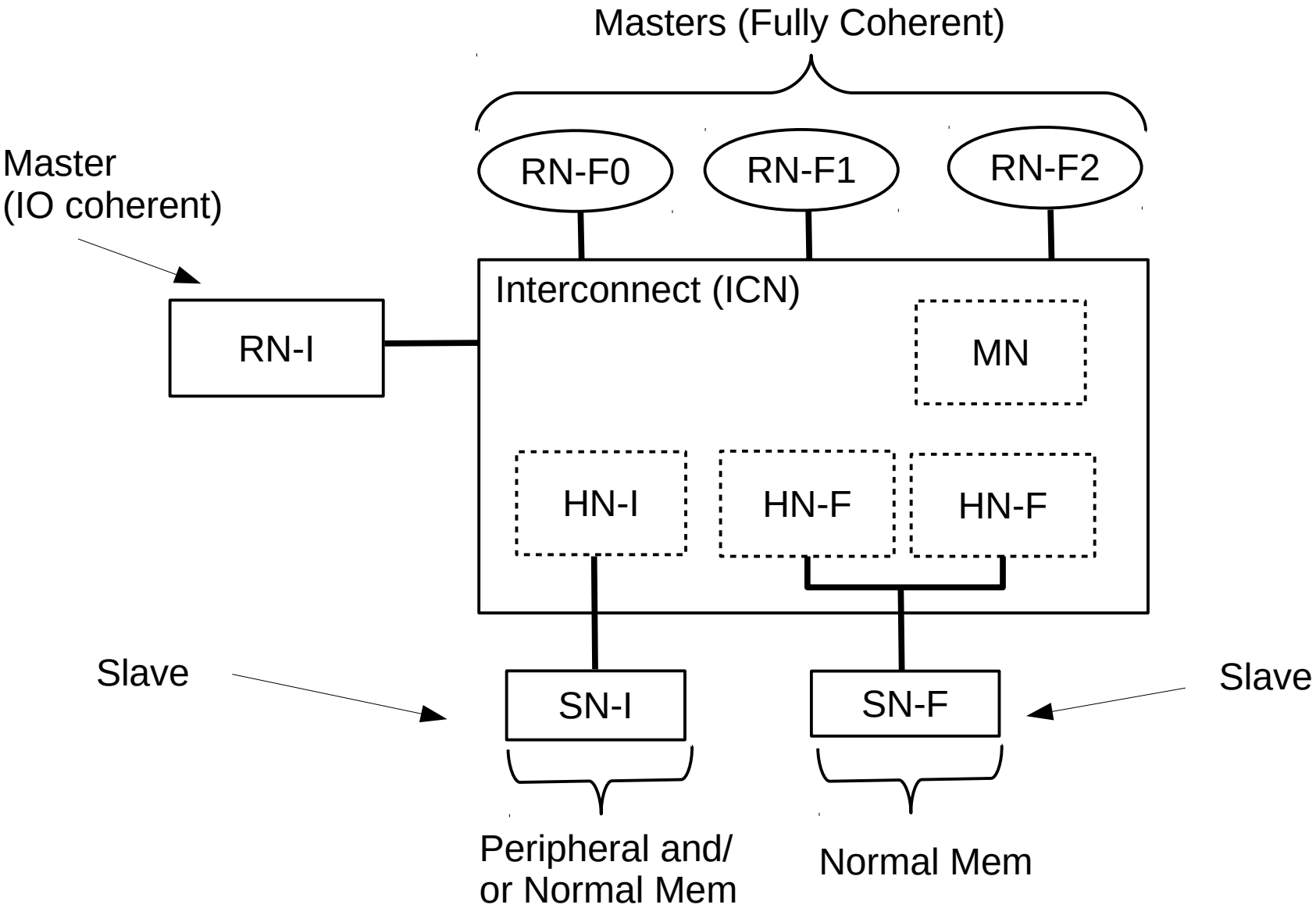
ARM AMBA 5 CHI NOTES

last updated: 28-08-19

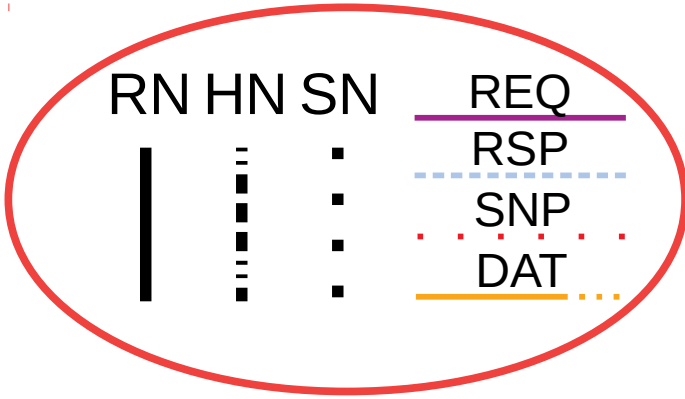
CHI Components (1/2)



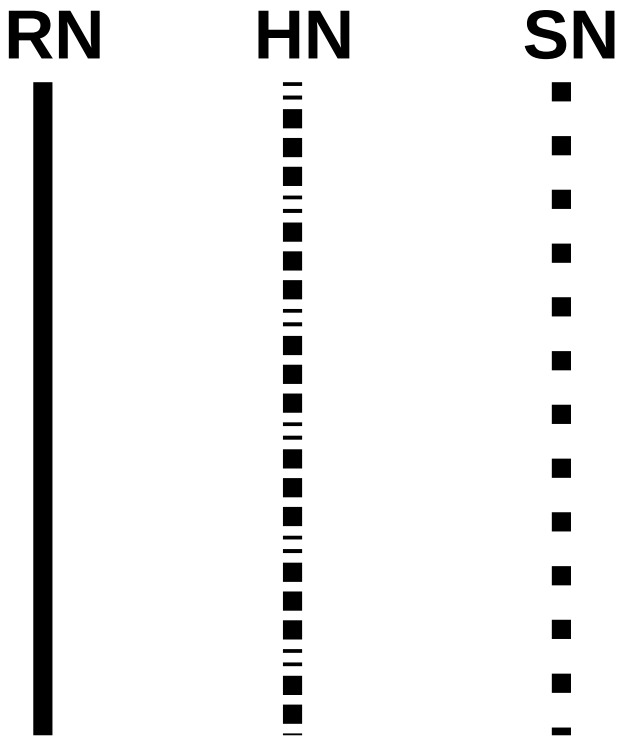
CHI Components (2/2)

- **RN:**
  - **RN-F:** Full coherent reqs, Incl. HW$, Gens all prot reqs, snoops trans
  - **RN-D:** IO coh, no HW$, recv DVM, gens sub of prot reqs
  - **RN-I:** IO coh, no HW$, no DVM, gens sub of prot reqs, no snoop
- **HN:**
  - **HN-F:** recvs all reqs, no DVM, incl. PoC (snoops RN-Fs, sends resp to RN), incl. PoS, might has dir or snoop filter
  - **HN-I:** mngs sub of prot reqs, incl PoS, no PoC, no snoop
  - **MN:** recv DVM
- **SN:**
  - **SN-F:** recv non-snooped, R/W atomic, CMO reqs from HN for normal mem
  - **SN-I**: recv non-snooped, R/W atomic, CMO reqs from HN for normal mem or peripherals

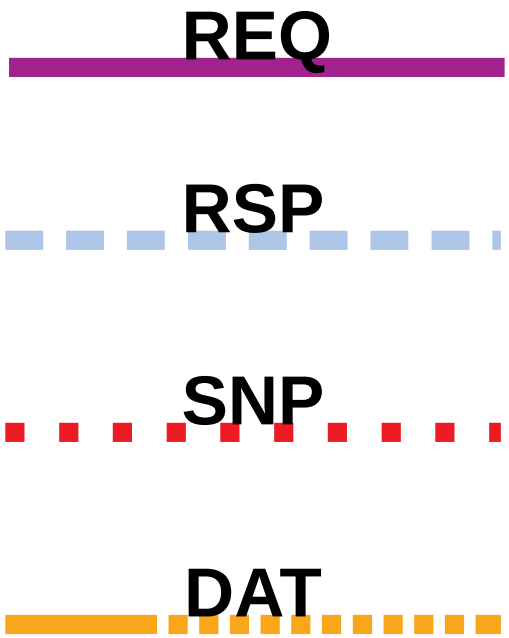
Channels & Nodes



Nodes:



Channels:

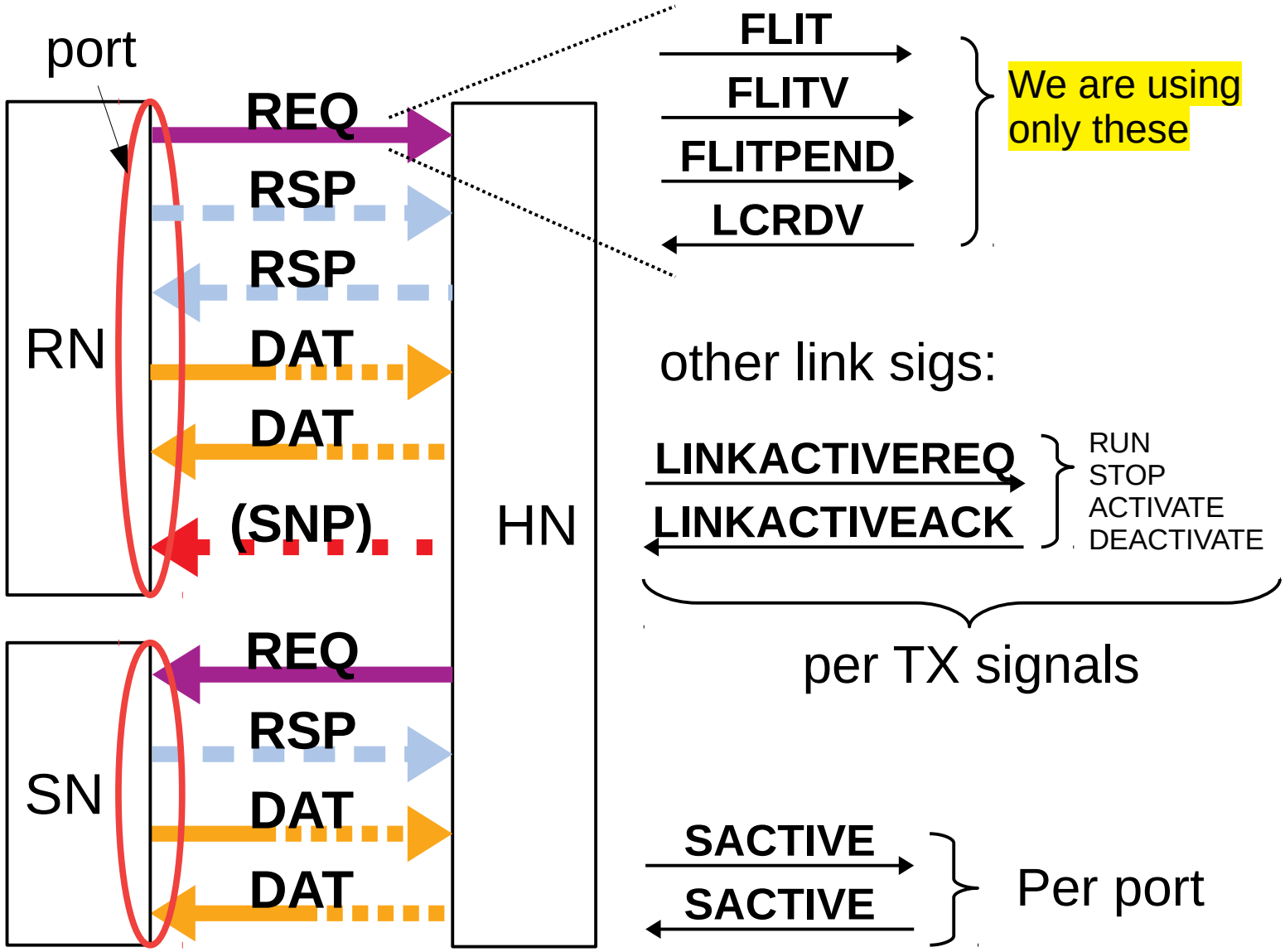


Channel dir & link sigs

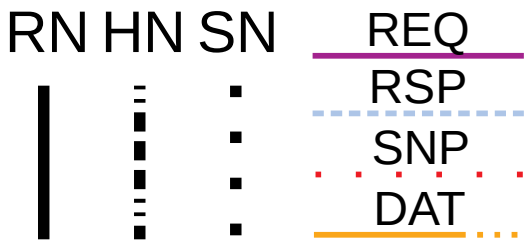
RN cannot:
- accept req
- Issue snp

HN cannot
- accept snp

SN cannot:
- issue req
- accept resp
- issue snp
- accept snp



Channel Ports



- **RN** utilizes **6** ports:
  - ISNP (input snoop)
  - OREQ (output request)
  - IRSP (input response)
  - ORSP (output response)
  - IDAT (input data)
  - ODAT (output data)
- **HN** utilizes **7** ports:
  - OSNP (output snoop)
  - IREQ (input request)
  - OREQ (output request)
  - IRSP (input response)
  - ORSP (output response)
  - IDAT (input data)
  - ODAT (output data)
- **SN** utilizes **4** ports:
  - IREQ (input request)
  - ORSP (output response)
  - IDAT (input data)
  - ODAT (output data)

Channel priorities for RN

HIGH	IRSP, IDAT
MID-HIGH	ORSP, ODAT
MID-LOW	ISNP
LOW	OREQ

Priority PROOF:

