Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Porovnávání síťových simulátorů

Plzeň 2012 Richard Malínský

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 10. května 2012

Richard Malínský

Abstract

This thesis deals with comparison of different network simulation tools. It is an overview of the most used and the most interesting network simulators currently available at the software market. It shows differencies and tries to make objective evaluation. Further it gives manual for qualified using these tools, from downloading and installing to creating sample network simulation. Choosing an appropriate simulator can be very hard as a lot of tools are available and the familiarisation with a simulator is very timeconsuming. So usually a researcher selects one tool and then uses it for all network research studies.

Obsah

1 Úvod

Cílem této práce je seznámit čtenáře s nejpoužívanějšími, popřípadě nejzajímavějšími síťovými simulátory dostupnými momentálně na softwarovém trhu, provést jejich porovnání, zjistit rozdíly mezi nimi a provést vyhodnocení. Dále pak poskytnout návod na použití nástroje od nalezení produktu, instalaci až po vytvoření vzorové síťové simulace.

Simulace se staly nepostradatelnou metodou při tvorbě počítačových sítí. Současné síťové technologie jsou vylepšovány, vznikají nové protokoly, můžeme upravovat, testovat a nastavovat stávající algoritmy pro potřeby současných sítí jako je vícesměrové vysílání, bezpečnost, mobilita, kvalita služeb a řízení přístupů. Díky stálé potřebě růstu počítačových sítí je možné s přispěním zvyšujícího se výkonu hardwarových prvků tvořit rychlejší, přesnější a složitější síťové topologie. A k tomu jsou nám užitečným pomocníkem síťové simulátory jako nástroje pro jejich testování.

Stejně tak jako u zrodu celosvětové počítačové sítě Internet stála i za vznikem, používáním a rozšířením simulací armáda. Proto je také v odborné literatuře mnoho příkladů simulací vztažených právě k tomuto odvětví. Poté simulace rychle pronikly do ostatních oborů jako je obchod, zábavní průmysl, vzdělávání, telekomunikace, návrhy hardwarových logických obvodů nebo doprava.

Při zkoumání chování sítí, vývoji nových síťových protokolů a testování nových postupů nelze vždy pracovat na fyzických zařízeních a je třeba použít simulátory. Důvodem je častá neproveditelnost testovacích scénářů v reálném prostředí, z důvodu vysokých finančních nákladů, společně s mobilitou a umístěním testovaných objektů. Navíc většina měření není opakovatelná a vyžaduje vysokou spolehlivost. Bohužel ve většině případů není jednoduché se nějaký simulátor naučit ovládat a nastavovat, a to je také důvod, proč tvůrci a uživatelé používají pouze jeden vybraný simulátor.

V praxi se setkáváme s již vytvořenými a aktivně využívanými počítačovými sítěmi, takže simulací je v tomto případě využíváno pro modifikaci stávajícího stavu a testování ke zjištění možného nárůstu výkonu při souběžném snižování nákladů. Simulátory umožňují řešit složité problémy, které nelze řešit běžnými analytickými metodami. Díky možnosti nastavení rychlosti simulace můžeme velmi efektivně procházet jednotlivé fáze simulace a zaměřit se tak na místa s očekávaným kritickým chováním. Zpomalení simulace slouží pro zkoumání detailů průběhu simulace zvláště při vyšetřování kritických míst síťového provozu a opačně zrychlení simulace slouží pro celkový náhled na chování systému v poměrně krátkém časovém úseku, což je častější případ využití simulace.

Důležitým faktorem je finanční úspora, která je spatřována v možnosti provedení různých variant řešení a zamítnutí neužitečných modifikací a pokusů na fyzické síti. Samozřejmě náklady na vytvoření simulace nesmí převýšit náklady na analýzu reálného systému. Při vytvoření dostatečně obecného řešení problému je možné opětovné použití simulace u podobných problémů v příbuzných oborech.

2 Základní pojmy

Pro zjednodušení a sjednocení oborové komunikace je potřeba upřesnit základní pojmy týkající se problematiky síťových simulací a distribuovaných systémů. Dále se předpokládá základní znalost síťového provozu a protokolů.

Systém - Skupina počítačů a periférií, navzájem propojených v síti [Havlenka(1997)]

Model - V oblasti počítačů jde o matematickou nebo grafickou napodobeninu objektů reálného světa. Modely mohou sloužit jednak k simulaci některých jevů v abnormálních nebo dlouhodobých podmínkách, jednak jako jeden ze způsobů vizuální kontroly některých fyzicky neexistujících objektů (stavby, předměty). [Havlenka(1997)]

Referenční model ISO OSI (norma ISO 7498) - Definice sedmivrstvého modelu pro počítače komunikující v síti. Vrstvy jsou řazeny podle vztahu k systému a k uživateli – nejnižší vrstvy pracují na hardwarové úrovni a nejvyšší pak maximálně komunikují s uživatelem. [Havlenka(1997)]

Distribuovaný systém - systém, kde je práce rozdělena mezi více počítačů v síti. [Havlenka(1997)]

Simulace - Napodobení procesu nebo objektu pomocí matematického popisu. Simulace umožňuje pomocí změny vstupních nebo jiných podmínek zkoumat změny a varianty chování objektu a předpovídat tak jeho reálnou činnost. Počítač je ideálním prostředkem pro provádění simulací vzhledem k tomu, že simulování reálných dějů vyžaduje množství ovlivňujících faktorů a jejich neurčitosti obrovskou výpočetní sílu. [Havlenka(1997)]

Emulace - Napodobování chodu programu nebo zařízení na prostředcích neodpovídajících přesně technické specifikaci napodobovaného zařízení. [Havlenka(1997)]

Výstup - Výsledek prakticky jakékoli operace, který je nějakým způsobem dán najevo – může být zobrazen na obrazovce, vytištěn na tiskárně či uložen do datového souboru. [Havlenka(1997)]

Událost - množina konkrétních vstupů následně zpracovaných simulátorem

3 Distribuované systémy a simulace

Systémy, jejichž výpočetní výkon je dislokovaný na různá místa, neboli je distribuován. V různých oblastech, jako je např. obchod, služby, bankovnictví, lékařství jsou určitým způsobem použity navzájem komunikující počítače, programovatelné automaty nebo jednočipové procesory a činnost celého systému je ovlivněna fungováním a vzájemnou komunikací jeho jednotlivých částí.

Každá oblast, kde se distribuované systémy používají, klade jiné nároky na výkonnost použité komunikační sítě, na rychlost odezvy, na množství přenášených dat a bezporuchovost přenosu. Distribuované systémy poskytují zdroje více uživatelům v různých sítích a mezi jejich základní vlastnosti patří[Hamilton (1996)]:

- Mnohonásobné snížení času při získávání výsledků simulace při použití více výpočetních jednotek.

- Jednotlivé počítače distribuovaného systému jsou často geograficky oddělené. Simulace zajistí šetření nákladů na přepravu osob a vybavení do příslušných destinací

- Možnost použití více výpočetních jednotek zajistí kontinuitu simulace i při případném výpadku některé z těchto jednotek, kdy si zbylé jednotky jsou schopny přerozdělit práci za nefunkční jednotku.

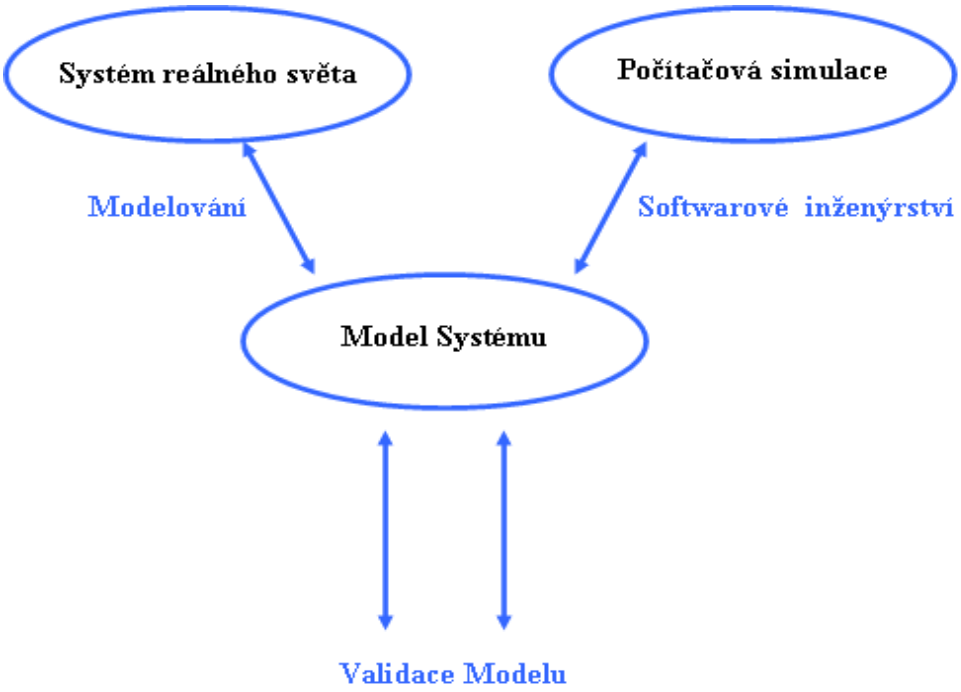
- Oddělení fyzických a logických úrovní mezi skupinami strojů, procesů, informací a uživatelů.

- Schopnost přidávat a přizpůsobovat různorodé hardwarové zdroje.

V distribuovaném systému tedy probíhá distribuované zpracování informací. Na zpracování výpočtu se podílí několik výpočetních jednotek, procesorů. Každý procesor zpracovává vlastní data a ta jsou v případě potřeby zasílána jiným procesorům pomocí zpráv. Počítač zapojený do distribuovaného systému je nazýván distribuovaný počítač. Pro jednoduchost budeme uvažovat počítač jako jednu výpočetní jednotku, s vlastní operační pamětí [Fujimoto (2000)].

3.1 Simulace, její úroveň a míra aproximace s reálným systémem

Z definice víme, že simulace je napodobení chování jiného systému v čase. Mluvíme-li o simulacích, které provádějí počítače, jedná se o počítačové simulace. Ty se staly důležitou součástí zkoumání a studiu nejrůznějších systémů. Obrázek 1 ukazuje schéma vstupů a výstupů při vytváření modelu.



Obrázek 1 Schéma vstupů a výstupů do/ze simulačního modelu

Simulace dělíme podle způsobu průběhu v čase. K tomu je třeba definovat pojem simulační čas. Simulační čas je čas, který je použitý pro průběh činnosti uvnitř simulátoru. Tento čas je možné nastavovat a měnit tak rychlost simulace. **Spojitá simulace** je taková, kdy čas běží rovnoměrně, ve stejných časových intervalech a v každém ukončeném intervalu je zjišťováno, zda a k jaké změně došlo. Oproti tomu **diskrétní simulace** se vyznačuje tím, že ke změnám dochází pouze na základě nějaké události bez možnosti předpovědi, kdy tato událost nastane [Fujimoto (2000)].

Simulátory modelují svět specifickým způsobem. Jejich smyslem je usnadnit chápání problému, případně jej zjednodušit a pozorovat chování a reakci na konkrétní události.

Jak již bylo řečeno, cílem simulace je vytvořit co možná nejreálnější situaci pro získání co nejuspokojivějšího výsledku. Jak tedy dosáhnout co největší přesnosti popisu reálného problému a tím získat co nejlepší výsledky simulace použitelné pro existující systém. Ve složitějších případech je to nelehký až nesplnitelný úkol, neboť nejsme schopni obsáhnout všechny vstupní informace. Proto musí být tato data filtrována a „zjednodušována“.

3.2 Otázka úrovně detailů

Úrovní detailů rozumíme dosažitelnou shodnost simulace s reálnou situací. Samozřejmě nám jde o co největší přiblížení se k realitě. V tomto ohledu je třeba zhodnotit, zda zvolená úroveň detailů pokrývá řešení problému a výsledky simulace jsou stále ještě použitelné a přenositelné na zkoumaný systém.

Na jedné straně je nedostatečný počet vstupů příčinou špatných výsledků a na straně druhé může být příliš veliký počet detailů ve výsledku nevyužitelný, neboť je potřeba hodnotit velké množství vstupů na úkor přehlednosti, nehledě na neúměrně dlouhý čas.

Tento problém je známý jako stavová exploze (angl. state space explosion problem). Jinými slovy jde o to, že je potřeba procházet různými stavy simulace při současném zaznamenávání těchto stavů. Změníme-li velikost simulovaného modelu, zvětšíme složitost, a tudíž i stavový prostoru modelu. Velikost stavového prostoru pak roste exponenciálně s velikostí simulovaného modelu a rostou i paměťové nároky na uchovávání stavů.

Obrovské množství stavů způsobuje nepřehlednost. Abychom této nepřehlednosti zamezili, musíme přijmout příslušná opatření. Můžeme využít dva přístupy k řešení problému. Jedním je použití úspornější prezentace výstupů stavových prostor nebo získávání jen výstupů bezprostředně souvisejících se zkoumanou částí systému. Druhým způsobem je zajištění vyššího výpočetního výkonu a operační paměti. Případně lze samozřejmě kombinovat oba přístupy. Optimalizace omezování stavového prostoru je navržena pro úzkou třídu problémů, což umožňuje lepší účinnost a vyšší efektivitu za cenu omezení rozsahu nasazení.

