CS 181AG
Lecture 16

Switching

Arthi Padmanabhan

Oct 31, 2022

Upcoming Schedule

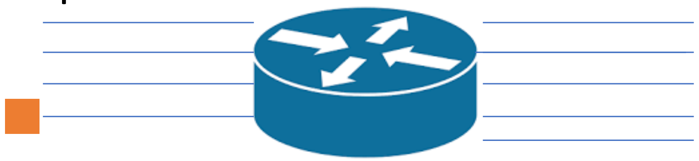
- Assignment 7 goes out Wednesday as usual, due next **Wednesday** 10pm (but please do reading before Wednesday’s class)

Big Picture: Router Functionality

Switching:
move packet to
correct port

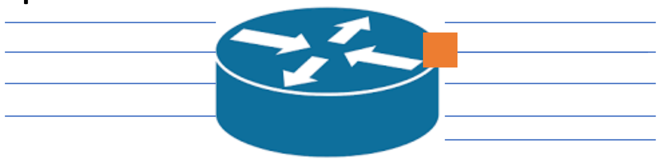
Input
ports

Output
ports



Longest Matching Prefix to decide which output port, Packet Classification to decide matching rule for packet

Big Picture: Router Functionality



Input ports

Output ports

Schedule packets in output port queue

Switching

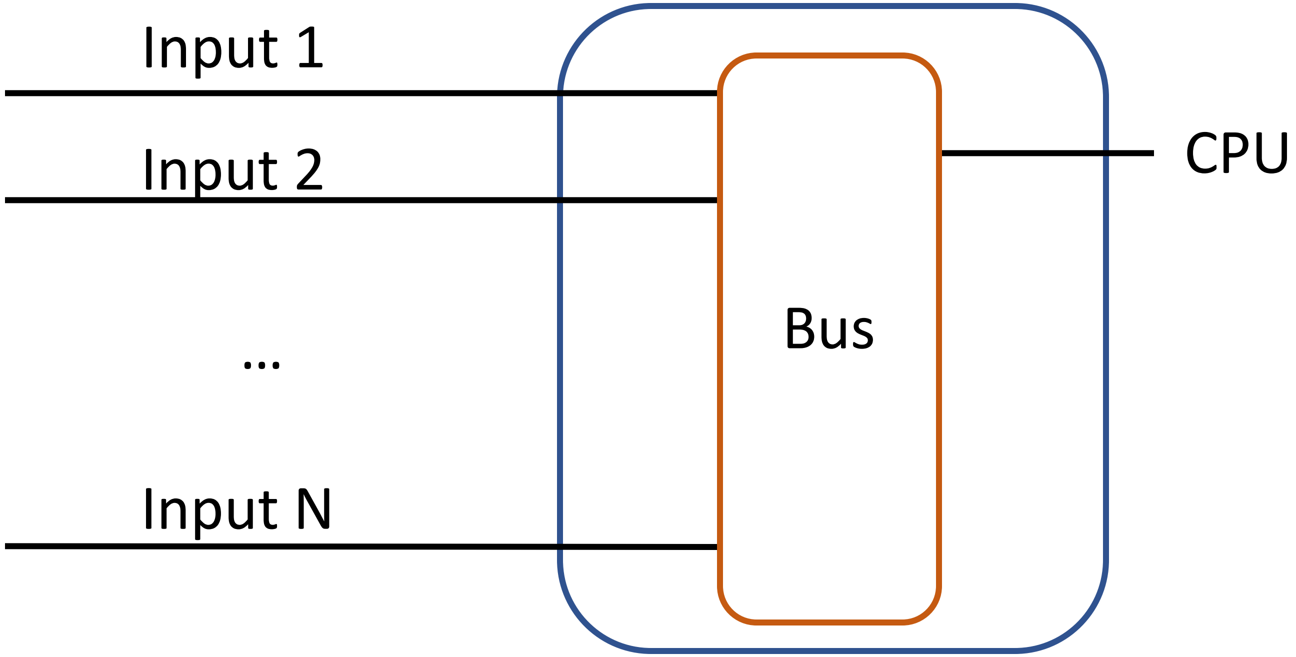
- Once router knows where a packet needs to go, it must physically move the packet to the correct output link

Simple Solution: Shared Memory Switch

- Each packet is read into memory and then read out of memory. Then the same is done for the next packet, etc

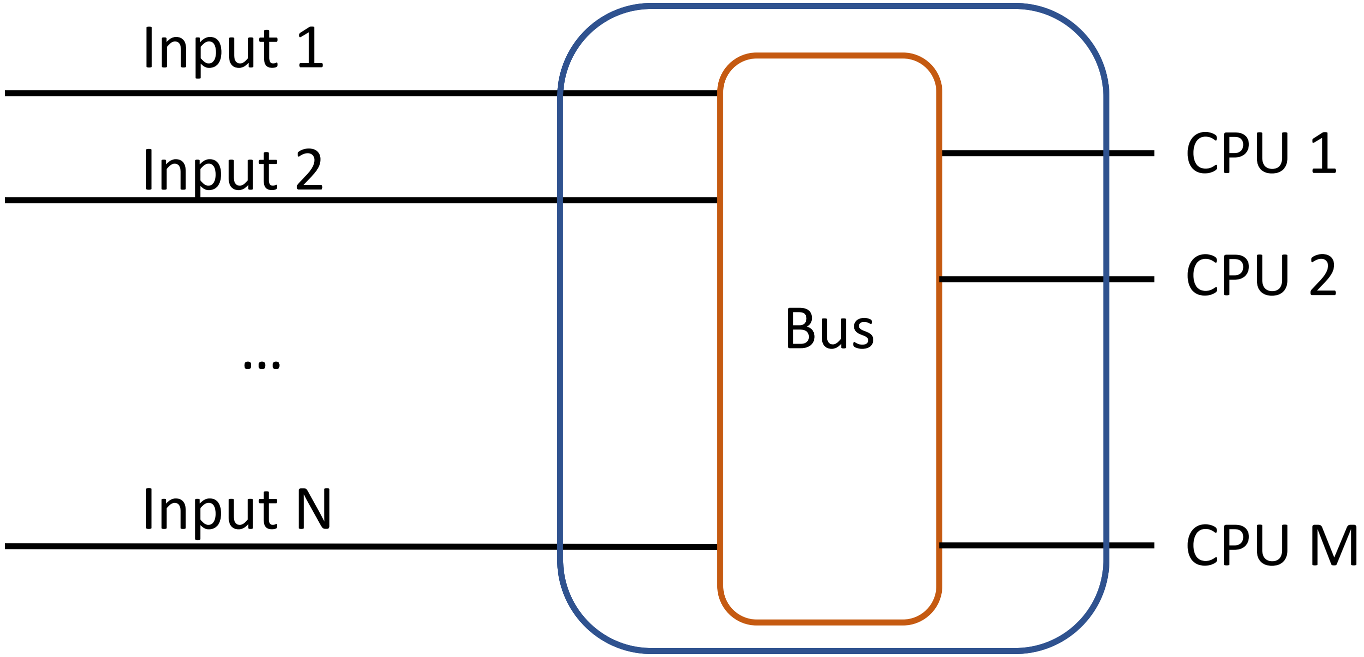
Problem:

- A single (general purpose) CPU is too slow



Shared Memory Switch with Multiple CPUS

- Use multiple CPUs to alleviate load on single CPU

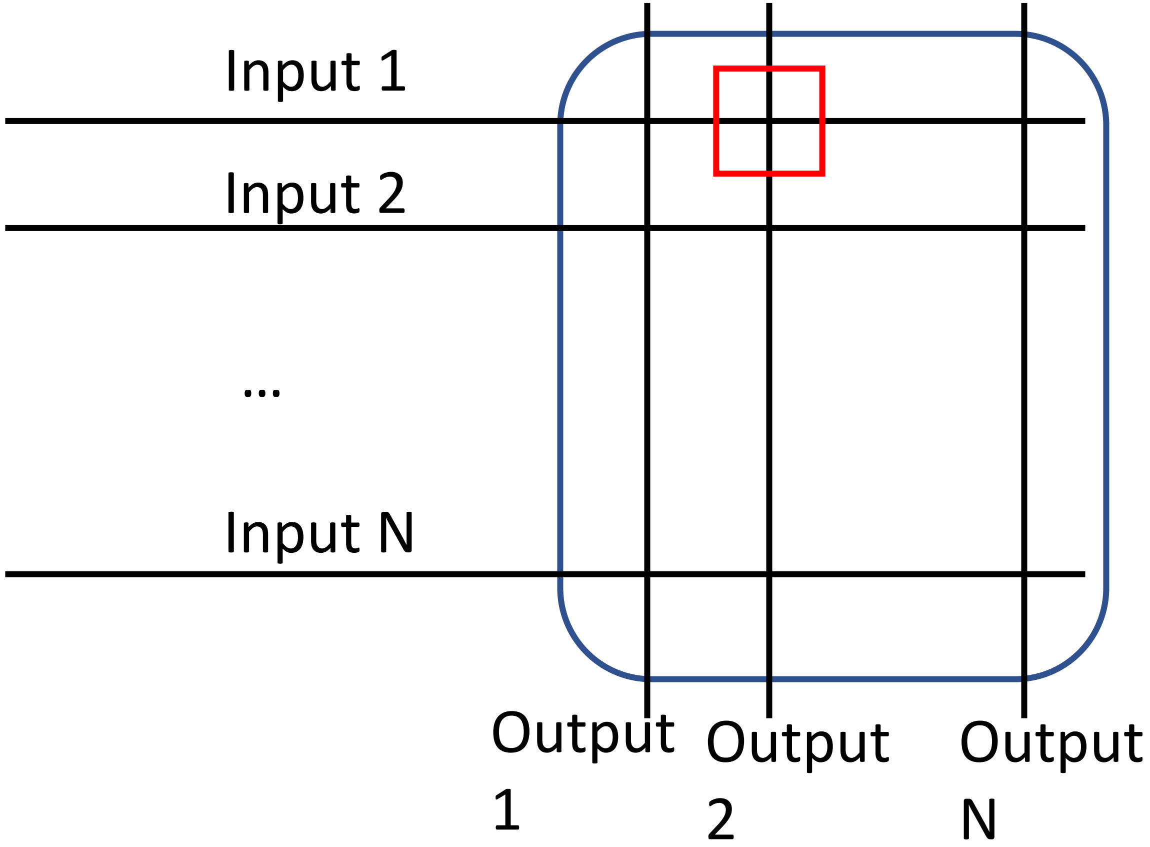


Problem:

- Packet still traverses bus twice – once to get to CPU, once to get back
- Bus: higher load -> lower speed

