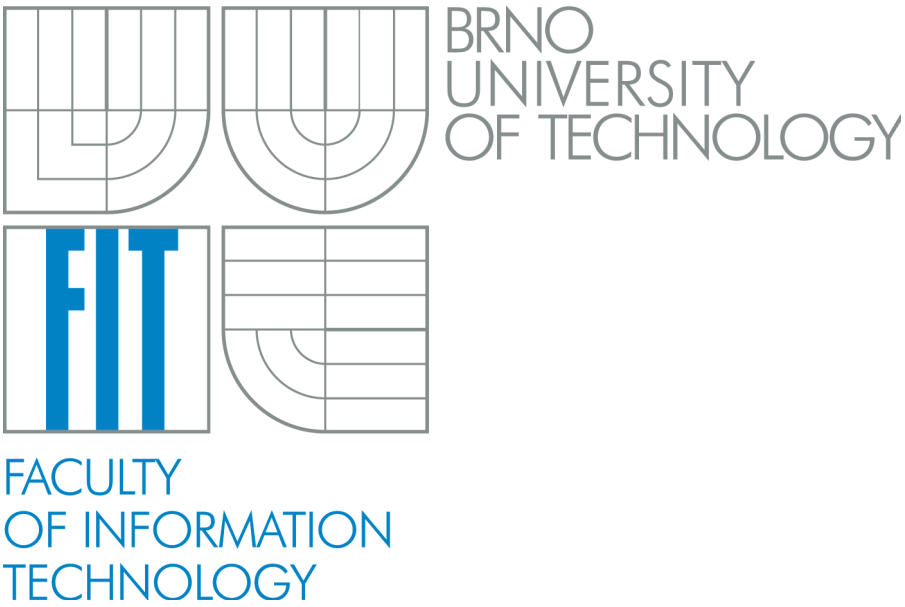

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Síťová služba Trace

Autor: Tomáš Válek

Login: xvalek02

Datum: 3. listopadu 2012

Obsah

1. Úvod do problematiky

Cílem bylo navrhnout program *trace,* který obdobným způsobem jako program *traceroute* umožňuje průchod paketu sítí, čili vypisuje směrovače od zdroje k cíli.

Trace pracuje s protokolem IPv4/IPv6 a využívá odpovídající ICMP(Internet Control Message Protocol). ICMPv4/ICMPv6 je protokol internetové vrstvy z rodiny protokolů TCP/IP a slouží k přenosu řídících informací a k signalizaci chybových stavů jako je zahození IP paketu, nedosažitelnost cílového uzlu atp. ICMPv4/ICMPv6 zpráva je přenášena v IP paketu. Trace využívá položky jako TTL(Time To Live) v IPv4 datagramu a Hop Limit v IPv6 datagramu. Na každém směrovači od zdroje k cíli se pokouší vyvolat ICMPv4/ICMPv6 zprávu Time Exceeded(čas vypršel) aby zjistil, že směrovač není koncový. Pokud směrovač koncový je, program trace zaznamená ICMPv4/ICMPv6 zprávu ECHO_REPLY a zná kompletní cestu od zdroje k cíli.

Na detailnější funkce programu trace se podíváme v návrhu aplikace na další straně.

2. Návrh aplikace

Aplikace je logicky rozvržena do několika funkcí z nich každá provádí definovanou operaci s daty, která vede ke správné činnosti programu trace. Uvedu přehled funkcí se stručným popisem. Zajímavější pasáže budou rozvedeny v kapitole Popis implementace.

- getParams() - Zpracuje parametry příkazové řádky, zkontroluje správně zadané parametry a ošetří případné chyby.
- error() - Pokud v průběhu programu nastane chyba, tato funkce zajistí dealokaci zdrojů a korektní ukončení programu.
- traceroute() - Hlavní funkce aplikace jejímž úkolem je zjistit průchod paketu sítí od zdroje k cíli.
- setTTL() - Nastaví na soketu TTL/Hop Limit.
- getIP() - Získá IPv4/IPv6 adresu.
- getHostName() - Získá jméno hosta z IP adresy.
- checksum() - Vypočítá kontrolní součet, který se doplní do ICMPv4/ICMPv6 hlavičky. Tato funkce je převzatá z RFC 1071 [1]
- lostMyPacket() - Analyzuje ztracený paket a zjišťuje podle „id“ a „sequence“, zda paket patří nám.
- terminated() - Obsluha signálů.
- response() - Počítá odezvu.

2.1 Jednoduchý algoritmus

Jedná se o stručný algoritmus na patřičné úrovni abstrakce.

Poznámka: N je časová jednotka v sekundách implicitně jedna, ale může být změněna parametrem příkazové řádky.

