VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Michal Marcin

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE OBJASŇUJÍCÍ ZÁKLADY FUNGOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ
APPLICATION CLARIFYING BASICS OF OPERATIONS OF COMMUNICATION PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michal Marcin**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Jeřábek, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Michal Marcin ***ID:*** 174350

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Aplikace objasňující základy fungování komunikačních protokolů

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku komunikačních protokolů a zejména různých režimů přenosu v paketových sítích a také mechanizmů ARQ (Automatic Repeat Request). Navrhněte a popište dva komplexní scénáře, na kterých bude možné si vyzkoušet tyto režimy přenosu. Musí být možné nastavovat různé atributy v různých situacích (propustnost, zpoždění, chybovost, velikost okna, způsob přenosu apod.). Předpokládá se realizace v podobě aplikace nebo webové aplikace, ve které bude možné emulovat chování daných komunikačních protokolů bez nutnosti přenosu po reálné síti a bez nutnosti editovat zdrojový kód. Výstupem práce budou dva kompletní scénáře včetně podrobných návodů pro studenty, prokonzultovaných s vedoucím práce, předpřipravených výchozích situací, jednoduchého grafického prostředí pro nastavení scénáře, komentovaných zdrojových kódů, doplňujících úkolů pro studenty a vzorového řešení. Předpokládá se, že délka realizace jedné úlohy bude pro studenta přibližně 2 hodiny času.

DOPORUČENÁ LITERATURA:

[1] Kurose, J. F., Ross, K. W., Computer networking: a top-down approach. 7th global ed. Essex: Pearson, 2017, 852 s. ISBN 978-1-292-15359-9.

[2] JEŘÁBEK, J. Komunikační technologie. Skriptum FEKT Vysoké učení technické v Brně, 2018. s. 1-172.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: doc. Ing. Jan Jeřábek, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Cieľom diplomovej práce bolo naštudovať problematiku komunikačných protokolov a viacerých režimov prenosu, ako aj mechanizmov ARQ (Automatic Repeat Request). Následne navrhnúť a popísať jednotlivé scenáre pre ich simuláciu. V rámci riešenia úlohy bolo potrebné vytvoriť aplikáciu umožňujúcu emuláciu chovania komunikačných protokolov bez nutnosti prenosu v reálnej sieti a potreby editovať zdrojový kód. Aplikácia bola vytvorená v prostredí Microsoft Visual Studio 2017 s použitím programovacieho jazyka C# a .NET frameworku a je zložená z knižnice a grafického rozhrania. Výstupom riešenia je aplikácia simulátora režimov prenosu dát v sieti s dvoma scenármi s pripravenými vstupnými situáciami v rámci grafického prostredia spolu s návodmi, doplňujúcimi úlohami a vzorovými riešeniami. Program umožňuje simuláciu správania sa komunikačných protokolov medzi klientom a serverom bez potreby prenosu v reálnej sieti. V závere možné konštatovať, že sa podarilo vytvoriť simulátor prenosu dát formou desktopovej aplikácie, ktorá obsahuje dva scenáre. Prvý slúžiaci na simuláciu ARQ mechanizmov a druhý simulujúci komutácie správ, okruhov, paketov alebo buniek.

KĽÚČOVÉ SLOVÁ

ARQ mechanizmus, Stop-and-Wait, Go-back-N, Selective repeat, komutácie okruhov, komutácie paketov, komutácie buniek, komutácie správ, simulácia, ISO/OSI, Visual Studio, objektovo orientované programovanie

ABSTRACT

The diploma thesis aimed at the study of the topic of communication protocols and several transmission modes, as well as ARQ (Automatic Repeat Request) mechanisms. Subsequently, the task was to design and describe individual scenarios for their simulation. As a part of solving the mentioned task, it was necessary to create an application that allows the emulation of the behaviour of communication protocols without the need for a transmission in the real network a nd the requirements for the edition of the source code. The application was created in the Microsoft Visual Studio 2017 development environment using the C# programming language and .NET framework and it consists of a library and a graphical interface. The output of the solution is the application of a mode simulator of the data transmission in the network with two scenarios with the prepared input situations in the frame of graphical environment together with instructions, additional tasks and sample solutions. The program allows the simulation of the behaviour of communication protocols between the client and the server without the need for a transmission in a real network. In conclusion, the simulator of data transfer was created in the form of the desktop application which contains two scenarios. The first scenario is used to simulate ARQ mechanisms and the second one is active in the simulation of commutation of messages, circuits, packets or cells.

KEYWORDS

ARQ mechanism, Stop-and-Wait, Go-back-N, Selective repeat, message switching, circuit switching, packet switching, cell switching, simulation, ISO/OSI, Visual Studio, object oriented programming,

MARCIN, Michal. *Aplikácia objasňujúca základy fungovania komunikačných protokolov*. Brno, 2020, 87 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Ing. Jan Jeřábek, Ph.D.

Vysadené pomocou balíčku `thesis` verzie 3.05; `http://latex.feec.vutbr.cz`

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Aplikácia objasňujúca základy fungovania komunikačných protokolov“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce pánovi doc. Ing. Janovi Jeřábkovi Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Rovnako tak chcem poďakovať svojej rodine za pomoc a podporu pri štúdiu.

Tato práca vznikla ako súčasť kľúčovej aktivity KA6 - Individuálna výuka a zapojenie študentov bakalárskych a magisterských študijných programov do výskumu v rámci projektu OP VVV Vytvorení double-degree doktorského študijného programu Elektronika a informační technológie a vytvorení doktorského študijného programu Informační bezpečnosť, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancovaný Európskou úniou.

Obsah

Zoznam obrázkov

Úvod

Pre správne fungovanie a výmenu dát je dôležité zaistiť rovnakú interpretáciu prijatých dát. Túto funkciu zaisťujú protokoly. Pre jasné oddelenie funkcionality jednotlivých protokolov bol vytvorený model TCP/IP a neskôr ISO/OSI. Hlavný rozdiel medzi danými modelmi je v počte vrstiev. Model TCP/IP popisuje štyri vrstvy zatiaľ čo model ISO/OSI obsahuje sedem vrstiev. Tieto modely sú bližšie popísané v kapitole 1. Na základe spomenutých modelov boli neskôr vytvorené protokoly a metódy na bezpečný a spoľahlivý prenos dát cez komplexnú sieť.

Nasledujúca kapitola 2 sa venuje problematike prenosu dát a sieťovým parametrom. Pri prenose informácií sa rozlišujú štyri spôsoby prenosu: komutácia okruhov, správ, paketov a buniek. Spomenuté techniky a ich rozdiely sú detailnejšie popísané v kapitole 3. Na základe spomenutých postupov boli navrhnuté viaceré simulátory, ktoré majú detailnejšie ukázať fungovanie týchto technológií. Pri návrhu siete je snaha docieliť bezchybný prenos dát medzi stanicami. Tento problém je možné minimalizovať použitím jedného z troch protokolov ARQ (Automatic repair request). Jednotlivé protokoly sú podrobnejšie popísané v podkapitole 4.1.

Pre jednoduchšie porozumenie funkcií spomenutých techník a komutácií boli vytvorené viaceré simulačné aplikácie. Jednotlivé programy sú následne popísané v kapitole 5. Aplikácie majú množstvo výhod ako aj nevýhod. Z toho dôvodu bol vytvorený simulátor, ktorý vznikol za účelom spojiť výhody a eliminovať ich nevýhody.

Vytvorený program popísaný v kapitole 7, bol zostavený s použitím objektovo orientovaného programovania, kde významné časti kódu boli vložené aj do prílohy. Naprogramovanie simulátora bolo možné s využitým vývojového prostredia Microsoft Visual Studio 2017. Aplikácia simuluje dva scenáre a to ARQ mechanizmus a štyri spôsoby komutácií. Jednotlivé funkcie a možnosti týchto scenárov sú následne popísané pomocou jednotlivých úloh v kapitole 8 a pre komutácie následne v kapitole 9.

1 Sieťový model ISO/OSI a TCP/IP

Prenos informácie medzi jednotlivými sieťovými prvkami je realizovaný na základe dohodnutých pravidiel tzv. protokolov. Protokolom sa chápe skupina pravidiel určujúcich formát a význam rámcov, paketov a správ vymieňaných medzi partnerskými entitami. Pri komunikácií v sieti je nutné rozlíšiť takzvanú vertikálnu a horizontálnu komunikáciu [16].

Vertikálna komunikácia prebieha formou požiadavky od najnižšej vrstvy po najvyššiu a naopak. V prípade komunikácie medzi dvoma účastníkmi sú vygenerované aplikačné dáta zapuzdrené, čím sa pridá smerovacia informácia kam majú byť dané dáta odoslané. Tento proces bude detailnejšie popísaný v kapitole 2.1. Jednotlivé dáta sú poslednou vrstvou zakódované a následne prenesené k príjemcovi. Druhá strana obdrží na najnižšej vrstve signál, ktorý je dekódovaný. Ten sa následne odovzdá vyšším vrstvám. V poslednej vrstve sa prijatý tok dát zobrazí užívateľovi v použitej aplikácií [16].

Horizontálnou komunikáciou chápeme prenos medzi dvoma zodpovedajúcimi úrovňami, kde sa nachádza spoločný protokol [13]. V praxi sa na zariadeniach používa vrstvový model ISO/OSI alebo TCP/IP, zobrazené na obr.1.1. Tieto modely budú bližšie popísané v následujúcich kapitolách [16].

1.1 Model ISO/OSI

Tento model predstavuje spôsob špecifikácie vrstvového modelu. Prvé štyri vrstvy popisujú spôsob prenosu dát medzi koncovými stanicami. Prvá vrstva modelu je fyzická. Táto vrstva špecifikuje konkrétne prenosové parametre, ktoré sú potrebné k prenosu informácií. Napríklad vhodne zvolené napäťové hodnoty v metalickom vedení alebo prenosovú rýchlosť [16].

Fyzická vrstva je rovnako tak zodpovedná za prenos informácie cez zvolené prenosové médium vo forme bitov. Prijatý signál je transformovaný do série jednotiek a núl a následne odoslaný druhej vrstve. Táto vrstva sa označuje ako spojová vrstva. Jej hlavná úloha je rozdelenie paketov do rámcov a naopak. Pri vytvorení rámca dochádza k pridaniu hlavičky v ktorej je unikátna zdrojová a cieľová MAC adresa (Media access control). Uložením tejto adresy je možné identifikovať odosielateľa a rovnako tak určiť príjemcu správy. MAC adresa je zložená z 48 bitového čísla a predstavuje jedinečným identifikátor sieťového adaptéra. Následujúcu vrstvou modelu je sieťová vrstva. Významnou úlohou danej vrstvy je vytvorenie logického spojenia od zdroja k cieľu. Na zostavenie spojenia sú potrebné smerovacie protokoly využívajúce logické adresy. Tieto adresy sú uložené v hlavičke dátovej jednotky datagramu. Najčastejšie sa jedná o adresy IP protokolu verzie 4 prípadne verzie 6.

Vytvorený datagram je následne odoslaný do transportnej vrstvy. Úlohou štvrtej vrstvy je riadenie toku dát, chybových stavov a adresovanie služieb použitím čísla portu. Pri doručení poškodených dát môže táto vrstva požiadať o opätovný prenos chybne doručenej dátovej jednotky. Nad transportnou vrstvou sa následne nachádza piata relačná vrstva, ktorá zaisťuje oddelenie rôznych dát aplikácií. Následne prijaté dáta je nutné prezentovať užívateľovi, prípadne zabezpečovať ich šifrovanie a dešifrovanie. Tento proces má na starosti prezentačná vrstva. Poslednou časťou modelu ISO/OSI je aplikačná vrstva [16].

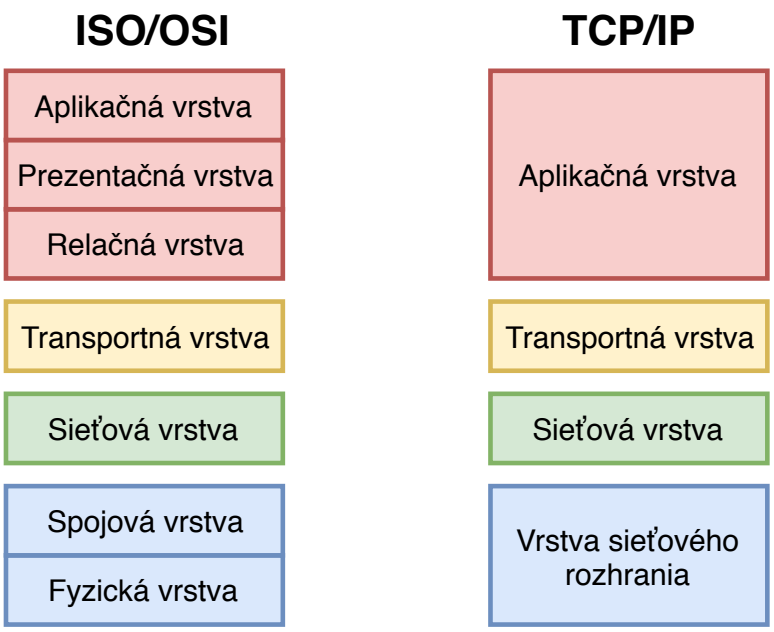
1.2 Model TCP/IP

Model TCP/IP vznikol z dôvodu udržania integrity dát a komunikácie. Na rozdiel od predchádzajúceho modelu tento sa skladá iba zo štyroch častí: aplikačnej, transportnej, sieťovej a vrstve sieťového rozhrania. Aplikačná vrstva združuje funkcie relačnej, prezentačnej a aplikačnej vrstvy. Transportná vrstva odpovedá štvrtej vrstve modelu ISO/OSI, ako aj následne internetová vrstva sieťovej vrstve. Spojová a fyzická vrstva je následne zlúčená do vrstvy sieťového rozhrania [1].

Každá zo spomenutých vrstiev implementuje jedinečný protokol pracujúci na zodpovedajúcej vrstve. Základnou, prvou vrstvou je vrstva sieťového rozhrania zodpovedná za odosielanie TCP/IP paketov do internetu. Je rovnako tak zodpovedná za príjem a kontrolu prijatých dát [1].

Nasledujúca sieťová vrstva, často označovaná ako internetová je zodpovedná za výber najvhodnejšej cesty od zdroja k cieľu. Vhodná cesta a smerovanie je dosiahnuté za použití internet protokolu (IP). Hlavnou funkciou je definovať paket použitím adresy, prenášať dáta medzi vrstvou sieťového rozhrania a transportnou vrstvou. Protokol je zodpovedný za rozdelenie a následné znovu zloženie paketov na strane príjemcu. IP protokol je nespojovo orientovaný, čo znamená, že nenadväzuje spojenie pred samotným odoslaním paketov. Rovnako tak protokol nepotvrdzuje správne prijaté správy. Funkcia nadviazania a spoľahlivého prenosu je implementovaná v transportnom protokole (TCP) [1].

Transportná vrstva predstavuje v poradí 3 vrstvu modelu TCP/IP. Jej hlavná úloha je podobná ako pri transportnej vrstve modelu ISO/OSI. Poskytuje schopnosť spojenia koncových zariadení. Na transportnej vrstve sú definované najpoužívanejšie protokoly, TCP a UDP. Protokol UDP je podobne ako IP protokol spojovo nespojovo orientovaný. Pri prenose nie je zaručený prenos všetkých dát medzi koncovými účastníkmi. Veľkou výhodou spomínaného protokolu je jeho rýchlosť nakoľko nenadväzuje spojenie pred zahájením prenosu a rovnako tak nekontroluje správnosť prijatých dát [1]. Protokol TCP je na rozdiel od UDP spojovo orientovaný zabezpečujúci spoľahlivý prenos. Pred zahájením komunikácie protokol TCP musí zostaviť



Obr. 1.1: Porovnanie sieťových modelov ISO/OSI a TCP/IP

spojenie medzi koncovými bodmi. Po skončení prenosu je dané spojenie ukončené a médium je voľné pre nasledujúcu komunikáciu. Zaistenie správneho fungovania a spoľahlivého prenosu je docielené opakovaným prenosom stratených alebo nedoručených segmentov [1].

Posledná vrstva modelu obsahuje veľkú skupinu protokolov a aplikácií využívajúcich transportné protokoly. Najviac používaný je napríklad protokol na prenos súborov (FTP) a prenosu elektronickej pošty (SMTP). Väčšina protokolov aplikačnej vrstvy je textovo orientovaná. Potencionálny útočník je schopný odoslané správy zachytiť a pozmeniť. Pre odstránenie tejto slabiny boli vyvinuté zabezpečené protokoly ako napríklad zabezpečený protokol na prenos hypertexotvého obsahu (HTTPS) [1].

2 Systém prenosu dát

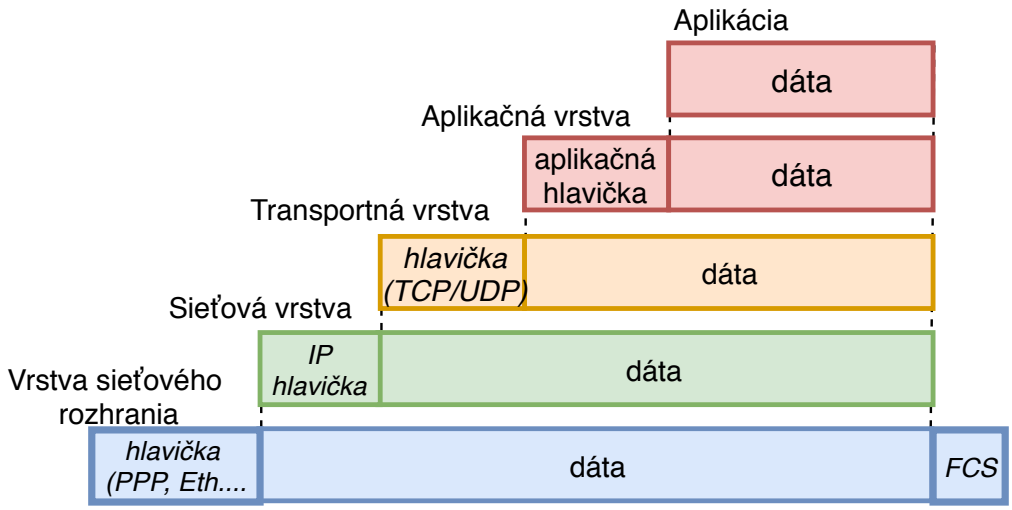
Systémom na prenos dát sa môže chápať súbor krokov, ktoré musia byť vykonané, aby sa informácia spoľahlivo preniesla od zdroja k cieľu. Jednotlivé časti a funkcie daného systému sú popísané v nasledujúcich podkapitolách.

2.1 Zapuzdrenie dát

Každá z vrstiev modelu ISO/OSI spomenutého v kapitole 1.1 komunikuje len so zodpovedajúcou partnerskou vrstvou prijímajúceho zariadenia. V praxi sa fyzicky dáta smerujú vertikálne od použitej aplikácie k sieťovej karte, ktorá ich presmeruje na príslušný výstupný port. Pre smerovanie takýchto dát slúžia prenesené dodatočné informácie, ktoré boli pripojené k pôvodným dátam [16].

V každej vrstve sa používajú dátové jednotky protokolu (protocol data unit, PDU). Tieto obsahujú riadiace informácie, potrebné pre správne fungovanie protokolu na jednotlivých vrstvách. Najčastejšie sú tieto informácie zapísané v hlavičke pred dátovým poľom, ale môžu byť takisto aj na konci [16].

Jednotka PDU sa teda pripája k dátam v príslušnej vrstve. Tento proces sa nazýva zapúzdrenie dát. Schematický je proces zapuzdrenia dát v modeli TCP/IP znázornený na obr. 2.1 [16].



Obr. 2.1: Zapuzdrenie aplikačných dát v modeli TCP/IP [10]

Užívateľské dáta sa postupne odovzdávajú z aplikačnej vrstvy na transportnú. Na tejto úrovni je základnou jednotkou segment a k dátam sa pripojí hlavičku transportnej vrstvy. Následne sa vytvárajú dáta, ktoré sú odovzdané sieťovej vrstve.

Tá potom pripojí zodpovedajúcu hlavičku, najčastejšie sa tak jedná o internet protokol verzie 4 alebo verzie 6. Takto zabalené dáta získajú hlavičku spojovej vrstvy a na konci pätu, kde je spravidla kontrolný súčet. Proces zapuzdrenia funguje analogicky aj pri použití modelu TCP/IP [16].

2.2 Zaistenie obojsmernej komunikácie

Obojsmerná komunikácia predstavuje prenos dát od zdroja k cieľu a naopak. Na fyzickej úrovni sa tento režim prenosu dá dosiahnuť napríklad použitím dvoch vlákien, pre každý smer jedno vlákno. V takomto prípade sa hovorí o duplexnom spojení [10].

Na vyšších vrstvách môže byť obojsmerná komunikácia zaistená výmenou signalizačných dát pred zahájením spojenia. Takto prenesené informácie slúžia pre rezerváciu prenosového média, prípadne rezervovanie sieťových prostriedkov, ktoré sú následne vyhradené len pre daný prenos. Po odoslaní všetkých potrebných dát obidve strany odošlú unikátnu sekvenciu dát, aby ukončili a uvoľnili rezervované prostriedky v sieti.

2.3 Sieťové parametre

Prenosovú sieť, internet, charakterizuje niekoľko parametrov. Na základe hodnôt jednotlivých veličín je možné použiť vhodnú prenosovú technológiu. Bližšie sú tieto parametre popísané v nasledujúcich podkapitolách.

2.3.1 Šírka pásma

Šírka pásma predstavuje maximálnu teoretickú hodnotu bitov prenesených za jednotku času. Napríklad pri použití technológie gigabit Ethernet môže dosiahnuť rýchlosť až 1000 Mb/s (125 megabitov za sekundu). Zatiaľ čo sa šírka pásma používa na popis rýchlosti siete, neznamená to skutočnú hodnotu, s ktorou sú bity prenesené z jednej stanice na druhú. Z dôvodu fyzikálnych vlastností prenosového kanála nie je možné túto hodnotu v praxi dosiahnúť [13].

2.3.2 Prenosová rýchlosť

Termín prenosová rýchlosť, často označovaná aj ako bitová rýchlosť, popisuje počet bitov prenesených od jednej stanice k druhej. Táto hodnota udáva koľko informácií je prenesených cez dátové médium za určitý čas. V praxi sa táto veličina udáva v počet bitov za jednotku času. Základnou jednotkou je následne bit za sekundu (b/s), prípadne kilo bit alebo mega bit. Napríklad pri digitálnom prenose informácií

po telefónnej sieti je možné sťahovať dáta bitovou rýchlosťou až 768 kilo bitov za sekundu [3].

2.3.3 Zdržanie (Delay)

Tento parameter patrí medzi najdôležitejšie v komunikačných sieťach. Zdržanie predstavuje dôležitý prvok pri prenose multimediálneho obsahu. Podľa doporučenia ITU-T G.114 pre zaistenie dostatočnej kvality prenosu by mali byť hodnoty oneskorenia medzi 0 – 150 ms v prípade bežných užívateľských aplikácií, 150 – 400 ms pre medzinárodné hovory a hodnota nad 400 ms by nemala byť prekročená [15].

Celkové zdržanie je možné rozdeliť do viacerých častí, kroré sú popísané v nasledujúcich kapitolách.

Zdržanie spôsobené zostavením aplikačného rámca

Pri zostavení rámca zdrojová stanica dokáže spracovať vždy definovaný úsek dát. Pri audio signále je možné vypočítať túto dobu zo vzťahu:

$$t_f = \frac{n_s}{f_s} \text{ [s]}. \tag{2.1}$$

Hodnota n_s predstavuje počet vzoriek v rámci vzorkovacej frekvencie f_s a výsledná doba zostavenia rámca je t_f [15].

Zdržanie spôsobené zostavením paketu

Po zakódovaní časového úseku zdrojovým kodérom sa musia dáta zapuzdriť do paketu. Najväčším oneskorením je doba čakania dát vo vyrovnávacej pamäti, kým sa nezíska dostatočne veľký počet na ich zostavenie do paketu [15].

Zdržanie spôsobené prekladaním paketu

Prekladanie paketu predstavuje možnosť zabezpečenia digitálneho signálu proti zhlukovým chybám. Pri tejto metóde prekladanie rámca o dĺžke n bitov je nutné uložiť N po sebe idúcich rámcov. Veľkosť vyrovnávacej pamäte bude n^2. Rovnaká podmienka platí aj na strane príjemcu kde k obnoveniu N pôvodných rámcov je potrebné uložiť N prekladaných rámcov s dĺžkou n bitov. Pretože prekladanie prebieha ako na strane vysielača tak i príjemcu a je nutné započítať toto zdržanie do celkového zdržania dvakrát [15].

Zdržanie spôsobené šírením signálu

Pri väčších vzdialenostiach je nutné brať do úvahy čas potrebný k šíreniu signálu od zdroja k príjemcovi cez komunikačné médium. Pre orientačný odhad tejto doby je možné použiť nasledujúci vzťah:

$$t_c = \frac{l}{c} \text{ [s]}. \tag{2.2}$$

Celkový čas šírenia médiom t_c je rovný podielu dĺžky trasy l a rýchlosti svetla vo vákuu c. Pre presnejšie hodnoty je potrebné zobrať do úvahy aj vplyv konkrétneho prenosového média [15].

2.3.4 Kolísavé zdržanie (Jitter)

Kolísavé zdržanie je spôsobené odosielaním dátových jednotiek v nepravidelných intervaloch. Tento jav môže nastať napríklad v prípade, ak je niekoľko dátových jednotiek odoslaných naraz. Dochádza tak v prípade zahltenia siete pri vzniku kolízií alebo pri interferencií. Kompenzácia kolísavého zdržania a odosielanie dátových jednotiek v pravidelných intervaloch môže byť docielená použitím dočasného úložiska (buffer). Aj po použití danej technológie môže dochádzať k neperiodickému odoslaniu dátových jednotiek. Tento jav môže mať za následok zvýšenie stratovosti v sieti. Ak príjemca nedokáže spracovať takýto veľký objem dát naraz, potom začne nové jednotky zahadzovať. Všetky zahodené jednotky musia byť znovu odoslané čo má za následok predĺženie celkového prenosu. V prípade prenosu dátového obsahu v reálnom čase môže dôjsť k výpadkom a zníženiu kvality [4].

