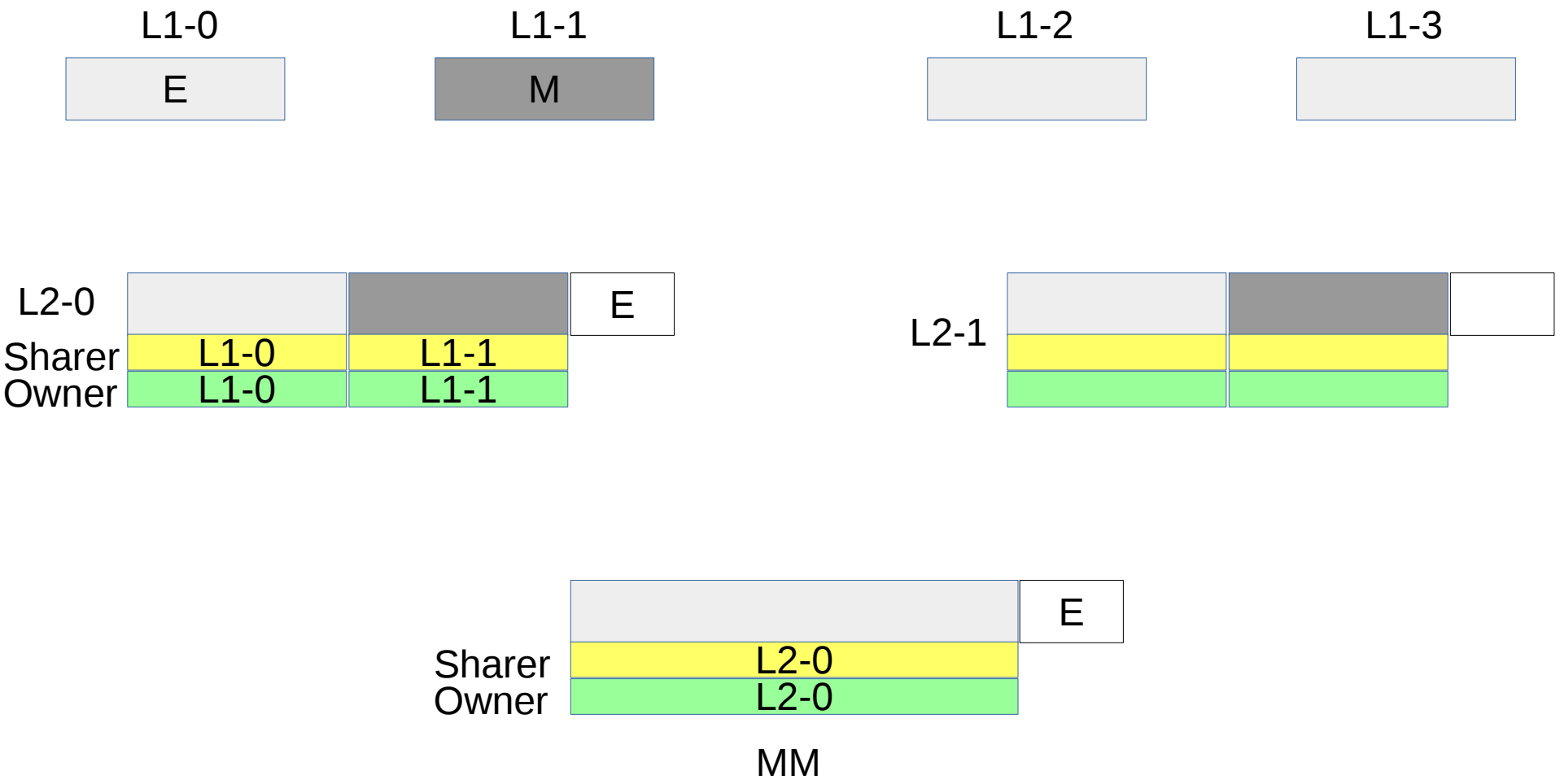
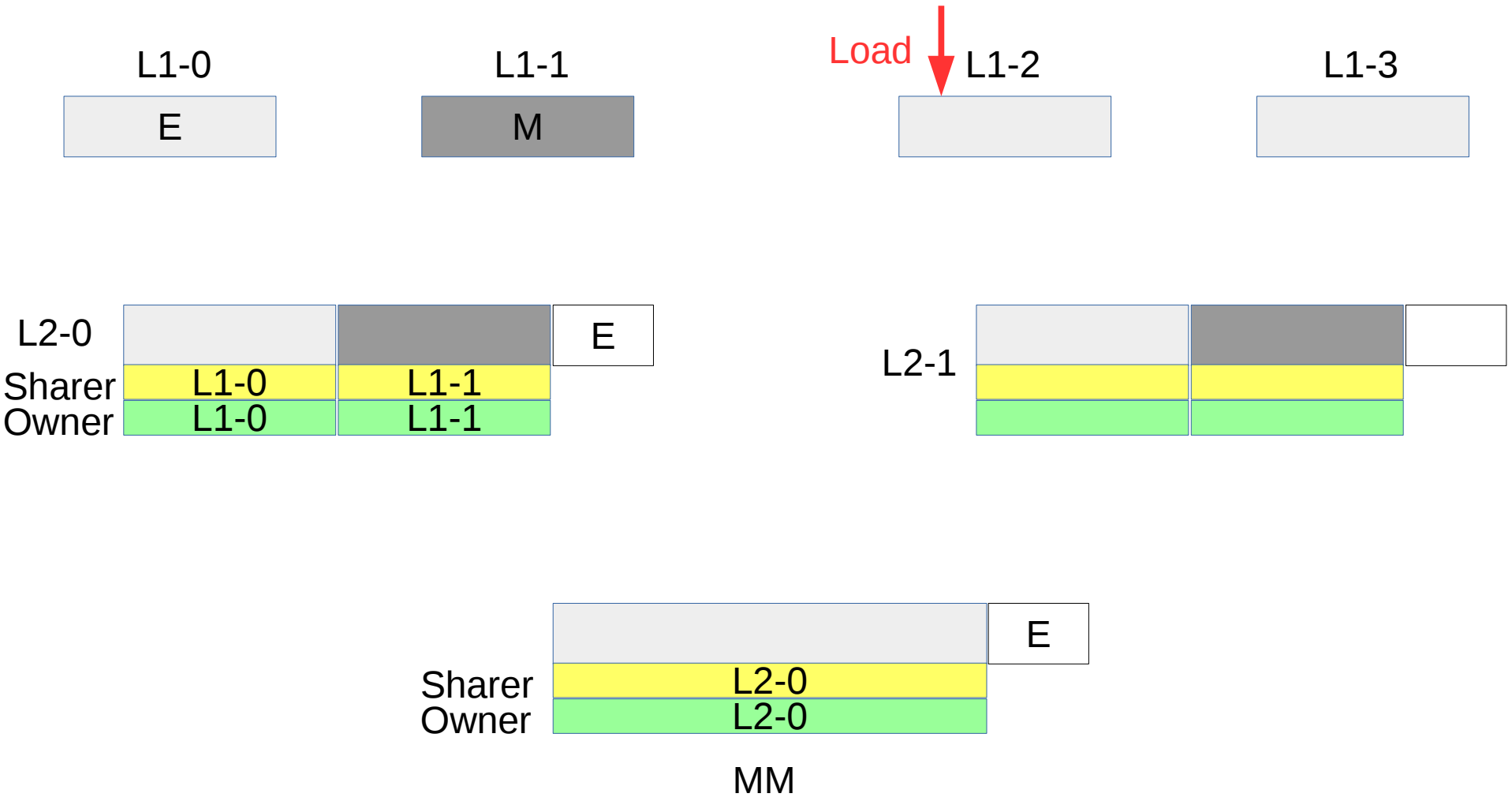
