The Balanced Sliding Window Protocol

CS60002: Distributed Systems

Pallab Dasgupta
Professor,
Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur

Definitions

- Two 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 received from process q

  Initially for all i, $out_p[i]$ = *udef*

  s_p : The lowest numbered word that *p* still expects to receive from *q*

  At any time, *p* has already written $out_p[0]$ through $out_p[s_p - 1]$

Required Properties

- ***Safe delivery***:
  - In every reachable configuration of the protocol

$$out_p[0 \ldots s_p - 1] = in_q[0 \ldots s_p - 1] \quad \text{and}$$
$$out_q[0 \ldots s_q - 1] = in_p[0 \ldots s_q - 1]$$

- ***Eventual delivery***:
  - For every integer $k \geq 0$, a configuration with $s_p \geq k$ and $s_q \geq k$ is eventually reached

The protocol

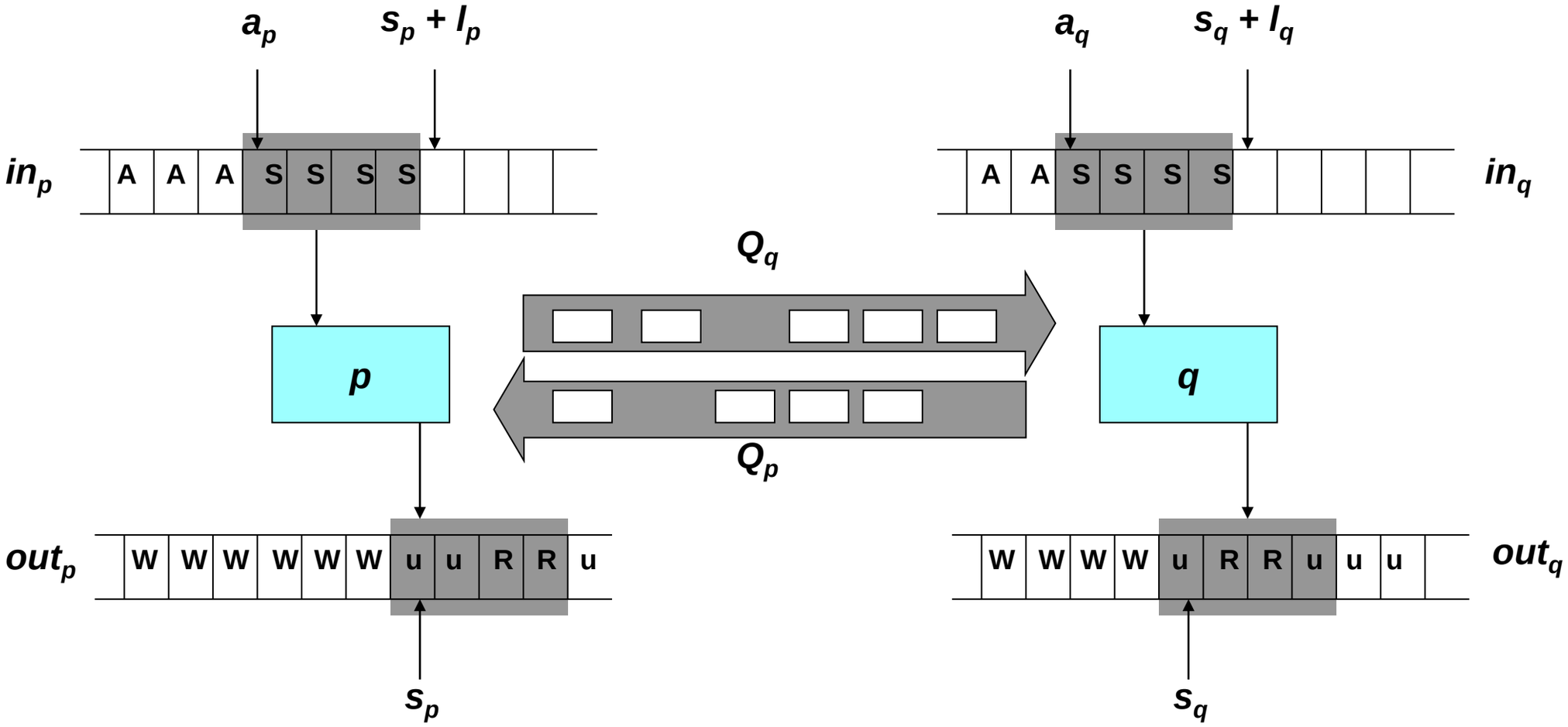
- The packet, < pack, *w*, *i* > , transmits the word $w = in_p[\, i \,]$ to *q*.