ISNP > OREQ (1)
ORSP > ISNP (2)
ODAT > ISNP (3)
IRSP > ALL (4)
IDAT > ALL (5)

(2,3) : ORSP, ODAT > ISNP (6)
(4,5) : IRSP, IDAT > ALL (7)

(1,6,7):
IRSP, IDAT > ORSP, ODAT > ISNP > OREQ

Based on page 12-301

Channel priorities for SN

HIGH	IDAT
MID-HIGH	ORSP
MID-LOW	IREQ
LOW	ODAT

Priority PROOF:

ORSP > IREQ (1)
IREQ > ODAT (2)
IDAT > ALL (3)
(1,2,3):
IDAT > ORSP > IREQ > ODAT

Based on page 12-302

CHI Requests (1/2)

Reads (9):
ReadNoSnp (RN→HN, HN→SN)
ReadNoSnpSep (HN→SN)
ReadnOnce (RN→HN-F)
ReadOnceCleanInvalid (RN→HN-F)
ReadOnceMakeInvalid (RN→HN-F)
ReadClean (RN-F→HN-F)
ReadNotSharedDirty (RN-F→HN-F)
ReadShared (RN-F→HN-F)
ReadUnique (RN-F→HN-F)

Writes(10):
WriteNoSnp[Ptl,Full] (RN→HN, HN→SN)
WriteUnique[Ptl,Full,PtlStash,FullStash] (RN→HN-F)
WriteBack[Ptl,Full] (RN-F→ HN-F)
WriteCleanFull (RN-F→ HN-F)
WriteEvictFull (RN-F→ HN-F)

CHI Requests (2/2)

Atomics (4) - (RN→HN, HN→SN)**:**
AtomicStore
AtomicLoad
AtomicSwap
AtomicCompare

Misc (3):
DVM (RN-F,RN-D → MN)
PrefetchTgt (RN→ SN-F)
PcrdReturn (RN→HN, HN→SN)

Dataless (9):
CleanUnique (RN-F→HN-F)
MakeUnique (RN-F→HN-F)
Evict (RN-F→HN-F)
StashOnceUnique (RN→HN-F)
StashOnceShared (RN→HN-F)
CleanShared (RN→HN, HN→SN)
CleanSharedPersist (RN→HN, HN→SN)
CleanInvalid (RN→HN, HN→SN)
MakeInvalid (RN→HN, HN→SN)

CHI Snoops

Snoops (18) - (HN-F → RN-F):
SnpOnce*Fwd*
SnpOnce
SnpClean*Fwd*
SnpClean
SnpNotSharedDirtyFwd
SnpNotSharedDirty
SnpShared*Fwd*
SnpShared
SnpUnique
SnpUnique*Fwd*
SnpCleanShared
SnpCleanInvalid
SnpUniqueStash
SnpStashUnique
SnpStashShared
SnpMakeInvalidStash
SnpMakeInvalid
SnpDVMOp

Read* (9 in total)

We are using the bold ones

ExpCompAck: Bit indicating if requestor will sen a CompAck (Completion Ack)

ReadNoSnp (RN→HN, HN→SN)

ReadNoSnpSep (HN→SN)

ReadOnce (RN→HN-F)

ReadOnceCleanInvalid (RN→HN-F)

ReadOnceMakeInvalid (RN→HN-F)

ReadClean (RN-F→HN-F)

ReadNotSharedDirty (RN-F→HN-F)

ReadShared (RN-F→HN-F)

ReadUnique (RN-F→HN-F)

ExpCompAck optional

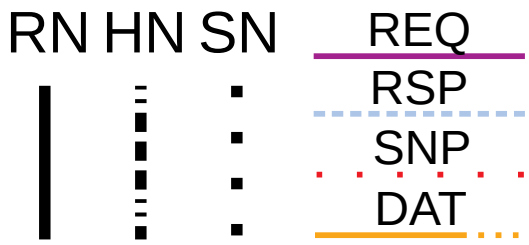
Snoopable w/ no $ alloc

Snoopable w/ $ alloc ExpCompAck must be used

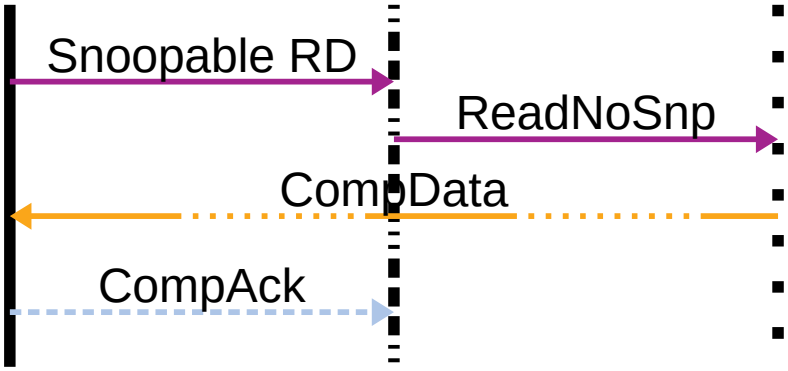
Can be X req w/ no DDT

DDT: Direct Data Transfer
"alloc": MSB bit of mem_attr field of request flit
X req: "excl" bit of request flit

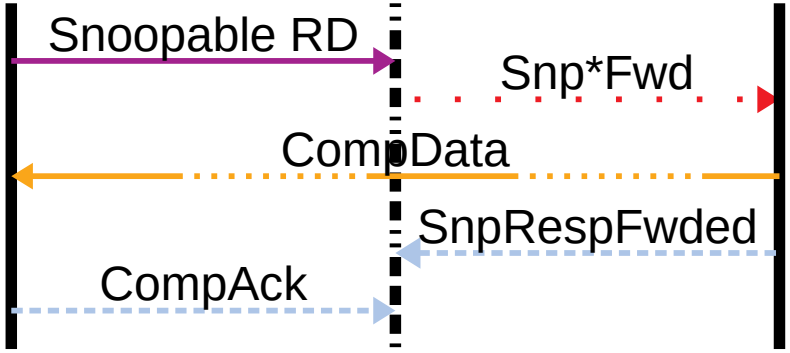
Snpable RD w/ DDT



w/ DMT:

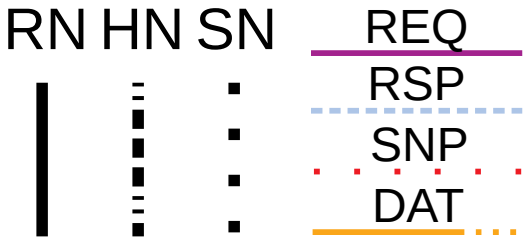


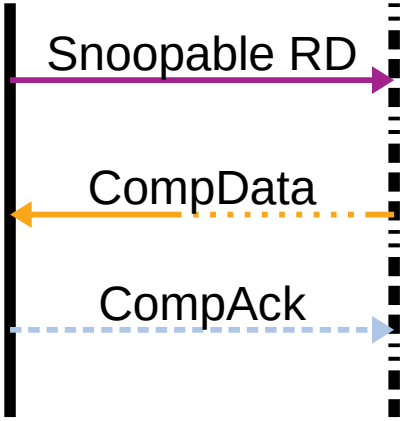
w/ DCT:



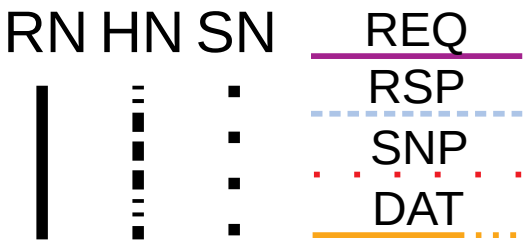
- CompAck can be sent after at least 1 CompData has been rcv'ed.
- The RN can re-use the TxnID only after the corresponding resp's have been cmplted.
- The HN can send DMT trans when:
  - no Snoop is needed to be sent
  - Snoop resp does not contain dirty $line
  - Any returned partial dirty data from snooping has been wr. back to SN
  - Any FwdSnp did not resulted in the RN's $line
  - If any DCT trans in pending, DCT should cmplt first.

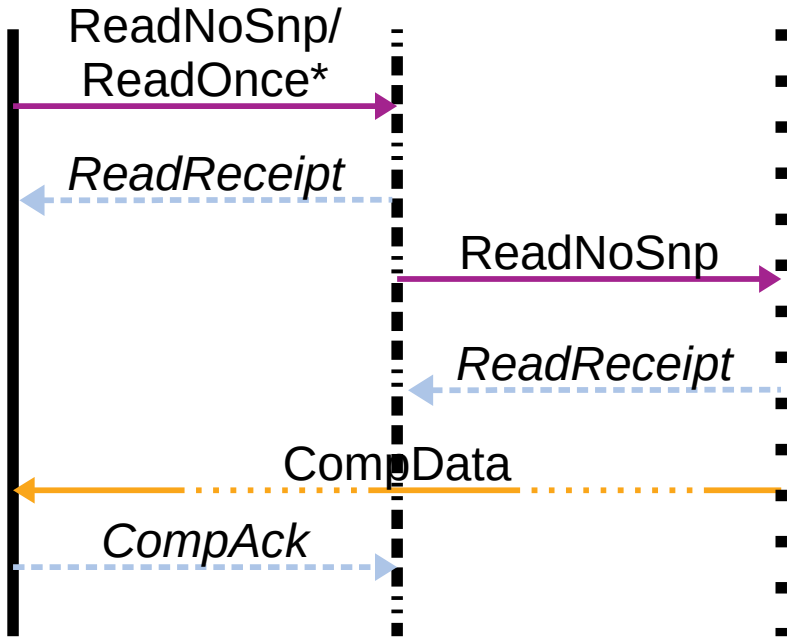
Snpable RD w/o DDT





ReadNoSnp, ReadOnce* w/ DMT

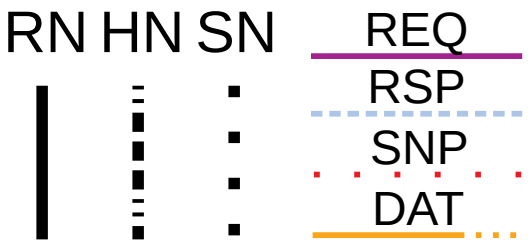


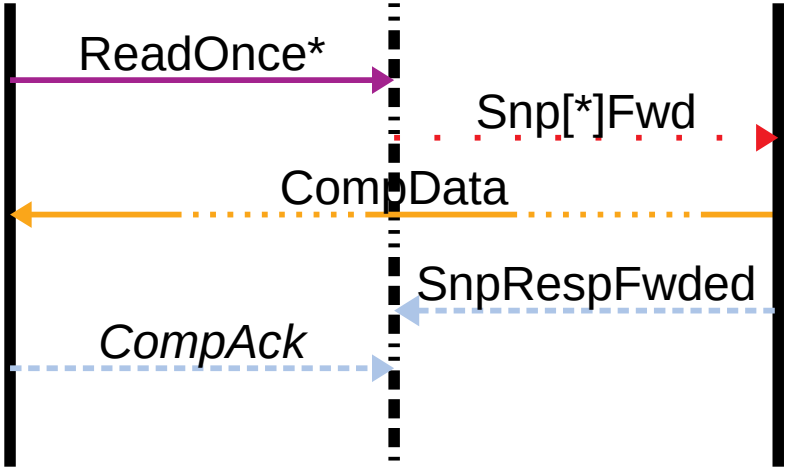


- ReadReceipt (optional) can be requested (order field of req packet) to deallocate the HN. If it is sent, the SN will not send RetryAck resp.
- The HN is able not to wait the CompAck if it has received a ReadReceipt. However it can accept it anyway.

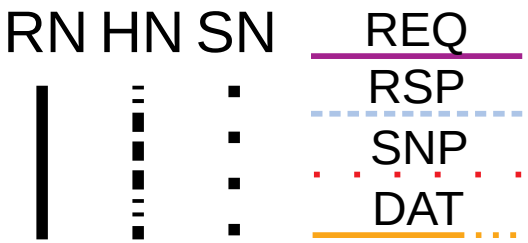
ReadNoSnp, ReadOnce* may have ordering requirement
ReadNoSnp, ReadOnce* may have the ExpCompAck asserted however it is not functionally required. It may be used when DMT and separate Comp/Data is required in some cases.