2.3.5 Stratovosť

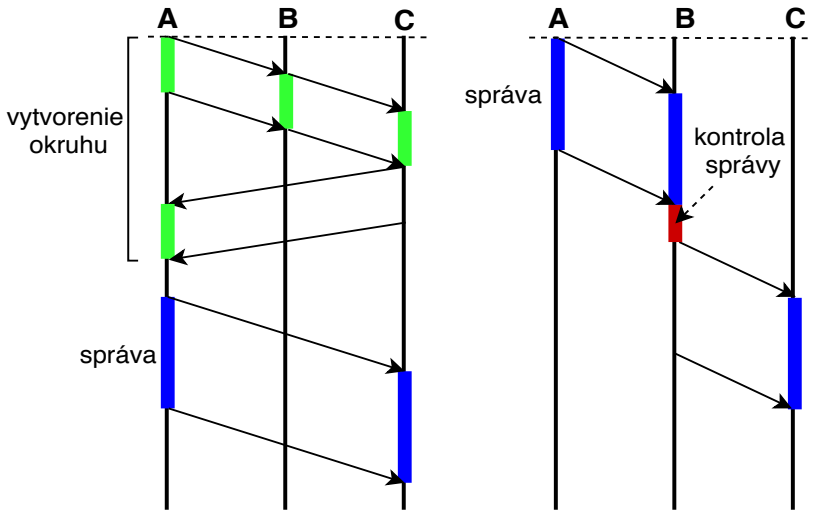
Stratovosť je možné vyjadriť, ako pomer stratených paketov a paketov odoslaných. K strate môže dochádzať spravidla pri vplyve vonkajšieho rušenia na prenosový kanál a tým dôjde k zmene bitov. Ďalšou z možností je napríklad kolízia na úrovni spojovej vrstvy. Na úrovni sieťovej vrstvy môže dochádzať k nesprávnemu smerovaniu paketov [15].

3 Spôsob prenosu informácie

Prenos informácie v telekomunikačnej sieti zjednodušene prebieha medzi dvoma účastníkmi prípadne dvoma procesmi. V závislosti na prenášaných dátach je nutné vybrať vhodnú metódu pre stabilný, spoľahlivý a pokiaľ možno bezchybný prenos. Pred samotným prenosom je potrebné zistiť o aký typ informácie sa bude jednať a následne určiť vhodný režim prenosu.

3.1 Komutácia okruhov

Pri komunikácií okruhov je vytvorený fyzický okruh medzi oboma účastníkmi. Táto fyzická cesta musí byť vytvorená pred zahájením prenosu. Rezervované fyzické prostriedky sú alokované aj vo vnútri samotných uzlov v sieti. Takto zostavené spojenie je finančne náročné aj preto, že účastníci musia platiť za celé spojenie, aj keď sa v danom okamihu nič neposiela [13].



Obr. 3.1: Schéma priebehu prenosu správy pri a) komutácii okruhov, b) komutácii správ [13]

3.2 Komutácia správ

Pri komutácií správ nie je nutné zostavovať celé spojenie ani rezervovať fyzické prostriedky pred prenosom. Ak stanica potrebuje odoslať informácie uloží adresu príjemcu a odošle celú správu k najbližšiemu uzlu. Tento uzol skontroluje prijatú správu a uloží ju do internej pamäte. Podľa smerovej tabuľky sa určí najbližší uzol

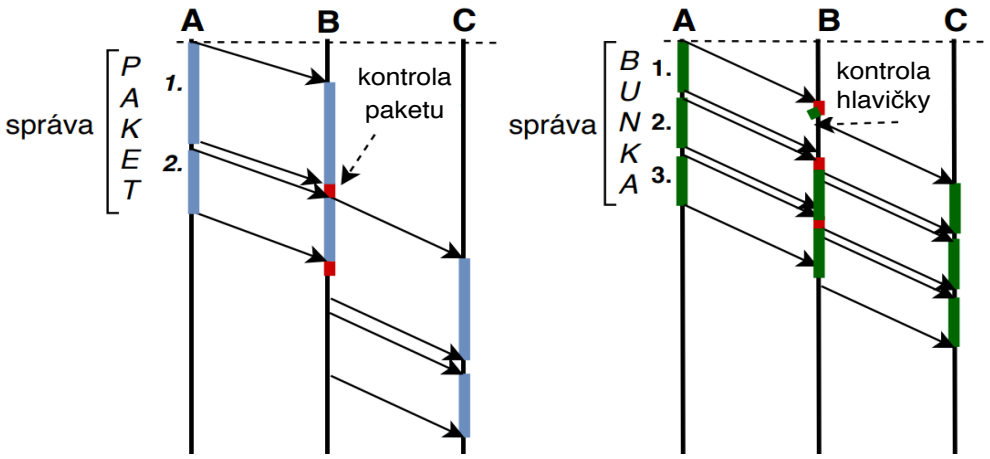
kam sa správa má odoslať. Tento proces pokračuje až, kým nie je správa doručená do cieľovej stanice. Celá správa je najprv uložená v uzle, čo zvyšuje nároky na pamäť zariadenia [13].

3.3 Komutácia paketov

Paketová komutácia predstavuje dnes najbežnejší spôsob prenosu informácie. Pred prenosom sa rozdelí správa na menšie časti a tie sa následne prenášajú v sieti. Na prijímacej strane sú tieto jednotky znova poskladané do ucelenej správy. V prípade ak správa presiahne zvolenú dĺžku rozdelí sa na pakety. Pri takomto rozdelení nie je zaručené, že všetky pakety dorazia v správnom poradí a preto je nutné implementovať dodatočné mechanizmy. Táto jednotka neobsahuje len adresu príjemcu, ale musí mať aj identifikačné číslo na základe, ktorého je možné určiť, o ktorú časť správy sa jedná [13].

3.4 Komutácia buniek

Pri tejto metóde prenosu sa správa rozdelí na presne definované menšie jednotky. Tie následne získajú hlavičku. Samotný uzol pri prenose skontroluje hlavičku, čo výrazne zrýchli prenos k cieľovej stanici. Komutáciu buniek je možné využiť napríklad pri asynchrónnom transportnom móde (ATM), kde táto jednotka má veľkosť 53 bajtov. Spomínaná technológia prináša rýchlejšiu odozvu, ale je obmedzená stanovenou veľkosťou jednotky [13].



Obr. 3.2: Schéma priebehu prenosu správy pri a) komutácii paketov, b) komutácii buniek [13]

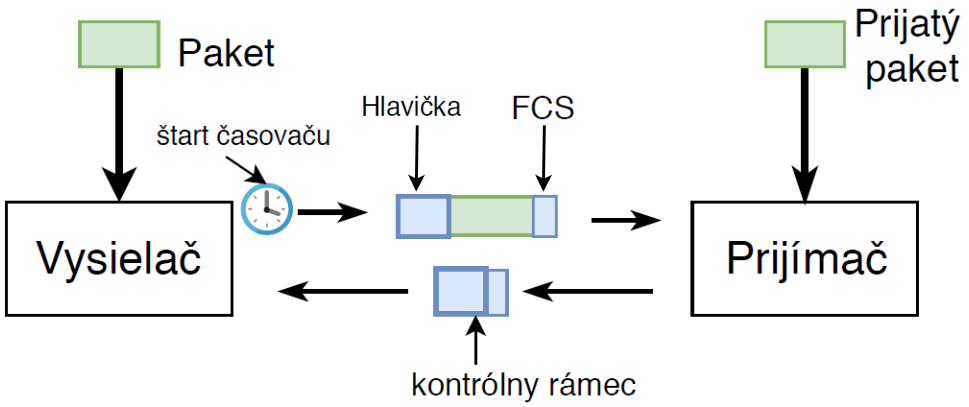
4 Detekcia a riadenie chybových stavov, mechanizmus ARQ

Ak chceme hovoriť o detekcií chýb v dátových komunikáciách musíme definovať, čo je chyba a kedy k nej dôjde. Chybu v telekomunikačnej sieti je možné definovať ako nesúlad medzi odoslanými a prijatými hodnotami. Tieto chyby môžu nastať na úrovni jednotlivých bitov alebo následne na úrovni protokolov. Bitová chybovosť významne závisí od prenosového média a je možné ju minimalizovať použitím protichybových kódov. Zabezpečená bitová postupnosť je potom menej náchylná na vznik chyby pri prenose. Ak sa chyba aj vyskytne, je možné ju relatívne ľahko detekovať a následne odstrániť. Vznik chýb na vyšších úrovniach môže nastať, ak sa jednotlivé dáta nedostali k príslušnému protokolu. V závislosti na použitom protokole môžu nastať dva prípady. V prvom scenári prijímacia strana požiada o zaslanie nových údajov alebo v druhom prípade protokol pracuje bez nich. Napríklad pri prenose multimediálneho obsahu. K chybe spôsobenej na aplikačnej úrovni často dochádza pri vyťaženosti sieťového prvku. Každé zariadenie pracuje s konečnou veľkosťou pamäte a pri jej zaplnení je nutné každú ďalšiu jednotku zahodiť.

Pre riadenie protichybových stavov bol vytvorený mechanizmus opätovného prenosu ARQ [9]. Tento spôsob môže byť použitý ako na linkovej vrstve tak aj na transportnej, kde sa použitím protokolu TCP potvrdzujú prenesené segmenty. Protokol ARQ je možné aplikovať aj na vyššie úrovne ISO/OSI modelu. Postup daného mechanizmu je uvedený v nasledujúcich kapitolách.

4.1 Mechanizmus ARQ

Mechanizmus ARQ vytvára spoľahlivé doručenie informácií medzi dvomi účastníkmi. ARQ je možné použiť na transportnej vrstve spolu s protokolom TCP, ako aj na nižšej spojovej vrstve. Schéma činnosti danej metódy je naznačená na obr. 4.1. Spojová vrstva vysielača prijme paket, ku ktorému pripojí hlavičku a kontrolný súčet FCS. Po odoslaní rámca sa spustí časovač opakovaného prenosu. Prijímač po prijatí rámca skontroluje doručené dáta. Ak nedošlo pri prenose k chybe rozbalený paket je smerovaný k sieťovej vrstve. V prípade chyby v rámci alebo pri jeho nedoručení sa prenos opakuje. V praxi sa rozlišujú tri modifikácie daného mechanizmu: Stop-and-Wait, Go-back-N ARQ a Selective Repeat ARQ [21][9].



Obr. 4.1: Prenos rámca mechanizmom ARQ

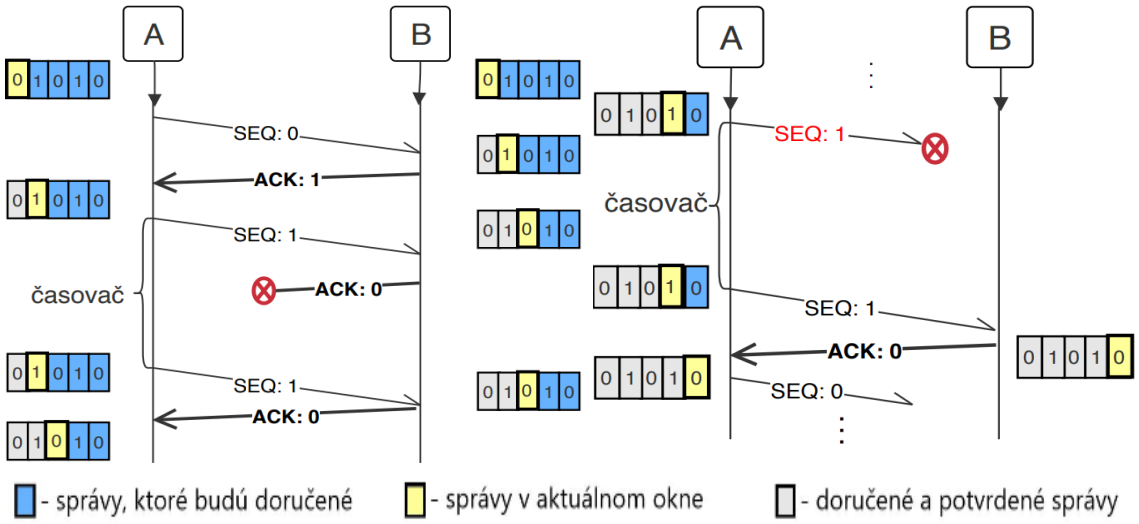
4.1.1 Protokol Stop-and-wait (SW)

Protokol SW predstavuje jednoduchý a spoľahlivý spôsob doručenia dát. Počas doručenia dochádza k prenosu jednej dátovej jednotky určitý čas. Stanice majú v pamäti uloženú len jednu dátovú jednotku, čo výrazne znižuje pamäťové nároky. Po správnom doručení správy je pamäť prepísaná novou dátovou jednotkou. Nevýhoda daného riešenia spočíva vo vysokej neefektivite, počas prenosu musia stanice čakať na doručenie správy až potom môžu zahájiť prenos ďalšej. Prenosové médium nieje tak dostatočne vyťažené. Výhodou takého protokolu je možnosť komunikovať aj pri poloduplexných spojeniach, kedy nie je možné zaistiť obojsmernú komunikáciu.

Popis protokolu SW je znázornený na obr. 4.2. Pri prenose budú označené správy iba jedným bitom: číslom 0 alebo 1. Odosielateľ, stanica A, odošle rámec s číslom 0 (SEQ: 0) a spustí časovač opakovaného prenosu. Hodnota časovača musí byť väčšia ako je doba prenosu správy od zdroja k cieľu a naopak, tzv. RTT(round-trip time). Stanica B po doručení a skontrolovaní rámca odošle potvrdenie s číslom 1 očakávanej správy (ACK: 1). Stanica A obdrží potvrdenie, následne nahradí staré dáta v pamäti s novou správou. Nasledujúci rámec dostane číslo 1 (SEQ: 1) a po odoslaní stanica znovu nastaví časovač opakovaného prenosu. Stanica B opäť skontroluje správu a odosiela potvrdenie (ACK: 0) [5].

Ak sa počas prenosu stratí potvrdenie ACK strana A nedostane informáciu o správnom doručení a musí čakať. Po uplynutí nastaveného časovača opakuje prenos rámca (SEQ: 1). Tento rámec je znovu doručený a strana B odosiela potvrdenie. V tomto prípade sa ACK správne doručí a prenos pokračuje. Podobná situácia nastáva, ak sa stratí alebo poškodí rámec (SEQ: 1). Takýto prípad je znázornený na obrázku 4.2 vpravo. Stanica B neodosiela potvrdenie. Po skončení časovača sa znova odosiela rámec (SEQ: 1), ktorý je správne doručený. Následne ako už bolo spomenuté stanica B odošle potvrdenie (ACK: 0) a prenos pokračuje. Celý protokol tak

využíva len 1 bitové sekvenčné číslo. Stanica buď prijme sekvenčné číslo s hodnotou 1 a odošle 0, alebo naopak [5].



Obr. 4.2: Princíp protokolu Stop and Wait s riadením chybových stavov

4.1.2 Protokol Go-back-N (GBN)

Go-back-N predstavuje modifikovanie algoritmu SW. Protokol podobne odosiela správy, ktoré sú následne potvrdzované. Pri tejto technike dochádza k prenosu viacerých dátových jednotiek z tzv. okna W v krátkom čase. Odosielateľ tak môže poslať nové dáta, aj keď neprial žiadne potvrdenie. Veľkosť okna sa môže počas prenosu dynamicky meniť. Maximálnu hodnotu W je možné vypočítať zo vzťahu:

$$W \leq 2^m - 1 \ [\text{-}], \tag{4.1}$$

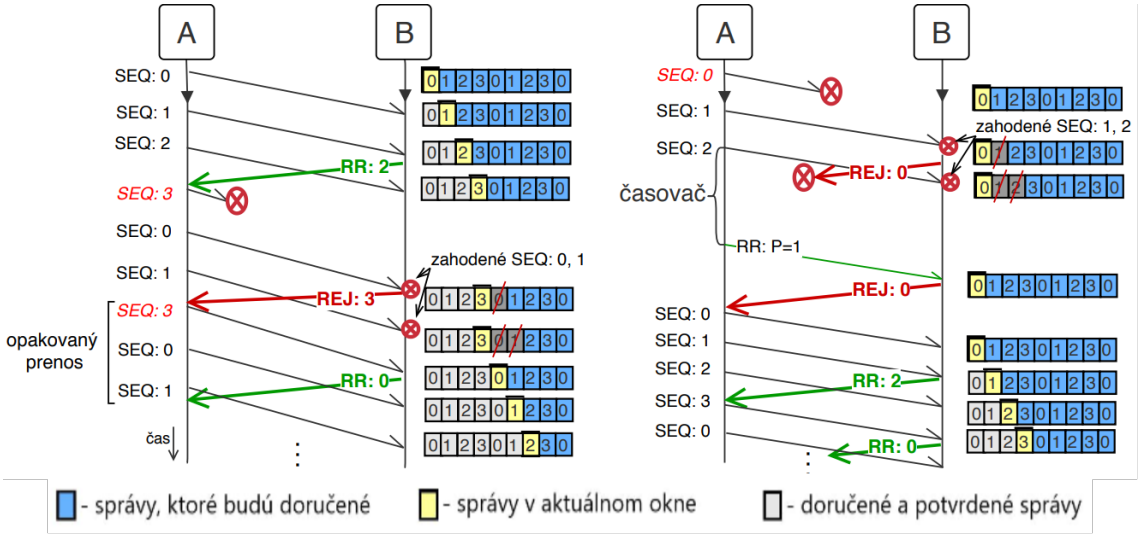
kde m predstavuje počet bitov sekvenčného čísla. Odosielateľ nemôže odoslať dátovú jednotu i+W, ak neprijal potvrdenie o doručení i-tej dátovej jednotky. Príjemca pracuje podobne, ako pri protokole SW. Ak vysielač príjme potvrdenie so sekvenčným číslom i, znamená to, že všetky dátové jednotky po i vrátane boli správne doručené [5].

Pre jednoduchosť sú všetky sekvenčné čísla v nasledujúcom príklade dvojbitové. Veľkosť okna je konštantná a nastavená podľa vzťahu 4.1 na hodnotu 3. Stanica A odosiela postupne všetky rámce ktoré sú v jednom okne. Jednotlivé rámce stanica B potvrdzuje odoslaním potvrdenia RR (Receiver ready) so sekvenčným číslom očakávaného rámca [5].

Na príklade 4.3 došlo k chybe alebo nesprávnemu doručeniu dátového rámca i (SEQ: 3). Vysielač pošle relatívne v krátkej dobe nasledujúce rámce i+1 (SEQ: 0)

a *i+2*(SEQ: 1). Stanica B zahodí novoprijaté rámce pretože ich sekvenčné čísla nie sú v súlade s očakávaným sekvenčným číslom *i* (SEQ: 3). Následne musí stanica A odoslať znovu všetky odoslané rámce od rámca *i* vrátane. Ak po opätovnom prenose stanica A neobdrží potvrdenie vyprší jej nastavený časovač a odošle správu RR s číslom 1. Príjemca vyhodnotí novú správu a posiela RR so sekvenčným číslom očakávaného rámca. Stanica A zaháji prenos nových správ vrátane správy *i* [15].

Potvrdenia RR majú kumulatívnu vlastnosť. Znamená to, že každé potvrdenie s číslom *i* potvrdzuje prijatie všetkých predchádzajúcich rámcov. Ak stanica A príjme potvrdenie *i+2* ešte pred vypršaním časovača pre potvrdenie *i*tého rámca dôjde k potvrdeniu súčasne rámcov *i* a *i+1*. Potvrdia sa aj rámce, ktoré neboli predtým korektne potvrdené [15].



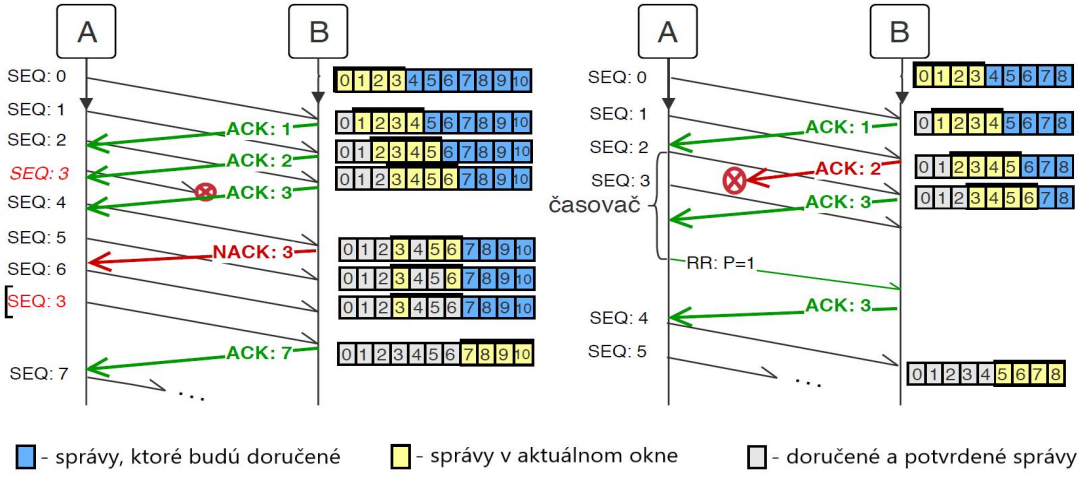
Obr. 4.3: Schéma prenosu dát pri protokole Go-back-N, zobrazenie straty dátovej jednotky a potvrdenia

Ak stanici A vyprší časovač pre potvrdenie rámca odosiela žiadosť na zaslanie potvrdenie *i*tého rámca správou RR s bitom P. Hodnota bitu je nastavená na 1. Po doručení stanica B odošle vyžiadané potvrdenie RR [15].

4.1.3 Protokol Selective repeat (SR)

Protokol SR je podobný protokolu GBN. Vysielač v pravidelných intervaloch odosiela rámce, ktoré sú uložené v dočasnej pamäti. Po prijatí kladného potvrdenia ACK sú vymazané. Rovnako, ako všetky protokoly ARQ aj SR protokol musí zahájiť spustenie časovača pre každý odoslaný rámec. Protokol môže pracovať v dvoch režimoch. V prvom kumulatívnom, dochádza k potvrdzovaniu každej n-tej dátovej

jednotky. V druhom režime protokol potvrdzuje každú prijatú správu, ako je to znázornené na obr. 4.4 [5]. V príklade bola veľkosť sekvenčného čísla nastavená na 3 bity a podľa vzorca 4.2 nastavená na hodnotu 4.



Obr. 4.4: Schéma prenosu dát použitím protokolu selective repeat s veľkosťou okna 4, zobrazenie straty dátovej správy a potvrdenia

$$W \leq 2^{m-1} \text{ [-]}, \tag{4.2}$$

Prenos začína prvou správou (SEQ: 0), ktorá je doručená na stranu B. Príjemca potvrdí prijem správy odoslaním potvrdenia (ACK: 1) a posunie okno na nasledujúce 4 segmenty. Prenos pokračuje až po stratu správy i (SEQ: 3), príjemca odosiela negatívne potvrdenie (NACK: 3) s číslom nedoručenej správy i. Všetky prijaté správy sú uložené v dočasnej pamäti, pričom okno ostáva na mieste správy 3. Prijatím negatívneho potvrdenia strana A odošle po uvoľnení média opäť správu (SEQ: 3). Tá je následne doručené a spolu s uloženými správami z dočasnej pamäti odoslaná vyššej vrstve [5].

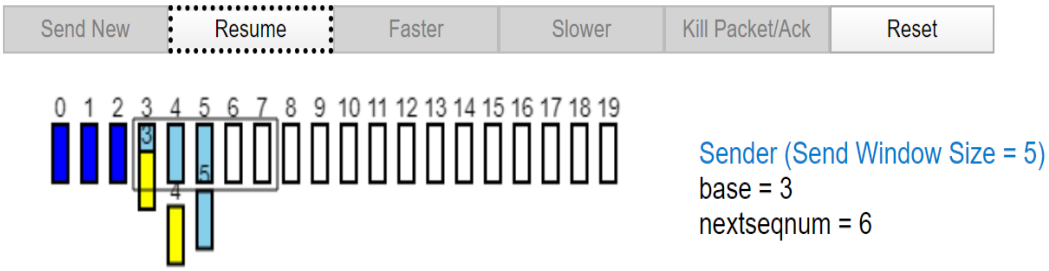
Pri strate potvrdenia stanica postupuje podobne ako v prípade GBN protokolu. V konkrétnom prípade obr. 4.4 došlo k strate správy (ACK: 2). Nastavený časovač odoslanej správy (SEQ: 2) uplynie. Po uvoľnení prenosového média nastáva prenos správy RR s príznakovým bitom P nastaveným na hodnotu 1. Príjemca, ktorý nevedel o strate potvrdenia opakuje tento prenos znovu. Opakovaným doručením správy na stranu A je stanica informovaná o úspešnom doručení správy (SEQ: 2). Prijatím potvrdenia posunie odosielateľ okno a začína odosielať nasledujúce dáta [5].

5 Simulačné techniky mechanizmu ARQ

Pre lepšie pochopenie jednotlivých techník spoľahlivého prenosu boli vytvorené simulátory. Jednotlivé simulačné programy a webové aplikácie sú predstavené v nasledujúcich podkapitolách.

5.1 Shatley simulátory protokolu SR a GBN

Simulačné programy vytvorené M. Shatleym a Ch. Hoffmanom predstavujú jednoduché internetové aplikácie [18][19]. Prenos paketov je znázornený farebnými obdĺžnikmi, ktoré neskôr zmenia farbu obr. 5.1. Na začiatku je každý paket bez farby pripravený na odoslanie.



Obr. 5.1: Shatley simulátory metód GBN a SR [19]

Spustenie simulácie a zároveň odoslanie paketu je možné tlačidlom „Send New“ (Odoslať nový). Udalosť je zaznamenaná do stavového okna na spodnej strane internetovej stránky. Odoslaný paket je svetlo modrý a po prijatí na strane príjemcu sa označí červenou. Príjemca následne potvrdí prijatie a odosiela nový paket označený žltou farbou. Každý odoslaný paket je možné zahodiť a simulovať tak opakovaný prenos. Túto akciu je možné spustiť kliknutím na pohybujúci sa paket, ktorý znení farbu na zeleno. Označený paket sa zahodí až po kliknutí tlačidla „Kill Packet/Ack“(Zmazanie paketu/ACK). Konzola zobrazí informáciu o strate paketu a po skončení časovača nastane opakovaný prenos.

