DiProNN: Distribuovaný programovatelný síťový prvek s podporou virtuálních strojů

Tomáš Rebok (xrebok@fi.muni.cz)

Fakulta informatiky
Masarykova univerzita v Brně
Botanická 68a, 602 00 Brno

CESNET, z.s.p.o.
Zikova 4, 160 00 Praha 6

Abstrakt

Koncept aktivních sítí navyšuje stávající pasivní architekturu počítačových sítí o novou síťovou vrstvu, která umožňuje provádění uživatelských programů na vnitřních prvcích sítě, nazývaných programovatelné/aktivní směrovače. Tyto uživatelské (aktivní) programy zpracovávají uživateli zasílaná data uvnitř sítě, takže funkcionalitu potřebnou pro jejich zpracování není potřeba zajišťovat na všech koncových uzlech podílejících se na dané komunikaci, ale pouze na vnitřním aktivním prvku. Příkladem možných aplikací je transkódování vysocekvalitního videa do nízké kvality pro účely videokonferencí tak, aby bylo dostupné i pro uživatele připojené linkami s nízkou propustností, nebo kódování dat procházejících nedůvěryhodným síťovým kanálem.

Díky stále narůstajícím rychlostem síťových linek, s nimiž ovšem narůstají i požadavky aplikací na jejich propustnost, se samostatný aktivní uzel stává pro zpracování vysokorychlostních dat v reálném čase nepoužitelným v případech, kdy je toto zpracování složitější.

Příspěvek obsahuje popis architektury síťového prvku nazvaného DiProNN (Distributed Programmable Network Node), který s využitím výhod plynoucích z návrhu jeho architektury založené na virtuálních strojích vylepšuje škálovatelnost takovýchto aktivních systémů jak co do počtu současně běžících aktivních programů, tak co do maximální rychlosti toku jednoho zpracovávaného síťového proudu. Prvek DiProNN je určen zejména pro zpracovávání síťových proudů, a proto pro usnadnění jeho programování v příspěvku navrhujeme i vhodný programovací model, který programování proudových aplikací pro DiProNN zejména díky virtualizaci jeho architektury značně usnadňuje.

1 Úvod

Na dnešní počítačové sítě lze nahlížet jako na pasivní transportní médium, které přenáší (případně v případě tzv. *best-effort* služeb se snaží přenášet) data od jejich odesílatele k příjemci, přičemž celý přenos je uskutečněn bez zásahu vnitřních prvků sítě do datové části paketů[1]. Nicméně, zejména pro menší až střední specializované skupiny může být možnost zpracování přenášených dat již na vnitřních prvcích sítě velmi žádoucí. Proto lze pro tyto účely pasivní sítě rozšířit o vrstvu, která pasivní transportní médium promění na médium aktivní schopné uvnitř sítě zpracovávat přenášená data dle požadavků jejich odesílatele (uživatele). Příkladem aplikací s výhodou využívajících tohoto principu jsou například transkódování vysocekvalitního videa do nízké kvality pro účely videokonferencí tak, aby bylo v reálném čase dostupné i pro uživatele připojené linkami s nízkou propustností, nebo kódování dat procházejících nedůvěryhodným síťovým kanálem.

Aktivní sítě (či *Programovatelné sítě*) jsou pokusem, jak takto inteligentní a flexibilní síť s využitím stávajících pasivních sítí jako komunikační infrastruktury vytvořit. V aktivních sítích mají uživatelé možnost

[1] V tomto případě nebereme v úvahu firewally, proxy servery a jim podobné prvky sítě, které sice pakety svým způsobem zpracovávají, avšak nikoli uživatelem řízeným způsobem.

