SYNrace: Load-adaptive Multi-path Routing Without Collecting Statistics

Arne Schwabe <arne.schwabe@uni-paderborn.de>
Holger Karl <holger.karl@uni-paderborn.de>
Data centers are changing

- Traditional single tree replaced by CLOS like topologies
- Redundant connections of servers
- Bundling multiple links for high data rates
- Multiple redundant paths from host to host
Challenges

• Goal: Use as much as capacity as possible
• Just use all paths in parallel

• Good Idea, does not work

• Problem: TCP
  • Does not like reorder
TCP Reorder Solution

- Pick a path per flow
  - 5 tuple identifies flow (IP Protocol, Source/Destination IP address and Port)
  - Solves TCP reordering problem

- New Problem: How to pick a path?
ECMP

- Just assume there are any many flows
- Assume “uniform” size and length distributions
- Then just pick a path at random
  - Equal cost multipath
- A lot of assumptions
- Still works relatively good in many cases
- Does not need statistics
Local knowledge

- Use actual usage of links
- Each switch uses least congested links to destination
  - Sounds good, performs terrible
  - Often least congested => Congestion on one of the next hops
- Need a global strategy
Global view

- Simple idea: Monitor link utilisation
- Pick least utilised link
- Need statistic collection and distribution
- Centralised (e.g. Hedera)
  - Scalability is problematic
  - Low statistics collection interval (1-10s)
  - Latency from controller to network switches
- Decentralised (e.g. Conga)
  - High number exchange message
  - Massive use of specialised hardware (custom silicon)
A way out

- Goals for SYNRace
  - Standard hardware
  - No central entity
  - No use of fast ageing statistics

- Absolute utilisation unimportant
- Only need to know the best path

- Search for different methods/effect to infer the best path
Latency as Proxy for Utilisation

- Indirectly measure the effect of utilisation
- TCP fills up buffers
- Increases latency on the path
- In summary: high utilisation $\Rightarrow$ Higher latency
Use Buffers to detect quickest path

- On demand measurement
- Needs only three probe packets
Challenges to solve

- Influence of other sources of delay?
- Generation of probe packets
- OpenFlow implementation
Delay sources

- Buffering: 10 packets (1500 Byte)
  - 1.2µs – 120µs

- Asymmetrical paths
  - Unproblematic in data centers, problematic in WANs

- Cable length
  - 200m difference: 1µs

- Different switches
  - Data centers are homogenous
  - Especially for a set of paths between two hosts
Implementation

- Straw man solution: Send one probe per path
- Problem first probe more likely to win
- No probe should be more likely to arrive first
- Duplicate packets on the way
OpenFlow Implementation

- Use the SYN packet as Probe packet
- Piggypack best path on SYN/ACK response/reverse probe
- Splitting flow packets is easy: `setOutport=1,2`
- Recording path a probe has taken: `add label=$ownid` (simplified)
- Preinstall all rules
  - `PACKET_IN` for probe packets screws up latency
Results (big file transfers)

![Graph showing flow completion time vs. ECDF for different loads and routing methods.](image-url)
Results (realistic traffic)

![Bar chart showing flow completion for different traffic sizes and comparison between SynRace and ECMP]

- **Flow completion** is shown for different traffic sizes: `<1kB`, `1kB-10kB`, `10kB-100kB`, `100kB-1MB`, and `>1MB`.
- The chart compares the performance of SynRace and ECMP for each traffic size.

The chart indicates that SynRace generally outperforms ECMP across all traffic sizes, with higher flow completion values.
Conclusion

- Can use buffering delay to infer network utilisation
- SDN allows to create sophisticated probe requests
- SYNrace is a useful tool to complement routing strategies
- As always technical details are in the paper