Analýza počítačových sietí (scale – free networks)

Přemysl Kouřil, Milan Malý

Ako náš projekt sme si vybrali preskúmať štruktúru počítačovej siete a ukázať, že daná sieť je skutočne scale-free. Zároveň popíšeme algoritmy používané pri zisťovaní topológie počítačových sietí.

Cieľom analýzy scale-free sietí je:

- ***nájsť vzťahy*** medzi jednotlivými údajmi
- ***vizualizácia linkov a vzťahov***

Jedným z mnohých typov scale-free sietí sú aj počítačové siete, či už v rámci malej (privátnej) siete alebo rozľahlej kostry internetu.

Dôvody na analýzu topológie počítačovej siete:

1. **Simulácia reálnej siete**. (Využíva sa na analýzu javov v sieti: pri pripojení ďalších užívateľov, pohyb pracovníkov v rámci siete, apod.)
2. **Správa siete** – pridávanie nových smerovačov, testovanie správneho nastavenia použitého hardvéru, hľadanie miest v sieti s nízkou priepustnosťou – úzke hrdlo.
3. **Pomáha pri zorientovaní sa pri pripojení do siete** – hľadanie najbližšieho servra, najrýchlejšieho spojenia apod.
4. **Umožňuje nasadenie algoritmov**, ktoré rozširujú poznatky o topológii v sieti k jednotlivým uzlom a na základe toho zvyšujú výkon siete.

Na hľadanie štruktúry počítačových sietí existuje niekoľko algoritmov, avšak žiaden z nich nie je priamo implementovaný vo voľne dostupnom automatizovanom softvéri. Preto sme sa rozhodli, že našu analýzu vykonáme na relatívne malej sieti, avšak ručne – krok po kroku podľa algoritmu, ktorý sme zvolili.

Algoritmy na zisťovanie topológie počítačových sietí

Existuje viacero algoritmov, väčšina z nich však predpokladá podporu istých mechanizmov v sieti. Avšak v dnešnej dobe hackerských útokov už len máloktorý administrátor ponechá sieť v stave, v ktorom by umožňovala viac funkčnosti, ako je nevyhnutné.

Typickým príkladom je zákaz odpovedať na príkaz **ping**, **broadcast ping** alebo **SNMP** protokol.

Význam jednotlivých príkazov:

- **ping** – príkaz vyšle k počítaču paket „echo request“ a počítač odpovie paketom „echo response“. Overí sa tak, či daný uzol v sieti existuje.
- **broadcast ping** – podobný ako príkaz ping, avšak rozošle sa všetkým PC v danej sieti, každý z uzlov následne odpovedá iniciátorovi. (Väčšinou je však blokovaný.)
- **traceroute** – príkaz vypíše cestu od zdrojového počítača(na ktorom je príkaz spustený) k cieľovému PC
- **SNMP *(simple network management protocol)*** – protokol, ktorý slúži na správu zariadení na sieti. (Nie je však implementovaný na starších PC a na nových PC je zakázaný z dôvodu bezpečnosti.)

Požiadavky na použité nástroje a algoritmy:

1. **efektivita** – čo najmenej pri skúmaní zaťažiť sieť
2. **rýchlosť** – vykonať analýzu topológie siete v čo najkratšom čase
3. **kompletnosť** – zistiť celú topológiu siete
4. **presnosť** – vyhnúť sa chybám

A. Všeobecný algoritmus na zisťovanie topológie siete:

1. Vytvoriť „**dočasnú**“ množinu IP adries, ktoré by mohli (ale tiež) nemusia odpovedať aktuálne vyskytujúcim sa počítačom a smerovačom
2. Pre každý z prvkov dočasnej množiny v cykle:
    a. otestovať platnosť IP adresy (či v sieti skutočne existuje)
    b. Ak je adresa platná, potom zistiť jej vzťah k ostatným adresám v „**konečnej**“ množine platných IP adries a pridať túto adresu do „**konečnej**“ množiny
    c. Použiť práve pridanú adresu na vygenerovanie ďalších IP adries a pridať ich do „dočasnej“ množiny

Po skončení algoritmus generuje topológiu siete, ktorá sa skladá zo zoznamu počítačov, smerovačov a podsietí.

