Network Protocols Designed for Optimizability

Jennifer Rexford
Computer Science Department
Princeton University
jrex@cs.princeton.edu

***Abstract*— Designing a communication network is a challenging task that requires selecting a network topology, as well as specific protocols and mechanisms, to meet current and future demands. Equally daunting, managing the network requires tuning these protocols and mechanisms over time in response to changing constraints and conditions. However, the protocols underlying today's data networks, such as the Internet, were not designed with manageability in mind. As a result, managing these networks is, at best, a black art practiced by an increasingly overwhelmed community of engineers. Optimization tools can help the operators tune the protocol configuration and diagnose performance problems, based on measurements of the underlying network. However, many of the existing protocols were not designed with optimization in mind, leading to computationally difficult optimization problems even for the simplest of objective functions. In this position paper, we argue that future protocols should be designed with optimization in mind from the beginning, to simplify the process of configuring the protocols and diagnosing performance problems.**

I. Introduction

A data network like an Internet Service Provider (ISP) backbone consists of a collection of routers and links, where the routers forward data packets and run routing protocols. Routers implement a wide variety of functions in the *data plane* at the level of individual packets, including filtering (based on access control lists), forwarding (based on forwarding tables), buffering (based on buffer management policies), and link scheduling (based on configurable weights or priorities). The routers also implement various *control plane* functions, such as the routing protocols that compute the forwarding tables that determine the paths the packets follow from one end of the network to the other.

A. Network Performance Depends on Configuration

Managing an IP network involves monitoring the routers to track changes in the topology and traffic loads, and adjusting the configuration of the routers accordingly, to satisfy network-wide objectives. Identifying an effective change to the network configuration requires accurate models of the underlying protocols, as illustrated in Figure 1. For example, the flow of traffic through an ISP backbone typically depends on routing protocols like Open Shortest Path First (OSPF) [1] and Intermediate System–Intermediate System (IS-IS) [2] that compute paths based on configurable link weights. Figure 2 shows an example network topology with an integer weight on each link. Each router learns the weighted graph and runs Dijkstra's shortest-path algorithm to compute a forwarding table that directs each packet to the next hop along a shortest path. The network operator influences how the routers forward traffic indirectly, through the setting of the link weights.

As another example, consider the way the Transmission Control Protocol (TCP) responds to congestion

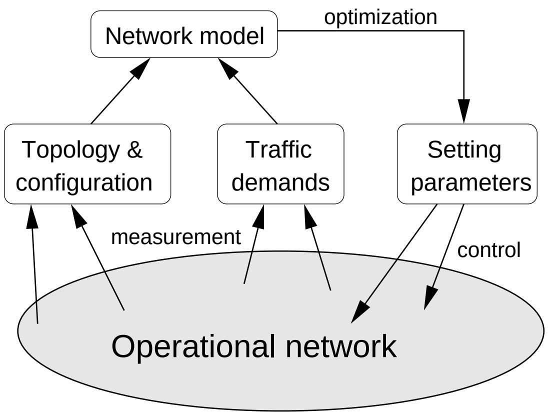

Fig. 1. Setting configurable parameters based on a network model

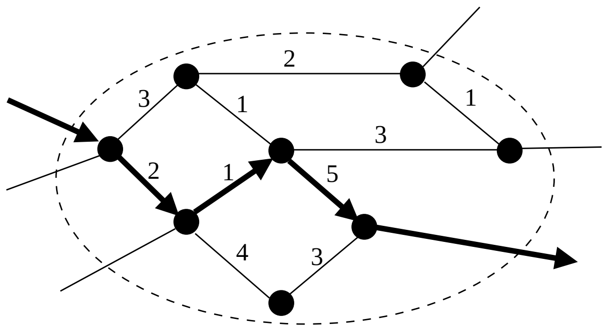

Fig. 2. Shortest path routing based on integer link weights

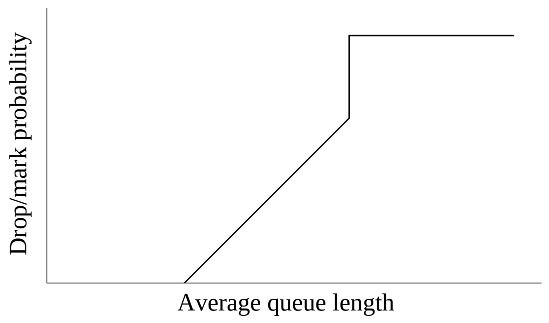

Fig. 3. RED drop probability as a function of average queue length

inside the network. When a link becomes overloaded, the router starts storing the packets in a queue until they can be served. Eventually, as the buffer becomes full, some incoming packets must be discarded. End hosts running TCP detect lost packets and infer the presence of congestion. A TCP sender responds by decreasing the sending rate, in the hope of alleviating the congestion. Rather than waiting until the buffer is full to provide feedback, the Random Early Detection (RED) [3] mechanism probabilistically drops (or marks) packets as the queue builds. Figure 3 shows an example of how the drop probability increases with the average queue length. RED has a number of configurable parameters, such as a the minimum and maximum queue-length thresholds for probabilistic dropping, the maximum drop probability, and the averaging interval for computing the queue length. In a network running RED, the operators need to select appropriate settings for these parameters.