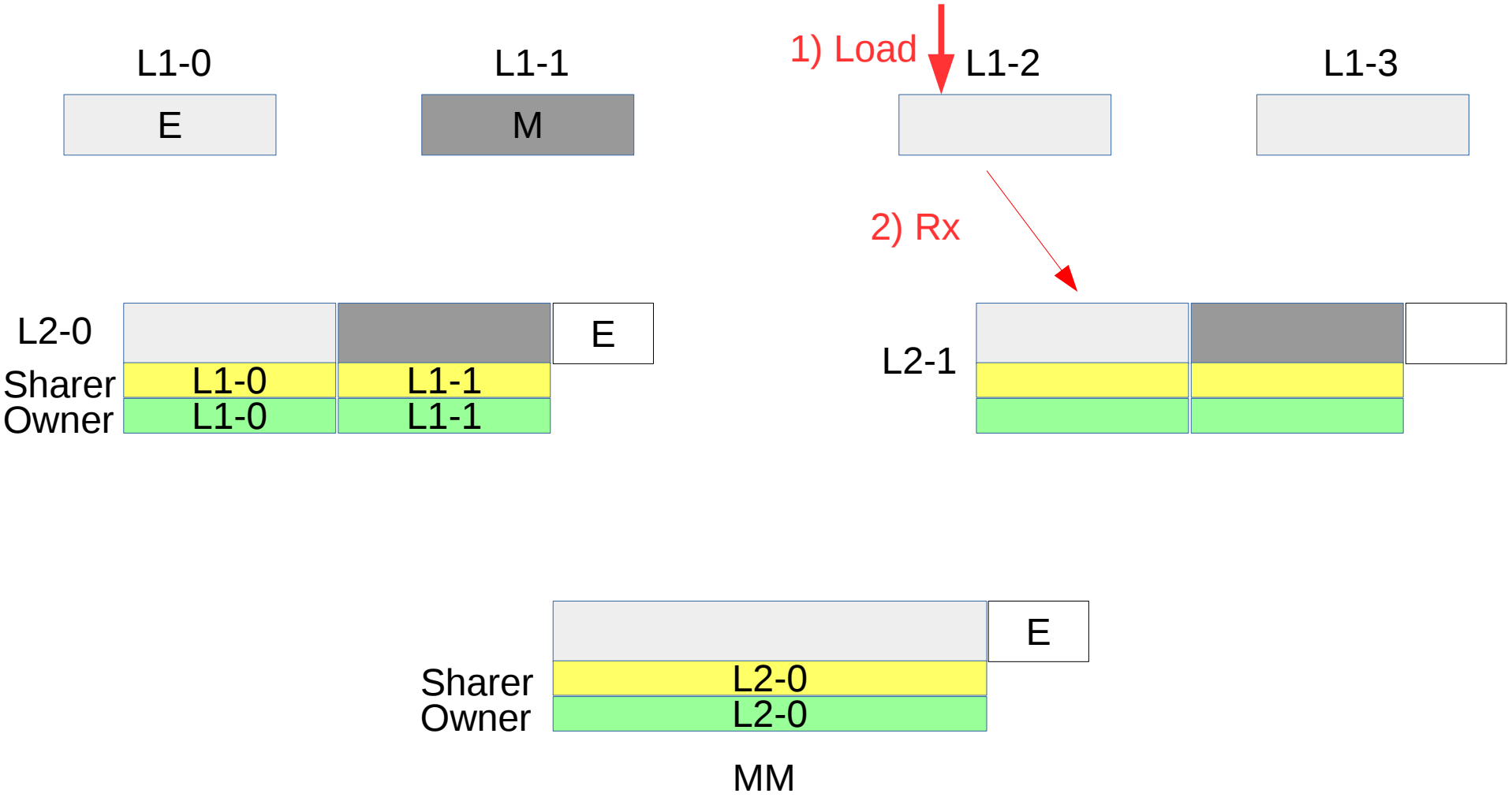
Baseline