Crossbar Switch

- Each input is connected to each output
- Connection is a transistor



Input 1

Input 2

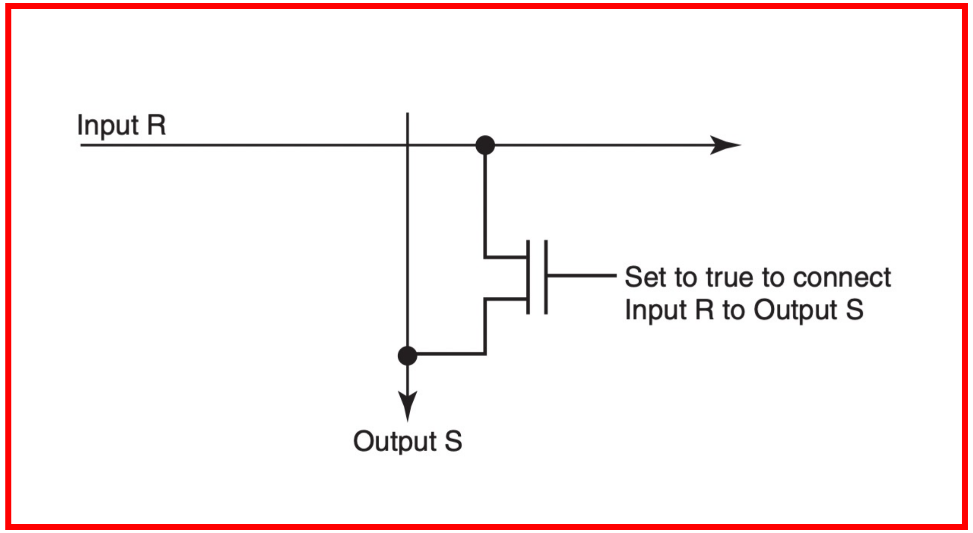
…

Input N

Output 1

Output 2

Output N



Input R

Set to true to connect Input R to Output S

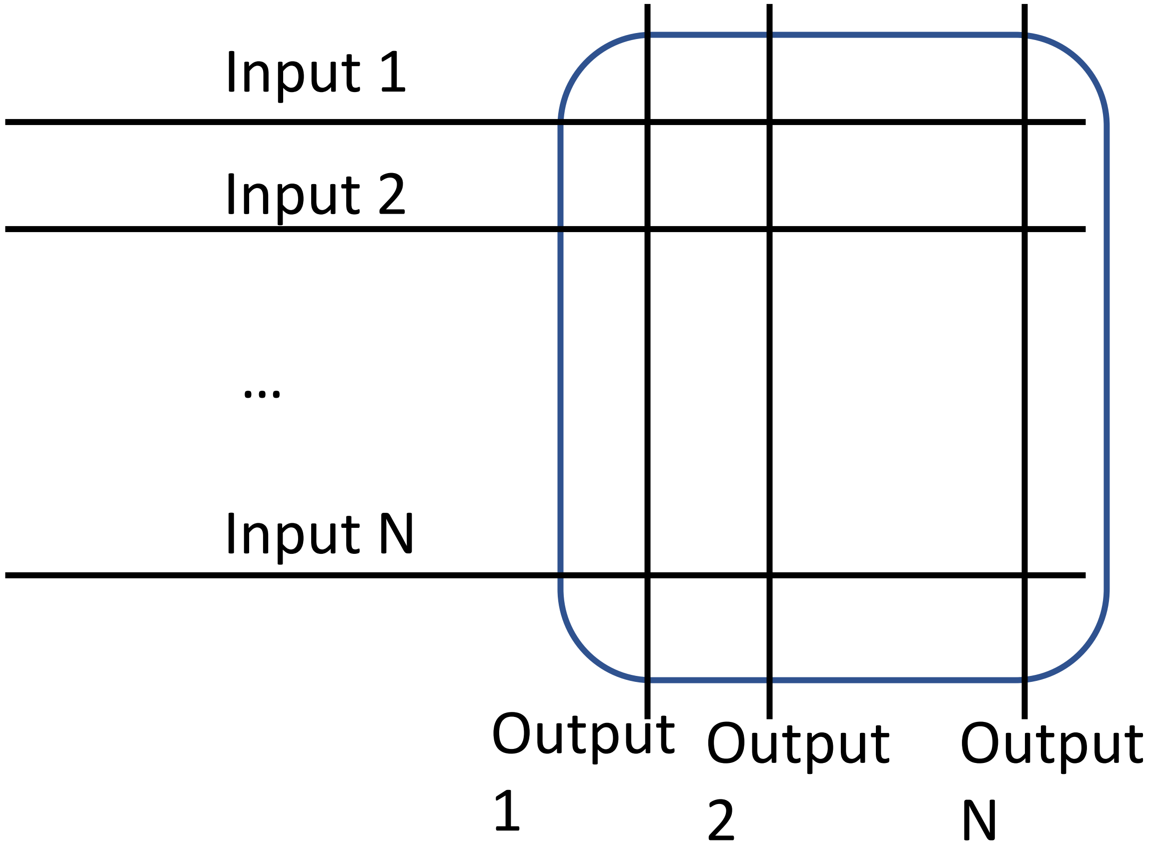
Output S

Crossbar Switch: Constraints

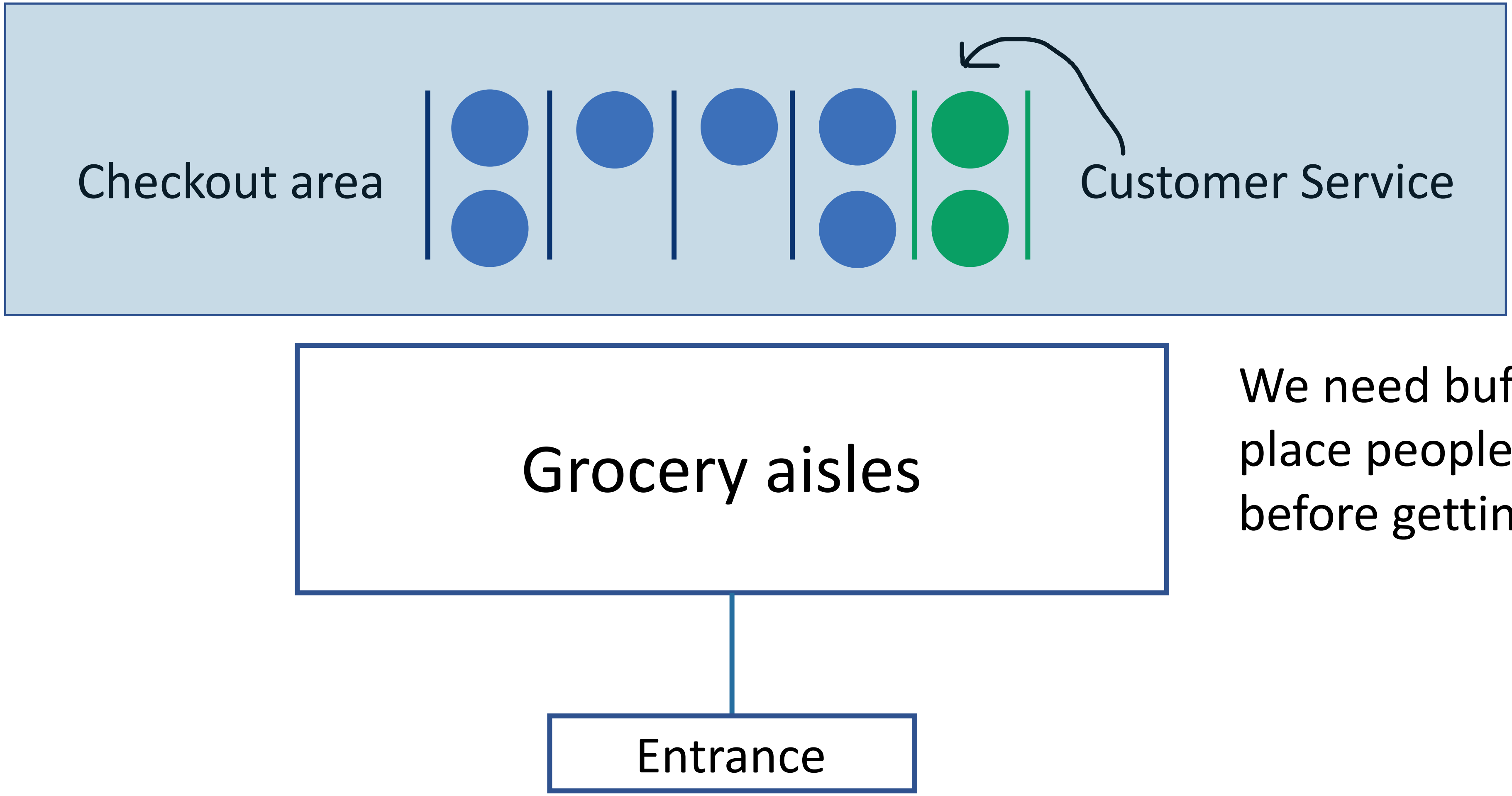
- Output should be connected to no more than one input
- Inputs can be connected to more than one output though
- Packets going to the same output should arrive at the output in the correct order

Crossbar Switch

- Best case: maximum parallelization -> N fold speedup (packets divided into cells)
- Requires finding N disjoint input-output pairs
- Why is this hard?
  - Several inputs may want to send to the same output at the same time
  - There may be outputs with no inputs wanting to send to them
- Could be reduced to bipartite matching problem (yay more algs class!)
  - All known bipartite matching algorithms are too slow in the context of switching packets