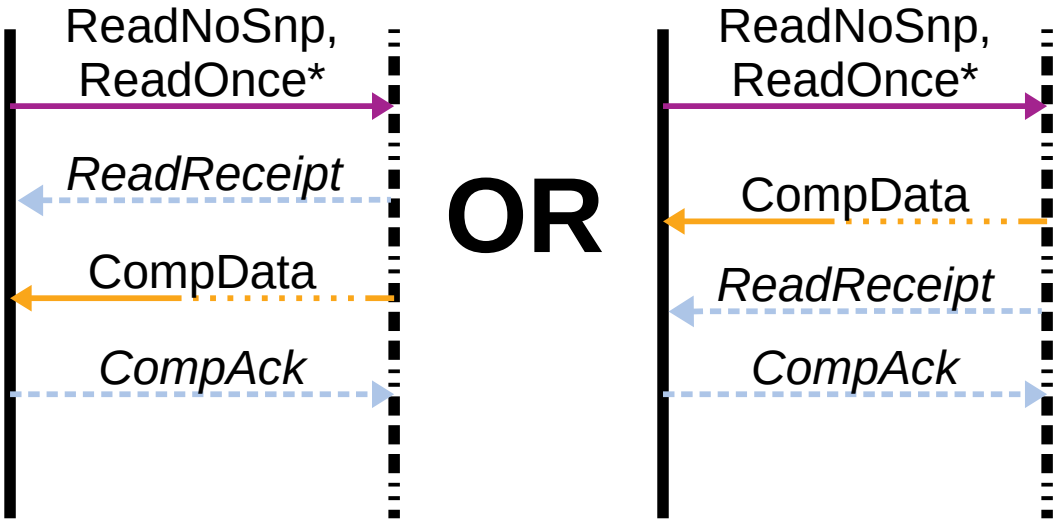
ReadOnce* w/ DCT





ReadNoSnp, ReadOnce* w/o DDT





- Requester can wait if he wants to, for the read receipt first before sending CompAck

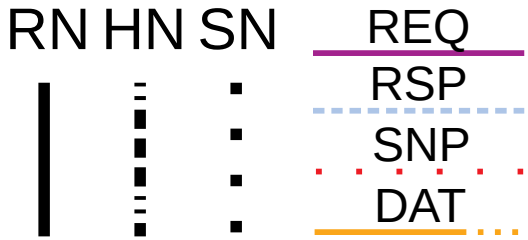
Italics format: optional usage

DMT/DCT under: order/CompAck for ReadNoSnp, ReadOnce* from RN

- order=2’b00, ExpCompAck=0:
  - DMT: HN must req to the SN ReadReceipt.
- order=2’b00, ExpCompAck=1:
  - DMT: HN may issue (optional) req to SN with ReadReceipt, its OK with the CompAck.
- order=2’b01:
  - No DDT is permitted.
- order=2’b10/2’b11, ExpCompAck=0:
  - DCT: HN uses SnpRespFwded / SnpRespDataFwded resp for trans. completion.
- order=2’b11, ExpCompAck=1:
  - DMT: HN uses CompAck resp for Completion.
  - DCT: HN uses SnpRespFwded / SnpRespDataFwded resp for trans. completion.

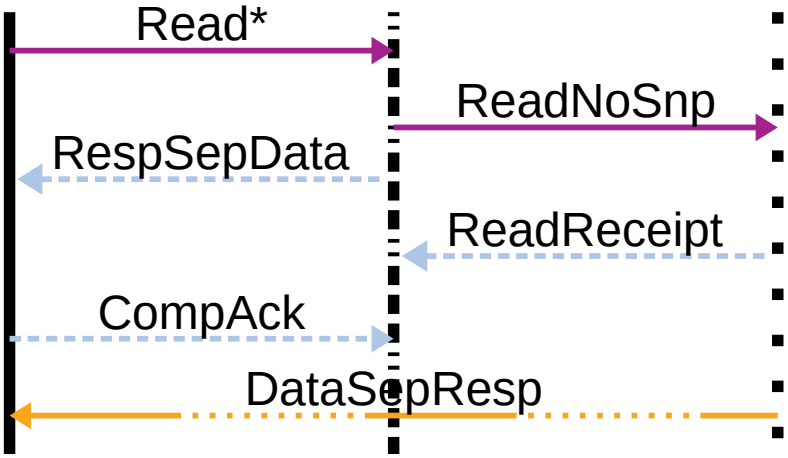
Order bits:
2’b00: no ordering. | **2’b01**: req accepted.
2’b10: Req order/ordered write observation required.
2’b11: Endpoint order required included req order.

Reads w/ sep. resps

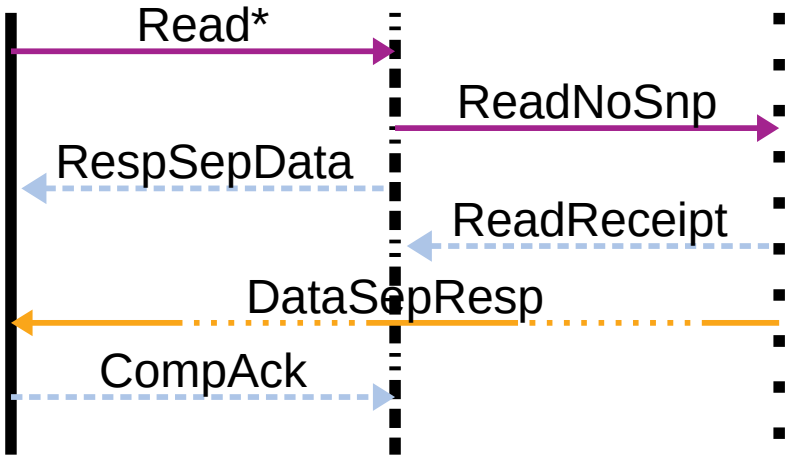


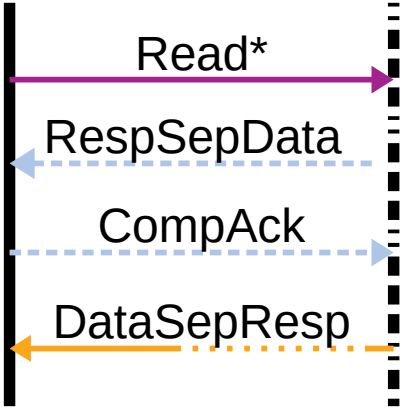
- TBA
- pages: 2-50 .. 2-54

Check page 2-99

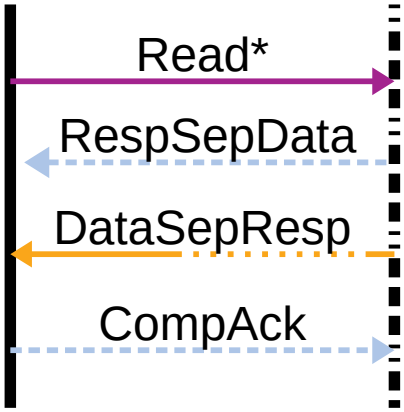


OR





OR

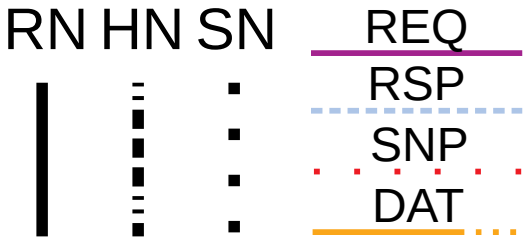


Note:
When ordering requirement is applied (Order field = b10/b11 + ExpCompAck = 1) CompAck is given after DataSepResp + RespSepData.

Dataless (9)

We are using the bold ones

Can be Excl=1

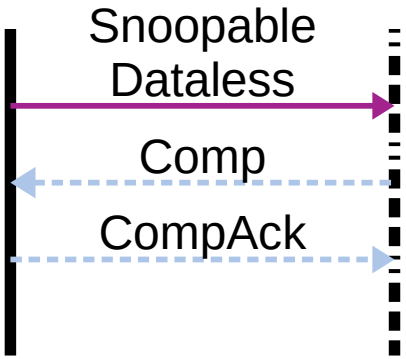


- **CleanUnique** (RN-F→HN-F)
- MakeUnique (RN-F→HN-F)

ExpCompAck must be used

- Evict (RN-F→HN-F)
- StashOnceUnique (RN→HN-F)
- StashOnceShared (RN→HN-F)
- **CleanShared** (RN→HN, HN→SN)
- CleanSharedPersist (RN→HN, HN→SN)
- **CleanInvalid** (RN→HN, HN→SN)
- **MakeInvalid** (RN→HN, HN→SN)

CMO



Write* (10)

Can be Excl=1

ExpCompAck
optional

- **WriteNoSnp**[Ptl,Full] (RN→HN, HN→SN)
- **WriteUnique**[Ptl,Full,PtlStash,FullStash] (RN→HN-F)
- **WriteBack**[Ptl,Full] (RN-F→ HN-F)
- **WriteCleanFull** (RN-F→ HN-F)
- WriteEvictFull (RN-F→ HN-F)

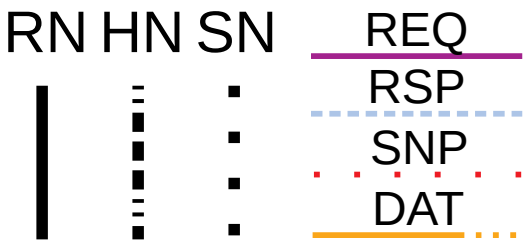
CopyBack (4)

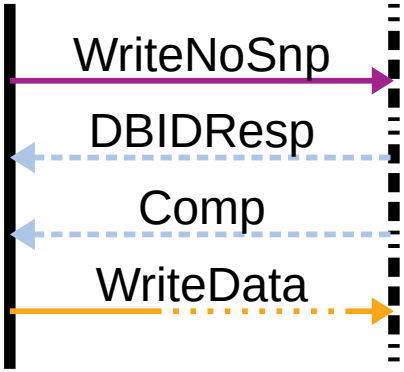
Update mem/$

Update mem only

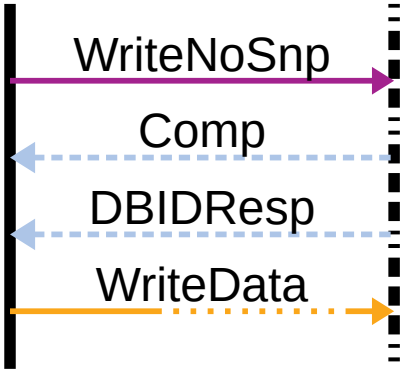
We are using the bold ones

WriteNoSnp (store w/o snp'n other msts)

