# Design for Optimizability: Traffic Management of a Future Internet

Jiayue He, Jennifer Rexford and Mung Chiang

**Abstract** :

As networks grow in size and complexity, network management has become an increasingly challenging task. Many protocols have tunable parameters, and optimization is the process of setting these parameters to optimize an objective. In recent years, optimization techniques have been widely applied to network management problems, albeit with mixed success. Realizing that optimization problems in network management are induced by assumptions adopted in protocol design, we argue that instead of optimizing existing protocols, protocols should be designed with optimization in mind from the beginning. Using examples from our past research on traffic management, we present principles that guide how changes to existing protocols and architectures can lead to optimizable protocols. We also discuss the trade-offs between making network optimization easier and the overhead these changes impose.

## 1 Introduction

Network management is the continuous process of monitoring a network to detect and diagnose problems, and of configuring protocols and mechanisms to fix problems and optimize performance. Traditionally, network management has been largely impenetrable to the research community since many of the problems appear both complex and ill-defined. In the past few years, the research community has made tremendous progress casting many important network management problems as optimization problems. Network optimization involves satisfying network management objectives by setting the tunable parameters that control network behavior. Solving an optimization problem involves optimizing an *objective function* subject to a set of *constraints*. Unfortunately, while *convex* optimization problems are easier

Jiayue He , Jennifer Rexford and Mung Chiang
Princeton University, Princeton e-mail: {jhe,jrex,chiangm}@princeton.edu

to solve, many problems that arise in data networks are nonconvex. Consequently, they are computationally intractable, with many local optima that are suboptimal.

In this paper, we argue that the difficulty of solving the key optimization problems is an indication that we may need to revise the underlying protocols, or even the architectures, that lead to these problem formulations in the first place. We advocate the design of *optimizable networks*—network architectures and protocols that lead to easy-to-solve optimization problems and consequently, optimal solutions. Indeed, the key difference between "network optimization" and "optimizable networks" is that the former refers to solving a given problem (induced by the existing protocols and architectures) while the latter involves formulating the "right" problem (by changing protocols or architectures accordingly).

The changes to protocols and architectures can range from minor extensions to clean-slate designs. In general, the more freedom we have to make changes, the easier it would be to create an optimizable network. On the other hand, the resulting improvements in network management must be balanced against other considerations such as *scalability* and *extensibility*, and must be made *judiciously*. To make design decisions, it is essential to quantify the trade-off between making network-management problems easier by changing the problem statement and the extra overhead the resulting protocol imposes on the network.

Network optimization has had a particularly large impact in the area of traffic management, which controls the flow of traffic through the network. Today, this spans across congestion control, routing and traffic engineering. In Section 2, we describe how optimization is used in traffic management today. In Section 3, we illustrate *design principles* which we have uncovered through our own research experiences on traffic management. Traffic management is an extremely active area of research, but we will not address related work in this paper since these examples are included to serve as illustrations of general principles. In Section 4, we discuss other aspects of traffic management, such as interdomain routing and active queue management, where the problems are even more challenging. We also examine the trade-off between performance achieved and overhead imposed when designing optimizable protocols. We conclude and point to future work in Section 5.

## 2 Traffic Management Today

In this section, we introduce how optimization is used in the context of traffic management inside a single Autonomous System (AS). Traffic management has three players: users, routers, and operators. In today's Internet, users run TCP congestion control to adapt their sending rates at the edge of the network based on packet loss. Congestion control has been reverse engineered to be implicitly solving an optimization problem, [1, 2, 3]. Inside the network, operators tune parameters in the existing routing protocols to achieve some network-wide objective in a process called traffic engineering, see Figure 1.

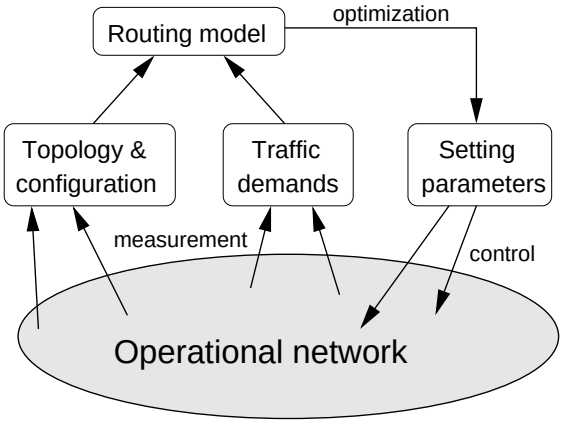


**Fig. 1** Components of the route optimization framework.

## *2.1 Traffic Engineering*

| Symbol | Meaning |
|---|---|
| $(i, j)$ | Pair of routers. |
| $x^{(i,j)}$ | Traffic demand between $i$ and $j$. |
| $l$ | A single link. |
| $w_l$ | Link weight $l$. |
| $c_l$ | Capacity of link $l$. |
| $y_l$ | Traffic load on link $l$. |
| $f(y_l/c_l)$ | Penalty function as a function of link utilization. |
| $r_l^{(i,j)}$ | Portion of the traffic from router $i$ to router $j$ that traverses the link $l$. |