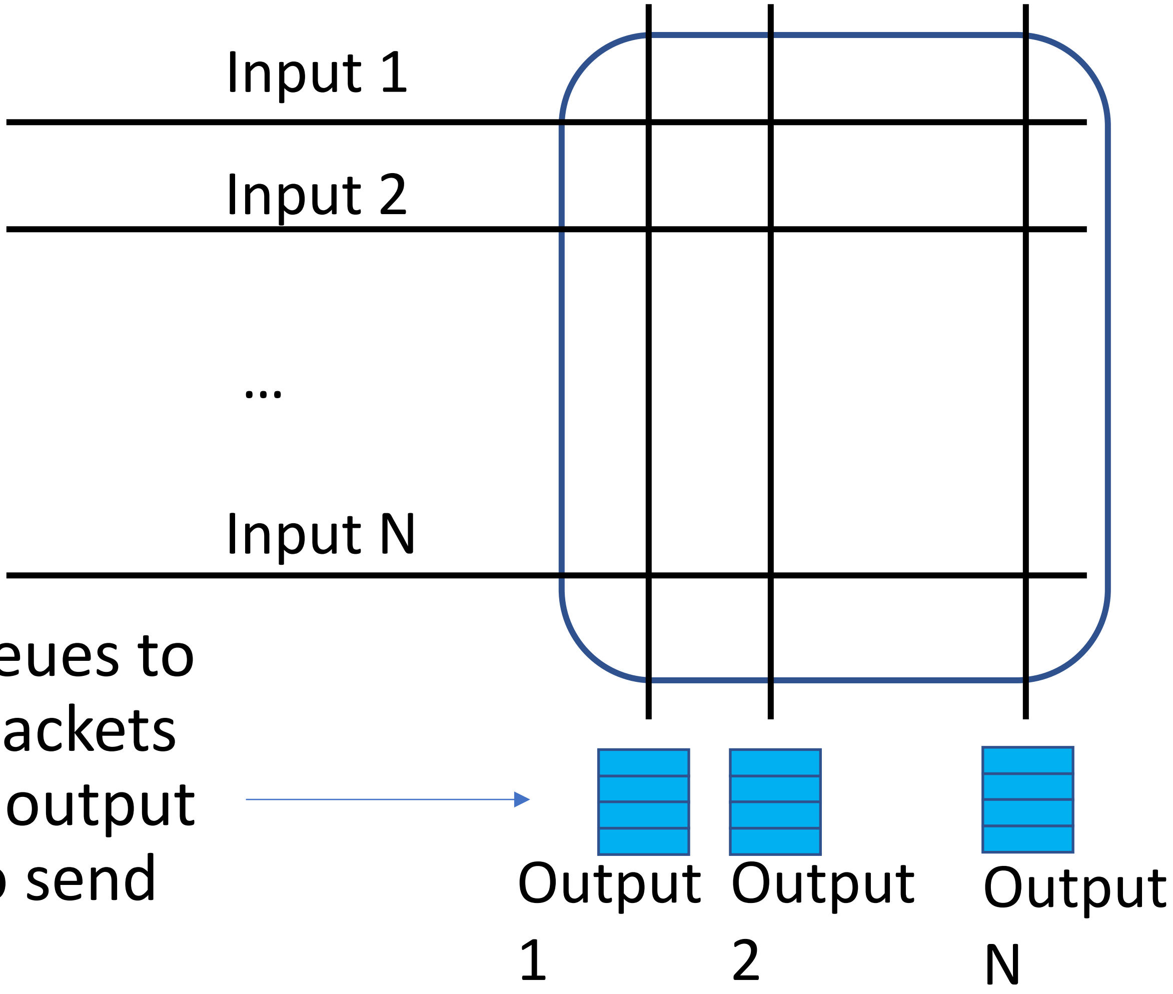


Example: Grocery Store



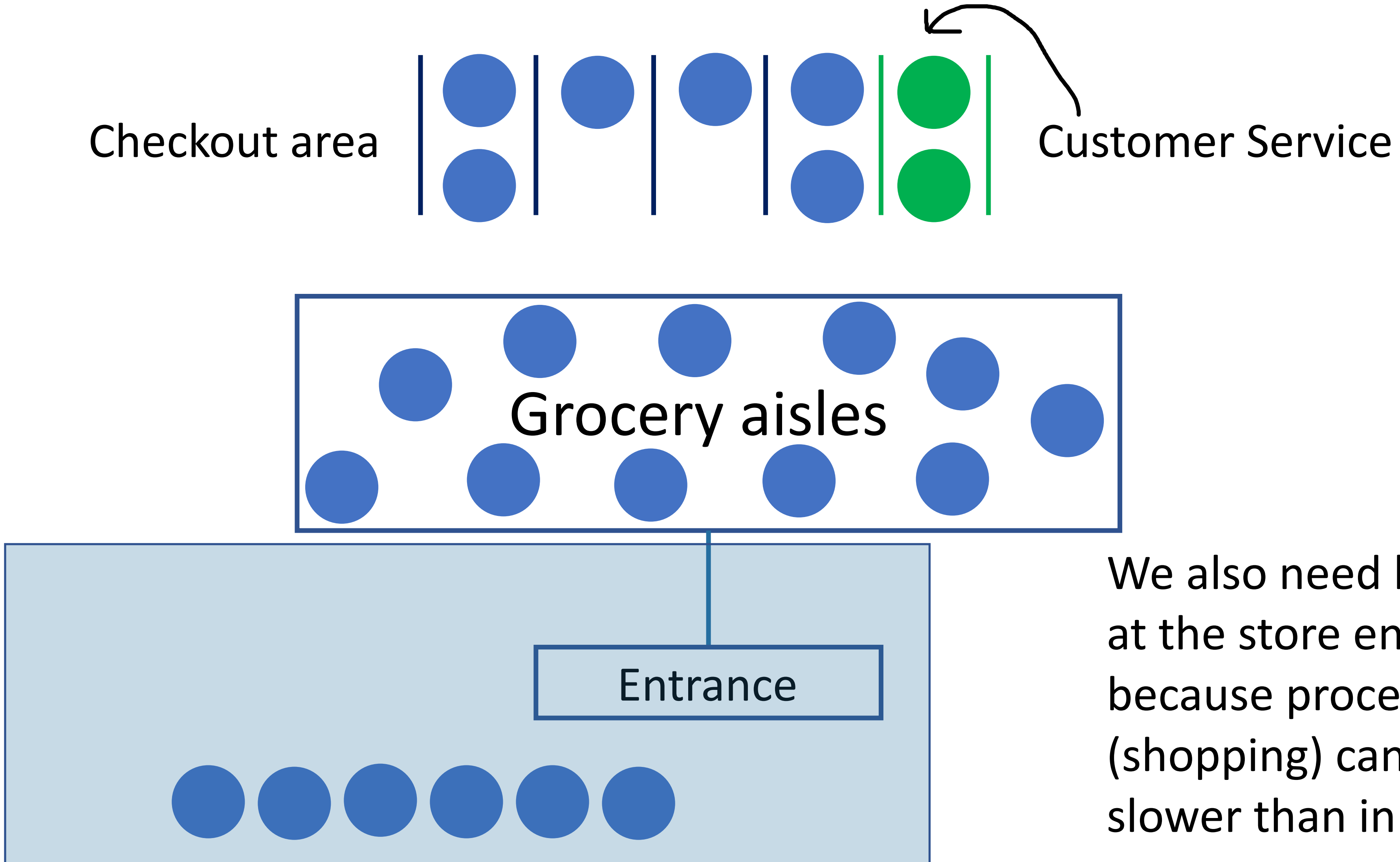
We need buffering, place people can wait before getting served

Crossbar Switch: Output queues



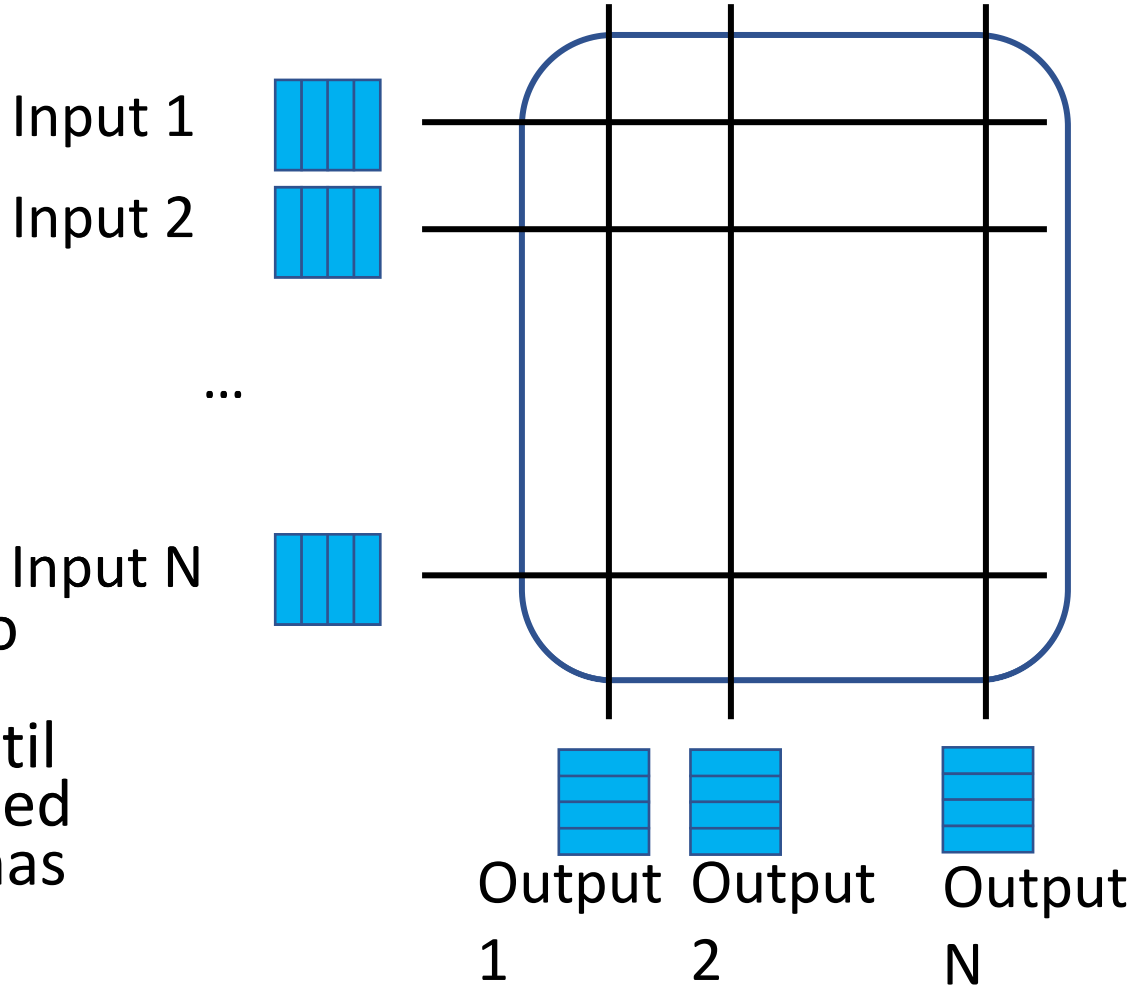
Add output queues to hold packets. Packets wait here until output port is ready to send

Example: Grocery Store



We also need buffering at the store entrance because processing (shopping) can be slower than input rate

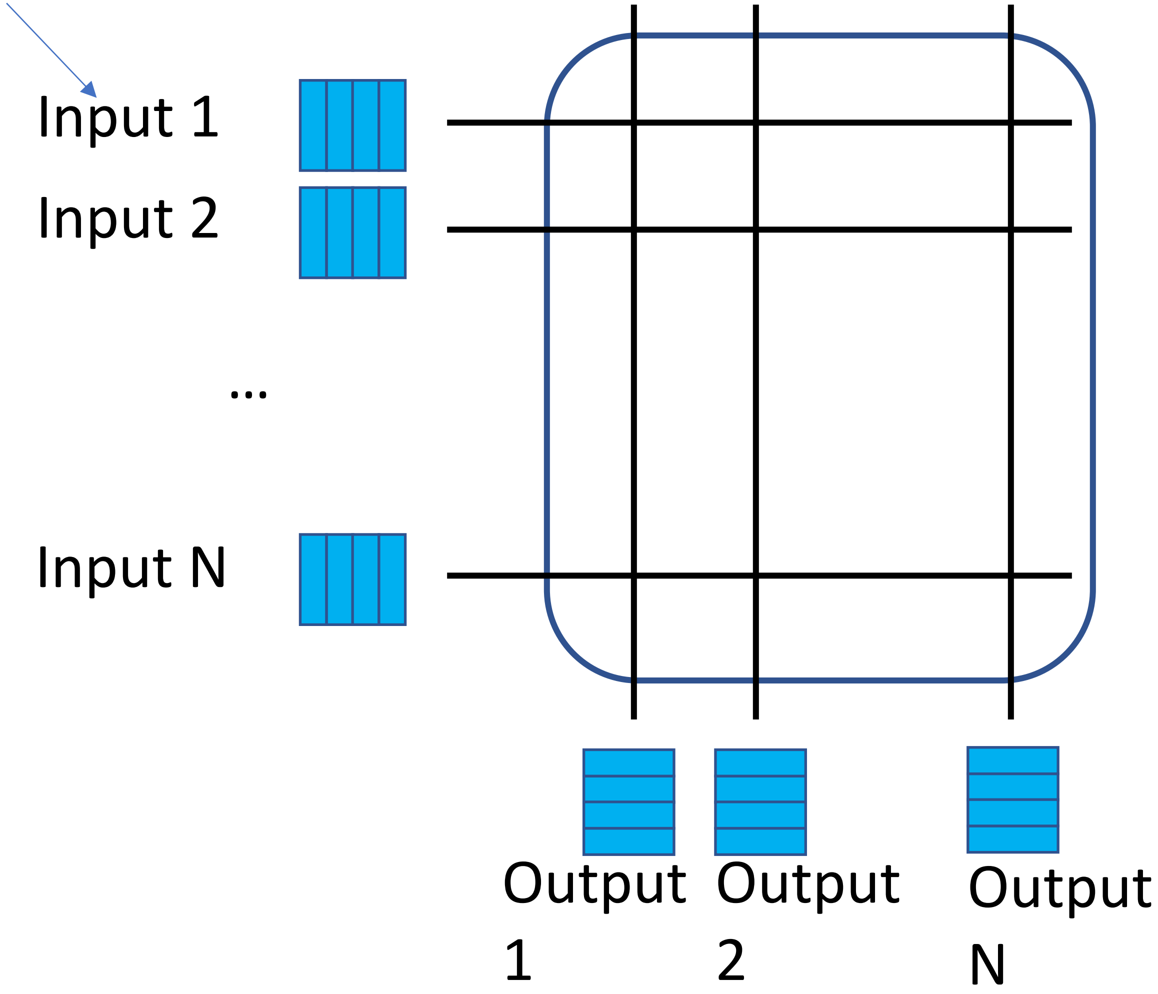
Crossbar Switch: Input Queues



Add input queues to temporarily hold received packets until they can be processed and output queue has free slot

