upisuje u izvor podataka. Upisuju se samo atributi obilježeni kao promijenjeni (*dirty*). Ako želimo izaći iz *edit* moda bez ikakvih promjena, pozivamo funkciju `Cancel` koja natrag kopira spremljene atributne varijable i otkazuje cijeli postupak. U tom slučaju, naravno, ne zovemo `Update` jer bi to izazvalo pogrešku.

Nakon toga smo pozicionirani na upravo mijenjanom zapisu.

Primjer:

```
// Pretpostavimo da imamo otvoreni recordset
// CXCSet rsXCSet i da smo postavljeni na zapis
```

```
// koji želimo izmijeniti.

// Ulazi u edit mod
rsXCSet.Edit();

// Mijenja naziv proizvođača
rsXCSet.m_MANUFACTURER = “Siemens”;

// Pošalji promjenu u izvor.
// (Kada bi pozvao rsXCSet.Cancel() promjena bi bila
//  poništena, u atributne varijable bi bile vraćene
//  spremljene vrijednosti, u izvor podataka ne bi bi
//  bilo ništa poslano.)
if(!rsXCSet.Update()) {
  AfxMessageBox(“Zapis nije promijenjen.”);
  return FALSE;
}
```

Dodavanje novih zapisa

Da li je moguće dodavati nove zapise u neki *recordset*, možemo saznati ako pozovemo funkciju `CRecordset::CanAppend` i `CanUpdate`.

Pozivom funkcije `AddNew` atributne varijable se spremaju u *buffer*, i označuju praznima (*Null* - nije im dodijeljena vrijednost. To nije C vrijednost `NULL`) i nepromijenjenima (*clean*).

Daljnji postupak je isti kao i kod izmjene zapisa: Postavimo atributne varijable i pozovemo `Update` kako bi se ostvario unos novog zapisa u izvor podataka (bazu).

Nakon `Update`, zapis koji je bio tekući prije poziva `AddNew` sada postaje tekući, bez obzira je li `Update` bio uspješan.

Primjer:

```
rsXCSet.AddNew();

// Postavi vrijednosti novog zapisa
rsXCSet.m_TYPE = “NOVI XC”;
rsXCSet.m_MANUFACTURER = “Siemens”;
rsXCSet.m_CROSSCONNECT_LEVEL = 64;

// Pošalji novi zapis u izvor podataka.
if(!rsXCSet.Update()) {
  AfxMessageBox(“Zapis nije dodan.”);
```

```
  return FALSE;
}
```

Ovisno o mogućnostima pogonitelja potrebno je pozvati `Requery` da bi novododani zapisi bili vidljivi u tom *recordset*-u.

Brisanje zapisa

Brisanje je moguće ako funkcija `CanUpdate` vraća `TRUE`. Nadalje, moramo biti pozicionirani na neki postojeći zapis (koji želimo obrisati), tj. `IsBOF, IsEOF` i `IsDeleted` moraju vratiti `FALSE`.

Brisanje zapisa je daleko jednostavnije od dodavanja ili izmjene. Dovoljno je postaviti se na željeni zapis i pozvati funkciju `Delete`**.** Nakon toga poželjno je pomaknuti se na neki drugi, postojeći zapis. Ne treba zvati `Update`.

Primjer:

```
rsXCSet.Delete();
rsXCSet.MoveNext();
```

4.3.4.2.3 Recordset u višekorisničkoj okolini

Kako promjene koje rade drugi korisnici utječu na naš recordset objekt

U višekorisničkoj okolini, moguć (i čest) je slučaj da drugi korisnici otvaraju *recordset*-e koji sadrže neke (ili sve) zapise našeg recordset-a. Postavlja se pitanje kako naše promjene (dodavanja, izmjene, brisanja) utječu na druge korisnike, i obratno, kako možemo vidjeti promjene koje rade drugi korisnici.

Starost svakog zapisa mjeri se od trenutka kada je zapis zadnji puta dohvaćen iz baze, tako da su zapisi koje vidimo slika stanja kakvo je bilo pri zadnjem dohvatu tih podataka iz baze.

`Dynaset` *recordset*-ovi dohvaćaju zapis iz baze svaki puta kada se želimo pozicionirati na njega, tako da su izmjene koje rade drugi korisnici vidljive. Ako je neki zapis obrisan, biti će jednostavno preskočen pri pomicanju kroz *recordset*. Zapisi koje su dodali drugi korisnici postaju vidljivi tek nakon poziva funkcije `Requery`.

`Snapshot` *recordset*-ovi dohvaćaju zapis samo kada se prvi puta želite pozicionirati na njega, tako da možete vidjeti samo one promjene koje su nastale prije nego ste se pozicionirali na taj zapis. Da bi vidjeli kasnije izmjene, brisanja ili dodavanja zapisa koje su izvršili drugi korisnici, potrebno je pozvati `Requery`.

Da bi vidjeli zapise koje ste sami dodali, ponekad (ovisno o mogućnostima pogonitelja) je također potrebno pozvati `Requery`.

*Zaključavanje zapisa (*locking*)*

Aplikacija može "zaključati" zapis čime onemogućuje drugim korisnicima promjenu ili brisanje toga zapisa. ODBC Klase omogućavaju dva moda (načina) zaključavanja:

1) optimističan (*optimistic*), i

2) pesimističan (*pessimistic*)

Optimistično zaključavanje djeluje samo za vrijeme poziva `Update` funkcije. Ovaj način zaključavanja je *default*, i nužan je jer sprečava istovremeni upis više različitih podatka na isto mjesto (isti zapis) čime bi rezultat bio nedefiniran.

Pesimistično zaključavanje djeluje tako da zaključa zapis pri pozivu `Edit` funkcije, i drži ga zaključanim sve do poziva `Update` funkcije nakon kojeg ga oslobađa (otključava). Ovaj način košta druge korisnike u toliko što onemogućava izmjene toga zapisa za vrijeme između poziva `Edit` i `Update` funkcije. Na žalost, trenutno malo koji pogonitelj podržava pesimistično zaključavanje.

Ako želite promijeniti mod zaključavanja, upotrijebite funkciju `SetLockingMode` sa parametrom `CRecordset::optimistic` ili `CRecordset::pessimistic`. Promjena moda zaključavanja treba se izvesti prije poziva funkcije `Edit`.

4.4 Komentar

U ovom poglavlju u kratkim crtama opisana su najčešće korištena programska sučelja baza podataka. Može se uočiti da gotovo svako od ovih sučelja ima pristup ODBC-u, ili se temelji na ODBC-u kao standardu za pristup relacijskim bazama podataka na Windows platformama. ODBC ima izvrsne performanse i nisku razinu kontrole, ali je relativno kompleksan i ima relativno nisku efikasnost koda (velik broj linija koda potreban za neki zadatak). MFC ODBC, sa svojim objektnim sučeljem, eliminira navedene mane uz minimalan gubitak na performansama. Iako će u budućnosti Microsoft sve više prelaziti na OLE DB i ADO tehnologije, u vrijeme pisanja aplikacija opisanih u ovom magistarskom radu još nisu postigli punu raširenost, stabilnost i funkcionalnost, pa je iz tih razloga, ali i iz svojih prednosti, MFC ODBC logičan izbor tehnologije za pristup bazama podataka koja se koristi u verifikaciji ovog rada.

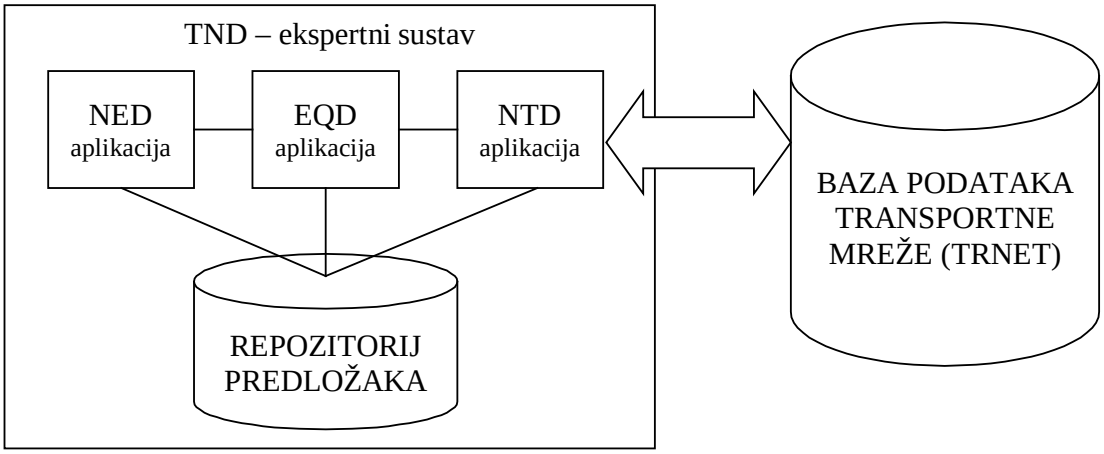
5. TND - ekspertni sustav za unos podataka o transportnoj mreži

5.1 Osnovni koncept

TND (***T**ransport **N**etwork **D**esigner*) je ekspertni sustav koji služi za efikasno prvo punjenje i ažuriranje baze podataka transportne mreže, njeno konfiguriranje, grafički prikaz i dizajniranje. Sastoji se od četiri komponente (slika 5.1):

1. Baze podataka nazvane "TND repozitorij predložaka" ili kraće "repozitorij", i triju aplikacija:
2. NED (***N**etwork **E**lement **D**esigner*),
3. EQD (*Network **Eq**uipment **D**esigner*) i
4. NTD (***N**etwork **T**opology **D**esigner*).

Ove komponente sustava međusobno su povezane i nadopunjavaju se u smislu da jedna aplikacija koristi servise one druge, iako su u potpunosti modularne i mogu funkcionirati i nezavisno.

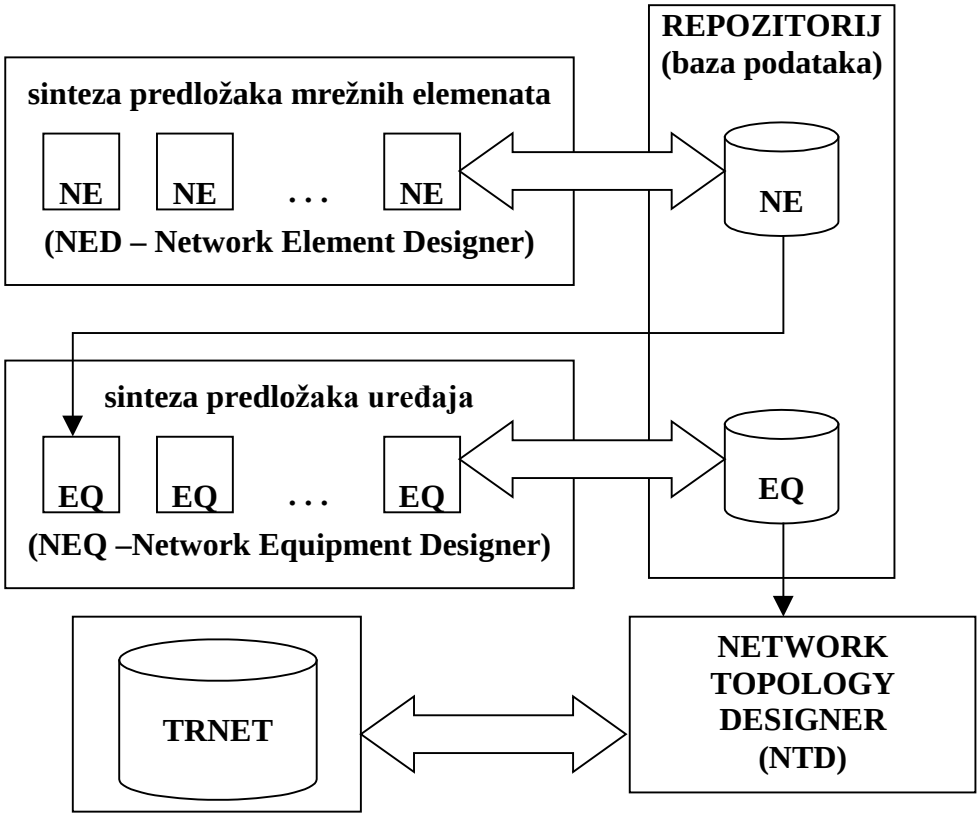


Slika 5.1 – Transport Network Designer – *komponente ekspertnog sustava i interakcija sa bazom podataka transportne mreže*

Temeljna ideja efikasnijeg punjenja baze podataka bazirana je na korištenju predložaka, intuitivnom grafičkom prikazu, aktivnim grafičkim sučeljem prema podacima u bazi podataka i ugrađenim znanjem kojim se nastoji što više automatizirati potrebne zadatke i spriječiti unos krivih podataka.

Čitav postupak shematski je prikazan na slici 5.2. Aplikacijom NED (***N**etwork **E**lement **D**esigner*) kreiramo tzv. predloške mrežnih elemenata i pohranjujemo ih u repozitoriju kako bi bili dostupni drugim klijent-aplikacijama koje koriste repozitorij. Te predloške mrežnih elemenata koristimo pri sintezi predložaka uređaja aplikacijom EQD (*Network **Eq**uipment **D**esigner*). Međusobnim povezivanjem mrežnih elemenata

definiramo funkcionalnost uređaja na transportnom nivou. Gotovi predlošci uređaja također se pohranjuju u repozitorij. Takve gotove predloške uređaja iz repozitorija možemo jednostavno preslikati u proizvoljan broj instancija uređaja u bazi transportne mreže pomoću aplikacije NTD (***N**etwork **T**opology **D**esigner*). Takvo preslikavanje automatski kreira instanciju uređaja, instancije svih pripadnih mrežnih elemenata, referentnih točaka i temeljnih link konekcija. Upravo taj korak ostvaruje cilj ekspertnog sustava TND da se jednostavnom komandom i na točno određeni, automatizirani način na temelju izabranog predloška uređaja iz repozitorija kreira potreban broj instancija uređaja u bazi transportne mreže (TRNET).



Slika 5.2 – Shematski prikaz postupka uporabe TND sustava u svrhu sinteze predložaka i punjenja baze podataka transportne mreže TRNET

Potrebno je naglasiti da repozitorij i pripadne aplikacije funkcioniraju nezavisno od baze transportne mreže (TRNET). Interakcija s bazom transportne mreže dešava se samo pri unosu instancija uređaja u bazu transportne mreže. Prema tome, relacijske sheme i druga rješenja primijenjena u repozitoriju predložaka ne moraju biti ista kao rješenja u bazi transportne mreže, ali je nužno da postoji jednoznačno definirani postupak kojim se elementi repozitorija (odnosno predlošci) preslikavaju u prilagođenom obliku u instancije u bazi transportne mreže.

5.2 Definicije osnovnih pojmova

5.2.1 Entitet, tip entiteta, instancija entiteta

Entitetom smatramo sve ono o čemu želimo prikupljati i pohranjivati informacije. Da bi mogli prikupljati, pohranjivati i obrađivati informacije o nekom entitetu mora

biti poznat njegov identifikator. U relacijskim bazama podataka u istoj relaciji nalaze se instancije entiteta istoga tipa entiteta. Dakle, tip entiteta karakteriziran je relacijskom shemom, a instancija entiteta predstavljena je n-torkom (retkom) u relaciji.

5.2.2 Mrežni element

Mrežni element u informacijskom modelu transportne mreže je osnovni entitet u kojem se obavlja definirana transportna funkcija. Mrežni element je dio uređaja, a može biti i cijeli uređaj ukoliko se uređaj sastoji od samo jednog mrežnog elementa.

Struktura mrežnog elementa opisana je referentnim točkama i konekcijama. Referentne točke mogu biti ulazne ili izlazne u odnosu na mrežni element. Konekcije između ulaznih i izlaznih referentnih točaka unutar mrežnog elementa opisuju funkciju mrežnog elementa i nazivaju se *temeljne podmrežne konekcije*.

Primjeri mrežnih elemenata su prospojnik 63x63 2Mb/s (obavlja funkciju prospajanja), multipleksor 63xVC12 na 1xSTM-1 (obavlja funkciju multipleksiranja), linijski terminal 1x1 (obavlja funkciju prilagodbe signala na liniju ili sa linije) itd.

5.2.3 Tip mrežnog elementa

Mrežni elementi navedeni u primjeru svi su različitog tipa. **Tip mrežnog elementa** određen je funkcijom koju mrežni element obavlja. Svi mrežni elementi koji obavljaju istovrsnu transportnu funkciju istoga su tipa. Npr. prospojnik 1024x1024 64Kb/s i prospojnik 16x16 STM-1 su mrežni elementi koji obavljaju funkciju prospajanja. Dakle, prema tipu to su prospojnici.

5.2.4 Predložak mrežnog elementa

Funkcijska analiza transportne mreže upućuje na zaključak da će se slični mrežni elementi u modelu mreže pojavljivati mnogo puta. Ti mrežni elementi biti će slični po strukturi, ali se mogu nezavisno konfigurirati. Dakle, iako su slični, moraju postojati kao različite instancije – nemaju nužno iste karakteristike. Npr. u mreži će biti puno prospojnika 64x64 VC12, dakle sličnih karakteristika, ali oni će se razlikovati prema prospojenosti, stanju ispravnosti, svaki će imati nezavisno definirane referentne točke itd.

Unos velikog broja sličnih elemenata u bazu transportne mreže, s obzirom na navedene početne postavke, može se optimizirati uvođenjem *predloška mrežnog elementa*. Predložak mrežnog elementa je entitet koji služi kao osnova za stvaranje instancija mrežnih elemenata sličnih karakteristika. Dakle, on služi kao kalup pomoću kojega se kreiraju instancije mrežnih elemenata u bazi transportne mreže. Postupak kojim se predlošci mrežnih elemenata iz repozitorija preslikavaju u mrežne elemente u bazi transportne mreže, biti će detaljnije opisan u nastavku rada.

5.2.5 Uređaj

Svaki mrežni element transportnog nivoa na fizičkom se nivou ostvaruje kao konkretni uređaj ili dio uređaja. Uređaj na transmisijskom nivou predstavlja fizički

element (opremu) mreže čija se funkcionalnost na transportnom nivou može opisati mrežnim elementima međusobno povezanim temeljnim link konekcijama.

5.2.6 Predložak uređaja

Vrijedi slično kao i za predložak mrežnog elementa. Korisnost uvođenja predloška za uređaje dolazi do još većeg izražaja jer je kompleksnost uređaja puno veća od kompleksnosti mrežnog elementa. Kreiranjem uređaja iz predloška uređaja automatski se kreira uređaj i svi mrežni elementi koje uređaj sadrži sa svim referentnim točkama unutar tih mrežnih elemenata i svim temeljnim link konekcijama unutar uređaja.

5.3 Repozitorij predložaka – baza podataka

Repozitorij predložaka je baza podataka koja je dio ekspertnog sustava TND. Ta baza podataka neovisna je o bazi podataka transportne mreže (TRNET). Repozitorij služi kao temeljni informacijski 'bazen' za aplikacije sustava TND.

Njegov sadržaj su:

- podaci o definiranim predlošcima mrežnih elemenata,
- podaci o definiranim predlošcima uređaja sa njihovom unutrašnjom strukturom,
- te podaci potrebni za grafički prikaz istih.

