

---

---

# Balanced and Timer Based Sliding Window

12 May 2010  
Lecture 9

Slide credits: R. H. Mak

---

# Topics for Today

---

- Balanced Sliding Window Protocol
- Timer-based Sliding Window Protocol

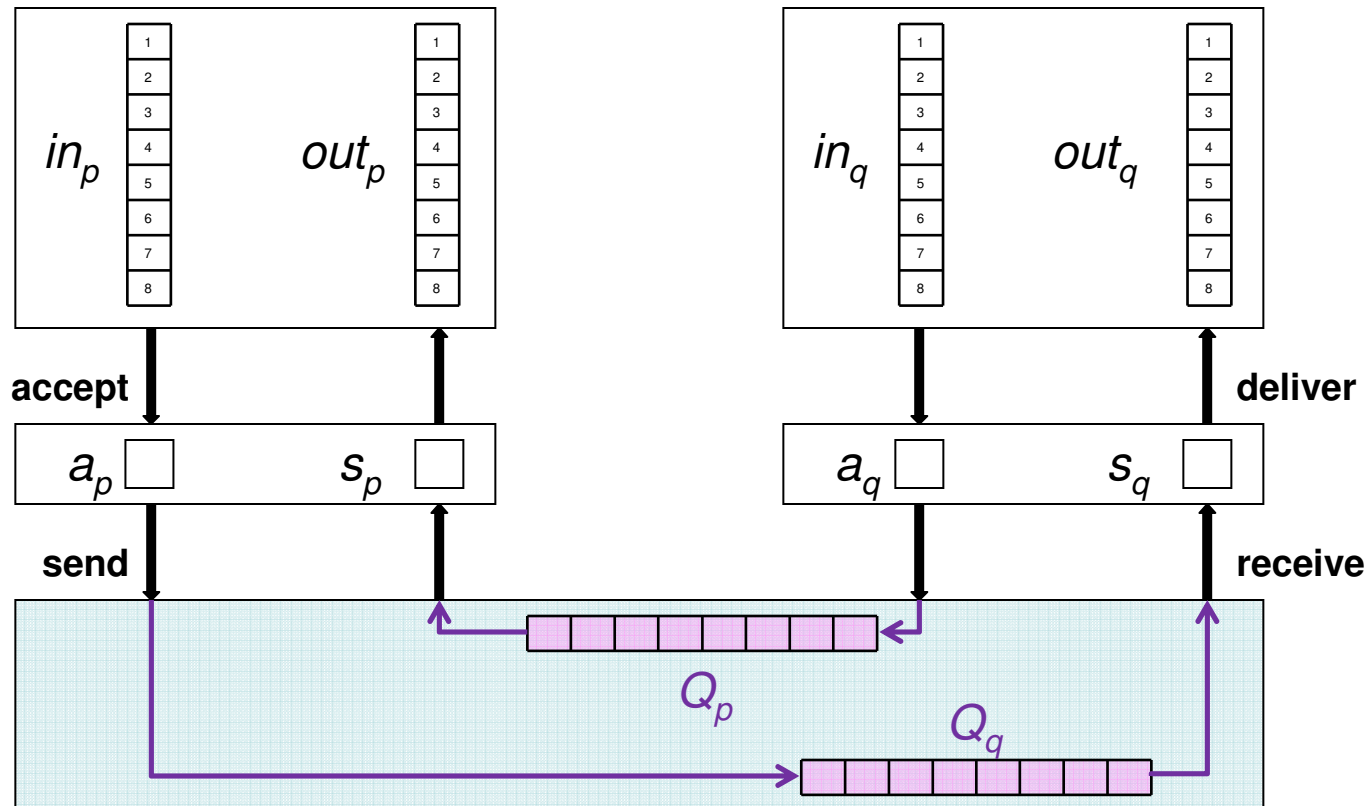
Source: Tel 3.1, 3.2

# Balanced Sliding-Window Protocol

---

- Assumptions
  - Processes are directly connected (data-link layer protocol)
    - or virtual channel on transport layer (TCP)
  - **bidirectional** connections
  - error model: **loss** (garbling = loss)
  - FIFO property:
    - channel  $pq$  is modeled by a pair of queues  $Q_p$  and  $Q_q$
- Requirements
  - Safety
    - all delivered data is correct
  - Liveness
    - all data is eventually delivered

# Protocol Model



1.  $in_p$ : infinite array of words to be delivered by  $p$
2.  $out_p$ : infinite array of words to be accepted by  $p$
3.  $a_p$ : the lowest numbered word not yet acknowledged by  $q$
4.  $s_p$ : the lowest numbered word not yet received from  $q$

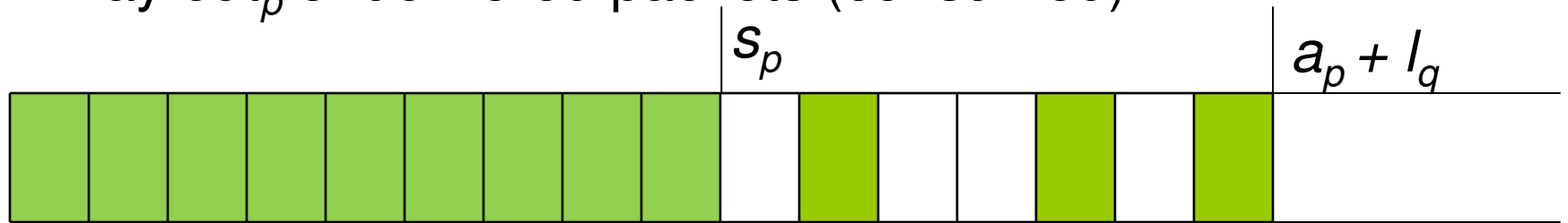
# BSWP characteristics

---

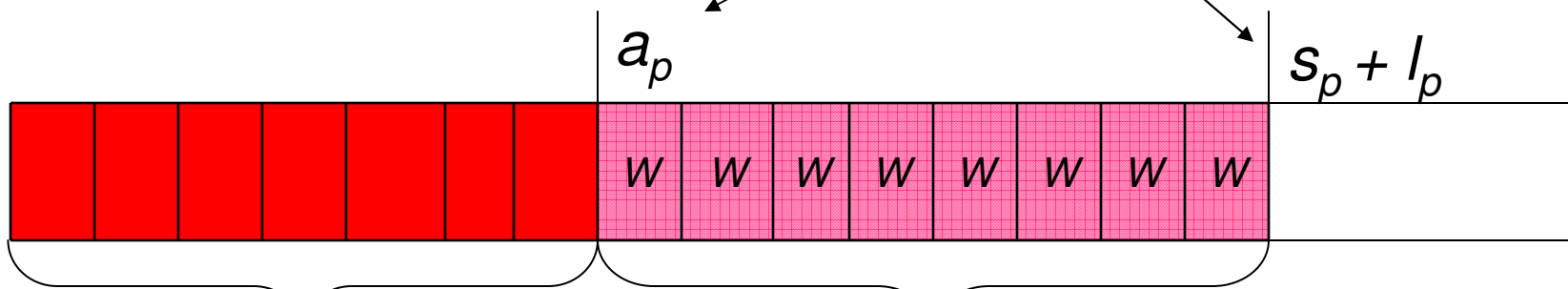
1. The protocol is **symmetric**
2. Messages used **both for data transmission and acknowledgement**
  - sending  $\langle \mathbf{pack}, w, i \rangle$  transmits data word  $w = in_p[i]$
  - sending  $\langle \mathbf{pack}, w, i \rangle$  acknowledges words numbered  $0 .. i - l_p$
3. Words that have not been acknowledged are **retransmitted indefinitely and in a non-deterministic way**, *i.e.*, the number and order of retransmissions is chosen arbitrarily (but some fairness is required) .
4. Process  $p$  can send  $in_p[i]$  only if  $i < s_p + l_p$
5. Process  $p$  stops retransmitting words from  $in_p[0 .. i - l_q]$  upon receipt of  $\langle \mathbf{pack}, w, i \rangle$

# BSWP State

Array  $out_p$  of delivered packets (consumed)



Array  $in_p$  of accepted packets (produced)



Delivery acknowledged by  $q$

Send window

# BSWP Program Text

---

$S_p: \{ a_p \leq i < s_p + l_p \}$   
**begin** send < **pack**,  $in_p[i]$ ,  $i$  > to  $q$  **end**

$R_p: \{ \langle \text{pack}, w, i \rangle \text{ in } Q_p \}$   
**begin** receive < **pack**,  $w$ ,  $i$  >;  
    **if**  $out_p[i] = \text{undef}$  **then**  
        **begin**  $out_p[i] := w$ ; (\*deliver  $w^*$ )  
             $a_p := \max ( a_p, i - l_q + 1 )$ ;  
             $s_p := \min \{ j \mid out_p[j] = \text{undef} \}$ ;  
        **end**  
**end**

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

# Invariants BSWP

---

**var**  $s_p, a_p$  : integer      **init** 0,0;  
 $in_p$  : array of words    (*\* Data to be sent \**);  
 $out_p$  : array of words **init** *undef, undef, ...*;

**invariants**

$0p : \forall i < s_p : out_p[i] \neq undef$

$0q : \forall i < s_q : out_q[i] \neq undef$

$1p : \forall w, \forall 0 \leq i : \langle \text{pack}, w, i \rangle \in Q_p \Rightarrow (w = in_q[i]) \wedge (i < s_q + l_q)$

$1q : \forall w, \forall 0 \leq i : \langle \text{pack}, w, i \rangle \in Q_q \Rightarrow (w = in_p[i]) \wedge (i < s_p + l_p)$

$2p : \forall 0 \leq i : out_p[i] \neq undef \Rightarrow (out_p[i] = in_q[i]) \wedge (a_p > i - l_q)$

$2q : \forall 0 \leq i : out_q[i] \neq undef \Rightarrow (out_q[i] = in_p[i]) \wedge (a_q > i - l_p)$

$3p : a_p \leq s_q$   $(a_p = i - l_q + 1) \text{ AND } (i < s_q + l_q) \rightarrow a_p \leq s_q$

$3q : a_q \leq s_p$

# Invariants BSWP

$S_p: \{ a_p \leq i < s_p + l_p \}$   
**begin** send < **pack**,  $in_p[i]$ ,  $i$  > to  $q$  **end**

$R_p: \{ \langle \text{pack}, w, i \rangle \text{ in } Q_p \}$   
**begin** receive < **pack**,  $w$ ,  $i$  >;  
 (\*  $w = in_q[i] \wedge (i < s_q + l_q)$  \*)  
**if**  $out_p[i] := undef$  **then**  
**begin**  
      $a_p := \max(a_p, i - l_q + 1);$   
     (\*  $w = in_q[i] \wedge a_p > i - l_q$  \*)  
      $out_p[i] := w;$   
     **while**  $out_p[s_p] \neq undef$  **do**  
          $s_p := s_p + 1;$   
**end**  
**end**

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

$0p: \forall i < s_p : out_p[i] \neq undef$

$0q: \forall i < s_q : out_q[i] \neq undef$

$1p: \forall w \forall 0 \leq i : \langle \text{pack}, w, i \rangle \in Q_p \Rightarrow$

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

$1q: \forall w \forall 0 \leq i : \langle \text{pack}, w, i \rangle \in Q_q \Rightarrow$

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

$2p: \forall 0 \leq i : out_p[i] \neq undef \Rightarrow$

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

$2q: \forall 0 \leq i : out_q[i] \neq undef \Rightarrow$

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

$3p: a_p \leq s_q$

$3q: a_q \leq s_p$

# Safe Delivery

---

**Thm.** In every reachable configuration of the protocol

1.  $out_p [ 0 \dots s_p - 1 ] = in_q [ 0 \dots s_p - 1 ]$
2.  $out_q [ 0 \dots s_q - 1 ] = in_p [ 0 \dots s_q - 1 ]$

**Proof.**

- Invariants (0p) and (2p) imply (1)
- Invariants (0q) and (2q) imply (2)

**Remark.** Note that deadlock is possible if

$$l_p = 0 \text{ and } l_q = 0$$

# Deadlock analysis

---

How did this happen? Actions  $S_p$  and  $S_q$  are both blocked

→ { preconditions do not hold }

$$(a_p \geq s_p + l_p) \wedge (a_q \geq s_q + l_q)$$

→ { Invariants (3p) and (3q) }

$$(a_p \geq a_q + l_p) \wedge (a_q \geq a_p + l_q)$$

→ { transitivity }

$$a_p \geq a_p + l_p + l_q$$

→

$$0 \geq l_p + l_q$$

**Conclusion:** If  $l_p + l_q > 0$ , then in any infinite computation either  $S_p$  or  $S_q$  is applicable (enabled) infinitely often

# Fairness Assumptions

---

F1 If the sending of a packet is applicable for an infinite long time, the packet is sent infinitely often.