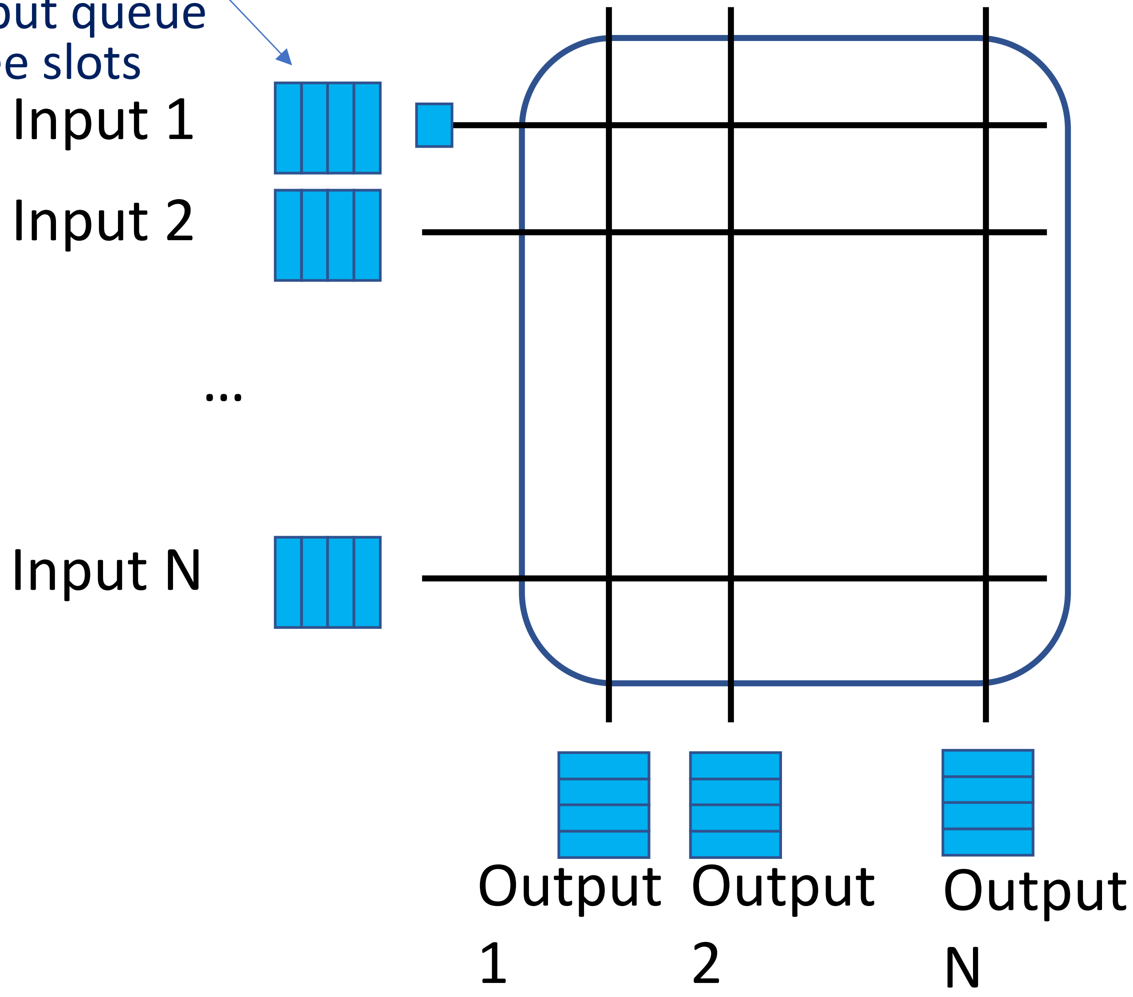
Crossbar Switch: Putting it together

Performs longest matching prefix and finds that packet should go to output 2

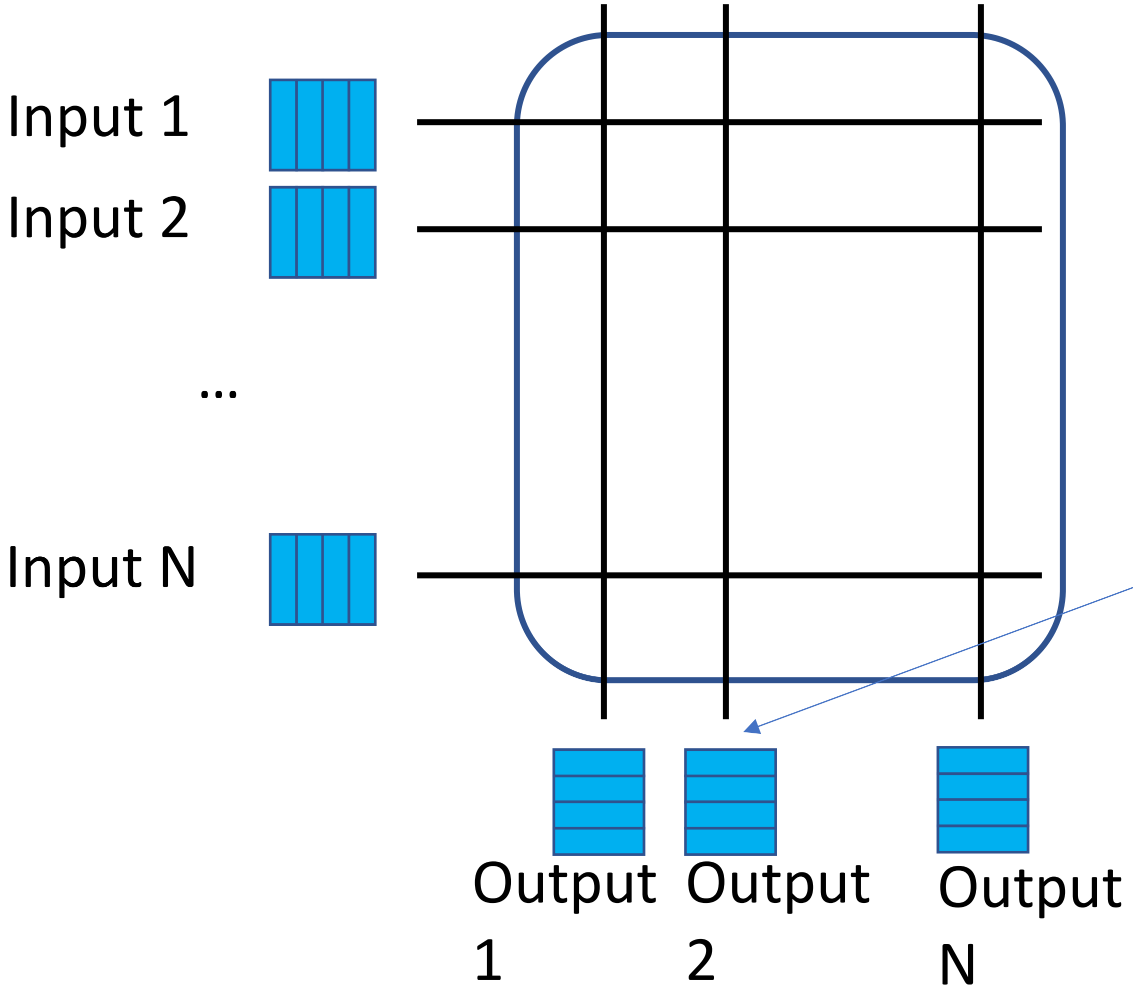


Crossbar Switch: Putting it together

Packet remains queued until input queue empties and output queue 2 has free slots