**Table 1** Summary of notation for Section 2.1.

Inside a single AS, each router is configured with an integer weight on each of its outgoing links, as shown in Figure 2. The routers flood the link weights throughout the network and compute shortest paths as the sum of the weights. For example, $i$ directs traffic to $k$ though the links with weights $(2,1,5)$Each router uses this information to construct a table that drives the forwarding of each IP packet to the next hop in its path to the destination. These protocols view the network inside an AS as a graph where each router is a node $n \in N$ and each directed edge is a link $l \in L$ between two routers. Each unidirectional link has a fixed capacity $c_l$, as well as a configurable weight $w_l$. The outcome of the shortest-path computation can be represented as $r_l^{(i,j)}$: the proportion of the traffic from router $i$ to router $j$ that traverses the link $l$.

Operators set the link weights in intradomain routing protocols in a process called traffic engineering. The selection of the link weights $w_l$ should depend on the offered

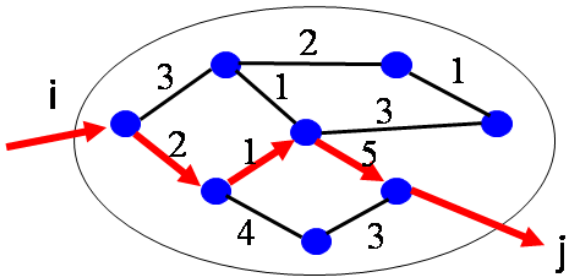


**Fig. 2** Network topology with link weights for shortest path routing.

traffic, as captured by a demand matrix whose entries $x^{(i,j)}$ represents the rate of traffic entering at router $i$ that is destined to router $j$. The traffic matrix can be computed based on traffic measurements [4] or may represent explicit subscriptions or reservations from users. Given the traffic demand $x^{(i,j)}$ and link weights $w_l$, the volume of traffic on each link $l$ is $y_l = \sum_{i,j} x^{(i,j)} r_l^{(i,j)}$, the proportion of traffic that traverses link $l$ summed over all ingress-egress pairs. An objective function can quantify the "goodness" of a particular setting of the link weights. For traffic engineering, the optimization considers a network-wide objective of minimizing $\sum_l f(y_l/c_l)$. The traffic engineering penalty function $f$ is a convex, non-decreasing, and twice-differentiable function that gives an increasingly heavy penalty as link load increases, such as an exponential function. The problem traffic engineering solves is to set link weights to minimize $\sum_l f(y_l/c_l)$, *assuming* the weights are used for shortest-path routing.

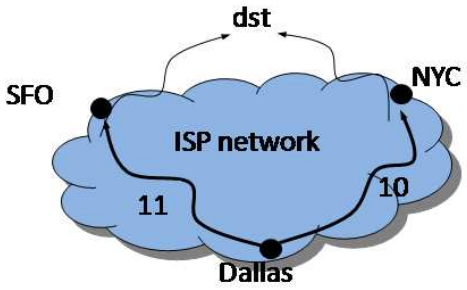


**Fig. 3** Traffic from Dallas exits via New York City (with a path cost of 10) rather than San Francisco (with a path cost of 11), due to hot-potato routing

So far, we have covered the impact of link weights inside an AS. When a network, such as an Internet Service Provider (ISP) backbone, can reach a destination through multiple egress points, a routing change inside the AS may change how traffic leaves the AS. Each router typically selects the closest egress point out of a

set of egress points which can reach a destination, in terms of the intradomain link weights $w_l$, in a practice known as early-exit or hot-potato routing [5]. In the example in Figure 3, suppose a destination is reachable via egress points in New York City and San Francisco. Then traffic from Dallas exits via New York City rather than San Francisco since the intradomain path cost from Dallas to New York City is smaller. If the traffic from Dallas encounters congestion along the downstream path from New York City in Figure 3, the network operators could tune the link weights to make the path through SanFrancisco appear more attractive. Controlling where packets leave the network, and preventing large shifts from one egress point to another, is an important part of engineering the flow of traffic in the network. Models can capture the effects of changing the link weights on the intradomain paths and the egress points, but identifying good settings of the weights is very difficult.

## *2.2 Pros and Cons of Traffic Management*

Traffic management today has several strengths. First, routing depends on a very *small amount of state per link* i.e., link weights. In addition, forwarding is done *hop-by-hop*, so that each router decides independently how to forward traffic on its outgoing links. Second, routers only disseminate information when link weights or topology change. Also, TCP congestion control is based only on *implicit feedback* of packet loss and delay, rather than explicit messages from the network. Third, the selection of link weights can depend on a wide variety of performance and reliability constraints. Fourth, hot-potato routing reduces internal resource usage (by using the closest egress point), adapts automatically to changes in link weights, and allows routers in the AS to do hop-by-hop forwarding toward the egress point. Last but not least, the decoupling of congestion control and traffic engineering reduces complexity through separation of concerns.

