VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Tomáš Bičák

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

NÁVRH A REALIZACE SÍŤOVÉHO TERMINÁLU
NETWORK TERMINAL DESIGN AND REALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Bičák**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jaroslav Koton, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Tomáš Bičák ***ID:*** 155737
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Návrh a realizace síťového terminálu

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte síťový terminál, který bude možné připojit do datové sítě a jím realizovat a vyhodnocovat příkaz ping na uživatelem zvolené uzly sítě. Zaměřte se především na možnost nastavení volitelných parametrů tohoto příkazu. Síťový terminál realizujte tak, aby pro potřeby jeho ovládání nebylo nutné samostatné PC.

DOPORUČENÁ LITERATURA:

[1] POSTEL, J., Internet Control Mesage Protocol, Network Working Group, RFC 792, 1981.

[2] VODA, Z.: Průvodce světem Ardurina, cit. [2015-10-19], dostupné online: http://arduino.cz/programujeme-arduino/

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: doc. Ing. Jaroslav Koton, Ph.D.
Konzultant:

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

ABSTRAKT

Tato bakalářská práce se věnuje návrhu a realizaci síťového terminálu. Cílem je sestrojit a realizovat síťový terminál, který bude generovat a vyhodnocovat příkaz ping, na zvolený síťový uzel. Síťový terminál funguje na platformě Banana Pi.

KLÍČOVÁ SLOVA

Banana Pi, dotykový displej, Ping, ICMP protokol

ABSTRACT

This bachelor thesis is about designing and construction of network terminal. The goal is to construct and implement a network terminal which will generate and evaluate the command ping of the selected network node. Network terminal based on Banana Pi platform.

KEYWORDS

Banana Pi, touchscreen, Ping, ICMP protocol

BIČÁK, Tomáš *Návrh a realizace síťového terminálu*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 83 s. Vedoucí práce byl doc. Ing. Jaroslav Koton, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Návrh a realizace síťového terminálu“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jaroslavu Kotonovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

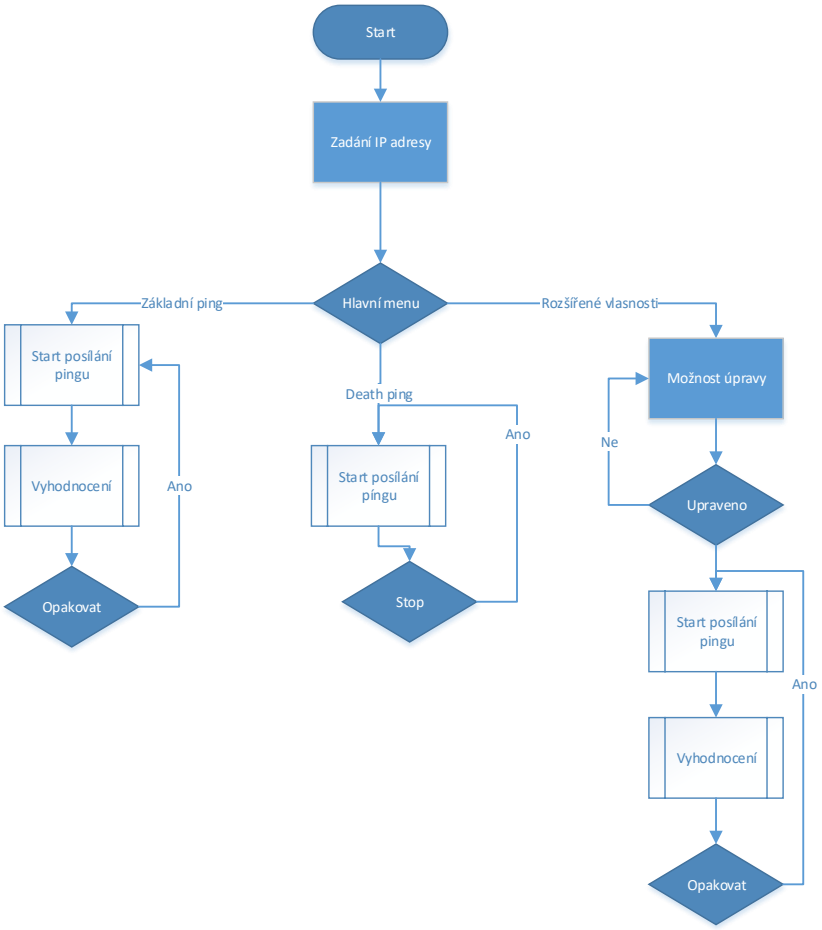
SEZNAM VÝPISŮ

ÚVOD

Tato bakalářská práce se věnuje oblasti návrhu a realizaci říditelného síťového terminálu.

Říditelný síťový terminál posílá a vyhodnocuje ping na zvolený síťový uzel. Účelem terminálu je posílat ping na zvolené síťové uzly a tím zjistit jejich dostupnost v síti. Terminál bude sloužit k testování dostupnosti přístupových uzlů a koncových stanic. Samotné testování dostupnosti stanic a uzlů se provádí pomocí pingu. Terminál vypisuje minimální a maximální zpoždění pingu v síti.

Zařízení je realizované na zařízení Banana Pi, ke kterému je připojen dotykový displej, přes který se celé zařízení bude ovládat. Volba pingu je rozdělena do několika částí. Je zde možnost navolit základní ping a nebo ping s ping s rozšířenýma vlastnostmi, které lze měnit. Na obrázku 1 je znázorněn vývojvý datagram programu.



Obr. 1: Vývojový datagram

1 PROTOKOL ICMP

Protokol ICMP (Protokol řídících hlášení – Internet Control Message Protocol) je určen k přenosu specifických zpráv týkajících se chyb a zvláštních okolností při přenosu datagramů.Cílem většiny ICMP hlášení není aplikace nebo uživatel, ale IP software (zde existují také výjimky a tou je např. ping).

Protokol ICMP používá služeb IP (Pracuje v rámci síťové vrstvy na IP). ICMP zprávy se v IP sítích přenášejí stejně jako jakékoliv jiné datagramy, nemají žádné prioritní zacházení, takže není zajištěno jejich doručení. ICMP zprávy se často vysílají v situacích, kdy je síť zahlcena nebo má nějaký problém. Proto jsou ICMP zprávy koncipované tak, aby samy nepřispívaly k dalšímu zkomplikování situace např. zbytečně vysokým počtem datagramů. ICMP zprávy se proto negenerují v souvislosti s problémy datagramů na všeobecnou nebo skupinovou adresu, zpráv se zdrojovou adresou neidentifikující konkrétní stanici (např. 127.0.0.0 nebo 0.0.0.0). ICMP zprávy se rovněž negenerují v souvislosti se směrováním a doručením ICMP zpráv, tj. ICMP rekurzivně nereagují na zprávy ICMP.

Pokud nějaký datagram způsobí problém, ICMP o něm může informovat pouze zdrojový uzel, který datagram vygeneroval. Zdroj pak musí sám reagovat na tuto informaci a zjednat nápravu. Chybová informace zaslaná zdrojové stanici je jediná možnost reakce na problém s přenosem datagramu nebo datagramem samotným v síti, protože nelze vysledovat celou cestu datagramu internetem, aby se zajistilo, kde po cestě může případně být problém.

1.1 Formát datagramu a typ zpráv ICMP

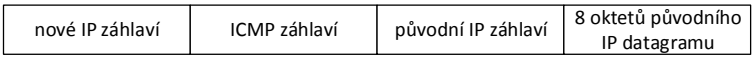
Protokol ICMP používá IP pro přenos zpráv, takže zprávy zapouzdřuje do IP datagramu. Kromě vlastního ICMP záhlaví obsahuje jeho zprávy také záhlaví IP datagramu, k němuž se vztahují, a počátečních 8 oktetů tohoto datagramu (viz obrázek 1.1).

Formát záhlaví ICMP zprávy je uveden na obrázku 1.2. Všechny zprávy ICMP mají první tři pole záhlaví fixní délky, za nimi mohou následovat další informace už v závislosti na typu zprávy.

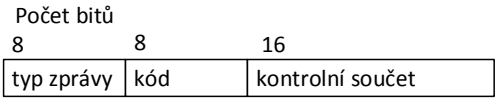
Zprávy ICMP má následující pole:

- **typ zprávy** – specifikuje typ zprávy (viz tabulka 1.2);
- **kód** – určuje parametry konkrétní zprávy;
- **kontrolní součet** – zabezpečení zprávy vůči chybám.

Další formát datagramu závisí na typu zprávy.



Obr. 1.1: ICMP zpráva zapouzdřená do IP datagramu



Obr. 1.2: Formát datagramu protokolu ICMP

Tab. 1.1: Kódy hlášení protokolu ICMP Redirect [2]

Kód	**Význam**
0	*Redirect datagrams for the net* = přesměrování datagramů pro síť
1	*Redirect datagarms for the host* = přesměrování datagramů pro počítač
2	*Redirect for the type of service and net* = přesměrování pro typ služby a síť
4	*Redirect for the type of service and host* = přesměrování pro typ služby a počítač

Tab. 1.2: Nejčastější typy ICMP zpráv

Typy ICMP hlášení	**Obsah zprávy**	**Query (Q) / Error (E)**
0	*Echo reply* = odpověď na žádost o odezvu (součást ping; kódy viz tabulka 1.4)	Q
3	*Destination unreachable* = cíl je nedostupný, protože cílový port, protokol, stanice nebo síť jsou nedostupné nebo neznámé, případně by byla nutná fragmentace po cestě k cíli, ale bit byla nutná fragmentace po cestě k cíli, ale bit DF v IP datagramu ji nepovoloval	E
4	*Source quech* = jednoduché řízení toku dat (protokou bez spojení) při zahlcení paměti směrovače v případě, kdy frekvence přijatých datagramů je tak velká, že je směrovač nestačí zpracovávat; další přijaté datagramy začne likvidovat, přičemž pro každý takový datagram vyšle jeho odesílateli ICMP zprávu*Source quecnh*. Každou takovou chybovou zprávu by měl odesílatel chápat jako žádost cílové stanice o snížení rychlosti přenostu. Podle RFC 1122 IP musí tento problém předat transportní vrstvě, a ta (případně aplikační vrstva) by měla zareagovat odpovídajícím způsobem. Dnes se místo *Source quench* používají moderní mechanismy řízení zahlcení.	E
5	Redict = přesměřování lepší cestou pro směřování (inzerována směrovačem zdrojové stanice, pokud je k síti připojen jiný směrovač s kratší cestou k cílové síti),viz tabulka 1.1. Předpokládá se, že směrovače mají pouze minimální IP konfiguraci a stačí jim znát jeden směrovač pro komunikaci vně vlastní sítě	E
8	*Echo request* = žádost o odezvu (součást ping)	Q

Pokračování na další stránce

Tab. 1.2 – *Pokračování z předešlé stránky*

Typy ICMP hlášení	**Obsah zprávy**	**Query (Q) / Error (E)**
9	*Router advertisment* = inzerování adresy směrovače; posílá se na skupinovou adresy 224.0.0.1 pro systémy s podporou skupinového vysílání; jinak na všeobecnou adresu 255.255.255.255, periodicky obvykle každých 10 minut	Q
10	*Router solicitation* = žádost o adresu směrovače; (v době čekání na *Router adverstisment*); posílá se na skupinovou adresu 224.0.0.2 směrovačů s podporou skupinového vysílání, jinak na všeobecnou adresu 255.255.255.255	Q
11	*Time exceeded* = životnost datagramu překročena (*Time-To-Live*), dosála na nulu nebo vypršel časový limit čekání na zbývající fragmenty téhož datagramu - vypršela doba života datagramu nebo vypršel časový limit pro znovu sestavení přijímaného datagramu z jeho jednotlivých fragmentů. Používá se pro detekci cyklických nebo extrémně dlouhých cest.	E
12	*Parametr problém* = problém v parametrech záhlaví datagramu, na který není možno zareagovat některou z předchozích ICMP zpráv (např. nesprávné záhlaví datagramu). Pokud je *Code* = 0, pak položka *Pointer* ukazuje na oktet datagramu, který problém způsobil, jinak *Code* = 1 znamená, že v datagramu chybí požadovaná volitelná položka.	E

Pokračování na další stránce

Tab. 1.2 – *Pokračování z předešlé stránky*

Typy ICMP hlášení	**Obsah zprávy**	**Query (Q) / Error (E)**
13	*Timestamp request* = žádost o označení času v datagramu (pro synchronizaci, zajištění doby cesty datovou sítí). Stanice A pošle stanici B žádost o zaslání (jejího) aktuálního času formou ICMP zprávy *Timestamp request* s vyplněnou položkou *Originate timestamp* obsahující její aktuální čas těsně před vysláním. Stanice B vyplní okamžitě po přijetí této žádosti položku *recive timestamp* a těsně před vysláním ICMP odpovědi *Timestamp reply* i položku *transmit time* příslušným aktuálním časovým údajem. Z rozdílu časových údajů může pak stanice A odhadnout dobu přenosu.	Q
14	*Timestamp reply* = odezva na žádost o označení času	Q
15	*Information request* = žádost o adresu sítě (tato zpráva je v současné době považována za zastaralou, neboť existují speciální protokoly, např. RARP, BOOTP nebo DHCP, které plně nahrazují její funkci)	Q
16	*Information reply* = odpověď na žádost o adrsu sítě	Q
17	*Address mask request* = žádost o informaci o podsíťové masce ICMP	Q
18	*Address mask reply* = odpověď na žádost o informaci o síťové masce	Q