Popis ďalších algoritmov:

B. Odhad masky podsiete na základe broadcast pingu:

for **dĺžka_masky** = 31 to 7 do

a.) predpokladajme, že maska podsiete má dĺžku **dĺžka_masky**
b.) skonštruujeme ‘0’ a ‘255’ broadcastové adresy pre masku dĺžky **dĺžka_masky**
c.) použijeme tieto adresy pri príkaze ping
d.) ak na každý z ping príkazov odpovedali viac ako 2 počítače, potom prehlásime premennú **dĺžka_masky** za skutočnú dĺžku masky siete
inak pokračujeme v cykle

C. Odhad masky podsiete na základe už známych podsietí:

Používa sa v prípade, keď už poznáme adresu nejakých počítačov v podsieti, ktorej masku chceme určiť. Tieto počítače patria do danej podsiete a sú od smerovača vzdialené na 1 hop.

Masku získame porovnaním výsledkov bitových AND a bitových OR operácií na adresách týchto počítačov.

(Pri počítaní masky vychádzame z poznatku, že maska siete musí byť kontinuálna, t.j. najskôr ‘1‘ a potom ‘0‘.)

Metóda funguje dobre za predpokladu, že známe IP adresy sú vybrané náhodne z celej sieti.
Pozn.: AND – odhalí rovnaké časti IP adresy – teda bity, ktoré by ešte mali patriť do masky
OR – odhalí rozdielne časti IP adresy – bity určujúce adresy uzlov v rámci podsiete

D. Vyhľadávanie aktívnych uzlov v sieti:

Pokiaľ **už máme nejaký uzol v sieti**, je pravdepodobné, že budú existovať v sieti aj ďalšie uzly s adresou o N väčšou, prípadne o N menšou. Pomocou príkazu ping testujeme tieto adresy.

Ak **ešte nepoznáme žiadnu z adries počítačov** v sieti, používame nasledujúcu heuristiku. Testujeme najskôr adresy, ktoré končia na **1, 63, 129, 193**. Tieto adresy sa zvyčajne používajú pre smerovače. Ak niektorá z týchto adries odpovedá, pridáme do „**dočasnej množiny**“ N náhodne zvolených adries (je pravdepodobné, že k smerovaču náležia nejaké ďalšie uzly).

E. Algoritmus využívajúci príkaz traceroute na zisťovanie topológie:

Vysvetlenie použitia jednotlivých operácií:

Funkciu XOR používame, aby sme zistili správny sieťový interface smerovača (DNS lookup nám môže vrátiť viacero rôznych sieťových rozhraní pre daný smerovač). Najlepšia zhoda medzi sieťovým interfaceom smerovača a sieťou obsahuje najmenej ‘1’.

V algoritme si udržiavame 2 hašovacie tabuľky:

Tabuľka AND obsahuje bitové AND všetkých IP adries počítačov, pre ktoré je sieťový interface predposledným hopom v traceroute.

Tabuľka OR obsahuje bitové OR všetkých, pre ktoré je sieťový interface predposledným hopom v traceroute.

Kľúčom v tabuľke AND{} (resp. OR{}) je brána podsiete, hodnotou je bitové AND (resp. OR) IP adries počítačov v danej podsieti.

1. Do „**dočasnej množiny**“ priraď náhodné adresy z domény, ktoré končia na ‘1’.
2. Inicializuj kumulatívne hašovacie tabuľky AND{} a OR{} na null
3. pre každý **uzol** z „dočasnej množiny“ vykonaj nasledujúce akcie:
    a. ping(**uzol**)
    b. Ak tento **uzol** existuje, potom do „konečnej množiny“ uzlov pridaj **uzol**
    c. Použi heuristiku na pridanie nových uzlov do „**dočasnej množiny**“ (vyber náhodne zvolené uzly, uzly s IP adresami vzdialenými ±N od existujúceho **uzlu**)
    d. traceroute(**uzol**)
    e. Vezmi za **smerovač** predposledný počítač podľa výpisu traceroute
    f. Nájdi všetky IP adresy tohto **smerovača** použitím DNSlookup (príkaz nslookup v systéme Windows)
    g. Do premennej **brána** priraď adresu, ktorá obsahuje minimálny počet ‘1’: v (IP adrese **smerovača** XOR adresa **uzlu**)
    h. Do premennej „**stará podsieť**“ priraď hodnotu z hašovacej tabuľky: AND{**brána**} (hľadáme podľa kľúča **brána**)
    i. Do tabuľky AND{**brána**} na miesto s indexom **brána** zapíš hodnotu:

