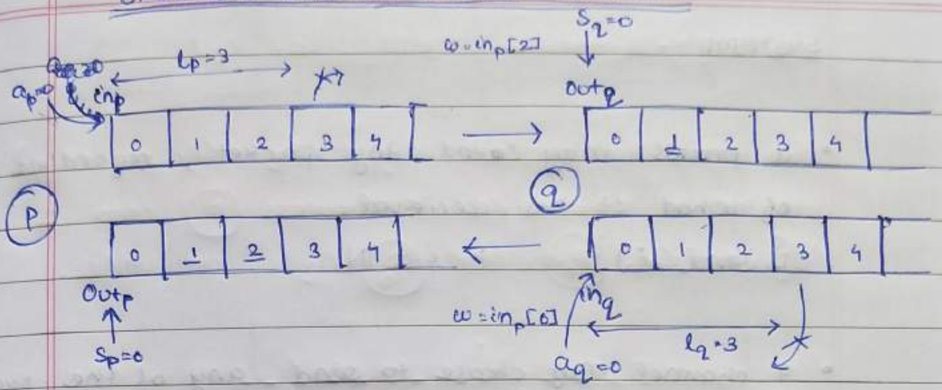
Balanced Sliding window

classmate
Date 12.2.2020
Page 1

BALANCED SLIDING WINDOW PROTOCOL



safe delivery:

$out_p[0 \ldots s_p-1] = in_q[0 \ldots s_p-1]$

$out_q[0 \ldots s_q-1] = in_q[0 \ldots s_q-1]$

eventual delivery:

$s_p \geq k, \quad s_q \geq k \quad \forall \; k \geq 0$

liveness:

$P \Rightarrow s_p - l_q \leq a_p \leq s_q \leq a_q + l_p \leq s_p + l_p$

$P \Rightarrow w = in_p[s_q]$ by p $\vee$ $w = in_q[s_p]$ by q is applic.
(no deadlock is possible).

protocol satisfies requirement of eventual delivery.

classmate
Date 12.2.2020
Page 2

SYSTEM

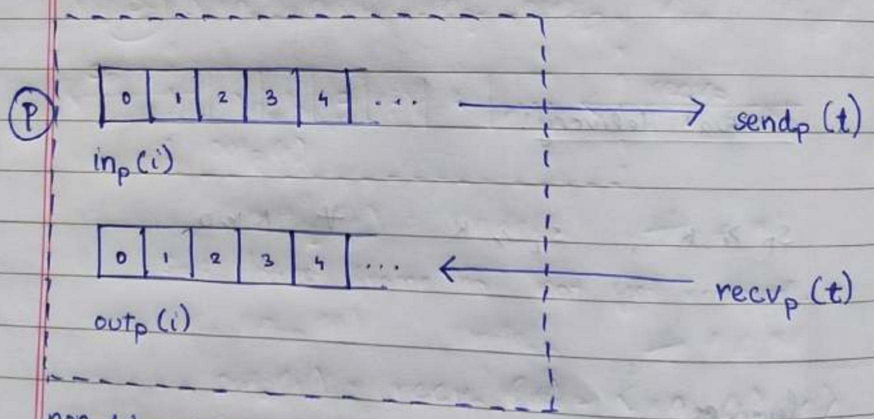
- a process may send any packet, regardless of what it has recieved.

$\{ send(i) \} \quad : i \in \mathbb{N}$

- a channel may choose to send any of the packets it has recieved at its input?

$\therefore C_{out} \in \{ in_p[i] \mid in_p[i] \text{ has been sent by } p \}$.

or, $out_c(i) = \{ in_c(j) \mid j < i \} \quad \forall\, i, j \geq 0$.



non-deterministic:

at $send_p(t)$ as it can choose which inp to send

i.e., $send_p(t) = in_p(i) \; : \; i \in \mathbb{N}$

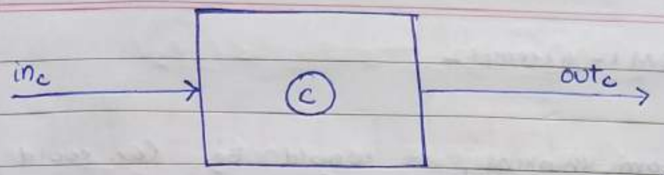
choice.

