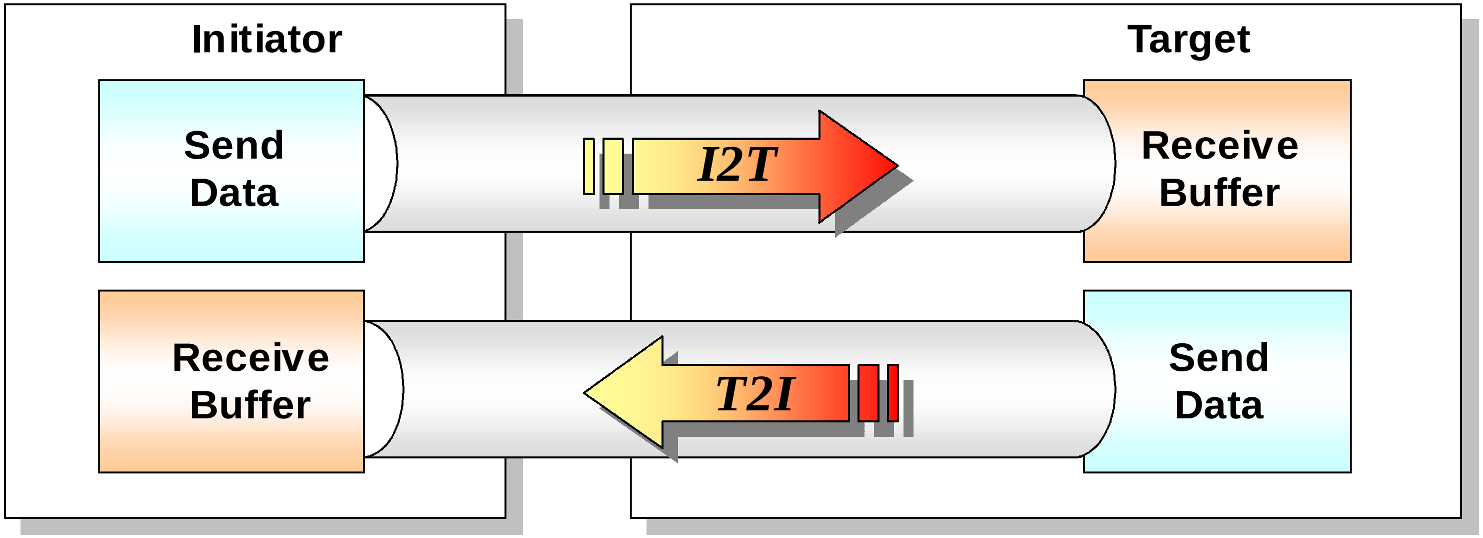
Flags across the Wire

September 20, 1999

Akihiro Shimura

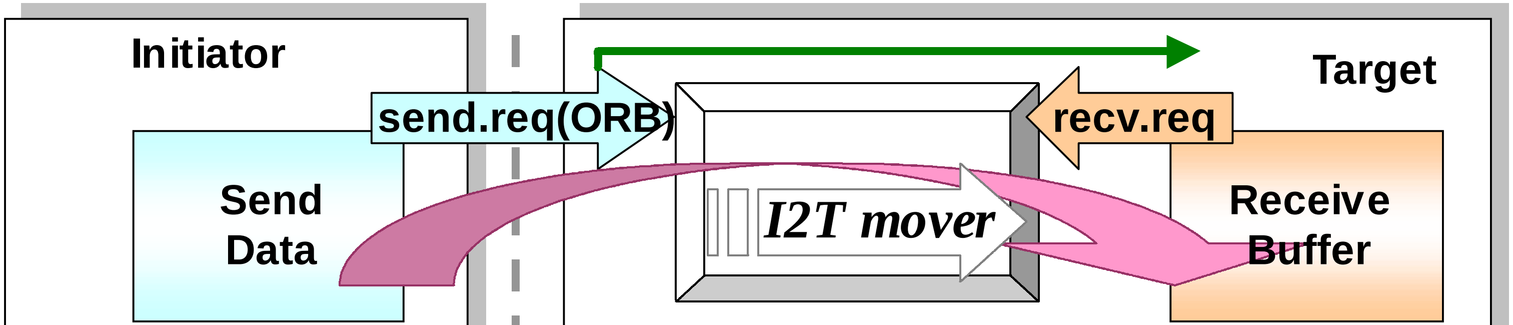
CANON INC.