Osnovni elementi (entiteti) pohranjeni u repozitoriju:
(redni broj, naziv relacije, pojašnjenje, aplikacije u kojima se koristi)

1. EQUIPMENT (predložak uređaja) – ***EQD, NTD***
2. BASIC_CONNECTION (temeljna link konekcija) - ***EQD, NTD***
3. NETWORK ELEMENT (mrežni element kao dio predloška uređaja) - ***EQD, NTD***

4. COMB_SPLIT (optički sprežnik ili rasprežnik – predložak mrežnog elementa) – ***NED, EQD, NTD***
5. CONVERTER (pretvorba signala iz opt. u el. ili obratno – predložak mrežnog elementa) – ***NED, EQD, NTD***
6. DROP-INSERT (predložak mrežnog elementa) – ***NED, EQD, NTD***
7. LT (linijski terminal – predložak mrežnog elementa) – ***NED, EQD, NTD***
8. MUX_DEMUX (multipleksor ili demultipleksor – predložak mrežnog elementa) – ***NED, EQD, NTD***
9. OH_IE (dodavanje zaglavlja ili 'skidanje' zaglavlja – predložak mrežnog elementa) – ***NED, EQD, NTD***
10. REG_AMP (regenerator ili pojačalo – predložak mrežnog elementa) – ***NED, EQD, NTD***
11. XC (prospojnik, optički ili električki – predložak mrežnog elementa) – ***NED, EQD, NTD***
12. OPTICAL_FILTER (optički filter – predložak mrežnog elementa) – ***NED, EQD, NTD***

13. WAVELENGTH_GROUP (grupa valnih duljina) – ***NED***
14. WAVELENGTH (valna duljina – pripada grupi valnih duljina) – ***NED***
15. LAYER (šifarnik slojeva) – ***NED, EQD, NTD***

16. CONNECTION_COORDINATES (koordinate za grafički prikaz konekcije) – ***EQD, NTD***

Popis relacija repozitorija s atributima koje sadrže opise gore navedenih entiteta (naziv atributa – tip, pojašnjenje):

1. EQUIPMENT (predložak uređaja)
   EQ_ID – NUMBER, primarni ključ
   EQ_NAME - VARCHAR2(20), unique

2. BASIC_CONNECTION (temeljna link konekcija)
   CONNECTION_ID – NUMBER, primarni ključ
   EQ_ID – NUMBER, strani ključ (EQUIPMENT.EQ_ID)
   PREVIOUS_NE_ID – NUMBER, strani ključ (NETWORK_ELEMENT.NE_ID)
   PREVIOUS_RP_FUNCTION - VARCHAR2(6), vrijednosti: SOURCE, SINK, DROP, INSERT
   PREVIOUS_RP_STARTNUM – NUMBER, s kojom ref. točkom počinje izvor grupe konekcija
   PREVIOUS_RP_NUMBER – NUMBER, redni broj točke potreban za njen grafički prikaz
   NEXT_NE_ID - NUMBER, strani ključ (NETWORK_ELEMENT.NE_ID)
   NEXT_RP_FUNCTION – VARCHAR2(6) , vrijednosti: SOURCE, SINK, DROP, INSERT
   NEXT_RP_STARTNUM – NUMBER, prva ponorna ref. točka za ovu grupu konekcija
   NEXT_RP_NUMBER – NUMBER, redni broj točke potreban za njen grafički prikaz
   NO_OF_CONNECTIONS – NUMBER, broj konekcija
   CONNECTION_TYPE – VARCHAR2(4), vrijednosti: FILC, LILC
   LABEL_X – NUMBER(4,0), koordinate za ispis oznake konekcije
   LABEL_Y – NUMBER(4,0)

3. NETWORK_ELEMENT (mrežni element kao dio predloška uređaja)
   NE_ID – NUMBER, primarni ključ
   NE_TYPE_TABLE - VARCHAR2(20), naziv relacije sa specifičnim podacima za takav tip mrežnog elementa: COMB_SPLIT, CONVERTER, DROP_INSERT, LT, MUX_DEMUX, OH_IE, REG_AMP, XC
   NE_TYPE_ID – NUMBER, strani ključ iz relacije određene sa NE_TYPE_TABLE
   EQ_ID – NUMBER, strani ključ (EQUIPMENT.EQ_ID)
   NE_LABEL - VARCHAR2(20), može biti NULL
   LEFT - NUMBER(4,0), NULL – grafičke koordinate mrežnog elementa unutar predloška uređaja
   TOP - NUMBER(4,0), NULL
   RIGHT - NUMBER(4,0), NULL
   BOTTOM - NUMBER(4,0), NULL

4. COMB_SPLIT (optički sprežnik ili rasprežnik - predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ
   NE_TYPE_NAME - VARCHAR2(30), unique
   FUNCTION - VARCHAR2(8), vrijednosti: COMBINER, SPLITTER
   NO_OF_RP – NUMBER

5. CONVERTER (pretvornik signala iz opt. u el. ili iz el. u opt. - predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ
   NE_TYPE_NAME – VARCHAR2(30), unique
   CONVERSION – VARCHAR2(18), vrijednosti: ELECTRICAL-OPTICAL, OPTICAL-ELECTRICAL
   ELECTRICAL_LAYER – NUMBER, strani ključ(LAYER.LAYER_ID)
   WAVELENGTH_GROUP_ID – NUMBER, strani ključ (WAVELENGTH_GROUP. WAVELENGTH_GROUP_ID)

6. DROP_INSERT (predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ

NE_TYPE_NAME - VARCHAR2(30), unique
DI_LAYER – NUMBER (NULL – optički DI), strani ključ(LAYER.LAYER_ID)
NO_OF_RP – NUMBER
NO_OF_DROP_RP – NUMBER (može biti NULL)

7. LT (linijski terminal - predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ
   NE_TYPE_NAME – VARCHAR2(30), unique
   LT_FUNCTION – VARCHAR2(9), vrijednosti: FROM LINE, ON LINE
   CHARACTERISTIC – VARCHAR2(9),
   INPUT_LAYER_ID – NUMBER, vrijednosti NULL = optical, strani ključ(LAYER.LAYER_ID)
   OUTPUT_LAYER_ID – NUMBER, vrijednosti NULL = optical, strani ključ(LAYER.LAYER_ID)

8. MUX_DEMUX (multipleksor ili demultipleksor - predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ
   NE_TYPE_NAME – VARCHAR2(30), unique
   HIGHER_LAYER – NUMBER, strani ključ(LAYER.LAYER_ID)
   LOWER_LAYER – NUMBER, strani ključ(LAYER.LAYER_ID)
   NO_OF_RP – NUMBER
   SIGNAL_TYPE - VARCHAR2(3), vrijednosti: PDH, SDH (može biti NULL)
   FUNCTION - VARCHAR2(5), vrijednosti MUX, DEMUX

9. OH_IE (dodavanje zaglavlja ili ‘skidanje’ zaglavlja - predložak mrežnog elementa)
   NE_TYPE_ID – NUMBER, primarni ključ
   NE_TYPE_NAME - VARCHAR2(30), unique
   OH_FUNCTION - VARCHAR2(3), vrijednosti: OHI, OHE
   SIGNAL_IN_LAYER – NUMBER, strani ključ(LAYER.LAYER_ID)
   SIGNAL_OUT_LAYER – NUMBER, strani ključ(LAYER.LAYER_ID)

10. REG_AMP (regenerator ili pojačalo - predložak mrežnog elementa)
    NE_TYPE_ID – NUMBER, primarni ključ
    NE_TYPE_NAME – VARCHAR2(30), unique
    FUNCTION – VARCHAR2(11), vrijednosti: REGENERATOR, OFA
    AMPLIFICATION – VARCHAR2(10)
    CHARACTERISTIC – VARCHAR2(9)
    INPUT_LAYER_ID – NUMBER, NULL = optical
    OUTPUT_LAYER_ID – NUMBER, NULL = optical

11. XC (prospojnik - predložak mrežnog elementa)
    NE_TYPE_ID – NUMBER, primarni ključ
    NE_TYPE_NAME - VARCHAR2(30), unique
    XC_LAYER - NUMBER (NULL – optički XC), strani ključ(LAYER.LAYER_ID)
    NO_OF_RP – NUMBER
    NO_OF_INPUT_RP – NUMBER (može biti NULL)
    PROTECTION_SWITCHING - NUMBER, vrijednosti: 0, 1
    POINT_TO_POINT – NUMBER, vrijednosti: 0, 1
    GROUP – NUMBER, vrijednosti: 0, 1
    BROADCAST – NUMBER, vrijednosti: 0, 1

12. OPTICAL_FILTER (optički filter)
    NE_TYPE_ID – NUMBER, primarni ključ
    NE_TYPE_NAME - VARCHAR2(30), unique
    WAVELENGTH_GROUP_ID – NUMBER, strani ključ (WAVELENGTH_GROUP. WAVELENGTH_GROUP_ID)
    TUNABLE - NUMBER(1), vrijednosti 0,1

13. WAVELENGTH_GROUP (grupa valnih duljina)
WAVELENGTH_GROUP_ID - NUMBER, primarni ključ

14. WAVELENGTH (valna duljina)
WAVELENGTH_GROUP_ID - NUMBER, strani ključ (WAVELENGTH_GROUP. WAVELENGTH_GROUP_ID), dio primarnog ključa (1/2)
WAVELENGTH - NUMBER (6) , dio primarnog ključa (2/2)

15. LAYER (šifarnik slojeva)
LAYER_ID – NUMBER, primarni ključ
LAYER - VARCHAR2(10), vrijednosti: 64 Kb, 2 Mb, ..., STM-1, ..., STM-16
SIGNAL_TYPE - VARCHAR2(3), vrijednosti: PDH, SDH

16. CONNECTION_COORDINATES (koordinate za grafički prikaz konekcije)
CONNECTION_ID – NUMBER, atribut primarnog ključa(1), ujedno i strani ključ (BASIC_CONNECTION.CONNECTION_ID)
KNEE_INDEX – NUMBER, atribut primarnog ključa(2)
KNEE_X – NUMBER(4,0), koordinate koljena odnosno točke loma konekcije
KNEE_Y – NUMBER(4,0)

Možemo primijetiti da su neki entiteti opisani u zajedničkoj relaciji (npr. MUX i DEMUX). Takvu organizaciju tih relacija dopušta sličnost opisa nekih entiteta.

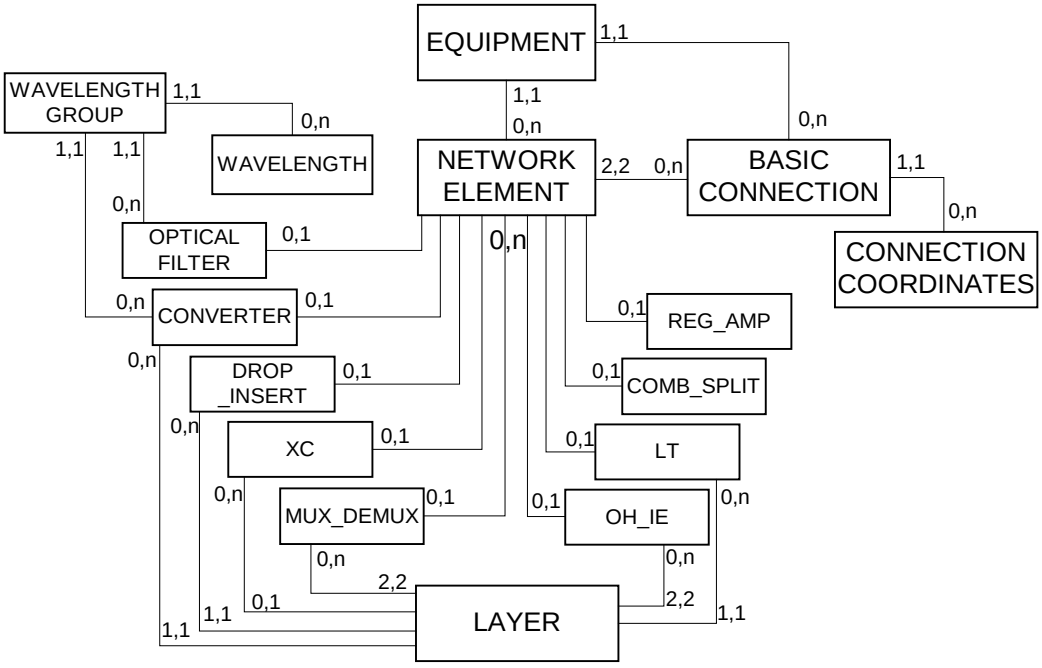
Relacije 4-12 koriste se u definiranju predložaka mrežnih elemenata (aplikacija NED), a relacije 1-16 (sve relacije) koriste se u definiranju predložaka uređaja (aplikacija EQD).

Važno je primijetiti da u repozitoriju nije pohranjen opis mrežnih elemenata niti uređaja već njihovih predložaka koji se kasnije koriste kao temelj za kreiranje instancija mrežnih elemenata i uređaja u bazi podataka TRNET.

Relacija NETWORK_ELEMENT opisuje predloške mrežnih elemenata kao dijelove predloška uređaja. Dakle, ona ne opisuje zajedničke karakteristike svih mrežnih elemenata – nije 'nadtip' mrežnog elementa, već predstavlja sliku (instanciju) predloška mrežnog elementa u predlošku uređaja. Naime, svaki predložak mrežnog elementa opisan u relacijama 4-15 može se više puta pojavljivati u istom predlošku uređaja. Kada predložak mrežnog elementa postane dio predloška uređaja, kreira se instancija mrežnog elementa sa identifikatorom NE_ID u relaciji NETWORK_ELEMENT koja se referencira na predložak mrežnog elementa iz kojega je izvedena pomoću atributa NE_TYPE_ID i NE_TYPE_TABLE. Kao što je već napomenuto, više takvih instancija može se referencirati na isti predložak mrežnog elementa, čime je ispunjen i smisao predloška kao višestruko iskoristivog resursa.

Na slici 5.3 možemo vidjeti odnose između tipova entiteta repozitorija. Brojčanim oznakama je naznačen najmanji i najveći broj instancija entiteta koji mogu pripadati instanciji entiteta sa suprotne strane veze. Pogledajmo odnos NETWORK_ELEMENT – BASIC_CONNECTION: (2,2) – (0,n). Svaka temeljna link konekcija mora imati izvorni i ponorni mrežni element, a svaki mrežni element može biti povezan preko više konekcija (ili nepovezan) sa drugim mrežnim elementima. Temeljne link konekcije zapravo povezuju referentne točke. Iako u repozitoriju ne postoji zasebna relacija za opis referentne točke, informacija o tome koje referentne točke povezuje konekcija nalazi se u atributima PREVIOUS_RP-

_STARTNUMBER i NEXT_RP_STARTNUMBER relacije BASIC_-CONNECTION. Instancija entiteta XC može i ne mora imati pripadajuću instanciju entiteta LAYER. XC opisuje mrežne elemente optičke i električke prospojnike. Ukoliko se radi o optičkom prospojniku, što je naznačeno u posebnom atributu, tada XC nema pripadajući sloj (LAYER). Svaka instancija entiteta NETWORK_-ELEMENT referencira se na točno jednu instanciju relacije koja opisuje taj mrežni element (relacije 4-12). To znači da ne mogu istovremeno postojati sve referencije relacije NETWORK_ELEMENT prema relacijama 4-12, već samo jedna od njih, ovisno o tome kojeg je tipa promatrani mrežni element (NETWORK_ELEMENT). Taj problem referencijskog integriteta biti će detaljnije razrađen u ovom tekstu.



Slika 5.3 – Dijagram veza između tipova entiteta u repozitoriju predložaka

5.3.1 Problem referencijskog integriteta

Relacije koje opisuju predloške mrežnih elemenata (relacije 4-12) imaju atribut jednakog naziva i tipa: NE_TYPE_ID koji im služi kao primarni ključ. Relacija NETWORK_ELEMENT referencira se na predloške mrežnih elemenata koji su definirani u različitim relacijama (4-12). Referencija je određena pomoću atributa NE_TYPE_TABLE koji definira na koju relaciju se referencira i NE_TYPE_ID koji identificira predložak mrežnog elementa unutar specificirane relacije. Takvo pravilo referencijskog integriteta nemoguće je implementirati korištenjem standardnih referencijskih ograničenja na navedene relacije jer referencirana relacija nije unaprijed određena (određuje je vrijednost atributa NE_TYPE_TABLE). Taj problem riješen je kreiranjem odgovarajućih okidača koji se brinu za održanje takvog složenog referencijskog integriteta. Pri brisanju svakog retka iz bilo koje relacije predložaka mrežnih elemenata pripadajući okidač briše sve retke iz relacije NETWORK_ELEMENT sa odgovarajućim NE_TYPE_ID. Isto tako, pri unosu ili izmjeni atributa NE_TYPE_ID i NE_TYPE_TABLE, odgovarajući okidač provjerava postojanje takvog predloška mrežnog elementa u specificiranoj relaciji.

Problem je moguće riješiti i kreiranjem nove relacije NE koja bi sadržavala zajedničke atribute svih predložaka mrežnih elemenata, a to je barem identifikator NE_TYPE_ID. NE_TYPE_ID bio bi primarni ključ u relaciji NE. U relacijama u kojima se opisuju atributi specifični za pojedini tip mrežnog elementa, atribut NE_TYPE_ID bio bi primarni ključ i ujedno strani ključ koji se referencira na NE.NE_TYPE_ID. Relacija NETWORK_ELEMENT referencirala bi se na NE_TYPE_ID iz relacije NE, a informacija o tome u kojoj relaciji su atributi koji nisu zajednički svim mrežnim elementima bila bi u vrijednosti atributa NE_TYPE_TABLE relacije NE.

NE (zajednički atributi svih predložaka mrežnih elemenata)
NE_TYPE_ID – NUMBER, primarni ključ
NE_TYPE_TABLE - VARCHAR2(20), jednoznačno određuje relaciju sa specifičnim podacima za takav tip mrežnog elementa. Vrijednosti: COMB_SPLIT, CONVERTER, DROP_INSERT, LT, MUX_DEMUX, OH_IE, REG_AMP, XC

To je minimalan (nužan) skup zajedničkih atributa predložaka mrežnih elemenata. U ovu relaciju može se dodati još zajedničkih atributa, kao na primjer NE_TYPE_NAME - VARCHAR2(30), unique. Pri određivanju koje atribute ćemo uvrstiti među 'zajedničke' za sve mrežne elemente moramo imati na umu rizik da se u budućnosti pojavi potreba za predloškom mrežnog elementa bez nekog od tih atributa, čime taj atribut više ne bi bio zajednički za sve predloške mrežnih elemenata.

Ovaj primjer alternativnog rješenja problema zahtijeva uvođenje još jedne tablice i mnoge promjene u postojećoj aplikaciji. Osim toga, bazira se na principu nasljeđivanja i iako je ostvariv i u relacijskom sustavu za upravljanje bazom podataka, jasno se vidi da je zbog njegove objektne prirode ovaj problem prikladnije riješiti u objektnom sustavu za upravljanje bazom podataka, no u vrijeme dizajniranja ove baze podataka takav sustav za upravljanje bazom podataka još nije postojao.

5.4 NED aplikacija (*Network Element Designer*)

NED je aplikacija u sklopu TND sustava koja omogućuje [5]:

- kreiranje novih predložaka mrežnih elemenata i njihovu pohranu u repozitorij
- pregled postojećih predložaka mrežnih elemenata
- obradu predložaka mrežnih elemenata pohranjenih u repozitoriju

NED je klijent aplikacija u klijent/poslužitelj odnosu koja preko ODBC API sučelja (***O**pen **D**atab**a**se **C**onnectivity*) pristupa repozitoriju predložaka. Nije ograničen broj NED klijenata koji istovremeno mogu obrađivati podatke repozitorija, što znači da je NED predviđen za simultani rad i dijeljenje (*sharing*) podataka među klijentima.

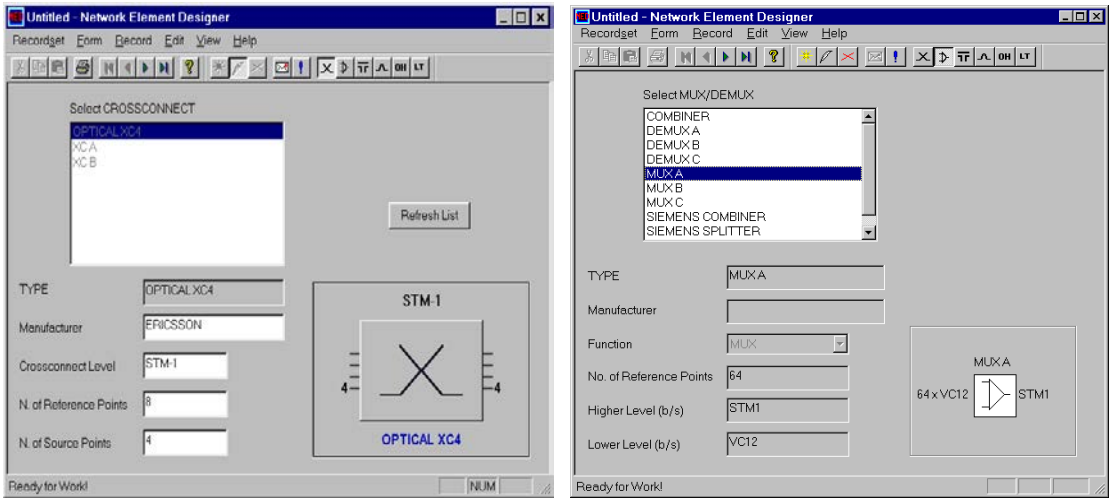
Predlošci mrežnih elemenata pohranjeni su u repozitoriju predložaka u sljedećim relacijama:

4. COMB_SPLIT (COMBINER + SPLITTER) – sprežnik ili rasprežnik
5. CONVERTER – pretvornik signala iz opt. u el. ili iz el. u opt.
6. DROP_INSERT
7. LT – linijski terminal

8. MUX_DEMUX (MUX + DEMUX) – multipleksor ili demultipleksor
9. OH_IE (OHI + OHE) – dodavanje zaglavlja ili ‘skidanje’ zaglavlja
10. REG_AMP (REGENERATOR + AMPLIFIER) – regenerator ili pojačalo
11. XC (prospojnik)
12. OPTICAL_FILTER

Svaka relacija opisuje predloške mrežnih elemenata istog tipa, s time da je s obzirom na sličnost opisa nekih tipova (npr. COMBINER i SPLITTER) izvršeno stapanje relacija i takvi tipovi opisani su u istoj relaciji (COMB_SPLIT).