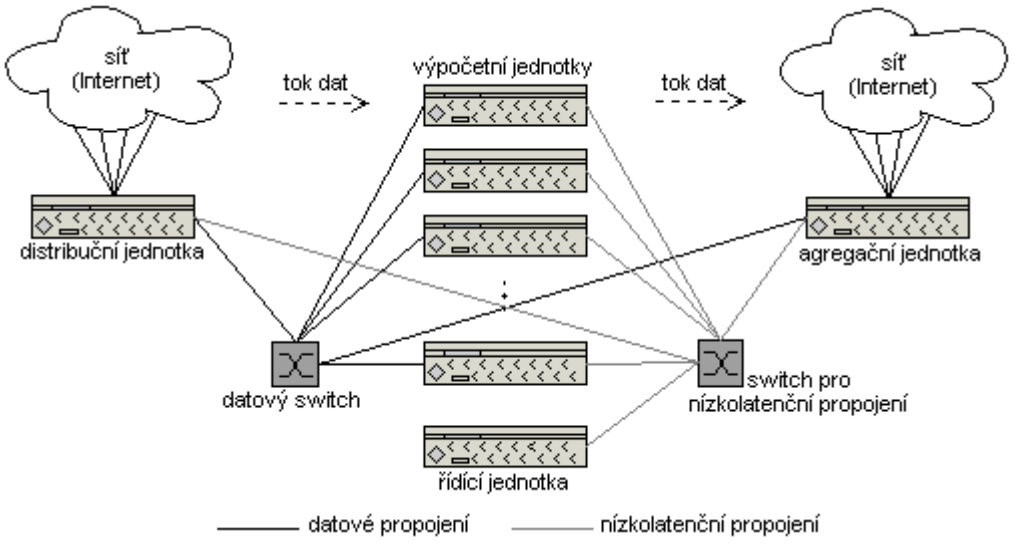
nahrání svých vlastních uživatelských programů do vnitřních prvků sítě, které jsou nad jimi zasílanými daty následně vykonávány. Tyto vnitřní prvky se nazývají *aktivní uzly* či *aktivní/programovatelné směrovače*.

Díky stále narůstajícím rychlostem síťových linek, s nimiž ovšem narůstají i požadavky aplikací na jejich propustnost, se samostatný aktivní uzel stává pro zpracování vysokorychlostních dat v reálném čase nepoužitelným v případech, kdy je toto zpracování složitější (zpracování videa, kódování dat, atp.). Pro zvýšení škálovatelnosti takovýchto aktivních systémů jak co do počtu současně běžících aktivních programů, tak co do maximální rychlosti toku jednoho zpracovávaného síťového proudu je zcela nezbytná distribuce procesorové a/nebo síťové zátěže.

Hlavním cílem naší práce je návrh architektury distribuovaného programovatelného síťového uzlu, který je s využitím clusteru složeného z několika PC propojených nízkolatenční komunikační sítí (známého pod pojmem *tightly coupled cluster*) schopen provádět distribuované zpracování jím procházejících dat. Architektura navrhovaného programovatelného síťového uzlu (pojmenovaného *DiProNN – Distributed Programmable Network Node*) je založena na principu virtuálních strojů [1], který navíc umožňuje dosažení vhodného stupně izolace souběžně běžících procesů různých uživatelů. Jelikož je prvek DiProNN určen zejména pro zpracovávání síťových proudů, navrhujeme v příspěvku pro usnadnění jeho programování i vhodný programovací model, který programování proudových aplikací zejména díky výhodám plynoucím z virtualizace jeho architektury značně usnadňuje.

Zbytek příspěvku je organizován následovně: sekce 2 je věnována popisu architektury navrhovaného distribuovaného programovatelného síťového uzlu, přičemž navrhovaný programovací model je představen v sekci 3. Současný stav v oblasti architektur existujících distribuovaných aktivních směrovačů je popsán v sekci 4 a závěrečné poznámky a návrhy naší budoucí práce jsou uvedeny v sekci 5.

2 DiProNN: VM-based Distributed Programmable Network Node



Obrázek 1.: Navrhovaná architektura síťového prvku DiProNN.

Architektura síťového uzlu DiProNN předpokládá architekturu zobrazenou na Obrázku 1. Jednotlivé komponenty DiProNNu tvoří výpočetní cluster propojený dvěma typy komunikačních kanálů:

- jedním *nízkolatenčním řídícím propojením* sloužícím pro interní komunikaci a synchronizaci uvnitř prvku DiProNN,

- alespoň jedním[2] *datovým propojením* sloužícím pro příjem a odesílání dat.

