Handling network traffic control in your applications

Target-Audience: C-Developers

Target-OS: *BSD/Linux/Cygwin

Time schedule: Tutorial half day

Location: Eurobsdcon 2008, Friday - October 17th 2008

Author: Dirk Meyer, FreeBSD user since 2.1.0, ports-committer since 2001

[dirk.meyer@dinoex.sub.org],[dirk.meyer@guug.de],[dinoex@FreeBSD.org]

Abstract

Demonstrate and show network traffic control in an file transfer application. Using ‘select()’ and buffers to adjust downloads and upload speed, while maintain incoming and outgoing connections and shaping traffic per connection and in total.

Problems of writing portable code, as well as tuning your compiler to the full potential of warnings. Implementing a small file sever over TCP from scratch.

Overview

- 1. Design
- 2. Implementation
- 3. Example Code
- 4. References
- 5. Questions

1. Design

- 1.1. select() API
- 1.2. IPv4 and IPv6
- 1.3. How to slow down sending
- 1.4. How to slow down receiving
- 1.5. Traffic Control Goals

1.1. select() API

Use of the classic select() interface.

- Allows the program to do multiple tasks
- No locking needed
- Very portable
- Avoiding racing bugs in the GCC optimizer
- Non blocking IO
- Drawback: only one CPU used

1.2. IPv4 and IPv6

Use the new API for IPv4 and IPv6 whenever possible.

- Unions for socket_addr
- getnameinfo()
- inet_pton()
- inet_ntop()

- err.h
- sysexits.h

1.3. How to slow down sending

- Count the bytes we sent
- Skip sending if we hit a limit

1.4. How to slow down receiving

- Count the bytes we received
- Skip polling if we hit a limit
  - TCP buffers will do the rest

1.5. Traffic Control Goals

- Maximum speed per transfer
- Overall speed
  - for sending
  - for receiving
- Not blocking the application

2. Implementation

- 2.1. Maximum speed per transfer
- 2.2. Overall speed per send
- 2.3. Overall speed per recv

2.1. Maximum speed per transfer

- Initialize a bucket for each time slots
- Count down bytes till the bucket is empty

2.2. Overall speed per send

- Ring buffer for smooth bandwidth control
- Initialize a buffer slot each second
- Count the sent bytes for the last 4 seconds
- Stop if we reach the average bandwidth

2.2. Overall speed per recv

- Ring buffer for smooth bandwidth control
- Initialize a buffer slot each second
- Count the received bytes for the last 4 seconds
- Stop if we reach the average bandwidth

3. Example Code

- High warning level
- bsd.prog.mk

4. References

[1] http://iroffer.org/
[2] http://iroffer.dinoex.net/

5. Questions

Feedback: Dirk Meyer
[dirk.meyer@dinoex.sub.org],[dirk.meyer@guug.de],[dinoex@FreeBSD.org]