Data flow Basics

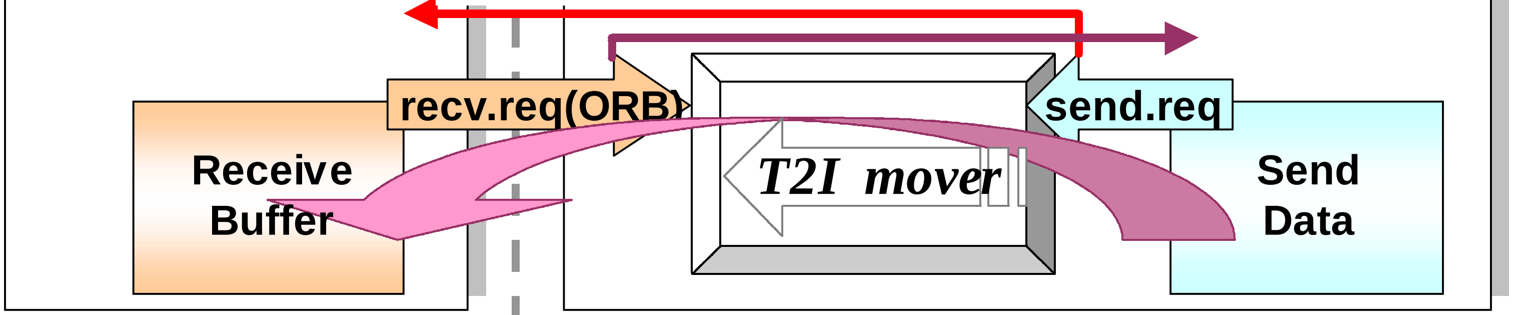


In both directions, data flows when
Receiver is ready to receive data *AND* Sender has data to send, regardless of the order that which becomes available first.
Data does not flow when Receiver is not ready,
even if Sender has data. [flow is blocked]
Data does not flow when Sender does not have sending data,
even if Receiver is ready. [idle]

PPDT flow model



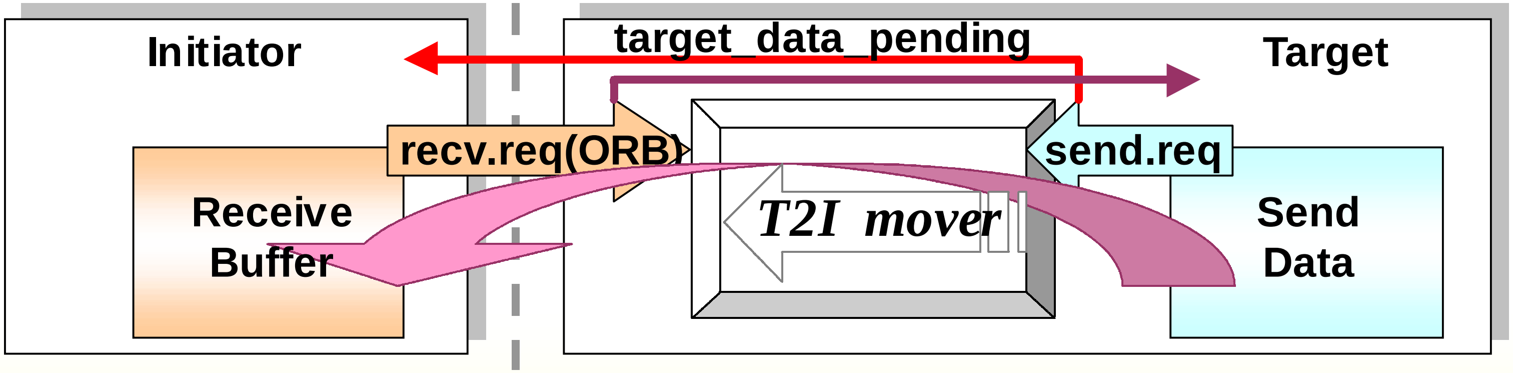
Actual data transfer is done by the *target* (I2T mover).
“I2T mover”(target) knows availability of sending data via ORB.
“I2T mover”(target) knows readiness of receiver internally.
Initiator does *NOT* know the readiness of receiver.