Crossbar Switch: Putting it together

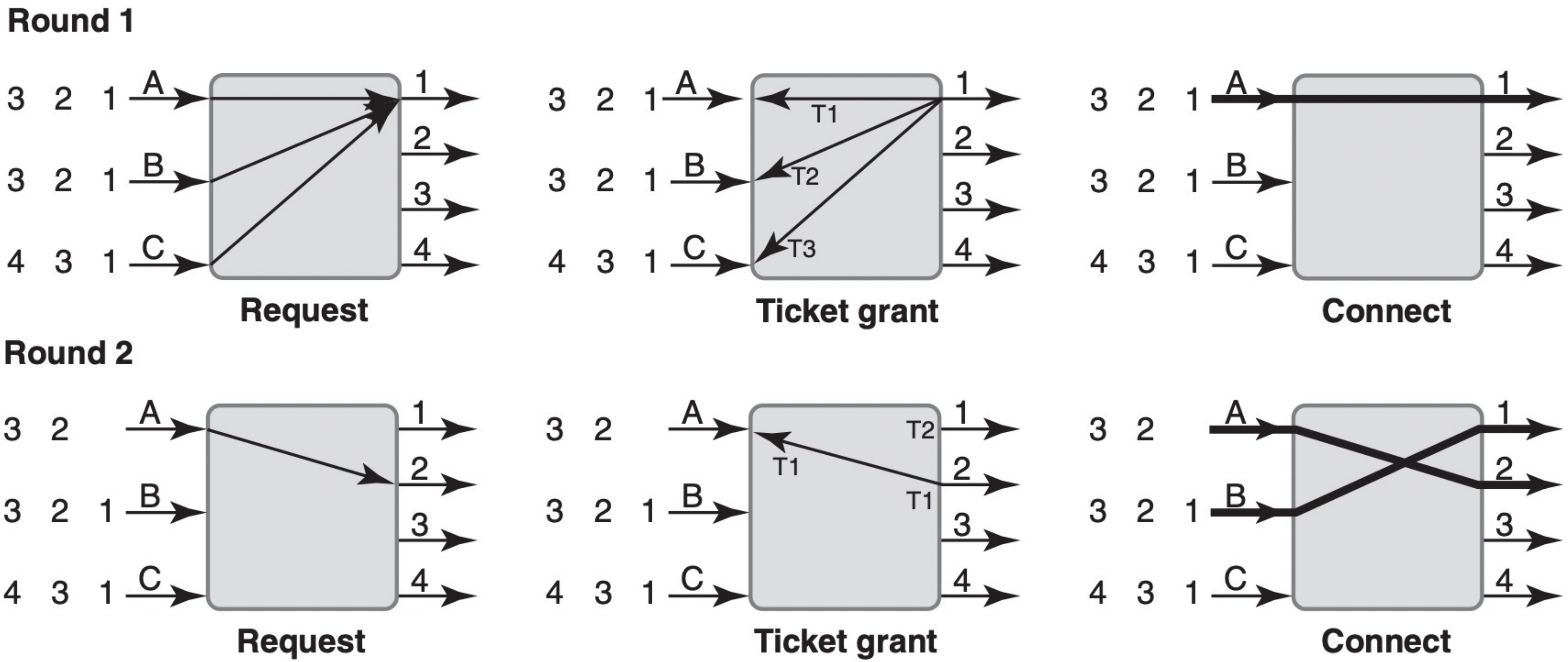


Packet remains queued until output is available to send

Take-a-Ticket

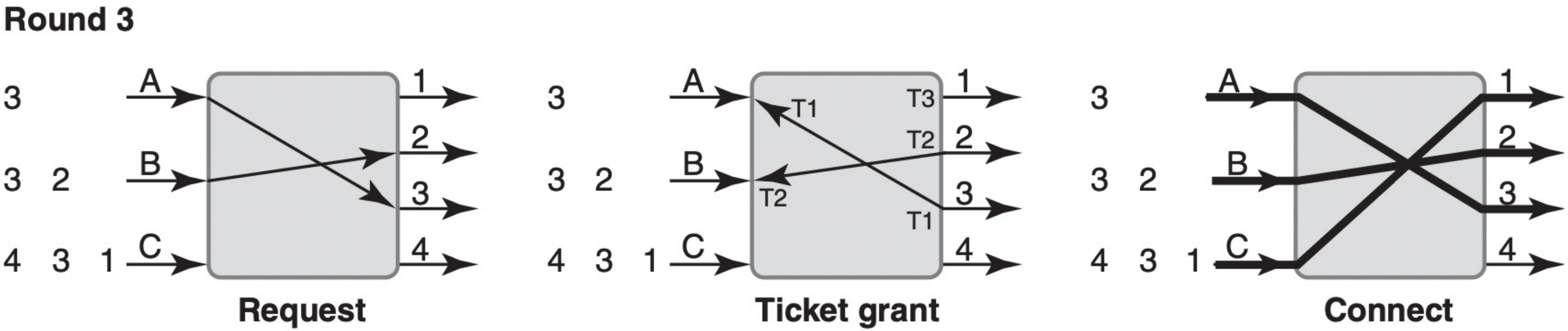
- How does input port know when to send the front element of its queue?
- Works like deli counter: each input R ”takes a ticket” for the output S at the front of its queue. S then calls out the ticket number it’s serving. When R hears its number, it sends the packet to S
- Requests, calling out numbers happens on separate control bus (very light load)

Example



Round 1
3 2 1 A
3 2 1 B
4 3 1 C
1
2
3
4
Request
T1
T2
T3
Ticket grant
Connect
Round 2
3 2 A
3 2 1 B
4 3 1 C
Request
T2
T1
T1
Ticket grant
Connect

Example (cont.)

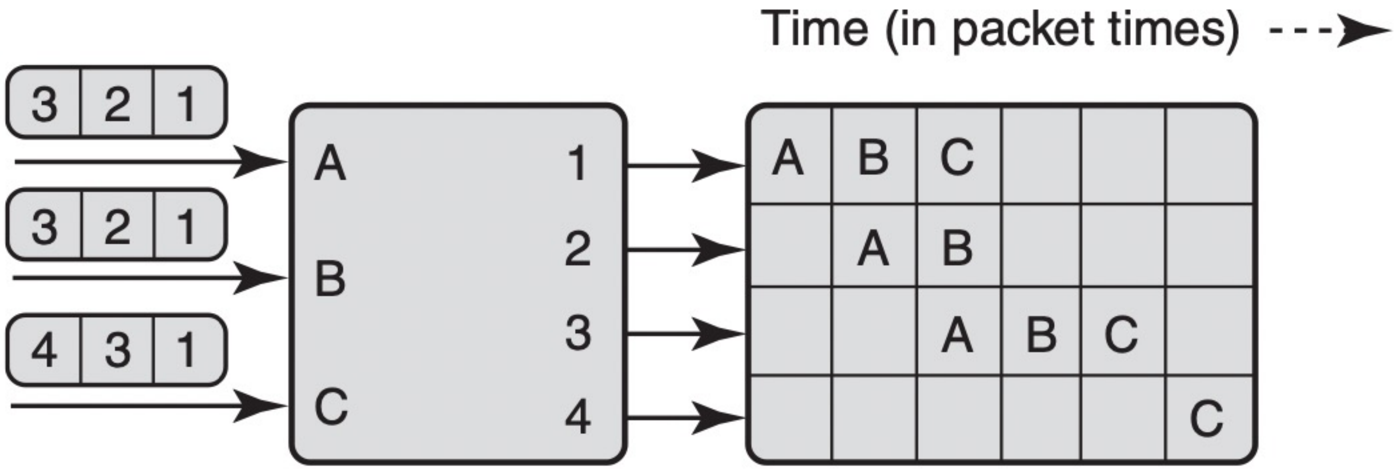


- How many more rounds are needed?
  - Draw out any remaining rounds

Example (cont)

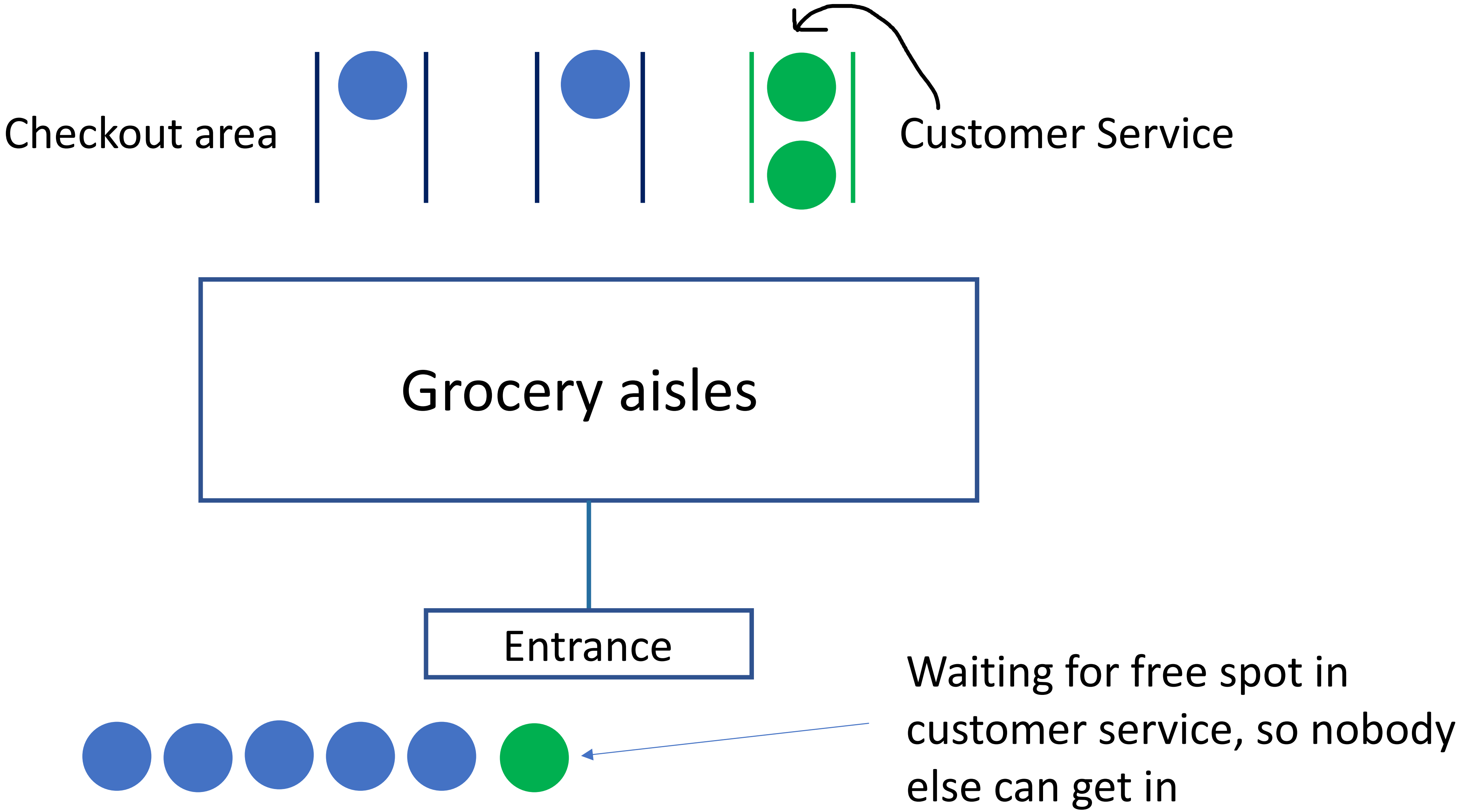
Head of Line Blocking

- What would have been the optimal number of rounds?



A	B	C			
	A	B			
		A	B	C	
					C

Example: Grocery Store



Checkout area

Customer Service

Grocery aisles

Entrance

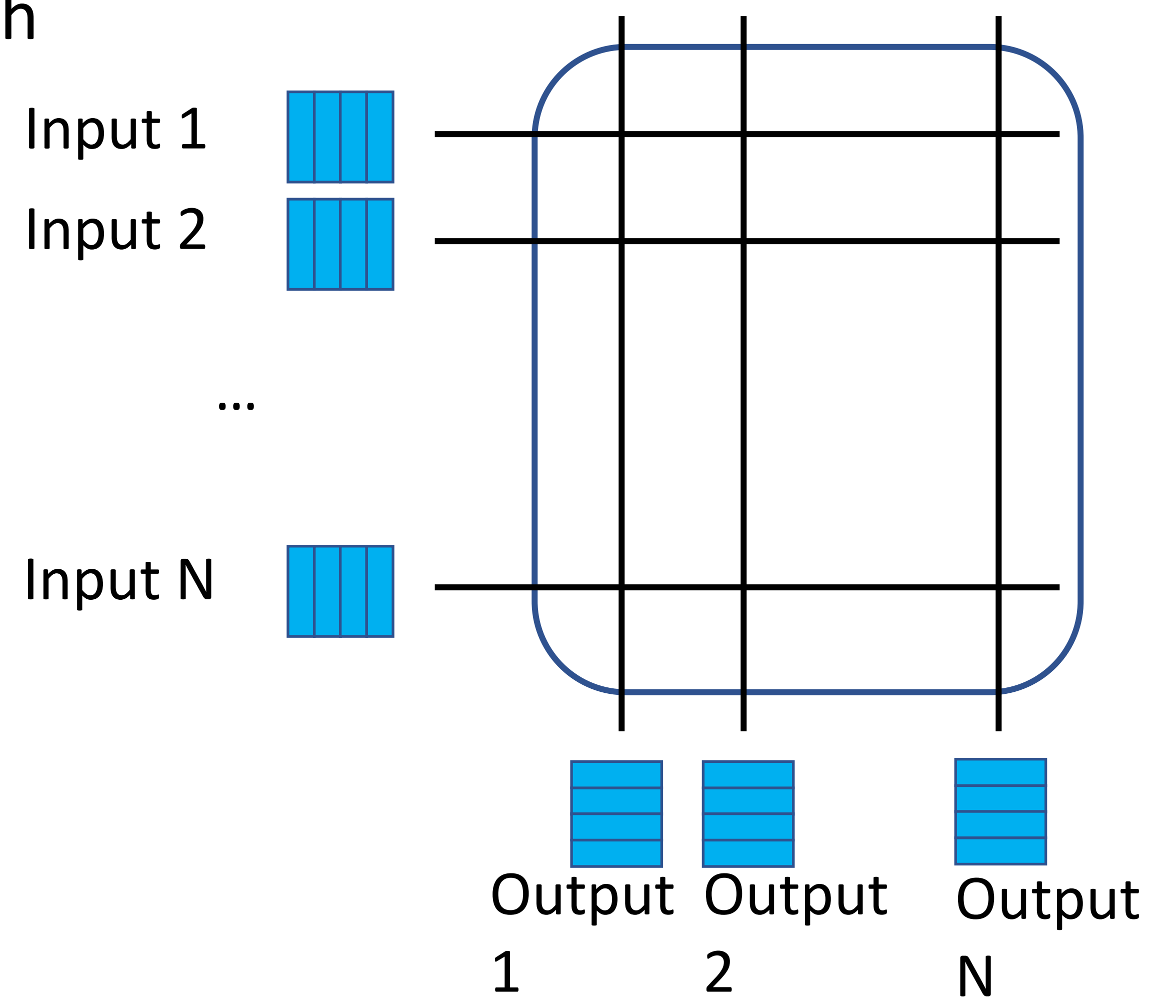
Waiting for free spot in customer service, so nobody else can get in

