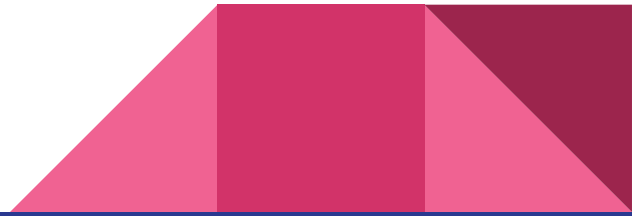
Online Collaborative Document Editing Application

Chen Li 27607798
Yujing Xie 33404144

Function

- A Web Application
- Real-time Collaborative edit, insert
- Operational transformation

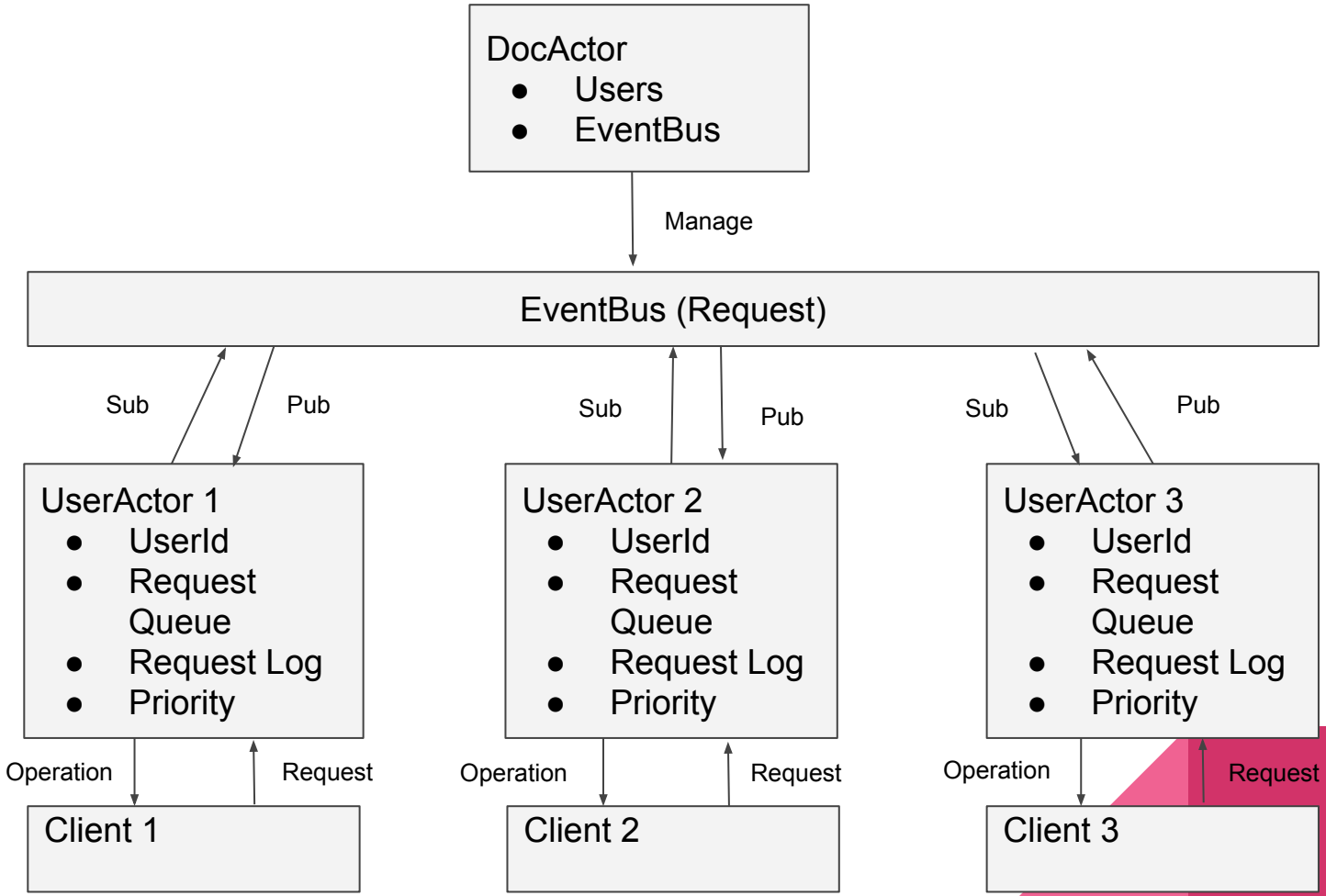


Technology Architecture

Play Framework: Non-blocking I/O Web Framework

Akka: Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM

AngularJS: Websocket, two-way data binding


DocActor
Users
EventBus
Manage
EventBus (Request)
Sub
Pub
Sub
Pub
Sub
Pub
UserActor 1
UserId
Request Queue
Request Log
Priority
UserActor 2
UserId
Request Queue
Request Log
Priority
UserActor 3
UserId
Request Queue
Request Log
Priority
Operation
Request
Operation
Request
Operation
Request
Client 1
Client 2
Client 3

The Algorithm

- Operation Transformation Model
- Example:
  - O1 = Insert[0, "x"] (to insert character "x" at position "0")
  - O2 = Delete[2, "c"] (to delete the character "c" at position "2")
- No lock required

$T_{11}(o_i, o_j, p_i, p_j) = o_i'$ where
 if($P_i < P_j$)
  o_i' = insert[X_i; P_i]
 else if($P_i > P_j$)
  o_i' = insert[X_i; $P_i + 1$]
 else /* $P_i = P_j$ */
  if($X_i = X_j$)
   $o_i' = \emptyset$
  else
   if($p_i > p_j$)
    o_i' = insert[X_i; $P_i + 1$]
   else /* $p_i < p_j$ */
