Name of the Client: RtBrick
Publication: Voice&Data
Date: 31 August 2020
URL: https://www.voicendata.com/end-router-know-it/
Page: 1 of 3

End of the router as we know it

By Hannes Gredler

Since its beginning in the 1980s, network routers have evolved significantly. The greatest leap in development was observed when the World Wide Web was introduced. Network providers were suddenly faced with the challenge of having to keep up with the rapid introduction of internet services and the demand for bandwidth-intensive applications. In order to meet the demand, they invested enormous amounts in networks, which they had great difficulty in refinancing. They asked network equipment manufacturers to help them build value-added services, which in the years that followed led to a flood of new features that are now available for almost every router.

Critics argue that by adding new features but not removing obsolete ones, the cost of the routers has risen so much that they are no longer in proportion to the actual benefits. James Hamilton, VP and Distinguished Engineer at Amazon Web Services expressed this observation as follows: "The network is anti-Moore." In this context, here are some functions that have become redundant in the course of the current technical developments.

Hardware-buffer

In their Paper Sizing Router Buffers (2004), researchers at Stanford University, describe their observations that buffers with a depth of up to 2000 ms are clearly suitable for services with low bandwidth and data stream diversity. But at speeds of 10 GBit per second and diversity of up to 10 million data streams, the advantages of buffering become extremely doubtful with a typical internet backbone connection.

The benefit is also questioned by the fact that there is no signalling to the Transmission Control Protocol (TCP) layer during "buffering". Most routers, however, still support a buffer depth of over 100 ms for 100 Gbps switching. A simple calculation shows that 1.25 GB DDR4-RAM is required for every 100 GBit/s port in a given router.

Exactly this DDR4 RAM, sometimes the most expensive form of RAM, makes the buffer the biggest cost driver for today's router hardware. Not only is it needed for a function that rarely works with today's internet backbone usage patterns. At the same time, it must be implemented as off-chip memory, which increases the cost of external I/O, power consumption, and cooling.

Hardware-forwarding tables

The second-largest cost factor for a router's data level is the size of its forwarding table. Modern hardware can store approximately two million forwarding entries in its IPv4, IPv6, and MPLS forwarding tables. The design of this forwarding engine is based on two basic ideas.

- One, a single forwarding entry can have a high data consumption. In fact, a single prefix can take up the entire bandwidth of a connection. This is still relevant today as content delivery networks and Web 2.0 companies direct a large part of their Internet traffic to just a few IP prefixes.
- Two, all forwarding entries can carry the full connection bandwidth. This no longer applies today. Traffic per prefix on the internet has increased exponentially, which means that the chip design for data forwarding has to be radically revised. Instead of treating each IP forwarding entry equally, a memory cache hierarchy is much more practical. This is comparable to today's computer designs: a tiered memory hierarchy with different levels of memory, each with different speeds, with adequate costs.

Modern IP routers still work at only one storage level, on the assumption that every forwarding entry must be fast. However, if you take a closer look at the analysis of the real backbone traffic data it is no longer the case. In fact, the forwarding tables for contemporary practical use are now 10 times oversized.

The good news is that the hardware can be easily optimized. Customers only need to clearly formulate what is usually required so that the next generation of forwarding hardware can be

adapted accordingly. Network software, on the other hand, is a very special kind of problem.

Software features

It is difficult to say which software functions are actually redundant and which are not, since these depend on the individual needs of the telecommunications company. Over time, manufacturers have developed a wide range of features at the request of network operators and inserted them directly into the code. However, this procedure makes it impossible to deactivate certain functions after implementation. This can quickly become a cost factor as network operators have to pay for these features even when they are not being used. At the same time, when a new function is introduced, interference tests must be carried out every time for all existing functions – even for those that are not required.

The background to this is that the software for the router was previously programmed as a monolithic system, whereby new functions were closely linked to the underlying infrastructure. The removal of such functions from the code base can be as complex as their original development. At the same time, the hurdle for a functional expansion increases with each new feature.

In today's highly competitive telecommunications market, however, service providers rely on their systems being agile, easy to maintain, and tailored to their needs and those of their customers. Accordingly, router system manufacturers have to develop a new, more cost-effective approach that enables network operators to flexibly and smoothly manage, update and, if necessary, remove functions.

The solution = Disaggregated Systems + Distributed SDNs + Modular Code

The first step in the right direction is the desegregation of hardware and software. This allows network operators to choose between different bare metal switches and validated network software for them, and to take advantage of the latest chip generations and thereby strengthen their innovation potential. However, this means that the responsibility for function management lies entirely with the network operators. A distributed software-defined network (SDN) offers the ideal conditions for this. It combines the advantages of an SDN with the advantages of a distributed control level and thus enables smooth management of the software.

To ensure that functions can be added and removed smoothly, the code should be structured in such a way that it can be put together from individual blocks of code. These should be able to be supplemented or taken out again as desired, whereby there should be no interdependencies between the blocks. An "internet-native" approach can help here. Independent micro services are used, which are carried out in containers. If a new function or an update is required, a corresponding container is supplied by the software developer, which updates or adds the respective feature within milliseconds and without interrupting the service. In this way, route processing, updating, and restarting are 20 times faster than with conventional router operating systems. If open interfaces are also available, network operators can even develop and implement their own functions.

Traditional routers and dynamic control systems are challenged by new concepts such as desegregation and distributed SDNs. These promises significantly faster implementation, automated control, and a shorter time to market. In order for future router designs to meet these challenges, fundamentally new router hardware and software must be developed, and modern software architectures and paradigms introduced.

The author Hannes Gredler is the Founder and CTO of RtBrick

###