Uppgift 2: Datakommunikationsprotokoll

Inledning

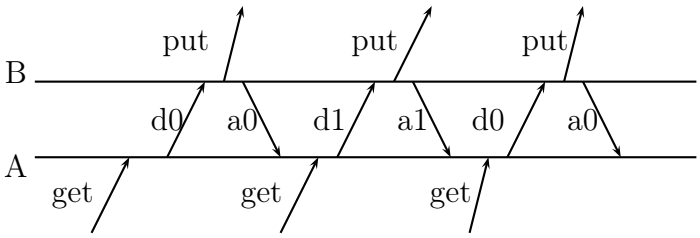
Vi betraktar en situation där datorn A sänder datapaket till datorn B med användande av en kanal som kan förorsaka datakommunikationsfel. Datapaketet är en sträng av bitar, dvs nollor och ettor. Det finns två slag av datakommunikationsfel: antingen förloras paketet helt och hållet, eller så ändras bitar i paketet (noll till ett eller ett till noll), och vi talar då om *paketets förvrängning*. Den normala situationen är sådan att kommunikationsfel sällan inträffar.

Det finns två hastigheter som är relevanta för datakommunikationen. För det första kan datorn skicka till kanalen med en viss maximal hastighet. Denna hastighet,*kanalens kapacitet* (i enheten bitar per sekund), beror på kanalens egenskaper (och delvis på datorns konstruktion, men i allmänhet är kanalens egenskaper avgörande). Den andra hastigheten hänför sig till hur snabbt bitarna rör sig i kanalen. Denna hastighet är vanligen ljusets hastighet i mediet (km/s). Om paketen förflyttas över långa distanser (tusentals kilometer), börjar fördröjningen som avståndet förorsakar bli sådan att den måste beaktas i vissa situationer. Med hjälp av dessa hastigheter kan man räkna ut hur länge det tar att sända och ta emot paketen.

Alternerande bitars protokoll

Datorerna använder följande algoritm, även kallat protokoll, för datakommunikationen (det så kallade *alternerande bitars protokoll*). Protokollet fungerar för sändarens del så att det tar emot datapaket från en applikation och sänder sen ut dem i kanalen. På mottagarens sida tar protokollet emot paket från kanalen och överlåter dem till applikationen. I praktiken placerar applikationen paketen i ett reserverat minnesområde kallat *buffert*, därifrån protokollet tar dem. På motsvarande sätt placerar protokollet på mottagarens sida paketen i en buffert, därifrån applikationen tar dem.

Vi antar i det som följer att den sändande maskinen är A och den mottagande B. När paketet som A sänt tas emot i B utan fel, kvitterar B att paketet mottagits med att sända tillbaka ett *kvitteringsmeddelande* till A. Först när kvitteringen mottagits av A, kan A ta ett nytt paket från bufferten och sända det. För att kvitteringsmeddelandet skall identifieras med rätt paket numrerar man paketen och kvitteringarna. Det visar sig att det räcker med numren 0 och 1. Följande schema visar hur protokollet funktionerar då inga kommunikationsfel inträffar.



Här tar A först ett datapaket från bufferten med ett get-meddelande och sänder datapaketet d försett med numret 0 och börjar vänta på att få kvitteringen. B tar emot datapaketet utan fel, överlåter det med ett put-meddelande till applikationen d.v.s. bufferten och sänder kvitteringen a försett med numret 0 och börjar vänta på följande paket. A tar emot kvitteringen utan fel med rätt nummer 0, tar ett nytt paket från bufferten och skickar därefter detta nya paket försett med numret 1. B tar emot detta paket, identifierar det som ett nytt paket med hjälp av numret, överlåter paketet till applikationen och sänder paketets kvittering med numret 1. Även denna kvittering kommer fram utan fel. Med hjälp av numret identifierar A det som kvitteringen för det föregående paketet, tar igen ett nytt paket från bufferten och sänder detta nya paket, som i sin tur får numret 0. B tar emot, överlåter och kvitterar.

I fortsättningen kallar vi ett sådant här schema som beskriver en situation i datakommunikation för *scenario*. I scenariot för detta protokoll löper alltså tiden från vänster till höger. Området mellan de horisontella linjerna beskriver situationen när meddelandena eller kvitteringarna är i kanalen. Put- och get-meddelandena hämtar respektive överlåter meddelanden i bufferten. Buffertarna har inte ritats ut i scenariot.

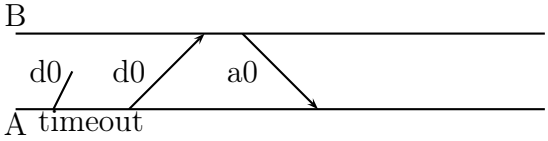
Protokollets funktion när kanalen förlorar eller förvränger meddelanden

I det föregående scenariot fungerade kommunikationskanalen utan fel. Protokollet måste dock vara berett också på fel. Vanliga fel är situationer, då paket förloras eller förvrängs, eller kvitteringen förloras eller förvrängs. Vi kan också anta att datakommunikationskanalen duplicerar paket. Detta betyder att när ett paket sänts, kan kanalen förmedla två eller t.o.m. flera kopior av samma paket. Det kan gå hur länge som helst mellan två kopior. Vi antar dessutom om datakommunikationskanalen att det kan finnas flera paket i kanalen samtidigt, men paketens ordning i kanalen kan inte ändras. Med andra ord, om paketen m_1 och m_2 har sänts till kanalen, så kommer paketen fram i samma ordning. Detta gäller också duplicering: Om kanalen duplicerar meddelandet

m till meddelandena m_1 och m_2, kommer de fram i samma ordning. Inget annat paket kan komma emellan m_1 och m_2.