Aplikacija ima ugrađeno dodatno znanje kojim provjerava unesene podatke i nastoji spriječiti unos neispravnih podataka. Pri unosu podataka predlaže najvjerojatnije vrijednosti, a odgovarajuću vrijednost možete odabrati iz ponuđene liste vrijednosti čime se olakšava i pojednostavljuje unos. Svaki predložak mrežnog elementa prikazan je i grafički sa svojim karakteristikama (slika 5.4). Grafički prikaz mrežnog elementa automatski se generira iz unesenih podataka o mrežnom elementu.



Slika 5.4 – NED aplikacija u radu

5.4.1 Opis osnovnih dijelova aplikacije kako je vidi korisnik

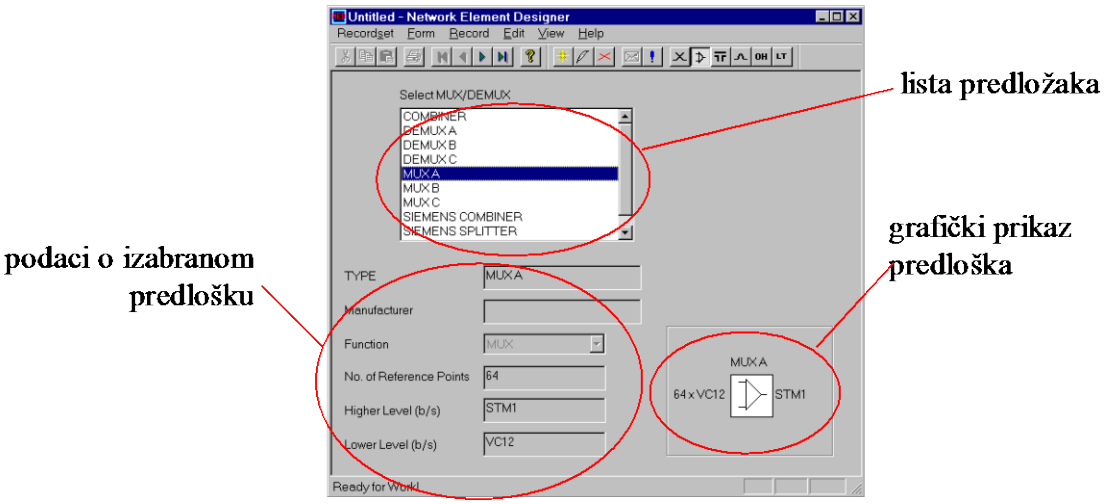
- **Izbornik tipa predloška mrežnog elementa** - Za svaki tip postoji gumb čijim se pritiskom prikaže forma odgovarajućeg sadržaja za izabrani tip.
  Ti gumbi su:

  - CROSSCONNECT
  - MUX/DEMUX
  - DROP-INSERT
  - CONVERTER
  - OVERHEAD INSERTION/EXTRACTION
  - LINE TERMINAL
  - REGENERATOR/AMPLIFIER
  - COMBINER/SPLITTER
  - OPTICAL FILTER

Možemo primijetiti da ovaj izbor korespondira relaciji iz repozitorija koja opisuje traženi tip predloška mrežnog elementa.

- **Forma za odgovarajući tip/relaciju predloška**
  Forme su za svaku relaciju različite, ali sve imaju jasno izdvojena tri dijela (sl. 5.5):

  - **lista postojećih predložaka** – u njoj su navedeni nazivi svih predložaka izabranog tipa (relacije) koji su pohranjeni u repozitoriju. Njome biramo predložak koji želimo vidjeti detaljnije, brisati ili mijenjati.
  - **detaljan prikaz podataka o predlošku** – različit za svaku formu, ovisno o atributima koji opisuju izabrani tip/relaciju
  - **grafički prikaz predloška** – na temelju zadanih podataka o predlošku, automatski se generira njegov simbolički izgled radi veće zornosti prikaza podataka



Slika 5.5 – Glavni dijelovi forme u aplikaciji NED

- **Komande:**

  - **CREATE** – kreiranje novog predloška
  - **EDIT** – promjene karakteristika izabranog predloška
  - **DELETE** – briše izabrani predložak
  - **COMMIT** – provjerava podatke i upisuje izmjenu/unos u bazu podataka (repozitorij)
  - **CANCEL** – poništava tekuću izmjenu/unos

5.4.2 Primjer: kreiranje predloška mrežnog elementa – demultipleksor

Kreiranje predloška demultipleksora koji signal 2 Mb/s demultipleksira na 31x64 kb/s opisati ćemo po koracima:

1) Prvo je potrebno izabrati formu koja odgovara tipu mrežnog elementa. U ovom slučaju to je MUX/DEMUX forma. U listi predložaka možete vidjeti već upisane multipleksore i demultipleksore iz repozitorija (baze podataka).

2) Pritisnite komandu CREATE. Time ulazite u način rada spreman za kreiranje predloška. Otvaraju se prazna polja za unos podataka.

3) Unesite podatke karakteristične za predložak koji želite opisati. Dakle, iz liste izaberite da li je funkcija mrežnog elementa multipleksor ili demultipleksor. Naziv predloška može biti npr. "DEMUX 2M < 31x64k". Tip signala isto se bira iz liste s ponuđenim vrijednostima "PDH" i "SDH", ali ga u ovom slučaju ne treba unositi jer je već upisan "PDH". Sada je potrebno izabrati viši i niži sloj signala iz liste ponuđenih vrijednosti. Ovisno o izboru tipa signala, u tim listama biti će ponuđeni svi mogući slojevi PDH, odnosno SDH. U ovom primjeru za viši sloj izabrati ćemo "2 Mb/s", a za niži "64 kb/s". Broj referentnih točaka neka bude 32 (31 + 1).

4) Pritisnite "COMMIT". Provjerava se logička ispravnost podataka i ako je sve u redu podaci o novom predlošku se unose u bazu podataka i predložak je na raspolaganju ostalim klijentima repozitorija (NED, EQD, TND).

Opisanim postupkom kreirali smo predložak mrežnog elementa i pohranili u bazu podataka – repozitorij predložaka. Te predloške mrežnih elemenata koristimo iz drugih aplikacija (EQD, NTD) za izgradnju predložaka i instancija uređaja.

5.5 EQD aplikacija (*Network **Eq**uipment **D**esigner*)

Kao što smo već napomenuli, transportne funkcije predstavljamo mrežnim elementima. Njihovim međusobnim povezivanjem možemo definirati funkcionalnost uređaja. Uređaj ponekad možemo opisati samo jednim mrežnim elementom, ali češće je realan uređaj potrebno opisati sa više, ponekad i mnogo mrežnih elemenata koje treba povezati temeljnim link konekcijama. Nadalje, broj mrežnih elemenata kojima će biti opisan neki uređaj ovisi i o tome koliko detaljno se žele prikazati funkcije toga uređaja, a to je odluka koja, naravno, ovisi o čovjeku. Opis svakog takvog mrežnog elementa može sadržavati opise stotina referentnih točaka i temeljnih podmrežnih konekcija sadržanih u tom mrežnom elementu. Ova analiza dovodi nas do zaključka da unos uređaja u bazu podataka transportne mreže predstavlja složen i dugotrajan postupak podložan greškama (ljudski faktor).

Da bi smanjili moguće pogreške pri unosu, vrijeme potrebno za unos uređaja, i da bi maksimalno pojednostavnili čitav postupak, potrebno je automatizirati taj postupak procedurama i aplikacijama kako bi ga korisnik (operater) provodio na što jednostavniji način.

Analiza transportne mreže pokazala je da će u bazi transportne mreže biti dosta istih ili sličnih uređaja koji se moraju moći nezavisno konfigurirati. Ta činjenica navodi nas na zaključak da je optimalan unos uređaja u bazu transportne mreže putem predloška uređaja. Osnovna ideja je u tome da predložak uređaja služi kao kalup, uzorak pomoću kojeg bi se moglo jednostavno kreirati i unositi instancije uređaja sličnih karakteristika.

Kreirani predlošci uređaja pohranjuju se u bazi podataka – repozitoriju predložaka, a aplikacija koja upravlja operacijama vezanim uz predloške uređaja je *Network **Eq**uipment **D**esigner* (EQD).

5.5.1 Osnovne postavke

Network Equipment Designer je aplikacija u sklopu ekspertnog sustava TND. Osnovna namjena ove aplikacije je:

- kreiranje novih predložaka uređaja i njihova pohrana u repozitorij
- grafički prikaz postojećih predložaka uređaja
- obrada i promjena konfiguracije predložaka uređaja pohranjenih u repozitoriju
- testiranje ispravnosti strukture predloška uređaja
- grafički dizajn predloška uređaja

Predlošci uređaja pohranjeni su u repozitoriju kako bi bili na raspolaganju drugim aplikacijama sustava TND koje koriste podatke iz repozitorija. Podatke o predlošcima možemo podijeliti u dvije skupine:

- podaci za opis logičke strukture predloška uređaja
- podaci potrebni za grafički prikaz predloška uređaja

Ti podaci pohranjeni su u sljedećim relacijama repozitorija:

1. EQUIPMENT - predložak uređaja
2. BASIC_CONNECTION - temeljna link konekcija
3. NETWORK ELEMENT - mrežni element kao dio predloška uređaja

4. COMB_SPLIT (COMBINER + SPLITTER) – sprežnik ili rasprežnik
5. CONVERTER – pretvornik signala iz opt. u el. ili iz el. u opt.
6. DROP-INSERT
7. LT - linijski terminal
8. MUX_DEMUX (MUX + DEMUX) – multipleksor ili demultipleksor
9. OH_IE (OHI + OHE) – dodavanje zaglavlja ili ‘skidanje’ zaglavlja
10. REG_AMP (REGENERATOR + AMPLIFIER) – regenerator ili pojačalo
11. XC – prospojnik
12. OPTICAL FILTER

13. WAVELENGTH_GROUP – grupa valnih duljina
14. WAVELENGTH – valna duljina
15. LAYER - šifarnik slojeva
16. CONNECTION_COORDINATES - koordinate za grafički prikaz konekcije

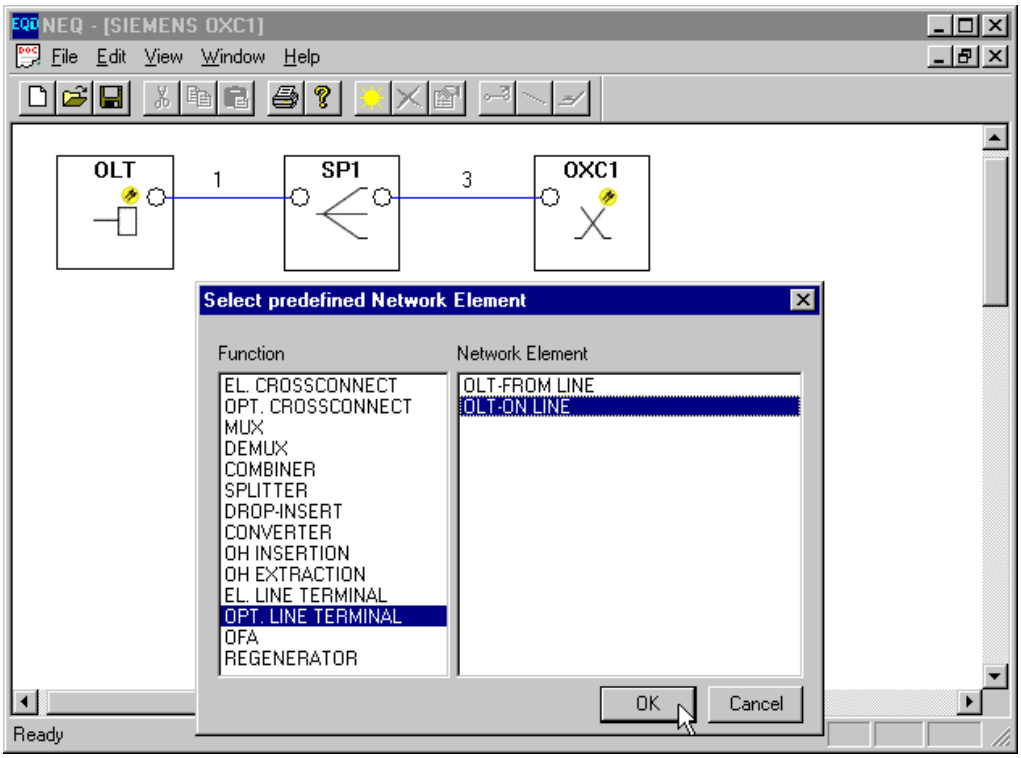
Osnovne su relacije 1-3. One su temelj opisa predloška uređaja. Tome skupu možemo dodati i relaciju CONNECTION_COORDINATES koja služi isključivo kod grafičkog prikaza uređaja. Relacije 4-12 su potrebne radi detaljnijih informacija o predlošcima mrežnih elemenata. Relacije 13 i 14 koriste se za opis optičkog signala tj. valnih duljina kojim se prenosi signal. Takva grupa valnih duljina u nekim (možda i čestim) slučajevima može sadržavati i samo jednu valnu duljinu. Relacija 15 se koristi radi pravilnog ispisa podataka o sloju konekcije ili referentne točke. Dakle, relacije iz kojih EQD koristi podatke su relacije 1-16, a u koje upisuje nove podatke ili izmjene su:

1. EQUIPMENT - predložak uređaja
2. BASIC_CONNECTION - temeljna link konekcija

3. NETWORK ELEMENT - mrežni element kao dio predloška uređaja
16. CONNECTION_COORDINATES - koordinate za grafički prikaz konekcije

Kao što smo već napomenuli, osnovni elementi od kojih u ovoj aplikaciji gradimo predložak uređaja su **mrežni elementi** i **temeljne link konekcije** koje ih povezuju. Temeljne podmrežne konekcije (elementi temeljne statičke fleksibilne konfiguracije – definiraju prospojenost XC mrežnog elementa) nećemo definirati u predlošku uređaja, nego u uređaju u kojeg će predložak biti preslikan. Razlog tome je taj da se svaki uređaj općenito različito konfigurira, ovisno o njegovoj poziciji i zahtjevima u mreži. Referentne točke, kao što smo već objasnili, ne opisujemo u posebnoj relaciji repozitorija već se potrebni podaci koji ih definiraju nalaze u relaciji BASIC_CONNECTION.

Za opis predloška uređaja koriste se postojeći predlošci mrežnih elemenata iz repozitorija kreirani NED aplikacijom i 'ubacuju' se u novi uređaj poput 'lego' kockica. Isti predložak mrežnog elementa možemo iskoristiti više puta u istom ili bilo kojem drugom uređaju. U tom cilju je u relaciji NETWORK_ELEMENT definiran ključ NE_ID nezavisan o predlošku mrežnog elementa identificiranim sa NE_TYPE_ID i NE_TYPE_TABLE. Na taj način se ne moraju posebno kreirati mrežni elementi svaki puta kada nam trebaju, ako predložak odgovarajućih karakteristika već postoji u repozitoriju. Odabravši opciju 'ubacivanja' novog mrežnog elementa, otvara se dijalog koji prikazuje postojeće predloške mrežnih elemenata iz repozitorija svrstane po tipu i funkciji (slika 5.6). Odavde jednostavno mišem izaberemo odgovarajući mrežni element.



Slika 5.6 - 'Ubacivanje' izabranog predloška mrežnog elementa u predložak uređaja

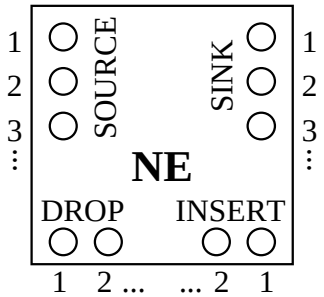
Moramo istaći da relacija NETWORK_ELEMENT ne predstavlja relaciju sa zajedničkim atributima svih mrežnih elemenata, već poseban entitet koji predstavlja

instanciju mrežnog elementa unutar predloška uređaja (vidi odnose između tipova entiteta na slici 5.3). Više takvih 'mrežnih elemenata' mogu kao temelj imati isti predložak mrežnog elementa na koji se referenciraju atributima NE_TYPE_ID i NE_TYPE_TABLE. Zapravo je relativno čest slučaj da uređaj sadrži više istih mrežnih elemenata, odnosno mrežnih elemenata s istim karakteristikama koji nastaju višestrukim korištenjem istog predloška mrežnog elementa. Upravo zbog tog relativno čestog slučaja postoji komanda REPEAT_INSERT_NE kojom se u uređaj ubacuje još jedna instancija zadnjeg ubačenog predloška mrežnog elementa što povećava efikasnost i brzinu izrade predloška uređaja.

Temeljne link konekcije opisane su u relaciji BASIC_CONNECTION. One jednosmjerno spajaju dva mrežna elementa unutar predloška uređaja. Svaka n-torka te relacije, radi optimalnog opisa prilagođenog grafičkom prikazu, zapravo opisuje grupu paralelnih konekcija između dva mrežna elementa. Koliko je konekcija u grupi definirano je atributom NO_OF_CONNECTIONS. Koje točno referentne točke spaja takva konekcija, definirano je atributima PREVIOUS_RP_STARTNUM, PREVIOUS_RP_FUNCTION, NEXT_RP_STARTNUM, NEXT_RP_FUNCTION. Grafičku poziciju referentnih točaka koje omeđuju konekciju određuju atributi PREVIOUS_RP_NUMBER i NEXT_RP_NUMBER na način opisan u tablici 5.1 i prikazan na slici 5.7.

RP_FUNCTION	pridruženje u odnosu na NE	ZNAČENJE U ODNOSU NA KONEKCIJE	Smjer porasta RP_NUMBER
SOURCE	lijevo	ponorna točka za ulazne konekcije u mrežni element	prema dolje
SINK	desno	izvorna točka za izlazne konekcije iz mrežnog elementa	prema dolje
DROP	dole lijevo	izvorna točka za izlazne konekcije iz mrežnog elementa	prema desno
INSERT	dole desno	ponorna točka za ulazne konekcije u mrežni element	prema lijevo

Tablica 5.1 - Grafički položaj referentnih točaka u odnosu na mrežni element



Slika 5.7 – Grafički položaj referentnih točaka u odnosu na predložak mrežnog elementa

Fiksno definirajući stranu mrežnog elementa na kojoj su prikazane referentne točke sa određenom RP_FUNCTION za svaku prikazanu konekciju određen je i njen

smjer koji je na taj način prepoznatljiv iz grafičkog prikaza. Ako je NO_OF_CONNECTIONS veći od 1, paralelizam opisanih konekcija odnosi se na uzastopnost pripadnih ..._RP_NUMBER atributa. Ovakav način označavanja referentnih točaka i grupiranja konekcija će se točno definiranim postupkom prilagoditi pri preslikavanju u odgovarajuće instancije u bazi transportne mreže. Na slici 5.8 možemo vidjeti kako se koristi grafičko sučelje za odabir i promjenu svojstava konekcije. Isto tako treba naglasiti da su u prikazu aplikacijom EQD konekcije koje provode optički signal označene plavom, a konekcije koje provode električki signal crnom bojom.

Na taj način smo opisali unutarnju funkcionalnu strukturu predloška uređaja na transportnom sloju. Opći podaci o predlošku uređaja nalaze se u relaciji EQUIPMENT. U ovom trenutku potreban je jedino atribut naziva predloška uređaja. Prema potrebi lako se mogu dodati dodatni atributi.

5.5.2 Aktivni objekti grafičkog sučelja

U grafičkom sučelju prema predlošku uređaja definirani su sljedeći grafički i logički objekti:

1) Predložak uređaja prikazan je kao prozor (*window*) u kojem su prikazani mrežni elementi međusobno spojeni konekcijama. EQD koristi MDI (***M**ultiple **D**ocument **I**nterface*) arhitekturu, što omogućuje istodoban rad na više ‘otvorenih’ predložaka uređaja.