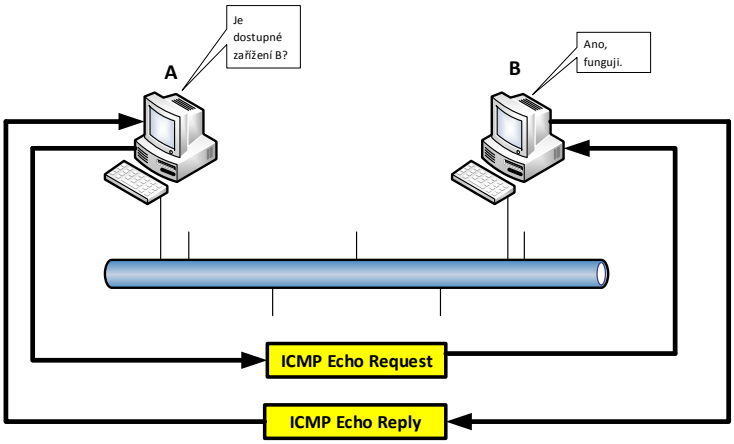
Tab. 1.3: Nejčastější typy ICMP zpráv [2]

Typ	**Kód**	**Význam**
0	0	odpověď na ping
3	0	cílová síť nedostupná
3	1	cílový protokol nedostupný
3	2	cílová stanice nedostupná
3	3	cílový port nedostupný
3	4	byla potřeba fragmentace, ale datagram ji specificky zakazoval
3	13	datagram administrativně zakázán
4	0	informace o zahlcení (podle RFC 1122 tuto zprávu může generovat stanice; podle RFC 1812 by ji neměli posílat směrovače)
5	0	přesměrovat datagram pro síť
5	1	přesměrovat datagram pro stanici
5	2	přesměrovat datagram pro síť a typ služby
5	3	přesměrovat datagram pro stanici a typ služby
8	0	ping
11	0	překročena životnost datagramu
11	1	překročen čas pro znovusestavení fragmentů do původního datagramu
12	0	problém s parametry datagramu

1.2 Příkaz PING

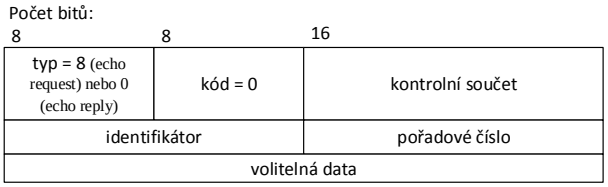
Protokol ICMP není pouze pro informace o chybách v datagramech a jejich směrování, ale slouží ke zjišťování doplňujících informací a také k ověření komunikace mezi stanicemi. Právě posledně jmenované možnosti využívá tzv. Ping (Packet Internet Groper) pro nejjednodušší zjištění dosažitelnosti cílové stanice/sítě (ve srovnání s jinými obdobnými možnostmi, jako Telnet nebo traceroute příkaz ping pracuje na nejnižší možné vrstvě, u cílové stanice je implementován v jádře operačního systému).

Na základě vysílání zpráv ICMP typu *Echo request* s adresou IP cílové stanice (případně s všeobecnou adresou pro danou síť) se očekává od cíle odpověď potvrzující funkčnost a dosažitelnost stanice (pro budoucí komunikaci), ICMP *Echo reply*, viz obrázek 1.3.

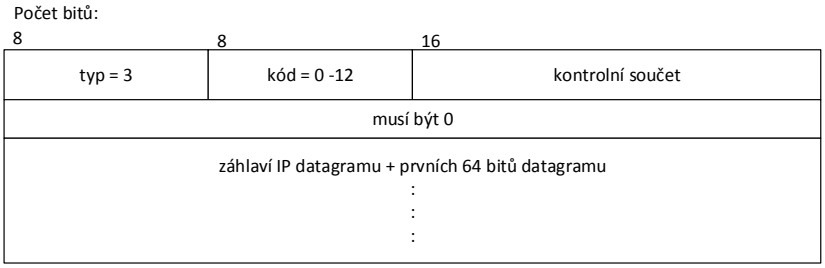


Obr. 1.3: PING

Formát zprávy *Echo* je uveden na obrázku 1.4; formát zprávy o nedostupnosti cíle je na obrázku 1.5



Obr. 1.4: Formát zprávy *Echo request* a *Echo replay*



Obr. 1.5: Formát ICMP zprávy *Destination unreachable*

1.2.1 Dostupnost v různém podání

Příkaz ping nedokáže sám o sobě „zajistit“ dostupnost vzdálené stanice pro budoucí komunikaci, pouze v pozitivním případě ukazuje na aktivní soubor protokolů TCP/IP na druhé straně, rep. pouze na funkční síťovou vrstvu.

Vždy je třeba rozlišovat mezi typy dostupnosti nebo nedostupnosti cíle (parametry zprávy ICMP *Echo reply*, viz tabulka 1.4):

- **nedostupná síť** – informace od směrovače na cestě k cíli, který cílovou síť nemá ve své tabulce; důvodem může být neexistence adresy sítě, její nedostupnost (oddělení od ostatní sítě nebo nefunkčnost), nebo špatná informace v tabulce směrovače; ohlašuje problémy se směrováním sítě
- **nedostupná stanice** – informace od posledního směrovače na cestě k cílové síti, který zjistí nedostupnost stanice (adresa stanice na síti neexistuje, stanice se nehlásí, nebo pro nějaký problém neodpovídá); ohlašuje problémy s doručením datagramu;

- **nedostupný port** – informace od cílové stanice, která je funkční, ale dotazovaný port (aplikační protokol) není dostupný (nesprávně označení portu, neexistence podpory protokolu).

Tab. 1.4: Kódy hlášení protokolu ICMP *Echo reply* [2]

Kód	Význam
0	*Network unreachable* = síť je nedosažitelná
1	*Host unreachable* = stanice je nedosažitelná
2	*Protocol unreachable* = protokol není k dispozici
3	*Port unreachable* = port je nedosažitelná
4	*Fragmentation needed and DF set* = návěst DF (don't fragment – zákaz fragmentace) v datagramu nastavena a přitom nutno fragmentovat
5	*Source route faild* = odesílatelem požadované směrování selhalo
6	*Destination network unknown* = cílová síť není známa
7	*Destination host unknown* = cílový počítač není znám
8	*Source host isolated*
9	*Komunikace s cílovou sítí administrativně zakázaná*
10	*Komunikace s cílovou stanicí administrativně zakázaná*
11	*Network unreachable for type of service*
12	*Host unreachable for type of service*

1.2.2 Porozumění významu ping

Pokud směrovač nevyšle zprávu o nedoručitelnosti datagramu, neznamená to, že byl datagram skutečně doručen. Ani směrovače nejsou schopny detekovat všechny případy, kdy datagram není doručen. Příkladem je cílová stanice připojena k síti Ethernet, která pro nějaký problém datagramy nepřijímá. Směrovač připojený k Ethernetu jí bude datagramy posílat, protože na této síti neexistuje žádný mechanizmus potvrzení přijetí datagramu a kromě problému s danou stanicí je síť plně funkční.

I v případě, že ping provede úspěšně, může při skutečných aplikacích nastat poslední zmíněný případ nebo může dojít k filtrování skutečných zpráv, takže výsledná komunikace nemusí být úspěšná.

Příkaz ping může správci jednoduché sítě napomoci zjistit, zda není zahlcená. Zpoždění, které vidíme ve výsledné odpovědi, v sobě zahrnuje zpoždění způsobené

paketizací/serializací, vlastním přenosem signálu po médiu, apod. Zprávy ping nabírají zpoždění také při každém zpracování směrovačem a ve frontách na vstupu a na výstupu, protože se typicky nezpracovávají prioritně, spíše naopak (objem ICMP zpráv může být na směrovači administrativně omezen).

Firewalls mohou *ping* blokovat. Na rozsáhlých sítích proto není *ping* dobrým mechanizmem pro testování zátěže. Ping ale nikdy nelze ztotožnit s rychlostí reakce konkrétní aplikace.

1.2.3 Příkaz ping

Typická podoba příkazu **ping** a nejběžnější volitelné možnosti vypadají následovně:
ping [-q] [-v][-R][-c Count][-i Wait][-s PacketSize] Host
kde:
-q **Quiet output** – nezobrazuje se nic kromě souhrnu při zahájení a zakončení;
-v **Verbose output** – výmluvný výstup, který zobrazuje obdržené ICMP zprávy vedle *Echo reply*
-R **Record route option** – zahrnuje volitelnou možnost RECORD ROUTE *Echo request* a zobrazuje vyrovnávací paměť vrácených datagramů;
-c **Count** – specifikuje počet zpráv *Echo request* k odesílání (implicitně lze ping přerušit kombinací kláves control-C);
-i **Wait** – specifikuje, kolik sekund se má počkat před vysláním dalšího datagramu (implicitně 1s);
-s **PacketSize** – označuje počet oktetů v části dat ICMP zprávy (implicitní hodnota je 56 oktetů, to znamená datagram o délce 64 oktetů, včetně 8 oktetů záhlaví ICMP);
Host – specifikuje IP adresu nebo jméno cílového systému.

1.3 Příkaz Tracert

Program traceroute slouží k analýze počítačové sítě. Vypisuje uzly (resp. směrovače) na cestě datagramu od zdroje až k zadanému cíli. uzly jsou zjišťovány pomocí snížení hodnoty TTL v hlavičce datagramu.

1.3.1 Popis Funkce

Traceroute zvyšuje hodnotu „time to live“ (TTL) po každém úspěšném odeslání balíčku paketú. První tři pakety mají hodnotu TTL nastavenou na 1 (odesílají se současně), další tři pakety mají hodnotu TTL 2 atd. při cestě k cíli paket prochází jednotlivými směrovači (uzly). Při průchodu směrovač sníží hodnotu TTL o 1 a pošle ho dál. Je-li hodnota TTL paketu nula a není v cílové IP síti, pak je paket zahozen a směrovač pošle chybnou ICMP zprávu odesílateli. Traceroute využívá právě těchto chybových hlášení, aby sestavil tabulku cesty paketu od odesílatele k cíli. Ve výpisu jsou tak zobrazeny všechny uzly, které položku TTL snižují (tj. směrovače).

1.3.2 Použití

Minoritně se používá pro zjišťování problémů se sítí. Díky výpisu jednotlivých uzlů, přes které paket prochází, se zjistí přesná cesta k počítači nebo nějaké stanici v síti. Toto pomáhá identifikovat problémy s routry nebo firewally, které mohou blokovat přístup do sítě.

1.3.3 Bezpečnost

Traceroute zpřístupňuje velmi detailní informace o jednotlivých bodech na cestě k nějakému cíli v síti. Na počátcích používaní internetu to bylo považováno za přijatelné, ale s následujícími problémy to vyvolalo debatu ohledně bezpečnosti a ochraně soukromých informacích. Traceroute začali totiž hojně využívat hackeři. Získávali tak podstatné informace o síťové architektuře různých společností. Pomocí použití příkazu traceroute mohli hackeři rychle zmapovat uzly, které daná společnost měla k dispozici, a určit si tak slabý bod, který mohli prolomit.

Kvůli těmto důvodům mnoho správců sítí zablokovalo odesílání programu traceroute ze svých routerů, proto některé uzly na cestě neodpovídají a trasování často končí na hranici lokální sítě.

2 VOLBA PLATFORMY

Tato kapitola se zabývá volbou vhodné platformy pro realizování sítového terminálu.

Mezi vhodné platformy pro realizování síťového terminálu patří Raspberry Pi, Banana Pi a Arduino. V tabulce 2.1 porovnávám jejich vlastnosti. Pro Bakalářskou práci jsem zvolil platformu Banana Pi, jelikož semestrální projekt je založený na platformě Arduino, tak zde uvedu pár jeho vlastností a verzí.

Tab. 2.1: Porovnání platforem

Název	**Arduino Mega 2560**	**Raspberry Pi B**	**Banana Pi M2**
Processor	ATmega2560	ARM11	4x Cortex-A7
Frekvence	16 MHz	700 MHz	1GHz
SoC	–	Broadcom	Allwinner
GPU	–	VideoCore	PowerVR
Ram	8 kB	512 mB	1 Gb
Memmory	256 kB	–	–
GPIO	54	26	40
Analog	16	–	–
Ethernet	+	10/100 Mb	10/100/1000Mb

2.1 Arduino

Vývoj prvního Arduina započal v roce 2005, když se lidé z italského Interaction Design Institute ve městě Ivrea rozhodli vytvořit jednoduchý a levný vývojové set pro studenty, kteří si nechtěli pořizovat, v té době rozšířené a drahé desky BASIC Stramp. Mezi studenty se Arduino uchytilo, a tak se tvůrci rozhodli poskytnout ho celému světu. (V roce 2010 vznikl zajímavý dokument o vzniku Arduina s rozhovory jeho tvůrci: Arduino The Documentary (2010) English HD). A to nejenom prodejem vlastních desek, ale i sdílením všech schémat a návodů (jedná se o Open Source projekt [3]).

