Školní rok 2011/2012 (verze 2012-02-19)

IPK

Studijní manuál

iBooks Author

Informace o kurzu

Cíl

Tento návod má za cíl pomoci studentům předmětu IPK při jeho úspěšném absolvování. V rámci textu jsou uvedeny základní podmínky pro zvládnutí kurzu a informace ke každé kapitole, které obsahují:

- Motivaci kapitoly
- Přehled látky, kterou je potřeba zvládnout pro minimální úspěšné hodnocení a oblasti pro další studium
- Komentovaný přehled studijní literatury
- Otázky pro kontrolu základního porozumění oblasti
- Příklady a otázky pro hlubší porozumění oblasti

Přednášky

Cílem přednášek je představení tématu takovým způsobem, který by zjednodušoval samotné studium. Přednáška toto studium nemůže nahradit. Prezentované informace nemohou být úplné a není možné považovat informace prezentované v rámci přednášky za dostatečné pro absolvování předmětu. Vymezení minimálních znalostí pro absolvování předmětu je uvedno v tomto dokumentu. Přednášky se týkají vybrané problematiky z každé oblasti na základě volby přednášejícího. Přestože snahou je zahrnout do obsahu přednášek nejdůležitější oblasti, je možné, že některé se do výběru nedostanou, ať již z časového nebo jiného důvodu. Některé oblasti mohou být také pro detailnější rozbor na přednášce nevhodné, například přehled Ehternet technologií.

Studijní literatura

Vzhledem k tomu, že ke kurzu nejsou k dispozici studijní opory, je studium doporučené literatury nezbytné. Pro každou oblast je literatura rozdělena na základní, doplňkovou a rozšířenou. Studium základní literatury je nezbytné pro zvládnutí učiva na minimální úrovni. Toto například znamená, že 75% otázek podle kterých je student v rámci kurzu hodnocen, je koncipováno tak, aby pro správnou odpoveď stačila znalost získaná ze základní literatury. Doplňková literatura rozšiřuje znalosti ve vybraných směrech. Její studium může přinést nové znalosti ve specifických oblastech a pomoci lépe pochopit vybrané technologie, metody či algoritmy. Rozšířená literatura je pak určena pro studium komplexnějších oblastí, které navazují tématicky na základní znalosti, ale svým charakterem přestavují náročnější oblast na pochopení. Pro získání excelentního hodnocení v předmětu jsou vyžadovány znalosti, které lze získat studiem těchto materiálů. Studijní literatura je, až na vyjímky, v anglickém jazyce. Vzhledem k tomu, že studium IT implicitně předpokládá alespoň pasivní znalost angličtiny, není uvažováno o překládání či dovysvětlování anglického textu ve studijních materiálech. Jelikož se vesměs jedná o knihy, které jsou určeny pro bakalařské studium na severoamerikcýh univerzitách, je

jejich forma a použité jazykové prostředky akceptovatelné pro bakalařské studium na FIT VUT v Brně. V případě nejasností či neporozumění při studiu jsou studentům k dispozici konzultační hodiny a on-line diskuze k předmětu.

V rámci předmětu jsou použity následující hlavní studijní materiály:

- **Kurose & Ross: COMPUTER NETWORKING: A Top-Down Approach, 2.vydání .**
  Dostupné v knihovně FIT, celkem 20ks.
- **Tanenbaum: Computer Networks, 4.vydání.**
  Dostupné ve výběru SAFARI Books
  (http://proquestcombo.safaribooksonline.com/0-13-066102-3)
- **Rita Pužmanová: Routing and Switching: Tima of convergence? Addison-Wesley, 2002.**
  Obsahuje přehled sítí a zejména pak oblasti mšerování, přepínání a kvality služeb.
- **Buford, Yu & Lua: P2P - Networking and Applications.**
  Vybrané pasáže budou poskytnuty.
- **Stallings: High-Speed Networks and Internets.**
  Vybrané pasáže budou poskytnuty.

Jako doplňkové materiály je možné použít také:

- **Rita Pužmanová: Moderní komunikační sítě od A do Z, 2.vydání, Computer Press, 2006.**
- **Kabelová A., Dostálek L.: Velký průvodce protokoly TCP/IP a systémem DNS (5. vydání). Computer Press, 2008.**

Projekty

Vypracování projektů je předpoklad pro získání praktických dovedností. Projekty mají různou složitost a jsou zaměřeny na programování síťové aplikace typu klient-server a implementaci mechanismů spolehlivé komunikace. Tímto lze získat praktické dovednosti s programováním siťových aplikací a také porozumět lépe principům protokolu TCP a spolehlivé komunikace obecně. Přestože vypracování projektů není povinné, je pro splnění podmínek zápočtu nutné některé projekty řešit. Témata projektů jsou:

1. Základní síťová aplikace
2. Pokročilá síťová aplikace
3. Implementace spolehlivé komunikace

Laboratorní cvičení

Laboratorní cvičení slouží pro získání praktických znalostí při základní konfiguraci síťových služeb. Jejich absolvováním lze získat představu o praktickcýh důsledcích a využití znalostí zís-

kaných studiem základní literatury. Témata laboratorních cvičení jsou:

1. Konfigurace uzlů v sítích, konfigurace směrování
2. Konfigurace bezdrátových sítí

Hodnocení

Hodnocení studenta v předmetu je rozděleno na hodnocení dovedností získaných a demonstrovaných v rámci řešení projektu a hodnocení jeho teoretických znalostí při závěrečné zkoušce.

Rozložení bodového hodnocení je následující:

- Projekt 1 - 4 bodů
- Projekt 2 - 6 bodů
- Projekt 3 - 10 bodů
- Laboratorní cvičení s úlohou - 2 x 3 = 6 bodů
- Průběžné testy - 3 x 3 = 9 bodů
- Závěrečná písemná zkouška - 65 bodů

Pro udělení zápočtu je potřeba 15 bodů za hodnocení, které je možné získat v průběhu semestru.

Průběžné testy budou psány na přednášce po ukončení výkladu. Jedná se o sadu testových otázek (cca 10-15).

Pro semestrální zkoušku je stanoveno minimum 25 bodů. Semestrální zkouška je mix testových otázek, odpovědních otázek a početních příkladů.

Přednášející

Jaroslav Ráb, rabj@fit.vutbr.cz

Ondřej Ryšavý, rysavy@fit.vutbr.cz

Konzultace on-line prostřednictvím IS, po skončení přednášky (je-li dostatek času) a v konzultačních hodinách.

Plán kurzu

Kurz obsahuje 13 témat rozvržených do 13 přednášek. Testy zahrnují témata předcházejících přednášek. Projekty se odevzdávají do konce uvedeného týdne. Na laboratoře se přihlašuje v IS.

