Rollback-Recovery for Middleboxes – Public Review

Jeff Chase
Duke University
Durham, NC USA
chase@cs.duke.edu

Middleboxes are important elements in today's networks, and increasingly their functions are virtualized (Network Function Virtualization or NFV). NFV middleboxes run as stateful applications on generic virtual server platforms. These platforms may fail: tolerating these failures is an important open problem. While the distributed systems literature offers many techniques to recover stateful services, it is an open question how to adapt these techniques for middleboxes, which have a particular structure and stringent performance requirements.

The Fault Tolerant MiddleBox (FTMB) paper contributes a recovery solution based on a few guiding assumptions about the problem space and middlebox structure. The inputs to the middlebox are incoming packets; in general, these packets flow through the system and are reinjected into the network at an output port with little modification. The middlebox code runs within a virtual machine (VM) and processes each packet independently, perhaps reading and/or writing a handful of state variables. Platform failures are presumed to halt operation of the VM without corrupting any outputs (fail-stop). The objective of FTMB is to recover the VM state after a failure so that the middlebox can restart where it left off, without disrupting the packet flow or compromising the function of the device.

FTMB uses an elegant adaptation of classical rollback-recovery, with input logging and output commit integrated into network elements (e.g., buffering switches) upstream and downstream from the middlebox. The upstream switch retains a copy of each incoming packet to replay on recovery. The downstream switch buffers each outgoing packet until it can be released (output commit). The middlebox VM snapshots periodically to a passive replica on a standby machine; recovery activates the replica and replays buffered inputs.

The key contribution of the work is to show how to adapt rollback-recovery for NFV middleboxes. Experiments show that the FTMB approach can recover a failed NFV middlebox without packet loss and without triggering congestion avoidance at the connection endpoints. It shows mean packet traversal latencies in the tens of microseconds and throughputs well over one million packets per second.

Achieving this level of performance requires some careful implementation choices. In particular, FTMB focuses on minimizing multi-core coordination overhead. FTMB targets middleboxes that use NICs with flow hashing so that each incoming packet is received and processed entirely by a single core. Because the cores share state variables, it is necessary to log various events (determinants) to recreate the original execution order for deterministic replay. The determinants include synchronization order events: FTMB injects logging annotations for these events into the application code. Each core logs its events independently; the downstream switch executes a parallel algorithm to release each output packet when it has all determinants needed to replay the execution to the packet's output point.

Naturally, it follows that FTMB's overhead varies with the number of state variables and determinants in the application. In addition, FTMB uses VM snapshot facilities to checkpoint the VM state. To recover, it is necessary to replay all input packets since the last snapshot. Therefore, these snapshots must be frequent enough to maintain acceptable input buffering requirements and replay times. There is more research to do on non-disruptive snapshots: although mean latency is excellent, the results show that suspending the middlebox VM for a snapshot is responsible for some uncomfortably high tail latencies.

What's next? The fail-stop assumption is common in the large body of work on recoverable services. Some classes of middlebox faults do not fit this model, so other approaches are needed to address them.