- The processes use constants l_p and l_q as follows:

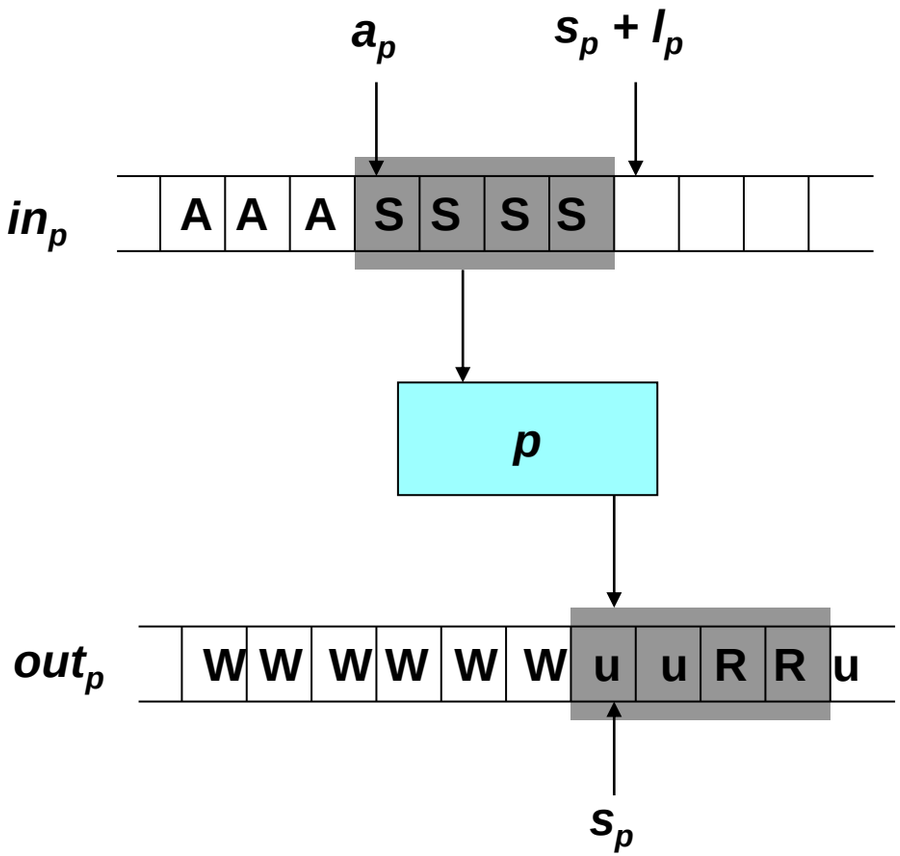
  - Process *p* can send the word $w = in_p[\, i \,]$ (as the packet, < pack, *w*, *i* >) only after storing all the words $out_p[0]$ through $out_p[\, i - l_p \,]$, that is, $i < s_p + l_p$.

  - When *p* receives < pack, *w*, *i* >, retransmission of words from $in_p[0]$ through $in_p[\, i - l_q \,]$ is no longer necessary.

The Sliding Windows



The Protocol



S_p: $\{ a_p \le i < s_p + l_p \}$

begin send < pack, $in_p[i]$, i > to q end

R_p: $\{ < pack, w, i > \in Q_p \}$

begin receive < pack, w, i > ;

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: $\{ < pack, w, i > \in Q_p \}$

begin $Q_p = Q_p \setminus \{ < pack, w, i > \}$ end

Protocol Invariant

$$
\begin{aligned}
P \equiv \quad & \forall i < s_p : out_p[i] \neq udef \\
\wedge \; & \forall i < s_q : out_q[i] \neq udef \\
\wedge \; & < \text{pack}, w, i > \in Q_p \Rightarrow w = in_q[i] \wedge (i < s_q + l_q) \\
\wedge \; & < \text{pack}, w, i > \in Q_q \Rightarrow w = in_p[i] \wedge (i < s_p + l_p) \\
\wedge \; & out_p[i] \neq udef \Rightarrow out_p[i] = in_q[i] \wedge (a_p > i - l_q) \\
\wedge \; & out_q[i] \neq udef \Rightarrow out_q[i] = in_p[i] \wedge (a_q > i - l_p) \\
\wedge \; & a_p \leq s_q \\
\wedge \; & a_q \leq s_p
\end{aligned}
$$

Results

Safety: The protocol satisfies the requirement of safe delivery

Liveness:

- P implies $s_p - l_q \leq a_p \leq s_q \leq a_q + l_p \leq s_p + l_p$
- P implies that the sending of <pack, $in_p[s_q]$, s_q> by p or the sending of <pack, $in_q[s_p]$, s_p> by q is applicable.
  - *Hence no deadlock is possible*
- The protocol satisfies the requirement of eventual delivery