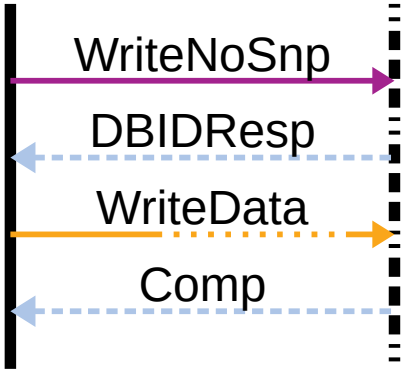


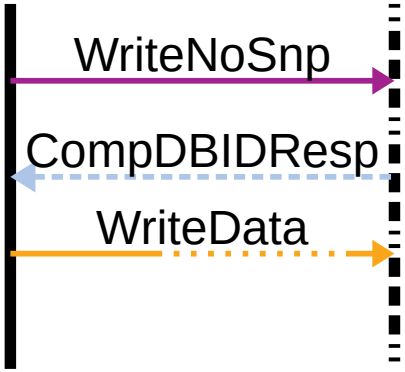


OR

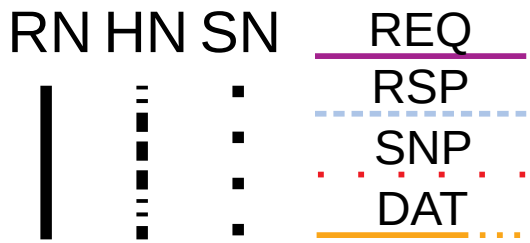


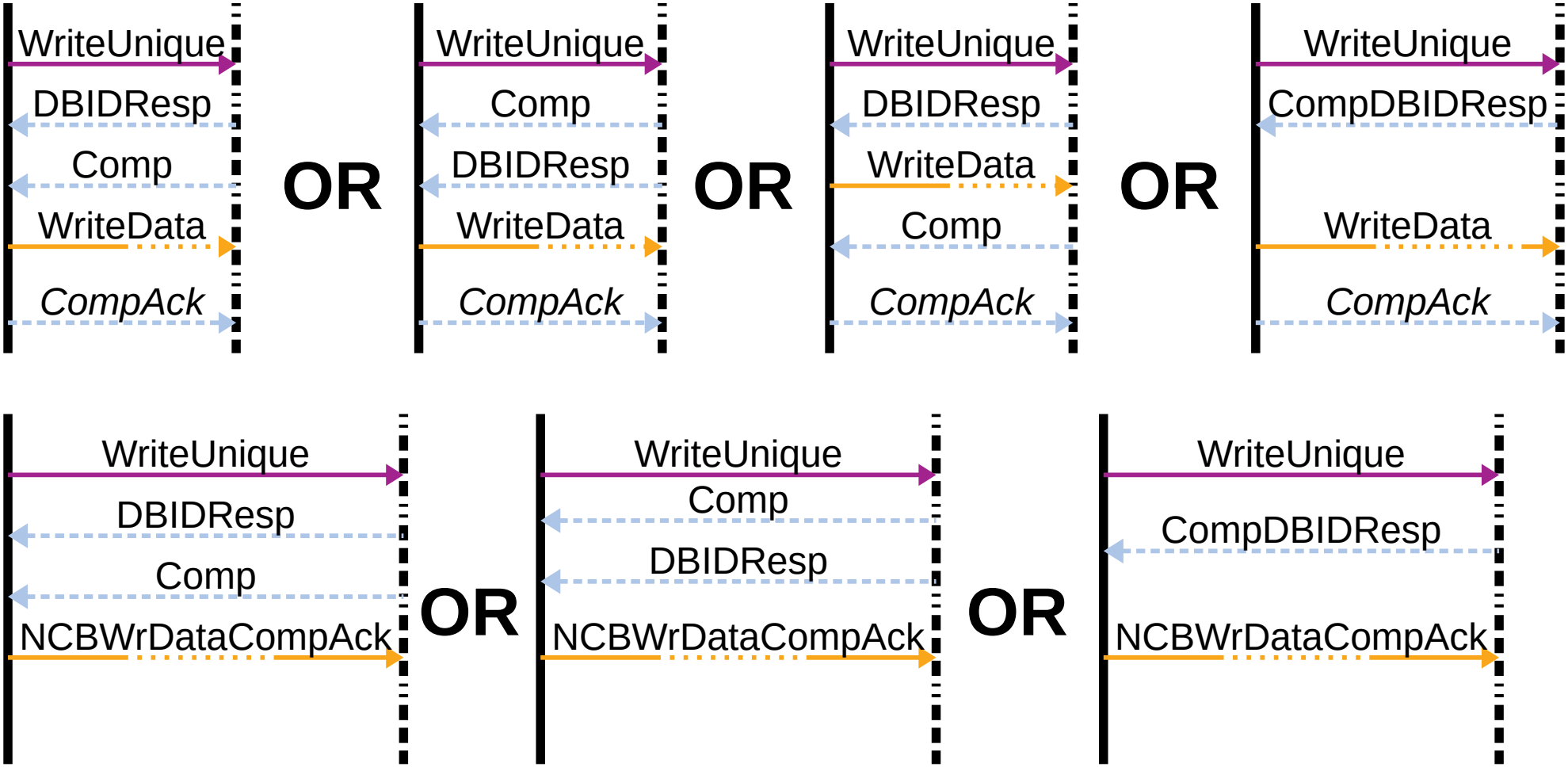
OR



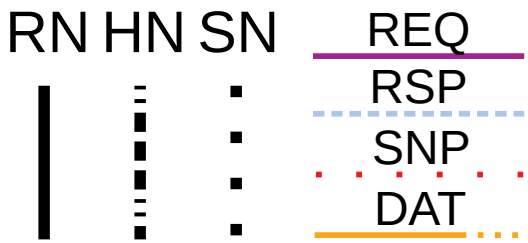


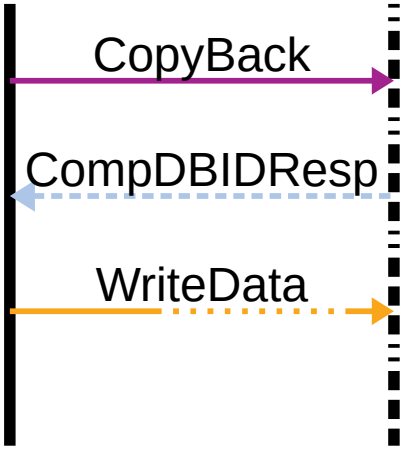
WriteUnique (store w/ snp'n other RN-Fs)





CopyBack





Recall CopyBack transactions:

WriteBack[Ptl,Full] (RN-F → HN-F)

WriteCleanFull (RN-F → HN-F) ← Update mem/$

WriteEvictFull (RN-F → HN-F) ← Update mem only

CompDBIDResp should be sent after any pending snoop request has been completed

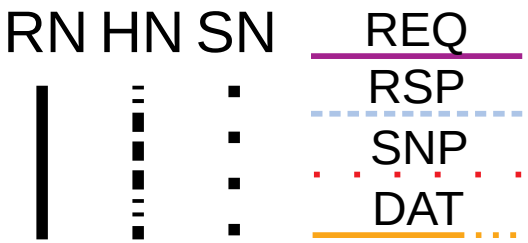
Atomics (4) - (RN→HN, HN→SN)

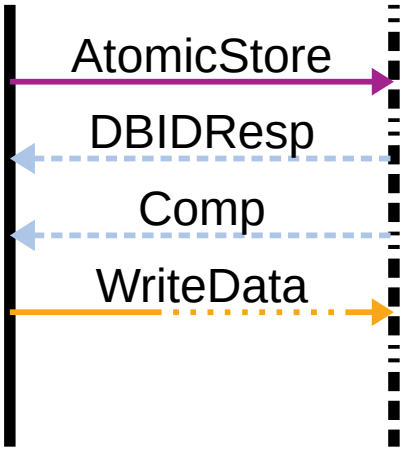
- **AtomicStore** → 1-way data
- **AtomicLoad**
- **AtomicSwap**
- AtomicCompare

} 2-way data

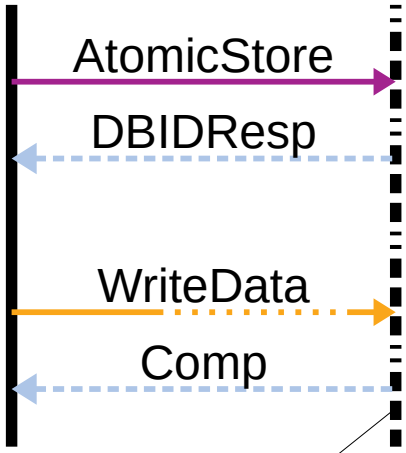
We are using the bold ones

Atomics



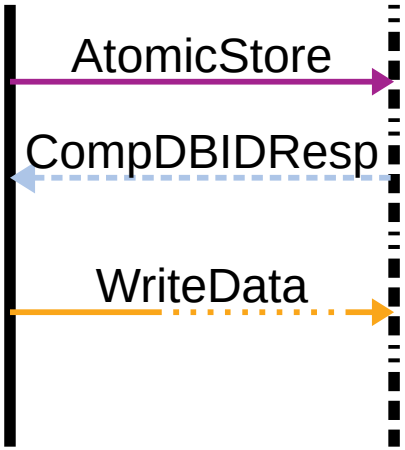


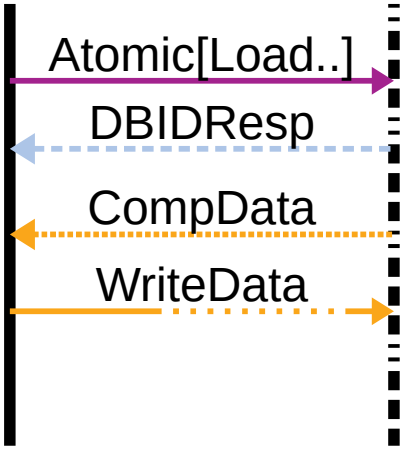
OR



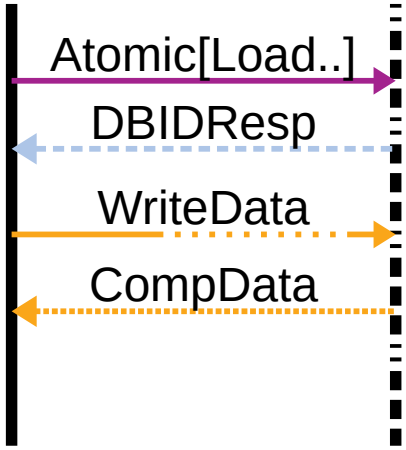
Permits the possibilty of error resp. (impl. defined)

OR

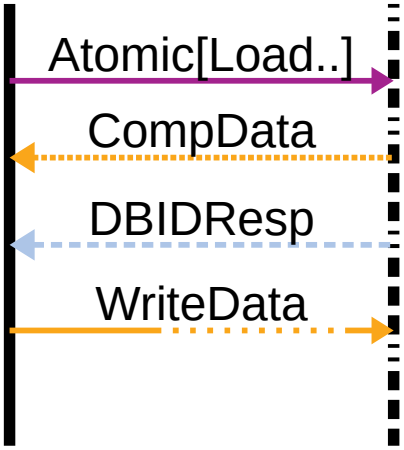




OR



OR



Self-snoop

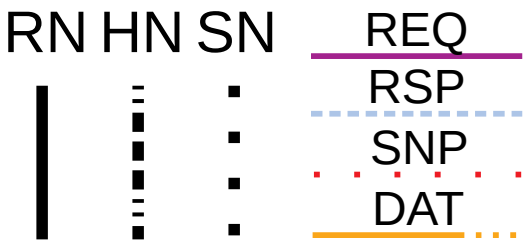
- Usage: When an RN does not invalidate its own $line before sending an atomic req, it can use the SnoopMe bit.
- HN:
  - Must send snoop to the RN if the $line resides elsewhere (RN).
  - May send snoop to the RN if the $line is not present elsewhere (RN).
  - May send snoop to the RN if the SnoopMe is zero.
  - May send SnpUnique / SnpCleanInvalid in response to an atomic req.
- RN:
  - May send CopyBack while an atomic req (SnoopMe=1) to the same addr is in progress
  - May issue an Atomic req (SnoopMe=1) while a CopyBack req to the same addr is in progress.

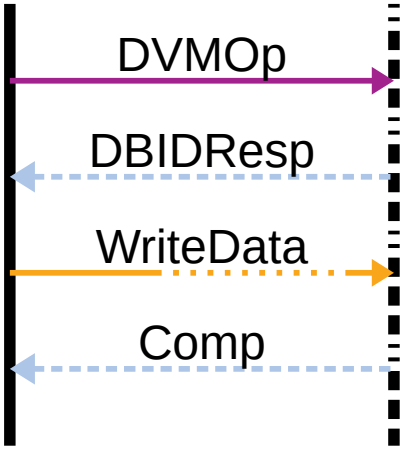
Misc

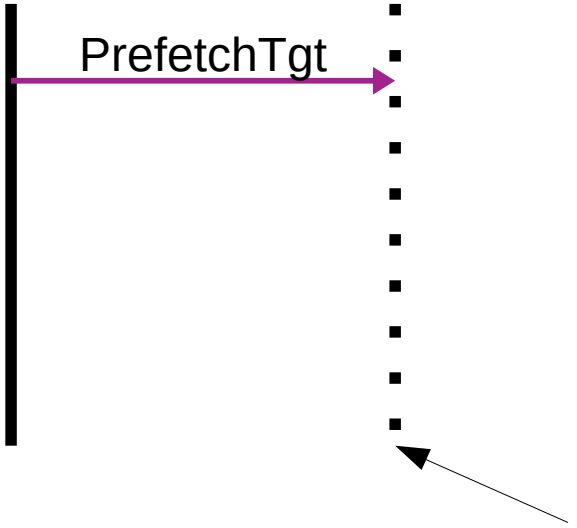
- DVMOp (RN-F,RN-D → MN)
- PrefetchTgt (RN→ SN-F)
- PcrdReturn

No retry seq

DVM, PrefetchTgt

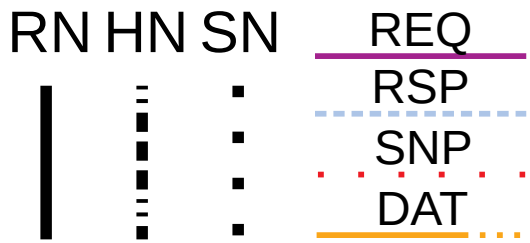




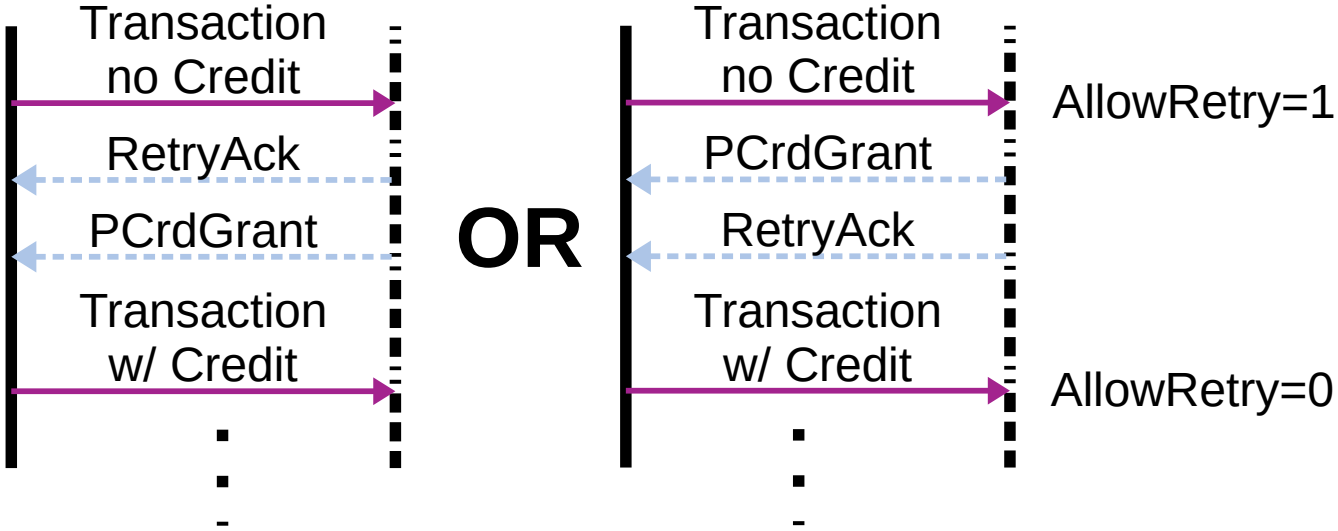


No retry seq

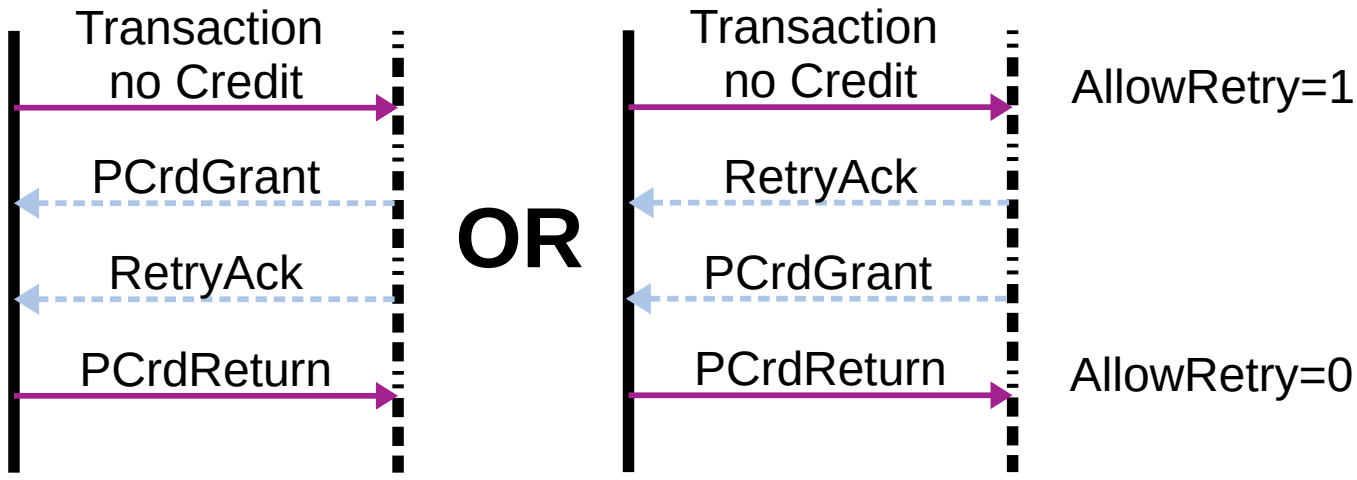
PcrdReturn (Retry)



