```
──────────────── MODULE OneUpdate ────────────────
EXTENDS OneUpdateMeta
──────────────────────────────────────────────────
 Helpers
```
$dir\_has\_action\_pending \triangleq dReqPending = 1$
$dir\_set\_action\_pending \triangleq dReqPending' = 1$
$dir\_rst\_action\_pending \triangleq dReqPending' = 0$

$upd\_dir\_state(s) \triangleq dState' = s$
$upd\_state(n, s) \triangleq cState' = [cState \text{ EXCEPT } ![n] = s]$

$rmv\_sharer(s) \triangleq dSharers' = dSharers \setminus \{s\}$
$add\_sharer(s) \triangleq dSharers' = dSharers \cup \{s\}$

```
upd_owner(o) ≜
      ∧ dOwner'   = o
      ∧ dSharers' = {o}

rmv_owner(o) ≜
      ∧ rmv_sharer(o)
      ∧ dOwner' = EMPTY_OWNER

rst_acks(n) ≜
      cRcvAcks' = [cRcvAcks EXCEPT ![n] = {}]
add_ack(n, m) ≜
      cRcvAcks' = [cRcvAcks EXCEPT ![n] = cRcvAcks[n] ∪ {m.sender}]

rst_dir_acks ≜
      dRcvAcks' = {}
add_dir_ack(m) ≜
      dRcvAcks' = dRcvAcks ∪ {m.sender}

rcv_upd_ack_msg(n, m) ≜
      ∧ m.receiver = n
      ∧ m.type = "UAck"

rcv_ack_msg(n, m) ≜
      ∧ m.receiver = n
      ∧ ∨ m.type = "SAck"
        ∨ m.type = "SDataAck"

_is_last_Ack_from_set(n, m, set) ≜
      set ⊆ (cRcvAcks[n] ∪ {m.sender})
```

```
$is\_last\_Ack(n, m) \triangleq$
    $\land\ rcv\_ack\_msg(n, m)$
    $\land\ \_is\_last\_Ack\_from\_set(n, m, dSharers \setminus \{n\})$

$is\_last\_upd\_Ack(n, m) \triangleq$
    $\land\ rcv\_upd\_ack\_msg(n, m)$
    $\land\ \_is\_last\_Ack\_from\_set(n, m, CORES \setminus \{n\})$

$is\_last\_dir\_Ack(m) \triangleq$
    $\land\ m.type = \text{“UAck”}$
    $\land\ dSharers \subseteq (dRcvAcks \cup \{m.sender\})$

$owner\_or\_min\_sharer \triangleq$
   IF $dOwner \neq EMPTY\_OWNER$
    THEN $dOwner$
    ELSE $Min(dSharers)$

$sharers\_no\_fwd \triangleq dSharers \setminus \{owner\_or\_min\_sharer\}$
```

Requests involving only Directory

Local write hit

```
$EtoM(n) \triangleq$   $E$ to $M$
      $\land\ cState[n] = \text{“E”}$
      $\land\ upd\_owner(n)$
      $\land\ upd\_state(n, \text{“M”})$
      $\land\ upd\_dir\_state(\text{“M”})$
      $\land\ unchanged\_gmMsgs$
      $\land$ UNCHANGED $\langle dReqPending, cData, cRcvAcks, dRcvAcks\rangle$
```

Eviction

```
$PutE(n) \triangleq$   $E$ to $I$
      $\land\ cState[n] = \text{“E”}$
      $\land\ rmv\_owner(n)$
      $\land\ upd\_state(n, \text{“I”})$
      $\land\ upd\_dir\_state(\text{“I”})$
      $\land\ unchanged\_gmMsgs$
      $\land$ UNCHANGED $\langle dReqPending, cData, cRcvAcks, dRcvAcks\rangle$

$PutM(n) \triangleq$   $M$ to $I$
      $\land\ cState[n] = \text{“M”}$
      $\land\ rmv\_owner(n)$
      $\land\ upd\_mem\_data(n)$
      $\land\ upd\_state(n, \text{“I”})$
      $\land\ upd\_dir\_state(\text{“I”})$
```

```
    $\land$ $unchanged\_gMsgs$
    $\land$ UNCHANGED $\langle dReqPending, cData, cRcvAcks, dRcvAcks \rangle$
```

```
$PutS(n) \triangleq$   S to I/S
    $\land$ $cState[n] = \text{"S"}$
    $\land$ $rmv\_sharer(n)$
    $\land$ $upd\_state(n, \text{"I"})$
    $\land$ IF $Cardinality(dSharers) = 1$
        THEN $upd\_dir\_state(\text{"I"})$
        ELSE $upd\_dir\_state(\text{"S"})$
    $\land$ $unchanged\_gmMsgs$
    $\land$ UNCHANGED $\langle dOwner, dReqPending, cData, cRcvAcks, dRcvAcks \rangle$
```

```
$PutO(n) \triangleq$   O to I/S
    $\land$ $cState[n] = \text{"O"}$
    $\land$ $rmv\_owner(n)$
    $\land$ $upd\_mem\_data(n)$
    $\land$ $upd\_state(n, \text{"I"})$
    $\land$ IF $Cardinality(dSharers) = 1$
        THEN $upd\_dir\_state(\text{"I"})$
        ELSE $upd\_dir\_state(\text{"S"})$
    $\land$ $unchanged\_gMsgs$
    $\land$ UNCHANGED $\langle dReqPending, cData, cRcvAcks, dRcvAcks \rangle$
```

