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

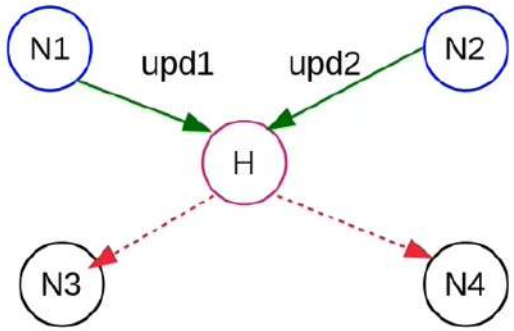
Serialisation to a location for coherence

- (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?

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?

See diff orders ...

- How to manage when multiple operations for the same block are in flight, and require service from different nodes?

- Several solutions can be used to ensure serialisation to a location. Most use additional directory states: Busy or Pending
- A block being in busy state at directory indicates that a previous request is still in progress (not completed). When new request comes, serialisation can be provided as follows: by using one of the following mechanisms:

- (i) Buffer at home
- (ii) Buffer at requestor
- (iii) NACK and Retry
- (iv) Forward to dirty node

(1) Buffer at Home

- New request is buffered at home until previous request in progress has completed
- Even if this request is being forwarded to owner or whether strict request-response used
- [Home must serve requests to other blocks]
- Ensures FIFO order at home, BUT
- (i) reduces concurrency
- (ii) requires home to be notified when write is completed, or i.e. when homes involvement is over
- (iii) increases danger of input buffer overflow at home (due to pending requests)
  - Can send pending to memory (if space => infinite buffering) and avoid deadlock problems
- Used in MIT Alewife processor

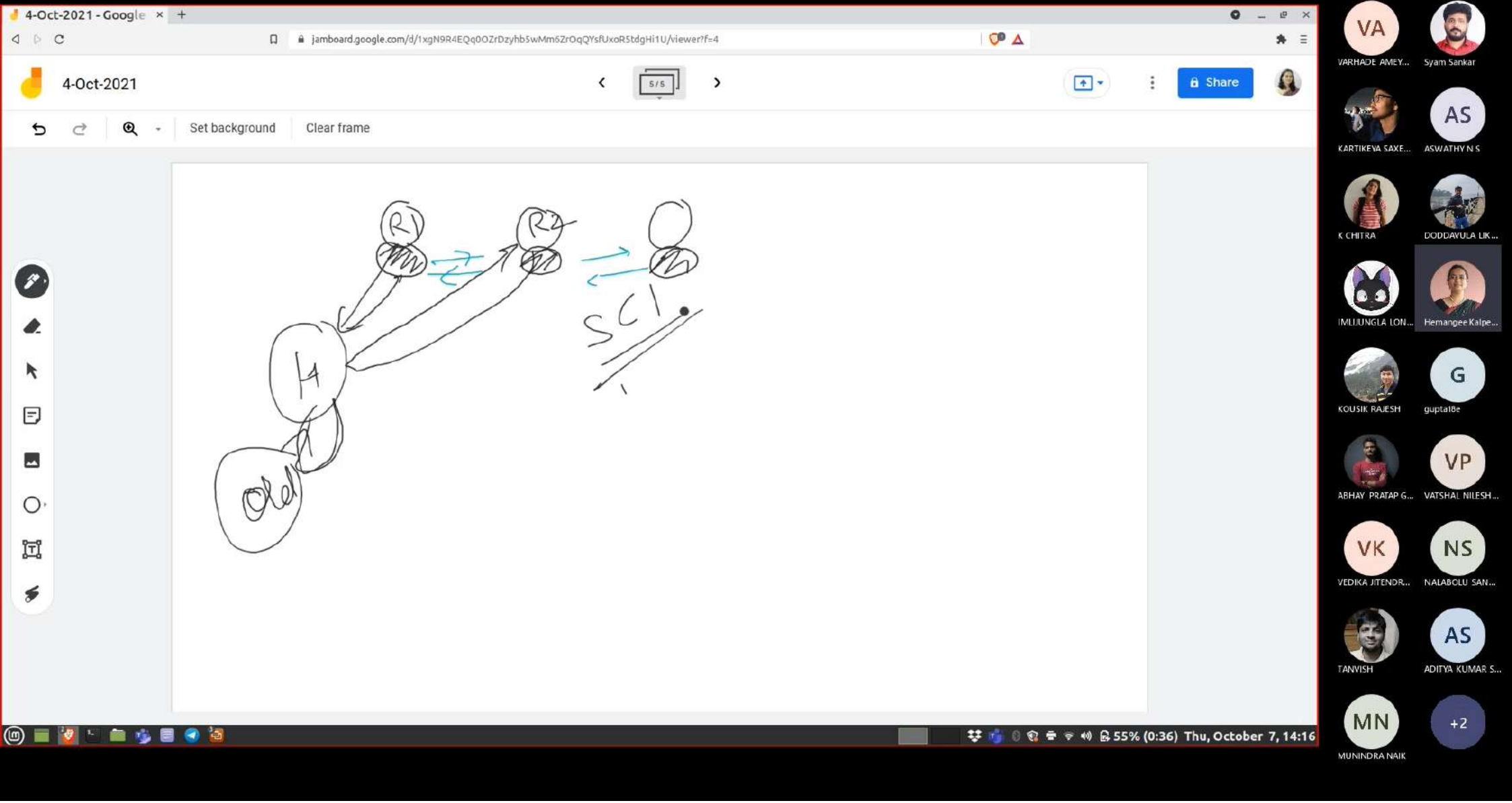
(2) Buffer at Requestor
(3) NACK and Retry