Case with useful 2nd req



Case with dummy 2nd req. a.k.a *Canceled trans*



Snoops (18) - (HN-F→RN-F)

SnpOnce
SnpClean
SnpNotSharedDirty
SnpShared
SnpUnique
SnpCleanShared
SnpCleanInvalid
SnpOnce*Fwd*
SnpClean*Fwd*
SnpNotSharedDirtyFwd
SnpShared*Fwd*
SnpUnique*Fwd*
SnpUniqueStash
SnpStashUnique
SnpStashShared
SnpMakeInvalidStash
SnpMakeInvalid
SnpDVMOp

Resp + data to HN

Resp + data (or not) to HN + data fwded to RN

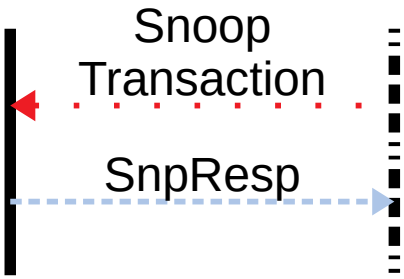
Permitted to be sent only to **ONE** RN-F

Generates 2 Snoop requests, returns 1 snoop resp. Initiated only by DVMOp req.

Resp ONLY to HN

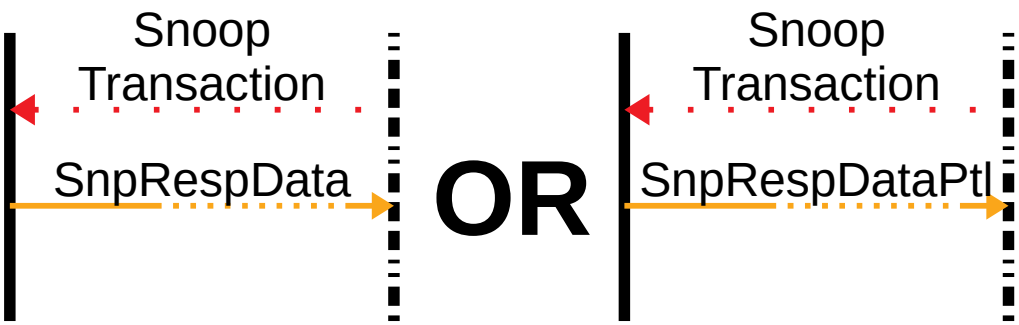
Data-less snoop

Regular Snoop:



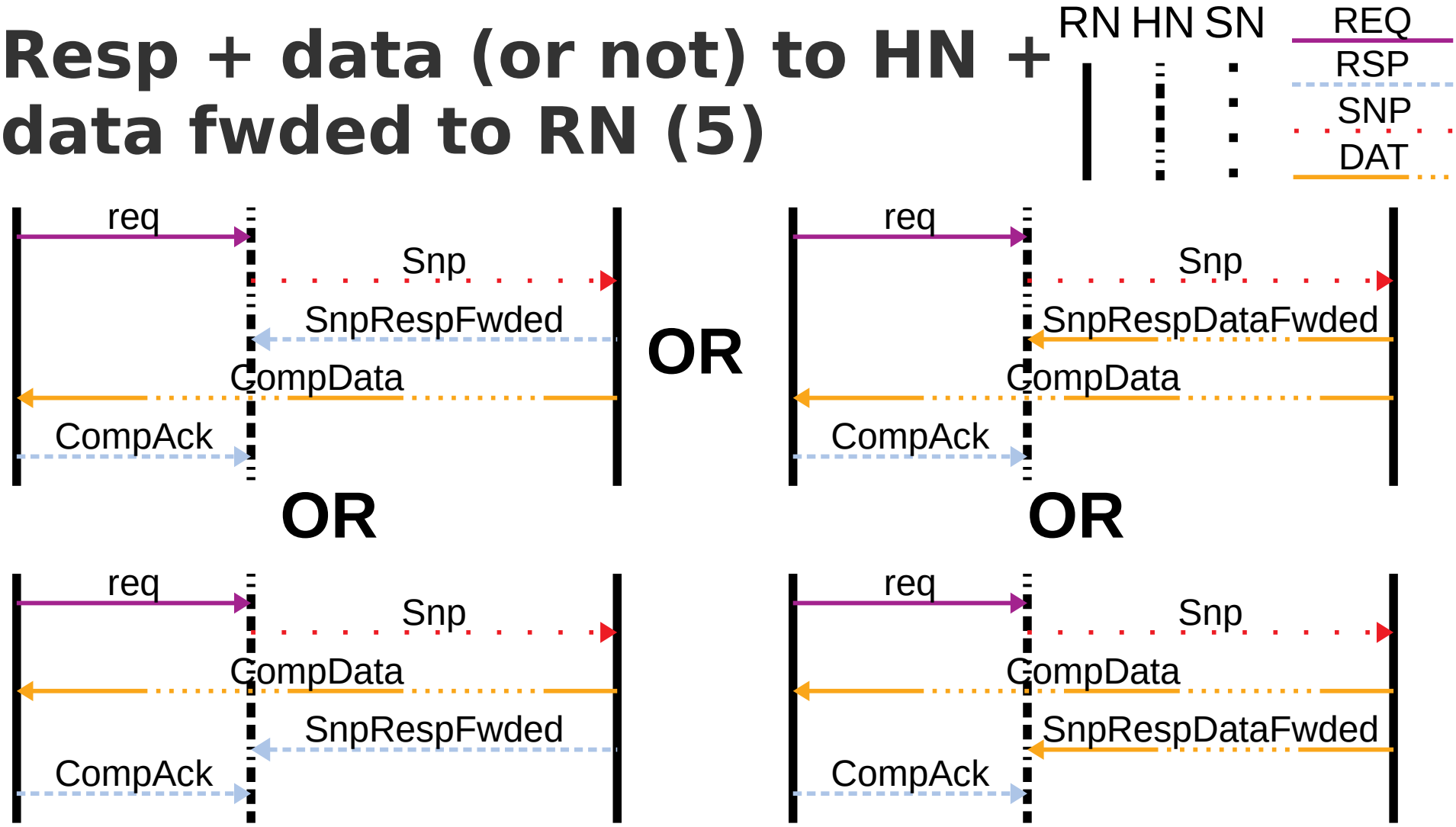
Snoop DVM:

resp+data to HN (7+5)



SnpOnce
SnpClean
SnpNotSharedDirty
SnpShared
SnpUnique
SnpCleanShared
SnpCleanInvalid
SnpOnce*Fwd*
SnpClean*Fwd*
SnpNotSharedDirtyFwd
SnpShared*Fwd*
SnpUnique*Fwd*

Resp + data (or not) to HN + data fwded to RN (5)



SnpOnce*Fwd*, SnpClean*Fwd*, SnpNotSharedDirtyFwd
SnpShared*Fwd*, SnpUnique*Fwd*

Stash snoops

- TBA
- page: 2-71

ALL Fields (47)

Order
Endian
DataSource
CCID
DataID
BE
Data
DataCheck
Poison
RSVDC
NS
DoNotGoToSD
DoNotDataPull
RetToSrc
QoS
TgtID

SrcID
DBID
PcrdType
TraceTag
LPID
ReturnNID
ReturnTxnID
StashNID
StashNIDValid
Addr
TxnID
FwdNID
FwdTxnID
StashLPID
StashLPIDValid
VMIDExt

HomeNID
Opcode
RespErr
Resp
FwdState
DataPull
Size
AllowRetry
PCrdType
ExpCompAck
MemAttr
SnpAttr
SnoopMe
LikelyShared
Excl

REQ pck fields (27)

QoS (4b)
TgtID (7-11b)
SrcID (7-11b)
TxnIN (8b)
LPID (5b)
ReturnNID (7-11b)
ReturnTxnID (8b)
StashNID (7-11b)
StashNIDValid (1b)
StashLPID (5b)
StashLPIDValid (1b)
Opcode (6b)
Addr (44-52b)

NS (1b)
Size (3b)
AllowRetry (1b)
PCrdType (4b)
ExpCompAck (1b)
MemAttr (4b)
SnpAttr (1b)
SnoopMe (1b)
LikelyShared (1b)
Excl (1b)
Order (2b)
Endian (1b)
TraceTag (1b)
RSVDC (0/4/12/16/24/32b)

Total = 117 .. 137 + RSVDC bits

RSP fields (12)

QoS (4b)
TgtID (7-11b)
SrcID (7-11b)
TxnID (8b)
DBID (8b)
PcrdType (4b)
Opcode (4b)
RespErr (2b)
Resp (3b)
FwdState (3b)
DataPull (3b)
TraceTag (1b)

Total = 51 .. 59 bits

DAT fields (20)

QoS (4b)
TgtID (7-11b)
SrcID (7-11b)
TxnID (8b)
HomeNID (7-11b)
DBID (8b)
Opcode (4b)
RespErr (2b)
Resp (3b)
FwdState (3b)
DataPull (3b)

DataSource (3b)
CCID (2b)
DataID (2b)
BE (16/24/32b)
Data (128/256/512b)
DataCheck (0/16/32/64b)
Poison (0/2/4/8b)
TraceTag (1b)
RSVDC (0/4/12/16/24/32b)

DW=128b: Total = 202 .. 214 + RSVDC+DataCheck+Poison bits
DW=256b: Total = 346 .. 358 + RSVDC+DataCheck+Poison bits
DW=512b: Total = 634 .. 646 + RSVDC+DataCheck+Poison bits

SNP Req pck fields (15)

QoS (4b)
TxnID (8b)
FwdNID (7-11b)
FwdTxnID (8b)
StashLPID (5b)
StashLPIDValid (1b)
VMIDExt (8b)
SrcID (7-11b)
Opcode (5b)
Addr (41-49b)
NS (1b)
DoNotGoToSD (1b)
DoNotDataPull (1b)
RetToSrc (1b)
TraceTag (1b)

Total = 84 .. 100 bits

Identifier fields

TgtID: REQ/ DAT/ RSP
SrcID: All

For Routing purpose.

identify individual processing agents within a single Requester.

LPID: REQ
StashLPID: REQ/SNP

node is the Stash target.

StashNID: REQ

TxnID: All
DBID: DAT/RSP
ReturnTxnID: REQ
FwdTxnID: SNP

These fields relate all the packets associated with a single trans.

Identify the node that the resp with data is to be sent to.

ReturnNID: REQ
FwdNID: SNP

DataID: DAT
CCID: DAT

Identify the individual data packets within a transaction.

Identify the node that the CompAck response is to be sent to.

HomeNID: DAT

TxnID/ ReturnNID/ FwdNID/ DBID

TxnID:
- unique per Requester (ex. PrefetchTgt)
- reuse when All corr. Resp. have been received or a Retry Ack resp. has been received
- not required to use the same TxnID when a trans. is retried

ReturnNID (similar to ReturnTxnID):
- ReadNoSnp(Sep) & non-store atomics HN→SN, else zero

FwdNID (similar to FwdTxnID):
- Valid for Snp*Fwd snps HN→RN, else zero
- RN uses it as TgtID for Data Response

DBID:
- Completer sends DBID to Requester. Requester sends back (DAT channel) TxnIN = DBID
- Unique for Write/Atomic/DVMOp & CompAck enabled trans.
- Permitted to have single DBID for 2 requesters
- Can be combined with SrcID in case 256 IDs are being used

CompACK field

- Sent ONLY by an RN!