Týden	Téma	Aktivity
6.2. + 9.2.	Počítačové sítě a Internet	
13.2. + 16.2.	Aplikační vrstva	Zadání projektů
20.2. + 23.2.	Transportní vrstva, Programování BSD soketů	
27.2 + 1.3.	Spolehlivý přenos	Test 1
5.3. + 8.3.	IPv4 a IPv6	
12.3. + 15.3.	Směrování I	Odevzdání P1
19.3. + 22.3.	Směrování II	Test 2,Laboratoř 1
26.3. + 29.3.	LAN Technologie	Laboratoř. 1
2.4. + 5.4.	Multicast	Odevz.P2, Laboratoř 2
9.4. + 12.4.	WLAN, WAN technologie	Laboratoř 2
16.4. + 19.4.	Bezpečnost	Test 3
23.4. + 26.4.	Protokoly pro QoS	Odevzdání P3
30.4. + 3.5.	Peer-to-peer sítě	

Projekty

Vypracování a odevzdání

Projekty je nutné vypracovat podle pokynů. V případě nedodržení pokynů riskujete, že Váš projekt nebude možné hodnotit. Odevzdání projektů je jejich nahráním do IS v daném termínu. Projekty odevzdané po termínu nebudou hodnoceny. Taktéž projekty nepřeložitelné nebudou hodnoceny. Je nutné si po odevzdání vyzkoušet, že odevzdávaný archív obsahuje veškeré nezbytné soubory potřebné pro odevzdání. **Nenechávejte odevzdání projektů na poslední chvíli.**

Pro každý projekt je vyžadován funkční Makefile. Jak vytvořit správně makefile si můžete přečíst například zde: http://www.khmere.com/freebsd_book/html/ch01.html

Zkontrolujte si, že jste odevzdali spravné a všechny soubory do IS, nejlépa tak, že si je z IS nahrajete do nového adresáře a zkusíte projekt přeložit.

Hodnocení

Projekty jsou hodnoceny na základě několika kritérií:

- funkčnost projektu
- splnění zadání
- implementace, srozumitelnost zdrojových kódů
- požadovaná dokumentace

Bodové hodnocení jednak vyjadřuje míru splnění požadavků uvedených v zadání, ale také zahrnuje kvalitu projektu.

Projekty jsou individuální. V případě zjištění podvodného jednání při vypracování projektu bude postupováno v souladu s disciplinárním řádem FIT VUT v Brně.

Celkové bodové hodnocení jednotlivých projektů:

Projekt	Hodnocení
Projekt 1	0 - 4 bodů
Projekt 2	0 - 6 bodů
Projekt 3	0 - 10 bodů
Celkem	0 - 20 bodů

Projekt 1: Webový klient

Vytvořte program (klient) s využitím rozhraní schránek (BSD sockets API), který implementuje získání informací zadaného objektu pomocí URL z WWW serveru s využitím HTTP 1.1 protokolu. Požadované informace budou vytištěny na standardní výstup (stdout).

Program vytvořte v jazyce C/C++, který je přeložitelný na studentském unixovém serveru eva.fit.vutbr.cz (FreeBSD) včetně funkčního Makefile souboru (program přeložitelný po zadání příkazu make). Program využívá spojovanou službu (protokol TCP). Jméno přeloženého programu klienta bude **webinfo**.

Program předpokládá jeden povinný parametr a to URL identifikující objekt, o kterém budou vypsány požadované informace. Pro zjištění vlastností objektu použijte metodu HEAD. Oznámení o chybách (stavové kódy 4xx, 5xx), které mohou nastat, budou vytištěny na standardní chybový výstup (stderr). Ze stavových kódů 3xx implementujte minimálně podporu pro kódy 301 a 302 (redirect), při kterých klient zopakuje dotaz na získání informací o požadovaném objektu pro hodnotu hlavičky Location. Pokud je program spuštěn bez modifikačního parametru, program vypíše celou hlavičku odpovědi včetně stavového řádku na standardní výstup. Pokud bude použit jeden nebo více modifikačních parametrů programu, program vypíše pouze požadované informace v pořadí, ve kterém byly uvedeny. V programu můžete použít pouze standardní C/C++ knihovny.

Synopsis: webinfo [-l] [-s] [-m] [-t] URL

Modifikační parametry:

Parametr	Význam
-l	velikost objektu
-s	identifikace serveru
-m	informace o poslední modifikaci objektu
-t	typ obsahu objektu

Pokud není požadovaná položka hlavičky v odpovědi dostupná, vypište hodnotu dané položky jako N/A .

Způsob odevzdání

Vypracovaný projekt (vas_login.tar.gz) odevzdejte elektronicky do WISu. Archiv vytvořte programy tar a gzip. Odevzdaný projekt musí obsahovat:

- soubor se zdrojovým kódem a komentářem
- Makefile pro standradní make na systému FreeBSD
- další potřebné soubory pro funkci programu (knihovny, ...)

Program se musí jmenovat webinfo.

Příklad použití

Vytištění celé hlavičky odpovědi na metodu HEAD pro specifikovaný objekt:

```
webinfo http://www.fit.vutbr.cz/images/logo_fit.png

HTTP/1.1 200 OK
Date: Wed, 09 Feb 2011 14:00:50 GMT
Server: Apache/1.3.42 Ben-SSL/1.59 (Unix)
Cache-Control: max-age=604800
Expires: Wed, 16 Feb 2011 14:00:50 GMT
Last-Modified: Mon, 04 Oct 2010 09:15:23 GMT
ETag: "7a1f9d-1aa3-4ca99b2b"
Accept-Ranges: bytes
Content-Length: 6819
Content-Type: image/png
```

Získání informací o datu poslední modifikace, typu a velikosti objektu pro specifikovaný objekt:

```
webinfo –m –t –l http://www.fit.vutbr.cz/images/logo_fit.png

Last-Modified: Mon, 04 Oct 2010 09:15:23 GMT
Content-Type: image/png
Content-Length: 6819
```

Získání informací o velikosti a datu poslední modifikace objektu pro specifikovaný objekt:

```
webinfo –l –m http://www.seznam.cz/st/img/logo-2.gif

Content-Length 2632
Last-Modified: Tue, 08 Feb 2011 12:34:53 GMT
```

Získání informací o velikosti a datu poslední modifikace objektu pro neexistující objekt (výpis je na stderr):

```
webinfo –l –m http://www.fit.vutbr.cz/img/unknown.gif

Chyba: 404 Not Found
```

Projekt 2: Klient/server

Vytvořte program pro klienta a server, s využitím rozhraní schránek (sockets API), který implementuje požadovanou službu dle výběru varianty projektu.

- navrhněte aplikační protokol pro komunikaci mezi klientem a serverem
- vytvořte oba programy v jazyce **C/C++** přeložitelné na studentském unixovém serveru **eva.fit.vutbr.cz** (**FreeBSD**) včetně funkčního **Makefile** souboru (oba programy přeložitelné po zadání příkazu **make**)
- vytvořte dokumentaci popisující aplikační protokol (max. 1 strana A4, formát pdf)

Programy využívají spojovanou službu (protokol TCP). Server musí být **konkurentní**, tzn. bude požadována současná obsluha více klientů. Jméno programu pro server po přeložení bude **server** a pro klienta **client**. Server předpokládá jeden povinný parametr **-p** následovaný číslem portu, na kterém bude očekávat spojení od klienta (příklad spuštění serveru: **server -p 10000 &**). Klient předpokládá dva povinné parametry, jméno serveru a číslo portu ve formátu jmeno_serveru:port a další parametry dle konkrétního zadání. Jméno serveru může být zadáno doménovým jménem nebo IPv4 adresou. Protokol síťové vrstvy použijte IPv4. Oznámení o chybách, které mohou nastat, bude vytištěno na standardní chybový výstup (stderr). Výpis názvu souborů v adresáři bude proveden na standardní výstup (stdout).