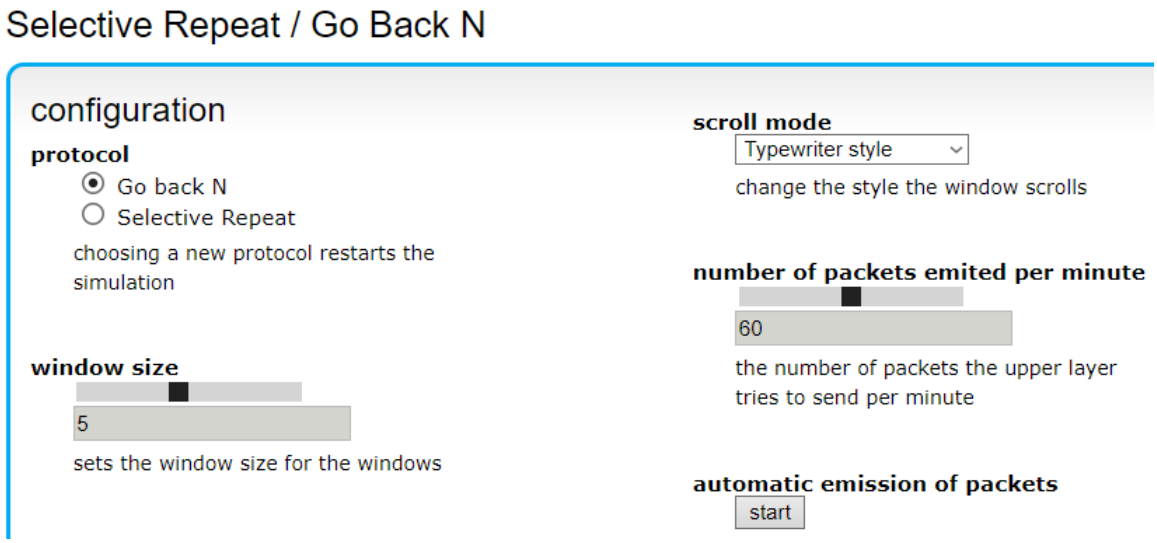
5.1.1 Výhody a nevýhody simulačného programu metód SR a GBN

Výhodou simulácií sú nízke pamäťové nároky a rovnako tak aj dostupnosť. Vďaka webovému serveru je možné obidva simulátory spustiť na akomkoľvek zariadení pripojenom na internet. Celú simuláciu možno urýchliť, spomaliť alebo pozastaviť a vidieť tak stav prenosu v určitom časovom okamihu.

Na druhej strane je veľkou nevýhodou konštantná veľkosť okna, ktorú nie je možné zmeniť. Podobne je to aj s počtom paketov. Simulácie nefungujú v automatickom režime a pre odoslanie paketu je nutné použiť tlačidlo „Send New“(Odoslať nový). Opakované odoslanie paketu nastane po uplynutí časovača na opakovaný prenos, ktorý žiaľ nie je nikde viditeľný. Časovač je spustený na pozadí a o jeho uplynutí alebo ukončení je užívateľ informovaný správou v konzole a opätovným odoslaním paketu.

5.2 Kessler simulačný program „Selective Repeat/Go Back N“

Druhým programom určeným na simuláciu mechanizmu ARQ je webová aplikácia vytvorená J. Kesslerom [14]. V porovnaní s predchádzajúcim simulátorom je v aplikácií možné upravovať väčší počet parametrov.



Obr. 5.2: Webová aplikácia „Selective Repeat/Go Back N“ [14]

Časť grafického rozhrania aplikácie je znázornený na obr. 5.2. Táto webová aplikácia ponúka simulovanie protokolu GBN ako aj SR. Nastavenie režimu je dostupné v pravom hornom rohu. Pod špecifikovaním použitého protokolu je posuvník, ktorým je možné upraviť veľkosť okna v rozsahu jedna až desať. Nasledujúca položka umožňuje zmenu hodnoty RTT v rozsahu od $10^3 - 20 \cdot 10^3$ sekúnd. Hodnota „timeout“(časovač) udáva hodnotu časovača pre opakovaný prenos. Ten môže nadobúdať hodnoty medzi $10^3 - 30 \cdot 10^3$ sekúnd. Časovač je následne zobrazený ako fialový kruh vedľa prvého odoslaného paketu na strane odosielateľa. Poslednou

možnosťou je počet emitovaných paketov za minútu. Aplikácia povoľuje zmenu tejto hodnoty aj po spustení čo pri ostatných hodnotách nieje možné. Zmenou danej hodnoty je možné ovplyvniť len rýchlosť aplikácie. Zrýchlenie nastane pri zväčšení čísla a naopak.

Po zvolení parametrov a stlačením tlačidla „start“ simulátor odosiela pakety znázornené v forme farebných obdĺžnikov. Celý prenos paketov prebieha po spustení automaticky a je znázornený v dolnej časti. Odosielateľ je nad príjemcom a v počiatočnom stave má všetky pakety označené svetlo modrou. Príjemca v dolnej časti má na začiatku označené pakety bielou. Odoslaný paket tak smeruje v vertikálnom smere a je označený svetlo modrou farbou. Po úspešnom doručení sa táto farba zmení na tmavo modrú. Potvrdenie je následne odoslané zelenou a všetky potvrdené pakety sú označené žltou farbou. Program je možné ukončiť tlačidlom „stop“.

5.2.1 Výhody a nevýhody simulačného programu Kesler Selective Repeat/Go Back N

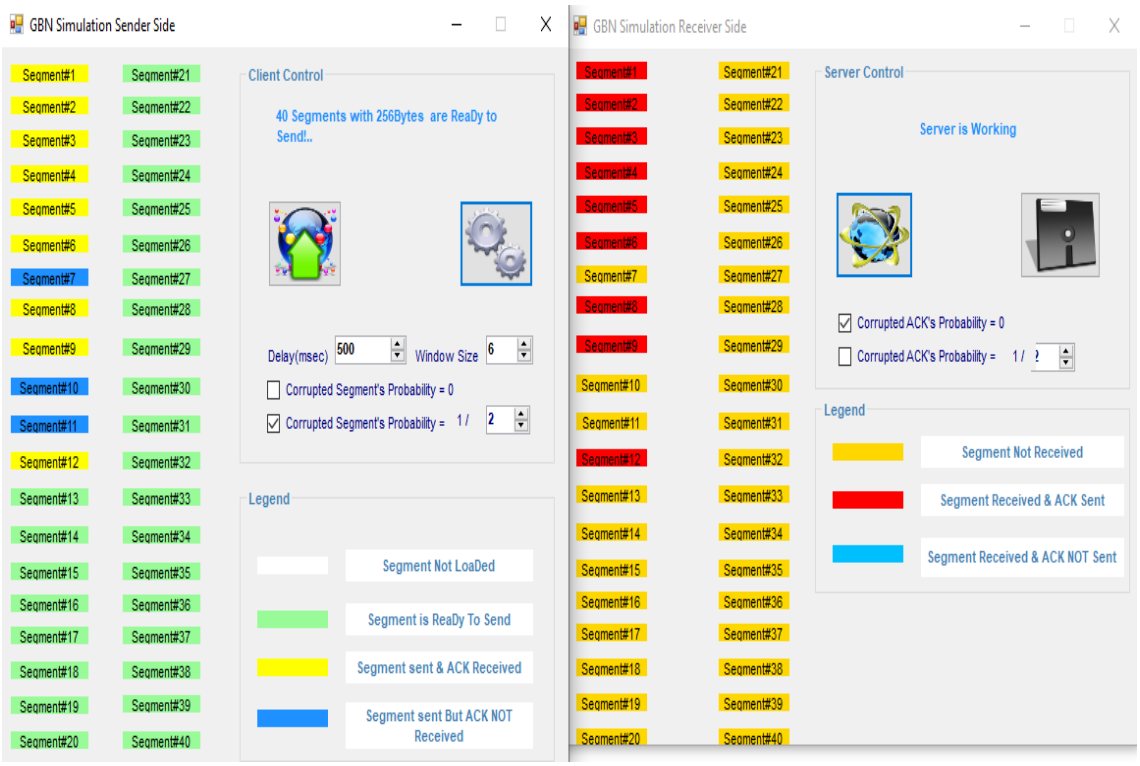
Aplikácie je dostupná online na webovej adrese [14]. Významnou výhodou je možnosť nastaviť základné parametre a tak upraviť priebeh simulácie. Pri zvolení veľkosti okna na hodnotu 1 sa simulácia prebieha ako základný ARQ protokol SW. Táto možnosť pri predchádzajúcom programe nebola možná. Jednou z výhod je aj automatické odosielanie paketov po spustení simulácie a viditeľné znázornenie časovača opakovaného prenosu.

Pri veľkom počte výhod aplikácia nezobrazuje jednotlivé sekvenčné čísla paketov, ktoré sú prenášané. Nevýhoda je obmedzenie pri nastavovaní parametrov. Nie je možné zvoliť hodnotu časovača menšiu ako 1000. Po ukončení aplikácie sa nevytvorí žiaden záznam o prenesených paketoch.

5.3 Simulátor GBN

Aplikácia bola vytvorená v programe C# verzia 2.0 [17]. Simulátor je zložený z dvoch nezávislých programov, klienta a server obr. 5.3.

Na začiatku klient musí načítať súbor o veľkosti 10240 bitov. Počet bitov je obmedzený na základe počtu segmentov, ktoré sa počas simulácie odošlú. V nastaveniach je možné zvoliť pravdepodobnosť chyby odoslania segmentov a veľkosť okna, ktorá udáva počet segmentov odoslaných v rovnakom čase. Ak boli pri prenose niektoré segmenty stratené server neodošle potvrdenie. Pri správnom doručení klient nemusí vždy prijať potvrdenie, ak bola nastavená pravdepodobnosť vzniku chýb na strane servera.



Obr. 5.3: Desktopová aplikácia protokolu Go-back-N [17]

5.3.1 Výhody a nevýhody simulačného programu

Program dokáže simulovať nespoľahlivý prenosový kanál použitím protokolu TCP a ARQ protokol Go-back-N, pri nastaveniach klienta, ako aj servera. Jednou z výhod je možnosť nastavovať základné parametre. Postup simulácie je prehľadne viditeľný v grafickom rozhraní.

Nevýhodou je obmedzená veľkosť vstupného súboru, ako aj zvýšené pamäťové nároky. Aplikácia využíva viacero vláken a potrebuje tak väčšiu výpočtovú kapacitu.

5.4 Porovnanie simulačných nástrojov mechanizmu ARQ

Popísané simulátory v predchádzajúcich kapitolách poskytujú vhodný nástroj na pochopenie mechanizmu ARQ. Vo všetkých simulátoroch je jednoduché rozlíšiť a nastaviť základné parametre. Výhodou týchto programov je ich dostupnosť na webovom serveri, až na simulátor GBN, spomenutý v podkapitole 5.3.

Dané riešenia predpokladajú dostupnosť internetového pripojenia. Ak by nebol server dostupný nie je možné spustiť simuláciu. Tento problém je možné vyriešiť

uložením simulačného programu na klientsku stanicu, čím sa dosiahne nezávislosť. Spomenutý problém vyriešila aplikácia popísané v kapitole 5.3.

Jednotlivé aplikácie simulujú len časť mechanizmu ARQ alebo komutácií paketov a preto bola vytvorená aplikácia realizujúca scenáre popísané v podkapitole 6. Tento program bol vytvorený v programovacom jazyku C# verzii 4.5 s použitím prostredia Microsoft Visual Studio Enterprise 2017. Novo vytvorená aplikácia umožňuje simulovať ARQ mechanizmy ako aj rôzne režimy prenosu dát. Tieto scenáre sú bližšie popísané v nasledujúcej kapitole.

6 Návrh simulačných scenárov mechanizmu ARQ a spôsobu prenosu informácií a komutácií

Pre simulovanie mechanizmu ARQ popísaného v podkapitole 4.1 a spôsobu prenosu informácií popísaného v kapitole 3, boli navrhnuté dva scenáre. Prvý z nich je scenár na simulovanie ARQ mechanizmu. Druhým je scenár zobrazujúci štyri druhy prenosu správ: komutácie okruhov, správ, paketov a buniek. Názov druhého scenára bude pre jednoduchosť označený ako scenár komutácií.

6.1 Scenár ARQ mechanizmu

Ako už bolo spomenuté, mechanizmus ARQ je realizovaný troma protokolmi. V tomto scenáre budú jednotlive protokoly podrobnejšie analyzované. Pri simulácií bude zvolený postupne každý z troch protokolov Stop-and-Wait, Go-back-N a Selective repeat. Zostavená aplikácia po spustení nastaví všetky potrebné hodnoty do takmer ideálneho stavu. Po spustení sa skontrolujú globálne nastavenia, spravidla či program generuje pcap súbor a následne je spustená simulácia jednotlivých protokolov. Priebeh je možné sledovať v simulačnej časti programu.

Prenos dát v sieti je ovplyvnený viacerými faktormi ako napríklad oneskorením, alebo stratovosťou. V prvej časti je simulovaný prenos dát v ideálnej sieti, z ktorého budú získané referenčné hodnoty. Tieto výsledky budú slúžiť na porovnanie medzi jednotlivými protokolmi. Pri simuláciách budú nastavované postupne rôzne hodnoty a odmeraná celková doba prenosu všetkých dát. Zmenou jednotlivých parametrov bude možné porovnať tri implementácie ARQ mechanizmu a určiť výhody a nevýhody.

V simulačnom scenári budú porovnané protokoly pri postupne zvyšujúcej sa „stratovosti“. Po ukončení každej simulácie sú jednotlivé hodnoty zaznamenané do tabuliek a zobrazené v grafoch. Všetky výsledky zo simulácií sú analyzované pomocou vygenerovaného pcap súboru.

6.2 Scenár komutácií

Simulačný scenár obsahuje štyri možnosti. Komutácia okruhov a správ je prezentovaná pomocou textovej časti na obrazovke. Pri obidvoch poliach sa následne zobrazí popis a užívateľ sa môže zoznámiť s príslušnou metódou pomocou animácie. Ostatné dva druhy komutácií pracujú pomocou na vyššie popísaného ARQ mechanizmu.

Na základe nastavených hodnôt klient môže odoslať v určitom časovom intervale viacero správ naraz.

Počas simulácie komutácií buniek a paketov bude prenesený rovnaký objem dát. Pri tomto scenári je klient a server zapojený v komplexnejšej sieti. Pri simulovaní je tak možné sledovať prenos správ cez jednotlivé uzly. Rozdiel a odlišné vlastnosti sú následne porovnané zvyšovaním objemu dát. Tieto dáta budú získané práve z rôzne veľkých textových súborov. Práve tento parameter simulácie ukáže rozdiely medzi prenosovými metódami. Prenesené dáta sú uložené do pcap súboru rovnako ako je to pri je pri scenári ARQ. Jednotlivé výsledky a typy budú porovnané a hodnoty zobrazené v grafoch.

7 Simulátor režimov prenosu dát v sieti

Simulátor predstavuje desktopovú aplikáciu, pomocou ktorej sú realizované dva scenáre. V prvom je možné simulovať prenos dát použitím ARQ protokolu medzi klientom a serverom. Druhý obsahuje komplexnejšiu topológiu, v ktorej prebieha výmena dát medzi stanicami a webovým serverom. Daný scenár zobrazuje spôsob prenosu dát pomocou komutácie správ, okruhov, paketov alebo buniek. Všetky správy sú prenášané vo virtuálnej sieti. Všetky správy sú po ukončení uložené a je možné dodatočne analyzovať. Dôležité časti programu sú bližšie popísané v nasledujúcich kapitolách a časti kódu sú následne zapísané v prílohách B.

Samotná aplikácia je zložená z dvoch základných častí, knižnice a grafického rozhrania. Jednotlivé sekcie a funkcie daných častí sú bližšie popísané v kapitole 7.3. Simulátor bol naprogramovaný použitím programovacieho jazyka C# a .NET frameworku verzie 4.5.

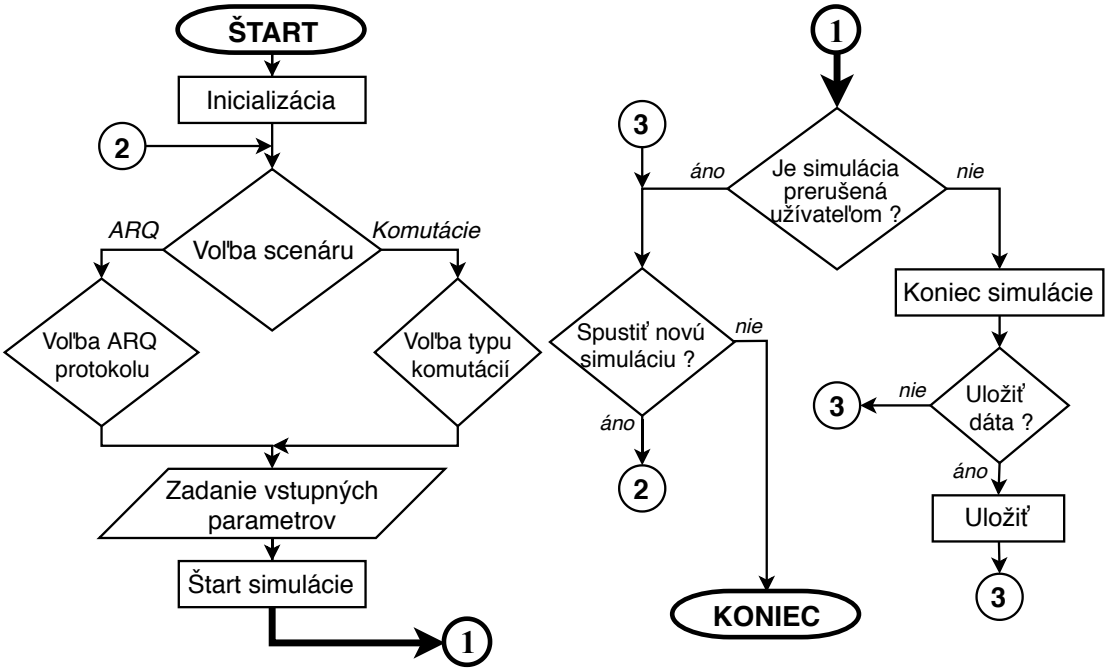
7.1 Programovací jazyk C#

Objektovo orientovaný programovací jazyk C# umožňuje vytvárať aplikácie spustiteľné na počítačoch s operačným systémom Windows. Jazyk C# ponúka viacero navrhových vzorov a programátorských konceptov, ako napríklad zapuzdrenie, dedičnosť alebo polymorfizmus. Všetky premenné a metódy sú zapuzdrené v rámci definície triedy. Tie tvoria predlohu k vytvoreniu jednotlivých objektov, ktoré následne alokujú pamäťové prostriedky počítača. Trieda môže dediť priamo len z jednej nadradenej triedy alebo implementuje niekoľko rozhraní. Rozloženie jednotlivých tried aplikácie na simulovanie prenosu dát bude následne popísané v kapitole 7.4. Syntax daného programovacieho jazyka je expresná a jednoduchá. Proces zostavenia programu C# je v porovnaní s jazykom C++ flexibilnejší a neexistuj tu hlavičkové súbory. Rovnako tak nie je nutné, aby metódy a typy boli deklarované v určitom poradí [12].

7.2 Vývojový diagram aplikácie

Samotný priebeh aplikácie od spustenia až po ukončenie nie je zložitý a je ho možné ilustrovať pomocou diagramu uvedeného na obr. 7.1.

Prvým krokom po spustení je inicializácia. Aplikácia upraví rozhranie na základe prednastavených hodnôt. Po tomto kroku sa zobrazí samotné grafické rozhranie aplikácie. Po inicializácií je spomenutá *Aplikačná časť* je pripravená na príkazy užívateľa.



Obr. 7.1: Vývojový diagram aplikácie

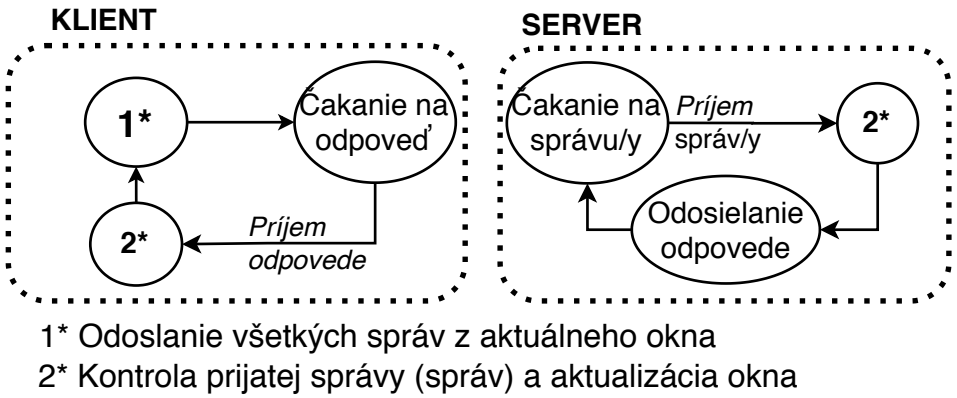
Do kroku inicializácie sa môže zaradiť aj skontrolovanie a prípadná zmena globálnych parametrov užívateľom. Nasledujúcim krokom je voľba samotného simulačného scenára. Po zvolení aplikácia zobrazí grafické rozhranie, ktoré zodpovedá príslušnej voľbe. Jednotlivé komponenty obsahujú základné prednastavené hodnoty, ktoré je možné zmeniť. Úpravou vstupných hodnôt je aplikácia pripravená.

Užívateľ spustí simuláciu tlačidlom *ŠTART*. Pretože priebeh samotnej simulácie nieje jednoduchý a záleží na viacerých faktoroch bude detailnejšie popísaný v nasledujúcej podkapitole 7.2.1. Program je možné kedykoľvek ukončiť tlačidlom *STOP*. Po danom kroku má užívateľ možnosť zmeniť vstupné parametre, alebo ukončiť aplikáciu. Druhá z možnosti je počkať na koniec simulácie. Ak bolo povolené generovanie *pcap* súboru v globálnych nastaveniach aplikácia automaticky uloží jednotlivé súbory. Vytvorenie týchto súborov a Korektné ukončenie simulácie umožní uložiť prenesené dáta do adresára. Po ich uložení je možné spustiť novú simuláciu alebo ukončiť aplikáciu.

Všetky prijaté a odoslané správy uložené v spomenutom *pcap* súbore sú generované pomocou open-source knižnice dostupnú z [2]. Jednotlivé súbory získajú príponu „pcap“ a je ich možné analyzovať použitím programu Wireshark uvedenom v kapitole 7.6 na zachytávanie správ. Druhou možnosťou je virtuálne okno simulovaného Wiresharku označeného v aplikácií tlačidlom **W**. Bližšie je dané okno popísané v podkapitole 7.5.

7.2.1 Popis priebehu simulácie

Priebeh simulácie je možné rozdeliť do nasledujúcich krokov: inicializácia, vytvorenie spojenia, prenos správ, a ukončenie spojenia. Aplikácia prejde do prvého kroku akonáhle užívateľ vydá pokyn tlačidlom *ŠTART*. Po tejto akcii sa program nachádza v prvej fáze, inicializácií. Týmto krokom program nastaví potrebné sieťové prvky, ktoré budú potrebné počas simulácie.



Obr. 7.2: Vývojový diagram činnosti klienta a serveru pri prenose dát

Po fáze inicializácie sieťové prvky naviažu spojenie a začína prenos samotných dát. Počas prenosu sa koncové prvky riadia schémou podľa obr. 7.2.

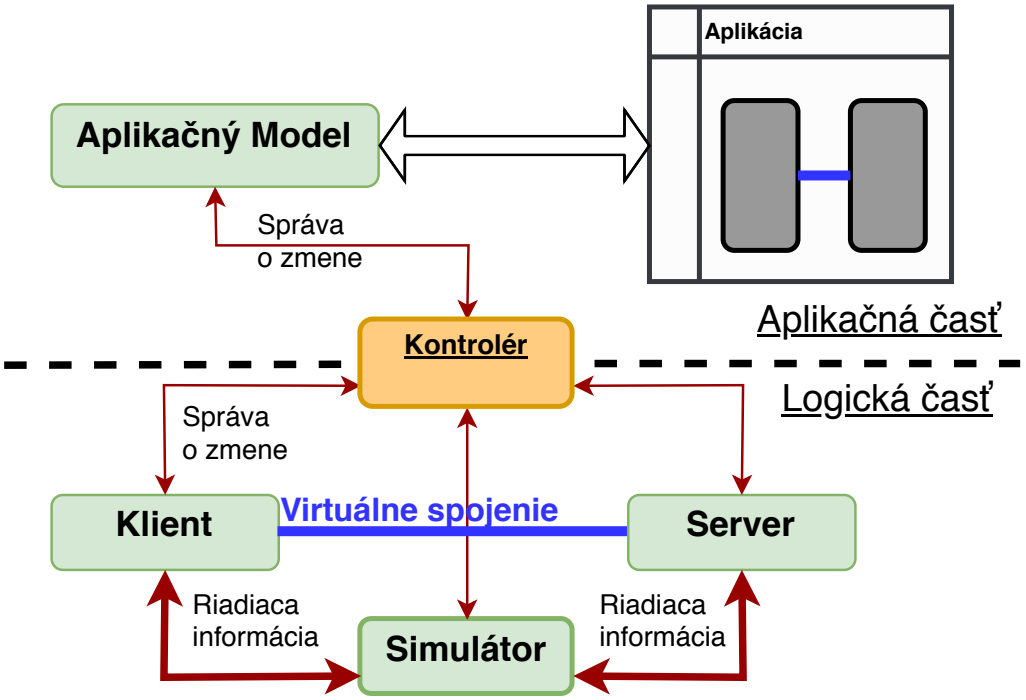
Klient znázornený v ľavej časti začína odosielať správy z aktuálneho. Po odoslaní príslušného počtu správ, ktoré záleží od nastavení užívateľa klient začína čakať na odpoveď. Ak by odpoveď neprišla v stanovenom čase klient považuje odoslané správy za stratené a opakuje ich prenos. Po prijatí odpovede vyhodnotí jej obsah a následne aktualizuje okno. Klient môže prijať bud potvrdenie alebo zamietnutie odoslaných správ. Podľa typu odpovede odošle buď nové správy z pamäte alebo ak boli predchádzajúce hodnoty zamietnuté opakuje ich prenos. Cyklus sa ukončí kladným potvrdením poslednej správy.

Server pri simulácií postupuje podobne. Po naviazaní spojenia prejde do stavu čakania. Tento dej preruší prvá prijatá správa od klienta. Následne server skontroluje sekvenčné číslo správy. Na základe daného čísla nasleduje kontrola, či prijatá správa je doručené v správnom poradí, alebo, či sa jedná o duplikát. Po kontrole server odosiela príslušnú odpoveď, ak je to možné a následne opäť čaká na príjem.

Ostatné sieťové prvky, umiestnené v simulačnom scenári komutácií sú nastavené statickým smerovaním. Podľa jednotlivých záznamov jednotlivé uzli smerujú doručenú správu ku serveru alebo ku klientovi.

7.3 Štruktúra simulačného programu

Simulátor má za úlohu simulovať komunikáciu v internete mechanizmom ARQ alebo komutáciou. Pre tento účel bol program rozdelený do dvoch samostatných modulov. Táto modularita uľahčila vývoj a ladenie chýb ako aj umožňuje rozšírenie o nové funkcie.



Obr. 7.3: Základná bloková schéma aplikácie

Na obr. 7.3 je znázornená základná bloková schéma objektov a ich vzťahov. Prvým modulom je časť grafického rozhrania *Aplikačná časť* a pod ním *Logická časť*. Moduly komunikujú na základe posielania správ, ktoré sa prenášajú cez blok *Kontrolér*. Bližšie popísaný proces prebieha analogicky aj pri simulačnom scenáre s viacerými uzlami.