(**uzol** AND AND{brána}). //spresňujeme odhad podsiete

j. Do tabuľky OR{**brána**} na miesto s indexom **brána** zapíš hodnotu:
(**uzol** OR OR{**brána**}). //spresňujeme odhad masky

k. Do premennej „**nová podsieť**“ priraď hodnotu: AND{**brána**}

l. Spočítaj „**masku novej podsiete**“ ako:
NOT (AND{**brána**} XOR OR{**brána**})

m. Ulož **uzol** do „**novej podsiete**“

n. Ak je to potrebné, presuň IP adresy počítačov v „konečnej množine“ zo „**starej podsiete**“ do „**novej podsiete**“

Algoritmus na určenie topológie siete využíva iba **traceroute a ping**, teda je takmer univerzálny. Problémom však je pridávanie nových uzlov do „dočasnej množiny“. Aby sme dosiahli dobrý výsledok, musí byť N pomerne vysoké, čo však zvyšuje počet neexistujúcich počítačov, ktorí musíme preskúmať pomocou príkazu ping. Ak počítač s danou IP v sieti neexistuje, čaká sa na vypršanie timeoutu, čo zvyšuje dĺžku trvania algoritmu.

Zisťovanie topológie počítačových sietí - postup:

Pokúsili sme sa zistiť topológiu viacerých počítačových sietí. Nám najdostupnejšie boli počítačová **sieť na koleji Hviezda**, **sieť na Jižním městě (na koleji Vltava a Otava)** a **sieť internetového providera (nwt.cz)**.

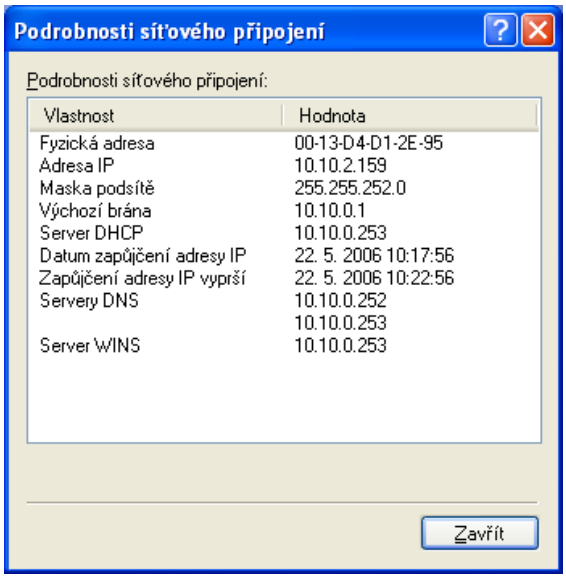
Pri zisťovaní topológie nás zaujímala ***štruktúra v rámci protokolu IP***, teda sme sa nepokúšali hľadať zariadenia na linkovej vrstve (ako sú napr. huby alebo swiche).

a.) **Zisťovanie topológie siete na koleji Hviezda** bolo zťažené tým, že sieť neodpovedala na traceroute a na broadcast ping.
Zistili sme však, že sa skladá zo ... podsietí, ktoré sú spojené cez smerovač.

	IP adresy	masky	Broadcast adresy
Podsiete			

b.) Zisťovanie topológie siete na Jižním městě:
Na začiatku sme mali len záznam z jediného počítača:

IP adresa:10.10.2.159
Maska posiete: 255.255.252.0 / 22 bitov
Brána: 10.10.0.1



Na základe tejto znalosti vlastnej IP adresy a masky siete sme zistili informáciu o možných podsieťach.

