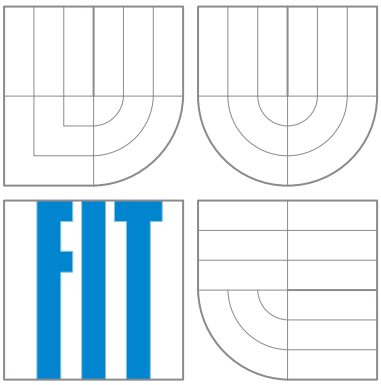
Dokumentace k projektu do předmětu ISA
My Path MTU Discovery



18. listopadu 2012

Jan Bednařík
xbedna45

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

Obsah

1 Úvod

Tento dokument se zabývá technikou zvanou Path MTU Discovery a implementací konkrétního programového řešení. V následujícím textu je shrnuta problematika zjišťování MTU a je diskutován možný způsob řešení. Popsána je rovněž konkrétní implementace programu `mypmtud`. V závěru dokumentu jsou uvedeny příklady spuštění a odpovídající výstupy programu.

2 Rozbor

2.1 Co je MTU

Pojem MTU neboli Maximum Transmission Unit představuje maximální velikost přenášeného datagramu na cestě od zdroje dat k cíli tak, aby jej nebylo nutné fragmentovat. Hodnota MTU je závislá především na použitém protokolu linkové vrstvy a na konfiguraci směrovače. Pro každou cestu (sekvenci síťových prvků mezi zdrojem dat a cílem) tedy může MTU nabývat rozdílných hodnot.

2.2 Proč hledat MTU

Znalost aktuálně platné hodnoty MTU pro danou síťovou cestu je nezbytná pro dosažení co nejnižšího vytížení síťových zařízení a přenosového pásma. Pakliže zdroj odesílá příliš velké datagramy, dochází na některém síťovém prvku k nežádoucí fragmentaci, jež zvyšuje riziko nutnosti opětovného zaslání celého datagramu. Naopak pokud zdroj odesílá příliš malé datagramy, zvyšuje na každém ze síťových prvků celkovou režii spjatou se směrováním či přepínáním paketu.

Technika dynamického vyhledávání MTU mezi zdrojem a cílem na dané síťové cestě se dle [5] označuje jako Path MTU Discovery. Operační systémy integrují implementaci Path MTU Discovery a při svém běhu tuto proceduru automaticky opakovaně provádějí a pro dané cesty uchovávají aktuální zjištěnou hodnotu MTU.

2.3 Jak hledat MTU

Hlavní myšlenkou techniky Path MTU Discovery je postupné odesílání datagramů zvolenému cíli a přizpůsobování jejich velikosti v závislosti na odezvě síťových prvků, jež se zdrojem komunikují prostřednictvím ICMP zpráv. Jinými slovy, pakliže odeslaný datagram dorazí korektně do cíle, zřejmě nepřesáhl hodnotu MTU a je možné jeho velikost zvýšit. Naopak pokud některý ze síťových prvků na cestě od zdroje k cíli datagram z důvodu přílišné velikost zahodí, zřejmě byla překročena

hodnota MTU a je nutné zaslat datagram menší velikosti.

Volba velikosti datagramu odesílaných dat probíhá již na transportní vrstvě, jež by tak mohla být využita pro implementaci Path MTU Discovery. Dle [5] se však nejedná o vhodné řešení, neboť by bylo nutné přepisovat algoritmus pro každý z protokolů transportní vrstvy. Jako příhodnější řešení se nabízí využití síťové vrstvy a protokolu IPv4, respektive IPv6.

Celé řešení stojí na práci se zprávami protokolu ICMP, nicméně vzhledem k odlišnostem protokolů IPv4 a IPv6 (potažmo ICMPv4 a ICMPv6) se řešení problému pro oba protokoly pricnipielně liší. Jednotlivým přístupům se věnují kapitoly 2.3.1 a 2.3.2.