Způsob odevzdání

Vypracovaný projekt (vas_login.tar.gz) odevzdejte elektronicky přes IS. Archiv vytvořte programy tar a gzip.

Odevzdaný projekt musí obsahovat

1. popis použitého aplikačního protokolu (formát pdf)
2. soubory se zdrojovým kódem a komentářem
3. Makefile
4. další potřebné soubory pro funkci programu nebo ukázkové soubory pro demonstraci činnosti

Varianta 1: Jednoduchá pošta

Jedná se o vytvoření jednoduché poštovní služby, kdy server lokálně ukládá poštu pro jednotlivé uživatele. Uživatel je identifikován pomocí login (dle UNIX standardu). V systému se přenáší jednoduché textové (pouze ASCII) zprávy. Použití klienta:

Synopsis: client HOST:PORT [-r UZIVATEL] [-s PRIJEMCE ZPRAVA] [-d UZIVATEL CISLO]

Modifikační parametry:

Parametr	Význam
-r	čtení zpráv
-s	posílání zpráv
-d	odstranění zprávy

Příklad použití:

Při odeslání zprávy je nutné uvést příjemce zprávy a samotnou zprávu. Jestliže daný příjemce neexistuje bude vytvořen.

```
client localhost:10000 –s xnovak00 “Toto je zprava”

Ok.
```

Čtení zpráv pro uživatele vypíše všechny uložené zprávy:

```
client localhost:10000 –r xnovak00

Toto je prvni zprava.
Toto je druha zprava.
Toto je treti zprava.
```

Jestliže uživatel daného jména nemá uloženu žádnou zprávu, pak výstupem nebude nic. V případě, že daný uživatel neexistuje, pak výstupem také nebude nic.

Odstranění zprávy je dle jejího indexu v rámci kolekce zpráv uživatele.

```
client localhost:10000 –d xnovak00 2

Ok.
```

V případě, že je index mimo rozsah, pak výstupem je chyba:

```
client localhost:10000 –d xnovak00 6

Err.
```

Hlášení o chybě (Err) nebo úspěšném provedení (Ok) se vypisují na STDERR. Data (uložené zprávy) se vypisují na STDOUT.

Varianta 2: Překlad doménových jmén

Vytvořte klient/server systém pro překlad doménových jmen na IP adresy. Server bude poskytovat službu překlad DNS jmen dle použitých voleb. Uvažujte IPv4 adresy a IPv6 adresy. Klient se dotazuje s uvedením doménového jména. Server používá lokální resolver pro překlad doménovách jmen pomocí API knihovny BSD sockets.

Pro otestování plné funkcionality je nutné dotazovat se na DNS, ke kterému je k dispozici AAAA záznam. Seznam takových domén je k nalezení například na: http://sixy.ch/

Na serveru je možné zkusit také, například:

```
kazi:~> nslookup -type=AAAA  nemesis.websitelive.net
Server:             ::1
Address:  ::1#53
Non-authoritative answer:
nemesis.websitelive.net   has AAAA address
2a00:85c0:1::241:21
...
```

Synopsis: client HOST:PORT [-4] [-6] DOMENJMENO

Kde možné parametry jsou:

Parametr	Význam
-4	vrací IPv4 adresu
-6	vrací IPv6 adresu

Příklad použití:

Pro dotaz je nutné specifikovat, které informace o doménovém jméně chceme zobrazit:

```
client localhost:10000 -4 www.fit.vutbr.cz

147.229.9.23
```

nebo:

```
client localhost:10000 -6 www.fit.vutbr.cz

2001:67c:1220:809::93e5:917
```

popřípadě, lze zobrazit obě informace (dle pořadí uvedení parametrů):

```
client localhost:10000 -4 -6 www.fit.vutbr.cz

147.229.9.23
2001:67c:1220:809::93e5:917
```

```
client localhost:10000 -6 -4 www.fit.vutbr.cz

2001:67c:1220:809::93e5:917
147.229.9.23
```

Odpovědi budou vypisovány na STDOUT. V případě nenalezení požadované odpovědi bude na STDERR vypsána chyba a výstup STDOUT bude prázdný:

```
client localhost:10000 -6 -4 www.neexistujici.domena.cz

Err6: Nenalezeno.
Err4: Nenalezeno.
```

Projekt 3: RDT

Implementujte protokol spolehlivého zřetězeného přenosu dat RDT dle přednášek. Projekt bude vypracován v jazyce C nebo C++, přeložitelný a spustitelný v prostředí FreeBSD na serveru eva.fit.vutbr.cz. Pro implementace využijte pouze knihoven a prostředků dostupných na tomto systému.

Protokol bude pro zajištění komunikaci používat služeb UDP a bude ve vlastní režii realizovat mechanismy pro zabezpečení spolehlivého přenosu a řízení toku dat.

Protokol bude textový, kdy pro posílané zprávy bude použito XML popisu. Protokol bude muset přenášet i binární data. K tomu bude nutné tyto binární data kódovat pomocí BASE64.

Každý segment bude mít následující formát:

```
<rdt-segment id="xnovak01">
 <header>

.. hlavička segmentu...

 </header>
 <data>

.. data v BASE64 ..

 </data>
</rdt-segment>
```

V hlavičce segmentu budou Vámi navržené atributy protokolu RDT. V segmentu je atribut id, do kterého uveďte svůj login.

Cíl řešení

Vytvořte dva programy, které představují odesílatele a příjemce dat. Jedná se tedy o jednosměrný spolehlivý přenos dat. Odesílatelem bude program:

```
rdtclient -s source_port -d dest_port
```

- Slouží pro vytvoření spojení na vzdálený port. Odpovídá implementaci klienta. Zdrojový port je specifikován pouze pro zjednodušení implementace a slouží pro naslouchání komunikace od serveru.
- Program po spuštění očekává aplikační data na standardním vstupu, do kterého se může přesměrovat libovolný soubor. Program odesílá data, dokud nenarazí na konec datového proudu. Po úspěšném odeslání všech dat se program korektně ukončí.

Příjemcem zpráv bude program:

```
rdtserver -s source_port -d dest_port
```

- Slouží pro naslouchání příchozím požadavků na zvoleném portu. Odpovídá implementaci serveru. Cílový port je specifikován pro zjednodušení a pouze na tento port budou odesílány odpovědi.

