Moving (and averaging) values over channels with message loss, replay, and re-ordering

Carlos Baquero

(Joint work with Paulo Almeida, Ali Shoker)

Presented at UPMC LIP6, January 2015
Moving/Handoff Problem

Nodes in a network have splittable value quantities, and the task is to reliably move quantities from node to node.

Each transfer involves only two parties, no global agreement. Possible uses include:

- Non-negative inc/dec shared counters (Positive PN-Counter)
- Stock escrow
- Token/lock transfers
- Distributed averaging and derived data aggregates
# Sketch of Handoff

## Source Node $i$

- **state:** $v$
- **on** `transfer(j, q)`
  - move $q$ to node $j$; $q \leq v$
  - $v := v - q$
  - `send_j(q)`

## Destination Node $j$

- **state:** $v$
- **on** `receive_i(q)`
  - $v := v + q$
Sketch of Handoff, commutative monoid with split

Split definition:

\[(v', q) = \text{split}(v, h) \text{ such that } v' \oplus q = v \text{ and } q \leq h\]

Source Node \(i\)

**state:** \(v\)  
any commutative monoid

**on** transfer \((j, h)\)  
move \(h\), or less, to node \(j\)

\((v, q) := \text{split}(v, h)\)

send\(_j(q)\)

Destination Node \(j\)

**state:** \(v\)  
any commutative monoid

**on** receive\(_i(q)\)

\(v := v \oplus q\)
Conservation of quantities requires an exactly-once delivery from each send to corresponding receive.

TCP mostly ensures exactly-once, but degrades to at-most-once upon connection break.

UDP can duplicate, drop and re-order messages.
Naive exactly-once over UDP

- Source assigns a unique id to each sent message
- Messages are re-transmitted until acknowledged
- Destination stores unique ids to avoid duplicated delivers
- (more compact sequence numbers ids can be used for FIFO)

+ Source can transmit immediately (one-way handshake)

− Node state at least linear on the number of (past) parties
TCP connection management

- No connection specific information between incarnations
- Three-way handshake to make connection
- Unbounded memory, to keep counters

A transfer over TCP pays a latency price and yet is still sensible to connection breaks
# Handoff

## System Model

- Network can **duplicate**, **drop** and **re-order**
- Nodes only have connection specific info during transfers
- Nodes can fail, but eventually recover

Three-way handshake is needed (Attiya, Rappoport. DC 1997)

## Strategy

Adapt (piggybacking) three-way handshake steps:

1. Announce available value and sender counter/clock
2. Prepare receive slot and request quantity hint
3. Split value, up to hint, and send exactly-once quantity
4. (Garbage collect at sender, upon acknowledge)
Handoff

i

val: 8
sck: 10

j

val: 4
dck: 20

Handoff

Time
Handoff

\[
(6,2) = \text{split}(8,2)
\]
Handoff

\[6 = 4 \oplus 2\]
Handoff

Duplicate Resilient Communication

i
val: 8
sck: 10

j
val: 4
dck: 20

val: 6
sck: 11

val: 6
dck: 21

Handoff

Time
Payload monoid data types
Value averaging

Positive reals that ask for half difference, give as much as possible

\[0 \equal 0\]
\[\bigoplus \equiv +\]
\[\text{needs}(x, y) \equiv \frac{y - x + |y - x|}{4}\]
\[\text{split}(x, h) \equiv (\frac{x - h + |x - h|}{2}, \frac{x + h - |x - h|}{2})\]

Derived aggregates include global sums and node counting
Monodic values might not be in total order

\[ X = \{ \text{single} \mapsto 8, \text{double} \mapsto 12 \} \]

\[ Y = \{ \text{single} \mapsto 1, \text{double} \mapsto 20 \} \]

Leading to transfers in both directions

\[ \{ \text{double} \mapsto 4 \} = \text{needs}(X, Y) \]

\[ \{ \text{single} \mapsto 3 \} = \text{needs}(Y, X) \]

Eventually stabilizing with

\[ X = \{ \text{single} \mapsto 5, \text{double} \mapsto 16 \} \]

\[ Y = \{ \text{single} \mapsto 4, \text{double} \mapsto 16 \} \]
Graph properties

- Graph with $n$ nodes and each with $2 \log n$ links
- (Symmetric forward and backward Chord)
- Small world topology. Low path lengths, High clustering
- Synchronous message model
- Initial values from integer uniform distribution 0 : 255
- All converge to average, about 128
Simple experiment that aims to check resilience to message drop and message duplication faults (dropping and duplication can also lead to re-ordering events), and show final GC of all connection meta-data.

- Execution with no faults
- Executions with 25, 50 and 75% message loss faults
- Executions with 25, 50 and 75% message replay faults
- Execution with 75% mixed faults

Storage probability for replay is at 20% (lower means older replays)

(Note: *need* and *split* functions not yet optimized for this topology)
Experiments
No loss

Showing linear meta-data size, excluding log growing clocks

1024 nodes, loss=0%, replay=0%
Experiments

25% loss

![Graph showing 1024 nodes, loss=25%, replay=0%](image)
Experiments
50% loss
Experiments
75% loss
Experiments
No loss

1024 nodes, loss=0%, replay=0%

Data size

Synchronous rounds

Slots
Tokens
Experiments
25% replay

1024 nodes, loss=0%, replay=25%
Experiments
50% replay

1024 nodes, loss=0%, replay=50%

![Graph showing data size over synchronous rounds. The graph compares slots and tokens.]
Experiments
75% replay

1024 nodes, loss=0%, replay=75%

Data size

Synchronous rounds
Experiments
No loss

1024 nodes, loss=0%, replay=0%

Data size vs. Synchronous rounds graph showing the behavior of slots and tokens over time.
Experiments
75% loss, 75% replay

1024 nodes, loss=75%, replay=75%
+/-  Base algorithm is not optimized for this experiment
+  Still, there is clear high resilience to faults
+  State after $t$ transfers is eventually $O(\log t)$
-  Topology must ensure symmetric exchanges
-  Uncontrolled churn impacts GC:
  -  Meta-data kept, linear with failed node peers $k$
  +  If degree is $\log n$ then $k \leq \log n$
+  Implemented in C++, for int, float and map payload
Related Work

Questions?

Email: cbm@di.uminho.pt
Twitter: @xmal