V modeli, blok *Aplikačný model* zodpovedá za všetky hodnoty a dáta, ktoré sú vykresľované v grafickom rozhraní. Po vykonaní akcie užívateľom, blok oznámi zmenu *Kontroléru*. Správa je následne odoslaná do bloku *Simulátora* a odpovedajúcich objektov. Po uskutočnený požadovaných akcií *Logická časť* odosiela výsledok pomocou správy cez *Kontrolér* až do grafického rozhrania.

7.3.1 Aplikačná časť

Aplikačná časť, modul znázornený v hornej časti obr. 7.3, je zodpovedný za získanie nastavení potrebných na simuláciu a následné zobrazenie jej priebehu. Fyzicky je tento komponent programu reprezentovaný súborom „NetworkSimulator.exe“. Modul obsahuje dve základné časti a to grafické rozhranie a dáta.

Blok *Aplikačný Model* je reprezentovaný súborom tried, ktoré sprístupňujú vlastnosti grafického rozhrania. Úlohou daného modelu je spravovať vstupné hodnoty zadané užívateľom. Pri spustení sú nastavené základné ideálne hodnoty pre správny priebeh simulácie. Samotný blok je zložený z viacerých objektov, ktoré kontrolujú grafické rozloženie aplikácie tiež zobrazenie priebehu a výsledkov samotnej simulácie.

7.3.2 Logická časť

Logická časť predstavuje samotné jadro programu. Funkciou tejto časti je vykonať potrebné operácie k správnemu priebehu simulácie. Výstup logickej časti je následne odoslaný pomocou správy do aplikačného modelu. Logická časť je po spustení programu inicializovaná a jej hodnoty sa môžu počas priebehu programu zmeniť na základe správ z aplikačnej časti.

Logická časť je zložená z troch základných blokov: simulátor, klient a server. Hlavný blok *Simulátor* obsahuje všetky parametre simulácie, ako sú napríklad počet správ, ktoré budú odoslané, ale aj samotný typ zvoleného scenáru. Na základe zadaných parametrov blok rozhodne o počte uzlov v simulácií, použitej technológií pri prenose dát a podobne. Spustením simulácie spomenutý blok zadá príkaz jednotlivým uzlom k prenosu správ. Komunikácia prebieha na základe výmeny riadiacej informácie medzi uzlami a *Simulátorom*.

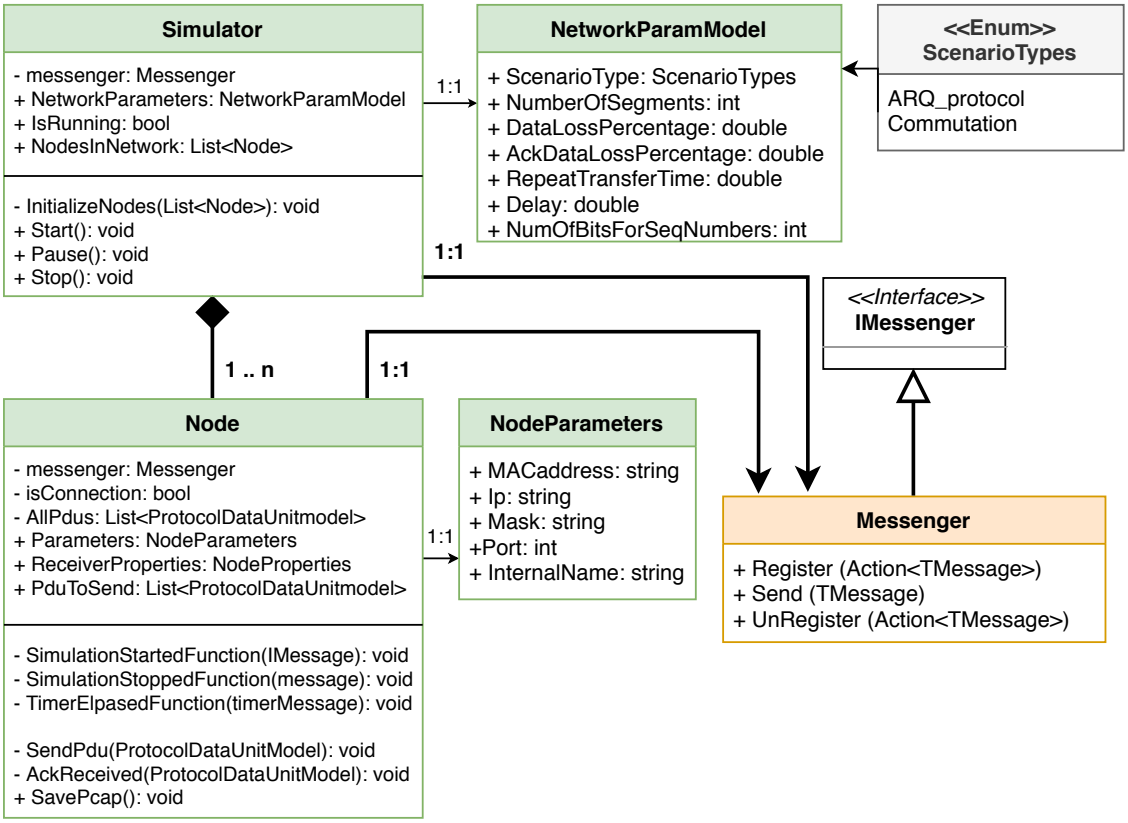
Jednotlivé uzly môžeme rozdeliť na dva typy: klient a server. Každý z uzlov je definovaný jedinečnou MAC adresou, sieťovou IP a prípadne portom. Po spustení samotnej simulácie spomenuté uzly komunikujú nezávisle od *Simulátora*.

7.4 Popis tried aplikácie

Na zostavenie programu bol zvolený objektovo orientovaný jazyk. Jednotlivé objekty sú vytvorené zo vzorov - tried. Každá z nich môže obsahovať časť atribútov a metód. Na zostavenie aplikácie boli vytvorené desiatky tried. V nasledujúcich častiach budú spomenuté len niektoré z nich, ktoré tvoria pomyslenú kostru programu a sú pre funkciu programu nevyhnutné. Jednotlivé triedy sú spolu prepojené na základe vzťahov, ako je to znázornené na obr. 7.4 pre logickú časť a pre aplikačnú na obr. 7.5 . Diagram obsahuje len vybrané atribúty a dôležité metódy. Podrobnejšie sú jednotlivé triedy zaznamenané v prílohách B.

7.4.1 Triedy v logickej časti

Logická časť obsahuje funkcie na výpočty a generuje dáta. Pred popisom jednotlivých tried je nutné objasniť prepojujúci prvok nazývaný aj *Kontrolér*. Tento element predstavuje jednu triedu „Messenger“, implementujúcu rozhranie „IMessenger“, uvedené v prílohe B.1. Pomocou danej triedy je pri inicializácií aplikácie vytvorený objekt, využitý ostatnými objektmi programu. *Kontrolér* následne obsahuje tri metódy: registráciu na určitú správu(„Register“), odoslanie („Send“) a odregistrovanie („UnRegister“). Funkcia je nasledujúca, jednotlivé objekty aplikácie sa registrujú na príjem určitej správy a definujú akciu, ktorá nastane po prijatí danej správy.



Obr. 7.4: Diagram významných tried logickej časti programu

Hlavnou triedou je *Simulator*. Slúži ako takzvaný ovládač logickej časti. Jeho hlavnými funkciami sú inicializovať potrebné uzly, spustiť, zastaviť simuláciu alebo odoslať správu o jej korektnom ukončení. Prvým z atribútov tejto triedy je kontrolér („messenger“) pomocou, ktorého trieda prijíma a odosiela správy. Nasledujúcim parameterom je „NetworkParametersModel“ obsahujúci informácie o súčasnej simulovanej sieti. Ďalší atribút udáva stav simulácie a v poslednom sú uložené všetky aktuálne aktívne uzly. Trieda pri spustení simulácie príjme správu od kontroléra

prebehne inicializácia uzlov („InitializeNodes“) štart („Start“). Ukončenie simulácie môže nastať na pokyn užívateľa alebo po úspešnom odoslaní potrebného počtu správ. Ak užívateľ ukončí aplikáciu *STOP* tlačidlom zavolá sa metóda Stop priamo. V opačnom prípade jednotlivé uzly informujú simulátor o ukončení práce a ten vykoná korektné ukončenie.

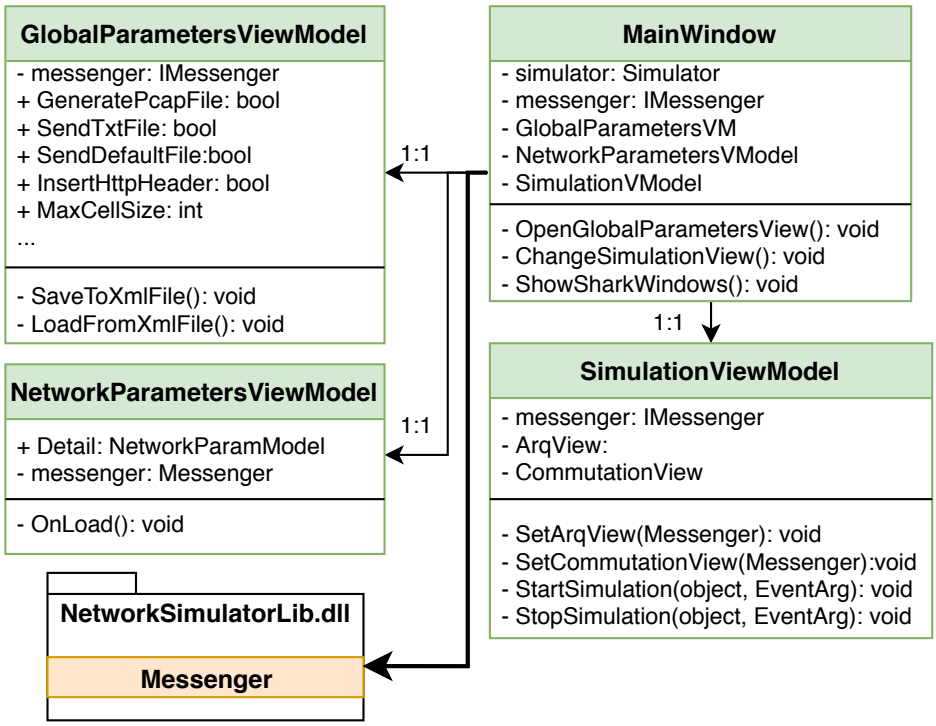
Rovnako ako trieda *Simulator* aj každý uzol implementujúci rozhranie („INode“) obsahuje kontrolér
(„messenger“) a sadu parametrov („NodeParametersModel“) (príloha B.2). Tieto parametre definujú sieťové vlastnosti uzla, ako je napríklad sieťová IP adresa alebo port a podobne. Každý uzol obsahuje vnútornú pamäť odoslaných a prijatých správ v liste a rovnako tak zoznam všetkých správ, ktoré má poslať. Prijatím signálu od simulátora trieda môže začať prenos správ. Jednotlivé správy sú v programe reprezentované komplexnov dátovou jednotkou „ProtocolDataunitModel“, ktorej štruktúra je zobrazená v prílohe B.3. Táto trieda obsahuje premenné, pre zostavenie kompletnej dátovej jednotky, ako napríklad zdrojovú, cieľovú MAC adresu,IP adresu, čísla portov a podobne. Každá táto jednotka je následne odoslaná pomocou nového vlákna, ktoré následne riadi procesor. Odoslaním veľkého objemu správ v krátkom časovom úseku (rádovo niekoľko milisekúnd), môže procesor spracovať vlákna v inom poradí ako boli vytvorené. To spôsobí, že príjemca prijme mimo poradia v akom boli odoslané, čo sa môže prejaviť pri použití GBN protokolu. Po ukončení simulácie sa všetky objekty vytvorené z danej triedy („ProtocolDataUnitModel“) naformátujú a uložia do súboru pcap, ak to užívateľ povolil.

7.4.2 Triedy aplikačnej časti

Aplikačná časť je tvorená jedným základným oknom *MainWindow*, ktoré obsahuje menšie jednotky nazývané „View“. Každá taká jednotka je kontrolovaná jednou triedou „ViewModel“.

Každá trieda bola inicializovaná pomocou takzvaného lokátora (*ViewModelLocator*) a odpovedá za jednu časť aplikácie. Každá z nich obsahuje spojenie na spomenutý kontrolér „messenger“. Týmto spojením môže jednotka grafického okna okamžite upozorniť logickú časť, napríklad aj o zmene vstupných údajov pri zadávaní užívateľom. Komunikácia medzi jednotkami a logickou časťou prebieha obojsmerne. Obsah jednotky sa mení nielen pri zásahu užívateľa, ale rovnako tak i pri prijatí správy z logickej časti (obr. 7.4).

Prvá trieda „GlobalParameterViewModel“ je určená k nastavovaniu globálnych nastavení, ako napríklad možnosť generovať pcap súbor, odoslať texotvý súbor, alebo vložiť HTTP hlavičku. Pre špecifické nastavenia jednotlivých simulačných scenárov bola vytvorená trieda „NetworkParametersViewModel“. Jej obsahom sú všetky



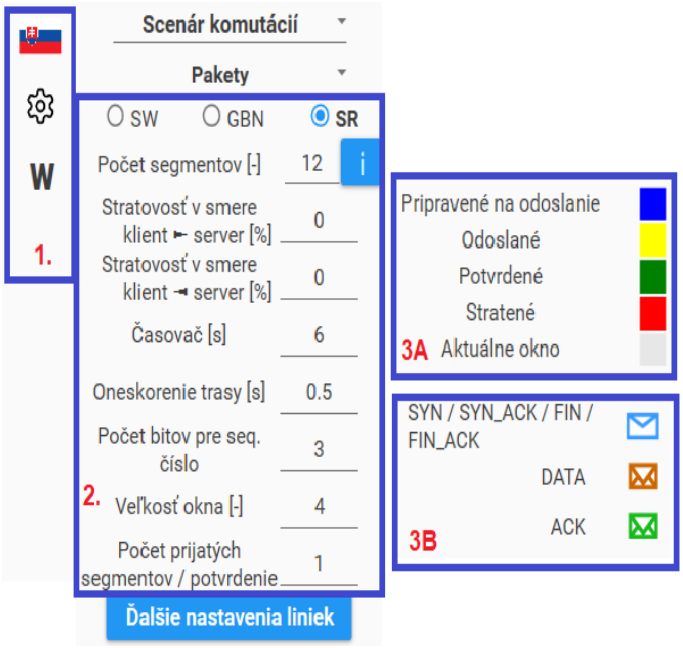
Obr. 7.5: UML Diagram dôležitých tried aplikačnej časti programu

vstupné hodnoty získané od užívateľa, ktoré sú následne použité pri simulácií. Danými hodnotami sú napríklad počet správ na odoslanie, typ simulačného scenára alebo zobrazovanie. Podrobnejšie sú tieto vstupy popísane v podkapitole 7.5.2. Najväčšiu časť grafického rozhrania tvorí jednotka slúžiaca k zobrazeniu priebehu simulácie a kontroluje ju trieda „SimulationViewModel“. Simulačná časť sa neskôr rozdeľuje na rozhranie pre zobrazenie ARQ scenára („ArqView“) a následne rozhranie pre režim komutácií („CommutationView“). Trieda „SimulationViewModel“ má za úlohu aj zahájiť simuláciu na pokyn užívateľa alebo ju prerušiť tlačidlami štart a stop.

7.5 Grafické rozhranie

Rozhranie aplikácie predstavuje súbor viacerých objektov s vlastnosťami a priradenými akciami. Po zvolení simulačného scenára sa môžu jednotlivé objekty premiestniť alebo sa nahradia novými. Základné grafické rozhranie pre simuláciu ARQ mechanizmu je následne znázornené v prílohe na obr. A.1. Pre úpravu a grafický dizajn bola použitá open-source knižnica dostupná z [20]. Pri druhom simulačnom scenári sú implementované dva druhy rozloženia. Prvý obsahuje textovú časť s animáciou obr. A.2, a druhý je zložený z sieťových prvkov v komplexnej sieti obr. A.3.

Jednotlivé rozhrania sú bližšie popísané v nasledujúcich častiach.



Obr. 7.6: Grafické rozhranie, 1. menu aplikácie, 2. nastavenia ARQ protokolov, 3A. legenda scenára ARQ, 3B. legenda scenára komutácii

Z zobrazeného rozloženia je možné vidieť jasné oddelenie nastavení a simulačnej plochy. Rozloženie vstupných parametrov pre GBN protokol a následne pre simulačný režim komutácií, podobný nastaveniam scenára ARQ, znázornený na obr. 7.6 vpravo. Umiestnenie komponentov je zvolené na základe poradia operácií. Po spustení programu je nutné zadať vstupné hodnoty až potom spustiť simuláciu.

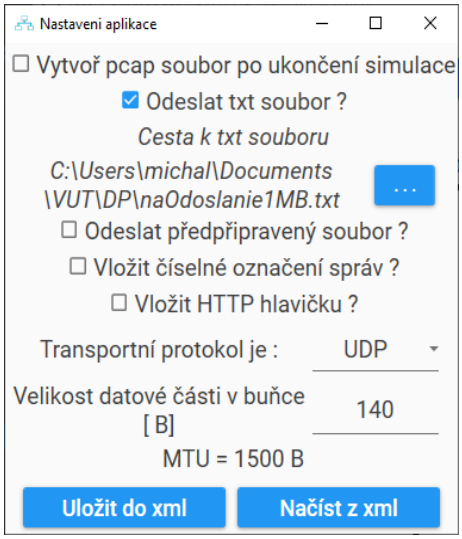
Prvá skupina objektov, označená číslom 1. na obr. 7.6 umožňuje nastavovať globálne parametre aplikácie 7.5.1. Pod voľbou nastavení tlačidlo zobrazí „Wireshark“ okna sieťových prvkov, popísaný v podkapitole 7.7.

Časť číslo 2 obsahuje nastavenia pre príslušné scenáre popísané v nasledujúcich kapitolách. Nad touto sekciou je možné zvoliť jeden z dvoch simulačných scenárov a prípadne typ komutácií. Posledné bloky 3A a 3B obsahujú legendy. Pri prenose sa jednotlivé správy označujú príslušnými farbami alebo obálkami. Napríklad správy, ktoré sú pripravené k odoslaniu, v scenári ARQ, majú modrú farbu počas prenosu žltú a ak sú potvrdené zmení sa ich označenie na zelenú. Červená farba označuje správy, ktoré budú stratené počas prenosu a posledná sivá označuje správy nachádzajúce sa v aktuálnom okne.

7.5.1 Globálne nastavenia aplikácie

Prvým z nastavení je voľba jazyka aplikácie. Po spustení programu je v ľavej časti pod názvom aplikácie vlajka štátu, ktorého jazyk bude použitý pre preklad textu. Zmenu jazyka je možné docieliť kliknutím na vlajku príslušného štátu. Aplikácia podporuje celkom 3 jazyky: češtinu, slovenčinu a angličtinu.

Pod voľbou jazyka sa nachádza v poradí druhé tlačidlo, slúžiace na definovanie prídavných funkcií. Kliknutím na ikonku pod vlajkou sa zobrazí zmenšené okno znázornené na nasledujúcom obr. 7.7.



Obr. 7.7: Globálne nastavenia aplikácie

Prvým z nastavení je možnosť povoliť alebo zakázať sieťovým prvkom aplikácie generovanie súboru s prenesenými správami. Pod daným tlačidlom užívateľ môže vybrať adresár do ktorého aplikácia uloží jednotlivé súbory. Ak je povolená predchádzajúca voľba, je možné nastaviť automaticé otvorenie vytvorených súborov v programe Wireshark. Spustenie tejto funkcie je možné iba ak bola zadaná cesta k spustiteľnému súboru „wireshark.exe“.

Nasledujúcim nastavením je vyber či sa bude počas simulácie prenášať súbor alebo nie. Dokument bude následne rozdelený na menšie časti a jeho obsah sa prenesie vygenerovanými správami počas simulácie. Ak je táto možnosť zvolená, v nasledujúcom kroku je nutné vybrať cestu k danému súboru, alebo zvoliť odoslanie predpripraveného súboru. Po týchto nastaveniach užívateľ môže povoliť alebo zakázať vkladanie informačného číselného označenia napríklad pre piatu správu z desiatich „[5/10]“. Ďalším nastavením je možné povoliť alebo zakázať vloženie HTTP hlavičky pre simulovanie použitia aplikačného protokolu.

Z pohľadu transportnej vrstvy aplikácia podporuje prenos protokolom ako UDP tak i TCP. Ak je zvolený ako transportný protokol TCP zobrazí sa možnosť „Zobraziť TCP spojenie“, ktorá ukáže pri simuláciách naviazanie a ukončenie spojenia.

Posledná možnosť je aplikovateľná pri komutácií buniek, kde je možné podľa [7] nastaviť veľkosť dátovej časti bunky s veľkosťou od 8 B do 1465 B pri UDP. Túto hodnotu je možné získať po odčítaní IP hlavičky (minimálne 20 B podľa [7]), UDP hlavičky (8 B podľa [6]) a označeným správy aplikáciou (minimálne 7 B pri jednocifernom číslovaní „[SEQ:7]“) od maximálnej veľkosti prenosovej jednotky nastavenej v programe na hodnotu 1500 B podľa [7]. Pri transportnom protokole TCP s veľkosťou hlavičky 20 B podľa [8] je maximálna možná hodnota vypočítaná ako 1500 B − 20 B − 20 B na hodnotu 1460 B. Pri protokole následne nedochádza dodatočnému označeniu správ ako to bolo pri UDP. Jednotlivé čísla správ a potvrdení sú vložené v samotnej hlavičke transportného protokolu.

Všetky spomenuté nastavenia je možné uložiť do súboru „xml“ tlačidlom „Uložiť do xml“. Aplikácia po následnom spustení načíta tento súbor, ak sa nachádzajú v rovnakom adresári ako spustiteľný súbor. V prípade ak nieje možné načítať nastavenia bude aplikácia nastavená do základných nastavení. Druhé zobrazené tlačidlo „Načítať z xml“ umožnuje načítať tento súbor manuálne z zvoleného adresára.

7.5.2 Nastavenia simulácie

Priebeh simulácie a jej výsledky sú ovplyvnené zadanými parametrami od užívateľa. Tieto hodnoty je možné vkladať pomocou zobrazených objektov obr. 7.6 časť označená číslom 2. Počet jednotlivých vstupov je rovnaký pri obidvoch simulačných scenároch. Jediný rozdiel medzi rozložením nastavení je pri simulačnom scenári komutácií. Po zvolení daného scenára sa zobrazí v novom riadku ponuka so zoznamom všetkých typov. Pri zvolení komutácií *správ* a *okruhov* budú všetky ostatné nastavenia skryté. Pre fungovanie aplikácie v týchto režimoch už nebudú potrebné. Ostatné dve možnosti, komutácie paketov a buniek, zobrazia rovnaké nastavenia ako je to pri simulačnom scenári ARQ.

Ako už bolo spomenuté v obidvoch scenároch je možné nastaviť rovnaké parametre. Prvou z možností je voľba jedného z troch ARQ protokolov: SW (Stop and wait), GBN (Go-back-N) alebo SR (Selective repeat). Pre prehľadnosť a čitateľnosť jednotlivé bloky obsahujú krátky popis, ktorý je doplnený nápovedou. Zobrazenie takého to textu je možné presunutím kurzora nad objekt. Napríklad pri tlačidle „SW“ bude užívateľovi zobrazený text „Stop and wait ARQ protokol“. Zvolením protokolu je možné zadať počet segmentov, ktoré budú odoslané počas simulácie. Ak bol zvolený súbor na odoslanie, tlačidlom napravo **i**, aplikácia na základe zadaných parametrov odporučí potrebný počet segmentov na prenos celého súboru.

V poradí tretie nastavenie, stratovosť správ odoslaných v smere od klienta k servera. Rovnakú funkciu má aj štvrtý blok upravujúci stratovosť v opačnom smere. Po odoslaní dátovej jednotky klient nastavý časovač opakovaného prenosu, ktorého veľkosť je možné nastaviť v nasledujúcom okne. Posledným zo série nastavení, ktoré sú spoločné pre všetky tri ARQ protokoly je oneskorenie trasy. Hodnota udáva čas nutný na prenos dát v jednom smere medzi dvoma sieťovými bodmi. V prípade simulačného scenára ARQ je možné túto hodnotu použiť na zrýchlenie alebo spomalenie simulácie a tak lepšie kontrolovať jej priebeh. Zmenou danej hodnoty je možné ukázať zmenu vplyv oneskorenia na celkovú dobu prenosu všetkých dát. Pri prvom simulačnom scenári je možné pomocou tejto hodnoty upravovať rýchlosť prenosu správ a tak spomaliť alebo zrýchliť simuláciu.

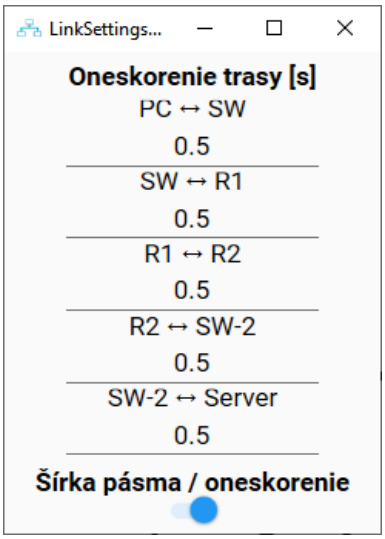
Posledné tri parametre sa aplikujú iba pri simulovaní protokolu GBN a SR. Parametre sú nastavene na hodnotu 1 pri simulovaní protokolu SW. Počet bitov pre sekvenčné číslo udáva celkový počet bitov, ktoré budú alokované pre veľkosť sekvenčného čísla. Nasledujúcimi hodnotami sú veľkosť okna a počet správ, ktoré budú odoslané aj bez získaného potvrdenia. Táto hodnota udáva počet správ, ktoré musia byť prijaté od klienta, aby server odoslal potvrdenie o ich doručení.

Pri prenose správ je použitý transportný protokol TCP, ktorý môže na prenos použiť protokol vyššej vrstvy a to HTTP protokol. Hlavičku daného protokolu zabalí spolu s dátami do segmentu. Pri prenose je nutné myslieť na veľkosť tejto hlavičky. Pri simulovaní komutácií paketov je jej veľkosť relatívne malá v porovnaní s veľkosťou dátovej časti TCP segmentu preto je možné nechať danú možnosť označenú. Problém môže nastať pri komutáciách buniek, kde je veľkosť dátovej časti obmedzená na 120 bajtov. V takom prípade je možné zrušiť vkladanie HTTP hlavičky a v bunke sa následne prenesie väčšie množstvo dát. Posledným tlačidlom je možnosť zobraziť zahájenie a ukončenie TCP spojenia.

