Directory based Cache Coherence

Topic-4
Chapter-8

Protocol Optimisations

Performance

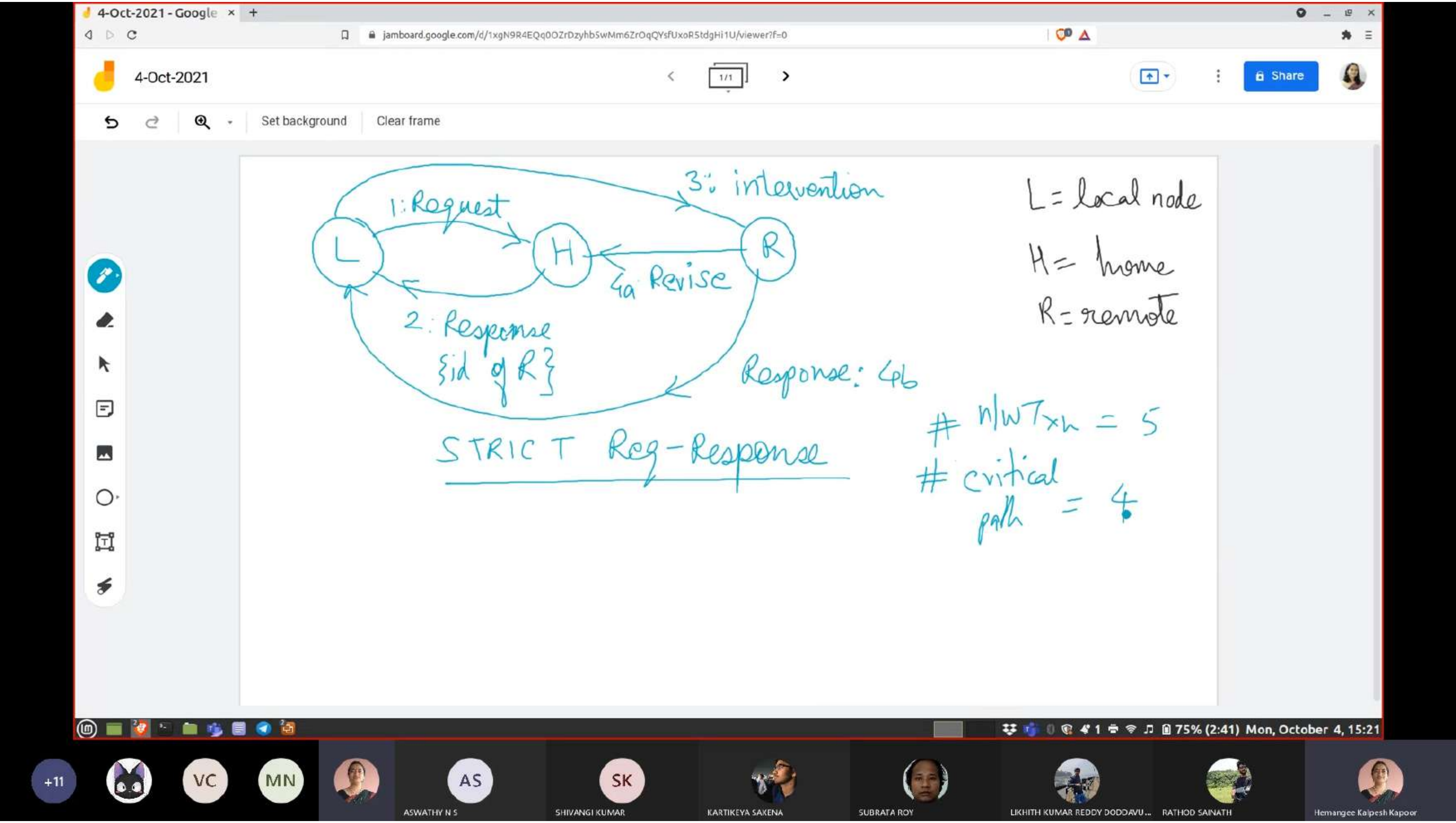
- Performance
  - Network transactions on which cache coherence protocols are built differ from message passing ones
  - (i) they are automatically generated by the communication assist or controllers
  - (ii) individually small in size, only req, response, ack or data
- Each network transaction incurs overhead at requesting processor, communication assist at end-points, network delay
- Processor involved only at requestor. Processor at dirty node or home node is not involved: here CA does the job
- Performance can be improved by
  - (i) protocol optimisation
  - (ii) high-level machine organisation
  - (iii) hardware specialisation to improve communication parameters

