Projekt 3: Reliable Protocol

Implementujte protokol spolehlivého přenosu dat.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 dle specifikace pro zvolenou variantu.

Protokol bude textový, kdy pro posílané zprávy bude použito XML popisu. Jako přenášená data je možné uvažovat pouze textové soubory.

Každý segment bude mít následující formát:

```
<rdt-segment id="xnovak01">
 <header sn="seriove cislo" ack="potvrzeni"  win="velikost okna"
tack="hodnota casovace v ms">

... ostatní informace potřebné k implementaci ...

 </header>
 <data>
.. textová data ..

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

V hlavičce segmentu budou požadované položky a 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. Dle varianty Vašeho zadání implementujte požadovaný mechanismus řízení 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)

Požadavky na program a 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.

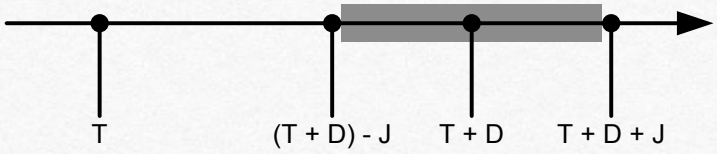

```
udtproxy -Q 10 -D 10 -J 5 -a 10000 -A 20000 -b 30000 -B
40000
```

- 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. pomocné funkce pro quotation/unquotation textových dat). 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 udtproxy. 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
```

Detaily implementace:

Protokol používá sekvenčních čísel k detekci ztrát, duplicit a přeuspořádání. Sekvenční čísla jsou packet-based, tedy každý nový paket má o jedno vetší SN než předchozí. (pozor TCP používá offset-based SN)

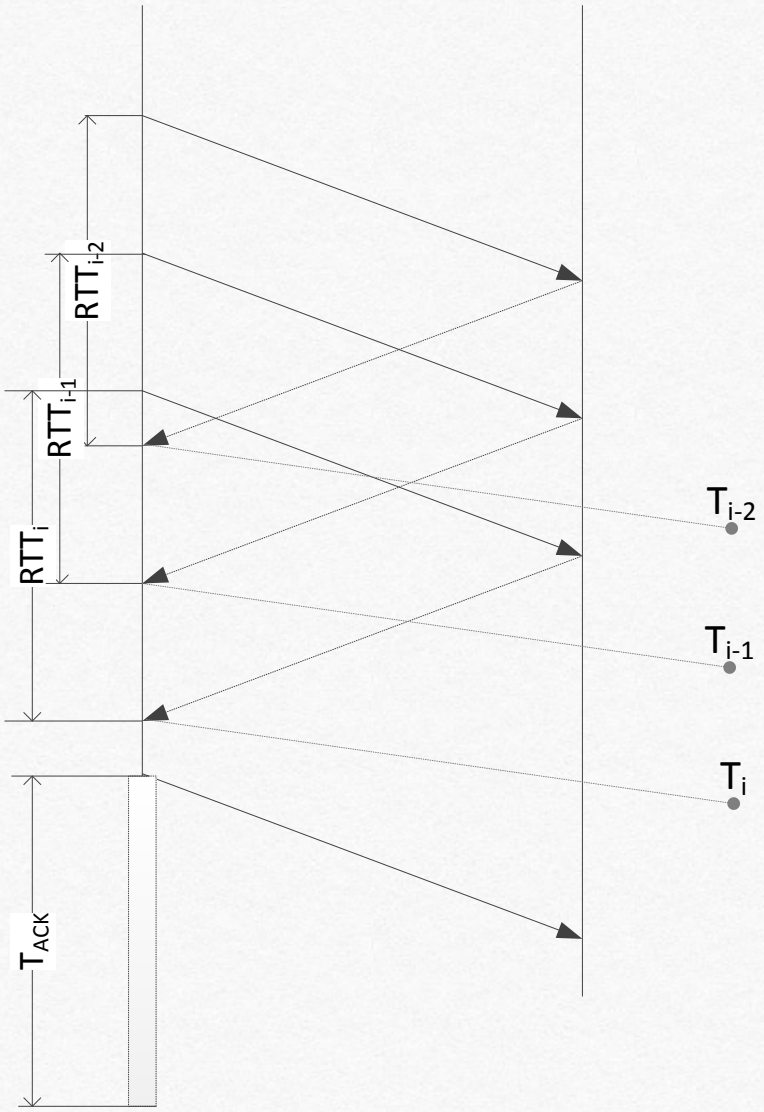
Protokol používá potvrzení posíláné pro každý doručený paket. V případě, že nepřijde potvrzení než dojde k vypršení časovače T_{ack}, je paket znovu zaslán.

Hodnota T_{ack} je určena z měření RTT pro jednotlivé odeslané pakety. Výpočet T_{ack} je následující:

Tedy poslední 3 měření RTT, kdy nejnovější má větší váhu je průměrováno s aktuální hodnotou časovače. Pro nastavení časovače Tack je pak tato hodnota zvětšena o 1/3.

$$T_i = \frac{T_{i-1} + \left(\frac{2 \cdot RTT_i + RTT_{i-1} + RTT_{i-2}}{4}\right)}{2}$$

$$T_{ack} = T_i \cdot 1.3$$



Protokol používá congestion window pro řízení toku. V případě ztráty paketu/potvrzení je okno zmenšeno na 1/3 své velikosti, nejméně však na velikost 1 paketu. V případě přijetí potrvzení je okno zvětšeno právě o 1 paket.

Varianta A:

Uvažujte schéma kumulativního potvrzování.

Varianta B:

Uvažujte schéma selektivního potvrzování.