- Usually implemented using timers
- Retransmit timer of TCP

F2 If the same packet is sent infinitely often, it is received infinitely often.

- Minimal requirement a channel must satisfy in order that it can be used for data transmission

# Eventual Delivery

---

**Thm.** For each integer  $k \geq 0$ , a configuration with  $s_p \geq k$  and  $s_q \geq k$  is eventually reached.

**Proof.** Consider an infinite computation  $C$  with maximal values  $\sigma_p$  and  $\sigma_q$  for  $s_p$  and  $s_q$  respectively. Let  $\gamma_m$  be the first configuration of  $C$  where these values are reached.

Then for all  $\gamma_i$  with  $i \geq m$  sending either  $msg = \langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$  or  $msg = \langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$  is applicable.

By F1 it follows that  $msg$  is sent infinitely often

By F2 it follows that  $msg$  is received infinitely often

Hence either  $s_p := \sigma_p + 1$  or  $s_q := \sigma_q + 1$  is executed.

# Balanced protocol

---

1.  $s_p - l_q$   
 $\leq \{ \text{Invariants (0p) and (2p) with } i = s_p - 1 \}$
2.  $a_p$   
 $\leq \{ \text{invariant (3p)} \}$
3.  $s_q$   
 $\leq \{ \text{invariants (0q) and (2q) with } i = s_q - 1 \}$
4.  $a_q + l_p$   
 $\leq \{ \text{invariant (3q)} \}$
5.  $s_p + l_p$

$$(i=s_p-1) \text{ AND } (a_p + l_q > i) \rightarrow a_p \geq s_q - l_q$$

Hence:

$$|s_q - s_p| \leq \max(l_p, l_q)$$

This implies that in the limit the effective transmission rate of both processes is the same!

# Bounded window sizes

---

---

From the computation on the previous slide

1.  $s_p - l_q \leq a_p$  (lines 1 & 3)
2.  $a_p \leq s_p + l_p$  (lines 2 & 5)

---

The size of the send window  
at process  $p$  is given by

$$s_p + l_p - a_p$$

$$s_p + l_p - a_p$$

$$\leq \{ \text{(a) above} \}$$

$$s_p + l_p + l_q - s_p$$

$$\leq \{ \text{calc.} \}$$

$$l_p + l_q$$

The size of the receive  
window at process  $p$  is  
given by  $a_p + l_p - s_p$

$$a_p + l_p - s_p$$

$$\leq \{ \text{(b) above} \}$$

$$s_p + l_p + l_q - s_p$$

$$\leq \{ \text{calc.} \}$$

$$l_p + l_q$$

# So Far

---

- Balanced Sliding Window Protocol
- Timer-based Sliding Window Protocol

# Timer-based protocol

---

- End-to-end protocol (transport protocol)
- Connection management
- Resilient against
  - Loss, duplication, reordering errors
- Ordered delivery within bounded time or reported loss
- Simplified Fletcher Watson  $\Delta t$ -protocol
  - One direction (data flow)
  - Single-word packets
  - Single-word receiving window
  - Simplified timing assumptions

# Time and timer assumptions

---

- Global time
  - Each event takes place at a certain moment in global time, not observable by the processes
  - Each event is instantaneous (has duration zero)
    - The time event included!?
- Bounded packet life time
  - A packet sent at  $\tau = \sigma$  and any of its duplicates, must be received before  $\tau = \sigma + \mu$  or becomes lost
- Timers
  - Processes cannot observe absolute time, but have access to timers  $Xt^{(t_1)} - Xt^{(t_2)} = t_2 - t_1$

# Protocol Description

---

1. A connection is opened
  - by the sender, when a word is accepted and no connection exists
  - by the receiver upon receipt of a word, when no connection exists
2. A connection is closed
  - by the sender, upon time-out after the last send action, provided all packages sent have been acknowledged (or reported lost, see point 7)
  - by the receiver, upon time-out after the last delivery of a word
3. A packet is lost in transmission
  - when the packet life time is exceeded
4. A packet is duplicated
  - At a random moment during transmission
  - A duplicate has the same remaining life time as the original!

# Protocol Description

---

## 5. Packet (re)transmission

1. The sender keeps track of the number of words accepted and delivered during the connection.
2. The sender (re-)transmits an accepted word that is not yet delivered only during a finite time interval after its acceptance.
3. The receiver keeps track of the lowest numbered that has not been delivered in the current connection.
4. The receiver discards (does not deliver) words that arrive out of order.
5. The receiver repeatedly notifies the sender of its last delivery (by sending acknowledgements) as long as the connection remains open.

## 6. Timer values are chosen such that an acknowledgement cannot be received when a connection is closed

# Protocol Description

---

7. Before a connection is closed all accepted words that have not been delivered are reported lost to the producer.
  1. The producer must decide whether it wants to open a new connection to retransmit the words that have not been delivered in previous connection.

# Constants and Network State

---

## Network Constant:

$\mu$  : real; (\* Maximum packet lifetime \*)

## Protocol Constants:

$U$  : real; (\* Length of send interval \*)

$R$  : real; (\* Receiver time-out value:  $R \geq U + \mu$  no duplicates can arrive \*)

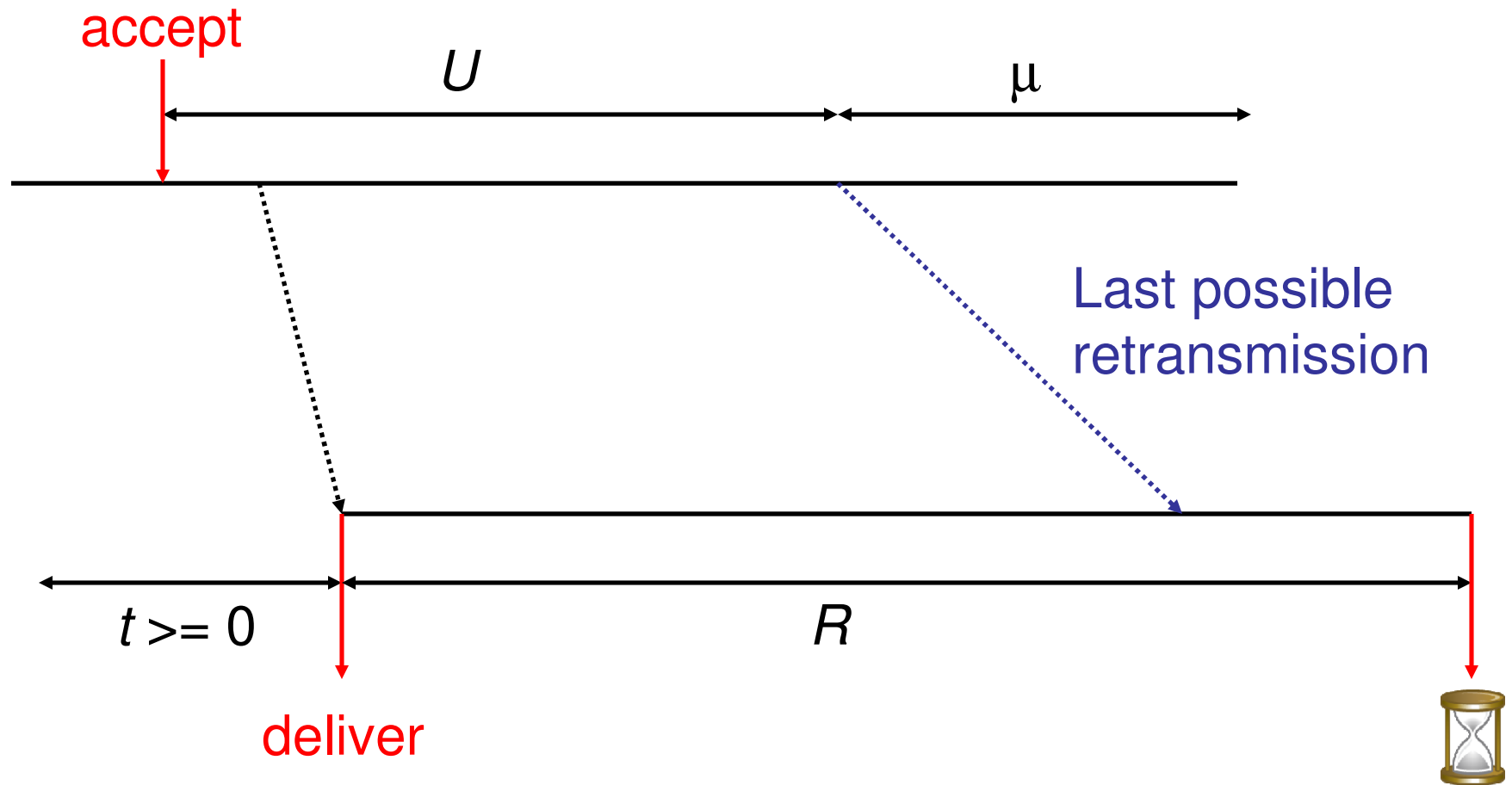
$S$  : real; (\* Sender time-out value:  $R \geq R + 2\mu$  no acknowledgement can arrive \*)