Nízkolatenční síťové propojení je nezbytné zejména z důvodu, že aktuální běžná síťová rozhraní (například gigabitový či 10Gbps Ethernet) poskytují sice velmi velkou šířku pásma, avšak zpoždění přenosu jimi generované je v řádech stovek μs, což je pro rychlou synchronizaci interních komponent uzlu DiProNN nepoužitelné. Z tohoto důvodu návrh DiProNNu počítá se specializovaným nízkolatenčním síťovým propojením (například Myrinet či InfiniBand), které umožní interní komunikaci uvnitř DiProNNu v řádech 10 μs pro Myrinet, v případě InfiniBandu dokonce až 4 μs.

Průchod dat uzlem DiProNN lze ve zkratce popsat následovně: příchozí data jsou nejprve přijata Distribuční jednotkou, která je dále přeposílá příslušné Výpočetní jednotce pro zpracování. Poté, co jsou data kompletně zpracována (zpracování se může dít na více Výpočetních jednotkách, viz dále), jsou agregována v Agregační jednotce a odeslána zpět do sítě dalšímu uzlu či příjemci. Z Obrázku 1. lze vyčíst, že architektura DiProNNu sestává ze čtyř hlavních komponent:

- **Distribuční jednotka (*Distribution Unit*) –** tato jednotka přijímá vstupní datový tok, který následně přeposílá příslušné (či příslušným) Výpočetním jednotkám pro zpracování. Výpočetní jednotka, na kterou mají být data přeposlána, je určena Řídící jednotkou DiProNNu (viz dále).
- **Výpočetní jednotky (*Processing Units*) –** každá výpočetní jednotka přijímá ARTP pakety zasílané Distribuční jednotkou či předchozím aktivním programem (detaily o komunikačním protokolu využitém v prvku DiProNN jsou uvedeny v podkapitole 2.2), které následně přeposílá pro zpracování do vhodného aktivního programu uvnitř ní běžícího. Zpracovaná data jsou následně přeposlána dalšímu aktivnímu programu pro další zpracování (který obecně může běžet uvnitř jiné Výpočetní jednotky) či Agregační jednotce, která data odesílá zpět do sítě.
  Každá Výpočetní jednotka je navíc s využitím nízkolatenčního propojení schopna komunikovat s ostatními jednotkami. Tato komunikace je mimo účely vyrovnávání zátěže či reakce na chyby využita zejména pro zasílání řídících zpráv uvnitř DiProNNu (například pro synchronizaci, řízení zpracování, sdílení stavu, atp.).
- **Řídící jednotka (*Control Unit*) –** Řídící jednotka je zodpovědná za řízení chování celého uzlu DiProNN a jeho komunikaci s okolím, včetně komunikace s jeho uživateli za účelem vyjednávání nových relací (detaily o relacích uzlu DiProNN jsou uvedeny v sekci 3), zjišťování jejich aktuálního stavu či jejich ukončování.
  V rámci interní komunikace DiProNNu Řídící jednotka prostřednictvím nízkolatenčního propojení přímo komunikuje s řídícím modulem běžícím uvnitř každé Výpočetní jednotky, čímž řídí zpracování dat v rámci dané relace. Mimoto pak Řídící jednotka komunikuje i s podsystémem pro správu zdrojů každé Výpočetní jednotky, čímž spravuje zdroje celého uzlu DiProNN. Na základě těchto znalostí pak může Řídící jednotka například rozhodnout o tom, zda požadavek na novou DiProNN relaci může být uspokojen či nikoli, nebo určit okamžik, kdy bude určitý virtuální stroj migrován [2] na jinou výpočetní jednotku z důvodu efektivního využívání zdrojů DiProNNu a umožnění jeho maximálně efektivního chování.
- **Agregační jednotka (*Aggregation Unit*) –** hlavním úkolem Agregační jednotky je skládání datových toků vystupujících z Výpočetních jednotek a jejich následné odesílání do sítě.