    o_i' = insert[X_i; P_i]
   fi
  fi

T_{22}(delete[P_i], delete[P_j], p_i, p_j) = o_i' where
 if($P_i < P_j$)
  o_i' = delete[P_i]
 else if ($P_i > P_j$)
  o_i' = delete[$P_i - 1$]
 else
  $o_i' = \emptyset$
 fi

T_{12}(insert[X_i; P_i], delete[P_j], p_i, p_j) = o_i' where
 if($P_i < P_j$)
  o_i' = insert[X_i; P_i]
 else
  o_i' = insert[X_i; $P_i - 1$]
 fi

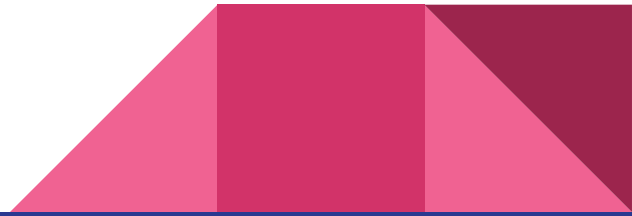
T_{21}(delete[P_i], insert[X_j; P_j], p_i, p_j) = o_i' where
 if($P_i < P_j$)
  o_i' = delete[P_i]
 else
  o_i' = delete[$P_i + 1$]

Operation Transformation

- **Transformation Matrix**
  1. A m*m matrix, m is the cardinality of the set O
  2. The component of the matrix is the transformation function from operation *o* to operation *p*, as $o_j' = T_{uv}(o_j, o_i, p_j, p_i)$
- **State Vector**
  1. Each site store a N dimension vector, the ***i***th component of the vector in site j indicate that how many operation site j has executed from site i
  2. Assure the precedence
  3. Relation of the site vection, given vector vi and vj :
     a. Vi = Vj : each component of Vi is equal to Vj
     b. Vi < Vj : each component of Vi is less than or equal to Vj and at least one is less than
     c. Vi > Vj : at least one component of Vi is greater than Vj
- **Priority:** User Id