L1-0
L1-1
L1-2
L1-3
E
M
L2-0
E
Sharer
L1-0
L1-1
Owner
L1-0
L1-1
L2-1
E
Sharer
L2-0
Owner
L2-0
MM

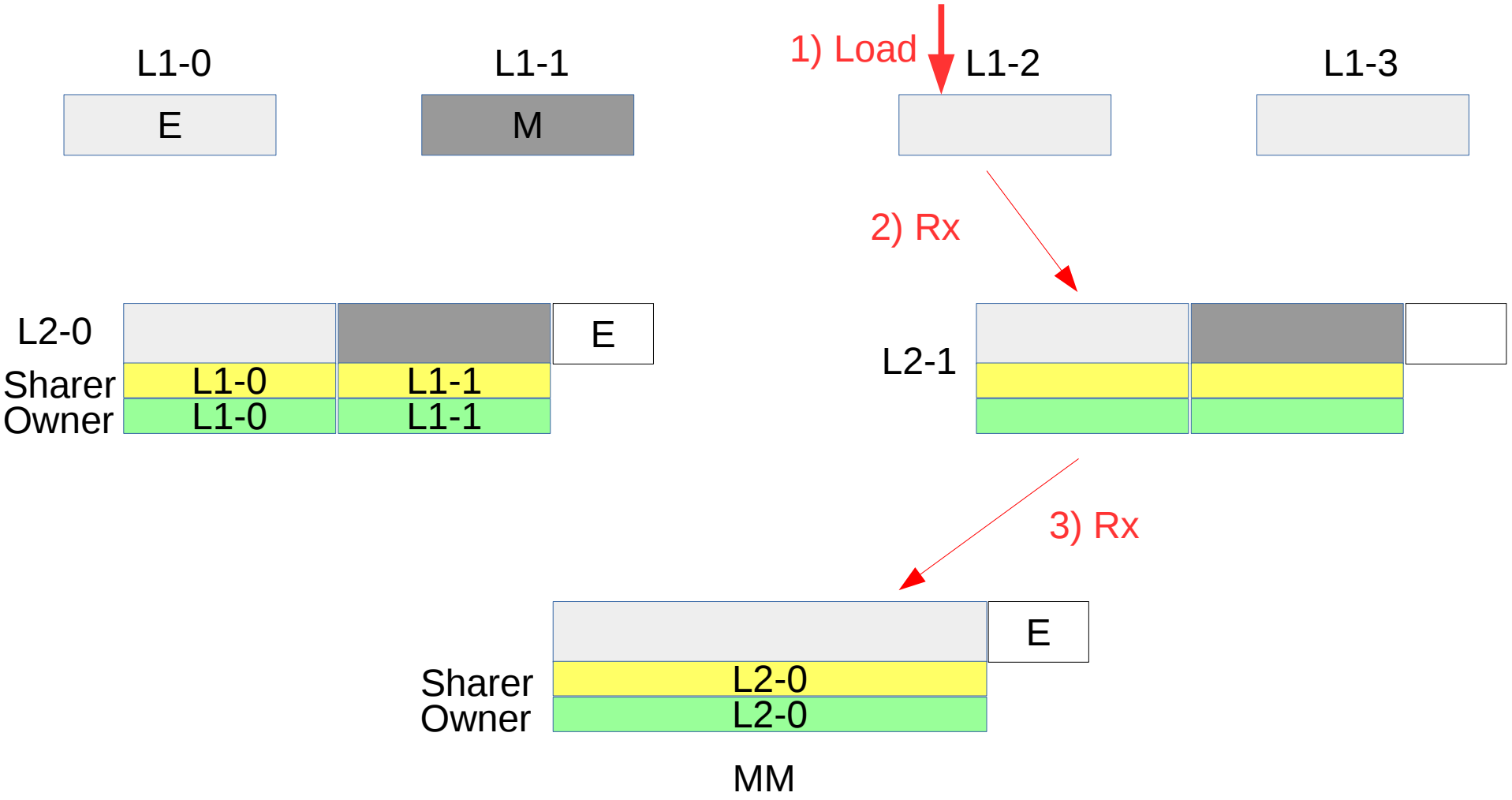
1 - Load


L1-0
L1-1
Load
L1-2
L1-3
E
M
L2-0
E
Sharer
L1-0
L1-1
Owner
L1-0
L1-1
L2-1
E
Sharer
L2-0
Owner
L2-0
MM

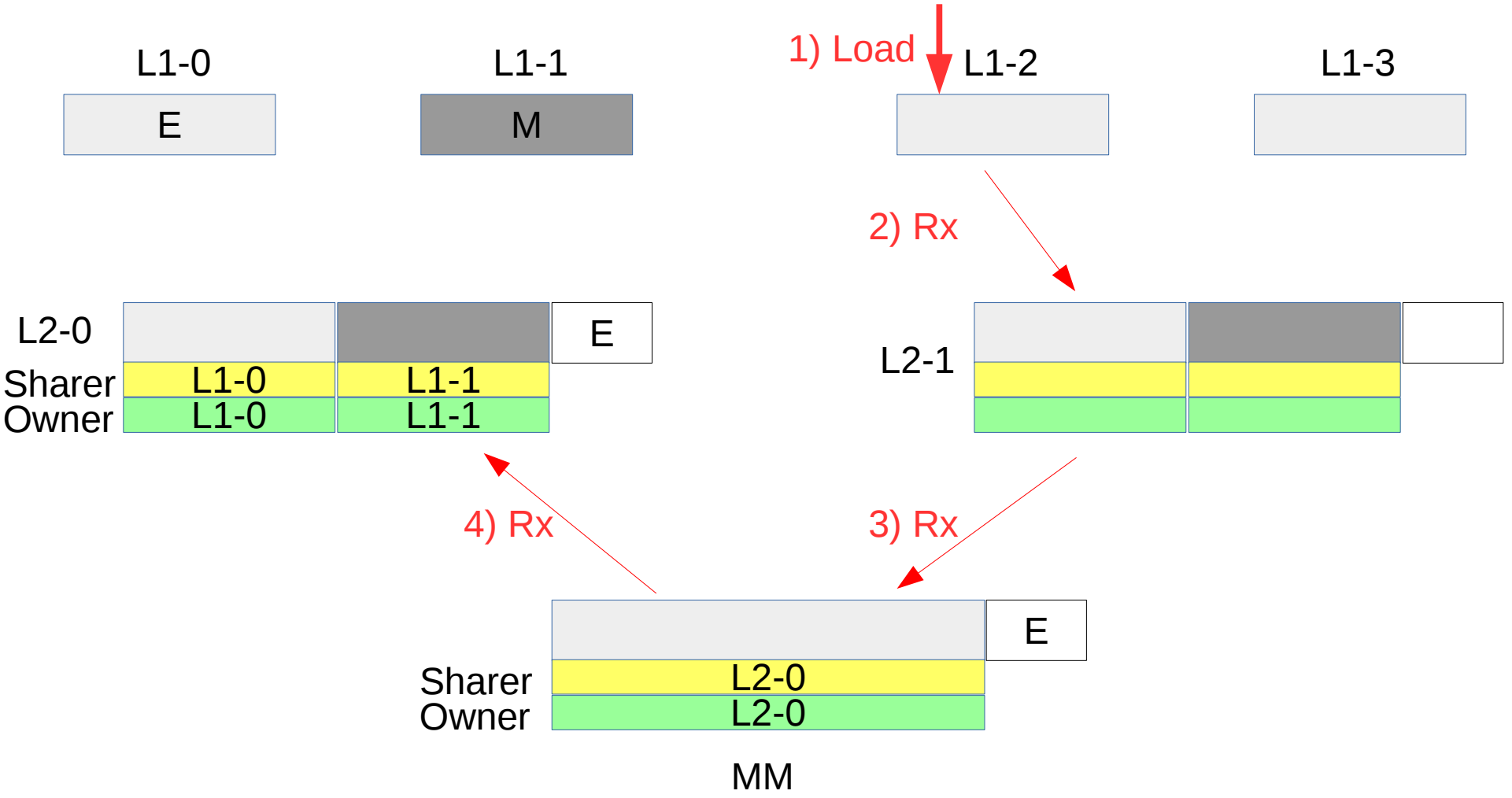
2- Rx L1-2 to L2-1
(L2-1 subblocks have no owner and sharers – sanity check)



