Task: Printing

Note: This page is converted with a scanner, so it might contain small errors.

Client and Server

There are two users, each having a computer. The computers are identified by their names: CLIENT(1) and CLIENT(2). The two computers share one or more printers named SERVER(1), SERVER(2) and so on. The print jobs from both computers can only be executed in succession. To coordinate the communication between both computers with a printer we will use a special object: a semaphore.

Semaphore

Each printer has one related semaphore. A semaphore is in one of two states: S1 or S2. When the printer is free to receive a print job, the semaphore is in state S1. As long as the printer is busy with executing a print job the semaphore is in state S2.

A semaphore can make two kinds of state transitions: `S1->S2' and `S2->S1'. When a user sends a print job to the computer, the computer will send a message "Are_you_open?" to the semaphore. If the state of the semaphore is S1, then the state of the semaphore will change to S2 and the semaphore will send a message "Open" to the computer that sent the message "Are_you_open?". If the state of the semaphore is S2 then the semaphore will send back a message: "Closed". After finishing a print job, the printer will send a message "Ready" to the semaphore. Upon receiving a message "Ready", the semaphore will change its state to S1.

Object type SEMAPHORE

In **Documentation 1** you will find the specification of the object type SEMAPHORE. The specification includes the possible identifiers and the states of an object of the type SEMAPHORE, the `Priority List', the `Communication Diagram', the `State Transition Diagram', and the 'Receive Procedures' for the messages: "Ready" and "Are_you_open?". The `Receive Procedures' describe how a semaphore responds to a message.

The `Priority List' is necessary because messages that are received at the same time by a semaphore can only be handled in succession. The `Priority List' in **Documentation 1** indicates that each message of a SERVER has a higher priority than a message of a CLIENT and that for instance a message of SERVER(2) has a higher priority than a message of SERVER(3).

Object type CLIENT

In **Documentation 2** you will find the specification of the object type CLIENT. An object of the type CLIENT is in one of three states: SA, SB or SC. A client is in state SA when this client did not send a print job to a server and the servers are not busy with a print job of this client. A client which is in state SB, wants to have access to a server. The client can only get access to a server via a semaphore. A client is in state SC when a server is executing a print job of this client.

A client can make three kinds of state transitions: `SA->SB', `SB->SC', `SC->SA'. When a client is in state SB, the client can receive a message "Closed" from the semaphore. After receiving this message the client waits during a period, the 'Waiting_Period', before the client will again send a message "Are_you_open?" to the semaphore. When the semaphore sends a message "Open" to the client, this client changes its state to SC and sends the print job with a message "S_Job" to the server related to the sending semaphore. This server then executes the print job. After finishing the print job, the server will send two messages at the same time, a message "Ready" to its semaphore and a message "C_Ready" to the client. This client then changes its state from SC to SA. You can assume that the printers are ideal printers. They will finish each received print job. For instance there is never the situation `out of paper'.

Communication

In **Documentation 3** you find in the `Communication Structure Diagram' all the message types that can be sent between the different object types. In the `Message List' you find the specification of each message type. Each message has an identifier, a sender, a receiver, and sometimes a content.

When a sender sends a message with the identifier A at time t, then the receiver will at time t+1 process the message by executing the `Receive Procedure' A.

If several senders send messages to the same receiver at time t, then the receiver will process all those messages at time t+1, in the same order in which the senders of these messages appear in the receiver's `Priority List'.

Subtask A

A Local Area Network (LAN) includes the following objects at time 0:
Object: CLIENT(1), Client.State = SA, Waiting_Period = 2, Number_of_Servers= 1

Object: CLIENT(2), Client.State = SA, Waiting_Period = 1, Number_of_Servers= 1

Object: SERVER(1)

Object: SEMAPHORE(1), Semaphore.State = S1

In this LAN there have been sent, among others, the following messages:

At time 1:

CLIENT(1) sends a message with the identifier "Are_you_open?"

At time 2:

CLIENT(2) sends a message with the identifier "Are_you_open?"

At time 4:

SERVER(1) sends a message with the identifier "Ready"

At time 5:

CLIENT(1) sends a message with the identifier "Are_you_open?"

Documentation 4 indicates for SEMAPHORE(1), CLIENT(1) and CLIENT(2) in a time table up to time 6, which messages these objects receive, which messages they send, and in which state they are or to which state they change.

Question A.1
What would have happened if CLIENT(1) receives a message "C_Job" at time 4? Write your answer in **Documentation 5**.

