Lab Report

Yang Chen: 1005747649

Jiping Li: 1005983269

Section 6

1. **Why are transient states necessary?**

   A transient state is necessary since there's delay between the request sent to directory. The BR/BW signal might cause out of order issue, there we need some kind of "lock" mechanism to prevent this situation. It is necessary for implementing a coherence protocol because they allow the coherence controllers to handle events that cannot be completed atomically.

2. **Why does a coherence protocol use stalls?**

   Keep all Load and Store in-order, handle situations where a cache controller cannot immediately respond to a request or transition to a desired state.

3. **What is deadlock and how can we avoid it?**

   Deadlock is when Event A depends on Event B and Event B depends on Event A, there Deadlock occurs. To solve this use Virtual Networks, use incoming/outgoing queues and buffer.

4. **What is the functionality of Put-Ack (i.e., WB-Ack) messages? They are used as a response to**

   which message types?

   It is used as Writeback Acknowledgement, as response to PUTM and PUTS

5. **What determines who is the sender of a data reply to the requesting core? Which are the possible options?**

   Determines by the Requestor's state; Request Messages: type of request, Multicast destination mask, and current status of Directory.

6. **What is the difference between a Put-Ack and an Inv-Ack message?**

A Put-Ack message is sent by the directory controller to the cache controller after the cache controller has sent a Replacement message to evict a block to the directory. An Inv-Ack message is sent by the cache controller to the directory controller after the cache controller has received an Inv message to invalidate a block.

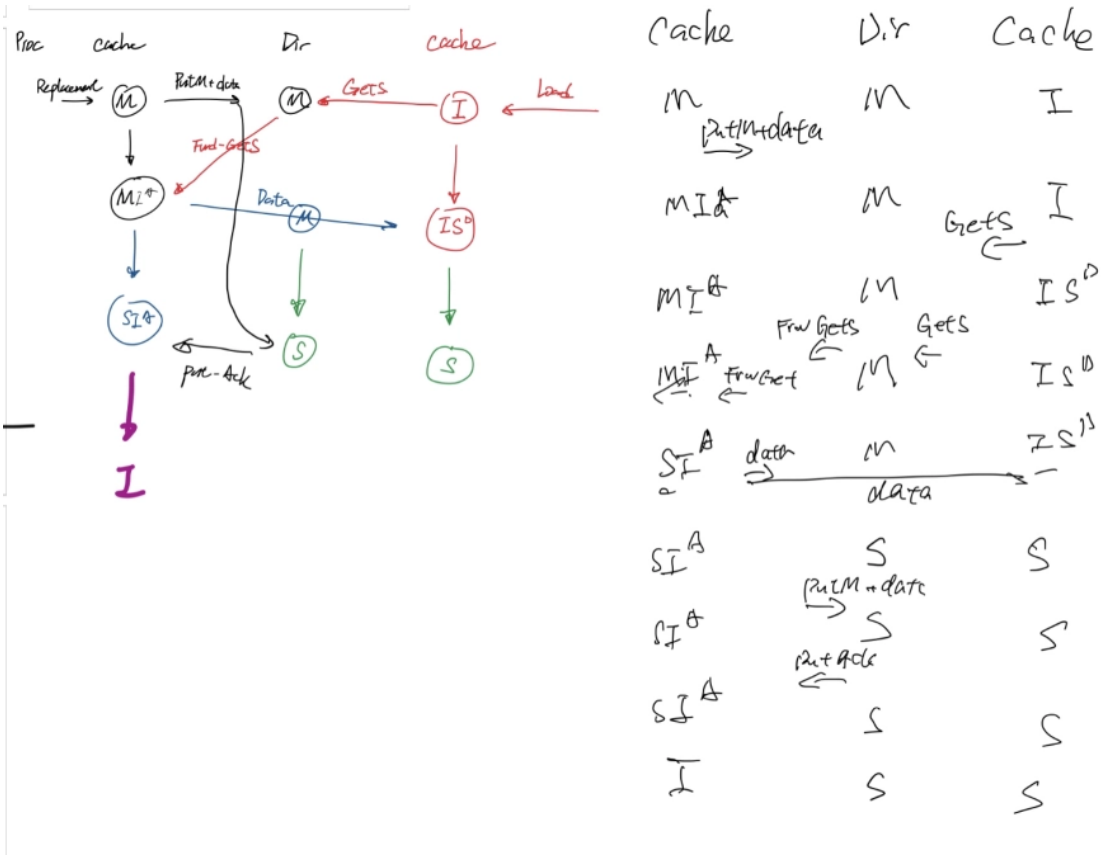
Section 5

1. **How does the FSM know it has received the last Inv Ack reply for a TBE entry?**

   FSM knows based on if the final Ack counter equals zero or not, For the case of S→M or I→M, in transient sate, we will set an ack counter we receive from Directory if ack>0, then it represents there ack # of sharers, which need to wait for thoses shares to invalidate by doing S→I. Thus, when those sharers completed invalidation, they will set Inv-Ack, each Inv ack from a invalidated core will decrement ack by one. Eventually ack will be zero. Telling our main core to leave transiant state into modified state.