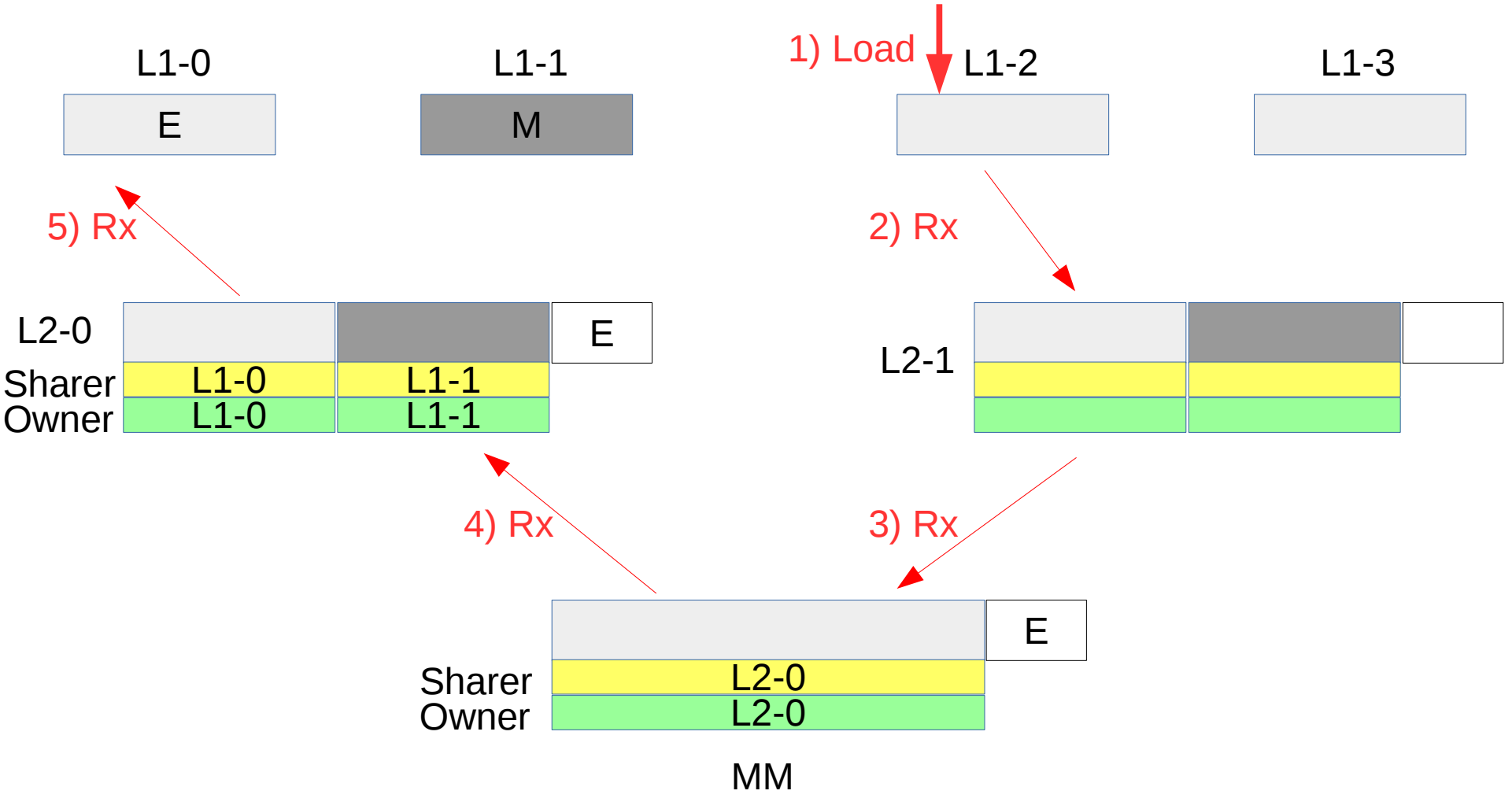
3- Rx L2-1 to MM


L1-0
L1-1
1) Load
L1-2
L1-3
E
M
2) Rx
L2-0
E
Sharer
L1-0
L1-1
Owner
L1-0
L1-1
L2-1
3) Rx
E
Sharer
L2-0
Owner
L2-0
MM