- **Required** (6x msgs):
  - ReadClean
  - ReadNotSharedDirty
  - ReadShared
  - ReadUnique
  - CleanUnique
  - MakeUnique

- **Optional** (8x msgs)
  - ReadNoSnp
  - ReadOnce
  - ReadOnce[CleanInvalid,MakeInvalid]
  - WriteUnique[Ptl,Full,PtlStash,FullStash]

Identifier field flows

- TBA..
- Pages 2-77 .. 2-94

LPID field

- Valid for (un$able & unsnpable, or dev or X accesses)
  - ReadNoSnp
  - WriteNoSnp
- Valid for (X accesses)
  - Read[Clean,Shared,NotSharedDirty]
  - CleanUnique

Ordering (1/6)

- **Multi-copy atomicity:**
  - All RNs observe the same WR order of the same loc.
  - A RD is returned if a all RNs have seen that write
  - Same loc: 2 $line addrs + NS attribute are the same
- **Ordering between N and N+1 req from the same agent or another agent:**
  - RD @ !$'ed addr-range.: RespSepData or CompData
  - RD @ $'ed loc.: CompData or DataSepResp
    - Furthermore: No snp will be sent to Requestor earlier than RespSepData and later than CompAck.
  - DT-less @ $'ed loc.: Comp
  - WR/atomic @ !$'ed range/Dev-nR(n)E: Comp or CompData
  - WR/atimic @ $'ed loc./Dev-RE: Comp or CompData
  - What if (i) WR @ !$'ed loc *AND* (ii) EWA = 1
    - Make a subsequent WR w/ Endpoint order
    - *OR* make a RD trans after WR

Ordering (2/6)

- **Ordering between Issued reqs and snoops:**
  - RN-F sends CompAck After receiving Comp / RespSepData / CompData or RespSepData + DataSepResp
  - HN-F waits for CompAck before sending a subsequent snoop to the same addr.
    - For CP-back trans:
      - HN-F sends CompDBIDResp <u>ONLY</u> when there are is **<u>NO</u>** pending snoop. That is WriteData acts as an implicit CompAck.
    - For ReadOnce*: HN can send a subsequent snp
  - When an RN-F has a pending req w/ CompAck (excpt for ReadNoSnp/ReadOnce*), it is guaranteed not to rcv a Snp (same addr) between Comp & CompAck.
  - An RN that wants to make ordered ReadNoSnp / ReadOnce* trans. + DMT MUST use CompAck.
  - An SN is not required to support the use of CompAck.

Ordering (3/6)

- **Requests and Snps ordering for Separate RD response messages:**
  - The HN, before sending RespSepData to the Rqster, must ensure no Snps are outstanding to that Rqster to the same addr
  - When the Rqster snds CompAck, this Rqster gets the responsibility to hazard snoops for any transaction that is scheduled after it.

Ordering - Transaction level (4/6)

- **2'b00:** *"no order".* Valid pairs: ALL
- **2'b01**:"***Request Accepted***": Completer sends an ACK only when the request is accepted.
  - Valid pairs: HN-F→SN-F. Valid msgs: ReadNoSnp(Sep), ReadOnce*
- **2'b10**:"***Request Order***": multiple trans from the same agent, to the same addr.
- **2'b10**:"***Ordered Write Observation***": observation by other agents of writes issued from a single agent.
- **2'b11**:"***Endpoint Order***": multiple trans from same agent to the same addr range (includes *Request Order*).

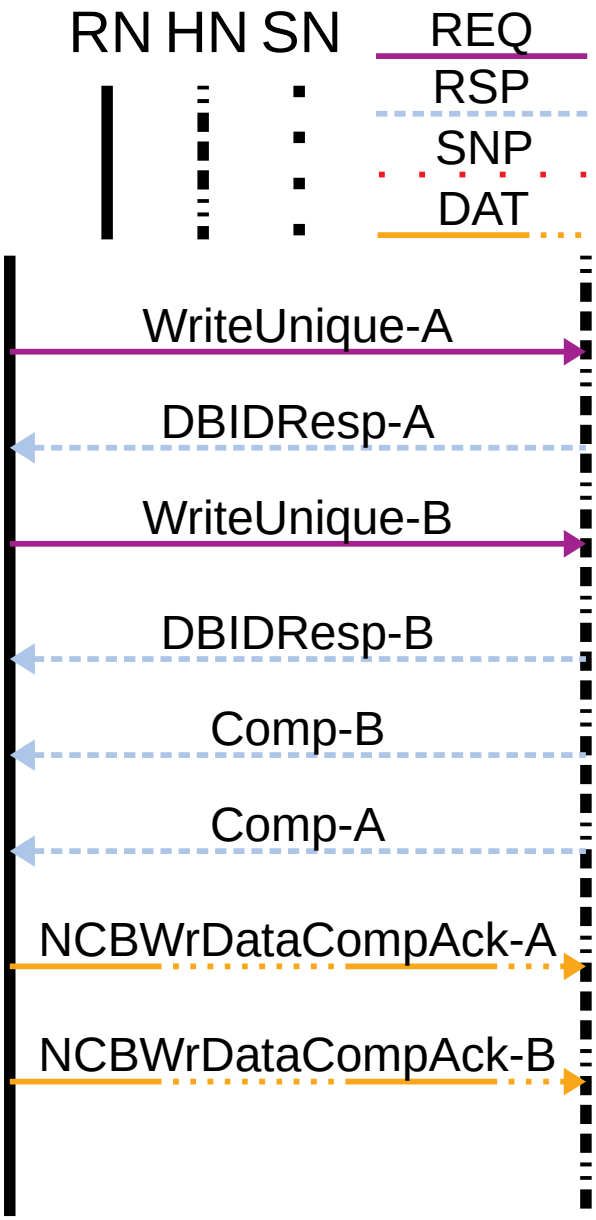
- Valid pairs:
  - RN→HN,
  - HN-I→SN-I
- Valid msgs:
  - ReadNoSnp(Sep)
  - ReadOnce*,
  - WriteNoSnp,
  - WriteUnique,
  - Atomic

Ordering (5/6)

- **Transaction level (continued):**
  - RO/EO under ReadNoSnp,ReadOnce*
    - ReadReceipt is required
  - RO/EO under WriteNoSnp/non-Snoopable Atomic
    - DBIDResp is required
    - write request has reached a PoS
  - RO under WriteUnique & ExpCompAck=1 or Snoopable Atomic
    - DBIDResp is required
  - OWO under WriteUnique
    - DBIDResp & ExpCompAck=1 is required
    - Completer is PoS
  - 2x Transactions from same agent with diff. Order
    - order guarantee the least restrictive of the 2

Ordering (6/6)

- **Transaction level (continued):**
  - RO under CopyBack
    - RN-F must wait for the CompDBIDResp before issuing another outstanding request to the same cache line.
      - an Atomic transaction (SnoopMe=1) can be issued before CompDBID resp. of CPback
  - OWO+CompAck under Write Unique
    - Wait for DBIDResp before sending next req.
    - Further optimization (Do not wait for the DBIDResp, send next req)
      - Devote a single Requestor OR
      - Use WriteDataCancel msgs



Address (Addr) & Non Secure (NS)

- **Address (Addr)**
  - PA: 44b, 45b, 46b .. 52b
  - VA: 49b, 51b, ??b .. 53b
  - Req_Addr_Width param specifies max PA. def val = 44
  - REQ Addr[MPA-1:0] : 52 bits
    - Read,Prefetch,Dataless,Write, Atm
  - SNP Addr[MPA-1:3] : 49 bits
    - Snp (exept for SnpDVMOp)
    - $line: Addr[43-51:6]
    - Critical Chunk Identifier: Addr[5:4]
    - Supplied but not used: Addr[3]
  - DVMOp/SnpDVMOp: Addr field related to DVM op
  - PCrdReturn: Unused, tie to zero
- **Non-Secure (NS) bit**
  - NS=1 non-secure trans (SNP: non-secure addr space)
  - NS=0 secure trans (SNP: secure addr space)
  - Not aplicable in DVMOp or PcrdReturn (tie to zero)

Memory attributes (MemAttr) (1/4)

- **EWA (1b)**
  - If “1”: WR resp comes from intermediate or end point
  - If “0”: WR resp comes from end point
  - Must be “1” for
    - Any Read (except for ReadNoSnp(Sep)=“0/1”)
    - Any dataless
      - exc. for CMO dataless:
        - CleanShared(Persist)
        - [Clean,Make]Invalid =“0/1”
    - Any Write (except for WriteNoSnp=“0/1”)
  - Inapplicable for
    - DVMOp=“0” & PcrdReturn=“0”, PrefetchTgt=“0/1

Memory attributes (MemAttr) (2/4)

- **Device (1b)**
  - Device-type (Device=”1”)
    - ReadNoSnp, WriteNoSnp[Ptl,Full], CleanShared(Persist), [Clean,Make]Invalid, Atomic
    - Not allowed to :
      - return more data than requested
      - Prefetch
      - Write merging
      - Read data from intermediate point
      - Forward read data from write
      - Combine req data from 1> trans
    - Completion (Writes) from an intermediate point must make the write data visible to the endpoint in a timely manner.
  - Normal-type (Device=”0”)
    - Read, Dataless, Write, PrefetchTgt, Atomics
    - Allowed to
      - Prefetch and Write merging
      - Forward read data from write (EWA asserted)

Memory attributes (MemAttr) (3/4)

- **Cacheable (1b)**
  - If “1” perform cache lookup else perform pure access
  - Don’t assert with Device attr = “1”
  - Must be “1” for
    - RDs (exc. for ReadNoSnp(Sep) =”0/1”)
    - dataless (exc. for CMO dataless)
      - CleanShared(Persist)= “0/1”
      - [Clean,Make]Invalid = “0/1”
    - for WRs (exc. for WriteNoSnp[Ptl,Full] =”0/1”)
  - Atomics = “0/1”
  - Inaplicable for DVMOp= “0”, PcrdReturn = “0”
  - Inaplicable for PrefetchTgt=“x”

Memory attributes (MemAttr) (4/4)

- **Allocate (1b)**
  - A hint for $allocation
  - If “1” recommend for $ allocation
  - If “0” recommend for non-$ allocation
  - Can be Asserted with $-attribute asserted
  - Must be Asserted for WriteEvictFull
    - It can be converted to dataless Evict with allocate attribute bit de-asserted
  - Must Not to be asserted when
    - Device attr = 1
    - Device attr = 0 & Cacheable = 0
  - Inaplicable for
    - DVMOp PcrdReturn and Evict (All tied to zero)
    - PrefetchTgt = “0/1”

Propagation of Attr

- EWA, Device, Cacheable, & Allocate, must be preserved on a HN req to SN which happened due to a req to HN unless:
  - the downstream memory is known to be Normal (Device=0)
  - The SnpAttr bit value
    - must be set to “0” .
- When a Prefetch generates ReadNoSnp/WriteNoSnp or an Eviction from System Cache takes place, then:
  - EWA, Cacheable, Allocate must be set to “1”
  - Device must be set to “0”
  - SnpAttr must be set to “0” (non-snoopable)

Trans. Attr combs (1/3)

- Dev/Alloc/$able/EWA - SnpAttr/LS/Order
  - **1/0/0/0 - 0/0/11 : Device nRnE**
    - RD/WR from endpoint,
    - RD/WR should not fetch/write more data than required, no prefetch,
    - no merge,
    - All Rds/Wrs must be in order
  - **1/0/0/1 - 0/0/11 : Device nRE**
    - Same as Device nRnE + WR ack from interm point
  - **1/0/0/1 - 0/0/X0 : Device RE**
    - Same as Device nRE + no order requirement for Rds/Wrs exc. for overlapped addresses
  - **0/0/0/0 - 0/0/X0 : non$able nonBuffable** (AXI mem type)
    - RD/WR from endpoint
    - Allow write merge
    - Order for overlapped RD/WR addresses from same src

Trans. Attr combs (2/3)

- Dev/Alloc/$able/EWA - SnpAttr/LS/Order
  - **0/0/0/1 - 0/0/X0 : non$able Buffable**
    - WR ack from interm point
    - Timely mannered WR visibility at endpoint
    - RD from endpoint ***or*** Forward read data from WR (most recent un$'ed version)
    - Allow write merge
    - Order for overlapped RD/WR addresses from same src
  - **0/0/1/1 - 0/0/X0 : nonSnpable WB noAlloc**
    - WR ack from interm point
    - Allow WR merge
    - Order for overlapped RD/WR addresses from same src
    - no WR visibility at endpoint
    - RD from $'ed copy
    - RD prefetch
    - $lookup for every RD/WR

Trans. Attr combs (3/3)

- Dev/Alloc/$able/EWA - SnpAttr/LS/Order
  - **0/1/1/1 - 0/0/X0 : nonSnpable WB Alloc**
    - Simlar to **nonSnpable WB noAlloc** except the fact that the alloc hint is passed to mem system as a hint for perf.
  - **0/0/1/1 - 1/X/X0 : Snpable WB noAlloc**
    - Similar to **WB noAlloc** but snoopable
  - **0/1/1/1 - 1/X/X0 : Snpable WB Alloc**
    - Similar to **WB Alloc** but snoopable

Likely Shared (LS) attribute

- Pass a hint directive for shared system caches
  - Valid msgs <u>ONLY</u>:
    - ReadShared
    - ReadNotSharedDirty
    - ReadClean
    - WriteBackFull
    - WriteCleanFull
    - WriteEvictFull
    - WriteUnique[[Ptl,Full,PtlStash,FullStash]]
    - StashOnceUnique
    - StashOnceShared
  - Not valid
    - For any other RD/WR msg
    - For any dataless/atomic msg
    - DVMOp PcrdReturn (tie to zero)
    - PrefetchTgt (tie to any value)

Snoop attribute (SnpAttr)

- If asserted the trans. Requires snooping
  - De-asserted for:
    - ReadNoSnp(Sep)
    - WriteNoSnp
    - DVMOp
    - Prefetch (any val)
    - Clean[Shared(Persist),Invalid], MakeInvalid (HN→SN)
  - Asserted for:
    - ReadOnce*,Read[Clean,Shared,NotSharedDirty,Unique]
    - CleanUnique,MakeUnique,StashOnce
    - Evict
    - Write[Back,Clean,EvictFull]
    - WriteUnique
  - Either Asserted or Not
    - Clean[Shared(Persist),Invalid], MakeInvalid (RN→HN)
    - Atomic

DoNotGoToSD attr

- For non-invalidating snoops
- Snoopie should not transition to SD
- Valid for :
  - SnpClean(Fwd)
  - SnpNotSharedDirty(Fwd)
  - SnpShared(Fwd)
- Always asserted (Tie to “1”)
  - SnpUnique(Fwd)
  - SnpClean[Shared,Invalid]
  - SnpMakeInvalid
- Don’t care value:
  - SnpOnce(Fwd)

Missmatched $able/SnpAble atributes to the same region

- SW protocol error
  - SW $ maintenance returns mem location to a defined state
- Undefined relationship between RN request and Snoop. For i.e. :
  - ReadNoSnp/WriteNoSnp VS. a Snoop performing at the same loc.

