Na počátku bylo vejce aneb Úvodem

Internet i UNIX jsou na světě již pěkných pár pátků. Počítače pracující pod nepřeberným množstvím unixovských systémů od nepaměti fungují jako internetové poštovní servery, name-servery a menší routery. Skoro by se zdálo, že si tyto dva světy už nemají co nového říci. V posledních letech ovšem výrazně vzrostl výkon běžných PC-ček a objevily se pro ně karty podporující nejrůznější síťové technologie, a tak se tyto počítače staly vhodnými kandidáty i pro funkci routerů. Na mnoha místech tak starší PC-čko s volně šiřitelným softwarem předstihlo svého staršího a dražšího komerčního sourozence. Ačkoliv podpora routingu, firewallingu a inteligentní queue management v jádrech těchto OS (a zejména Linuxu) je již velice vyspělá, ještě nedávno byly notně zanedbávány protokoly pro dynamický routing, tradiční unixovskou koncepcí umístěné do uživatelského prostoru v podobě routovacích daemonů.

Tento nedostatek se pokoušela mezi jinými napravit také skupina nadšenců na pražské Matematicko-fyzikální fakultě UK, která se rozhodla jako svůj studijní projekt vytvořit nového routovacího daemona, který by se stal něčím víc než jen důstojným konkurentem komerčních routerů – nejen kopíroval už známá řešení, ale také přispěl něčím novým; na jedné straně být stabilním a efektivním programem pro praxi, na straně druhé ovšem i laboratoří pro snadné experimentování s novými myšlenkami. A tak se narodil projekt *BIRD Internet Routing Daemon* (zkráceně Bird[0]).

Původní plány byly smělé a rozhodné: Vytvořit nejlepšího routovacího daemona na světě, který by fungoval na všech platformách, vyžadoval minimum strojového času a paměti, podporoval všechny routovací protokoly atd., zkrátka sen, jaký má být. Přeskočme nyní dva roky vývoje[1] a podívejme se, co z toho vzešlo.

Když ptáčka lapají, pěkně mu zpívají aneb Feature list

Bird dostal od svých sudiček do vínku mnohé dary:

- Schopnost komunikovat protokoly IPv4 i IPv6.
- Podporu všech základních routovacích protokolů:
  - RIPv2 (Routing Information Protocol, viz RFC 1723 a 2080)
    Určen pro interní routing, dnes významu spíše historického.
    Algoritmus: distance vector (Bellman-Ford).
    Velmi pomalá konvergence, použitelý pouze pro maličké sítě.
  - OSPFv2 (Open Shortest Path First, viz RFC 2328)
    Určen pro interní routing, v současné době asi nejběžnější.
    Algoritmus: link state (Dijkstra).
    Rychlá konvergence, ale velice komplikovaný.
    Zatím podporujeme pouze pro IPv4.
  - BGP4 (Border Gateway Protocol, viz RFC 1771 a 2283)
    Standardní protokol pro externí routing mezi AS.
    Rychlá konvergence, možnost administrativní filtrace.
    Algoritmus: path vector (upravený Bellman-Ford).
- Propojování protokolů a výměnu dat mezi nimi.

