Achieving CCI Efficiently by Combining OT and Dynamic Locking with Lazy Consistency in a Peer-to-Peer CES

Jon A Preston and Sushil K Prasad

Department of Computer Science
Georgia State University
Atlanta, GA USA

jon.preston@acm.org and sprasad@gsu.edu
Motivation

- Better achieve CCI
- Avoid reduced concurrency of locking
- Reduce communication/computation cost associated with OT
Better Achieve CCI

- Consistency & causality preservation via OT
- Intention preservation more difficult
  - Recent work of Sun/Sosič, Ignat, & Li
- Combine OT and locking
  - Cache changes when possible
  - Reduce costs of applying OT globally
  - Local history buffers
Combining OT and Locking

- Model document as tree
- Promote and demote locks when possible
- Apply OT
  - At an atomic level (when demotion not possible)
  - As desired by users (higher in the document tree)
Dynamic Locking: Avoiding Reduction of Concurrency

- Locks dynamically “resize”
- Users are always granted the largest sub-tree within the document
  - Demote until contention on lock is removed
  - Promote when user leaves and contention removed
- Allow users to maintain multiple locks (i.e. retain lock if expected to return within a short time)
Maintaining the Locks
Visualizing the Lock Maintenance
Distributing the Document Tree Among Peers

- Changes are cached & only broadcast to other peers within the same section
- History buffers sent and replayed at other peers
  - When demotion occurs (send $\Delta x$ to $U_2$ above)
  - When promotion occurs (send $\Delta$ to newly-promoted peer)
Improving Intention Preservation

- As identified by others, intention preservation is difficult to achieve via OT
  - Changes may achieve intention locally
  - But when converged, the changes invalidate the local intentions

- We promote $\Delta$ (history buffers) up the document tree
  - Rely upon semantic structure of document
  - Selectively apply $\Delta$s at the higher level in tree
    - Query user(s)
    - Semantically-aware algorithm for detecting and automatically resolving conflict
Promoting $\Delta$s (History Buffers)

- $U_1$ leaves, $\Delta w$ is transferred to $U_2$ as $U_2$ is promoted to $v$
- Can choose to keep $\Delta w$ and/or $\Delta x$ (or some hybrid combination)

$U_1$ and $U_2$ are depicted with $\Delta w$ and $\Delta x$ respectively, pointing to $v$. The diagram shows $U_2$ being promoted to $v$ with $\Delta v = \Delta w + \Delta x$.
Modeling the Peers and Changes to $\Delta$

- Changes to local $\Delta_v$ come from local user, peers within $v$ and children of $v$
- Promotion of $\Delta$s up the tree can be improved by reduction
Reducing Communication Cost of OT

- Changes cached locally, so we avoid global multicast
- Selectively multicast among “interested” peers
- Can replay set of changes when needed
  - \( \Delta \) (history buffer) sent in one message rather than individual changes in many messages (Yang et al)
  - Local undo’s reduce \( \Delta \), so \( \Delta \) is smaller
Reducing Costs via Promotion and Reduction

- Promotion of $\Delta$ can occur without a lock promotion

$\Delta w$ is reduced to $\Delta w'$ and sent to $v$  
$\Delta v = \Delta w' + \Delta x'$

$\Delta x$ is reduced to $\Delta x'$ and sent to $v$  
$v' = v + \Delta v$

$U_3$ joins at $x$
$U_2$ sends $\Delta v$ and $\Delta x''$ to $U_3$
Conclusions

- We believe CCI is best achieved when combining OT and locking
  - The semantic structure of the document can be leveraged to better achieve intention preservation
- Communication costs can be reduced
  - Avoid global broadcast of changes (cache them)
- Computation costs can be reduced
  - Cached Δs can be reduced and merged up the tree
- Response times among peers still being investigated
- Reduction algorithm still being investigated