2.1 Architektura výpočetní jednotky

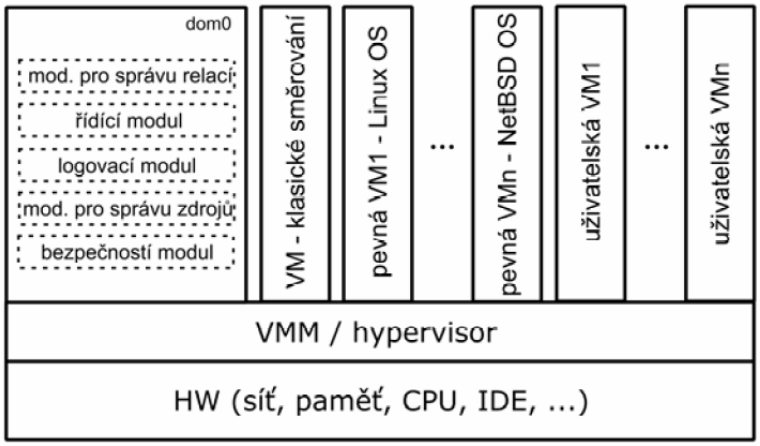
Jelikož lze na výpočetní jednotku DiProNNu nahlížet jako na samostatný programovatelný síťový uzel, pro její návrh jsme se nechali inspirovat architekturou aktivního směrovače [3], který byl vyvíjen na Masarykově univerzitě v Brně ve spolupráci se zájmovým sdružením CESNET. Mezi nejvýznamnější vy-

[2] Vstupní datové rozhraní může být totožné s výstupním.

lepšení, kterými původní návrh prošel, patří zejména úprava jeho základní architektury vedoucí k podpoře virtuálních strojů (VM).

Využití virtuálních strojů umožňuje uživatelům DiProNNu nahrávat do něj nejen aktivní programy, které jsou poté spouštěny uvnitř nějakého virtuálního stroje běžícího na DiProNNu, ale zejména přináší možnost nahrání celého uživatelského virtuálního stroje se svým vlastním operačním systémem a množinou aktivních programů uvnitř něho, které pak procházející uživatelská data zpracovávají. Tento přístup přináší řadu výhod i pro administrátory DiProNNu, kterým je umožněno spustit množinu fixních („zabudovaných“) virtuálních strojů, obecně se zcela odlišným operačním systémem a se zcela odlišnou funkcionalitou, provádějící žádané služby. Princip virtuálních strojů navíc umožňuje striktní plánování zdrojů (například CPU, paměť či přístup na diskový subsystém) jednotlivým virtuálním strojům.

Architektura výpočetní jednotky DiProNNu je zobrazena na Obrázku 2. Mezi hlavní úkoly systému pro správu virtuálních strojů (*VMM – Virtual Machine Monitor*, *hypervisor*) dané výpočetní jednotky je nahrávání, startování a ukončování virtuálních strojů, komunikace s Řídící jednotkou DiProNNu a komunikace a předávání nezbytných informací virtuálnímu stroji spravujícímu danou Výpočetní jednotku (dom0). Jak již bylo napsáno dříve, virtuální stroje běžící uvnitř DiProNNu a řízené modulem pro správu relací (*session management module*) mohou být jak fixní virtuální stroje poskytující pevnou funkcionalitu danou administrátorem DiProNNu, tak virtuální stroje nahrané uživateli. Příkladem virtuálního stroje poskytujícího administrátorem určenou fixní funkcionalitu může být virtuální stroj určený pro klasické směrování „neaktivních“ paketů (viz Obrázek 2.). Kromě toho může administrátor DiProNNu spustit množinu virtuálních strojů určených pro provádění uživateli nahraných aktivních programů (těch, které nejsou nahrány s celým virtuálním strojem). Tento přístup uživatelům DiProNNu nevnucuje nezbytnost nahrání celého virtuálního stroje v případech, kdy je nahrání pouhého aktivního programu dostačující.



Obrázek 2.: Architektura Výpočetní jednotky prvku DiProNN.

2.2 ARTP: Komunikační protokol DiProNNu

Komunikačním protokolem využitým v prvku DiProNN je transportní protokol nazvaný ARTP (*Active Router Transmission Protocol*, [4]), který byl původně navržen a implementován pro aktivní směrovač popsaný v [3].