Rozlišujeme pět základních optimalizačních technik [Clacke1999]. **Kompozitní přístup** je způsob ověřování systému, který je složen ze samostatných komponent, popřípadě komponent s minimálním počtem vazeb. Ověřujeme, že zkoumaná vlastnost je obsažena v konkrétní komponentě, a pokud ano, je tedy i součástí celku. Čím je vzájemných vazeb mezi komponenty méně, tím je použití tohoto přístupu výhodnější. **Abstrakce** jsou dalším způsobem redukce stavového prostoru a používají se již před vytvořením vlastního modelu. Princip spočívá ve zjišťování vlivu proměnných na zkoumanou vlastnost a její případné odstranění, nebo pomocí datové abstrakce nalezneme a vhodně nahradíme skupinu základních datových typů proměnných abstraktním datovým typem proměnné. Ta bude vyjadřovat souhrnný stav těchto základních datových typů. Výsledný stavový prostor bude podstatně menší oproti původnímu stavu. Použití **Symetrie** přichází v úvahu v situacích, kdy máme opakující se komponenty (paměti, soubory registrů nebo komunikační protokoly). Tyto komponenty mají stejné přechodové funkce stavů. Vytvořením ekvivalentní funkce pro celou skupinu možných stavů výrazně zmenšíme celkovou velikost stavového prostoru. **Symbolický** model checking spočívá v nahrazení množiny stavů specifickou symbolickou reprezentací. Tento způsob je hojně využíván pro ověřování hardwarových systémů. **On-the-fly** technika trochu vybočuje z řady přístupů. Technika spočívá ve spuštění generování stavů a současné ověřování zájmové vlastnosti s tím, že jakmile se podaří existenci vlastnosti systému potvrdit (nebo vyvrátit), je generování stavů ukončeno. Analýza stavů s využitím vyššího výpočetního výkonu je založena na souběžném zpracovávání, generování a vyhodnocování jednotlivých stavů.

V praxi postupujeme tak, že provádíme postupně jednotlivé kroky, kdy na základním modelu simulace získáme a ověříme platnost omezeného množství dat a poté přidáme další požadovaná vstupní data. Všechno závisí na schopnosti simulace tato data zpracovávat. S rostoucím objemem dat roste také výpočetní náročnost, schopnost ověřování a jejich následná implementace [Hamilton (1996)].

3.3 Postup od záměru k realizaci simulačního modelu

Rozhodneme-li se při řešení problému sáhnout po simulaci, je vhodné se držet definované osnovy, neboť proces tvorby simulace může být velice komplikovaný, obzvláště potřebujeme-li použít velké množství komponent. Postup pro tvorbu simulace můžeme rozdělit do několika přehledných fází.

1. **Definování a analýza problému** – znamená jasnou formulaci problému s přesně definovanými hranicemi, stanovení cílů a časového rozvrhu. Definice hranice úzce souvisí se zvolenou rozlišovací úrovní – úrovní detailů. Během vlastní simulace se ve většině případů dojde k závěru, že je potřeba rozlišení upravit. Co se týče důkladné analýzy problému, je tato nezbytná pro co nejhladší průběh dalších fází realizace simulace, neboť je pevným základem pro zajištění kvalitních výstupů při očekávaném chování systému. Důležitý je zde sběr dat pro zajištění kvalitní analýzy. Při této analýze můžeme dojít k závěru, že je třeba rozlišovací úroveň změnit [Hamilton (1996)]:

   Pro zvýšení rozlišovací úrovně se rozhodneme pokud

   - je aktuální úroveň simulačního modelu příliš abstraktní.
   - potřebujeme vysoké rozlišení pro jeden konkrétní speciální proces
   - potřebujeme stanovit hranice pro parametrickou analýzu (př. počet pokusů o doručení paketu)
   - budeme potřebovat kalibraci rozlišení - poznáním, že mám znalosti světa na všech úrovních detailů

   Pro snížení rozlišovací úrovně se rozhodneme pokud

   - vyžadujeme rychlou analýzu alternativních směrů.
   - potřebujeme obecnější přehled nad rozlišovací úrovní detailní simulace
   - nám dynamická změna rozlišení poskytuje flexibilitu v modelování a analýze a může zlepšit výpočetní efektivitu simulace.

   Nesmíme ovšem zapomínat, že případné zvyšování úrovně detailů je výpočetně drahé a narůstá také množství dat pro analýzu, proto se jako praktické jeví prostudování pouze části síťového provozu a ostatní provoz nechat běžet na nižším rozlišení.

2. **Sestavení simulačního modelu –** Samozřejmostí je mít znalosti o funkčnosti systému v teoretické rovině a aplikovat příslušná zjednodušení a manipulaci s daty (strukturami) vyplývající z analýzy. Z pohledu simulací sítí je základní jednotkou simulačního modelu jeden segment sítě, tzn. použití protokolů fyzické a linkové vrstvy referenčního modelu ISO/OSI. Pro modelování segmentů sítě je zapotřebí zapojit i další vrstvy jako je síťová a transportní vrstva. S přibývajícími požadavky je nutné zvážit použití i dalších vrstev. Důležitým úkolem je výběr vhodného softwarového prostředí.

3. **Validace a verifikace modelu –** určení rozdílů mezi simulačním modelem a modelovaným reálným systémem a odhalení chyb modelu, což je základní

předpoklad pro zodpovězení otázky zda model umožňuje získávat smysluplné výsledky.

4. **Návrh experimentu (přesnost, náklady) –** příprava jednotlivých simulačních scénářů, experimentálních údajů pro navržený model s cílem prověřit jeho správnost při implementaci.

5. **Provádění experimentu –** jde o sledování modelu v čase, kdy na vstupu máme vhodně připravenou sadu počátečních podmínek pro tento model.

6. **Vyhodnocení výsledků a postupů -** analýza jednotlivých experimentů s ohledem na zadání řešeného problému a jejich statistické zpracování.

7. **Realizace -** k úspěšnému zvládnutí všech fází procesu simulace je potřeba znát mnoho IT oborů: softwarové inženýrství, problematika počítačových sítí, tvorba designu, stejně tak porozumění prostředí, ve kterém se má systém vytvářet. Realizace je posledním krokem, kdy jsme provedli dostatečné množství experimentů a uspokojivě vyhodnotili jejich výsledky a je možné tyto výstupy přenést do reálného prostředí.

4 Definice srovnávacích kritérií simulátorů

V této kapitole je vytvořen seznam kritérií, jejichž existence bude zkoumána při porovnávání simulátorů, specializujících se na modelování počítačových sítí.

Licence – popis způsobu distribuce softwaru

Uživatelská přívětivost instalace – jednoduchost při získávání programu a jeho přípravě k použití

Náročnost použití nástroje – kvalita návrhu uživatelského rozhraní programu

Požadavek na použití zdrojového kódu při budování simulované sítě – potřeba znalosti programovacího jazyka při tvorbě simulace

Vizualizace topologie sítě – grafické zpracování simulace

Rozšiřitelnost síťových komponent – možnost tvorby a definice vlastních modelovacích prvků

Rozšiřitelnost dostupných protokolů – možnosti tvorby vlastních protokolů

Vizualizace toku dat – zpracování průběžných výsledků simulace do grafické podoby

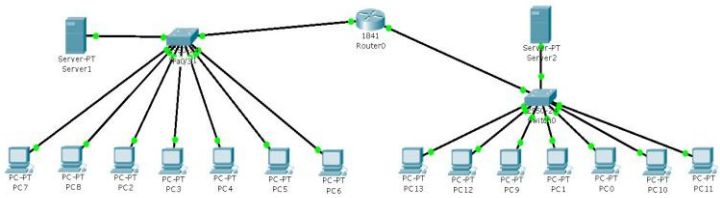
Propojení simulované sítě do reálného prostředí – možnost napojení simulované sítě do reálného prostředí

Zdrojové kódy – možnost získání zdrojových kódů programu pro vlastní úpravu

Operační systémy – typy systémů podporujících simulátor

Lokalizace produktu – národní jazykové mutace pro použití programu

Pro demonstraci použití jednotlivých simulátorů, popisu možností a zachycení rozdílů mezi nimi byla předem nadefinována vzorová laboratorní počítačová síť. Z pohledu fyzického umístění se jedná o dvě kanceláře v budově, kde bude propojení síťovými prvky realizováno následujícím způsobem. Základním zařízením bude směrovač jakožto propojovací prvek do veřejné sítě. K tomuto směrovači budou připojeny dva přepínače, pro každou kancelář jeden, a na tyto přepínače budou připojena koncová zařízení – osobní počítače a servery v počtu osmi zařízení na každém přepínači. Na obrázku 3 je tato síť vyobrazena.



Obrázek 2 Vzorová počítačová síť

5 Porovnání simulačních nástrojů

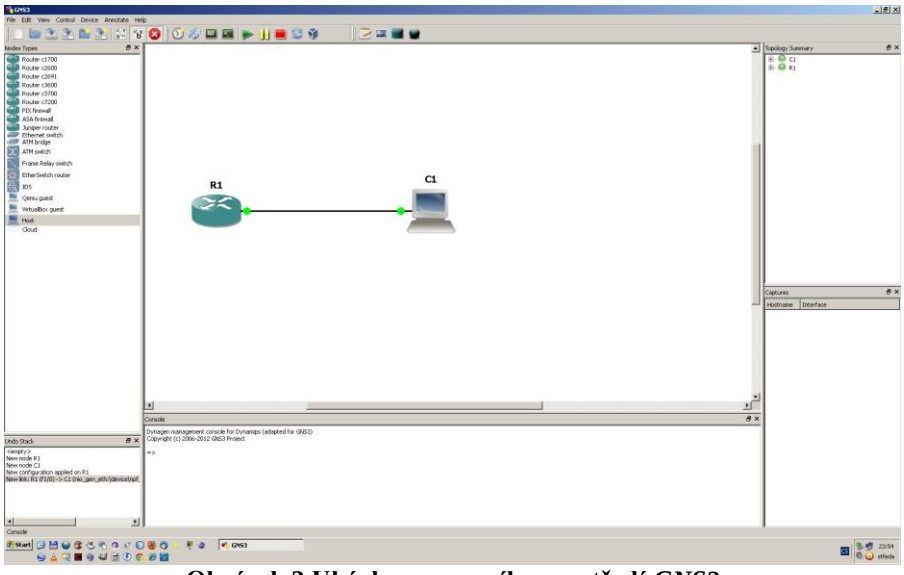
5.1 GNS3

GNS3 je grafický síťový simulátor umožňující vytvářet rozlehlé sítě. Je schopen emulovat operační systém široké škály směrovačů a PIX firewallů firmy Cisco, hlavního poskytovatele síťových prvků na světě. *GNS3* je grafická nadstavba programu Dynagen, který běží nad jádrem simulačního programu Dynamips v konzolovém prostředí. Tento simulátor je možné provozovat jak v systému Windows, na uniových systémech, tak i na MacOS X. Program je poskytován zdarma a je vyvíjen pro laboratorní účely jako testovací a výukové prostředí. Používá veliké množství protokolů a příkazů. Jediným omezením je nutnost vlastnit operační systém firmy Cisco.

GNS3 umožňuje simulaci Ethernetu, ATM a Frame Relay přepínačů, emulaci mnoha vývojových platforem, spojení simulované a reálné sítě, zachytávání paketů pomocí programu Wireshark. Ten je možné získat na webové adrese: http://www.wireshark.org.

5.1.1 Instalace softwaru

GNS3 lze použít jak v operačních systémech Windows tak i Linux. Prvním krokem k získání tohoto simulátoru je přístup na webovou stránku http://www-gns3.net a přímo na hlavní stránce kliknout na zelené tlačítko Download. Poté se zobrazí volba pro výběr balíčku ke stažení. Po stažení požadované verze pro systém Windows, nejlépe té se suffixem all-in-one dojde ke stažení spouštěcího souboru instalace. Program je dodáván v 32bitové i 64bitové verzi. Po spuštění instalace se stačí držet průvodce instalací a potvrzovat nabízené možnosti, k úspěšnému nainstalování programu od odsouhlasení licenčního ujednání, výběru umístění programu a instalaci pomocné knihovny pro zachytávání paketů přes tlačítka Agree, Next a Install k dokončení instalace - tlačítko Finish. Během instalace dojde k dotazu instalování programu Wincap (knihovna pro zachytávání paketů na síťovém rozhraní), což stačí pouze potvrdit. Nyní je aplikace připravena k použití.



Obrázek 3 Ukázka pracovního prostředí *GNS3*

5.1.2 *Prvotní nastavení*

Prvním krokem po spuštění je zvolení cesty k obrazu operačního systému příslušného síťového zařízení. Tu nastavíme v menu Edit - IOS image and hypervisors. Zde si celkem intuitivně přidáme potřebný IOS, v následující nabídce vybereme model přidávaného zařízení, který se zobrazí v seznamu. Změny uložíme a vrátíme se na hlavní obrazovku. Nyní jsme připraveni na vytvoření naší první síťové topologie.

5.1.3 *Začínáme*

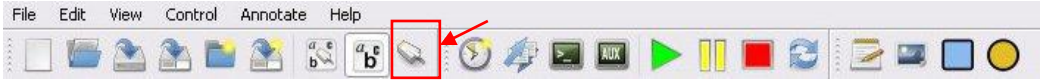
Začneme umístěním routeru R1 do pracovního okna, jeho spuštěním a nastavením konfigurace v konsoli. Pracovní okno se nachází přímo uprostřed aplikace a je obklopeno z levé strany oknem s výběrem dostupných síťových zařízení, z pravé strany seznamem zařízení v naší topologii a indikací jejich stavu a zespoda oknem konsole, které informuje o činnosti programu Dynagen. Přidání routeru do pracovního okna se provede prostým přetažením mezi okny. Pravým tlačítkem klikneme na vložený router a z nabídky vybereme volbu Configure. Ze seznamu po levé straně vybereme náš router a poté přidáme potřebná rozhraní. Vrátíme se do pracovního prostředí a přes pravé tlačítko myši spustíme Start. Tím jsme router aktivovali.