2.3.1 IPv4

Algoritmus musí zjišťovat, zda odeslaný datagram dorazil do cíle. V případě komunikace skrze protokol IPv4 lze k tomuto účelu využít ICMPv4 zprávy. Pokdu by odeslaný datagram přesáhl (doposud neznámé) MTU síťové cesty, daný síťový prvek, který není schopný datagram odeslat vcelku, provede jeho fragmentaci na menší datagramy. Aby bylo možné MTU zjišťovat, je nutné tomuto chování zabránit nastavením bitu DF (Don't Fragment) v IPv4 hlavičce, který dle [7] zamezí síťovým prvkům fragmentaci provádět.

Zdroj odesílá ICMPv4 zprávy Echo Request (typ 8, kód 0). Pakliže datagram dorazí k cíli, obdrží zdroj ICMPv4 zprávu Echo Reply (typ 0, kód 0) a může vhodným způsobe zvýšit velikost odesílaného datagramu. Pakliže je datagram přesahuje velikost MTU, některý ze síťových prvků jej zahodí a zdroji odpoví ICMPv4 zprávou Destination Unreachable (typ 3 kód 4). V takovém případě je nutné velikost odesílaného datagramu vhodně snížit.

Dle [6] obsahuje ICMPv4 hlavička zprávy Destination Unreachable 32bitové pole označené jako *unused*. Jak uvádí [5], na výrobce směrovačů je apelováno, aby dolních 16 bitů pole *unused* použili pro uložení hodnoty MTU následujícího skoku známé směrovači, který zahodil příliš velký datagram. Zdroj by tak mohl ihned zjistit, jakou hodnotu MTU má následující datagram zvolit. Na přítomnost této infromace v ICMPv4 hlavičce se nicméně nelze spolehnout a nedá se tak pro urychlení algoritmu použít. Namísto toho je nutné využít jinou techniku, již popisuje kapitola 3.1.

2.3.2 IPv6

Analogicky s 2.3.1 pracuje protokol IPv6 s ICMPv6 zprávami. Narozdíl od protokolu IPv4 však není nutné explicitně zakazovat fragmentování datagramu, neboť se dle specifikace [4] v případě IPv6 jedná o implicitní chování.

Podobně, jako tomu bylo u IPv4, odesílá zdroj ICMPv6 zprávy Echo Request (typ 128, kód 0) a při korektním doručení přijímá Echo Reply (typ 129, kód 0). V případě přesáhnutí MTU obrdží zdroj ICMPv6 zprávu Packet Too Big (typ 2, kód 0), jež ve svém těle dle [1] musí nést hodnotu MTU pro následující skok. Tato informace výrazně urychluje celý algoritmus, neboť není nutné MTU zjišťovat stylem pokus omyl.

2.3.3 Problém černé díry

Dokument [3] zmiňuje několik problémů, které se vztahují k procesu Path MTU Discovery. Zřejmě nejvýznamnější problém, který je pro správnou funkci algoritmu nutné implementačně řešit, je pojmenován jako *Black Hole Detection*. Jedná se o to, že některé směrovače jsou administrátory kvůli bezpečnosti konfigurovány tak, aby neposílaly zpět ICMP zprávy, nebo posílali pouze některé. Algoritmus Path MTU Discovery tento problém musí vhodně reflektovat. Konkrétní řešení je představeno v kapitole 3.2.

3 Implementace

Implementace programu `mypmtud` se drží výše zmíněných principů pro zjišťování hodnoty MTU při použití protokolu IPv4 i IPv6. Program lze přeložit a spustit na unixových operačních systémech rodiny *Linux*. Staví na využití BSD soketů, konkrétně pracuje se soketem typu `RAW` a ke svému běhu tudíž vyžaduje práva uživatele root. Jako implementační jazyk byl zvolen C/C++.

