

Layering As Optimization Decomposition

Mung Chiang^{*}, Steven H. Low[†], A. Robert Calderbank[‡], John C. Doyle[§]

April 2, 2006

Abstract

Network protocols in layered architectures have historically been obtained on an ad hoc basis, and many of the recent cross-layer designs are conducted through piecemeal approaches. They may instead be holistically analyzed and systematically designed as distributed solutions to some global optimization problems. This paper presents a survey of the recent efforts towards a systematic understanding of “layering” as “optimization decomposition”, where the overall communication network is modeled by a generalized Network Utility Maximization (NUM) problem, each layer corresponds to a decomposed subproblem, and the interfaces among layers are quantified as functions of the optimization variables coordinating the subproblems. There can be many alternative decompositions, each leading to a different layering architecture. This paper summarizes the current status of horizontal decomposition into distributed computation and vertical decomposition into functional modules such as congestion control, routing, scheduling, random access, power control, and channel coding. Key messages and methods arising from many recent work are listed, and open issues discussed. Through case studies, it is illustrated how “Layering as Optimization Decomposition” provides a common language to think about layered network architecture and a unifying approach to holistically design protocol stacks.

Keywords: Adaptive coding, Cross-layer design, Congestion control, Distributed algorithm, Lagrange duality, MAC, Network utility maximization, Optimization, Power control, Reverse engineering, Routing, TCP/IP, Scheduling, Stochastic control, Wireless ad hoc networks.

Contents

1 Introduction	4
1.1 Overview	4
1.1.1 Structures of layered protocol stack	4

^{*}Electrical Engineering Department, Princeton University

[†]Computer Science and Electrical Engineering Departments, California Institute of Technology

[‡]Electrical Engineering and Mathematics Departments, Princeton University

[§]Control and Dynamic Systems, California Institute of Technology

1.1.2	Network utility maximization	6
1.1.3	Summary	8
1.2	From Theory To Practice	8
1.2.1	Convex optimization	8
1.2.2	Practice	10
1.3	Notation	12
2	Horizontal Decomposition	12
2.1	TCP Congestion Control	12
2.1.1	Congestion control protocols	12
2.1.2	Reverse Engineering: Congestion control as distributed solution of basic NUM	16
2.1.3	Stability of distributed solution	18
2.1.4	Heterogeneous congestion control protocols	22
2.1.5	Forward Engineering: FAST	27
2.2	Medium Access Control	28
2.2.1	Reverse engineering: MAC as non-cooperative game	28
2.2.2	Forward engineering: Utility-optimal MAC protocol	33
3	Vertical Decomposition	38
3.1	Case Studies	38
3.1.1	Case 1: Jointly optimal congestion control and routing	38
3.1.2	Case 2: Jointly optimal congestion control and physical layer resource allocation	42
3.1.3	Case 3: Jointly optimal congestion and contention control	53
3.1.4	Case 4: Jointly optimal congestion control, routing, and scheduling	57
3.2	Decomposition Methods	59
3.2.1	Decoupling coupled constraints	59
3.2.2	Decoupling coupled objective	63
3.2.3	Alternative decompositions	65
3.3	Related Work	69
4	Future Research Directions	70
4.1	Modeling and Complexity Challenges	70

4.2	Exploration of Alternative Decompositions	70
4.3	Research Issues Involving “Time”	71
4.4	Stochastic NUM	71
4.4.1	Session level stochastic	72
4.4.2	Packet level stochastic	73
4.4.3	Channel level stochastic	74
4.4.4	Topology level stochastic	74
4.5	Nonconvex NUM	74
4.6	Network X-ities	75
5	Conclusion	76

1 Introduction

1.1 Overview

1.1.1 Structures of layered protocol stack

Layered architectures is one of the most fundamental and influential structures of network design. It adopts a modularized and often distributed approach to network coordination and resource allocation. Each layer controls a subset of the decision variables, and observes a subset of constant parameters and variables from other layers. Intuitively, layered architectures enable a scalable, evolvable, and implementable network while introducing potential risks to its manageability. There are clearly more than one way to “divide and conquer” the network design problem. From a data-plane performance point of view, some layering schemes may be more efficient or more fair than others. Focusing on resource allocation functionalities and using only performance metrics, this paper examines the question of “how to” and “how not to” layer. The limitations of our focus, in terms of semantics functionalities and “network X-ities” metrics, will be discussed at the end of the paper.

Each layer in the protocol stack hides the complexity of the layer below and provides a service to the layer above. While the general principle of layering is widely recognized as one of the key reasons for the enormous success of data networks, there is little quantitative understanding to guide a systematic, rather than an ad hoc, process of designing layered protocol stack for wired and wireless networks. One possible perspective to rigorously and holistically understand layering is to integrate the various protocol layers into a single coherent theory, by regarding them as carrying out an asynchronous distributed computation over the network to implicitly solve a global optimization problem. Different layers iterate on different subsets of the decision variables using local information to achieve individual optimality. Taken together, these local algorithms attempt to achieve a global objective. Such a framework of “Layering as Optimization Decomposition” exposes the interconnection between protocol layers and can be used to study rigorously the performance tradeoff in protocol layering, as different ways to modularize and distribute a centralized computation. Even though the design of a complex system will always be broken down into simpler modules, this theory will allow us to systematically carry out this layering process and explicitly trade off design objectives.

Based on the concepts of “networks as optimizers” and “layering as optimization”, the key idea in “Layering as Optimization Decomposition” is as follows. Different *decompositions* of an optimization problem, in the form of a generalized Network Utility Maximization (NUM), are mapped to different *layering* schemes in a communication network, with each decomposed subproblem corresponding to a layer, and functions of primal or Lagrange dual *variables* coordinating the subproblems correspond to the *interfaces* among the layers. Since different decompositions lead to alternative layering architectures, we can tackle the question “how to and how not to layer” by investigating the pros and cons of decomposition techniques. Furthermore, by comparing the objective function values under various forms of optimal decompositions and suboptimal decompositions, we can seek “separation theorems” among layers: conditions under which layering incurs no loss of optimality. Robustness of these separation theorems can be further characterized by sensitivity analysis in optimization theory: how much will the differences in the objective value (between different layering schemes) fluctuate as constant parameters in the

generalized NUM formulation are perturbed.

The mentality of “network as an optimizer” and “protocol as a distributed solution” to some global optimization problem (in the form of the basic NUM) has already been successfully tested in trials for Transmission Control Protocol (TCP) [40, 109, 41]. The key innovation from this line of work [46, 65, 70, 53, 69, 118, 64, 62] is to view each variant of congestion control protocol as a distributed algorithm to maximize the sum of source utilities over their transmission rates subject to capacity constraints. Other recent results also show how to reverse engineer Border Gateway Protocols (BGP) as solving the Stable Path Problem [32], and contention-based Medium Access Control (MAC) protocols as game-theoretic selfish utility maximization [56, 97]. Starting from a given protocol originally designed based on engineering heuristics, reverse engineering discovers the underlying mathematical problems being solved by the protocols and allows the application of derived insights to forward engineer improvements of the protocols.

These reverse engineering successes provide one of the justifications to employ generalized versions of NUM for systematic cross-layer design. Furthermore, utility of allocated resources to end users and elasticity of application traffic can both be modeled through general utility functions. Utility functions also provide a measure of resource allocation efficiency and lead to allocations satisfying fairness definitions. In general, they can be coupled across users, and may depend on not just rates, but also other metrics such as reliability, latency, jitter, and energy.

While the application needs give rise to the objective function, *i.e.*, network utility to be maximized, restrictions in the communication infrastructure are translated into many constraints of a generalized NUM problem. The resulting problem may be a very difficult nonconvex optimization with integer constraints. These generalized NUM problems highlights the end user utilities as the ultimate objective in network design. For example, benefits of innovations in the physical layer through better modulation and coding schemes are now characterized by the enhancement to applications rather than just the drop in bit error rates, which the users do not directly observe. An optimal solution to a generalized NUM formulations automatically establishes the benchmark for all layering schemes. Indeed, layering is a human engineering effort. The problem itself does not have any pre-determined layering architecture.

How to attain an optimal solution to a generalized NUM in a *modularized* and *distributed* way then becomes an overarching question. *Vertical* decompositions across modules and *horizontal* decompositions across disparate network elements can be conducted systematically through the theory of decomposition for nonlinear optimization. Implicit or explicit message passing quantifies the amount of information sharing and decision coupling required for a particular decomposition. There are many different ways to decompose a given problem, each of which corresponds to a different layering architecture. Even a different representation of the same NUM problem can lead to different decomposability structures even though the optimal solution remains the same. These decompositions, *i.e.*, layering schemes, have different characteristics in efficiency, robustness, asymmetry of information and control, and tradeoff between computation and communication. Some are “better” than others depending on the criteria set by the network users and operators. A systematic exploration in the space of alternative decompositions is possible, where each particular decomposition represents a holistically designed protocol stack.

Given the layers, crossing layers is tempting. For example, layers can be crossed for wired or wireless networks in at least the following ways:

- Information may be passed from one layer to another. For example, a TCP proxy at the base station of a wireless cellular network may be informed by the physical layer that a packet loss is due to channel fading and not congestion.
- Information from one layer may be used in another layer to either adapt its existing algorithm or create new diversity. For example, if the medium access layer informs the routing layer about its performance, multipath routing may be used to provide spatial diversity.
- Tasks may be jointly accomplished across the layers. For example, joint routing in the network layer and data compression in application layer may leverage the spatial redundancy in the sensor network measurements to reduce the network traffic load.
- Tasks may be re-divided among the layers. For example, error correction is performed in different forms in each of the application, transport, network, link, and physical layers. The task of ensuring the accuracy of the received bits may be re-allocated across the layers and some error checking functions may be removed from certain layers.

As evidenced by the large and ever growing number of papers on cross layer design over the last few years, we expect that there will be no shortage of cross layer ideas. However, any piecemeal design jointly over multiple layers does not contribute to the understanding of the structures of network architecture any more than the ad hoc design of just one layer. What seems to be lacking is a level ground for fair comparison among the variety of cross layer designs, a unified view on how to and how not to layer, basic principles rigorously quantified, and fundamental limits on the impacts of layer-crossing on network performance and robustness metrics.

“Layering as Optimization Decomposition” provides a candidate for such a unified framework.¹ What is unique about this framework is that it views the network as the optimizer itself, highlights the application needs as the optimization objective, provides a globally optimal performance benchmark, and leads to a systematic design of decomposed solution to attain the benchmark. Carrying the intellectual thread of “forward engineering” (solve a given problem) and “reverse engineering” (find the problem being solved by a given protocol) one step further, the framework naturally suggests “design for optimizability”. The difficulty of solving a particular set of subproblems can be an indication of a poorly constructed decomposition and should prompt the search for a better alternative.

1.1.2 Network utility maximization

The basic NUM problem is the following formulation [46], known as monotropic programming and studied since 1960s. As will be presented in Section 2.1, TCP variants have recently been reverse engineered to show that they are implicitly solving this problem, where source rate vector \mathbf{x} is the only optimization variable, and routing matrix

¹It is important to note that “Layering as Optimization Decomposition” is *not* the same as the generic phrase of “cross-layer optimization”.

\mathbf{R} and link capacity vector \mathbf{c} are both constants:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && \mathbf{R}\mathbf{x} \leq \mathbf{c}. \end{aligned}$$

Utility functions U_s are often assumed to be smooth, increasing, concave, and depends on local rates only, although recent investigations have removed some of these assumptions for applications where they are invalid. Utility functions can be determined based on any of the following: reverse-engineering (a given protocol description implicitly dictates the underlying utility function), user perception behavior models, application traffic elasticity, efficiency of resource allocation, and fairness among competing users.

Many of the papers on ‘‘Layering as Optimization Decomposition’’ are special cases of the following generic problem [13], one of the possible formulations of a generalized NUM for the entire protocol stack:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s, P_{e,s}) + \sum_j V_j(w_j) \\ & \text{subject to} && \mathbf{R}\mathbf{x} \leq \mathbf{c}(\mathbf{w}, \mathbf{P}_e) \\ & && \mathbf{x} \in \mathcal{C}_1(\mathbf{P}_e), \quad \mathbf{x} \in \mathcal{C}_2(\mathbf{F}) \text{ or } x \in \Pi \\ & && \mathbf{R} \in \mathcal{R}, \quad \mathbf{F} \in \mathcal{F}, \quad \mathbf{w} \in \mathcal{W}. \end{aligned} \tag{1}$$

Here, x_s denotes the rate for source s and w_j denotes the physical layer resource at network element j . The utility functions $\{U_s\}$ and $\{V_j\}$ may be any nonlinear, monotonic functions. \mathbf{R} is the routing matrix, and \mathbf{c} are the logical link capacities as functions of both physical layer resources \mathbf{w} and the desired decoding error probabilities \mathbf{P}_e . The issue of signal interference and power control can be captured in this functional dependency. The rates must also be constrained by the interplay between channel decoding reliability and other error control mechanisms like ARQ. This constraint set is denoted as $\mathcal{C}_1(\mathbf{P}_e)$. The issue of rate-reliability tradeoff and coding is captured in this constraint. The rates are further constrained by the medium access success probability, represented by the constraint set $\mathcal{C}_2(\mathbf{F})$ where \mathbf{F} is the contention matrix or the schedulability constraint set Π . The issue of packet collision and medium access control is captured in this constraint. The sets of possible physical layer resource allocation schemes, of possible scheduling or contention based medium access schemes, and of single-path or multi-path routing schemes are represented by $\mathcal{W}, \mathcal{F}, \mathcal{R}$, respectively. The optimization variables are $\mathbf{x}, \mathbf{w}, \mathbf{P}_e, \mathbf{R}, \mathbf{F}$. Holding some of the variables as constants and specifying some of these functional dependencies and constraint sets will then lead to a special class of this generalized NUM formulation.