- Program po spuštění očekává připojení klienta. Po připojení a vytvoření spolehlivého přenosového kanálu jsou přijatá dat vypisována na standardní výstup. Nevypisujte žádné další informace na standardní výstup! Pro ladění či jiné informace použijte stderr.

Po ukončení přenosu se musí `rdtclient` korektně ukončit. Program `rdtserver` bude reagovat na signál `SIGTERM`.

Je bezpodmínečně nutné dodržet jména souborů a názvy argumentů.

Postup řešení

1. **Navrhněte mechanismus spolehlivého přenosu dat:** Tento mechanimsmus bude implementovaný na straně klienta a serveru. Můžete zvolit libovolný ze známých principů spolehlivé komunikace založený na potrvzování (pozitivní, negativní, kumulativní, selektivní). Při implementaci je nezbytné myslet na efektivitu přenosu - tedy realizovat nějakou vhodnou formu zřetězeného přenosu.

2. **Navrhněte hlavičku RDT protokolu:** V hlavičce protokolu pro spolehlivý přenos dat se musí objevit informace potřebné pro zajištění spolehlivého doručení dat, řízení toku a detekci chyb. Pro inspiraci se můžete podívat do hlavičky protokolu TCP. Množství a charakter potřebných informací bude dán volbou mechanismu spolehlivého přenosu dat provedeného v kroku 1.

3. **Implementujte protokol nad UDP:** Pro samotný přenos dat využijte transportního protokolu UDP. Předpokládejte, že přenosový kanál povoluje UDP datagramy, které obsahují maximálně 512B dat.

Potřebné prostředky:

- knihovna BSD sockets, UDP sokety (viz. udtdemo.c)
- UNIX časovače (viz timer.c)
- parsování vstupních parametrů (funkce getopt)
- manipulace s XML (knihovna libxml2)
- kódování BASE64 (funkce b64_ntop, b64_pton)

Požadavky na program a testování

Pro otestování chování je možné využít programu udtproxy, jenž simuluje nespolehlivost při přenosu UDP datagramů. Implicitně se předpokládá, že komunikace probíhá na lokálním stroji.

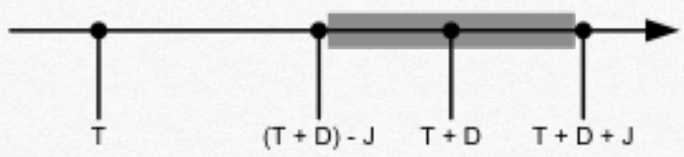

```
udtproxy -Q 10 -D 10 -J 5 -a 10000 -A 20000 -b 30000 -B 40000
```

- Q - délka vstupní fronty udávaná v “paketech”
- D – průměrné zpoždění paketů [ms]
- J – rozptyl zpoždění paketů [ms]
- a – lokální port strany A
- A – vzdálený port strany A
- b – lokální port strany B
- B – vzdálený port strany B

```
rdtclient -s 10000 -d 20000 < soubor_in.txt
rdtserver -s 30000 -d 40000 > soubor_out.txt
```

Nespolehlivost přenosu je možné simulovat kombinací parametrů:

- Vhodnou volbou zpoždění a rozptylu je možno simulovat přeházení pořadí paketů.
- Kombinací velikosti bufferu a zpoždění je možno simulovat ztrátovost paketů.



Paket bude zpožděn v intervalu $\langle (T+D)-J, T+D+J \rangle$, kde T je čas příchodu paketu na UDT proxy.

Není nutné uvažovat, že by při přenosu docházelo k bitovým chybám uvnitř přenášených dat. Pakety se tedy mohou pouze ztrácet, přijít v nesprávném pořadí nebo být výrazně zpožděny.

Poznámky k odevzdání

Vypracovaný projekt uložený v archivu .tar.gz a se jménem xlogin00.tar.gz odevzdejte elektronicky přes IS. Termín odevzdání je uveden v IS. Odevzdání emailem po uplynutí termínu není možné. Odevzdaný projekt musí obsahovat:

- soubory se zdrojovým kódem (dodržujte jména souborů uvedená v zadání),
- funkční Makefile pro překlad zdrojového souboru,
- a soubor readme.xml, který bude obsahovat stručný popis navrženého protokolu (formát zpráv, pořadí zpráv, zabez-

pečení spolehlivosti přenosu), seznam zdrojových souborů s uvedením autorů. Vyplněte šablonu, kterou dostanete k dispozici.

- na začátku každého zdrojového soboru musí být uveden jeho autor. Je možné převzít soubory, které přímo nesouvisí s implementací protokolu spolehlivého přenosu (např. funkce pro kódování BASE64). Toto musí být explicitně uvedeno v readme.xml.

Poznámky k hodnocení

Hodnocení bude provedeno na základě testů funkčnosti programu na sadě testů, které definují rozdílné parametry pro udt-proxy. Hodnocení projektu může až z 50% ovlivnit také způsob implementace, tedy kvalita a přehlednost zdrojových kódu. V případě, že implementace nebude odpovídat základním parametrům, tzn, nejedná se o zřetězenou komunikaci, není implementováno potvrzování, či mechanismus sliding-window, pak výsledné hodnocení může být 0bodů. Hodnocení bude také reflektovat dodržení dalších požadavků na program, například pojmenování programů a parametrů, korektní ukončení programu, atd.

Pro základní otestování, že Váš program je kompatibilní s testovacím prostředí vyzkoušejte následující skript.

```
udtproxy -Q 10 -D 10 -J 5 -R 10 -a 10000 -A 20000 -b 30000 -B 40000 &
pid_proxy=$!
rdtserver –s 30000 –d 40000 > soubor_out.txt &
pid_server=$!
rdtclient –s 10000 –d 20000 < soubor_in.txt &
pid_client=$!
sleep 60
kill -9 $pid_server
kill -9 $pid_client
kill -2 $pid_proxy
if diff –q soubor_out.txt soubor_in.txt >/dev/null
then
 echo “Soubor se prenesl spravne“
else
 echo “Soubor se neprenesl spravne“
fi
```

Počítačové sítě a Internet

1

V této kapitole je představena počítačová síť a zmíněny základní pojmy, tak aby byl vytvořen kontext pro hlubší studium problematiky, které bude následovat v dalších kapitolách. Cílem je vytvořit si základní rámcovou představu o počítačové síti jako celku a jejich hlavních součástí.

Cíle

Cílem úvodní kapitoly je získání základních informací o počítačových sítích, včetně vytvoření nezbytného kontextu a terminologie, na které pak bude navazováno v dalších kapitolách. Tyto kapitoly se detailněji věnují jednotlivým síťovým vrstvám a technologiím.