1) vyplníme ICMPv4/ICMPv6 hlavičku
2) ICMPv4/ICMPv6 hlavičku odešleme(IPv4/IPv6 hlavičku vyplní jádro OS)
3) získáme časový bod A
4) čekáme N dlouho na příchozí paket:
   4a) doba vypršela – vypíšeme „* * *“ a jdeme na bod 10)
   4b)paket dorazil – pokračuj na bod 5)
5) přijmeme paket

6) získáme časový bod B
7) zjistíme zda paket který dorazil je k naší žádosti
   7a)ANO: pokračuj na bod 8)
   7b)NE: – paket zahodíme, pokračuj na bod 10)

8) zjistíme informace z hlavičky ICMPv4/ICMPv6
   8a)ECHO_REPLY k naší žádosti – konec, cesta nalezena
   8b)chybová zpráva – pokračuj na bod 9)
9) vypíšeme informace o odesílateli ICMPv4/ICMPv6 a jeho odezvu vypočítanou z bodu 3) a 6)
10) pokud hodnota TTL/Hop Limit nepřekročila maximum, zvýšíme TTL/Hop Limit o jedničku a jdeme na bod 1)

11) konec, cesta nenalezena

3. Popis implementace

3.1 ICMP

Jelikož aplikace podporuje IPv4/IPv6 je potřeba použít odpovídající protokoly internetové vrstvy ICMPv4 a ICMPv6. Implementace struktur *icmphdr* a *icmp6_hdr* je na první pohled rozdílná.

3.2 ICMPv4 [4]

Program dokáže rozpoznat tyto ICMPv4 zprávy:

Typ	Kód	Význam	Označení
0	0	Echo Reply	
11	0	Time to Live exceeded in Transit	
3	1	Host Unreachable	H!
3	2	Protocol Unreachable	P!
3	6	Destination Network Unknown	N!
3	7	Destination Host Unknown	H!
3	8	Source Host Isolated	H!
3	9	Commun. with Dest. Network is Admin. Prohibited	N!
3	10	Commun. with Dest. Host is Admin. Prohibited	H!
3	11	Destination Network Unreachable for Type of Service	N!
3	12	Destination Host Unreachable for Type of Service	H!
3	13	Communication administratively prohibited	X!
3	14	Host Precedence Violation	H!

3.3 ICMPv6 [5]

V RFC ICMPv6 již nenajdeme tolik zpráv jako u ICMPv4. Došlo k vypuštění nepoužívaných ICMP zpráv, tudíž jich je daleko méně na rozdíl od ICMPv4. Program dokáže rozpoznat tyto ICMPv6 zprávy:

Typ	Kód	Význam	Označení
129	0	Echo Reply	
3	0	Hop limit exceeded in transit	
1	1	Commun. with destination admin. Prohibited	X!

3.4 Soket

Vytváříme soket typu RAW s protokolem ICMPv4/ICMPv6. Na soketu nastavujeme vlastnost IP protokolu: TTL/Hop Limit.

3.5 Rozlišení ICMP zpráv

Pro rozlišení příchozích ICMP zpráv vkládáme do hlavičky jednoznačný identifikátor, který představuje číslo procesu a sekvenci odeslané ICMP zprávy.

3.6 Výpočet odezvy

Patrně by pro výpočet odezvy šla použít položka časové razítko v IPv4 hlavičce. Jde o volitelnou část IPv4 hlavičky. Tato volba byla ale zamítnuta, protože:

- směrovač mi položku časové razítko nemusí vůbec do hlavičky IPv4 zapsat nebo mi může podvrhnout falešnou hodnotu, což by vedlo na nesprávný výpočet odezvy
- v IPv6 hlavičce žádná taková možnost neexistuje

Pro zjištění aktuálního času byla použita funkce gettimeofday() z <time.h>. Princip je jednoduchý:

- odešleme žádost
- získáme čas A
- přijmeme žádost
- získáme čas B

Tyto data pošleme do funkce response(), která nám spočítá odezvu.

3.7 Čtení ICMPv4 zpráv

U chybových stavů ICMPv4 abychom mohli přečíst přijatou zprávu, musíme přeskočit IPv4 hlavičku. Délku IPv4 hlavičky zjistíme z položky IHL, kterou vynásobíme čtyřmi, protože atribut IHL je v délce půl-bajtu.

Pokud chceme číst odeslanou ICMPv4 zprávu v těle přijaté ICMPv4 zprávy, musíme přeskočit IPv4 hlavičku + ICMPv4 hlavičku.

U ECHO_REPLY dorazí ICMPv4 zpráva bez IP hlavičky.