On the other hand, today's protocols also have a few shortcomings. To start with, optimizing the link weights in shortest-path routing protocols based on the traffic matrix is NP-hard, even for simplest of objective functions [6]. In practice, local-search techniques are used for selecting link weights [6]; however, the computation time is long and, while the solutions are frequently good [6], the deviation from the optimal solution can be large. Finding link weights which work well for egress point selection is even more challenging, as this adds even more constraints on how the weights are set.

There are other limitations to today's traffic management. The network operator can only *indirectly* influence how the routers forward traffic, through the setting of the link weights. Further, traffic engineering is performed assuming that the offered traffic is inelastic. In reality, end hosts adapt their sending rates to network congestion, and network operators adapt the routing based on measurements of the traffic matrix. Although congestion control and routing operate independently, their decisions are coupled. The joint system is stable, but often suboptimal [7]. Furthermore, traffic engineering does not necessarily adapt on a small enough timescale

to respond to shifts in user demands. In addition to timescale alternatives, there are also choices as to geographically which part of traffic management work should be carried out inside the network, and which by the sources. These limitations suggest that revisiting architectural decisions is a worthy research direction.

## 3 Design Optimizable Protocols

In this section, we illustrate three design principles through proposed protocols. The three principles also correspond to the three parts of an optimization problem formulation: objective, variables and constraints. In a generic optimization problem formulation, the objective is to minimize $g(x)$ over the variable $x$, subject to constraints on $x$:

$$\begin{array}{ll} \text{minimize} & g(x) \\ \text{subject to} & x \in S \\ \text{variable} & x \end{array} \tag{1}$$

From optimization theory, it is well established that a local optimum of (1) is also a *global optimum*, which can be found in *polynomial time* and often very fast, if $S$ is a convex set and $g$ is a convex function. The intuition is as follows: searching for an optimum on a nonconvex set is challenging as it would be difficult to "cross" any gaps as seen in Figure 4. In addition, a convex objective function is necessary for a global optimum to exist as seen in Figure 5.

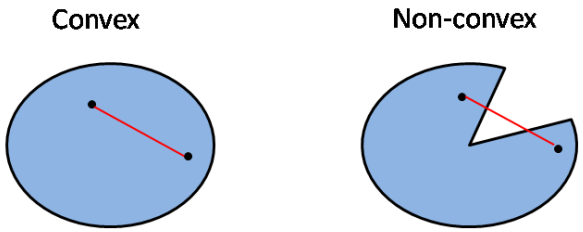


**Fig. 4** Convex and nonconvex sets. A convex set $S$ is defined as if $x, y \in S$, then $\theta x + (1-\theta)y \in S$, for all $\theta \in [0,1]$.

In other words, a convex optimization problem leads to both *tractability* and *optimality*. Due to single-path routing, an artifact of the current system, the constraint set is not convex for most traffic management problems. In our first example, we tackle this problem head-on by changing the shape of the constraint set. In our second example, we avoid the problem because the particular problem formulation falls under a special class of integer programming problems. In our third example, we change the system to allow routing to be per path multi-commodity flow, so that decom-

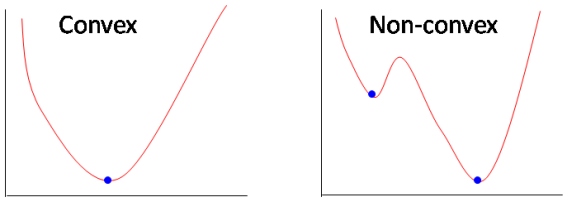


**Fig. 5** Convex and nonconvex functions. A function $g$ is a convex function if domain of $g$ is a convex set and $f(\theta x+(1-\theta)y) \leq \theta g(x)+(1-\theta)g(y)$.

position techniques can be applied to derive stable and fast-timescale interaction between routing and congestion control.

## *3.1 Changing the Shape of the Constraint Set*

| Symbol | Meaning |
|---|---|
| $(i, j)$ | Pair of routers. |
| $x^{(i,j)}$ | Traffic demand between $i$ and $j$. |
| $l$ | A link. |
| $r_l^{(i,j)}$ | Portion of the traffic from router $i$ to router $j$ that traverses the link $l$. |
| $k$ | A path between $i$ and $j$. |
| $w_k^{(i,j)}$ | Path weight of path $k$ between $i$ and $j$. |
| $x_k^{(i,j)}$ | Traffic demand between $i$ and $j$, that will be placed on path $k$. |