B. Challenges of Selecting Good Configuration Settings

Network performance is very sensitive to how the operators set the tunable parameters. However, selecting good parameter settings is very challenging in practice, for four main reasons:

Lack of predictive models for how the parameter settings affect performance: In some cases, there is no known model for predicting how the parameter settings affect performance. For example, although simulation studies have provided insight into the behavior of RED, algorithms for setting the configurable parameters remain elusive. This makes it difficult for network operators to use RED in practice, short of following general guidelines for how to tune the parameters. The appropriate RED parameters may depend on a variety of factors, including the number of active data transfers and the distribution of round-trip times of these traffic flows, but the exact relationship between the properties of the traffic and the parameter setting is not known. In other cases, accurate models exist but are extremely complicated [4].

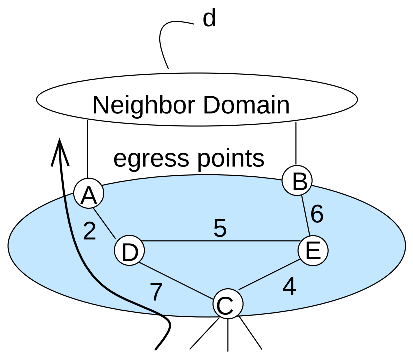

Fig. 4. Router C directs traffic to destination d via router A (with a path cost of 9) rather than router B (with a path cost of 10), due to hot-potato routing

Limited support for measuring the "inputs" to the models: Even when a simple and accurate model exists, the inputs to the model may be difficult to measure in an operational network. For example, today's IP routers do not provide accurate measurements of the number of active flows or their round-trip times. As another example, it is difficult to collect accurate measurements of the traffic matrix—the offered traffic from each ingress point to each egress point—which would be useful for setting the link weights in Figure 2. Over the years, a variety of techniques have been created for inferring the traffic matrix from other measurement data, such as link utilization statistics [5], with varying degrees of accuracy, but the problem remains challenging. As an another example, the Border Gateway Protocol (BGP)—the interdomain routing protocol for the Internet—has a tie-breaking rule that depends on the *order* the router learns about candidate paths. This kind of esoteric information is typically difficult to measure [6].

Computationally difficult optimization problems: Even with an accurate model and the necessary input data, the resulting optimization problems may be computationally intractable. For example, setting the link weights in shortest-path routing protocols based on the traffic matrix is an NP-hard problem, even for relatively simple objective functions [7], [8], [9], such as minimizing the sum of a convex function of the link utilizations. In practice, local-search techniques are often quite effective for selecting link weights, at least on certain network topologies, but the optimality gap can be high in the worst case. However, selecting weight settings that are robust in the face of topology or traffic changes is more difficult [10]. In addition, other protocols such as BGP have so many configurable parameters that the search space quickly becomes far too large to explore effectively [6].

Non-linear reactions in the network protocols: For some of the protocols, network behavior is sensitive to very small changes in the configurable parameters. For example, a small change in a link weight, or the failure of a link, may have significant influence on the resulting shortest paths. When an ISP can reach a destination through multiple egress points, each router typically selects the *closest* egress point, in a practice known as *early exit* or *hot-potato* routing [11]. For example, in Figure 4, router C directs traffic to destination d via router A since the path cost from C to A is smaller than the cost from C to B. However, a small change in the internal topology would cause C to direct traffic to B instead. The sensitivity of the routing protocol makes it difficult to select parameter settings that are robust to small changes in the network.

Since the operators cannot easily change the underlying protocols, they are forced to select the configurable parameters as effectively as possible. Common strategies include disabling protocol features that are difficult to model and using optimization tools based on local-search techniques to identify good parameter settings.

II. Designing Future Protocols With Optimization in Mind

The configurable parameters in network protocols have a profound influence on network performance and robustness. In light of this, we argue that future protocols, and the measurement infrastructure that supports them, should be designed with optimization in mind. That is, protocols should be judged, at least in part, by the optimization problems they induce. The design of the protocol can go hand in hand with the formulation of the optimization problems, and the difficulty of the optimization problems can inform revisions to the design of the protocol. Initial forays into redesigning IP routing with optimization in mind include:

Routing protocols with simpler optimization problems: Optimizing the link weights in a shortest-path routing protocol is an NP-hard problem. The work in [12] considers a simple variant of OSPF and IS-IS routing, where the routers forward traffic on paths in *inverse proportion* to the sum of the weights, rather than sending all traffic on shortest paths. This small change leads to an optimization problem that can be solved in polynomial time for simple objective functions, as well as a protocol that has smaller reactions to small changes in the path costs. The use of randomization, in general, may be an effective way to make protocols easier to tune.