Nyní provedeme konfiguraci routeru, pravým kliknutím zvolíme Console, čímž spustíme klienta Telnet. Po dotazu pro vstup do inicializačního okna napíšeme No a stiskneme Enter. V tomto okamžiku musíme z důvodu úspory operačního času procesoru nastavit opět přes pravé tlačítko myši volbu IdlePC. Program automaticky spočte a doporučí nejlepší hodnoty pro optimální rozdělení zatížení procesoru. Hodnota preferovaná programem je zvýrazněna hvězdičkou. Vypočtené hodnoty jsou uloženy na stejné kartě jako IOS obrazy, tedy IOS images and hypervisors. Kontrolu zatížení procesoru můžeme ověřit ve Správci úloh systému. Program emuluje činnost procesorů jednotlivých síťových prvků společné s jejich konfigurací.

Po nepříliš složitém úvodním nastavení a zorientování se v nabídkách programu, není obtížné již se samotným programem pracovat, i když se předpokládá znalost práce v prostředí IOS Cisco, kde lze využívat všechny podporované příkazy.

5.1.4 *Vytvoření síťové topologie*

Pro vytvoření složitější sítě zopakujeme předchozí postup přidání zařízení, a poté tato zařízení spojíme. K tomu nám slouží tlačítko Add a link. Dostaneme se do módu označování zdrojového a cílového zařízení. V tomto automatickém módu po označení jednoho a poté i druhého zařízení, dojde k jejich propojení, což je indikováno v pravém okně.



Obrázek 4 Použití tlačítka Add a link z lišty ikon funkcí

Pro hromadné připojení konzole ke směrovačům použijeme tlačítko s obrázkem příkazového řádku. Otevře se okno s klientem Telnet, na dotaz inicializace odpovíme No. A dále nastavujeme dle vlastních požadavků jako v reálném prostředí. Po dokončení konfigurace provedeme test spojení zařízení příkazem Ping.

Pro připojení počítačů do sítě je potřeba stáhnout, nainstalovat a spustit program Virtual PC Simulator, ten podporuje celkem devět virtuálních počítačů a je vhodné jej spustit ještě před programem *GNS3*. Návod k použití lze nalézt na webových stránkách, nicméně jeho použití je velice triviální. Pro přepínání mezi počítači stačí stisknout příslušné číslo a nastavit IP adresu, bránu a masku. (ip 10.0.0.5 10.0.0.1 24)

Pro integraci virtuálního simulátoru je třeba použít knihovnu symbolů (Symbol Library). V hlavním menu vybereme Edit a poté Symbol Manager. V levém okně klikneme na symbol počítače a přidáme pomocí tlačítka šipky do pravého okna.

Po návratu k pracovnímu oknu přetáhneme požadovaný počet počítačů ke směrovačům. Přes pravé tlačítko myši vybereme počítač a stiskneme Configure. V otevřeném okně nastavíme číslo místního portu, vzdáleného portu a vzdáleného počítače (ip 127.0.0.1). Analogicky nastavíme i ostatní počítače. Poté provedeme konfiguraci rozhraní routeru a počítače a zkusíme test spojení.

V případě, že si nevystačíme s poskytnutým množstvím počítačů, je možné použít jako počítač i router. Jen je potřeba jej správně nakonfigurovat. Příklad konfigurace lze najít v tutoriálu.

5.1.5 Propojení simulace s Internetem

Zajímavou vlastností aplikace je možnost připojení vlastní virtuální sítě do prostředí reálného světa, kupříkladu ke spojení virtuálních počítačů běžících ve Virtual PC. Pokud se k tomu kroku rozhodneme, postupujeme podobným způsobem jako při nastavení virtuálních počítačů. Přidáme nový počítač do pracovního okna (musí být definován jako Cloud), v konfiguraci zvolíme záložku NIO Ethernet. Zde vybereme síťový adaptér, který požadujeme. Stiskneme tlačítko Add a poté OK. Nastavíme IP adresu adaptéru, stiskneme Add a link a je hotovo. (Pozn. Alternativou je použití Loopback Adapteru od MS Windows)

5.1.6 Frame Relay Switch a Ethernetový switch

Kromě routeru můžeme použít i další zařízení jako je Frame Relay Switch. Ten je poskytován jako standardní zařízení (bez potřeby operačního systému). Konfiguraci provedeme opět přes volbu Configure a nastavíme port a identifikátor spojení mezi WAN. Po přesunu zařízení do pracovního okna je automaticky aktivní. Podobný postup uplatníme i v případě ATM Switche.

Simulátor nabízí také ethernetový switch se základní funkcí přepínání s podporou virtuální sítě na protokolu 802.1q. Konfigurace nabízí možnost nastavení až do 10000 portů a virtuálních sítí. Můžeme se také připojit k reálné síti a používat příkazy pro zobrazení a vymazání tabulky MAC adres.

Chceme-li používat více funkcí switche, zvolíme vložení routeru a v nastavení rozhraní zvolíme z výběru volbu NM-16ESW. Výsledkem bude například rozhraní pro linkovu vrstvu, switch virtual interface, trunk protokol, spanning tree protokol, port security, flow control.

5.1.7 PIX Firewall

Dalším nabízeným zařízením pro simulace je PIX Firewall, jehož použití rovněž vyžaduje vlastní image, pro neomezený přístup také sériové číslo a aktivační klíč. Pokud toto máme k dispozici, můžeme opět konfigurovat. V preferencích, menu Edit zvolíme Pemu, zkontrolujeme zaškrtnutí políček pro použití manažerů a do pole PIX Image vložíme cestu k našemu obrazu, vložíme klíč, sériové číslo a potvrdíme OK. Aktivační klíč je oddělován čárkami bez mezer a sériové číslo musí být zadáno v hexadecimálním tvaru.

Přetáhneme ikonu PIX Firewall do projektu a nakonfigurujeme. Vybereme obraz, vložíme klíč a číslo a potvrdíme OK. Firewall spustíme příkazem Start. (Firewall disponuje pouze rozhraním pro Ethernet nebo FastEthernet)

Stejně jako u směrovačů i Firewall používá téměř 100 % času procesoru, proto zde použijeme obdobu IdlePc. Příkladem může být program BES (http://mion.faireal.net/BES/), který tuto práci spolehlivě zastane. Použití je intuitivní, stačí vybrat proces, který chceme limitovat, obecně pro každý běžící program, v našem případě postačí qemu.exe.

5.1.8 Ošetření paměťových nároků aplikace

Aplikace je při použití většího množství zařízení velice paměťově náročná.
Z tohoto důvodu je zde zavedena funkce Ghost IOS pro uložení jedné kopie IOS pro identické operační systémy. Pak je zde ještě funkce "sparsemem" pro zmenšení velikosti virtuální paměti spuštěné instance routeru. Pro optimální běh simulátoru je nutné mít obě funkce zapnuté, neboť používají mapování souborů do virtuální paměti mmap.

Při každém bootování routeru se nejdříve musí IOS dekomprimovat a dochází ke zbytečnému zdržení, toto lze obejít a ušetřit tak čas načtením již extrahovaných IOS.

5.1.9 Zachytávání paketů

GNS3 umožňuje sledování cesty paketů mezi jednotlivými rozhraními použitých síťových prvků a výstupy zapisuje do souboru *libpcap,* který je možné číst programem WireShark. (http://www.wireshark.org)

Stačí mít tento síťový analyzér nainstalovaný a poté vybrat zájmovou linku navrženého a aktivního spojení, kliknout pravým tlačítkem myši, dále na volbu Capture a dojde k automatickému spuštění analyzéru. Nastavení pro zachytávání paketů lze nastavit v Preferencích menu Edit záložka Capture. Zde nastavujeme pracovní adresář pro zachytávání a upřesňující spouštěcí příkaz pro WireShark. (V závislosti na rozhraní můžeme zvolit zachytávání protokolů FR, HDLC nebo PPP)

5.1.10 Uložení konfigurace simulací

GNS3 umožňuje nahrávat a ukládat vytvořené síťové topologie do souboru s příponou *net,* v textovém formátu vyhovujícímu programu Dynagen. Můžeme též projekt uložit v grafické podobě ve formátu obrázku s příponou souboru *png* volbou

Snapshots v menu File. V případě potřeby uložení a nahrání nastavení jednotlivých zařízení použijeme příkazy export/import v příkazovém řádku Dynagen.

Veškerá činnost odehrávající se v aplikaci se zobrazuje v příkazovém okně pracovního prostředí (uprostřed dole). Ovšem pro vlastní činnost v simulátoru není potřeba probíhající komunikaci znát. Jen pro představu, zobrazení dostupných příkazů získáme odesláním znaku „?“, nápověda k příkazu „<příkaz> ?“.

Dále poskytuje *GNS3* mnoho dalších ikon zařízení, které ovšem slouží pouze pro dekorativní účely, s možností si další potřebné ikony vytvořit. Doporučené hardwarové požadavky jsou alespoň 2 GB RAM a procesor o taktovací frekvenci 2.5 GHz.

5.1.11 Vytvoření vzorové sítě

Vytvoření vzorové sítě spočívá v přetažení požadovaných ikon zařízení do pracovního prostoru a jejich nastavení. Simulátor nedisponuje možností vložení a konfigurace serveru do prostředí, proto byl ve výsledku nahrazen dalším osobním počítačem. Pro vytvoření osobního počítače je potřeba použít emulátor. Doporučeným emulační programem je virtualpc, který nabízí devět vituálních počítačů. Osm těchto virtuálních počítáčů bylo použito pro vytvoření jedné větve počítačů. Pro vytvoření druhé větvě byl použit obraz unixového systému tinycore-2.11.5 pro jednotlivé počítače. Pro konfiguraci routeru jsou použity základní příkazy nastavení rozhraní, adresy a aktivity linky.

5.1.12 Shrnutí

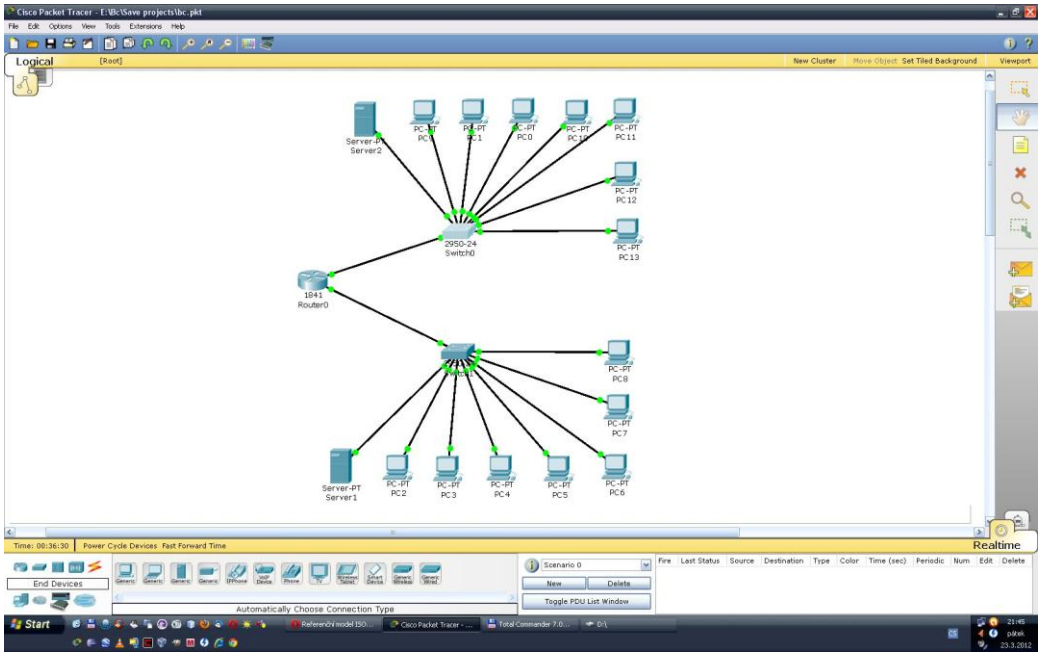
Pro začínající uživatele je zpracován vcelku podrobný manuál pro oba operační systémy a nechybí ani podrobná dokumentace. Použití tohoto simulátoru je vhodné k získání zkušeností se zařízeními síťové komunikace, pro testování a experimentování se softwarem reálných zařízení nebo ověřování správnosti konfigurací zařízení.

Ovládání a práce se simulátorem nečiní větší obtíže a pracovní prostředí je přehledně uspořádáno. Práce programu s operační pamětí počítače je z důvodu náročnosti ošetřena vestavěnými nástroji, přidání a zprovoznění velkého množství síťových prvků je časově náročné z důvodu častého čekání na reakci počítače. Program svým zpracováním nabízí seznámení hlavně se směrovacími protokoly a testováním průchodnosti sítě při nastavených parametrech a nemá možnost přidávání nových protokolů ani programování.

5.2 Cisco Packet Tracer

Cisco Packet Tracer je grafický síťový simulátor firmy Cisco zaměřený především na výuku síťové komunikace, tvorbu simulací, vizualizaci a animaci síťového provozu. Umožňuje navrhovat a konfigurovat počítačové sítě a učí řešit problémy s tím spojené. Simulátor je praktickým nástrojem výukového programu Cisco Networking Academy, na jehož konci je možnost získání světově uznávaných certifikátů CCNET, CCNA a CCNP. Program nabízí širokou škálu síťových protokolů počínaje linkovými přes směrovací, směrované, bezpečnostní až po aplikační.