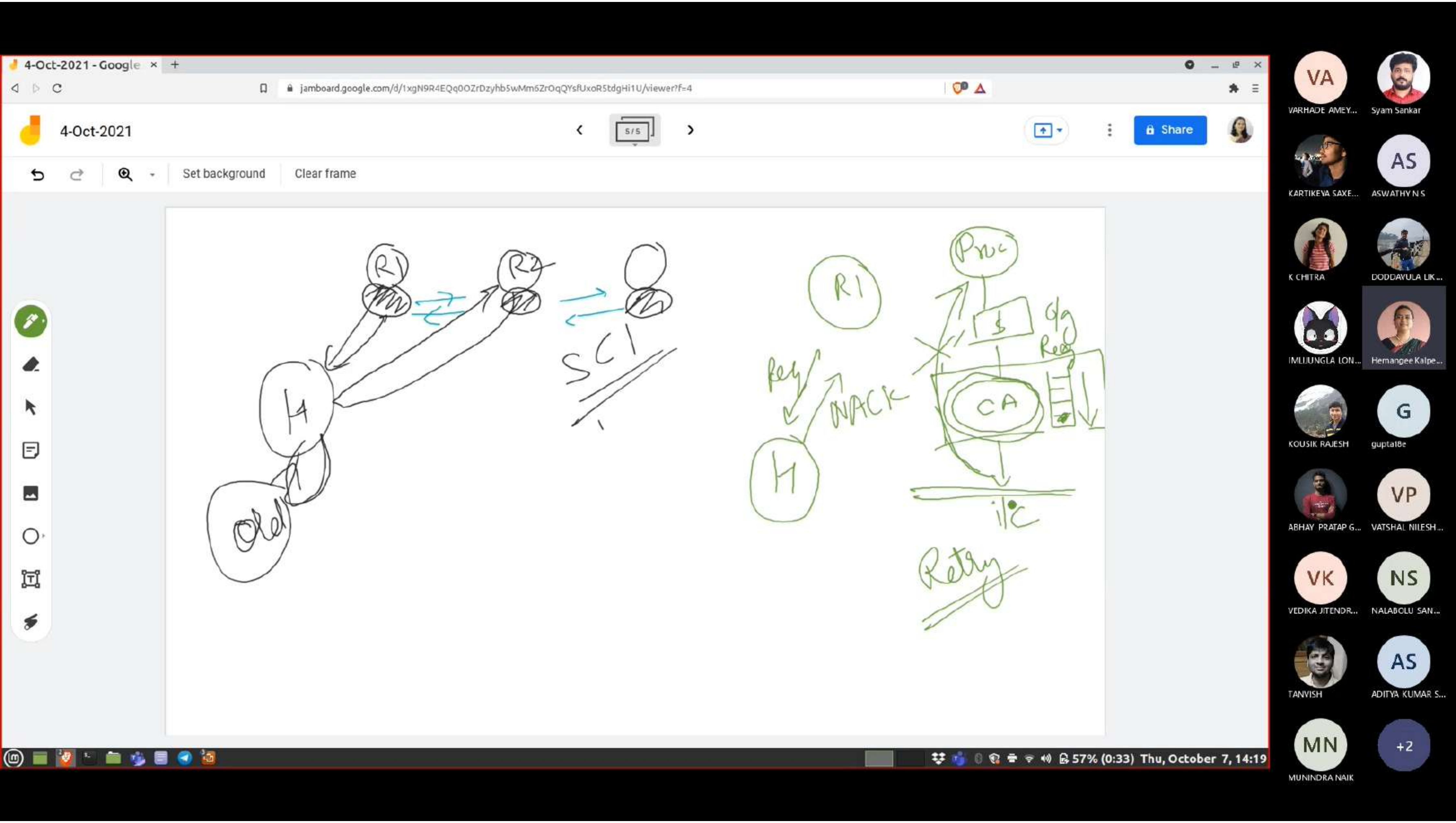
(2)

- Pending requests are buffered at requestor and not at home, by constructing a distributed linked list like in cache-based directory
- Used in SCI protocol standard of IEEE
- Number of pending requests that each node needs to keep track of is small

(3)