Programová část Arduina byla založena na Processing, což je programovací jazyk s vlastním editorem, určený k výuce programování. Důkazem, že tato platforma není mrtvá, může být i to, že nedávno byl ohlášen vývoj nové a výkonné desky Arduino Galileo, která vzniká ve spolupráci s Intelem. Za osm let vývoje již vzniklo spoustu různých typů Arduina, Jelikož se jedná o opensource projekt, vznikalo společně s hlavní linií projektů i spoustu dalších, neoficiálních typů, takzvaných klonů.

2.2 Raspberry pi

Raspberry Pi je „mini“ počítač, který využívá architekturu ARM. Na trh se dostal rokem 2011. Od té doby vzniklo několik verzí, které se od sebe odlišují velikostí paměti, počtem USB portů, ethernetovým portem aj.

Raspberry Pi bylo vyvinuto britskou nadací Raspberry Pi Foundation pro vyuku informatiky na školách. Raspberry má velké množství výběru operačního systému. Nejčastěji se používá za OS Raspbian což je upravený Linuxový operační systém přímo pro Raspberry Pi. V současné době se dá využít jako malý nenáročný server nebo přenosný počítač. Dá se využít jako domácí multimediální centrum, nebo jako NAS. Ale asi nejčastější použití má jako v automatizaci domácnosti.

2.3 Banana Pi

Banana Pi je jednodeskový mini počítač velmi podobný populárnímu Raspberry Pi nebo Orange Pi a předchůdce nového modelu Banana Pro od společnosti Lemaker s podporou WiFi. Vychází z velmi podobné architektury, avšak nabízí mnohem větší výkon. Základem počítače je dvoujádrový procesor ARM Cortex-A7 s integrovaným GPU jádrem Mali400MP2. Deska obsahuje integrovanou operační paměť 1 GB DDR3. Deska neobsahuje žádnou interní paměť pro operační systém nebo ukládání souborů, nabízí však možnost připojení SD karty či SATA rozhraní pro připojení pevného disku. K připojení zobrazovací jednotky slouží konektor HDMI nebo kompozitní RCA. Zvukový výstup lze napojit skrze 3,5 mm JACK nebo HDMI. Na desce se nachází také integrovaný mikrofon. Oproti Raspberry Pi má Banana Pi přímo napojený ethernet adaptér 10/100/1000 s konektorem RJ45. Základní deska obsahuje dva konektory USB 2.0 včetně jednoho konektoru USB OTG a jednoho konektoru USB micro sloužícího pro napájení.

Celý počítač se rozměry 90 mm x 60 mm a rozložením konektorů velmi podobá Raspberry Pi a je kompatibilní s většinou podporovaných příslušenství určených právě k Raspberry Pi. Najdeme zde i sadu 26 programovatelných pinů pro připojení rozšiřujících modulů a konektor na připojení LCD dotykového panelu. Součástí desky jsou i tři tlačítka - zapnutí, reset a bootování. Deska obsahuje IR přijímač pro dálkové ovládání.

Základní deska počítače Banana Pi je napájena pomocí napájecího adaptéru 5V s doporučeným výstupním proudem 2A.

2.3.1 Historie

Projekt Banana Pi vznikl na počátku roku 2013 jako reakce na vznikající komunitu Raspberry Pi s odezvou na poptávaný větší výkon a s lépe řešeným připojením k počítačové síti prostřednictvím ethernetu s podporou rychlosti 10/100/1000. Nevýhodou ethernetového rozhraní Raspberry Pi byl fakt, že toto rozhraní bylo připojeno interně jako další USB periferie, což zvyšovalo nároky na další systémové prostředky.

2.3.2 Operační systém

Banana Pi podporuje tak jako Raspberry Pi následující operační systémy:

- Linux (Banana Pi OS, Raspbian, Debian, Arch-Linux, Ubuntu a další upravené verze)
- Android

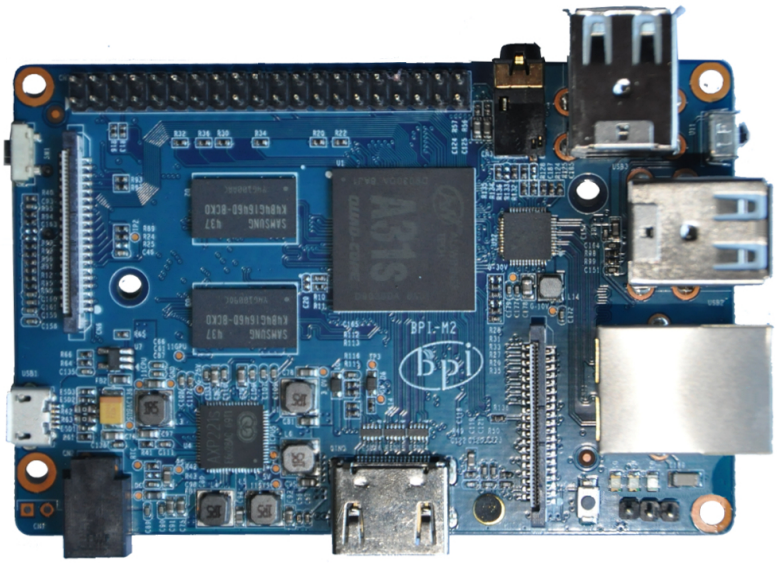
2.3.3 Využití

V současné době se dá využít jako malý nenáročný server nebo přenosný počítač. Kvůli hojné komunitě uživatelů na internetu vznikají různé projekty. Například domácí NAS server nebo dá se využít v domácí automatizaci. Zajímavým projektem může být například i sestavení výkonného clusteru, který již může konkurovat silnějším počítačovým sestavám při zachování nízké spotřeby s vysokým výkonem.

2.3.4 Banana Pi M2

Čtyřjádrový ARM Cortex-A7 1GHz Quad Core, PowerVR SGX544MP2, 1GB DDR3, 1x Gigabit LAN, 1x WiFi, 4x USB, HDMI...

Tento model disponuje čtyřjádrovým procesorem ARM Cortex-A7 s čipsetem PowerVR SGX544MP2. Oproti prvnímu modelu má již slot na Micro SDHC karty (odpadá použití redukce nebo velké SD karty). Novinkou je také podpora integrovaného bezdrátového adaptéru WiFi B/G/N. V rámci tohoto rozšíření naopak tento model postrádá konektor SATA.



Obr. 2.1: Banana Pi M2

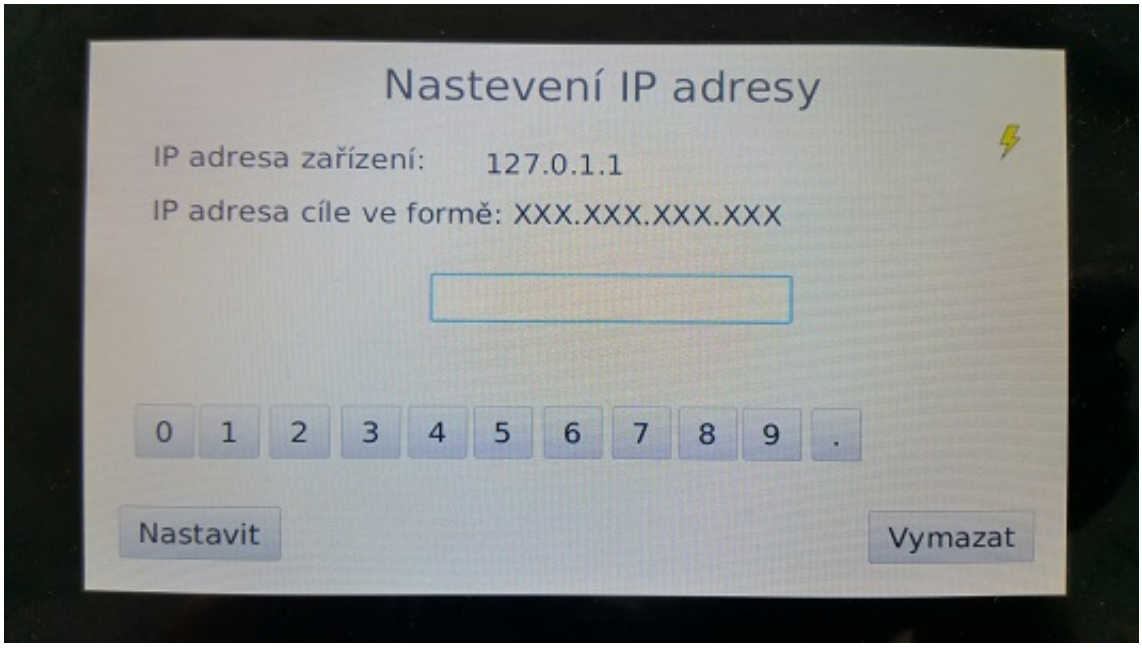
3 REALIZACE

Tato kapitola se zabývá realizací celého projektu. Realizace projektu je na zařízení Banana Pi M2 s dotykovým displejem. Operační systém Banana Pi je verze Raspbianu určená pro LCD displej s dotykovou plochou. Celý program je naprogramován v jazyce Java. Je rozdělena do několika částí. První část se věnuje zadání cílové IP adresy a základního menu. Druhá část se zabývá naprogramování a odesílání běžného pingu. Třetí část se zabývá naprogramování rozšířeného pingu, kde se mění a upravují jeho vlastnosti. Čtvrtá část se zabývá naprogramováním nekonečné smyčky pingu, který se dá ukončit zásahem uživatele. Poslední části této kapitoly se zabývá naprogramováním traceroutu.

Celé ovládání projektu je řešeno pomocí dotykového displeje.

3.1 Základní menu a zadání IP adresy

Tato část se zabývá zadáním cílové IP adresy koncového zařízení (uzlu). Zadání IP adresy je řešeno tak, že po stisknutí čísla na obrazovce se číslo zapíše do TextFieldu a je ve formátu string. Tato hodnota se předává do další třídy programu. Ip adresa se předává mezi jednotlivými třídami pomocí ukazatelů. Celý zdrojový kód se nachází v příloze A.4 na stránce 51.



Obr. 3.1: Obrazovka zadání IP adresy

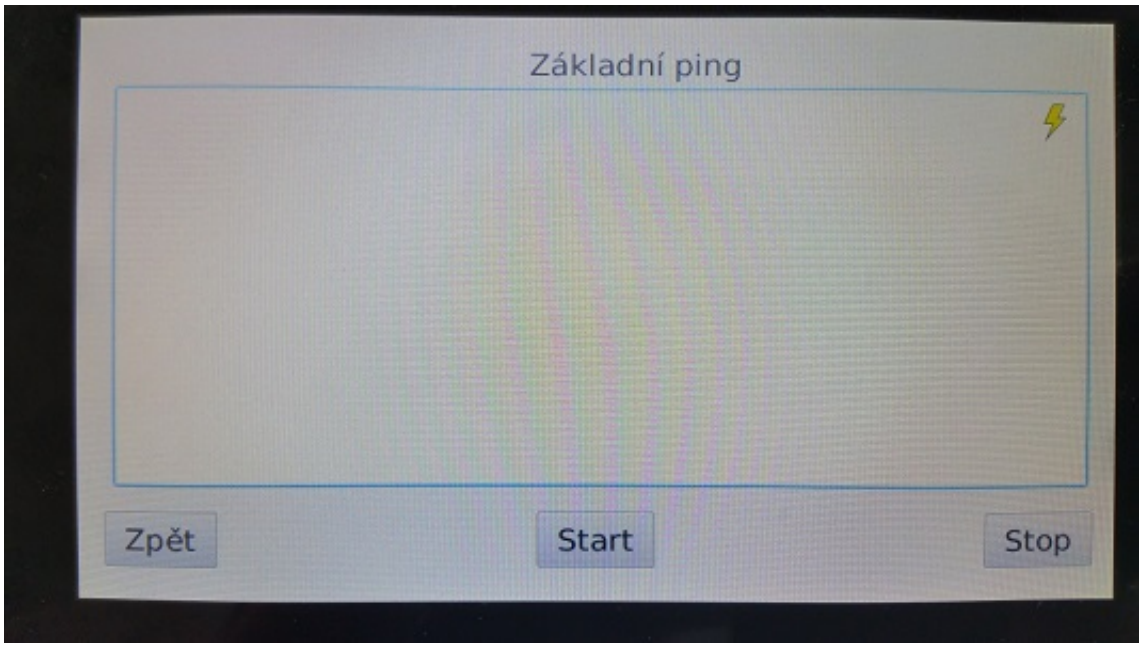
Pomocí tlačítka Clear se TextField vymaže a můžeme znovu zadat novou IP adresy. Pokud se nezadá žádná IP adresa, tak se objeví na obrazovce nové okno,

které upozorňuje, že se musí zadat IP adresa. Po stisknutí tlačítka Set se zobrazí hlavní menu programu. Zdrojový kód hlavního menu je v příloze Celý zdrojový kód najdete v příloze A.6 na stránce 55

Obr. 3.2: Menu programu

3.2 Základní příkaz ping

Tato část se věnuje naprogramování základního příkazu ping. Příkaz ping je řešen tak, že do ArrayListu se přidají požadované parametry jako je IP adresa cílového zařízení a počet celkových odeslaných parametrů. V základním pingu se celkem odesílá 5 paketů. A ty se následně vyhodnocují. Až jsou všechny parametry nastaveny, tak se volá doCommand. V doCommand je ProcessBuilder, který spouští zadané příkazy z ArrayListu v příkazové řádce a vypisuje hodnoty do TextArea.