Avoiding HOL Blocking

- One proposal – queue only at output
- Requires fabric to run N times faster than input link (where N is number of input links)
  - If k is small, can be realized with k parallel buses
  - Can be very expensive

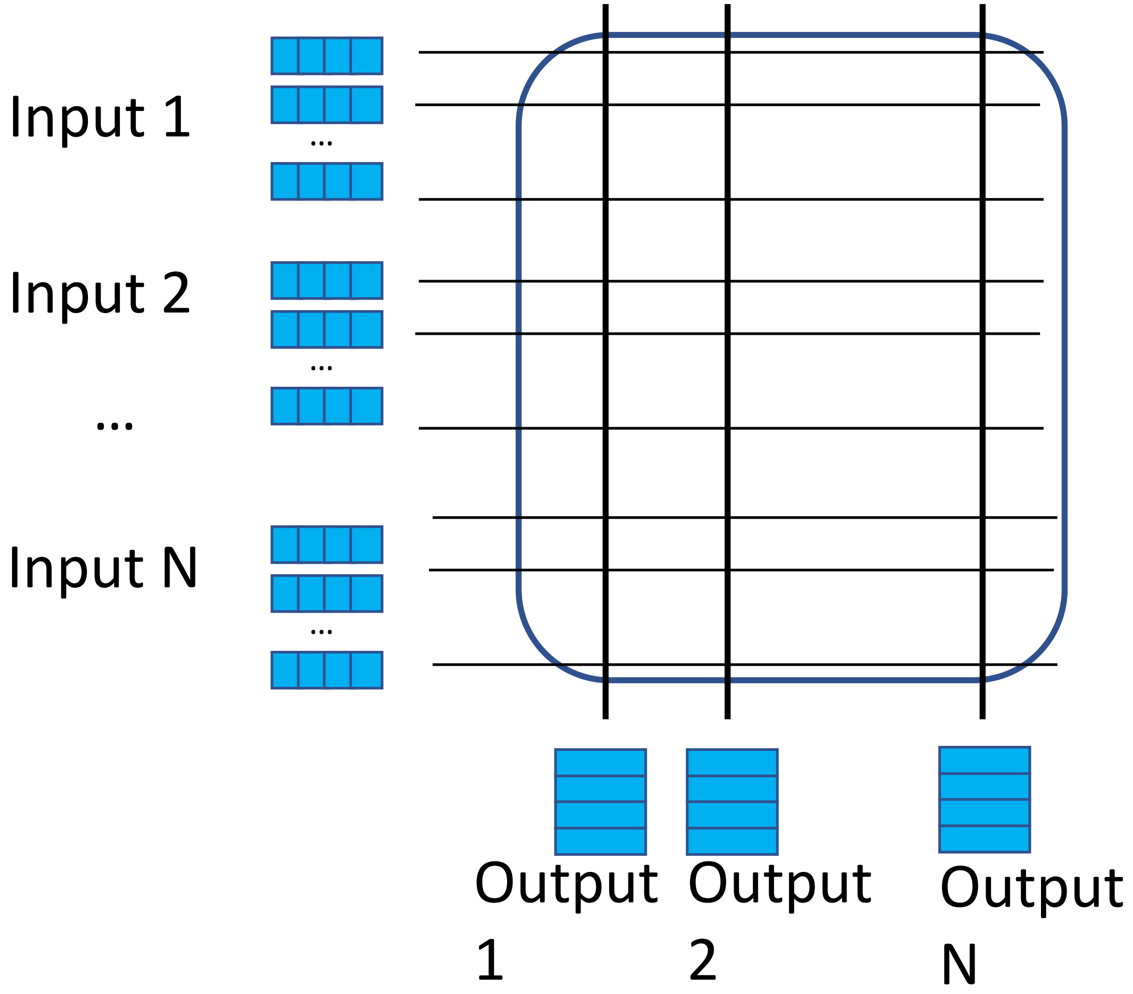
Avoiding HOL Blocking: Virtual Output Queues

- Keep separate queue for each output



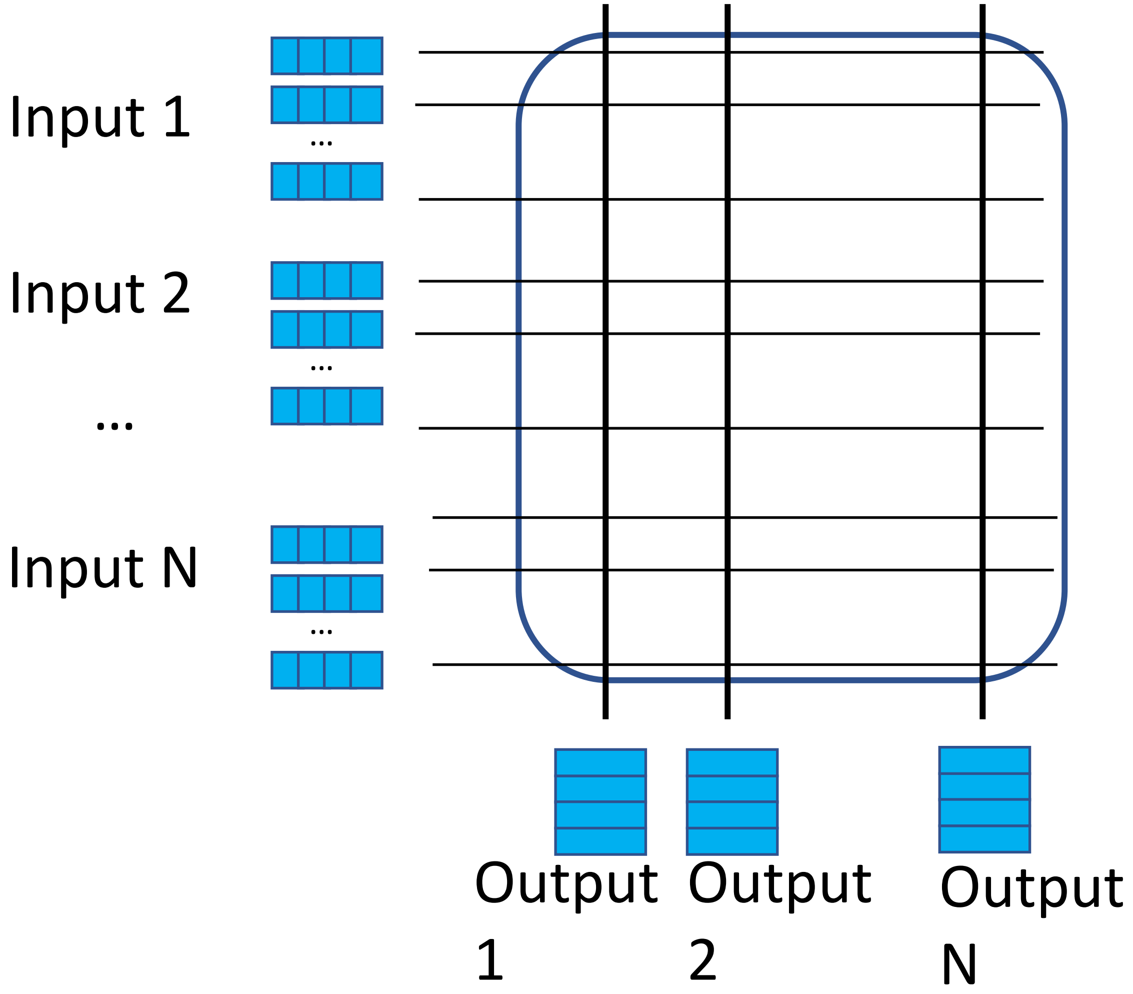
Avoiding HOL Blocking: Virtual Output Queues

- Keep separate queue for each output
    - Can make progress on each output queue separately
    - Can express request to all ports in one bitmap