Data transfer (Data payload)

- Valid for RD/WR/Atomic and Snp Responses
- Size[2:0]: Fixed for Snps: 64B, others: upto 64B
  - Size=0 (1B). Size=1 (2B)...Size=6 (64B)
- Byte access (mem VS. device - MemAttr[1])
  - Mem
    - AlignedAddr to (AlignedAddr+ 2^Size -1)
      - AlignedAddr=RoundDwn(Addr/2^Size) * 2^Size
  - Dev
    - Addr to (AlignedAddr+2^Size) -1

Byte Enable (BE)

- Valid for WR,SnpResp with Data and Atomics
- If BE zero, corresponding Data Byte must be zero
- BE should be deassrted beyond the data window (specified by Addr,Size) for all msgs
- WriteDataCancel msgs should have BE=zero
- The msgs which allow a cancel between a wr req and sending data are (pg. 4-166):
  - WriteUnique[Ptl, PtlStash], WriteNoSnpPtl
- Any BE comb is allowed for
  - Any *Ptl* WR msgs (4 in total), SnpRespDataPtl
- All BE=”1”or All BE=”0” (chk spec confict 4-116 vs. 4-166):
  - Any *Full* WR msgs (4 in total)
- All BE=”1”
  - SnpRespData

Data packet (DataID & CCID)

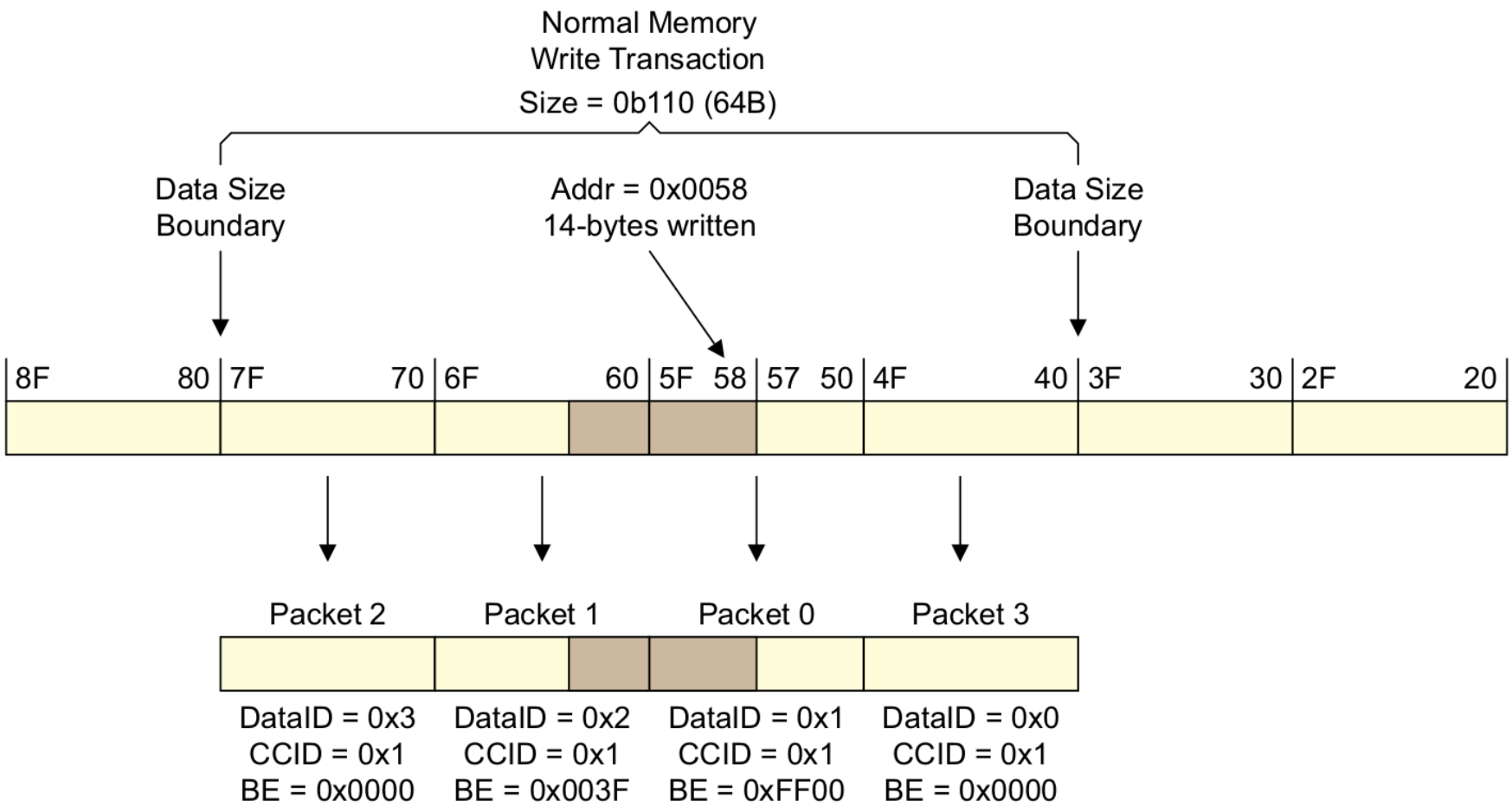
- Packet = Flit = Data Bus Width
- Allowed Data Bus Widths: 128b/256b/512b
- DataID[1:0] field: specifies the flit Addr[5:4]
  - If flit width =128b: DataID = [0..3]
  - If flit width =256b: DataID = 0,2
  - If flit width =512b: DataID = 0
- CCID[1:0]
  - Represents which flit is critical
  - Valid only for 128b or 256b flits
    - 128b case: critical data when DataID=CCID
    - 256b case: critical data when DataID[1]=CCID[1]
- Critical chunk first wrap order
  - CCF_Wrap_Order param= TRUE/F at sender / ICN / Receiver
- Data Beat Ordering
  - Check pg. 2-121

Size,Addr & Data in atomics

- Endianess field is valid for atomic msgs only
- Size
  - AtomicStore: 1/2/4/8bytes (out-only)
  - Atomic[Load,Swap]: 1/2/4/8bytes (in/out)
  - AtomicCompare: 2/4/8/16/32 bytes (out: ½ compare ½ swap, in: ½ swap)
    - check Figure 2-36 pg. 2-119

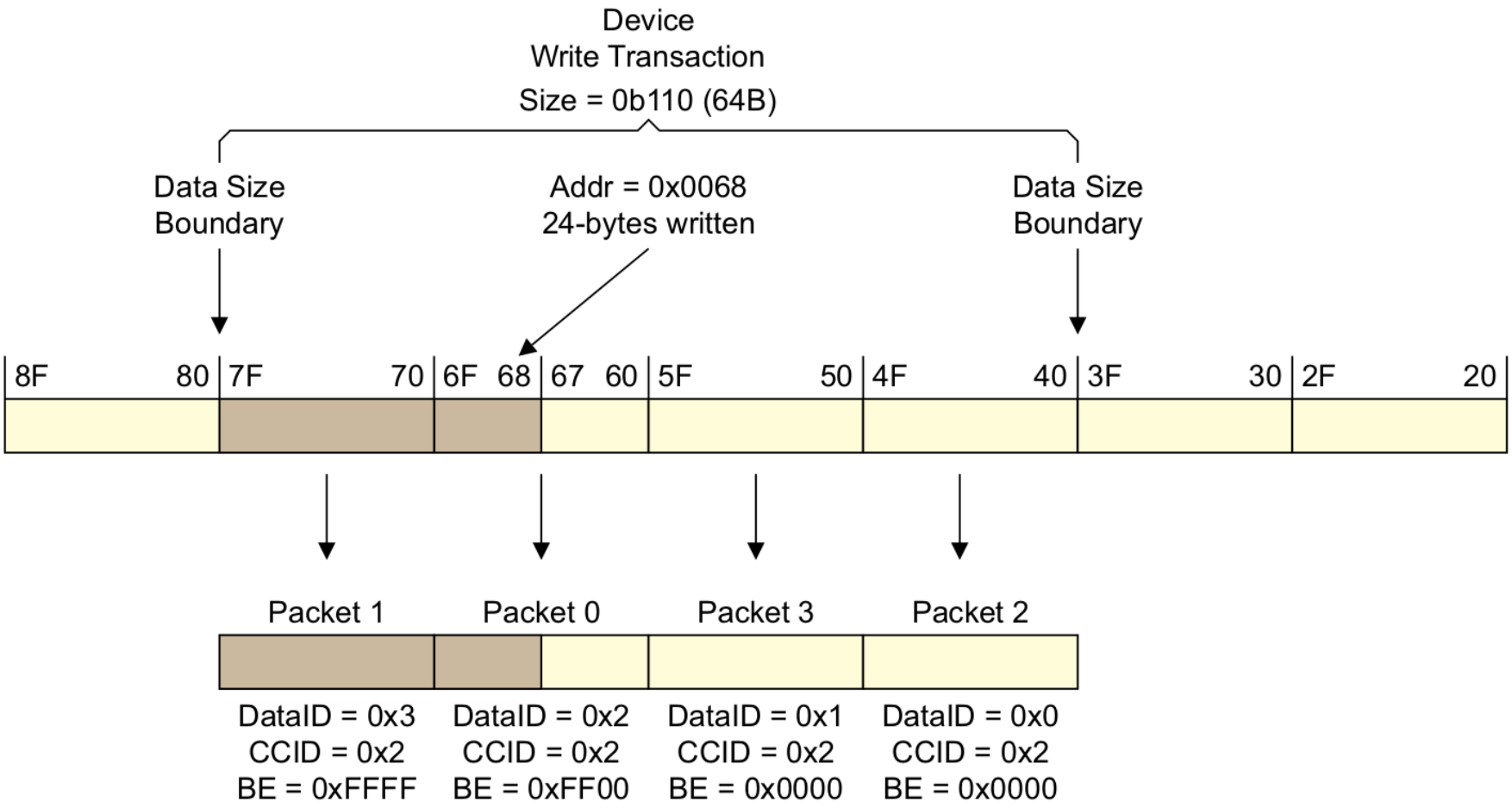
Data transfer example

Example 2-4 Normal memory 14-byte consecutive write transaction from an unaligned address



Data transfer example

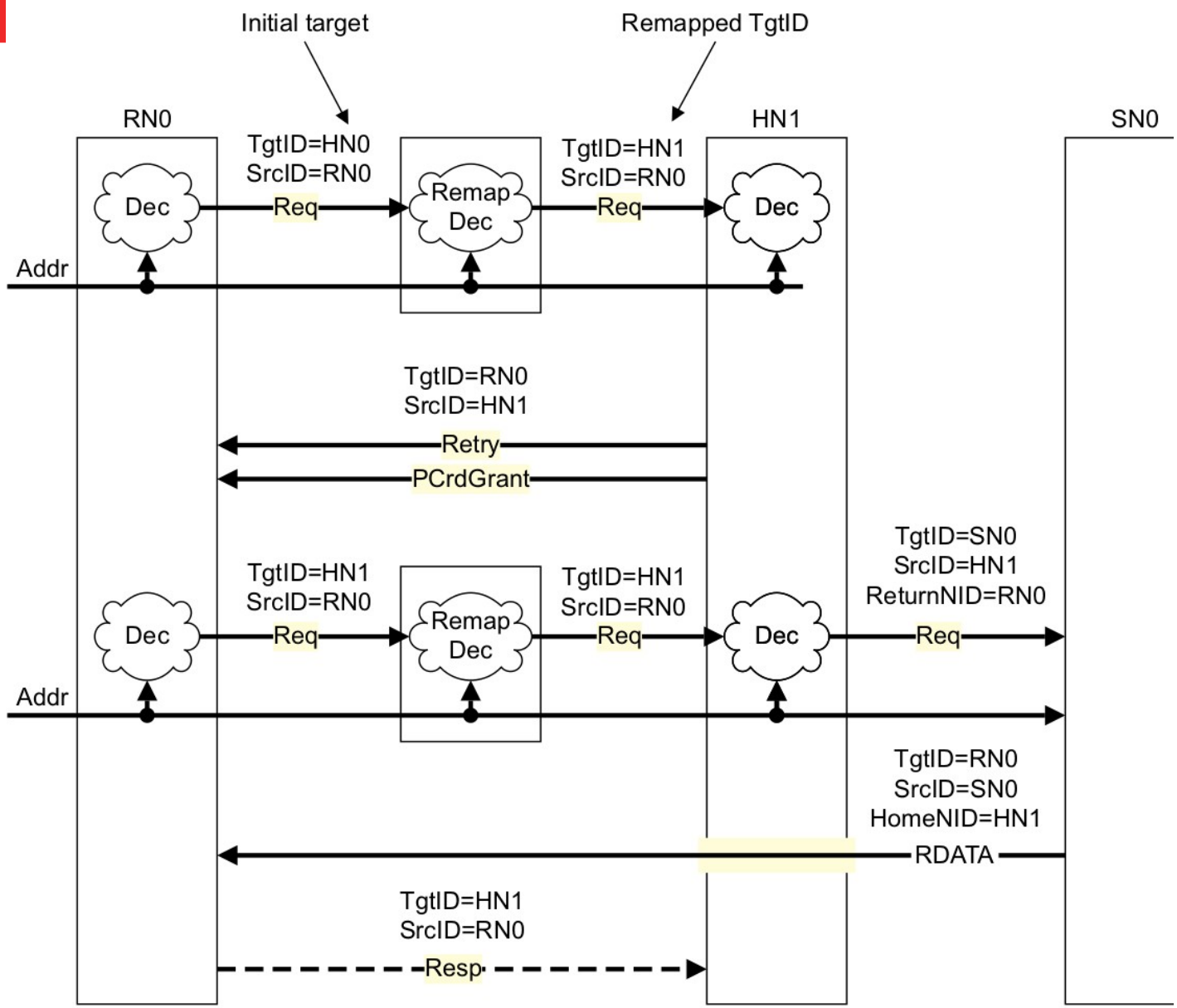
Example 2-6 Device write transaction to an unaligned address