3.8 Čtení ICMPv6 zpráv

Stejné jako u ICMPv4 zpráv s rozdílem délky IPv6 hlavičky, která má konstantních 20bytů.

3.9 Problémy při implementaci

- ICMPv4 obsahuje atributy „identifier“ a „sequence“, které jsem v ICMPv6 hlavičce nemohl najít. Nakonec se povedlo najít, kde se tyto hodnoty uchovávají.
- Čísla z ICMP hlavičky byly jiné, než jsem předpokládal. Problém vyřešen htons, ntohs.
- Výpočet odezvy jsem řešil funkcí clock() z <time.h>. Tato funkce se ukázala jako velmi nepřesná. Navíc měří počet „tiků“, jež strávil program na procesoru. V dnešních systémech s více procesory ji tedy nelze použít. Východiskem byla tedy funkce gettimeofday() z <time.h>, která měří s přesností na mikro-vteřiny.
- Další problém nastal při přijímání ICMP zpráv. Při ECHO_REPLY dorazí ICMP zpráva bez těla, ale při chybě dorazí ICMP s tělem. Zprvu jsem to nerozlišoval a přistupoval do cizí paměti.
- Volba mezi AF_INET a PF_INET byla nejasná. Pokud jsem si dal vypsat tyto konstanty, zjistil jsem, že jsou stejné.

4. Návod k použití

Protože aplikace používá RAW sockety, je potřeba být přihlášený jako root.

trace [-f TTL] [-m max_TLL] host

-f: velikost TTL pro první paket - implicitně 1

-m: maximální velikost TTL/Hop Limit - implicitně 30

host: povinný parametr, IPv4/IPv6/DNS cíl

5. Příklady spuštění

IPv6:

```
sudo ./trace www.fit.vutbr.cz
 1    aaabcw-1.tunnel.tserv27.prg1.ipv6.he.net (2001:470:6e:28a::1) 15.405
ms
 2    gige-g2-20.core1.prg1.he.net (2001:470:0:221::1) 35.560 ms
 3    nix2-20ge.ipv6.cesnet.cz (2001:7f8:14::1:1) 14.802 ms
 4    2001:718:0:c003::2 (2001:718:0:c003::2) 22.854 ms
 5    hp-kou.net.vutbr.cz (2001:67c:1220:f534::aff:601) 18.125 ms
 6    pe-ant.net.vutbr.cz (2001:67c:1220:f712::aff:4701) 20.909 ms
 7    2001:67c:1220:f526::2 (2001:67c:1220:f526::2) 19.769 ms
 8    www.fit.vutbr.cz (2001:67c:1220:809::93e5:917) 18.735 ms
```

IPv4:

```
sudo ./trace czc.cz
 1    Sucre.lan (192.168.1.1) 0.519 ms
 2    * * *
 3    static-84-242-127-1.net.upcbroadband.cz (84.242.127.1) 9.786 ms
 4    84.116.221.37 (84.116.221.37) 36.834 ms
 5    nix1.bluetone.cz (91.210.16.206) 14.719 ms
 6    xe10-1-0-100-mx-site2.bluetone.cz (84.244.126.169) 30.005 ms
 7    xe11-0-0-100-mx-stra2.bluetone.cz (84.244.126.7) 17.767 ms
 8    ge0-1-tiskar2.bluetone.cz (82.99.164.234) 20.654 ms
 9    www.czechcomputer.cz (82.99.173.173) 21.830 ms
```

6. Literatura

[1] Computing the Internet Checksum, September 1988, strana 6 bod 4.1 „C“ RFC 1071 .

[2] INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September 1981 RFC 791 .

[3] Internet Protocol, Version 6 (IPv6) Specification, December 1998 RFC 2460 .

[4] INTERNET CONTROL MESSAGE PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September 1981 RFC 792 .

[5] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification, March 2006 RFC 4443 .

[6] Raw socket Wikipedia .

[7] getaddrinfo(3) - Linux man page linux.die .

7. Závěr

Projekt považuji za velice přínosný. Upevnil mi základy IPv4 [2] a IPv6 [3] datagramu. Naučil jsem se pracovat s RAW sokety [6] . Nyní vím na jakém principu funguje ICMPv4-RFC 792 [4] a ICMPv6-RFC 4443 [5] .

Na druhou stranu jsem nad projektem strávil zhruba čtyřikrát více času, než jsem předpokládal. Bylo to z důvodů překladů RFC a následného studování RFC dokumentů, seznámení se s funkcí getaddrinfo() [7], jež nahrazuje zastaralou funkci gethostbyname().

Co v zadání nebylo implicitně řečeno, jsem implementoval po zamyšlení dle vlastního uvážení.