ARTP je spojově orientovaný transportní protokol poskytující důvěryhodný oboustranný komunikační kanál, který nezaručuje, že přenášená data budou přijata ve stejném pořadí, v jakém byla odeslána. Protokol ARTP podporuje přenos dvou typů dat:

- *řídících dat* určených pro správu koncových aplikací aktivního směrovače (tyto zprávy nemohou být fragmentovány) a
- *vlastních dat* zasílaných mezi oběma komunikujícími partnery.

Data jsou aplikací protokolu ARTP předávána ve formě datových bloků nazývaných *ARTP datagramy*, které mohou být libovolné velikosti. V případě, že je vzhledem k jejich velikosti nelze do sítě odeslat najednou, zajistí protokol ARTP jejich fragmentaci do *ARTP paketů* na straně odesílatele, které přenese k určenému příjemci, kde jsou ARTP pakety opět složeny do původního ARTP datagramu (v okamžiku, kdy jsou všechny jeho fragmenty úspěšně přijaty), který je předán přijímající aplikaci. Pořadí přenášených ARTP datagramů není jednoznačně určeno – ARTP protokol pouze zajišťuje jejich správnou fragmentaci a zpětnou defragmentaci.

3 Programovací model prvku DiProNN

V této sekci je představen programovací model využitý pro programování DiProNNu. Jeho základní idea je založena na principech workflow (viz [5]), přičemž rovněž využívá i myšlenek prezentovaných v rámci projektu StreamIt ([6]). StreamIt je programovací jazyk a překladač speciálně navržený pro programování moderních proudových aplikací.

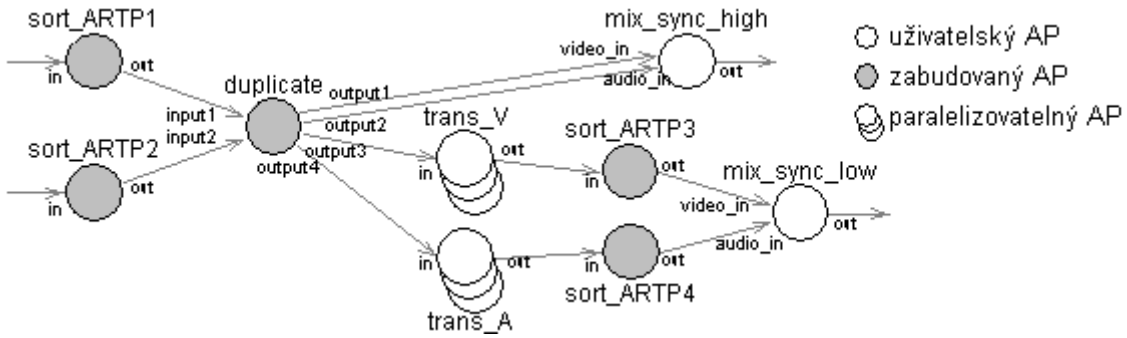
Programovací model uzlu DiProNN přebírá myšlenku nezávislých výpočetních bloků (filtrů v terminologii jazyka StreamIt) podílejících se na zpracování celého datového proudu, které při vhodném složení do grafu zpracování představují požadovanou komplexní funkcionalitu. Procesními bloky v uzlu DiProNN jsou již zmíněné aktivní programy a datová komunikace mezi těmito programy je díky virtualizačním mechanismům poskytována za pomoci hypervisora (pro tuto komunikaci jsou využity pouze standardní prostředky síťové komunikace – detaily jsou uvedeny v sekci 3.1). Takto propojené aktivní programy pak utvářejí tzv. „*DiProNN relaci (DiProNN session)*“ popsanou svým „*DiProNN relačním grafem (DiProNN session graph)*“, což je grafická reprezentace „*DiProNN programu*“ (příklad je uveden na obrázku 3.).

DiProNN program určuje aktivní programy (případně i s virtuálními stroji, ve kterých dané aktivní programy běží), které jsou nezbytné pro zpracování DiProNN relace, a zároveň definuje datovou a řídící komunikaci mezi nimi. Dále může DiProNN program definovat další parametry aktivních programů či celé DiProNN relace a/nebo požadavky na zdroje, které jednotlivé aktivní programy či celá DiProNN relace pro svůj korektní běh vyžadují. Speciálním parametrem aktivních programů je příznak „*parallelizable*“, kterým uživatel určuje, že daný aktivní program může běžet paralelně[3].