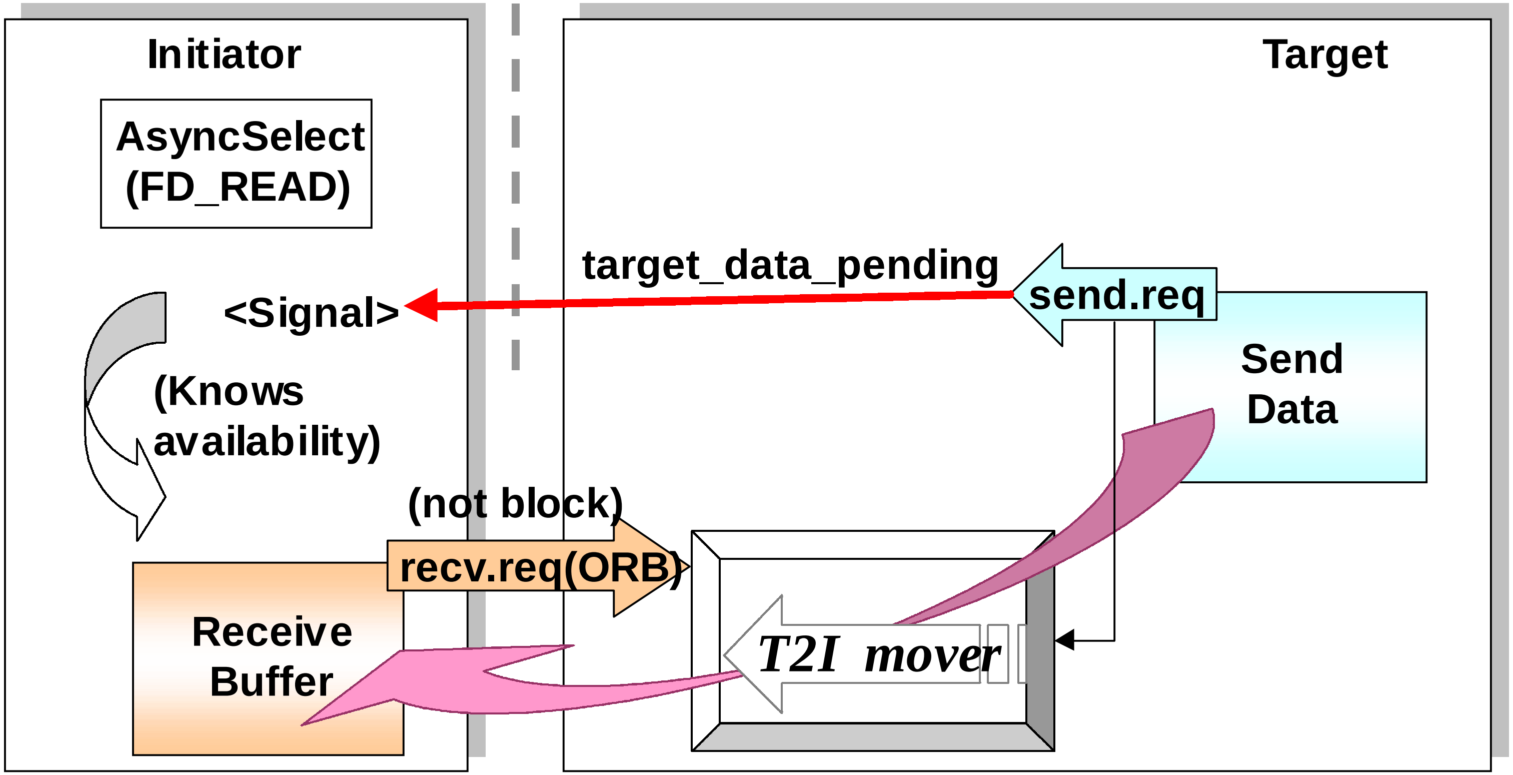
Actual data transfer is done by the *target* (T2I mover).
“T2I mover”(target) knows availability of sending data internally.
“T2I mover”(target) knows readiness of receiver via ORB.
Initiator *DOES* know the availability of sending data via flag.