3.1 Algoritmus

Samotné zjištění výsledného MTU je řešeno formou iterativního zasílání ICMP zpráv a dynamickému přizpůsobení jejich velikosti v závilosti na ICMP odpovědi od cíle nebo některého ze síťových prvků na cestě od zdroje k cíli. Změna velikosti datagramu se řídí dle algoritmu binárního vyhledávání, který zmiňuje dokument [5] jako jedno z možných řešení.

V každé iteraci se tak posouvá spodní či horní hranice možného rozsahu hodnot MTU. Spodní hranice je při spuštění programu omezena hodnotou `68 B` v případě

IPv4 (jak specifikuje [5]), respektive hodnotou `1280 B` v případě IPv6 (jak specifikuje [2]). Horní hranice je implicitně nastavena na hodnotu `1500 B`, uživatel ji však může při spuštění změnit pomocí parametru `-m`. Maximální možná hodnota MTU je omezena 16b rozsahem pole `Total Length` (určujícím velikost dat včetně hlavičky) v případě protokolu IPv4, respektive 16b rozsahem pole `Payload Length` (určujícím velikost dat bez hlavičky) v případě protokolu IPv6.

3.2 Nespadnout do černé díry

Jak bylo zmíněno v kapitole 2.3.3, je nezbytné řešit problém *Black Hole Detection*. Program `mypmtud` implementuje časově omezené čekáním na ICMP odpověď s implicitní hranicí 3 sekundy. Pakliže do této doby neobdrží některou z očekávaných ICMP zpráv, vyhodnotí situaci jako problém černé díry, tedy odeslaný datagram zřejmě přesahoval hodnotu MTU síťové cesty, tudíž jej některý směrovač zahodil, ale neposlal zpět ICMP zprávu. Program na situaci reaguje zmenšením velikosti datagramu podle algoritmu binárního vyhledávání a pokračuje další iterací.

3.3 Změna velikosti datagramu

Jak popisuje kapitola 3.1, v každé iteraci algoritmu je nutné změnit velikost odesílaného datagrmu. Velikost datagramu stanovuje IPv4 hlavička v poli `Total Length`, respektive hlavička IPv6 v poli `Payload Length`. Při použití BSD soketů tuto hodnotu doplňuje sám operační systém na základě množství odesílaných dat. Implementačně tak stačí udržovat hodnotu aktuálního odesílaného MTU a tu předávat jako parametr funkci `sendto()`. Odesílaný datagram pak obsahuje pouze patřičné hlavičky a datová část je vynulována.

3.4 Síťová komunikace

Uživatelem zadané URL cíle lze v mnoha případech přeložit na několik IP adres (pakliže nebyla zadána přímo IP adresa), případně je cíl dosažitelný skrze IPv4 i IPv6. V takových případech implementace programu `mypmtud` vyhodnotí MTU pouze pro jednu IP adresu, přičemž upřednostňuje protokol IPv6.

Jak bylo uvedeno v kapitole 2.2 operační systém sám provádí Path MTU Discovery. Aby mohl program `mypmtud` správně fungovat, je nutné operačnímu systému provádění Path MTU Discovery explicitně zakázat. Toho lze dosáhnout patřičným nastavením soketu za pomocí funkce `setsockopt()` s hodnotou `IP_PMTUDISC_PROBE` pro volbu `IP_MTU_DISCOVER`, respektive s hodnotou `IPV6_PMTUDISC_PROBE` pro volbu `IPV6_MTU_DISCOVER`.

3.5 Podporované ICMP zprávy

Kromě očekávaných ICMP zpráv reaguje program `mypmtud` také na další ICMP zprávy, jež se obecně pojí s výskytem fatálního stavu vylučujícího se s dalším během programu. Vyskytne-li se taková situace, program se ukončí s chybovým kódem.