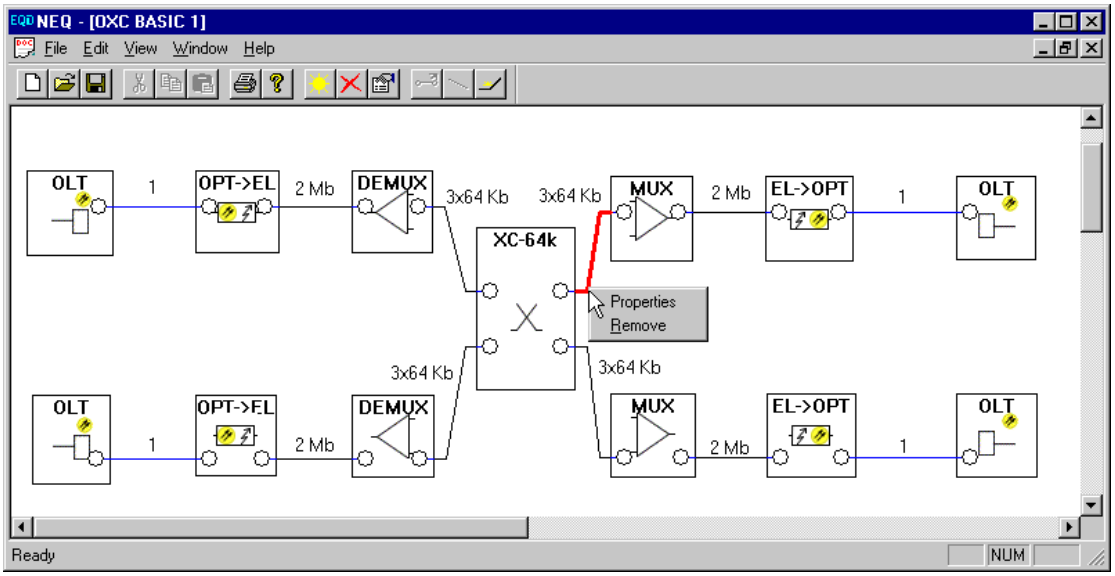
2) Mrežni elementi prikazani su pravokutnicima čije se koordinate prikaza pohranjuju u atribute LEFT, RIGHT, TOP i BOTTOM relacije NETWORK_ELEMENT. To su logičke koordinate sa ishodištem u lijevom gornjem uglu predloška uređaja, a rastu prema desno, odnosno dolje. Svakom mrežnom elementu može se jednostavno mišem mijenjati pozicija i veličina pri čemu navedene promjene slijede sve spojene konekcije. Pozicija se može mijenjati i strelicama na tipkovnici, a finije pomake (1 *pixel*) dobijemo ako držimo tipku SHIFT.

3) Konekcije počinju i završavaju referentnim točkama. Postoji mogućnost ‘lomljenja’ takve konekcije u dužine omeđene točkama ‘loma’ tzv. ‘koljenima’ čime se postiže veća sloboda u grafičkom prikazu. Svakom ‘koljenu’ može se jednostavno mišem mijenjati pozicija u čemu dodatno pomaže i opcija koja poravnava poziciju ‘koljena’ tako da susjedne linije budu sasvim vodoravne ili okomite ukoliko su vrlo blizu vodoravnog/okomitog položaja. Također, jednostavno ih je dodati i maknuti, a njihov broj nije ograničen. Koordinate takvih ‘koljena’ pohranjuju se u relaciji CONNECTION_COORDINATES kojoj je upravo za to i namijenjena.

4) Labela, tj. oznaka konekcije automatski se generira na principu “<broj_konekcija>x<sloj>”, kao npr. “32xVC-12”. Sloj se određuje pretraživanjem relacija koje opisuju mrežne elemente s kojima je konekcija spojena. Ako je broj konekcija 1, tada se on izostavlja i umjesto npr. “1x2Mb” imamo “2Mb”. Konekcije koje prenose optički signal kao oznaku imaju samo broj konekcija u grupi, bez oznake sloja signala. Kao i ostale grafičke elemente, i poziciju oznake konekcije moguće je lako mijenjati. Koordinate sredine donjeg

ruba labele pohranjuju se u atributima LABEL_X i LABEL_Y relacije BASIC_CONNECTION.

5) Referentne točke prikazuju se samo na krajevima konekcija, i imaju već spomenuto značenje grupnih referentnih točaka koje nisu opisane u zasebnoj relaciji. Njih također možemo jednostavno pomicati mišem na točno određene pozicije čime se na najjednostavniji način mijenja podatak o njihovoj poziciji. Taj podatak pohranjen je u atributima PREVIOUS_RP_NUMBER i NEXT_RP_NUMBER relacije BASIC_CONNECTION.



Slika 5.8 – Korištenje grafičkog sučelja aplikacije EQD – odabir i promjena atributa konekcije

5.5.3 Komande raspoložive u aplikaciji

NEW_EQUIPMENT – kreira novi, prazan predložak uređaja. Korisnik (operater) unaša opće atribute uređaja i on je spreman za daljnje definiranje unutarnje strukture.

OPEN_EQUIPMENT – 'otvara' postojeći predložak uređaja. Na posebnoj formi prikazani su predlošci uređaja iz repozitorija od kojih se može izabrati željeni uređaj koji želimo vidjeti, rekonfigurirati ili brisati. Pri otvaranju predloška uređaja čitaju se podaci iz 13 relacija repozitorija (relacije 1-13).

SAVE_EQUIPMENT – načinjene promjene ili novokreirani predložak uređaja pohranjuje u repozitorij kako bi bio dostupan ostalim aplikacijama koje mu pristupaju. Pri pohrani predloška uređaja, podaci se upisuju u 4 relacije repozitorija (relacije 1-3, 13).

SAVE_AS_EQUIPMENT - načinjene promjene ili novokreirani predložak uređaja pohranjuje u repozitorij pod nekim drugim nazivom. Time se redovito kreira novi predložak uređaja. Tom komandom možemo lako kreirati slične predloške uređaja na način da otvorimo neki postojeći predložak, učinimo izmjene koje ga razlikuju od novog predloška uređaja i pohranimo novonastali predložak uređaja.

INSERT_NE – ‘ubacuje’ predložak mrežnog elementa u predložak uređaja čime on postaje novi entitet sa vlastitim identifikatorom. Predložak se bira u posebnoj formi koja prikazuje sve postojeće predloške mrežnih elemenata iz repozitorija svrstane po tipu i funkciji. Isti predložak mrežnog elementa može se više puta ‘ubaciti’ u uređaj čime nastaje više različitih instancija mrežnih elemenata temeljenih na istom predlošku.

REPEAT_INSERT_NE – ‘ubacuje’ još jedan predložak mrežnog elementa koji je zadnji ubačen u predložak uređaja. Ova komanda praktična je ako uređaj sadrži više mrežnih elemenata istih karakteristika, odnosno koji potiču od istog predloška mrežnog elementa, zato što preskače korak u kojem se otvara dijalog za izbor predloška mrežnog elementa, već direktno ubacuje onaj predložak mrežnog elementa koji je zadnji ubačen.

REMOVE_NE – briše mrežni element iz predloška uređaja. Time se ne briše predložak mrežnog elementa na kojem je mrežni element temeljen.

NE_PROPERTIES – u posebnoj formi prikazuje svojstva mrežnog elementa od kojih se neka ovdje mogu i mijenjati. Zbog specifičnosti svakog tipa mrežnog elementa, za svaki tip mrežnog elementa postoji zasebna forma. Dakle, za sada ih ima 8.

MAKE_CONNECTION – kreira novu konekciju između dvaju mrežnih elemenata. Prije izbora ove komande mora biti odabran izvorni mrežni element. Odabirom ove komande mrežni elementi s kojima je izvorni mrežni element moguće povezati iscrtani su zelenom, a oni s kojima se ne može povezati (npr. zbog različitih slojeva, nedostatka slobodnih referentnih točaka, različitog tipa signala itd.) crvenom bojom. Nakon što mišem izaberete ponorni mrežni element, otvara se forma u kojoj možete izmijeniti predložene parametre konekcije i ukoliko tako zadana konekcija prođe provjere ispravnosti, konekcija se kreira i prikazuje.

REMOVE_CONNECTION – briše izabranu konekciju

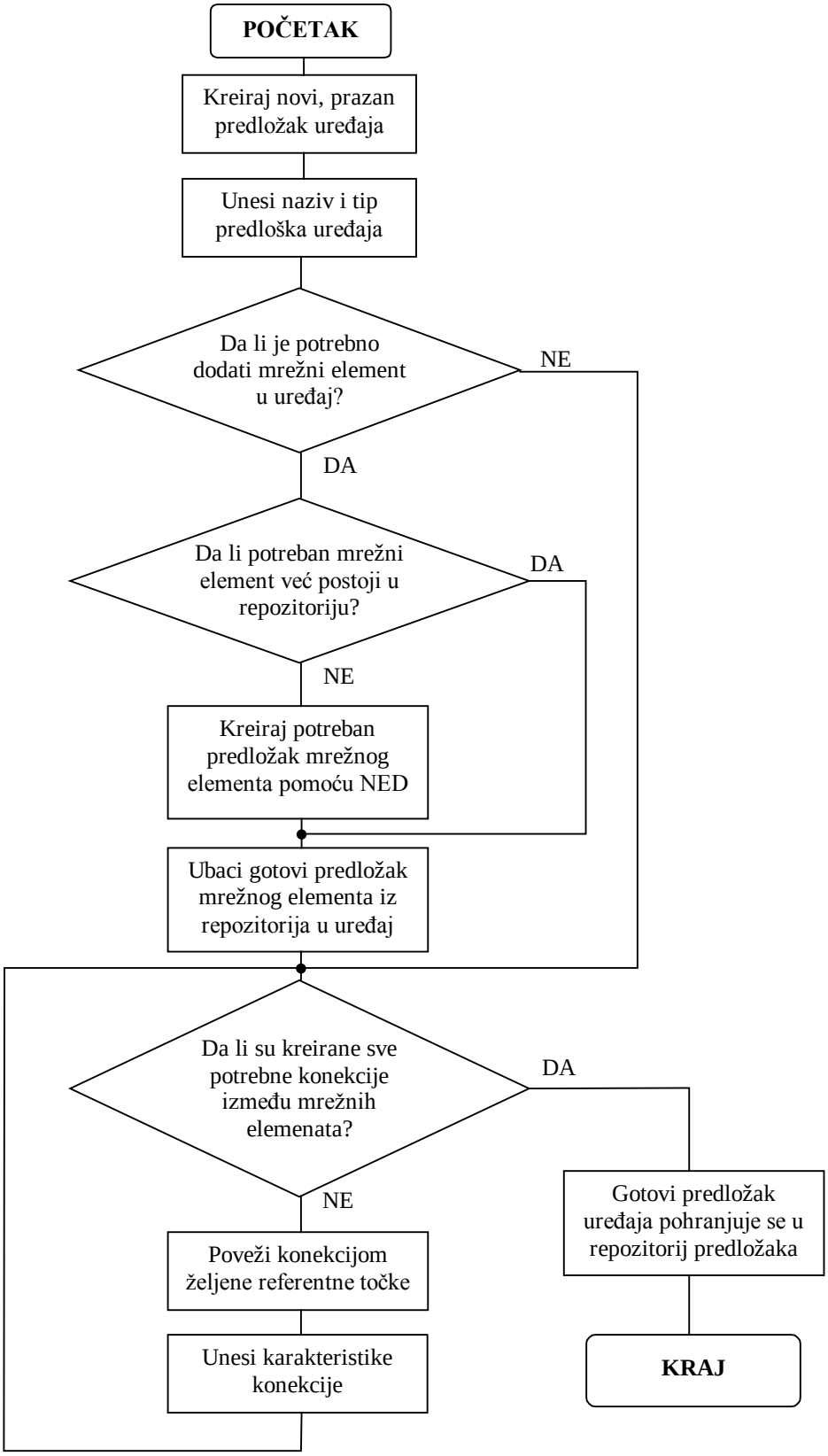
CONNECTION_PROPERTIES – otvara formu na kojoj možete vidjeti i/ili izmijeniti parametre konekcije. Moguće je mijenjati i koje referentne točke konekcija povezuje.

ADD_KNEE – konekciji dodaje podesivu točku loma, tzv. ‘koljeno‘. Broj koljena nije ograničen.

REMOVE_KNEE – kao što joj i naziv govori, ova komanda briše koljeno (ukoliko postoji).

5.5.4 Kreiranje predloška uređaja

Općeniti postupak pri kreiranju predloška uređaja skiciran je dijagramom toka na slici 5.9. Kroz taj postupak proći ćemo na primjeru kreiranja predloška SM uređaja. Radi jednostavnosti, na nekim mjestima ću upotrebljavati riječ “uređaj” umjesto “predložak uređaja” i “mrežni element” umjesto “predložak mrežnog elementa” na mjestima gdje se iz konteksta može zaključiti o kojem se pojmu radi i gdje ne može doći do zabune.



Slika 5.9 – Postupak kreiranja predloška uređaja

Odaberimo komandu kreiranja novog predloška uređaja. Nakon unosa naziva predloška "SM basic" otvara se prazan uređaj. Kreiranje smo mogli provesti i tako da otvorimo postojeći predložak sličnog uređaja kao polazišnu točku. Uređaj je sada spreman za unos mrežnih elemenata i konekcija.

Potrebne mrežne elemente kreiramo iz postojećih predložaka mrežnih elemenata pohranjenih u repozitoriju. Na slici 5.6 vidimo kako se odabire potrebni predložak mrežnog elementa od raspoloživih predložaka u repozitoriju. U našem slučaju za početak će nam biti potrebna četiri optička linijska terminala. Pošto linijski terminal odgovarajućih karakteristika već postoji u repozitoriju, možemo ga jednostavno 'ubaciti' četiri puta u uređaj. Nakon toga biti će nam potreban optički prospojnik u svrhu zaštitnog prospajanja, pa i njega odaberemo i na temelju njega kreiramo mrežni element.

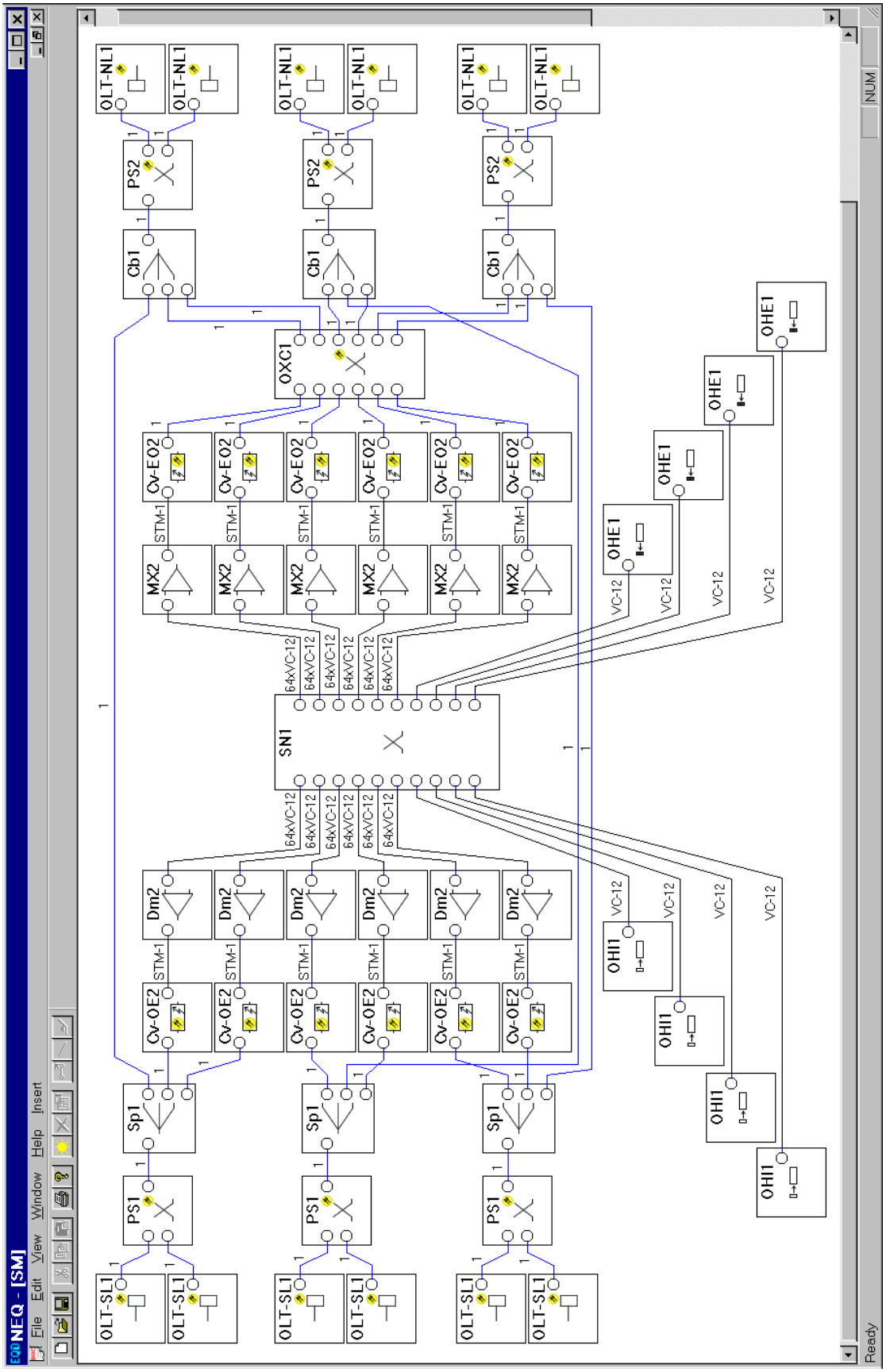
Sada možemo povezati linijski terminal i prospojnik konekcijom. Odaberemo linijski terminal i kliknemo na komandu "MAKE CONNECTION". Primijetimo da je zaštitni prospojnik iscrtan zelenom bojom što znači da je moguće odabrani linijski terminal spojiti s njime. Kliknemo na zaštitni prospojnik. Otvara se dijalog sa parametrima konekcije. Broj konekcija neka ostane 1, tip je "PILC" jer ju je moguće fizički identificirati. Stisnemo "OK" i konekcija je kreirana. Na taj način možemo kreirati i ostale konekcije.

Nakon što smo na opisani način kreirali sve potrebne mrežne elemente i konekcije, možemo grafički urediti prikaz takvog uređaja poravnavanjem mrežnih elemenata, dodavanjem koljena konekcijama itd.

Postupak se završava provjerom unesenih podataka i pohranjivanjem novonastalog predloška u repozitorij komandom "SAVE_EQUIPMENT". Kreirani uređaj time postaje raspoloživ svim EQD klijentima i aplikacijama koje bi mogle koristiti takav predložak, kao što su npr. NTD (***N**etwork **T**opology **D**esigner*) pomoću kojeg bi se predložak uređaja preslikavao u uređaje u bazi podataka transportne mreže. Na slici 5.10 vidimo gotov predložak SM uređaja onako kako izgleda u aplikaciji EQD.

Primjena EQD i NED u opisu optičkih elemenata mreže elaborirana je u [6].

Nakon što smo kreirali predloške uređaja i pohranili ih u repozitorij, takve predloške možemo koristiti za kreiranje instancija uređaja u TRNET bazi podataka uz pomoć aplikacije NTD.



Slika 5.10 – Predložak SM uređaja kreiranog aplikacijom EQD

5.6 NTD aplikacija (*Network Topology Designer*)

5.6.1 Osnovne postavke

Rezultat ili izlaz aplikacije EQD jesu gotovi predlošci uređaja čiji je informacijski opis pohranjen u bazi podataka - repozitoriju predložaka. Opis takvih predložaka uređaja ima kompleksnu strukturu jer mora sadržavati informacijski opis uređaja na više razina: razini mrežnih elemenata, referentnih točaka i internih link konekcija kao i informacije potrebne za njihovu grafičku prezentaciju. Dodatnu kompleksnost unosi i činjenica da mrežni elementi različitih vrsta imaju različita svojstva i funkcionalnost, i opisuju se različitim atributima. Opisi različitih vrsta mrežnih elemenata pohranjuju se u različitim tablicama, pa tako u TRNET bazi imamo 15 tablica samo za opis različitih vrsta mrežnih elemenata.

Imajući, dakle, takav opis predložaka uređaja u repozitoriju predložaka, javlja se potreba za aplikacijom kojom bi mogli iskoristiti takav predložak za njegovu osnovnu namjenu, a to je kreiranje instancija uređaja u bazi podataka transportne mreže (TRNET). Postupak kreiranja instancije uređaja u TRNET bazi je onaj krajnji korak i zapravo najvažnija funkcija TND sustava kojom on obavlja svoju funkciju prvog punjenja TRNET baze. Pri tom automatiziranom i za korisnika jednostavnom procesu čitaju se podaci o predlošku uređaja iz 16 tablica repozitorija, podaci se transformiraju i generiraju se, ovisno o kompleksnosti uređaja, tisuće ili čak deseci tisuća entiteta u 20 tablica TRNET baze. Ručni unos tih entiteta od kojih svaki u prosjeku ima desetak atributa je gotovo nezamisliv, ne samo zbog velike količine čovjek-vremena potrebne za unos, nego i zbog nemale vjerojatnosti pogrešaka koje bi na tako velikom ručnom unosu podataka bile neizbježne.

U ovom poglavlju biti će opisana NTD aplikacija, njene funkcije, a djelomično i neka tehnološka rješenja koja su primijenjena.

