MODULE *OneUpdate*

EXTENDS *OneUpdateMeta*

Helpers

$dir\_has\_action\_pending \triangleq dReqPending = 1$
$dir\_set\_action\_pending \triangleq dReqPending' = 1$
$dir\_rst\_action\_pending \triangleq dReqPending' = 0$

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

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

$upd\_owner(o) \triangleq$
$\quad \land dOwner' = o$
$\quad \land dSharers' = \{o\}$

$rmv\_owner(o) \triangleq$
$\quad \land rmv\_sharer(o)$
$\quad \land dOwner' = EMPTY\_OWNER$

$rst\_acks(n) \triangleq$
$\quad cRcvAcks' = [cRcvAcks \text{ EXCEPT } ![n] = \{\}]$
$add\_ack(n, m) \triangleq$
$\quad cRcvAcks' = [cRcvAcks \text{ EXCEPT } ![n] = cRcvAcks[n] \cup \{m.sender\}]$

$rcv\_upd\_ack\_msg(n, m) \triangleq$
$\quad \land m.receiver = n$
$\quad \land m.type = \text{“UAck”}$

$rcv\_ack\_msg(n, m) \triangleq$
$\quad \land m.receiver = n$
$\quad \land \lor m.type = \text{“SAck”}$
$\quad\quad \lor m.type = \text{“SDataAck”}$

$\_is\_last\_Ack\_from\_set(n, m, set) \triangleq$
$\quad set \subseteq (cRcvAcks[n] \cup \{m.sender\})$

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

$is\_last\_upd\_Ack(n, m) \triangleq$
$\quad \land rcv\_upd\_ack\_msg(n, m)$

$\land$ *_is_last_Ack_from_set*$(n,\ m,\ CORES \setminus \{n\})$

$owner\_or\_min\_sharer \triangleq$
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*
$\land\ cState[n] =$ "E"
$\land\ upd\_owner(n)$
$\land\ upd\_state(n,$ "M")
$\land\ upd\_dir\_state($"M")
$\land\ unchanged\_gmMsgs$
$\land$ UNCHANGED $\langle dReqPending,\ cData,\ cRcvAcks\rangle$

Eviction
$PutE(n) \triangleq$ *E to I*
$\land\ cState[n] =$ "E"
$\land\ rmv\_owner(n)$
$\land\ upd\_state(n,$ "I")
$\land\ upd\_dir\_state($"I")
$\land\ unchanged\_gmMsgs$
$\land$ UNCHANGED $\langle dReqPending,\ cData,\ cRcvAcks\rangle$

$PutM(n) \triangleq$ *M to I*
$\land\ cState[n] =$ "M"
$\land\ rmv\_owner(n)$
$\land\ upd\_mem\_data(n)$
$\land\ upd\_state(n,$ "I")
$\land\ upd\_dir\_state($"I")
$\land\ unchanged\_gMsgs$
$\land$ UNCHANGED $\langle dReqPending,\ cData,\ cRcvAcks\rangle$

$PutS(n) \triangleq$ *S to I/S*
$\land\ cState[n] =$ "S"
$\land\ rmv\_sharer(n)$
$\land\ upd\_state(n,$ "I")
$\land$ IF $Cardinality(dSharers) = 1$
THEN $upd\_dir\_state($"I")

ELSE $upd\_dir\_state(\text{“S”})$
$\land$ $unchanged\_gmMsgs$
$\land$ UNCHANGED $\langle dOwner,\ dReqPending,\ cData,\ cRcvAcks\rangle$

$PutO(n) \triangleq$ *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 $\langle dReqPending,\ cData,\ cRcvAcks\rangle$

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 $\langle dOwner,\ dReqPending,\ cRcvAcks\rangle$

$GetM\_dI(n) \triangleq$ 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 $\langle dReqPending,\ cRcvAcks\rangle$