**Table 2** Summary of notation for Section 3.1.

Some optimization problems involve *integer* constraints, which are not convex, making them intractable and their solutions suboptimal. Relaxing the integer constraint to approximate a convex constraint can lead to a more tractable problem and a smaller optimality gap. In the original link-weight setting problem where link weights are set to minimize $\sum_l f(y_l/c_l)$, assuming the weights are used for shortest-path routing, the constraints are nonconvex. The network usually has a single shortest path from $i$ to $j$, resulting in $r_l^{(i,j)} = 1$ for all links $l$ along the path, and $r_l^{(i,j)} = 0$ for the remaining links. An OSPF or IS-IS router typically splits traffic evenly along one or more outgoing links along shortest paths to the destination, allowing for limited fractional values of $r_l^{(i,j)}$, but the constraint set is still highly nonconvex. The ability to split traffic arbitrarily over multiple paths would make the constraints convex, i.e., $r_l^{(i,j)} \in [0,1]$. The downside is this approach would sacrifice the simplicity

of OSPF and IS-IS, where routers compute paths in a distributed fashion based on link weights alone.

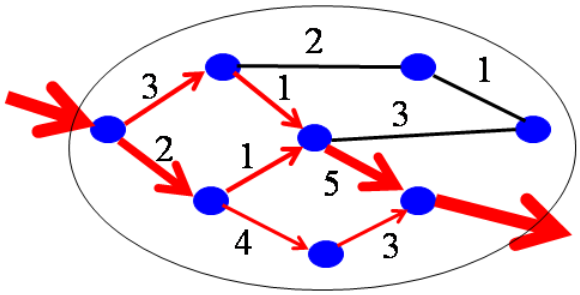


**Fig. 6** Routers forwarding traffic with exponentially diminishing proportions of the traffic directed to the longer paths. Arrows indicate paths that make forward progress toward the destination, and the thickness of these lines indicates the proportion of the traffic that traverses these edges.

Rather than supporting arbitrary splitting, a recent proposal advocates small extensions to OSPF and IS-IS to split traffic over multiple paths [8]. Under this proposal, the routers forward traffic on multiple paths, with exponentially diminishing proportions of the traffic directed to the longer paths, as shown in Figure 6. The goal is still to minimize $\sum_l f(y_l/c_l)$, but allowing any routing protocol based on link weights instead of assuming only shortest-path routing.

More formally, given multiple paths between routers $i$ and $j$, indexed by $k$, to keep the protocols simple, the constraint is to have $x_k^{(i,j)}/x^{(i,j)}$, the ratio of traffic placed on path $k$, be computable using only link weight information. At each router $i$, the following computation is performed:

$$\frac{x_k^{(i,j)}}{x^{(i,j)}} = \frac{e^{-w_k^{(i,j)}}}{\sum_m e^{-w_m^{(i,j)}}}, \tag{2}$$

where $w_k^{(i,j)}$ is the sum of the link weights on the $k$th path between router $i$ and $j$. So as in OSPF and IS-IS today, each router would compute all the path weights for getting from $i$ to $j$, there is just an extra step to compute the splitting ratios. For example, in Figure 6, consider the two lower paths of costs 8 (i.e., $2+1+5$) and 9 (i.e., $2+4+3$), respectively. The path with cost 8 will get $e^{-8}/(e^{-8}+e^{-9})$ of the traffic, and the path with cost 9 will get $e^{-9}/(e^{-8}+e^{-9})$ of the traffic.

Under this formulation, both link weights and the flow splitting ratios are variables. This enlarges the constraint set, and the resulting constraints are much easier to *approximate* with convex constraints. Consequently, the link-weight tuning problem is tractable, i.e., can be solved much faster than the local search heuristics today. In addition, the modified protocol is optimal, i.e., makes the most efficient use of link capacities, and is more robust to small changes in the path costs. The optimality

result is unique to this particular problem where there is an intersection between the set of optimal protocols and protocols based on link weights. In general, the optimality gap is reduced by enlarging the constraint set, as seen in a similar proposed extension to OSPF and IS-IS [9]. By *changing the constraint set*, [8] and [9] retains the simplicity of link-state routing protocols and hop-by-hop forwarding, while inducing an optimization problem that is both faster to solve and lead to a smaller optimality gap.

## *3.2 Adding Variables to Decouple Constraints*

| Symbol | Meaning |
|---|---|
| $(i, j)$ | Ingress-egress pair. |
| $w^{(i,j)}$ | Path cost between $i$ and $j$. |
| $d$ | Destination. |
| $q_d^{(i,j)}$ | Ranking metric for paths between $i$ and $j$. |
| $\alpha_d^{(i,j)}$ | Tunable parameter to support automatic adaptation to topology changes. |
| $\beta_d^{(i,j)}$ | Tunable parameter to support static ranking of egress points $j$ per ingress router $i$. |

**Table 3** Summary of notation for Section 3.2.

Some optimization problems can involve many tightly-coupled constraints, making it difficult to find a feasible solution. Introducing extra variables can decouple the constraints, and increase the size of the feasible region. As an example, setting the link weights is highly constrained, since the weights are used to compute both the forwarding paths between the routers inside the domain and the egress points where the traffic leaves the domain. Weakening the coupling between intradomain routing and egress-point selection is the key to simplifying the optimization problem and improving network performance.