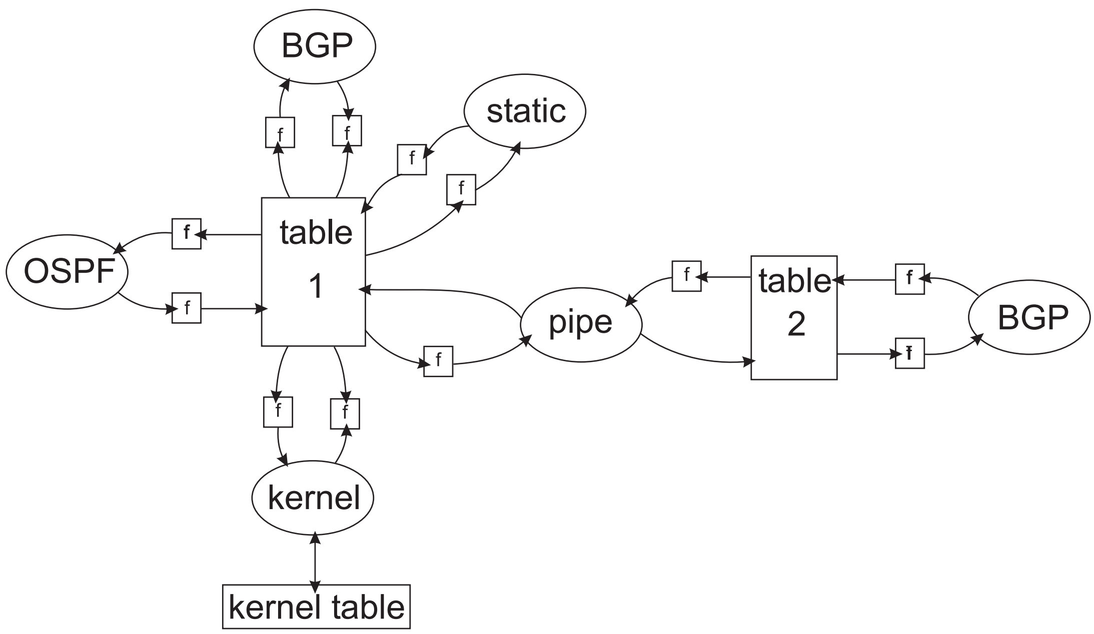
[0] Projekt si samozřejmě vysloužil nejrůznější přezdívky – od Ptáčete až po Ptákovinu.

[1] během nichž z některých požadavků samozřejmě sešlo – například logem projektu se nakonec nestal ptáček nesoucí v zobáčku snítku byliny řečené routa …

- Schopnost uplatňovat routing policy – určovat, které routovací informace budou od kterých protokolů přijímány a do kterých protokolů vysílány.
- Snadnou konfiguraci a údržbu, ale o tom až později.
- Modulární architekturu umožňující snadné doplňování nových protokolů, filtrů i jednoduché portování na další systémy. (Interface k routovacím službám jádra je bohužel i mezi jednotlivými unixovskými systémy natolik různorodý, že dokonce i Linux 2.0 a Linux 2.2 jsou naprogramovány jako dva odlišné porty.)
- A v neposlední řadě kvalitní dokumentaci jak uživatelskou, tak programátorskou (to je vítaný důsledek bytí školním projektem).

Svět z ptačí perspektivy aneb Jak to všechno dát dohromady?

Hlavním problémem ovšem není ani tak všechny tyto vymoženosti naprogramovat a popropojovat, nýbrž udělat to tak, aby z toho nevznikl neudržovatelný a neefektivní zmatek[2]. Z toho se zrodil modulární pohled na svět routingu a následně i celá modulární architektura programu:



Základem pohledu na Internet z naší „ptačí perspektivy“ je routovací tabulka. Na ní jsou připojeny jednotlivé routovací protokoly – jak protokoly reálné, tak i několik virtuálních, jako je například protokol Kernel zajišťující synchronizaci tabulky s tabulkou jádra nebo protokol Static generující podle své konfigurace statické směrování. Každý z protokolů posílá do tabulky položky pro všechny routy, které se na základě svých informací o topologii sítě dozvídá. Tabulka pro každou síť vybírá na základě preferencí jednotlivých protokolů a jejich metrik optimální směr a ten oznamuje zpět protokolům. V cestě ovšem v obou směrech stojí filtry, které mohou tok dat regulovat – některé položky odmítat, jiné upravovat (například jim přenastavovat metriky či tagy; to se dokonce vztahuje i na atributy doplňované cílovým protokolem, protože nejprve cílový protokol nastaví implicitní hodnoty svých atributů, pokud je již položka neobsahovala, a teprve pak dojde k filtrování).

[2] Nádherným příkladem je Mendelův pes, jak ho kdysi nakreslil pan Kantorek.