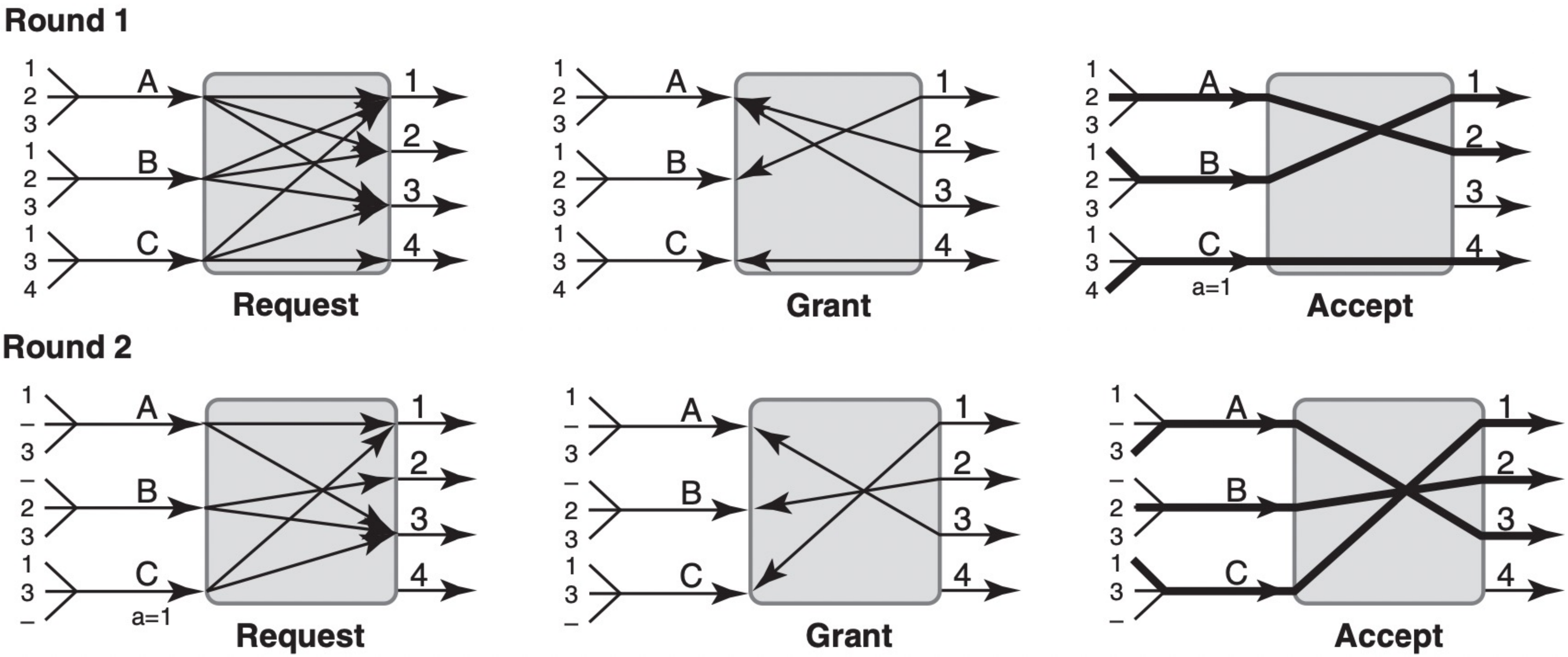


Avoiding HOL Blocking: Virtual Output Queues + Parallel Iterative Matching

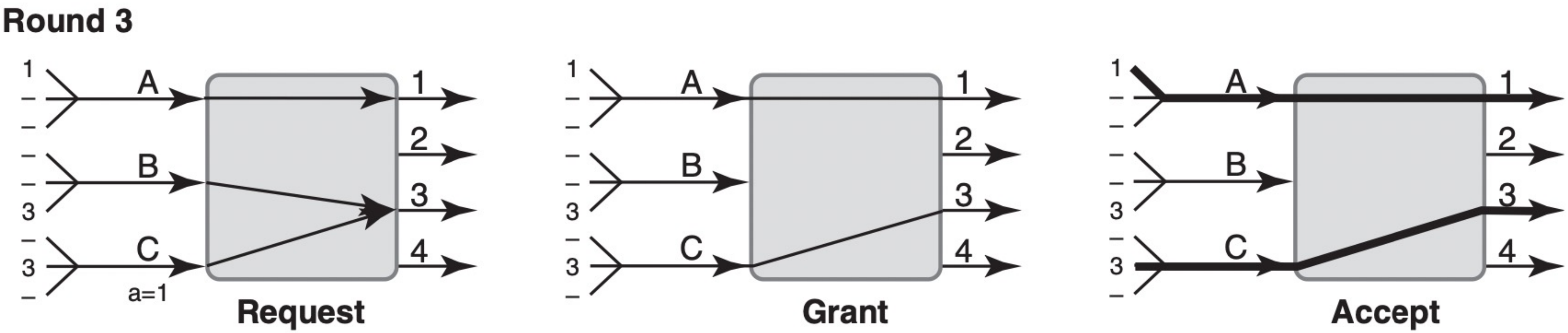
- Keep separate queue for each output
  - Can make progress on each output queue separately
  - Can express request to all ports in one bitmap



Parallel Iterative Matching



Parallel Iterative Matching



Avoiding Randomness

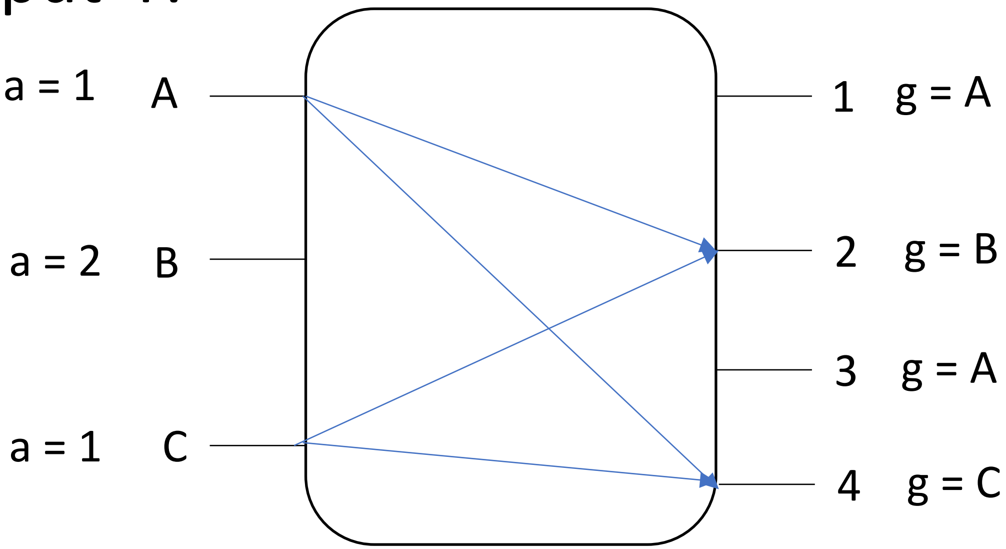
- Why avoid randomness?
  - Hard to generate random numbers fast enough
  - Multiple iterations to attain maximal matches

iSLIP

- In each step that involves choosing (Grant and Accept), choose winner in round robin manner using a rotating pointer
- Each output keeps a grant pointer, g, initialized to first input. When it has to choose which input to grant to, it chooses the input with lowest port number that is greater than or equal to g
- *After* accept phase, if output port was matched with input X, grant pointer is at (X+1) mod (number of input ports)
- Input ports each keep an accept pointer that works in the same way

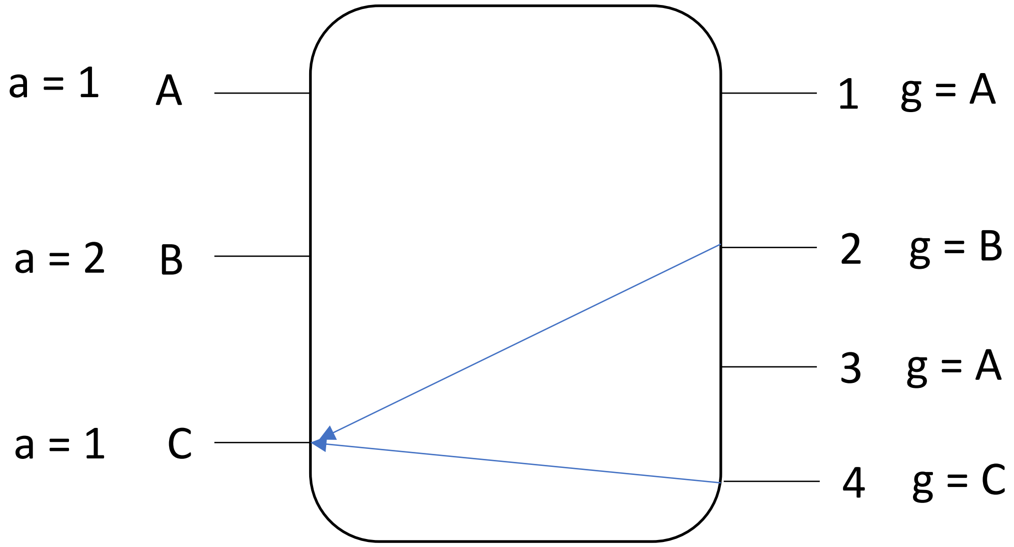
iSLIP Example

- Which input’s request does output 2 grant?
- Output 4?



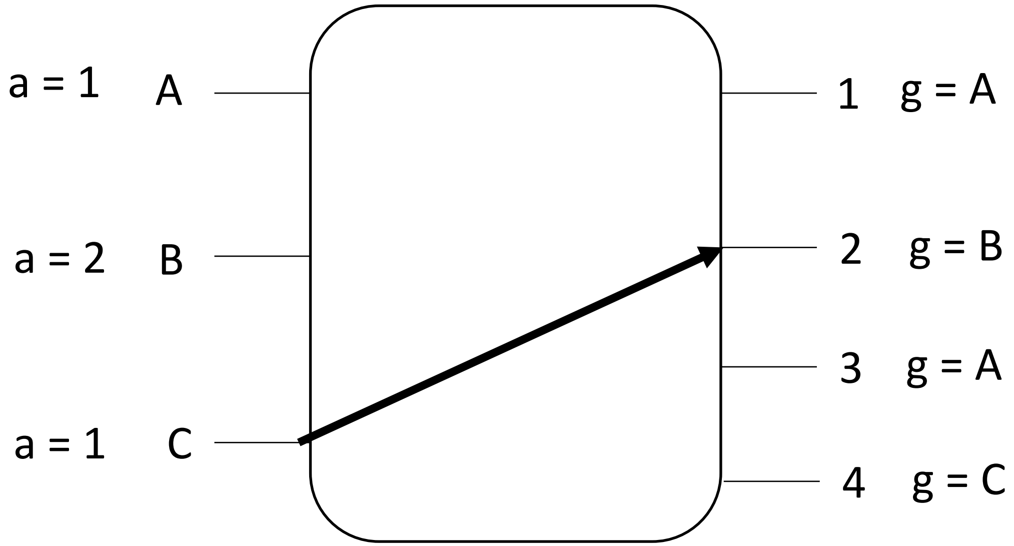
iSLIP Example

- Which output's grant does C accept?



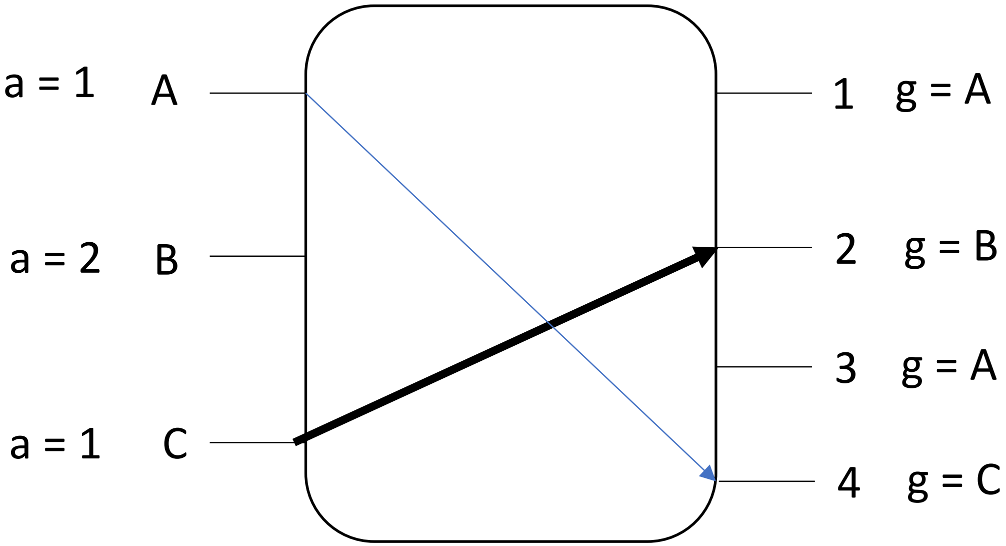
2nd Iteration?

- Which inputs would send where in a 2nd iteration?