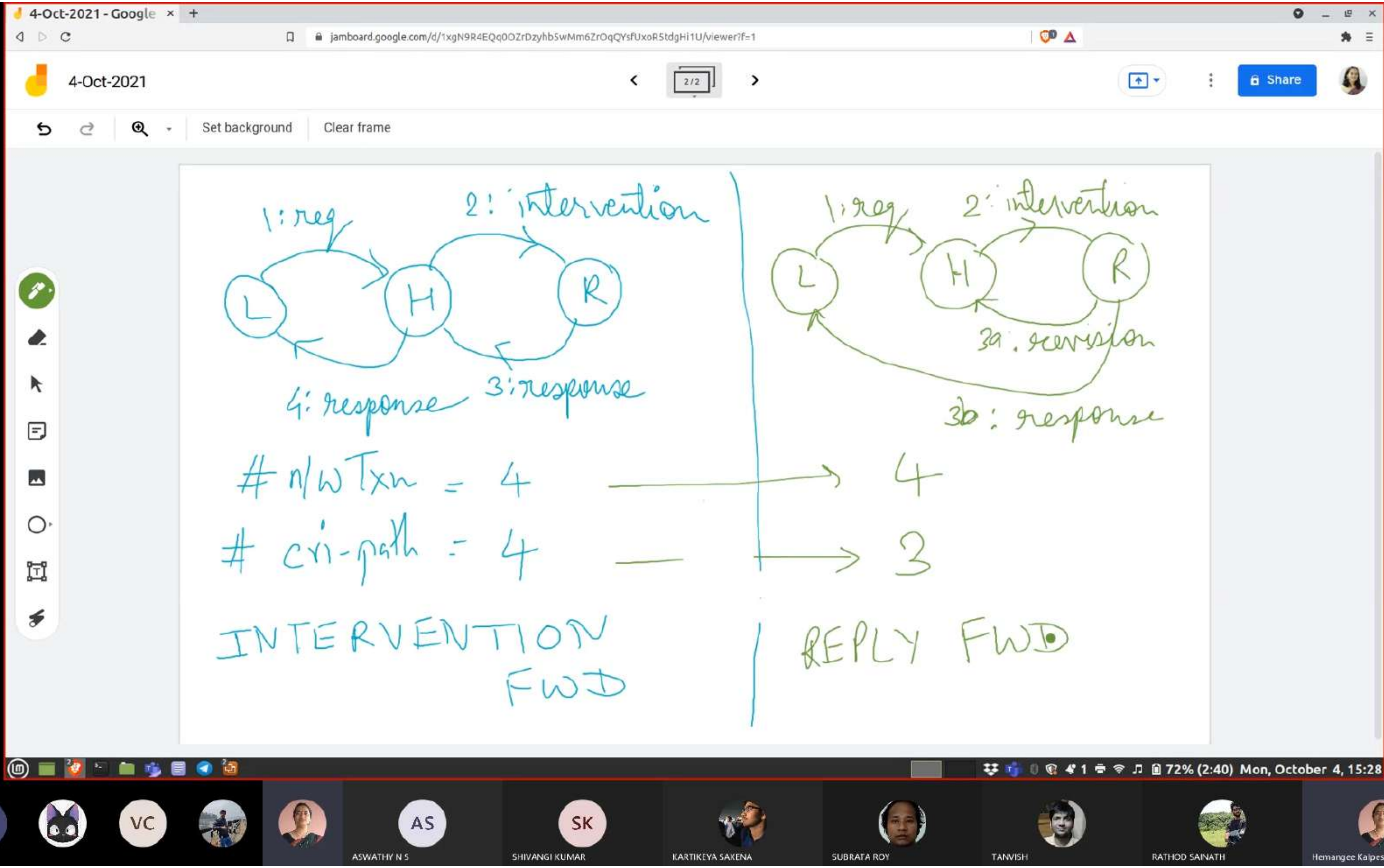
Performance

- Performance can be improved by
  - (i) protocol optimisation
    - We will see 3 variants ...
  - (ii) high-level machine organisation
    - mostly 2-level hierarchy with multiprocessor nodes
    - Each node can be snooping or directory
    - Snooping nodes have longer latency but still used
  - (iii) hardware specialisation to improve communication parameters
    - Keep assist occupancy small, tighter integration with memory unit, allow assist to start new transaction while data of older transation is being fetched, make the assist more specialised
    - Use SRAM directory to reduce look-up time
    - a single bit can be kept with memory to keep track of whether the block is clean or not (so that communication assist is not invoked on a read miss)
    - If assist occupancy is high then pipeline its design or overlap its actions