but, $recv_p(t) = out_p(j) \; : \; j \in \mathbb{N}$

no choice as

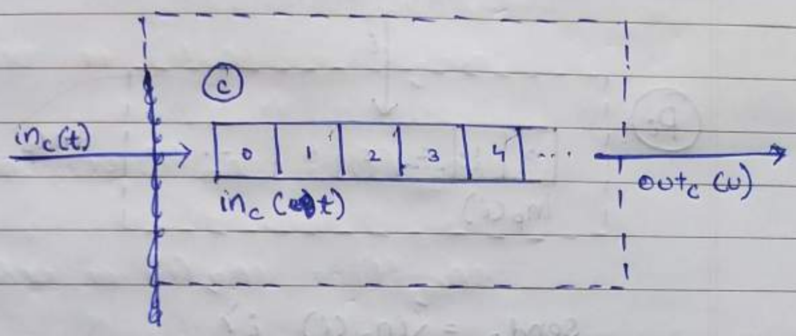
as $out_p(j) = in_q(j)$ choice

classmate
Date 12.2.2020
Page 3



choice (nd.)

$out_c(t) = in_c(u)$ s.t. $u < t$ $\quad t, u \geq 0$



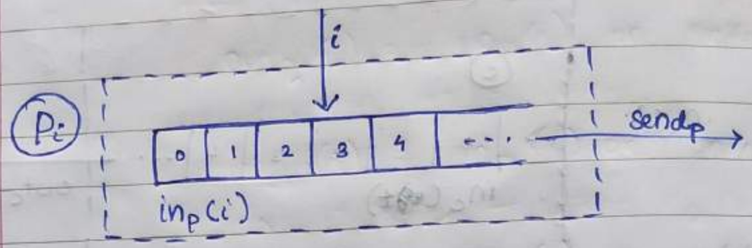
not allowing $out_c(u) = \phi$
as that could lead to no progress.

assumption: the channel transmits the message it recieved, as its, without any modification.

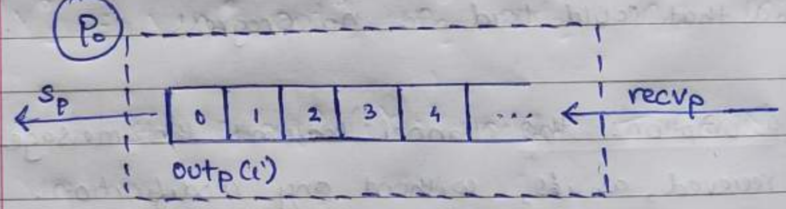
classmate
Date 12.2.2020
Page 4

SYSTEM 1 - attempt 2

now i am thinking, it would be (or could be) simpler to subdivide a process (P) into 2 subprocesses (P_i) and (P_o).



$$send_p = \langle in_p(i), i \rangle$$



s_p: lowest frame no. not yet recieved.

$$out_p(i) = recv_p = \langle in_q(j), i \rangle$$

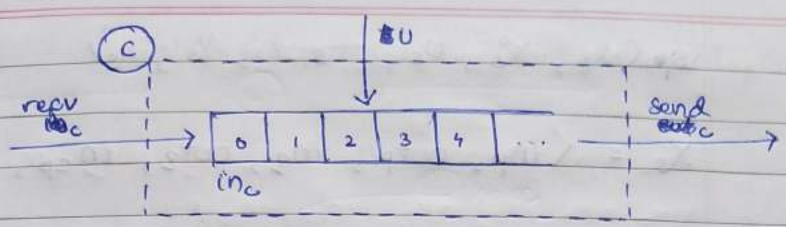
$$s_p = \min(\{ i \mid i \neq undef. \})$$