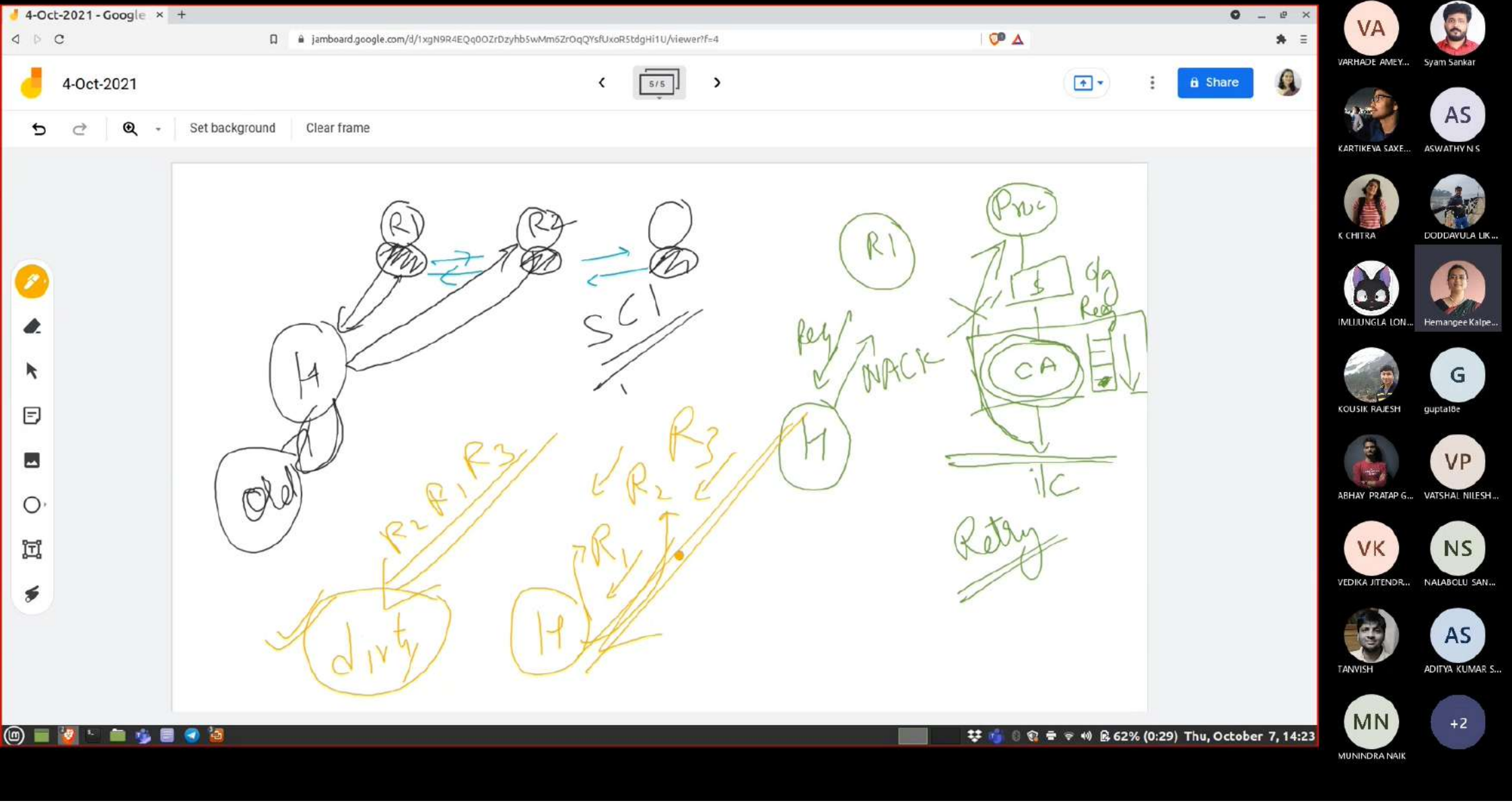
- Home need not buffer new request
- It can send negative-ack when the state is Busy
- The requestors' CA will retry the request later
- The serial order will be determined by when the home accepts this request
- Attempts that are NACKed do not enter the serialisation order
- Used in Origin 2000