“target_data_pending” flag

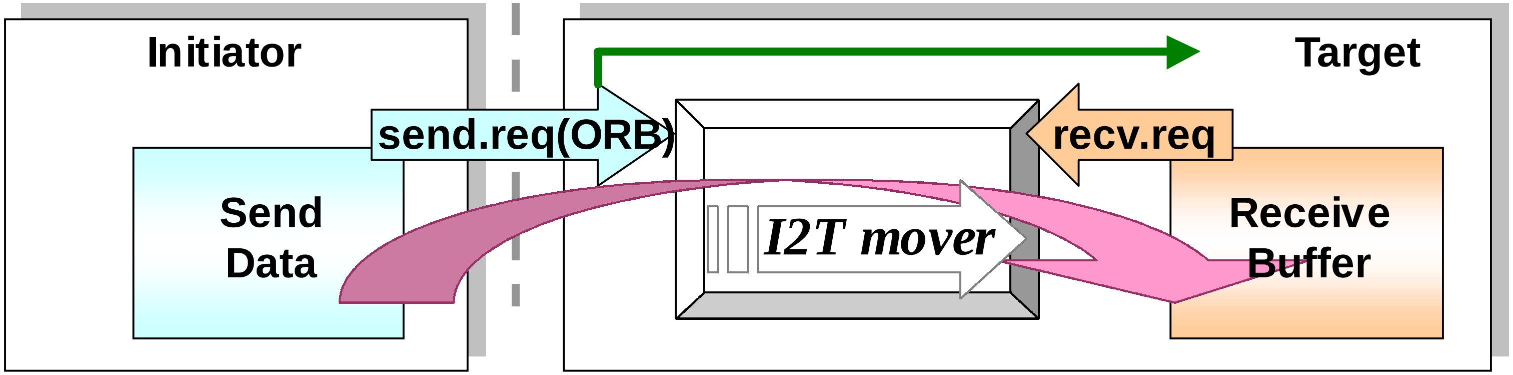
- The *“target_data_pending”* flag does not help data flow control itself.
- By priorly knowing the availability of target’s sending data, initiator may be able to avoid the possible block of the *receive OPERATION.* (=supports Select() API operation)
- To improve the performance, target should be informed whether the request should immediately complete or not. (Otherwise, initiator will need to signal receive request with single byte buffer to ensure the request complete immediately.)



“*target_data_pending*” and Select() API



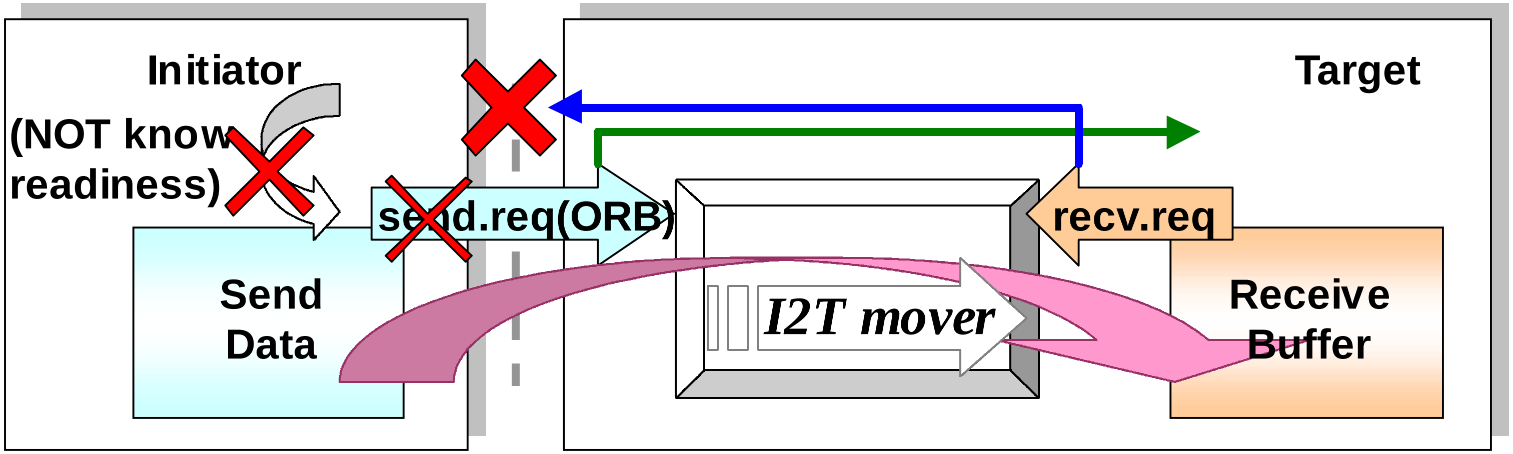
Flag indicates target's readiness?