## Communication Subsystem;

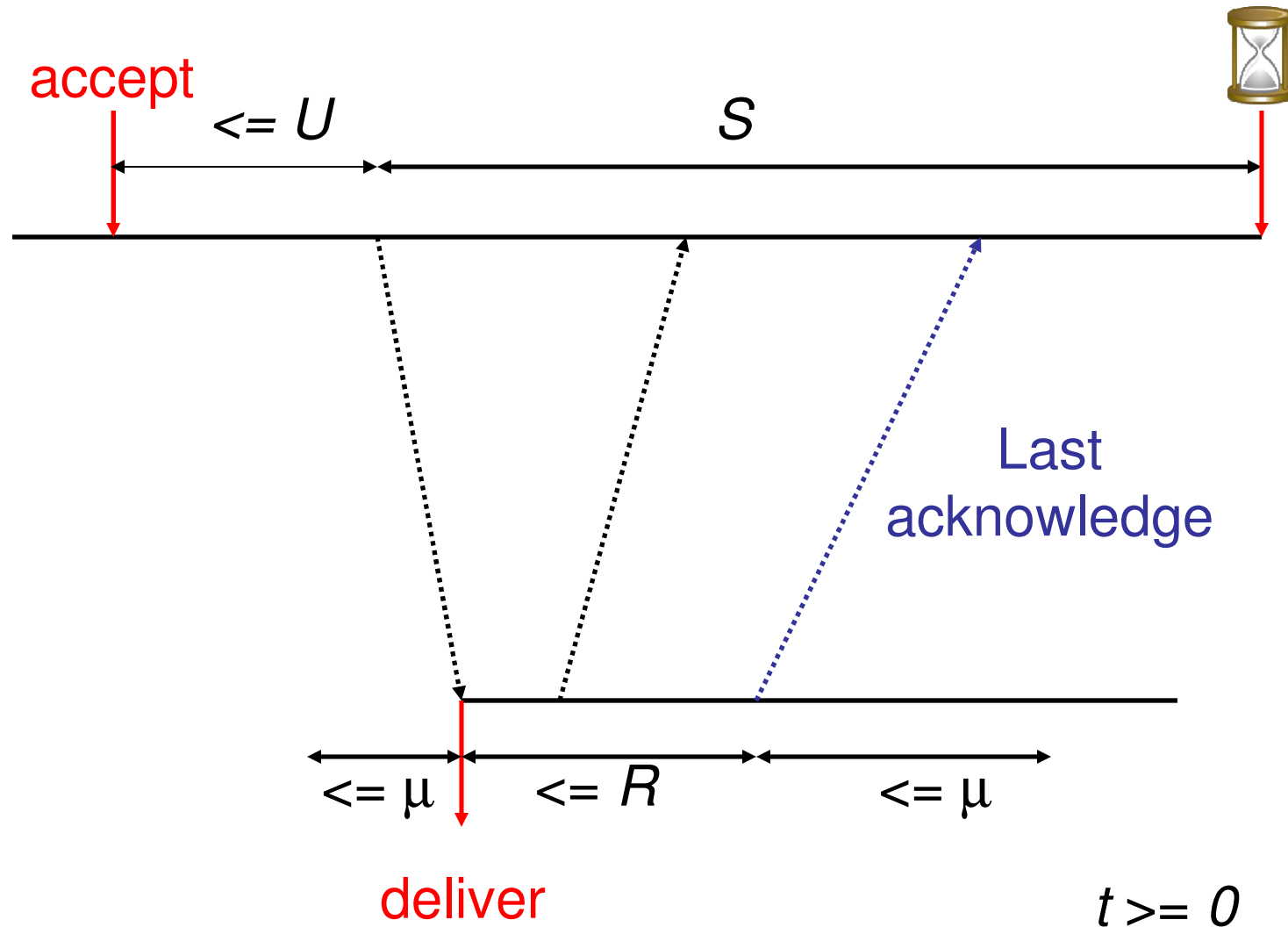
$M_q$  : channel; (\* Data packets for  $q$  \*)

$M_p$  : channel; (\* Acknowledge packet for  $p$  \*)

# Reason for $R \geq U + \mu$



# Reason for $S \geq R + 2\mu$



# Process State

---

Connection record of sender:

*Low* : int; (\* Acknowledged words current connection \*)

*High* : int; (\* Accepted words current connection \*)

*St* : timer (\* Connection timer \*)

Connection record of receiver:

*Exp* : int; (\* Next expected sequence number \*)

*Rt* : timer; (\* Connection timer \*)

Auxiliary variables:

*B* : int **init** 0; (\* Words in previous connections \*)

*cr* : boolean **init** *false* (\* Connection exists at receiver \*)

*cs* : boolean **init** *false* (\* Connection exists at sender \*)

*Ut* [ *i* ] : timer; (\* For all indices  $Low \leq i < High$  \*)

# Sender Protocol (i)

---

$A_p$ : (\* Accept next word \*)

**begin if !cs then**

**begin** (\* Connection is opened first \*)

        create ( $St$ ,  $High$ ,  $Low$ ); (\*  $cs := true$  \*)

$Low := High := 0$ ;  $St := S$ ;

**end;**

$Ut [ B + High ] := U$ ;  $High := High + 1$ ;

**end;**

$S_p$ : (\* Send  $i^{\text{th}}$  word from current connection \*)

{ $cs \wedge Low \leq i < High \wedge Ut [ B + i ] > 0$ }

**begin**            send <data, ( $i = Low$ ),  $i$ ,  $in_p [ B + i ]$ ,  $v$ >;

$St := S$ ;

**end;**

# Sender Protocol (ii)

---

$R_p$  : (\* Receive and acknowledgement \*)

$\{cs \wedge \langle \mathbf{ack}, i, \rho \rangle \in M_p\}$

**begin** receive  $\langle \mathbf{ack}, i, \rho \rangle$ ;  $Low := \max(Low, i)$  **end**;

$E_p$  : (\* Generate error report for possibly lost word\*)

$\{cs \wedge Ut [ B + Low ] < -2\mu - R\}$

**begin** error  $[ B + Low ] := true$ ;  $Low := Low + 1$  **end**;

$C_p$  : (\* Close the connection \*)

$\{cs \wedge St < 0 \wedge Low = High\}$

**begin**  $B := B + High$ ; delete  $(St, High, Low)$  **end**;

(\*  $cs := false$  \*)

# Receiver Protocol

---

$R_q$  : (\* Receive a data packet \*)  
    {<data,  $s$ ,  $i$ ,  $w$ ,  $\rho$ >  $\in M_q$ }  
    **begin** receive <data,  $s$ ,  $i$ ,  $w$ ,  $\rho$ >;  
        **if**  $cr$  **then**  
            **if**  $i = Exp$  **then**  
                **begin**  $Rt := R$ ;  $Exp := i+1$ ; deliver  $w$  **end**;  
            **else if**  $s = true$  **then**  
                **begin** create < $Rt$ ,  $Exp$ >; (\*  $cr := true$  \*)  
                     $Rt := R$ ;  $Exp := i + 1$ ; deliver  $w$   
                **end**  
        **end**  
    **end**

$S_q$  : (\* Send an acknowledgement \*)  
    { $cr$ }  
    **begin** send <ack,  $Exp$ ,  $\mu$ > **end**;

(\* Close connection if  $Rt$  times out; see action **Time** \*)

# Channel and Time Actions

---

**Loss** :  $\{m \in M\}$  (\*  $M$  is either  $M_p$  or  $M_q$  \*)  
    **begin** remove  $m$  from  $M$ ; **end**;

**Dupl**:  $\{m \in M\}$  (\*  $M$  is either  $M_p$  or  $M_q$  \*)  
    **begin** insert  $m$  in  $M$ ; **end**;

**Time** : (\*  $\delta > 0$  \*)  
    **begin forall**  $i$  **do**  $Ut[i] := Ut[i] - \delta$ ;  
         $St := St - \delta$ ;  
         $Rt := Rt - \delta$ ;  
        **if**  $Rt \leq 0$  **then** delete ( $Rt$ ,  $Exp$ ); (\*  $cr := false$  \*)  
        **forall**  $\langle \dots, \rho \rangle \in M_p, M_q$  **do**  
            **begin**  $\rho := \rho - \delta$ ;  
                **if**  $\rho \leq 0$  **then** remove packet  
            **end**  
    **end**

# Conclusion

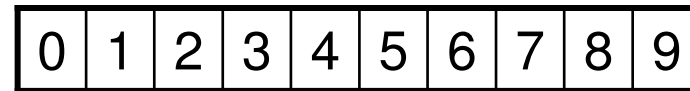
---

- Balanced Sliding Window Protocol
- Timer-based Sliding Window Protocol

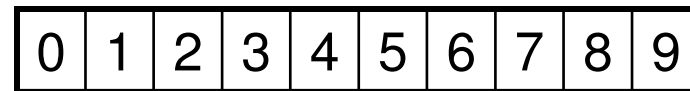
# Balanced Targil: Example Setup

---

P:  $l_p = 3, a_p = 0, s_p = 0$

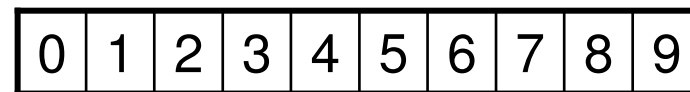


$in_p$

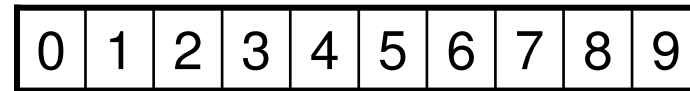


$out_p$

---



$out_q$



$in_q$

Q:  $l_q = 2, a_q = 0, s_q = 0$

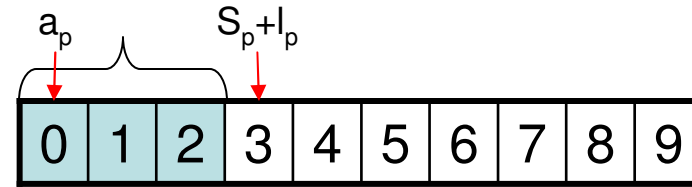
---

# Balanced Targil: Step 1

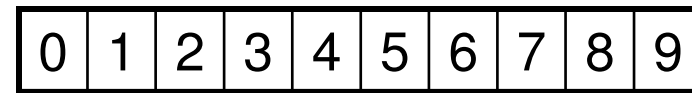
P:  $l_p = 3, a_p = 0, s_p = 0$

$s_p + l_p = 0 + 3 = 3$

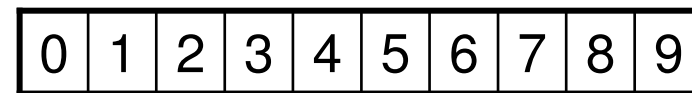
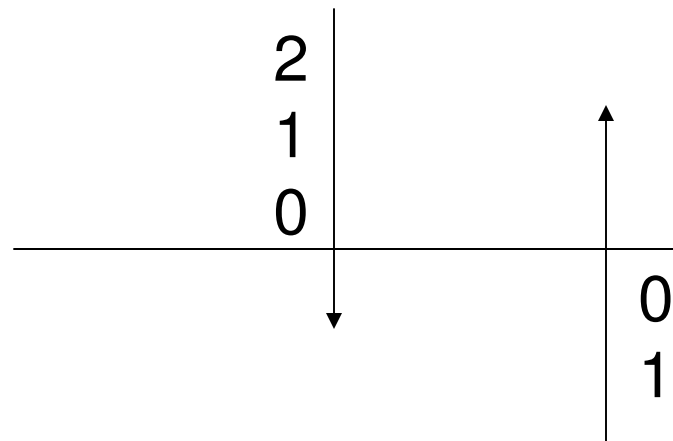
Can send:  $i: 0 \leq i < 3$