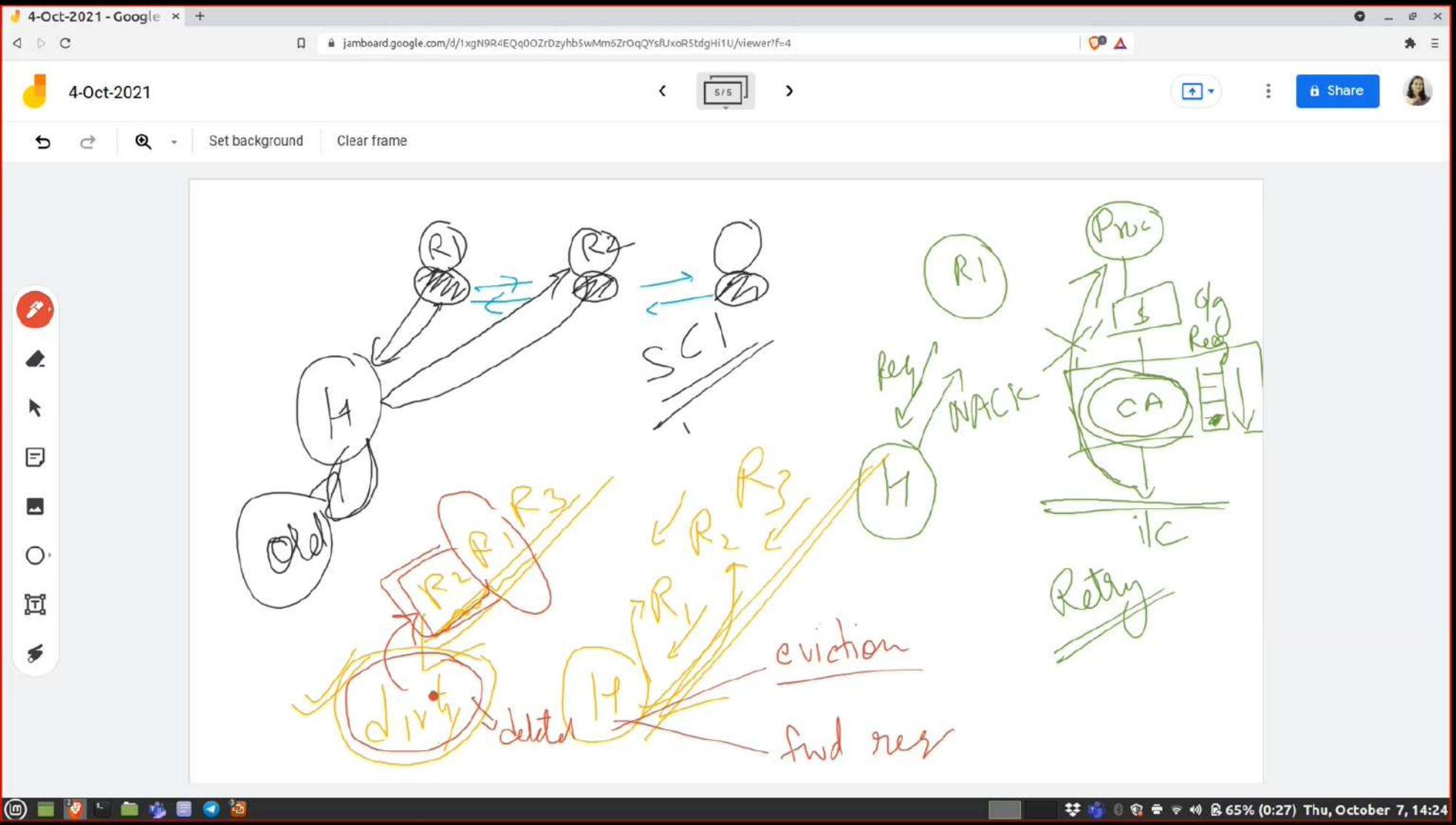

4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=4
4-Oct-2021
5/5
Share
Set background
Clear frame
R1
R2
SC
H
Old
VA
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
AS
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
G
guptal8e
ABHAY PRATAP G...
VP
VATSHAL NILESH...
VK
VEDIKA JITENDR...
NS
NALABOLU SAN...
TANVISH
AS
ADITYA KUMAR S...
MN
MUNINDRA NAIK
+2
55% (0:36) Thu, October 7, 14:16


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=4
4-Oct-2021
5 / 5
Share
Set background
Clear frame
R1
R2
H
Old
SCI
R1
H
Req
NACK
Proc
$
Req
CA
Retry
VA
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
AS
ASWATHY N S
K CHITRA
DODDAVULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
G
gupta8e
ABHAY PRATAP G...
VP
VATSHAL NILESH...
VK
VEDIKA JITENDR...
NS