Pri simulačnom scenári komutácií pracuje aplikácia s komplexnejšou sieťou. Všetky správy sú medzi jednotlivými uzlami odoslané s rovnakým oneskorením. Túto možnosť je možné zmeniť v okne na obr. 7.8 tlačidlom *Ďalšie nastavenia liniek*. V dolnej strane je možné následne voliť medzi úpravou oneskorenia alebo šírky pásma príslušnej trasy.

7.5.3 Simulačná časť aplikácie

Za nastaveniami parametrov sa nachádza druhá časť programu, simulačná časť. Sekcia slúži na zobrazenie priebehu simulácie. Rozloženie jednotlivých komponentov sa upravuje na základe zvolených parametrov. Oblasť obsahuje jediné dve tlačidla, ktoré sú stále viditeľné, *ŠTART* a *STOP*. Druhé tlačidlo je vždy neaktívne a povolí sa až po spustení simulácie. Každá simulácia sa po samotnom priebehu automaticky

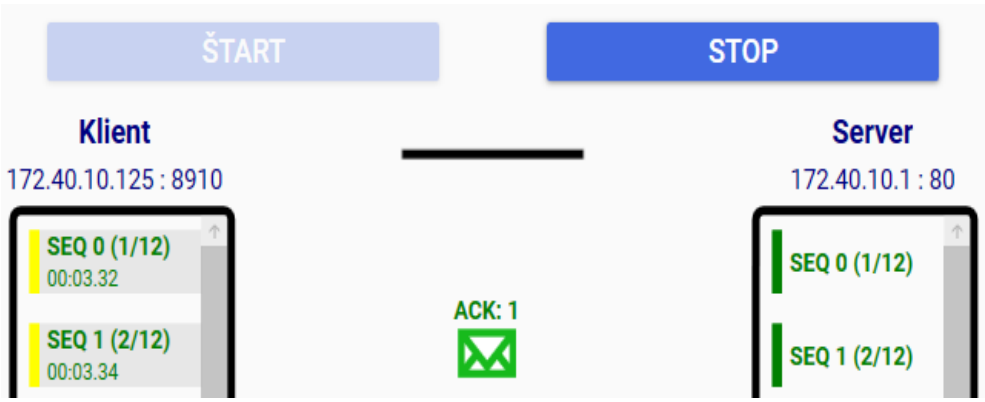


Obr. 7.8: Zmena nastavení jednosmerného oneskorenia medzi jednotlivými uzlami

ukončí, preto dané tlačidlo slúži len na predčasné ukončenie simulácie. Rozloženie plochy sa automaticky upravuje na základe zadaných hodnôt a zvoleného scenára. V nasledujúcej časti sú popísané jednotlivé rozhrania a ich funkcie.

Simulačná časť scenára ARQ

Rozloženie plochy pri simulovaní ARQ protokolu je znázornené na obr. 7.9. Plocha je rozdelená na dve samostatné sekcie. Prvá vpravo znázorňuje časť klienta na ľavej strane s príslušnou IP adresou a druhá prestavuje časť serveru. Stlačením tlačidla *ŠTART*, program zobrazí príslušný počet správ a označí ich modrou farbou.



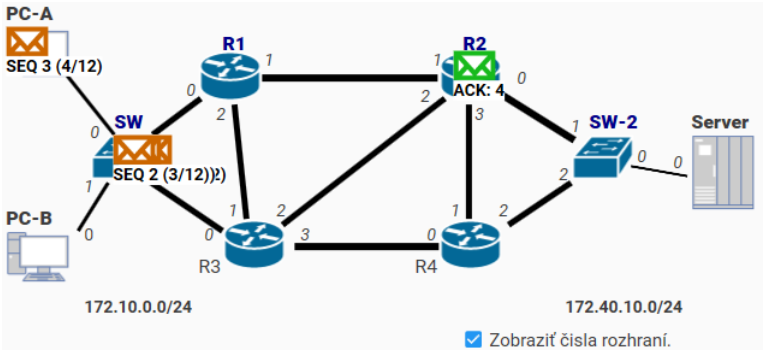
Obr. 7.9: Časť grafického rozhrania pre simulovanie ARQ scenára

Každá zo správ je označená sekvenčným číslom a popisným číslom, uvedeným v zátvorke. Prvé číslo udáva poradie aktuálnej správy a druhé celkový počet správ

v simulácií. Správy pripravené na odoslanie sú označované modrou farbou, následne počas prenosu žltou a po potvrdení alebo doručení zelenou obr. 7.9. Pri označení správy v zozname klienta dôjde k jej zmene na červenú a správa bude počas prenosu zahodená. Jednotlivé správy môžu byť prenesené iba ak sa nachádzajú v dočasnej pamäti tzv. okne. Toto okno je vyznačené sivou farbou. Jednotlivé farby sú rovnako tak uvedené aj v legende zobrazenej na obr. 7.6 časť 3. Posledným dôležitým prvkom správ je časovač opakovaného prenosu. Hodnota je zobrazená v minútach, sekundách a stotinách oddelených dvojbodkou.

Simulačná časť scenára komutácií

Pri voľbe druhého scenára aplikácia zobrazuje jedno z nasledujúcich rozložení. Zvolením komutácií okruhov užívateľ uvidí vysvetlenie a jednotlivé kroky, ktoré charakterizujú danú technológiu. Pod popisom je možné vidieť topológiu siete a dve tlačidla. V prílohe obr. A.4 je znázornené dané rozloženie, počas spustenej animácie. Pri priebehu sa užívateľovi automaticky zobrazuje popis jednotlivých krokov. Rovnaké rozloženie komponentov a topológia bola použitá aj pri komutácií správ obr. A.5.



Obr. 7.10: Simulačná časť pre scenár komutácií paketov a buniek počas prenosu správ a 1 potvrdenia

Komutácia paketov a buniek ukazuje nasledujúcu sieť obr. 7.10. V simuláciách je následne zahájený prenos od klientskej stanice na ľavej strane cez smerovače až k serveru vpravo. Všetky sieťové prvky sú spojené virtuálnymi spojeniami cez očíslované rozhrania. Označenie týchto rozhraní je možné kedykoľvek skryť odznačením tlačidla pod topológiou. Zobrazené čísla rozhraní slúžia pre ľahšiu orientáciu pri otvorených Wireshark oknách, ktoré sú bližšie popísané v nasledujúcej podkapitole 7.7. Po spustení simulácie klient generuje správy označené sekvenčnými číslami napríklad **SEQ 3 (4/12)** a server odosiela zelené potvrdenia **ACK: 4**.

7.6 Program Wireshark

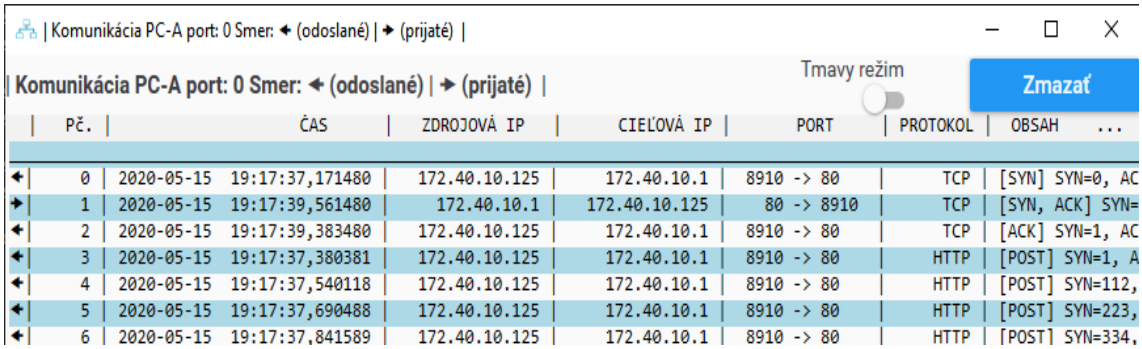
Aplikácia Wireshark je nástroj na zachytávanie sitovej komunikácie. Program môže jednotlivé správy zaznamenávať zo zvoleného sieťového rozhrania alebo zobraziť uložené hodnoty zo *pcap* súboru. Jednotlivé dátové jednotky sú zobrazené do tabuľky.

V prvom stĺpci sa nachádza poradové číslo, za ním nasleduje čas kedy bol segment zachytený. Pre rýchlu analýzu účastníkov slúži zdrojová a cieľová IP adresa, ktoré sú zobrazené v stĺpcoch *Zdroj* a *Cieľ*. Nasleduje stĺpec *Protokol*, ktorý označuje použitý protokol najvyššej vrstvy dátového modelu ISO/OSI. V danom prípade sa jedná o TCP protokol. Stĺpec *Dĺžka* udáva celkový počet prenesených bitov a posledná hodnota *Info* zobrazuje stručný popis preneseného paketu.

7.7 Virtuálny Wireshark v aplikácii

Virtuálny Wireshark bol navrhnutý, ako náhrada funkcií a možnosti originálneho programu Wireshark. Cieľom daného okna je jednoduché a prehľadné zobrazovanie prenesených správ počas simulácie v reálnom čase. Pomocou daného nástroja užívateľ nemusí filtrovať komunikáciu rovnako tak ani inštalovať nový externý program.

Na obr. 7.11 je znázornené okno virtuálneho Wiresharku, ktoré obsahuje prehľadnú tabuľku s dôležitými údajmi dátovými jednotkami. V každom novom okne sa nachádza názov zariadenia a jeho port, na ktorom sú jednotlivé správy zachytávané do tabuľky.



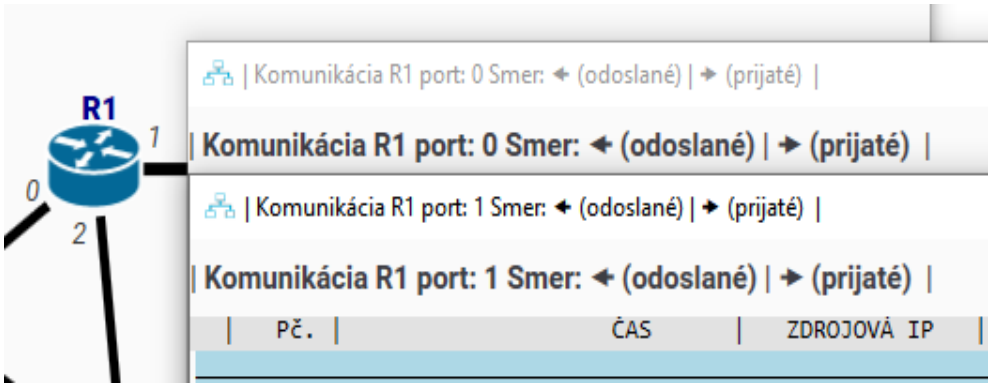
Obr. 7.11: Zachytená komunikácia vo virtuálnom programe Wireshark

Samotná tabuľka obsahuje celkom sedem stĺpcov. Prvý obsahuje poradové číslo paketu podľa poradia v akom bol uložený v tabuľke. Za touto hodnotou nasleduje čas. Stĺpec zobrazuje presný čas prijatia alebo odoslania dátovej jednotky s rozlíšením na milisekundy. Tretí a štvrtý údaj obsahuje zdrojovú a cieľovú adresu paketu. V nasledujúcom stĺpci sú zobrazené čísla portov. Zdrojový port je zaznamenaný na

ľavej strane šipky a následne na pravej je cieľový. Za týmto údajom sa nachádza stĺpec zobrazujúci protokol najvyššej vrstvy ISO/OSI modelu, ktorý je prenesený. Názov posledného stĺpca je *Obsah*. V stĺpci sa nachádzajú ostatné položky TCP hlavičky ako aj prenesené dáta. Všetky zaznamenané hodnoty tabuľky je možné kedykoľvek v prípade potreby vymazať tlačidlom nad tabuľkou.

7.7.1 Implementácia virtuálneho Wireshark programu

Nové okno virtuálneho Wiresharku je možné otvoriť dvoma spôsobmi. Prvým je tlačidlo **W** v hlavnom menu programu. Pred zobrazením jednotlivých okien je užívateľ vyzvaný či chce otvoriť okna všetkých sieťových uzlov alebo nie. Výberom možnosti nie sa zobrazí iba okno klienta a serveru.



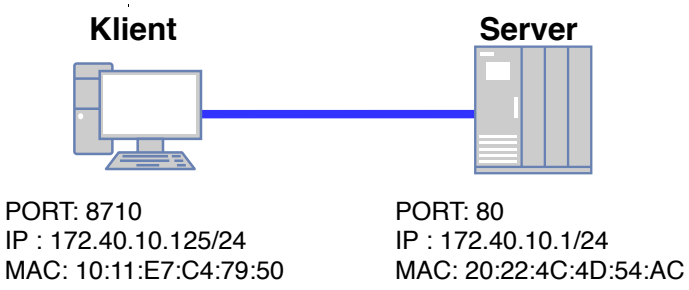
Obr. 7.12: Zobrazenie virtuálnych okien Wireshark pre port 0 a 1 rutra R1

Druhou možnosťou, ako zobraziť jednotlivé okná, je dvojklik na vybraný sieťový prvok. Každý uzol následne zobrazí jednotlivé okná pre príslušné aktívne rozhrania (obr. 7.12). Ak sú jednotlivé okná aktívne ukladajú zachytené správy do jednotlivých riadkov. Zobrazené správy je možné vymazať tlačidlom *Zmazať* (obr. 7.11).

Každé okno predstavuje samostatnú jednotku, ktorá prijíma správy od sieťových uzlov. Jednotlivé okná pri inicializácií dostanú globálny identifikátor a meno uzla („ovnerHostname“), pre ktoré budú zaznamenávať správy. Spolu s danými hodnotami je nutné uložiť rovnako tak aj číslo rozhrania, ktorého správy sú zaznamenávané („InterfaceId“). Počas simulácie jednotlivé uzly pred prijatím alebo odoslaním správy odosielajú informačnú správu do *Kontroléru* (obr. 7.3). Informačná správa je ďalej odoslaná jednotlivým oknám. Každý sieťový uzol spolu s dátovou jednotkou zabalí do tejto správy aj svoj globálny identifikátor. Každé okno po prijatí správy skontroluje globálny identifikátor. Ak sa číslo zhoduje z uloženou hodnotou, uloží príslušný segment do tabuľky.

8 Scenár ARQ protokolu v simulačnom programe

Simulačný program obsahuje dva simulačné scenáre. V tejto kapitole sa nachádza popis prvého scenára ARQ protokol. Jednotlivé funkcie a možnosti programu budú popísané vo forme úloh spolu s podrobným pracovným postupom. Pri scenári je použité dvojbodové zapojenie klienta so serverom, znázornené na obr. 8.1. Počas prenosu majú tieto stanice nastavené statické IP adresy, čísla portov a MAC adresy zobrazené na obr. 8.1.



Obr. 8.1: Topológia klient-server pri simulácií ARQ protokolu

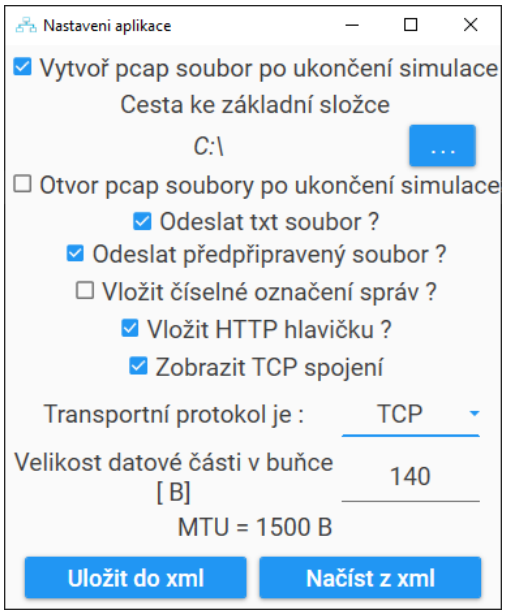
8.1 Úlohy v simulačnom scenári

Po zvolení simulačného scenára je možné simulovať a testovať rôzne nastavenia a možnosti jednotlivých ARQ protokolov. V tejto časti budú popísané jednotlivé úlohy a ich postup s cieľom bližšie ukázať možnosti daného protokolu, jeho výhody a nevýhody. Pri nasledujúcich úlohách bude použitý vytvorený program spolu s ná

8.1.1 Základné nastavenia aplikácie

Pri plnení jednotlivých úloh je nutné mať nainštalovaný vytvorený program a externý nástroj na zachytávanie komunikácie napríklad Wireshark. Pri nasledujúcich úlohách bola použitá verzia 3.2.0. Nastavenia simulácie v jednotlivých úlohách nezodpovedajú reálnym nastaveniam a vlastnostiam prenosových sieti. Tieto hodnoty boli zvolené za účelom jasného zobrazenia rozdielov a vlastností ARQ mechanizmu v interaktývnej simulácií. Pri nasledujúcich úlohách bude použitý ako transportný protokol TCP, ktorý následne prenesie dáta s HTTP hlavičkou.

Pred spustením akejkoľvek simulácie je vhodné skontrolovať globálne nastavenia. To je možné po kliknutí na ozubené koliesko pod vlajkou.



Obr. 8.2: Globálne nastavenia pre simulačný scenár ARQ protokolu

V otvorenom okne obr. 8.2 je vhodné nechať označenú prvú možnosť a vytvoriť pcap súbor po ukončení simulácie. Následne je zobrazená cesta k adresáru, kam sa budú generované súbory. Kliknutím na modré tlačidlo s troma bodkami sa zobrazí dialógové okno v ktorom je možné túto cestu zadať. Po vybratí súboru je vhodné overiť, či má prihlásený užívateľ právo zápisu na dané miesto. Väčšina operačných systémov povoľuje užívateľom vytvárať súbory na pracovnú plochu, preto bude zvolené práve toto miesto. Ďalšiu položku nie je nutné označovať.

Po týchto nastaveniach je možné zvoliť odosielanie textového súboru. Dané nastavenie bude v priebehu jednotlivých úloh zmenené podľa zadania, preto ho nie je nutné meniť. Posledná možnosť je uloženie nastavených parametrov do súboru. Kliknutím na tlačidlo *Uložiť do xml* sa zmenené hodnoty uložia a pri opätovnom spustení budú automaticky načítané. Pre fungovanie aplikácie to nie je nevyhnutné, pretože všetky nastavenia sa aplikujú okamžite po ich zmenení. Po úprave všetkých nastavení by malo zobrazené okno vyzerať ako na obr. 8.2.

8.1.2 Simulovanie prenosu dát v bezchybnej sieti

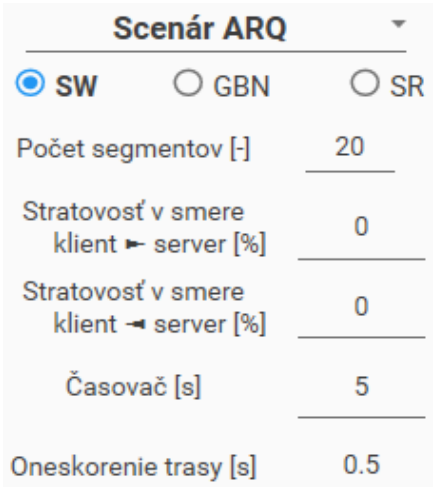
Simulovanie pri bezchybnom prenose slúži na vytvoreniu referenčných hodnôt. Pred spustením simulácie je nutné upraviť základné nastavenia. Počas úlohy budú spustené postupne jednotlivé ARQ protokoly. Nastavením hodnôt spustíme simuláciu a určíme dobu prenosu dát. Ktorý z ARQ protokolov preniesol rovnaký počet správ najrýchlejšie a prečo?

Zadané nastavenia simulácie:

1. počet odoslaných správ: 20
2. pravdepodobnosť opakovaného prenosu klienta a serveru: 0 %
3. časovač opakovaného prenosu: 5 s
4. doba prenosu správy (oneskorenie): 0,5 s
5. počet bitov pre sekvenčné číslo: 4
6. veľkosť okna (pri GBN a SR): 6
7. počet prijatých správ na jedno potvrdenie (pri GBN a SR): 1

Nastavenie programu pred simuláciou SW protokolu

Spustením programu overime nastavenie scenára ARQ. Ako prvý bude simulovaný Stop-and-wait protokol, ktorý je už zvolený tlačidlom **SW**. Po výbere protokolu môžme pokračovať na nasledujúce nastavenia počet odoslaných správ zmeníme na hodnotu 20. Nasleduje stratovosť v smere od klienta k serveru a následne opačne. Obe tieto hodnoty budú nastavené v tejto časti na 0. Postupne postupujeme pri nasledujúcich parametroch časovača a oneskorenia. Hodnoty nastavíme na 5 a 0,5 sekúnd (obr. 8.3). Nastavením všetkých zadaných hodnôt je možné spustiť simuláciu tlačidlom *ŠTART*.



Obr. 8.3: Nastavenia aplikácie pre simulovanie bezchybného prenosu pri SW protokole v simuloačnom scenári ARQ

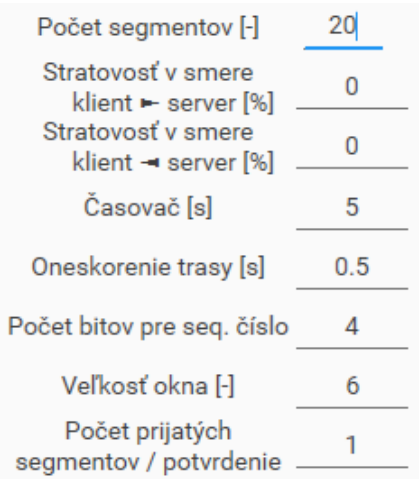
Vyhodnotenie výsledkov simulácie pri zvolenom SW protokole

Po ukončení simulácie sú vygenerované dve pcap súbory, jeden pre klienta s názvom „*node_PC-A_ARQ_StopAndWait..*“ a druhý pre server s názvom „*node_Server_ARQ*

_StopAndWait..“, kde na konci sú zaznamenané časy vytvorenia súboru. Pretože prenos inicializoval klient použijeme na nasledujúce výpočty jeho pcap súbor. Celkový čas prenosu všetkých dát na server získame odčítaním posledného prijatého potvrdenia od času odoslania prvej správy, bez naviazania a ukončenia TCP spojenia. Výsledná hodnota 20,06 sekúnd je následne zaznamenaná v grafe 8.7. Čas prenosu bude neskôr porovnaný s ostatnými protokolmi.

Nastavenie parametrov pre použitie GBN a SR protokolu

Simulovanie protokolu GBN a následne SR si vyžaduje zadanie viacerých parametrov. Ako prvé bude označená skratka protokolu **GBN**, analogicky pre SR. Pri výbere sa v aplikácií zobrazia nové nastavenia. Prvých 5 políčok ostane rovnakých ako v predchádzajúcej simulácií, ostatné je nutné upraviť. Počet bitov pre sekvenčné číslo zmeníme na hodnotu 4 a veľkosť okna na hodnotu 6. Posledný parameter *Počet prijatých sgmentov/ACK* nastavíme na hodnotu 1. Po vložení požadovaných parametrov bude sekcia vstupných parametrov nasledujúca obr. 8.4. Ostatné nastavenia ostanú rovnaké z predchádzajúcej simulácie. Nastavením všetkých hodnôt môžme spustiť simuláciu. Po jej ukončení overíme funkčnosť protokolu pomocou zachytenej komunikácie na strane serveru.



Obr. 8.4: Nastavenia aplikácie pre simulovanie bezchybného prenosu pri GBN alebo SR protokole v simuloačnom scenári ARQ

Vyhodnotenie výsledkov simulácií

Prenos začal naviazaním TCP spojenia, ktoré inicioval klient s IP adresou 172.40.10.125 na server 172.40.10.1. Po zostavení spojenia server správne prijal 6 správ, po ktorých

odoslal potvrdenie obr. 8.5 (riadok číslo 10). Po odoslaní potvrdenia klient znovu začal odosielať 6 správ, ktoré následne prijíma server.

1	2020-05-21 01:12:16,62…	172.40.10.125	172.40.10.1	TCP	54	8910 → 80 [SYN] Seq=
2	2020-05-21 01:12:16,63…	172.40.10.1	172.40.10.125	TCP	54	80 → 8910 [SYN, ACK]
3	2020-05-21 01:12:17,64…	172.40.10.125	172.40.10.1	TCP	54	8910 → 80 [ACK] Seq=
4	2020-05-21 01:12:18,31…	172.40.10.125	172.40.10.1	HTTP	1516	POST http://www.loca
5	2020-05-21 01:12:18,46…	172.40.10.125	172.40.10.1	HTTP	1109	POST http://www.loca
6	2020-05-21 01:12:18,61…	172.40.10.125	172.40.10.1	HTTP	118	POST http://www.loca
7	2020-05-21 01:12:18,76…	172.40.10.125	172.40.10.1	HTTP	118	POST http://www.loca
8	2020-05-21 01:12:18,91…	172.40.10.125	172.40.10.1	HTTP	118	POST http://www.loca
9	2020-05-21 01:12:19,06…	172.40.10.125	172.40.10.1	HTTP	118	POST http://www.loca
10	2020-05-21 01:12:19,07…	172.40.10.1	172.40.10.125	TCP	54	80 → 8910 [ACK] Seq=
11	2020-05-21 01:12:20,29…	172.40.10.125	172.40.10.1	HTTP	118	POST http://www.loca

Obr. 8.5: Časť zachytených paketov na strane servera pri bezchybnom prenose protokolom GBN pri veľkosti okna 6

Záver a diskusia úlohy

Aká je celková doba prenosu pri jednotlivých protokoloch?
Z vygenerovaných pcap súborov sme určili celkovú dobu prenosu na približne 20,06 sekúnd. Tento čas následne výrazne klesla pri protokolch GBN a SR až na hodnotu približne 4 sekúnd (obr. 8.7).
Čím bol spôsobený rozdiel medzi jednotlivými simuláciami?
Rozdiel doby prenosu medzi SW protokolom a ostatnými je približne 80 %. Táto výrazná odchýlka bola spôsobená odlišnými implementáciami jednotlivých protokolov. V prvom prípade SW odosiela iba jednu správu cez prenosový kanál a následne čaká na jej potvrdenie. Pri ostatných protokolov bolo prenesených naraz 6 správ takmer v rovnakom čase až potom stanica čaká na potvrdenia.

