Industriautomation

Tentamen SSY 065, onsdag 20/12, 08:30-12:30, VV

Examinator: Martin Fabian, (772) 3716

Tider för lärarens närvaro: 09:30, 11:30

Fullständig lösning ska lämnas på samtliga uppgifter. I förekommande fall av tvetydigt formulerade tentamensuppgifter ska den föreslagna lösningen och eventuella antaganden motiveras. Examinator förbehåller sig rätten att godkänna rimligheten i antaganden och motiveringar.

Totalt omfattar tentamen 25 poäng. För betygen tre, fyra och fem krävs 10, 15 resp 20 poäng.

Lösningar anslås 21/12 på kursens hemsida i Studieportalen.

Tentamensresultaten anslås senast 8/1 på institutionens anslagstavla, kl. 12:30.

Granskning av rättningen får ske 8/1, 9/1 kl. 12:30 – 13:30 på institutionen.

OBS. Inga hjälpmedel är tillåtna.

Uppgift 1. Flödessimulering

a. Vad är syftet med flödessimulering? (2p)

Testa:

_ Olika layouter

_ Olika processtider

_ Olika egenskaper

_ Eliminera flaskhalsar

_ Kassationer

Dvs variera parametrar som skall undersökas enl problemformulering / Detaljeringsgrad

Nya problemområden kan uppdagas

- Skapar ordning och reda
- Pekar åt rätt riktning
- Tidssparande

- Beslutsstöd (utveckling /styrningen)
- Samlingsplats för idéer/kommunikation
- Övning/träning.

b. Vad menas med en *warmup*? (1p)

- Används för att ta bort data som inte är relevant för helheten
- Används inte i system med återkallande status; ex bank, service centraler
- Warm-up är den tid det tar för modellen att ”svänga” in sig (Steady state)

c. Vilka är de huvudsakliga medlen man har för att påverka resultatet av en flödessimulering och förklara på vilket sätt dessa påverkar resultatet? (2p)

Användning av buffrar vilket leder till att osäkerheter i en maskin inte påverkar en efterföljande lika mycket.

Arbeta med batcher vilket gör att man minskar omställningstider.

Titta på vilken rutt produkterna tar vilket gör att maskiner som lämpar sig bäst för en viss produkt används

Uppgift 2. Robotprogrammering och simulering

a. Förklara vad ett *workobjekt* är och vad det används till. (2p)

Ett workobject är ett koordinatsystem som exempelvis kan läggas i hörnet på ett bord eller inne i en fixtur. Det finns alltid ett world workobject, wobj0. Anledningen till att jobba med andra workobjects än wobj0 är att man kan vilja gruppera flera targets. De placeras då i ett gemensamt workobject. Om man sen flyttar objektet flyttas alla targets med, vilket gör att roboten fortfarande kan hitta alla dessa targets. Exempelvis kan det vara lämpligt att ha ett workobject för häftapparaten, eftersom den då enkelt skulle kunna flyttas, utan att man behöver ändra koordinaterna för de tre häftpunkterna.

b. Förklara vad ett *target* är. (1p)

Ett target är en koordinat med en position och en orientering (samt en konfigurering), exempelvis en hemmaposition för en robot.

c. Förklara vad en *path* är. (1p)

En path är en sekvens av rörelseinstruktioner (t.ex. MoveL, MoveJ) längs vilken roboten rör sig till sina targets.

d. Beskriv för- och nackdelar med offline programmering gentemot onlineprogrammering. (2p)

Fördelar:

Minimerat robotstillestånd

Kraftfulla simuleringsmöjligheter

Ett programmeringssystem för olika robottillverkare

Undviker kollisioner vilket medför bättre arbetsmiljö för operatörerna.