Pro dosažení vhodné úrovně abstrakce jsou všechny aktivní programy, stejně jako vstupní/výstupní datové/řídící kanály odkazovány za pomoci hierarchických jmen tak, jak je zobrazeno v DiProNN programu následujícího příkladu.

Příklad. Uvažujme následující situaci: mějme jeden vstupní proud video dat ve vysoké kvalitě (a tudíž využívající velké kapacity síťových linek, například nekomprimovaný HD proud využívající 1,5 Gbps) a jeden vstupní proud audio dat (opět ve velmi vysoké kvalitě). Nechť jsou oba tyto proudy přenášeny v reálném čase (například v rámci videokonference) za pomoci protokolu ARTP. Cílem je mít uvnitř sítě aktivní prvek (například uzel DiProNN), který dokáže oba proudy dat transkódovat do nižší kvality pro specifikovanou (předem neznámou) množinu uživatelů majících připojení do sítě s nízkou přenosovou rychlostí tak, aby byly i pro ně v reálném čase dostupné; pro uživatele připojené vysokorychlostním připojením však musí oba proudy zůstat dostupné v původní kvalitě. Na výstupu z transkódovacího prvku pak musí být audio a video proudy dané kvality (z určitých důvodů) složeny do jednoho výstupního proudu a tudíž také s definovanou přesností synchronizovány. Na následujícím obrázku lze vidět relační graf možného DiProNN programu včetně fragmentu samotného DiProNN programu.

[3] Zde je vhodné poznamenat, že DiProNN relace si musí sama určit, jakým způsobem budou data mezi paralelně běžící aktivní programy distribuována (uživatel může využít některý z nabízených distribučních principů, například *round-robin* či jednoduchou duplikaci mezi všechny paralelní instance, nebo si vhodné distribuční schéma může sám navrhnout).




```
Project first_project.HD_transcode;
# parametry projektu (vlastník, notifikace, požadované zdroje, ...)
{AP name="duplicate" ref=localService.duplicate;
    # parametry aktivního programu
    inputs = input1, input2;
    output1 = my_VM.mix_sync_high.video_in;
    output2 = my_VM.mix_sync_high.audio_in;
    output3 = localService.trans_V.in;
    output4 = localService.trans_A.in;
}
{VM name="my_VM" ref=my_VM_image;
    # parametry virtuálního stroje
    {AP name="mix_sync_high" ref=mixer_syncer;
        inputs = video_in, audio_in;
        precision = 0.001;  # 1ms
        out = DiProNN_OUT;
                        # speciální výstupní interface – výstup z uzlu DiProNN
    }
    # další aktivní programy uvnitř daného virtuálního stroje...
}
# další virtuální stroje/aktivní programy ...
```

Obrázek 3.: Příklad možného DiProNN relačního grafu spolu s fragmentem příslušného DiProNN programu.

3.1 Tok dat uzlem DiProNN

V okamžiku příchodu požadavku na novou DiProNN relaci Řídící jednotka na základě znalosti aktuálního stavu zdrojů či bezpečnostní politice daného DiProNN uzlu rozhodne, zda může být příchozí požadavek uspokojen či nikoli. V prvním případě, kdy požadavek uspokojen být může, nastává fáze sestavení nové DiProNN relace, jejíž součástí je například nahrání aktivního/aktivních programů včetně DiProNN programu, rozhodnutí, na které/kterých Výpočetních jednotkách budou běžet, alokace zdrojů, spuštění příslušných aktivních programů, ustavení komunikačního rozhraní, a podobně.