Operation Transformation

- Request
  - <i, s,o,p>
- Request Queue
  - Requests waited to be executed
- Request Log
  - Request already been executed



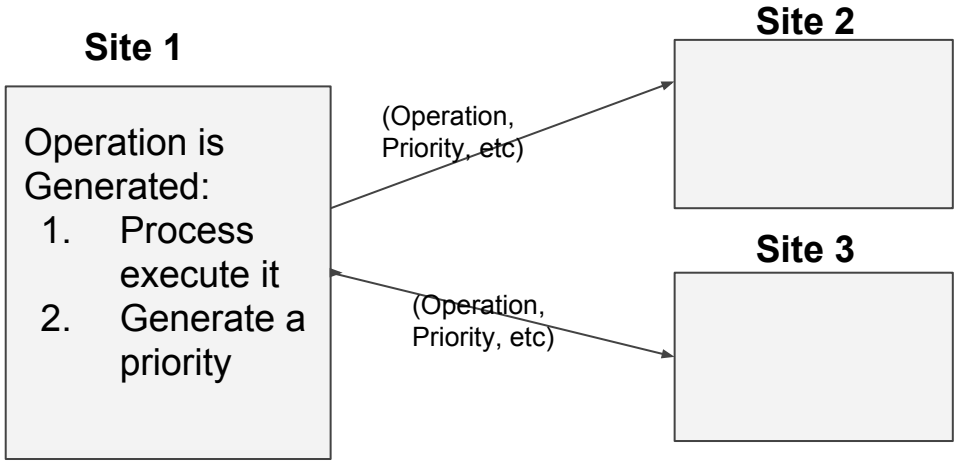
Site Process

Perform Three Activities :

- Operation generation
- Operation reception
- Operation execution

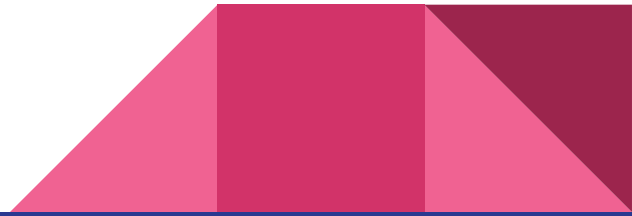
Generate Operation

- Operations: Insert(a,2), Delete(0)
- When an operation is generated at a site, the site process add the request to its local request queue, then generate a priority for the operation and send requests including the operation and its priority to all other sites



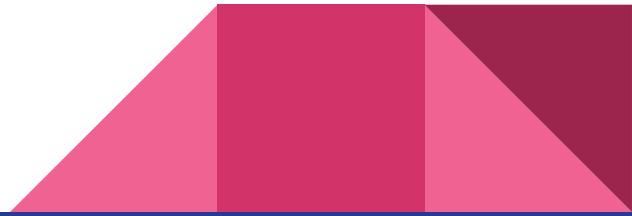
Receive Operation

- When a site i receive a request from site j, add the request to the request queue.



Execute Operation

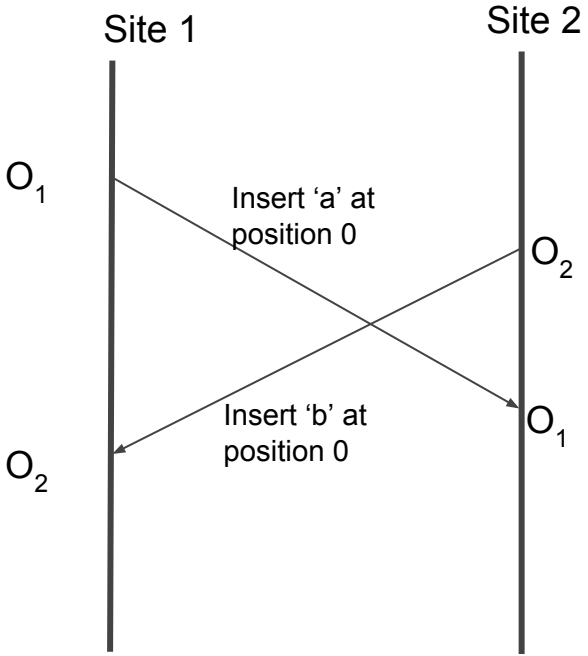
- Dequeue from request queue
- At first it will check whether this operation is future operation which has been executed by site j but not by site i, or past operation(which site i has executed but not site j), or current operation
- ***Future operation***: queue the operation
- ***Past operation:*** transform the operation
- ***Current operation:*** execute the operation