- Without priorly knowing the readiness of target's receive, initiator cannot avoid the possible block of the *send OPERATION.*
- To avoid the possible block of the operation, target should be informed whether the request should immediately complete or not. (Otherwise, initiator has no way to avoid blocking of the operation.)

– Current solution seems to assume that receive operation is driven by the availability of the other end, while send operation does not care the readiness of the other end.

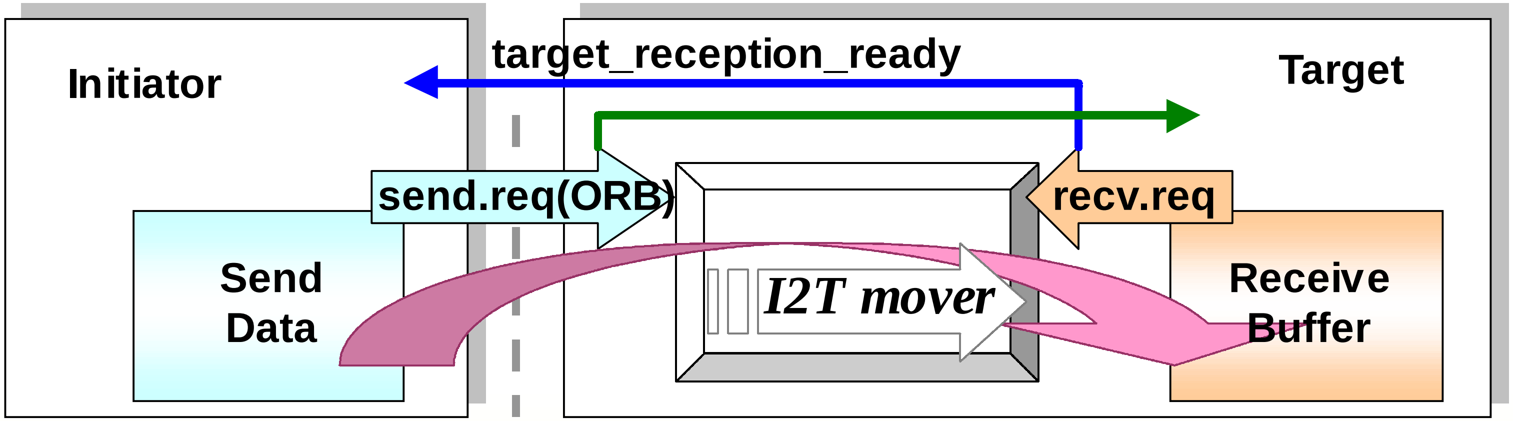
Select() API?