8.1.3 Simulovanie prenosu dát v sieti s chybami

Prenos správ v reálnej sieti je ovplyvnený viacerými faktormi. Jeden z nich je aj stratovosť. Vplyvom takého to faktoru stanice, ak je to nutné musia opakovať prenos, aby zaistili doručenie všetkých dát. Táto úloha bude porovnávať pôsobenie stratovosti pri jednotlivých protokoloch. Program simuluje túto vlastnosť v dvoch smeroch od klienta pri nastavení hodnoty „Stratovosť v smere klienta > server“, alebo v smere od serveru „Stratovosť v smere klienta < server“. Pred odoslaním každej správy program vygeneruje pseudonáhodné číslo z rozsahu $0 - 100\,\%$ a ak je hodnota menšia alebo rovná zadanej hodnote stratovosti v príslušnom smere správa bude zahodená. Úloha bude preto rozdelená do dvoch častí v prvej bude postupne

zvyšovaná stratovosť dát odoslaných od klienta. V druhej časti sa táto hodnota vynuluje a zvyšovať sa bude iba stratovosť v smere od serveru.

Nastavenie parametrov pre stratovosť dát v smere od klienta

Počas simulácie je nutné použiť rovnaké nastavenia aby sme boli schopný porovnať výsledky s ideálnym stavom siete. Pri úlohe bude zmenená pravdepodobnosť opakovaného prenosu klienta postupne na hodnoty 5, 10 a 20 %. Tieto hodnoty nezodpovedajú hodnotám reálnych sieti, boli zvolené za účelom zobrazenia rozdielov medzi protokolmi a ich implementáciou. Na základe výsledkov zistite o koľko percent sa zvýši doba prenosu v porovnaní s prvou simuláciou bezchybného prenosu. Nastavené hodnoty simulácie overíme pomocou vygenerovaných pcap súborov.

Vypracovanie a vyhodnotenie výsledkov úlohy

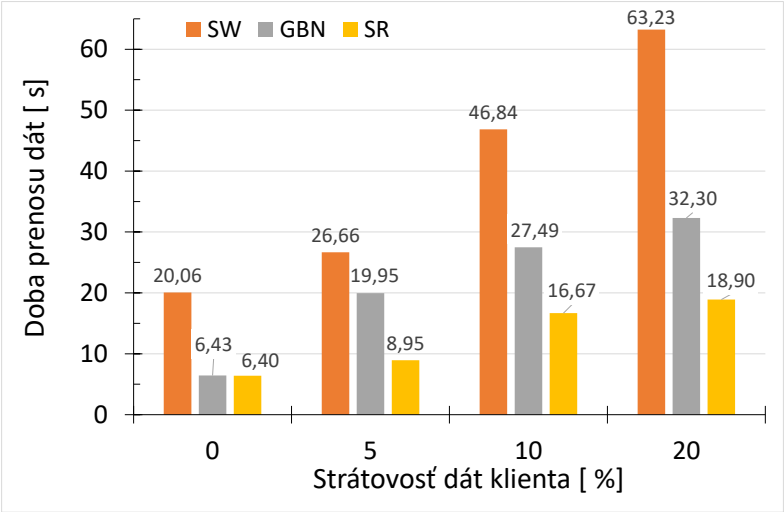
Podľa zadania bude nutné upraviť hodnotu stratovosti. Počet odoslaných segmentov, časovač opakovaného prenosu a ostatné hodnoty ostanú rovnaké ako v predchádzajúcej úlohe obr. 8.4. Hodnotu „Stratovosť v smere klienta > server“ upravíme najprv na hodnotu 5 % a spustíme simuláciu. Po ukončení vypočítame celkový čas prenosu a rovnako pokračujeme s ostatnými dvoma protokolmi.

P.č.	Čas	Zdroj	Cieľ	Protokol	Celková dĺžka	Info
4	19:56:55,…	172.40.10.1…	172.40.10.1	HTTP	1515	POST http://www.local…
5	19:56:56,…	172.40.10.1	172.40.10.1…	TCP	54	80 → 8910 [ACK] Seq=1…
6	19:56:56,…	172.40.10.1…	172.40.10.1	HTTP	1123	POST http://www.local…
7	19:57:01,…	172.40.10.1…	172.40.10.1	TCP	1123	[TCP Retransmission] …
8	19:57:02,…	172.40.10.1	172.40.10.1…	TCP	54	80 → 8910 [ACK] Seq=1…
9	19:57:02,…	172.40.10.1…	172.40.10.1	HTTP	124	POST http://www.local…

Obr. 8.6: Časť zachyteného prenosu klienta pri 5 % stratovosti použitím protokolu SW

Po simulovaní otvoríme postupne jednotlivé pcap súbory a určíme celkovú dobu prenosu, ktorú sme zaznamenali v grafe(obr. 8.7). Jednotlivé znovu odoslané správy zobrazí Wireshark s označením „[TCP Retransmission]...“ (obr. 8.6). V otvorenom súbore sú znovu odoslané správy označené ako . V súbore paket číslo 21 bol odoslaný serveru v čase 36,66 sekúnd a potvrdenie bolo doručené približne o 1,02 sekúnd neskôr. Z hodnoty vyplýva, že na prenos správy jedným smerom trval približne 0,51 sekúnd. Odmeraný čas sa odlišuje od nastavenej hodnoty oneskorenia o 0,01 sekúnd čo je spôsobené spracovaním paketu na jednotlivých stranách.

Doručením potvrdenia (na riadku 5, obr. 8.6) klient odosiela nasledujúci paket, ktorý nebol potvrdení a je nutné ho znova odoslať. Po pakete nasleduje zvýraznený záznam číslo 7, kde sú zhodné zdrojové a cieľové IP adresy. Rozdielu časových značiek paketov 6 a 7 zobrazuje nastavenú hodnotu časovača. Klient približne po 5 sekundách opakoval prenos. Presne zhodný čas bol nastavený aj v aplikácií pred spustením simulácie. Odoslané dáta boli následne úspešne doručené čo dokazuje prijaté potvrdenie na riadku číslo 8.



Obr. 8.7: Porovnanie ARQ protokolov pri bezchybnom prenose a pri postupne zvyšujúcej sa stratovosti dát odoslaných klientom

Záver a zhodnotenie výsledkov prvej časti

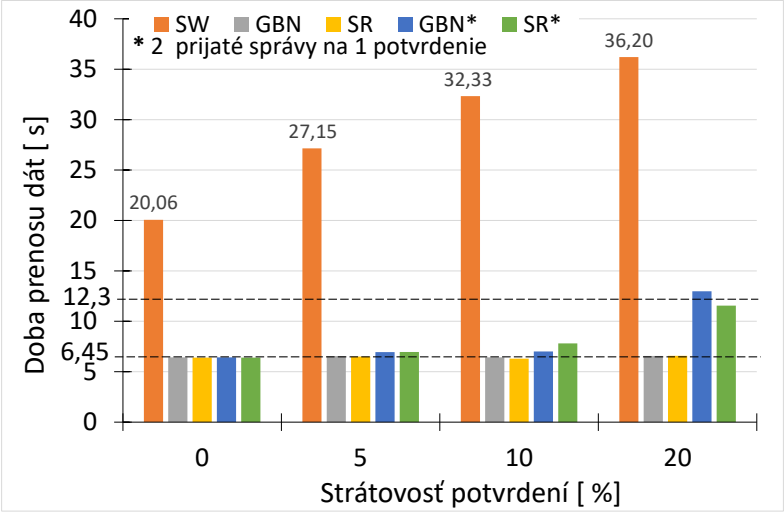
Aký je percentuálny nárast doby prenosu v porovnaní simuláciou bezchybnej siete? Z výsledkov (obr. 8.7) získame čas prenosu a pozorujeme,že nastal výrazný nárast oproti simulácií bezchybného prenosu. Pri prvej simulácií s 5 % stratovosťou je to 33 %, pri GBN následne 212 % a pri SR 40 %. Z percentuálnych hodnôt nárastu vychádza protokol GBN najhoršie a najvýhodnejším protokolom z pohľadu percentuálneho nárastu je SW. Avšak keď porovnáme výslednú dobu prenosu s SW protokolom je jednoznačne vidieť rozdiel v čase. Rozdiel sa následne zväčšoval zo zväčšujúcou sa stratovosťou. Pri 10 % je možné určiť nárast pri protokole SW približne o 130 %, GBN o 330 % a pri protokole SR nastal nárast o 160 %. V poslednej simulácií je tento nárast najvýraznejší kde stúpla doba prenosu približne o 215 %, 405 % a 195 % pri SW,GBN a SR.

Ktorý protokol je najúčinnejší z pohľadu celkovej doby prenosu a ktorý z percentuálneho nárastu doby prenosu v porovnaní s bezchybným prenosom?

Z výsledkov simulácií je možné jednoznačne určiť protokol SR ako najrýchlejší. Pri najväčšej nastavenej hodnote stratovosti preniesol všetky súbory za čas 18,9 sekúnd. Percentuálny náraste doby prenosu je 195 % čo znamená že celkový prenos sa zvýšil približne 3,1 krát oproti prenosu bez chýb.

Druhá časť úlohy - stratovosť dát v smere od serveru

Program bude nastavení rovnako ako v prvej časti s jedným rozdielom, stratovosť v smere od klienta bude 0 % a stratovosti v smere od serveru budú postupne zmenené na 5 %, 10 % a 20 %. Pri úlohe postupujeme rovnako ako v prvej časti. Počet prijatých správ na jedno potvrdenie nastavíme najprv na hodnotu 1 a následne na hodnotu 2.



Obr. 8.8: Porovnanie ARQ protokolov pri bezchybnom prenose a pri postupne zvyšujúcej sa stratovosti potvrdení odoslaných serverom

Jednotlivé časy prenosu všetkých dát sa môžu mierne odlišovať v závislosti na vygenerovanom náhodnom čísle. Pri simulovaní sme zaznamenali hodnoty do grafu obr. 8.8. V grafe sú zobrazené hodnoty, kde server odoslal potvrdenie po každej správe a následne po prijatí dvoch správ.

Z výsledkov je možné vidieť približne rovnaký nárast doby prenosu pri SW protokole ako tomu bolo v prvej časti úlohy. Pri ostatných protokoloch je celkový čas prenosu približne v priemere 6,45 sekúnd. Pri stratovosti 20 % je doba prenosu približne 12,3 sekúnd.

Záver a zhodnotenie výsledkov druhej časti

Aký je rozdiel medzi dobou prenosu dát pri protokole Stop-and-Wait?
Pri simulácií sú odoslané správy potvrdzované serverom, ktorého potvrdenia sa ná-

hodne stratia. Nastavením SW protokolu je možné pozorovať výrazný nárast celkovej doby prenosu. Je to spôsobené implementáciou daného protokolu. Ak bolo odoslané potvrdenie stratené, klient nevie, že server prijal správu. Po uplynutí časovača, musí klient odoslať správu znova a opäť čakať na potvrdenie.

Aký je rozdiel medzi dobou prenosu dát pri ostatných dvoch protokoloch? Každé kladné potvrdenie oznámi stanici prijatie všetkých predchádzajúcich správ. Táto vlastnosť sa výrazne prejavila pri protokoloch GBN a SR. Klient odosielal postupne jednotlivé správy, ktoré boli postupne potvrdzované. Pri strate potvrdenia správy, prijatie správ potvrdzuje nasledujúce prijaté potvrdenie. Pri nastavení dvoch správ na jedno potvrdenie je možné pozorovať miernu odchýlku v dobe prenosu.

Nastavením stratovosti na hodnotu 20 % nastal dvojnásobný nárast doby prenosu. Tento jav bol spôsobený stratov niekoľkých potvrdení. Z jednotlivých pcap súborov klienta a serveru bolo možné pozorovať stratu posledného potvrdenia. Daná správa mala potvrdiť posledné 2 správy klienta. Po ich nedoručení vypršal 5 sekundový časovač a prenos správ sa zopakoval.

8.1.4 Vplyv veľkosti okna na čas prenosu dát

Cieľom úlohy je zistiť vplyv veľkosti okna protokolov GBN a SR na dobe prenosu dát. Veľkosť okna na strane klienta budeme postupne zvyšovať. Táto hodnota udáva, počet správ, ktoré môže naraz odoslať klient a budú nastavené následovne: $1, 2, 4, 8$ a 16. Počet prijatých správ na jedno potvrdenie bude rovnaký a následne o polovicu menší ako je veľkosť okna.

Nastavenia simulácie:

1. počet odoslaných správ: 32
2. pravdepodobnosť opakovaného prenosu klienta a serveru: 0 %
3. časovač opakovaného prenosu: 5 s
4. doba prenosu správy (oneskorenie): 0,5 s
5. počet bitov pre sekvenčné číslo: 5

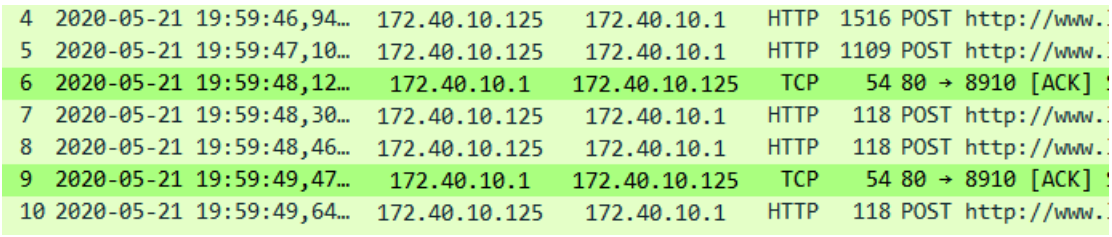
Postup úlohy a vyhodnotenie výsledkov úlohy

Pred spustením simulácie je nutné nastaviť jednotlivé parametre podľa zadania. Ako prvý bude simulovaný protokol GBN. Pred zadaním parametrov je nutné zistiť potrebný počet bitov pre sekvenčné číslo. Pri riešení budeme vychádzať z samotnej implementácie protokolu, kde je na strane klienta veľkosť okna postupne $1, 2, 4, 8$ a 16 a strane serveru 1.

Ako prvý vyberieme protokol GBN a nastavíme jednotlivé parametre podobne ako v predchádzajúcich úlohách. Veľkosť okna upravíme na hodnotu 1 rovnako tak počet správ prijatých na jedno potvrdenie. Spustením simulácie je možné sledovať

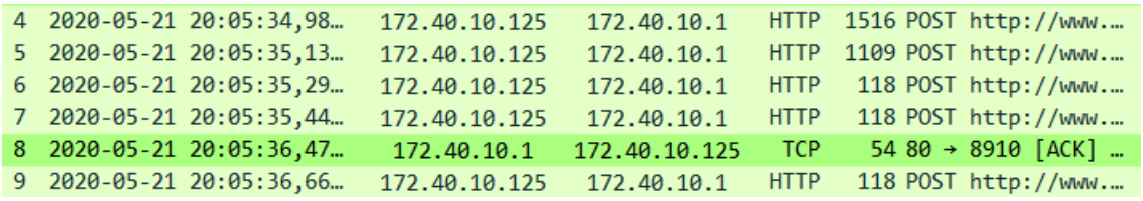
rovnaký priebeh ako pri SW protokole. Celkový čas simulácie je následne pri oboch protokoloch rovnaký približne 34,5 sekúnd.

Následne upravíme veľkosť okna na hodnotu 2 a spustíme simuláciu. Po simulovaní zistíme dobu prenosu, ktorá je 21,45 sekúnd. Po simulovaní SR protokolu je táto hodnota podobná 21,51 sekúnd.



Obr. 8.9: Prenos prvých správ klienta pri GBN s veľkosťou okna 2 a prijatí dvoch potvrdení

Prvé odoslané a prijaté správy klienta sú následne znázornené na obr. 8.9. Z komunikácie je možné vidieť odoslanie dvoch paketov v krátkom časovom úseku po sebe a následne ich potvrdenie paketom na riadku číslo 6. Pri prenose môžeme pozorovať, že čas potrebný na odoslanie jedného okna až po jeho potvrdenie je približne 1,19 sekúnd. Podobne postupujeme aj pri veľkosti okna 4.



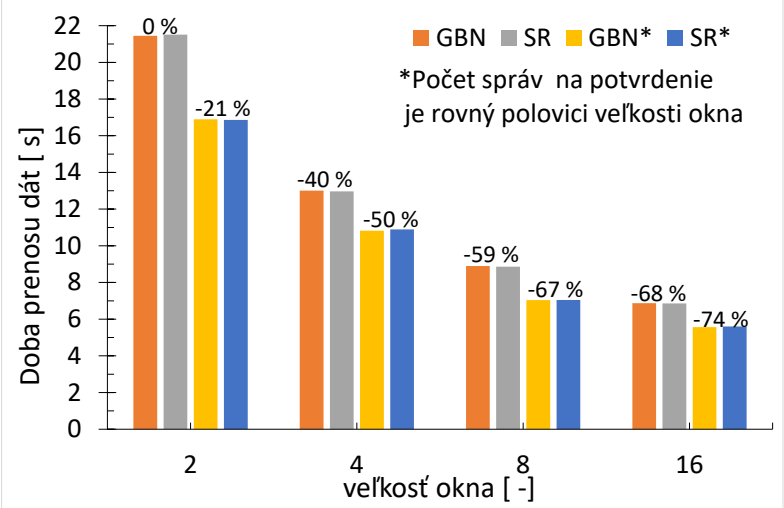
Obr. 8.10: Prenos prvých správ klienta pri GBN s veľkosťou okna 4 a prijaté potvrdenie

Z zachytenej komunikácie obr. 8.10 je možné pozorovať zhluk 4 paketov, ktoré boli odoslané v krátkom časovom okamihu. Pre porovnanie s predchádzajúcou simuláciou odčítame čas prijatia potvrdenia na riadku 8 od času, kedy bol odoslaný prvý paket z daného okna. Výsledkom je hodnota približne 1,5 sekúnd. Prenos jedného okna s počtom 4 paketov narástol o približne 26,05 %. Postup opakujeme následne pre veľkosti okna 8 a 16.

Záver a diskusia úlohy

Ako sa zmenil čas prenosu dát pri zvyšujúcej sa veľkosti okna?
Pri porovnaní s predchádzajúcim protokolom zisťujeme, že hodnoty sa líšia len o 6 stotín sekundy v ideálnych podmienkach, kde nedochádza k chybám, alebo stratám paketov. Táto odchýlka je spôsobené chybou merania nakoľko podstata protokolov GBN a SR je rovnaká. Pri oboch server čaká na prijatie stanoveného počtu správ až následne odošle potvrdenie.

Zmena by nastala až v prípade výskytu chýb v sieti, kedy by pri GBN server začal zahadzovať všetky prijaté správy, ktoré by nasledovali za aktuálne nedoručenou správou. Celková doba prenosu by sa tým pádom predĺžila o tento počet správ. V prípade SR by nastalo opakovanie len vybranej správy a tým by celková doba prenosu bola nižšia. Spomenuté prípady boli bližšie skúmané v predchádzajúcej úlohe 8.1.3.



Obr. 8.11: Závislosť veľkosti okna a doby prenosu pri GBN a SR pri prenose 32 správ

Aký je percentuálny pokles doby prenosu pri zvyšujúcej sa veľkosti okna?
Doba prenosu pri zväčšení veľkosti okna z 2 na 4 klesla pri oboch protokoloch približne o 40 %. Následne bol nastavená dvojnásobná veľkosť 8, kde nastal pokles o 59 %. Ako posledná bola testovaná veľkosť okna 16, kde klient musel odoslať 16 správ až po ich doručení mohol server odoslať potvrdenie. Doba prenosu je najmenšia z ostatných simulácií. Ak porovnáme túto hodnotu s prvou simuláciou dostaneme pokles približne 63 %. Jednotlivé hodnoty sme následne zaznamenali do grafu 8.11.

Aký vplyv mala zmena hodnoty „počet správ na potvrdenie“ na simuláciu?
Pri simulácií je možné ako už bolo spomenuté sme simulovali prenos 32 správ. Počet

správ na jedno potvrdenie bol rovnaký ako veľkosť okna, čo sme popísali v predchádzajúcej otázke. Po simulovaní sme túto hodnotu znížili na polovicu. Ako referenčnú hodnotu zvolíme prvú simuláciu kde bola celková doba prenosu približne 21,5 sekúnd a veľkosť okna rovná 2. Ak server pri prenose odosiela potvrdenie na každú prijatú správu zníži sa doba prenosu o 21 %. Pri nasledujúcich simuláciách je tento pokles približne 10, 8 a 6 %. Z jednotlivých údajov následne vyplýva, že vhodne zvolenou veľkosťou okna je možné znížiť dobu prenosu pri na úkor pamäťových nárokov.

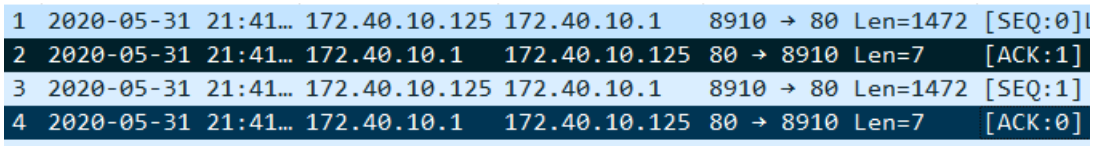
8.1.5 Veľkosť okna pri ARQ protokoloch a voľba vhodného rozsahu sekvenčných čísel

Pri prenose dát jedným z ARQ protokolov je dôležitým faktorom veľkosť okna, ktorá udáva počet správ ktoré môžu stanice naraz odoslať. Počas prenosu správ v sieti tieto správy nemusia byť doručené v poradí v akom boli odoslané a preto je nutné ich označiť. Server tak bude presne vedieť, ktoré správy prijal a ktoré budú ešte doručené. Na označenie jednotlivých správ je v segmente vyhradený počet bitov. Aplikácia zobrazuje konkrétne sekvenčné číslo zjednodušene v zozname správ klienta a serveru za skratkou **SEQ** (sekvencia), alebo v dátovej časti UDP segmentu. Zistite koľko najmenej bitov je potrebných pre správne fungovanie protokolov GBN a SR, pri výpočtoch uvažujeme veľkosť okna 3 ($W = 3$). Následne odôvodnite veľkosť okna pri protokole SW. Počas prenosu použite transportný protokol UDP a zobrazte prvé prenesené správy klienta pri protokoloch Go-back-N a Selective repeat.

Určenie veľkosti okna a sekvenčných pri SW protokole

Aký je minimálny počet bitov pre sekvenčné číslo pri protokole SW a odôvodnite prečo?
Použitím protokolu SW sme schopný odoslať jednu správu za určitý časový úsek. Odoslaním správy stanica nastaví časovač opakovaného prenosu a počká na potvrdenie. Po doručení odosiela nasledujúci rámec v poradí. V prípade vypršania časovača stanica opakuje prenos len danej správy uloženej v pamäti. Pri prenose jednej správy tak stačí použiť iba jeden bit a počet sekvenčných čísel tak bude len 0 a 1. Server prijatím správy s číslom 0 očakáva nasledujúcu správu s číslom 1 a následne po jej doručení opäť 0.

Klient odošle správu 0 a čaká potvrdenie. Ak sa potvrdenie stratilo klient odošle správu znovu, ktorá bude doručená na server. Ten tak obdrží správu s číslom 0, preto, že správa neobsahuje číslo 1 odpovie a duplicitnú správu zahodí. Výsledný počet bitov pre sekvenčné číslo je 1 hodnota sa následne zobrazí v zozname správ klienta a následne v dátovej časti UDP obr. 8.12.

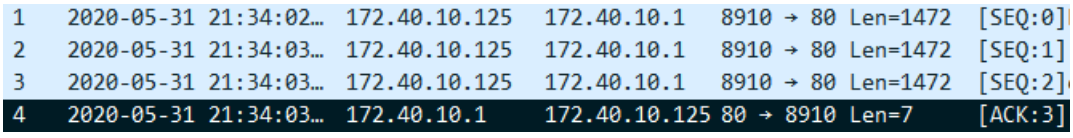


Obr. 8.12: Časť komunikácie serveru pri číslovaní správ 1 bitovým sekvenčným číslom

Zistenie minimálneho počtu bitov pre sekvenčné čísla pri GBN a SR protokole

Pri protokole GBN a SR je situácia zložitejšia, preto že nastáva prenos viacerých správ naraz. Začneme protokolom GBN veľkosť okna na strane klienta je zvolená na hodnotu 3 a server ju nastaví na hodnotu 1, čo vyplýva z implementácie protokolu popísanej v predchádzajúcich častiach. Pri zistení minimálneho počtu bitov pre sekvenčné čísla budeme vychádzať z nerovnosti 4.1. Po vyjadrení neznámej m získame nerovnosť $m \geq log_2 (W + 1)$ a následne dosadíme zadanú veľkosť okna W. Výsledná hodnota je $m \geq 2$.

Aký je minimálny počet bitov pre sekvenčné číslo pri protokole GBN?
Výpočtom sme zistili že pri veľkosti okna nastavenej na hodnotu 3 budú potrebné minimálne 2 bity pre sekvenčné číslo. Rozsah použiteľných sekvenčných čísel tak budú všetky celé čísla od 0 do 3 vrátane. Výpočet overíme v simuláciami voľbou GBN protokolu a zadaním veľkosti okna na hodnotu 3. Vykonali sme to nastavením zadaných hodnôt a spustením simulácie. prvé štyri pakety boli sú následne zobrazené na obr. 8.13. V poslednom stĺpci je možné vidieť dvoj bitové sekvenčné čísla dátových správ a následného potvrdenia.



Obr. 8.13: Časť komunikácie serveru pri číslovaní 2 bitovým sekvenčným číslom

Protokol SR musíme uvažovať aj veľkosť okna na strane serveru. Pri prenose sa nezávisle posúva a tak klient aj server majú informácie o doručených a potvrdených správach. Ak by sme pri výpočte rozsahu sekvenčných čísel postupovali rovnako ako pri GBN protokole, môže nastať nasledujúca situácia.

Klient aj server majú nastavenú veľkosť okna na hodnotu 3 a počet bitov pre sekvenčné číslo na hodnotu 2. Klient odosiela prvé správy z okna očíslované 0, 1 a 2 a čaká na ich potvrdenie. Server príjme správy a odosiela kumulatívne potvrdenie po jeho odoslaní aktualizuje svoje okno. V tomto okamihu server očakáva nové správy

očíslované hodnotami 3, 0* a 1*. Pre malý počet sekvenčných čísel server musí znovu použiť hodnoty 0 a 1 (pre odlíšenie sú nové správy 0 a 1 označené *). Počas prenosu potvrdenia nastala chyba a klient musel potvrdenie zahodiť. Po uplynutí časovača klient odosiela opäť správy 0, 1 a 2. Po ich prijatí server nemôže rozoznať či príslušné správy 0 a 1 sú nové alebo už boli odoslané. Ich čísla sa nachádzajú v okne a tak sú akceptované. Pri prenose tak niky nebudú správy 0* a 1* doručené.

Aký je minimálny počet bitov pre sekvenčné číslo pri protokole SR a odôvodnite prečo?
Problém je možné vyriešiť zväčšením počtu sekvenčných čísel. Pri nasledujúcom výpočte požadovaného počtu bitov budeme vychádzať z nerovnosti 4.2. Odvodíme premennú m ako $m \geq log_2(W) + 1$ a následne za hodnotu W dosadíme opäť 3. Výsledná nerovnosť pre m bude: $m \geq 2.5849$.