Obr. 3.3: Základní ping

Výpis 3.1: Zdrojový kód příkazu ping

```
                                                                    1
public void startControll(ActionEvent event)                       2
        throws IOException {                                        3
                                                                    4
        List<String> commands = new ArrayList<>();                  5
        commands.add("ping");                                       6
        commands.add(ip);                                           7
        commands.add("-c");                                         8
        commands.add("5");                                          9
        doCommand(commands);                                        10
    }                                                               11
public void doCommand(List<String> command)                        12
        throws IOException {                                        13
                                                                    14
        ProcessBuilder pb = new ProcessBuilder(command);            15
        this.process = pb.start();                                  16
                                                                    17
        Reader reader = new InputStreamReader                       18
                        (process.getInputStream());                 19
        ProcessThread ort = new ProcessThread                       20
                            (reader, console);                      21
        ort.start();                                                22
```

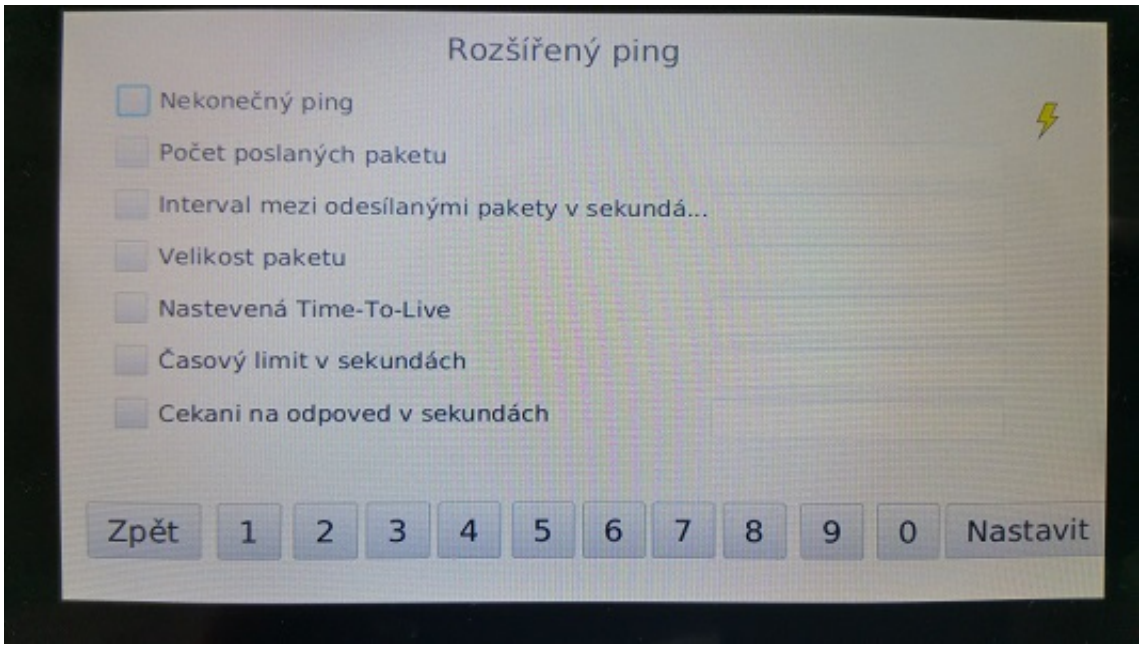
```
                                                                    23
    }                                                               24
```

Reader čte vypsané hodnoty z console a pomocí ProcessThreadu , který je jedno vlákno programu, tak vypisuje hodnoty do TextArea. Když reader přečte hodnotu z příkazového řádku tak ho automaticky zapiše do TextArea. Je tu vlákno programu, protože bez vlákna by program čekal než proběhne celý příkaz ping a pak by se teprve vypsal do TextArea. Vlákno zajišťuje automatickou aktualizaci TextArea. Celý zdrojový kód najdete v příloze A.7 na stránce 60.

Výpis 3.2: Zdrojový kód ProcessTread

```
static class ProcessThread extends Thread {                         1
                                                                    2
        private Reader reader;                                      3
        private TextArea ta;                                        4
                                                                    5
        public ProcessThread(Reader reader, TextArea ta) {          6
            this.reader = reader;                                   7
            this.ta = ta;                                           8
            this.setDaemon(true);                                   9
        }                                                           10
                                                                    11
        @Override                                                   12
        public void run() {                                         13
                                                                    14
           try {                                                    15
             BufferedReader br = new BufferedReader(reader);        16
             String cr = null;                                      17
             while ((cr = br.readLine()) != null) {                 18
                 if (cr.length()>0) {                               19
                    ta.setText(ta.getText() + "\n" + cr);           20
                    ta.selectPositionCaret(ta.getLength());         21
                    ta.deselect();                                  22
                    }                                               23
                }                                                   24
             } catch (IOException ex) {                             25
                 ex.printStackTrace();                              26
             }                                                      27
                                                                    28
        }                                                           29
    }                                                               30
```

3.3 Rozšířený příkaz ping



Obr. 3.4: Natavení parametrů pinu

Rozšířený ping je podobný jako základní ping, ale má dvě obrazovky. V první se mění a nastavují parametry příkazu ping a v druhé se vyhodnocují. Druhé okno je podobné jako u základního příkazu ping, ale před přidáním hodnoty do ArrayListu se testuje jestli se daný parametr přidává nebo je v základním nastavení. Pokud není v základním nastavení, tak se přidá do ArrayListu s danou hodnotou.

V nastavení se dá nastavit počet odeslaných paketů, interval mezi jednotlivými pakety, velikost paketu, délka životnosti paketu (TTL), dá se nastavit timeout, čekací doba na odpověď paketu. Je zde i možnost nekonečného posílání paketů. Celý zdrojový kód najdete v příloze A.9 na stránce 70.

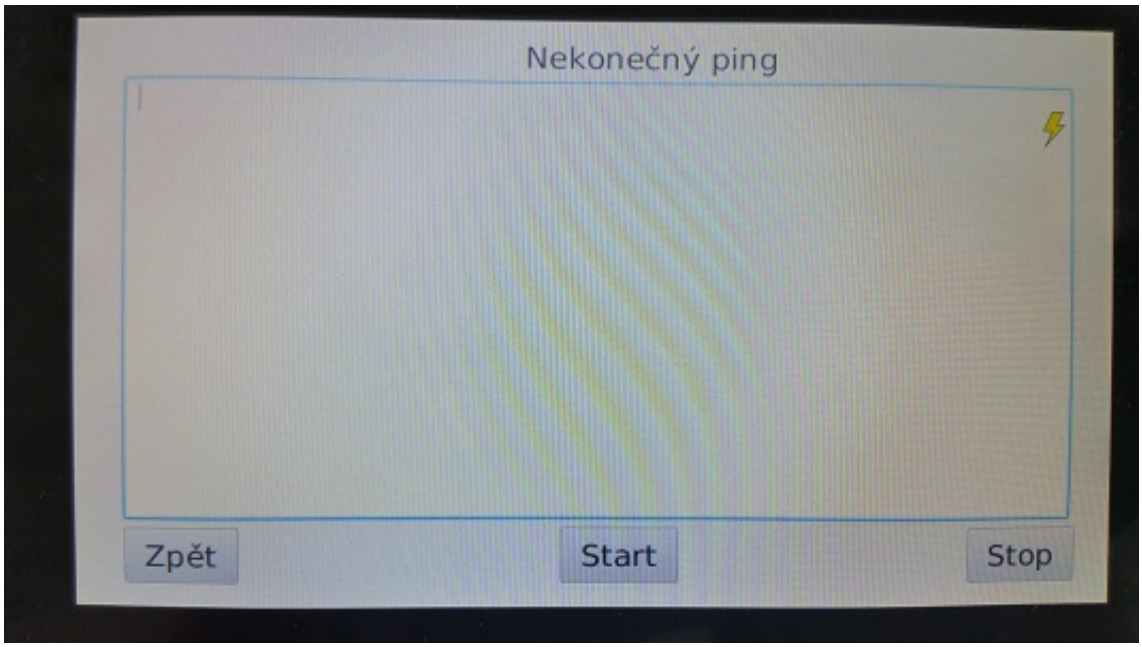
Výpis 3.3: Zdrojový kód rozsireneho pingu

```
public void startControll(ActionEvent event)                    1
        throws IOException {                                     2
                                                                 3
        List<String> commands = new ArrayList<>();               4
        commands.add("ping");                                    5
        commands.add(ip);                                        6
         if (counter == true)                                    7
        {                                                        8
            commands.add("-c");                                  9
            commands.add(counterText);                           10
```

```
  }                                                                    11
        if (intevals == true)                                          12
        {                                                              13
            commands.add("-i");                                        14
            commands.add(intervalsText);                               15
  }                                                                    16
        if (packetSize == true)                                        17
        {                                                              18
            commands.add("-s");                                        19
            commands.add(packetSizeText);                              20
  }                                                                    21
        if (ttl == true)                                               22
        {                                                              23
            commands.add("-t");                                        24
            commands.add(ttlText);                                     25
  }                                                                    26
        if (deadline == true)                                          27
        {                                                              28
            commands.add("-w");                                        29
            commands.add(deadlineText);                                30
  }                                                                    31
        if (timeout == true)                                           32
        {                                                              33
            commands.add("-W");                                        34
            commands.add(timeoutText);                                 35
  }                                                                    36
        doCommand(commands);                                           37
    }                                                                  38
public void doCommand(List<String> command)                           39
        throws IOException {                                           40
                                                                       41
        ProcessBuilder pb = new ProcessBuilder(command);               42
        this.process = pb.start();                                     43
                                                                       44
        Reader reader = new InputStreamReader                          45
                        (process.getInputStream());                    46
        ProcessThread ort = new ProcessThread                          47
                            (reader, console);                         48
        ort.start();                                                   49
    }                                                                  50
```

3.4 Nekonečný ping

Nekonečný ping funguje na principu nekonečného posíláni paketu ping, při kterém nečeká na odpověď koncové stanice. Tento druh pingu je občas nazýván Death ping. Tento ping právě využívají hackeři k zahlcení koncové stanice. Kvůli tomuto je doporučováno aby příkaz ping byl blokován v sítích, tak aby nedocházelo k útokům na stanice.



Obr. 3.5: Nekonecny ping

Nekonečný ping se zadává v linuxu tak že nemá zadaný počet paketů, které se mají odeslat. V operačním systému Windows se právě musí za příkaz ping zadat parametr na nekonečný ping. Celý zdrojový kód najdete v příloze A.10 na stránce 76.

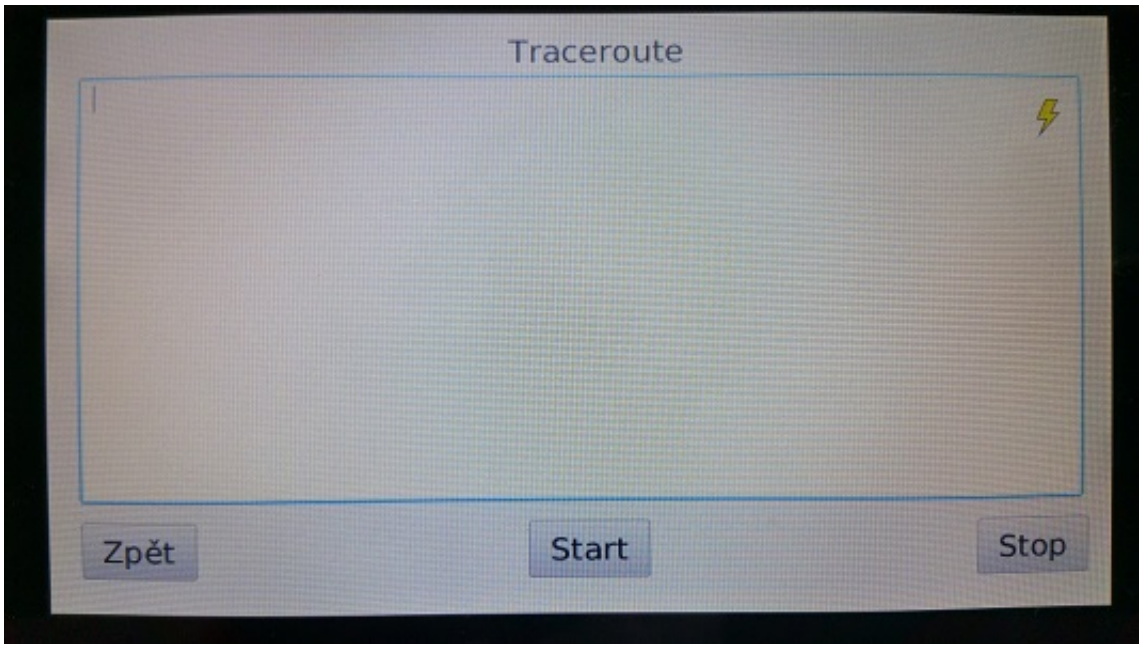
Výpis 3.4: Zdrojový kód nekonecneho ping

```
public void startControll(ActionEvent event)                    1
        throws IOException {                                    2
                                                                3
        List<String> commands = new ArrayList<>();              4
        commands.add("ping");                                   5
        commands.add(ip);                                       6
        doCommand(commands);                                    7
    }                                                           8
public void doCommand(List<String> command)                     9
        throws IOException {                                    10
```

```
                                                                    11
        ProcessBuilder pb = new ProcessBuilder(command);            12
        this.process = pb.start();                                  13
                                                                    14
        Reader reader = new InputStreamReader                       15
                        (process.getInputStream());                 16
        ProcessThread ort = new ProcessThread                       17
                            (reader, console);                      18
        ort.start();                                                19
    }                                                               20
```

3.5 Příkaz traceroute

Příkaz traceroute nebyl součástí zadání bakalářské práce, ale patří mezi jedny z nejzákladnějších příkazů na testování stanic v síti a na tvorbu tabulky s zapojenými směrovači (routry) v síti, přes které se směruje paket k cílové stanici. Celý zdrojový kód najdete v příloze A.11 na stránce 79.



