Tweaking TCP’s Timers

Kieran Mansley
Laboratory for Communication Engineering
Context

• Researching user-level TCP for my PhD.
• Focusing on how to implement it efficiently at user level, particularly in a server room.
• Timers are a small but interesting part of that.
Historical Context

• TCP was first specified in the early 1980s
  • OS support for time was poor and costly
  • Networks were slower, so time intervals longer.
• Portability very important, all leads to...

  TCP has weak requirements of the OS
What are TCP Timers?

- *Not* measuring time (usually).
- Enable an action to be performed later.
- Mostly used to deal with inactivity:
  - Timer *set* when activity is expected
  - Timer *cancelled* when activity occurs
  - If timer expires, recovery action is executed.
Delayed Acknowledgments

- TCP sends acks for reliability and flow control.
- Can either be a separate packet or piggyback on a data packet.
- Acks are delayed to encourage piggybacking.
- Timer used to ensure delay is limited.
Delayed Ack Illustration
Timer Ticks Illustrated

[Diagram showing a timer with ticks and two laptops]
Timer Techniques

• How to implement timers?

• Trad. scheme based on 100ms clock ticks
  • Maintain flags in per-connection state.
  • Each tick, check list of connections for timers.

• Modern scheme based on hashed hierarchical timing wheel.
  • Ordered list of timers, use hardware clock to trigger the check and schedule operation.
Problems

• Inaccurate delay of acks: from 0ms to 200ms

• List of connections must be searched each clock tick.

• A busy connection will still regularly send separate ack packets.
Potential Solution (i)

- Change profile of delay:
Potential Solution (ii)

• Use two timer buckets to achieve delay limits:

![Diagram showing the use of two timer buckets to achieve delay limits.](Diagram)
Potential Solution (iii)

- Implementation of buckets:
  - Lazy switch avoids need for scheduling.
  - Timer execution when blocking op encountered.
  - Data thread used for active connections.
  - Time checking done using “rtdsc” counter.
Potential Solution (iv)

- Handle timers for active connections from the data thread.
- Removes need for locking, other than for handing connections between threads.
- No list searching, but...
- Increased set/clear timer complexity.
CPU Usage Tradeoff

- CPU to iterate list (%) measured inline: $0.0000754x + 0.0001191$
- CPU to iterate list (%) measured with 'time': $0.0000692x + 0.001060$
- CPU for bucket scheme (%)

Graph showing the relationship between connection list size and CPU usage (%) for different scenarios.
Summary

- Timers at user level can benefit from a different solution.

- Change the way timers are implemented to:
  - Give guaranteed lower, reduced upper bound;
  - Avoid locking by checking timers in data thread.

- Minor performance issue for current TCPs
  - May be more important in future.
Questions/Comments?

Kieran Mansley
kjm25@cam.ac.uk
http://www-lce.eng.cam.ac.uk/~kjm25

Available as technical report CUED/F-INFENG/TR.487