Routovacích tabulek může dokonce existovat vícero a pomocí protokolu Pipe si mohou přeposílat vybrané položky. Tím můžeme zajistit například routing závislý na zdrojové adrese či vstupním interfacu: u Linuxu 2.2 či novějšího nakonfigurujeme jádru více routovacích tabulek a každou z nich připojíme na jednu tabulku naši.

Ptačí zpěv aneb User interface

Síť se mění a s ní se musí měnit i konfigurace síťových komponent. Ta se pozvolna stala tradičním kamenem úrazu většiny routerů. U tohoto kamene se totiž scházejí dva odvěcí nepřátelé: snaha o snadnost a flexibilitu konfigurace a snaha o nepřetržitý provoz sítě. A jednou sleví ten, podruhé onen – autoři některých routerů vsadili na sílu textových konfiguračních souborů za cenu toho, že se po každé změně konfigurace musí router restartovat a po nějakou dobu neroutuje, jiní zase podřídili non-stop provozu vše ostatní a stvořili konfigurování nepřeberným množstvím online příkazů, pomocí nichž se některé věci provádějí snadno, jiné, na které autoři speciální příkaz nevymysleli, už obtížněji a ještě jiné vyžadují do výsledného stavu dospět postupným provedením mnoha změn, což má obvykle za důsledek daleko delší výpadek než jaký by byl vznikl reloadem celého routeru.

Bird se snaží dosáhnout obojího: používá textové konfigurační soubory, uživateli tak dává možnost vytvářet konfiguraci víceméně libovolným způsobem – ať již ručním editováním, interaktivními programy či automatickým generováním pomocí scriptů. Po změně konfigurace je ovšem schopen si novou verzi souboru přečíst, porovnat s právě používanou konfigurací a změnám se za běhu přízpůsobit (pokud jsou změny příliš velké, může to způsobit restart některého z protokolů, ale provoz těch, kterých se změna netýkala, to nijak neohrozí).

Konfigurace každého protokolu (lépe řečeno každé jeho instance – protokoly mohou být spuštěny několikrát na různých rozhraních či nad různými tabulkami) vypadá přibližně takto:

```
protocol bgp TestBGP {                # instanci si můžeme nazvat
   local as 65000;                    # náš AS
   neighbor 195.39.3.64 as 5588;      # sousední AS
   export all;                        # posílat budeme všechno
   import filter {                    # na vstupu některé odmítneme
      if !(bgp_path ~ / ? 5588 ? /) then reject;
      if net ~ [ 10.0.0.0/8+, 192.168.0.0/16+ ] then reject;
      preference = 101;               # zbylým nastavíme preferenci
      accept;                         # a přijmeme je
   };
}
```

Vždy popisuje protokol, jeho parametry, připojení k tabulkám (v našem případě používáme tabulku implicitní) a nastavení vstupních i výstupních filtrů.

Filtry jsou popisovány jednoduchým procedurálním programovacím jazykem, který má k dispozici všechny informace o právě zpracovávané položce, může se podle nich rozhodovat a libovolně je měnit. Díky tomu je možné jednoduše zadávat i velice složitá pravidla, definovat si podprogramy sdílené filtry patřícími k více protokolům, i počítat metriky pro jeden protokol na základě metrik protokolů ostatních.

Vítaným pomocníkem při odhalování problémů v sítích je rovněž volitelné logování důležitých událostí a trasování činnosti protokolů: zde si je možno pro každý protokol vyžádat prakticky cokoliv počínaje základními informacemi o běhu protokolu (připojování a odpojování sousedů apod.), přes výměnu položek mezi protokolem, filtry a jeho tabulkou, a konče detailním výpisem všech přijatých i odeslaných paketů.

