Object Transfer using Path Reversal: Implementation program

Shankar

May 19, 2014
Object-transfer implementation program

Proving implements: see text
program ObjTrPathReversalDist (ADDR, OID, OVAL, initObj)  
// implements ObjTransferService with same params  
{cj} ← start FifoChannel(ADDR)  
for j in ADDR  
  vj ← start ObjTrPathReversal (ADDR, j, cj, ...)  
return {vj}

ObjTrPathReversal: await program  
- implements an independent path-reversal alg for each object  
- input fns: acq(oid), rxReq(), rel(oid, oval)  
- output calls: tx, rx of channel access system
Program ObjTrPathReversal – 1

- **Parameters:**
  - local addr j, cj, ADDR, OID, OVAL, initObjs

- **Main**
  - req msgs: [REQ, addr, oid]
  - obj msgs: [OBJ, oid, oval]
  - lst: last-ptrs map  // oid entry iff last ptr non-nil
  - nxt: next-ptrs map  // oid entry iff next ptr non-nil
  - objs: set <OID>  // local objs at user or in buffer
  - objBuff: map <OID, OVAL>  // objs for local user
  - reqBuff: seq <OID>  // reqs for local user
  - startThread (doRx())  // rcvs msgs from channel

- **Update last/next ptrs upon rcving msg**
  - so objBuff, reqBuff part of user wrt path-reversal alg
Program ObjTrPathReversal – 2

- atomicity assumption: awaits
- progress assumption: weak fairness

Input `acq(x)`:
- `await (true)`
  - `cj.tx(lst_x, [REQ, j, x])`
  - `remove lst_x`
- `await (objBuff_x)`
  - `remove objBuff_x`
- `return it`

// Rule: become hungry(x)

// already eating(x)
Program ObjTrPathReversal – 3

- input rxReq():
  await (non-empty reqBuff)
  remove reqBuff.head
  return it

- input rel(x, oval):
  await (true)  // Rule: become thinking(x)
  cj.tx(nxtX, [OBJ, x, oval])
  remove nxtX
  return
function doRx(): // executed by local thread
    while true:
        msg ← cj.rx()
        await (true)
        if msg is [OBJ, x, oval] // Rule: become eating(x)
            objBuff_x ← oval
            add x to objs
        else if msg is [REQ, k, x] // Rule: rcv request(k, x)
            if (lst_x)
                cj.tx(lst_x, msg)
                lst_x ← k
            else
                nxt_x ← lst_x ← k
                append x to reqBuff
Choose to update last/next ptrs upon rcving msg
- so objBuff, reqBuff are part of user wrt path-reversal alg
- eating wrt oid: oid in objs or in objBuff
- hungry wrt oid: ongoing j.acq(oid) and oid not in objBuff
- thinking wrt oid: not eating nor hungry wrt oid

acq(oid):
- send [REQ,j,oid] to lst oid
- remove oid entry from lst
- wait for oid entry in objBuff
- remove oid entry from objBuff and return it
Object-transfer implementation program

Proving implements: see text