Question A.2
What would have happened if CLIENT(2) receives a message "C_Job" at time 4? Write your answer in **Documentation 5**.

Question A.3
Complete the timetable in **Documentation 4** up to time 13 if the following happens:

At time 8:

SERVER(1) sends a message with the identifier "Ready".

At time l0:

CLIENT(1) receives a message with the identifier "C_Job".

At time 12:

SERVER(1) sends a message with the identifier "Ready".

Subtask B

The LAN is extended. It now includes two semaphore-server pairs: (SEMAPHORE(1), SEMAPHORE(2), SERVER (1), SERVER(2)). For each CLIENT the Number_of_Servers equals 2.

To use both printers a change of the definition of the object type CLIENT described in **Documentation 2** is necessary.
In **Documentation 6** you find the changed `Receive Procedures': C_Job and Wait.
In **Documentation 6** you also find a description of the situation at time 0.
At the following times the following messages are received:
At time 0:

CLIENT(1) receives a message "C_Job" of a user.

At time 0:

CLIENT(2) receives a message "C_Job" of a user.

At time 4:

SEMAPHORE(1) receives a message "Ready".

What happens in the extended LAN according to the changed object type CLIENT?
Mark the correct answer in **Documentation 6**.

Subtask C

The definition change of the object type CLIENT in subtask B was not efficient for an extended LAN with more than one semaphore-server pair.

Subtask C.1
Change the definition of the object type CLIENT (see **Documentation 2**) so that the LAN with more than one semaphore-server pair can function as follows:
An object CLIENT(1) may use any of the servers in the LAN, but CLIENT(i) can only have one print job executed at a time by the servers. Each print job of a client is printed only once.
An object CLIENT(i) sends a message "Are_you_open?" successively to the several semaphores until a certain threshold value of received messages "Closed" has been reached or until the object CLIENT(i) receives one message "Open".

When the object CLIENT(i) reaches the threshold value, the object CLIENT(i) waits during a certain Waiting_Period before it again sends a series of messages "Are_you_open?".
Write your solution in **Documentation 7**.

Subtask C.2
Change the object type CLIENT in such a way, that object CLIENT(i) now may have several print jobs executed at a time by several servers, too. However, the number of print jobs of each CLIENT(i) should be limited to CLIENT(i).Job_Maximum.
Write your solution in **Documentation 8**.

Documentation 1

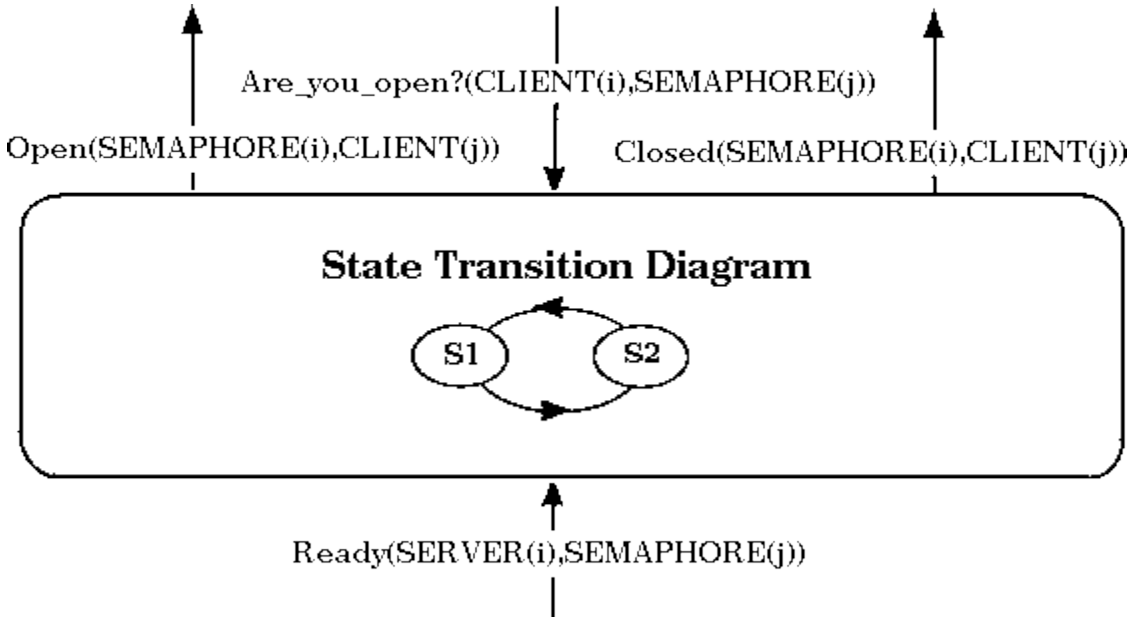
Object type: SEMAPHORE

Possible Identifiers: (SEMAPHORE(1),SEMAPHORE(2),SEMAPHORE(3),...)