Po úspěšném zvládnutí látky této kapitoly by měl být student schopen:

- charakterizovat současný Internet
- vysvětlit co je síťový protokol a jeho význam pro komunikaci v počítačových sítích
- definovat pojem síťové rozhraní (network edge) a způsob jakým se k síti přstupuje
- definovat pojem páteř sítě (network core)
- vysvětlit rozdíl mezi přepínáním paketů a okruhů
- pochopit úlohu ISP a strukturu Internetu
- vysvětlit proč dochází ke ztrátam a zpoždění dat při jejich přenosu
- zdůvodnit vrstvovou architekturu současných počítačových sítí, včetně odkazů na ISO/OSI a TCP/IP model
- popsat jednotlivé druhy zpráv přenášené na různých vrstvách
- pochopit problematiku základních typů útoků na síť a její služby
- porozumět historickému vývoji Internetu

Výše uvedené znalosti jsou v rozsahu úvodní kapitoly a v průběhu kurzu se budou v jednotlivých oblastech prohlubovat.

Nejdůležitější koncepty

- **Počítačové sítě** jsou tvořeny koncovými stanicemi, paketovými přepínači a komunikačnimi linkami. Koncové systémy (end systems, hosts) jsou počítače, laptopy, smartphony, sensory a servery. Koncové systémy jsou propojeny sítí komunikačních linek. Komunikační linky jsou realizovány různou technologií jako drátové či bezdrátové.
- **Distribuované aplikace** využívají počítačové sítě, pomocí které si vyměňují potřebná data a koncových systémů, na kterých jsou prováděny. Příkladem jsou web, email, instant messaging, internetová telefonie, peer-to-peer sdílení dat nebo videokonference.
- **Přepínání paketů** je základní metoda používaná při doručování dat v Internetu. Posílaná data jsou rozdělena do datových paketů a každý paket je pak přenášen sítí nezávisle na ostatních. Kvůli tomu je nutné, aby každý paket identifiko-

val adresu příjemce, kterou má uloženu ve své hlavičce. Když paket dorazí na paketový přepínač, toto zařízení určí na základě cílové adresy výstupní linku kam má být paket přepnut. Často se používá metoda store-and-forward.

- **Protoko**l definuje formát a pořadí zpráv vyměňované komunikujícími stranami. Dále protokol definuje akce, které jsou provedeny při odeslání nebo přijetí zprávy.

- **Přepínání okruhů** je technika komunikace vlastní tradičním digitalním telefoním sítím. V těchto sítích je nutné před tím, než může dojít k přenosu dat mezi koncovými stanicemi, ustanovit spojení a zároveň rezervovat dostatečné přenosové pásmo v rámci každé linky podél cesty.

- **Fyzické médium** a **přístup k síti** se může lišit v závislosti na použité technologii. Modemové linky, DSL a nejčastější typ Ethernetu používají měděných spojů. Internetové páteřní spoje jsou nejčastěji realizovány na bázi optických spojů. Bezdrátové spoje jsou zastoupeny Wi-Fi technolgiemi, Bluetooth a satelitními spoji. V rámci Internetu jsou podporovány různé technologie.

- **Internet** se skládá z velkého množství propojených sítí. Každá tato síť se nazývá Internet Service Provider (ISP). Každé ISP je síť přepínačů a komunikačních linek. Z tohoto pohledu je Internet síť sítí. ISP jsou hierarchicky organizovány. ISP na nejnižší úrovni představuje například univerzitní ISP, podnikové ISP, rezidenční ISP. Naproti tomu, ISP na nejvyšší úrovni jsou spravovány organizacemi na národní a nadnárodní úrovni, propojující geograficky velmi vzdálená místa. Každý ISP na n-té úrovni poskytuje služby IPS na n+1 úrovni. Každý ISP je samostatně spravovaná síť v rámci Internetu. Pro vzájemnou kompatibilitu musí všichni ISP pro přenos dat používat stejný protokol - **Internet protokol** (IP).

- **Zpoždění přenosu** je důležitá vlastnost komunikačních sítí, kterou je nutné zohlednit při implementaci distribuovaných aplikací. Doba přenosu (**propagation delay**) je doba potřebná pro přenesení bitu informace z jednoho konce linky na druhý. Doba odeslání paketu (**transmission delay**) je doba potřebaná pro vyslání paketu, Celková doba potřebná pro přenos paketu je tedy součtem doby přenosu a doby odeslání.

- Přepínače obsahují **fronty**, které řeší situaci, kdy na přepínač přijde zhruba ve stejný okamžik více paketů určených pro stejnou výstupní linku. Pro obsloužení čekají pakety ve frontě dokud není výstupní médium volné pro jejich odeslání. V době, kdy paket čeká ve frontě, je jeho přenos zpožděn (**queue delay**). V případě zaplnění fronty se může stát, že paket není možné zpracovat a je přepínačem zahozen (**packet loss**). Obě situace mají vliv na výkonnost uvažované distribuované aplikace.

- Typická počítačová síť používá mnoho protokolů. Protokoly jsou organizovány do **vrstev**, které se nazývají **protokolový zásobník**. Internet definuje protokolový zásobník tvořený pěti vrstvami: aplikační, transportní, síťová, linková a fyzická. Vrstva nižší poskytuje přenosové služby vrstvě vyšší.

- Jestliže chce aplikace odeslat data, použije služeb transportní vrstvy. Aplikace předá datovou jednotku transportní vrstvě, kde data představují **payload** segmentu transportní vrstvy. Tento segment obsahuje další informace v přidané hlavičce, například o aplikaci, které jsou data nesená segmentem určena na cílové stanici. **Segment** transportní vrstvy si lze představit jako obálku, do které byla vložena data aplikační vrstvy. Tento proces se nazývá **Encapsulation**. Podobným způsobem se přidávají informace při předání segmentu síťové vrstvě. Segment se stává obsahem **datagramu**, což je datová jednotka síťové vrstvy, V hlavičce datagramu jsou například uvedeny informace o zdrojové a cílové adrese. Nakonec je datagram předán linkové vrstvě, která jej vloží do **rámce**.

Literatura

Kurose & Ross: COMPUTER NETWORKING: A Top-Down Approach.
Kapitola 1 této knihy představuje základní studijní materiál. Struktura přednášky i množství potřebných znalostí odpovídá přesně obsahu kapitoly. Na konci kapitoly je uvedeno množství otázek, domácích úkolů a dalších aktivit k posílení znalostí. V závěrečném testu se mohou objevit otázky z nich vybrané či jim podobné.

Další literatura se skládá z článků a specifikací, které se věnují dílčím problémům týkajících se studované problematiky:

Gettys, J. (2011). Bufferbloat: Dark buffers in the internet. Internet Computing, IEEE, 95-96.
Popularizační článek o problematice bufferů v současném Internetu a ve vztahu k požadavků na kvalitu služeb.

Jiří Balej (2010). Simulace zpoždění při přenosu dat mezi stanicemi v IP sítích, Diplomová práce, FEKT VUT v Brně.
Práce zahrnuje popis míst vzniku zpoždění při přenosu informací, uvádí paramtry tohoto zpoždění a dává příklady pro různé typické sítě. Zajímavé jsou i výsledky měření zpoždění a jejich porovnání s vypočtenými hodnotami.

