Počítačové komunikace a sítě

Projekt 3: Implementace zřetězeného RDT

15.2.2011

Implementujte protokol spolehlivého zřetězeného přenosu dat RDT dle přednášek. Projekt bude vypracován v jazyce C (tedy žádné C++, java, python, atd.), přeložitelný a spustitelný v prostředí FreeBSD na serveru eva.fit.vutbr.cz.

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

Cíl řešení

Vytvořte dva programy:

```
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. Bude se pouze jednak o textové řetězce v maximální délce 80 znaků. Oddělovačem je konec řádků (\n). Každý celý řádek bude tvořit segment, jenž bude přenášen UDP datagramem společně s odpovídající hlavičkou RDT. Program odesílá data, dokud nenarazí na konec datového proudu. Po úspěšném odeslání všech dat se program korektně ukončí.

```
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 program korektně ukončit.

Je bezpodmínečně nutné dodržet jména souborů a názvy argumentů.

Postup řešení

1. Navrhněte strukturu paketu RTP protokolu
2. Navrhněte mechanismus spolehlivého přenosu
3. Navrhněte a implementujte mechanismus řízení toku principem sliding window
4. Implementujte pomocí nad UDP, je možné využít funkcí definovaných v udt.h.

Při řešení se můžete inspirovat příkladem `udtdemo.c`, který sice neimplementuje žádný z rysů protokolu RDT, ale ukazuje jak pracovat s UDP a I/O. Dále budete potřebovat časovače. Program `timer.c` ukazuje použití standardního UNIX časovače. Pro parsování parametrů použijte funkci `getopt`, jak je uvedeno v `udtdemo.c`. Jelikož je nutné ověřit, že příchozí datagram neobsahuje

chybu (uvažujte pouze 1- bitové chyby) budete potřebovat implementovat vhodný algoritmus kontrolního součtu. Jako inspiraci uvažujte RFC 1071.

Při návrhu časovače uvažujte následující limity:

- Maximální zpoždění na lince 500ms
- Maximální velikost přenášených dat 1MB
- Maximální přípustná doba přenosu 60 s

Testování

Pro otestovaní 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 -R 10 -a 10000 -A 20000 -b 30000 -B 40000
```

Q - délka vstupní fronty udávaná v paketech

D – průměrné zpoždění paketů v ms

J – rozptyl zpoždění paketů v ms

R – definuje podíl chybně přenesených paketů

a – lokální port strany A

A – vzdálený port strany A

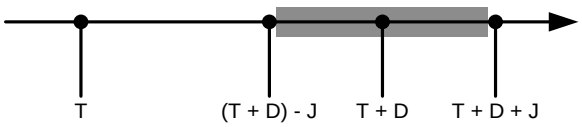
b – lokální port strany B

B – lokální 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.
- Pomocí volby R se nastavuje kolik procent paketů bude obsahovat jednobitovou chybu.



Paket bude zpožděn v intervalu [(T+D)-J, T + D + J], kde T je čas příchodu paketu do UDTProxy.

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.txt**, který bude obsahovat základní informace o vlastnostech programu a stručný popis navrženého protokolu.

Co není v zadání jednoznačně uvedeno, můžete implementovat podle svého vlastního výběru. Zvolené řešení stručně popište v souboru Readme.

Na začátku každého zdrojového souboru musí být uveden autor, například:

```
/*
 * Projekt IPK3
 * Autor: Jan Novak, xnovak04@stud.fit.vutbr.cz
 * Datum: 10.4.2010
 */
```

Poznámka k hodnocení

Pro základní kontrolu zkuste následující test:

```
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
```

Odkazy a studijní materiály

1. Stevens et.el. Unix Network Programming: The Sockets Network API. Volume 1. Addison-Wesley Professional,2004.

2. Parziale, et.el. TCP/IP Tutorial and Technical Overview. IBM Redbooks serie, available online: http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/gg243376.html

3. Přednášky IPK.