A deterministic fluid model is used in the above formulations. Stochastic network utility maximization is an active research area, as discussed in Section 4.4, where stochastic models are imposed at session, packet, channel, and topology levels, raising new questions such as stochastic stability, average optimality, and outage performance.

Whether it is the basic, general, or stochastic NUM, there are three steps in the process: first formulate a specific NUM problem that also establishes the benchmark independent of layering possibilities, then devise a modularized and distributed solution following a particular decomposition, and finally explore the space of alternative decompositions that provide a choice of layered protocol stack and coupling across the layers.

In general, there are two types of objective functions: sum of utility functions by end users, which can be functions of rate, reliability, delay, jitter, or power level, and a network-wide cost function by network operators, which can be functions of congestion level, energy efficiency, network lifetime, or collective estimation error. Some of these utility functions may not have an additive structure. Maximizing a weighted sum of these utility functions, which is the focus of this paper, is only one of the possible formulations. An alternative is multi-objective optimization to characterize the Pareto-optimal tradeoff between the user objective and operator objective. Another formulation is game-theoretic between users and operators, or among users or operators themselves.

1.1.3 Summary

More than just an ensemble of specific cross-layer designs for existing protocol stacks, “Layering as Optimization Decomposition” is a mentality that views networks as optimizers, a common language that allows researchers to quantitatively compare alternative network architectures, and a suite of methods that facilitates a systematic design approach for modularized resource allocation.

	<i>Reverse Engineering</i>	<i>Forward Engineering</i>
<i>Horizontal Decomposition</i>	Sections 2.1.2 and 2.2.1	Sections 2.1.5 and 2.2.2
<i>Vertical Decomposition</i>	Section 3.1.1	Section 3.1

Table 1: Summary of content.

The power of “Layering as Optimization Decomposition” has been illustrated through many case studies carried out by various research groups in the last couple of years, generating considerable general insights in addition to the specific cross-layer designs. The summary list of key messages illustrate the conceptual simplicity in this rigorous and unifying framework, which is more important than any specific cross layer design. The summary list of main methods developed in many recent publications aims at popularizing these analytic techniques so that future research can invoke them readily.

1.2 From Theory To Practice

1.2.1 Convex optimization

Linear programming has found important applications in communication systems for several decades. In particular, network flow problems, *i.e.*, minimizing linear cost subject to linear flow conservation and capacity constraints, include important special cases such as the shortest path routing and maximum flow problems. Recently, there have been many research activities that utilize the power of recent developments in nonlinear convex optimization to tackle a much wider scope of problems in the analysis and design of communication systems. These research activities are driven by both new demands in the study of communications and networking, and new tools emerging from optimization theory. In particular, a major breakthrough in optimization over the last two decades has been the

<i>Key Message</i>	<i>Section</i>
Existing protocols in layers 2,3,4 have been reverse engineered	Section 2.1.2, 2.2.1
Reverse engineering leads to better design	Section 2.1.5, 2.2.2
There is one unifying approach to cross-layer design	Section 3.1
Loose coupling through layering price	Section 3.1
Queue length often a right layering price, but not always	Section 3.1
Many alternatives in decompositions and layering architectures	Section 3.2
Convexity is key to proving global optimality	Section 3.1
Decomposability is key to designing distributed solution	Section 3.2
Still many open issues in modeling, stochastic dynamics, and nonconvex formulations	Section 4

Table 2: Summary of 9 key messages.

<i>Main Method</i>	<i>Section</i>
Reverse engineering cooperative protocol as an optimization algorithm	Section 2.1.2
Lyapunov function construction to show stability	Section 2.1.3
Proving convergence of dual descent algorithm	Section 2.1.3
Proving stability by singular perturbation theory	Section 2.1.3
Proving stability by passivity argument	Section 2.1.3
Proving equilibrium properties through vector field representation	Section 2.1.4
Reverse engineering non-cooperative protocol as a game	Section 2.2.1
Verifying contraction mapping by bounding the Jacobian's norm	Section 2.2.1
Analyzing cross-layer interaction through generalized NUM	Section 3.1.1
Log change of variables for decoupling, and computing minimum curvature needed	Section 2.2.2
Dual decomposition for jointly optimal cross layer design	Section 3.1.2
Computing conditions under which a general constraint set is convex	Section 3.1.2
Introducing an extra "layer" to decouple the problem	Section 3.1.2
End user generated pricing	Section 3.1.2
Different timescales of protocol stack interactions through different decomposition methods	Section 3.1.3
Maximum differential congestion pricing for node-based back-pressure scheduling	Section 3.1.4
Absorbing routing functionality into congestion control and scheduling	Section 3.1.4
Primal and dual decomposition for coupling constraints	Section 3.2.1
Consistency pricing for decoupling coupled objective	Section 3.2.2
Partial and hierarchical decomposition for architectural alternatives	Section 3.2.3

Table 3: Summary of 20 main methods.

development of powerful theoretical tools, as well as highly efficient computational algorithms like the interior-point method, for nonlinear convex optimization, *i.e.*, minimizing a convex function (or maximizing a concave function as is often seen in this paper) subject to upper bound inequality constraints on other convex functions and affine equality constraints:

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, m \\ & && \mathbf{A}\mathbf{x} = \mathbf{c} \end{aligned} \tag{2}$$

where the variables are $\mathbf{x} \in \mathbf{R}^n$. The constant parameters are $\mathbf{A} \in \mathbf{R}^{l \times n}$ and $\mathbf{c} \in \mathbf{R}^l$. The objective function f_0 to be minimized and m constraint functions f_i are convex functions.

Since the early 1990s, it has been recognized that the watershed between efficiently solvable optimization problems and intractable ones is *convexity*. It is well known that for a convex optimization problem, a local minimum is also a global minimum. The Lagrange duality theory is also well-developed for convex optimization. For example, the duality gap is zero under constraint qualification conditions, such as Slater’s condition [7] that requires the existence of a strictly feasible solution to nonlinear inequality constraints. When put in an appropriate form with the right data structure, a convex optimization problem can also be efficiently solved numerically, such as the primal-dual interior-point method, which has worst-case polynomial-time complexity for a large class of functions and scales gracefully with problem size in practice.

Special cases of convex optimization include convex Quadratic Programming (QP), Second Order Cone Programming (SOCP), and Semidefinite Programming (SDP) [7], as well as seemingly non-convex optimization problems that can be readily transformed into convex problems, such as Geometric Programming (GP) [12]. The last decade has witnessed the appreciation-application cycle for convex optimization, where more applications are developed as more people start to appreciate the capabilities of convex optimization in modeling, analyzing, and designing communication systems.

The phrase “optimization of communication systems” in fact carries three different meanings. In the most straight-forward way, an analysis or design problem in a communication system may be formulated as either minimizing a cost or maximizing a utility function over a set of variables confined within a constraint set. In a more subtle and recent approach, emphasized in this paper, a given network protocol may be interpreted as a distributed algorithm solving an implicitly defined global optimization problem. In yet another approach, the underlying theory of a network control method or a communication strategy may be generalized using nonlinear optimization techniques, thus extending the scope of applicability of the theory.

1.2.2 Practice

Industry adoption of “Layering as Optimization Decomposition” has already started. For example, insights from reverse-engineering TCP has led to an improved version of TCP in the FAST Project (Fast AQM Scalable TCP) [40, 109, 41]. Putting end-user application utilities as the objective function has led to a new way to leverage innovations in the physical and link layers beyond the standard metrics such as bit error rate, *e.g.*, in the “FAST Copper” Project (here FAST stands for Frequency, Amplitude, Space, Time) for an order-of-magnitude boost to

rates in fiber/DSL broadband access systems [25].

FAST [24] is a joint project between CS, CDS, EE and Physics departments at Caltech and UCLA, and involves partners at various national laboratories around the world. It integrates theory, algorithms, implementation, and experiment so that they inform and influence each other intimately. Its goal is to understand the current TCP congestion control, design new algorithms, implement and test them in real high-speed global networks. We have developed a duality model that interprets any TCP/AQM algorithm as a distributed asynchronous primal-dual algorithm carried out over the Internet in real-time to solve the basic NUM. Different algorithms differ merely in the utility functions they implicitly optimize. The model allows us to understand the limitations of the current TCP and design new algorithms. Until about six years ago, the state of the art in TCP research has been simulation-based using simplistic scenarios, with often a single bottleneck link and a single class of algorithms. We have now a theory that can predict the equilibrium behavior of an arbitrary network under any TCP-like algorithm. Moreover, we can prove, and design, their stability properties in the presence of feedback delay for arbitrary networks. As explained in detail in Section 2.1.5, we have implemented the insights from this series of theoretical work in a software prototype FAST TCP and it has been used to break world records in data transfer in the last few years.

FAST Copper [25] is a joint project at Princeton, Stanford, and Fraser Research Institute, aiming at providing at least an order of magnitude increase in DSL broadband access speed to 50-100 Mbps, through a joint optimization of Frequency, Amplitude, Time and Space dimensions to overcome the attenuation and crosstalk bottlenecks in today's DSL systems. The key idea is to treat the DSL network as a multiple input multiple output system rather than a point-to-point channel, thus leveraging the opportunities of multi-user cooperation and mitigating the current bottleneck due to multi-user competition. The overarching research challenge is to understand how the resulting highly nonlinear and complicated optimization problems across the application layer, transport layer, link layer, and physical layer can be dynamically solved. "Layering as Optimization Decomposition" provides an analytic framework for these dynamic resource allocation problems in the interference environment of DSL broadband access.

Clean-slate design of the entire protocol stack is another venue of application of "Layering as Optimization Decomposition". For example, Internet 0 [39] is a project initiated at the Center for Bits and Atoms at MIT and jointly pursued by an industrial consortium. Its goal is to develop theory, algorithms, protocols, and implementations to connect a large number of small devices. The focus here is not the transmission speed, which will be much smaller than on the wired Internet, but the sheer number of small devices that need simple connectivity but are constrained by limited power, heterogeneity, and complex interactions with the physical environment. Eventually, "Layering as Optimization Decomposition" may be used to develop CAD (Computer Aided Design) tools for protocol design and implementation. The idea is to treat protocols for Internet 0 as distributed algorithms to solve an appropriately generalized NUM problem, and automatically compile and optimize computer codes to be downloaded onto Internet 0 nodes from high level protocol specifications.

1.3 Notation

Unless otherwise specified, vectors are denoted in boldface small letters, e.g., \mathbf{x} with x_i as its i th component; matrices are denoted by boldface capital letters, e.g., \mathbf{H} , \mathbf{W} , \mathbf{R} ; and sets of vectors or matrices are denoted by script letters, e.g., \mathcal{W}_n , \mathcal{W}_m , \mathcal{R}_n , \mathcal{R}_m . Inequalities between two vectors denote component-wise inequalities. We will use the terms “user”, “source”, “session”, and “connection” interchangeably.

Due to the wide coverage of materials in this survey paper, notational conflicts occasionally arise. Consistency is maintained within any section, and main notation is summarized in the tables of notation for each section.

2 Horizontal Decomposition

It is well-known that physical layer algorithms try to solve the data transmission problem formulated by Shannon: maximizing data rate subject to the constraint of vanishing error probability. Widely used network protocols, such as TCP, BGP, and IEEE 802.11 DCF, were instead designed based primarily on engineering intuitions and ad hoc heuristics. Recent progress have put protocols in layers 2-4 of the standard reference model on a mathematical foundation as well:

- The congestion control functionality of TCP has been reverse engineered to be implicitly solving the basic NUM problem [62, 91]. While *heterogeneous* congestion control protocols do not solve an underlying NUM problem, its equilibrium and dynamic properties can still be analyzed through a vector field representation and the Poincare-Hopf index theorem [101], which show that bounded heterogeneity implies global uniqueness and local stability of network equilibrium.
- IGP (Interior Gateway Protocol) of IP routing is known to solve variants of the shortest path problem, and the policy-based routing protocol in BGP (Border Gateway Protocol) has recently been modeled as the solution to the Stable Path Problem [32].
- Scheduling based MAC protocols are known to solve variants of maximum weight matching, and random access (contention based MAC) protocols have recently been reverse engineered as a non-cooperative utility maximization game [56, 97].

In this section, the reverse and forward engineering results for TCP congestion control and random access MAC are summarized.