Z uvedeného výpočtu volíme počet bitov pre sekvenčné číslo m na najbližšiu celú hodnotu 3. Jednotlivé správy tak bude možné označiť celými číslami od 0 po 7 vrátane. Vypočítaná hodnota bola nastavená v programe a opäť simulovaná. Časť komunikácie je zobrazená na obr. 8.14, v poslednom stĺpci sú následne správy označené vypočítanými číslami.

```
5  2020-05-31 21:45… 172.40.10.125 172.40.10.1   8910 → 80 Len=1472 [SEQ:3]
6  2020-05-31 21:45… 172.40.10.125 172.40.10.1   8910 → 80 Len=1472 [SEQ:4]
7  2020-05-31 21:45… 172.40.10.125 172.40.10.1   8910 → 80 Len=1472 [SEQ:5]
8  2020-05-31 21:45… 172.40.10.1   172.40.10.125 80 → 8910 Len=7    [ACK:6]
9  2020-05-31 21:45… 172.40.10.125 172.40.10.1   8910 → 80 Len=1472 [SEQ:6]
```

Obr. 8.14: Časť komunikácie serveru pri číslovaní 3 bitovým sekvenčným číslom

8.1.6 Vplyv oneskorenia na celkovú dobu prenosu

Prenos dát v sieti je vždy ovplyvnený viacerými parametrami. Jedným z nich je čas potrebný na zostavenie a odoslanie dátovej jednotky príjemcovi. Ak prenášame dáta na väčšie vzdialenosti vo väčšej sieti, sú tieto správy vždy doručené s zväčšujúcim sa oneskorením. Po simulácií následne porovnajte celkový čas prenosu všetkých dát pri jednotlivých protokoloch.

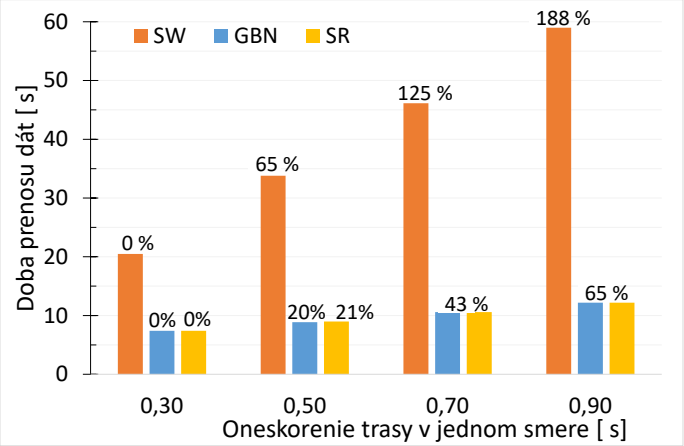
Nastavenia simulácie:

1. počet odoslaných správ: 32
2. pravdepodobnosť opakovaného prenosu klienta a serveru: 0 %
3. časovač opakovaného prenosu: 5 s
4. doba prenosu správy (oneskorenie): 0,3; 0,5; 0,7; 0,9 s
5. počet bitov pre sekvenčné číslo: 8

6. veľkosť okna a počet prijatých správ na jedno potvrdenie: 8

Postup úlohy a vyhodnotenie výsledkov

Počas úlohy bude v jednotlivých simuláciách doba potrebná na prenos dát od zdroja k cieľu postupne narastať. Pred spustením prvej simulácie nastavíme jednotlivé parametre podľa zadania. Ako prvý bude simulovaný SW protokol. Po simulácií podobne ako v predchádzajúcich úlohách určíme dobu prenosu pri prvej sieti s na 20,49 sekúnd. Pri protokole GBN a SR je to len 7,40 sekúnd.



Obr. 8.15: Závislosť zväčšujúceho sa oneskorenia trasy na dobe celkového prenosu dát odoslaním 32 správ

Záver a diskusia

Ako sa menila doba prenosu pri jednotlivých simuláciách?
Po ukončení všetkých simulácií a vyhodnotení bolo zistené, že celková doba prenosu dát ako je znázornené aj na grafe 8.15. Zo zväčšujúcou sa hodnotou z 0,3 na 0,5 sekúnd vzrástla aj celkový čas. Použitím SW protokolu bol dosiahnutý nárast približne o 65 %, ale pri ostatných dvoch to je len 20 % a 21 %. Neskôr sme zvyšovali dobu prenosu na hodnotu 0,7 sekúnd. Tretia simulácia ukázala jednoznačnú nevýhodu Stop-and-Wait, kde bol opäť výrazný nárast oproti prvej simulácií približne o 125 %. Pri ostatných protokoloch je táto hodnota rovnaká a to približne 43 %. V poslednej simulácií je oneskorenie nastavené na hodnotu 0,9 sekúnd. Prenos všetkých správ pri prvom protokole opäť výrazne stúpol o 188 % a o 65 % rovnako pri ostatných dvoch protokoloch.

Ktorý z ARQ protokolov dosiahol najnižší čas prenosu všetkých dát a prečo?
Z výsledkov simulácií vyplýva, že doba prenosu protokolom SW sa neporovnateľne

zvyšuje, čo je spôsobené jeho implementáciou kedy je možné odoslať iba jednu správu za daný čas. Pri GBN aSR je tento nárast menší. Tieto protokoly dosahujú najlepší čas prenosu ale majú vyššie výpočtové a pamäťové nároky v porovnaní s SW. Stanice majú v pamäti iba 1 správu a jednoduchší mechanizmus kontroly správ.

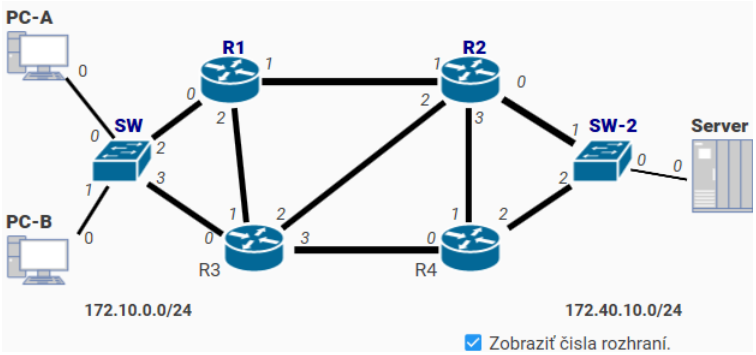
Ako by bol ovplyvnený výsledok ak časovač opakovaného prenosu bol nastavený na hodnotu napríklad 1 sekunda?
Zmena hodnoty časovača opakovaného prenosu na hodnotu jednej sekundy by pri prvej simulácií nepredstavovala žiaden problém. Odoslaná správa je doručená serveru po 0,3 sekundách, a následne potvrdenie obdrží klient v čase približne 0,6 sekúnd po odoslaní. Nastavený časovač by ostal na hodnote 0,4 sekúnd.

Hraničnou hodnotou je čas jednosmerného oneskorenia 0,5 sekúnd kedy záleží na presnosti a rýchlosti zariadení. Potvrdenie správy môže byť prijaté niekoľko milisekúnd pred vypršaním časovača a tak túto správu nieje nutné opakovať. Ak by sa potvrdenie omeškalo, časovač by vypršal a správa by bola znovu odoslaná.

Pri posledných dvoch simuláciách bude zaťaženie siete najväčšie. Počas prenosu bude stanica odosielať každú správu dvakrát, nakoľko vždy vyprší časovač opakovaného prenosu pred doručením potvrdenia.

9 Scenár Komutácie v simulačnom programe

Scenár komutácií sa skladá zo štyroch častí: komutácie okruhov, správ, paketov a buniek. Jednotlivé časti sú v nasledujúcej kapitole postupne popísané spolu s otázkami. Pri všetkých častiach aplikácia pracuje s topológiou zobrazenou na obr. 9.1. Počas simulácií komunikujú koncové stanice *PC-A* a *PC-B* v podsieti 172.10.0.0/24 so serverom v podsieti 172.40.10.0/24.



Obr. 9.1: Topológia siete pre simulačný scenár komutácií

9.1 Úlohy v simulačnom scenári

Po spustení programu je nastavený ako simulačný scenár ARQ protokol, ktorý je nutné zmeniť. po kliknutí na možnosť **ARQ scenár** zvolíme z zobrazenej ponuky možnosť *Scenár komutácií*. Program následne zobrazuje rozloženie pre simuláciu paketov znázornené v prílohe na obr. A.3.

Pri jednotlivých úlohách sa budú postupne meniť nastavenia na základe požiadavkov a samotné rozhranie programu. Podrobné kroky k splneniu úloh a ich výsledky sú uvedené v nasledujúcich podkapitolách.

9.1.1 Komutácie okruhov a správ

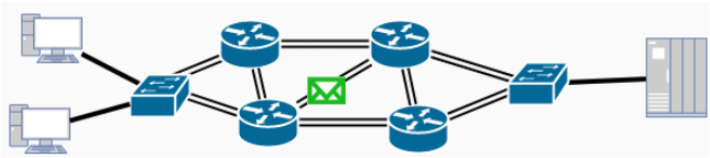
Po zvolení scenára aplikácia zobrazí druhú sadu nastavení, kde budú zvolené *Okruhy* a neskôr *Správy*. Pri simulačnej úlohe nieje nutné upravovať nastavenia aplikácie ani meniť globálne nastavenia.

Postup a vyhodnotenie výsledkov úlohy

Po spustení programu je zobrazený scenár ARQ protokolu, čo nezodpovedá zadaniu. Kliknutím na názov aktuálneho scenáru sa nám zobrazí ponuka, kde následne zvo-

líme scenár *komutácií*. Výberom program upravil rozloženie komponentov a pridal novú položku pod výber scenára.

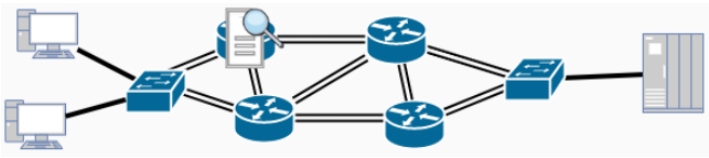
Ako prvé zobrazíme rozloženia pre komutáciu okruhov zobrazenú v prílohe na obr. A.4. Program zobrazil stručný popis mechanizmu, ktorý následne overíme spustením animácie. Tlačidlom pod obr. 9.2 animácia odošle správu a spustí rezervovanie sieťových prostriedkov. Pri doručení správy serveru, potvrdí rezervovanie a je zostavený okruh vyznačený zelenou.



Obr. 9.2: Rezervovanie sieťových prostriedkov pri komutácií okruhov

Vytvorením okruhu nastáva prenos správ po trase. Zo simulácie môžeme pozorovať určité zdržanie, ktoré je spôsobené rezervovaním prostriedkov pred odoslaním správy. V animácií bol znázornený ideálny stav, ktorý v reálnych podmienkach nenastáva a pri prenose môže byť táto správa poškodená. Následne by muselo nastať obnovenie a odoslanie novej správy. Rezervovaná trasa je vyhradená po celú dobu prenosu, čo zvyšuje cenu a rovnako tak znižuje efektivitu.

Ako druhý spôsob komutácie zvolíme z ponuky *Správy*. Aplikácia zobrazí podobné rozloženie ako v predchádzajúcom prípade zobrazené v prílohe na obr. A.4. Opäť naštudujeme popis tejto technológie a spustíme animáciu. Počas prenosu nenastáva rezervácia spojenia od účastníka až k príjemcovi. Z animácie je možné pozorovať prenos celej správy naraz. Po prijatí sieťovým prvkom je celá správa rozbalená, skontrolovaná ako je znázornené na obr. 9.3 a odoslaná nasledujúcemu uzlu.



Obr. 9.3: Kontrola správy sieťovým uzlom pri komutácií správ

Záver a diskusia úlohy

Aké sú rozdieli medzi komutáciou okruhov a správ?
V úlohe boli porovnané dve metódy komutácií. Výhodou prenosu správy pri komutácií okruhov je stabilita prenosu, kedy sa prenesú správy po rezervovanom spojení.

Počas prenosu môže nastať situácia, kedy stanica nevysiela. V tom prípade je rezervovaná trasa nevyužitá. Spomenutý problém rieši metóda, komutácie správ, pri prenose sa nerezervuje spojenie. Stanice príjmu celú správu, ktorú následne prepošlú a sú pripravené na prenos ostatných správ od iných staníc. Táto technológia má vysoké nároky na pamäť jednotlivých uzlov. Správa musí byť celá uložená až potom je možné ju preposlať.

9.1.2 Komutácie paketov a buniek, porovnanie a zistenie rozdielov

Prvé dve metódy boli založené na prenose celej správy bez jej delenia. Pri komutácií paketov budú tieto dáta rozdelené na menšie úseky s stanovenou veľkosťou. V prvej časti simulujte prenos textového súboru s veľkosťou kilobajtov za použitia paketov a následne v druhej časti pomocou buniek. Po simuláciách zistite maximálnu veľkosť dátovej časti bunky a následne paketu. Pri simulácií ostanú nastavené základné nastavenia, ich hodnoty nemajú vplyv na požadované výsledky. Ako transportný protokol bude použitý TCP a veľkosť dátovej časti bunky 120 B.

Prenos súboru pomocou komutácie paketov, nastavenia

Pred spustením simulácie vygenerujeme text s požadovanou veľkosťou, ktorý bude následne odoslaný. V aplikácií zobrazíme globálne nastavenia, kde zvolíme možnosť „Odoslať text súbor?“. Následne sa zobrazí prázdne pole s popisom „Cesta k súboru“, vedľa ktoré ho je zobrazené tlačidlo „**...**“, s ktorým vyberieme vytvorený súbor.

Koľko paketov bude potrebných pre prenos zvoleného súboru?
Použitím dialógového okna sme nahrali do programu súbor s veľkosťou 2 kilobajty. Program má nastavenú veľkosť maximálnej dátovej jednotky na hodnotu 1500 bajtov po odčítaní IP hlavičky získame 1480 B dát v jednom pakete. Pri zvolení TCP protokolu musíme následne odčítať 20 B tvoriacich jeho hlavičku. Celkový objem dát bude 1460 B. Pri súbore s veľkosťou 2000 B budeme potrebovať celkom dva pakety. Hodnotu vložíme do programu a spustime simuláciu.

Vyhodnotenie výsledkov simulácie

Po simulovaní s nastavenými základnými hodnotami otvoríme vygenerovaný záznam serveru, súbor označený „node_Server_Packets...“. Z súboru obr. 9.4 je možné vidieť oba prenesené pakety a následne ich potvrdenia.

Aká je celková dĺžka prenesených dát zobrazených v Wireshark programe?
Po naviazaní spojenia ako prvý bol prenesený paket s číslom 4. Celková dĺžka je 1514 bajtov spolu s hlavičkou spojovej vrstvy. Tá je dlhá presne 14 bajtov z čoho

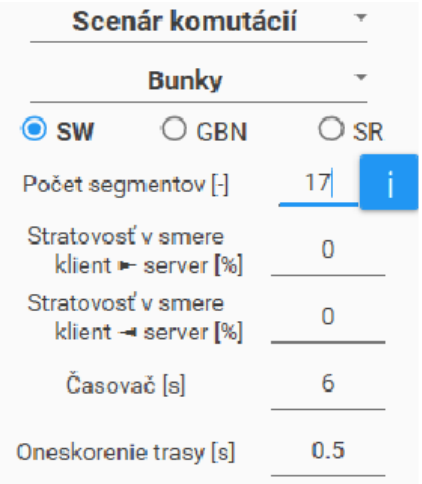
P.č.	Čas	Zdroj	Cieľ	Protokol	Info	Celková dĺžka	Hlavička segm.	Dĺžka dát
4	08:42:30…	172.10.10.125	172.40.10.1	TCP	8910 → 80…	1514	20	1460
5	08:42:35…	172.40.10.1	172.10.10.125	TCP	80 → 8910…	54	20	0
6	08:42:35…	172.10.10.125	172.40.10.1	TCP	8910 → 80…	594	20	540
7	08:42:40…	172.40.10.1	172.10.10.125	TCP	80 → 8910…	54	20	0

Obr. 9.4: Zachytené správy servera pri komutácií paketov

vyplýva, že veľkosť IP paketu je presne 1500 bajtov. Ako sme vypočítali pred simuláciou v pakete sa nachádza zapuzdrený transportný protokol TCP. Celkový objem dát textového súboru, ako sme určili je následne 1460 bajtov (stĺpec *Dĺžka dát*). Druhý paket prenášal následne menší objem dát. Z záznamu môžme zistiť celkovú veľkosť dát na hodnotu 594 Bveľkosť. Odčítaním príslušných hlavičiek získame hodnotu 540 bajtov. Tieto údaje sú vložené do nasledujúceho paketu s číslom 6 (obr. 9.4). Odčítaním jednotlivých hlavičiek protokolov získame presne počet bajtov, ktoré bolo potrebné odoslať.

Prenos súboru pomocou komutácie buniek

Pri komutácií buniek vyberieme z zobrazenej ponuky, ktorá sa nachádza pod názvom zvoleného scenára, možnosť *Bunky*. Ostatné nastavenia ostanú nezmenené.



Obr. 9.5: Základné nastavenia pre simuláciu komutácii buniek s protokolom SW

Koľko buniek bude potrebných pre prenos zvoleného súboru?
Výpočet počtu buniek bude jednoduchší. Pri stanovenej veľkosti dátovej časti bunky na hodnotu 120 B budeme potrebovať iba 17 buniek (2000 B/120 B $\doteq$ 17). Výpočet

je možné overiť programom po kliknutí na talčidlo **i** zobrazené vedľa poľa pre zadávanie počtu segmentov. V zobrazenej správe vidíme odporúčanú hodnotu 17. Túto správu potvrdíme tlačidlom *„YES“* a spustíme simuláciu. Výsledné nastavenia pre simuláciu sú znázornené na obr. 9.5.

Vyhodnotenie výsledkov simulácie

Po simulovaní prenosu opäť otvoríme pcap súbor serveru obr. 9.6.
Aká je celková dĺžka prenesených buniek zobrazených v Wireshark programe?
Zo záznamu je možné určiť celkovú dĺžku zaznamenaných dát na 174 bajtov. Podobne ako pri predchádzajúcej simulácií ak odčítame veľkosť jednotlivých hlavičiek, ktoré boli vložené do správy získame celkovú dĺžku dát. Táto hodnota 120 bajtov je pri všetkých správach rovnaká. Posledná bunka má následne vloženú výplň aby bola zachovaná potrebná veľkosť.

P.č.	Čas	Zdroj	Cieľ	Protokol	Info	Celková dĺžka	Hlavička segm.	Dĺžka dát
34	09:35:25…	172.10.10.126	172.40.10.1	TCP	8710 → 80…	174	20	120
35	09:35:27…	172.40.10.1	172.10.10.126	TCP	80 → 8710…	54	20	0
36	09:35:27…	172.10.10.126	172.40.10.1	TCP	8710 → 80…	174	20	120
37	09:35:29…	172.40.10.1	172.10.10.126	TCP	80 → 8710…	54	20	0

Obr. 9.6: Záznam posledných komunikácie servera pri komutácií buniek pri odoslaní dvoch paketov

Záver a diskusia úlohy

Pri jednotlivých simuláciách bol odoslaní súbor s veľkosťou 2000 bajtov. Zvolením komutácie paketov bol prenesený súbor v dvoch paketoch, kde druhý nebol naplnený na maximálnu kapacitu. V druhej simulácií rovnaký súbor prenieslo celkom 17 buniek, ktoré mali konštantnú veľkosť 174 bajtov.

Aký je rozdiel medzi posledným prenesením paketom a poslednou prenesenou bunkou?
Klient v poslednom pakete odoslal iba 540 B textového súboru a potrebné hlavičky. Posledná odoslaná bunka preniesla 80 bajtov z vloženého textového súboru a ostatné bajty bunky tvorila výplň.

9.1.3 Prenos súborov rôznych veľkostí pomocou komutácie paketov a buniek

Rozdieli medzi jednotlivými komutáciami je možné pozorovať pri odoslaní rôzne veľkých súborov protokolom SR. Pri úlohe preneste tri rôzne veľké súbory jednotlivými

metódami. Ako transportný protokol bude použitý UDP.

Nastavenia simulácie:

1. počet odoslaných správ bude definovaný na základe súboru
2. stratovosť siete: 0 %
3. časovač opakovaného prenosu: 7 s
4. doba prenosu správy medzi uzlami (oneskorenie): 0,5 s
5. počet bitov pre sekvenčné číslo: 5
6. veľkosť okna: 10
7. počet prijatých správ na jedno potvrdenie postupne: 1
8. veľkosť dátovej časti bunky: 140 B

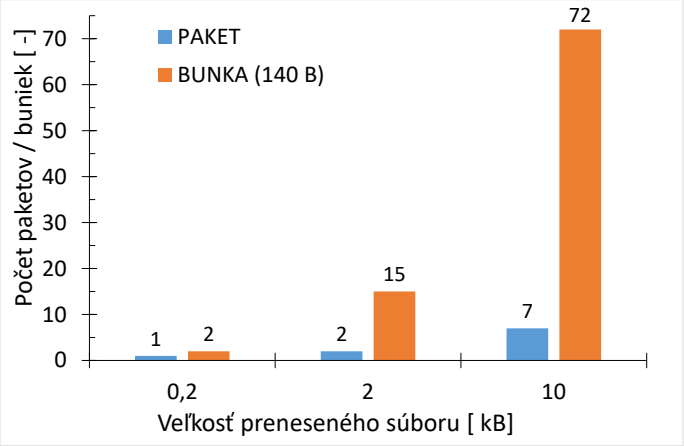
Priebeh simulácie a porovnanie počtu dátových jednotiek

Pred jednotlivými simuláciami budú vygenerované textové súbory s veľkosťami 200 B, 2 kilobajty a 10 kilobajtov. Do aplikácie pomocou globálnych nastavení nahráme prvý súbor, rovnako ako to bolo popísané v predchádzajúcej úlohe.

Aká je maximálna veľkosť dátovej časti paketu pri použití UDP protokolu?
Ako to bolo spomenuté v predchádzajúcej úlohe maximálna veľkosť dát IP paketu je 1480 B. Použitý UDP protokol má následne 8 bajtovú hlavičku a k nej bude aplikáciou vložené označenie aktuálnej správy. Identifikátor bude zložený s 6 bajtov alokovaných pre text a určitého počtu bajtov pre samotné číslo. Zo zadania je určený počet bitov pre dané číslo na hodnotu 5. To znamená, že maximálne sekvenčné číslo bude mať binárny tvar 11111, po prevode do desiatkovej sústavy hodnota odpovedá číslu 31. Pre najväčšie číslo dostaneme nasledujúci formát označenia dátovej správy „[SEQ:31]“. Celková dĺžka označenia bude následne 8 bajtov. Výsledný objem dát uložených v pakete bude môžme určiť ako $1480 - 8 - 8 = 1464$ B. Každým paketom budeme môcť preniesť až 1464 bajtov. Zvolený súbor obsahuje 200 bajtov, ktoré je možné vložiť do 1 paketu.

Pri komutáciách buniek bola zvolený veľkosť dátovej časti bunky na hodnotu 140 bajtov. Potrebný počet buniek na prenos súboru získame z výpočtu $200/140 = 1,43$, hodnotu následne zaokrúhlime nahor a súbor bude rozdeliť na dve bunky. Prvá bude zaplnená s 140 B textového súboru a ostatných 60 B je prenesených v druhej bunke spolu s výplňou.

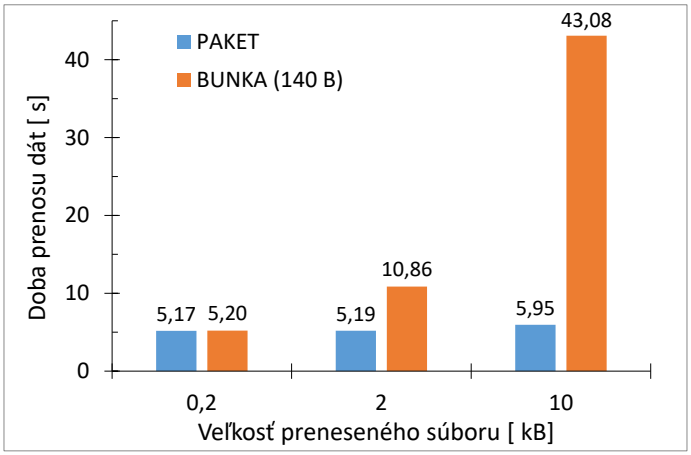
Rovnaký postup opakujeme postupne pre ostatné súbory. Na prenos 2 000 B dát bude potrebné 2 pakety a 15 buniek. Pri prenose posledného súboru bude potrebných 7 paketov a 72 buniek. Rozdiel jednotlivých postupov pri prenose dát je možné pozorovať v grafe 9.7 so zaznamenanými výsledkami.



Obr. 9.7: Závislosť veľkosti súboru na počte odoslaných dátových jednotiek pri komutácií paketov a buniek

Porovnanie a zhodnotenie výsledkov

So zväčšujúcou sa veľkosťou dát počet buniek rastie mnohonásobne rýchlejšie ako počet paketov. Pri prenose posledného súboru bol tento nárast približne 928,57 %. Nastavením konštantného oneskorenia jednotlivých uzlov na hodnotu 0,5 s získavame hodnoty zaznamenané následne v tabuľke 9.8.



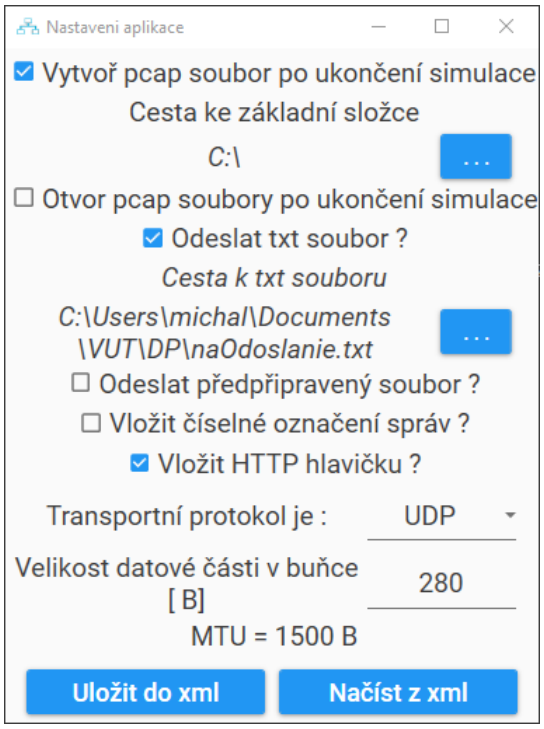
Obr. 9.8: Závislosť veľkosti súboru na celkovej dobe prenosu pri komutácii paketov a buniek, použitím SR protokolu s veľkosťou okna 10

Ako sa zmenila doba prenosu pri jednotlivých simuláciách?
Z výsledkov je možné sledovať nárast doby prenosu buniek postupne polovicu a následne pri prenose posledného súboru stúpla doba prenosu približne o 75 %. Pri

prenose paketov vznikol nárast o približne 0,4 % a následne o 14,64 %. Tieto hodnoty závisia od viacerých parametrov ako je napríklad zvolená veľkosť okna pri SR. Na dobu prenosu má vplyv rovnako tak doba spracovania jednotlivých paketov/buniek sieťovými uzlami. V programe je táto hodnota konštantná pri obidvoch typoch komutácií. V praxi sa daný údaj môže výrazne odlišovať.

