Implementace P2P sítě zrcadel v prostředí JXTA

Michal Procházka[1] (xprocha7@fi.muni.cz)
Tomáš Rebok[1,2] (xrebok@fi.muni.cz)
Petr Holub[1,2] (hopet@ics.muni.cz)

Fakulta informatiky[1]
Masarykova univerzita v Brně
Botanická 68a, 602 00 Brno

CESNET, z.s.p.o.[2]
Zikova 4, 160 00 Praha

Modulární uživatelem řízené zrcadlo (reflektor, nebo také aktivní element) se ukázalo jako vhodný nástroj pro síťovou podporu prostředí pro spolupráci s malým až středně velkým počtem uživatelů.

Samostatný reflektor se však vzhledem ke své centralistické koncepci potýká s problémem škálovatelnosti a robustnosti; z tohoto důvodu byl navržen model překryvné sítě zrcadel, který má výše uvedené problémy překonat a navíc přinést možnost heterogenní funkcionality propojených zrcadel. Pro zachování uživatelem řízeného přístupu jsme na organizaci překryvné sítě využili technologii peer-to-peer (P2P), pomocí níž si reflektory organizují a průběžně upravují topologii distribuce dat. Ta obvykle probíhá nezávisle na dané P2P síti, která je sice robustní, ale pro přenos dat silně neefektivní. Jako vhodný prostředek pro implementaci samoorganizující se P2P sítě byla zvolena technologie JXTA, která je platformově nezávislá, pervazivní (proniká různými síťovými prostředími bez ohledu na firewally a NAT), uživatelem řízená a robustní vzhledem k výpadkům sítě.

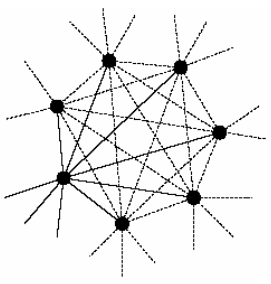
Pro distribuci dat byla navržena a implementována řada modelů s různým poměrem robustnosti a škálovatelnosti – jednoduché úplné grafy, vrstvené úplné grafy a na robustnost optimalizované minimální kostry (využívající Dijkstrův algoritmus). Tím byl vytvořen robustní uživatelem řízený nástroj podporující prostředí pro synchronní komunikaci větších skupin uživatelů.

1. Úvod

Komunikační zrcadlo (neboli reflektor, aktivní element) je vhodným alternativním řešením k neexistující globální podpoře nativního multicastu na dnešních počítačových sítích. Je založeno na jednoduchém principu zrcadlení (reflektování) veškerého příchozího toku dat (obvykle ve formě UDP datagramů) ke všem připojeným klientům s výjimkou toho, který daná data odeslal. Jeho první implementace vytvořená Julianem Highfieldem v roce 1998 byla využívána především v prostředí malých virtuálních konferencí. První zásadní inovací této myšlenky bylo převedení reflektoru do modulární architektury, díky které se stal velmi mocným a lehce rozšiřitelným uživatelsky řízeným nástrojem podporujícím možnosti libovolného zpracování procházejících toků dat.

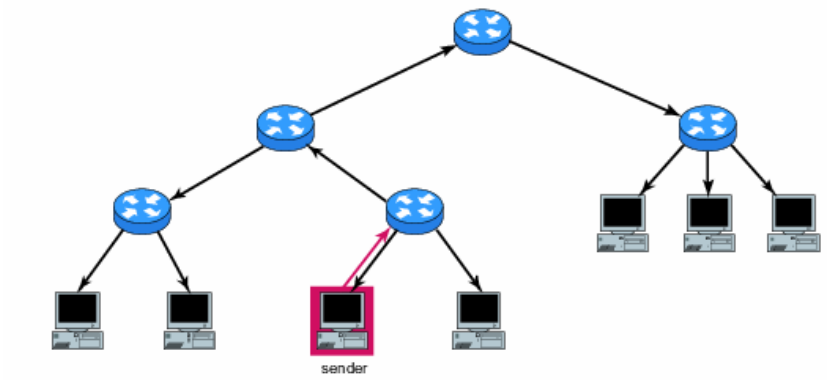
Nynější podoba reflektoru se dá popsat jako programovatelný směrovač na aplikační úrovni s možností přidávání modulů. Jeho modularita dává obrovské možnosti rozšiřitelnosti – moduly mohou zajišťovat jednak zpracování přeposílaných dat, ale také řídit samotný reflektor. Přestože je tento modulární reflektor v porovnání se svou původní verzí výkonnější, díky své centralistické koncepci se stále potýká s problémem škálovatelnosti a robustnosti v případě použití pro podporu spolupráce větších skupin komunikujících uživatelů (limity reflektoru jsou dosahovány především při přenosu multimediálních dat náročných na šířku pásma, jako například přenos digitálního videa). Kolaborativní prostředí je navíc velice náročné na latenci, proto vyvstala nutnost navrhnout systém, který by na stávajících unicastových sítích provoz takto náročných aplikací umožnil.