State: (S1,S2); initial state is S1.

Priority List: SERVER(1),SERVER(2),...,CLIENT(1),CLIENT(2),...

Communication Diagram



Receive Procedures

```
procedure Are_you_open?(Client,Semaphore)
begin
   if    State = S1
   then  State <- S2
         Send("Open(Semaphore,Client)")
   else
   if    State = S2
   then  Send("Closed(Semaphore,Client)")
end


procedure Ready(Server, Semaphore)
begin
   State <- S1
end
```

Documentation 2

Object type: CLIENT

Possible Identifiers: (CLIENT(1),CLIENT(2),CLIENT(3),...)

State: (SA,SB,SC); initial state is SA.

Priority List: CLIENT,SERVER(1),SERVER(2),...,SEMAPHORE(1), SEMAPHORE(2),...,USER(1),USER(2),...

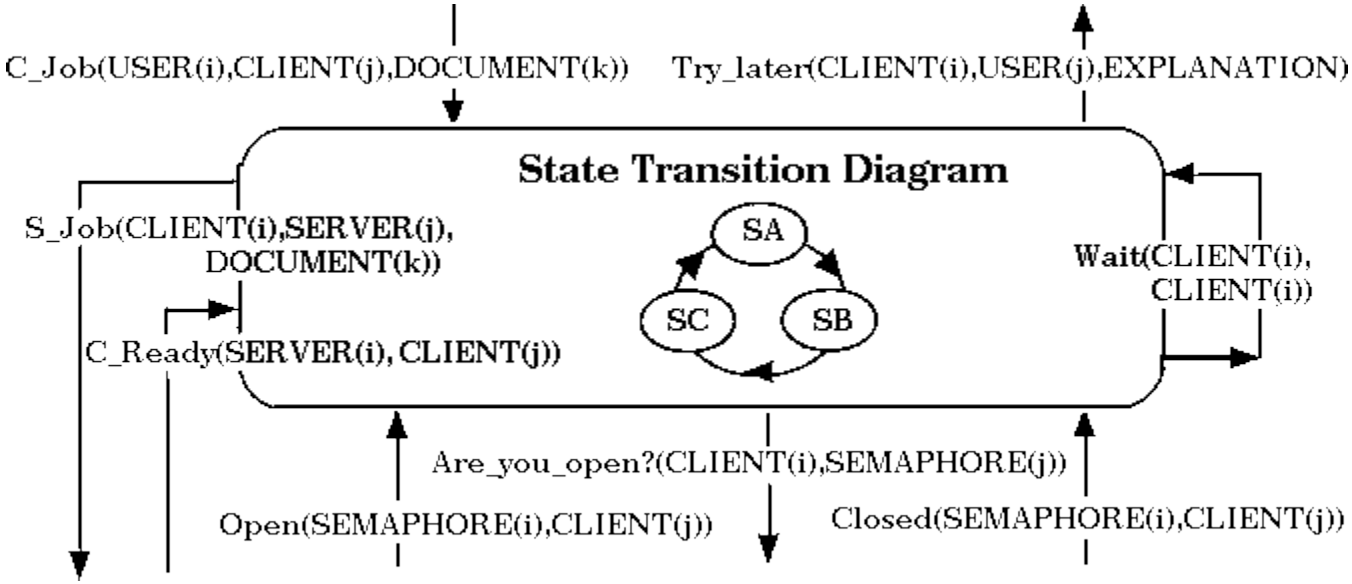
Countdown: { t | t in N }; initial value is 0.