Execute Operation

```
for each <j, s_j, o_j, p_j> ∈ Q_i where s_j ≤ s_i begin
  Q_i ← Q_i - <j, s_j, o_j, p_j>
  if s_j < s_i
    <k, s_k, o_k, p_k> ← most recent entry in L_i
                 where s_k ≤ s_j (or ∅ if none)
    do while <k, s_k, o_k, p_k> ≠ ∅ and o_j ≠ ∅
      if the k'th component of s_j
                 is ≤ the k'th component of s_k
        let u be the index of o_j (o_j is an instance of O_u)
        let v be the index of o_k (o_k is an instance of O_v)
        o_j ← T_uv(o_j, o_k, p_j, p_k)
      fi
      <k, s_k, o_k, p_k> ← next entry in L_i (or ∅ if none)
    od
  fi
  perform operation o_j on i's site object
  L_i ← L_i + <j, s_i, o_j, p_j>
  s_i ← s_i with j'th component incremented by 1
end
```

Queue & Log



Initial
L_1 : { }
Q_1 : { }

Operation: user 1 insert a at position 0 , user 2 insert b at position 0 at the same time

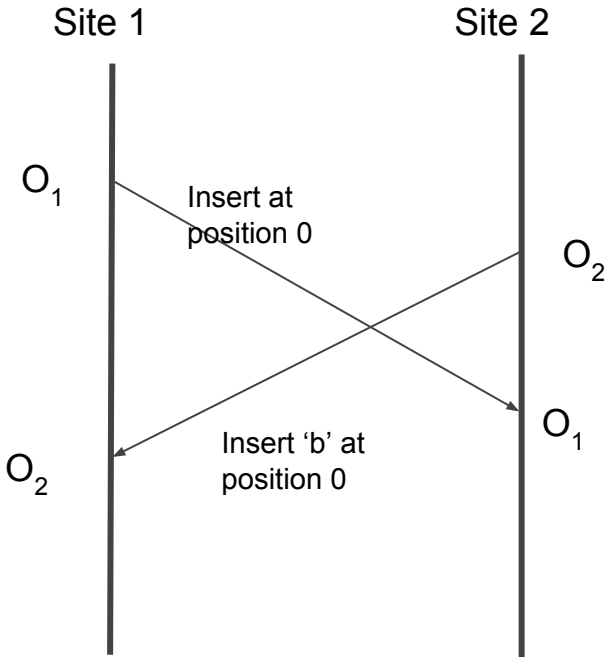
After user 1 insert a
L_1 : {}
Q_1 : {<1,<1,0>,O_1,1>}

Then **dequeue** request <1,<1,0>,O_1,1>, execute it on site 1
L_1 : {<1,<1,0>,O_1,1>}
Q_1 : {}
Receive request from site 2,<2,<0,1>,O_2,2>
L_1 : {<1, <1,0>, O_1, 1>}
Q_1 : { <2, <0,0>, O_2, 2>}

Dequeue:
Operational Transformation:
Because s_2 is <0,0>, which is less than s_1 <1,0>, so transform to insert b at position 1