Dir
$GetS\_Fwd(n) \triangleq$
$\land$ $dState \neq \text{“I”}$
$\land$ $cState[n] = \text{“I”}$
$\land$ $dir\_set\_action\_pending$
$\land$ $ucst\_FwdGetS(n,\ owner\_or\_min\_sharer)$
$\land$ IF $(dState = \text{“E”} \lor dState = \text{“S”})$
THEN $\land$ $upd\_dir\_state(\text{“S”})$

```
        ELSE  upd_dir_state("O")
    ∧ unchanged_gmc
    ∧ UNCHANGED ⟨dOwner, dSharers⟩

GetS(n) ≜
     ∨ GetS_dI(n)
     ∨ GetS_Fwd(n)

 Sharers
RcvFwdGetS(n, m) ≜
    ∧ rcv_FwdGetS(m, n)
    ∧ resp_SData(m)
    ∧ IF (cState[n] = "E" ∨ cState[n] = "S")
        THEN upd_state(n, "S")
        ELSE upd_state(n, "O")
    ∧ unchanged_gmd
    ∧ UNCHANGED ⟨cData, cRcvAcks⟩

 Requester
RcvData(n, m) ≜
    ∧   rcv_SData(m, n)
    ∧   deliver_Msg(m)
    ∧   add_sharer(n)
    ∧   upd_state(n, "S")
    ∧   upd_core_data(n, m.data)
    ∧   dir_rst_action_pending
    ∧   unchanged_gm
    ∧   UNCHANGED ⟨dOwner, dState, cRcvAcks⟩
```

```
 Dir
GetM_Invs(n) ≜
    ∧ dState ≠ "I"
    ∧ cState[n] ≠ "M"
    ∧ cState[n] ≠ "E"
    ∧ Cardinality(dSharers \ {n}) > 0
    ∧ rst_acks(n)
    ∧ dir_set_action_pending
    ∧ upd_dir_state("M")
    ∧ unchanged_m
    ∧ UNCHANGED ⟨dOwner, dSharers, cState, cData⟩
    ∧ IF (dState = "E" ∨ dState = "M")
        THEN  ∧ ucst_FwdGetM(n, owner_or_min_sharer)      single remote owner case
              ∧ unchanged_g
        ELSE  IF (dState = "S" ∨ dOwner = n)
                THEN  ∧ bcst_DInv(n, dSharers \ {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*$(n,$ "I")
$\land$ IF *rcv_DInv*$(m,\ n)$
THEN *resp_SAck*(m)
ELSE *resp_SDataAck*(m)
$\land$ *unchanged_gmd*
$\land$ UNCHANGED $\langle cData,\ cRcvAcks\rangle$

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 $\langle dState\rangle$
$\land$ IF *rcv_SDataAck*$(m,\ n)$
THEN *upd_core_data*$(n,\ m.data)$
ELSE UNCHANGED $\langle cData\rangle$ *TODO*
$\land$ 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$(n) \triangleq$

$\wedge\ rst\_acks(n)$
$\wedge\ dir\_set\_action\_pending$
$\wedge\ bcst\_Upd(n,\ CORES \setminus \{n\},\ cData[n])$
$\wedge\ unchanged\_mMsgs$
$\wedge$ UNCHANGED $\langle cData,\ cState,\ dOwner,\ dSharers,\ dState\rangle$

$RcvUpd(n,\ m)\ \triangleq$
$\wedge\ rcv\_Upd(m,\ n)$
$\wedge\ resp\_UAck(m)$ todo may add rejection of sharing with *Nacks* and not transitioning to S
$\wedge\ upd\_state(n,\ \text{“S”})$
$\wedge\ upd\_core\_data(n,\ m.data)$
$\wedge\ unchanged\_gmd$
$\wedge$ UNCHANGED $\langle cRcvAcks\rangle$

$RcvUpdAck(n,\ m)\ \triangleq$
$\wedge\ cState[n] = \text{“M”}$
$\wedge\ rcv\_upd\_ack\_msg(n,\ m)$
$\wedge\ deliver\_Msg(m)$
$\wedge\ unchanged\_gm$
$\wedge$ UNCHANGED $\langle cData\rangle$
$\wedge$ IF $\neg is\_last\_upd\_Ack(n,\ m)$
THEN $\wedge\ add\_ack(n,\ m)$
$\wedge\ unchanged\_d$
$\wedge$ UNCHANGED $\langle cState\rangle$
ELSE $\wedge\ rst\_acks(n)$
$\wedge\ upd\_state(n,\ \text{“O”})$
$\wedge\ dState' = \text{“O”}$
$\wedge\ dOwner' = n$
$\wedge\ dSharers' = CORES$
$\wedge\ dir\_rst\_action\_pending$

$must\_update(n)\ \triangleq$
$\wedge\ cState[n] = \text{“M”}$
$\wedge\ cData[n] = WRITE\_TO\_UPDATE$

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

```
                    ∨ PutO(n)

SharerActions(n, m) ≜
    ∨ RcvUpd(n, m)
    ∨ RcvInv(n, m)
    ∨ RcvFwdGetS(n, m)

RequesterActions(n, m) ≜
    ∨ RcvAck(n, m)
    ∨ RcvData(n, m)
    ∨ RcvUpdAck(n, m)

MessageActions(n) ≜
    ∃ m ∈ Msgs :
        ∨ SharerActions(n, m)
        ∨ RequesterActions(n, m)

PerformBcast ≜
        ∧ gBcstMsg ≠ {}
        ∧ ∃ m ∈ gBcstMsg :
            ∧ _send_Msg(m)
            ∧ unchanged_mcd
            ∧ IF gBcstMsgRcvers = {}
              THEN ∧ gBcstMsg′ = {}
                   ∧ UNCHANGED ⟨gBcstMsgRcvers⟩
              ELSE LET rcver ≜ CHOOSE x ∈ gBcstMsgRcvers : TRUEIN
                   ∧ gBcstMsg′ = {[m EXCEPT !.receiver = rcver]}
                   ∧ gBcstMsgRcvers′ = gBcstMsgRcvers \ {rcver}
```

```
WriteData(n) ≜
    ∧ cState[n] = "M"
    ∧ cData[n] < MAX_WRITES
    ∧ ¬must_update(n)
    ∧ cData′ = [cData EXCEPT ![n] = cData[n] + 1]
    ∧ unchanged_gdmMsgs
    ∧ UNCHANGED ⟨cState, cRcvAcks⟩
```

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

```
ANext ≜
        IF   gBcstMsg ≠ {}
         THEN PerformBcast
         ELSE ∃ n ∈ CORES :
                 ∨ Requests(n)
                 ∨ WriteData(n)
                 ∨ MessageActions(n)
```

The complete definition of the algorithm

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

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

THEOREM $Spec \Rightarrow Invariants$