2.1 TCP Congestion Control

2.1.1 Congestion control protocols

Congestion control is a distributed mechanism to share link capacities among competing users. In this section, a network is modeled as a set L of links (scarce resources) with finite capacities $\mathbf{c} = (c_l, l \in L)$. They are shared by

a set N of sources indexed by s . Each source s uses a set $L(s) \subseteq L$ of links. Let $S(l) = \{s \in N \mid l \in L(s)\}$ be the set of sources using link l . The sets $\{L(s)\}$ define an $L \times N$ routing matrix²

$$R_{ls} = \begin{cases} 1 & \text{if } l \in L(s), \text{ i.e., source } s \text{ uses link } l \\ 0 & \text{otherwise} \end{cases}$$

Associated with each source s is its transmission rate $x_s(t)$ at time t , in packets/sec. Associated with each link l is a scalar congestion measure $\lambda_l(t) \geq 0$ at time t . We will call $\lambda_l(t)$ the link (*congestion*) *price*.

A congestion control algorithm consists of two components: a source algorithm that dynamically adjusts its rate $x_s(t)$ in response to prices $\lambda_l(t)$ in its path, and a link algorithm that updates, implicitly or explicitly, its price $\lambda_l(t)$ and sends it back, implicitly or explicitly, to sources that use link l . On the current Internet, the source algorithm is carried out by Transmission Control Protocol (TCP), and the link algorithm is carried out by (active) queue management (AQM) schemes such as DropTail or RED [29]. Different protocols use different metrics to measure congestion, *e.g.*, TCP Reno [30, 93] and its variants, use loss probability as the congestion measure, and TCP Vegas [8] and FAST [40, 109] use queueing delay as the congestion measure [64]. Both are implicitly updated at the links and implicitly fed back to sources through end-to-end loss or delay, respectively.

In this section, we show that a large class of congestion control algorithms can be interpreted as distributed algorithms to solve a global optimization problem. Specifically, we associate with each source s a utility function $U_s(x_s)$ as a function of its rate x_s . Consider the network utility maximization problem proposed in [46], which we will refer to as the basic NUM:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && \mathbf{R}\mathbf{x} \leq \mathbf{c}, \end{aligned} \quad (3)$$

and its Lagrangian dual problem [65]:

$$\text{minimize}_{\boldsymbol{\lambda} \geq 0} \quad D(\boldsymbol{\lambda}) := \sum_s \max_{x_s \geq 0} \left(U_s(x_s) - x_s \sum_l R_{ls} \lambda_l \right) + \sum_l c_l \lambda_l. \quad (4)$$

We now present a general model of congestion control algorithms and show that they can be interpreted as distributed algorithms to solve NUM (3) and its dual (4).

Let $y_l(t) = \sum_s R_{ls} x_s(t)$ be the aggregate source rate at link l and let $q_s(t) = \sum_l R_{ls} \lambda_l(t)$ be the end-to-end price for source s . In vector notation, we have

$$\mathbf{y}(t) = \mathbf{R}\mathbf{x}(t) \quad \text{and} \quad \mathbf{q}(t) = \mathbf{R}^T \boldsymbol{\lambda}(t).$$

Here, $\mathbf{x}(t) = (x_s(t), s \in N)$ and $\mathbf{q}(t) = (q_s(t), s \in N)$ are in \mathbf{R}_+^N , and $\mathbf{y}(t) = (y_l(t), l \in L)$ and $\boldsymbol{\lambda}(t) = (\lambda_l(t), l \in L)$ are in \mathbf{R}_+^L .

In each period, the source rates $x_s(t)$ and link prices $\lambda_l(t)$ are updated based on local information. Source s can observe its own rate $x_s(t)$ and the end-to-end price $q_s(t)$ of its path, but not the vector $\boldsymbol{\lambda}(t)$, nor other components

²We abuse notation to use L and N to denote both sets and their cardinalities.

of $\mathbf{x}(t)$ or $\mathbf{q}(t)$. Similarly, link l can observe just local price $\lambda_l(t)$ and flow rate $y_l(t)$. The source rates $x_s(t)$ are updated according to

$$x_s(t+1) = F_s(x_s(t), q_s(t)) \quad (5)$$

for some nonnegative functions F_s . The link congestion measure $\lambda_l(t)$ is adjusted in each period based only on $\lambda_l(t)$ and $y_l(t)$, and possibly some internal (vector) variable $\mathbf{v}_l(t)$, such as the queue length at link l . This can be modeled by some functions (G_l, \mathbf{H}_l) : for all l ,

$$\lambda_l(t+1) = G_l(y_l(t), \lambda_l(t), \mathbf{v}_l(t)) \quad (6)$$

$$\mathbf{v}_l(t+1) = \mathbf{H}_l(y_l(t), \lambda_l(t), \mathbf{v}_l(t)) \quad (7)$$

where G_l are non-negative so that $\lambda_l(t) \geq 0$. Here, F_s model TCP algorithms (e.g., Reno or Vegas) and (G_l, \mathbf{H}_l) model AQM's (e.g., RED, REM). We will often refer to AQM's by G_l , without explicit reference to the internal variable $\mathbf{v}_l(t)$ or its adaptation \mathbf{H}_l . We now present some examples.

TCP Reno/RED

The congestion control algorithm in the large majority of current TCP implementations is (enhanced version of) TCP Reno first proposed in [30]. A source maintains a parameter called *window size* that determines the number of packets it can transmit in a round-trip time (RTT), the time from sending a packet to receiving its acknowledgment from the destination. This implies that the source rate is approximately equal to the ratio of window size to RTT, in packets per second. The basic idea of (the congestion avoidance phase of) TCP Reno is for a source to increase its window by one packet in each RTT and half its window when there is a packet loss. This can be modeled by (see e.g., [62, 53]) the source algorithm $F_s(t+1) := F_s(x_s(t), q_s(t))$:

$$F_s(t+1) = \left[x_s(t) + \frac{1}{T_s} - \frac{2}{3} q_s(t) x_s^2(t) \right]^+ \quad (8)$$

where T_s is the RTT of source s , i.e., the time it takes for s to send a packet and receive its acknowledgement from the destination. Here we assume T_s is a constant even though in reality its value depends on the congestion level and is generally time-varying. The quadratic term signifies the property that, if rate doubles, the multiplicative decrease occurs at twice the frequency with twice the amplitude.

The AQM mechanism of RED [29] maintains two internal variables, the instantaneous queue length $b_l(t)$ and average queue length $r_l(t)$. They are updated according to

$$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+ \quad (9)$$

$$r_l(t+1) = (1 - \omega_l)r_l(t) + \omega_l b_l(t) \quad (10)$$

where $\omega_l \in (0, 1)$. Then, (the ‘‘gentle’’ version of) RED marks a packet with a probability $\lambda_l(t)$ that is a piecewise linear increasing function of $r_l(t)$ with constants $\rho_1, \rho_2, M_l, \underline{b}_l, \bar{b}_l$:

$$\lambda_l(t) = \begin{cases} 0 & r_l(t) \leq \underline{b}_l \\ \rho_1(r_l(t) - \underline{b}_l) & \underline{b}_l \leq r_l(t) \leq \bar{b}_l \\ \rho_2(r_l(t) - \bar{b}_l) + M_l & \bar{b}_l \leq r_l(t) \leq 2\bar{b}_l \\ 1 & r_l(t) \geq 2\bar{b}_l. \end{cases} \quad (11)$$

The equations (9)–(11) define the model (\mathbf{G}, \mathbf{H}) for RED.

TCP Vegas/DropTail

A duality model of Vegas has been developed and validated in [64]; see also [70]. We consider the situation where the buffer size is large enough to accommodate the equilibrium queue length so that Vegas sources can converge to the unique equilibrium. In this case, there is no packet loss in equilibrium.

Unlike TCP Reno, Vegas uses queueing delay as congestion measure, $\lambda_l(t) = b_l(t)/c_l$, where $b_l(t)$ is the queue length at time t . The update rule $G_l(y_l(t), \lambda_l(t))$ is given by (dividing both sides of (9) by c_l):

$$\lambda_l(t+1) = \left[\lambda_l(t) + \frac{y_l(t)}{c_l} - 1 \right]^+. \quad (12)$$

Hence, AQM for Vegas does not involve any internal variable. The update rule $F_s(x_s(t), q_s(t))$ for source rate is given by:

$$x_s(t+1) = x_s(t) + \frac{1}{T_s^2(t)} \mathbf{1}(\alpha_s d_s - x_s(t) q_s(t)) \quad (13)$$

where α_s is a parameter of Vegas, d_s is the round-trip propagation delay of source s , and $\mathbf{1}(z) = 1$ if $z > 0$, -1 if $z < 0$, and 0 if $z = 0$. Here $T_s(t) = d_s + q_s(t)$ is the RTT at time t .

FAST/DropTail

The FAST algorithm is developed in [40, 109, 41]. Let d_s denote the round-trip propagation delay of source s . Let $\lambda_l(t)$ denote the queueing delay at link l at time t . Let $q_s(t) = \sum_l R_{ls} \lambda_l(t)$ be the round-trip queueing delay, or in vector notation, $\mathbf{q}(t) = \mathbf{R}^T \boldsymbol{\lambda}(t)$. Each source s adapts its window $W_s(t)$ periodically according to:

$$W_s(t+1) = \gamma \left(\frac{d_s W_s(t)}{d_s + q_s(t)} + \alpha_s \right) + (1 - \gamma) W_s(t) \quad (14)$$

where $\gamma \in (0, 1]$ and $\alpha_s > 0$ is a protocol parameter. A key departure from the model described above and those in the literature is that, here, we assume that a source's *send rate* cannot exceed the *throughput* it receives. This is justified because of self-clocking: within one round-trip time after a congestion window is increased, packet transmission will be clocked at the same rate as the throughput the flow receives. A consequence of this assumption is that the link queueing delay vector, $\boldsymbol{\lambda}(t)$, is determined implicitly by the instantaneous window size in a *static* manner: given $W_s(t) = W_s$ for all s , the link queueing delays $\lambda_l(t) = \lambda_l \geq 0$ for all l are given by:

$$\sum_s R_{ls} \frac{W_s}{d_s + q_s(t)} \quad \begin{cases} = c_l & \text{if } \lambda_l(t) > 0 \\ \leq c_l & \text{if } \lambda_l(t) = 0 \end{cases} \quad (15)$$

where again $q_s(t) = \sum_l R_{ls} \lambda_l(t)$.

Hence, FAST is defined by the discrete-time model (14)–(15) of window evolution. The sending rate is then defined as $x_s(t) := W_s(t)/(d_s(t) + q_s(t))$.

2.1.2 Reverse Engineering: Congestion control as distributed solution of basic NUM

Under mild assumptions on $(\mathbf{F}, \mathbf{G}, \mathbf{H})$, it can be shown using Kakutani's fixed point theorem that equilibrium $(\mathbf{x}, \boldsymbol{\lambda})$ of (5)–(7) exists and is unique [70, 101]. The fixed point of (5) defines an implicit relation between equilibrium rate x_s and end-to-end congestion measure q_s :

$$x_s = F_s(x_s, q_s).$$

Assume F_s is continuously differentiable and $\partial F_s / \partial q_s \neq 0$ in the open set $A := \{(x_s, q_s) | x_s > 0, q_s > 0\}$. Then, by the implicit function theorem, there exists a unique continuously differentiable function f_s from $\{x_s > 0\}$ to $\{q_s > 0\}$ such that

$$q_s = f_s(x_s) > 0. \quad (16)$$

To extend the mapping between x_s and q_s to the closure of A , define

$$f_s(0) = \inf \{q_s \geq 0 \mid F_s(0, q_s) = 0\}. \quad (17)$$

If $(x_s, 0)$ is an equilibrium point, $F_s(x_s, 0) = x_s$, then define

$$f_s(x_s) = 0 \quad (18)$$

Define the utility function of each source s as

$$U_s(x_s) = \int f_s(x_s) dx_s, \quad x_s \geq 0, \quad (19)$$

which is unique up to a constant.

Being an integral, U_s is a continuous function. Since $f_s(x_s) = q_s \geq 0$ for all x_s , U_s is nondecreasing. We assume that f_s is a nonincreasing function – the more severe the congestion, the smaller the rate. This implies that U_s is concave. If f_s is strictly decreasing, then U_s is strictly concave since $U_s''(x_s) < 0$. An increasing utility function implies a greedy source (a larger rate yields a higher utility) and concavity implies diminishing marginal return.

We assume

- C1: For all $s \in S$ and $l \in L$, F_s and G_l are non-negative functions. F_s are continuously differentiable and $\partial F_s / \partial q_s \neq 0$ in $\{(x_s, q_s) | x_s > 0, q_s > 0\}$; moreover, f_s in (16) are strictly decreasing.
- C2: \mathbf{R} has full row rank.
- C3: If $\lambda_l = G_l(y_l, \lambda_l, \mathbf{v}_l)$ and $\mathbf{v}_l = \mathbf{H}_l(y_l, \lambda_l, \mathbf{v}_l)$, then $y_l \leq c_l$ with equality if $\lambda_l > 0$.

