Object Transfer Service

Shankar

September 18, 2014

Overview: Object Transfer Service

- Object: id and mutable value // eg, page # and contents
- Service allows systems to share objects
  - acquire an object, change its value, release it
  - acquired value equals last-released value
- Objects "at rest" reside with users, not service // unlike lock
  - object's owner: user that currently holds it
  - object is unowned if it currently has no owner
  - objects have initial owners
  - user can acquire an object // blocking
  - service can request user for an object
  - user releases object only when requested

Object transfer service – 1

- Parameters
    - ADDR: set of addresses
    - OID: set of object ids (oids)
    - OVAL: possible values of an object
    - {initObjs$_j$}: oids of objects with user at j

- Main
    - objs$_j$ ← initObjs$_j$ // objects at user j
    - reqs$_j$ ← set() // objects requested by user j
    - val$_{oid}$, for unowned oid // value of obj at last release
    - return {v$_j$ ← sid()} // access system at j

Object transfer service – 2

- v_j.acq(oid) // acquire object and its value
  - ic { no ongoing v_j.acq(oid) and oid not in objs_j }
  - output rval

    oc { val_{oid} exists and rval $=$ val_{oid} }

    move oid from val to objs_j

    return rval

- v_j.rel(oid, oval) // release object and its value
  - ic { oid in objs_j and in reqs_j }

    remove oid from objs_j and from reqs_j

    $\text{val}_{\text{oid}} \leftarrow$ oval
  - oc { true }

    return

Object transfer service – 3

- v_j.rxReq()   // rcv request for object
  - ic { no ongoing v_j.rxReq() }
  - output oid

```
    oc { ( oid not in reqs_j )  and
         ( oid in objs_j  or  ongoing v_j.acq(oid) ) }
    add oid to reqs_j
    return oid
```

- atomicity assumption: input parts and output parts

Object transfer service: progress assumption

- every rel call returns
  - ongoing `j.rel(x,v)` *leads-to* no ongoing `j.rel(x,v)`
- if a user wants an object then the owner is informed, provided the owner maintains an ongoing rxReq call
  - (`objs`$_j$ not empty *leads-to* ongoing `j.`rxReq) $\Rightarrow$
    (x in `objs`$_j$ and ongoing `k.`acq(x)) *leads-to* x in reqs$_j$
- if a user wants an object then it gets it provided the owner rcvs a request and then releases the object
  - (x in `objs`$_j$ and ongoing `k.`acq(x) *leads-to* x in reqs$_j$)
    and (x in reqs$_j$ *leads-to* x not in reqs$_j$)
    $\Rightarrow$ (ongoing `j.`acq(x) *leads-to* no ongoing `j.`acq(x))