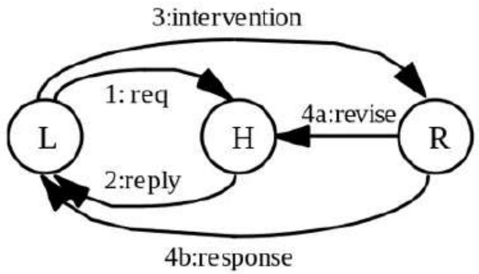
Protocol optimisations

- Protocol optimisation goals
  - Reduce #network transactions
    - Reduces bandwidth demand on the network and the communication assist
  - Reduce #transactions on critical path of processor
    - Can be done by overlapping the transactions needed for a memory operation as much as possible
- Manner in which directory information stored, determines the number of network transactions in the critical path of processor
    - Memory based: can be overlapped
    - Cache based: cannot be overlapped
    - In both cases options of improvement are possible
- Methods of communication
    - (i) strict Request-Response
    - (ii) intervention forwarding
    - (iii) reply forwarding

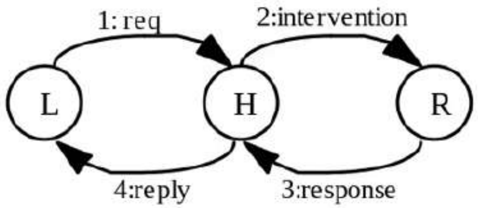

4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhbSwMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=0
4-Oct-2021
1/1
Share
Set background
Clear frame
1: Request
3: intervention
L
H
R
4a Revise
2: Response {id of R}
Response: 4b
STRICT Req-Response
L = local node
H = home
R = remote
# n/w Txn = 5
# critical path = 4
75% (2:41) Mon, October 4, 15:21
+11
VC
MN
AS
SK
ASWATHY N S
SHIVANGI KUMAR
KARTIKEYA SAXENA
SUBRATA ROY
LIKHITH KUMAR REDDY DODDAVU...
RATHOD SAINATH
Hemangee Kalpesh Kapoor


4-Oct-2021
Set background
Clear frame
1: req
2: intervention
L
H
R
4: response
3: response
n/w Txn = 4
cri-path = 4
INTERVENTION FWD
1: req
2: intervention
3a: revision
3b: response
4
3
REPLY FWD
Mon, October 4, 15:28

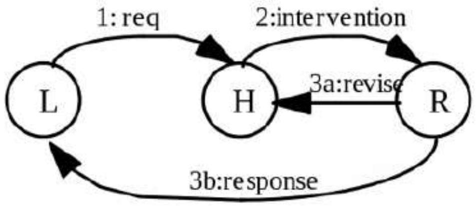
Protocol enhancements for latency



(a) Strict request-reply



(a) Intervention forwarding



(a) Reply forwarding

L = local requesting node

H = home node

R = remote, dirty node

Strict Req-Response

- 1: request
- 2: carriers owner indentity
- 3: intervention
- 4a: sends data
- 4b: sends data to update "home" and change directory-state

- => we have 4 network transactions in critical path of Read operation and total 5 network transactions

- Reduce this using (2) intervention forwarding ..

+19 VC ASWATHY N S AS SHIVANGI KUMAR SK KARTIKEYA SAXENA SUBRATA ROY TANVISH RATHOD SAINATH Hemangee Kalpesh Kapoor

Intervention forwarding

- 1: request, 2: intervention, 3: response, 4: response

- Home does not respond to requestor
- But forwards request as intervention to owner asking for data
- Owner sends data to home. Updates state. Home sends data to requestor

- *"Intervention is like a request but is issued in reaction to a request and is directed at a cache rather than memory" [similar to inv but seeks data]*
- Total network transactions reduced from 5 to 4
  - Bandwidth requirement reduced
  - #transactions in critical path is still 4
- So need more agressive method = Reply forwarding