Rather than selecting egress points $j$ from ingress router $i$ based only on the intradomain path costs $w^{(i,j)}$ (sum of all link weights on the path from $i$ to $j$), a variable $q_d^{(i,j)}$ is introduced for router $i$, across all destinations $d$ and egress points $j$. To support flexible policy while adapting automatically to network changes, the metric $q_d^{(i,j)}$ includes both configurable parameters and values computed directly from a real-time view of the topology. In particular, $q_d^{(i,j)} = \alpha_d^{(i,j)} w^{(i,j)} + \beta_d^{(i,j)}$ where $\alpha$ and $\beta$ are configurable values [10]. The first component of the equation supports automatic adaptation to topology changes, whereas the second represents a static ranking of egress points per ingress router. Providing separate parameters for each destination prefix allows even greater flexibility, such as allowing delay-sensitive traffic to use the closest egress point while preventing unintentional shifts in the egress points for other traffic.

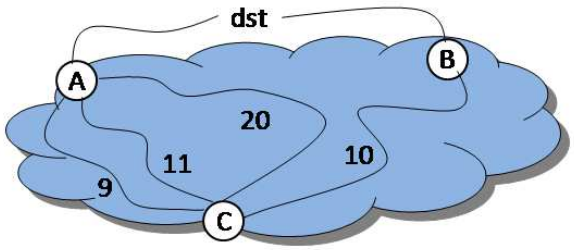


**Fig. 7** Ingress router *c* can reach destination through egresses *a* and *b*.

Consider a scenario where $\alpha$ and $\beta$ are tuned to handle failure scenarios. As seen in Figure 7, the ingress router *c* can reach a destination through egress routers *a* and *b*. There are three paths from *c* to *a* with paths costs 9, 11, and 20, respectively, the path cost from *c* to *b* is 11. The goal is to not switch the traffic from leaving egress router *a* if the path with cost of 9 fails, but do switch to egress *B* if the path with cost 11 fails also. This can be expressed as a set of conditions as in (3):

$$\begin{aligned} 9\alpha_d^{c,a} + \beta_d^{c,a} &< 10\alpha_d^{c,b} + \beta_d^{c,b} \\ 11\alpha_d^{c,a} + \beta_d^{c,a} &< 10\alpha_d^{c,b} + \beta_d^{c,b} \\ 20\alpha_d^{c,a} + \beta_d^{c,a} &> 10\alpha_d^{c,b} + \beta_d^{c,b} \end{aligned} \tag{3}$$

One set of $\alpha$ and $\beta$ values to achieve the conditions in (3) is $\alpha_d^{c,a} = 1$, $\beta_d^{c,a} = 1$, $\alpha_d^{c,b} = 1$, and $\beta_d^{c,b} = 0$.

In general, the resulting integer multicommodity-flow problem is still nonconvex and consequently intractable. This problem formulation happens to correspond to a very special subset of integer programming problems where relaxing the integrality constraints would still produce integer solutions [11], thus side-stepping the convexity issue. That is, the optimization problem becomes solvable in polynomial time if we allow an ingress point *i* to split traffic destined to *d* over multiple egress points *e*, rather than forcing all traffic from *i* to go to a single egress point; in practice, solving the relaxed problem produces integer solutions that do, in fact, direct all traffic from *i* to *d* via a single egress point *e*. Overall, by *increasing the degrees of freedom*, a management system can set the new parameters under a variety of constraints that reflect the operators' goals for the network [10]. Not only does the network become easier to optimize, but the performance improves as well, due to the extra flexibility in controlling where the traffic flows.

| Symbol | Meaning |
|---|---|
| $(i, j)$ | Pair of routers. |
| $x^{(i,j)}$ | Traffic demand between $i$ and $j$. |
| $U(x^{(i,j)})$ | Utility of traffic demand between $i$ and $j$. |
| $l$ | A single link. |
| $c_l$ | Capacity of link $l$. |
| $r_l^{i,j}$ | Portion of the traffic from router $i$ to router $j$ that traverses the link $l$. |
| $f(\sum_{i,j} x^{(i,j)} r_l^{(i,j)}/c_l)$ | Penalty function as a function of link utilization. |
| $v$ | Weight between utility and penalty functions. |
| $k$ | A path between $i$ and $j$. |
| $x_k^{(i,j)}$ | Traffic demand between $i$ and $j$, that will be placed on path $k$. |
| $H_{l,k}^{(i,j)}$ | Matrix capturing the available paths between $i$ and $j$. |

**Table 4** Summary of notation for Section 3.3.

## *3.3 Combining Objectives to Derive Protocols*

In a system, there can be multiple interacting optimization problems with different objectives. Combining the objectives of multiple problems can allow for a better solution to the overall problem. In today's traffic management system, congestion control and traffic engineering have different objectives. Congestion control tries to maximize aggregate user utility, and as a result tends to push traffic into the network so that multiple links are used at capacity. In contrast, traffic engineering uses a link cost function which heavily penalizes solutions with bottleneck links.