Ideálním modelem škálovatelné sítě je nativní multicast. Ten však nevytváří robustní síť, neboť jeho stromová struktura přenosu dat v případě výpadku jedné linky znamená, že dojde k odpojení celého podstromu uzlů (data jsou mezi uzly přenášena vždy právě jedinou cestou). Dalším velkým problémem použití nativního multicastu na Internetu je jeho podpora ze strany ISP.



Obrázek 1: Jednoduchá síť reflektorů

Reflektory lze mezi sebou propojovat „tunely“ (viz Obrázek 1). Jednoduchá síť reflektorů je velice robustní a minimalizuje latenci, avšak stále se potýká s omezenou škálovatelností. Možným vylepšením je propojení reflektorů do topologie stromu, čímž lze dosáhnout velice dobré škálovatelnosti (viz Obrázek 2). Na druhou stranu však ztrácíme robustnost celého systému stejně jako v případě nativního multicastu.



Obrázek 2: Reflektory uspořádané do stromu (převzato z [1])

Všechny tyto problémy vedly k návrhu *aktivního elementu* (AE), který je zobecněním původního modulárního reflektoru doplněného o dva moduly – *Network Management module* (NM) a *Network Information module* (NIS). Úkolem NM modulu je vytvoření sítě aktivních elementů spolu s její správou a reorganizací v případě výpadku linky. Naproti tomu NIS modul slouží k více účelům – shromažďuje a publikuje informace o daném AE (např. dostupnou síťovou a procesorovou kapacitu) či o vlastnostech důležitých pro synchronní přenos multimediálních dat (zpoždění linky, RTT, odhadovaná kapacita linky, atp.). Dále pak poskytuje informace o speciálních schopnostech daného AE (např. schopnost překódování multimediálních dat).

Aktivní elementy si tak nad hostující sítí (ethernet, Internet, atd.) vytváří překryvnou síť, která zajišťuje jak robustnost, tak škálovatelnost celého systému, a dynamicky reaguje na změny stavu původní sítě. Pro řídící zprávy používají samoorganizující se principy úspěšně implementované v běžných *peer to peer* (P2P) síťových prostředích (konkrétně pro zjišťování nových AE, zjišťování dostupnosti služeb, správu topologie sítě, atp.). P2P přístup navíc vyhovuje požadavkům robustnosti a uživatelem řízeného přístupu, a jeho nižší efektivita nemá žádný významný vliv na výkon celého systému, neboť je překryvná síť využívána pouze pro přenos řídících dat (vlastní data jsou distribuována po hostující síti). V další části tohoto příspěvku bude popsána technologie JXTA, která byla díky svým vhodným vlastnostem na vytvoření této P2P sítě vybrána. Pro distribuci dat na hostující síti pak byly navrženy tři základní modely, které budou představeny dále.

2. Překryvné sítě, Peer to Peer sítě

Překryvné sítě (*overlay networks*) vytvářejí vrstvu mezi hostující sítí (ve většině případů se jedná o Internet) a aplikacemi. Původní síť se využívá pouze pro přenos paketů – jako spojení mezi uzly překryvné sítě. Informace specifické pro překryvnou síť jsou obsaženy v datové části paketů původní sítě, takže je tato síť zpracovává, aniž by o překryvné síti věděla. Hlavní předností překryvných sítí je možnost provozu v uživatelském režimu, takže lze budovat lehce konfigurovatelné sítě bez nutnosti administrátorských zásahů do jejich fyzické struktury. Uzly jsou v překryvných sítích adresovány pomocí jedinečných identifikátorů, což jim umožňuje identifikovat se v síti vždy stejně (nezávisle na tom, jakou mají v hostující síti adresu). V překryvných sítích lze navíc využít vlastních směrovacích pravidel, čímž se otevírá cesta k vytvoření virtuálního multicastu, který je provozován na unicastových spojích hostující sítě. Návrh překryvné sítě lze koncipovat tak, že se do jisté míry stane odolná vůči výpadkům hostující sítě (udržováním alternativních cest, seznamů okolních uzlů atp.). Překryvné sítě však mají velké problémy s latencí, která je pro multimediální přenosy, konkrétně pro videokonference, kritickým faktorem.

Pro odstranění problémů s latencí využívají aktivní elementy jak hostující síť, tak překryvnou. Překryvná síť je využívána k přenosu servisních a administrativních zpráv mezi jednotlivými aktivními elementy a jejich klienty, protože tyto zprávy nejsou tolik náchylné na latenci, ale zejména vyžadují spolehlivost doručení. Při výpadku překryvné sítě tak není ohrožen samotný přenos multimediálních dat na hostující síti, naopak při výpadku hostující sítě mohou aktivní elementy díky informacím distribuovaným pomocí překryvné sítě obnovit přenos po záložních trasách.