NALABOLU SAN...
TANVISH
AS
ADITYA KUMAR S...
MN
MUNINDRA NAIK
+2
57% (0:33) Thu, October 7, 14:19

(4) Forward to the dirty node

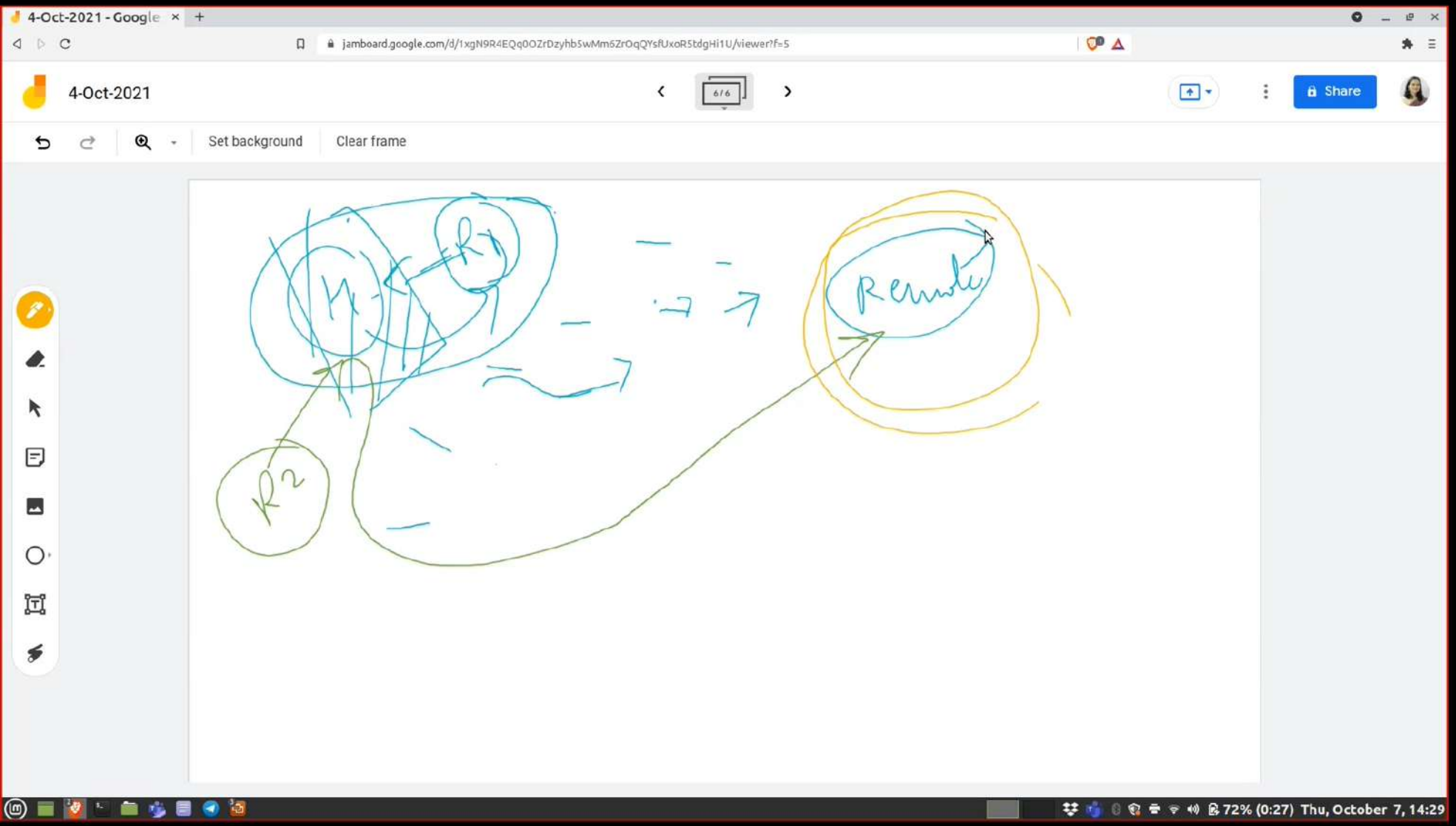
- When a new request comes and directory is busy, forward the request to the dirty node
- Subsequent requests are not buffered at home or NACKed. But are buffered at dirty node
- Dirty node decides the order of serialisation
- Order of serialisation is determined by home for clean blocks and by the order in which the requests reach the dirty-node for dirty blocks
- Meanwhile, if the dirty node deletes the dirty block (write-back or due to some previous forwarded request) then the dirty node NACKs the new requests
- These will be later retried
- Later they will be serialised at home or dirty node when the retry is successful
- Used in Stanford DASH protocol