User utility $U(x^{(i,j)})$ is a measure of "happiness" of router pair $(i, j)$ as a function of the total sending rate $x^{(i,j)}$. $U$ is a concave, non-negative, increasing and twice-differentiable function, e.g., logarithmic function, that can also represent the elasticity of the traffic or determine fairness of resource allocation. As mentioned earlier, the objective for traffic engineering is a convex function of link load. The objective function has two different practical interpretations. First, $f$ can be selected to model M/M/1 queuing delay and thus the objective is to minimize average queuing delay. Second, network operators want to penalize solutions with many links at or near capacity and do not care too much whether a link is 20% loaded or 40% loaded [6]. One way to combine the objectives of traffic engineering and congestion control is to construct a weighted sum of utility and link cost functions as the overall objective for traffic management [12], where $v$ is the weight between the two objectives.

$$\begin{aligned} &\text{maximize } \textstyle\sum_i U(x^{(i,j)}) - v \sum_l f(\sum_{i,j} x^{(i,j)} r_l^{(i,j)}/c_l) \\ &\text{subject to } \textstyle\sum_{i,j} x^{(i,j)} r_l^{(i,j)} \leq c_l,\ \mathbf{x} \succeq \mathbf{0}. \end{aligned} \tag{4}$$

In [12], we revisit the division of labor between users, operators and routers. In this case, we allow for a per path multi-commodity flow solution, hence resulting in a convex problem, and opens up many standard optimization techniques that derive distributed and iterative solutions. In its current form, (4) has a non-convex constraint set, which can be transformed into a convex set if the routing is allowed to be

multipath. To capture multipath routing, we introduce $x_k^{(i,j)}$ to represent the sending rate of router $i$ to router $j$ on the $k$th path. We also represent available paths by a matrix $\mathbf{H}$ where

$$H_{l,k}^{(i,j)} = \begin{cases} 1, \text{ if path } k \text{ of pair } (i,j) \text{ uses link } l \\ 0, \text{ otherwise.} \end{cases}$$

$\mathbf{H}$ does not necessarily present all possible paths in the physical topology, but a subset of paths chosen by operators or the routing protocol. Using the new notation, the capacity constraint is transformed into $\sum_{i,j,k} x_k^{(i,j)} H_{l,k}^{(i,j)} \leq c_l$, which is convex.

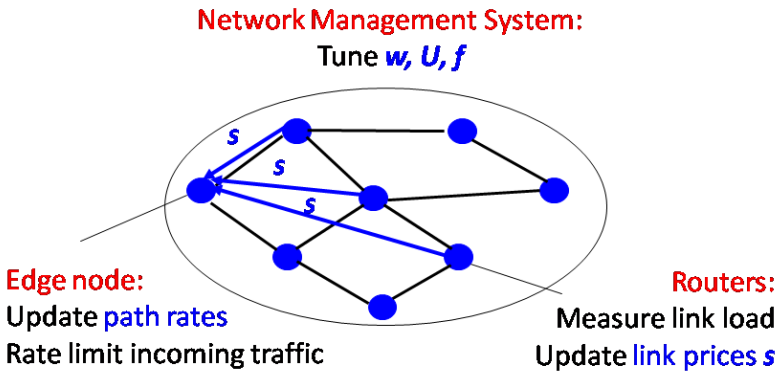


**Fig. 8** A high-level view of how the distributed traffic-management protocol works.

Decomposition is the process of breaking up a single optimization problem into multiple ones that can be solved independently. As seen in Figure 8, decomposing the overall traffic management optimization problem, a distributed protocol is derived that splits traffic over multiple paths, where the splitting proportions depend on feedback from the links. The links send feedback to the edge routers in the form of a price $s$ that indicates the local congestion level, based on local link load information. Although there are multiple ways to decompose the optimization problem, they all lead to a similar divions of functions between the routers and the links [12].

By *embedding the management objectives in the protocols*, the link-cost function is now automated incorporated by the links themselves as part of computing the feedback sent to the edge routers, rather than by the network-management system. As seen in Figure 8, there are no link weights at all in this distributed protocol. As such, the network-management system merely specifies $U$, $f$ and $v$, instead of adapting the link weights over time.

## 4 Open Challenges in Traffic Management Optimization

The principles introduced in the previous section are a useful first step towards designing optimizable protocols, but are by no means comprehensive. The merits of proposed optimizable protocols should always be balanced with any extra overhead in practical implementation and robustness to changing network dynamics. In addition, the principles introduced in the previous section focuses on intradomain traffic management, and do not address all the challenges in end-to-end traffic management. Finally, when deriving new architectures, the balance between performance and other factors is even more delicate.