Nackdelar:

Kräver kalibrering

Dyra

Inlärningssvårigheter

Uppgift 3. Kommunikation

a. Förklara relationen mellan de olika protokollen TCP, UDP och IP. (3p)

TCP/IP is (currently) the protocol that makes the Internet tick. Though there exists one protocol acronymed TCP, and another acronymed IP, TCP/IP actually names a protocol stack. The stack includes the *transmission control protocol*, TCP, and the *internet protocol*, IP, as well as a number of other well-known protocols, such as the *hypertext transfer protocol*, HTTP, and the *Ethernet* access protocol. A number of other well-known protocols are typically (see Figure 1) also considered to belong to the TCP/IP protocol suite. These are the *simple mail transfer protocol*, SMTP, the *real-time protocol*, RTP, the *domain name* system, DNS, and the *user datagram protocol*, UDP. These latter all are application and transport layer protocols that run over IP, see Figure 1.

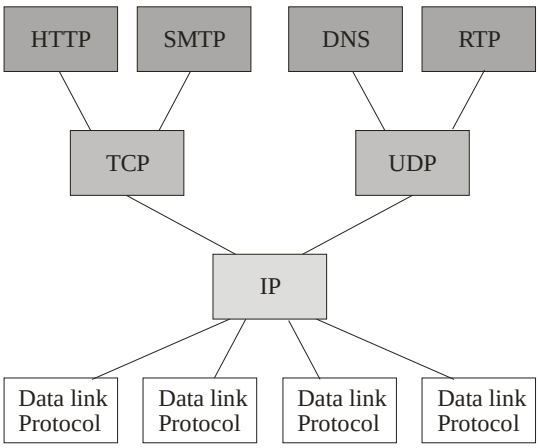


Figure 1. **TCP/IP protocol graph.**

IP is concerned with getting packets from one computer to another, as opposed to TCP and UDP that concern themselves with getting the packets from the sender to the receiver. The packet may have to pass through many computers on the way from the sender to the receiver, and IP figures out which computer to send the packet to next. This is known as a *hop*, and TCP/IP transmission may include many, many hops between the sender and the receiver. Figuring out which hop to make next (to which computer) is known as *routing*, and this is the main concern of IP. Thus, IP is also *connectionless*.

IP is an unreliable protocol. This means that lost or distorted packets are not resent; there are no guarantees that a sent packet actually reaches the intended receiver. Using IP, the TCP protocol implements a *reliable* transmission, guaranteeing that the sent data actually reches the receiver. This TCP does by using acknowledgments and having the possibility to ask for retransmission of lost packets. At the receiver end, TCP puts the packets together in the correct order so that the actual data is recovered.

In contrast to TCP, UDP does not care about reliability. UDP ads just a minimal overhead on the IP packet, and does not sequence the data or retransmit. UDP does also not chop the data up into manageable pieces, like TCP does, but rather expects the application to hand over the data in appropriate chunks. The aim of UDP is to provide sender to receiver streaming of data, such as real-time information (video, music, sensor-signals). In these cases a lost packet or two is not crucial, since a new one will soon arrive and the earlier data will be outdated anyway.

b. Beskriv *token ring/bus* metoden för access av det fysiska mediet. (2p)

Från kursboken, sid 44.

In the *Token Ring* (IEEE 802.5) access method a *token* is sent from node to node, and the node owning the token may transmit data. After it has transmitted the data, the token is passed on. The line is, thus, always free and there are no collisions.

Token Ring has the benefit that it is deterministic in the sense that if we know the number of nodes and we know the time it takes to pass a token, then we can calculate an upper bound on the time it takes for a node to transmit its data. However, as the number of nodes increases, so does this upper bound. Furthermore, a node can actually have to wait for exactly that time before being able to transmit its data.

The token ring protocol is also simple to implement in wireless networks. As long as a node can communicate with its nearest neighbors, it can pass on the token, and no “hidden station problem” can occur, as in the case of the Ethernet.