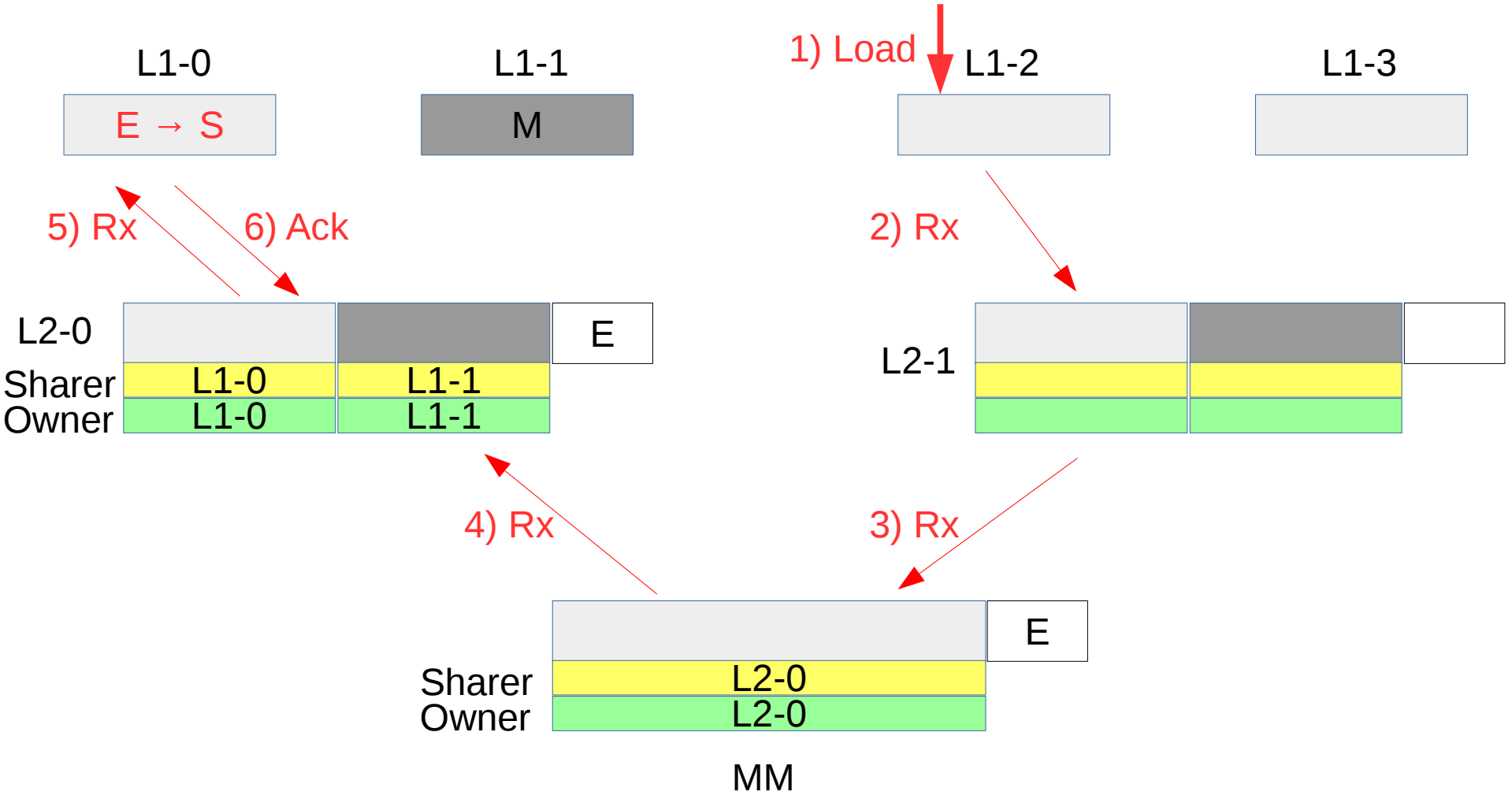
4-Rx from MM to L2-0
(based on owner (the owner is set when the upper block is in either M/O/E))



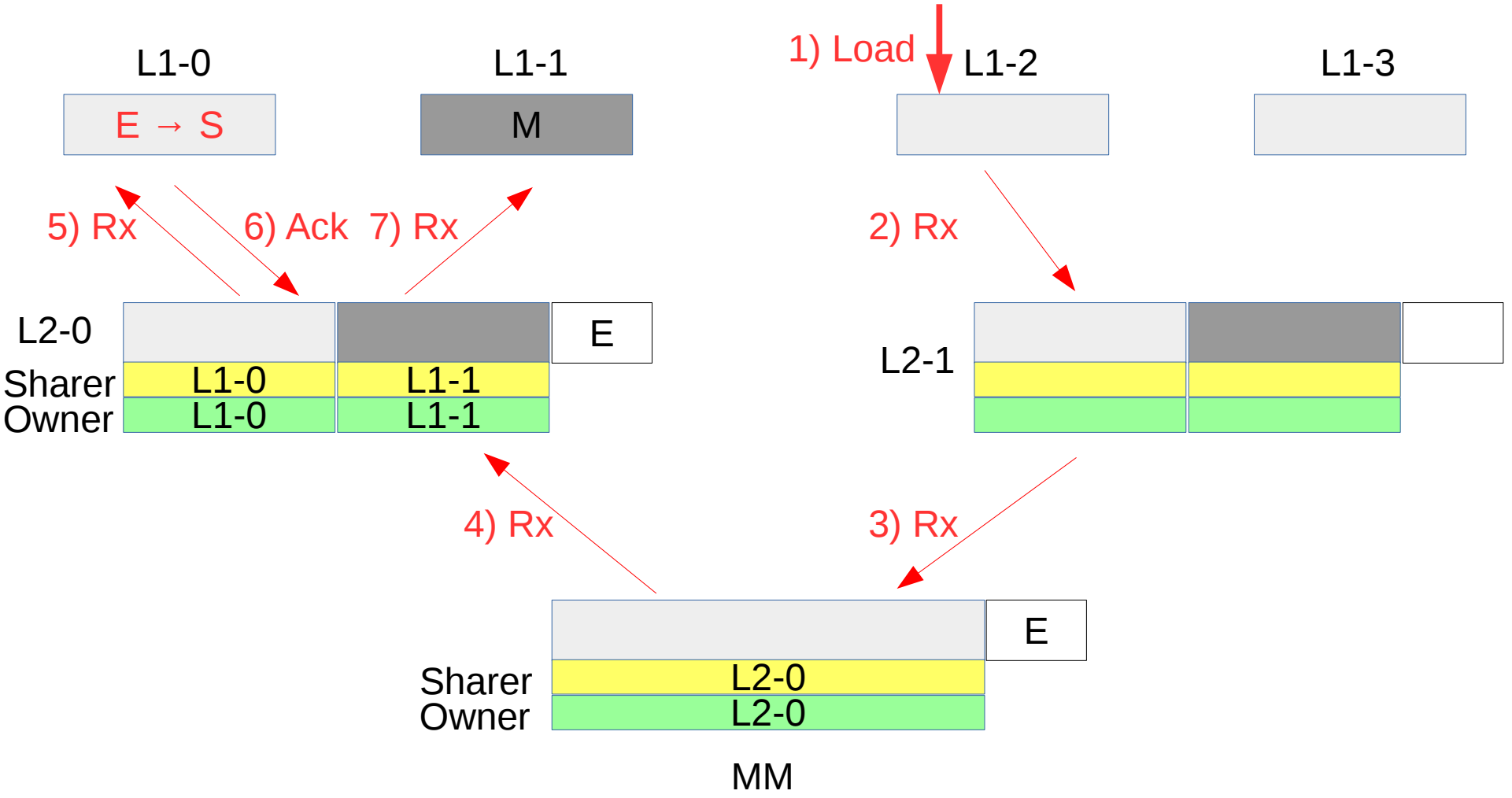
5) Rx from L2-0 to L1-0 (starting with sub-block 0 based on owner filed)
(The L2-0 or L1-0 states can not be I/S/N – only the owner states M/E/O)