Intervention forwarding

- 1: request, 2: intervention, 3: response, 4: response

- Home does not respond to requestor
- But forwards request as intervention to owner asking for data
- Owner sends data to home. Updates state. Home sends data to requestor

- *"Intervention is like a request but is issued in reaction to a request and is directed at a cache rather than memory" [similar to inv but seeks data]*
- Total network transactions reduced from 5 to 4
  - Bandwidth requirement reduced
  - #transactions in critical path is still 4
- So need more agressive method = Reply forwarding

+19 VC AS SK

ASWATHY N S SHIVANGI KUMAR KARTIKEYA SAXENA SUBRATA ROY TANVISH RATHOD SAINATH Hemangee Kalpesh Kapoor

Reply forwarding

- 1: request, 2: intervention: 3a: revision, 3b: response

- Home forwards the intervention message to owner but here intervention message contains identity of requestor
- Owner replies directly to requestor
- Owner also sends revision message to home to update memory data and directory state

- But 3a is not in critical path of Read miss
- Total #transactions is 4 and <u>#transactions in critical path is 3</u>
- Called three-message miss

Which to use?

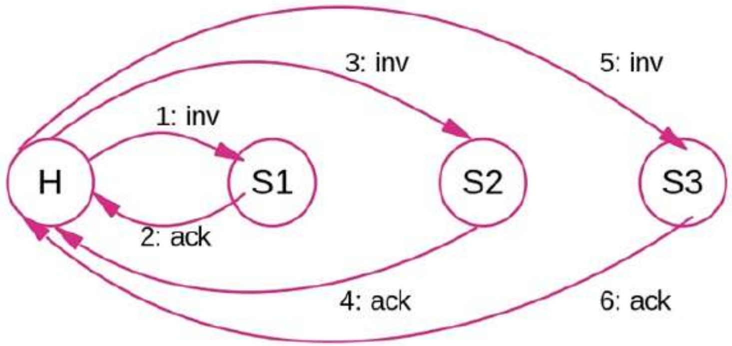
- Type (2) and (3): intervention forwarding and reply forwarding are NOT strict request-response
- Since a request at home generates another request (to owner) which in turn generates reponse. This complicates deadlock avoidance

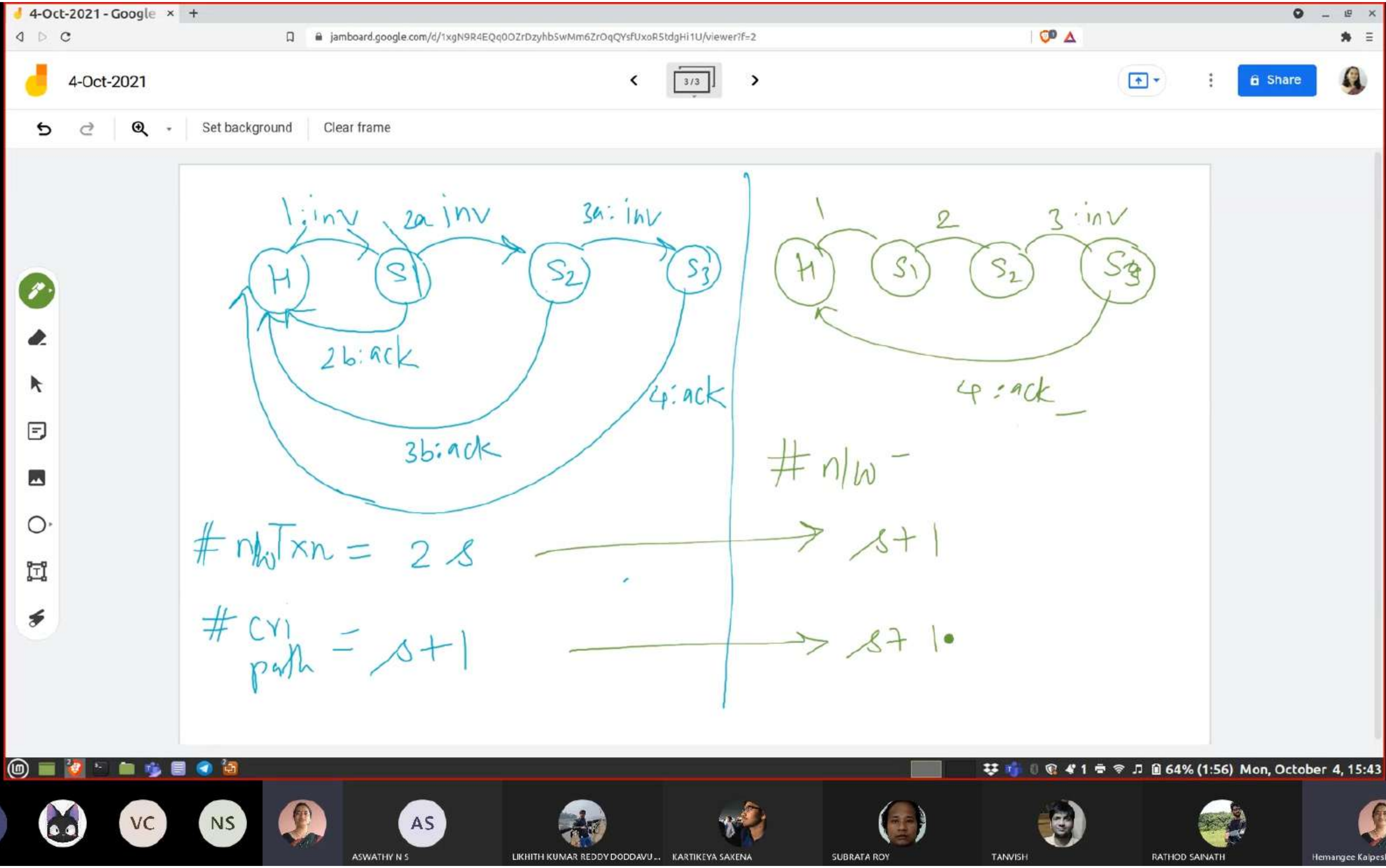
- Intervention forwarding is intermediate in its latency and traffic characteristics
- At the same time has disadvantage that all outstanding interventions are kept track-of at home node
- Home has to keep track of k*P requests as there can be 'k' requests from each node in a 'P' node system
- Requestor has to keep track of only 'k' outstanding requests