Program lze nalézt na adrese http://www.cisco.com, po registraci ke vzdělávacímu programu Cisco Networking Academy. Na výběr je dána samotná verze programu nebo verze s tutoriálem.



Obrázek 5 Ukázka pracovního prostředí *Cisco Packet Tracer*

5.2.1 Instalace softwaru a popis prostředí

Po stažení instalačního balíčku a jeho spuštění se otevře průvodce instalací, v jehož dialogovém okně proběhne samotná bezproblémová instalace. Spuštěním programu se dostaneme do základního grafického simulačního rozhraní. To je rozděleno na několik částí.

V horní části jsou umístění dvě lišty, lišta Hlavní Menu s dostupnými příkazy aplikace a pod ní lišta s ikonami pro rychlý přístup k vybraným činnostem.
Pod touto lištou je umístěn přepínač mezi fyzickým a logickým pracovním prostředím a navigační menu. Ve fyzickém pracovním prostředí je možné vytvářet nové lokace – nové budovy, města, místnosti. Ty se dají různě přemístit, nastavit jejich pozadí a přepnout do nového pracovního prostředí. Práce v logickém pracovním prostředí je obdobná s tím, že je možné přecházet mezi místnostmi.

Po pravé straně je umístěna lišta pro práci přímo související s návrhovým prostředím (viz níže). Jsou zde volby pro výběr, přesun, mazání síťových prvků, změnu velikosti prvků, změnu umístění pomocných textů.

Hlavní pracovní okno se nachází uprostřed aplikace a zaujímá největší prostor. Zde probíhá největší část práce se simulátorem. Vytváří se zde simulovaná síť, probíhá její testování a zobrazují se zde pomocné informace a statistiky.

Další ovládací prvek je přepínač mezi prostředím simulace a pracovním oknem. Jedná se o žlutý pruh umístění v dolní části obrazovky. V režimu simulace se zde testuje navržená síť. Obsahuje ovládání pro rychlosti simulace, zobrazují se zde výše uvedené statistiky a zaznamenávají se zde jednotlivé události s ohledem na simulační čas.

Pod tímto pruhem se nachází dvě paralelně zobrazená okna pro výběr síťových prvků, v levém okně je to výběr obecného zařízení (přepínač, směrovač, koncové zařízení) společně s dostupnými druhy vzájemných připojení zařízení, v pravém okně je konkrétní zařízení pro výběr a přetažení na pracovní plochu simulátoru.

Vedle těchto oken po pravé straně je umístěno okno ovládání paketů vložených do simulace. Okno je použitelné po přepnutí do simulačního módu.

Přizpůsobení vzhledu těchto oken je čistě na vůli uživatele programu. Další nastavení nalezneme v hlavním menu v záložce Nastavení – Preference. Jde hlavně o nastavení doplňujících informací o vložených zařízeních, nastavení vlastností zařízení, paketů nebo fyzických spojů. Nalezneme zde nastavení pro změnu lokalizace programu nebo logování událostí, uzamykání zvolených nastavení, skrývání a zobrazování ovládacích prvků, nastavování písma nebo omezení nových spojení po dosažení zadané hodnoty.

5.2.2 Vytvoření síťové topologie

Pro demonstraci základních funkcí programu použijeme spojení mezi koncovými zařízeními, a to osobním počítačem a serverem. V listu zařízení zvolíme End Devices a na pracovní plochu přetáhneme postupně Generic PC a Generic Server. Poté se přepneme do záložky Connections a vybereme ikonu položeného blesku pro automatické připojení spojovacím kabelem a spojíme jím obě zařízení. Zelené světlo na koncích spojů značí funkční spojení, červená barva značí opak. Kontrolu spojení můžeme provést také umístěním kurzoru nad vybrané zařízení a zobrazí se nám stav linky "Up" pro aktivní spojení. Nyní se poklepáním na ikonu osobního počítače dostaneme do konfiguračního nastavení a vybereme záložku Config. Nastavíme adresu DNS serveru na 192.168.0.1. U položky Interface klikneme na rozhraní FastEthernet a nastavíme IP adresu 192.168.0.102. Program automaticky dopočítá další parametry. Zkontrolujeme, že je zašktrnuto políčko u položky Port Status. Postup opakujeme i pro přidaný server. Otevřeme konfigurační okno, nastavíme rozhraní FastEthernet a IP adresu na 192.168.0.1. Opět zkontrolujeme Port Status. Dále klikneme na DNS a nastavíme doménové jméno na www.firstlab.com, IP adresu na 192.168.0.1 a klikneme na Add. Ujistíme se, že je služba DNS zapnuta a tím je spojení vytvořeno a nakonfigurováno.

Nyní jsme připraveni na test spojení. Klikneme na ikonu obálky se znaménkem plus (Add Simple PDU) a pošleme jednorázový test spojení pomocí příkazu ping směrem k serveru. Server odpoví a zašle potvrzovací zprávu. Kurzor se změní na obálku, se kterou postupně klikneme na osobní počítač a pak na server. V informační liště na pravé straně spodního panelu můžeme získat informace o stavu a pozici zaslané zprávy. Můžeme také provést test na opačnou stranu, ze serveru na osobní počítač. Testování spojení je zaznamenáváno a automaticky ukládáno jako scénář pro pozdější zopakování testu. Program nám dává možnost vytvořit více testovacích scénářů pro jeden projekt. Stačí jen kliknout na tlačítko New na informačním panelu v jeho levé části a zvolit nový postup testování.

Program nám umožňuje prohlížení jednotlivých fází simulace a jejich animaci. Po odeslání zprávy se přepneme do Simulation Mode, klikneme na Edit Filters a vybereme z nabízených protokolů jen na ICMP, pro zobrazení pouze těchto paketů. Spuštěním simulace tlačítkem Play můžeme sledovat průběh konkrétní simulace, kdy je současně zobrazen stav paketu v textové podobě s podrobnými informacemi a v pracovním okně animované zobrazení s označením úspěšnosti přenosu zelenou

"fajfkou" nebo červeným křížkem. Aktuálně probíhající přenos je v textové části označen obrázkem oka. Přidání dalších zpráv pro posílání je zařazeno do fronty zpráv.

Pokud by nás zajímala cesta paketu, je sledována z pohledu sedmivrstvého referenčního modelu ISO/OSI, můžeme kliknout na ikonu obálky v pracovním okně nebo aktuální záznam v textové tabulce. Otevře se nám okno s tabulkou zmíněného referenčního modelu a zobrazením, na jaké vrstvě se paket momentálně nachází. Mezi jednotlivými vrstvami se můžeme pohybovat tlačítky Next/Previous Layer a zkoumat zpracovávání paketu na jejich úrovni. Přepneme-li se do další záložky v tomto okně, získáme detailní informace o vnitřní struktuře uložení informací v paketu.

Další informace o průběhu komunikace mezi těmito zařízeními lze sledovat povolením protokolu ARP v simulačním módu, ve výše zmíněném Edit Filters. Naplnění tabulek ARP uvidíme v pracovním okně po kliknutí nástrojem Inspect (ikona Lupy) z pravého svislého menu na vybrané zařízení. Vymazání tabulek ARP provedeme kliknutím na Power Cycle Devices, který provede vypnutí všech zařízení a tím i reset jejich konfigurací.

5.2.3 Vytvoření vzorové sítě

Vytvoření vzorové sítě spočívá v přetáhnutí ikon příslušných zařízení do pracovního prostředí, jejich konfigurace v základním rozsahu jako u reálných zařízení, připojení propojovacími kabely, spuštěním a otestováním průchodnosti paketů přes jednotlivá zařízení. Pro nastavení síťových prvků je nutné samozřejmě znát technologický postup společně s dostupnými příkazy konfigurace. Konfigurace zařízení je výhodné mít předem uloženy a poté je do nastavovaného zařízení pouze přehrát. Tím se výrazně zkrátí čas na vytvoření a zprovoznění sítě.

5.2.4 Shrnutí

Práce v tomto simulátoru je zajímavá, prostředí nabízí mnoho možností nastavení jak systému, tak samotné simulace a práce je zpříjemněna pěkně zpracovaným prostředím s důrazem na jednoduchost a intuitivnost činností. Simulátor je dodáván v konečné podobě, což znamená, že nelze vytvářet vlastní rozšíření a vylepšení prostředí ani přidávání protokolů, zato se mu ale dostává podpora ze strany vývojářů. Od předchozího simulátoru se liší zejména tím, že používá integrované nástroje na přenos dat a neumožňuje připojit vytvořenou síť k reálnému zařízení.

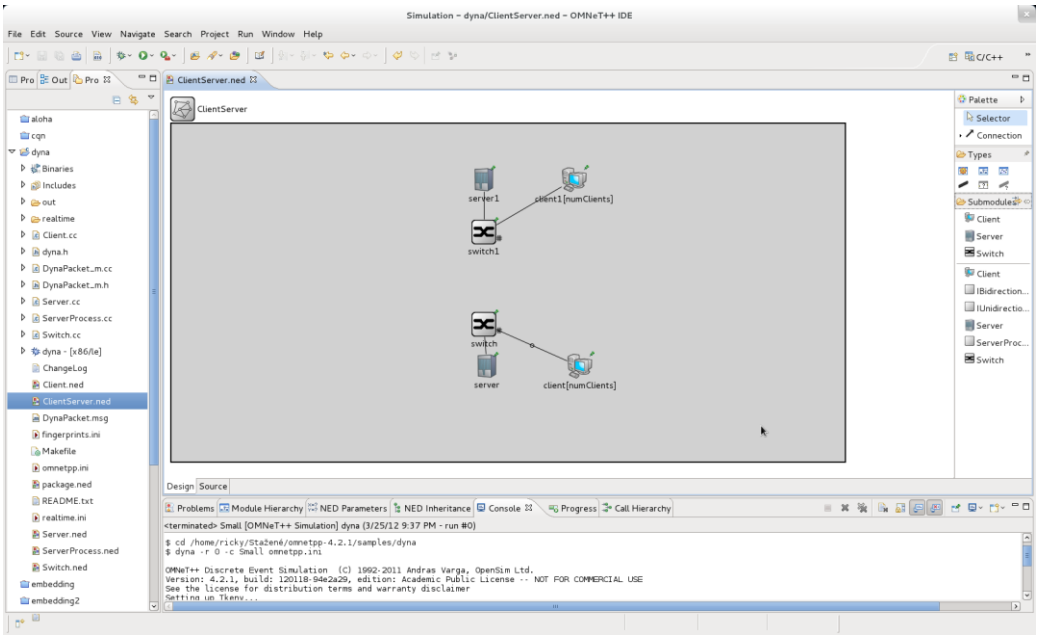
Program je dodáván s kvalitně zpracovaným tutoriálem v podobě webových stránek a ukázkových videí, a je určen pro testování a výuku scénářů předkládaných při skládání mezinárodně uznávaných certifikátů.

5.3 Omnetpp

Omnetpp je opensource grafický simulátor využívající diskrétní simulaci objektů, založený na komponentách jazyka C++. Hlavní použití spočívá v modelování počítačových sítí, drátový i bezdrátových, modelování ad-hoc sítí, síťových protokolů. Svým univerzálním zpracováním umožňuje simulace nejen počítačových sítí, ale i jiných oblastí, jako například dopravní infrastrukturu nebo multiprocesory. V případě použití pro nekomerční a akademické účely je poskytován bezplatně na

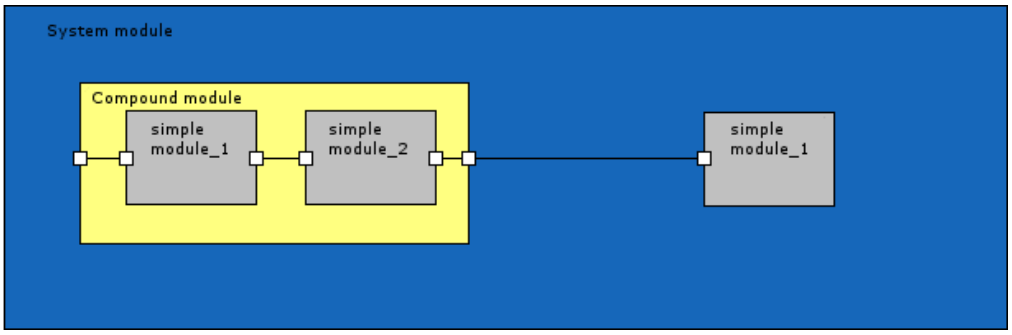
http://www.omnetpp.org/. V komerční verzi jej lze najít pod názvem Omnest. K simulátoru je možné připojit podpůrné programy pro snadnější užívání. Jde hlavně o rozšiřující moduly s přidanými protokoly a grafická vylepšení.

Program pracuje pod operačním systémem Windows i pod unixovými systémy.



Obrázek 6 Ukázka vývojového prostředí *Omnetpp*

Prostředí simulátoru je organizováno do modulů, které se do sebe hierarchicky vkládají bez omezení hloubky vnoření. Základní moduly se nazývají Simple modules, dva a více základních modulů tvoří submoduly a ze submodulů jsou tvořeny systémové moduly. Tímto způsobem je možné vytvořit přehlednou strukturu simulačního modelu.