$in_p$



$out_p$

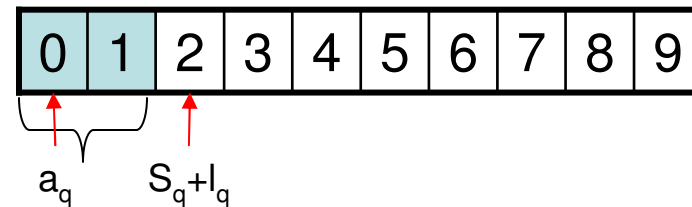


$out_q$

Q:  $l_q = 2, a_q = 0, s_q = 0$

$s_q + l_q = 0 + 2 = 2$

Can send:  $i: 0 \leq i < 2$



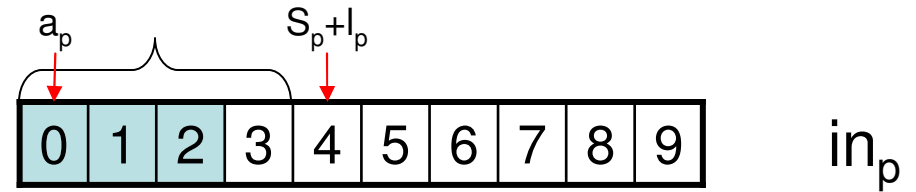
$in_q$

# Balanced Targil: Step 2

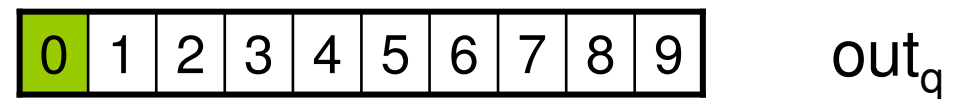
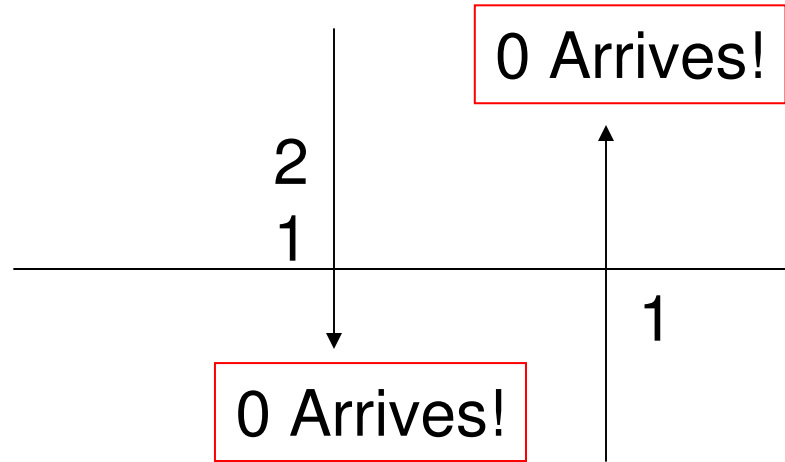
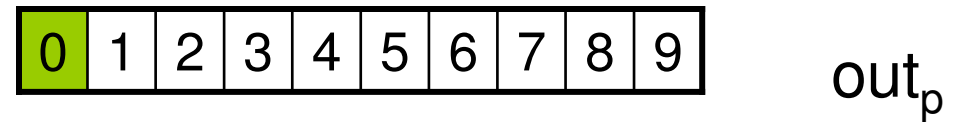
P:  $l_p = 3$ ,  $a_p = 0$ ,  $s_p = 1$

$s_p + l_p = 1 + 3 = 4$

Can send:  $i: 0 \leq i < 4$



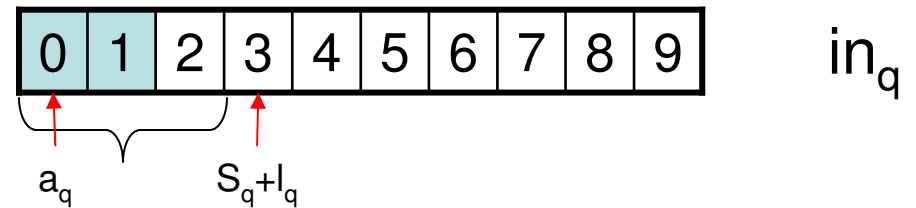
0 Arrives!



Q:  $l_q = 2$ ,  $a_q = 0$ ,  $s_q = 1$

$s_q + l_q = 1 + 2 = 3$

Can send:  $i: 0 \leq i < 3$

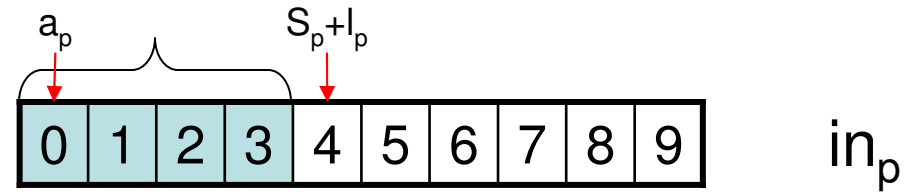


# Balanced Targil: Step 3

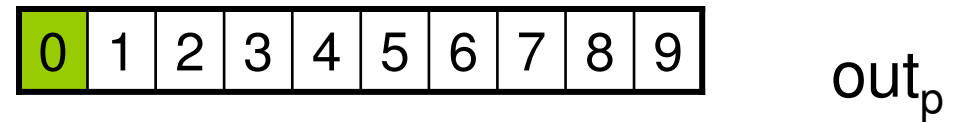
P:  $l_p = 3$ ,  $a_p = 0$ ,  $s_p = 1$

$s_p + l_p = 1 + 3 = 4$

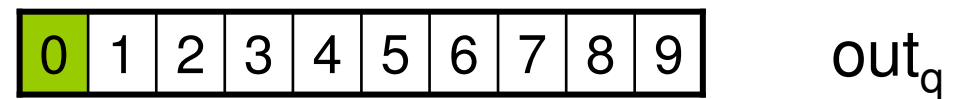
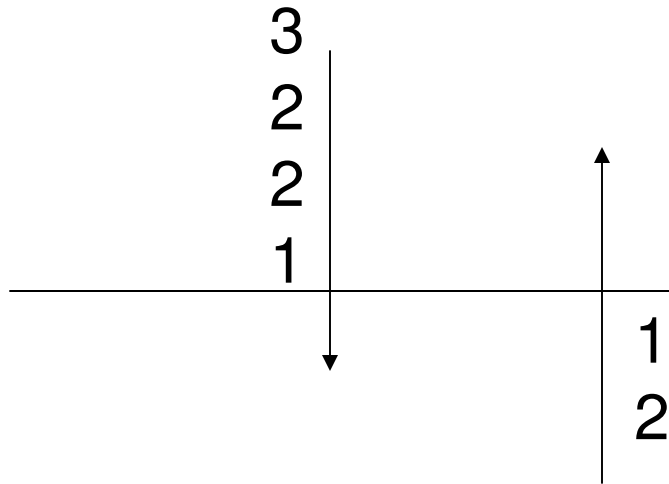
Can send:  $i: 0 \leq i < 4$



$in_p$



$out_p$

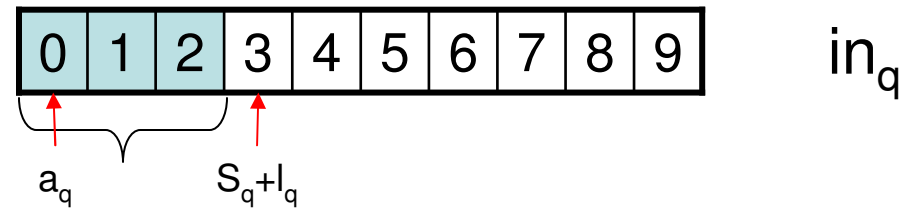


$out_q$

Q:  $l_q = 2$ ,  $a_q = 0$ ,  $s_q = 1$

$s_q + l_q = 1 + 2 = 3$

Can send:  $i: 0 \leq i < 3$



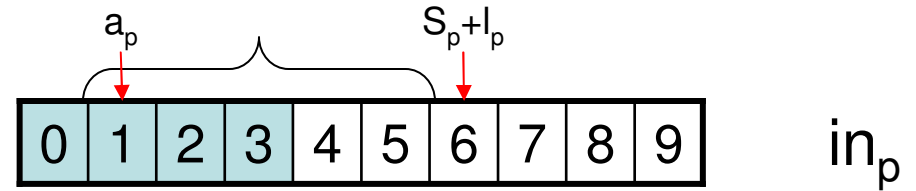
$in_q$

# Balanced Targil: Step 4

P:  $l_p = 3$ ,  $a_p = 1$ ,  $s_p = 3$

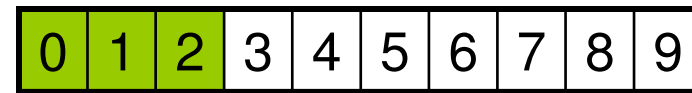
$s_p + l_p = 3 + 3 = 6$

Can send:  $i: 1 \leq i < 6$



$in_p$

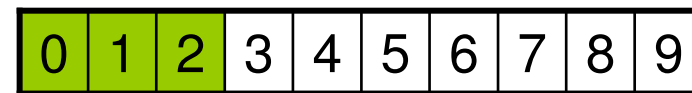
1,2 Arrive!



$out_p$

3  
2

1,2 Arrive!

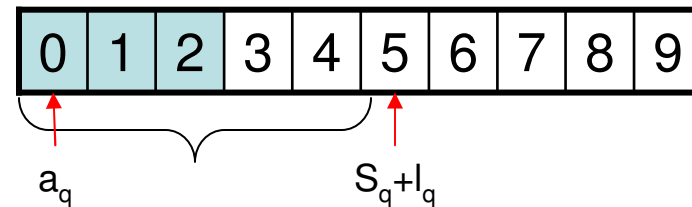


$out_q$

Q:  $l_q = 2$ ,  $a_q = 0$ ,  $s_q = 3$

$s_q + l_q = 3 + 2 = 5$

Can send:  $i: 0 \leq i < 5$



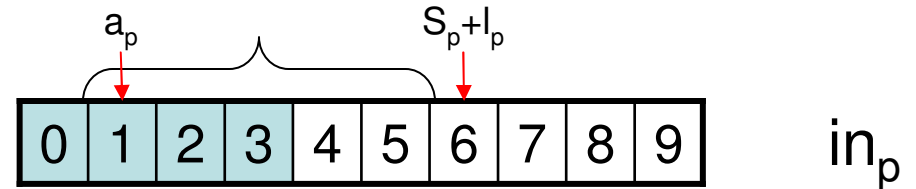
$in_q$

# Balanced Targil: Step 5

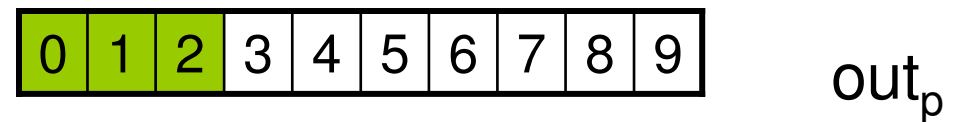
P:  $l_p = 3$ ,  $a_p = 1$ ,  $s_p = 3$

$s_p + l_p = 3 + 3 = 6$

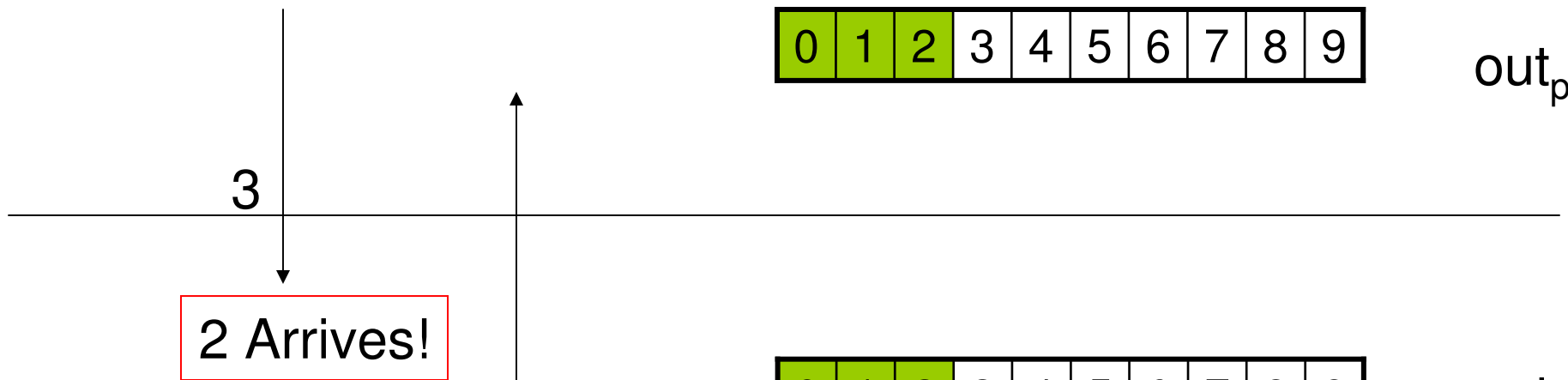
Can send:  $i: 1 \leq i < 6$



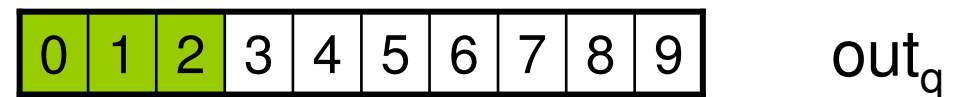
$in_p$



$out_p$



Nothing!