5.6.2 Glavne funkcije NTD aplikacije

Aplikaciju možemo najbolje opisati tako da opišemo funkcije koje ona obavlja, odnosno čemu služi i koje su njene mogućnosti. Funkcije koje NTD aplikacija obavlja možemo podijeliti u nekoliko skupina:

- generiranje instancija uređaja u TRNET bazi pomoću predložaka iz TND repozitorija
- grafički prikaz topologije transportne mreže iz njenog opisa pohranjenog u bazi podataka transportne mreže (TRNET)
- povezivanje postojećih uređaja u TRNET bazi konekcijama tipa LLC i LC

5.6.3 Kreiranje instancija uređaja pomoću predložaka

Kao što je već napomenuto, ovo je jedna od najvažnijih funkcija čitavog TND sustava jer se time ispunja i osnovni cilj TND sustava a to je omogućiti efikasno prvo punjenje baze podataka transportne mreže. Njegova važnost leži i u tome da od ukupnog broja entiteta u svim tablicama TRNET baze velika je većina entiteta koji

sudjeluju u opisu instancija uređaja, a ostalih entiteta kvantitativno ima puno manje (npr. broj mreža i podmreža, čvorova može biti za nekoliko redova veličina manji od broja referentnih točaka). Ovaj izuzetno kompleksan proces s gledišta korisnika izgleda relativno jednostavno upravo zahvaljujući TND sustavu.

Taj proces kreiranja instancija uređaja u TRNET bazi pomoću predložaka uređaja iz repozitorija obavlja aplikacija NTD (***N**etwork **T**opology **D**esigner*). NTD povezuje bazu podataka – repozitorij predložaka sa bazom podataka transportne mreže TRNET. NTD čita podatke o danom predlošku iz svih 16 tablica repozitorija, transformira ih u odgovarajući oblik i na temelju tih podataka, ugrađenog znanja i informacija od samog korisnika kreira instanciju uređaja i čitave njegove strukture u TRNET bazi podataka.

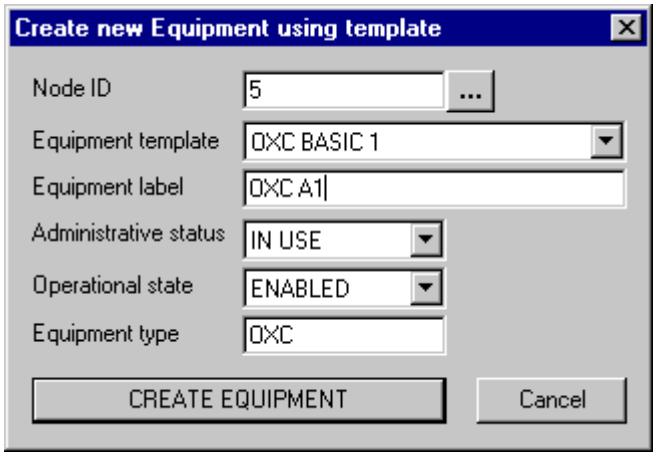
U repozitoriju predložaka opis uređaja je u komprimiranom obliku, u smislu da nije opisana pojedinačna konekcija, već se konekcije grupiraju u grupe konekcija s istim svojstvima. Slično je i s mrežnim elementima jer se isti predložak mrežnog elementa može više puta upotrijebiti u predlošku uređaja. Referentne točke su opisane grupno, zajedno sa svakom grupom konekcija. Tako da pri kreiranju instancije uređaja dolazi do značajne ekspanzije i broj entiteta koji nastaju u TRNET bazi za opis instancije uređaja je često puno veći nego u repozitoriju za opis predloška.

Neki mrežni elementi koji imaju slične atribute u repozitoriju su opisani u istoj tablici (npr. MUX i DEMUX su opisani u tablici MUX_DEMUX, COMBINER i SPLITTER u COMB_SPLIT tablici itd..), dok je u TRNET bazi podataka svaki tip mrežnog elementa opisan u zasebnoj tablici. Zbog takvih i drugih razlika u strukturi veći je i broj tablica u TRNET bazi koje treba popuniti pri unosu nekog uređaja i taj broj iznosi do 20, ovisno o tome od koliko vrsta mrežnih elemenata se uređaj sastoji.

U samom procesu kreiranja instancije uređaja istovremeno se odvijaju tri procesa:

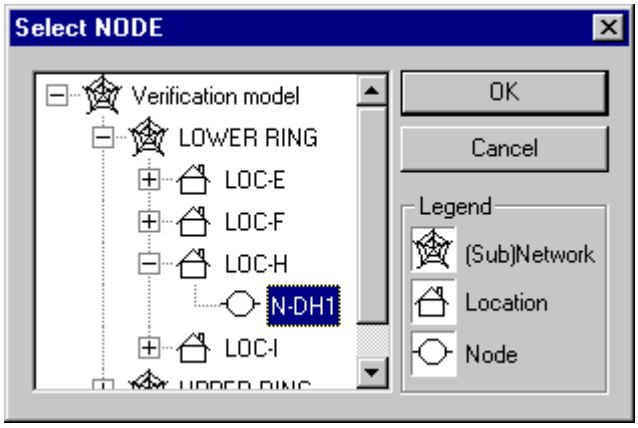
1. čitanje strukture predloška uređaja,
2. konverzija strukture predloška,
3. generiranje novih entiteta u bazi podataka transportne mreže koji sudjeluju u opisu instancije uređaja

Preko posebnog dijaloga, korisnik bira predložak uređaja i unosi ostale podatke potrebne za kreiranje određene instancije uređaja (slika 5.11). Odabir predloška vrši se izborom iz ponuđene liste predložaka uređaja iz repozitorija. Unose se i ostali parametri instancije kao što su njegova oznaka (labela), administrativni i operacijski status, i njegov tip. Svi podaci koji se unose biraju se iz liste ponuđenih vrijednosti tamo gdje je moguće ponuditi takvu listu, čime se olakšava i ubrzava unos. U istu svrhu su u polja već upisane inicijalne vrijednosti za koje se pretpostavlja da su najčešći izbor.



Slika 5.11 – Dijalog TND aplikacije za kreiranje instancije uređaja u kojem korisnik odabire predložak uređaja i unosi druge potrebne podatke

Izbor identifikatora čvora u kojem se generira instancija uređaja je također olakšan posebnim dijalogom za odabir čvora (slika 5.12). U tom dijalogu najvažniju ulogu ima stablo u kojem su hijerarhijski prikazane mreže i podmreže, lokacije i napokon čvorovi koji su već uneseni u TRNET bazi podataka. Mreže, podmreže, lokacije i čvorovi označene su i odgovarajućim grafičkim simbolima. Korisnik jednostavno odabire odgovarajuću mrežu, podmrežu, lokaciju i tako ulazi u sve niže razine sve dok napokon ne dođe do odgovarajućeg čvora kojeg odabire i čiji se identifikator upisuje u odgovarajuće polje kao povratna informacija.



Slika 5.12 – Odabir čvora u kojem će se kreirati instancija uređaja

Nakon što je odabran predložak, čvor i uneseni ili promijenjeni drugi parametri, pritiskom na tipku CREATE EQUIPMENT počinje proces generiranja instancije uređaja.

Na temelju podataka u predlošku i podataka koje je unio korisnik najprije se generira sam entiteta uređaja. Zatim se čitaju specifični podaci o svakom pojedinom mrežnom elementu. To je dosta zahtjevan proces. Različiti tipovi mrežnih elemenata opisani su različitim atributima i u različitim tablicama, odnosno, svaki tip mrežnog elementa ima specifičnu proceduru kreiranja instancije, sa različitim postupcima konverzije podataka, razdvajanja opisa u zajedničkim tablicama i odlučivanja o vrijednostima pojedinih atributa koji nisu direktno opisani u predlošku mrežnog

elementa. Dakle, 10 tipova entiteta repozitorija koji opisuju predloške mrežnih elemenata ovim se postupkom preslikavaju u 16 tipova entiteta u TRNET bazi podataka koji opisuju instancije mrežnih elemenata.

Za svaki predložak mrežnog elementa generiraju se i referentne točke. Referentne točke u repozitoriju nisu opisane u zasebnoj tablici, već se potrebni podaci crpe iz informacija o predlošcima mrežnih elemenata i grupa konekcija koje se spajaju na taj predložak mrežnog elementa. Vrijednosti atributa generiranih referentnih točaka određuju se na temelju podataka o specifičnom mrežnom elementu, odnosno broju *source*, *sink*, *drop* i *insert* referentnih točaka, tipu mrežnog elementa i ostalim podacima iz mrežnog elementa, za što je potrebno određeno znanje koje je ugrađeno u sustav.

Grupe konekcija opisane u tablici repozitorija BASIC_CONNECTION se ekspandiraju u jednu ili više instancija konekcija u TRNET bazi na temelju podataka o konekciji. Pritom je bilo potrebno posebno voditi računa o identifikatorima referentnih točaka. Naime, aplikacija NTD generira sve identifikatore i vodi računa o tome koji se predložak mrežnog elementa preslikao u koje instancije mrežnih elemenata. Na koje referentne točke se konekcije zapravo trebaju spajati što nikako nije trivijalan problem.

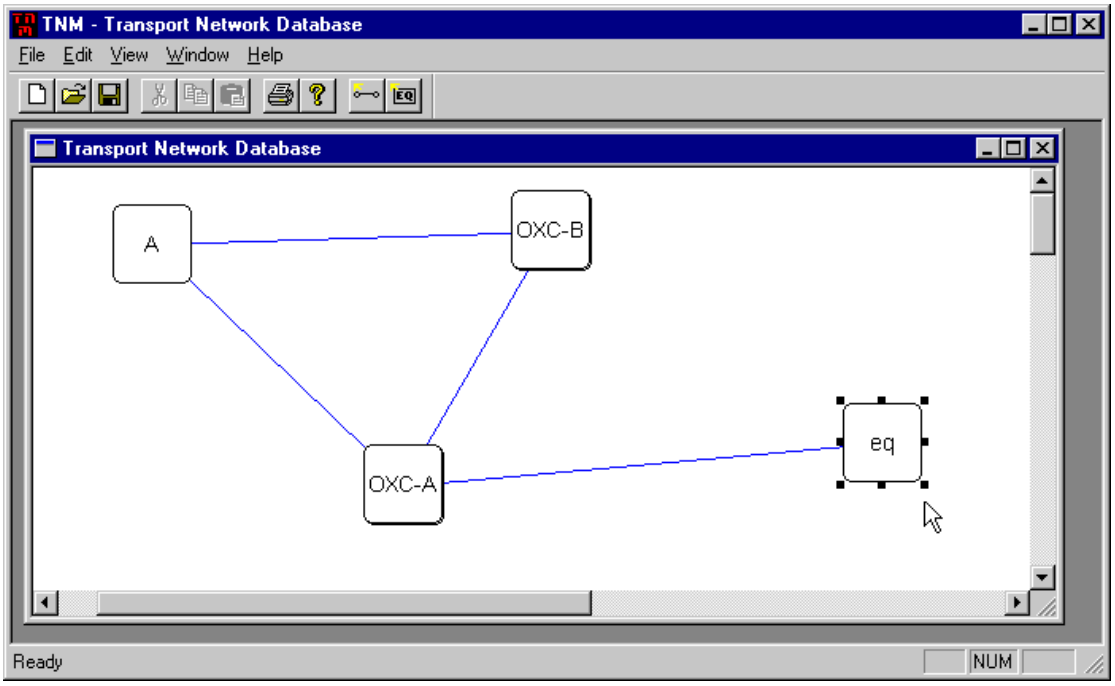
Cijeli postupak je zaštićen unutar transakcije, odnosno, ako dođe do pogreške nakon što su već kreirani neki entiteti, transakcija se "odmota natrag" (*rollback*) kako u TRNET bazi ne bi ostala djelomično kreirana instancija uređaja.

Zbog tako velike količine podataka i entiteta koje treba kreirati, i konverzija koje treba napraviti, za takvo kreiranje instancije uređaja računalu treba i određeno vrijeme - npr. 30 sekundi za kompleksnije uređaje, ovisno o kompleksnosti uređaja i brzini hardvera na kojem se izvodi navedeni proces, što je zanemarivo u usporedbi s vremenom koje bi trebalo za manualni unos.

5.6.4 Grafički prikaz topologije transportne mreže

Aplikacija NTD omogućuje i prikaz topologije transportne mreže na temelju postojećih podataka u TRNET bazi. Ovaj prikaz je prikaz čitave mreže na razini uređaja (vidi sliku 5.13). Uređaji su predstavljeni kvadratima sa zaobljenim rubovima, dok su konekcije između dva uređaja grupirane i prikazane jednom plavom linijom između uređaja. Ovakav prikaz je zapravo aktivno grafičko sučelje prema bazi podataka jer su uređaji i konekcije aktivni elementi grafičkog sučelja i moguće im je pristupati i mijenjati karakteristike preko njega. Npr. moguće je mijenjati poziciju i veličinu uređaja, pristupati njihovim podacima, mišem birati uređaje koje ćemo spojiti konekcijom (biti će opisano u nastavku). Isto tako klikom na prikazanu konekciju dobivamo informacije o konekcijama koje prikazana linija predstavlja.

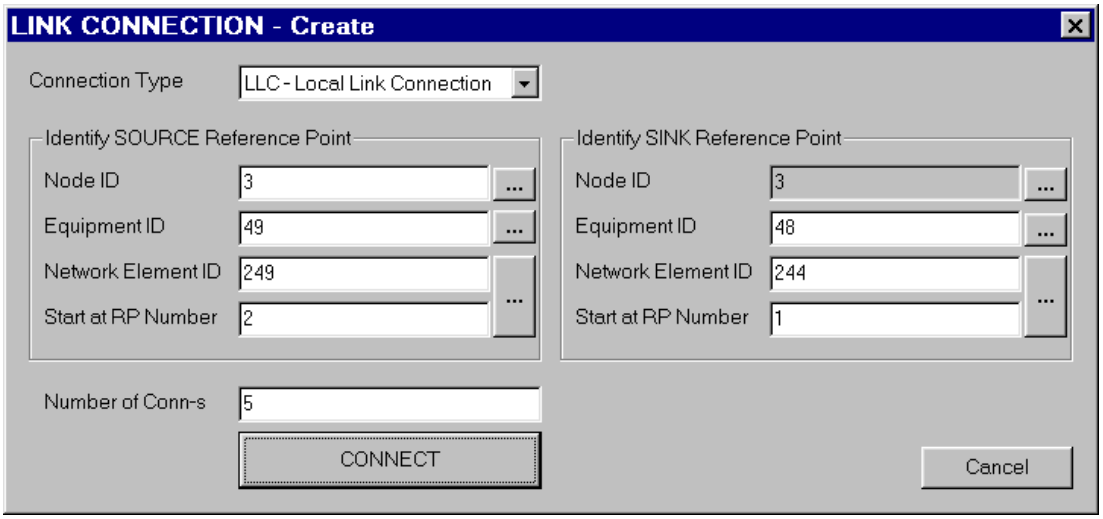
Podaci za grafički prikaz također postoje, ali ne kao sastavni dio TRNET baze podataka jer to ne bi imalo smisla, nego izdvojeno, jer su te informacije zapravo vezane uz TND sustav. Ako za pojedini uređaj nema podataka potrebnih za prikaz, NTD aplikacija ga smješta na prvo slobodno mjesto na ekranu. Nakon što korisnik eventualno promijeni poziciju ili veličinu prikazanog uređaja, promjene se pohranjuju u za to namijenjenu tablicu.



Slika 5.13 – Prikaz unesenih instancija uređaja i konekcija u TRNET bazi podataka kako izgleda u NTD aplikaciji

5.6.5 Povezivanje uređaja konekcijama

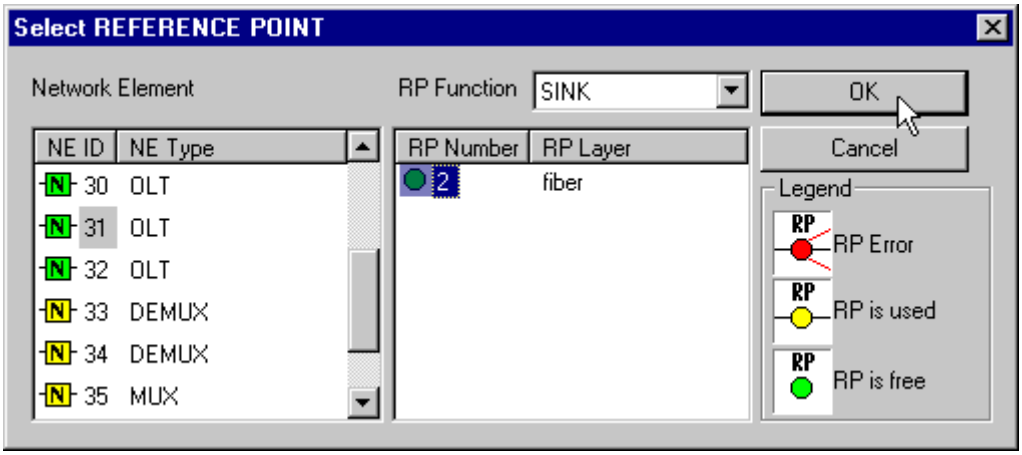
NTD aplikacija omogućuje i povezivanje kreiranih instancija uređaja konekcijama. To su konekcije tipa LLC (***L**ocal **L**ink **C**onnection*) ako se radi o uređajima unutar istog čvora, ili LC (***L**ink **C**onnection*) ako se radi o uređajima koji pripadaju različitim transportnim čvorovima.



Slika 5.14 – Dijalog za kreiranje LC i LLC konekcija u TRNET bazi podataka

Korisnik mišem odabire uređaje koje želi povezati (vidi sliku 5.13) i odabire komandu za kreiranje konekcija. Budući da svaka konekcija zapravo povezuje dvije referentne točke, potrebno je odrediti te referentne točke. Njih određuje korisnik uz pomoć dijaloga prikazanog na slici 5.14. U tom dijalogu su automatski uneseni

identifikatori odabranih uređaja i pripadajućih čvorova, kao i vrsta konekcije. Da bi lakše odabrali mrežne elemente i konačno referentne točke koje treba povezati, klikom na gumb pored tih polja pojavljuje se dijalog za odabir referentnih točaka prikazan na slici 5.15. Korisniku se olakšava izbor na način da su prikazani mrežni elementi odabranog uređaja sa slobodnim referentnim točkama naznačeni zelenom bojom, a oni koji su zauzeti žutom. Nakon što korisnik izabere referentne točke, automatski se upisuje maksimalan broj konekcija koje je moguće kreirati između odabranih mrežnih elemenata, počevši od odabranih referentnih točaka. Korisnik može modificirati taj broj ako želi kreirati manje konekcija. Pritiskom na CONNECT (vidi sliku 5.14) aplikacija kreira željeni broj konekcija i postavi odgovarajuće atribute. Ako se radi o LC konekciji, aplikacija dodatno pronalazi u kojim se mrežama nalaze povezani uređaji i upisuje i te atribute.



Slika 5.15 – Odabir referentne točke prilikom kreiranja konekcije

Temeljne podmrežne konekcije (konekcije unutar mrežnih elemenata) se u TRNET modelu transportne mreže ne opisuju jer su jednoznačno definirane samim mrežnim elementom. Izuzetak su mrežni elementi drop/insert i prospojnik. Temeljnim podmrežnim konekcijama u prospojniku definira se statička fleksibilna konfiguracija mreže, a budući da je ovaj model za sada ograničen na statičku nefleksibilnu konfiguraciju mreže, za sada nije podržan unos tih konekcija TND sustavom.

5.7 Komentar

Sve tri opisane aplikacije sustava su razvijene, testirane i potpuno funkcionalne. Oznaka njihove verzije je 2.0. Aplikacije sustava TND omogućuju korisniku jednostavan dizajn predložaka mrežnih elemenata i uređaja i kreiranje njihovih instancija u TRNET bazi podataka. Entiteti koje sustav kreira u TRNET bazi su:

1. EQUIPMENT
2. NETWORK_ELEMENT
3. REFERENCE_POINT
4. BASIC_LINK_CONNECTION
5. LINK_CONNECTION
6. NE_COMBINER
7. NE_CONVERTER
8. NE_DEMULTIPLEXER
9. NE_ELECTRICAL_DROP_INSERT
10. NE_ELECTRICAL_LINE_TERMINAL

11. NE_MULTIPLEXER
12. NE_OVERHEAD_EXTRACTION
13. NE_OVERHEAD_INSERTION
14. NE_OPTICAL_DROP_INSERT
15. NE_OPTICAL_FIBRE_AMPLIFIER
16. NE_OPTICAL_LINE_TERMINAL
17. NE_OPTICAL_CROSSCONNECT
18. NE_REGENERATOR
19. NE_SPLITTER
20. NE_CROSSCONNECT

Detaljniji popis relacija s atributima je u prilogu ovog rada.