Cache miss (fetching from memory)

```
$GetS\_dI(n) \triangleq$   I to E
    $\land$  $dState = \text{"I"}$
    $\land$  $cState[n] = \text{"I"}$
    $\land$  $add\_sharer(n)$
    $\land$  $rd\_mem\_data(n)$
    $\land$  $upd\_state(n, \text{"E"})$
    $\land$  $upd\_dir\_state(\text{"E"})$
    $\land$  $unchanged\_gmMsgs$
    $\land$  UNCHANGED $\langle dOwner, dReqPending, cRcvAcks, dRcvAcks \rangle$
```

```
$GetM\_dI(n) \triangleq$   I to M
    $\land$  $dState = \text{"I"}$
    $\land$  $cState[n] = \text{"I"}$
    $\land$  $upd\_owner(n)$
    $\land$  $rd\_mem\_data(n)$
    $\land$  $upd\_state(n, \text{"M"})$
    $\land$  $upd\_dir\_state(\text{"M"})$
    $\land$  $unchanged\_gmMsgs$
    $\land$  UNCHANGED $\langle dReqPending, cRcvAcks, dRcvAcks \rangle$
```

```
Dir
$GetS\_Fwd(n) \triangleq$
    $\land\ dState \neq$ "I"
    $\land\ cState[n] =$ "I"
    $\land\ dir\_set\_action\_pending$
    $\land\ ucst\_FwdGetS(n,\ owner\_or\_min\_sharer)$
    $\land$ IF $(dState =$ "E" $\lor\ dState =$ "S"$)$
        THEN $\land\ upd\_dir\_state($"S"$)$
        ELSE $upd\_dir\_state($"O"$)$
    $\land\ unchanged\_gmc$
    $\land$ UNCHANGED $\langle dOwner,\ dSharers,\ dRcvAcks\rangle$

$GetS(n) \triangleq$
      $\lor\ GetS\_dI(n)$
      $\lor\ GetS\_Fwd(n)$

Sharers
$RcvFwdGetS(n,\ m) \triangleq$
    $\land\ rcv\_FwdGetS(m,\ n)$
    $\land\ resp\_SData(m)$
    $\land$ IF $(cState[n] =$ "E" $\lor\ cState[n] =$ "S"$)$
        THEN $upd\_state(n,$ "S"$)$
        ELSE $upd\_state(n,$ "O"$)$
    $\land\ unchanged\_gmd$
    $\land$ UNCHANGED $\langle cData,\ cRcvAcks,\ dRcvAcks\rangle$

Requester
$RcvData(n,\ m) \triangleq$
    $\land$   $rcv\_SData(m,\ n)$
    $\land$   $deliver\_Msg(m)$
    $\land$   $add\_sharer(n)$
    $\land$   $upd\_state(n,$ "S"$)$
    $\land$   $upd\_core\_data(n,\ m.data)$
    $\land$   $dir\_rst\_action\_pending$
    $\land$   $unchanged\_gm$
    $\land$   UNCHANGED $\langle dOwner,\ dState,\ cRcvAcks,\ dRcvAcks\rangle$
```

```
Dir
$GetM\_Invs(n) \triangleq$
    $\land\ dState \neq$ "I"
    $\land\ cState[n] \neq$ "M"
    $\land\ cState[n] \neq$ "E"
    $\land\ Cardinality(dSharers \setminus \{n\}) > 0$
    $\land\ rst\_acks(n)$
    $\land$ $dir\_set\_action\_pending$
    $\land$ $upd\_dir\_state(\text{"M"})$
    $\land$ $unchanged\_m$
    $\land$ UNCHANGED $\langle dOwner, dSharers, cState, cData, dRcvAcks \rangle$
    $\land$ IF $(dState = \text{"E"} \lor dState = \text{"M"})$
       THEN $\land$ $ucst\_FwdGetM(n, owner\_or\_min\_sharer)$        single remote owner case
            $\land$ $unchanged\_g$
       ELSE IF $(dState = \text{"S"} \lor dOwner = n)$
               THEN $\land$ $bcst\_DInv(n, dSharers \setminus \{n\})$ is owner but w/ sharers
                    $\land$ $unchanged\_Msgs$
               ELSE $\land$ $ucst\_FwdGetM(n, owner\_or\_min\_sharer)$ (remote) owner and sharers
                    $\land$ IF $Cardinality(dSharers \setminus \{owner\_or\_min\_sharer, n\}) > 0$
                       THEN $bcst\_DInv(n, dSharers \setminus \{owner\_or\_min\_sharer, n\})$
                       ELSE $unchanged\_g$

$GetM(n) \triangleq$
      $\lor$ $EtoM(n)$
      $\lor$ $GetM\_dI(n)$
      $\lor$ $GetM\_Invs(n)$


Sharers $\rightarrow$ $rcvInv$ or $FwdGetM$
$RcvInv(n, m) \triangleq$
    $\land$ $(rcv\_DInv(m, n) \lor rcv\_FwdGetM(m, n))$
    $\land$ $upd\_state(n, \text{"I"})$
    $\land$ IF $rcv\_DInv(m, n)$
          THEN $resp\_SAck(m)$
          ELSE $resp\_SDataAck(m)$
    $\land$ $unchanged\_gmd$
    $\land$ UNCHANGED $\langle cData, cRcvAcks, dRcvAcks \rangle$

Requester $\rightarrow$ normal $Ack$ or $DataAck$
$RcvAck(n, m) \triangleq$
    $\land$ $rcv\_ack\_msg(n, m)$
    $\land$ $deliver\_Msg(m)$
    $\land$ $unchanged\_gm$
    $\land$ UNCHANGED $\langle dState, dRcvAcks \rangle$
    $\land$ IF $rcv\_SDataAck(m, n)$
       THEN $upd\_core\_data(n, m.data)$
       ELSE UNCHANGED $\langle cData \rangle$
    $\land$ IF $\neg is\_last\_Ack(n, m)$
       THEN $\land$ $add\_ack(n, m)$
            $\land$ $unchanged\_d$
            $\land$ UNCHANGED $\langle cState \rangle$
       ELSE $\land$ $rst\_acks(n)$
            $\land$ $upd\_owner(n)$
```

```
        $\land$ $upd\_state(n,$ "M"$)$
        $\land$ $dir\_rst\_action\_pending$
```

