Jellyfish: Networking Data Centers Randomly

Paper Note
Junzhi Gong

September 22, 2018

1 Backgrounds

- A well provisioned data center network is critical 1) to ensure that servers do not face bandwidth bottlenecks to utilization; 2) to help isolate services from each other; 3) and to gain more freedom in workload placement, rather than having to tailor placement of workloads to where bandwidth is available.
- One crucial problem that these designs encounter is incremental network expansion, *i.e.*, adding servers and network capacity incrementally to the data center.
- Current structure of network topologies hinder incremental expansion, e.g., the fat-tree topology.

2 Target

Design a random network interconnect. Additional componentsracks of servers or switches to improve capacity can be incorporated with a few random edge swaps. The design naturally supports heterogenety, allowing the addition of newer network elements with higher port-counts as they become available, unlike past proposals which depend on certain regular port-counts. The topology also allows construction of arbitrary-size networks, unlike topologies discussed above which limit the network to very coarse design points dictated by their structure.

3 Challenges

- Routing (schemes depending on a structured topology are not applicable) and congestion control.
- Physical construction.
- Cabling layout.

4 Solution

- **Jellyfish topology.** The Jellyfish approach is to construct a random graph at the top-of-rack (ToR) switch layer.
  - *Construction.* In the simplest case, every switch has the same number of ports (k) and servers ($k-r$). In this case, the network is a *random regular graph*, which we denote as RRG(N, k, r).
  - *Intuition.* The two key goals are bandwidth and flexibility. Achieving flexibility is simple. Jellyfish has high end-to-end throughput because it has low mean path length.
- **Jellyfish topology properties.** Key findings are:
  - Jellyfish can support 27% more servers at full capacity than a (same-switching-equipment) fat-tree at a scale of ¡900 servers.
  - Jellyfish's network capacity is ¿91% of the best- known degree-diameter graphs.
  - Paths are shorter on average in Jellyfish than in a fat-tree, and the maximum shortest path length (diameter) is the same or lower for all scales we tested.
  - Incremental expansion of Jellyfish produces topologies identical in throughput and path length Jellyfish topologies generated from scratch.
  - Jellyfish provides a significant cost-efficiency advantage over prior work (LEGUP) on incremental network expansion in Clos networks.
  - Jellyfish is highly failure resilient, even more so than the fat-tree.

  For incremental expandability, Jellyfish only needs to choose a random link, break it, and then rewire it via two ports in the new ToR switch.
- **Routing and congestion control.** There are two layers which can affect performance in real deployments: routing and congestion control.
  - *ECMP is not enough.* ECMP performs poorly for Jellyfish, not providing enough path diversity. After using k-shortest paths, each congestion control protocol works as least as well for Jellyfish as for the fat-tree.
  - *k-shortest-paths with MPTCP.* Each switch needs to maintain a routing table containing for each other switch, k shortest paths. Choices include OpenFlow, SPAIN, and MPLS.
- **Physical construction and cabling.** Key considerations in data center wiring include:
  - Number of cables.
  - Length of cables.
  - Cabling complexity.

In deployment of topologies, there are several common concerns:

- *Handling wiring errors.* Given Jellyfish's sloppy topology, a small number of miswirings may not even require fixing in many cases. Nevertheless, fixing miswirings is relatively inexpensive.
- *Small clusters and CDCs.* The key observation is that in a high-capacity Jellyfish topology, there are more than twice as many cables running between switches than from servers to switches. Thus, placing all the switches in close proximity to each other reduces cable length, as well as manual labor.
- *Jellyfish in massive-scale data centers.* In this setting, as the number of containers grows, most Jellyfish cables are likely to be between containers. To address this problem, Jellyfish applies random graph between containers, instead of switches.

5 Related Work

Several recent proposals for high-capacity networks exploit special structure for topology and routing.

- Folded-Clos (or fat-tree) designs. (Fat-tree, VL2, and Portland)
- Designs that use servers for forwarding. (BCube, DCell, and MDCube)
- Designs using optical networking technology. (Helios and c-Through)
- High performance computing literature has also studied carefully-structured expander graphs.

However, none of these architectures address the incremental expansion problem.

6 Summary of Key Metrics for Topology

- Bandwidth and capacity.
- Routing and congestion control.
- Physical construction and cabling.
- Incremental expansion.
- Fault tolerance.