Aplikacija NTD zahtijeva da su prije unosa instancija uređaja već uneseni entiteti iznad razine uređaja (čvorovi, podmreže i mreže, vidi sliku 3.4 na str. 37). Unos podataka o transportnim čvorovima i (pod)mrežama za sada nije podržan i planira se za buduće verzije jer je brojnost čvorova, mreža i podmreža vrlo mala naspram brojnosti uređaja, mrežnih elemenata, a pogotovo referentnih točki i temeljnih link konekcija, pa ih je moguće i «ručno» unijeti.

Aplikacije NED, EQD i NTD razvijene su u *MS Visual C++* za *Windows* platforme (Win95, 98, Me, NT, 2k) [4]. Kao programsko sučelje (API – ***A****pplication* ***P****rogramming* ***I****nterface*) prema bazama podataka (repozitoriju i bazi transportne mreže) korišten je ODBC (***O****pen* ***D****ata****b****ase* ***C****onnectivity*) i odgovarajuće MFC klase (***M****icrosoft* ***F****oundation* ***C****lasses*) [1]. Repozitorij je implementiran u ORACLE RDBMS, verzija 8.0.

6. VERIFIKACIJA

Cilj verifikacije je provjeriti i pokazati na konkretnom modelu transportne mreže korištenje aplikacija TND sustava u svrhu modeliranja predložaka mrežnih elemenata i uređaja i na kraju izvršiti prvo punjenje baze podataka transportne mreže koristeći aplikacije sustava. Upotrebom predložaka i korištenjem grafičkog sučelja postupak prvog punjenja TRNET baze postaje brz i jednostavan, dok su zbog znanja ugrađenog u sustav pogreške pri unosu svedene na minimum. Nad unesenim podacima mogu se vršiti razni upiti da bi se provjerila njihova ispravnost.

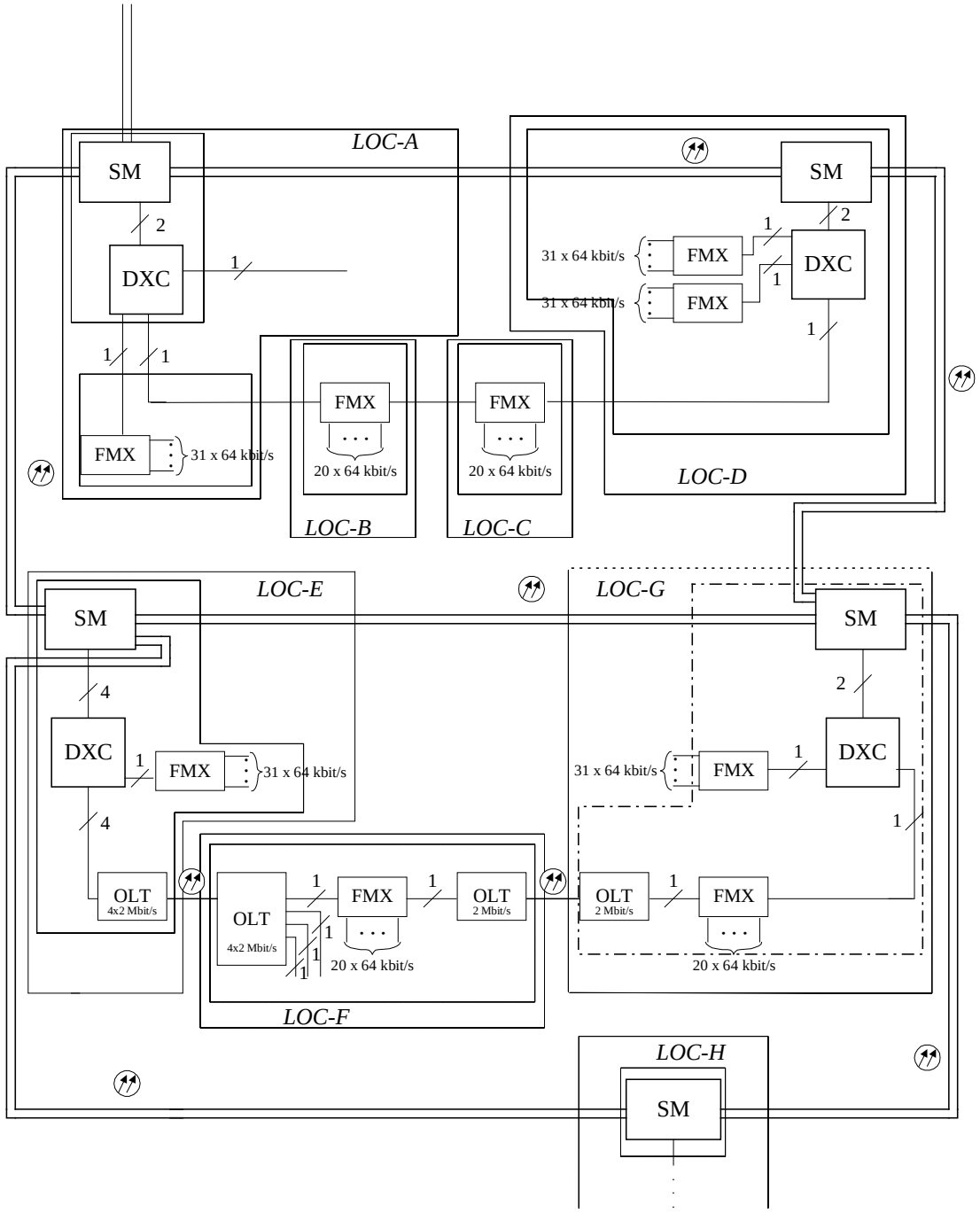
Za potrebe verifikacije definirana je mreža prikazana na slici 6.1 [7]. Ona predstavlja ograničeni model transportne mreže koji sadrži svu semantiku mreže i samo je kvantitativno slabiji od neke realne mreže. U opisu je korištena *top-down* metoda, krenuvši od strukture mreže na razini uređaja kako je prikazano na slici 6.1, zatim opisa pojedinih uređaja na razini mrežnih elemenata, i konačno opisa samih mrežnih elemenata. Na kraju su definirane konekcije između uređaja (lokalne link konekcije koje povezuju uređaje u istom čvoru i link konekcije koje povezuju uređaje različitih čvorova). Nakon toga je pokazano automatizirano kreiranje opisa takve mreže korištenjem TND sustava bez kojeg bi unos tolike količine podataka kompleksne strukture bio vrlo težak i dugotrajan uz veliku mogućnost pogreške.

6.1 Model mreže

Slika 6.1 prikazuje model mreže koji je upotrijebljen za verifikaciju TND sustava. Prikazana mreža je zamišljena kao PDH mreža sa dva SDH prstena. Na slici 6.1 prikazani su sljedeći uređaji: SM – sinkroni multipleksor, FMX – fleksibilni multipleksor, OLT – optički linijski terminal, te DXC – digitalni prospojnik. Naravno, nazivi ovih uređaja su komercijalni, tako da npr. SM nije samo multipleksor, već obavlja funkcije prospajanja, terminacije … itd, što znači da je svaki od njih sastavljen (funkcionalno) od više mrežnih elemenata. Točan prikaz svakog od ova četiri uređaja pomoću mrežnih elemenata bit će dan nakon objašnjenja slike.

Model mreže predstavlja prstenastu strukturu mreže i sastavljen je od dva prstena. Prsteni su definirani SM uređajima, od kojih 4 sudjeluju u formiranju prvog prstena (gornji prsten na slici 6.1), a tri sudjeluju u formiranju drugog prstena. Pri tome SM uređaji na lokacijama LOC-E i LOC-G povezuju oba prstena. Da bi lakše razlikovali SM uređaje, označeni su prema lokaciji u kojoj se nalaze (pa će npr. SM-E označavati SM uređaj na lokaciji LOC-E).

Osim prstenom, SM-E i SM-H uređaji te SM-A i SM-D uređaji su povezani preko "lokalne" strukture koja ide paralelno optičkom kablu koji se nalazi u prstenu. Te se strukture sastoje od FMX uređaja te OLT uređaja za prijenos preko optičkog kabela (manje brzine prijenosa nego što je u optičkom kabelu prstena), tamo gdje je on postavljen. Optičke linije označene su standardnom oznakom (krug u kojem se nalaze dvije strelice). Može se primijetiti da su FMX uređaji na različitim lokacijama različiti po broju ulaza i funkciji: FMX na lokaciji E ima 31 ulaz i služi kao krajnji multipleksor, dok FMX na lokacijama B i C služi kao Drop/Insert multipleksor i ima 20 ulaza (slobodni kapaciteti).



Slika 6.1 - Model mreže za verifikaciju TND sustava

U svakoj od lokacija nalazi se po jedan čvor, osim u LOC-A. Ona se sastoji od dva čvora: u jednom se nalaze SM i DXC uređaj, a u drugom dva FMX uređaja. Čvorovi su označeni linijom crta-točka unutar crtkane linije koja označava lokaciju.

Sam izbor uređaja i njihov položaj u mreži u smislu funkcionalnosti mreže izveden je tako da se realno oslikaju stvarni odnosi u mreži, te da sve situacije koje se u njoj javljaju mogu realno prikazati.

Za daljnji opis potrebno je definirati strukturu uređaja koji se nalaze u modelu mreže, kao i strukturu svih mrežnih elemenata. Naravno, koliko duboko ćemo ulaziti

u strukturu uređaja, odnosno koliko detaljno ćemo opisivati funkcionalnost uređaja putem mrežnih elemenata, ovisi o namjeni tog informacijskog opisa i procjeni koliko detaljan opis je potreban da bi nam dao relevantne informacije. Imajući to u vidu, funkcionalnost uređaja može se opisati sa više ili manje mrežnih elemenata, ovisno o tome koliko detaljno neke funkcije želimo razložiti.

6.2 Analiza potrebnih predložaka uređaja

Da bi precizno opisali strukturu uređaja potrebno je definirati sljedeće:

1. funkcije koje uređaj obavlja
2. mrežne elemente kojima se te funkcije opisuju
3. odnosi između pojedinih mrežnih elemenata
4. povezivanje mrežnih elemenata unutar uređaja čime se definira funkcionalnost čitavog uređaja.

U informacijskom modelu postoje bidirekcionalne veze između svih uređaja, osim između SM uređaja u prstenima mreže. Bez obzira na koji bi način taj sustav bio realiziran, interesantan je samo funkcionalni opis svakog uređaja. Tako se u uređajima kroz koje prolaze bidirekcionalne veze upotrebljavaju zasebni mrežni elementi za svaki smjer. Ako se, npr., radi o uređaju u kojem se multipleksira/demultipleksira signal, svaki smjer ima svoj mrežni element za multipleksiranje odnosno demultipleksiranje.

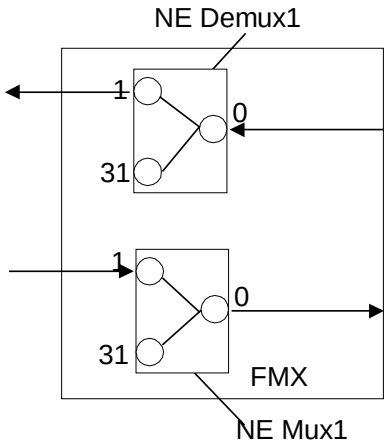
Uz opis svakog uređaja putem shematskog prikaza mrežnih elemenata i veza među njima, biti će opisani i svi potrebni predlošci mrežnih elemenata koji sudjeluju u kreiranju predloška tog uređaja, zajedno s opisom konekcija i referentnih točki.

6.2.1 FMX uređaj

FMX u modelu prikazanom na slici 6.1 radi u dva moda: kao krajnji uređaj te kao Drop/Insert multipleksor. Kako različiti modovi rada znače različitost u funkcioniranju uređaja, to povlači i različite mrežne elemente kojima se oni opisuju ili odnose između definiranih mrežnih elemenata unutar uređaja. FMX kao Drop/Insert multipleksori na lokacijama LOC-B i LOC-C moraju u sebi imati i linijske terminale kao zaključenje lokacije. Stoga FMX prikazujemo trima uređajima koji imaju sličan naziv kako bi se naznačilo da se zapravo radi o jednom proizvodu koji ima različitu funkcionalnost. To su FMX kao krajnji uređaj (FMX), FMX kao Drop/Insert multipleksor (FMX D/I), i FMX kao Drop/Insert multipleksor s linijskim terminalima (FMX D/I LT).

1. FMX kao krajnji uređaj (mux/demux) – FMX

Funkcija koju obavlja na lokacijama LOC-A, LOC-D, LOC-E, LOC-G je multipleksiranje odnosno demultipleksiranje 2Mb/s signala u 64Kb/s. Dakle, radi se o bidirekcionalnoj vezi prikazanoj različitim mrežnim elementima (slika 6.2) za svaki smjer.



Slika 6.2 – FMX kao krajnji uređaj

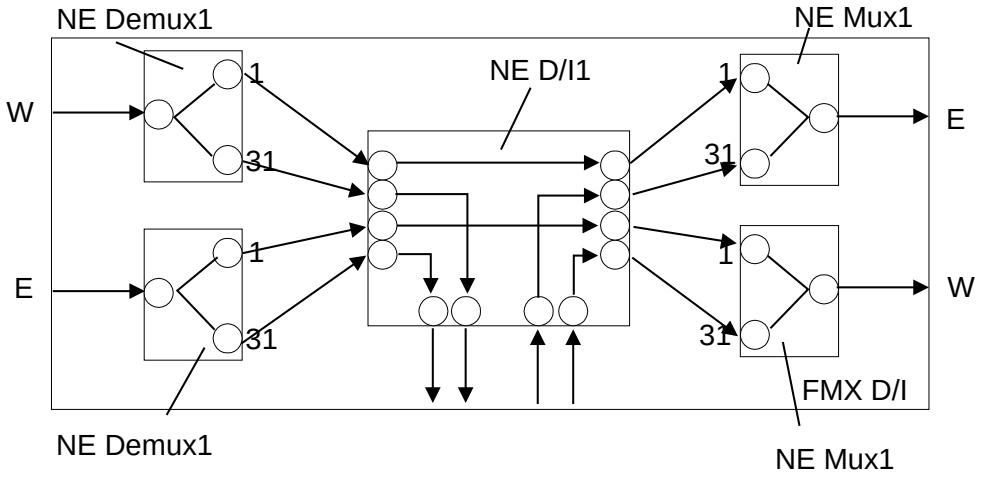
Potrebni predlošci mrežnih elemenata su: (naziv predloška - opis)

- NE Demux1 - demultipleksira 2Mbit/s u 31x64kbit/s za lokalni promet
- NE Mux1 - multipleksira lokalni promet 31x64kbit/s u 2Mbit/s

Nakon što upotrijebimo aplikaciju NED za unos predložaka mrežnih elemenata, možemo kreirati predložak uređaja aplikacijom EQD. Predlošci mrežnih elemenata koje smo kreirali biti će upotrijebljeni i u ostalim predlošcima uređaja prema potrebi.

2. FMX kao drop/insert multipleksor – FMX D/I

FMX uređaj kao drop/insert multipleksor upotrijebljen je na lokacijama LOC-F i LOC-G. Drop/Insert mrežni element D/I1 odvaja dio signala u lokalni promet i to u oba smjera. Dakle, uređaj prikazan na slici 6.3 sadrži bidirekcionalne veze, a smjerovi su označeni oznakama E (*East*) i W (*West*). Mrežni elementi Demux1 i Mux1 već su definirani pri opisu FMX predloška uređaja, pa ih nije potrebno ponovo opisivati. Također, kreiranje ovog predloška uređaja biti će vrlo jednostavno i brzo jer trebamo unijeti samo D/I1 predložak mrežnog elementa, a već unesene predloške Demux1 i Mux1 višestruko upotrijebiti u predlošku ovog uređaja.



Slika 6.3 – FMX kao drop/insert multipleksor

Predlošci upotrijebljenih mrežnih elemenata koji su ranije opisani i kreirani:

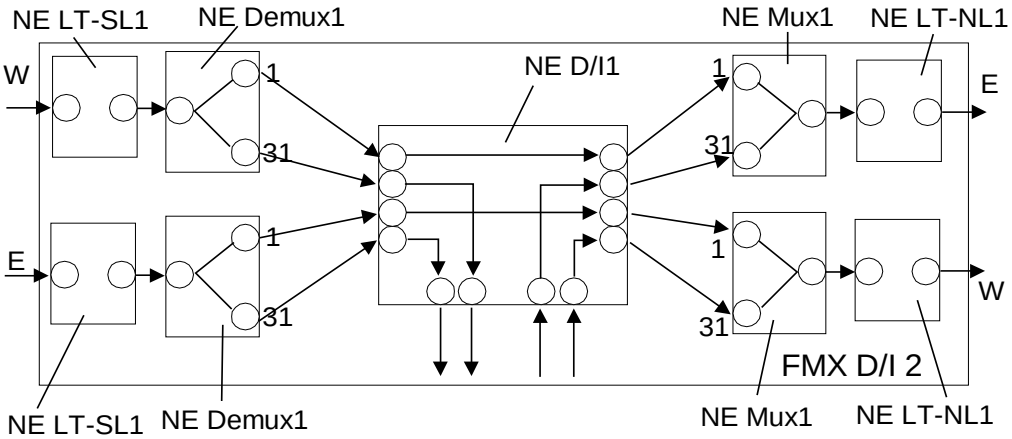
- NE Demux1
- NE Mux1

Predlošci mrežnih elemenata koje treba kreirati:

- NE D/I1 - 64kbit/s, 62 source, 62 sink, 20 drop, 20 insert točaka

3. FMX kao drop/insert multipleksor s linijskim terminalima – FMX D/I LT

Kao što je već napomenuto ovakav uređaj nalazimo na lokacijama LOC-B i LOC-C i razlikuju se od FMX D/I po tome što imaju dodane linijske terminale kao zaključenje lokacije (slika 6.4).



Slika 6.4 – FMX kao drop/insert multipleksor s linijskim terminalima

Predlošci upotrijebljenih mrežnih elemenata koji su ranije opisani i kreirani:

- NE Demux1
- NE Mux1

- NE D/I1

Predlošci mrežnih elemenata koje treba kreirati:

- NE LT-SL1 - prilagodba 2Mbit/s sa linije
- NE LT-NL1 - prilagodba 2Mbit/s na liniju

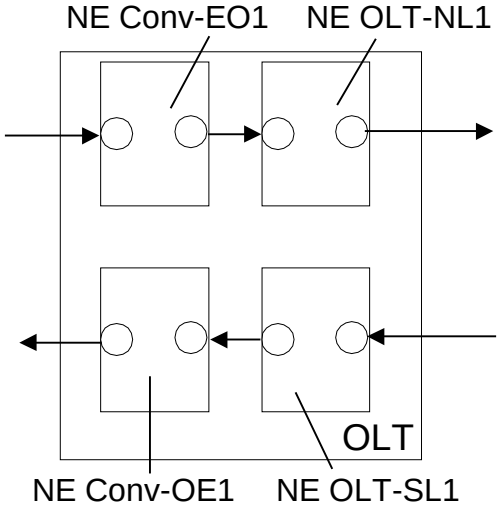
Unos ovog predloška uređaja je vrlo jednostavan i brz jer nakon što aplikacijom NED unesemo predloške mrežnih elemenata LT-SL1 i LT-NL1, u aplikaciji EQD možemo upotrijebiti predložak FMX D/I uređaja kojem dodamo linijske terminale i iskoristimo komandu SAVE_AS kako bi ga pohranili u repozitorij kao predložak novog, različitog uređaja.

6.2.2 OLT uređaj (optički linijski terminal)

Prema slici 6.1 postoje dva tipa OLT uređaja: uređaj koji ima jedan ulaz odnosno izlaz za svaki smjer na nivou 2 Mbit/s (lokacije LOC-F i LOC-G), te OLT koji ima 4 ulaza odnosno izlaza istog nivoa (lokacije LOC-E i LOC-F). Oba uređaja vrše konverziju električnog signala 2Mb/s u optički (i obratno) i prilagodbu optičkog signala na liniju, odnosno s linije.

4. OLT sa jednim ulazom i jednim izlazom - OLT

Slika 6.5 prikazuje uređaj OLT koji se sastoji od dva mrežna elementa tipa optički linijski terminal, označena sa NE OLT-NL1 i NE-OLT-SL1, te dva mrežna elementa tipa Converter (pretvornik) koje označavamo oznakama NE Conv-OE1 i NE Conv-EO1.