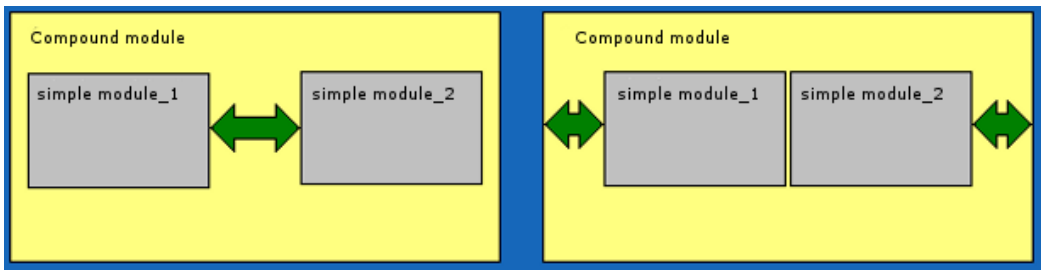


Obrázek 7 Vzor systémového modulu

Program poskytuje návrhové vzory modulů, které velice dobře slouží k usnadnění a urychlení práce v prostředí. Struktura modulu je popisována pomocí jazyka NED

5.3.1 *Komunikace modulů*

Jednotlivé moduly mezi sebou komunikují pomocí zpráv. Po přijetí zprávy provede modul požadovanou činnost jako například odpověď na přijatou zprávu nebo požadovanou činnost. V případě síťových simulací jsou zprávy reprezentovány rámci nebo datovými pakety. Způsob posílání zpráv je realizován pomocí cílovému modulu nebo definováním cest přes brány. Jednotlivé brány chápeme jako vstupně-výstupní rozhraní modulů. Cesta je potom konkrétní spojení jednotlivých bran. Cestám neboli spojením lze nastavovat různé parametry jako je rychlost, zpoždění nebo chybovost. Jejich použití je nepovinné. Nastavení parametrů je možné pro každé jednotlivé spojení nebo z předem nadefinovaných šablon spojení a tyto pak použít pro konkrétní případ. Nastavené parametry můžeme najít v globálním souboru *omnetpp.ini* nebo NED souboru pro moduly.



Obrázek 8 Vzájemná komunikace modulů

5.3.2 *Jazyk NED*

Jazyk NED (Network description) se používá pro popis struktury modulů. Umožňuje deklarovat základní moduly, propojovat je a stavět do složitějších struktur. Tyto struktury je možno označit jako "sítě", jako samostatné simulační modely. NED nabízí několik vlastností umožňující vystavět složitější moduly.
Jsou to zejména:

Hierarchie - umožňuje zmíněnou výstavbu modulů od jednodušších ke složitějším

Použití komponent - zajišťuje opakované použití modulů a tvorbu modulových knihoven. Příkladem je rozšiřující modul INET Framework obsahující moduly pro simulaci protokolů IP, TCP, UDP, Ethernet, MPLS, LDS. Pro bezdrátové sítě se využívá modul Mobility Framework.

Rozhraní - je možno použít jako zástupce modulu nebo kanálu, které se nastavuje při startu simulace.

Dědičnost - umožňuje přidávat nové parametry, vlastnosti a spojení do stávajících modulů.

Balíčky - podpora dělení modulů do balíčků pro odstranění rizika kolizí jmen modulů, pro specifikaci závislostí mezi moduly slouží NEDPATH (viz CLASSPATH z jazyka Java)

Vnitřní typy - obdoba použití lokálních proměnných, v tomto případě lokálních modulových typů

Metadata - popis vlastností modulů, kanálů, parametrů, bran. Jedná se o doplňující (pomocné) informace pro různé nástroje, prostředí i moduly, zejména pro grafické výstupy.

Příklad jazyka NED definujícího síť se šesti uzly a jejich vzájemným spojením:

```
network Network
{
    submodules:
        tic[6]: Txc10;
    connections:
        tic[0].out++ --> {delay = 100 ms;} -->  tic[1].in++;
        tic[0].in++  <-- {delay = 100 ms;} <--  tic[1].out++;

        tic[0].out++ --> {delay = 100 ms;} -->  tic[2].in++;
        tic[0].in++  <-- {delay = 100 ms;} <--  tic[2].out++;

        tic[1].out++ --> {delay = 100 ms;} -->  tic[2].in++;
        tic[1].in++  <-- {delay = 100 ms;} <--  tic[2].out++;

        tic[1].out++ --> {delay = 100 ms;} -->  tic[4].in++;
        tic[1].in++  <-- {delay = 100 ms;} <--  tic[4].out++;

        tic[3].out++ --> {delay = 100 ms;} -->  tic[4].in++;
        tic[3].in++  <-- {delay = 100 ms;} <--  tic[4].out++;

        tic[4].out++ --> {delay = 100 ms;} -->  tic[5].in++;
        tic[4].in++  <-- {delay = 100 ms;} <--  tic[5].out++;
}
```

5.3.3 Základní modul

Je bázový modul pro veškeré stavební činnosti při budování sítě. Je aktivním prvkem simulace. V jeho definici je uveden název, parametry a komunikační brány. Příkladem základního modulu je:

```
simple Module
{
    parameters:
        @display("i=block/routing");
    gates:
        input in[];
        output out[];
}
```

5.3.4 Kanál

Nastavení parametrů přenosového kanálu je volitelné, slouží k vytvoření spojení s přednastavenými hodnotami. Máme možnost nastavovat tři parametry, a to zpožďovací kanál, který nastavuje hodnotu zpoždění průchodu zprávy, a dále pak chybu přenosu simulující rušení kanálu a přenosový parametr určující přenosovou rychlost kanálu v bitech za sekundu, kdy pomocí šířky pásma vypočítává dobu přenosu paketu. Pokud je parametr nastaven na nulu, je uvažována nekonečná šířka pásma.

5.3.5 Simulace v Omnet++

Základem simulace je soubor NED popisující topologii sítě a soubor *omnetpp.ini* nastavující vlastnosti jednotlivých modulů. Pokud by byl soubor *omnetpp.ini* prázdný, musely by se všechny parametry modulů zadávat ručně po spuštění simulace.

5.3.6 Popis vývojového prostředí Omnet++

Vývojové prostředí pro simulace je podobné jako u většiny dalších IDE, to znamená, že je rozděleno na několik částí. Horní část vyplňuje lišta se všemi dostupnými funkcemi a nástroji programu, pod ní je lišta s ikonami pro urychlení výběru funkcí a nástrojů. V levé části se nachází okno Project Explorer sloužící pro vytváření a editaci souborů a jejich zařazování do přehledné struktury projektu simulace. Pod ním je okno zobrazující vlastnosti aktuálně vybraného souboru. V prostředním, centrálním okně probíhá tvorba simulace, kdy lze přepínat mezi návrhovým prostředím a prostředím psaním zdrojového kódu. V pravé části aplikace se nachází paleta nástrojů pro tvorbu. Dolní část je vyhrazena záložkám zobrazujícím chyby kompilace simulace, parametry modulů, jejich hierarchii a závislosti, záložku konsole, logování událostí.

Všechna tato okna lze libovolně přesouvat a individuálně upravit tak, aby plně vyhovovala pohodlné práci uživatele.

5.3.7 Instalace prostředí

V současné době je k dispozici verze softwaru 4.2.1, a je možno si vybrat instalaci pro Unix nebo Windows. Původně jsem dal přednost balíčku ve formátu zip pro systém Windows. Během instalace však antivirový program nahlásil nález virové infekce, proto jsem instalaci ukončil a dále pracoval s verzí pro linux, konkrétně s distribucí Fedora 16. Po stažení příslušného archivu tgz je potřeba jej extrahovat do požadované složky. Do příkazového řádku zadáme

```
tar xvfz omnetpp-4.2.1-src.tgz
```

Tento příkaz vytvoří stejnojmennou složku s potřebnými soubory. Pro správný průběh instalace a následné činnosti simulátoru je nutné si nainstalovat přídavné balíčky a aktualizovaný systém.

Aktualizaci povedeme příkazem:

```
su yum update
```

poté získáme vyžadované knihovny a aplikace příkazem:

```
su – c yum install build-essential gcc g++ bison flex perl tcl-dev tk-dev blt libxml2-dev zlib1g-dev openjdk-6-jre doxygen graphviz openviz openmpi-bin libopenmpi-dev libpcap-dev
```

Přesuneme se do složky se simulátorem a do příkazového řádku zadáme

```
. setenv
```

pro nastavení prostředí. Dále nastavíme systémové proměnné v souboru bashrc.

Soubor otevřeme příkazem pro jeho editaci

```
gedit ~/.bashrc
```

a na konec souboru přidáme řádky:

```
export PATH=$PATH:$HOME/omnetpp-4.2.1/bin
export TCL_LIBRARY=/usr/share/tcltk/tcl8.5
```

A můžeme provést konfiguraci a kompilaci:

```
./configure
make
```

O správnosti kompilace jsme informováni v příkazovém řádku. Je-li kompilace úspěšná, jsme vyzváni k zadání příkazu *omnetpp* pro spuštění simulátoru, v opačném případě je zobrazena informace s chybovým hlášením a způsobem nápravy.

Funkčnost programu ověříme nejlépe spuštěním vzorové simulace ve složce samples. Například takto:

```
cd samples/dyna
./dyna
```

5.3.8 Vytvoření síťové topologie

Při vytváření sítě se omezíme na zobrazení dvou uzlů a ukázku jejich vzájemné komunikace. Jeden uzel vytvoří paket a pošle jej druhému uzlu a ten jej pošle zpět. Nejprve si vytvoříme pracovní složku s názvem tiktak a přepneme se do ní. Vytvoříme zde soubor s příponou *tiktak.ned* a popíšeme v něm zamýšlenou topologii následujícím způsobem:

```
// Definice sítě network Tiktak1 pro dvě instance (submodules) představujících uzly (tik a tak) základního modulu Txc1 se vstupem in, výstupem out a vzájemným propojením (connections) a posíláním zpráv
// s nastavením zpoždění (delay)
```

```
simple Txc1
{
    gates:
        input in;
        output out;
}

network Tiktak1
{
    submodules:
        tik: Txc1;
        tak: Txc1;
    connections:
        tik.out --> {  delay = 100ms; } --> tak.in;
        tik.in <-- {  delay = 100ms; } <-- tak.out;
}
```

Nyní přidáme nový soubor *txc1.cc*, který nám zajistím funkcionalitu základního modulu

```
//
#include <string.h>
#include <omnetpp.h>


class Txc1 : public cSimpleModule
{
  protected:
    // Opětovná definice virtuálních funkcí pro provedení algoritmu
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);
};

// Registrace třídy Txc1
Define_Module(Txc1);

void Txc1::initialize()
{
    // Initialize je volána na začátku simulace.
    // Nastavení modulu tik pro první zaslání zprávy

    if (strcmp("tik", getName()) == 0)
    {
        // vytvoření zprávy a její zaslání na výstup out z modulu
        cMessage *msg = new cMessage("tiktakMsg");
        send(msg, "out");
    }
}
```

```
void Txc1::handleMessage(cMessage *msg)
{
    // Metoda handleMessage() method je zavolána po přijetí zprávy
    // Zde je zpráva beze změny odeslána na výstup
    send(msg, "out");
}
```

Dále si vytvoříme soubor *Makefile*, pomocí kterého zkompilujeme a nalinkujeme program pro spuštění. Zadáme do příkazové řádky:

```
opp_makemake
```

V pracovní složce se vytvoří soubor *tiktak*.

Zadání make proběhne kompilace a linkování.

```
make
```

Nyní si vytvoříme soubor *omnetpp.ini* pro nastavení počátečních podmínek simulace a nastavení parametrů, v našem případě jde pouze o tyto dvě řádky:

```
[General]
network = Tiktak1
```

V případě potřeby je možné využít jeden soubor *omnetpp.ini* pro nastavení počátečních podmínek pro více simulací s následující strukturou:

```
[General]
# komentář

[Config Tiktak1]
network = Tiktak1

[Config Tiktak2]
network = Tiktak2
Tiktak2.tak.limit = 5
```

Po dokončení všech předchozích kroků, spustíme grafické prostředí simulace příkazem:

```
./tiktak
```

Objeví se nám simulační okno *Omnetpp*, kde spuštěním tlačítka Run zahájíme již samotnou simulaci. Výsledkem je vizuální výměna zprávy mezi uzly tik a tak.

5.3.9 Vytvoření vzorové sítě

Vytvoření definované vzorové sítě nebylo náročné. Předpřipravené moduly a použití vzorových příkladů simulací poskytly dostatečný základ pro tento úkol. Zde

stačilo propojit dvě simulované sítě, rozšířit jejich vzájemné propojení a definovat komunikaci.

5.3.10 Shrnutí

Výstupy ze simulace jsou ukládány do textových souborů, rozdělených dle vektorových a skalárních veličin, umístěních ve složce Results. Výstupy lze dále zpracovávat do podoby grafů, a to buď přímo v programu, nebo v jiném, externím programu. Jako příklad je uveden Matlab. Aplikace vyžaduje alespoň základní znalost unixového prostředí, způsob kompilace a spouštění programů a znalost programování v jazyce C.

Co se týče vytváření simulací, je zde připraveno množství modulů k použití, a tudíž vystavění modulů do obrovských celků. Vývojové prostředí je přehledné a grafické uživatelské rozhraní je podobné konkurenčním IDE tohoto typu. Simulátor je navržen pro širší použití, nejen pro studium a analýzu provozu v počítačových sítích. Vytvoření a zprovoznění vzorové počítačové sítě nebylo díky předdefinovaným modulům obtížným úkolem.