Aktivity

A1: Nainstalujte si Packet Sniffer - nástroj pro sledování provozu. K nejrozšířenějším patří Wireshark (Windows/Linux/MAC) nebo Microsoft Network Monitor (Windows). Naučte se ovládat tento nástroj a sledujte komunikaci na vašem PC. Pro Vám ivybrané protokoly zjistěte informace, které se přenášejí. Zkuste například zachytit HTTP komunikaci Vašeho prohlížeče a nalézt data, které prohlížeč odesílá a přijímá.

Otázky

O1: Stučně popište princip komunikace mezi aplikací A a B při přenosu souboru. Jaký je postup vzniku rámce odesláného ze stanice A? Co se stane, když rámec přijde na přepínač? Co se děje s rámcem na koncovém systému B?

O2: Jaké multiplex techniky se používá pro přepínání okruhů?

O3: Je výhodnější použít ořeoínání okruhů nebo paketů, jestliže data jsou odesílána rovnoměrně s konstantním bit rate? A co v případě, kdy data jsou posíálna nepravidelně ve shlucích?

O4: Uživatelé sdílejí linku 1Mbps. Každý uživatel potřebuje 500kbps při odesílání dat. Tato komunikace trvá 10% celkového času.

> Kolik uživatelů je možné obsaloužit v případě přepínání okruhů?
>
> Pro přepínání paketů, jaké bude zpoždění queue delay, když budou vysílat současně 2,3 nebo 4 uzly?
>
> Pro tři uzly urči pravděpodobnost, že uživatelé odesílájí současně.

O5: Co charakterizuje ISP úrovně 1?

O6: Jak dlouho trvá přenos paketu délky 1000 bajtů na lince o délce 5000km (rychlost šíření signálu je $2.5\times10^8 \, m/s$) a přenosová rychlost je 1Mbps.

O7: N paketů každý délky L přijde současně na linku s přenosovou rychlostí R. Jaké je průměrné zpoždění N paketů?

O8: Přenosová rychlost R= 1Mbps, velikost paketu L = 1250bajtů, Pakety přichází náhodně v intervalu 1 paket/s. Intenzita přenosu je $I = L \times a/R$. Průměrné zpoždění ve frontě je modelováno jako: $[I/(1-I)]\times[L/R]$. Jaké je zpoždění ve frontě a celkové zpoždění (zpoždění ve frontě + zpoždění přenosu) pro a=30,60 a 90 paketů/s.

O9: Uvažujte spojovanou přenosovou službu v síti s paketovým přepínáním. Obsah paketu má velikost F bitů. Každý paket má hlavičku o velikosti $h \times F$, kde $0 < h < 1$. Cesta obsahuje Q linek. Linka *i* má přenosovou rychlost R_i . Doba přenosu je zanedbatelná. Uvažujte, že v síti nedochází ke zpoždění ve frontě. Jak dlouho bude trvat odeslání paketu?

O10: Vyjmenujte vrstvy modelu Internetu? Jak se jmenujtí datové jednotky pro jednotlivé vrstvy? Na jaké vrstvě “pracuje” koncový systém? Na jaké vrstvě “pracuje” směrovač? Na jaké vrstvě pracuje switch?

Aplikační vrstva

2

Síťové aplikace definují požadavky na vlasnotsti počítačových sítí. Charakter aplikací se neustále vyvíjí. V této kapitole budou představeny konceptuální a implementační aspekty vybraných síťových aplikací.

Cíle

V této kapitole je prezentována aplikační vrstva. Tato vrstva zahrnuje distribuované aplikace, které využívají komunikačních prostředků Internetu k výměně dat mezi jednotlivými výpočetním uzly. Principy jsou představeny pomocí nejpoužívanějších Internetových aplikací:

- Web, protokol HTTP
- Přenos souborů, FTP, TFTP
- Vzdálený přístup, Telnet
- Systém doménových jmen, DNS
- Elektronická pošta, SMTP, POP3, IMAP, MIME

Studiem materiálů této kapitoly by měl student:

- získat znalosti o tom jak pracují výše uvedené aplikace/služby
- poznat a porozumět klíčovým komunikačním mechanismům a vzorům používaným na aplikační vrstvě
- seznámit se s konkrétními protokoly a způsobem jejich definice dle příslušných RFC
- pochopit význam služby DNS pro adresování služeb v rámci Internetu

Uvedené znalosti a kompetence budou prohloubeny v rámci projektu č.1, jehož tématem je implementace komunikace vybrané služby.

Nejdůležitější koncepty

- **Protokol aplikační vrstvy** definuje formát, pořadí zpráv a akce odesílatele a příjemce zpráv a reakci na možné další událost. Pro Internet jsou protokoly specifikovány v příslušném RFC. Například HTTP/1.1 (RFC2612), FTP (RFC959+3695), DNS (RFC 1034,1035,...), SMTP (RFC 2821).
- Komunikace **klient/server** definuje rozdílné role pro komunikující strany. Klient žádá o službu zasláním zpráv serveru. Server přijímá zprávy a provádí požadované úkony. Vetšinou je výsledek zaslán zpět ke klientovi. Příkladem může být žádost Internetového prohlížeče (klient) o získání určité webové stránky. V případě **peer-to-peer** komunikace mají obě strany stejnou roli.
- Aplikace používají transportní vrstvu Internetu pro přenos svých dat. K dispozici mají službu **spolehlivého, řízeného datového přenosu (TCP)** a **nespolehlivého datového přenosu (UDP)**. Transportní vrstva neposkytuje jiné přenosové služby, které by například garantovaly maximální zpoždění nebo minimální šířku pásma.

- Protokol **HTTP** pracuje v režimu **žádost/odpověď (request/response)**. Tento režim je často používaný i jinými aplikacemi. HTTP klient definuje požadavek pomocí GET zprávy. Webový server odpovídá zasláním obsahu požadovabé stránky. Pro komunikaci se používá TCP datového přenosu, což zjednodušuje implementaci protokolu HTTP. Před zasláním požadavku je nutné ustanovit TCP spojení. HTTP umožnuje pracovat ve dvou režimech:
  - **persistent HTTP**: jedno TCP spojení je použito pro celou komunikaci klient/server. To znamená, že série žádostí a odpovědí je přenášena v rámci jediného vytvořeného TCP spojení.
  - **non-persistent HTTP**: pro každou dvojici žádost/odpověď se vytváří samostatné TCP spojení, které je po vyřízení žádosti ukončeno.
- V síťových aplikacích se často využívá ukládání získané informace pro její další využžití. Toto se označuje jako **caching**. Například v DNS je získaná informace o adrese a příslušejícím doménovém jméně lokálně uložena, což šetří čas a komunikační prostředky v případě, že se později objeví dotaz na stejnou informaci. Webový klient spravuje **cache** nedávno navštívených stránek a v případě nového přístupu se nejprve kontroluje jejich aktuálnost před tím než je celá stránka přenášena ze serveru. Podobně webové proxy udržují ve paměti cache nedávnou navštěvované stránky, které pak mohou přímo poskytovat různým klientům.
- **Služba DNS** poskytuje překlad doménových jmen a IP adres. Tato služba je realizována aplikačním protokolem, který používá UDP a TCP datových přenosů. Tato služba je tvořena mnoha DNS servery, které udržují aktuální informace o mapování doménových jmen a IP adres. Přestože je tato služba klíčová pro fungování dnešního Internetu je implementována stejnými prostředky jako ostatní Internetové aplikace.
- Přenos souborů **službou FTP** představuje zajímavý příklad protokolu, který **odděluje komunikaci od řízení (out-of-band control)**. Datová komunikace používá oddělené TCP spojení od řídící komunikace.
- Knihovna BSD sockets definuje dva typy schránek (sockets). **TCP socket** je vytvořen v okamžiku, kdy aplikace serveru dostane řízení ze systémového volání accept(). Tento socket je možné použít pro komunikaci s klientem. **UDP socket** je vytvořen okamžitě a přijímá veškerou komunikaci přicházející na příslušný port systému. Je na plikaci, aby odlišila různé komunikující klienty.
- Aplikace si vyměňují data. Základními principy, které řídí způsob vyměny dat jsou metody **pull** nebo **push**. V systémech typu pull musí aplikace, která data potřebuje, explicitně požádat zprostředkovatele dat. V systémech typu