Slika 6.5 – OLT uređaj sa jednim ulazom i jednim izlazom

Predlošci mrežnih elemenata koje treba kreirati:

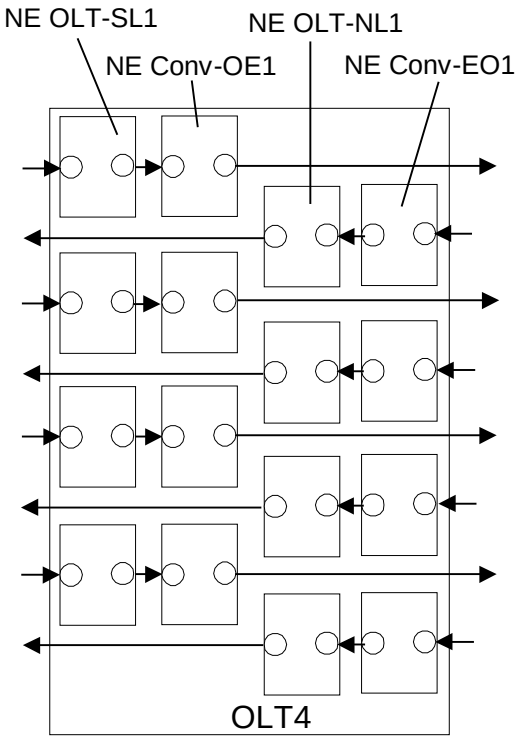
- NE OLT-NL1 - prilagodba optičkog signala na liniju
- NE OLT-SL1 - prilagodba optičkog signala sa linije

- NE Conv-EO1 - konvertira električki signal 2 Mbit/s u optički signal
- NE Conv-OE1 - konvertira optički u električki signal 2 Mbit/s

Treba naglasiti razliku između mrežnih elemenata NE OLT-NL1 odnosno NE OLT-SL1 upotrijebljenih u ovom predlošku uređaja i NE LT-NL1 odnosno NE LT-SL1 upotrijebljenih u FMX D/I LT predlošku uređaja: mrežni elementi NE LT-NL1 i NE LT-SL1 služe za prilagodbu električkog signala, a mrežni elementi tipa NE OLT-NL1 i NE OLT-SL1 služe za prilagodbu optičkog signala na liniju odnosno sa linije (NL – prilagodba na liniju, SL – prilagodba sa linije).

5. OLT s po četiri ulaza odnosno izlaza za svaki smjer – OLT4

Slika 6.6 prikazuje OLT uređaj sa četiri ulaza i izlaza za jedan smjer i 4 ulaza/izlaza za drugi. Uređaj se zapravo sastoji od po 4 linijska terminala tipa NE OLT-SL1 i NE OLT-NL1 i po četiri konvertera tipa NE Conv-OE1 i NE Conv-EO1, koji su već definirani pri opisu predloška uređaja OLT. Na slici 6.6 je zbog preglednosti označen samo po jedan od svakog tipa mrežnog elementa.



Slika 6.6 – OLT uređaj sa četiri ulaza i izlaza

Predlošci potrebnih mrežnih elemenata koji su svi ranije opisani i kreirani:

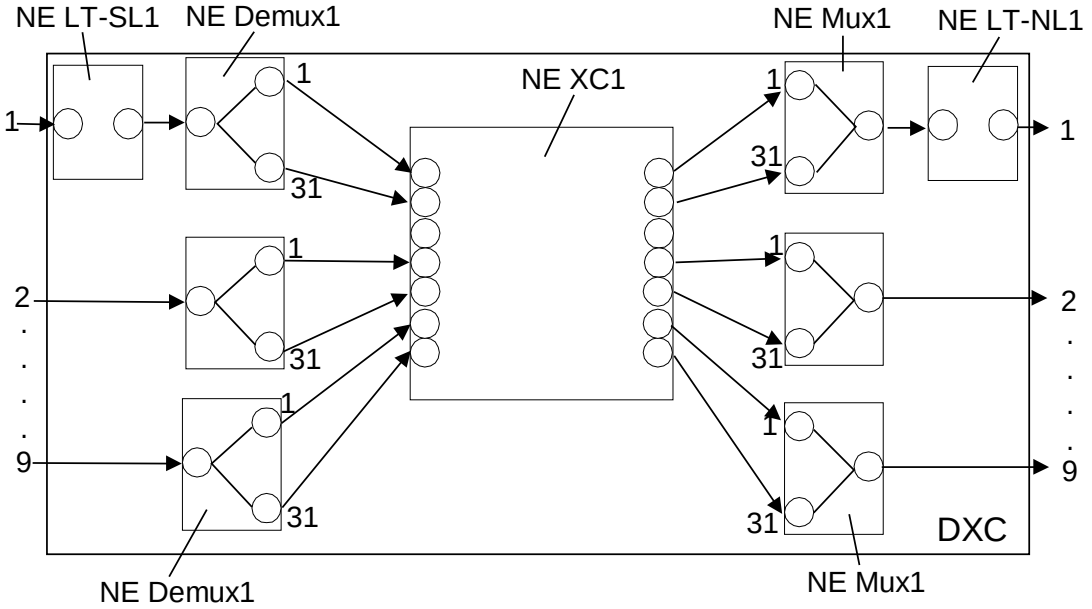
- NE OLT-NL1
- NE OLT-SL1
- NE Conv-EO1
- NE Conv-OE1

Svi potrebni predlošci mrežnih elemenata su već opisani i kreirani prilikom kreiranja predloška uređaja OLT, pa je ovaj predložak uređaja jednostavno izgraditi višestrukom upotrebom već unesenih predložaka mrežnih elemenata.

6.2.3 DXC

DXC uređaj možemo naći na slici 6.1 na lokacijama LOC-A, LOC-D, LOC-E i LOC-G. Uređaj je zamišljen kao prospojnik koji usmjerava sav lokalni promet prije preusmjeravanja u SM uređaje (tj. na prstene). To znači da se npr. promet u lokaciji E usmjerava ili na lokalni dio lokacije E, ili u lokaciju F lokalno, bez prijenosa prstenom, jer bi prstenom taj promet prolazio dužim putem i zauzimao nepotrebno njegove kapacitete. Lokalnim prometom smatramo sav promet koji dolazi na određenu lokaciju preko uređaja koji nisu dio prstena.

Za model mreže podrazumijeva se da su svi DXC uređaji na slici 6.1 istog tipa i unutrašnje strukture. Ta struktura (mrežni elementi i povezanost mrežnih elemenata unutar uređaja) je prikazana na slici 6.7.



Slika 6.7 – DXC uređaj

DXC je zamišljen kao uređaj koji prospaja dolazni promet na nivou 64 kbit/s, što znači da mora sadržavati demultipleksore sa 2Mbit/s na 64 kbit/s brzinu prijenosa signala, mrežni element - prospojnik dovoljnog kapaciteta s obzirom na broj ulaznih 2Mbit/s signala, te multipleksor za izlazno multipleksiranje signala. Obzirom da smo već definirali NE Mux1 i NE Demux1, potrebno je samo definirati karakteristike mrežnog elementa – prospojnika. On je označen sa NE XC i može prospajati 279 signala brzine prijenosa 64 kbit/s. To znači da ima 558 referentnih točaka! Ovaj je mrežni element izabran kako bi se pokazala složenost sustava u kojem se u svakom trenutku može dobiti položaj signala unutar signala veće brzine prijenosa.

Broj ulaznih 2Mbit/s signala je 9, što je proizvoljno odabran broj, prema čemu je proizvoljno i određen kapacitet NE XC mrežnog elementa. Primijetimo da je kapacitet ovako definiranog prospojnika prema slici 6.1 u potpunosti iskorišten tek u lokaciji E,

dok se u lokacijama A, D i G iskorištava samo dio kapaciteta, pa postoji mogućnost priključivanja dodatnog lokalnog prometa preko istog uređaja (postoje neiskorišteni instalirani kapaciteti). Također primijetimo da je u opisu ovog uređaja dodan jedan linijski terminal, jer je u lokacijama LOC-A i LOC-D uređaj DXC vezan na izlaznu link konekciju (prema LOC-B i LOC-C).

Predlošci potrebnih mrežnih elemenata koji su ranije opisani i kreirani:

- NE Demux1
- NE Mux1
- NE LT-SL1
- NE LT-NL1

Predlošci mrežnih elemenata koje treba kreirati:

- NE XC1 - prospojnik 64 kbit/s signala , 279 source i 279 sink referentnih točaka

6.2.4 SM uređaj

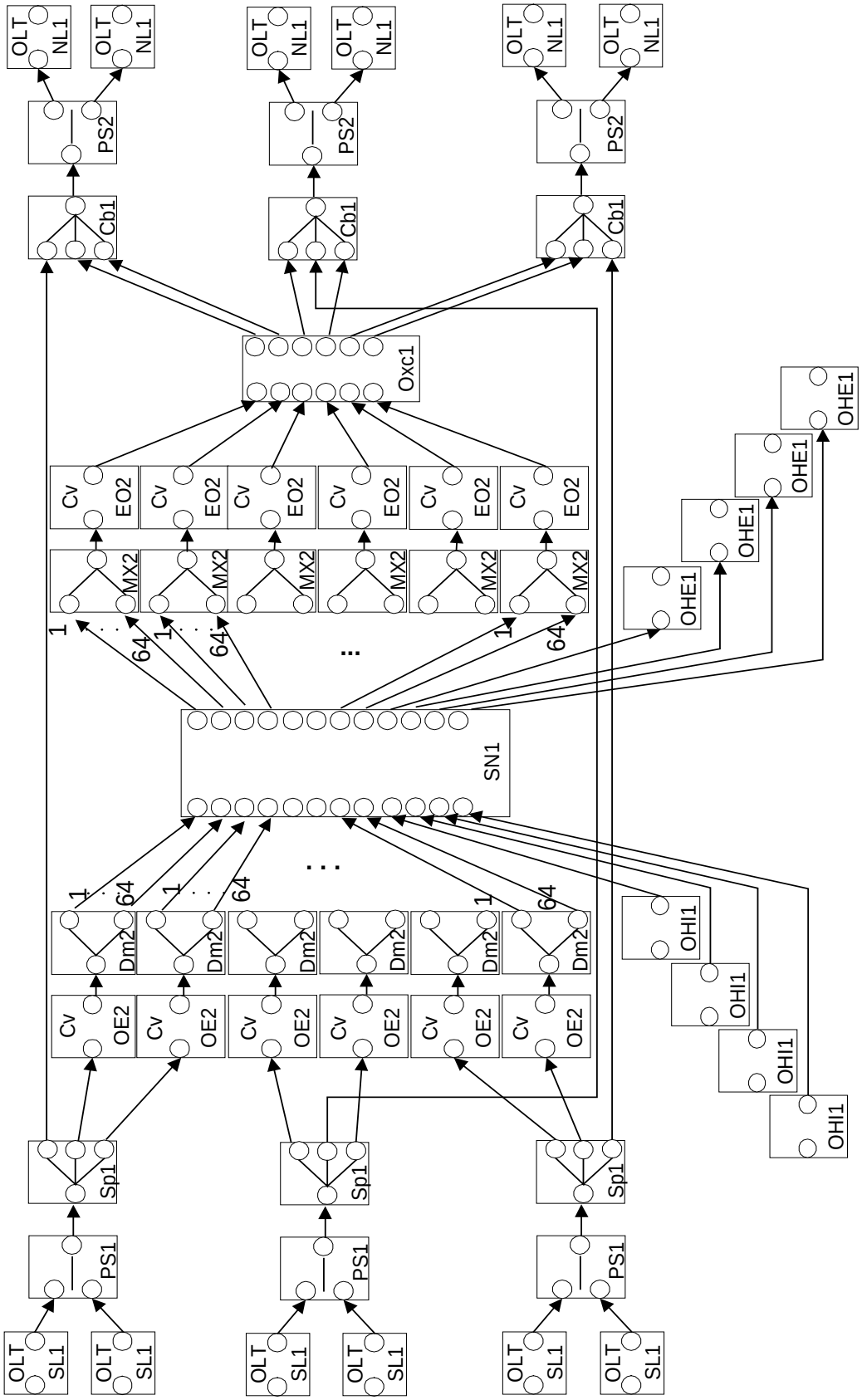
SM uređaj je najsloženiji uređaj u modelu mreže. Na slici 6.1 ga možemo naći na lokacijama A, D, E, G i H. Vidimo da se SM uređajem ostvaruje funkcionalnost prstena. Ovi uređaji služe kao centri lokalnog prometa koji se preko njih usmjerava na optičke prstene. Istodobno, promet koji preko prstena dolazi do SM uređaja se u tom uređaju može prospojiti za daljnji prijenos preko prstena, ili se može odvojiti kao lokalni promet.

Prema ovom se kratkom opisu osnovnih transportnih funkcija koje se obavljaju u SM uređaju može zaključiti da se on sastoji od sljedećih mrežnih elemenata:

- prospojnika,
- optičkih linijskih terminala,
- pretvornika signala,
- sprežnika,
- rasprežnika, te
- OHI i OHE tipova mrežnih elemenata.

Opis strukture SM uređaja sa mrežnim elementima prikazan je na slici 6.8. Pod-tipovi mrežnih elemenata su na slici označeni bez oznake “NE” (npr. Cv2 umjesto NE Cv2) iz razloga preglednosti prikaza.

Svaki mrežni element na slici 6.8 ima oznaku koja je proizvoljno određena, a koja će je bitna za opis karakteristika tog pod-tipa mrežnog elementa. Vidljivo je da se svi mrežni elementi osim dva (NE SN1 i NE Oxc1) pojavljuju više puta u ovom uređaju. Stoga pri kreiranju predloška i ovog uređaja dolazi do izražaja učinkovitost koncepta predložaka kojim se isti predložak mrežnog elementa jednostavno može upotrijebiti više puta u nekom predlošku uređaja.



Slika 6.8 – SM uređaj

Predlošci potrebnih mrežnih elemenata koji su ranije opisani i kreirani:

- NE OLT-SL1
- NE OLT-NL1

Predlošci mrežnih elemenata koje treba kreirati:

- NE PS1 – optički prospojnik s funkcijom zaštitnog komutiranja (struktura prstena je 1+1); dvije ulazne i jedna izlazna referentna točka;
- NE PS2 – optički prospojnik s funkcijom zaštitnog komutiranja (struktura prstena je 1+1); jedna ulazna i dvije izlazne referentne točke;
- NE Sp1 – rasprežnik; jedna ulazna i tri izlazne referentne točke;
- NE Cb1 – sprežnik; tri ulazne i jedna izlazna referentna točka;
- NE Cv-OE2– pretvornik optičkog u električki signal STM-1 sloja; jedna ulazna i jedna izlazna referentna točka;
- NE Cv-EO2– pretvornik električkog signala STM-1 sloja u optički; jedna ulazna i jedna izlazna referentna točka;
- NE Dm2 – demultipleksor sa STM-1 na VC12 signal; jedna ulazna i 64 izlazne referentne točke;
- NE MX2 – multipleksor sa VC12 na STM1 sloj; 64 ulazne i jedna izlazna referentna točka;
- NE SN1 – glavni prospojnik VC12 signala; sa po 64*6+4 = 388 ulaznih i izlaznih referentnih točki = ukupno 776 točaka;
- NE OXC1 – prospojnik optičkog signala; po šest ulaznih i izlaznih referentnih točaka;
- NE OHI1 – dodavanje zaglavlja 2 Mbit/s signalu, čime postaje VC12 signal; jedna ulazna i jedna izlazna referentna točka;
- NE OHE1 – skidanje zaglavlja VC12 signala čime postaje 2 Mbit/s signal; jedna ulazna i jedna izlazna referentna točka.

6.3 Unos podataka u TRNET bazu

Nakon što su aplikacijama NED i EQD kreirani svi predlošci uređaja, možemo pomoću aplikacije NTD i kreiranih predložaka iz repozitorija kreirati instancije uređaja u TRNET bazi podataka.

Dakle, možemo zaključiti da nam je potrebno 7 predložaka uređaja:

1. FMX – multipleksor kao krajnji uređaj (lokacije LOC-A, LOC-D, LOC-E, LOC-G)
2. FMX-D/I – drop/insert multipleksor (lokacije LOC-F i LOC-G)
3. FMX-D/I-LT – drop/insert multipleksor s linijskim terminalima (lokacije LOC-B i LOC-C)
4. OLT – optički linijski terminal, konverzija optičkog u el. signal i obratno (LOC-E, LOC-F, LOC-G)
5. OLT4 – optički linijski terminal sa po 4 ulaza i izlaza, konverzija optičkog u el. signal i obratno (LOC-E, LOC-F)
6. DXC – prospojnik (LOC-A, LOC-D, LOC-E, LOC-G)

7. SM – uređaj koji je glavna karika u optičkim prstenima, a služe i za dovođenje i odvođenje lokalnog prometa u prsten (LOC-A, LOC-D, LOC-E, LOC-G, LOC-H)

Aplikacijom NTD jednostavno odabiremo navedene predloške i čvorove u kojima bi trebali kreirati njihove instancije. Eventualno se mogu promijeniti još neki parametri i pritiskom na CREATE_EQUIPMENT će biti kreirana instancija uređaja.

Nakon što smo unijeli sve instancije možemo aplikacijom NTD kreirati i konekcije između unesenih instancija uređaja prema slici 6.1. Postupak kreiranja konekcija između instancija uređaja u TRNET bazi podataka opisan je u paragrafu 5.6.5 (str. 79). Pri povezivanju uređaja posebnu pažnju treba obratiti na to da su sve konekcije dvosmjerne, odnosno potrebno je kreirati jednak broj konekcija za svaki smjer.

Da bi provjerili unesene podatke možemo napraviti niz upita nad bazom podataka transportne mreže TRNET. Jedan od takvih upita je prikazan u tablici 6.1 koja prikazuje kreirane instancije uređaja iz tablice EQUIPMENT.

Tablica 6.1 – Rezultat upita nad tablicom EQUIPMENT u TRNET bazi podataka koji prikazuje instancije uređaja unesenih aplikacijom NTD

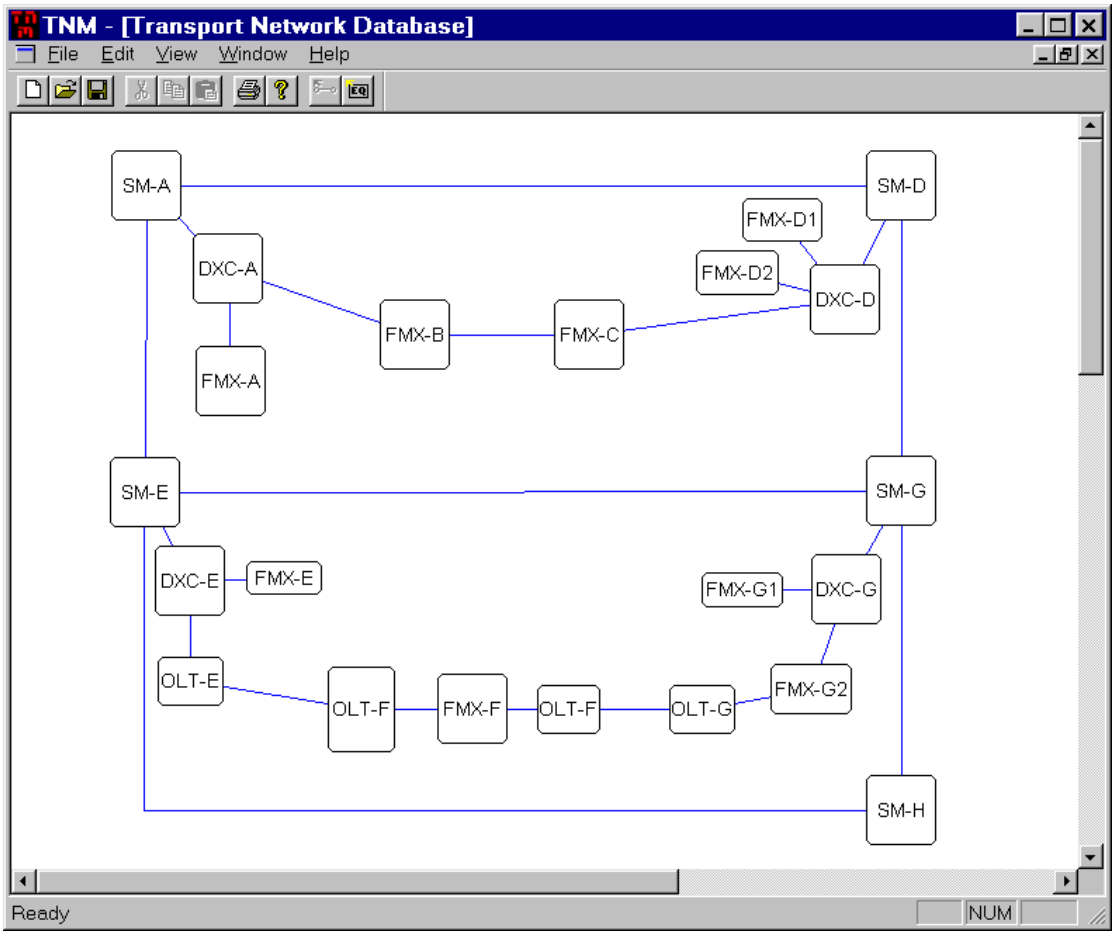
EQ_ID	NODE_ID	LABEL	ADMINISTRATIVE _STATUS	OPERATIONAL _STATE	EQ_TYPE
46	1	SM-A	IN USE	ENABLED	SM
47	1	DXC-A	IN USE	ENABLED	DXC
48	2	FMX-A	IN USE	ENABLED	FMX
49	3	FMX-B	IN USE	ENABLED	FMX
50	4	FMX-C	IN USE	ENABLED	FMX
61	5	SM-D	IN USE	ENABLED	SM
81	5	DXC-D	IN USE	ENABLED	DXC
82	5	FMX-D1	IN USE	ENABLED	FMX
83	5	FMX-D2	IN USE	ENABLED	FMX
84	6	SM-E	IN USE	ENABLED	SM
85	8	SM-G	IN USE	ENABLED	SM
101	6	DXC-E	IN USE	ENABLED	DXC
102	6	FMX-E	IN USE	ENABLED	FMX
103	6	OLT-E	IN USE	ENABLED	OLT
104	7	OLT-F1	IN USE	ENABLED	OLT4
105	7	FMX-F	IN USE	ENABLED	FMX
106	7	OLT-F2	IN USE	ENABLED	OLT
107	8	OLT-G	IN USE	ENABLED	OLT
108	8	DXC-G	IN USE	ENABLED	DXC
109	8	FMX-G1	IN USE	ENABLED	FMX
110	8	FMX-G2	IN USE	ENABLED	FMX
111	9	SM-H	IN USE	ENABLED	SM

Kao rezultat upita nad unesenim modelom mreže u TRNET bazi podataka dobiveni su i sljedeći podaci: Jedna instancija SM uređaja ima 58 mrežnih elemenata, 827 temeljnih link konekcija (BLC) i 1674 referentnih točaka, a takvih SM uređaja u ovom modelu ima pet i sve instancije su unesene na temelju istog predloška. Time se želi pokazati efikasnost koncepta predložaka. Model se sastoji od 9 čvorova, u kojima

su ukupno 22 uređaja (tablica 6.1), 504 mrežna elementa, 6973 temeljnih link konekcija i 14610 referentnih točaka. Koje entitete i atribute u TRNET bazi kreira TND sustav, naznačeno je u dodatku.