Dir
SharedUpdate

predicate
For simplicity now we always make every core a sharer here

```
$MtoO(n) \triangleq$
    $\land$ $dir\_set\_action\_pending$
    $\land$ $bcst\_Upd(n,\ CORES \setminus \{n\},\ cData[n])$
    $\land$ $unchanged\_mMsgs$
    $\land$ IF $ENABLE\_DIR\_ACKS$
         THEN $\land$ $upd\_state(n,$ "O"$)$  update eagerly to $O$ state if dir collects $ACKs$
              $\land$ $dRcvAcks = \{n\}$  add the requester to rcved acks for easier check of all acks predicate
         ELSE $\land$ $rst\_acks(n)$
              $\land$ UNCHANGED $\langle cState,\ dRcvAcks \rangle$
    $\land$ UNCHANGED $\langle cData,\ dOwner,\ dSharers,\ dState \rangle$

$RcvUpd(n,\ m) \triangleq$
    $\land$ $rcv\_Upd(m,\ n)$
    $\land$ $resp\_UAck(m)$
    $\land$ $upd\_state(n,$ "S"$)$
    $\land$ $upd\_core\_data(n,\ m.data)$
    $\land$ $unchanged\_gmd$
    $\land$ UNCHANGED $\langle cRcvAcks,\ dRcvAcks \rangle$

$RcvUpdAck(n,\ m) \triangleq$
    $\land$ $\neg ENABLE\_DIR\_ACKS$
    $\land$ $cState[n] =$ "M"
    $\land$ $rcv\_upd\_ack\_msg(n,\ m)$
    $\land$ $deliver\_Msg(m)$
    $\land$ $unchanged\_gm$
    $\land$ UNCHANGED $\langle cData,\ dRcvAcks \rangle$
    $\land$ IF $\neg is\_last\_upd\_Ack(n,\ m)$
         THEN $\land$ $add\_ack(n,\ m)$
              $\land$ $unchanged\_d$
              $\land$ UNCHANGED $\langle cState \rangle$
         ELSE $\land$ $rst\_acks(n)$
              $\land$ $upd\_state(n,$ "O"$)$
              $\land$ $dState'$   $=$ "O"
              $\land$ $dOwner'$   $= n$
              $\land$ $dSharers'$ $= CORES$
              $\land$ $dir\_rst\_action\_pending$
```

```
$DirRcvUpdAck(m) \triangleq$
    $\land$ $ENABLE\_DIR\_ACKS$
    $\land$ $dir\_has\_action\_pending$
    $\land$ $dState = \text{"M"}$
    $\land$ $m.type = \text{"UAck"}$
    $\land$ $deliver\_Msg(m)$
    $\land$ $unchanged\_gmc$
    $\land$ UNCHANGED $\langle dOwner \rangle$
    $\land$ IF $\neg is\_last\_dir\_Ack(m)$
        THEN $\land$ $add\_dir\_ack(m)$
             $\land$ UNCHANGED $\langle dSharers, dReqPending, dState \rangle$
        ELSE $\land$ $rst\_dir\_acks$
             $\land$ $dState' = \text{"O"}$
             $\land$ $dSharers' = CORES$
             $\land$ $dir\_rst\_action\_pending$
```

```
$must\_update(n) \triangleq$