Počet možných podsietí: 64

Network ID's	Broadcast ID's	Network ID's	Broadcast ID's
10.10.0.0	**10.10.3.255**	10.10.128.0	10.10.131.255
10.10.4.0	**10.10.7.255**	10.10.132.0	10.10.135.255
10.10.8.0	**10.10.11.255**	10.10.136.0	10.10.139.255
10.10.12.0	10.10.15.255	10.10.140.0	10.10.143.255
10.10.16.0	10.10.19.255	10.10.144.0	10.10.147.255
10.10.20.0	10.10.23.255	10.10.148.0	10.10.151.255
10.10.24.0	10.10.27.255	10.10.152.0	10.10.155.255
10.10.28.0	10.10.31.255	10.10.156.0	10.10.159.255
10.10.32.0	10.10.35.255	10.10.160.0	10.10.163.255
10.10.36.0	10.10.39.255	10.10.164.0	10.10.167.255
10.10.40.0	10.10.43.255	10.10.168.0	10.10.171.255
10.10.44.0	10.10.47.255	10.10.172.0	10.10.175.255
10.10.48.0	10.10.51.255	10.10.176.0	10.10.179.255
10.10.52.0	10.10.55.255	10.10.180.0	10.10.183.255
10.10.56.0	10.10.59.255	10.10.184.0	10.10.187.255
10.10.60.0	10.10.63.255	10.10.188.0	10.10.191.255
10.10.64.0	10.10.67.255	10.10.192.0	10.10.195.255
10.10.68.0	10.10.71.255	10.10.196.0	10.10.199.255
10.10.72.0	10.10.75.255	10.10.200.0	10.10.203.255
10.10.76.0	10.10.79.255	10.10.204.0	10.10.207.255
10.10.80.0	10.10.83.255	10.10.208.0	10.10.211.255
10.10.84.0	10.10.87.255	10.10.212.0	10.10.215.255
10.10.88.0	10.10.91.255	10.10.216.0	10.10.219.255
10.10.92.0	10.10.95.255	10.10.220.0	10.10.223.255
10.10.96.0	10.10.99.255	10.10.224.0	10.10.227.255
10.10.100.0	10.10.103.255	10.10.228.0	10.10.231.255
10.10.104.0	10.10.107.255	10.10.232.0	10.10.235.255
10.10.108.0	10.10.111.255	10.10.236.0	10.10.239.255
10.10.112.0	10.10.115.255	10.10.240.0	10.10.243.255
10.10.116.0	10.10.119.255	10.10.244.0	10.10.247.255
10.10.120.0	10.10.123.255	10.10.248.0	10.10.251.255
10.10.124.0	10.10.127.255	10.10.252.0	10.10.255.255

V ďalšom kroku sme sa snažili **preskúmať** pomocou heuristiky **všetky počiatočné adresy podsietí**. Na týchto miestach sa zvyčajne nachádzajú smerovače.

Použili sme teda heuristiku a skúmali pre každú podsieť adresy tvaru: ***10.10.c.1***, kde c sa menilo podľa daného zoznamu podsietí.
Na každú zo zadaných adries sme poslali príkaz ping.

Na prikaz PING odpovedali tieto adresy:

10.10.0.1
10.10.4.1
10.10.8.1

Predpokladáme, že ide o smerovače. Väčšiu istotu nám dáva aj to, že príkaz **nslookup** (ktorý vyhľadáva meno počítača na základe IP adresy) nedokázal určiť meno pre žiadnu z týchto adries.

Na broadcast ping odpovedali:

10.10.3.255 - odpovedá 10.10.0.1,
10.10.7.255 - odpovedá 10.10.0.1,
10.10.11.255 - žiadna odpoveď

Z toho usudzujeme, že sa v skutočnosti jedná o jediný smerovač s viacerými sieťovými rozhraniami pre každú z 3 podsietí.

V ďalšom kroku máme nasledujúce možnosti:
Skenovať rozsah adries:

a.) 10.10.0.1 - 10.10.0.255
10.10.1.0 - 10.10.1.255
10.10.2.0 - 10.10.2.255
10.10.3.0 - 10.10.3.254

b.) 10.10.4.1 - 10.10.4.255
10.10.5.0 - 10.10.5.255
10.10.6.0 - 10.10.6.255
10.10.7.0 - 10.10.7.254

c.) 10.10.8.1 - 10.10.8.255
10.10.9.0 - 10.10.9.255
10.10.10.0 - 10.10.10.255
10.10.11.0 - 10.10.11.254

Náhodne vyberať adresy, zisťovať ich platnosť a následne, či sa v ich okolí nachádzajú iné odpovedajúce IP adresy.