5.4 ns-3

ns-3 je všeobecný simulátor pro diskrétní simulace, který je vytvořen v programovacím jazyce C++. Stejně jako ostatní simulátory je určen hlavně pro pokročilou výuku síťové komunikace a dále pak pro vývoj a testování nových protokolů. Jedná se o zcela nový open-source simulátor, který převzal od svých předchůdců pouze některé modely. Je určen pro prostředí uniových systémů.

ns-3 může být jako knihovna přilinkován k hlavnímu C++ programu, stejně jako k programovacímu jazyku Python, do kterého je exportováno API. Simulátor podporuje jak drátové tak i bezdrátové protokoly. Komunikace uživatele se simulátorem probíhá pomocí jazyka TCL(). Odkaz ke stažení simulátoru je na internetové adrese http://www.nsnam.org/ .

5.4.1 Získání simulátoru

Získání simulátoru může probíhat dvěma způsoby. Prvním je stažení balíčku tar a jeho extrakce. Postup je následující:

Vytvoříme si složku, do které si balíček uložíme a poté i rozbalíme. Do příkazového řádku zadáme:

```
cd
mkdir tarballs
cd tarballs
wget http://www.nsnam.org/releases/ns-allinone-3.14.tar.bz2
tar xjf ns-allinone-3.14.tar.bz2
```

Po přepnutí do složky ns-allinone-3.14 bychom měli vidět následující obsah:

```
build.py ns-3.14/ pybindgen-0.15.0/ util.py
constants.py nsc-0.5.2/ README
```

Druhým způsobem získání programu je použití verzovacího a archivačního softwaru Mercurial, který již musí být v systému nainstalován. Nepsaným zvykem je vytvoření obecné složky repos pro ukládání takto získaných distribucí. Postup je následující:
Do příkazového řádku zadáme:

```
cd
mkdir repos
cd repos
hg clone http://code.nsnam.org/ns-3-allinone
```

Po spuštění se na displeji zobrazí informace o přidání (naklonování) dat potřebných pro samotné stažení programu v této podobě:

```
destination directory: ns-3-allinone
requesting all changes
adding changesets
adding manifests
adding file changes
added 31 changesets with 45 changes to 7 files
7 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

Ve složce ns-3-allinone pod repos nalezneme soubory:

```
build.py* constants.py dist.py* download.py* README util.py
```

Jedná se o skripty jazyka Python, pomocí kterých stáhneme a sestavíme distribuci simulátoru. Pro tento úkon je potřeba mít nainstalovaný jazyk Python s potřebnými knihovnami.
Do příkazového řádku napíšeme:

```
./download.py
```

Po spuštění bychom měli vidět podobné informace jako po naklonování složky, zakončené statistikou o aktualizovaných, přidaných, sloučených nebo odstraněných souborech. V naší složce přibudou nové soubory.

5.4.2 Popis kompilace, testování a spuštění programu

Nyní přistoupíme ke kompilaci programu, kdy otestujeme jeho funkčnost na přiložených příkladech. Do příkazového řádku zadáme:

```
./build.py --enable-examples --enable-tests
```

Před námi proběhne řada kompilačních informací zakončená zprávou o úspěšnosti kompilace s dobou jejího trvání. Pro kompilaci nově vytvořených projektů (námi vytvořených) se musíme přepnout do složky ns-3-dev a provést kompilaci odtud. Použijeme k tomu příkaz Waf, což je obdoba příkazu make. Příkaz Waf je součást nové generace kompilačních příkazů jazyka Python, který odbourává složitosti kompilace a konfigurace u rozsáhlých projektů a pomáhá při její optimalizaci. Vzorový příklad optimalizuje kompilaci projektu i s přiloženými příklady a testy a kontroluje nejrůznější závislosti zdrojového kódu a přítomnost použitých knihoven. Zadáme:

```
./waf -d optimized --enable-examples --enable-tests configure
```

O úspěšnosti konfigurace jsme opět informováni. Nyní provedeme konfiguraci pro ladění se zahrnutím příkladů a testů. Do příkazového řádku zadáme:

```
./waf -d debug --enable-examples --enable-tests configure
```

Po tomto kroku můžeme zkompilovat naše laděné programy odesláním příkazu:

```
./waf
```

Příkaz waf má řadu přepínačů použitelných jak v konfigurační tak i kompilační fázi. Pro jejich bližší vysvětlení můžeme použít přepínač --help.
Pro otestování našich zkompilovaných programů slouží příkaz:

```
./test.py -c core
```

Proběhne test všech přítomných programů. Testování je obdoba použití příkazu Waf, jen probíhá ještě kontrola alokace, použití a uvolňování paměti.

A konečně se dostáváme ke spuštění simulací. Vše probíhá pod kontrolou Waf, což nám zajistí kontrolu přítomnosti vyžadovaných sdílených knihoven. Spustíme Waf s přepínačem --run a názvem simulace:

```
./waf --run hello-simulator
```

Výstupem v tomto případě je výpis **Hello Simulator**.

Pozn. Pokud se i přes úspěšný build nic nezobrazí, budeme se pravděpodobně muset přepnout zpět do ladícího módu, viz postup výše.

5.4.3 Popis základní struktury souboru pro simulaci

Základní struktura souboru vychází z jazyka C++ a stejně tak i standardní styl zdrojového kódu, který je zde zmiňován jako simulační skript. V záhlaví se nachází reference na hlavičkové soubory použitých knihoven. Tyto soubory nalezneme ve složce ns3.
Následuje označení jmenného prostoru ns3 a poté povolení logování probíhajících simulačních událostí. Další v pořadí je vstupní bod programu s funkcí main.
Pod ní se nachází funkce pro logování jednotlivých komponent opět s uvedením jejich jména a nastavením úrovně detailů výpisů.

Dále již následuje vytvoření jednotlivých uzlů reprezentujících počítače pomocí příkazů:

```
NodeContainer nodes;
nodes.Create (2);
```

Byly vytvořeny dva uzly.
Tyto uzly musíme propojit mezi sebou. Nejdříve si nadefinujeme spojovací kanál a nastavíme vlastnosti pomocí objektu PointToPointHelper. Ten nám umožní nastavení přenosové rychlosti a zpoždění:

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

Nastavení konkrétních hodnot v příkladu je u přenosové rychlosti 5Mbps a 2ms zpoždění.
Nyní pomocí dalšího objektu vytvoříme propojení mezi dvěma uzly s definovanou komunikační linkou. Vytvoříme si seznam zařízení pro připojení:

```
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

Tím získáme požadovaný kontejner v objektu devices.
Pro komunikaci uzlů potřebujeme nadefinovat protokolový zásobník příkazy:

```
InternetStackHelper stack;
stack.Install (nodes);
```

Dále přiřadíme každému z uzlů IP adresu. Nejdříve nastavíme základní IP adresu a síťovou masku:

```
Ipv4AddressHelper address;
address.SetBase ("192.168.1.0", "255.255.255.0");
```

Od této adresy se nastavují IP adresy pro jednotlivé uzly inkrementované o jednotku. Veškeré definované adresy jsou monitorovány simulátorem pro vyloučení případných kolizí. Následující kód zajišťuje zmíněné přiřazení IP adres uzlům a vytvoření kontejnerové rozhraní, ve kterém bude toto nastavení uloženo.

Z hlediska tvorby topologie a její konfigurace máme vše potřebné zajištěno, jediné co zbývá je vytvoření síťového provozu. K tomuto účelu musíme použít další dva objekty zvané UdpEchoServerHelper a UdpEchoClientHelper, a to následujícím způsobem:

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

Nejdříve vytvoříme objekt serveru a přiřadíme mu číslo portu (zde port 9). K serveru připojíme pomocí metody Install druhý z uzlů (nodes.Get(1)), nastavíme čas spuštění a vypnutí generování provozu v sekundách.

```
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

Na druhé straně využijeme objekt UdpEchoClientHelper a spojíme jej obdobným způsobem s druhým uzlem jako v případě serveru:

```
UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

Vytvoříme instanci echoClient a nastavíme IP adresu rozhraní a port 9. Nastavíme další parametry jako maximální počet paketů, které budou odeslány, doba mezi odesláním paketů, velikost paketu v bytech a také čas spuštění a zastavení klienta, zde jednu sekundu po spuštění serveru. Po tomto kroku jsme připraveni na spuštění simulace. K tomu použijeme globální funkci:

```
Simulator::Run ();
```

Systém projde seznam naplánovaných událostí a na základě zvoleného časového parametru je spustí. Po jejich provedení jsou příkazem Stop ukončeny, a tím i celá simulace. Dále ještě dáme příkaz ke zrušení všech vytvořených objektů a systém se sám postará o jejich likvidaci z paměti. Ukončíme zdrojový kód a přejdeme ke kompilaci a spuštění programu.

```
Simulator::Destroy ();
```

```
return 0;
```

Nejdříve si náš zdrojový kód zkopírujeme do složky scratch a pak už jen spustíme příkaz:

```
./waf
```

Výsledkem je úspěšné přeložení projektu. Konečné spuštění programu se provádí mimo složku scratch příkazem waf s parametrem --run:

```
./waf --run scratch/test
```

Proběhne automatická kontrola, zda byl proveden překlad projektu a spustí program test. Výpis stavů simulace je zobrazen díky povolení logování v konzole. V tomto případě:

```
Waf: Entering directory ‘/home/craigdo/repos/ns-3-allinone/ns-3-dev/build’
Waf: Leaving directory ‘/home/craigdo/repos/ns-3-allinone/ns-3-dev/build’
’build’ finished successfully (0.418s)
Sent 1024 bytes to 192.168.1.2
Received 1024 bytes from 192.168.1.1
Received 1024 bytes from 192.168.1.2
```

Klient odeslal paket serveru na adresu 192.168.0.2, server paket přijal a vrátil paket na adresu klienta 192.168.1.1.

5.4.4 Vytvoření vzorové sítě

Při vytváření vzorové sítě byl z topologie vyjmut síťový prvek switch, který není podporován a byl nahrazen hubem. Ostatní náležitosti nebyly změněny. Práce na vytvoření sítě byla z důvodu teoretické přípravy časově náročná.

5.4.5 Shrnutí

Dokumentace je dostupná v systému Doxygen a je dodávána společně s instalací. Pro samotnou práci se simulátorem je potřeba dostatečná teoretická příprava, která je srovnatelná s výukou příkazů programovacího jazyka, neboť má obsáhlé programové rozhraní. Velice propracovanou kapitolou je zpracování výsledků v podobě podrobných textových výpisů a trasování jednotlivých paketů. Výsledky je možné si prohlédnout pomocí programu tcpdump nebo Wireshark a jsou uloženy v souboru s příponou *pcap*. Při použití programu gnuplot dostaneme informace o přenosu dat v čase pro vyhodnocení do podoby grafů. Simulátor podporuje také tvorbu přenosů dat přes bezdrátovou síť. Přestože se jedná o textovou formu simulace, můžeme po instalaci programu NetAnim zobrazovat průběh simulace v reálném čase. K tomu je zapotřebí

doplnit metody pro výstupy simulace do souboru ve formátu xml. Jedná se o zajímavý projekt na poli simulátorů počítačových sítí a je určen pro zkušenější uživatele.

5.5 cnet

Grafický síťový simulátor *cnet* pochází ze Západoaustralské univerzity a je v první řadě určen studentům pro výuku. Simulátor pracuje v uniovém operačním systému a umožňuje experimentování na různých vrstvách referenčního modelu OSI, tvorbu vlastních protokolů pro klasické i bezdrátové sítě. Simulátor je dostupný z adresy http://www.csse.uwa.edu.au/cnet/download.html, kde je podmínkou pro jeho stažení vyplnění krátkého formuláře s uvedením emailové adresy a důvodu zájmu o tento software.

5.5.1 Instalace software

Před samotnou instalací simulátoru je potřeba v systému zkontrolovat přítomnost knihovny libelf pro práci s objektovými soubory, dále pak programovací jazyk Tcl s knihovnou Tk ve verzi nejméně 8.4 a to i se soubory pro vývoj. Pro stažení a instalaci zapíšeme do příkazového řádku:

```
sudo apt-get install tcl8.5 tcl8.5-dev tk8.5 tk8.5-dev libelf-dev
```

Ze stránek projektu stáhneme simulátor a rozbalíme jej:

```
tar zxvpf cnet-3.2.4.tgz
```

Přepneme se do nově vytvořené složky cnet-3.2.4 a zkontrolujeme nastavení konstant v souboru *make* spjatých s cestami ke složkám knihoven, www stránek, ke složce bin a nastavení prefixů a případně je aktualizujeme. V souboru *src/preferences.h* zkontrolujeme úplnost cesty ke kompilátoru a linkeru systému. Poté postupně spustíme příkazy:

```
make
make install
```

Pro ověření úspěšnosti instalace spustíme jeden ze vzorových příkladů, například příkazem:

```
cnet TICKTOCK
```

Výsledkem bude pracovní okno s připravenou síťovou topologií o dvou uzlech a vzájemným přeposíláním informací.

5.5.2 Popis vlastností simulátoru

Primární prostředí pro tento simulátor je terminál, který je v případě požadavku možné nahradit grafickým výstupem. Ten poskytuje nástroje pro změnu parametrů jednotlivých uzlů během simulace. Parametry najdeme v informativním okně po kliknutí na příslušný uzel. Zde se objeví i statistické údaje uzlu o přenesených datech a stavu přenosové linky. Změna parametrů probíhá nastavením rolovacích lišt nebo zaškrtávacích políček. Můžeme měnit stav uzlu. Simulované stavy jsou pracovní, odstavení (pause), porucha, restart.

