Rollback Recovery for Middleboxes

Justine Sherry, Peter Gao, Soumya Basu, Aurojit Panda, Arvind Krishnamurthy, Christian Maciocco, Maziar Manesh, João Martins, Sylvia Ratnasamy, Luigi Rizzo, Scott Shenker
The Goal: Recovery from Failure
Three Challenges

(1) Statefulness

(2) Nondeterminism

(3) Latency
The Solution:
Our system, FTMB, implements **rollback recovery** with low overhead through **inband logging** and ordered release.
FTMB performs stateful recovery.
What happens when recovery is stateless?

- **Network Address Translator**: All active connections are reset/dropped.

- **Intrusion Prevention System**: lost history and gaps in security record
Stateful Recovery: Classical Definition

“A system recovers correctly if its internal state is consistent with the observable behavior of the system before the failure.”

[Strom & Yemeni 1985]
Stateful Recovery: Layman’s Terms

Never allow a packet to egress the system unless the system is able to immediately wake up a replica which “remembers” transmitting that packet.

This property is called “output commit.”
FTMB performs stateful recovery, for multithreaded, non-deterministic middleboxes.
Nondeterminism

- Data Races (accesses to “shared state”)

Diagram:

1. P1 SYN
   - Active Connections = 4
2. P2 SYN
   - Active Connections = 5
3. P2 SYN
   - Active Connections = 5
   - P1 SYN
   - Active Connections = 5
   - P1 SYN
   - Active Connections = 4
FTMB performs stateful recovery for multithreaded, non-deterministic middleboxes with low latency overhead.
Latency under HA Solutions

CDF of Packets

Latency (us)

MazuNAT (Baseline)

Median latency: 70us
Latency under HA Solutions

CDF of Packets vs. Latency (us)

- MazuNAT (Baseline)
- Pico (reported)
- MazuNAT, under Remus

App-Layer Checkpointing: 8.5ms
VM Checkpointing: 50ms
Latency under HA Solutions

FTMB: 100us

CDF of Packets vs. Latency (us)

- MazuNAT (Baseline)
- MazuNAT, with FTMB and Snapshots
- Pico (reported)
- MazuNAT, under Remus
An Active:Active system gets Bogged down with checkpoints.

Worst of all!
How does FTMB achieve all this?

FTMB, implements rollback recovery with low overhead through inband logging and ordered release.
Classic Design: Checkpoint + Log

Input Logger

Hypervisor

Click

Master

Output Logger

Output (Steady State)
Classic Design: Checkpoint + Log

- Input Logger
- Hypervisor
  - Click
  - Master
- Output Logger
- Input Packets (Recovery)
- VM Checkpoints (Steady State)
- Hypervisor
  - Shadow VM
  - Output Verification
  - Backup
- Output (Post-Recovery)
- Output (Steady State)
Classic Design: Checkpoint + Log

Input Logger → Hypervisor [Click] → Output Logger

Input Packets (Recovery) → VM Checkpoints (Steady State) → Output Packets PALs (Recovery)

Master

Output (Steady State)

Backup

Output (Post-Recovery)

Determinants
Packet Access Logs

• FTMB records determinants via a “Packet Access Log” containing:
  – An ID number for the packet
  – An ID number for the variable accessed
  – A sequence number enumerating the number of accesses at the variable ("this is the $n^{th}$ packet to accesses variable $v$")

• $\text{PAL} = [p_i, v_j, s_{ij}]$
Stateful Recovery: Layman’s Terms

Never allow a packet to egress the system unless the system is able to wake up a replica which “remembers” transmitting that packet.

This property is called “output commit.”
FTMB

• A rollback-recovery approach with lightweight logging of determinants

Key techniques:
  – Inband logging
  – Ordered release

• Median latency penalty? 30µs.
Inband Logging

Master

Logger
Inband Logging

Master

Logger

[P2, V1, S12]
Inband Logging

- Uses the existing packet processing pipeline: take advantage of the path we’ve already optimized.
- Packets and logs arrive at same output log so that output commit decision can be made easily.
- PALs may leave before their packets, even if the packet is withheld or dropped within the middlebox.
FTMB

• A rollback-recovery approach with lightweight logging of determinants

Key techniques:
  – Inband logging
  – Ordered release

• Median latency penalty? 30µs.
Ordered Release

• Makes output commit decision fast
• Need to decide, given a packet at the logger, whether or not it is `safe’ to release it.
  – Check whether all PALs it depends on have been logged.
Packet Dependencies