Vybrali sme si zdĺhavejšiu možnosť a preskenovali všetky adresy pre niekoľko prvých rozsahov adries – *počet reagujúcich počítačov v zadaných podsieťach sa menil v závislosti na dni* (a síce na tom, či sú zapnuté alebo vypnuté). Cez víkend je kolej prázdnejšia ako počas týždňa. :-)

c.) **Analýza siete internetového providera (nwt.cz)**

Začínáme s adresou: 217.197.144.135
Brána 213.197.144.9 – hostynek

Hostynek:
217.197.149.135
217.197.149.134
217.197.149.128

Výpočet masky podsítě
1000 0111
1000 0110
1000 0000

1000 0000 – AND
1000 0111 – OR

0000 0111 – XOR
1111 1000 - NOT

První odhad subsítě
Hostynek – 217.197.149.128 / *29*
Výsledný adresní prostor
217.197.149.128 – 217.197.149.135

Nalezen nový host: 217.197.149.126
•Upřesněný odhad
•Hostynek – 217.197.149.0/24

nový host: 217.197.149.226
patří do subsítě hostynek ale podle traceroute má jinou bránu: 217.197.149.25

nová brána: 217.197.149.25 - host25
objevili jsme novou subsíť

host25
217.197.149.226
217.197.149.229
217.197.149.237

host25 - 217.197.149.224/28
Adresní prostor
217.197.149.224 - 217.197.149.238

silo - 217.197.144.5
subsíť 217.197.150.0/24
aktivní uzly:
–217.197.150.2
–217.197.150.150
–217.197.150.250

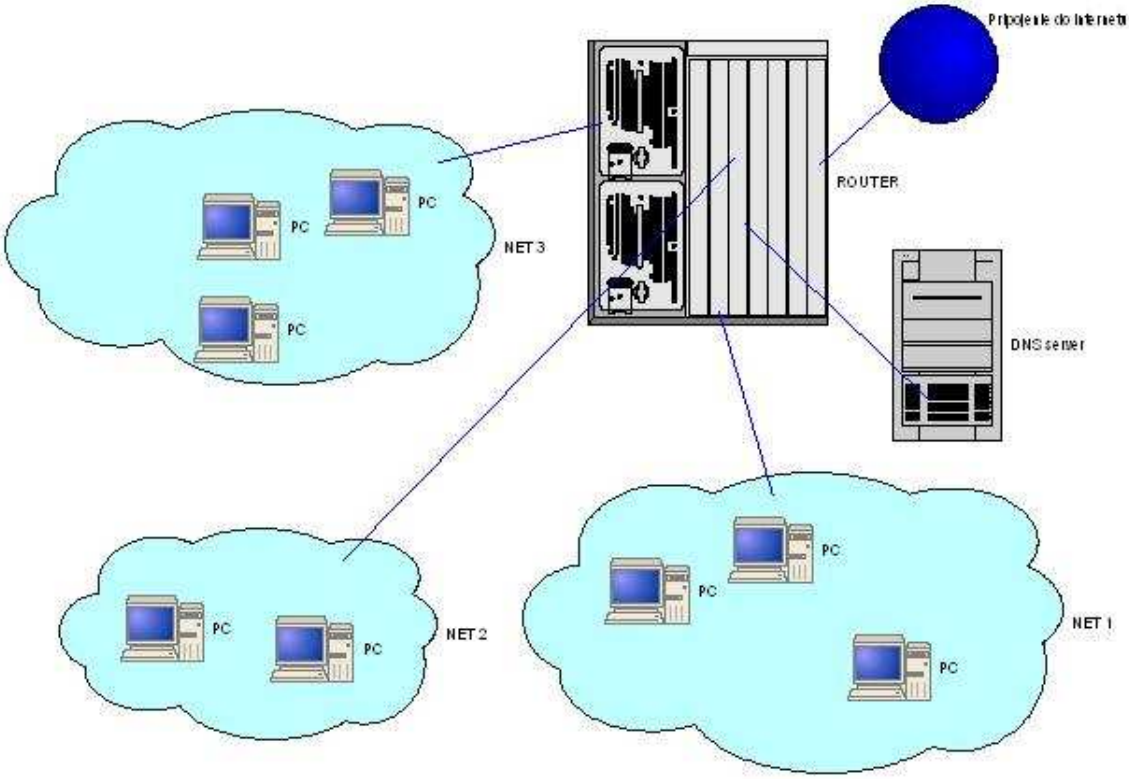
nwtcomputer1.sloane.cz - 213.192.4.130
subsíť 217.197.144.0/20
Adresní prostor
–217.197.144.1 - 217.197.159.254