- Reply forwarding does not require home to keep track and has better performance
- Therefore systems prefer to use reply forwarding

Reducing latency in flat, cache-based protocol

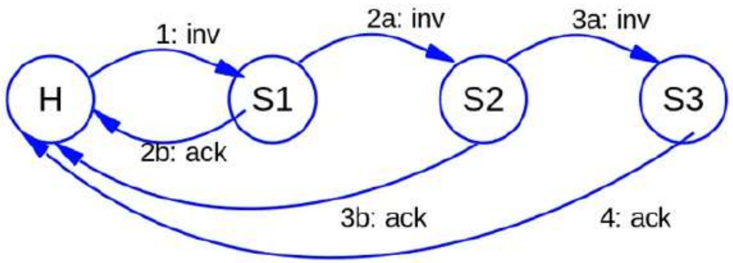
- Scenario: write request. Inv to be sent to i sharers
- Strict request-response
  - In each ‘ack’ identity of next sharer is sent to home
  - Total #transactions is 2s : s = #sharers
  - All 2s are in critical path



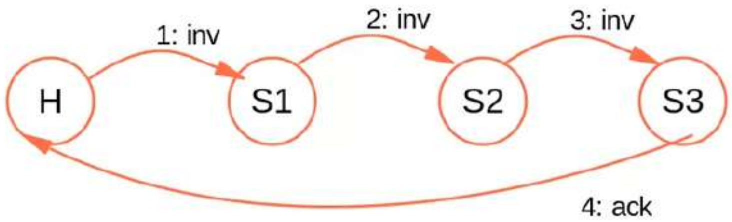

4-Oct-2021
Set background
Clear frame
Share
1: inv
2a inv
3a: inv
H
S1
S2
S3
2b: ack
3b: ack
4: ack
n/w Txn = 2s
cri path = s+1
1
2
3: inv
4: ack
n/w
s+1
s+1
ASWATHY N S
LIKHITH KUMAR REDDY DODDAVU...
KARTIKEYA SAXENA
SUBRATA ROY
TANVISH
RATHOD SAINATH
Hemangee Kalpesh Kapoor
Mon, October 4, 15:43

Intervention fwd and reply fwd

- Intervention fwd
  - Total transactions = 2s
  - In critical path = s+1
  - As each s_i sends inv to next sharer and in parallel sends ack to home