Jakmile je nová relace ustavena, tok dat DiProNNem lze ve stručnosti popsat následovně: v okamžiku, kdy na vstupní rozhraní Distribuční jednotky dorazí ARTP paket, je s využitím mechanismů GRE (*Generic Routing Encapsulation*) zapouzdřen do nového paketu, který je následně odeslán prvnímu aktivnímu programu podílejícímu se na zpracování dané DiProNN relace (respektive na vstupní rozraní příslušného virtuálního stroje, ve kterém daný aktivní program běží. Tam je původní ARTP paket z příchozího zapouzdřeného paketu opětovně vybalen; po příchodu všech nutných ARTP paketů jsou tyto složeny do ARTP datagramu, který je následně předán příslušnému aktivnímu programu pro zpracování). Po zpracování je ARTP da-

tagram[4] opět fragmentován do ARTP paketů, které jsou s využitím GRE zapouzdřeny do nových paketů a přeposlány následujícímu aktivnímu programu pro další zpracování. Nakonec jsou tyto zapouzdřené pakety směrovány Agregační jednotce, kde jsou vybaleny a odeslány do sítě příjemci či dalšímu DiProNN uzlu.

Jelikož navrhovaný programovací model pro komunikaci mezi aktivními programy využívá symbolických jmen (jak pro samotné aktivní programy, tak pro komunikační kanály), musí být tato symbolická jména při startu DiProNN relace asociována s příslušnými čísly síťových portů. Tato asociace je uskutečněna s využitím řídícího modulu (součást každé výpočetní jednotky, viz Obrázek 2.), kde jsou registrovány všechny potřebné dvojice *(symbolické jméno, číslo portu)*. Na základě této informace spolu příslušným DiProNN programem a s informacemi aplikací předávanými jako součást ARTP datagramu řídící modul sestavuje IP hlavičku nového paketu, do kterého budou fragmenty ARTP datagramu (tj. ARTP pakety) zapouzdřeny. Tyto pakety jsou poté přeposlány určenému aktivnímu programu.

Díky využití symbolických jmen pro popis aktivních programů a komunikačních rozhraní mezi nimi nejsou aktivní programy nuceny znát informace jak o celé DiProNN relaci, tak o svém bezprostředním okolí. Jsou tak na sobě zcela nezávislé – musí pouze znát symbolická jména portů, se kterými chtějí komunikovat, a symbolická jména svých portů zaregistrovat u řídícího modulu Výpočetní jednotky, na které běží.

4 Související projekty

Díky svým mnoha možným aplikacím se aktivní sítě staly velmi populární, a tak byly zkoumány řadou vědeckých týmů. Výsledkem jejich práce je mnoho navržených architektur aktivních směrovačů/prvků – v této sekci budou stručně popsány pouze ty, které s naší prací nejvíce souvisí.

C&C Research Laboratories navrhly uzel CLARA (*CLuster-based Active Router Architecture*) [7] – prototyp směrovače v tzv. JOURNEY síti. Architektura CLARy je (prototyp využívá síť Myrinet). CLARA tak svým uživatelům nabízí možnost transkódování proudu video dat dle jejich požadavků. Poskytuje však pouze pevnou funkcionalitu stanovenou administrátorem daného uzlu a nezaručuje zpracování všech zaslaných paketů (toto musí být zajištěno přídavnými technikami implementovanými na koncových stanicích).

LARA (*Lancaster Active Router Architecture*) [8] zahrnuje návrh jak softwarové, tak hardwarové architektury aktivního směrovače. Jeho nástupce, LARA++ (*Lancaster's 2nd-generation Active Router Architecture*) [9], který oproti původní architektuře, která pro svůj běh vyžadovala podporu speciálního hardwaru, klade hlavní důraz na softwarový návrh architektury aktivního směrovače. Tím se LARA++ stal nezávislým na hardware, na kterém běží, a tak dokáže běžet jak na jednoprocesorovém stroji, tak využívat například výpočetní cluster pro distribuované zpracování. Nicméně, ani jedna z těchto architektur svým uživatelům neposkytuje možnost nahrávání libovolných aktivních programů určených pro zpracování jimi zasílaných dat.

5 Závěr a plány do budoucna

V tomto příspěvku jsme navrhli architekturu distribuovaného programovatelného síťového prvku využívajícího principů virtuálních strojů, nazvaného DiProNN.