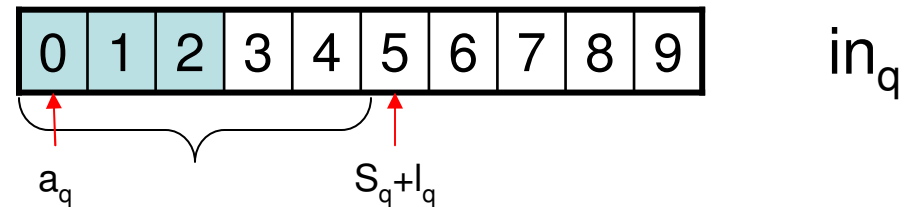


$out_q$

Q:  $l_q = 2$ ,  $a_q = 0$ ,  $s_q = 3$

$s_q + l_q = 3 + 2 = 5$

Can send:  $i: 0 \leq i < 5$



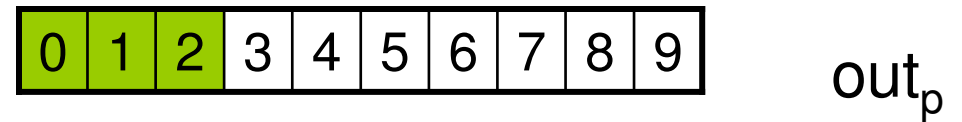
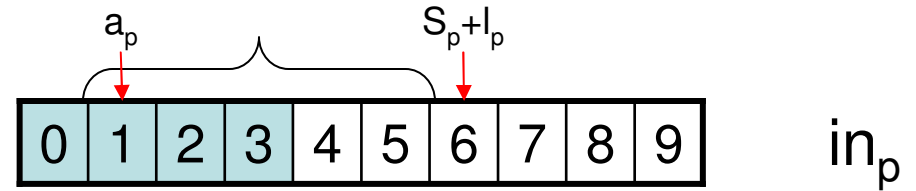
$in_q$

# Balanced Targil: Step 6

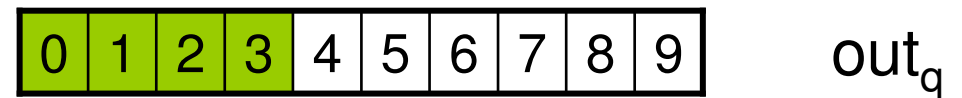
P:  $l_p = 3$ ,  $a_p = 1$ ,  $s_p = 3$

$s_p + l_p = 3 + 3 = 6$

Can send:  $i: 1 \leq i < 6$



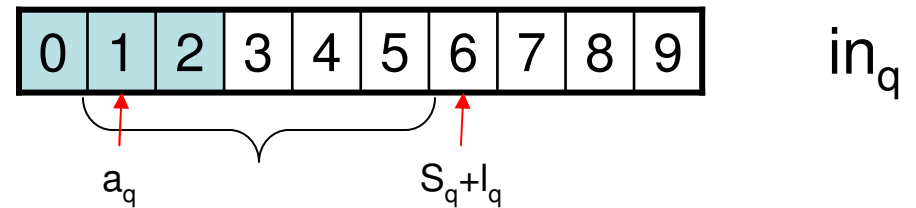
3 Arrives!



Q:  $l_q = 2$ ,  $a_q = 1$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 1 \leq i < 6$

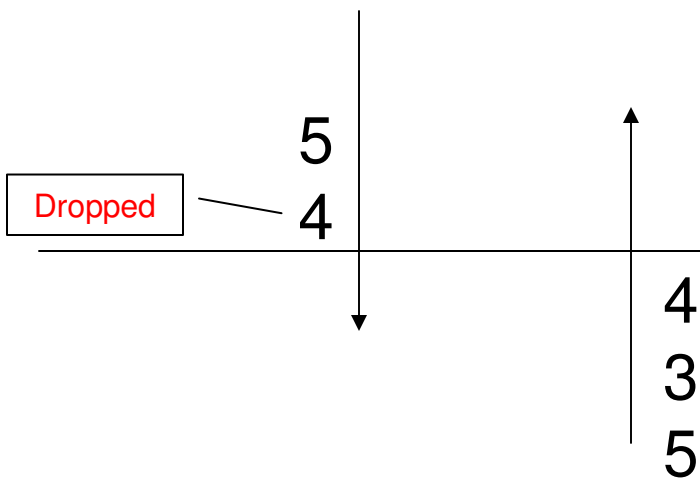
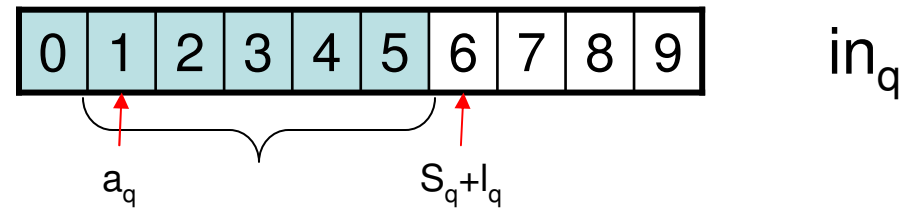
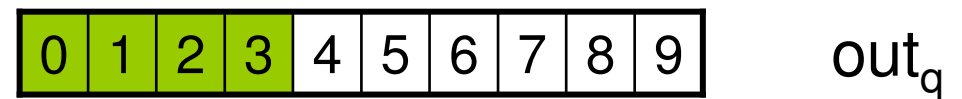
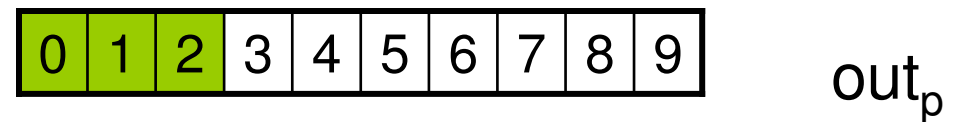
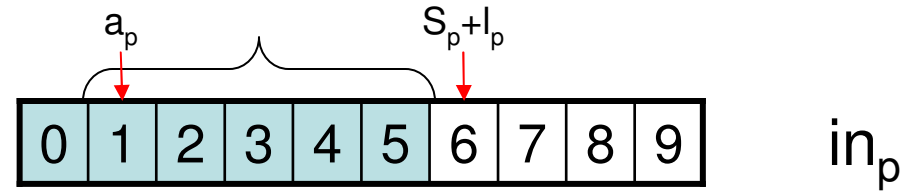


# Balanced Targil: Step 7

P:  $l_p = 3$ ,  $a_p = 1$ ,  $s_p = 3$

$s_p + l_p = 3 + 3 = 6$

Can send:  $i: 1 \leq i < 6$



Q:  $l_q = 2$ ,  $a_q = 1$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

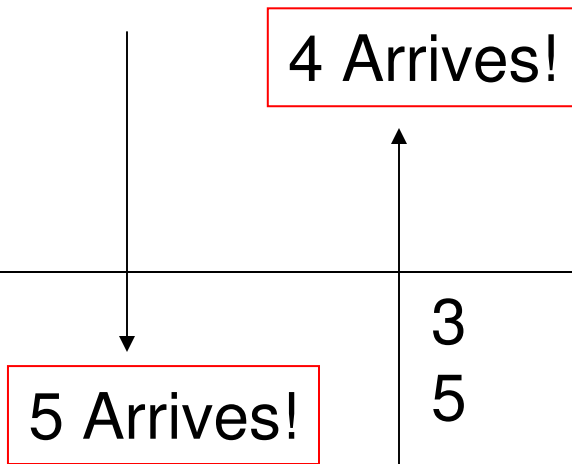
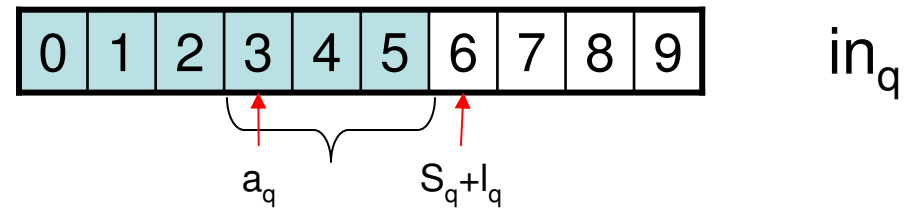
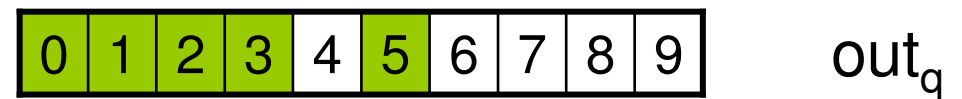
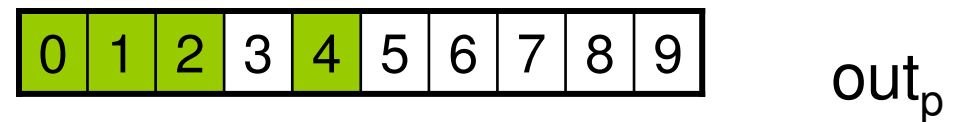
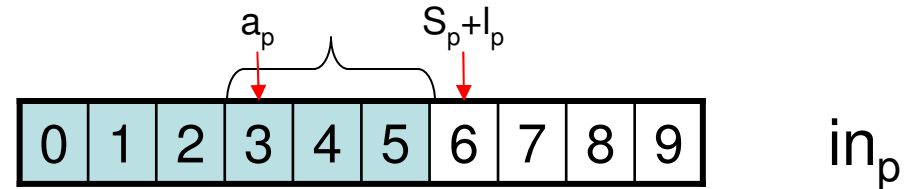
Can send:  $i: 1 \leq i < 6$