Obr. 3.6: Příkaz traceroute

Výpis 3.5: Zdrojový kód traceroute

```
public void startControll(ActionEvent event)                    1
        throws IOException {                                    2
                                                                3
        List<String> commands = new ArrayList<>();              4
        commands.add("traceroute");                             5
        commands.add(ip);                                       6
        doCommand(commands);                                    7
    }                                                           8
public void doCommand(List<String> command)                     9
        throws IOException {                                    10
                                                                11
        ProcessBuilder pb = new ProcessBuilder(command);        12
        this.process = pb.start();                              13
                                                                14
        Reader reader = new InputStreamReader                   15
```

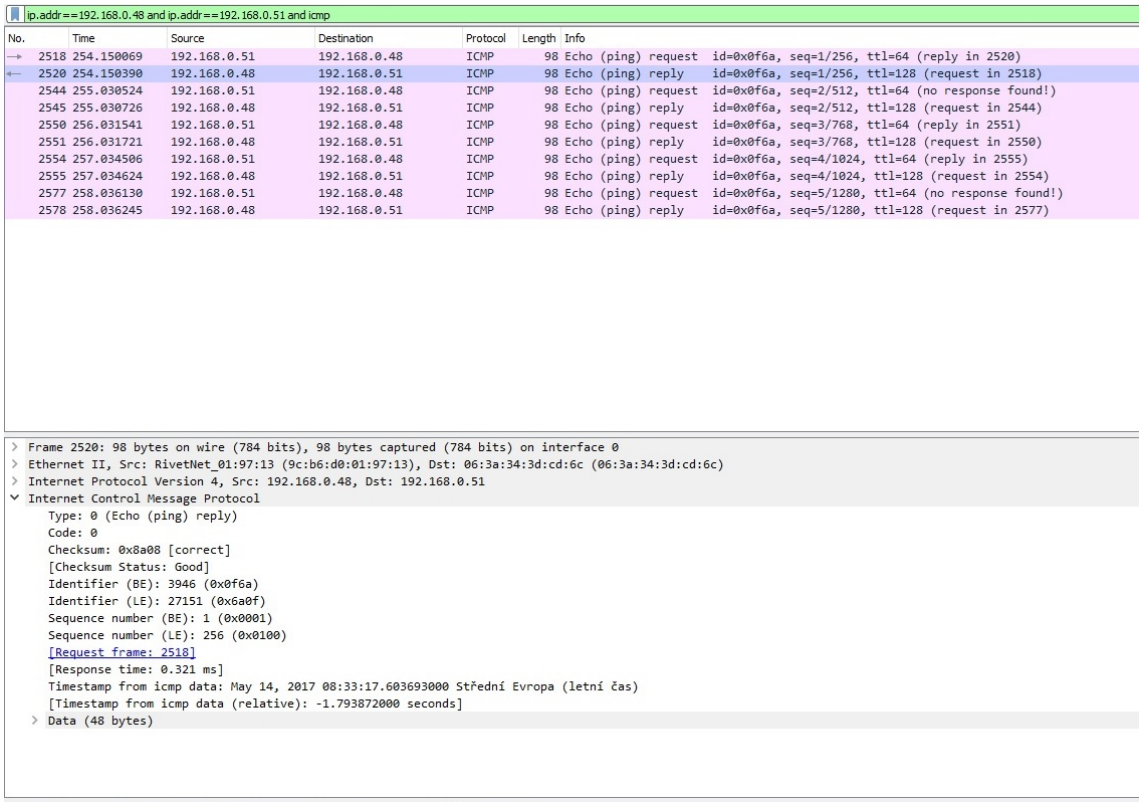
```
                    (process.getInputStream());
        ProcessThread ort = new ProcessThread
                            (reader, console);
        ort.start();
    }
```

4 VÝSLEDKY

Tato kapitola se zabývá zhodnocením naměřených výsledků. V první části se budu věnovat základnímu pingu a následně pingu s upravenými vlastnostmi. V poslední části se budu věnovat nekonečnému pingu.

4.1 Základní ping

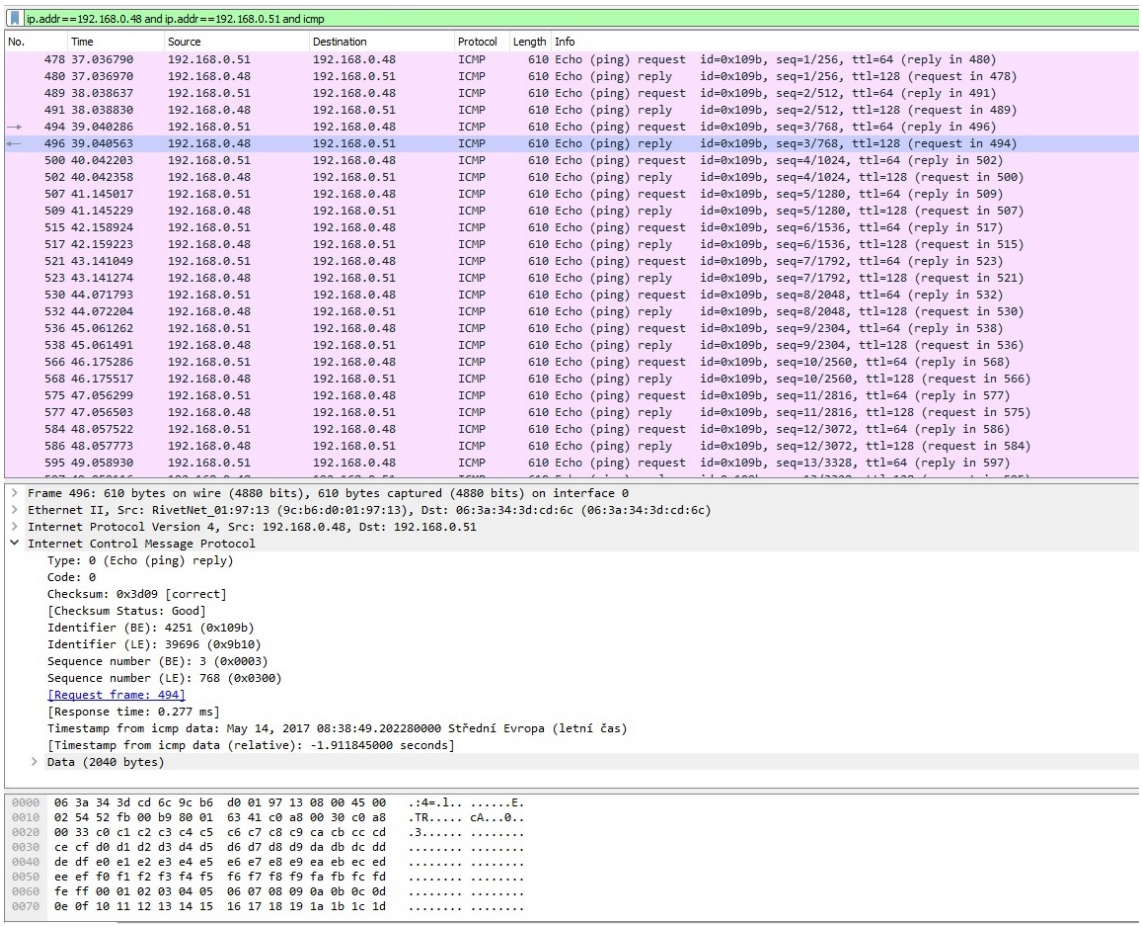
Základní ping je nastavený na 5 celkem odeslaných paketů. Z výsledků vyplývá že celkem odešlo 5 paketů, ale některé ping (request) nebyly zachycený, protože se nestačily zachytnout a proběhly rychle. Na výpisu obrazovky výsledků se zobrazily, že byly přijaty a žádný paket nebyl zahozen.



Obr. 4.1: Základní ping

4.2 Rozšířený ping

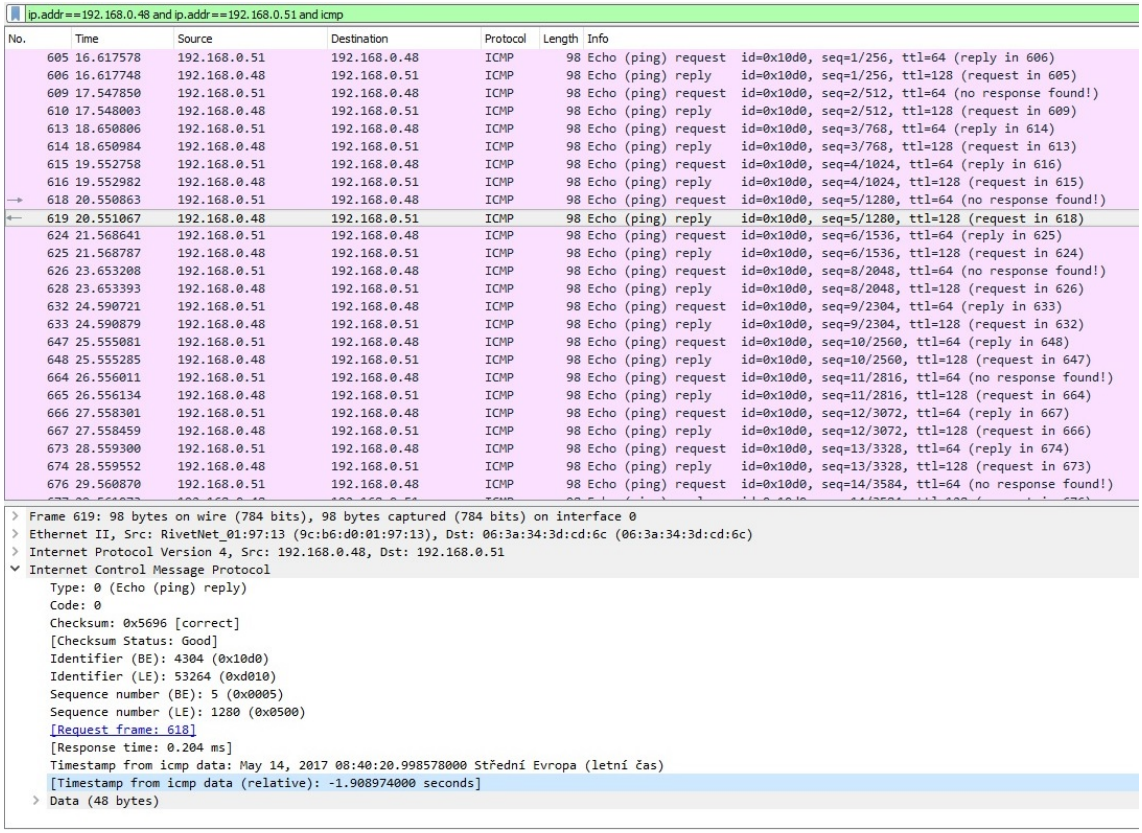
Na rozšířený ping jsem nastavil velikost celého paketu na 2048 B a celkový počet odeslaných paketů na 15. Z naměřených výsledků z programu Wireshark vyplývá že celkový počet paketů se shoduje s nastaveným počtem odeslaných paketů. Velikost paketu odpovídá nastavené hodnotě.



Obr. 4.2: Ping s upravenyma vlastnostma

4.3 Nekonečný ping

Nekonečný ping byl nastaven defaultně, protože využívám platformu Linux a tam je ze základu nastaven ping na nekonečný, takže jsem u něho neupravoval vlastnosti. nekonečný ping jsem měl spuštěný asi 10 minut, abych si ověřil jeho funkčnost. Z výsledků co mi vyšly z programu Wireshark vyplývá že nekonečný ping probíhal v pořádku jen se mi nepodařilo zachytit některé (ping) request. Ale z toho co jsem pozoroval na obrazovce zařízení, tak ty pakety byly úspěšně přijaty. Nekonečný ping se mi podařilo zdárně ukončit.