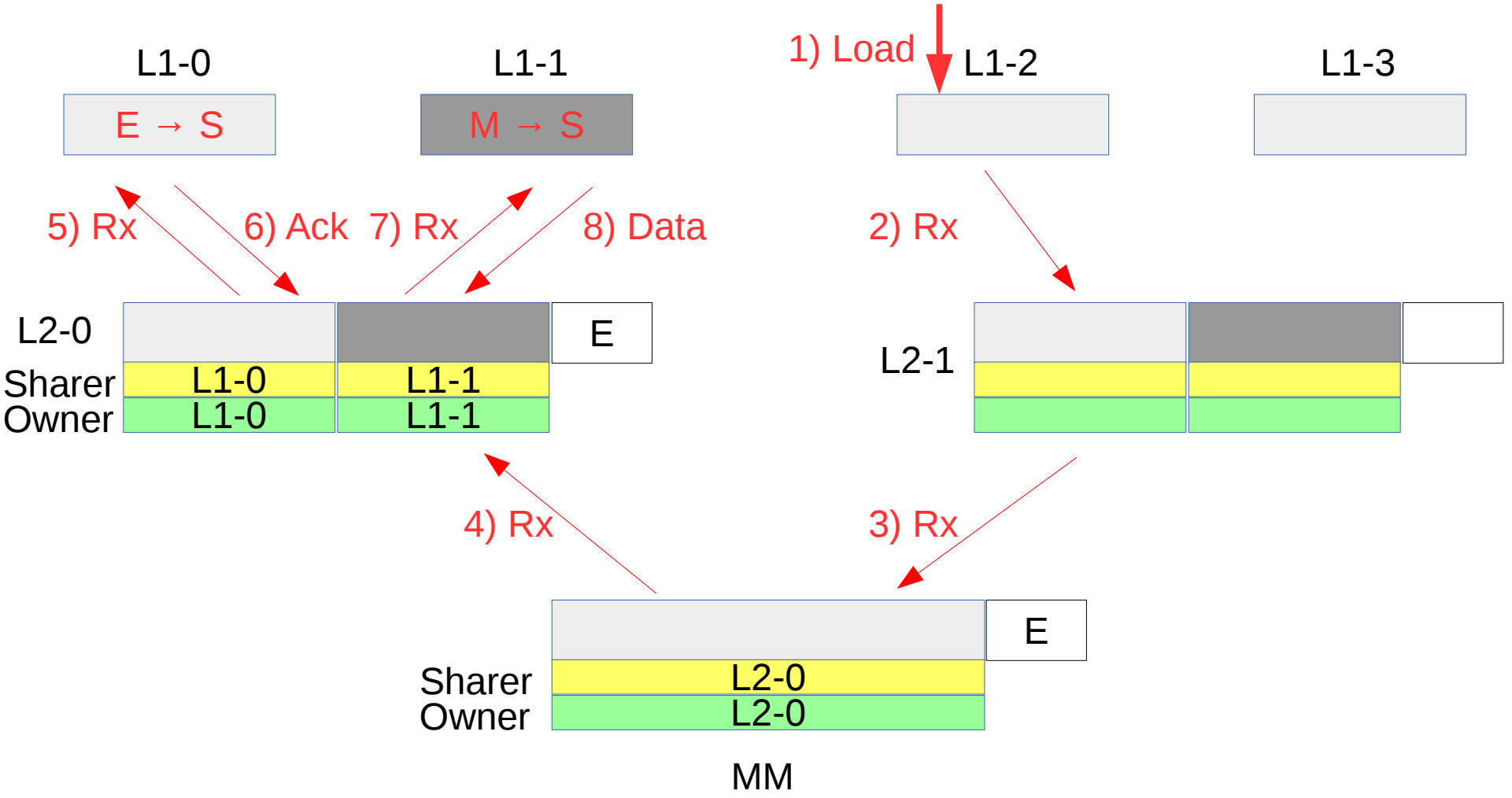
6) Respond from L1-0 to L2-0
(The state is *not* M/O so just an Ack is sent, and stage changes to S)



7) Rx from L2-0 to L1-1 (based on owner field of subblock-1)