9.1.4 Zmena dátovej časti bunky a jej závislosť na celkovom oneskorení

V predchádzajúcej úlohe bola zvolená veľkosť dátovej časti bunky na hodnotu 140 B. Pri prenose súboru o veľkosti 10 kilobajtov bolo potrebných celkom 72 buniek. Pri simulácií postupne upravujte túto hodnotu aby bola dosiahnutý čo najmenšia doba prenosu. Následne bude v grafe zobrazený pomer užitočných dát textového súboru k celkovému počtu odoslaných bajtov. Všetky ostatné nastavenia aplikácie sú rovnaké ako v predchádzajúcej úlohe. Veľkosť bunky budeme upravovať v okne globálnych nastavení (obr. 9.9). V zobrazenom okne označíme vytvorenie súboru pcap a odoslanie textového súboru. Ako transportný protokol bude zvolený UDP.



Obr. 9.9: Zmena veľkosti dátovej časti bunky na hodnotu 280 B v globálnych nastaveniach aplikácie

Zmena veľkosti bunky

Ako ovplyvňuje zmena veľkosti bunky dobu prenosu dát?
Z predchádzajúcej simulácie bola určená doba prenosu na hodnotu 43,08 s pri prenose 72 buniek. Každá bunka preniesla celkom 140 bajtov dát. Zníženie doby prenosu je možné zväčšením veľkosti dátovej časti bunky na dvojnásobok.

P.č.	Čas	Zdroj	Cieľ	Protokol	Info
1	16:35:24,3…	172.10.10.126	172.40.10.1	UDP	8710 → 80 Len=288
2	16:35:24,4…	172.10.10.126	172.40.10.1	UDP	8710 → 80 Len=288
3	16:35:24,6…	172.10.10.126	172.40.10.1	UDP	8710 → 80 Len=288
4	16:35:24,7…	172.10.10.126	172.40.10.1	UDP	8710 → 80 Len=288

Obr. 9.10: Časť komunikácie stanice PC-B pri komutáciách buniek s veľkosťou 280 B

Veľkosť dátovej časti bunky pri prvej simulácií je 280 B. Ostatné nastavenia niesu pre simuláciu dôležité. Po ukončení simulácie overíme nastavenie veľkosti bunky z vygenerovaného súboru stanice PC-B. Prenesené dáta sú zobrazené na obr. 9.10, kde je možné vidieť prenos piatich správ klienta s IP adresou 172.10.10.126 na server s adresou 172.40.10.1 protokolom UDP. Každý záznam (obr. 9.10) má konštantnú veľkosť 288 bajtov, z ktorých musíme odčítať označenie jednotlivých buniek s veľkosťou 8 bajtov. Výsledná hodnota 280 B zodpovedá nastavenej hodnote v programe. Z zachytených správ sme určili celkovú dobu prenosu, ktorá je približne 21,64 sekúnd. Hodnota poklesla o polovicu, ako aj počet odoslaných správ. Postup opakujeme s veľkosťou bunky 1000 B a následne s hodnotou blízkou veľkosti paketu 1400 B.

Zväčšením dátovej časti bunky na 1400 B bola dosiahnutá približne rovnaká doba prenosu. V porovnaní s komutáciou paketov musela byť posledná bunka vyplnená hodnotami a tak vznikla nadbytočná zátaž.

Záver

Diplomová práca sa zaoberá spôsobom komunikácie v globálnej sieti. V práci boli popísané základné princípy prenosu dát ako sú komutácie správ, okruhov, paketov a buniek. Následne bola analyzovaná metóda ARQ. Nasadením protokolu možno zabezpečiť prenos a doručiť všetky odoslané dáta. V praxi sa dajú použiť pri plnom duplexnom spojení protokoly s posuvným oknom: GBN (Go-back-N) alebo SR (Selective repeat). Ak medzi koncovými účastníkmi nie je možné zaistiť duplexné spojenie, ale iba polovičný duplex, potom je vhodné použiť protokol SW (Stop-and-Wati). Jednotlivé znalosti spomenutých protokolov a komutácii boli spracované a implementované do výslednej desktopovej aplikácie. Program bol zostavený použitím objektovo orientovaným programovacím jazykom C#.

V aplikácií sú využité dva simulačné scenáre. Prvý scenár slúži pre simuláciu prenosu dát s použitím ARQ mechanizmu medzi klientom a serverom. Druhý scenár obsahuje topológiu, v ktorej sú jednotlivé správy prenášané cez uzly virtuálnej siete. Prenos týchto správ je realizovaný pomocou komutácie paketov alebo buniek s nastaviteľnou veľkosťou.

V programe sa dajú následne upraviť viaceré sieťové parameter, ako napríklad oneskorenie, stratovosť alebo celkový počet správ, ktoré budú odoslané. Tieto správy sa podľa potreby môžu uložiť do súboru a neskôr analyzovať externým programom Wireshark. Jednotlivé funkcie, ako aj samotné scenáre boli popísané formou úloh s podrobným postupom vypracovania.

Výstupom jednotlivých simulácií sú súbory s komunikáciou sieťových prvkov, ktoré sa dajú zobraziť v programe Wireshark. Možno skonštatovať, že sa podarilo zostaviť simulátor virtuálnej siete schopný simulovať jednotlivé ARQ protokoly a rôzne typy komutácii.

Literatúra

[1] ALANI, Mohammed M. Guide to OSI and TCP/IP models. Cham: Springer, [2014], 50s. ISBN 978-3-319-05151-2. [cit. 25. 11. 2019]. Dostupné z URL: https://link.springer.com/book/10.1007%2F978-3-319-05152-9

[2] BRICKNER, Boaz. *Pcap.Net - the open-source: .NET wrapper for WinPcap.* In: *Github* [online]. 2009 [cit. 2020-05-29]. Dostupné z URL: https://github.com/PcapDotNet/Pcap.Net

[3] CHRISTENSSON, Per. *Bitrate Definition.* In: *TechTerms* [online]. Sharpened Productions, 2016 [cit. 1. 12. 2019]. Dostupné z URL: https://techterms.com/definition/bitrate.

[4] CHRISTENSSON, Per. *Jitter Definition.* In: *TechTerms* [online]. Sharpened Productions, 2019 [cit. 1. 12. 2019]. Dostupné z URL: https://techterms.com/definition/jitter.

[5] *Communication networks: fundamental concepts and key architectures.* 2nd ed. Boston: McGraw-Hill, 2004, s. 286-287. ISBN 0071198482.

[6] Doporučenie RFC 768: *User Datagram Protocol Internet Standard* 28 August 1980, [cit. 28. 5. 2020]. Dostupné z URL: https://tools.ietf.org/html/rfc768.

[7] Doporučenie RFC 791: *Internet protocol Darpa Internet Program* september 1981, [cit. 28. 5. 2020]. Dostupné z URL: https://tools.ietf.org/html/rfc791.

[8] Doporučenie RFC 793: *Transmission control protocol Darpa Internet Program* september 1981, [cit. 20. 10. 2019]. Dostupné z URL: https://tools.ietf.org/html/rfc793#section-3.3.

[9] Doporučenie RFC 3366: *Advice to link designers on link Automatic Repeat reQuest (ARQ)* august 2002, [cit. 1. 11. 2019]. Dostupné z URL: https://tools.ietf.org/html/rfc3366.

[10] DOSTÁLEK, Libor a Alena KABELOVÁ. *Velký průvodce protokoly TCP/IP a systémem DNS.* 2. aktualiz. vyd. Praha: ComputerPress, 2000. Komunikace & sítě. ISBN 80-7226-323-4.

[11] HASAN, Osman a Sofiene TAHAR. *Performance Analysis of ARQ Protocols using a Theorem Prover.* ISPASS 2008 - IEEE International

Symposium on Performance Analysis of Systems and software. IEEE, 2008, s. 85-94. DOI: 10.1109/ISPASS.2008.4510741. ISBN 978-1-4244-2232-6, [cit. 19. 11. 2019]. Dostupné z URL: http://ieeexplore.ieee.org/document/4510741/.

[12] *Introduction to the C# Language and the .NET Framework* [online]. docs.microsoft: Microsoft, 2015, 20.7. 2015 [cit. 25. 11. 2019]. Dostupné z URL: https://docs.microsoft.com/en-gb/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[13] JEŘÁBEK, J. *Komunikační technologie*. Skripta FEKT Vysoké učení technické v Brne, 2019. s. 1-175, ISBN 978-80-214-4713-4.

[14] KESLER, J., *Simulačný program pre protokoly* SR a GBN 2012, [cit. 29. 10. 2019] Dostupné z URL: http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

[15] KOTON, J., *Moderní síťové technologie*. Skripta FEKT Vysoké učení technické v Brne, 2014. s. 1-191, ISBN 978-80-214-5026-4.

[16] LAMMLE, Todd. *CCNA: výukový průvodce přípravou na zkoušku 640-802*. Brno: Computer Press, 2010. ISBN 978-80-251-2359-1.

[17] OZEN, M.S., *Go-Back-N Simulator*. [cit. 29. 10. 2019] Dostupné z URL: https://www.codeproject.com/Articles/14531/Go-Back-N-Simulator

[18] SHATLEY, M., HOFFMAN, Ch. , *Simulačný program* GBN , [cit. 29. 11. 2019] Dostupné z URL: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/go-back-n-protocol/index.html

[19] SHATLEY, M., HOFFMAN, Ch., *Simulačný program* SR , [cit. 29. 11. 2019] Dostupné z URL: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html

[20] WILLOCK, James, Mulholland Software a spol. *Material Design In XAML Toolkit*. In: *Github* [online]. 2020 [cit. 2020-05-29]. Dostupné z URL: https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit/blob/master/LICENSE

[21] ZORZI, M., R.R. RAO a L.B. MILSTEIN. ARQ error control for fading mobile radio channels. *IEEE Transactions on Vehicular Technology.* **46**(2), 445-455.

DOI: 10.1109/25.580783. ISSN 00189545, [cit. 29. 10. 2019]. Dostupné z URL: http://ieeexplore.ieee.org/document/580783/.

Zoznam symbolov, veličín a skratiek

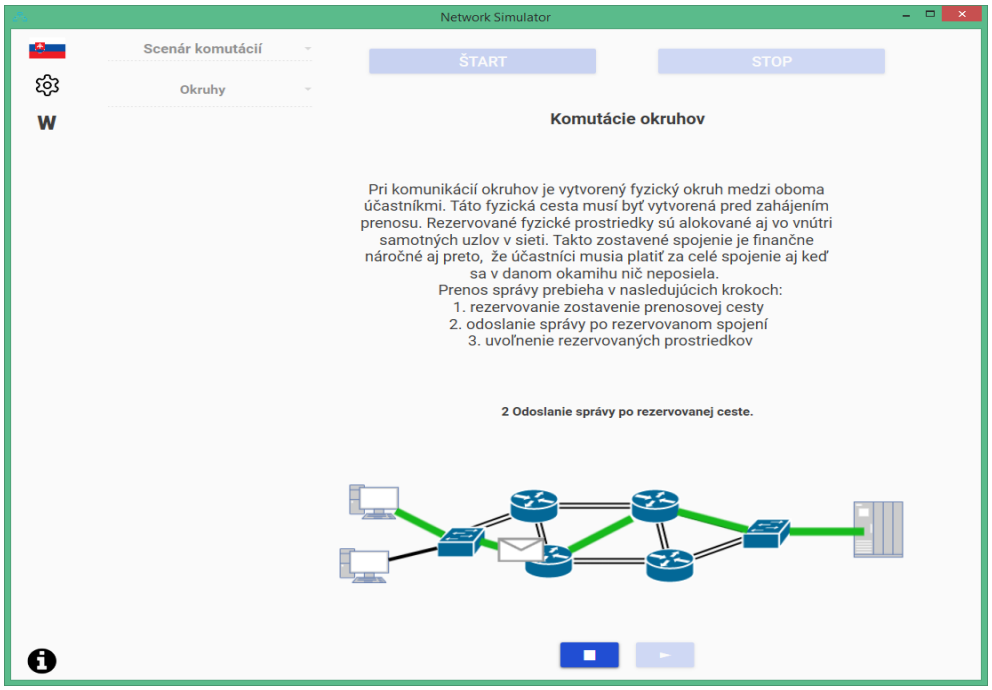
ACK	Kladné potvrdenie (Acknowledgement)
ARQ	Spetná väzba s automatickým opakovaním (Automatic repeat request)
ATM	Asynchrónny transportný mód (Asynchronous Transfer Mode)
B	bajt
FCS	Kontrolný súčet (Frame check sequence)
f_s	vzorkovací kmitočet
FTP	Protokol na prenos súborov (File transfer protocol)
GBN	Metóda Go-back-N
HTTP	Hypertextový prenosový protokol (Hypertext transfer protocol)
HTTPS	Zabezpečený hypertextový prenosový protokol (Hypertext transfer protocol secured)
IP	Internetový protokol (Internet protocol)
ISO/OSI	Referenčný model ISO/OSI
l	dĺžka trasy
m	počet bitov sekvenčného čísla
MAC	Jedinečné identifikačné číslo sieťového adaptéra (Media access control)
n	dĺžka rámca
NACK	Záporné potvrdenie (Negative Acknowledgement)
n_s	počet vzoriek v rámci
PDU	Dátová jednotka protokolu (protocol data unit)
REJ	Zamietnutie (Reject)
RTT	Obojsmerné zdržanie (round-trip time)
RR	Príjemca pripravený (Receiver ready)
SEQ	Sekvenčné číslo (Sequence number)
SMTP	Jednoduchý protokol na prenos pošty (Simple Mail Transfer Protocol)
SR	Protokol Selective repeat
SW	Protokol Stop-and-Wait
TCP/IP	Referenčný model TCP/IP
TCP	Protokol riadenia prenosu (Transmission Control Protocol)
t_c	celkový čas šírenia signálu médiom
t_f	dĺžka trvania rámca
UDP	Používateľský datagramový protokol (User datagram protocol)
W	veľkosť okna ARQ protokolu

Zoznam príloh

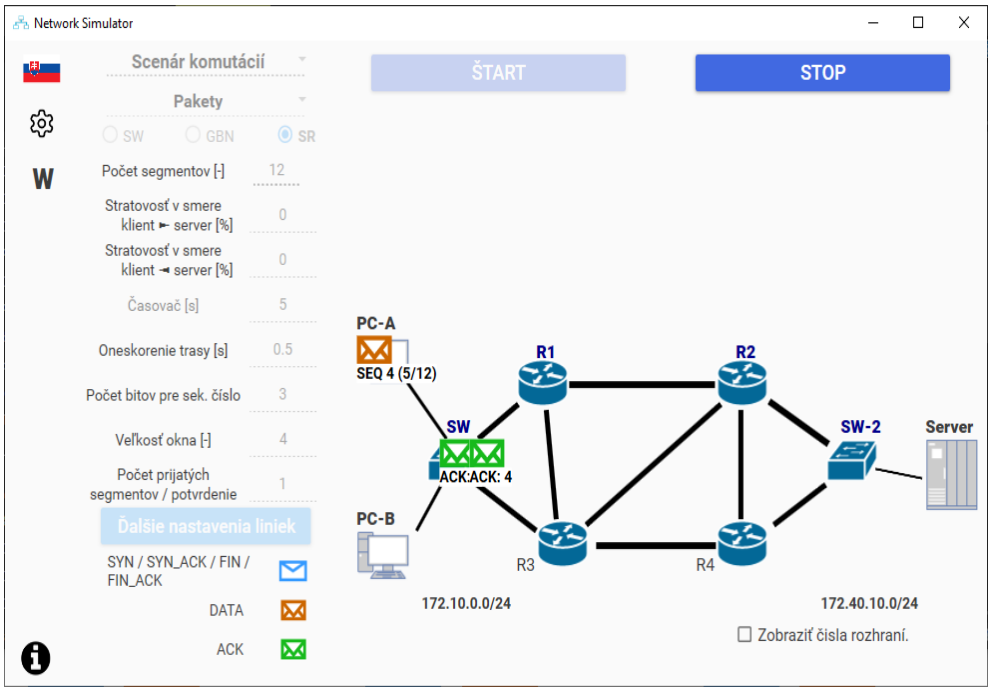
A Obrázky aplikácie

A.1 Grafické rozhranie aplikácie

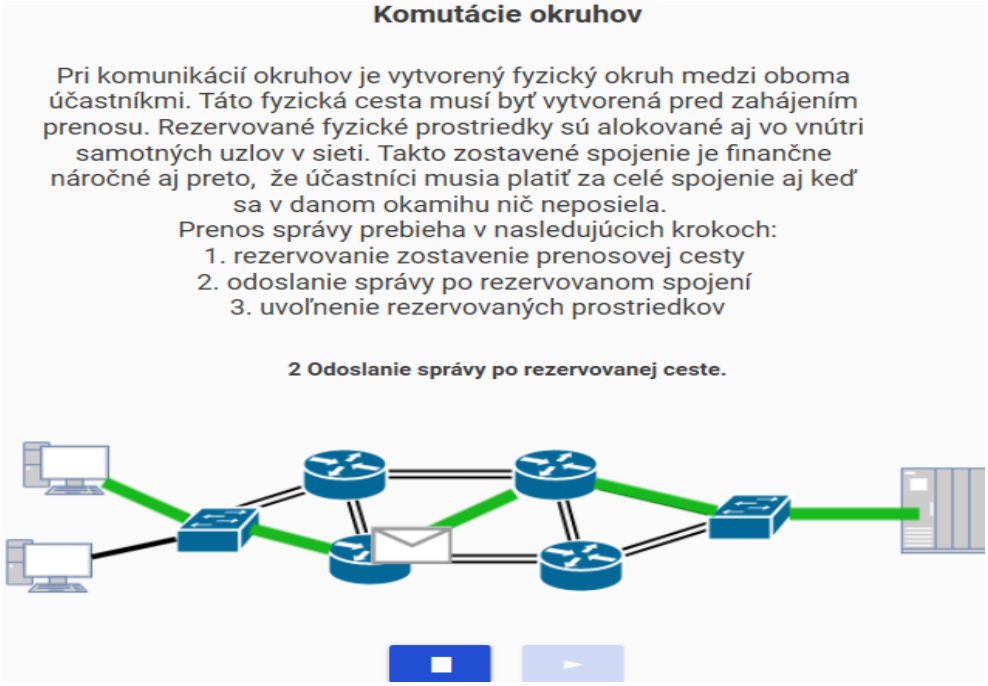
Obr. A.1: Základné grafické rozhranie aplikácie pre simulovanie ARQ mechanizmu



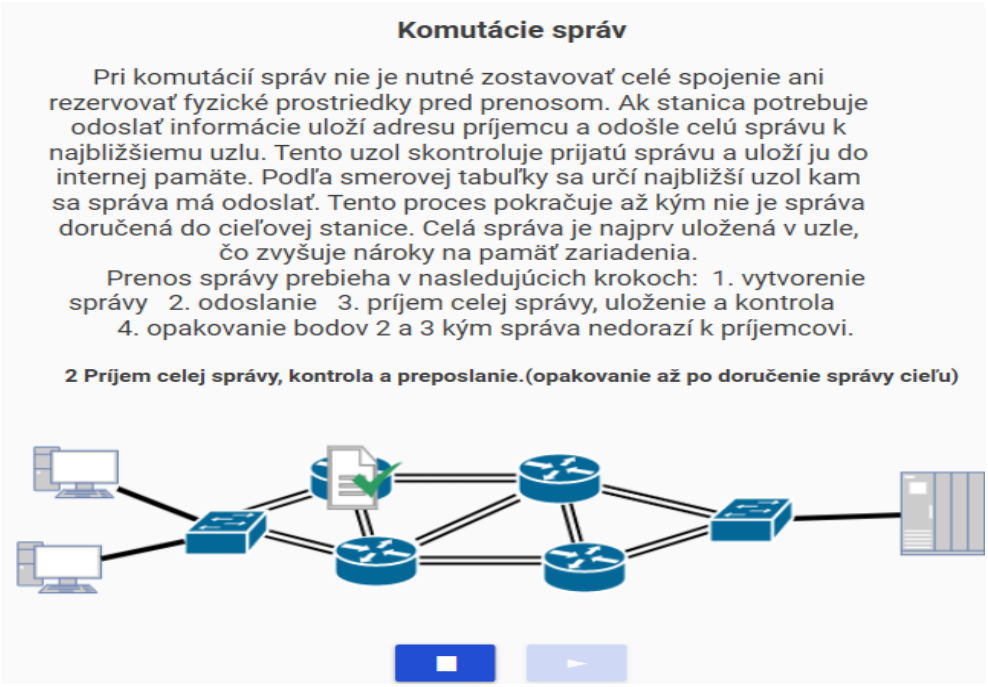
Obr. A.2: Základné grafické rozhranie aplikácie pri zvolenom scenári komutácií okruhov



Obr. A.3: Základné grafické rozhranie aplikácie pri zvolenom scenári komutácií paketov s zobrazenými správami, zobrazené po spustení simulácie



Obr. A.4: Časť grafického rozhrania pre simulovanie komutácií okruhov



Obr. A.5: Časť grafického rozhrania pre simulovanie komutácií správ

B Ukážky kódou

Výpis B.1: Rozhranie „IMessenger“

```
1 public interface IMessenger
2 { // trieda pre komunikáciu logickej časti a GUI
3 // registracia na správy v Messengeri
4     void Register<TMessage>(Action<TMessage> action);
5 // odregistrovanie primu správ
6     void UnRegister<TMessage>(Action<TMessage> action);
7 // odoslanie spravy
8     void Send<TMessage>(TMessage message);
9 }
```

Výpis B.2: Základné nastavenia simulovanej siete logickej časti „NetworkSimulatorLib.dll“, trieda „NetworkParametersModel“

```
1 public class NetworkParametersModel
2 { //trieda obsahujúca sietové parametre simulácie
3 // aktuálne nastavený simulačný scenár
4     public NetworkSimulatorScenarios
          NetworkSimulatorScenario {get; set;}
5 // zvolený ARQ protokol
6     public ArqProtocols ArqProtocol {get; set;}
7 // zvolený typ komutácii
8     public ComutationTypes ComutationType {get; set;}
9 // zvolený typ transportného protokolu TCP/UDP
10    public TransportProtocols TransportProtocol {get; set;}
11 // počet segmentov prenesených počas simulácie
12    public int NumberOfSegments {get; set;}
13 // stratovosť dátových segmentov smere od klienta
14    public double DatatLossPercentage {get; set;}
15 // stratovosť potvrdení smere od servra
16    public double AckDatatLossPercentage{get; set;}
17 // velkosť časovača opakovaného prenosu
18   public double RepeatTransferTimer{get; set;}
19 // jednosmerné oneskorenie medzi dvoma uzlami
20    public double Delay{get; set;}
21 // počet bitov sekvenčného čísla
22    public int NumOfBitsForSeqNumbers{get; set;}
23 // velkosť okna
```

```
24      public int WindowSize{get; set;}
25  // počet prijatých správ po ktorých bude odoslané potvrdenie
26      public int SendAckAfterNpdus{get; set;}
27    ...
28  }
```

Výpis B.3: Model dátovej jednotky, trieda „ProtocolDataUnitModel“, prenášanej počas simulácie

```
1   public class ProtocolDataUnitModel
2    { // dátová jednotka založená na TCP/IP
3   // prenesená počas simulácie
4   // 2. vrstva zdrojová/cielová MAC
5      public string SourceMAC {get; set;} = String.Empty;
6      public string DestinationMAC {get; set;} = String.Empty;
7   // 3. vrstva zdrojová/cielová IP
8      public string SourceIP {get; set;} = String.Empty;
9      public string DestinationIP {get; set;} = String.Empty;
10  // 4. vrstva zdrojový/cielový port
11     public ushort SourcePORT {get; set;} = 0;
12     public ushort DestinationPORT {get; set;} = 1;
13  // protokol transportnej vrstvy
14     public TransportProtocols TransportProtocol {get; set;} =
          TransportProtocols.TCP;
15     public TcpControlBits ControlBit {get; set;} =
          TcpControlBits.Synchronize;
16  // sekvenčné čísla
17     public uint SequenceNo {get; set;} = 0;
18     public uint AckNo {get; set;} = 0;
19     public ushort WindowSize {get; set;} = 1
20  // dáta aplikačnej vrstvy
21     public PduPayloadModel PayloadData {get; set;}
22  // čas odoslania dátovej jednotky
23     public DateTime PduTimestamp {get; set;} = DateTime.Now;
24  // označenie stratenej dátovej jednotky
25     public bool IsLost {get; set;} = false;
26  // ccelková dlžka jednotky v bajtoch
27     public int ByteLength {get; set;} = 58;
28  // interný identifikátor
29     public int PduId {get; set;}
30   ...
```

C Obsah priloženého CD

- / ..... koreňový adresár priloženého CD
  - Marcin_DP.pdf ..... elektornická verzia diplomovej práce
  - Zdroj_textu_DP.zip ..... zdrojové dokumenty pre diplomovú prácu
  - ARQ_ulohy_cj.pdf ..... laboratórna úloha ARQ mechanizmu v čj.
  - Komutacie_ulohy_cj.pdf ..... laboratórna úloha komutácii v čj.
  - ARQ_Komutacie_cj.docx ..... laboratorne úlohy v čj
  - Zdrojové_kódy ..... koreňový adresár s zdrojovými súbormi
    - NetworkSimulator.sln ..... spustiteľný „Solution“ vo Visual Studiou
    - packages ..... adresár s prídavnými balíčkami programu
    - NetworkSimulator ..... adresár s súbormi aplikačnej časti programu
    - NetworkSimulatorLib ..... adresár s súbormi logickej časti programu
  - NetworkSimulator_bin ..... binárne a spustiteľné súbory
    - NetworkSimulator.exe ..... spustiteľný súbor programu
    - NetworkSimulatorLib.dll
    - MaterialDesignColors.dll
    - MaterialDesignThemes.Wpf.dll
    - MaterialDesignColors_license.txt
    - PcapDotNet.Base.dll
    - PcapDotNet.Packets.dll
    - PcapDotNet_license.txt
    - System.Windows.Interactivity.dll