Kreiranje sedam predložaka uređaja i 22 predloška mrežnih elemenata aplikacijama EQD i NED trajalo je otprilike 3 sata, od čega je znatan dio bio grafičko oblikovanje. Sa gotovim predlošcima uređaja bilo je jednostavno i zaista brzo generirati 22 instancije uređaja u TRNET bazi podataka pomoću aplikacije NTD. Taj proces trajao je otprilike pola sata, što je zaista izuzetno malo s obzirom na količinu unesenih podataka.

Na slici 6.9 je prikazan model mreže unesen u TRNET bazu podataka kako ga prikazuje NTD aplikacija. Treba naglasiti da se radi o prikazu mreže na razini uređaja na temelju podataka iz TRNET bazu. Dakle, i taj prikaz je određena provjera unesenih podataka. Ako usporedimo ovaj prikaz podataka TRNET baze dobiven NTD aplikacijom sa modelom mreže na slici 6.1, možemo vidjeti njihovu sličnost, odnosno poklapanje.



Slika 6.9 – Model mreže u TRNET bazi podataka kako ga prikazuje NTD aplikacija

7. Zaključak

Na temelju generičkog informacijskog modela transportne mreže sa PDH i SDH elementima razvijena je i implementirana baza podataka transportne mreže nazvana TRNET [3] [7] kao prvi korak prema informacijskom sustavu za upravljanje transportnom mrežom.

U ovom magistarskom radu razvijen je, implementiran i testiran inteligentni sustav za punjenje baze podataka transportne mreže TRNET nazvan TND (***T****ransport* ***N****etwork* ***D****esigner*). Osim efikasnog i brzog punjenja TRNET baze podataka kroz grafičko sučelje pri čemu je zbog ugrađenog znanja i provjera vjerojatnost unosa pogrešnih podataka svedena na minimum, sustav pruža i grafički prikaz podataka unesenih u TRNET bazu. Nadalje, sustav pruža i određeni nivo upravljanja konfiguracijom jer je moguće mijenjati topologiju transportne mreže kreiranjem konekcija između uređaja u TRNET bazi. Sustav je modularan, proširiv s novim tipovima mrežnih elemenata, novim funkcijama, modulima ili aplikacijama.

Verifikacija je obavljena na odgovarajućem modelu transportne mreže čiji je informacijski opis uspješno unesen u TRNET bazu podataka korištenjem TND sustava.

Za izradu aplikacija sustava korišten je *MS Visual C++*, a repozitorij je implementiran u ORACLE RDBMS-u.

Literatura

[1] Blaszczak, M., “Professional MFC with Visual C++ 5”, Wrox Press, 1997.

[2] Microsoft Developer Network (MSDN) Library v6.0

[3] Kesegić, V., I. Matasić, B. Mikac, Z. Skočir, E. Šehović, B. Vrdoljak, “Baza podataka o telekomunikacijskoj mreži HEP-TMN s pratećim aplikacijama, Verzija 2.1”, Zagreb, travanj 1999.

[4] Petzold, Ch., “Programming Windows 95”, Microsoft Press, 1996.

[5] Skočir, Z., E. Šehović, I. Matasić, V. Kesegić, “Application for Information Design of Network Elements”, *Proceedings of International Telecommunications Symposium VITEL 98*, pp. E.33-E.37, Ljubljana – Slovenia, October 5-6, 1998.

[6] Matasić, I., Z. Skočir, V. Kesegić, “Framework for Information Modeling of Optical Network”, *Proceedings of Workshop on All-optical Networks WAON 98*, pp. 51-60, Zagreb, 6-8 svibnja 1998.

[7] Matasić, I. «Informacijski sustav transportne mreže s elementima sinkrone digitalne hijerarhije», magistarski rad, FER, Zagreb, 1999.

[8] W.Widle, K. Woldegiorgis: “In Search of Managed Objects”, Ericsson Review, 69:1-2, 1992. pp. 34-56.

[9] Z. Skočir, “Postupak informacijske analize i formalnog opisa telekomunikacijske mreže”, ITA, 9, 69-114, 1990.

[10] Z. Skočir, B. Mikac, B. Vrdoljak: “An Approach to Information Description of Telecommunication Network”. Proceedings of IASTED International Conference on Networks and Communication Systems, 26-30, Pittsburgh, USA, May 13-16, 1998.

[11] E. Šehović, Z. Skočir, I. Lovrek, M. Kunštić: "An Approach to Power Utility Telecommunication Network Management", Proc. of CIGRE SYMPOSIUM Integrated Control and Communication Systems, 300-01, Helsinki, August 1995.

[12] M. Varga, “Baze podataka – Konceptualno, logičko i fizičko modeliranje podataka”, DRIP, Zagreb, 1994.

DODATAK

U ovom dodatku je popis relacija TRNET baze podataka kakvu je koristi TND sustav. Podebljanim slovima su naznačeni atributi koje popunjava sustav TND. Zvjezdicom (*) su označeni oni atributi čije će se popunjavanje omogućiti TND sustavom nakon definiranja valnih duljina i grupa valnih duljina koje treba obaviti na konceptualnom nivou (optički dio sustava). Nakon naziva relacije, u zagradama slijede atributi sa svojim tipom i oznakom da li je NULL vrijednost dozvoljena.

```
BASIC_LINK_CONNECTION (
  PREVIOUS_NE_ID      NUMBER        NOT NULL,
  PREVIOUS_RP_NUMBER  NUMBER        NOT NULL,
  NEXT_NE_ID          NUMBER        NOT NULL,
  NEXT_RP_NUMBER      NUMBER        NOT NULL,
  TYPE                VARCHAR2 (4)
)
BASIC_SUBNETWORK_CONNECTION (
  PREVIOUS_RP_NUMBER  NUMBER        NOT NULL,
  NEXT_RP_NUMBER      NUMBER        NOT NULL,
  NE_ID               NUMBER        NOT NULL,
  ACTIVITY_STATUS     VARCHAR2 (10),
  TYPE                VARCHAR2 (2)
)
EQUIPMENT (
  EQ_ID                  NUMBER        NOT NULL,
  NODE_ID                NUMBER,
  LABEL                  VARCHAR2 (20),
  ADMINISTRATIVE_STATUS  VARCHAR2 (10),
  OPERATIONAL_STATE      VARCHAR2 (8),
  EQ_TYPE                VARCHAR2 (15)
)
HIERARCHY_LEVEL (
  NETWORK_HIERARCHY_LEVEL  VARCHAR2 (20)  NOT NULL
)
LAYER (
  LAYER_ID     NUMBER (3)  NOT NULL,
  LAYER_LABEL  VARCHAR2 (6)
)
LINK_CONNECTION (
  LC_ID                NUMBER        NOT NULL,
  LC_HIERARCHY_LEVEL   VARCHAR2 (20),
  PREVIOUS_NE_ID       NUMBER        NOT NULL,
  PREVIOUS_RP_NUMBER   NUMBER        NOT NULL,
  NEXT_NE_ID           NUMBER        NOT NULL,
  NEXT_RP_NUMBER       NUMBER        NOT NULL,
  PREVIOUS_NETWORK_ID  NUMBER,
  NEXT_NETWORK_ID      NUMBER
)
LOCATION (
```

```
  LOCATION      VARCHAR2 (15)  NOT NULL,
  X_COORDINATE  NUMBER,
  Y_COORDINATE  NUMBER,
  Z_COORDINATE  NUMBER
)
GTP (
  GTP_ID           NUMBER        NOT NULL,
  NE_ID            NUMBER        NOT NULL,
  ACTIVITY_STATUS  VARCHAR2 (10),
  NUMBER_OF_RP         NUMBER (10)
)
MPXC (
  MPXC_ID           NUMBER        NOT NULL,
  NE_ID             NUMBER        NOT NULL,
  SOURCE_RP_NUMBER  NUMBER,
  NUMBER_OF_RP          NUMBER (10),
  ACTIVITY_STATUS   VARCHAR2 (10)
)
NETWORK (
  NETWORK_ID               NUMBER        NOT NULL,
  NETWORK_HIERARCHY_LEVEL  VARCHAR2 (20),
  SUPERIOR_NETWORK_ID      NUMBER,
  ADMINISTRATION           VARCHAR2 (20),
  NETWORK_LABEL            VARCHAR2 (20)
)
NETWORK_ELEMENT (
  NE_ID                   NUMBER        NOT NULL,
  ADMINISTRATIVE_STATUS  VARCHAR2 (10),
  SIGNAL_TYPE            VARCHAR2 (18),
  TYPE                  VARCHAR2 (5),
  EQ_ID                    NUMBER,
  AVAILABILITY_STATUS    VARCHAR2 (13),
  OPERATIONAL_STATE      VARCHAR2 (8)
)
NE_COMBINER (
  NE_ID         NUMBER        NOT NULL,
  NUMBER_OF_RP  NUMBER ( 10 )
)
NE_CONVERTER (
  NE_ID                NUMBER        NOT NULL,
  ELECTRICAL_LAYER_ID  NUMBER (3),
  WAVELENGTH_GROUP_ID*  NUMBER,
  TYPE                 VARCHAR2 (18)
)
NE_DEMULTIPLEXER (
  NE_ID            NUMBER        NOT NULL,
  LOWER_LAYER_ID   NUMBER (3),
  NUMBER_OF_RP     NUMBER ( 10 ),
  DEMUX_TYPE       VARCHAR2 (3),
  HIGHER_LAYER_ID  NUMBER (3)
)
```

```
NE_ELECTRICAL_DROP_INSERT (
  NE_ID          NUMBER        NOT NULL,
  EDI_LAYER_ID   NUMBER (3),
  NUMBER_OF_RP   NUMBER ( 38 ),
  NUMBER_OF_DROP_RP  NUMBER ( 38 )
)
NE_ELECTRICAL_LINE_TERMINAL (
  NE_ID                   NUMBER        NOT NULL,
  INPUT_SIGNAL_LAYER_ID   NUMBER (3),
  LT_CHARACTERISTIC      VARCHAR2 (9),
  LT_FUNCTION            VARCHAR2 (9),
  OUTPUT_SIGNAL_LAYER_ID  NUMBER (3)
)
NE_MULTIPLEXER (
  NE_ID            NUMBER        NOT NULL,
  LOWER_LAYER_ID   NUMBER (3),
  HIGHER_LAYER_ID  NUMBER (3),
  MUX_TYPE         VARCHAR2 (3),
  NUMBER_OF_RP     NUMBER ( 38 )
)
NE_OPTICAL_DROP_INSERT (
  NE_ID          NUMBER        NOT NULL,
  NUMBER_OF_RP   NUMBER ( 38 ),
  NUMBER_OF_DROP_RP  NUMBER ( 38 )
)
NE_OPTICAL_FIBRE_AMPLIFIER (
  NE_ID               NUMBER        NOT NULL,
  AMPLIFICATION       VARCHAR2 (15),
  AMP_CHARACTERISTIC  VARCHAR2 (9)
)
NE_OVERHEAD_EXTRACTION (
  NE_ID                   NUMBER        NOT NULL,
  INPUT_SIGNAL_LAYER_ID   NUMBER (3),
  OUTPUT_SIGNAL_LAYER_ID  NUMBER (3)
)
NE_OVERHEAD_INSERTION (
  NE_ID                   NUMBER        NOT NULL,
  INPUT_SIGNAL_LAYER_ID   NUMBER (3),
  OUTPUT_SIGNAL_LAYER_ID  NUMBER (3)
)
NE_OPTICAL_LINE_TERMINAL (
  NE_ID               NUMBER        NOT NULL,
  LT_FUNCTION         VARCHAR2 (9),
  LT_CHARACTERISTIC  VARCHAR2 (9)
)
NE_OPTICAL_CROSSCONNECT (
  NE_ID                 NUMBER        NOT NULL,
  PROTECTION_SWITCHING  VARCHAR2 (3),
  POINT_TO_POINT        VARCHAR2 (3),
  GTP                   VARCHAR2 (3),
  BROADCAST             VARCHAR2 (3),
```

```
  NUMBER_OF_RP          NUMBER ( 38 ),
  NUMBER_OF_INPUT_RP    NUMBER ( 38 )
)
NE_REGENERATOR (
  NE_ID                   NUMBER        NOT NULL,
  AMPLIFICATION           VARCHAR2 (15),
  AMP_CHARACTERISTIC      VARCHAR2 (9),
  INPUT_SIGNAL_LAYER_ID   NUMBER (3),
  OUTPUT_SIGNAL_LAYER_ID  NUMBER (3)
)
NE_SPLITTER (
  NE_ID         NUMBER        NOT NULL,
  NUMBER_OF_RP  NUMBER ( 38 )
)
NE_CROSSCONNECT (
  NE_ID                 NUMBER        NOT NULL,
  XC_LAYER_ID           NUMBER (3),
  PROTECTION_SWITCHING  VARCHAR2 (3),
  POINT_TO_POINT        VARCHAR2 (3),
  GTP                      VARCHAR2 (3),
  BROADCAST             VARCHAR2 (3),
  NUMBER_OF_RP          NUMBER ( 38 ),
  NUMBER_OF_INPUT_RP    NUMBER ( 38 )
)
NODE (
  NODE_ID     NUMBER        NOT NULL,
  NETWORK_ID  NUMBER,
  LOCATION    VARCHAR2 (15),
  NODE_LABEL  VARCHAR2 (20)
)
PATHWAY (
  PATHWAY_ID           NUMBER        NOT NULL,
  AP_SOURCE_NE_ID      NUMBER,
  AP_SOURCE_RP_NUMBER  NUMBER,
  AP_SINK_NE_ID        NUMBER,
  AP_SINK_RP_NUMBER    NUMBER,
  TRAIL_ID             NUMBER        NOT NULL,
  QOS                  VARCHAR2 (15)
)
PATHWAY_ELEMENT (
  TRAIL_ID    NUMBER        NOT NULL,
  PATHWAY_ID  NUMBER        NOT NULL,
  SORT        NUMBER        NOT NULL,
  NE_ID       NUMBER,
  RP_NUMBER   NUMBER,
  TYPE_OF_RP  VARCHAR2 (3)
)
REFERENCE_POINT (
  NE_ID           NUMBER        NOT NULL,
  RP_NUMBER       NUMBER        NOT NULL,
  RP_FUNCTION     VARCHAR2 (6),
```

```
  RP_TYPE          VARCHAR2 (3),
  RP_STATUS        VARCHAR2 (10),
  RP_SIGNAL_TYPE   VARCHAR2 (10),
  RP_LAYER_ID      NUMBER (3),
  WAVELENGTH_GROUP_ID*  NUMBER,
  GTP_ID           NUMBER,
  MPXC_ID          NUMBER,
  RP_ROLE          VARCHAR2 (19),
  TUNABLE          VARCHAR2 (3) DEFAULT 'NO' NOT NULL
)
WAVELENGTH (
  WAVELENGTH_GROUP_ID*  NUMBER    NOT NULL,
  WAVELENGTH*     NUMBER (9,3)          NOT NULL
)
TRAIL (
  TRAIL_ID            NUMBER        NOT NULL,
  NUMBER_OF_PATHWAYS  NUMBER ( 38 ),
  CAPACITY            NUMBER,
  ACTIVITY_STATUS     VARCHAR2 (21),
  ALTERNATIVE_TRAIL   NUMBER,
  TYPE                VARCHAR2 (20)
)
```

Sažetak

Zbog kompleksnosti informacijskog opisa transportne mreže sa SDH i PDH elementima i velike količine podataka potrebnih za opis realne mreže, javlja se potreba za ekspertnim sustavom. U ovom radu opisan je inteligentni informacijski sustav za punjenje baze podataka transportne mreže (TRNET), nazvan *Transport Network Designer* (TND). TND se sastoji od baze podataka nazvane repozitorij i triju aplikacija: *Network Element Designer* (NED), *Network Equipment Designer* (EQD) i *Network Topology Designer* (NTD). Ovim aplikacijama preko intuitivnog grafičkog sučelja korisnik kreira i povezuje komponente mreže. Verifikacija je uspješno provedena korištenjem TND ekspertnog sustava za punjenje baze podataka transportne mreže (TRNET) podacima realnog modela mreže.

NED, EQD i NTD su aplikacije programirane u *Visual C++* jeziku za Windows platforme. Repozitorij je implementiran u ORACLE RDBMS.

INTELLIGENT SYSTEM FOR LOADING A
TRANSPORT NETWORK DATABASE

Summary

The extreme complexity of informational description of transport networks with SDH and PDH elements and vast amount of data in such informational description requires an expert system. This thesis presents the intelligent information system for loading a transport network database (TRNET), named Transport Network Designer (TND). TND consists of a repository database and three applications: Network Element Designer (NED), Network Equipment Designer (EQD) and Network Topology Designer (NTD) which are user-friendly intelligent graphical tools for creating and interconnecting data of the network components. Verification was successfully carried out using the TND expert system for loading TRNET database with data of a real network configuration model.

NED, EQD and NTD are programmed in Visual C++ for Windows platforms. Repository database is implemented in ORACLE RDBMS.

Ključne riječi

Uređaj, mrežni element, dizajn topologije mreže, TND, informacijski sustav

Keywords

Equipment, Network Element, network topology design, TND, information system

Životopis

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxx.

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.

BESPLATNI GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RAD.

RADOVI IZ SVIH OBLASTI, POWERPOINT PREZENTACIJE I DRUGI EDUKATIVNI MATERIJALI.

WWW.SEMINARSKIRAD.ORG

WWW.MATURSKIRADOVI.NET

WWW.MATURSKI.NET

WWW.SEMINARSKIRAD.INFO

WWW.MATURSKI.ORG

WWW.ESSAYSX.COM

WWW.FACEBOOK.COM/DIPLOMSKIRADOVI

NA NAŠIM SAJTOVIMA MOŽETE PRONAĆI SVE, BILO DA JE TO SEMINARSKI, DIPLOMSKI ILI MATURSKI RAD, POWERPOINT PREZENTACIJA I DRUGI EDUKATIVNI MATERIJAL. ZA RAZLIKU OD OSTALIH MI VAM PRUŽAMO DA POGLEDATE SVAKI RAD, NjEGOV SADRŽAJ I PRVE TRI STRANE TAKO DA MOŽETE TAČNO DA ODABERETE ONO ŠTO VAM U POTPUNOSTI ODGOVARA. U BAZI SE NALAZE GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RADOVI KOJE MOŽETE SKINUTI I UZ NJIHOVU POMOĆ NAPRAVITI JEDINSTVEN I UNIKATAN RAD. AKO U BAZI NE NAĐETE RAD KOJI VAM JE POTREBAN, U SVAKOM MOMENTU MOŽETE NARUČITI DA VAM SE IZRADI NOVI, UNIKATAN SEMINARSKI ILI NEKI DRUGI RAD RAD NA LINKU IZRADA RADOVA. PITANjA I ODGOVORE MOŽETE DOBITI NA NAŠEM FORUMU ILI NA MATURSKIRADOVI.NET@GMAIL.COM