Condition C1 guarantees that $(\mathbf{x}(t), \boldsymbol{\lambda}(t)) \geq 0$ and $(\mathbf{x}^*, \boldsymbol{\lambda}^*) \geq 0$, and that utility functions U_s exist and are strictly concave. C2 guarantees uniqueness of equilibrium price vector $\boldsymbol{\lambda}^*$. C3 guarantees the primal feasibility and complementary slackness of $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$. We can regard congestion control algorithms (5)–(7) as distributed primal-dual algorithms to solve the NUM (3) and its dual (4) [62].

Theorem 1. *Suppose assumptions C1 and C2 hold. Then (5)–(7) has a unique equilibrium $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$. Moreover, it solves the primal problem (3) and the dual problem (4) with utility function given by (19) if and only if C3 holds.*

Hence the various TCP/AQM protocols can be modeled as different distributed solutions $(\mathbf{F}, \mathbf{G}, \mathbf{H})$ to solve (3) and its dual (4), with different utility functions U_s . Theorem 1 characterizes a large class of protocols $(\mathbf{F}, \mathbf{G}, \mathbf{H})$ that admits such an interpretation. This interpretation is the consequence of end-to-end control: it holds as long as the end-to-end congestion measure to which the TCP algorithm reacts is the *sum* of the constituent link congestion measures, and that the link prices are independent of sources (this would not be true in the heterogeneous protocol case as in Subsection 2.1.4). Note that the definition of utility function U_s depends only on TCP algorithm F_s . The role of AQM (\mathbf{G}, \mathbf{H}) is to ensure that the complementary slackness condition of problem (5)–(7) is satisfied (condition C3). The complementary slackness has a simple interpretation: AQM should match input rate to capacity to maximize utilization at every bottleneck link. Any AQM that stabilizes queues possesses this property and generates a Lagrange multiplier vector $\boldsymbol{\lambda}^*$ that solves the dual problem.

The utility functions of several proposed TCP algorithms turn out to belong to a simple class of functions defined in [70] that is parameterized by a scalar parameter $\alpha_s \geq 0$:

$$U_s(x_s) = \begin{cases} w_s \log x_s & \alpha_s = 1 \\ w_s(1 - \alpha_s)^{-1} x_s^{1-\alpha_s} & \alpha_s \neq 1 \end{cases}$$

where weight $w_s > 0$. In particular, TCP Vegas, FAST, and Scalable TCP correspond to $\alpha_s = 1$, HTCP to $\alpha_s = 1.2$, TCP Reno to $\alpha_s = 2$, and maxmin fairness to $\alpha_s = \infty$. Maximizing α -fair utility leads to optimizers that satisfy the definition of α -fair resource allocation in the economics literature.

Method 1. *Reverse engineering cooperative protocol as an optimization algorithm.*

The potentials and risks of networks comes from the interconnection of local algorithms. Often, interesting and counter-intuitive behaviors arise in such a setting where users interact through multiple shared links in intricate and surprising ways. Reverse engineering of TCP/AQM has also led to a deeper understanding of throughput and fairness behavior in large scale TCP networks. For example, there is a general belief that one can design systems to be efficient or fair, but often not both. Many papers in the networking, wireless, and economics literature provide concrete examples in support of this intuition. We proved in [98] an exact condition under which this conjecture is true for general TCP networks using the duality model of TCP/AQM. This condition allows us to produce the first counter-example and trivially explains all the supporting examples we found in the literature. Surprisingly, in our counter-example, a fairer throughput allocation is always more efficient. It implies for example that maxmin fair allocation can achieve higher aggregate throughput on certain networks. Intuitively, we might expect that the aggregate throughput will always rise as long as some links increase their capacities and no links decrease theirs.

This turns out not to be the case, and we characterize exactly the condition under which this is true in general TCP networks. Not only can the aggregate throughput be reduced when some link increases its capacity, more strikingly, it can also be reduced even when all links increase their capacities by the same amount. Moreover, this holds for all fair bandwidth allocations. This paradoxical result seems less surprising in retrospect: according to the duality model of TCP/AQM, raising link capacities always increases the aggregate utility, but mathematically there is no a priori reason that it should also increase the aggregate throughput. If all links increase their capacities proportionally, however, the aggregate throughput will indeed increase, for α -fair utility functions.

2.1.3 Stability of distributed solution

Theorem 1 characterizes the equilibrium structure of congestion control algorithms (5)–(7). We now discuss its stability. We assume conditions C1 and C2 in this subsection so that there is a unique equilibrium $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$.

Roughly speaking, an algorithm is said to be locally asymptotically stable if it converges to the unique equilibrium starting from a neighborhood of the equilibrium, and globally asymptotically stable if it converges starting from any initial state. Global asymptotic stability in the presence of feedback delay is desirable but generally hard to prove. Most papers in the literature analyze global asymptotic stability in the absence of feedback delay, or local stability in the presence of feedback delay. Proof techniques that have been used for global asymptotic stability in the absence of feedback delay include Lyapunov stability theorem, gradient decent method, passivity technique, and singular perturbation theory. In the following, we summarize some representative algorithms and illustrate how these methods are used to prove their stability in the absence of feedback delay. For analysis with delay, see, e.g., [78, 79, 103, 104] for local stability of linearized systems and [65, 81, 80, 84] for global stability; see also surveys in [45, 66, 91] for further references. In particular, unlike the Nyquist argument, [81] handles nonlinearity and delay with Lyapunov functionals.

Consider the algorithm (using a continuous-time model) of [46]:

$$\dot{x}_s = \kappa_s x_s(t) (U'_s(x_s(t)) - q_s(t)) \quad (20)$$

$$\lambda_l(t) = g_l(y_l(t)) \quad (21)$$

where $\kappa_s > 0$ is a constant. This is called a primal algorithm which has come to mean that there is dynamics only in the source control law but not the link control law. To motivate (20)–(21), note that $q_s(t)$ is the unit price for bandwidth that source s faces end-to-end. The marginal utility $U'_s(x_s(t))$ can be interpreted as source s 's willingness to pay when it transmits at rate $x_s(t)$. Then, according to (20), source s increases its rate (demand for bandwidth) if the end-to-end bandwidth price is less than s 's willingness to pay, and decreases it otherwise. Since g_l is an increasing function, the price increases as the aggregate demand for bandwidth at link l is large. To prove that (20)–(21) is globally asymptotically stable, consider the function

$$V(\mathbf{x}) := \sum_s U_s(x_s) - \sum_l \int_0^{y_l} g_l(z) dz. \quad (22)$$

Using (20)–(21), it is easy to check that

$$\dot{V} := \frac{d}{dt}V(\mathbf{x}(t)) = \begin{cases} > 0 & \text{for all } \mathbf{x}(t) \neq \mathbf{x}^* \\ = 0 & \text{if } \mathbf{x}(t) = \mathbf{x}^* \end{cases}$$

where \mathbf{x}^* is the unique maximizer of the strictly concave function $V(\mathbf{x})$. Hence $V(\mathbf{x})$ is a Lyapunov function for the dynamical system (20)–(21), certifying its global asymptotic stability. The function $V(\mathbf{x})$ in (22) can be interpreted as the penalty-function version of the NUM (3). Hence the algorithm (20)–(21) can also be thought of as a gradient ascent algorithm to solve the approximate NUM.

Method 2. *Lyapunov function construction to show stability.*

A dual algorithm is proposed in [65]:

$$\lambda_l(t+1) = [\lambda_l(t) + \gamma(y_l(t) - c_l)]^+ \quad (23)$$

$$x_s(t) = U_s'^{-1}(q_s(t)) \quad (24)$$

where $U_s'^{-1}$ is the inverse of U_s' . The algorithm is derived as the gradient projection algorithm to solve the dual (4) of NUM. The source algorithm (24) is called the demand function in economics: the larger the end-to-end bandwidth price $q_s(t)$, the smaller the demand $x_s(t)$. The link algorithm (23) is the law of supply and demand: if demand $y_l(t)$ exceeds supply, increase the price $\lambda_l(t)$; otherwise, decrease it. By showing that the gradient $\nabla D(\boldsymbol{\lambda})$ of the dual objective function in (4) is Lipschitz, it is proved in [65] that, provided the stepsize γ is small enough, $\mathbf{x}(t)$ converges to the unique primal optimal solution of NUM and $\boldsymbol{\lambda}(t)$ converges to its unique dual solution. The idea is to show that the dual objective function $D(\boldsymbol{\lambda}(t))$ strictly decreases in each step t . Hence one can regard $D(\boldsymbol{\lambda})$ as a Lyapunov function in discrete time³. The same idea is extended in [65] to prove global asymptotic stability in an asynchronous environment where the delays between sources and links can be substantial, diverse, and time-varying, sources and links can communicate at different times and with different frequencies, and information can be outdated or out of order.

Method 3. *Proving convergence of dual descent algorithm through descent lemma.*

Several variations of the primal and dual algorithms above can all maintain local stability in the presence of feedback delay [78, 79, 103, 104]. They are complementary in the sense that the primal algorithm has dynamics only at the sources, allows arbitrary utility functions and therefore arbitrary fairness, but typically has low link utilization, whereas the dual algorithm has dynamics only at the links, achieves full link utilization, but requires a specific class of utility functions (fairness) to maintain local stability in the presence of arbitrary feedback delays. The next algorithm has dynamics at both. It allows arbitrary utility functions, achieves arbitrarily close to full link utilization, and can maintain local stability in the presence of feedback delay. Algorithms that have dynamics at both links and sources are called primal-dual algorithms. The algorithm of [52] extends the primal algorithm (20)–(21) to a primal-dual algorithm and the algorithm of [79] extends the dual algorithm (23)–(24) to a primal-dual

³Indeed, for a continuous-time version of (23)–(24), it is trivial to show that $D(\boldsymbol{\lambda})$ is a Lyapunov function.

algorithm. The paper [79] focuses on local stability in the presence of feedback delay. We now summarize the proof technique in [52] for global stability in the absence of feedback delay.

The algorithm of [52] uses a source algorithm that is similar to (20):

$$\dot{x}_i(t) = w_i - \frac{1}{U'_i(x_i(t))} \sum_l R_{li} \lambda_l(t). \quad (25)$$

Its link algorithm AVQ (adaptive virtual queue) maintains an internal variable at each link called the virtual capacity \tilde{c}_l that is dynamically updated:

$$\dot{\tilde{c}}_l = \begin{cases} \frac{\gamma}{\partial g_l / \partial \tilde{c}_l} (c_l - y_l(t)) & \text{if } \tilde{c}_l \geq 0 \\ 0 & \text{if } \tilde{c}_l = 0 \text{ and } y_l(t) > c_l \end{cases} \quad (26)$$

where $\gamma > 0$ is a gain parameter and g_l is a link “marking” function that maps aggregate flow rate $y_l(t)$ and virtual capacity \tilde{c}_l into a price:

$$\lambda_l(t) = g_l(y_l(t), \tilde{c}_l(t)). \quad (27)$$

Using singular perturbation theory, it is proved in [52] that, under (25)–(27), $x(t)$ converges exponentially to the unique solution of the NUM, provided γ is small enough. Furthermore, $\lambda(t)$ then converges to the optimal (Lagrange multiplier) of the dual problem. The idea is to separately consider the stability of two approximating subsystems that are at different timescales when γ is small. The boundary-layer system approximates the source dynamics and assumes that the virtual capacity \tilde{c}_l are constants at the fast timescale:

$$\dot{x}_s = w_s - \frac{1}{U'_s(x_s(t))} \sum_l R_{ls} g_l(y(t), \tilde{c}_l). \quad (28)$$

The reduced system approximates the link dynamics and assumes the source rates x_s are the unique maximizers of (22):

$$\dot{\tilde{c}}_l = c_l - y_l \quad (29)$$

where $y_l = \sum_s R_{ls} x_s$ are constants and x_s are the unique maximizers of $V(\mathbf{x})$ defined in (22). Now we already know from above that the boundary-layer system (28) is asymptotically stable. In [52], it is further shown that it is exponentially stable uniformly in \tilde{c} , and that the reduced system (29) is exponentially stable provided the trajectory remains in a compact set. Singular perturbation theory then implies that the original system (25)–(27) is globally exponentially stable provided γ is small enough (and the initial state $(\mathbf{x}(0), \boldsymbol{\lambda}(0))$ is in a compact set).