Master

PAL Queue

Pkt Queue

Logger
Packet Dependencies

Master

Logger

PAL Queue

Pkt Queue
Packet Dependencies

\[ DEP(p_i) = DEP(p_i, \text{last}(p_i)) \]

\[ DEP(p_i, v_j) = \text{PAL}(p_i, v_j) \ldots \]
\[ \cup \text{DEP}(\text{pkt}(v_j, s_{ij} - 1), v_j) \ldots \]
\[ \cup \text{DEP}(p_i, \text{prev}(p_i, v_j)) \]

[We define dependencies formally in our Technical Report.]
Packet Dependencies

A (provably) equivalent, but easier to enforce set of conditions:

(1) \[ PAL(pkt(v_j,s_{ij} - 1), v_j) \prec PAL(p_i, v_j) \]

(2) \[ PAL(p_i, prev(p_i, v_j)) \prec PAL(p_i, v_j) \]
Packet Dependencies

Master

PAL Queue

Pkt Queue

Output Commit

Logger
Ordered Release, Single Queue Edition

Recall: PALs are placed in the PAL queue immediately, as they are generated, before their packets progress.
• PALs leave first.
• Data leaves second.
  – Hence: All PALs that a data packet depends on will leave the master before the data packet does.
Ordered Release,
Single Queue Edition

Master

Logger

PAL Queue

Pkt Queue

Output

Commit

?
Ordered Release, Single Queue Edition

Master

PAL Queue

Pkt Queue

"Tail Marker"

Output Commit

Logger
Ordered Release, Single Queue Edition

- The output commit decision:
  - Does the log have every PAL up to and including the tail marker specified at the packet
  - If yes, release.
  - If no, enqueue until the reliability channel retransmits the missing PAL(s).
Ordered Release, Multiqueue Edition

• This algorithm is easily parallelized!
• [Details how in technical report.]

• Key properties:
  – Deadlock-free
  – Low latency
  – Fully parallel
  – No redundancy
FTMB

• A rollback-recovery approach with lightweight logging of determinants

Key techniques:
  – Inband logging
  – Ordered release

• Median latency penalty? 30μs.
Thank You!

Contact:
justine@eecs.berkeley.edu
@justinesherry
Prototype

• Built on Xen 4.2 for checkpointing
• Applied PALs/Recording to 7 Click Middleboxes
• Logger runs over DPDK-Click
• Local testbed: one I/O logger, one master, one replica, one packet-generator.
What is the throughput overhead introduced by FTMB?
Throughput

- MazuNAT
- SimpleNAT
- WAN Opt.
- Monitor
- QoS
- Adaptive LB

Packets Per Second

<table>
<thead>
<tr>
<th></th>
<th>Xen Baseline</th>
<th>Xen + FTMB</th>
<th>Xen + FTMB + Snapshotting</th>
</tr>
</thead>
<tbody>
<tr>
<td>MazuNAT</td>
<td>3e+06</td>
<td>3e+06</td>
<td>3e+06</td>
</tr>
<tr>
<td>SimpleNAT</td>
<td>2e+06</td>
<td>2e+06</td>
<td>2e+06</td>
</tr>
<tr>
<td>WAN Opt.</td>
<td>1e+06</td>
<td>1e+06</td>
<td>1e+06</td>
</tr>
<tr>
<td>Monitor</td>
<td>6e+06</td>
<td>6e+06</td>
<td>6e+06</td>
</tr>
<tr>
<td>QoS</td>
<td>5e+06</td>
<td>5e+06</td>
<td>5e+06</td>
</tr>
<tr>
<td>Adaptive LB</td>
<td>7e+06</td>
<td>7e+06</td>
<td>7e+06</td>
</tr>
</tbody>
</table>

5%-30% overhead, depending on the application.
How long does it take FTMB to replay the original execution?
Replay Time

O(1-2) Wide-Area RTTs
Tunable with Checkpoint Interval
Latency

CDF of Packets

Latency (us)

MazuNAT (Baseline)
with I/O Loggers
w/ FTMB w/o Snapshots
w/ FTMB + Snapshots

Tail comes from periodic VM suspension for snapshots
Is FTMB Efficient?
Our evaluation showed that one backup could host up to 50 virtual middlebox replicas.
Ordered Release, Multiqueue Edition

Recall: dependencies are transitive
Ordered Release, Multiqueue Edition

Master

Two egress queues

Output Commit

Logger
Ordered Release, Multiqueue Edition

Two log buffers

Master

Logger

Output Commit
Ordered Release, Multiqueue Edition

Master

Two sets of sequence numbers

Logger
Ordered Release, Multiqueueue Edition

Two tail markers per packet
Ordered Release, Multiqueue Edition

Two tail markers per packet
Ordered Release, Multiqueue Edition

- The output commit decision:
  - Do all logs have every PAL up to and including the tail marker specified at the packet (for that log)?
  - If yes, release.
  - If no, enqueue until the reliability channel(s) retransmits the missing PAL(s).