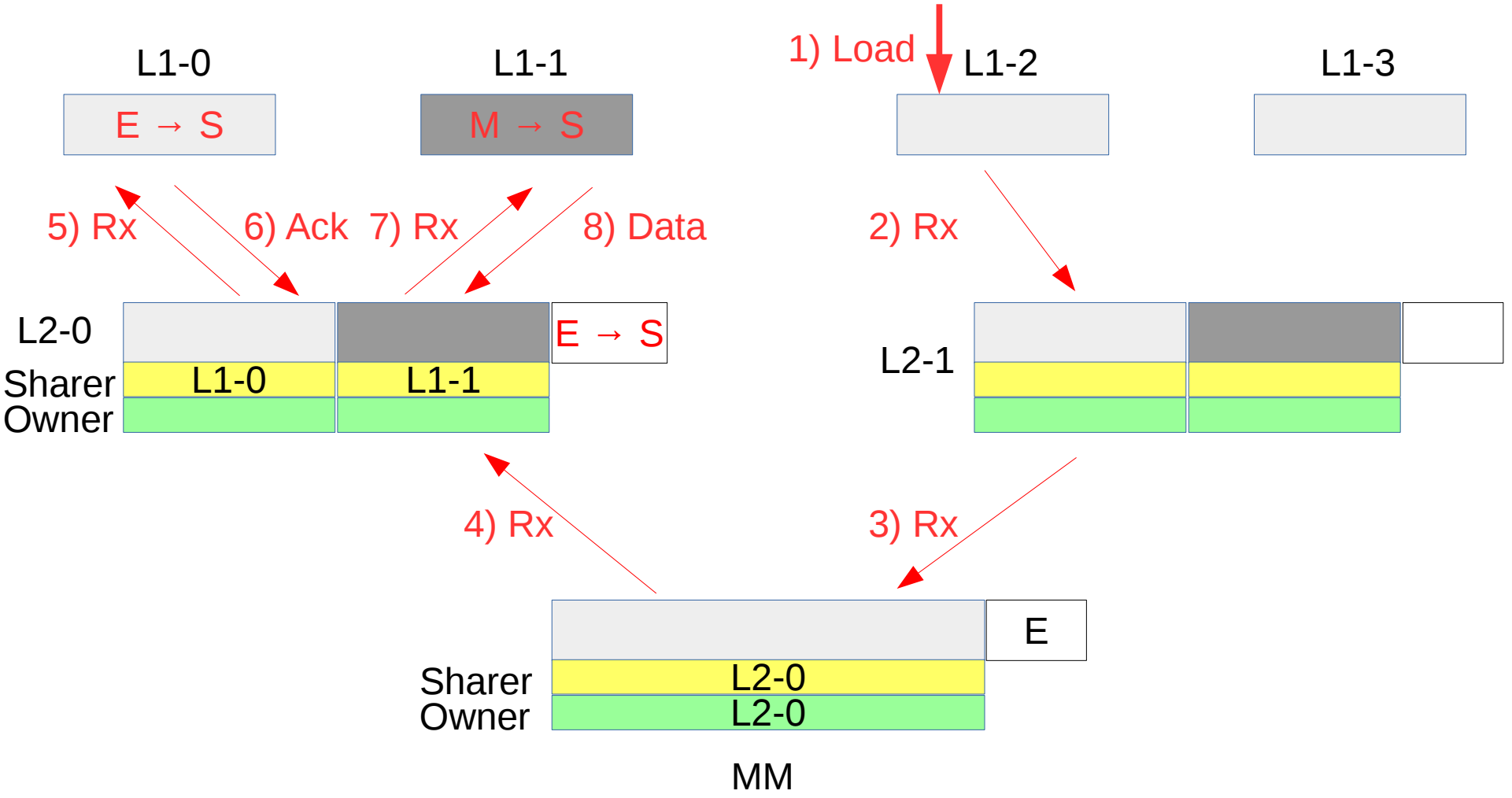


8) Data from L1-1 to L2-0
(The state is M/O so an Ack+Data is sent, and stage changes to S. If we had peer-transfer, the data was sent to peer and state changes to O – but don't know why?)