Method 4. *Proving stability by singular perturbation theory.*

A different approach to prove global asymptotic stability for primal-dual algorithms uses passivity techniques developed in [110]. A system, described by its state $\mathbf{z}(t)$, input $\mathbf{u}(t)$ and output $\mathbf{v}(t)$, is called *passive* if there are positive semidefinite functions $V(\mathbf{x}) \geq 0$ and $W(\mathbf{x}) \geq 0$ such that

$$\dot{V}(\mathbf{x}(t)) \leq -W(\mathbf{x}(t)) + \mathbf{u}^T(t)\mathbf{v}(t).$$

$V(\mathbf{x})$ is called a storage function. The passivity theorem states that the feedback interconnection of two passive systems is globally asymptotically stable and

$$V(\mathbf{x}) := V_1(\mathbf{x}) + V_2(\mathbf{x})$$

is a Lyapunov function for the feedback system, provided one of the storage functions V_1, V_2 of the individual systems are positive definite and radially unbounded. Consider the following variants of the primal algorithm (20)–(21):

$$\dot{x}_s(t) = \kappa_s(U'_s(x_s(t)) - q_s(t)) \quad (30)$$

$$\lambda_l(t) = g_l(y_l(t)). \quad (31)$$

To show that it is the feedback interconnection of two passive systems, the trick is to consider the forward system from $\lambda(t) - \lambda^*$ to $\dot{\mathbf{y}}(t)$, and the backward system from $\dot{\mathbf{y}}(t)$ to $\lambda(t) - \lambda^*$. From $\lambda(t) - \lambda^*$ to $\dot{\mathbf{y}}(t)$, the storage function is

$$V_1(\mathbf{x}) = \sum_s x_s q_s^* - U_s(x_s).$$

Then $V_1(x)$ is a positive definite function since its Hessian is a positive definite matrix for all \mathbf{x} . Moreover, it can be shown, using $\mathbf{q}(t) = \mathbf{R}^T \lambda(t)$, that

$$\dot{V}_1(\mathbf{x}) = - \sum_s \kappa_s (q_s(t) - U'_s(x_s(t)))^2 + (\lambda(t) - \lambda^*)^T \dot{\mathbf{y}}$$

and hence the forward system from $\lambda(t) - \lambda^*$ to $\dot{\mathbf{y}}(t)$ is passive. For the reverse system, consider the storage function

$$V_2(\mathbf{y} - \mathbf{y}^*) = \sum_l \int_{y_l}^{y_l^*} g_l(z) - g_l(z^*) dz.$$

V_2 is positive semidefinite function since its Hessian is a positive semidefinite matrix. Moreover

$$\dot{V}_2 = (\lambda(t) - \lambda^*)^T \dot{\mathbf{y}},$$

and hence the reverse system is passive. Then $V(\mathbf{x}) := V_1(\mathbf{x}) + V_2(\mathbf{x})$ can be used as a Lyapunov function for the feedback system, because

$$\dot{V} = - \sum_s \kappa_s (q_s(t) - U'_s(x_s(t)))^2 < 0 \quad \text{except for } \mathbf{x}(t) \equiv \mathbf{x}^*.$$

This implies global asymptotic stability.

The same argument proves the global asymptotic stability of the dual algorithm ((23),24) [110]. Moreover, since primal source algorithm from $\lambda - \lambda^*$ to $\mathbf{y} - \mathbf{y}^*$ and the dual link algorithm from $\mathbf{y} - \mathbf{y}^*$ to $\lambda - \lambda^*$ are both

passive, the passivity theorem asserts the global asymptotic stability of their feedback interconnection, *i.e.*, that of the following primal-dual algorithm:

$$\begin{aligned}\dot{x}_s &= \kappa_s (U'_s(x_s(t)) - q_s(t))_{x_s}^+ \\ \dot{\lambda}_l &= \gamma_l (y_l(t) - c_l)_{\lambda_l}^+\end{aligned}$$

where $(h)_z^+ = 0$ if $z = 0$ and $h < 0$, and $= h$ otherwise. The global asymptotic stability of the AVQ algorithm (25)–(26) is similarly proved in [110].

Method 5. *Proving stability by passivity argument.*

2.1.4 Heterogeneous congestion control protocols

A key assumption in the current model (5)–(7) is that the link prices $\lambda_l(t)$ depend only on links but not sources, *i.e.*, the sources are homogeneous in that, even though they may control their rates using different algorithms F_s , they all adapt to the same type of congestion signals, *e.g.*, all react to loss probabilities, as in TCP Reno, or all to queueing delay, as in TCP Vegas or FAST. When sources with *heterogeneous* protocols that react to different congestion signals share the same network, the current convex optimization and duality framework is no longer applicable. This is modeled in [101, 99] by introducing price mapping functions m_l^s that maps link prices λ_l to “effective prices” seen by sources s . However, one can no longer interpret congestion control as a distributed solution of the basic NUM when there are heterogeneous protocols. In this subsection, we summarize the main results of [101] on the equilibrium structure of heterogeneous protocols. Dynamic properties have also recently been characterized.

Suppose there are J different protocols indexed by superscript j , and N^j sources using protocol j , indexed by (j, s) where $j = 1, \dots, J$ and $s = 1, \dots, N^j$. The total number of sources is $N := \sum_j N^j$. The $L \times N^j$ routing matrix R^j for type j sources is defined by $R_{l's}^j = 1$ if source (j, s) uses link l , and 0 otherwise. The overall routing matrix is denoted by

$$\mathbf{R} = [\mathbf{R}^1 \quad \mathbf{R}^2 \quad \dots \quad \mathbf{R}^J].$$

Every link l has an “intrinsic price” λ_l . A type j source reacts to the “effective price” $m_l^j(\lambda_l)$ in its path, where m_l^j is a price mapping function, which can depend on both the link and the protocol type. By specifying functions m_l^j , we can let the link feed back different congestion signals to sources using different protocols, for example, Reno with packet losses and Vegas with queueing delay. Let $\mathbf{m}^j(\boldsymbol{\lambda}) = (m_l^j(\lambda_l), l = 1, \dots, L)$ and $\mathbf{m}(\boldsymbol{\lambda}) = (\mathbf{m}^j(\boldsymbol{\lambda}), j = 1, \dots, J)$.

The aggregate prices for source (j, s) is defined as

$$q_s^j = \sum_l R_{l's}^j m_l^j(\lambda_l). \quad (32)$$

Let $\mathbf{q}^j = (q_s^j, s = 1, \dots, N^j)$ and $\mathbf{q} = (\mathbf{q}^j, j = 1, \dots, J)$ be vectors of aggregate prices. Then $\mathbf{q}^j = (\mathbf{R}^j)^T \mathbf{m}^j(\boldsymbol{\lambda})$ and $\mathbf{q} = \mathbf{R}^T \mathbf{m}(\boldsymbol{\lambda})$. Let \mathbf{x}^j be a vector with the rate x_s^j of source (j, s) as its s th entry, and \mathbf{x} be the vector of \mathbf{x}^j :

$$\mathbf{x} = [(\mathbf{x}^1)^T, (\mathbf{x}^2)^T, \dots, (\mathbf{x}^J)^T]^T.$$

Source (j, s) has a utility function $U_s^j(x_s^j)$ that is strictly concave increasing in its rate x_s^j . Let $\mathbf{U} = (U_s^j, s = 1, \dots, N^j, j = 1, \dots, J)$. We call $(\mathbf{c}, \mathbf{m}, \mathbf{R}, \mathbf{U})$ a *network* with heterogeneous congestion control protocols.

A network is in equilibrium, or the link prices $\boldsymbol{\lambda}$ and source rates \mathbf{x} are in equilibrium, when each source (j, s) maximizes its net benefit (utility minus bandwidth cost), and the demand for and supply of bandwidth at each bottleneck link are balanced. Formally, a network equilibrium is defined as follows.

Given any prices $\boldsymbol{\lambda}$, we assume that the source rates x_s^j are uniquely determined by

$$x_s^j(q_s^j) = \left[(U_s^j)'^{-1}(q_s^j) \right]^+.$$

This implies that the source rates x_s^j uniquely solve $\max_{z \geq 0} [U_s^j(z) - zq_s^j]$. As usual, we use $\mathbf{x}^j(\mathbf{q}^j) = (x_s^j(q_s^j), s = 1, \dots, N^j)$ and $\mathbf{x}(\mathbf{q}) = (\mathbf{x}^j(\mathbf{q}^j), j = 1, \dots, J)$ to denote the vector-valued functions composed of x_s^j . Since $\mathbf{q} = \mathbf{R}^T \mathbf{m}(\boldsymbol{\lambda})$, we often abuse notation and write $x_s^j(\boldsymbol{\lambda}), \mathbf{x}^j(\boldsymbol{\lambda}), \mathbf{x}(\boldsymbol{\lambda})$. Define the aggregate source rates $\mathbf{y}(\boldsymbol{\lambda}) = (y_l(\boldsymbol{\lambda}), l = 1, \dots, L)$ at links l by:

$$\mathbf{y}^j(\boldsymbol{\lambda}) = \mathbf{R}^j \mathbf{x}^j(\boldsymbol{\lambda}), \quad \mathbf{y}(\boldsymbol{\lambda}) = \mathbf{R} \mathbf{x}(\boldsymbol{\lambda}). \quad (33)$$

In equilibrium, the aggregate rate at each link is no more than the link capacity, and they are equal if the link price is strictly positive. Formally, we call $\boldsymbol{\lambda}$ an *equilibrium price*, a *network equilibrium*, or just an *equilibrium* if it satisfies (from (32)–(33))

$$\text{diag}(\boldsymbol{\lambda})(\mathbf{y}(\boldsymbol{\lambda}) - \mathbf{c}) = 0, \quad \mathbf{y}(\boldsymbol{\lambda}) \leq \mathbf{c}, \quad \boldsymbol{\lambda} \geq 0. \quad (34)$$

The current theory corresponds to $J = 1$. When there are $J > 1$ types of prices, the current duality theory breaks down because there cannot be more than one Lagrange multiplier at each link. In general, an equilibrium no longer maximizes aggregate utility, nor is it unique. It is proved in [101] that, under mild assumptions, an equilibrium always exists. There can be networks $(\mathbf{R}, \mathbf{c}, \mathbf{m}, \mathbf{U})$ that have uncountably many equilibria, but except for a set of measure zero, all networks have finitely many equilibria. Moreover, Poincare-Hopf index theorem implies that the number of equilibria is necessarily odd. Specifically, suppose the following assumptions hold:

C4: Price mapping functions m_l^j are continuously differentiable in their domains and strictly increasing with $m_l^j(0) = 0$.

C5: For any $\epsilon > 0$, there exists a number λ_{\max} such that if $\lambda_l > \lambda_{\max}$ for link l , then

$$x_i^j(\boldsymbol{\lambda}) < \epsilon \text{ for all } (j, i) \text{ with } \mathbf{R}_{li}^j = 1.$$

C6: Every link l has a single-link flow (j, i) with $(U_i^j)'(c_l) > 0$.

Assumption C6 can be relaxed; see [90]. We call an equilibrium λ^* *locally unique* if $\partial \mathbf{y} / \partial \lambda \neq 0$ at λ^* . We call a network $(\mathbf{c}, \mathbf{m}, \mathbf{R}, \mathbf{U})$ *regular* if all equilibrium points are locally unique.

Theorem 2. 1. *There exists an equilibrium price λ^* for any network $(\mathbf{c}, \mathbf{m}, \mathbf{R}, \mathbf{U})$.*

2. *Moreover, the set of link capacities \mathbf{c} for which not all equilibrium points are locally unique (i.e., the network is not regular) has Lebesgue measure zero in \mathbf{R}_+^L .*

3. *A regular network has finite and odd number of equilibrium points.*

Method 6. *Proving equilibrium properties through vector field representation and Poincare-Hopf Index Theorem.*

Despite the lack of an underlying NUM, heterogeneous protocols are still Pareto efficient for general networks. Moreover, the loss of optimality can be bounded in terms of the slope of the price mapping functions m_l^j . Specifically, suppose we use the objective value V^* of the following NUM as a measure of optimality for heterogeneous protocols

$$\begin{aligned} V^* &:= \text{maximize} && \sum_j \sum_s U_s^j(x_s^j) \\ &\text{subject to} && \mathbf{R}\mathbf{x} \leq \mathbf{c} \end{aligned} \quad (35)$$

Let $V(\lambda^*) := \sum_j \sum_s U_s^j(x_s^j(\lambda^*))$ be the utility achieved by any equilibrium λ^* of the heterogeneous protocol. Then it can be shown that, for any equilibrium λ^* ,

$$\frac{V(\lambda^*)}{V^*} \geq \frac{\min \dot{m}_l^j(\lambda)}{\max \dot{m}_l^j(\lambda)}$$

where \dot{m}_l^j denotes the derivative of m_l^j , and the minimization and maximization are over all types j , all links l used by all type j flows, and all prices λ . For common AQM schemes such as RED with (piecewise) linear m_l^j , the bound reduces to a simple expression in terms of their slopes.

For a homogeneous congestion control protocol, the utility functions determine how bandwidth is shared among all the flows. For heterogeneous protocols, how is bandwidth shared among these protocols (inter-protocol fairness), and how is it shared among flows within each protocol (intra-protocol fairness)? It is shown in [99] (and a generalization of results there) that any desired degree of fairness among the different protocols is achievable in general networks by appropriate linear scaling of utility functions. Within each protocol, the flows would share the bandwidth among themselves as if they were in a single-protocol network according to their own utility functions, except that the link capacities are reduced by the amount consumed by the other protocols. In other words, intra-protocol fairness is unaffected by the presence of other protocols.