### *4.1 Performance vs. Overhead Trade-off*

Characterizing a network architecture in terms of the tractability of network-management problems is just one piece of a complex design puzzle. The design of optimizable networks introduces tension between the ease of network optimizability and the overhead on network resources. Some of the architectural decisions today make the resulting protocols simple. For example, protocols which rely on *implicit feedback* e.g., TCP congestion control, do not have message passing overhead. Further, *hop-by-hop forwarding* does not depend on the upstream path, requiring less processing at the individual routers. It would be desirable to capture such notions of simplicity mathematically, so we can learn to derive optimizable protocols which retain them.

Our example in Section 3.1 manages to retain the simplicity of hop-by-hop forwarding while resulting in a tractable optimization problem. In this particular case, optimality gap was significantly reduced with very little extra overhead. However, some approaches make the protocol more optimizable at the expense of additional overhead. For example, adding flexibility in egress-point selection in Section 3.2 introduces more parameters that the network-management system must set. Similarly, revisiting the division of functionalities in Section 3.3 leads to a solution that requires explicit feedback from the links. Imposing extra overhead on the network may be acceptable, if the improvement in performance is sufficiently large.

Furthermore, ensuring a completely tractable optimization problem is sometimes unnecessary. An NP-hard problem may be acceptable, if good heuristics are available. For striking the right trade-offs in the design of optimizable networks, it is important to find effective ways to quantify the acceptable amount of deviation from the optimal solution. There are also well-established, quantitative measures of the notions of how easily-solvable an optimization is. These quantitative measures can help determine *how much* the protocols and architectures need to change to better support network management.

The protocols today are designed with certain assumptions in mind, e.g., single-path routing and hop-by-hop forwarding. Some of these assumptions cause the resulting optimization problem to be intractable e.g., single-path routing, while others do not, e.g., hop-by-hop forwarding. By perturbing the underlying assumptions in

today's protocols, we can achieve a different point in the trade-off space of optimality versus simplicity. Therefore, it is worth exploring the alternatives, even if at the end the decision is to keep the original protocol and architectures. In order to choose between protocol designs, the key is to gain a deeper understanding of the trade-offs. As such, we believe that *design for optimizability* can be a promising, new interdisciplinary area between the systems and theory communities.

## *4.2 End-to-End Traffic Management*

Our examples thus far focused on optimization problems in intradomain traffic management. Routing within a single domain side-steps several important issues that arise in other aspects of data networking, for several reasons:

- A single domain has the authority to collect measurement data (such as the traffic and performance statistics) and tune the protocol configuration (such as the link weights).
- The routing configuration changes on the timescale of hours or days, allowing ample time to apply more computationally intensive solution techniques.
- The optimization problems consider highly aggregated information, such as link-level performance statistics or offered load between pairs of routers.

When these assumptions do not hold, the resulting optimization problems become even more complicated, as illustrated by the following two examples.

*Optimization in interdomain traffic management:* In the Internet, there are often multiple Autonomous Systems (AS) in the path between the sender and the receiver. Each AS does not have full view of the topology, only the paths which are made visible to it through the routing-protocol messages exchanged in the Border Gateway Protocol (BGP). In addition, each AS has a set of private policies that reflect its business relationships with other ASes. Without full visibility and control, it is difficult to perform interdomain traffic management. For example, to implement DATE in the Internet, the ASes would need to agree to provide explicit feedback from the links to the end hosts or edge routers, and trust that the feedback is an honest reflection of network conditions. Extending BGPs to allow for multiple paths would simplify the underlying optimization problem, but identifying the right incentives for ASes to deploy a multipath extension to BGP remains an open question.

*Optimization in active queue management:* A router may apply active queue management schemes like Random Early Detection [13] to provide TCP senders with early feedback about impending congestion. RED has many configurable parameters to be selected by network operators, e.g.,, queue-length thresholds and maximum drop probability. Unfortunately, predictive models for how the tunable parameters affect RED's behavior remain elusive. In addition, the appropriate parameter values may depend on a number of factors, including the number of active data transfers and the distribution of round-trip times, which are difficult to measure on high-speed links. Recent analytic work demonstrates that setting RED pa-

rameters to stabilize TCP is fundamentally difficult [14]. It is appealing to explore alternative active-queue management schemes that are easier to optimize, including self-tuning algorithms that do not require the network-management system to adjust any parameters.

From these two examples, it is clear that there remains open challenges in end-to-end traffic management. Outside the context of traffic management, network optimization's role is even less understood. We argue for a principled approach in tackling these challenges so that, in time, protocol design can be less of an art and more of a science.