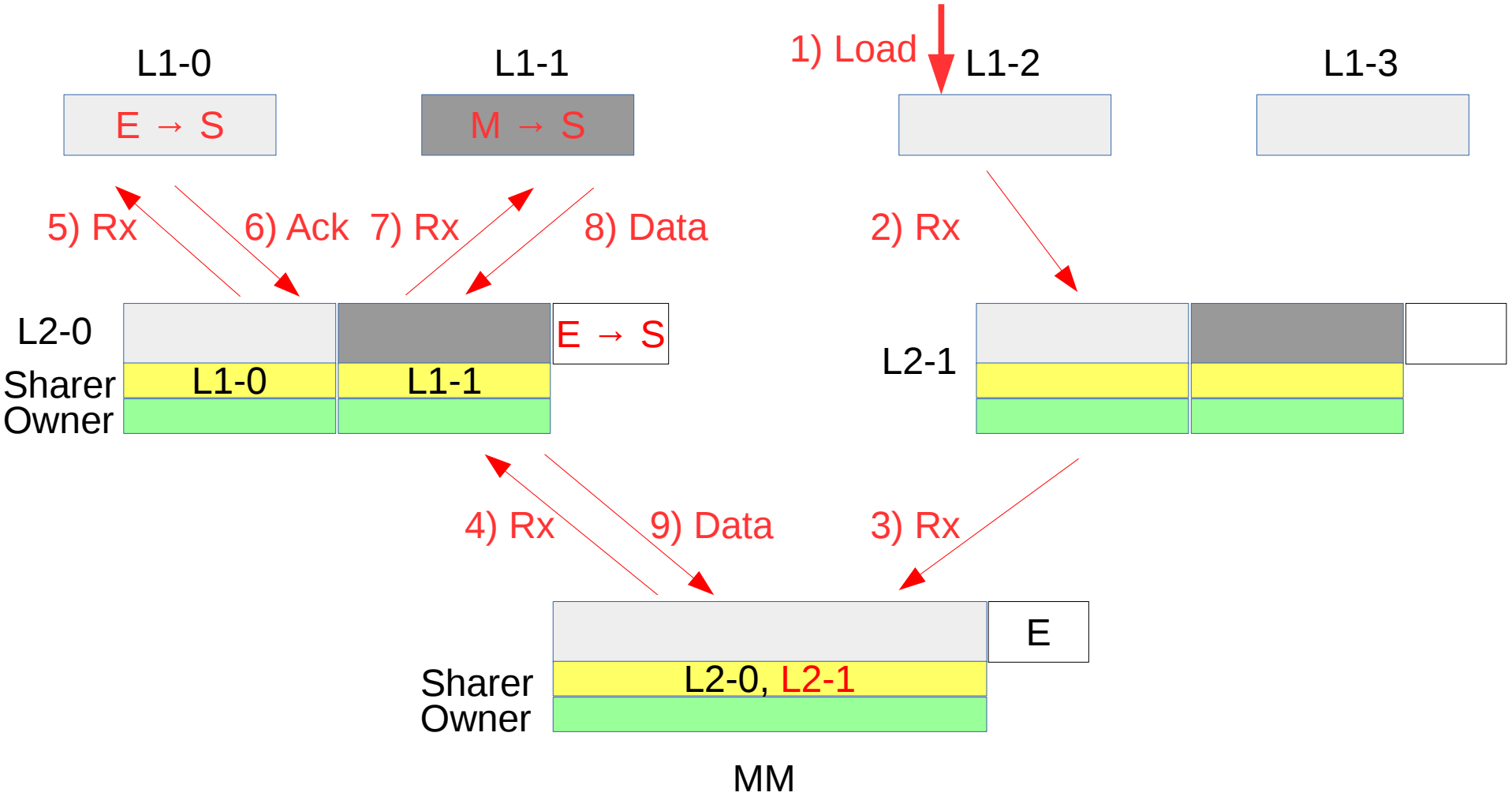


Setting the L2 states and fields

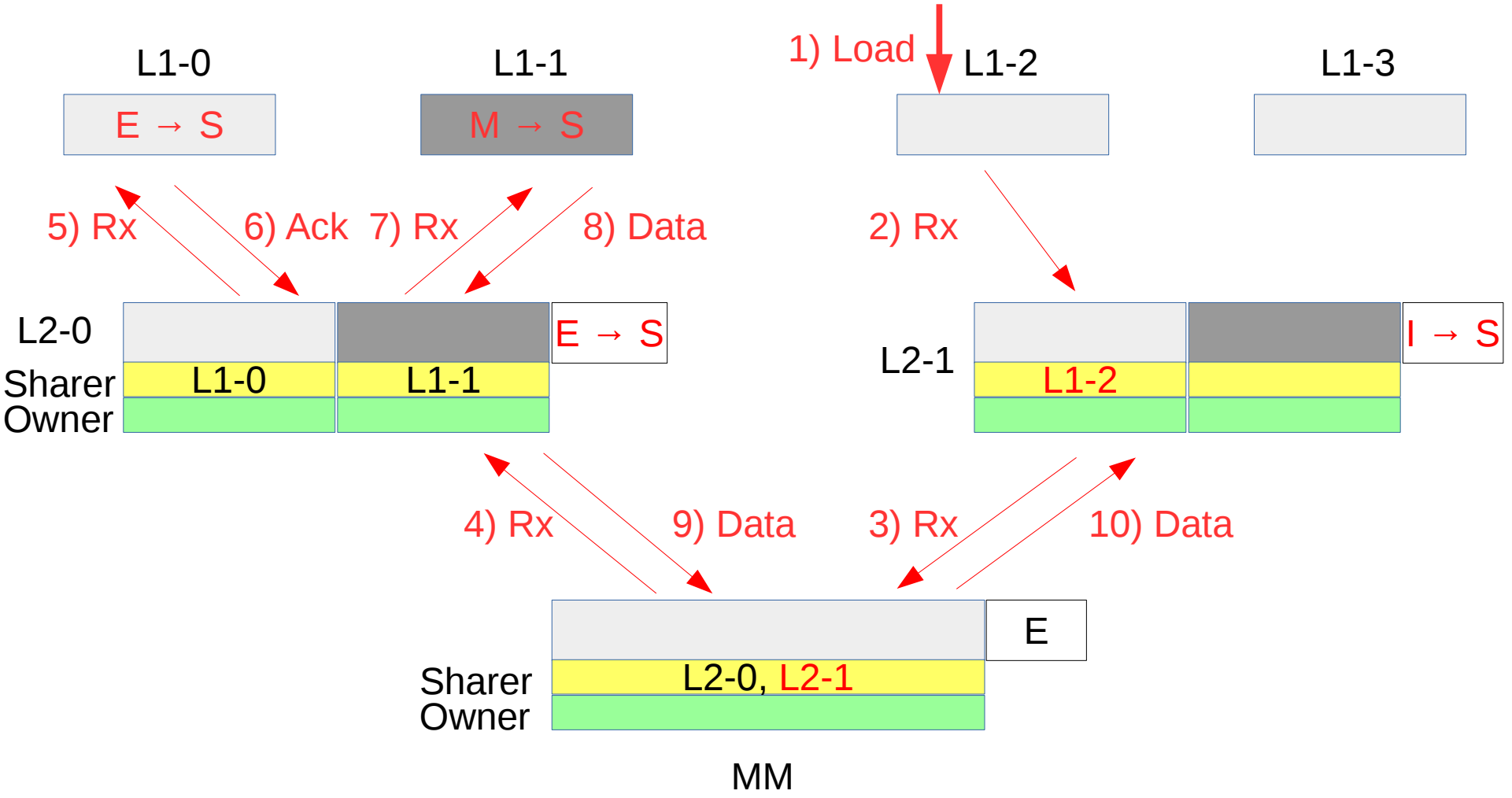
It receives two responds. The Ack (from L1-0) turns the L2-0 from E → S. The Data (from L1-1) again set the state to S but responds with a Data packet to a lower level. (owners are set to none)



9) Data from L2-0 to MM (updown finish)
The peer (requester, or L2-1) is set as a sharer, owner is set to none, and respond to L2-1 is set to Data.



10) Data from MM to L2-1 (updown miss)
The state is set to S (since shared was true), the requester (L1-2) is set as a sharer of the subblock, and response is set to Data.
Ret.shared = true



11) Data from L2-1 to L1-2 (updown miss)
The state is set to S, the requester is NULL (set in Load).
Ret.Shared was true so the state goes to S