2. **How is it possible to receive a PUTM request from a NonOwner core? Please provide a set of events that will lead to this scenario.**

   From the output of PutM to the input of Put-Ack at cache controller side (cc1), that specific block state remains effectively M in directory, and the cache controller respond to Fwd-GetS and Fwd-GetM (send from cc2). These two input will change cc1 state into SIA or IIA, and forward the data to cc2. In this case cc2 will successfully transient to S, or M. This functionality maintains property of 1 writer OR multiple reader.

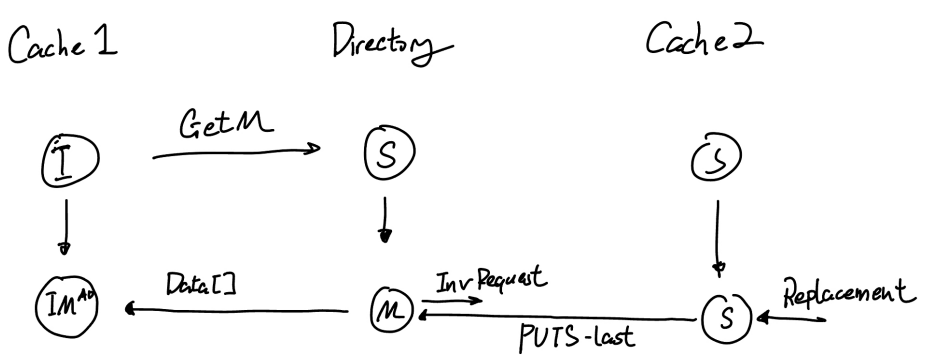


3. **Why do we need to differentiate between a PUTS and a PUTS-Last request?**

   We need to differentiate PUTS and PUTS-last, since directory is managing the Sharers for a certain block. PUTS will removes the Sharers from the list and once it removes the last Sharer from the list, the Directory should change the state of that specific block to Invalid.

4. **How is it possible to receive a PUTS-Last request for a block in modified state in the directory? Please provide a set of events that will make this possible.**

   When initials state of Directory is S, Cache 1 acquires for Store to this Block, GetM sent to Directory which it immediate change to M state. Meanwhile proc2 created a Replacement for cache 2 and created a PUTS-Last request sent to Directory, before the Invalid Request arrives at Cache 2.



5. **Why is it not possible to get an Invalidation request for a cache block in a Modified state? Please explain.**

   The SWMW methodology states that, there are are 3 scenarios, for all caches, all invalid, 1 modified + remains invalid, or multiples shares + remains invalid.

   When there is an I→M, Directory determines to use Fwd-GetM or Inv based on directory global state for that block. From Q2, we prove Dir and Cache state are always “aligned”, meaning if there global state is M, then there is still a state as modified exist or under transition, which has no synchronization issue. Thus, we can say it is not possible to get an Invalidation request for a cache block in a Modified state.

6. **Why is it not possible for the cache controller to get a Fwd-GetS request for a block in SI A state? Please explain.**

   From a higher prespective understanding, in considering synchronization issues as question 2, we need to ensure that S and M can not exist at the same time, for both cache and directory.

   In the detailed understanding, the Fwd-GetS can only received by Owner, while getting into SI A have 2 method, first is S→SI A, this no matter what the owner is not set originally. Second is from MI A→SI A, but in order to get into MI A, it will goes to directory Action[M, PutM + dataFromOwner] this case it will clear Owner and thus, both case the directory doesn’t contains owner. Therefore the SI A core controller will never receive Fwd-GetS.

7. **Was your verification testing exhaustive? How can you ensure that the random tester has exercised all possible transitions (i.e., all {State, Event} pairs)?**

   We employed test scripts to evaluate our protocol across various configurations, ensuring that a range of architectural setups were included. However, it is important to acknowledge that this may not encompass every possible transition. Therefore, we cannot assert that our verification is exhaustive or claim that our program is completely free of bugs.

Modifications

8.2 Add IS_mD transient state	I + GETS ⇒ IS_mD, it's waiting for data from memory, during IS_mD stall other requests.
8.2 Add IM_mD transient state	I + GETM ⇒ IM_mD, it's waiting for data from memory, during IM_mD stall other requests.
8.2 S_D transient state	split into MS_cD and MS_mA two transitent state, details below
8.2 Add MS_cD transient state	M + GetS ⇒ MS_cD, M→MS_cD sent Fwd signal to requester and in MS_cD states waits for Data from cache
8.2 Add MS_mA transient state	once Data arrived from cache MS_cD + data ⇒ MS_mA and here sent WB signal to memory, where waits for Mem_WB signal arrive at Directory. Once arrived MS_mA → S
8.2 Add MI_mA transient state	M+PutM ⇒ MI_mA, here sent WB signal to memory, where waits for Mem_WB signal arrive at Directory. Once arrived MI_mA→ I

Work Completed

work are evenly spilt, Code and Report are written together, Yang take are of cache side debug and Jiping take create of Directory side debug.