Uppgift 4. SFC programmering

Vi har två maskiner M_1 och M_2 som använder ett gemensamt verktyg. Maskinerna tar åt sig verktyget automatiskt, men bara då de har fått tillåtelse av styrenheten (PLCn). Respektive maskin frågar styrenheten om lov genom att sätta en signal a_i ( ) hög. Styrenheten svarar då maskinen genom att i sin tur sätta en signal b_i hög. Då tar den maskin som fick tillåtelse verktyget och använder det tills den är färdig, varefter den lämnar tillbaka det och meddelar styrenheten att den är klar med verktyget. Detta gör den genom att sätta en tredje signal, c_i hög. Observera att vi *inte* är garanterade att a_i och c_i för en maskin M_i aldrig är höga “samtidigt” (ur PLCns synpunkt).

Maskinernas ”protokoll” kan alltså beskrivas enlig följande.

1. Använder ej verktyget.
2. Begär verktyg, sätt a_i.
3. Få tillgång, vänta på b_i.
4. Använd verktyget.
5. Lämna tillbaka verktyget, sätt c_i.
6. Gå till 1.

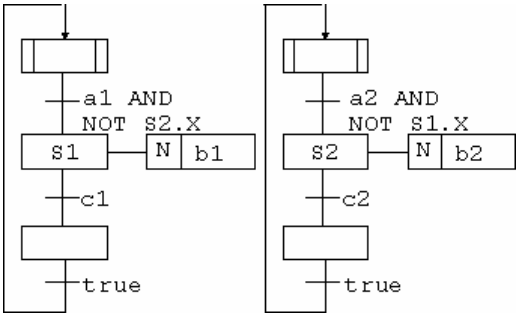
Initialt använder ingen av maskinerna verktyget. Notera att vi inte vet nånting om vilken av maskinerna som begär verktyget först.

Formulera ett funktionsdiagram, IEC 61131-3 SFC, som styr maskinernas utnyttjande av verktyget så att

1. båda maskinerna aldrig tillåts använda verktyget samtidigt, och att
2. den ena maskinen inte hela tiden får tillgång till verktyget på bekostnad av den andra.

Förklara också under vilka förutsättningar din kod fungerar. (5p)

Det kan hända att PLCn inte kan observera vilken av maskinerna som begär verktyget först. Detta, tillsammans med att en maskin kan lämna tillbaka verktyget och sen omedelbart begära det igen (dvs ha både a_i och c_i höga samtidigt ur PLCns perspektiv), gör att man måste garantera att man inte alltid låter den ena maskinen få tillgång till verktyget på bekostnad av den andra (man måste undvika "svält"). Ett sätt är följande, vilket förutsätter "immediate transit"-tolkning av funktionsdiagramexekveringen.



Om a1 och c1 är satta "samtidigt" tillsammans med a2, kan M2 släppas emellan när sekvensen för M1 befinner sig i det tomma steget. Notera att vi inte kan garantera att maskinerna alltid får tillgång till verktyget i den ordning de begär det; scan-cykeln medför att vi kan se a1 och a2 satta samtidigt utan att kunna urskilja i vilken ordning detta skett.

Uppgift 5. PLC, teori

a) Beskriv de huvudsakliga delarna av en modern PLC. (2p)

I/O-enhet (med in- och utgångsmoduler, plus buffertminne), processor (CPU), programminne (RAM), operativsystemminne (ROM).

b) Förklara den principiella idén bakom *in/utsignalskopiering* som (så gott som) alla moderna PLCer utför? (2p)

Alla insignalers värde kopieras till buffertminnet, hela programmet exekveras med insignalernas sparade värde och utsignalernas nya värde sparas i utsignalsbufferten. Därefter kopieras utsignalbuffertens innehåll till de egentliga utsignalerna. Detta gör för att simulera det parallella beteendet som reläkopplingar och fast trådad logik uppvisar. Från en utomstående betraktare verkar det som om alla utsignaler ändrar sina värden beroende på insignalerna (och interna minnen), samtidigt.