or

$$s_p(t) = \min(\{ i \text{ in } recv_p(t) = \langle -, i \rangle \})$$

$$= j \text{ st. } j \geq i \ \forall i \in recv_p(t) = \langle -, i \rangle$$

classmate
Date 12·2·2020
Page 5



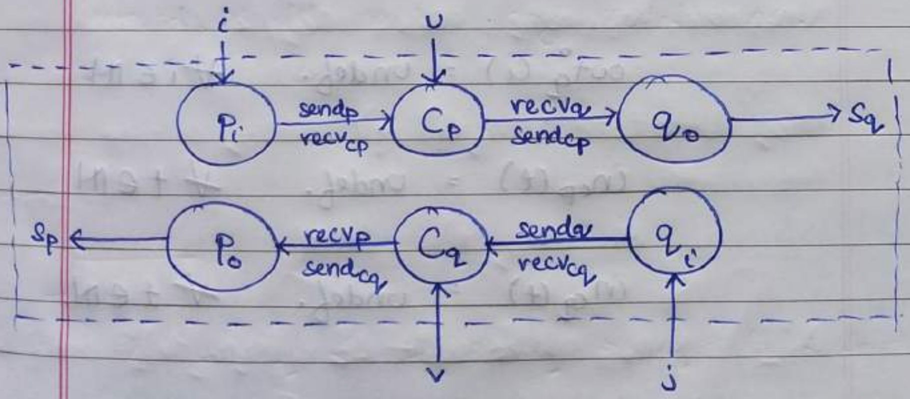
$out_c(t) = recv_c$

$send_c(t) = recv_c(u)$ st. $u < t$ $u, t \geq 0$.

or

$in_c(t) = recv_c(t)$

$send_c(t) = in_c(u)$ st. $in_c(u) \neq undef$.



4 inputs to interact with.

classmate
Date 12.2.2020
Page 6

$S_p \langle X_s, X_s^0, U_s, \xrightarrow{s}, Y_s, h_s \rangle$

$X_s = \langle in_p, out_p, in_q, out_q, in_{cp}, in_{cq} \rangle$

st. $out_p(i) = in_q(i)$ or undef.

$out_q(i) = in_p(i)$ or undef.

$in_{cp}(t) = in_p(i)$ or undef. $i \in \mathbb{N}$

$in_{cq}(t) = in_q(i)$ or undef. $i \in \mathbb{N}$.

$X_s^0 = \langle in_p^0, out_p^0, in_q^0, out_q^0, in_{cp}^0, in_{cq}^0 \rangle$

st. $out_p^0(i) =$ undef. $\forall i \in \mathbb{N}$

$out_q^0(i) =$ undef. $\forall i \in \mathbb{N}$

$in_{cp}(t) =$ undef. $\forall t \in \mathbb{N}$

$in_{cq}(t) =$ undef. $\forall t \in \mathbb{N}$.

classmate
Date 12.2.2020
Page 7

$U_s = \{ send_p(i),\ send_q(j),\ send_{cp}(u),\ send_{cq}(v) \}$

st. $in_p(i) \neq undef.$

$in_q(j) \neq undef.$

$in_{cp}(u) \neq undef.$

$in_{cq}(v) \neq undef.$

Objective:

$$out_p(j) = in_q(j) \quad \forall \ j \le k$$

$$out_q(i) = in_p(i) \quad \forall \ i \le k$$

$k \in \mathbb{N}$

K: frames to be transferred.

in_p: frames of process p

in_q: frames of process q

out_p: frames recieved by process p (of q).

out_q: frames recieved by process q (of p).

in_{cp}: frames sent through channel by process p
in_{cq}: frames sent through channel by process q

Date 12.2.2020
Page 8

(i) $\langle in_p, out_p, in_q, out_q, in_{cp}, \cancel{out} \, in_{cq} \rangle$

$$\Big\downarrow send_p(i)$$

$\langle in_p, out_p, in_q, out_q, in'_{cp}, in_{cq} \rangle$

$in'_{cp} = \{ t \rightarrow \langle in_p(i), i \rangle \} \, in_{cp}$

st. $in_{cp}(t) = undef.$

$\wedge \quad t \leq x \quad \forall \{ x \mid in_{cp}(x) = undef. \}$

(ii) $\langle in_p, out_p, in_q, out_q, in_{cp}, \cancel{out} \, in_{cq} \rangle$