- Reply fwd
  - Only last sharer sends ack to home
  - Total transactions = s+1
  - In critical path = s+1



- These two types are NOT strict-request-response cases

Correctness

- Similar to snoop, here correctness concerns are
- (1) Protocol ensures relevant blocks are invalidated and data retrieved, state transitions happen (we assume this holds, we will not prove it here)
- (2) serialisation and ordering relationships defined by coherence and consistency model must be preserved
- (3) protocol implementation must be free from deadlock, livelock and (ideally) starvation
- Points (2) and (3) complicated by scalable protocols because
  - (i) multiple copies of block but no single agent that sees all relevant transactions and serialises them
  - (ii) with many processors, large number of requests may be directed towards a single node, creating input buffer problems

Correctness

- Scalable systems have high latency which aggravate the problem and makes us use protocol optimisations (As shown earlier)

- Optimisations allow
  - (i) more transactions to be in parallel simultaneously, and
  - (ii) do not preserve a strict request-response nature, thus complicating correctness

Serialisation to a location for coherence

- A given processor must be able to construct a serial order out of all the operations to a given location
  - At least out of all write operations and its own read operation AND
- All processors must see the writes to a given location as having happened in the same order
- For serialisation we need one entity that sees the necessary operations to a given location from all processors and determines the serial order
- (i) easy in BUS based, as bus is the entity
- (ii) in distributed system that does not cache shared data: memory at home can be entity
  - Order in which writes reach home is the order visible to all processors
  - Which write's value a read sees is determined by when the read reaches home memory
- (iii) in distributed system with coherent caching ..

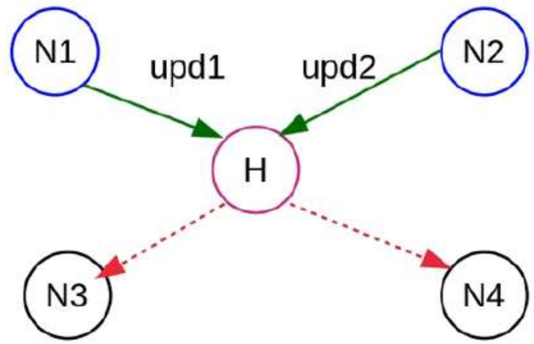
Serialisation to a location for coherence

- (iii) in distributed system with coherent caching
  - Home memory is likely candidate as serialising entity, atleast in flat-directory
  - (a) if home can satisfy all requests itself then it processes them in FIFO order and determines serialisation
    - But with multiple copies, visibility at home does not imply visibility by all processors
  - Easy to generate scenario when different processors see different orders than what are seen by home
    - Examples: (1) update protocol (2) inv protocol write request

Ex: see diff orders: update prot

- (1)

  Update based protocol and network does not preserve point-to-point order of transactions between the end-points



- If two write requests for shared data arrive at home in one order, the updates they generate may arrive at the copies in different orders