Obr. 4.3: Nekonečný ping

5 ZÁVĚR

Tento práce mi pomohl se ponořit do problematiky počítačových sítí a pomohl mi hlouběji pochopit ICMP protokol. Celý program původně byl navrhnut na Arduino, ale vyskytly se chyby v použitém naprogramovaném protokolu ICMP. Po úpravách protokolu ICMP pro Arduino se začaly vyskytovat chyby v předávání parametrů z jedné třídy do druhé a nepodařilo se mi to opravit. Při programování na Arduino jsem se radil s lidmi na různých fórech na internetu a dospěli jsem k závěru, že programovat složitý síťový terminál na Arduino je velice nepohodlné a nepraktické.

Chtěl bych se omluvit, že nemám v textu moc citací. To je způsobeno tak, že knížky, které jsem měl vypůjčené k tvorbě této bakalářské práce jsem musel vrátit a když jsem si je chtěl znovu půjčit, abych mohl upravit text, tak už byly půjčené někomu jinému a nestihl bych odevzdat práci v čas. Proto se tu omlouvám, že v tecte nemám moc citací.

Jelikož naprogramovat síťový terminál na Arduino bylo zcela nepraktické, jsem začal uvažovat o náhradě za Arduino a našel jsem Raspberry Pi. Které jsem ze začátku práce uváděl jako náhradní zařízení na které by se dal naprogramovat síťový terminál. Dále jsem našel na internetu zařízení, které je podobné Raspberry Pi a to je Banana Pi. Banana Pi se liší od Raspbery Pi velice málo a má gigabytový ethernetový port, který se dá použít pro měření v rychlých počítačových sítích. Než jsem dostal zařízení Banana Pi s dotykovým displejem tak jsem celý program zprovoznil na zařízení Raspberry Pi.

Po obdržení zařízení BananaPi jsem začal zprovozňovat celý projekt. Zařízení jsem sestavil a naprogramoval a otestoval. Funkčnost této práce dokazují přiložené výpisy z programu Wireshark. Bohužel se mi z nějakého důvodu nepodařilo zachytit získávání IP adresy od DHCP serveru, takže bohužel ji nemohu zveřejnit jak byla získána.

Při kompletaci práce se vyskytly problémy s zařízením Banana Pi, které při připojení do elektrické sítě začalo bzučet a dotykový displej nereagoval. Později jsem si všiml, že kabel, který propojuje Banana Pi s dotykovým displejem je poškozený. Domnívám se, že i Banana Pi je poškozené.

Na závěr bych chtěl dodat, že ping ani tracerout nemusí být v některých počítačových sítích povolen kvůli bezpečnosti. Ping a tracerout je hojně využíván k prolomení bezpečnosti počítačových sítích. Před použitím v dané počítačové síti doporučuji kontaktovat správce dané sítě a zeptat se ho jestli je v dané síti povolen ping nebo je zakázaný.

LITERATURA

[1] POSTEL, J. *Internet Control Mesage Protocol, Network Working Group, RFC 792, 1981.*

[2] PUŽMANOVÁ, Rita TCP/IP v kostce. 2., upr. a rozš. vyd. České Budějovice: KOPP,2009 s. ISBN 978-80-7232-388-3

[3] VODA, Z. *Průvodce světem Arduina*, cit.[2015-10-19], dostupné online: <`http://arduino.cz`>

[4] Herout, Pavel Učebnice jazyka Java, KOPP, České Budějovice, 2013 s. ISBN 978-80-7232-398-2

[5] Herout, Pavel Java - grafické uživatelské prostředí a čeština, KOPP, České Budějovice, 2012 s. ISBN 978-80-7232-328-9

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ICMP	Protokol řídících hlášení – Internet Control Message Protocol
SPI	Externí sériová sběrnice – Serial Peripheral Interface
Ping	Packet Internet Groper
Telnet	Telecommunication Network
traceroute	Analýza počítačové sítě
TCP/IP	Primární přenosový protokol síťové vrstvy– Transmission Control Protocol/Internet Protocol
DF	don‘t fragment – zákaz fragmentace

SEZNAM PŘÍLOH

A ZDROJOVÉ KÓDY PROGRAMU

A.1 Zdrojový kód třídy Bakalarka

Výpis A.1: Zdrojový kód tridy Bakalarka

```
package bakalarka;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

/**
 *
 * @author tomas
 */
public class Bakalarka extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass()
        .getResource("FirstPage.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

A.2 Zdrojový kód třídy FirstPageController

Výpis A.2: Zdrojový kód třídy FirstPageController

```
package bakalarka;

import java.io.IOException;
import java.net.InetAddress;
import java.net.URL;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.stage.Stage;

/**
 *
 * @author tomas
 */
public class FirstPageController implements Initializable {

    @FXML
  private Label nadpis;

  @FXML
  private Button startButton;

  @FXML
  private Button closeButton;

  @FXML
  private void startApplication(ActionEvent event)
  throws IOException
  {
    String myIp = InetAddress.getLocalHost()
```

```
    .getHostAddress();
    FXMLLoader loader = new FXMLLoader();
    loader.setLocation(getClass()
    .getResource("setIPAddress.fxml"));
    try
    {
      loader.load();

    } catch (IOException ex)
    {
      Logger.getLogger(FirstPageController.class.getName())
      .log(Level.SEVERE, null, ex);
    }
    SetIPAddressController nastav = loader.getController();
    nastav.setMyIp(myIp);
    Parent p = loader.getRoot();
    Stage stage = new Stage();
    stage.setScene(new Scene(p));
    stage.show();

    Stage stage2 = (Stage)startButton.getScene().getWindow();
    stage2.close();
  }

  @FXML
    private void closeApplication(ActionEvent event)
    {
        FXMLLoader Loader = new FXMLLoader();
        Loader.setLocation(getClass()
        .getResource("closeEnd.fxml"));
        try
        {
            Loader.load();
        } catch (IOException ex)
        {
            Logger.getLogger(FirstPageController.class
            .getName()).log(Level.SEVERE, null, ex);
        }
        Parent p = Loader.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(p));
```

```
        stage.show();
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

A.3 Zdrojový kód třídy CloseEndController

Výpis A.3: Zdrojový kód třídy CloseEndController

```
package bakalarka;

import com.sun.jndi.dns.DnsContextFactory;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author tomas
 */
public class CloseEndController implements Initializable {

    /**
     * Initializes the controller class.
     */

    @FXML
```

```
    private Button yesButton;

    @FXML
    private Button noButton;

private Process process;

    @FXML
    private void noControll (ActionEvent event)
    {
        Stage stage = (Stage) noButton.getScene().getWindow();
        stage.close();
    }

    public void yesControll (ActionEvent event)
    throws IOException
    {/*
        CloseEndController END = new CloseEndController();
        List<String> commands = new ArrayList<>();
        commands.add("sudo");
        commands.add("shutdown");
        commands.add("-h");
        commands.add("now");
        END.doCommand(commands);*/

        Platform.exit();
        System.exit(0);
    }

    public void doCommand(List<String> command)
    throws IOException
    {

      ProcessBuilder pb = new ProcessBuilder(command);
        this.process = pb.start();
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }
```

```
                                                                    69
}                                                                   70
```

A.4 Zdrojový kód třídy SetIPAddressController

Výpis A.4: Zdrojový kód třídy SetIPAddressController

```
package bakalarka;                                                  1
                                                                    2
import java.io.IOException;                                         3
import java.net.URL;                                                4
import java.util.ResourceBundle;                                    5
import java.util.logging.Level;                                     6
import java.util.logging.Logger;                                    7
import javafx.event.ActionEvent;                                    8
import javafx.fxml.FXML;                                            9
import javafx.fxml.FXMLLoader;                                      10
import javafx.fxml.Initializable;                                   11
import javafx.scene.Parent;                                         12
import javafx.scene.Scene;                                          13
import javafx.scene.control.Button;                                 14
import javafx.scene.control.Label;                                  15
import javafx.scene.control.TextField;                              16
import javafx.stage.Stage;                                          17
                                                                    18
/**                                                                 19
 * FXML Controller class                                            20
 *                                                                  21
 * @author tomas                                                    22
 */                                                                 23
public class SetIPAddressController implements Initializable {     24
                                                                    25
    /**                                                             26
     * Initializes the controller class.                            27
     */                                                             28
                                                                    29
    @FXML                                                           30
  private Label labelMyIp;                                          31
                                                                    32
  @FXML                                                             33
  private TextField ipAddress;                                      34
```

```
                                                                    35
@FXML                                                               36
private Button setButton;                                           37
                                                                    38
@FXML                                                               39
private Button clearButton;                                         40
                                                                    41
@FXML                                                               42
private void number(ActionEvent event) {                            43
  String value = ((Button) event.getSource())                       44
  .getText();                                                       45
  ipAddress.setText(ipAddress.getText() + value);                   46
}                                                                   47
                                                                    48
@FXML                                                               49
private void set(ActionEvent event) {                               50
  String ipAddress_text = ipAddress.getText();                      51
                                                                    52
  if (ipAddress.getText().isEmpty())                                53
  {                                                                 54
    FXMLLoader loader1 = new FXMLLoader();                          55
    loader1.setLocation(getClass()                                  56
    .getResource("errorIP.fxml"));                                  57
    try {                                                           58
      loader1.load();                                               59
                                                                    60
    } catch (IOException ex) {                                      61
      Logger.getLogger(SetIPAddressController.class                 62
      .getName()).log(Level.SEVERE, null, ex);                      63
    }                                                               64
    Parent n = loader1.getRoot();                                   65
    Stage stage1 = new Stage();                                     66
    stage1.setScene(new Scene(n));                                  67
    stage1.showAndWait();                                           68
                                                                    69
  } else {                                                          70
    FXMLLoader Loader2 = new FXMLLoader();                          71
    Loader2.setLocation(getClass()                                  72
    .getResource("mainPage.fxml"));                                 73
    try {                                                           74
      Loader2.load();                                               75
```

```
      } catch (IOException ex) {
        Logger.getLogger(SetIPAddressController.class
        .getName()).log(Level.SEVERE, null, ex);
      }
      MainPageController display = Loader2.getController();
      display.setIp(ipAddress_text);
      Parent p = Loader2.getRoot();
      Stage stage = new Stage();
      stage.setScene(new Scene(p));
      stage.show();
      Stage stage2 = (Stage) setButton.getScene().getWindow();
      stage2.close();

    }
  }

  @FXML
  private void clear(ActionEvent event) {
    ipAddress.clear();
  }

  public void setIp(String ipAddress) {
    this.ipAddress.setText(ipAddress);
  }

  public void setMyIp(String myIp) {
    this.labelMyIp.setText(myIp);
  }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

A.5 Zdrojový kód třídy ErrorIPController

Výpis A.5: Zdrojový kód třídy ErrorIPController

```
package bakalarka;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author tomas
 */
public class ErrorIPController implements Initializable {

    /**
     * Initializes the controller class.
     */

    @FXML
  private Button backButton;

  @FXML
  private void backAction(ActionEvent event)
  {
          Stage stage = (Stage) backButton.getScene()
         .getWindow();
          stage.close();
  }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

A.6 Zdrojový kód třídy MainPageController

Výpis A.6: Zdrojový kód třídy MainPageController

```
package bakalarka;

import java.io.IOException;
import java.net.InetAddress;
import java.net.URL;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author tomas
 */
public class MainPageController implements Initializable {

    /**
     * Initializes the controller class.
     */

    @FXML
    private Label ipAddress;

    @FXML
    private Button setIpAddressButton;

    @FXML
    private Button basicPingButton;