Theorem 2 guarantees local unique equilibrium points for almost all networks under mild conditions. If the *degree of heterogeneity*, as measured by the slopes \dot{m}_l^j of the price mapping functions m_l^j is small, then global

uniqueness is guaranteed: if \dot{m}_l^j do not differ much across source types at each link, or they do not differ much along links in every source's path, the equilibrium is globally unique. Moreover, under this condition, global uniqueness is equivalent to local stability. Specifically, consider the dual algorithm (in continuous-time):

$$\begin{aligned}\dot{\lambda}_l &= \gamma(y_l(t) - c_l) \\ x_s^j(t) &= U_s'^{-1}(q_s^j(t))\end{aligned}$$

where the effective prices $q_s^j(t)$ is defined by (32) (compare with (23,24) in the homogeneous case). The linearized system with a small perturbation $\delta\lambda$ around an equilibrium point λ^* is, in vector form,

$$\delta\dot{\lambda} = \gamma \frac{\partial y}{\partial \lambda}(\lambda^*) \delta\lambda. \quad (36)$$

The equilibrium λ^* is called *locally stable* if all the eigenvalues of $\partial y / \partial \lambda(\lambda^*)$ are in the left-half plane. Given the price mapping functions m_l^j , we say their degree of heterogeneity is small if they satisfy any one of the following conditions:

1. For each $l = 1, \dots, L, j = 1, \dots, J$

$$\dot{m}_l^j(\lambda^*) \in \left[a_l, 2^{\frac{1}{L}} a_l \right] \quad \text{for some } a_l > 0 \text{ for any equilibrium } \lambda^*. \quad (37)$$

2. For all $j = 1, \dots, J, l = 1, \dots, L$

$$\dot{m}_l^j \in \left[a^j, 2^{\frac{1}{L}} a^j \right] \quad \text{for some } a^j > 0 \text{ for any equilibrium } \lambda^*. \quad (38)$$

Theorem 3. *For almost all networks $(\mathbf{c}, \mathbf{m}, \mathbf{R}, \mathbf{U})$,*

1. *Suppose their degree of heterogeneity is small, then the equilibrium is globally unique. Moreover, it is locally stable.*
2. *Conversely, if all equilibrium points are locally stable, it is also globally unique.*

Asymptotically when $L \rightarrow \infty$, both conditions (37) and (38) converge to a single point. Condition (37) reduces to $\dot{m}_l^j = a_l$ which essentially says that all protocols are the same ($J = 1$). Condition (38) reduces to $\dot{m}_l^j = a^j$, which is the case where price mapping functions m_l^j are linear and link independent. Various special cases are shown to have a globally unique equilibrium in [101].

First recall that since a network of homogeneous protocols solves the basic NUM, it always has a unique equilibrium point as long as the routing matrix R has full row rank. The equilibrium source rates \mathbf{x}^* does not depend on link parameters, such as buffer size, as long as the AQM guarantees complementary slackness condition for the basic NUM. Moreover, \mathbf{x}^* does not depend on the flow arrival pattern. These properties no longer hold in the heterogeneous case. We now present a simulation using ns2 (Network Simulator 2) that shows that \mathbf{x}^* can depend on the flow arrival pattern because of the existence of multiple equilibria.

The topology of this network is shown in Figure 1. All links run the RED algorithm. Links 1 and 3 are each configured with 9.1pkts per ms capacity (equivalent to 111 Mbps), 30 ms one-way propagation delay and a buffer of 1500 packets. RED parameter is set to be $(b, \bar{b}, \rho_1) = (300, 1500, 10^{-4})$. Link 2 has a capacity of 13.8 pkts per ms (166 Mbps) with 30 ms one-way propagation delay and buffer size of 1500 packets. RED parameter is set to $(0, 1500, 0.1)$. There are 8 Reno flows on path 3 utilizing all the three links, with one-way propagation delay of 90 ms. There are two FAST flows on each of paths 1 and 2. Both of them have one-way propagation delay of 60 ms. All FAST flows use a common parameter value $\alpha = 50$ packets. Two sets of simulations have been carried

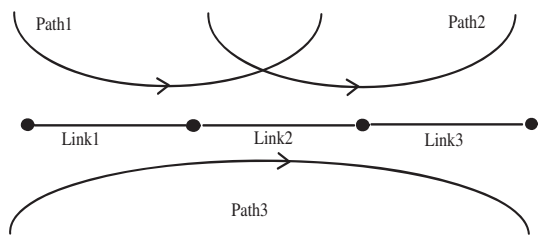


Figure 1: Multiple equilibria scenario.

out with different starting times for Reno and FAST flows. One set of flows (Reno or FAST) starts at time zero, and the other set starts at the 100-th seconds. Figure 2 shows the sample throughput trajectories of one of FAST flows and one of Reno flows. The large difference in the rate allocations of FAST and Reno between these two

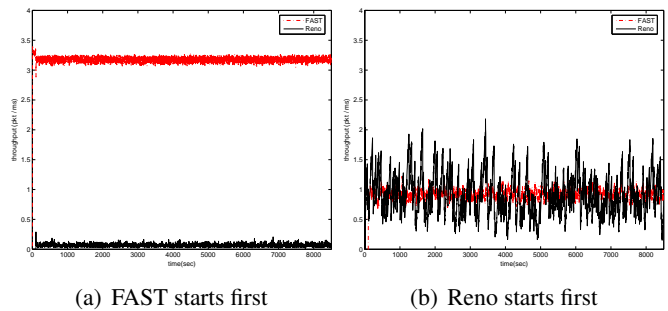


Figure 2: Sample throughput trajectories of FAST and Reno.

scenarios results from that the network reaches two different equilibrium points depending on which type of flows starts first.

The model introduced in [101, 99] is critical in deepening our understanding of such complex behavior, and providing design guidelines to manage it in practice.

2.1.5 Forward Engineering: FAST

The congestion control algorithm in the current TCP, which we refer to as Reno, was developed in 1988 [30] and has gone through several enhancements since. It has performed remarkably well and is generally believed to have prevented severe congestion as the Internet scaled up by six orders of magnitude in size, speed, load, and connectivity. It is also well-known, however, that as bandwidth-delay product continues to grow, TCP Reno will eventually become a performance bottleneck itself. Even though, historically, TCP Reno was designed, implemented, and deployed without any consideration of network utility maximization, and its equilibrium, fairness, and dynamic properties were understood only as an afterthought, it indeed solves a NUM implicitly.

Several new algorithms have been proposed in the last few years to address the problems of Reno, including TCP Westwood, HSTCP [28], FAST TCP [40], STCP [49], BIC TCP [113], HTCP [60], MaxNet [111, 112], XCP [44], and RCP [22], etc. (see [40] for other references). Some of these designs were explicitly guided by the emerging theory surveyed in this paper, which has become indispensable to the systematic design of new congestion control algorithms. It provides a framework to understand issues, clarify ideas and suggest directions, leading to more understandable and better performing implementations. One of the proposals, FAST, has been used to break world records of wide area network data transfer in 2003, 2004 and 2005 [40, 109, 41].

The congestion control mechanism of FAST TCP is separated into four components as shown in Figure 3. These four components are functionally independent so that they can be designed separately and upgraded asynchronously. The *data control* component determines *which* packets to transmit, *window control* determines *how*

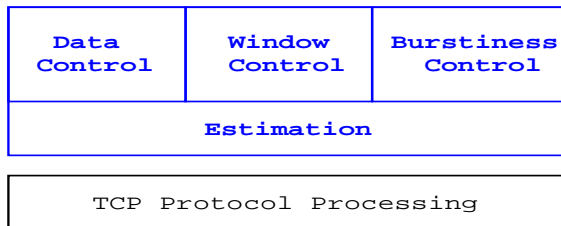


Figure 3: FAST TCP architecture.

many packets to transmit, and *burstiness control* determines *when* to transmit these packets. These decisions are made based on information provided by the *estimation* component. More specifically, the estimation component computes two pieces of feedback information for each data packet sent – a multibit queueing delay and an one-bit loss-or-no-loss indication – which are used by the other three components. Data control selects the next packet to send from three pools of candidates: new packets, packets that are deemed lost (negatively acknowledged), and transmitted packets that are not yet acknowledged. Window control regulates packet transmission at the RTT timescale, while burstiness control smoothes out the transmission of packets at a smaller timescale. The theory surveyed in this paper forms the foundation of the window control algorithm. FAST periodically updates the congestion window based on the average RTT and average queueing delay provided by the estimation component, according to (14) in Section 2.1.1.

The equilibrium values of windows w^* and delays λ^* of the network defined by (14)–(15) are the unique

solutions to the utility maximization problem

$$\begin{aligned} & \max_{\mathbf{x} \geq 0} && \sum_s \alpha_s \log x_s \\ & \text{subject to} && \mathbf{R}\mathbf{x} \leq \mathbf{c} \end{aligned}$$

and its Lagrangian dual problem:

$$\min_{\boldsymbol{\lambda} \geq 0} \quad \sum_l c_l \lambda_l - \sum_s \alpha_s \log \sum_l R_{ls} \lambda_l$$

This implies that the equilibrium rate \mathbf{x}^* is α_s -weighted proportionally fair. In equilibrium, source s maintains α_s packets in the buffers along its path. Hence, the total amount of buffering in the network must be at least $\sum_s \alpha_s$ packets in order to reach the equilibrium. FAST TCP is proved in [107] to be locally asymptotically stable for general networks if all flows have the same feedback delay, no matter how large the delay is. It is proved in [17] to be globally asymptotically stable in the presence of heterogeneous feedback delay at a single link.

We have implemented the insights from this series of theoretical work in a software prototype FAST TCP [40, 109] and have been working with our collaborators to test it in various networks around the world [41]. Physicists have been using FAST TCP to break various world records in data transfer in the last few years. Figure 4 shows its performance in several experiments conducted over 2002–05 over a high speed trans-Atlantic network, over a home DSL (digital subscriber line), and over an emulated lossy link.

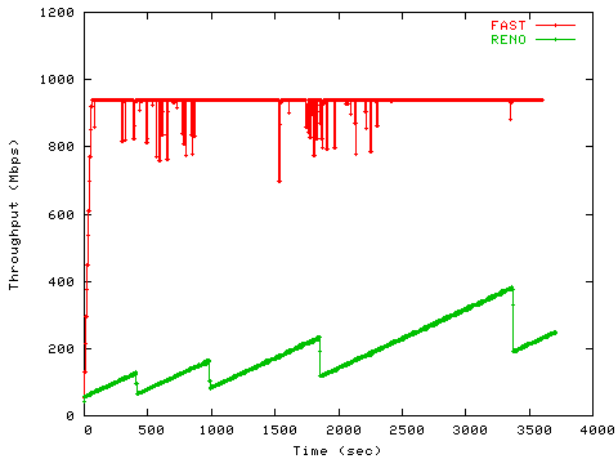
2.2 Medium Access Control

2.2.1 Reverse engineering: MAC as non-cooperative game

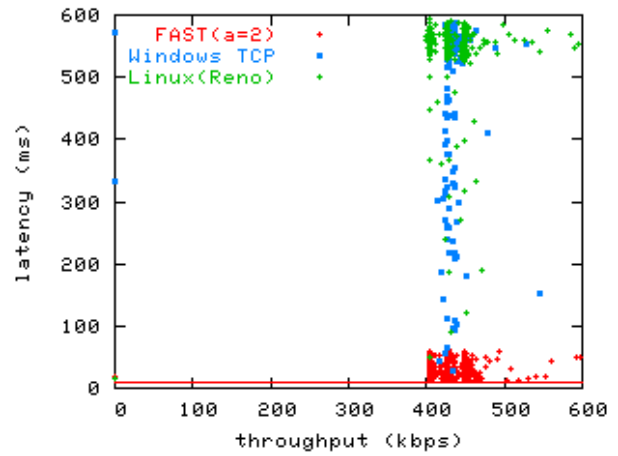
If contentions among transmissions on the same link in wired networks or across different links in wireless networks are not appropriately controlled, a large number of collisions may occur, resulting in waste of resources such as bandwidth and energy, as well as loss of system efficiency and fairness. There are two major types of medium access control (MAC): scheduling-based contention-free mode and random-access-based contention-prone mode. The first is often shown to solve certain maximum weight matching problems. The second has been extensively studied through the perspective of queuing-theoretic performance evaluation, but is only recently reverse engineered to recover the underlying utility maximization structure [55, 97].

In TCP reverse engineering considered in the last subsection, the utility function of each source depends only on its data rate that can be directly controlled by the source itself. TCP/AQM can be modeled as a distributed algorithm that solves the basic NUM problem and its Lagrange dual.