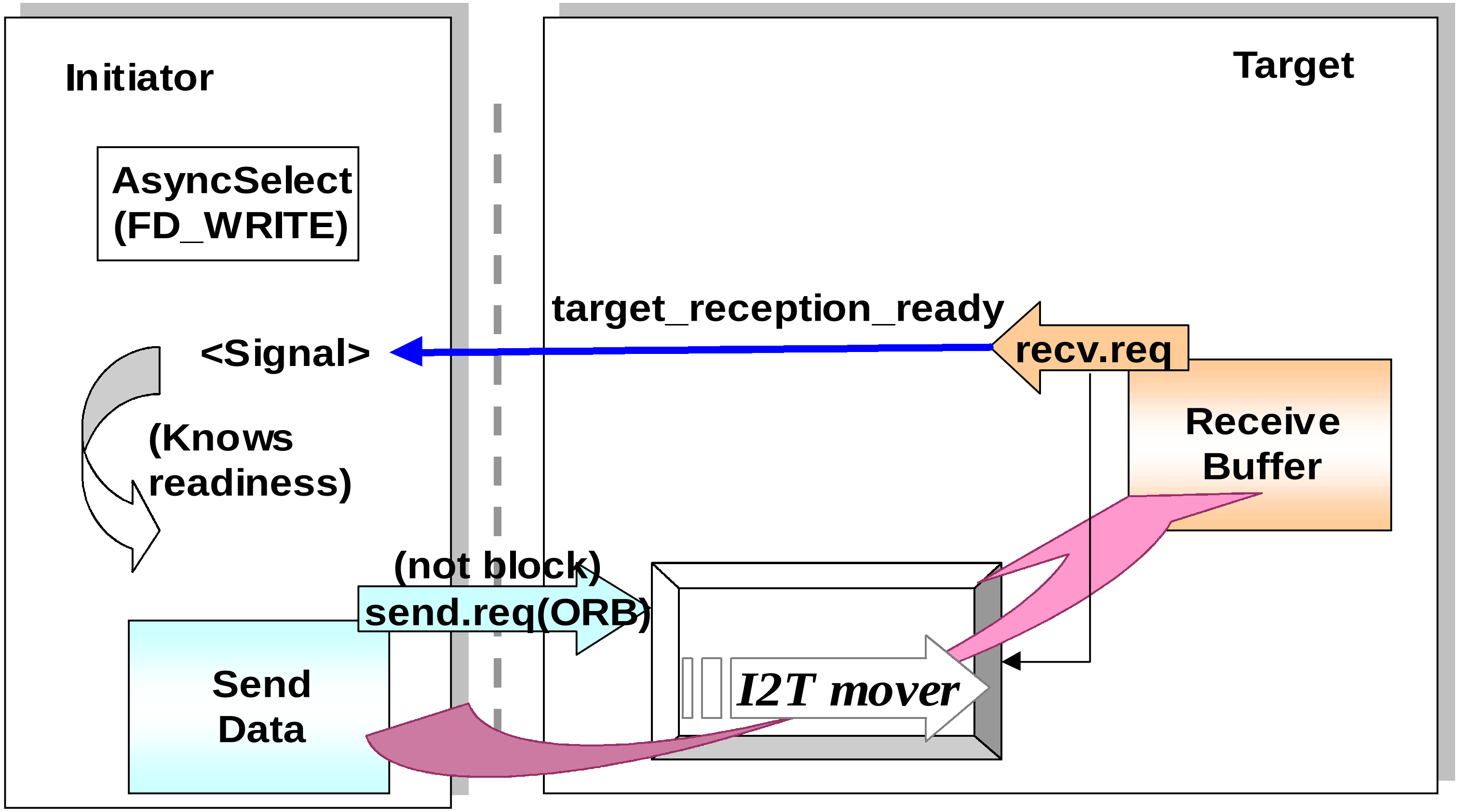
- **Because Initiator has no knowledge about the readiness of the target receive, Initiator won’t be able to support Select() API operation without assuming the existence of additional intermediate buffer in the initiator.**

If initiator knows the readiness...