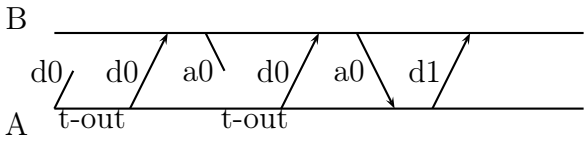
För att protokollets ska kunna återhämta sig från datakommunikationsfel tar vi i användning en *timeout*. Timeouten är på sändaren A:s sida. Ifall A inte får kvitteringen inom utsatt tid, utlöses timeouten, d.v.s. protokollet får ett *timeout-meddelande* (eller t-out-meddelande) och A sänder då det föregående datapaketet på nytt. Förvrängda datameddelanden och kvitteringar tolkas som förlorade paket, d.v.s. förvrängda paket förorsakar inga reaktioner. Protokollet måste fungera så att det inte överlåter paket som redan tidigare kommit och överlåtits. (Protokollet är en del av en större programhelhet. De andra delarna av programvaran behandlar datapaketen noggrannare och vårt protokoll sköter enbart mottagande av paket från andra programmoduler, förmedlar paket till datakommunikationskanalen och överlåter paket i den mottagande ändan till andra moduler.)

Låt oss i det följande studera med ett scenario hur protokollet funktionerar om ett datapaket förloras. I detta och följande scenarier lämnas get- och put-meddelandena bort för tydlighetens skull. Vi bör dock komma ihåg att alltid när ett nytt paket sänds, måste det först tas från bufferten med en get-operation. Likaså när ett nytt paket tas emot, överlåts det först till applikationen med ett put-meddelande och först efter det sänds kvitteringen. Om ett meddelande med samma nummer kommer två gånger efter varandra, så är det andra en duplicering, och det överlåts inte mera en andra gång till applikationen.



I scenariot försvinner det första datapaketet $d0$. Mottagaren B fortsätter att vänta, men i något skede utlöses A:s timeout, eftersom ingen kvittering kommer. A skickar samma paket $d0$ på nytt och nu går det fram till B. B kvitterar, kvitteringen kommer fram utan fel och vi är igen i ett normalt läge och A kan fortsätta med att skicka ett nytt datapaket med nummer 1. En situation där kvitteringen förloras är analog. A väntar på kvitteringen som inte kommer, och A:s timeout utlöses. A skickar det föregående datapaketet på nytt. Om det nu kommer fram utan fel till B, noterar B med hjälp av numret att det är samma gamla paket och överför det inte på nytt, men skickar kvitteringen på nytt. Om den kommer fram utan fel, fortsätter situationen igen normalt med försändelsen av nästa datapaket.

Låt oss ännu titta på ett scenario där datapaketet förloras och genast efter det förloras också kvitteringen. Protokollet återhämtar sig även från denna situation med hjälp av timeouten:



I scenariot förloras det första datapaketet $d0$, så ingen kvittering kommer och A:s timeout utlöses. A sänder datapaketet $d0$ på nytt och denna gång kommer det fram. Dock förloras kvitteringen $a0$ som hör till paketet. Således utlöses timeouten på nytt och datapaketet $d0$ sänds på nytt. Även detta datapaket kommer fram utan fel. B noterar att samma paket kommit på nytt, överlåter det inte, men skickar kvitteringen. Nu kommer kvitteringen $a0$ fram utan fel och protokollet fortsätter normalt med att sända nästa paket.

Frågor

Fråga 2.1.
Visa med ett scenario hur protokollet sköter en situation där kvitteringen förvrängs. Lägg med också get- och put-meddelandena i detta och följande scenarier. (maximalt antal poäng 2)

Fråga 2.2.
Visa med ett scenario och därtillhörande skriftlig förklaring att protokollet fungerar inkorrekt om meddelandena inte numreras. (maximalt antal poäng 3)

Fråga 2.3.
Vi antar att timeouten utlöses fastän kvitteringen är på kommande till A utan fel. Visa med ett scenario att protokollet återhämtar sig också från denna situation. (maximalt antal poäng 4)

Fråga 2.4.
Visa med ett scenario att protokollet fungerar inkorrekt om kanalen förutom att förlora meddelanden och duplicera meddelanden också kunde ändra på meddelandenas ordning i kanalen. (maximalt antal poäng 4)

Vänd!

Fråga 2.5.
Vi antar att protokollet används för att förflytta en fil på 10 megabyte från maskin A till maskin B. Avståndet mellan maskinerna är 4000 km och signalens hastighet i kanalen är (för enkelhetens skull) 200000 km/s. Kanalen kan sända med hastigheten 10 megabitar i sekunden. Hur länge tar det att sända filen, när vi använder alternerande bitars protokoll och filen delas upp i paket på 1000 byte och dessa skickas separat. Kvitteringarnas storlek är 1000 bitar. Kanalen fungerar utan fel.

Ett byte är 8 bitar, ett kilobyte är (för enkelhetens skull) 1000 byte, ett megabyte är 1000 kilobyte. (maximalt antal poäng 5)

Fråga 2.6.
Liksom i föregående fråga, men nu förlorar kanalen 10% av meddelandena, av vilka hälften är datameddelanden och hälften kvitteringar. Timeouten utlöses efter 5 sekunder, om kvitteringen inte kommit fram. (maximalt antal poäng 5)

Fråga 2.7.
Beskriv vilka faktorer som bör beaktas vid inställningen av timeoutens tidsgräns, då vi har som målsättning en effektiv timeout, som inte utlöses för tidigt, men inte heller blir och väntar onödigt länge. (maximalt antal poäng 2)