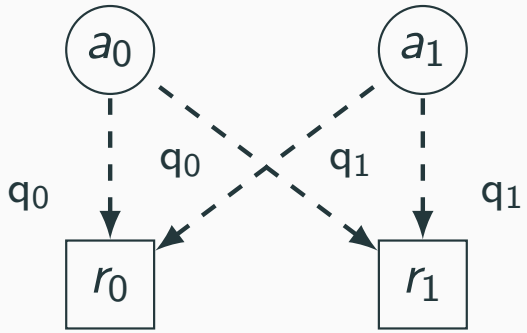
Distributed Resource Reservation - Problems

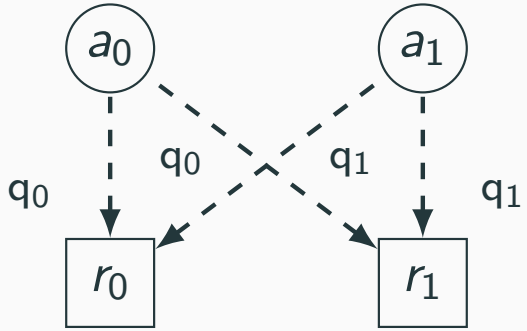


	r_0
0	
1	
2	
⋮	
n	

	r_1
1	
2	
3	
⋮	
n	

Remark: Suppose, we have a scenario where a_0 and a_1 completed lane negotiation step, and now can write (create lanes) $a_0 \to 1$, $a_0 \to 2$.

Distributed Resource Reservation - Problems

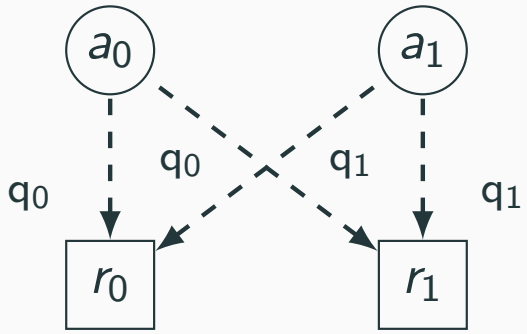


	r_0
0	
1	
2	a_0
⋮	
n	

	r_1
1	
2	
3	
⋮	
n	

Remark: When r_0 receives write(a_0, 2) messages, it stores a_0 at position slot 2 and since there is none else in r_0 request pool it will send a ready message to a_0 (a_0 is still waiting ready message from r_1 to starting consuming resources).

Distributed Resource Reservation - Problems

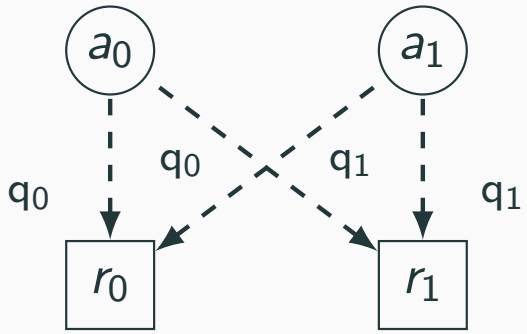


	r_0
0	
1	
2	a_0
⋮	
n	

	r_1
1	a_1
2	
3	
⋮	
n	

Remark: Now let's say the next a_1 write message arrives at r_1 and same happens again. Ready message is sent to a_1 by r_1 (a_1 still waits another ready message from r_0 to consume).

Distributed Resource Reservation - Problems



	r_0
0	
1	a_1
2	a_0
⋮	
n	

	r_1
1	a_1
2	
3	
⋮	
n	

Remark: Now let's say a_1 is written to r_0 but ready message already has been sent to a_0 - so system deadlocks.

Distributed Resource Reservation - Problems

- In the previous version of the protocol once a ready message is sent another one couldn't be sent until release message would be received.
- I think allowing ready messages to be sent if someone else writes to the smaller index to the current one, still might result in unsafe situation too.
- Solution is to introduce few more message exchanges with pre-ready step and also prohibiting ready message sending while someone has locked a resource.

Distributed Resource Reservation - 2^{nd} Protocol Part

- Resource keeps sending pre-ready message to agents, if:
  - Read-pointer is not locked;
  - A received write message contains the new minimum value of the request pool.
- Once agent received all pre-ready message:
  - it sends lock messsages to all interested resources;
- If resource receives a lock message it can reply in two ways:
  - READY message if read pointer still points at that agent (also stops sending pready messages).
  - DENY message if someone wrote to the resource with the new minimum value.
- Once agent receives all DENY/READY messages it can act in two ways:
  - if all messages were READY then consume resources;
  - if exists a DENY message then send unlock messages to resources which sent READY message.
- If resource receives a unlock message it continues sending pre-ready messages.