DiProNN je aktivní prvek sítě umožňující uživateli řízené zpracování jimi zasílaných dat již uvnitř sítě, nikoli až na koncových uzlech. Mezi jeho hlavní přednosti patří zejména možnost jeho jednoduchého modulárního programování popsaného tzv. DiProNN programem, kdy jednotlivé moduly (v daném případě aktivní programy) spolu komunikují s využitím standardních síťových služeb popsaných abstraktním programem (komunikační kanály jsou pro snadnější programování odkazovány pouze s využitím symbolických jmen). Potřebné aktivní programy jsou do DiProNNu nahrávány buď samostatně, nebo spolu s virtuálním

[4] Jednotkou komunikace mezi aplikací a komunikační vrstvou DiProNNu je ARTP datagram, který je díky tomu, že může mít libovolnou velikost, před odesláním do sítě na základě znalosti maximální velikosti odesílaného paketu (*MTU – Maximum Transmission Unit*) fragmentován na menší bloky dat (ARTP pakety).

strojem (a tím i operačním systémem), ve kterém pak uvnitř DiProNNu běží. Díky své architektuře umožňující jeho implementaci na clusteru složeném z několika PC propojených nízkolatenční komunikační sítí DiProNN navíc umožňuje i paralelní běh určených aktivních programů, které tak souběžným zpracováním navyšují objem dat, který je DiProNN schopen zpracovat.

Co se týče našich budoucích záměrů, navrženou architekturu DiProNNu plánujeme implementovat s využitím Xen VMM (*Xen Virtual Machine Monitor*) [10]. Navíc chceme prozkoumat problematiku zajištění kvality služby uvnitř navrženého uzlu a mechanizmy plánování zdrojů DiProNNu tak, aby byly využity co nejefektivněji. Plánování zdrojů DiProNNu chceme zkoumat ze všech tří perspektiv – plánování aktivních programů do vhodných virtuálních strojů (těch, které nemají určený virtuální stroj, ve kterém mají běžet), plánování virtuálních strojů na vhodné Výpočetní jednotky a plánování aktivních programů/virtuálních strojů na vhodné DiProNN uzly (v případě, kdy se na cestě od odesílatele k příjemci dat vyskytuje více DiProNN uzlů schopných podílet se na požadovaném zpracování).

6 Poděkování

Tento projekt je podporován výzkumným záměrem „Integrovaný přístup k výchově studentů DSP v oblasti paralelních a distribuovaných systémů“ (kód projektu 102/05/H050) a výzkumným záměrem „Optická síť národního výzkumu a její nové aplikace“ (MŠM 6383917201) zájmového sdružení CESNET.

7 Odkazy a literatura

[1] Jim E. Smith a Ravi Nair. *Virtual Machines: Versatile Platforms for Systems and Processes.* Elsevier Inc., 2005.

[2] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt a Andrew Warfield. *Live Migration of Virtual Machines.* Ve sborníku 2. ročníku konference ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), strany 273-286, Boston, MA, květen 2005.

[3] Eva Hladká a Zdeněk Salvet. *An Active Network Architecture: Distributed Computer or Transport Medium.* Ve sborníku konference ICN2001: First International Conference Colmar, strany 612-619, Francie, červenec 2001.

[4] Tomáš Rebok. *Active Router Communication Layer.* Technická zpráva zájmového sdružení CESNET, 29 stran, Praha, 2004.

[5] Andrzej Cichocki, Marek Rusinkiewicz a Darrell Woelk. *Workflow and Process Automation: Concepts and Technology.* Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[6] William Thies, Michal Karczmarek a Saman P. Amarasinghe. *StreamIt: A Language for Streaming Applications.* Ve sborníku 11. ročníku konference International Conference on Compiler Construction, strany 179-196, 2002.

[7] Girish Welling, Maximilian Ott a Saurabh Mathur. *A Cluster-Based Active Router Architecture.* Časopis IEEE Micro, IEEE Computer Society Press, svazek 21, číslo 1, 2001, Los Alamitos, CA, USA.

[8] R.Cardoe, Joe Finney, Andrew C. Scottand a Doug Shepherd. *LARA: A Prototype Systém for Supporting High Performance Active Networking.* Ve sborníku konference IWAN’99, strany 117-131, 1999.

[9] Stefan Schmid. *LARA++ Design Specification.* Lancaster University DMRG Internal Report, MPG-00-03, leden 2000.

[10] Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Ian Pratt, AndrewWarfield, Paul Barham a Rolf Neugebauer. *Xen and the Art of Virtualization.* Ve sborníku ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 2003.