Increasing the degrees of freedom: In some cases, the careful addition of more tunable parameters can turn an intractable optimization problem into a tractable one. For example, hot-potato routing selects between multiple egress points based on the path costs, as shown earlier in Figure 4. A more flexible egress-selection mechanism would allow each router to select the egress point based on a weighted sum of the link weights and a constant term. The ability to tune these extra parameters enables the application of conventional integer-programming techniques for setting the parameters that control the selection of egress points [13].

Automatic adaptation or negotiation based on feedback: When a configuration change causes a router to direct traffic to a different egress point, the neighboring domain starts receiving traffic at a different ingress point. Rather than make configuration changes independently, neighboring networks could coordinate their activities to find mutually beneficial ways to direct the traffic [14]. However, this approach depends on having an effective way to satisfy the objectives of both networks, while ensuring that neither domain has an incentive to provide misleading information to the other. Similarly, the difficulties in tuning the Random Early Detection (RED) mechanism may be surmountable in self-tuning versions of the algorithm, where the routers "learn" the appropriate settings of the parameters by adapting over time, even without the benefit of a predictive model.

Logically-centralized control over path selection: The routing protocols in today's IP networks were designed, first and foremost, to be implemented in a distributed fashion. More recently, the increasing capabilities of computers makes it possible to select the paths for a large collection of routers in a separate platform with a network-wide view of the topology and traffic [15], [16], [17]. Rather than emulating today's distributed protocols, these platforms could define new frameworks for computing paths in a logically-centralized fashion to satisfy network engineering goals directly. These platforms are a natural place to run optimization algorithms that compute the forwarding tables, rather than the configurable parameters that determine how the routers compute the forwarding tables.

We believe that designing protocols with optimization in mind is a promising area for future research. Similarly, designing protocols with *diagnosis* in mind would be very useful. Diagnosing network problems, such as a failing link or a denial-of-service attack, often devolves to solving "inverse problems" to identify possible explanations for the anomaly based on indirect observations of the underlying system. Designing both the network protocols and the measurement systems with these inverse problems in mind is another interesting direction for future work.

References

[1] J. Moy, “OSPF Version 2,” Request For Comments 2328, IETF, April 1998.

[2] R. Callon, “Use of OSI IS-IS for Routing in TCP/IP and Dual Environments,” Request For Comments 1195, IETF, December 1990.

[3] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, August 1993.

[4] N. Feamster, J. Winick, and J. Rexford, “A model of BGP routing for network engineering,” in *Proc. ACM SIGMETRICS*, June 2004.

[5] M. Grossglauser and J. Rexford, “Passive traffic measurement for IP operations,” in *The Internet as a Large-Scale Complex System*, pp. 91–120, Oxford University Press, 2005.

[6] N. Feamster, J. Borkenhagen, and J. Rexford, “Guidelines for interdomain traffic engineering,” *ACM SIGCOMM Computer Communication Review*, vol. 33, October 2003.

[7] B. Fortz and M. Thorup, “Increasing Internet capacity using local search,” *Computational Optimization and Applications*, vol. 29, no. 1, pp. 13–48, 2004.

[8] J. Rexford, “Route optimization in IP networks,” in *Handbook of Optimization in Telecommunications*, Springer Science + Business Media, February 2006.

[9] B. Fortz, J. Rexford, and M. Thorup, “Traffic engineering with traditional IP routing protocols,” *IEEE Communication Magazine*, October 2002.

[10] B. Fortz and M. Thorup, “Robust optimization of OSPF/IS-IS weights,” in *Proc. International Network Optimization Conference*, pp. 225–230, October 2003.

[11] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, “Dynamics of hot-potato routing in IP networks,” in *Proc. ACM SIGMETRICS*, June 2004.

[12] J. H. Fong, A. C. Gilbert, S. Kannan, and M. J. Strauss, “Better alternatives to OSPF routing,” *Algorithmica*, vol. 43, no. 1-2, pp. 113–131, 2005.

[13] R. Teixeira, T. Griffin, M. Resende, and J. Rexford, “Tie breaking: Tunable interdomain egress selection,” in *Proc. CoNEXT*, October 2005.

[14] R. Mahajan, D. Wetherall, and T. Anderson, “Negotiation-based routing between neighboring ISPs,” in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, May 2005.

[15] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, “The case for separating routing from routers,” in *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2004.

[16] O. Bonaventure, S. Uhlig, and B. Quoitin, “The Case for More Versatile BGP Route Reflectors.” Expired Internet Draft draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.

[17] A. Farrel, J.-P. Vasseur, and J. Ash, “A Path Computation Element (PCE) Based Architecture.” Internet Draft draft-ietf-pce-architecture-04.txt, work in progress, January 2006.