$$\Big\downarrow send_q(j)$$

$\langle in_p, out_p, in_q, out_q, in_{cp}, in'_{cq} \rangle$

$in'_{cq} = \{ t \rightarrow \langle in_q(j), j \rangle \} \, in_{cq}$

st. $in_{cq}(t) = undef.$

$\wedge \quad t \leq x \quad \forall \{ x \mid in_{cq}(x) = undef. \}$

(iii) $\langle in_p,$

$\langle in_p,$

out'_q

(iv) $\langle in_p,$

$\langle in_p,$

out'_p

objec

we c

the c

Date 12.2.2020
Page 9

(iii) $\langle in_p, out_p, in_q, out_q, in_{cp}, out_{cp} \rangle$

$$\Big\downarrow send_{cp}(u)$$

$\langle in_p, out_p, in_q, out_q', in_{cp}, out_{cp} \rangle$

$out'_q = \{ i \rightarrow F \}\ out_q$

s.t. $i = i$ in $in_{cp}(u) = \langle F, i \rangle$

$F = F$ in $in_{cp}(u) = \langle F, i \rangle$

(iv) $\langle in_p, out_p, in_q, out_q, in_{cp}, in_{cq} \rangle$

$$\Big\downarrow send_{cq}(v)$$

$\langle in_p, out_p', in_q, out_q, in_{cp}, out_{cp} \rangle$

$out'_p = \{ j \rightarrow F \}\ out_q$

st. $j = j$ in $in_{cq}(v) = \langle F, j \rangle$

$F = F$ in $in_{cq}(v) = \langle F, j \rangle$

objective:
we want to eliminate the choice of 'i' in $send_p(i)$, and the choice of j in $send_q(j)$. this would make processes p & q deterministic.

classmate
Date 17.2.2020
Page 10

(2) $S_2 \langle X_{S2}, X^{o}_{S2}, U_{S2}, \xrightarrow{S2}, \rangle$

$X_{S2} = \langle in_p, out_p, S_p, l_p, in_q, out_q, in_{cp}, in_{cq} \rangle$

here, S_p represents that all frames till $S_p - 1$ have been recieved at out_p and that S_p is the first frame yet to be sent.

l_p represents the frames window for sending, implying that any frame $\geq S_p + l_p$ cannot be sent, and process p must ensure first that frame S_p is recieved at out_p, after which it can increment S_p, and then will be able to send the frame $S_{p_{old}} + l_p$.

(this is a constant)

the above repporesents additional 2 contrants that apply to the choice of $send_p(i)$. i is still choosable, but its choice must satisfy above constraints.

$X^{o}_{S2} = \langle in_p, out^{o}_p, s^{o}_p, l^{o}_p, in_q, out^{o}_q, in^{o}_{cp}, in^{o}_{cq}$

st. (additional)

$S^{o}_p = 0 \qquad S^{o}_q = 0$

$l^{o}_p = L_p$ (say 3) $\quad l^{o}_q = L_q$

classmate
Date 17.2.2020
Page 21

$U_{S_2} = U_{S_1}$

(i) $\langle in_p, out_p, s_p, l_p, in_q, out_q, s_q, l_q, in_{cp}, in_{cq} \rangle$

$\big\downarrow send_p(i)$

$\langle in_p, out_p, s_p, l_p, in_q, out_q, s_q, l_q, in'_{cp}, in_{cq} \rangle$

iff $i \geq s_p \wedge i < s_p + l_p$

$\therefore$ same as for S_1

(ii) $\langle in_p, out_p, s_p, l_p, in_q, out_q, \ldots \rangle$

$\big\downarrow send_q(j)$

$\langle \ldots, in'_{cq} \rangle$

iff $s_q \leq i < s_q + l_q$

$\therefore$ Same as for S_1.

(iii) $\langle \ldots \rangle$

$\big\downarrow send_{cp}(v)$

$\langle \ldots, out'_q, s'_q, \ldots \rangle$

$s'_q = v$ iff $v > s_q$

$\wedge\ out'_q(i) \neq undef.\ \forall i < v.$

$\therefore$ Same as for S_1

Date 17.2.2020
Page 12

(iv) $\langle \ldots \rangle$
$\downarrow send_{cq}(v)$
$\langle \ldots, out'_p, s'_p, \ldots \rangle$

$s'_p = v$ iff $v > s_p$
$\wedge\ out'_p(i) \neq undef.\ \forall\ i < v.$

$\therefore$ same as for S_1

(3) $S_3 \langle X_{S3}, X^0_{S3}, U_{S3}, \xrightarrow{S3}, \rangle$

$X_{S3} = \langle in_p, out_p, s_p, l_p, f_p, in_q, out_q, s_q, l_q, f_q, in_{cp}, in_{cq}$

f_p represent the frame that was last sent.
$X^0_{S3}: f_{p_0} = -1.$

$U_{S3} = \{send_p, send_q, \underbrace{send_{cp}(v), send_{cq}(v)}\}$

channel is still non deterministic.