Simulátor disponuje širokou škálou chybových hlášení a má propracovaný systém jejich detekce, což při tvorbě simulace umožňuje jejich efektivní odstraňování. *cnet* je napsán v programovacím jazyku C a také jej vyžaduje při tvorbě síťových protokolů. Tyto jsou po zkompilování, nejlépe s gcc přilinkovány ke spuštěnému simulátoru.

cnet podporuje mobilní a bezdrátové sítě, povoluje použití více síťových karet a umožňuje nastavení jejich parametrů, frekvenci přenosu dat, intenzitu přijímaného signálu, spotřebu elektrické energie. Pro zařízení v pohybu zjišťuje a nastavuje jejich polohu. Veškeré činnosti uzlů jsou řízeny modelem událostí.

Pro simulování komunikace po Ehernetu je zde implementována podpora ethernetových segmentů, kde každý segment obsahuje nejméně dva uzly. Tyto segmenty se spojují pomocí routerů do větších celků. Adresu síťové karty lze nakonfigurovat. *cnet* podporuje detekci kolizí a zahlcení sítě, všesměrové vysílání, permanentní přenosovou rychlost. Nastavování těchto parametrů probíhá v příslušných strukturách ve zdrojovém kódu a jejich deklarace je umístěna ve stěžejním hlavičkovém souboru *cnet.h*. K aktualizaci dat v těchto strukturách dochází před zařazením uzlu do fronty pro spuštění. Tato data můžeme měnit již výše zmíněnými posuvnými lištami.

U parametrů linek si můžeme vybrat z nastavení nákladů na přenos jednoho rámce a počet bytů v jednom rámci.

5.5.3 Vytvoření prvního programu

Zdrojový kód simulovaných protokolů je psán v jazyce C, přesto jsou zde odlišnosti ve struktuře programu. Soubory neobsahují vstupní bod pro provádění příkazů v podobě funkcí *main () a exit (),* ani návratové hodnoty.

```
#include <cnet.h>

void reboot_node(CnetEvent ev, CnetTimerID timer, CnetData data)
{
    printf("hello world\n");
}
```

V tomto případě je uzel po zadání požadavku restartován a na výstupu všech uzlů je vytištěna zpráva. Parametry pro zajištění této činnosti jsou ev = EV_REBOOT, timer = NULLTIMER a data = 0, tedy samotný požadavek, časovač jeho spuštění a bez potřeby zpracování dat. Pak už stačí jen v příkazové řádce zapsat

```
cnet -C helloworld.c -r 2
```

a program se spustí. Po provedení tisku zprávy je simulace stále aktivní, ale bez dalších požadavků na obsluhu a zpracování událostí. V tom případě je *cnet* schopen rozpoznat tento stav a ukončit činnosti programu.

5.5.4 Parametry příkazové řádky

Příkazová řádka slouží k nastavení volitelných parametrů pro spuštění simulace. Pro spuštění simulace existují tři obecné způsoby:

```
cnet [command-line-options] TOPOLOGYFILE [args-to-reboot-function]
cnet [command-line-options] - [args-to-reboot-function]
cnet [command-line-options] -r NNODES [args-to-reboot-function]
```

Položky v hranatých závorkách jsou volitelné. Pro vysvětlení použijeme zápis na příkazovou řádku z prvního programu. Na prvním místě je příkaz *cnet* následovaný vstupními parametry, názvem souboru s topologií a argumenty pro funkci reboot (restartování uzlu). Parametr – *C* deklaruje použití interních vrstev referenčního modelu bez nutnosti jejího definování programátorem, zároveň se očekává jméno zdrojového souboru pro kompilaci a jeho nalinkování k *cnet* ve formě souboru se stejnojmennou příponou *cnet*, neboť k linkování zkompilovaných protokolů dochází za běhu programu. Při použití parametru – *C* lze přidat přepínač – r, který umožní vytvořit náhodnou síťovou topologii o zadaném počtu uzlů. Následují argumenty pro restart uzlu. Nahradíme-li jméno souboru pomlčkou (prostředí vzorový případ), je zahrnut standardní vstupní soubor *protocol.c*.

5.5.5 Popis základních ovládacích prvků simulace

Simulátor pracuje na základě obsluhy událostí. Tyto vznikají při restartu uzlu, požadavku na poslání zprávy aplikační vrstvou, obdržení rámce na úrovni fyzické vrstvy, uplynutí stanoveného času, vypnutí uzlu nebo požadavku na ladění kódu. Události jsou zpracovávány v naplánovaném pořadí, bez možnosti nastavení jejich priority. Pokud nedojde k návratu události z obsluhy je simulace zablokována a musí být uživatelsky ukončena z příkazové řádky. Nejdříve musí být zaregistrována obsluha události. Ta je později vyvolána ke zpracování požadavku a obsahuje tři parametry - typ události, časování a uživatelská data. Jedinou povinnou obsluhou události je restartování uzlu - reboot_node() funkce. Ta zajišťuje čas pro inicializaci protokolů, jejich alokaci v paměti, informuje simulátor o použitých protokolech a dalších událostech. Při vytváření obsluhy událostí je doporučováno používat makro z hlavního hlavičkového souboru <*cnet.h*> pro lepší čitelnost kódu. Příklad ukazuje použití makra a deklaraci obsluhy události.

```
 #define EVENT_HANDLER(name)
        void name(CnetEvent ev, CnetTimerID timer, CnetData data)
```

```
EVENT_HANDLER(timeouts)
{
    ....
}
```

Důležitým prvkem simulace je časování. K tomu jsou určeny časovací události. V následujícím příkladu je ukázka použití časovače při restartu uzlu:

```
#include <cnet.h>

EVENT_HANDLER(reboot_node)
{
    CHECK(CNET_set_handler(EV_TIMER1, timeouts, 0));
    CNET_start_timer(EV_TIMER1, (CnetTime)1000000, 0);
}
```

Funkce CHECK kontroluje správnost průběhu vnitřní funkce CNET_set_handler a zachytává případné chyby. Funkce CNET_set_handler zajišťuje provedení funkce timeouts() při každém vzniku události EV_TIMER1. Na druhém řádku je nastaven čas pro generování události EV_TIMER1 v mikrosekundách.

Následuje funkce pro obsluhu události timeouts(), která může vypadat například takto:

```
static EVENT_HANDLER(timeouts)
{
    static int poradi = 0;

    printf("%3d.\t%s\n", poradi, (poradi%2) == 0 ? „tik" : "\ttak");
    ++poradi;

    CNET_start_timer(EV_TIMER1, (CnetTime)1000000, 0);
}
```

Vstupní parametry pro tuto funkci jsou převzaty z funkce reboot_node(). Činnost této funkce je taková, že po prvním zpracování pošle na standardní výstup řetězec „0. tik“. Inkrementuje se vnitřní proměnná pro počítání pořadí a znovu se zavolá funkce CNET_start_timer() v čase jedné sekundy.

Dalším důležitým prvkem při tvorbě simulací je používání ladících událostí. Ve funkci reboot_node() zaregistrujeme událost EV_DEBUG1 a propojíme ji s funkcí posli_zpravu. Dále si nastavíme název „Poslat“ na jedno z tlačítek v grafickém prostředí. Po jeho stisku vyvoláme požadovanou událost, zde posli_zpravu, kterou budeme muset definovat.

```
EVENT_HANDLER(reboot_node)
{
    CHECK(CNET_set_handler(EV_DEBUG1, posli_zpravu, 0));
    CHECK(CNET_set_debug_string( EV_DEBUG1, "Poslat"));
}
```

Následuje příklad funkce pro odeslání zprávy a informativním výpisem:

```
static EVENT_HANDLER(posli_zpravu)
```

```
{
    char         ramec[256];
    size_t       delka;

    sprintf(ramec, "zprava %d is %s", ++pocet, „Posilam zpravu“);
    delka      = strlen(ramec) + 1;

    printf("posilam %u bytu, kontrolni_soucet=%6d\n\n",
               delka,
       CNET_IP_checksum((unsigned short *)ramec,(int)delka));
    CHECK(CNET_write_physical_reliable(1, ramec, &delka));
}
```

Pro komunikaci mezi uzly se používá pouze fyzická vrstva, tudíž musí být implementována vždy. Při provádění simulace používají všechny zainteresované uzly stejnou kopii protokolu a mění se pouze vnitřní proměnné. Pokud se provádí čtení dat je potřeba udržovat statistické informace o množství požadované a použité paměti.

5.5.6 Vytvoření síťové topologie

Pro definování síťové topologie použijeme zvláštní soubor, který bude obsahovat požadované parametry sítě společně s názvy uzlů, jejich vzájemné propojení a použité síťové protokoly. Následující text zobrazuje vytvoření sítě o dvou uzlech s použitím protokolu StopAndWait.

```
compile          = "stopandwait.c"

mapimage         = "australia.gif"
messagerate      = 500ms
probframecorrupt = 4

host Perth {
    wan to Melbourne
}

host Melbourne {
    east of Perth
}
```

Nalezneme zde globální parametry pro celou topologii. V první řádce je nastaven použitý protokol, poté nastavíme grafické pozadí simulace, rychlost přenosu zpráv a pravděpodobnost ztracených (poškozených) zpráv. Následují jména uzlů a způsob jejich připojení do sítě. Můžeme zde nastavit vlastní parametry připojení. Takto budou globální parametry ignorovány. Vše je zapisováno ve stylu jazyka C.

Při návrhu rozlehlé síťové topologie můžeme nastavit parametry spojení jako je šířka pásma, pravděpodobnost hardwarové chyby a doba na její opravu. Tyto parametry jsou platné pro oba směry linky, stačí tedy nastavení na jedné straně spojení. Příklad nastavení globálních parametrů sítě WAN:

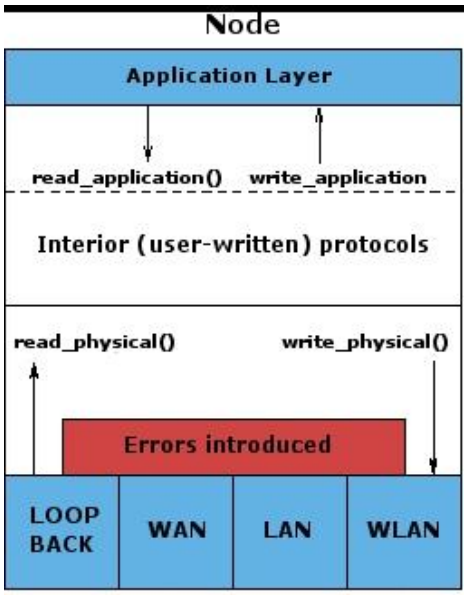
```
wan-bandwidth = 128Kbps
wan-mtbf      = 3600sec
```

```
wan-mttr       = 60sec
```

5.5.7 *Simulační model cnet*

cnet umožňuje použití čtyř typů uzlu - host, router, mobile a accesspoint. Pro každý typ uzlu lze nastavit trochu odlišné parametry. Pro simulování spojení můžeme použít nabízené tři typy linek, lokální síť LAN, rozlehlou síť WAN nebo bezdrátovou síť WLAN opět s příslušnými parametry. Úplný výčet parametrů pro jednotlivá zařízení, linky a globální nastavení lze nalézt na manuálových webových stránkách projektu.

cnet poskytuje dvě vrstvy referenčního modelu ISO/OSI, a to fyzickou vrstvu pro všechny uzly a aplikační vrstvu pouze uzly host a mobile. Dále nabízí vlastní vrstvu pro simulaci a detekci chyb Error Layer pro výše zmíněné chybové parametry. Obrázek 10 ukazuje rozložení jednotlivých vrstev v uzlu [*cnet*].



Obrázek 9 Rozložení jednotlivých vrstev uzlu

Jak je vidět, mezi poskytnutými vrstvami je velký prostor pro tvorbu vlastních protokolů. Jednotlivé uzly jsou bez vytvoření dodatečných protokolů téměř izolovány jeden od druhého, nevědí, kde se nachází ostatní uzly, neznají jejich jména ani parametry. Je potřeba naučit tyto uzly získávat tyto informace a posílat je dál ostatním uzlům.

5.5.8 Vytvoření vzorové sítě

S ohledem na popsané vlastnosti sítového simulátoru *cnet* byla vytvořena počítačová síť s předepsaným počtem koncových zařízení, avšak s absencí switche jako aktivního síťového prvku, zároveň byl server nahrazen dalším hostem. Řešením by mohlo být vytvoření vlastního protokolu popř. protokolů, které by byly schopné tuto činnost zastoupit.

5.5.9 Shrnutí

Hlavní náplní činnosti simulátoru je tvorba vlastních síťových protokolů. Pro práci v tomto simulátoru musí být uživatel/programátor vybaven dostatečnými znalostmi v oblasti provozu počítačových sítí, aby mohl úspěšně implementovat vlastní protokoly do připravené struktury komunikačních vrstev.

5.6 Přehled vlastností simulátorů

Následující tabulka shrnuje srovnávací kritéria pro vyhodnocování simulátorů. Doplňující informace jsou zapsány pod tabulkou.

	Verze SW	Licence	Uživatelská přívětivost instalace	Náročnost použití nástroje	Využití znalosti programování	Vizualizace topologie sítě	Rozšiřitelnost síťových komponent	Rozšiřitelnost dostupných protokolů	Vizualizace toku dat	Možnost propojení simulované sítě do reálného prostředí	Zdrojové kódy	Operační systémy	Lokalizace produktu
GNS3	0.8.2	volná	Průvodce instalací	1	Ne	Ano	Ne	Ne	Ne	Ano	Ne	Windows	Ano
Packet Tracer	5.3.2	volná	Průvodce instalací	2	Ne	Ano	Ne	Ne	Ano	Ne	Ne	Windows	Ne
Omnet++	4.2.2	volná	Dekomprese, konfigurace, kompilace*	4	Ano	Ano	Ano	Ano	Ano	Ne	Ano	Windows, Unix, Mac OS	Ne
ns-3	3.14	volná	Dekomprese, konfigurace, kompilace	5	Ano	Ano	Ano	Ano	Ano	Ne	Ano	Unix, Mac OS	Ne
cnet	3.2.4	volná	Dekomprese, konfigurace, kompilace*	3	Ano	Ano	Ano	Ano	Ano	Ne	Ano	Unix	Ne

Tabulka vlastností simulačních nástrojů

Poznámky k tabulce:

1) * (znak hvězdička) - programy vyžadují dodatečnou instalaci grafických knihoven
2) Stupnice náročnosti (číslo 1 znamená nejméně náročné).