# Balanced Targil: Step 8

P:  $l_p = 3$ ,  $a_p = 3$ ,  $s_p = 3$

$s_p + l_p = 3 + 3 = 6$

Can send:  $i: 3 \leq i < 6$



Q:  $l_q = 2$ ,  $a_q = 3$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

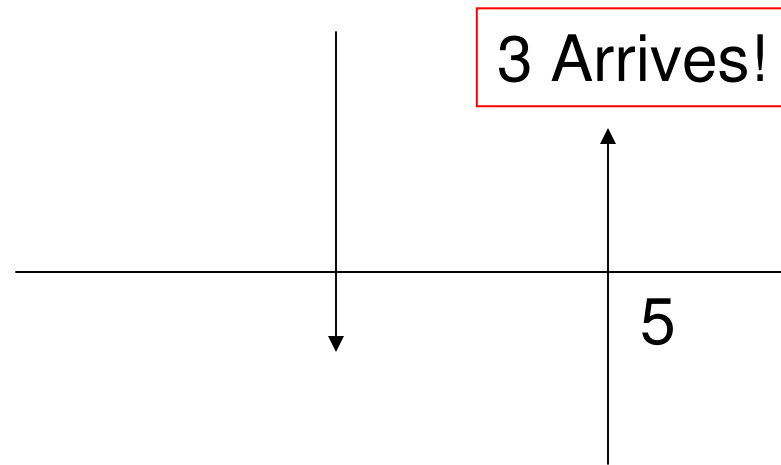
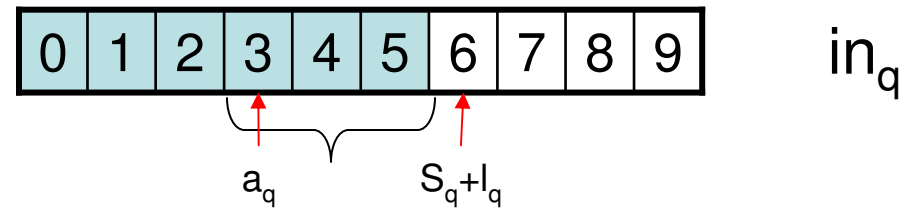
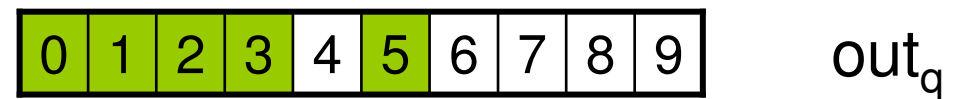
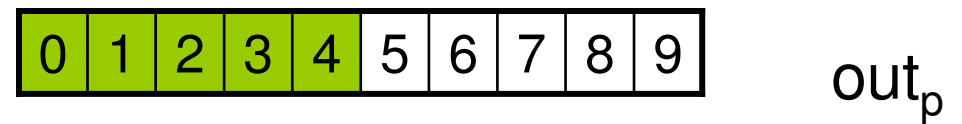
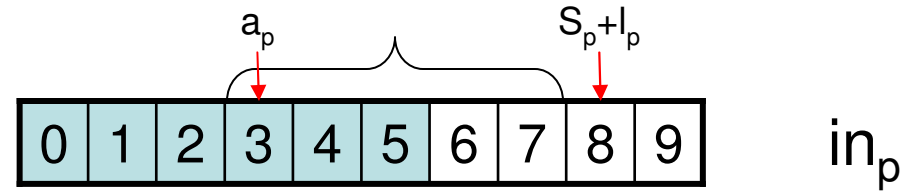
Can send:  $i: 3 \leq i < 6$

# Balanced Targil: Step 9

P:  $l_p = 3$ ,  $a_p = 3$ ,  $s_p = 5$

$s_p + l_p = 5 + 3 = 8$

Can send:  $i: 3 \leq i < 8$



Q:  $l_q = 2$ ,  $a_q = 3$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

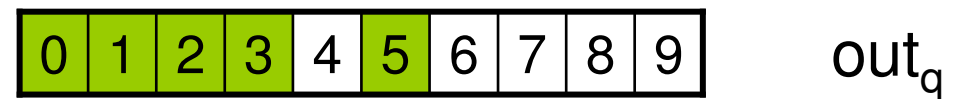
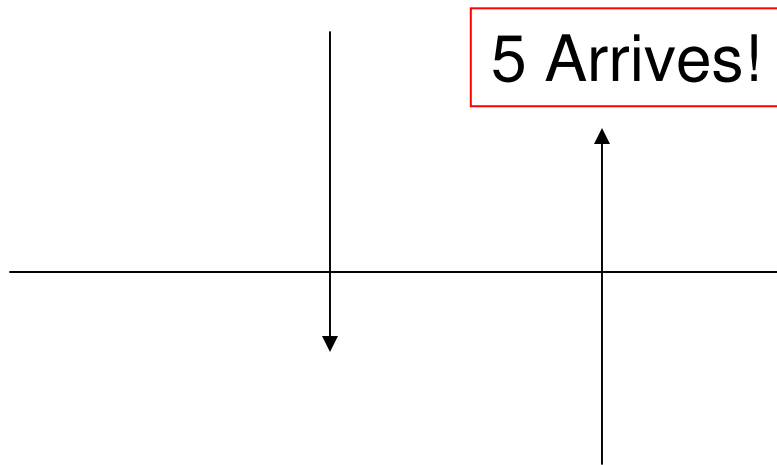
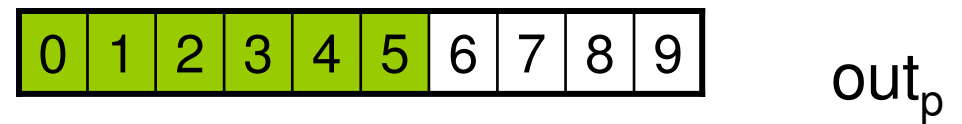
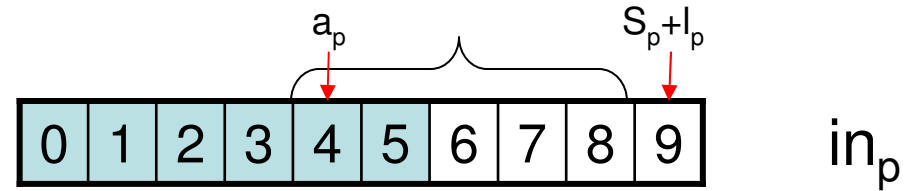
Can send:  $i: 3 \leq i < 6$

# Balanced Targil: Step 10

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

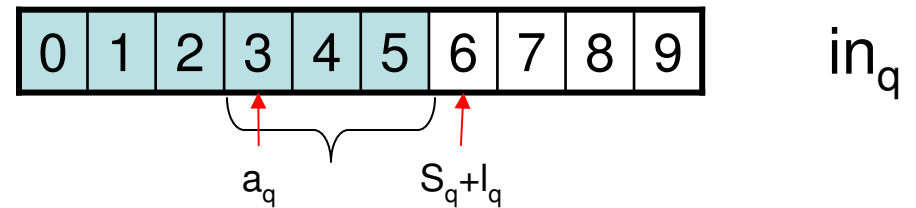
Can send:  $i: 4 \leq i < 9$



Q:  $l_q = 2$ ,  $a_q = 3$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 3 \leq i < 6$

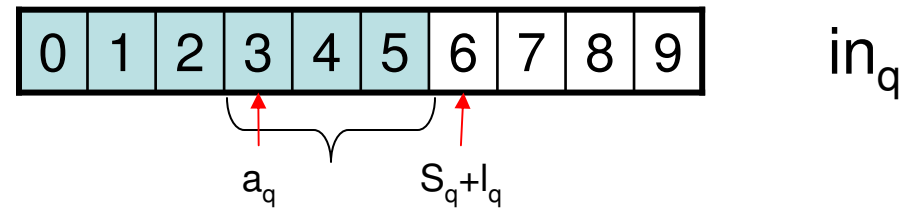
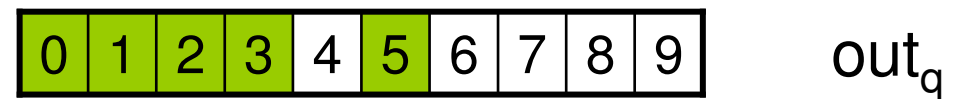
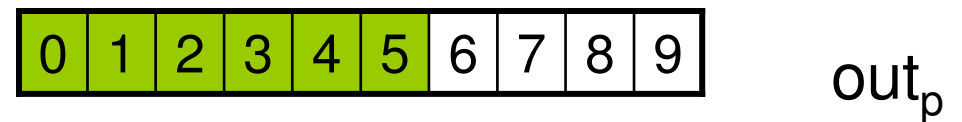
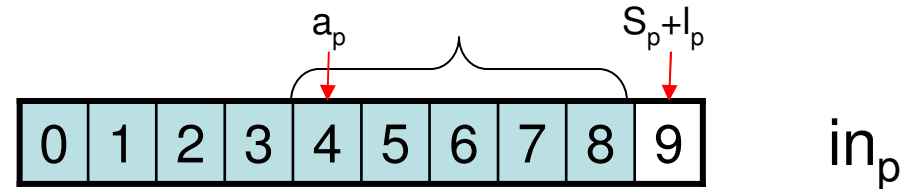


# Balanced Targil: Step 11

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

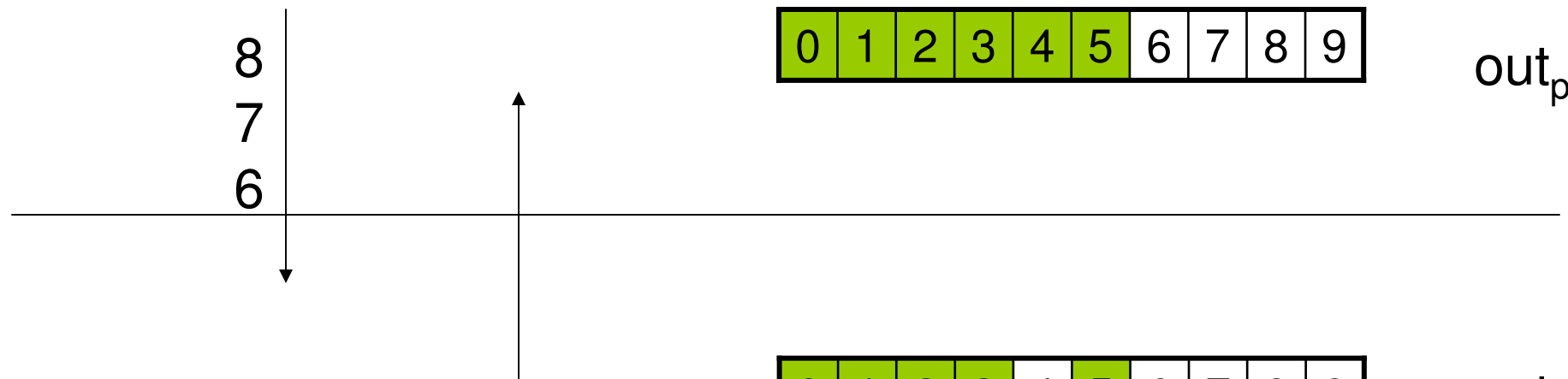
Can send:  $i: 4 \leq i < 9$



Q:  $l_q = 2$ ,  $a_q = 3$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 3 \leq i < 6$

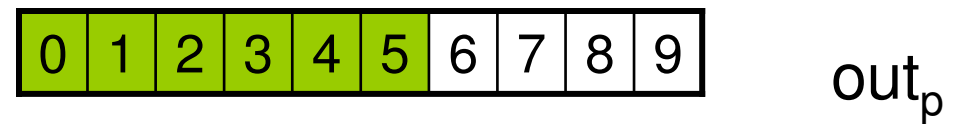
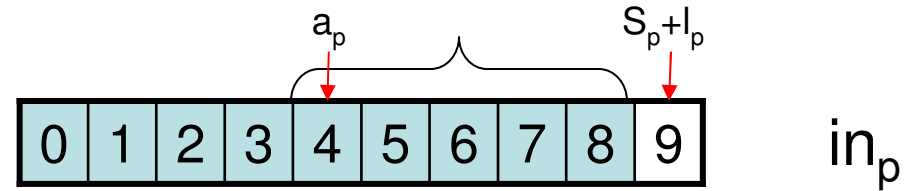


# Balanced Targil: Step 12

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

Can send:  $i: 4 \leq i < 9$



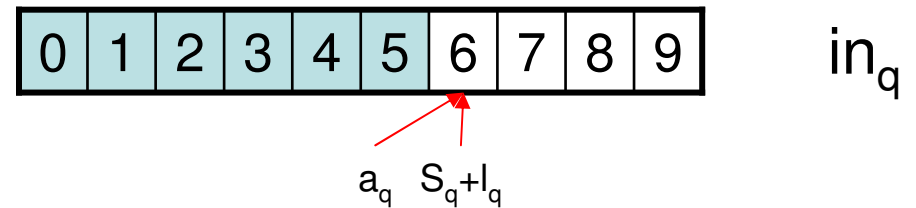
6,7,8 Arrive!