(i) $\langle \ldots \rangle$
$\downarrow send_p$
$\langle \ldots f'_p, \ldots, in'_{cp}, \ldots \rangle$

$in'_{cp} = send_p(i)$ where $i = \max(f_p + 1, s_p)$
$f'_p = i$

Date 17.2.2020
Page 13.

(ii) similarly for send$_2$.

(iii) (iv) same as for S_2.

RTP:

① safety (all frames are recieved as is, in order).

② eaventuality (atleast k frames can be recieved by both process with a finite no. of steps).

classmate

Date 7.4.2020
Page 1

THE BALANCED SLIDING WINDOW PROTOCOL

DEFINITIONS

2 processes (p) and (q), each sending an infinite array of words to the other.

FOR PROCESS P:

in_p: an infinite array of words to be sent to process q.

out_p: an infinite array of words being recieved from process q.

initially $\forall i \quad out_p[i] = \phi$

S_p: the lowest number word that p still expects to recieve from q.

at any time, p has already written $out_p[0]$ through $out_p[i]$.

Date 7.4.2020
Page 2

REQUIRED PROPERTIES

SAFE DELIVERY

- in every reachable configuration of the protocol.

$out_p[0 \ldots s_p - 1] = in_q[0 \ldots s_p - 1]$, and

$out_q[0 \ldots s_q - 1] = in_p[0 \ldots s_q - 1]$

EVENTUAL DELIVERY

- for every integer $k \geq 0$, a configuration with

$s_p \geq k$ and $s_q \geq k$

is eventually reached.

classmate
Date 7.4.2020
Page 3

THE PROTOCOL

- the packet $\langle pack, w, i \rangle$, transmits the word $w = in_p[i]$ to q.

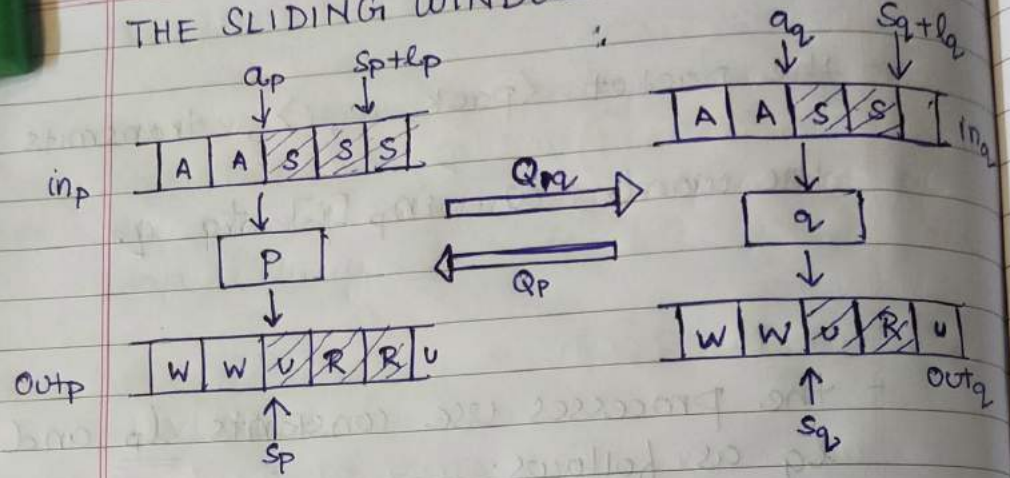
- the processes use constants $\underline{\underline{l_p}}$ and $\underline{\underline{l_q}}$ as follows:

• process p can send the word $w = in_p[i]$ (as $\langle pack, w, i \rangle$) only after storing all the words $out_p[0] \dots out_p[i-l_p]$ $(i < s_p + l_p)$

• when p recieves $\langle pack, w, i \rangle$ retransmission of $in_p[0] \dots in_p[i-l_p]$ words is no longer necessary.

classmate
Date 7.4.2020
Page 4

THE SLIDING WINDOWS



THE PROTOCOL

S_p: $\{ a_p \le i < s_p + l_p \}$
begin send $\langle pack, in_p[i], i \rangle$ to q end