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=4
4-Oct-2021
5/5
Share
Set background
Clear frame
R1
R2
SCI
H
Old
R2 R1 R3
dirty
R3
R2
R1
H
R1
Prvc
S
Reg
Req
NACK
CA
I/C
Retry
VA
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
AS
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
G
guptal8e
ABHAY PRATAP G...
VP
VATSHAL NILESH...
VK
VEDIKA JITENDR...
NS
NALABOLU SAN...
TANVISH
AS
ADITYA KUMAR S...
MN
MUNINDRA NAIK
+2
62% (0:29) Thu, October 7, 14:23


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=4
4-Oct-2021
5/5
Share
Set background
Clear frame
R1
R2
SCI
H
Old
R3
R2
R1
div
delete
H
R3
R2
R1
H
eviction
fwd req
R1
Proc
S
Reg
Req
NACK
CA
I/C
Retry
VA
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
AS
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
G
guptal8e
ABHAY PRATAP G...
VP
VATSHAL NILESH...
VK
VEDIKA JITENDR...
NS
NALABOLU SAN...
TANVISH
AS
ADITYA KUMAR S...
MN
MUNINDRA NAIK
+2
65% (0:27) Thu, October 7, 14:24

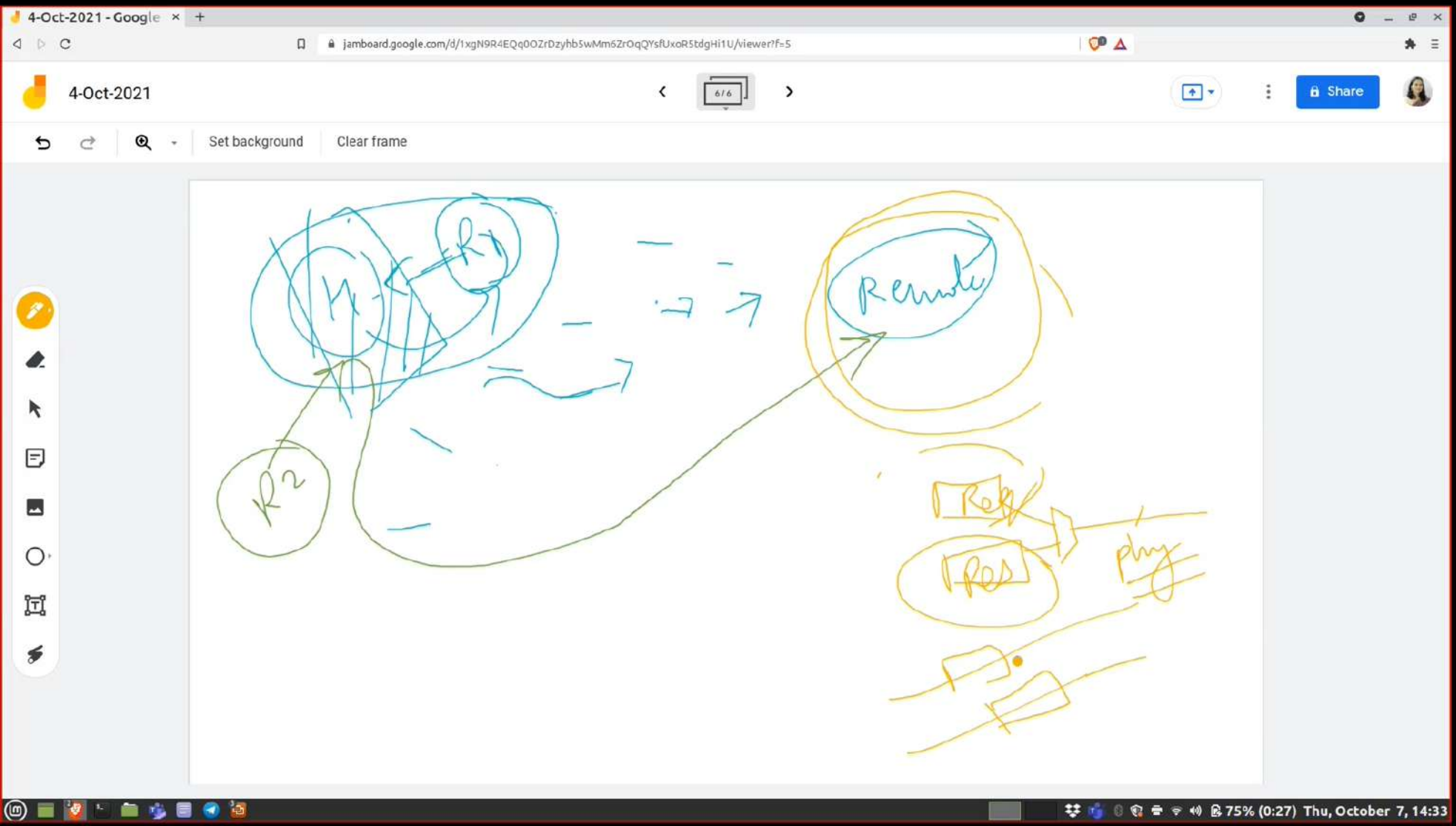
Serialising entity is not sufficient

- With multiple copies in a distributed network, simply identifying a serialising entity is not enough
- The problem is that, the home or serialising agent may know when its involvement with a request is completed, BUT this does not mean that a request is completed with respect to other nodes
- Some transactions for the next request to that block may reach other nodes and perform with respect to them, before some remaining transactions for the previous request
- We will see examples and solutions for this in the 2 case studies...
  - In SGI Origin and Sequent NUMA-Q .. later in the course ...
- This shows that in addition to the system providing a global serialising entity for a block, individual nodes (ex requestors) should also preserve a local serialisation wrt each block.
  - Eg: they should not apply incoming transactions to a block while they still have a transaction outstanding for that block


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=5
4-Oct-2021
6/6
Share
Set background
Clear frame
R1
R2
Result
72% (0:27) Thu, October 7, 14:29
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
guptal8e
ABHAY PRATAP G...
VATSHAL NILESH...
VEDIKA JITENDR...
NALABOLU SAN...
TANVISH
ADITYA KUMAR S...
MUNINDRA NAIK
+2

Proving Correctness

- Deadlock freedom - Source of deadlock in strict request-response is filling up of finite input buffer space
- 3 solutions:
- (i) provide enough buffers considering all combinations of maximum transactions
- (ii) use NACKs
- (iii) Provide separate request-response network. Physical or multiplexed with separate buffers
- Two separate networks are sufficient for strict request-response --> request generate only response and response generates no further transactions
- However, for performance we break away from strict request-response model
- Number of networks required will be the longest chain of transaction types needed to complete a given operation, so that the transaction at the end of the chain makes process
- Multiple networks are expensive and most of the time they are under-utilised


4-Oct-2021 - Google
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=5
4-Oct-2021
6/6
Share
Set background
Clear frame
R1
R2
Remote
Rok
Res
play
VA
VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
AS
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
G
guptal8e
ABHAY PRATAP G...
VP
VATSHAL NILESH...
VK
VEDIKA JITENDR...
NS
NALABOLU SAN...
TANVISH
AS
ADITYA KUMAR S...
MN
MUNINDRA NAIK
+2
75% (0:27) Thu, October 7, 14:33

Correctness : deadlock freedom

- For protocols that are not strict request-response two solutions: NACK or become strict
- Pretend that its strict request-response and
- (i) provide two real/virtual networks
- (ii) then detect that deadlock may happen and try to avoid deadlock by NACK or strict
- How to detect potential deadlock situation?
- (i) when both input and output request buffer fill up beyond a threshold and the request at head of input request buffer is one that may generate further requests like intervention / invalidation
- (ii) when output request buffer is full and has not had a transaction removed from it for 'T' cycles

4-Oct-2021 - Google

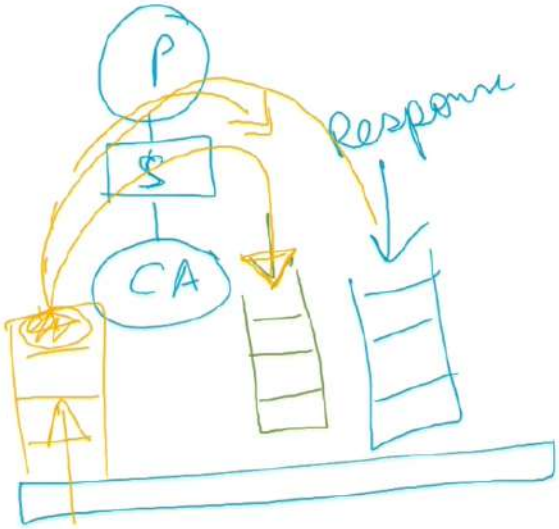
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=6

4-Oct-2021

7/7

Share

Set background | Clear frame



VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
guptal8e
ABHAY PRATAP G...
VATSHAL NILESH...
VEDIKA JITENDR...
NALABOLU SAN...
TANVISH
ADITYA KUMAR S...
MUNINDRA NAIK
+2

83% (0:27) Thu, October 7, 14:43

On potential deadlock?