Peer to peer (P2P) sítě jsou jedním z typů překryvných sítí. Jedná se o distribuované prostředí, ve kterém uzly mezi sebou komunikují přímo (bez nutnosti existence centrálního bodu). P2P sítí existuje více typů – od centrálních přes decentralizované až po hybridní (více viz [2]). Mezi stěžejní části P2P sítí patří lokalizace (hledání) určitého prostředku, služby nebo zdroje. Nejpoužívanějšími přístupy pro hledání jsou:

Flooding

Flooding (nebo-li záplava) patří k nejhorším způsobům vyhledávání[1], protože hledající uzel zasílá dotaz všem svým sousedům, kteří jej dále replikují. Pokud dotazovaný uzel požadovaná data má, pak odpovídá přímo hledajícímu uzlu. Každý dotaz je opatřen TTL (*Time to Live* – doba platnosti), po jehož expiraci se dále nepřeposílá (zabraňuje se tak zahlcení sítě již neplatnými dotazy). Do této kategorie patří například sítě Gnutella a Kazaa.

DHT (Distributed Hash Table)

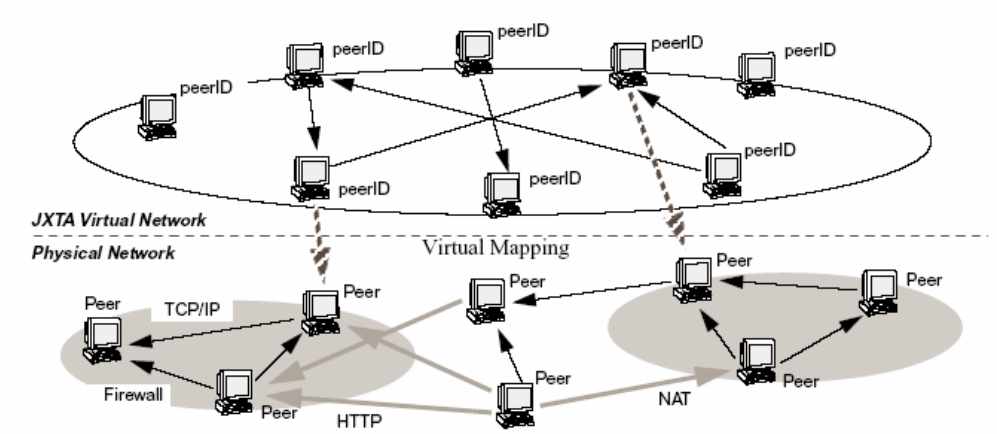
DHT je velice sofistikovaný systém, kdy všechny entity v síti, které je potřeba adresovat, mají svůj jedinečný identifikátor (ID). Každý uzel si udržuje seznam svých sousedních uzlů a jejich ID, přičemž vyhledávání je založeno na porovnávání nejdelšího společného prefixu nebo sufixu. Hledáme-li konkrétní ID, pak je dotaz zaslán na uzel, který má v lokální tabulce nejdelší společný prefix s vyhledávaným dotazem. Pokud takový uzel není, je dotaz zaslán na náhodně vybraný uzel. Takto dotaz dosáhne svého cíle s průměrným počtem *log N* skoků. Dotazy v sítích využívajících DHT tak generují velice malý provoz. Typickými představiteli sítí využívajících tento princip vyhledávání jsou sítě Pastry, Tapestry a Chord.

Loosly-Consistent DHT

Tento typ vyhledávání používá síť JXTA a bude podrobněji popsán v následující kapitole.

3. JXTA

Cílem projektu JXTA je vytvořit uniformní prostředí, ve kterém mohou počítače, služby a aplikace mezi sebou vzájemně komunikovat, aniž by musely řešit problémy s transportem dat (například Firewally, NAT či dynamicky se měnící hostující síť). Zjednodušeně lze říci, že JXTA je sada protokolů, které spolu tvoří virtuální síťové prostředí (viz Obrázek 3) postavené na základech technologie P2P. Jelikož mají protokoly minimální nároky, mohou tuto virtuální síť využívat i jednodušší zařízení jako jsou senzory, PDA apod. JXTA byla navržena především s ohledem na nezávislost na programovacím jazyce (C, Java, Perl5, SmallTalk, Python), systémové platformě (Windows, Linux, Unix, Mac OS, FreeBSD, Solaris), síťových protokolech (TCP/IP, Bluetooth, IrDA, HTTP) a síťových službách (RMI, WSDL). Veškerá komunikace je založena na formátu XML, čímž je do JXTA protokolů umožněno přidávat vlastní informace, aniž by tím byla narušena funkčnost sítě. K další nesporné výhodě patří absence administrativních zásahů jak do operačních systémů, tak do sítě. Celá JXTA totiž běží v uživatelském režimu a pro možnost aktivně komunikovat i za NAT a firewallem využívá HTTP protokol spolu s *relay service*, který bude popsán dále.



