How to use SO_BUSY_POLL socket option

Overview

There is a new low latency socket option available in RHEL 7.1 called SO_BUSY_POLL that can be used with FIX Antenna C++ to reduce the latency of message receiving. The socket option can be applied to a small number of FIX session connections created in AGGRESSIVE_RECEIVE or AGGRESSIVE_SEND_AND_RECEIVE modes. The number of connections covered should be kept small because each socket in this mode keeps one CPU core completely busy.

SO_BUSY_POLL option should be supported by the appropriate network device driver, RHEL 7.1 documentation lists the drivers that currently support this option, see “6.3.3.1. Configuring busy polling” for actual driver list: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Performance_Tuning_Guide/sect-Red_Hat_Enterprise_Linux-Performance_Tuning_Guide-Networking-Configuration_tools.html

You can find more details about this option here: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/open-source-kernel-enhancements-paper.pdf

Linux kernel support:

Kernel should be built with CONFIG_NET_RX_BUSY_POLL compilation option which is enabled by default in RHEL 7.x, no changes are needed for this OS.

To check if this option is enabled, the following command can be used:

```
# cat /boot/config-3.10.0-229.el7.x86_64 | grep CONFIG_NET_RX_BUSY_POLL
```

Expected result is: “CONFIG_NET_RX_BUSY_POLL=y”

To check if busy poll feature is available for a specific network device, run:

```
# ethtool -k device | grep "busy-poll"
```

Expected result is: “busy-poll: on [fixed]”

Configuration Steps

In order to use this option in Fix Antenna C++ the following steps should be made:

1. Configure the system to use this specific busy poll wait time value:

   ```
   # sysctl net.core.busy_poll=600000
   ```

2. Create an AGGRESSIVE_RECEIVE or AGGRESSIVE_SEND_AND_RECEIVE FIX session

   ```
   Engine::SessionExtraParameters params;
   params.socketPriority_ = Engine::AGGRESSIVE_ RECEIVE_SOCKET_OP_PRIORITY;
   ```

3. Enable busy poll option via Fix Antenna API for this session:

   ```
   params.socketBusyPollTime_ = 600000         //   set 600 milliseconds
   ```

   set aggressive receive delay value to 500ms.

   ```
   params. aggressiveReceiveDelay_ = 500   // set 500 milliseconds
   ```

4. Alternatively you can set the above properties in engine.properties file rather than via the API:

```
Session.target/sender.SocketBusyPollTime = 600000
Session.target/sender.AggressiveReceiveDelay = 500
```

⚠ Note that SO_BUSY_POLL socket option is supported only by FIX Antenna 2.15.0 C++ and higher.

Test Results

A modified Benchmark/Latency test was used in these tests. There were inserted 10ms delays between posts to make the test more realistic and the mess age rate similar to what occurs in practice. Number of messages = 1000.

The test was conducted on RHEL7.1, i7 3.6 GHz CPU, Myricom 10G NICs; packages were passed through 10G cross-over cable, everything was installed on a single machine.

1. **SO_BUSY_POLL is enabled using *SocketBusyPollTime*. This is a new mode.**

   ```
   Session.target/sender.SocketBusyPollTime = 600000
   Session.target/sender.AggressiveReceiveDelay = 500
   System::Thread::sleep(10)
   iterations = 1,000
   ```

   Total latency (nanoseconds): <- time from sending till receiving the message
   MIN: 13549
   MAX: 130952
   AVG: 16396

2. **SO_BUSY_POLL is disabled.**
   Total latency (nanoseconds):
   MIN: 39530
   MAX: 166082
   AVG: 47499

3. **SO_BUSY_POLL is disabled, but “busy poll” is enabled inside of Antenna using installation.**

   ```
   Session.target/sender.AggressiveReceiveDelay = 0
   ```

   Total latency (nanoseconds):
   MIN: 15854
   MAX: 123549
   AVG: 18831

As we can see the enabled SO_BUSY_POLL option is the best choice.

The results of one more test are provided below. It was the test in which messages were sent without pauses. The latency is less when the SO_BUSY_POLL option is enabled though this test is not from the real world.

1. **SO_BUSY_POLL is enabled using *SocketBusyPollTime*.**

   ```
   Session.target/sender.SocketBusyPollTime = 600000
   Session.target/sender.AggressiveReceiveDelay = 500
   iterations = 1,000,000
   ```

   Total latency (nanoseconds):
   MIN: 12153
   MAX: 119149
   AVG: 15602
2. **SO_BUSY_POLL is disabled.**

   ```
   Session.target/sender.AggressiveReceiveDelay = 0
   ```

Total latency (nanoseconds):
MIN: 13968
MAX: 409688
AVG: 17255