Advanced Topics in IP Networks
Lecture 02

Lecture by Dr. Anat Bremler-Barr
Typeset by Steven Karas

2018-10-25
Last edited 20:59:48 2018-10-25

Disclaimer These notes are based on the lectures for the course Advanced Topics in IP Networks, taught by Dr. Anat Bremler-Barr at IDC Herzliyah in the fall semester of 2018/2019. Sections may be based on the lecture slides prepared by Dr. Anat Bremler-Barr.

1 Agenda

- Recap
- Network Algorithmics
- Routers

2 Recap

Software Defined Networking pulled the control from the individual switch/router to a central authority that would send out routing instructions to individual components. This separated the "control plane" from the "data plane" that handles actual packet routing. This started in or around 2009.

Network Function Virtualization was the second revolution in IP Networks, where specialized middleboxes were consolidated into shared hardware that runs multiple "appliances".

3 How to read papers

As part of the course, we will be required to read papers, some in order to answer homework questions. The point is to teach us how to conduct research on a small scale.

The suggested approach is the 3 pass approach. In this approach, the first pass only covers the abstract and the section headers to determine relevancy and interest. The second pass skips over proofs and other details to get the gist of the paper. The final pass includes reconstructing the proofs to build a critique of the paper.

4 Network Algorithmics

This subfield deals with finding performant solutions that are easy to use. In general, endpoint solutions can be slower, but need to cover more layers, and are more complex. Network solutions on the other hand, need to handle

Units Network speeds are always measured in terms of bits or in well established physical line speeds. E.g. OC-768 is 768 times faster than a single optical connection (OC), which is equal to 40Gbps.

Ethernet frames have 24 header bytes. IP packets have 20 header bytes. TCP packets have 20 header bytes.

4.1 Example: Buffer Overflow Detection

We can discover suspicious traffic by counting the character frequency of packets. A naive approach will count character frequencies, but this is slow because of the second pass over the character array. A more performant approach will read, increment, and track the maximum threshold.

$$\text{PacketLength} \cdot \text{Threshold} > \text{Count} \iff \text{PacketLength} > \frac{\text{Count}}{\text{Threshold}}$$

However, this cannot report which character caused the packet to over a threshold, or if there were multiple. This approach is still not performant enough, because if we store the thresholds in a separate array, then we perform 2 reads and 1 write for each byte in the packet.

We can pack the threshold and the count into a single record. This is great because we're now doing 1 read and 1 write, but we're still doing a division for each byte.

We generalize the thresholds to be inexact and use a shift-count and shift left instead of divide. This is great because we're doing bit ops which are fast, but we still need to go over the entire array to zero the counts after each packet.

To solve this, we add another field to the record to track the generation of the count. If the current generation is different than the one recorded, it's treated as a 0. This can be a single bit, and only adds a small cost for the branch in each byte (and has branchless solutions).

As a rule of thumb, for network algorithmics, we are concerned with the number of main memory accesses and with reducing that to a constant factor (which also matters a lot!).

4.2 Hardware

Link speeds largely followed Moore's Law, and demand followed it also. However, Moore's law no longer applies to CPUs, but rather to the number of cores.

Router capacity is limited by the memory speed, which has not significantly improved in the last 10 years. This is why we care about memory access more than CPU operations (1 memory access is approximately equivalent to 100 operations)

Registers are very small (32-64 bits), but take 0.5-1 nsec to access. On-chip SRAM is limited in size, but costs approx 1-2 nsec to access. Off-chip SRAM takes around 5-10 nsec ot access. DRAM can take between 40-80 nsec to access. SRAM is often used to cache accesses to DRAM. TCAM is a type of memory than is similar to SRAM, and can be found in almost every router today.

DRAM requires decoding, and if we split this into 2 decoders we can reduce the needed gates from $O(n)$ to $O(\sqrt{n})$, however this increases decode delay. External memory access can often be batched to read a large chunk of memory at once. So designing algorithms with spatial locality for memory access is a good idea.

Pipelining splits jobs into chunks that can be run in parallel. This increases throughput, but is generally a tradeoff with latency. The industry tends to be interested in throughput, whereas the user is typically more interested in latency.

Example: Pipelined Flow ID lookups Network sessions are called "flows". If we assume a requirement of assigning a 96-bit packet id for accounting purposes at OC-48 speeds.

$$2.5\,\text{Gbps} = 2.5 \cdot 2^{30}/40\,\text{byte} \cdot 8 = 7812500\,\text{pps}$$

$$\text{Time per packet} = 128 \cdot 10^{-9} = 120\,\text{nsec}$$

If we use a balanced binary tree, then we need lots of memory, and the tree height is 20 layers. With a DRAM cycle of 50 nsec, we would need 1000 nsec.

To solve this, we can take RDRAM (with 16 banks) and 60 nsec delay, and store nodes of height i in bank i. This gives us a $16 \cdot 60 = 960$ nsec latency per packet, but throughput of 1 flow per 60 nsec. Worse, we can only support 2^{16} flows.

To solve this, we can use RAMBUS in page variation mode, which allows 3-way branching and can support 3^{16} flows.

5 Routers

Backbone (non SDN, non virtualized) routers are large devices with a slotted chassis. Interface linecards go into the slots, which handle the actual traffic, and have a builtin control unit. Early routers were essentially general purpose computers, whereas modern routers resemble supercomputers with specialized hardware. Modern routers can reach 400Gbps over 32 linecards.

Routers are meant to forward packets to the next network. It does this by looking at the headers. It may adjust some of these headers (such as the ttl), and uses the destination address and other fields to decide where to forward the packet to.

5.1 History of Router Architectures

1st generation routers were general purpose hardware with multiple NICs, where the entire router needed to work at the backplane speed.

2nd generation routers were general purpose hardware with specialized NICs that handled most of the routing on their own. However, they still needed to communicate between them and this shared bus was a bottleneck.

3rd generation routers were specialized hardware that used a switched backplane. A crossbar switch, then the traditional solution, exploits parallelism, but requires a scheduling algorithm. This switching fabric must work at n times the line rate, and because out lines may have more traffic than the line rate traffic buffers are needed.

If we put the switching fabric after the traffic buffers, this can reduce the required size of the buffers, but can cause head of line blocking. This is solved by keeping a buffer per output line, and having separate arbiters for each outline.

4th generation routers are multi-chassis, multi-rack, and use optics for internal lines/switching.

Next generation routers may use optical switching fabric.

5.2 Forwarding

Identifiers can usually be broken down into 3 different types: host names, IP addresses, and MAC addresses.

We skipped a few slides due to time constraints.

Pre-1994, addresses were allocated in a classful manner (only in /8s, /16s, and /24s). This is extremely inefficient, and was replaced with CIDR (Classless Inter-Domain Routing). This meant that we need to maintain a list of all the allocated prefixes and the next hops for them. There are approximately 740000 prefixes in the global routing table, as of October 20, 2018. This can be consolidated into approximately 400000 routes using CIDR.

IANA is responsible for allocating IP addresses to RIRs. The RIRs then allocate to various providers, who can allocate smaller prefixes to organizations.

References

[1] Mark Crovella and Balachander Krishnamurthy. *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2006.

[2] James F. Kurose and Keith Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

[3] George Varghese. *Network Algorithmics,: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.