Obrázek 3. Virtuální síťové prostředí (převzato z [4])

3.1. Charakteristika

Základní stavební kameny JXTA jsou obdobné jako v ostatních sítích založených na technologii P2P: *ID, Advertisements, Peers a Peer Groups, Services, Pipes* a *Messages.*

ID

Adresování v JXTA je založeno na uniformním logickém adresním modelu. Každému síťovému prostředku (*Peer* – uzel, *Peer Group* – skupina uzlů, *Pipe* – komunikační kanál atd.) je přiřazen jedinečný identifi-

[1] Dle studie Matei Rîpeanu z University of Chicago (viz [3]) dělaly v roce 2000 dotazy sítě Gnutella o velikosti 50 000 uzlů až 1,7 % provozu americké páteřní sítě (tj. téměř 330 TB měsíčně).

kátor (ID). Zjednodušeně řečeno – každá entita, kterou je v síti JXTA potřeba adresovat, musí mít přiřazeno ID. Pro JXTA ID je využito 128bitového UUID (*Universal Unique IDentifier*), který je možno vygenerovat z libovolných dat (IP adresy, sériového čísla zařízení, atp.). Přiřazení jedinečného ID každému uzlu JXTA sítě jej umožňuje adresovat nezávisle na jeho fyzickém umístění.

Advertisemenets

Každá entita je v JXTA síti identifikována pomocí XML dokumentu nazývaného *advertisement* (ADV). Entitou v tomto smyslu může být v zásadě cokoliv – uzel, skupina uzlů, komunikační uzly, služba, binární data, části zdrojového kódu, objektové třídy, atd. Na ADV je nahlíženo jako na objekty, a proto mohou být libovolně přetypovány a rozšiřovány o další data. Každý ADV je spojen s expirační dobou; tím je zajištěno, že neexistující entity přestanou být po určité době označovány za dostupné. Prodleva mezi nedostupností entity a jejím faktickým odstraněním ze sítě je nutnou cenou za to, že ke správě existence prostředků není potřeba žádný centrální registr.

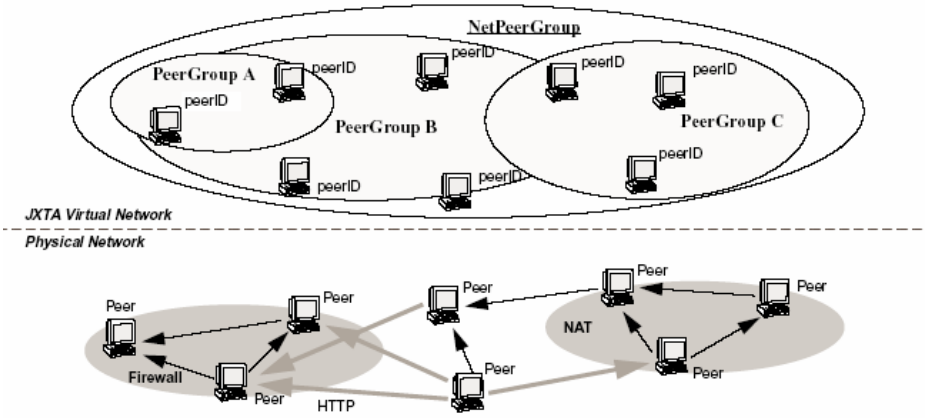
Následující příklad představuje ADV komunikačního kanálu:

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PipeAdvertisement>
<jxta:PipeAdvertisement xmlns:jxta="http://jxta.org">
<Id>urn:jxta:uuid-59616261646162614E504720503250338E3E786229EA460DADC1A176B69B731504</Id>
<Type>JxtaUnicast</Type>
<Name>TestPipe.end1</Name>
</jxta:PipeAdvertisement>
```

Peers a Peer Groups

Každá P2P síť sestává z uzlů. Uzel (*Peer*) je stejně jako každá jiná entita v JXTA síti identifikován pomocí ADV, který obsahuje jméno daného uzlu, jeho popis, jedinečné ID a seznam adres, pod kterými je dostupný na hostující síti.

V klasických sítích (jako například Internet) se počítače seskupují do skupin podle lokality, které jsou dány přidělenou IP adresou. Naproti tomu jsou ve většině P2P sítí všechny uzly umístěny v právě jedné skupině. JXTA zavedla pojem tzv. *Peer Group* – virtuální skupiny uzlů, které jsou v dané skupině soustředěny podle společných zájmů (viz Obrázek 4). Skupin a podskupin lze vytvářet libovolný počet; každý uzel navíc povinně náleží do skupiny s názvem *NetPeerGroup*, která zastupuje kořen stromu všech skupin. Kořenová skupina obsahuje standardní sadu služeb a s nimi spojené protokoly (*discovery*, *resolver*, *pipe*, *peer info* a *rendezvous*). JXTA explicitně neurčuje kdy, kde a z jakého důvodu mají být skupiny vytvářeny, ale pouze definuje jak skupinu vytvořit, publikovat či vyhledávat.



Obrázek 4. Skupiny uzlů – *Peer Groups* (převzato z [4])

Services

Services jsou služby, které mohou využívat uzly uvnitř každé skupiny. Kromě předdefinovaných umožňuje JXTA vytvářet i vlastní služby. Předdefinované služby jsou:

Pipe	– zajišťuje komunikaci mezi uzly; jedná se o abstrakci pro jednosměrné asynchronní zasílání zpráv
Membership	– stará se o připojování a odpojování uzlů ve skupině
Access	– bezpečnostní služba kontrolující oprávnění pro přístup ke službám uvnitř skupiny
Discovery	– služba umožňující uzlům nacházet ostatní uzly, skupiny, komunikační kanály, služby atp.
Resolver	– jelikož se v JXTA síti zdroje hledají pomocí ADV, *resolver* zajišťuje mapování ADV na konkrétní implementaci

Pipes

Pipes (komunikační kanály) jsou virtuální jednosměrné tunely mezi službami a aplikacemi, kterými se posílají data. Každý komunikační kanál má vstupní část (*input pipe*), výstupní část (*output pipe*) a je publikován a identifikován pomocí *pipe advertisement*.

Messages

Messages (zprávy) jsou základním prvkem výměny informací mezi uzly. JXTA pro výměnu dat využívá tzv. *binary wire format*, který v sobě dovoluje přenášet data jakéhokoliv druhu. V síti JXTA se nejčastěji používá přenos textového XML a čistě binárních zpráv.

3.2. Protokoly

Jak již bylo řečeno, JXTA je sada protokolů. Konkrétně jich je právě šest a jsou rozděleny do dvou skupin:

Core Specification Protocols

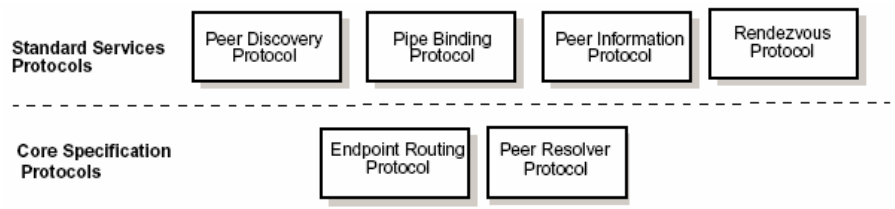
Každý, kdo se chce stát součástí JXTA sítě, musí implementovat následující dva protokoly (uzel se pak nazývá *Minimal Edge Peer*):

Endpoint Routing Protocol – Protokol určený ke hledání cest v síti JXTA.
Peer Revolver Protocol – Protokol zajišťující odesílání obecných dotazů službám a příjem jejich odpovědí.

Standard Service Protocols

Sada protokolů, které je doporučeno implementovat do aplikací využívajících síť JXTA kvůli lepší funkcionalitě a interoperabilitě s ostatními aplikacemi.

Rendezvous Protocol – Jeho úkolem je propagace dotazů a cachování SRDI indexů (viz dále) v rámci skupiny uzlů.
Peer Discovery Protocol – Díky tomuto protokolu mohou uzly publikovat vlastní ADV, a zároveň ostatní ADV vyhledávat.
Peer Information Protocol – Tento protokol umožňuje uzlům získávat informace o ostatních uzlech (stavové informace, vytížení, doba provozu atp.).
Pipe Binding Protocol – Pomocí tohoto protokolu mohou uzly vytvářet virtuální komunikační kanály mezi jedním nebo více uzly.



Obrázek 5: Protokoly JXTA (převzato z [4])

3.3. Bezpečnost

Potřeba autentizace a autorizace v *ad-hoc* sítích vede k nutnosti použití *role-base trust* modelu, ve kterém se jeden uzel autentizuje vůči druhému za pomoci ověření u třetí strany. XML zprávy v JXTA lze rozšířit o bezpečnostní metada jako jsou digitální certifikáty, hashe, veřejné klíče atp. Zprávy díky tomu mohou být podepsány pro ověření pravosti a nepopíratelnosti, případně i šifrovány. JXTA podporuje ověřené bezpečnostní mechanismy jako jsou *Secure Socket Layer* (SSL) a *Transport Layer Security* (TLS).