H: upd1 ; upd2

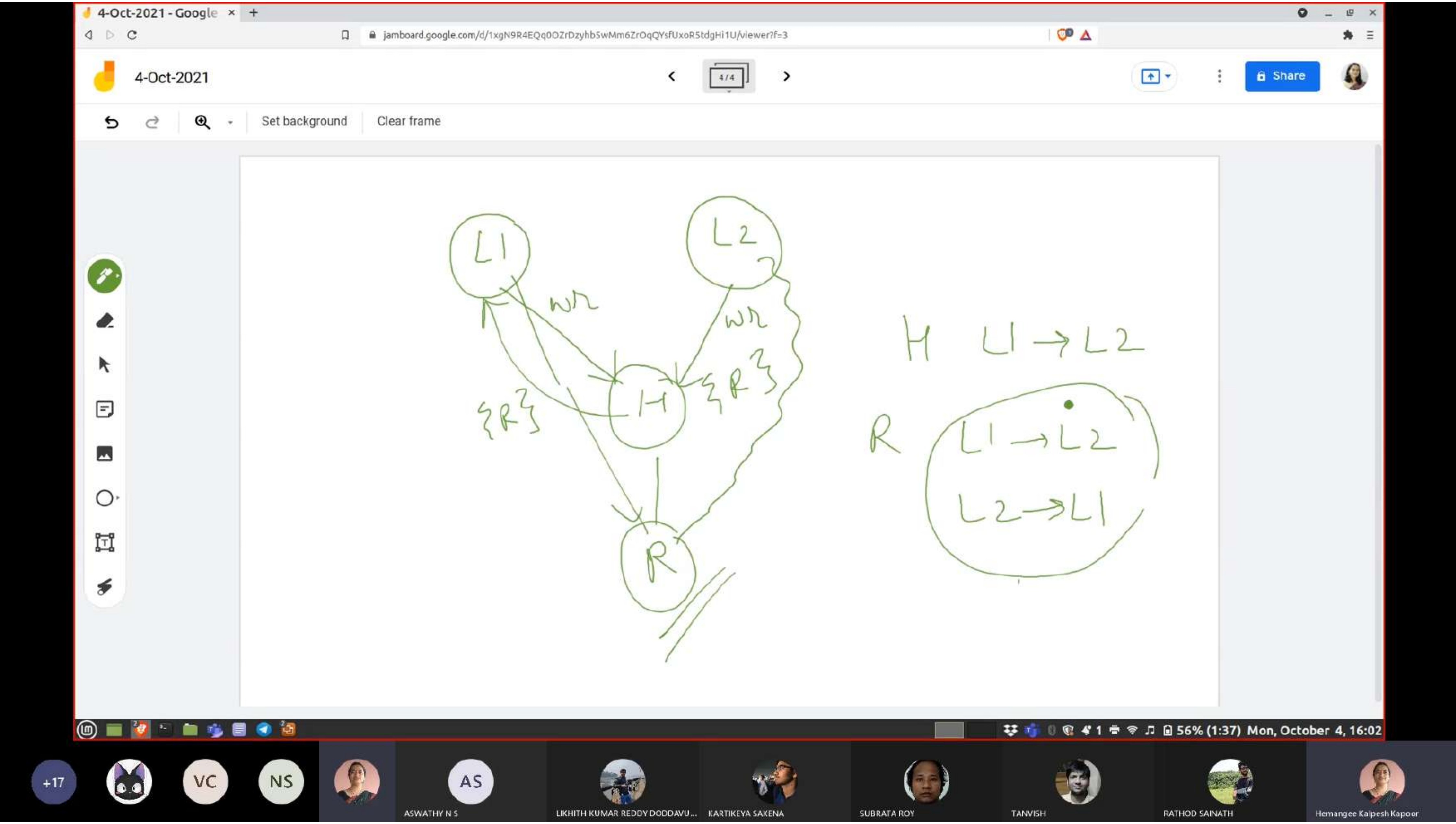
N3: upd1 ; upd2

N4: upd2 ; upd1

Ex: see diff orders: inv protocol

- (2) in inv-based protocol, block in dirty state in a node. Two other nodes issue a Read exclusive request
  - In strict request-response protocol, the home will give identity of owner to both requestors
    - Note that the Home node will not get updated with new owner until the revision message comes from owner. In the mean time a new requestor can come and Home will give the same owner ID to this new requestor.
  - Then the requestor will go to owner to get data
  - Order in which requestor reached home and order in which they reach owner may be different
  - Which entity provides globally consistent serialisation in this case?

+17 VC NS AS
ASWATHY N S LIKHITH KUMAR REDDY DODDAVU... KARTIKEYA SAXENA SUBRATA ROY TANVISH RATHOD SAINATH Hemangee Kalpesh Kapoor


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhbSwMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=3
4-Oct-2021
4 / 4
Share
Set background
Clear frame
L1
L2
wr
wr
{R}
{R}
H
R
H L1 → L2
R
L1 → L2
L2 → L1
56% (1:37) Mon, October 4, 16:02
+17
VC
NS
AS
ASWATHY N S
LIKHITH KUMAR REDDY DODDAVU...
KARTIKEYA SAXENA
SUBRATA ROY
TANVISH
RATHOD SAINATH
Hemangee Kalpesh Kapoor

Ex: see diff orders: inv protocol

- (2) in inv-based protocol, block in dirty state in a node. Two other nodes issue a Read exclusive request
  - In strict request-response protocol, the home will give identity of owner to both requestors
    - Note that the Home node will not get updated with new owner until the revision message comes from owner. In the mean time a new requestor can come and Home will give the same owner ID to this new requestor.
  - Then the requestor will go to owner to get data
  - Order in which requestor reached home and order in which they reach owner may be different
  - Which entity provides globally consistent serialisation in this case?