    $\land$ $cState[n] = \text{"M"}$
    $\land$ $cData[n] = WRITE\_TO\_UPDATE$

$Requests(n) \triangleq$
    $\land$ $\neg dir\_has\_action\_pending$
    $\land$ IF $must\_update(n)$
        THEN $MtoO(n)$
        ELSE $\lor$ $GetM(n)$
             $\lor$ $GetS(n)$
             $\lor$ $PutE(n)$
             $\lor$ $PutM(n)$
             $\lor$ $PutS(n)$
             $\lor$ $PutO(n)$

$SharerActions(n, m) \triangleq$
    $\lor$ $RcvUpd(n, m)$
    $\lor$ $RcvInv(n, m)$
    $\lor$ $RcvFwdGetS(n, m)$

$RequesterActions(n, m) \triangleq$
    $\lor$ $RcvAck(n, m)$
    $\lor$ $RcvData(n, m)$
    $\lor$ $RcvUpdAck(n, m)$

$DirActions(m) \triangleq DirRcvUpdAck(m)$

$MessageActions(n) \triangleq$
    $\exists m \in Msgs :$
```

```
        $\lor\ DirActions(m)$
        $\lor\ SharerActions(n, m)$
        $\lor\ RequesterActions(n, m)$

$PerformBcast \triangleq$
        $\land\ gBcstMsg \neq \{\}$
        $\land\ \exists\, m \in gBcstMsg :$
            $\land\ \_send\_Msg(m)$
            $\land\ unchanged\_mcd$
            $\land$ UNCHANGED $\langle dRcvAcks \rangle$
            $\land$ IF $gBcstMsgRcvers = \{\}$
                THEN $\land\ gBcstMsg' = \{\}$
                     $\land$ UNCHANGED $\langle gBcstMsgRcvers \rangle$
                ELSE LET $rcver \triangleq$ CHOOSE $x \in gBcstMsgRcvers :$ TRUEIN
                     $\land\ gBcstMsg' = \{[m$ EXCEPT $!.receiver = rcver]\}$
                     $\land\ gBcstMsgRcvers' = gBcstMsgRcvers \setminus \{rcver\}$
```

```
$WriteData(n) \triangleq$
    $\land\ cState[n] =$ "M"
    $\land\ cData[n] < MAX\_WRITES$
    $\land\ \neg must\_update(n)$
    $\land\ cData' = [cData$ EXCEPT $![n] = cData[n] + 1]$
    $\land\ unchanged\_gdmMsgs$
    $\land$ UNCHANGED $\langle cState, cRcvAcks, dRcvAcks \rangle$
```

Modeling 1-Update protocol (Directory, memory and core/cache actions)

```
$ANext \triangleq$
        IF $gBcstMsg \neq \{\}$
         THEN $PerformBcast$
         ELSE $\exists\, n \in CORES :$
                $\lor\ Requests(n)$
                $\lor\ WriteData(n)$
                $\lor\ MessageActions(n)$
```

The complete definition of the algorithm

$Spec \triangleq AInit \land \Box[ANext]_{vars}$

$Invariants \triangleq \land (\Box ATypeOK) \land (\Box INVARIANTS)$

THEOREM $Spec \Rightarrow Invariants$
