An Industry Standard Benchmark Consortium

Networking Version 2.0 Benchmark Name: QoS

Highlights

- **Based on NetBSD kernel code**

Application

This benchmark simulates the processing undertaken by bandwidth management software used to "shape" traffic flows to meet Quality of Service (QoS) requirements. The system paces the delivery of the packets to the desired speed, based on a set of predefined rules. This shaping is achieved via the use of a variant of the Weighted Fair Queuing (WFQ) algorithm. Random Early Detection (RED) queue management is also supported to provide flow control.

Benchmark Description

The overall structure for the QoS system is as follows (largely based on documentation provided with the Dummynet QoS system):

In the QoS system, egress packets are selected based on rules established during the initialization phase, and passed to two different objects: "pipe" or "queue."

A queue is just a queue with configurable size and queue management policy. It is also associated with a mask (to discriminate among different flows), a weight (used to give different shares of the bandwidth to different flows) and a pipe, which essentially supplies the transmit clock for all queues associated with that pipe.

A pipe emulates a fixed-bandwidth link, whose bandwidth is configurable. The "clock" for a pipe is incremented every iteration in the benchmark. A pipe is also associated with one (or more, if masks are used) queue, where all packets for that pipe are stored.

The bandwidth available on the pipe is shared by the queues associated with that pipe (only one in case the packet is sent to a pipe) according to the WF2Q+ scheduling algorithm and the configured weights.

Egress packets are stored in the appropriate queue, which is then placed into one of a few heaps managed by a scheduler to decide when the packet should be extracted. The scheduler is run once per iteration, and grabs queues from the head of the heaps when they are ready for processing.

There are three data structures defining a pipe and associated queues:

1. `dn_pipe`, which contains the main configuration parameters related to bandwidth
2. `dn_flow_set`, which contains WF2Q+ configuration information
3. `dn_flow_queue`, which is the per-flow queue (containing the packets)

Multiple `dn_flow_set` can be linked to the same pipe, and multiple `dn_flow_queue` can be linked to the same `dn_flow_set`. All data structures are linked in a linear list which is used for housekeeping purposes.

Benchmark Description (continued)

During configuration, the `dn_flow_set` and dn_pipe structures (a `dn_pipe` also contains a `dn_flow_set`) are created and initialized.

At runtime, packets are sent to the appropriate `dn_flow_set` (either WFQ ones, or the one embedded in the `dn_pipe` for fixed-rate flows), which in turn dispatches them to the appropriate `dn_flow_queue` (created dynamically according to the masks).

The transmit clock for fixed rate flows (`ready_event()`) selects the `dn_flow_queue` to be used to transmit the next packet. For WF2Q, `wfq_ready_event()` extracts a pipe which in turn selects the right flow using a number of heaps defined into the pipe itself.

The current dataset sends packets directly to a pipe and utilizes fixed rate flows.

Analysis of Computing Resources

QoS is a memory intensive benchmark – large memory requirements test data cache misses, while long sequences of dependent loads test memory latency.