TCP and Routing Developments in the Linux Kernel

David S. Miller
(davem@davemloft.net)

TCP Congestion Control

- Several new algorithms
- TCP Reno was showing it's age
- Poor handling of long delay paths
- The larger the number of packets in the congestion window the poorer Reno handled things.
- Newer algorithms attempt to recover more gracefully from packet loss within a large window.

TCP Westwood+

- Calculates a bandwidth estimate.
- This estimate is sent through a low-pass filter, identical to the one used for TCP RTO calculations.
- The bandwidth estimate guides congestion window and slow-start threshold decisions.
- This scheme is effective but not as much so as BIC-TCP (discussed later). Note however this could be due to implementation errors.

TCP Vegas

- An algorithm which is quite old
- Most notable feature is that it tries to measure the available bandwidth without packet loss.
- Traditional congestion control works by detecting packet loss and pulling back the send rate accordingly.
- The major flaw of this technique is that it performs poorly when there is traffic in the reverse path.

BIC-TCP

- A very new algorithm, current kernels implement version 1.0
- Version 1.1 released recently, will merge
- Uses a combination of binary search and linear growth to find the appropriate congestion window safely yet fast.
- Provides high levels of TCP fairness compared to other schemes.
- It is the best performer in our testing.

TCP Segmentation Offload

- Basic idea:
  - Send one IP+TCP header template plus huge data portion
  - Card uses template to produce multiple TCP packets
- The advantage:
  - Less cpu processing, building headers
  - Less bus/memory bandwidth usage
- Initial implementation had serious problems, it worked but violated congestion control rules.

Fixing TSO Congestion Control

- Keep track of how many real packets each TSO packet contains.
- Use this 'packet count' in congestion control decisions.
- When ACKs arrive, trim sub-packets from TSO frames and liberate socket send buffer space.
- Never allow TSO packet size to exceed some fraction of congestion window.

Remaining TSO Issues

- Major episodes of loss, or other indications that connection is 'sick' cause TSO to be disabled.
- We can remove this disabling only by adding sophisticated Selective ACK tracking state for the sake of TSO frames. Re-segmentation is too expensive and not an option.
- TSO, even with current fixes, can still be a bit too bursty. Not an easy problem to solve.

TCP Routing

- Current layered architecture:
  - FIB Rules, which point to
  - FIB Nodes, which are used to generate
  - Routing Cache entries
- FIB Node layer algorithms are showing their age, and need complexity improvement.
- When Routing Cache meets DoS level traffic patterns, performance is limited by two things:
  - Routing Cache recycling performance
  - FIB Node lookup complexity

State of Routing Lookups

- Longest Matching Prefix is a well researched topic.
- Unfortunately, it's a well patented topic too :-(
- For most solutions, one must pick two out of:
  - Fast lookup performance
  - Fast table update
  - Low memory usage
- Tree Bitmap is the one exception currently and is considered state of the art right now.

Routing Lookup Papers

- 'Survey and Taxonomy of IP Address Lookup Algorithms' by Miguel A. Ruix-Sanchez et al.
- 'Tree Bitmap: Hardware/Software IP Lookups with Incremental Updates' by Will Eatherton, George Varghese and Zubin Dittia
- 'Scalable High-Speed Prefix Matching' by Marcel Waldvogel et al.
- 'nCRT and Bonsai: Two Fast Longest Prefix Match Algorithms' by Abhishek Singh et al.

Patents patents patents...

- Tree Bitmap patented by Cisco, an attempt was made to obtain permission to use in GPL code but this has failed.
- Binary Search on Hash Tables is patented by the University of Washington and work is nearly complete to get permission to use this algorithm in GPL code.
- LCP-Trie algorithm we have full permission to use already, Robert Olsson working on this.

Preliminary Issues in Routing

- Current FIB Node lookup layer was not friendly enough to add new algorithms
- Work proceeds to try and build a pluggable route lookup architecture
- Major issue is separation of pure destination IP address longest matching prefix lookup from other details of routing:
  - TOS and priority sub-keys
  - FIB Semantics

Current Plan

- Complete routing lookup abstraction
- Integrate Binary Search and LPC-Trie implementations
- Benchmark, test, and optimize
- Once mid-level lookup complexity is minimized, reanalyze performance characteristics of routing cache.
- Continue to follow research on LMP algorithms, and perhaps make our own :-)

Binary Search on Hash Tables

- 32 hash tables, one for each ipv4 prefix length
- 'marker' nodes are added to the hash tables to guide the binary search
- Seeing a 'marker' tells the lookup algorithm which direction to take the binary search
- Lookup performance is great, for ipv6 too
- Table updates are very complex, we believe it can be made to perform acceptably, however.
- Memory usage is not that bad.

LPC Trie

- Level and Path Compressed Trie
- LMP search is a two dimensional problem, each node traversal in LPC search attempts to make progress in both dimensions at same time.
- Lookup complexity is O(W/k) where 'W' is width in bits of the keys (32 for IPV4) and 'k' is the Trie stride chosen by the algorithm.
- Table update and memory complexity is high

Credits

- Mashimaro, official mascot of Linux networking development
- Stephen Hemminger for work on TCP congestion control integration
- Robert Olsson and Jamal Hadi Salim for routing and traffic classification work
- Yoshifuji and entire USAGI team for ipv6 work
- Herbert Xu for keeping IPSEC in shape
- Alexey Kuznetsov, he gave us the foundation

Thank You

- Tetsuro Yogo for inviting me here and making this trip as comfortable as possible.
- All conference organizers and sponsors for making this event possible
- Linus Torvalds, for giving us something to be talking about today