Waiting_Period: { t | t in N and t > 0 }

Semaphore_Index: { i | i = 1,2,...,Number_of_Servers }

Number-of-Servers : { i | i in N and i > 0 }

Communication Diagram



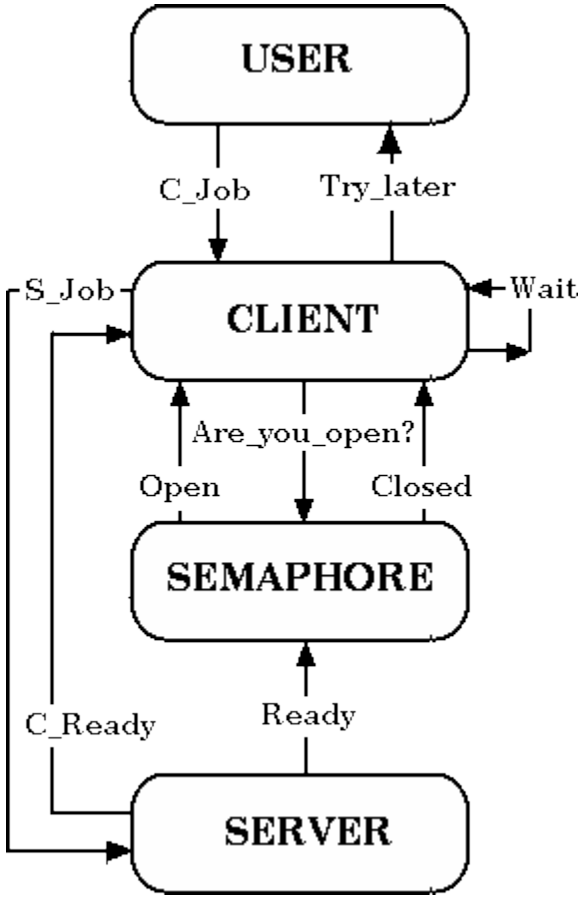
Receive Procedures

```
procedure C_Job(User,Client,Document)
begin
   if    State = SA
   then  State <- SB
         Send("Are_you_open?(Client,
         SEMAPHORE(Semaphore_Index))")
   else
   if    State = SB
   then  Send("Try_later(Client,User,
              Client_is_busy)")
   else
   if    State = SC
   then  Send("Try_later(Client,User,
            All_Servers_are_busy)")
end

procedure Open(Semaphore,Client)
```

```
begin
    if    State = SB
    then  State <- SC
          Send("S_Job(Client,Server,
                   Document)")
end

procedure Closed(Semaphore,Client)
begin
   Countdown <- Waiting_Period
   Send("Wait(Client,Client)")
end

procedure Wait(Client,Client)
begin
   Countdown <- Countdown - 1
   if    Countdown > 0
   then  Send("Wait(Client,Client)")
   else  Send("Are_you_open?(Client,
         SEMAPHORE(Semaphore_Index))")
end


procedure C_Ready(Server,Client)
begin
   State <- SA
end
```

Documentation 3

Communication Structure Diagram



Message List

(i,j,k in N)

identifier	sender	receiver	content
Are_you_open?	CLIENT(i)	SEMAPHORE(j)	-

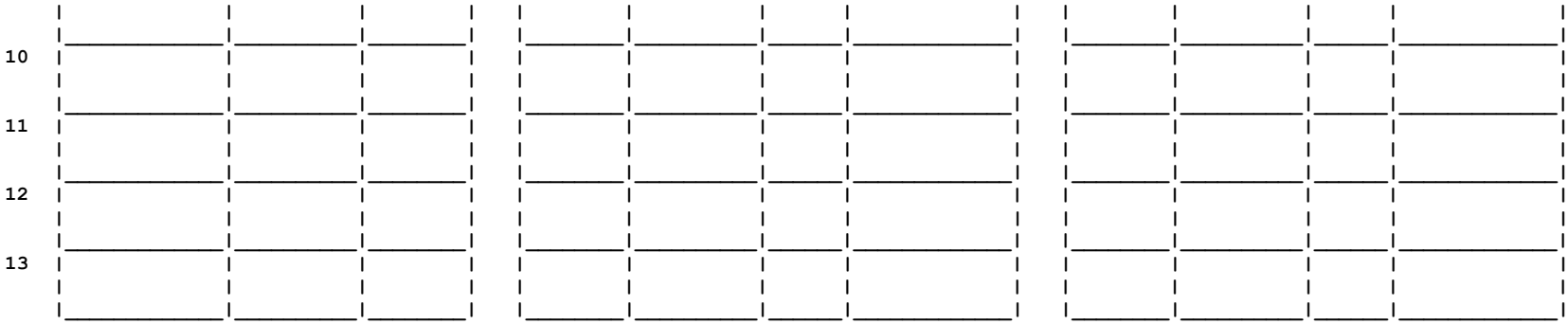
C_Job	USER(i)	CLIENT(j)	DOCUMENT(k)
C_Ready	SERVER(i)	CLIENT(j)	-
Closed	SEMAPHORE(i)	CLIENT(j)	-
Open	SEMAPHORE(i)	CLIENT(j)	-
Ready	SERVER(i)	SEMAPHORE(j)	-
S_Job	CLIENT(i)	SERVER(j)	DOCUMENT(k)
Try_later	CLIENT(i)	USER(j)	EXPLANATION
Wait	CLIENT(i)	CLIENT(i)	-