5.7 Zamítnuté síťové simulátory

Pro hledání a zjišťování informací o dostupných simulátorech sítí byl jako zdroj použit Internet. Vyhledávání vhodných simulátorů byla obtížná činnost, neboť bylo v první řadě potřeba zjistit aktuálnost softwaru, a v mnoha případech byla tato informace velmik těžko dostupná. Vodítkem bylo srovnávání simulátorů v odborných tuzemských a zahraničních publikacích, nalezení jejich poslední verze a poskytovaná podpora produktu. Výsledkem je jednak vytvoření seznamu zamítnutých simulátorů a také seznamu těch, které byly použity v této práci.

Tabulky, které jsou součástí přílohy, shrnují nalezené síťové simulátory, které nebyly použity pro vzájemnou komparaci s uvedením důvodu jejich zamítnutí, odkaz na jejich úložiště a jsou součástí přílohy. Důvody jsou uvedeny v nadpisu tabulky.

6 Závěr

Softwarový trh nabízí široký výběr simulátorů nejrůznějšího zaměření. Největší zastoupení zde mají simulátory dopravní infrastruktury, počítačových sítí nebo simulace výrobních procesů, zábavního průmyslu a telekomunikací. Mnoho z nich jsou krátkodobé projekty realizované převážně univerzitami celého světa a orientují se hlavně na nižší vrstvy referenčního modelu ISO/OSI. V případě komerčních produktů jsou to protokoly vrstev vyšších. Obecně lze říci, že srovnávání simulátorů sítí je velmi obtížnou záležitostí. Důvodem je právě jejich rozmanitost a rozdílnost zaměření. Pro srovnávání simulátorů v této práci byly použity produkty firem a výzkumných zařízení dislokovaných převážně na akademické půdě, jejichž použití je v případě *Cisco Packet Tracer* podmíněno členstvím ve vzdělávacím institutu Cisco Networking Academy, *Omnetpp* je vyhrazeno pouze studentům a pro fungování GNS3 je potřeba získat obraz operačního systému pro emulovaná zařízení opět po registraci v Cisco Networking Academy.

Pro vzájemné porovnání simulátorů byly předem nadefinovány parametry, které byly poté vyhledávány a zaznamenávány. Výsledky byly ukládány do přehledové tabulky pro zjištění odlišností. Současně byla definována vzorová počítačová síť, která také sloužila jako měřítko vzájemného posuzování. Z široké nabídky simulátorů bylo vybráno pět, u kterých byla zřejmá aktuálnost softwaru, jejich uživatelská podpora a grafické uživatelské prostředí.

Jak již bylo předesláno v úvodu práce, simulátory mají své specifické zaměření, což nám znemožňuje závěrečné vyhodnocení vypovídající o tom, který simulátor je lepší, komplexnější a věrněji odráží realitu.

Obecně vzato, simulátory síťového provozu v prostředí unixových systémů předpokládají dobrou znalost komunikací v počítačových sítích a programování v jazyce C, popř. C++, které využijeme jak při tvorbě topologie tak i při vytváření vlastních protokolů. Grafické uživatelské prostředí slouží jako nadstavba a není pro samotný běh simulace potřebné. Na druhé straně je použití vybraných simulátorů v prostředí Windows vhodnější pro výuku síťové komunikace.

Konkrétně v případě *GNS3* jde o nastavování parametrů síťových prvků a následný test průchodnosti dat. U *CiscoPackerTracer* je předchozí funkcionalita rozšířena o vizualizaci testovacích dat s možností výběru typu dat dle protokolu a možnost změny parametrů přenosu. Obdobou je linuxový *Omnet++* používající ale pro tvorbu topologie programovací jazyk NED. Simulátor *cnet* je zaměřen hlavně na tvorbu vlastních protokolů, a *ns-3* disponující rozsáhlou knihovnou simulačních nástrojů je vhodný pro tvorbu simulací v obecném slova smyslu.

S vývojem sledovaných produktů je počítáno i do budoucnosti, což zmiňují autoři ve svých dokumentacích. Tento fakt potvrzuje vydání aktualizačních balíčků simulátorů *ns-3*a *GNS3* během vzniku této práce.

Využití této práce spočívá v prvotní orientaci na poli simulačních nástrojů a získání přehledu o jejich dostupnosti a zaměření pro kvalitní rozhodnutí při potřebě vlastní realizace simulace. Do budoucnosti by však bylo potřeba pro kvalitnější rozhodování o výběru simulátoru vytvořit více vzorových testovacích sítí, případně doplnit kritéria porovnávání. Každopádně, používání jakýchkoliv simulátorů klade zvýšené nároky na výkonnost hardware, neboť zpracovává a generuje velké množství dat.

Literatura a další informační zdroje

[Hamilton (1996)] HAMILTON, J. A. *Distributed simulation*. CRC Press, Inc. Boca Raton, FL, USA 1996. ISBN 0-8493-2590-0.

[Fujimoto (2000)] FUJIMOTO, R. M. *Parallel and distributed simulation systems*. New York: John Wiley & Sons, Inc., 2000. ISBN 0-471-18383-0.

[Havlenka (1997)] HAVLENKA Jiří a kolektiv. *Výkladový slovník výpočetní techniky a komunikací*. Praha: Vydavatelství a nakladatelství Computer Press, 1997. ISBN 80-7226-023-5.

[Clacke1999] Clarke, E. M., Grumberg, O., Peled, D. A. *Model Checking*. The MIT Press, Cambridge, Massachusetts, United States of America, 1999. 314 s. ISBN 0-262-03270-8.

GNS3 [Počítačový program]. Ver.0.8.2 for Windows, Unix. Dostupné z URL: <http://www-gns3.net> [cit. 2012-1-20]. Tool for computer simulation.

CiscoPacketTracer [Počítačový program]. Ver.5.3.2 for Windows, Dostupné z URL: < http://www.packettracer.info/> [cit. 2012-2-2]. Tool for computer simulation.

Omnetpp [Počítačový program]. Ver.4.2.2 for Windows, Unix. Dostupné z URL: <http://www.omnetpp.org/> [cit. 2012-2-25]. Tool for computer simulation.

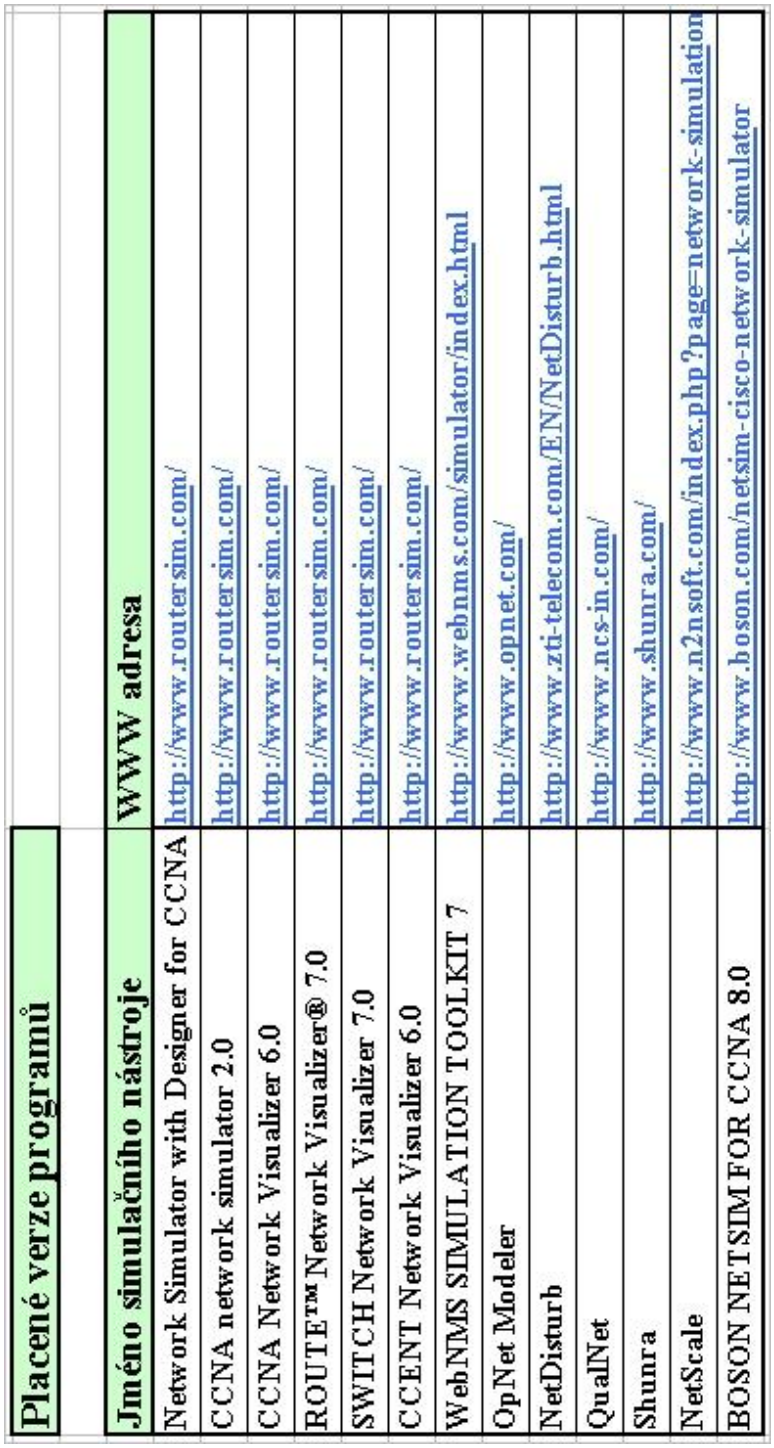
ns-3 [Počítačový program]. Ver.3.14 for Unix. Dostupné z URL: <http://www.nsnam.org/ > [cit. 2012-3-17]. Tool for computer simulation.

cnet [Počítačový program]. Ver.3.2.4 for Unix. Dostupné z URL: <http://www.csse.uwa.edu.au/cnet/> [cit. 2012-4-10]. Tool for computer simulation.

Seznam obrázků

Přílohy

A: Tabulka Placené verze simulátorů



Placené verze programů	
Jméno simulačního nástroje	**WWW adresa**
Network Simulator with Designer for CCNA	http://www.routersim.com/
CCNA network simulator 2.0	http://www.routersim.com/
CCNA Network Visualizer 6.0	http://www.routersim.com/
ROUTE™ Network Visualizer® 7.0	http://www.routersim.com/
SWITCH Network Visualizer 7.0	http://www.routersim.com/
CCENT Network Visualizer 6.0	http://www.routersim.com/
WebNMS SIMULATION TOOLKIT 7	http://www.webnms.com/simulator/index.html
OpNet Modeler	http://www.opnet.com/
NetDisturb	http://www.zti-telecom.com/EN/NetDisturb.html
QualNet	http://www.ncs-in.com/
Shunra	http://www.shunra.com/
NetScale	http://www.n2nsoft.com/index.php?page=network-simulation
BOSON NETSIM FOR CCNA 8.0	http://www.boson.com/netsim-cisco-network-simulator

B: Tabulka Zastaralé, nepodporované simulátory

Zastaralý, nepodporovaný software	
Jméno simulačního nástroje	**WWW adresa**
NCTUns	http://nsl.csie.nctu.edu.tw/nctuns.html
Jasper	http://www.cs.stir.ac.uk/~kjt/software/comms/jasper.html
NS2	http://www.isi.edu/nsnam/ns/
SSFNet	http://www.ssfnet.org/homePage.html
PDNS	http://www.cc.gatech.edu/computing/compass/pdns/
Dynamic Network Emulation Backplane Project	http://www.cc.gatech.edu/computing/pads/netemulation/
Parsec	http://pcl.cs.ucla.edu/projects/parsec/
Dartmouth SSF (DaSSF)	http://users.cis.fiu.edu/~liux/research/projects/dassf/index.html
GtnetS	http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/
J-Sim	http://www.j-sim.zcu.cz/
HEGONS - Optical Network Simulator	http://www.solostuff.net/hegons.php
Network cash simulator	http://ncs.codeplex.com/
JiST / SWANS	http://jist.ece.cornell.edu/
SNS	http://www.cs.cornell.edu/people/egs/sns/
GLOMOSIM	http://pcl.cs.ucla.edu/projects/glomosim/
Narses	http://de.sourceforge.jp/projects/sfnet_narses/
3LS P2P	http://www.computer.org/portal/web/csdl/abs/proceedings/p2p/2003/2023/00/202
NeuroGrid	https://lists.sourceforge.net/lists/listinfo/neurogrid-simulation
PeerSim	http://peersim.sourceforge.net/
P2PSim	http://pdos.csail.mit.edu/p2psim/

C: Obsah CD

CD obsahuje konfigurační nastavení vytvořených počítačových sítí, zdrojové kódy a příklad testovacího síťového provozu. Postup pro spuštění jednotlivých konfigurací je analogický se způsobem popsaným uvnitř této práce. Dále je zde tisknutelná verze bakalářské práce ve formátu pdf s společně se zdrojovým textem.