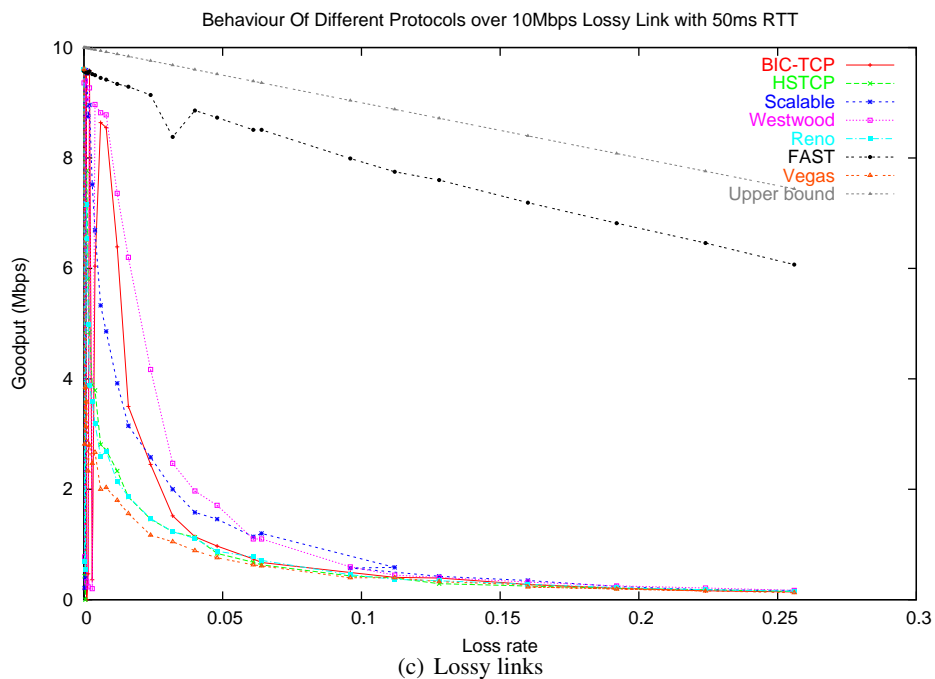
In contrast, in the Exponential-Backoff (EB) MAC protocol, the utility of each link directly depends not only on its own transmission (*e.g.*, persistence probability) but also transmissions of other links due to collisions. We show that the EB protocol can be reverse engineered through a non-cooperative game in which each link tries to maximize, using a stochastic subgradient formed by local information, its own utility function in the form of



(a) High speed WAN



(b) DSL



(c) Lossy links

Figure 4: Performance of FAST TCP. (a) At 1 Gpbs, FAST TCP utilized 95% of a trans-Atlantic network bandwidth while maintaining a fairly constant throughput. Linux TCP on average used 19% of the available bandwidth, while producing a throughput that fluctuates from 100 Mbps to 400 Mbps. (b) At an 512-Kbps DSL uplink, data transfer using FAST TCP increased the latency from 10 ms to around 50 ms, while Linux and Windows TCP increased it to as high as 600 ms, an order of magnitude larger. (c) Over an emulated lossy link, FAST TCP achieved close to optimal data rate while other (loss-based) TCP variants collapsed when loss rate exceeded 5%. Figure from unpublished work by Bartek Wydrowski, Sanjay Hegde and Cheng Jin.

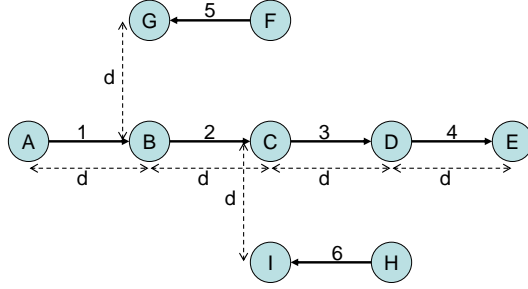


Figure 5: Logical topology graph of a network illustrating contention.

expected net reward for successful transmission. While the existence of Nash equilibrium can be proved, neither convergence nor social welfare optimality is guaranteed. We then provide sufficient conditions on user density and backoff aggressiveness that guarantee uniqueness and stability of Nash equilibrium (*i.e.*, convergence of the standard best response strategy).

Consider an ad hoc network represented by a directed graph $G(V, E)$, *e.g.*, as in Figure 5, where V is the set of nodes and E is the set of logical links. We define $L_{out}(n)$ as a set of outgoing links from node n , $L_{in}(n)$ as a set of incoming links to node n , t_l as the transmitter node of link l , and r_l as the receiver node of link l . We also define $N_{to}^I(l)$ as the set of nodes whose transmission cause interference to the receiver of link l , excluding the transmitter node of link l , (*i.e.*, t_l), and $L_{from}^I(n)$ as the set of links whose transmission get interfered from the transmission of node n , excluding outgoing links from node n (*i.e.*, $l \in L_{out}(n)$). Hence, if the transmitter of link l and a node in set $N_{to}^I(l)$ transmit data simultaneously, the transmission of link l fails. If node n and the transmitter of link l in set $L_{from}^I(n)$ transmit data simultaneously, the transmission of link l also fails.

Random-access protocols in such wireless networks usually consist of two phases: contention avoidance and contention resolution. We focus only on the second phase. The EB protocol is a prototypical contention resolution protocol. For example, in the IEEE 802.11 DCF (Distributed Coordination Function) implementation, the EB protocol is window-based: each link l maintains its contention window size W_l , current window size CW_l , and minimum and maximum window sizes W_l^{min} and W_l^{max} . After each transmission, contention window size and current window size are updated. If transmission is successful, the contention window size is reduced to the minimum window size (*i.e.*, $W_l = W_l^{min}$), otherwise it is doubled until reaching the maximum window size W_l^{max} (*i.e.*, $W_l = \min\{2W_l, W_l^{max}\}$). Then, current window size CW_l is chosen to be a number between $(0, W_l)$ uniformly at random. It decreases in every time-slot, and when it becomes zero, the link transmits data. Since the window size is doubled after each transmission failure, the random access protocol in DCF is called the Binary Exponential Backoff (BEB) protocol, which is a special case of EB protocols.

We study the window-based EB MAC protocol through a persistence probabilistic model, an approach analogous to the source rate model for the window-based TCP congestion control protocol in Subsection 2.1.2. Here each link l transmits data with a probability p_l , which we refer to as the persistence probability of link l . After each transmission attempt, if the transmission is successful without collisions, then link l sets its persistence probability

to be its maximum value, p_l^{max} . Otherwise, it multiplicatively reduces its persistence probability by a factor β_l ($0 < \beta_l < 1$) until reaching its minimum value p_l^{min} . This persistence probability model is a memoryless one that approximates the average behavior of EB protocol.

Since in the window-based BEB protocol the current window size CW_l of link l is randomly selected between $(0, W_l)$, when its window size is W_l , we may think that link l transmits data in a time-slot with an attempt probability $1/W_l$, which corresponds to the persistence probability p_l in our model for the average behavior of the EB protocols. In the window-based protocol, after every transmission success, the attempt probability is set to be its maximum value (*i.e.*, $1/W_l^{min}$), which corresponds to p_l^{max} in our model, and after every transmission failure, the attempt probability is set to be a fraction of its current value until it reaches its minimum value, which corresponds to reducing the persistence probability by a factor of $\beta = 0.5$ in BEB (and in general $\beta \in (0, 1)$ in EB) until reaching the minimum persistence probability p_l^{min} .

The update algorithm for the persistence probability described above can be written as:

$$p_l(t+1) = \max\{p_l^{min}, p_l^{max} \mathbf{1}_{\{T_l(t)=1\}} \mathbf{1}_{\{C_l(t)=0\}} + \beta_l p_l(t) \mathbf{1}_{\{T_l(t)=1\}} \mathbf{1}_{\{C_l(t)=1\}} + p_l(t) \mathbf{1}_{\{T_l(t)=0\}}\} \quad (39)$$

where $p_l(t)$ is a persistence probability of link l at time-slot t , $\mathbf{1}_a$ is an indicator function of event a , and $T_l(t)$ and $C_l(t)$ are the events that link l transmits data at time-slot t and that there is a collision to link l 's transmission given that link l transmits data at time-slot t , respectively. In the rest of this subsection, we will examine the case when $p_l^{min} = 0$. Given $\mathbf{p}(t)$, we have

$$\text{Prob}\{T_l(t) = 1 | \mathbf{p}(t)\} = p_l(t)$$

and

$$\text{Prob}\{C_l(t) = 1 | \mathbf{p}(t)\} = 1 - \prod_{n \in L_{to}^l(t)} (1 - p_n(t)).$$

Since the update of the persistence probabilities for the next time-slot depends only on the current persistence probabilities, we will consider the update conditioning on the current persistence probabilities. Note that $p_l(t)$ is a random process whose transitions depend on events $T_l(t)$ and $C_l(t)$. We first study its expected trajectory and will return to (39) later in this subsection. Slightly abusing the notation, we still use $p_l(t)$ to denote the expected persistence probability. From (39), we have

$$\begin{aligned} p_l(t+1) &= p_l^{max} \mathbb{E}\{\mathbf{1}_{\{T_l(t)=1\}} \mathbf{1}_{\{C_l(t)=0\}} | \mathbf{p}(t)\} + \beta_l \mathbb{E}\{p_l(t) \mathbf{1}_{\{T_l(t)=1\}} \mathbf{1}_{\{C_l(t)=1\}} | \mathbf{p}(t)\} + \mathbb{E}\{p_l(t) \mathbf{1}_{\{T_l(t)=0\}} | \mathbf{p}(t)\}\} \\ &= p_l^{max} p_l(t) \prod_{n \in L_{to}^l(t)} (1 - p_n(t)) + \beta_l p_l(t) p_l(t) \left(1 - \prod_{n \in L_{to}^l(t)} (1 - p_n(t))\right) + p_l(t)(1 - p_l(t)), \end{aligned} \quad (40)$$

where $\mathbb{E}\{a|b\}$ is the expected value of a given b and $\mathbf{1}$ denotes the indicator function of probabilistic events.

We now reverse engineer the update algorithm in (40) as a game, in which each link l updates its strategy, *i.e.*, its persistence probability p_l , to maximize its utility U_l based on strategies of the other links, *i.e.*, $\mathbf{p}_{-l} = (p_1, \dots, p_{l-1}, p_{l+1}, \dots, p_{|E|})$. Formally, the game is $G_{EB-MAC} = [E, \times_{l \in E} A_l, \{U_l\}_{l \in E}]$, where E is a set of players, *i.e.*, links, $A_l = \{p_l \mid 0 \leq p_l \leq p_l^{max}\}$ is an action set of player l , and U_l is a utility function of player l to be determined through reverse engineering.

Theorem 4. *The utility function is the following expected net reward (expected reward minus expected cost) that the link can obtain from its transmission:*

$$U_l(\mathbf{p}) = R(p_l)S(\mathbf{p}) - C(p_l)F(\mathbf{p}), \quad \forall l \quad (41)$$

where $S(\mathbf{p}) = p_l \prod_{n \in L_{to}^I(l)} (1 - p_n)$ is the probability of transmission success, $F(\mathbf{p}) = p_l (1 - \prod_{n \in L_{to}^I(l)} (1 - p_n))$ is the probability of transmission failure, and $R(p_l) \stackrel{\text{def}}{=} p_l (\frac{1}{2} p_l^{max} - \frac{1}{3} p_l)$ can be interpreted as the reward for transmission success, $C(p_l) \stackrel{\text{def}}{=} \frac{1}{3} (1 - \beta_l) p_l^2$ can be interpreted as the cost for transmission failure.

Furthermore, there exists a Nash equilibrium in the EB-MAC Game $G_{EB-MAC} = [E, \times_{l \in E} A_l, \{U_l\}_{l \in E}]$ characterized by the following:

$$p_l^* = \frac{p_l^{max} \prod_{n \in L_{to}^I(l)} (1 - p_n^*)}{1 - \beta_l (1 - \prod_{n \in L_{to}^I(l)} (1 - p_n^*))}, \quad \forall l. \quad (42)$$

Note that the expressions of $S(\mathbf{p})$ and $F(\mathbf{p})$ come directly from the definitions of success and failure probabilities, while the expressions of $R(p_l)$ and $C(p_l)$ (thus exact form of U_l) are in fact *derived* in the proof by reverse engineering the EB protocol description.

In the EB protocol, there is no explicit message passing among links, and the link cannot obtain the exact information to evaluate the gradient of its utility function. Instead of using the exact gradient of its utility function as in (40), each link attempts to approximate it using (39). It can be shown [56, 97] that the EB protocol described by (39) is a stochastic subgradient algorithm to maximize utility (41).

Method 7. *Reverse engineer a non-cooperative protocol as a game.*

The next step is to investigate uniqueness of Nash equilibrium together with the convergence of a natural strategy for the game: the best response strategy, commonly used to study stability of Nash equilibrium. In best response, each link updates its persistence probability for the next time-slot such that it maximizes its utility based on the persistence probabilities of the other links in the current time-slot:

$$p_l^*(t+1) = \underset{0 \leq p_l \leq p_l^{max}}{\operatorname{argmax}} U_l(p_l, \mathbf{p}_{-l}^*(t)). \quad (43)$$

Hence, $p_l^*(t+1)$ is the best response of link l given $\mathbf{p}_{-l}^*(t)$. The connection between the best response strategy and stochastic subgradient update strategy has been quantified for EB MAC Game [97].

Let $K = \max_l \{|L_{to}^I(l)|\}$, which captures the amount of potential contention among links. We have the following theorem that relates three key quantities: amount of potential contention K , backoff multiplier β (speed of backoff), and p^{max} that corresponds to the minimum contention window size (minimum amount of backoff).