Documentation 4

Answer Subtask A.3

Note: This table should be printed in portrait mode.

time	SEMAPHORE(1) received messages	SEMAPHORE(1) State/Transition	SEMAPHORE(1) sent messages	CLIENT(1) received messages	CLIENT(1) State/Transition	CLIENT(1) Count-down	CLIENT(1) sent messages	CLIENT(2) received messages	CLIENT(2) State/Transition	CLIENT(2) Count-down	CLIENT(2) sent messages
0		S1			SA	0			SA	0	
1		S1		C_Job	SA->SB	0	Are_you_open?		SA	0	
2	Are_you_open?	S1->S2	Open		SB	0		C_Job	SA->SB	0	Are_you_open?
3	Are_you_open?	S2	Closed	Open	SB->SC	0	S_Job		SB	0	
4		S2			SC	0		Closed	SB	0->1	Wait
5	Ready	S2->S1		C_Ready C_Job	SC->SA SA->SB	0 0	Are_you_open?	Wait	SB	1->0	Are_you_open?
6	Are_you_open? Are_you_open?	S1->S2 S2	Open Closed								
7											
8											
9											



Documentation 5

Question A.1

What would have happened if CLIENT(1) receives a message "C_Job" at time 4 ? _____

_____
_____
_____
_____
_____
_____
_____

Question A.2

What would have happened if CLIENT(2) receives a message "C_Job" at time 4 ? _____

_____
_____
_____
_____
_____
_____
_____

Documentation 6

Subtask B

The changed `Receive Procedures': C_Job and Wait.

Note: The changed code is made italic in this page.

```
procedure C_Job(User,Client)
begin
   if     State = SA
   then   State <- SB
          for   Semaphore_Index <- 1 step 1 until Number_of_Servers
                Send("Are_you_open?(Client,SEMAPHORE(Semaphore_Index))")
   else
   if     State = SB
   then   Send("Try_later(Client,User,Client_is_busy)")
   else
   if     State = SC
   then   Send(,Try_later(Client,User,All_Servers_are_busy)')
end

procedure Wait(Client,Client)
begin
  if      State = SB
  then    Countdown <- Countdown -1
          if       Countdown > 0
          then     Send("Wait(Client,Client)")
          else
          if       Countdown < 0
          then     Countdown <- 0
          else     for  Semaphore_Index <- 1 step 1 until Number_of_Servers
                        Send("Are_you_open?(Client,SEMAPHORE(Semaphore_Index))")
end
```

At time 0 the situation in the LAN is as follows:
Object: CLIENT(1), Client.State SA, Waiting_Period = 2, Number_of_Servers=2
Object: CLIENT(2), Client.State SA, Waiting_Period = 1, Number_of_Servers=2
Object: SEMAPHORE(1), Semaphore.State = S1
Object: SEMAPHORE(2), Semaphore.State = S1
At the following times the following messages are received:
At time 0: CLIENT(1) receives a message "C_Job" of a user.

At time 0: CLIENT(2) receives a message "C_Job" of a user.
At time 4: SEMAPHORE(1) receives a message "Ready".
What happens in the extended LAN according to the changed object type CLIENT?

Mark the correct answer.

(a)

The print job of CLIENT(1) will be printed on SERVER(1) and SERVER(2).
The print job of CLIENT(2) will not be printed.

(b)

The print job of CLIENT(1) will be printed once on SERVER(1).
The print job of CLIENT(2) will be printed once on SERVER(2).

(c)

The print job of CLIENT(1) will be printed once on SERVER(1).
The print job of CLIENT(2) will be printed once on SERVER(1).

(d)

The print job of CLIENT(1) will be printed once on SERVER(2).
The print job of CLIENT(2) will be printed once on SERVER(2).

(e)

The print job of CLIENT(1) will not be printed.
The printjob of CLIENT(2) will be printed on SERVER(1) and SERVER(2).

Documentation 7

Subtask C.1

_____
_____
_____
_____
_____
_____
_____
_____

Documentation 8

Subtask C.2