Retry Scheme

- AllowRetry must be asserted the first time (REQ) and PcrdType filed should be tied to zero
  - If RetryAck msg is sent (RSP) back, the requester must wait for PcrdGrant with credit. Then he can
    - replay the original transaction w/ credit or
    - abort trans sending PCrdReturn msg w/ credit
  - It may happen PcrdGrant arrive before RetryAck
  - Credit = PCrdType field
- If RetryAck has been received from the requestor, the next req must have the AllowRetry de-asserted and the PCrdType field value equal to the RetryAck msg's PCrdType field value

Network Layer



SAM:
Map Addr to ID

An RN or HN should have a SAM

Any unmapped region should be sent to an err slave (send back err resp.)

Coherence Protocol (C.4)

- CHI $line steady states
  - **I (Invalid)**
  - **UC (Unique Clean)**
  - UCE (Unique Clean Empty)
  - **UD (Unique Dirty)**
  - UDP (Unique Dirty Partial)
  - **SC (Shared Clean)**
  - SD (Shared Dirty)

Coherence Protocol (C.4)

- TBA

Trans. Flows (C.5)

- TBA

X acceses & monitors (1/3)

- A seq. of LDX trans., COMP, STX trans. to snp'd/nonSnp'd addr
  - Pass(RespErr =2b01) or fails (RespErr =2b00). Starts with LDX instr. ends with STX instr. Not every LDX/STX requires a LDX/STX trans.
- Snp range (Read[Clean,NotSharedDirty,Shared], CleanUnique)
  - LP monitor: RN-F side
    - Set by LDX
    - Reset by a rcv'ed invldt snp OR same LP (i.e. a ST) impl def.
  - PoC monitor: HN-F side
    - Parallel monitoring of all LPs, Registers any X seq.
    - Responses accordingly to X stores
  - An LP can restart X seq. after a CompAck is rcv'ed
- Non-Snp range (ReadNoSnp,WriteNoSnp):
  - System Monitor:PoS side or Endpoint side
- Address matching can be used also for registering an X seq.
- Decouple secure and non-secure reqs. with X seq

X acceses & monitors (2/3)

- X load on $'d loc (Unique,Shared)
  - Not required to perform X load trans.
- X load on un$'d loc
  - X load trans. is required, but a Read[Clean, Shared, NotSharedDirty] can be issued w/o excl bit asserted
- An X seq can be aborted if desired w/o issuing STX
- An LP must wait any current X seq before issuing another STX
- X store on $'d loc
  - Unique: (if LP monitor is set no need to make STX trans)
  - Shared: make a CleanUnique STX trans
    - Case pass resp from HN:
      - LP mon set –> X seq pass
      - LP mon reseted (due to $upd/$evict) –> X seq restart
    - Case fail resp from HN:
      - check LP mon (reset) → restart X seq.
      - Check LP mon (if set) → reissue X store trans. only
      - Do not check LP mon → restart X seq. anyway

X acceses & monitors (3/3)

- WriteNoSnp/ReadNoSnp
  - A pair of the affordmentioned trans. Should have identical fields of:
    - Addr, MemAtrr,SnpAttr, size, LPID
  - Outstanding RD/WR trans. Are not allowed even for diff. Addresses
  - An SN which does not support X accesses:
    - If ReadNoSnp returns Xfail then
      - the WR will update val if WriteNoSnp returns Xfail
    - If ReadNoSnp returns Xpass then
      - the WR will not update val if WriteNoSnp returns Xfail

$ Stashing (C.7)

- TBA

DVM for Virtual mem protocol management (C.8)

- TBA

Error Handling (C.9)

- TBA

QoS (C.10)

- TBA

Data Source & Trace Tag (C.11)

- TBA

Link Handshake (C.13)

- TBA

System Coherency Interface (C.14)

- TBA

Properties, Parameters & Broadcast signals (C.15)

- TBA

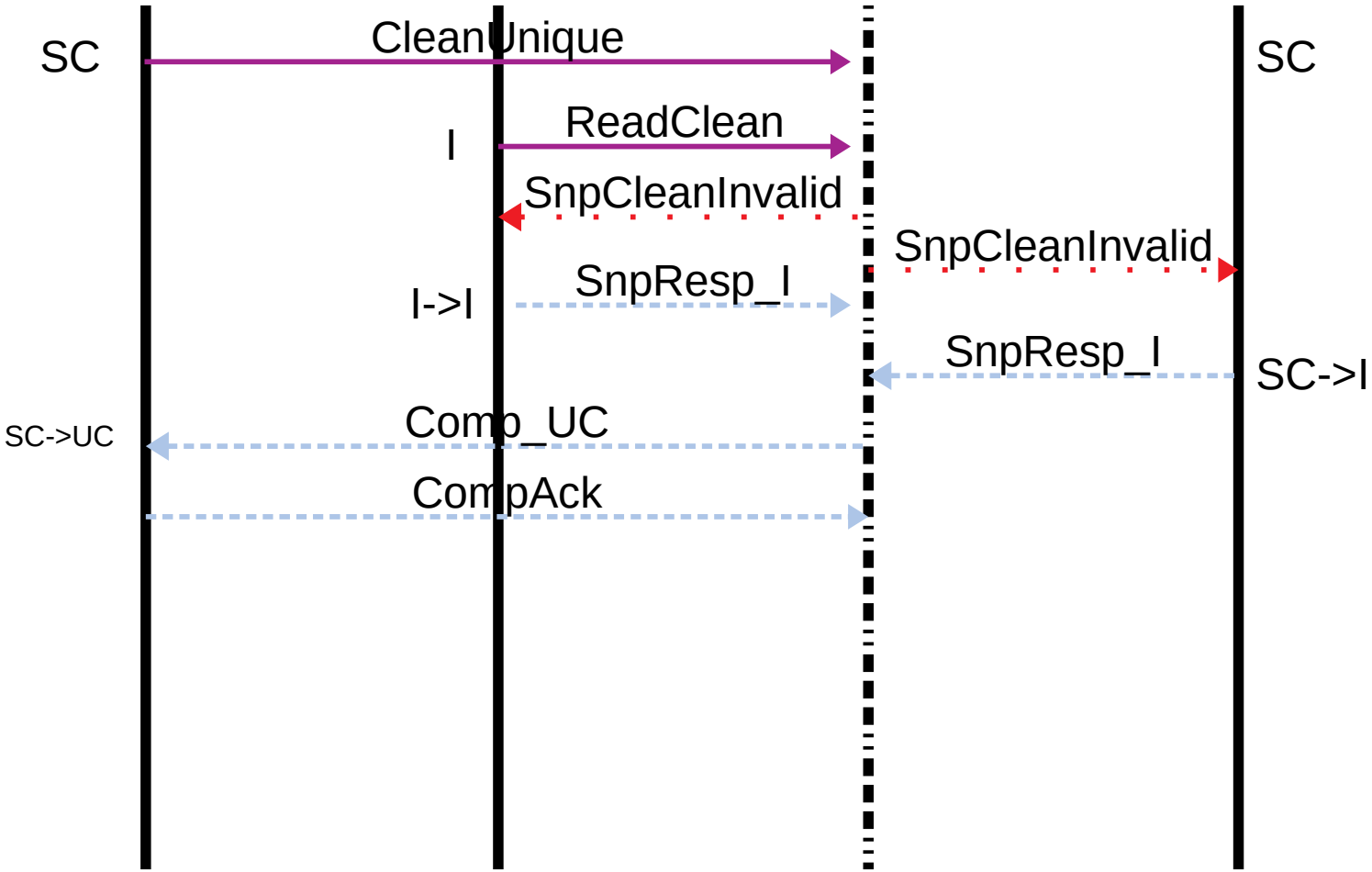
Message Field Mappings (C.A)

- TBA

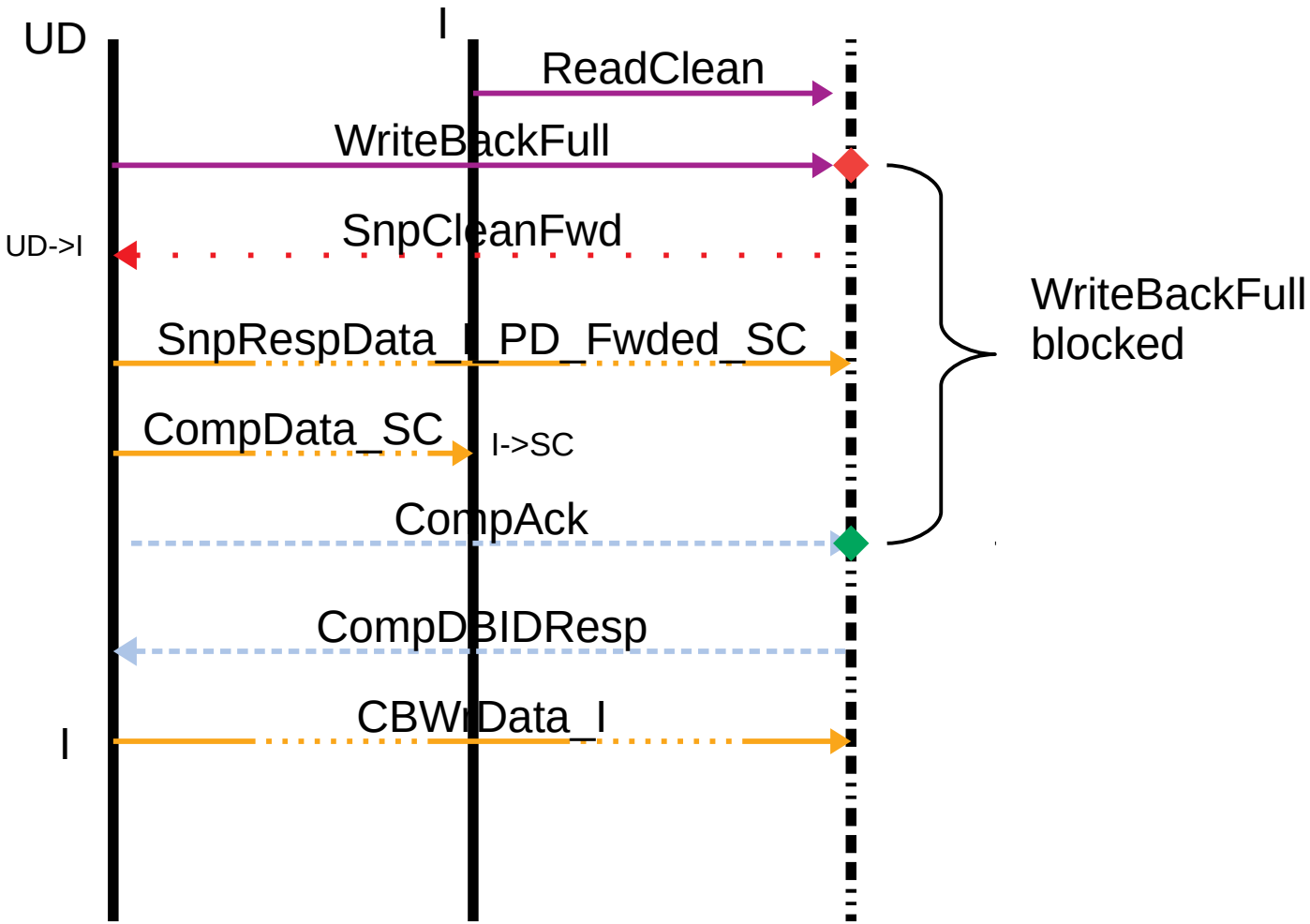
Communicating Nodes (C.B)

- TBA

Hazard #1


RN HN SN
REQ
RSP
SNP
DAT
SC
CleanUnique
SC
I
ReadClean
SnpCleanInvalid
SnpCleanInvalid
I->I
SnpResp_I
SnpResp_I
SC->I
SC->UC
Comp_UC
CompAck

Hazard #2


RN HN SN
REQ
RSP
SNP
DAT
UD
I
ReadClean
WriteBackFull
UD->I
SnpCleanFwd
SnpRespData_I_PD_Fwded_SC
CompData_SC
I->SC
CompAck
WriteBackFull
blocked
CompDBIDResp
CBWrData_I
I