- When potential deadlock is detected
- DASH system sends NACK
  - Takes requests from its input queue and sends NACK one-by-one
  - Until head of queue has request that no longer generates further requests OR until it finds that output request queue is no longer full
  - NACKed requestors will retry later
- Origin 2000 does not NACK
  - The node sends a response asking the requestor to do the invalidations or intervention directly with the sharers, i.e. it dynamically backs-off to strict request-response
  - Advantage of this is that NACK is not robust solution and may result in congestion when several retries are required
    - Increases network traffic and latency
  - Dynamic back-off also helps for livelock

4-Oct-2021 - Google

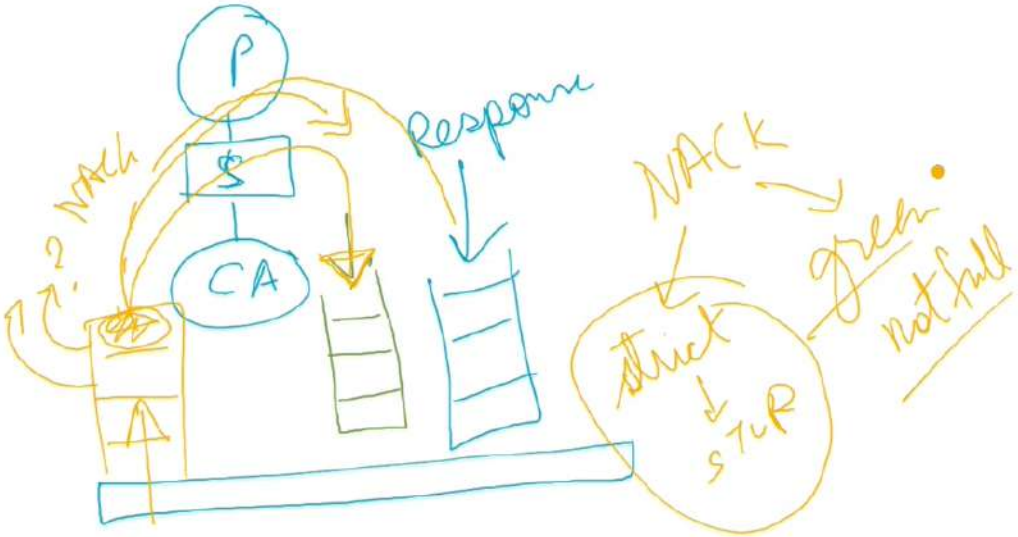
jamboard.google.com/d/1xgN9R4EQq0OZrDzyhb5wMm6ZrOqQYsfUxoR5tdgHi1U/viewer?f=6

4-Oct-2021

7/7

Share

Set background | Clear frame



VARHADE AMEY...
Syam Sankar
KARTIKEYA SAXE...
ASWATHY N S
K CHITRA
DODDAYULA LIK...
IMLIJUNGLA LON...
Hemangee Kalpe...
KOUSIK RAJESH
guptal8e
ABHAY PRATAP G...
VATSHAL NILESH...
VEDIKA JITENDR...
NALABOLU SAN...
TANVISH
ADITYA KUMAR S...
MUNINDRA NAIK
+2

84% (0:27) Thu, October 7, 14:46

Livelock freedom

- Protocols that avoid deadlock by enough buffering of requests, have livelock+starvation taken care of automatically as long as buffers are FIFO

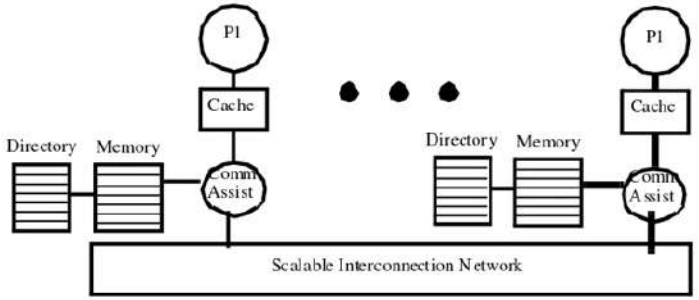
- Else livelock due to multiple nodes trying to write to same location at the same time
  - Solution= allow request going to home FIRST to complete and NACK others
- NACKs good to avoid livelock due to race conditions
- But, if NACK is used to for avoiding deadlock, then they may lead to livelock ! [DASH sends NACK to avoid deadlock]
  - As all NACK requests may be retried at the same time
- Alternative solution to deadlock that does not cause livelock is to use strict request-response once deadlock is predicted. It guarantees forward progress

Starvation freedom

- Unlikely in well designed protocols
- Use of FIFO is better
- But if not FIFO then problem
- Use NACKs Retry
- But some don't get chance
  - Don't do anything
  - Wait for variability of delay in network or system to solve starvation
  - Add random delay between retries
  - Assign priority to request based on number of times they are NACKed