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 6 \leq i < 6$

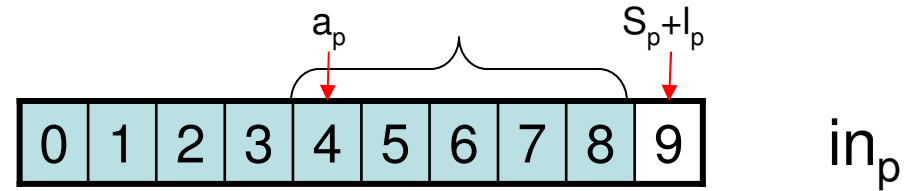


# Balanced Targil: Step 13

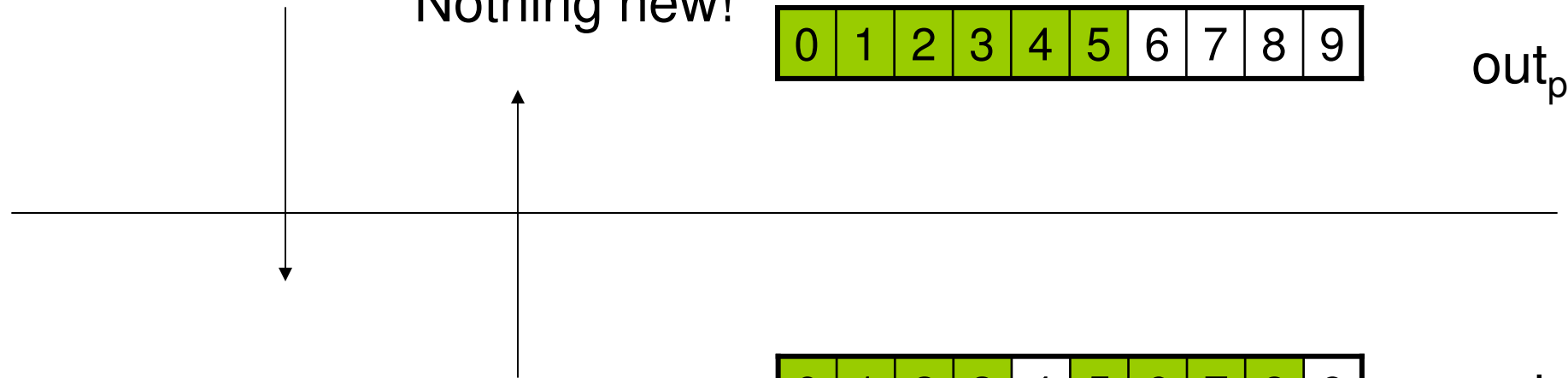
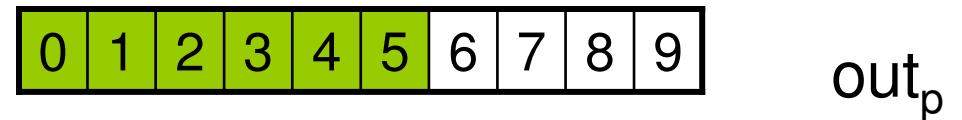
P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

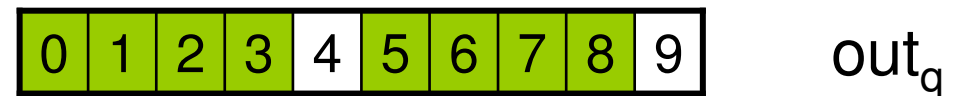
Can send:  $i: 4 \leq i < 9$



Nothing new!



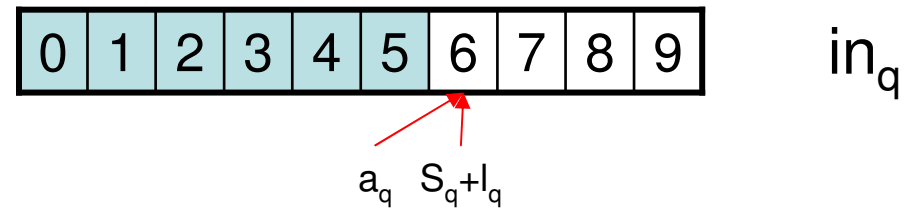
Can't send!



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 6 \leq i < 6$

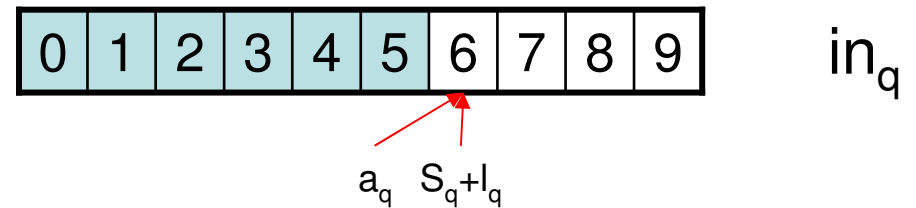
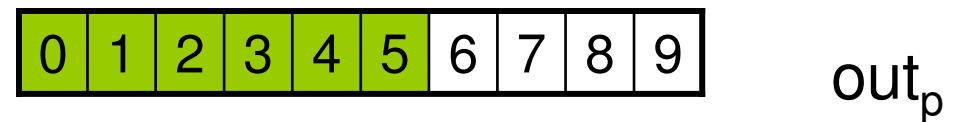
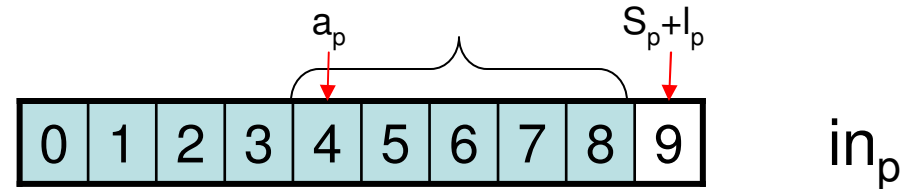


# Balanced Targil: Step 14

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

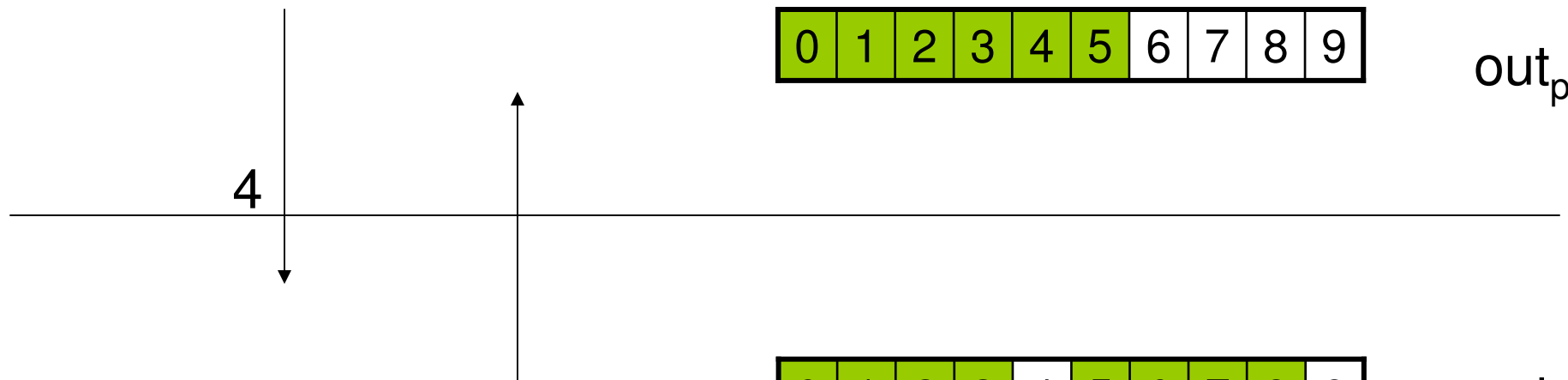
Can send:  $i: 4 \leq i < 9$



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 4$

$s_q + l_q = 4 + 2 = 6$

Can send:  $i: 6 \leq i < 6$

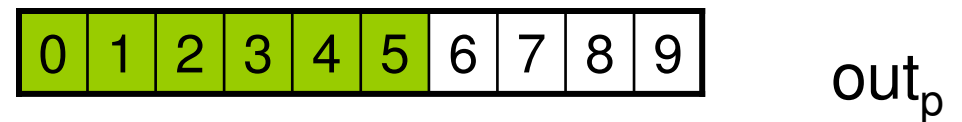
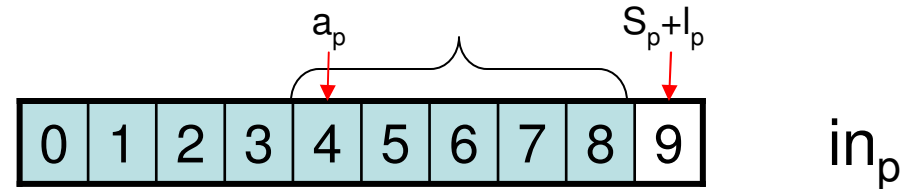


# Balanced Targil: Step 15

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

Can send:  $i: 4 \leq i < 9$



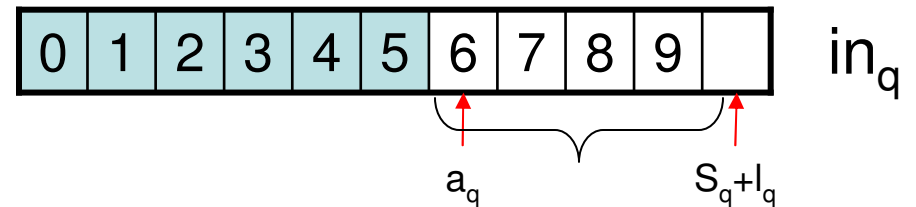
4 Arrives!