3.4. Pervazivita

Pervazivita sítě JXTA (tj. schopnost sítě umožnit komunikaci s uzly, které jsou za NAT či firewallem), je dosažena pomocí dvou technologií – protokolu HTTP a *relay service*. HTTP umožňuje komunikaci z uzavřených systémů směrem ven, jelikož se předpokládá, že je ve většině sítí povolen. Největším problémem je nedosažitelnost uzlů umístěných za NAT či firewallem – JXTA tuto problematiku řeší elegantně pomocí *relay service*.

Relay service je služba, kterou lze spustit pouze na uzlech dostupných v rámci hostující sítě (v případě Internetu mají veřejnou IP). K nim se pak přihlásí uzly, které jsou z hostující sítě nedosažitelné. *Relay service*

funguje jako schránka – všechny zprávy určené pro přihlášené uzly jsou u *relay* serveru ukládány a cílový uzel se na ně periodicky dotazuje.

3.5. Vyhledávání

Na vyhledávání v JXTA se podílí služby *SRDI index* a uzly fungující jako *rendezvous server* (RDV) spolu s algoritmy *Limited-range walker* a *Distributed hash table* (DHT).

RDV uzly si v rámci jedné skupiny udržují seznam několika dalších RDV uzlů a indexy SRDI. Tento seznam se v kontextu JXTA nazývá *Rendezvous Peer View* (RPV). Dostupnost uzlů uvedených v RPV se periodicky kontroluje a ve větších časových intervalech se náhodně vyměňují samotné RPV. Jelikož je RPV distribuováno přes všechny RDV uzly, výpadek jednoho uzlu způsobí dočasnou nekonzistentnost celého RPV (odtud název *loosly-consistent*). V případě dramatické změny RPV se stane SRDI index na daném RDV nedostupným; tehdy je použit algoritmus *Limited-range walker*, který rozesílá dotaz na všechny sousední RDV (algoritmus je podobný vyhledávací metodě *flooding* popsané dříve).

Dojde-li k publikování ADV do sítě, nepřenáší se celý XML dokument. Služba SRDI (*Shared Ressource Distributed Index*) zajistí indexaci klíčů z ADV (jako je např. ID nebo jméno) a zašle tyto indexy RDV uzlu. Zasíláním pouhého indexu se minimalizuje množství dat, které RDV uzly musí uchovávat. SRDI indexy jsou součástí RPV, a proto jsou distribuovány na další RDV.

Vyhledávání v síti JXTA pak probíhá tak, že uzel zašle dotaz na připojený RDV a zároveň, pokud je to možné, pomocí multicastu a broadcastu rozešle dotaz uzlům na místní segment sítě. Podrobnější informace o vyhledávání v síti JXTA lze nalézt v [5].

3.6. Samoorganizace, robustnost a škálovatelnost

Robustnost sítě JXTA je zajištěna existencí RPV. Každý uzel si udržuje seznam několika záložních RDV uzlů pro případ výpadku aktuálně používaného. Proto i výpadek velkého množství uzlů nemusí znamenat výpadek celé sítě.

Dobré úrovně škálovatelnosti dosahuje JXTA díky segmentaci celé své sítě na skupiny uzlů, čímž jsou dotazy propagovány pouze v rámci této skupiny (nejsou posílány po celé síti). Dále pak JXTA minimalizuje počet potřebných skoků a tím i přenášených zpráv k nalezení cíle pomocí *loosly-consistent DHT*.

Uzly se samoorganizují díky periodickému dotazování na dostupnost, vzájemnému vyměňování informací o svém okolí a úpravou RPV.

4. Implementace

K implementaci informačního (NIS) a management (MN) modulu jsme zvolili implementaci JXTA v jazyce C (JXTA-C) především z důvodu, že samotný reflektor je v tomto programovacím jazyce také kompletně naprogramován. Problémem JXTA-C je zatím neúplná podpora všech funkcí „hlavní“ JXTA, která je vyvíjena v jazyce Java.

Návrh NIS a MN modulů je koncipován modulárně, přičemž sestávají z následujících částí:

Modul generování ID – Tento modul je využíván při vytváření identity uzlu. Standardně se využívá integrovaného generování ID v JXTA, ale díky tomuto modulu lze generované ID ovlivnit. Do budoucna se počítá s generováním ID, které v sobě bude zahrnovat informace o umístění uzlu (pouze pro uzly se statickým umístěním), na základě kterých bude možno upravovat směrování a informace o topologii sítě.