Závěr:

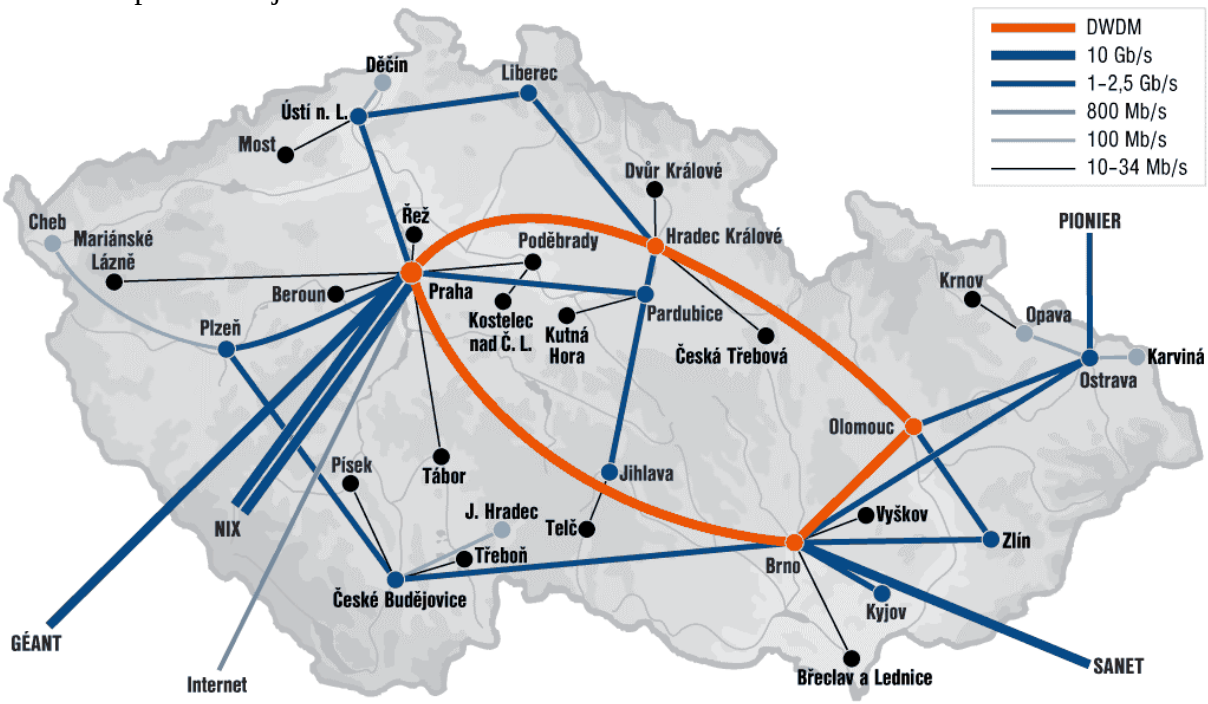
Část sítě, kterou jsme analyzovali, má poměrně jednoduchou strukturu. Aproximace subsítí zde fungovala velice dobře, a celkový výsledek analýzy poměrně věrohodně zobrazuje skutečnost. Na Internetu se však vysktují daleko složitější struktury (četná křížení apod.) které jsou na analýzu daleko složitější.

Príloha:

1. Obrázkový popis siete na Jižním městě:



2. Príklad počítačovej siete CESNET:



3. Príklad Európskej počítačovej siete – GEANT:

4. Príklad zložitejšej topológie Internetu