NETWORK ALGORITHMICS

Network algorithmics is the use of an interdisciplinary systems approach, seasoned with algorithmic thinking, to design fast implementations of network processing tasks at servers, routers, and other networking devices

Network algorithmic is a fundamental way of crafting solutions to internet bottlenecks. It provides an insight to design different implementations for specific contexts and to deal with new bottlenecks that will undoubtedly arise in the changing world of networks. Network algorithmics goes beyond the design of efficient algorithms for networking tasks, though this has an important place. In particular, network algorithmics recognizes the primary importance of taking an interdisciplinary systems approach to streamlining network implementations. The problems addressed by network algorithmics are fundamental networking performance bottlenecks. The solutions advocated by network algorithmics are a set of fundamental techniques to address these bottlenecks.

NETWORK BOTTLENECKS

The main problem is how to make networks *easy to use* while at the same time realizing the *performance* of the raw hardware. Ease of use comes from the use of powerful network abstractions, such as socket interfaces and prefix-based forwarding. Unfortunately, without care such abstractions exact a large performance penalty when compared to the capacity of raw transmission links such as optical fiber. To study this performance gap in more detail we examine two fundamental categories of networking devices, *endnodes* and *routers.*

Endnode Bottlenecks

Endnodes are the endpoints of the network. They include personal computers and workstations as well as large servers that provide services. Endnodes are specialized toward computation, as opposed to networking, and are typically designed to support *general-purpose* computation.Thus endnode bottlenecks are typically the result of two forces: *structure and scale*.

• *Structure:* To be able to run arbitrary code, personal computers and large servers typically have an operating system that mediates between applications and the hardware. To ease software development, most large operating systems are carefully structured as *layered software*; to protect the operating system from other applications, operating systems implement a set of *protection mechanisms*; finally, core operating systems routines, such as schedulers and allocators, are written using *general mechanisms* that target as wide a class of applications as possible. Unfortunately, the combination of layered software, protection mechanisms, and excessive generality can slow down networking software greatly, even with the fastest processors

• *Scale:* The emergence of large servers providingWeb and other services causes further performance problems. In particular, a large server such as aWeb server will typically have thousands of concurrent clients. Many operating systems use inefficient data structures and algorithms that were designed for an era when the number of connections was small.

Bottleneck	Cause	Sample solution
Copying Protection	Protection, structure	Copying many data blocks without OS intervention e.g RDMA
Context switching	Complex scheduling	User-level protocol implementations Event driven web servers.
System calls	Protection , structure	Direct channels from applications to drivers (e.g., VIA)
Timers	Scaling with number of timers	Timing wheels
Demultiplexing	Scaling with number of endpoints	BPF and Pathfinder
Checksums (CRC)	Generality Scaling with link speeds	Multibit computation
Protocol code	Generality	Header prediction

Figure above previews the main endnode bottlenecks , together with causes and solutions.

The first bottleneck occurs because conventional operating system structures cause packet data *copying* across protection domains; the situation is further complicated in Web servers by similar copying with respect to the file system and by other manipulations, such as checksums, that examine all the packet data. The second major overhead is the *control overhead* caused by switching between threads of control (or protection domains) while processing a packet.

Networking applications use timers to deal with failure. With a large number of connections the timer overhead at a server can become large. Similarly, network messages must be demultiplexed (i.e., steered) on receipt to the right end application. Finally, there are several other common protocol processing tasks, such as buffer allocation and checksums.

Router Bottlenecks

The techniques previewed apply equally well to any other network devices, such as bridges, switches, gateways, monitors, and security appliances, and to protocols other than IP, such as

Unlike end nodes, these are special-purpose devices devoted to networking. Thus there is very little structural overhead within a router, with only the use of a very lightweight operating system and a clearly separated forwarding path that often is completely implemented in hardware. Instead of structure, the fundamental problems faced by routers are caused by *scale* and *services*. Fiber Channel.

• *Scale:* Network devices face two areas of scaling: *bandwidth scaling* and *population scaling*. Bandwidth scaling occurs because optical links keep getting faster, as the progress from 1-Gbps to 40-Gbps links shows, and because Internet traffic keeps growing due to a diverse set of new applications. Population scaling occurs because more endpoints get added to the Internet as more enterprises go online.

• *Services:* The need for speed and scale drove much of the networking industry in the 1980s and 1990s as more businesses went online (e.g., Amazon.com) and whole new online services were created (e.g., Ebay). But the very success of the Internet requires careful attention in the next decade to make it more effective by providing guarantees in terms of performance, security, and reliability. After all, if manufacturers (e.g., Dell) sell more online than by other channels, it is important to provide network guarantees—delay in times of congestion, protection during attacks, and availability when failures occur. Finding ways to implement these new services at high speeds will be a major challenge for router vendors in the next decade.

Bottleneck	Cause	Sample solution
Exact lookups	Link speed scaling	Parallel hashing
Prefix lookups	Link speed scaling Prefix database size scaling	Compressed multibit tries
Packet classification	Service differentiation Link speed and size scaling	Decision tree algorithms Hardware parallelism (CAMs)
Switching	Optical-electronic speed gap Head-of-line blocking	Crossbar switches Virtual output queues
Fair queueing	Service differentiation Link speed scaling Memory scaling	Weighted fair queueing Deficit round robin DiffServ, Core Stateless
Internal bandwidth	Scaling of internal bus speeds	Reliable striping
Measurement	Link speed scaling	Juniper's DCU
Security	Scaling in number and intensity of attacks	Traceback with bloom filters Extracting worm

		signatures

Figure above previews the main router (bridge/gateway) bottlenecks covered in this book, together with causes and solutions.

First, all networking devices forward packets to their destination by looking up a forwarding table. The simplest forwarding table lookup does an exact match with a destination address, as exemplified by bridges. Unfortunately, population scaling has made lookups far more complex for routers. To deal with large Internet populations, routers keep a single entry called a *prefix* (analogous to a telephone area code) for a large group of stations. Thus routers must do a more complex *longest-prefix-match* lookup. Many routers today offer what is sometimes called *service differentiation*, where different packets can be treated differently in order to provide service and security guarantees. Unfortunately, this requires an even more complex form of lookup called *packet classification*, in which the lookup is based on the destination, source, and even the services that a packet is providing. Next, all networking devices can be abstractly considered as switches that shunt packets coming in from a set of input links to a set of output links. Thus a fundamental issue is that of building a high-speed switch. This is hard, especially in the face of the growing gap between optical and electronic speeds. The standard solution is to use parallelism via a *crossbar switch*. Unfortunately, it is nontrivial to schedule a crossbar at high speeds, and parallelism is limited by a phenomenon known as *head-of-line blocking*. Worse, population scaling and optical multiplexing are forcing switch vendors to build switches with a large number of ports (e.g., 256), which exacerbates these other problems. While the previous bottlenecks are caused by scaling, the next bottleneck is caused by the need for new services. That raises an issue of providing performance guarantees at high speeds ,for which the issue of implementing so-called QoS (quality of service) mechanisms is to be studied. Another bottleneck that is becoming an increasing problem: the issue of bandwidth within a router. It describes sample techniques, such as striping across internal buses and chip-to-chip links.

CHARACTERISTICS OF NETWORK ALGORITHMICS

Network algorithmics is an *interdisciplinary* approach because it encompasses such fields as architecture and operating systems (for speeding up servers), hardware design (for speeding up network devices such as routers), and algorithm design (for designing scalable algorithms). Network algorithmics is also a *systems* approach, because it is described in this book using a set of 15 principles that exploit the fact that routers and servers are systems, in which efficiencies can be gained by moving functions in time and space between subsystems.