push, se data posílají příjemci v případě, že jsou data k dispozici (například SMTP, kdy email je zaslán příjemci bez předchozí žádosti).

- Peer-to-peer sítě nemají centralizovanou architekturu, která by byla použitelná například pro indexaci a lokalizaci služeb a informací, tak jak je tomu v klient/server systémech (DNS). Pro lokalizaci informace v P2P systémech se používají techniky:
  - query flooding - pro nalezení informace se používá dotazování na odstupné uzly (peer) a přeposílání dotazů na další uzly dokud není informace lokalizována
  - directory system - existují vyhrazené uzly, které spravují index. Obyčejné uzly se dotazují těchto uzlů.
  - hybrid system - používají se tkz. superuzly, které poskytují index pro určitou doménu. V případě nenalezení se dotazují další superuzly.

Literatura

Kurose & Ross: COMPUTER NETWORKING: A Top-Down Approach.
Kapitola 2 této knihy představuje základní studijní materiál. Tato kapitola také obsahuje informace k programování BSD schránek. Ty jsou předmětem následujícího modulu. Jejich detailnější studium je možné odložit.

Aktivity

A1: Prostudujte základní formát a sémantiku požadavků protokolu HTTP/1.1. Pomocí nástroje telnet se připojte na Vámi vybraný webový server a vytvořte dotaz a obdrženou odpověď uložte. Zobrazte odpověď pomocí prohlížeče.

A2: Pomocí nástroje telnet se připojte na SMTP server a pokuste se odeslat email. Dodržujte pravidla pro používání elektronické pošty.

Otázky

O1: Vyjmenujte, které služby poskytuje TCP aplikační vrstvě.

O2: Vyjmenujte, které služby poskytuje UDP aplikační vrstvě.

O3: Jakou transportní službu byste použili v případě, že:

a. Sesor čte hodnotu každých 500ms. Je důležité mít tuto informaci co nejdříve na přijímači. Přitom není nutné přijmout všechny hodnoty.

b. Všechny transakce, které se na serveru provádí jsou kritické a je nutné zachovat jejich pořadí a úplnost.

c. Aplikace pro přenos hlasu po Internetu.

d. Apikace pro přenos rychlých zpráv mezi uživateli po Internetu (ICQ, IM, Google Talk).

e.Aplikace pro přenos elektronické pošty.

O4: V zachyceném HTTP segmentu je tento obsah:

GET /vyuka/index.html HTTP/1.1
Host: www.fit.vutbr.cz
User-Agent: Mozilla/5.0 (Windows;U; Windows NT 5.1; en-US; rv:1.7.2)
Accept:ext/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: en-us,en;q=0.5..Accept- Encoding: zip,deflate
Accept-Charset: ISO -8859-1,utf-8;q=0.7,*;q=0.7..Keep-Alive: 300
Connection:keep-alive

a.Jaké URL má dokument, o který klient žádá?

b.Jakou verzi HTTP klient použil?

c.Jaký prohlížeč klent používá?

d.Jaký typ HTTP spojení je použito?

e.Jakou IP adresu má klient and server?

O5: Uvažujte persistentní a nepersistentní HTTP spojení. Stránka, kterou bude prohlížeč stahovat má velikost 100k bitů a obsahuje 10 obrázku, každý o velikosti 200k bitů. RTT serveru je 300ms. Cesta mezi klientem a serverem má kapacitu 100Mbps. Jaká bude celková doba potřebná pro stáhnutí stránky v případě:

a.nepersistentního HTTP, žádné paralelní spojení

b.nepersistentního HTTP, paralelní spojení

c.persistentní HTTP, žádný pipelining

d.persistentní HTTP, pipelining

O6: Jaký je rozdíl mezi in-band a out-of-band řízením? Uveďte příklady aplikačním protokolů, které používají různé řízení.

O7: Popište jakým způsobem se používá caching v systému DNS.

O8: Jaký je rozdíl mezi zprávou MAIL FROM: v komunikaci SMTP a From: jak je uvedeno v emailu?

O9: Jaká je posloupnost DNS a HTTP zpráv, pozorovatelná od okamžiku kdy se do prohlížeče zadá URL dokumentu, než je tento dokument zobrazen?

O10: Popište princip stavového chování protokolu HTTP. Kde a jaké informace se ukládají? K čemu je možné cookies použít?

BSD schránky

3

Programování síťových aplikací vyžaduje pochopení základních principů komunikace. Jako abstrakce zde slouží schránka (socket), které je věnována tato kapitola. Budou představeny základní přístupy síťového programování, které budou demonstrovány na knihovně BSD sockets.

Cíle

V této kapitole se zaměříme na programování síťových aplikací pomocí knihovny BSD sockets. Bude vysvětleno:

- princip a architektura knihovny BSD socket
- programování klient/server komunikace protokolem TCP
- programování klient/server komunikace protokolem UDP
- principy vytváření iterativního a konkurentního serveru
- ukázka implementace jednoduché klient/serve komunikace v C a v Javě.