```

```
@FXML                                                                   39
private Button advancePingButton;                                       40
                                                                        41
@FXML                                                                   42
private Button deathPingButton;                                         43
                                                                        44
@FXML                                                                   45
private Button traceRouteButton;                                        46
                                                                        47
@FXML                                                                   48
private Button closeButton;                                             49
                                                                        50
@FXML                                                                   51
private void SetIPAddress(ActionEvent event)                            52
throws IOException                                                      53
{                                                                       54
    String IpAddress_text=ipAddress.getText();                          55
    String myIp = InetAddress.getLocalHost()                            56
    .getHostAddress();                                                  57
    FXMLLoader Loader = new FXMLLoader();                               58
    Loader.setLocation(getClass()                                       59
    .getResource("setIPAddress.fxml"));                                 60
    try                                                                 61
    {                                                                   62
        Loader.load();                                                  63
    } catch (IOException ex)                                            64
    {                                                                   65
      Logger.getLogger(MainPageController.class                         66
      .getName()).log(Level.SEVERE, null, ex);                          67
    }                                                                   68
    SetIPAddressController nastav = Loader                              69
    .getController();                                                   70
    nastav.setIp(IpAddress_text);                                       71
    nastav.setMyIp(myIp);                                               72
    Parent p = Loader.getRoot();                                        73
    Stage stage = new Stage();                                          74
    stage.setScene(new Scene(p));                                       75
    stage.show();                                                       76
    Stage stage2 = (Stage) setIpAddressButton                           77
    .getScene().getWindow();                                            78
    stage2.close();                                                     79
```

```
}                                                                   80
                                                                    81
@FXML                                                               82
private void Basic(ActionEvent event)                               83
{                                                                   84
    String ipAddress_text = ipAddress.getText();                    85
                                                                    86
        FXMLLoader Loader = new FXMLLoader();                       87
        Loader.setLocation(getClass()                               88
        .getResource("basicPing.fxml"));                            89
        try                                                         90
        {                                                           91
            Loader.load();                                          92
        } catch (IOException ex)                                    93
        {                                                           94
            Logger.getLogger(MainPageController.class               95
            .getName()).log(Level.SEVERE, null, ex);                96
        }                                                           97
        BasicPingController nastav = Loader                         98
        .getController();                                           99
        nastav.setIP(ipAddress_text);                               100
        Parent p = Loader.getRoot();                                101
        Stage stage = new Stage();                                  102
        stage.setScene(new Scene(p));                               103
        stage.showAndWait();                                        104
                                                                    105
}                                                                   106
                                                                    107
@FXML                                                               108
private void AdvancePing(ActionEvent event)                         109
{                                                                   110
    String ipAddress_text = ipAddress.getText();                    111
                                                                    112
    FXMLLoader loader = new FXMLLoader();                           113
    loader.setLocation(getClass()                                   114
    .getResource("advancedPingSetup.fxml"));                        115
    try {                                                           116
        loader.load();                                              117
    } catch (IOException ex)                                        118
    {                                                               119
        Logger.getLogger(MainPageController.class                   120
```

```
            .getName()).log(Level.SEVERE,null,ex);
        }
        AdvancedPingSetupController setup = loader
        .getController();
        setup.setIP(ipAddress_text);
        Parent p = loader.getRoot();
        Stage stage = new Stage();
        stage.setScene(new Scene(p));
        stage.showAndWait();

    }

    @FXML
    private void DeathPing (ActionEvent event)
    {
        String ipAddress_text=ipAddress.getText();

            FXMLLoader Loader = new FXMLLoader();
            Loader.setLocation(getClass()
            .getResource("deathPing.fxml"));
            try
            {
                Loader.load();
            } catch (IOException ex)
            {
                Logger.getLogger(MainPageController.class
                .getName()).log(Level.SEVERE, null, ex);
            }
            DeathPingController IP = Loader.getController();
            IP.setIP(ipAddress_text);
            Parent p = Loader.getRoot();
            Stage stage = new Stage();
            stage.setScene(new Scene(p));
            stage.showAndWait();

    }

    @FXML
    private void Tracert(ActionEvent event)
    {
        String ipAddress_text=ipAddress.getText();
```

```
                                                                    162
        FXMLLoader Loader = new FXMLLoader();                       163
        Loader.setLocation(getClass()                               164
        .getResource("traceroute.fxml"));                           165
        try                                                         166
        {                                                           167
            Loader.load();                                          168
        } catch (IOException ex)                                    169
        {                                                           170
            Logger.getLogger(MainPageController.class               171
            .getName()).log(Level.SEVERE, null, ex);                172
        }                                                           173
        TracerouteController IP = Loader.getController();          174
        IP.setIP(ipAddress_text);                                   175
        Parent p = Loader.getRoot();                                176
        Stage stage = new Stage();                                  177
        stage.setScene(new Scene(p));                               178
        stage.showAndWait();                                        179
                                                                    180
                                                                    181
}                                                                   182
                                                                    183
@FXML                                                               184
private void Close(ActionEvent event)                               185
{                                                                   186
    FXMLLoader Loader = new FXMLLoader();                           187
    Loader.setLocation(getClass()                                   188
    .getResource("closeEnd.fxml"));                                 189
    try                                                             190
    {                                                               191
        Loader.load();                                              192
    } catch (IOException ex)                                        193
    {                                                               194
        Logger.getLogger(MainPageController.class                   195
        .getName()).log(Level.SEVERE, null, ex);                    196
    }                                                               197
    Parent p = Loader.getRoot();                                    198
    Stage stage = new Stage();                                      199
    stage.setScene(new Scene(p));                                   200
    stage.show();                                                   201
}                                                                   202
```

```
                                                                    203
    public void setIp(String ipAddress)                            204
    {                                                               205
        this.ipAddress.setText(ipAddress);                          206
    }                                                               207
                                                                    208
    @Override                                                       209
    public void initialize(URL url, ResourceBundle rb) {            210
        // TODO                                                     211
    }                                                               212
                                                                    213
}                                                                   214
```

A.7 Zdrojový kód třídy BasicPingController

Výpis A.7: Zdrojový kód třídy BasicPingController

```
package bakalarka;                                                  1
                                                                    2
import java.io.BufferedReader;                                      3
import java.io.IOException;                                         4
import java.io.InputStreamReader;                                   5
import java.io.Reader;                                              6
import java.net.URL;                                                7
import java.util.ArrayList;                                         8
import java.util.List;                                              9
import java.util.ResourceBundle;                                    10
import javafx.event.ActionEvent;                                    11
import javafx.fxml.FXML;                                            12
import javafx.fxml.Initializable;                                   13
import javafx.scene.control.Button;                                 14
import javafx.scene.control.TextArea;                               15
import javafx.stage.Stage;                                          16
                                                                    17
/**                                                                 18
 * FXML Controller class                                            19
 *                                                                  20
 * @author tomas                                                    21
 */                                                                 22
public class BasicPingController implements Initializable {        23
                                                                    24
```

```
/**
 * Initializes the controller class.
 */

private Process process;
private String ip;

@FXML
private TextArea console;

@FXML
private Button startButton;

@FXML
private Button backButton;

@FXML
private Button stopButton;

@FXML
private void Back(ActionEvent event) {
    Stage stage = (Stage) backButton.getScene()
    .getWindow();
    stage.close();
}

public void startControll(ActionEvent event)
throws IOException {
    List<String> commands = new ArrayList<>();
    commands.add("ping");
    commands.add(ip);
    commands.add("-c");
    commands.add("5");
    doCommand(commands);
}

public void stopControll(ActionEvent event)
throws IOException {
    if (process!=null) {
        process.destroy();
        process = null;
```

```
        }
    }

public void setIP(String ip)
{
        this.ip = ip;
    }

public void doCommand(List<String> command)
throws IOException {

        ProcessBuilder pb = new ProcessBuilder(command);
        this.process = pb.start();

        Reader reader = new InputStreamReader(process
        .getInputStream());
        ProcessThread ort = new ProcessThread(reader
        , console);
        ort.start();

    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    static class ProcessThread extends Thread {

        private Reader reader;
        private TextArea ta;

        public ProcessThread(Reader reader, TextArea ta) {
            this.reader = reader;
            this.ta = ta;
            this.setDaemon(true);
        }

        @Override
        public void run() {
```

```
            try {                                                   107
                BufferedReader br =                                 108
                new BufferedReader(reader);                         109
                String cr = null;                                   110
                while ((cr = br.readLine()) != null) {              111
                    if (cr.length()>0) {                            112
                        ta.setText(ta.getText() + "\n" + cr);       113
                        ta.selectPositionCaret(ta                   114
                        .getLength());                              115
                        ta.deselect();                              116
                    }                                               117
                }                                                   118
            } catch (IOException ex) {                              119
                ex.printStackTrace();                               120
            }                                                       121
                                                                    122
        }                                                           123
    }                                                               124
}                                                                   125
```

A.8 Zdrojový kód třídy AdvancedPingSetupController

Výpis A.8: Zdrojový kód třídy AdvancedPingSetupController

```
package bakalarka;                                                  1
                                                                    2
import java.io.IOException;                                         3
import java.net.URL;                                                4
import java.util.ResourceBundle;                                    5
import java.util.logging.Level;                                     6
import java.util.logging.Logger;                                    7
import javafx.event.ActionEvent;                                    8
import javafx.event.Event;                                          9
import javafx.fxml.FXML;                                            10
import javafx.fxml.FXMLLoader;                                      11
import javafx.fxml.Initializable;                                   12
import javafx.scene.Parent;                                         13
import javafx.scene.Scene;                                          14
import javafx.scene.control.Button;                                 15
```

```
import javafx.scene.control.CheckBox;
import javafx.scene.control.TextField;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author tomas
 */
public class AdvancedPingSetupController
implements Initializable {

    /**
     * Initializes the controller class.
     */

    private String ip;

    @FXML
    private TextField selectedTextField;

    @FXML
    private Button backButton;

    @FXML
    private Button setButton;

    @FXML
    private CheckBox unlimitedBox;

    @FXML
    private CheckBox countBox;

    @FXML
    private TextField countField;

    @FXML
    private CheckBox intervalBox;

    @FXML
    private TextField intervalField;
```

```
                                                                    57
@FXML                                                               58
private CheckBox packetSizeBox;                                     59
                                                                    60
@FXML                                                               61
private TextField packetSizeField;                                  62
                                                                    63
@FXML                                                               64
private CheckBox ttlBox;                                            65
                                                                    66
@FXML                                                               67
private TextField ttlField;                                         68
                                                                    69
@FXML                                                               70
private CheckBox deadlineBox;                                       71
                                                                    72
@FXML                                                               73
private TextField deadlineField;                                    74
                                                                    75
@FXML                                                               76
private CheckBox timeoutBox;                                        77
                                                                    78
@FXML                                                               79
private TextField timeoutField;                                     80
                                                                    81
@FXML                                                               82
private void backAction(ActionEvent event)                          83
{                                                                   84
    Stage stage = (Stage) backButton.getScene()                     85
    .getWindow();                                                   86
    stage.close();                                                  87
}                                                                   88
                                                                    89
@FXML                                                               90
public void number(ActionEvent event)                               91
{                                                                   92
    selectedTextField.setText(selectedTextField                     93
    .getText()+((Button)event.getSource()).getText());              94
}                                                                   95
                                                                    96
@FXML                                                               97
```

```
    private void setFocus(Event event)                                  98
    {                                                                   99
        selectedTextField = ((TextField) event.getSource());           100
    }                                                                   101
                                                                        102
    @FXML                                                               103
    private void unlimiteCount(ActionEvent event)                       104
    {                                                                   105
        CheckBox cb = (CheckBox) event.getSource();                     106
        if ("unlimitedBox".equals(cb.getId())) {                        107
            boolean selected = ((CheckBox) event                        108
            .getSource()).isSelected();                                 109
            countBox.setDisable(!selected);                             110
        }                                                               111
    }                                                                   112
                                                                        113
    @FXML                                                               114
    private void counter(ActionEvent event)                             115
    {                                                                   116
        CheckBox cb = (CheckBox) event.getSource();                     117
        if ("countBox".equals(cb.getId())) {                            118
            boolean selected = ((CheckBox) event                        119
            .getSource()).isSelected();                                 120
            countField.setDisable(!selected);                           121
            unlimitedBox.setDisable(selected);                          122
            if (!selected) {                                            123
                countField.clear();                                     124
            }                                                           125
        }                                                               126
                                                                        127
    }                                                                   128
                                                                        129
    @FXML                                                               130
    private void intervals(ActionEvent event)                           131
    {                                                                   132
        CheckBox cb = (CheckBox) event.getSource();                     133
        if ("intervalBox".equals(cb.getId())) {                         134
            boolean selected = ((CheckBox) event                        135
            .getSource()).isSelected();                                 136
            intervalField.setDisable(!selected);                        137
            if (!selected) {                                            138
```

```
            intervalField.clear();
        }
    }
}

@FXML
private void packetSize(ActionEvent event)
{
    CheckBox cb = (CheckBox) event.getSource();
    if ("packetSizeBox".equals(cb.getId())) {
        boolean selected = ((CheckBox) event
        .getSource()).isSelected();
        packetSizeField.setDisable(!selected);
        if (!selected) {
            packetSizeField.clear();
        }
    }
}

@FXML
private void timeToLive(ActionEvent event)
{
    CheckBox cb = (CheckBox) event.getSource();
    if ("ttlBox".equals(cb.getId())) {
        boolean selected = ((CheckBox) event
        .getSource()).isSelected();
        ttlField.setDisable(!selected);
        if (!selected) {
            ttlField.clear();
        }
    }
}

@FXML
private void deadline(ActionEvent event)
```

```
{                                                                    180
    CheckBox cb = (CheckBox) event.getSource();                      181
    if ("deadlineBox".equals(cb.getId())) {                          182
        boolean selected = ((CheckBox) event                         183
        .getSource()).isSelected();                                  184
        deadlineField.setDisable(!selected);                         185
        if (!selected) {                                             186
            deadlineField.clear();                                   187
                                                                     188
        }                                                            189
                                                                     190
    }                                                                191
}                                                                    192
                                                                     193
@FXML                                                                194
private void timeout(ActionEvent event)                              195
{                                                                    196
    CheckBox cb = (CheckBox) event.getSource();                      197
    if ("timeoutBox".equals(cb.getId())) {                           198
        boolean selected = ((CheckBox) event                         199
        .getSource()).isSelected();                                  200
        timeoutField.setDisable(!selected);                          201
        if (!selected) {                                             202
            timeoutField.clear();                                    203
                                                                     204
        }                                                            205
                                                                     206
    }                                                                207
}                                                                    208
                                                                     209
@FXML                                                                210
private void setAction(ActionEvent event)                            211
{                                                                    212
  boolean countBoxResolve = countBox                                 213
  .isSelected();                                                     214
  boolean intervalBoxResolve = intervalBox                           215
  .isSelected();                                                     216
  boolean packetsizeBoxResolve = packetSizeBox                       217
  .isSelected();                                                     218
  boolean ttlBoxResolve = ttlBox.isSelected();                       219
  boolean deadlineBoxResolve = deadlineBox                           220
```

```
.isSelected();                                                    221
boolean timeoutBoxkResolve = timeoutBox                           222
.isSelected();                                                    223
String countTextResolve = countField.                             224
getText();                                                        225
String intervalTextresolve = intervalField                        226
.getText();                                                       227
String packetSizeTextResolve = packetSizeField                    228
.getText();                                                       229
String ttlTextResolve = ttlField.getText();                       230
String deadlineTextResolve = deadlineField                        231
.getText();                                                       232
String timeoutTextResolve = timeoutField                          233
.getText();                                                       234
  FXMLLoader Loader = new FXMLLoader();                           235
  Loader.setLocation(getClass()                                   236
  .getResource("advancedPingResult.fxml"));                       237
  try                                                             238
  {                                                               239
      Loader.load();                                              240
  } catch (IOException ex)                                        241
  {                                                               242
      Logger.getLogger(AdvancedPingSetupController.class          243
      .getName()).log(Level.SEVERE, null, ex);                    244
  }                                                               245
  AdvancedPingResultController setup = Loader                     246
  .getController();                                               247
  setup.setIP(ip);                                                248
  setup.setCounter(countBoxResolve);                              249
  setup.setCounterText(countTextResolve);                         250
  setup.setIntervals(intervalBoxResolve);                         251
  setup.setIntervalsText(intervalTextresolve);                    252
  setup.setPacketSize(packetsizeBoxResolve);                      253
  setup.setPacketSizetext(packetSizeTextResolve);                 254
  setup.setTTL(ttlBoxResolve);                                    255
  setup.seTtltext(ttlTextResolve);                                256
  setup.setDeadline(deadlineBoxResolve);                          257
  setup.setDeadlineText(deadlineTextResolve);                     258
  setup.setTimeout(ttlBoxResolve);                                259
  setup.setTimeoutText(timeoutTextResolve);                       260
  Parent p = Loader.getRoot();                                    261
```

```
        Stage stage = new Stage();
        stage.setScene(new Scene(p));
        stage.showAndWait();
    }

    public void setIP(String ip)
    {
        this.ip = ip;
    }
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

}
```