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 9$

$s_q + l_q = 9 + 2 = 11$

Can send:  $i: 6 \leq i < 11$

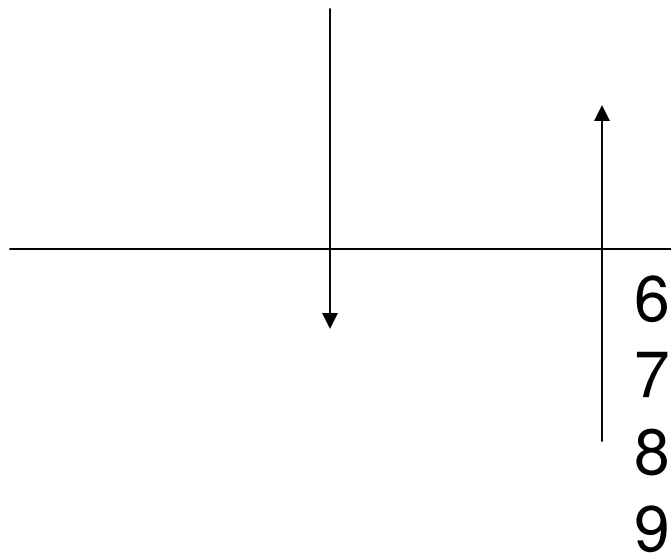
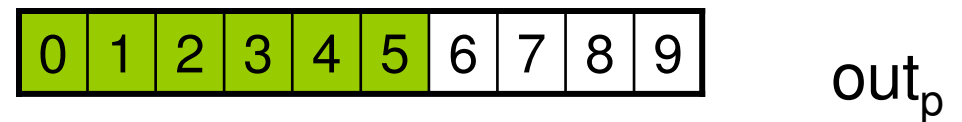
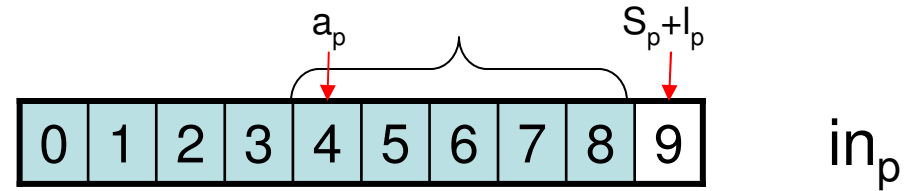


# Balanced Targil: Step 16

P:  $l_p = 3$ ,  $a_p = 4$ ,  $s_p = 6$

$s_p + l_p = 6 + 3 = 9$

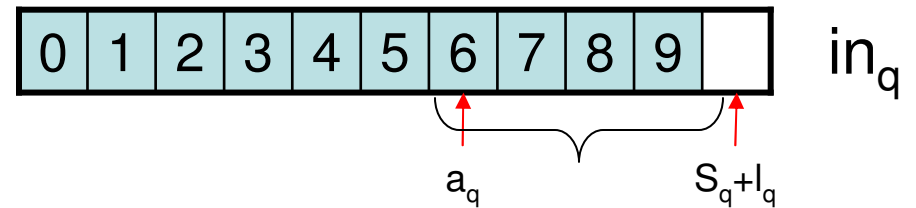
Can send:  $i: 4 \leq i < 9$



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 9$

$s_q + l_q = 9 + 2 = 11$

Can send:  $i: 6 \leq i < 11$

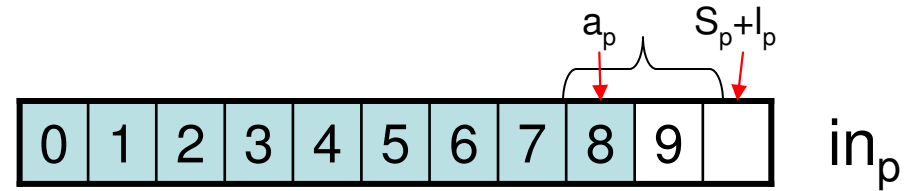


# Balanced Targil: Step 17

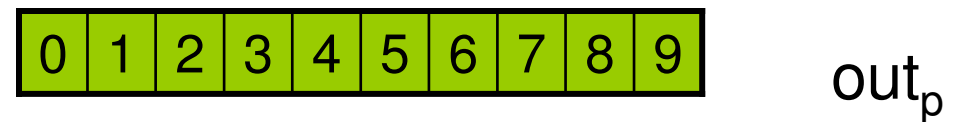
P:  $l_p = 3$ ,  $a_p = 8$ ,  $s_p = 10$

$s_p + l_p = 10 + 3 = 13$

Can send:  $i: 8 \leq i < 13$



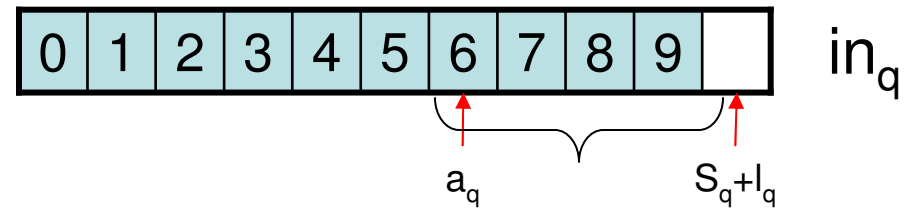
6,7,8,9 Arrive!



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 9$

$s_q + l_q = 9 + 2 = 11$

Can send:  $i: 6 \leq i < 11$

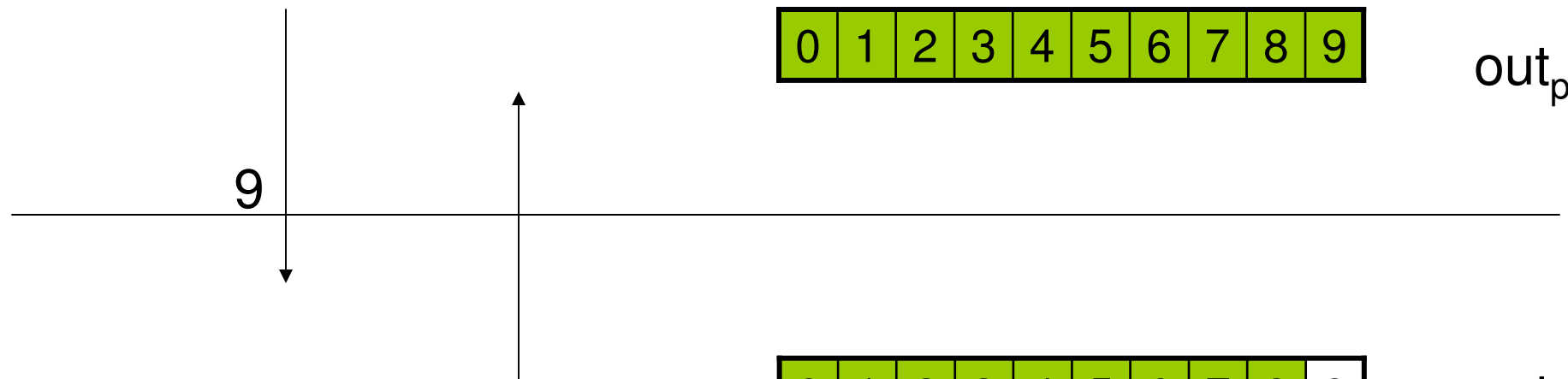
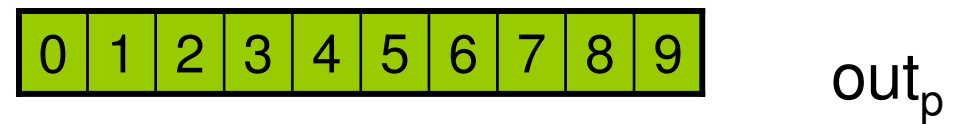
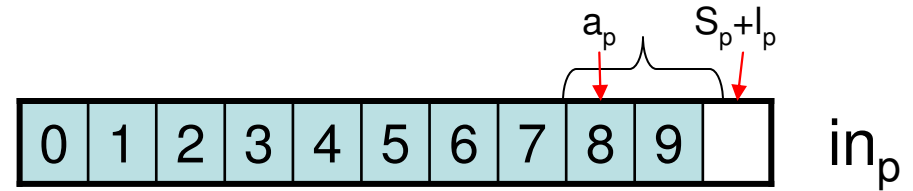


# Balanced Targil: Step 18

P:  $l_p = 3$ ,  $a_p = 8$ ,  $s_p = 10$

$s_p + l_p = 10 + 3 = 13$

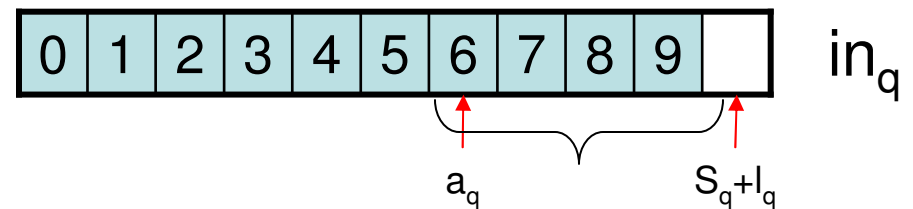
Can send:  $i: 8 \leq i < 13$



Q:  $l_q = 2$ ,  $a_q = 6$ ,  $s_q = 9$

$s_q + l_q = 9 + 2 = 11$

Can send:  $i: 6 \leq i < 11$

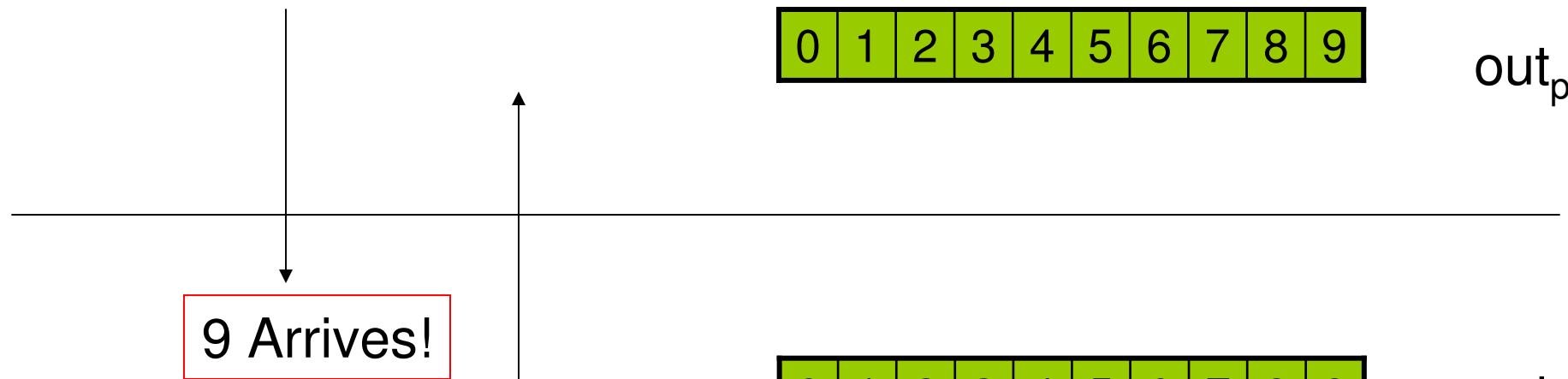
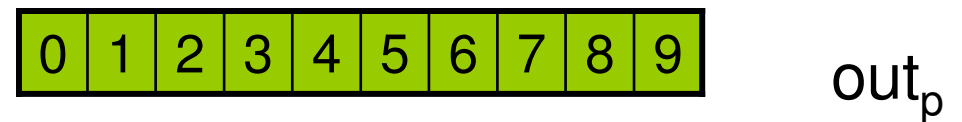
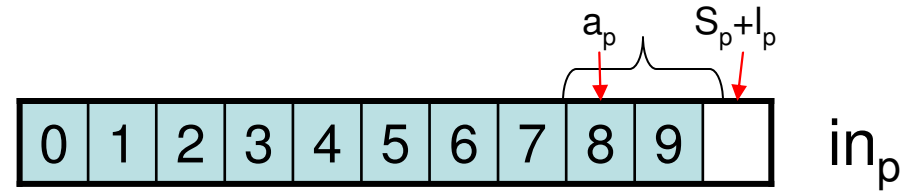


# Balanced Targil: Step 19

P:  $l_p = 3$ ,  $a_p = 8$ ,  $s_p = 10$

$s_p + l_p = 10 + 3 = 13$

Can send:  $i: 8 \leq i < 13$



Q:  $l_q = 2$ ,  $a_q = 7$ ,  $s_q = 10$

$s_q + l_q = 10 + 2 = 12$

Can send:  $i: 7 \leq i < 12$