Mimo to BIRD disponuje „dálkovým ovládáním“ – jednoduchým příkazovým rozhraním, přes které se mohou připojovat různí klienti a vydávat jak některé řídící příkazy (restarty protokolů, reload konfigurace, přepínání trasování, ...), tak požadavky na vypisování stavu routovacích tabulek a protokolů. U těchto příkazů je možno využívat plné síly filtrovacího jazyka, takže chceme-li například znát všechny routy směrované pomocí BGP na jednoho konkrétního souseda, stačí položit dotaz typu

```
bird> show route where source=RTS_BGP && gw=62.168.0.1
```

a dozvíme se ihned vše, co jsme potřebovali. Ladění filtrů navíc usnadní i to, že se můžeme zeptat na obsah routovací tabulky z pohledu některého z běžících protokolů.

Cesty tažných ptáků aneb Za hranicemi všedních dnů

Hotová modulární implementace routovacího daemona samozřejmě svádí i k méně tradičnímu využití. Zde se fantazii meze nekladou, autory samé během vývoje napadly například tyto triky:

- *multirouter* – „schizofrenní“ zařízení pracující na různých skupinách interfaců jako různé routery řídící se různými pravidly, ležící třeba i v různých autonomních systémech. To je možno zařídit patřičným nastavením jádra a buďto více běžícími BIRDy nebo dokonce jedním pracujícím s několika různými routovacími tabulkami. Tak by mohl například celý pražský NIX běžet na jediném routeru (pravda, běžné PC by na to nestačilo), a přesto by si každý provider mohl sám určovat a konfigurovat svou vlastní routovací politiku.
- *šedá eminence* – BIRD by ani nemusel běžet přímo na stroji, který pakety routuje, mohl by také hotové routovací tabulky diktovat nějakému dedikovanému routeru disponujícím výkonnějším hardwarem, a tak skloubit špičkovou propustnost „velkého železa“ s flexibilitou routeru bežícího pod unixovským systémem.
- *inteligentní mirror* – aplikace se nemusí zastavovat u routingu jako takového, mnohdy je možno informace získané z provozu routeru využívat k dalším účelům, například k inteligentnímu přesměrovávání klientů na nejbližší mirror vašeho archivu.

Čas ptáčat aneb Co dál?

Přestože po úspěšném obhájení projektu a prvních několika desítkách spokojených uživatelů (a samozřejmě i pár opravených chybách) nyní vývoj spíše stagnuje, autoři již spřádají plány do budoucnosti, které by měly BIRDovi přinést mimo jiné také routování multicastů, agregaci síťových prefixů, OSPFv3 pro IPv6, on-demand linky a porty na další systémy.

Snůška odkazů na závěr

BIRD TEAM pod vedením RNDr. Libora Forsta (*forst@cuni.cz*) tvořili:

Ondřej Filip	*feela@ipex.cz*	*http://feela.ipex.cz/*
Martin Mareš	*mj@ucw.cz*	*http://atrey.karlin.mff.cuni.cz/~mj/*
Pavel Machek	*pavel@ucw.cz*	*http://atrey.karlin.mff.cuni.cz/~pavel/*

Domácí stránku projektu najdete na *http://bird.network.cz/*, odtamtud již vedou odkazy na aktuální verzi, online dokumentaci i mailing list uživatelů. Všechna zmiňovaná RFC i mnoho dalších naleznete na *http://www.rfc-editor.org/*, první myšlenky o využívání ptactva v Internetu sahají až k RFC 1149. BIRD není na světě sám, má i příbuzné; mezi nimi za zmínku stojí daemoni GATED (*http://www.gated.org/*), ZEBRA (*http://www.zebra.org/*) a MRTD (*http://www.mrtd.net/*). O IPv6 se bohužel zatím jen píše, na jeho celointernetové nasazení se stále čeká a mezitím přibývá address-translatorů a dalších zvěrstev. Snad jediná větší síť používající tento protokol je experimentální virtuální síť 6-bone (viz *http://www.6bone.net/* a odkazy odtamtud).

Dotazy a připomínky posílejte buďto autorovi nebo celému BIRD TEAMu na adresu *bird@bird.network.cz*.