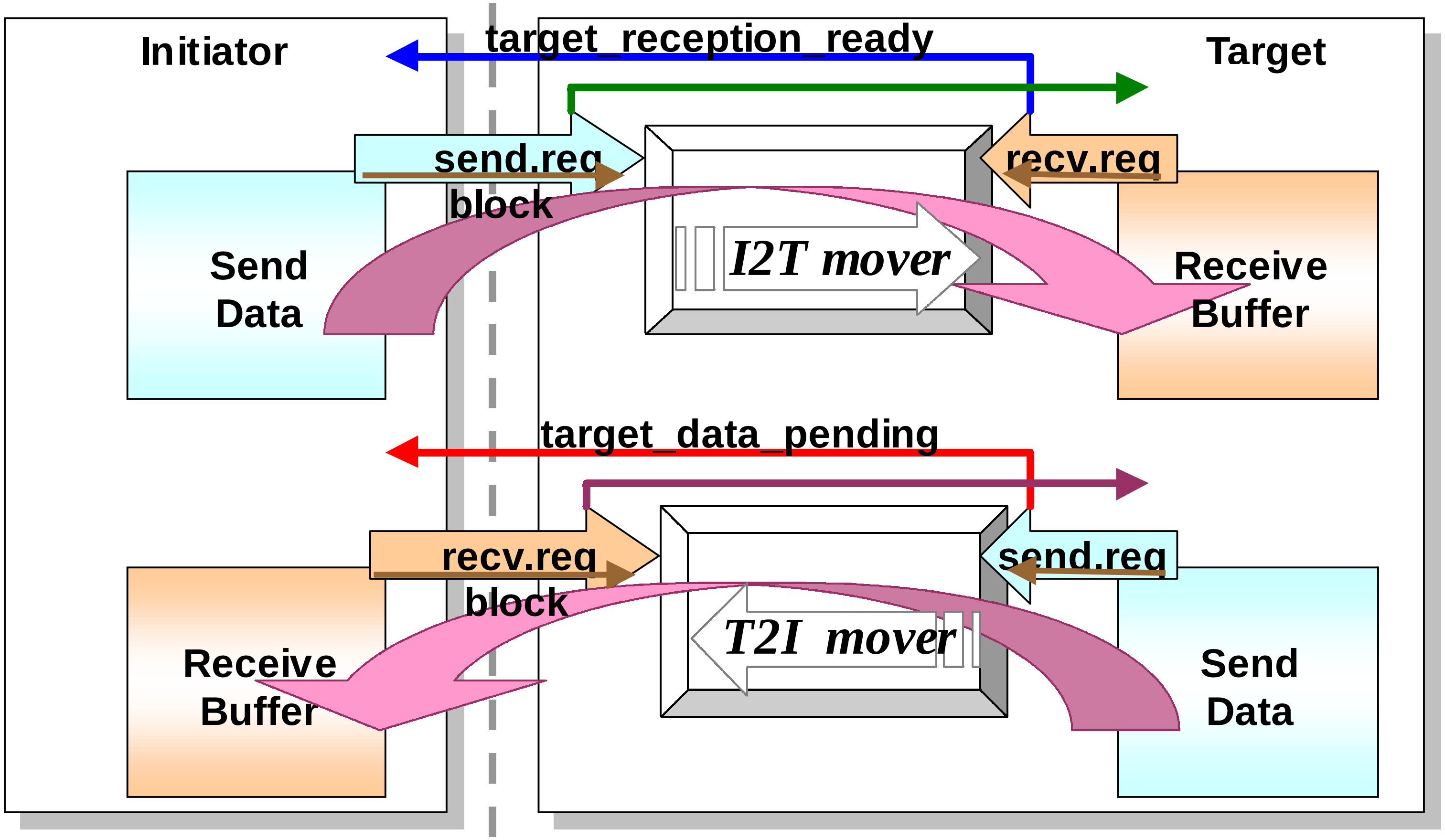
- **By priorly knowing the readiness of target's receive, initiator may be able to avoid the possible block of the *send OPERATION. (=Support Select() API Operation)***
- **To improve the performance, target should be informed whether the request should immediately complete or not. (Otherwise, initiator will need to signal send request with single byte buffer to ensure the request completes immediately.)**



“target_reception_ready” and Select() API



Flags


Initiator
target_reception_ready
Target
send.req
recv.req
block
I2T mover
Send
Data
Receive
Buffer
target_data_pending
recv.req
send.req
block
T2I mover
Receive
Buffer
Send
Data

Summary

- **Solution**
  - **Add “target_reception_ready” flag in the status_block and QUEUE_INFO.**
  - **Add “block” flag in the ORB.**
- **Others ...**