Modul zasílání zpráv – Dovoluje uzlu zasílat zprávy jak konkrétním uzlům, tak celým skupinám uzlů. Zprávy jsou rozděleny do dvou skupin – řídící a informační. Řídící zprávy jsou chápány jako žádost o změnu stavu adresovaného uzlu, zatímco informační zprávy znamenají žádost o poskytnutí informací o aktuálním stavu uzlu. Tento modul se také stará o zpracování příchozích zpráv (funguje pouze jako analyzátor zpráv, pro její zpracování volá příslušné externí funkce).

Modul správy sítě – Shromažďuje informace o připojených klientech a sousedních reflektorech. Periodicky vyhodnocuje dostupnost a latenci připojených uzlů a aktivně jedná v případě výskytu jakéhokoli problému na síti. Tento modul částečně zasahuje do implementace JXTA, kde upravuje chování RPV a seznamu záložních RDV uzlů.

Modul informační – Pro vizualizaci topologie sítě je použit nástroj *jxtanetmap*. Jelikož jeho klientská i serverová část je napsána v jazyce Java, byla implementace jeho protokolu pře-

psána do jazyka C a zabudována do informačního modulu. Dále byl původní protokol upraven pro potřeby reflektoru – byly přidány informace o RTT pro vyhodnocování optimálního rozložení uzlů v síti, informace o dostupné šířce pásma, aktuálním vytížení procesoru, počtu připojených klientů a počtu probíhajících přenosů.

Modul správy skupin – Spravuje skupiny uzlů (*Peer Groups*). Skupiny jsou definovány dynamicky podle potřeb služeb podporovaných reflektorem (např. skupina reflektorů s podporou překódovaní video streamů či skupina reflektorů participujících na konkrétní videokonferenci).

Součástí tvorby modulů je také vytvoření klientské části, která bude uživatelům reflektoru poskytovat informace o dostupných zdrojích v síti, o stavu reflektoru, a zároveň bude umožňovat své řízení reflektorem a bude řešit obnovení spojení při problémech sítě či reflektoru. Klientská aplikace bude postavena na základech informačního a management modulu. Bude se však jednat o zcela samostatnou aplikaci, která bude navíc schopna spouštět externí programy a předávat jim parametry. Uživatel i aplikace tak budou schopni zjišťovat služby dostupné v síti reflektorů, monitorovat jejich stav a během připojení k reflektoru sledovat jeho stavové informace.

5. Modely distribuce dat

Pro distribuci dat mezi aktivními elementy je zapotřebí takové decentralizované řešení, ve kterém je pečlivě vyvážena robustnost a výkon celého systému. Jelikož není výpočetní náročnost pro malé skupiny (uživatelů, linek, aktivních elementů, atp.) kritická, byly navrženy modely, které jsou redundantní a velmi důmyslné. Pro rozlehlé sítě typu Internet je však zapotřebí daleko jednodušších algoritmů.

***Poznámka**: Na počítačovou síť je při popisu následujících modelů nahlíženo jako na graf, jehož uzly reprezentují aktivní elementy a hrany představují fyzické spoje mezi nimi. Detailnější informace k představeným modelům lze nalézt v [6, 7].*

5.1. Multicast schemes

Ideálním a nejefektivnějším modelem distribuce dat více klientům je použití principů multicastu. Pro uživatele je však velmi obtížné rozmístit aktivní elementy do sítě tak, aby žádná data nešla po téže lince více jak jednou (výjimkou může být implementace AE jako aktivního směrovače, která však neumožňuje uživatelem řízený přístup, který požadujeme). Multicast se tak stává horním odhadem dosažitelné efektivity distribuce dat více klientům.

Pro použití schématu distribuce dat multicastu na unicastových sítích je zapotřebí vybudovat tzv. *multicast tree* – strom, po kterém se data na dané síti posílají podobně, jak je tomu u multicastu. Jednou z možností, jak takový strom vybudovat, je výpočet minimální kostry sítě aktivních elementů[2] (ideálně pomocí distribuovaných algoritmů), pomocí které jsou pak data rozesílána. Tento přístup se však podobně jako multicast potýká s problémem robustnosti – při výpadku linky, která leží na minimální kostře daného grafu, je odpojena celá část grafu ležící za vypadlou linkou a pro další pokračování v přenosu je zapotřebí zjistit novou minimální kostru. Pro urychlení konvergence sítě po výpadku je možno předem spočítat a po celou dobu udržovat více vzájemně disjunktních minimálních koster. V případě výpadku pak není potřeba novou kostru počítat, ale uzly se mezi sebou pouze dohodnou na přechodu na některou ze záložních koster (za vypadlou kostru je poté na pozadí přenosu vypočtena náhrada). Výrazného zvýšení robustnosti lze v tomto modelu dosáhnout používáním dvou či více minimálních koster současně, na které jsou data duplikována[3] (při výpadku používané linky se tak žádná data neztratí). Po výpadku je opět zapotřebí zvolit (či vypočítat) za rozpadlou kostru náhradu.