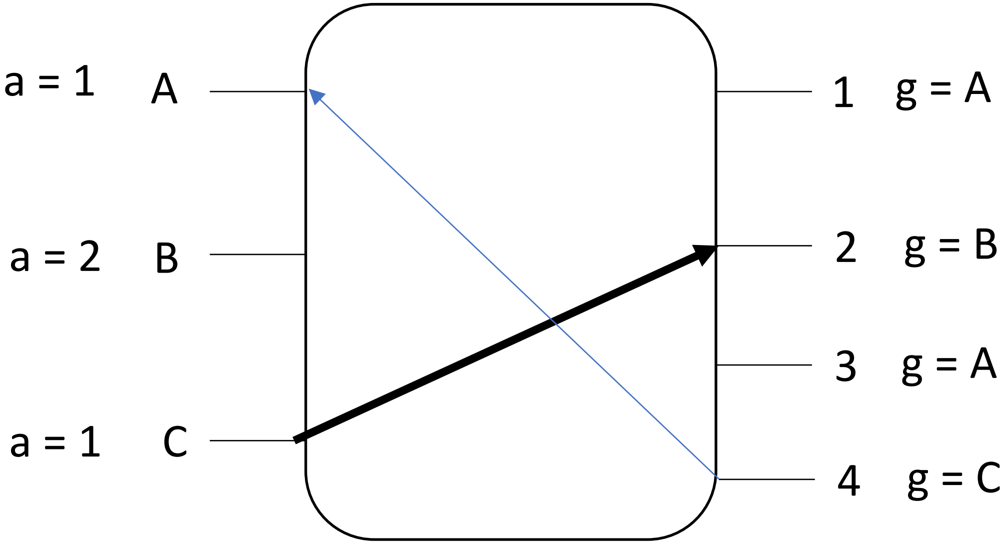


2nd Iteration?

- Which inputs would send where in a 2nd iteration?

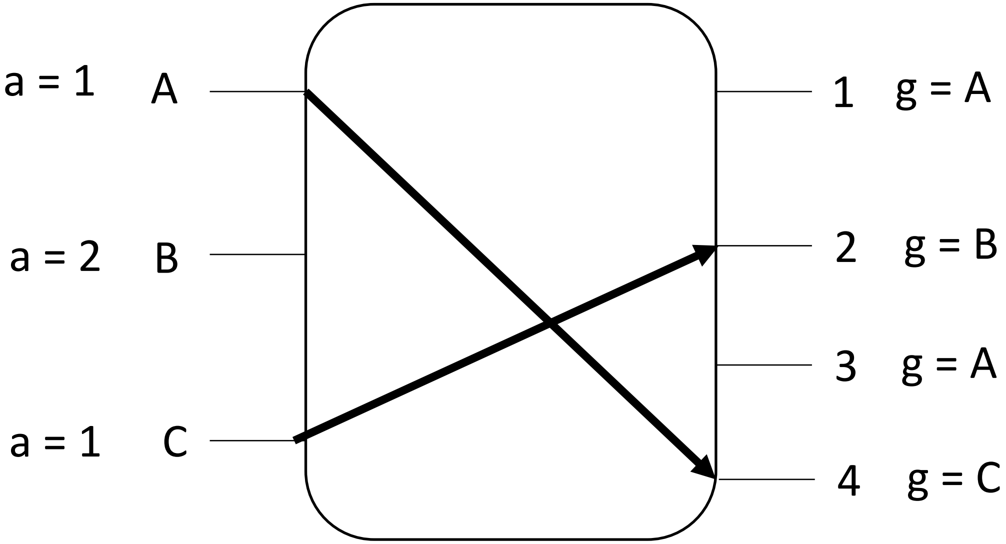


2nd Iteration

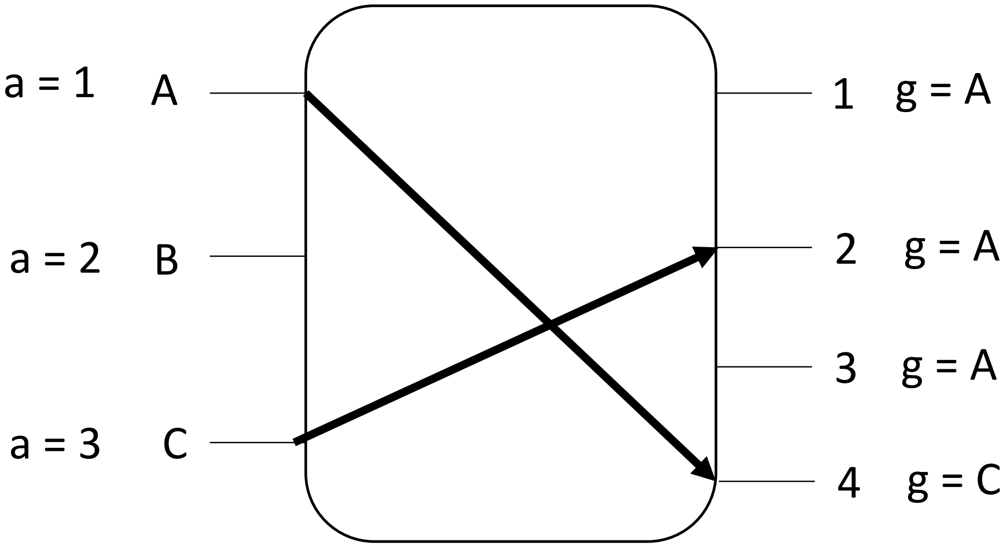


2nd Iteration

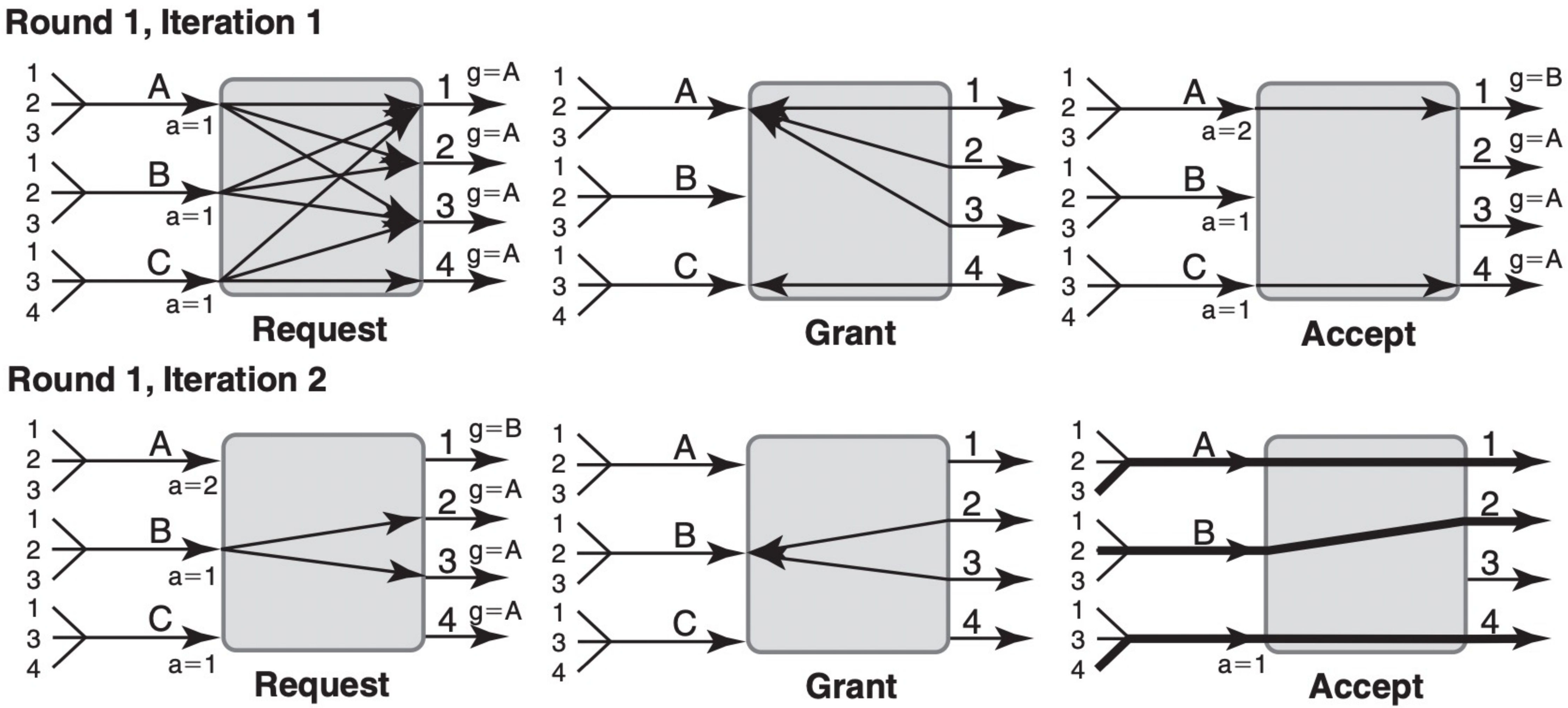
- Note: grant/accept pointers only increment after 1st iteration
- Where would grant/accept pointers be after this round?



2nd Iteration



iSLIP Round 1



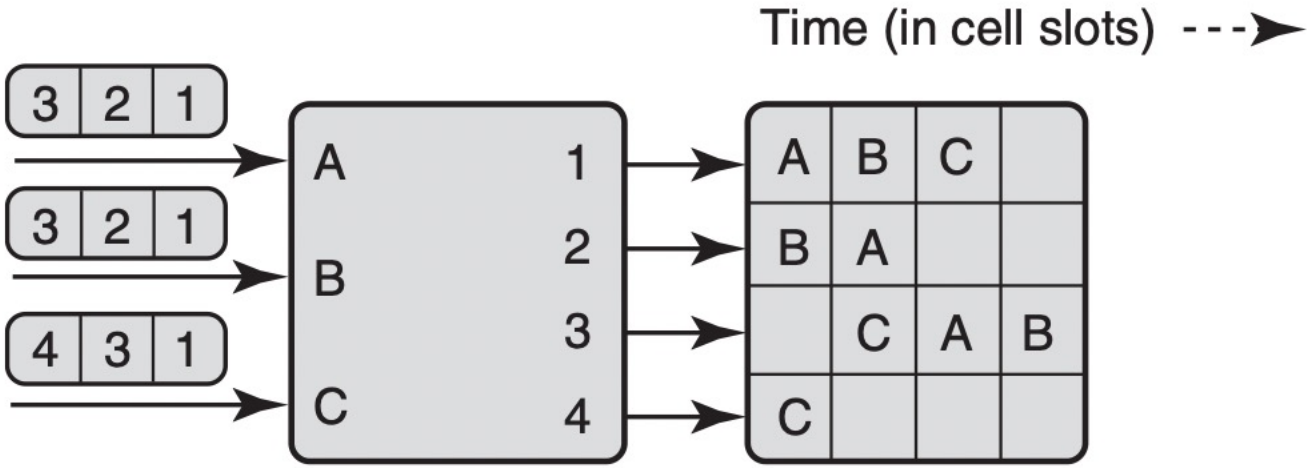
Round 1, Iteration 1

Round 1, Iteration 2

iSLIP Remaining Rounds

iSLIP Advantages

- Avoids HOL blocking
- Rotating priority provides long-term fairness (pointers are synchronized at the beginning but long-term lack of synchronization provides performance improvement)



A	B	C	
B	A		
	C	A	B
C			

Recap

- Switching is the process of physically moving packets from input to output ports
- Using a crossbar switch, N-fold speedup is possible, but finding N disjoing input-output pairs is difficult
- Take-a-Ticket system provides communication protocol between inputs/outputs, but is subject to Head-of-Line Blocking
- Parallel Iterative Matching (PIM) avoid HOL blocking by using virtual output queus (VOQs) and randomization
- iSLIP removes randomization from PIM by introducing concept of rotating priority