Object Transfer using Path Reversal: Implementation program

Shankar

May 19, 2014

Outline

Object-transfer implementation program

Proving implements: see text

Distributed program

- program ObjTrPathReversalDist(ADDR,OID, OVAL, initObjs)
  // implements ObjTransferService with same params
  $\{c_j\}$ ← start FifoChannel(ADDR)
  for j in ADDR
    v_j ← start ObjTrPathReversal(ADDR, j, c_j, ...)
  return $\{v_j\}$

- 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, c_j, 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

- atomicity assumption: awaits
- progress assumption: weak fairness
- input acq(x):

```
await (true)                                  // Rule: become hungry(x)
    c_j.tx( lst_X, [REQ, j, x] )
    remove lst_X
await (objBuff_X)                             // already eating(x)
    remove objBuff_X
return it
```

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

- input rel(x, oval):
  await (true)
    c_j.tx(nxt_X, [OBJ, x, oval])
    remove nxt_X
  return

// Rule: become thinking(x)

- function doRx(): // executed by local thread

```
while true:
  msg ← c_j.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)
        c_j.tx( lst_x, msg)
        lst_x ← k
      else
        nxt_x ← lst_x ← k
        append x to reqBuff
```

Refining alg-level to program

- 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 $\texttt{lst}_{\texttt{oid}}$
  remove oid entry from lst
  wait for oid entry in objBuff
  remove oid entry from objBuff and return it

Outline

Object-transfer implementation program

Proving implements: see text