5.2. 2D Full-Mesh

Model předpokládající kompletní graf, ve kterém každý aktivní element přímo komunikuje se všemi zbývajícími. Jedná se o nejjednodušší model s velkou redundancí dat, a proto slouží co se týče škálovatelnosti jako ukázka nejhoršího principu. Při výpadku linky na síti mezi aktivními elementy jsou klienti požádáni o přesun na alternativní AE. V případě výpadku celého AE se klient přesouvá zcela samostatně.

Přestože se tento model zdá jako zcela triviální a nezajímavý, má dvě zásadní výhody. Je totiž velice robustní (výpadek uzlu ovlivní pouze klienty k tomuto uzlu připojené) a navíc přináší minimální latenci, jelikož

[2] Minimální může být podle RTT, jednosměrného zpoždění, počtu skoků, atd.
[3] Duplicitní data jsou pak koncovými aktivními elementy či samotnými aplikacemi zahazována.

data tečou maximálně přes dva AE). Problémy škálovatelnosti se zachováním těchto dvou vítaných vlastností řeší třetí model distribuce dat nazývaný *3D Layered-Mesh*.

5.3. 3D Layered-Mesh

3D Layered-Mesh model na síti vytváří několik vrstev, v nichž jsou distribuována data od právě jednoho AE. Každý klient je připojen do právě jedné vrstvy pro příjem i odesílání a do všech ostatních vrstev pro příjem. Pro distribuci dat v rámci jedné vrstvy je pak využit *2D Full-Mesh* model.

Jelikož si každá vrstva monitoruje svou konektivitu, je informace o problémech v případě potřeby broadcastována všem jejím klientům. Klienti připojení do této vrstvy pro příjem i odesílání jsou požádáni o přechod na jinou náhodně zvolenou vrstvu; klienti připojení pouze pro příjem se z této vrstvy jednoduše odpojí.

Hlavní problém tohoto modelu spočívá v kvadratickém navýšení počtu používaných aktivních elementů. Zřejmě se však jedná o poslední model, který nepřidává žádné mezilehlé uzly[4] a tím minimalizuje latenci.

6. Závěr

Distribuce velkého objemu dat za pomocí aplikačního multicastu realizovaného pomocí sítě reflektorů, které se samoorganizují díky P2P technologii, se v dnešní době zdá být velice nadějným řešením. Spojuje se zde škálovatelnost aplikačního multicastu a robustnost P2P sítě, přičemž tento přístup navíc přináší možnosti heterogenní funkcionality propojených reflektorů.

JXTA aspiruje na vedoucí technologii v oblasti *ad-hoc* P2P sítí, a to především díky velice promyšlenému návrhu a silné podpoře komunity, komerčních a akademických subjektů. Poskytuje ucelenou sadu funkcí potřebných pro budování P2P sítí, kterou lze libovolně rozšiřovat a tím vytvářet širokou škálu aplikací. Pro organizaci sítě reflektorů je využívána většina funkcionality JXTA, která byla doplněna o námi potřebné funkce.

Naše další práce bude směřovat ke zdokonalení infomačního a řídícího (management) modulu tak, aby bylo dosaženo co nejefektivnějšího využití kapacit přenosových linek a reflektorů.

7. Poděkování

Tento výzkum je podporován výzkumným záměrem „Optická síť národního výzkumu a její nové aplikace“ (MŠM 6383917201).

8. Reference

[1] E. Hladká a J. Denemark: „On-demand secure collaborative support“, *ITIM 2003 (Italian Association of Temedicine and Medical Informatics)*, Řím, Itálie, 2003.

[2] *http://www.cs.cf.ac.uk/user/I.J.Taylor/DistributedSystems/PPTSlides/PeerOrganization.ppt (duben 2005)* – Organizace uzlů

[3] *http://www.internet2.edu/presentations/20020130-P2P-Ripenau.ppt (duben 2005)* – Studie sítě Gnutella

[4] *http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf (duben 2005)* – Projekt JXTA 2.0

[5] *http://www.jxta.org/docs/jxta-dht.pdf (duben 2005)* – Loosely consistend distributed hash table

[6] P. Holub, E. Hladká, L. Matyska: „Scalability and Robustness of Virtual Multicast for Synchronous Multimedia Distribution“, příspěvek přijatý na konferenci ICN'05

[7] P. Holub: „Notes on Scalable Models for Synchronous Multimedia Distribution“, CESNET technical report number 10/2004

[4] Lepší škálovatelnosti tohoto modelu lze dosáhnout využitím mezilehlých uzlů, čímž je však navyšována latence. Bližší podrobnosti lze nalézt v [7].