Scalable CC-NUMA Design case study
SGI Origin 2000

Recap

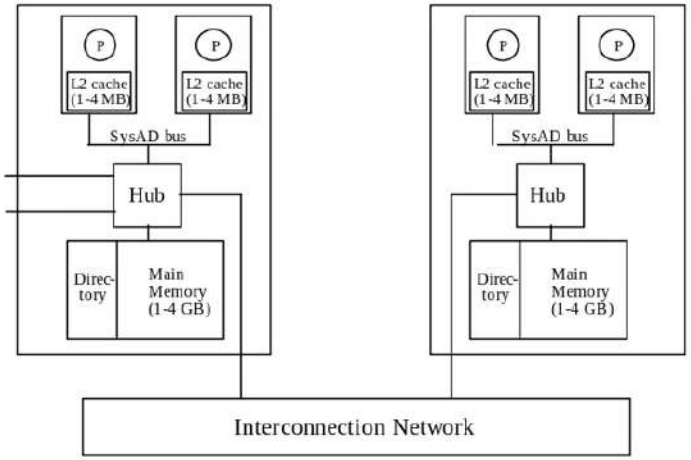


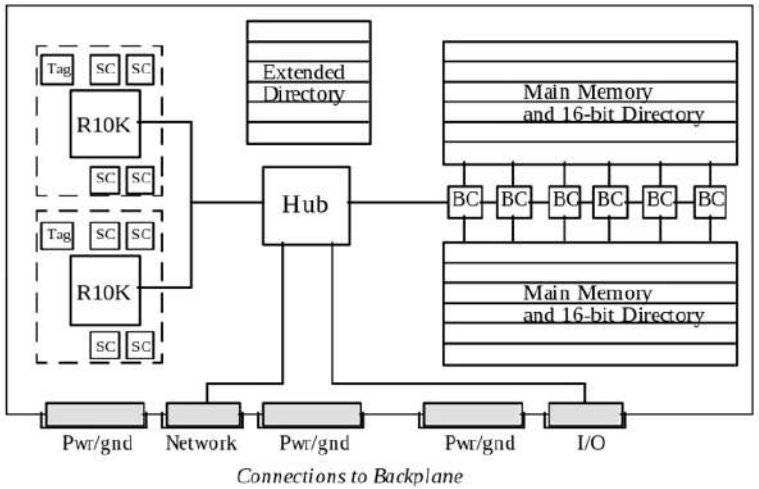
- Flat, memory-based directory schemes maintain cache state vector at block's home
- Protocol realized by network transactions
- State transitions serialized through the home node
- Completion requires waiting for invalidation acks

Origin 2000 system overview

- Single 16”-by-11” PCB
- Directory state in same or separate DRAMs, accessed in parallel
- Upto 512 nodes (1024 processors)
- Each node has 2 processors: MIPS R10000
- With 195MHz R10K processor, peak 390MFLOPS or 780 MIPS per proc
- Peak SysAD bus b/w is 780MB/s, so also Hub-Mem
- Hub to router chip and to Xbow is 1.56 GB/s (both are off-board)
- Hub has 4 outstanding transaction buffers
  - Connects processor, memory, network and I/O
  - Provides synch primitives
- 2 processors treated independently

Origin 2000 system overview

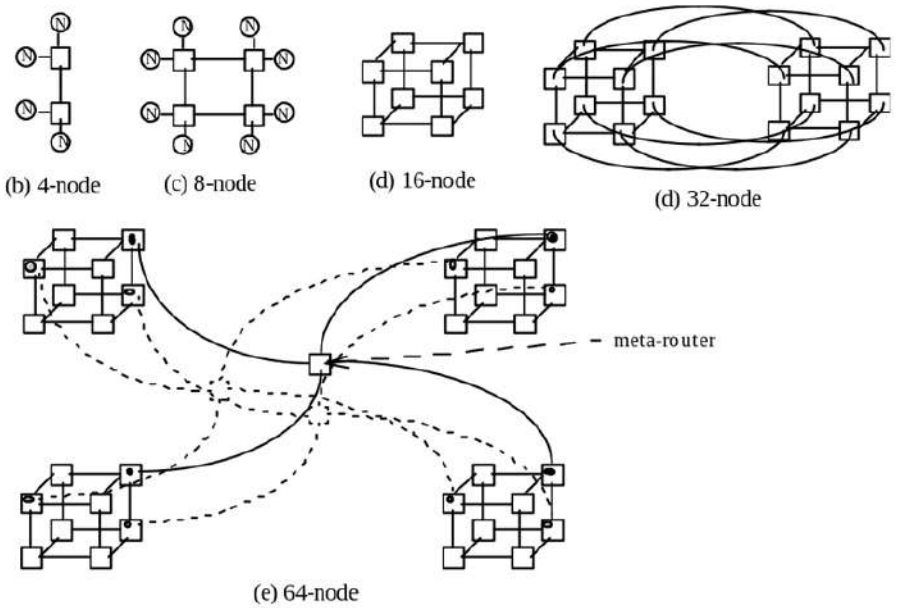




Origin Directory Structure

- Flat, Memory-based: All directory information is at the home
- Complex as needs to scale to more than 64-nodes with 64-bit entry
- Three possible formats or interpretations
  - Exclusive
    - If a block is in exclusive state in a processor-cache then the rest of the directory-entry is not a bit-vector but an explicit pointer to the processor (not to the node)
  - Shared: bit-vector
    - If the block state is shared the directory entry is a bit-vector
    - Bits correspond to nodes (not processors)
    - The 2-proc in node are not snoop-coherent but unit of visibility to the directory is the node
    - Invalidation to a node are broadcast by Hub to both processors
    - Bit vector sizes: 16-bit format (32 procs): keep in main memory, DRAM ; 64-bit format (128 procs): extra bits in extension memory
  - Larger system: coarse vector ..

Origin network



- Each router has six pairs of 1.56GB/s unidirectional links
- Two to nodes, four to other routers
- latency: 41ns pin to pin across a router
- Flexible cables up to 3 ft long
- Four “virtual channels”: request, reply, other two for priority or I/O