A.9 Zdrojový kód třídy AdvancedPingResultController

Výpis A.9: Zdrojový kód třídy AdvancedPingResultController

```
package bakalarka;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

/**
 * FXML Controller class
```

```
 *
 * @author tomas
 */
public class AdvancedPingResultController
implements Initializable {

    /**
     * Initializes the controller class.
     */
    private Process process;

    private String ip;

    private boolean counter;

    private String counterText;

    private boolean intevals;

    private String intervalsText;

    private boolean packetSize;

    private String packetSizeText;

    private boolean ttl;

    private String ttlText;

    private boolean deadline;

    private String deadlineText;

    private boolean timeout;

    private String timeoutText;

    @FXML
    private TextArea console;

    @FXML
```

```
    private Button startButton;
    @FXML
    private Button backButton;
    @FXML
    private Button stopButton;
    @FXML
    private void backAction(ActionEvent event) {
        Stage stage = (Stage) backButton.getScene()
        .getWindow();
        stage.close();
    }
    public void startControll(ActionEvent event)
    throws IOException {
        List<String> commands = new ArrayList<>();
        commands.add("ping");
        commands.add(ip);
        if (counter == true)
        {
            commands.add("-c");
            commands.add(counterText);
}
        if (intevals == true)
        {
            commands.add("-i");
            commands.add(intervalsText);
}
        if (packetSize == true)
        {
            commands.add("-s");
            commands.add(packetSizeText);
}
        if (ttl == true)
        {
            commands.add("-t");
            commands.add(ttlText);
}
        if (deadline == true)
        {
```

```
            commands.add("-w");
            commands.add(deadlineText);
}
        if (timeout == true)
        {
            commands.add("-W");
            commands.add(timeoutText);
}
        doCommand(commands);
    }

    public void stopControll(ActionEvent event)
    throws IOException {
        if (process!=null) {
            process.destroy();
            process = null;
        }
    }

    public void setCounter(boolean counter)
    {
        this.counter = counter;
    }

    public void setCounterText(String counterText)
    {
        this.counterText = counterText;
    }

    public void setIntervals(boolean intervals)
    {
        this.intevals = intervals;
    }

    public void setIntervalsText(String instervalsText)
    {
        this.intervalsText = instervalsText;
    }

    public void setPacketSize(boolean packetSize)
    {
```

```
        this.packetSize = packetSize;                                    143
    }                                                                    144
                                                                         145
    public void setPacketSizetext(String packetSizetext)                 146
    {                                                                    147
        this.packetSizeText = packetSizetext;                            148
    }                                                                    149
                                                                         150
    public void setTTL(boolean ttl)                                      151
    {                                                                    152
        this.ttl = ttl;                                                  153
    }                                                                    154
                                                                         155
    public void seTtltext(String ttltext)                                156
    {                                                                    157
        this.ttlText = ttltext;                                          158
    }                                                                    159
    public void setDeadline(boolean deadline)                            160
    {                                                                    161
        this.deadline = deadline;                                        162
    }                                                                    163
                                                                         164
    public void setDeadlineText(String deadlineText)                     165
    {                                                                    166
        this.deadlineText = deadlineText;                                167
    }                                                                    168
    public void setTimeout(boolean timeout)                              169
    {                                                                    170
        this.timeout = timeout;                                          171
    }                                                                    172
                                                                         173
    public void setTimeoutText(String timeoutText)                       174
    {                                                                    175
        this.timeoutText = timeoutText;                                  176
    }                                                                    177
                                                                         178
    public void setIP(String ip)                                         179
    {                                                                    180
        this.ip = ip;                                                    181
    }                                                                    182
                                                                         183
```

```
public void doCommand(List<String> command)                    184
throws IOException {                                           185
                                                               186
    ProcessBuilder pb = new ProcessBuilder(command);           187
    this.process = pb.start();                                 188
                                                               189
    Reader reader = new InputStreamReader                      190
    (process.getInputStream());                                191
    BasicPingController.ProcessThread ort =                    192
    new BasicPingController.ProcessThread(reader, console);    193
    ort.start();                                               194
                                                               195
}                                                              196
                                                               197
@Override                                                      198
public void initialize(URL url, ResourceBundle rb) {          199
    // TODO                                                    200
}                                                              201
                                                               202
static class ProcessThread extends Thread {                    203
                                                               204
    private Reader reader;                                     205
    private TextArea ta;                                       206
                                                               207
    public ProcessThread(Reader reader, TextArea ta) {         208
        this.reader = reader;                                  209
        this.ta = ta;                                          210
        this.setDaemon(true);                                  211
    }                                                          212
                                                               213
    @Override                                                  214
    public void run() {                                        215
                                                               216
        try {                                                  217
            BufferedReader br =                                218
            new BufferedReader(reader);                        219
            String cr = null;                                  220
            while ((cr = br.readLine()) != null) {             221
                if (cr.length()>0) {                           222
                    ta.setText(ta.getText() + "\n" + cr);      223
                    ta.selectPositionCaret                     224
```

```
                        (ta.getLength());
                        ta.deselect();
                    }
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }

        }
    }

}
```

A.10 Zdrojový kód třídy DeathPingController

Výpis A.10: Zdrojový kód třídy DeathPingController

```
package bakalarka;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.stage.Stage;

/**
 * FXML Controller class
 *
 * @author tomas
 */
public class DeathPingController implements Initializable {

```

```
/**
 * Initializes the controller class.
 */

private Process process;
private String ip;

@FXML
private TextArea console;

@FXML
private Button startButton;

@FXML
private Button backButton;

@FXML
private Button stopButton;

@FXML
private void Back(ActionEvent event) {
    Stage stage = (Stage) backButton.getScene()
    .getWindow();
    stage.close();
}

public void startControll(ActionEvent event)
throws IOException {
    List<String> commands = new ArrayList<>();
    commands.add("ping");
    commands.add(ip);
    doCommand(commands);
}

public void stopControll(ActionEvent event)
throws IOException {
    if (process!=null) {
        process.destroy();
        process = null;
    }
}
```

```
                                                                    66
public void setIP(String ip)                                        67
{                                                                   68
      this.ip = ip;                                                 69
  }                                                                 70
                                                                    71
public void doCommand(List<String> command)                        72
throws IOException {                                               73
                                                                    74
      ProcessBuilder pb = new ProcessBuilder(command);              75
      this.process = pb.start();                                    76
                                                                    77
      Reader reader = new InputStreamReader                         78
      (process.getInputStream());                                   79
      ProcessThread ort = new ProcessThread                         80
      (reader, console);                                            81
      ort.start();                                                  82
                                                                    83
  }                                                                 84
                                                                    85
  @Override                                                         86
  public void initialize(URL url, ResourceBundle rb) {              87
      // TODO                                                       88
  }                                                                 89
                                                                    90
  static class ProcessThread extends Thread {                       91
                                                                    92
      private Reader reader;                                        93
      private TextArea ta;                                          94
                                                                    95
      public ProcessThread(Reader reader, TextArea ta) {            96
          this.reader = reader;                                     97
          this.ta = ta;                                             98
          this.setDaemon(true);                                     99
      }                                                             100
                                                                    101
      @Override                                                     102
      public void run() {                                           103
                                                                    104
          try {                                                     105
              BufferedReader br =                                   106
```

```
                new BufferedReader(reader);                          107
                String cr = null;                                    108
                while ((cr = br.readLine()) != null) {               109
                    if (cr.length()>0) {                             110
                        ta.setText(ta.getText() + "\n" + cr);        111
                        ta.selectPositionCaret                       112
                        (ta.getLength());                            113
                        ta.deselect();                               114
                    }                                                115
                }                                                    116
            } catch (IOException ex) {                               117
                ex.printStackTrace();                                118
            }                                                        119
                                                                     120
        }                                                            121
    }                                                                122
}                                                                    123
```

A.11 Zdrojový kód třídy TarerouteController

Výpis A.11: Zdrojový kód třídy TracerouteController

```
package bakalarka;                                                   1
                                                                     2
import java.io.BufferedReader;                                       3
import java.io.IOException;                                          4
import java.io.InputStreamReader;                                    5
import java.io.Reader;                                               6
import java.net.URL;                                                 7
import java.util.ArrayList;                                          8
import java.util.List;                                               9
import java.util.ResourceBundle;                                     10
import javafx.event.ActionEvent;                                     11
import javafx.fxml.FXML;                                             12
import javafx.fxml.Initializable;                                    13
import javafx.scene.control.Button;                                  14
import javafx.scene.control.TextArea;                                15
import javafx.stage.Stage;                                           16
                                                                     17
/**                                                                  18
 * FXML Controller class                                            19
```

```
 *
 * @author tomas
 */
public class TracerouteController implements Initializable {

    /**
     * Initializes the controller class.
     */

    private Process process;
    private String ip;

    @FXML
    private TextArea console;

    @FXML
    private Button startButton;

    @FXML
    private Button backButton;

    @FXML
    private Button stopButton;

    @FXML
    private void Back(ActionEvent event) {
        Stage stage = (Stage) backButton.getScene()
        .getWindow();
        stage.close();
    }

    public void startControll(ActionEvent event)
    throws IOException {
        List<String> commands = new ArrayList<>();
        commands.add("traceroute");
        commands.add(ip);
        doCommand(commands);
    }

    public void stopControll(ActionEvent event)
    throws IOException {
```

```
        if (process!=null) {
            process.destroy();
            process = null;
        }
    }

public void setIP(String ip)
{
        this.ip = ip;
    }

public void doCommand(List<String> command)
throws IOException {

        ProcessBuilder pb = new ProcessBuilder(command);
        this.process = pb.start();

        Reader reader = new InputStreamReader
        (process.getInputStream());
        ProcessThread ort = new ProcessThread
        (reader, console);
        ort.start();

    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    static class ProcessThread extends Thread {

        private Reader reader;
        private TextArea ta;

        public ProcessThread(Reader reader, TextArea ta) {
            this.reader = reader;
            this.ta = ta;
            this.setDaemon(true);
        }
```

```
        @Override                                                       102
        public void run() {                                             103
                                                                        104
            try {                                                       105
                BufferedReader br =                                     106
                new BufferedReader(reader);                             107
                String cr = null;                                       108
                while ((cr = br.readLine()) != null) {                  109
                    if (cr.length()>0) {                                110
                        ta.setText(ta.getText() + "\n" + cr);           111
                        ta.selectPositionCaret                          112
                        (ta.getLength());                               113
                        ta.deselect();                                  114
                    }                                                   115
                }                                                       116
            } catch (IOException ex) {                                  117
                ex.printStackTrace();                                   118
            }                                                           119
                                                                        120
        }                                                               121
    }                                                                   122
}                                                                       123
```

B OBSAH PŘILOŽENÉHO CD

Na CD naleznete dokumentaci, vytvořený program a projekt k programu vytvořený v programu NetBeans, který je zazipovaný.

- `/` ..... kořenový adresář přiloženého CD
  - `Bakalarka.jar` ..... pragram projektu
  - `Bakalarka.zip` ..... projekt v programu NetBeans
  - `Dokumentace` ..... dokumentace k projektu
    - `sablona-prace.pdf`