R_p: $\{ \langle pack, w, i \rangle \in Q_p \}$
begin recieve $\langle pack, w, i \rangle$;
  if $out_p[i] = udef$ then
  begin $out_p[i] = w$;
    $a_p = max(a_p, i - l_q + 1)$
    $s_p = min(j \mid out_p[j] = udef)$
  end
  // else ignore - retransmission
end

L_p: $\{ \langle pack, w, i \rangle \in Q_p \}$
begin $Q_p = Q_p \setminus \{ \langle pack, w, i \rangle \}$ end

classmate
Date 7.4.2020
Page 5

PROTOCOL INVARIANT

$$P = \forall i < s_p : out_p[i] \neq udef$$

$$\wedge \ \forall i < s_q : out_q[i] \neq udef$$

$$\wedge \ \langle pack, w, i \rangle \in Q_p \Rightarrow$$

$$w = in_q[i] \ \wedge \ (i < s_q + l_q)$$

$$\wedge \ \langle pack, w, i \rangle \in Q_q \Rightarrow$$

$$w = in_p[i] \ \wedge \ (i < s_p + l_p)$$

$$\wedge \ out_p[i] \neq udef \Rightarrow$$

$$out_p[i] = in_q[i] \ \wedge \ (a_p > i - l_q)$$

$$\wedge \ out_q[i] \neq udef \Rightarrow$$

$$out_q[i] = in_p[i] \ \wedge \ (a_q > i - l_p)$$

$$\wedge \ a_p \leq s_q$$

$$\wedge \ a_q \leq s_p$$

classmate
Date 7.4.2020
Page 6

RESULTS

SAFETY

the protocol satisfies the requirement of safe delivery.

LIVENESS

- P implies $s_p - l_p \le a_p \le s_q \le a_p + l_p \le s_p + l_p$
- P implies that the sending of $\langle pack, in_p[s_q], s_q \rangle$ by p or the sending of $\langle pack, in_q[s_p], s_p \rangle$ by q is applicable.

hence no deadlock is possible.

- the protocol satisfies the requirement of eventual delivery.

classmate
Date 7.4.2020
Page

ELM DESIGN

MODEL?

$n = 6$

psend	→	7	1	3	4	2	8	int
qsend	→	-7	-1	-3	-4	-2	-8	
precv	→	0	0	0	0	0	0	
qrecv	→	0	0	0	0	0	0	
ps	→	F	F	F	F	F	F	bool
qs	→	F	F	F	F	F	F	
pr	→	F	F	F	F	F	F	
qr	→	F	F	F	F	F	F	
cprecv	→	0	0	0	0	0	0	int
cqrecv	→	0	0	0	0	0	0	
cpr	→	F	F	F	F	F	F	bool
cqr	→	F	F	F	F	F	F	

classmate
Date 7.4.2020
Page 2

TOO MANY FIELDS?

could be better to group propeties / fields by:

- process (separate types)
- channel.

PROCESS TYPE

- send - sent? or just s.
- recv - recieved? (rcvd) or just r
- also, ω $\rightarrow$ window size

$\therefore\ a_p = \min\{i \mid s[i] = \text{false}\}$ nextSend

$s_p = \min\{i \mid r[i] = \text{false}\}$. nextRecv

$l_p = \omega$.

classmate
Date 7.4.2020
Page 3

CHANNEL TYPE

- recv - rcvd /r

also if we force the channel to send each recieved value atleast once,

or set a maximum duplicate send limit, we can guarentee eventual delivery.

also need to guarentee safety, i.e, whatever data has been sent, is recieved as is,

but this should be straightforward as the channel (we assume) does not modify any data passing through it.

it may choose, at a certain step, not to send a recieved packet, but it may not change the content of the packet.

- sent /s an integral sent field could be used to count the no. of times a packet has been retransmitted by a channel.

classmate
Date 7.4.2020
Page 4

TAKING TURNS

how should the processes and the channel take turns in submitting an action to the model?

if arbitrary turn taking is allowed, it is possible that the channel ~~..~~ may not recieve any turns (or not sufficient enough), and we will not be able to guarentee eventual delivery.

a possible solution is to give turns to each process, and the channels one after the other, in a cycle, like in a turn-based game. this had been discussed in the class. (the plant and the controller taking alternate turns).