Uvedené znalosti jsou potřebné pro realizaci projektů. Samotná realizace jednoduché kleint/server aplikace je přímočará. Místa, která vyžadují pozornost jsou vysvětlena důkladněji. Jedná se zejména o:

- převod doménového jména na ip adresu
- navázání na socket
- vytvoření neblokující komunikace
- paralelní zpracování příchozích požadavků

Nejdůležitější koncepty

Mezi nejdůležitější koncepty představené v této kapitole patří:

- **Schránka (socket)** je abstrakce operačního systému, která zpřístupňuje funkce transportní vrstvy aplikacím. V UNIXových operačních systémech je schránka reprezetována jako file descriptor.
- Pro **TCP schránk**y platí, že server vytvoří socket, na kterém očekává požadavky na nová spojení od klientů (welcome socket). Pro osbluhu požadavku ok konkrétního klienta je poté vytvořen nový socket (connection socket), určený pouze pro tuto komunikaci. Tento socket je vrácen z funkce **accept**.
- TCP komunikace poskytuje aplikaci službu spolehlivého přenosu proudu bajtů (**reliable byte-stream**). Pro čtení dat se používá funkce **read**. Tato funkce vždy vrací pouze dostupná data (stejně jako v případě vstupu od uživatele). Toto je rozdíl od čtení dat ze souboru, kdy data jsou vždy dostupná.
- Schránky mohou pracovat v **blokujícím** nebo **neblokujícím** režimu. V blokujícím režimu je provádění programu při zavolání blokující operace přerušeno, do té doby, než je operace dokončena. Například, při požadavku na čtení (operace read) je provádění programu přerušeno, než jsou k dispozici ve vstupním bufferu data. V neblokujícím režimu je řízení vráceno zpět programu okamžitě. Návratová hodnota E_WOULDBLOCK značí, že operace nebyla úspěšně provedena.

- Knihovna BSD schránek obsahuje funkce nezbytné pro nastavení parametrů komunikace a pomocné funkce pro podporu komunikace. Jedná se například o funkce pro získání IP adresy z doménového jména (**gethostbyname**). Tato funkce využívá lokální DNS resolver pro převod. Dále jsou k dispozici funkce pro převod integer datových typů do síťové neutrální reprezentace (**ntohl**, **ntohs, inet_aton**).

Literatura

Kurose & Ross: COMPUTER NETWORKING: A Top-Down Approach.
Kapitola 2 této knihy obsahuje informace k programování BSD schránek. Je uveden příklad jednoduché klient/server aplikace v jazyce C. V novějších edicích je příklad uveden v Javě.

Stevens: UNIX Network Programming, Volume 1: The Sockets Networking API. 3 vydání, Addision-Wesley, 2004.
Kniha obsahuje vyčerpávající popis programování síťových aplikací s pomocí BSD sockets, včetně referenční příručky této knihovny.

Gilligan, Thomson, Bound, McCann, Stevens: Basic Socket Interface Extensions for IPv6, RFC3439, 2003.
V tomto dokumentu je popsáno revidované API, které obsahuje podporu pro IPv6 a zároveň je kompatibilní s IPv4.

Aktivity

A1: Seznamte se s demo aplikaci udt_demo. Podívějte se na způsob komunikace pomocí UDP přenosu.

A2: Seznamte se s demo aplikací messenger. Pomocí nástroje Wireshark, zachyťte komunikaci mezi SMTP serverem a tímto klientem. Podívejte se, kdy se vytváří TCP zprávy a jakým způsobem dochází k jejich potvrzování.

Otázky

O1: Jak je schránky (socket) reprezentována v operačním systému?

O2: Jaké základní operace se schránkami poskytuje BSD Sockets API?

O3: Čím se identifikuje spojení mezi síťovými aplikacemi?

O4: Co obsahuje struktura **sockaddr_in** pro protokol IPv4.

O5: Vysvětlete nutnos použití konverzních funkcí **hto***.

O6: Jaké typy socketů lze vytvořit voláním funkce socket?

O7: Jaké funkce BSD Sockets API vrací popisovač schránky (socket descriptor) jako svojí návratovou hodnotu?

O8: Jaké operace jsou mohou být blokující? Vysvětlete proč.

O9: Jaké operace použité při vytváření spojované komunikace se nepoužijí pro nespojovanou komunikaci?

O10: Popište rozdíl mezi konkurentním a iterativním serverem. Jakým způsobem lze realizovat konkurentní server?

Spolehlivý přenos

4

Spolehlivý přenos dat je v TCP/IP modelu implementován na transportní vrstvě. Tato kapitola představí problematiku zajištění spolehlivého přenosu. Postupně budou demonstrovány různé implementace zajišťující spolehlivost na různé úrovni. Bude také představen další problém který je řešen na transportní vrstvě - řízení množství přenášených dat.

IPv4 a IPv6

5

Tato kapitola se zabývá adresování, což je jeden k konceptů realizovaný na síťové vrstvě. Bude představen protkol IPv4 a IPv6, včetně problematiky adresování uzlů v Internetu. S tím souvisí další techniky, například překlad adres, či koexistence a přechod IPv4 na IPv6 sítě.

Směrování I

6

Síťová vrstva se zabývá doručením paketů k cíli. Vzhledem k tomu, že Internet je paketově přepínanou sítí, je potřeba znát informace potřebné krozhodnutí kam má být paket přepnut, což se děje na každém směrovači v síti. Tato kapitola je první částí zabývající se problematikou přepínání paketů a nalézání směrovaích informací v síti.

Směrování II

7

Směrování v Internetu představuje komplexní problém. Předchozí kapitola se zabývala základními principi směrování v Internetu. Tato kapitola přináší další informace a zabývá se především směrováním mezi ISP. Je zde předtaven směrovací protokol BGP.

Multicast

8

Multicast umožňuje posílat setjná data více příjemcům a snižuje tak požadavky na přenosové pásmo v distribuční síti. Tato kapitola se zabývá problematikou multicastové komunikace z praktického pohledu. Budou představeny protokoly pro správa multicastové komunikace a různé komunikační modely včetně příkladů rlzných konfigurací multicastové komunikace.

LAN, Ethernet, Ethernet

Linková vrstva a lokální sítě jsou představeny v této kapitole. Je zde diskutováno jak jsou data přenášena fyzickými linkami, jaké významné technologie lze nalézt v dnešních LAN a jaké přístupové metody se používají v závislosti na charakteru linky (vyhrazené či sdílené médium). Více prostoru dostane nejrozšířenější LAN technologie - Ethernet.

WLAN, WAN

10

V této kapitole budou představeny základní informace o bezdrátových sítích a technologiích používaných v páteřních sítích.

Bezpečnost

V této kapitole budou představeny prstředky pro zabezpečení komunikace před některými bezpečnostními hrozbami. Stručně budou diskutovány základní bezpečností mechanismy, které se při zabezpečení komunikace implementují. Představení zabezpečení bude obsahovat autentizaci komunikujících stran, zabezpečení emailu, TCP spojení (SSL), a virtuální privátní sítě.

Protokoly pro QoS

12

Zajištění kvality služeb (QoS) předpokládá použití mechanismů. které mohou garantovat splnění potřebných požadavků na datový přenos. Vzhledem k charakteru Internetu je toto nutné řešit (podobně jako bezpečnost) dodatečnými mechanismy. V této kapitole budou představeny tři protokoly,, které se v souvislosti s QoS používají. Jedná se o RSVP, MPLS a RTP.

Peer-to-peer sítě

13

Peer-to-peer komunikace je odlišné paradigma od klasické klient-server komunikace, které se v poslední době velmi rychle rozvíjí v Internetu. Množství aplikací používá tento způsob komunikace. V této kapitole budou představeny základní principy při implementaci P2P architektury a komunikace, včetně diskuze souvisejících problémů její začlenění do současného Internetu.