3.5.1 ICMPv4

Korektní stav:

- Echo Reply (typ = 0, kód = 0)
- Destination Unreachable - Fragmentation required (typ = 3, kód = 4)

Nekorektní stav:

- Destination Unreachable (typ = 3, kód $\bar{4}$)
- Time Exceeded (typ = 11, kód = 1-2)

3.5.2 ICMPv6

Korektní stav:

- Echo Reply (typ = 129, kód = 0)
- Packet Too Big(typ = 2, kód = 0)

Nekorektní stav:

- Destination Unreachable (typ = 1, kód = 0-7)
- Time Exceeded (typ = 3, kód = 0-1)

4 Testování a ladění

Program byl sestaven a testován na referenčním stroji běžícím na 32bitovém operačním systému Ubuntu 12.04. Pro exaktní testování správného vyhodnocení hodnoty MTU byl využit softwarový simulátor počítačových sítí GNS3 a jednoduchá síť obsahující klientskou stanici, na níž běžel program `mypmtud`, a dva směrovače *Cisco 2691*, na nichž bylo možné měnit hodnotu MTU. Pro ladění byl využit protokolový analyzér a paketový sniffer Wireshark.

5 Příklady použití

5.1 Spuštění

```
mypmtud [-h] [-m MAXMTU] DESTINATION

        -h
           Výpis nápovědy.

        -m MAXMTU
           MAXMTU specifikuje horní hranici hledaného MTU na cestě
           od zdroje k DESTINATION.
```

5.2 Příklady výstupů

```
./mypmtud www.google.cz
Error: socket(): Attempt to create socket failed. (root required)
—
sudo ./mypmtud www.google.cz
Trying 1390 B ... message too big
Trying 1280 B ... ok

resume: 1280 bytes
—
sudo ./mypmtud -m 5000 www.google.cz
Trying 3140 B ... message too big
Trying 1280 B ... ok

resume: 1280 bytes
—
sudo ./mypmtud -m 500 www.google.cz
Error: Unsupported maximal MTU size (parameter -m).
—
sudo ./mypmtud -m 70 www.idos.cz
Trying 69 B ... ok
Trying 70 B ... ok

resume: 70 bytes
```

6 Závěr

Program `mypmtud` provádí algoritmus Path MTU Discovery na základě změny velikosti odesílaných datagramů v závislosti na příchozích ICMP zprávách či vypršení časovače. Podporuje protokol IPv4 i IPv6 a umožňuje uživateli specifikovat velikost maximálního hledaného MTU. Implementace dodržuje doporučení příslušných dokumentů RFC, ať už se jedná o vyhledávací algoritmus nebo reakce na známé problémy spjaté s Path MTU Discovery. Funkčnost programu byla ověřena v simulovaném síťovém prostředí za pomocí simulátoru GNS3.

Reference

[1] Conta, A.; Deering, S.; Gupta, M.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), Březen 2006, aktualizováno v RFC 4884.
URL http://www.ietf.org/rfc/rfc4443.txt

[2] Deering, S.; Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Prosinec 1998, updated by RFCs 5095, 5722, 5871, 6437, 6564.
URL http://www.ietf.org/rfc/rfc2460.txt

[3] Lahey, K.: TCP Problems with Path MTU Discovery. RFC 2923 (Informational), Září 2000.
URL http://www.ietf.org/rfc/rfc2923.txt

[4] McCann, J.; Deering, S.; Mogul, J.: Path MTU Discovery for IP version 6. RFC 1981 (Draft Standard), Srpen 1996.
URL http://www.ietf.org/rfc/rfc1981.txt

[5] Mogul, J.; Deering, S.: Path MTU discovery. RFC 1191 (Draft Standard), Listopad 1990.
URL http://www.ietf.org/rfc/rfc1191.txt

[6] Postel, J.: Internet Control Message Protocol. RFC 792 (Standard), Září 1981, aktualizováno v RFC 950, 4884, 6633.
URL http://www.ietf.org/rfc/rfc792.txt

[7] Postel, J.: Internet Protocol. RFC 791 (Standard), Září 1981, aktualizováno v RFC 1349, 2474.
URL http://www.ietf.org/rfc/rfc791.txt