Execute operation O_2,
L_1 : {<1, <1,0>, O_1, 1>, <2, <0,0>, O_2, 2>}
Q_1 : { }

Insert & Insert

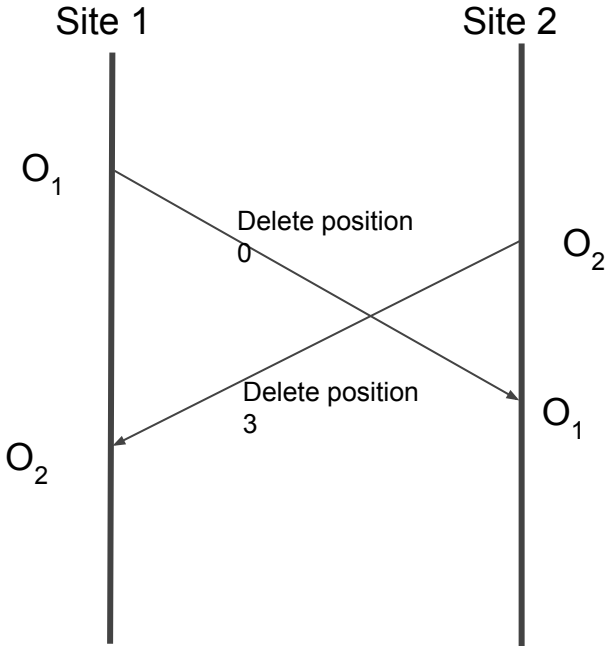


Operation: user 1 insert at position 0 a, user 2 insert b at position 0 at the same time

Non-OT:
User 1: ba
User 2: ab

OT:
User 1: ab
User 2: ab

Delete & Delete

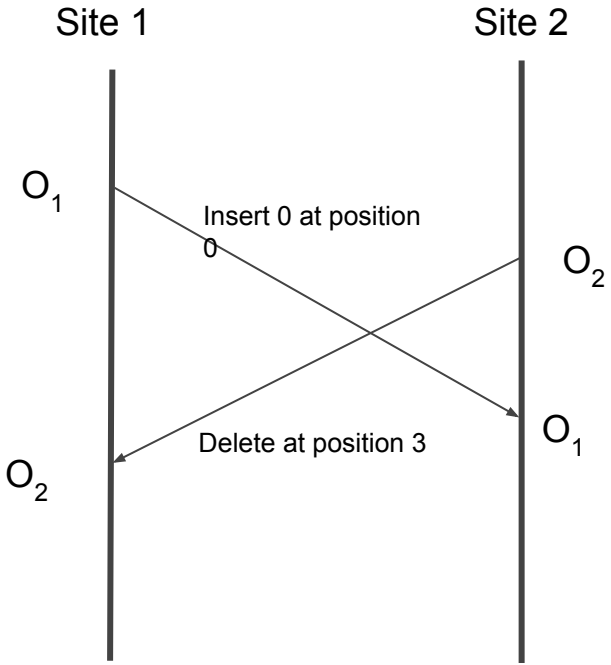


Operation:
Initial state: abcde
Operation:
user 1: delete position 0
user 2: delete 3

Non-OT:
User 1: bcd
User 2: bce

OT:
User 1: bce
User 2: bce

Insert & Delete



Operation:
Initial status: abcdef
user 1 insert a at position 0, user 2 delete at position 3 at the same time

Non-OT:
User 1: aabdef
User 2: aabcef

OT:
User 1: aabcef
User 2: aabcef

Thank you

Lamport Partial Order

Convergence Property

- ***Quiescent*** iff all generated operation has been executed at all sites
- All sites are identical at all sites at quiescent

Correct

- Satisfy both ***Precedence and Convergence***

Lamport Partial Order

- **Precedence**

  Operation *o* and *p*, Site *s* and *t*, *o* precede *p* iff:

  1. s=t and *o* happens before *p*
  2. s=!t and *o* execute before *p* generation

If one operation *o* precede *p*, then at each site the execution of *o* happens before *p*