### *4.3 Placement of Functionality*

The challenges are not just limited to protocols, but extend to architectural decisions regarding the placement of functionality. Architecturally, the DATE example represents one extreme where most of computation and coordination is moved into the distributed protocols that run in the routers. In the context of Figure 1, this means much of the measurement, control and optimization is pushed down into the network. One can consider another extreme, where the network-management systems bear all the responsibility for adapting to changes in network conditions, as in [15]. Both approaches redefine the division of labor between the management system and the routers, where one moves most of the control into the distributed protocols and the other has the management systems directly specify how the routers handle packets.

In some cases, having the management system bear more responsibility would be a natural choice. For example, if an optimization problem is fundamentally difficult, consequently leading to distributed solutions that are complicated or suboptimal, or both. Unlike the routers, a management system has the luxury of a global view of network conditions and the ability to run centralized algorithms for computing the protocol parameters. Today's traffic engineering uses the centralized approach and allows operators to tailor the objectives to the administrative goals of the network. This leads to a more evolvable system, where the objective function and constraints can differ from one network to another, and change over time. In addition, the operators can capitalize on new advances in techniques for solving the optimization problems, providing an immediate outlet for promising research results.

The network-management system can apply centralized algorithms based on a global view of network conditions, at the expense of a slower response based on coarse-grain measurements. Yet some parts of traffic management, such as detecting link failures and traffic shifts, must occur in real time. In order to understand which functions must reside in the routers to enable adaptation on a sufficiently small timescale, it is important to quantify the loss in performance due to slower adaptation. For functions which require fast adaptation, an architecture where end user load balance across multiple paths would be desirable. For functions that can operate on a slower timescale, the control of flow distribution can be left to operators. In general,

determining the appropriate division of labor between the network elements and the management systems is an avenue for future research.

## 5 Conclusions and Future Work

In recent years, optimization has played an increasingly important role in network management. In this paper, we argue that, instead of just trying to optimize existing protocols, new protocols should be designed *for* the ease of optimization. If a set of architectures and protocols lead to intractable optimization problems for network management, we argue that, instead of trying to solve these problems by ad hoc heuristics, we should revisit some of the underlying assumptions in the architectures and protocols. Such explorations can lead to easier network optimization problems and may provide superior simplicity-optimality tradeoff curves.

Drawing from our own research experiences in traffic management, we propose three guiding principles for making optimizable protocols which correspond to three aspects of an optimization problem i.e., constraints, variables and objective. First, changing the constraint set can turn an NP-hard optimization problem into an easier problem and reduce the optimality gap. Second, increasing degrees of freedom (by introducing extra parameters) can break tightly coupled constraints. Finally, embedding management objectives in the protocol can lead to alternative architectures. Still, protocols changes must be made judiciously to balance the gain in performance with the extra consumption of network resources.

Ultimately, the design of manageable networks raises important architectural questions about the appropriate division of functionalities between network elements and the systems that manage them. This paper represents a first step toward identifying design principles that can guide these architectural decisions. The open challenges which remain suggest that the design of manageable networks may continue to be somewhat of an art, but hopefully one that will be guided more and more by design principles. We believe that providing a new, comprehensive foundation for the design of manageable networks is an exciting avenue for future research.

## Acknowledgments

We would like to thank Constantine Dovrolis, Nick Feamster, Renata Teixeira and Dahai Xu for their feedback on earlier drafts. This work has been supported in part by NSF grants CNS-0519880 and CCF-0448012, and DARPA Seedling W911NF-07-1-0057.

## References


1. F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. of Operational Research Society*, vol. 49, pp. 237–252, March 1998.
2. S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 11, pp. 525–536, August 2003.
3. R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.
4. 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.
5. R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proc. ACM SIGMETRICS*, June 2004.
6. B. Fortz and M. Thorup, "Increasing Internet capacity using local search," *Computational Optimization and Applications*, vol. 29, no. 1, pp. 13–48, 2004.
7. J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards multi-layer traffic engineering: Optimization of congestion control and routing," *IEEE J. on Selected Areas in Communications*, June 2007.
8. D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," in *Proc. IEEE INFOCOM*, May 2008.
9. D. Xu, M. Chiang, and J. Rexford, "DEFT: Distributed exponentially-weighted flow splitting," in *Proc. IEEE INFOCOM*, May 2007.
10. R. Teixeira, T. Griffin, M. Resende, and J. Rexford, "TIE breaking: Tunable interdomain egress selection," *IEEE/ACM Trans. Networking*, August 2007.
11. A. Ozdaglar and D. P. Bertsekas, "Optimal solution of integer multicommodity flow problems with application in optical networks," *Frontiers in Global Optimization*, vol. 74, pp. 411–435, 2004.
12. J. He, M. Bresler, M. Chiang, and J. Rexford, "Rethinking Internet traffic management: From multiple decompositions to a practical protocol," in *Proc. CoNEXT*, December 2007.
13. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, August 1993.
14. S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Computer Networks*, vol. 43, pp. 633–647, December 2003.
15. A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, October 2005.