Theorem 5. *If $\frac{p^{max} K}{4\beta(1-p^{max})} < 1$, then*

1. *The Nash equilibrium is unique;*

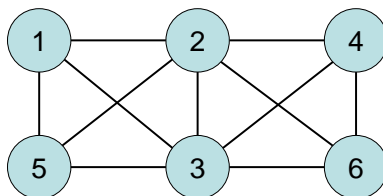


Figure 6: A contention graph.

2. Starting from any initial point, the iteration defined by best response converges to the unique equilibrium.

There are several interesting engineering implications from the above theorem. For example, it provides guidance on choosing parameter in the EB protocols, and quantifies the intuition that with a large enough β (*i.e.*, links do not decrease the probabilities suddenly) and a small enough p^{max} (*i.e.*, links backoff aggressively enough), uniqueness and stability can be ensured. The higher the amount of contention (*i.e.*, a larger value of K), the smaller p^{max} needs to be. The key idea in the proof is to show the updating rule from $p(t)$ to $p(t + 1)$ is a contraction mapping by verifying the infinity norm of the Jacobian \mathbf{J} of the update dynamics in the game is less than one.

Method 8. *Verifying contraction mapping by bounding the Jacobian’s norm.*

As will be discussed in Section 4, session level stochastic effects need to be incorporated in the above reverse engineering model to include the arrival statistics of finite-duration sessions. Then MAC protocols can be analyzed and designed through a union of stochastic stability results in traditional queuing models and optimality results in the utility maximization models.

2.2.2 Forward engineering: Utility-optimal MAC protocol

Nash equilibrium attained by existing EB MAC protocols may not be socially optimal. This motivates forward engineering where adequate feedback is generated to align selfish utility maximization by each logical link to maximize the social welfare in terms of total network utility. By imposing different utility functions, different types of services and different efficiency-fairness tradeoffs can be provisioned. Two suites of protocols are possible: scheduling-based and random-access-based. We again focus on the second in this subsection on forward engineering.

Contentions among links can be modeled by using a contention graph first proposed in [72]. An example is shown in Figure 6, which is obtained from Figure 5 assuming that if the distance between the receiver of one link and the transmitter of the other link is less than $2d$, there is interference between those two links. Each vertex in the contention graph corresponds to a link in the network topology graph. If two links’ transmissions interfere with each other, the vertices corresponding to them in the contention graph are connected with an edge. Only one link

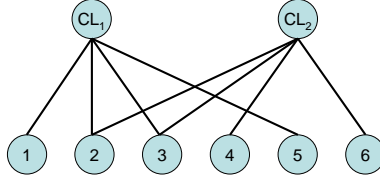


Figure 7: A bipartite graph between maximal cliques and links in the contention graph.

at a time among links in the same maximal clique in the contention graph can transmit data without collision. This constraint can be visualized by using a bipartite graph, as in Figure 7, where one partition of vertices corresponds to links in the network (*i.e.*, nodes in the contention graph) and the other corresponds to maximal cliques in the contention graph. An edge is established in the bipartite graph if a node in the contention graph belongs to a maximal clique. Hence, only network links represented by the nodes in the bipartite graph that are covered by a matching can transmit data simultaneously without collisions.

In [72, 26, 10], a fluid approximation approach is used where each maximum clique is defined as a resource with a finite capacity that is shared by the links belonging to the clique. Capacity of a clique is defined as the maximum value of the sum of time fractions such that each link in the clique can transmit data without collision. Consequently, a generalized NUM problem has been formulated as follows, with capacity constraint C_{CL_i} at each maximal clique CL_i :

$$\begin{aligned} & \text{maximize} && \sum_l U_l(x_l) \\ & \text{subject to} && \sum_{l \in L(CL_i)} \frac{x_l}{c_l} \leq C_{CL_i} \quad \forall i. \end{aligned} \quad (44)$$

This problem formulation essentially takes the same structure as the basic NUM (3) for TCP congestion control, and can be solved following the same dual-decomposition algorithm.

An alternative approach is to explicitly model collision probabilities, as shown in [42] for log utility and in [56] for general concave utility. Consider a random-access-based MAC protocol in which each node n adjusts its own persistence probability and also the persistence probability of each of its outgoing links. Since persistent transmission decisions are made distributively at each node, we need a shift from graph models based on logical communication links to graph models that incorporate nodes as well. Let P^n be the transmission probability of node n , and p_l be that of link l . The appropriate generalized NUM thus formulated is as follows, with variables $\{P^n\}, \{p_l\}$:

$$\begin{aligned} & \text{maximize} && \sum_l U_l(x_l) \\ & \text{subject to} && x_l = c_l p_l \prod_{k \in N_{to}^I(l)} (1 - P^k), \quad \forall l \\ & && \sum_{l \in L_{out}(n)} p_l = P^n, \quad \forall n \\ & && 0 \leq P^n \leq 1, \quad \forall n \\ & && 0 \leq p_l \leq 1, \quad \forall l. \end{aligned} \quad (45)$$

Without loss of generality, we can replace the equality in the first constraint with an inequality. This is because such an inequality will always be achieved with an equality at optimality. The next step of problem transformation

is to take the log of both sides of the first constraint in problem (45) and a log change of variables and constants: $x'_l = \log x_l$, $U'_l(x'_l) = U_l(e^{x'_l})$, and $c'_l = \log c_l$. This reformulation turns the problem into:

$$\begin{aligned}
& \text{maximize} && \sum_{l \in L} U'_l(x'_l) \\
& \text{subject to} && c'_l + \log p_l + \sum_{k \in N_{to}^I(l)} \log(1 - P^k) - x'_l \geq 0, \quad \forall l \\
& && \sum_{l \in L_{out}(n)} p_l = P^n, \quad \forall n \\
& && 0 \leq P^n \leq 1, \quad \forall n \\
& && 0 \leq p_l \leq 1, \quad \forall l.
\end{aligned} \tag{46}$$

Note that problem (46) is now separable but still may not be a convex optimization problem, since the objective $U'_l(x'_l)$ may not be a strictly concave function, even though $U_l(x_l)$ is a strictly concave function. However, the simple sufficient condition guarantees its concavity:

$$\frac{\partial^2 U_l(x_l)}{\partial x_l^2} < -\frac{\partial U_l(x_l)}{x_l \partial x_l},$$

which states that the curvature (degree of concavity) of the utility function needs to be not just non-positive but bounded away from 0 by as much as $-\frac{\partial U_l(x_l)}{x_l \partial x_l}$, i.e., the application represented by this utility function must be elastic enough.

Method 9. *Log change of variables for decoupling, and computing minimum curvature needed for concavity after change of variable.*

Following dual decomposition and subgradient method, the NUM problem (45) for random access MAC protocol design can be solved by the following algorithm.

Algorithm 1. Utility Optimal Random Access Algorithm

Each node n constructs its local interference graph to obtain sets $L_{out}(n)$, $L_{in}(n)$, $L_{from}^I(n)$, and $N_{to}^I(l)$, $\forall l \in L_{out}(n)$.

Each node n sets $t = 0$, $\lambda_l(1) = 1, \forall l \in L_{out}(n)$, $P^n(1) = \frac{|L_{out}(n)|}{|L_{out}(n)| + |L_{from}^I(n)|}$, and $p_l(1) = \frac{1}{|L_{out}(n)| + |L_{from}^I(n)|}$, $\forall l \in L_{out}(n)$.

For each node n , do

- 1: Set $t \leftarrow t + 1$.
- 2: Inform $\lambda_l(t)$ to all nodes in $N_{to}^I(l)$, $\forall l \in L_{out}(n)$ and $P^n(t)$ to t_l , $\forall l \in L_{from}^I(n)$.
- 3: Set $k_n(t) = \sum_{l \in L_{out}(n)} \lambda_l(t) + \sum_{k \in L_{from}^I(n)} \lambda_k(t)$ and $\beta(t) = \frac{1}{t}$.

4: Solve the following problems to obtain $P^n(t+1)$, and $x'_l(t+1)$, $p_l(t+1)$, and $\lambda_l(t+1)$, $\forall l \in L_{out}(n)$:

$$\begin{aligned}
P^n(t+1) &= \begin{cases} \frac{\sum_{l \in L_{out}(n)} \lambda_l(t)}{\sum_{l \in L_{out}(n)} \lambda_l(t) + \sum_{k \in L_{from}^I(n)} \lambda_k(t)}, & \text{if } k_n(t) \neq 0 \\ \frac{|L_{out}(n)|}{|L_{out}(n)| + |L_{from}^I(n)|}, & \text{if } k_n(t) = 0 \end{cases}, \\
p_l(t+1) &= \begin{cases} \frac{\lambda_l(t)}{\sum_{l \in L_{out}(n)} \lambda_l(t) + \sum_{k \in L_{from}^I(n)} \lambda_k(t)}, & \text{if } k_n(t) \neq 0 \\ \frac{1}{|L_{out}(n)| + |L_{from}^I(n)|}, & \text{if } k_n(t) = 0 \end{cases}, \\
x'_l(t+1) &= \operatorname{argmax}_{x'_l \in [x'_l{}^{min}, x'_l{}^{max}]} \{U'_l(x'_l) - \lambda_l(t)x'_l\}, \\
&\text{and} \\
\lambda_l(t+1) &= \left[\lambda_l(t) - \beta(t) \left(c'_l + \log p_l(t) + \sum_{k \in N_{to}^I(l)} \log(1 - P^k(t)) - x'_l(t) \right) \right].
\end{aligned}$$

5: Set its persistence probability $P^{n*} = P^n(t)$ and the conditional persistence probability of each of its outgoing links $q_l^* = p_l(t)/P^n(t)$.

6: Decide if it will transmit data with a probability P^{n*} , in which case it chooses to transmit on one of its outgoing links with a probability q_l^* , $\forall l \in L_{out}(n)$.

while (1)

Note that the above algorithm is conducted at each node n to calculate P^n , and p_l , λ_l , and x'_l for its outgoing link l (i.e., $\forall l \in L_{out}(n)$). Hence, the above algorithm is conducted at the transmitter node of each link. If we assume that two nodes within interference range can communicate with each other (i.e., if nodes within distance $2d$ in Figure 5 can establish a communication link), in the above algorithm each node requires information from nodes within two-hop distance from it. To calculate P^n and p_l for its outgoing link l (i.e., $\forall l \in L_{out}(n)$), node n needs λ_m from the transmitter node t_m of link m that is interfered from the transmission of node n (i.e., from t_m , $\forall m \in L_{from}^I(n)$). Note that t_m is within two-hop from node n .

Alternatively, if λ_l and x'_l for each link l are calculated at its receiver node r_l instead of its transmitter node t_l , a modified version of Algorithm 1 can be devised in which each node requires information only within *one-hop* distance [56].

Theorem 6. Algorithm 1 converges to a globally optimal solution of (45) for sufficiently concave utility functions.

We now show a numerical example of the desired tradeoff between efficiency and fairness that can be achieved by appropriately adjusting the parameters of utility functions. In this experiment, the utility function for each link

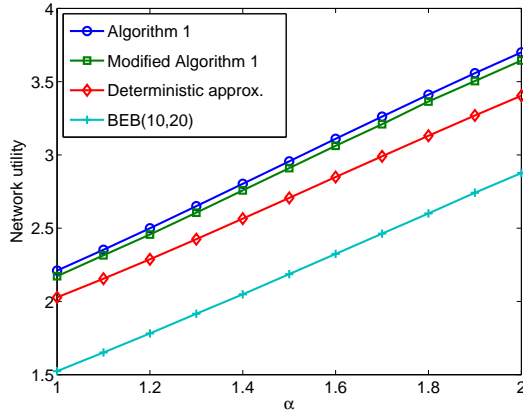


Figure 8: Comparison of network utilities.

l , $U_l(x_l)$ is in the following standard form of concave utility parameterized by α , shifted such that $U_l(x_l^{min}) = 0$ and $U_l(x_l^{max}) = 1$:

$$U_l(x_l) = \frac{x_l^{(1-\alpha)} - x_l^{min(1-\alpha)}}{x_l^{max(1-\alpha)} - x_l^{min(1-\alpha)}}.$$

We set $x_l^{min} = 0.5$ and $x_l^{max} = 5$, $\forall l$, varying the value of α from 1 to 2 with a step size 0.1.

We compare the performances of Algorithm 1 and its one-hop message passing variant (modified Algorithm 1, not shown here) with deterministic fluid approximation and the BEB protocol in IEEE 802.11 standard.⁴

In Figure 8, we compare the network utility achieved by each protocol. We show the tradeoff curve of rate and fairness for each protocol in Figure 9. Here, the fairness index is obtained by $f(\mathbf{x}) = \frac{(\sum_l x_l)^2}{|L| \sum_l x_l^2}$. For each protocol shown in the graph, the area to the left and below of the tradeoff curve is the achievable region (*i.e.*, every (rate, fairness) point in this region can be obtained), and the area to the right and above of the tradeoff curve is the infeasible region (*i.e.*, it is impossible to have any combination of (rate, fairness) represented by points in this region). It is impossible to operate in the infeasible region and inferior to operate in the interior of the achievable region. Operating on the boundary of the achievable region, *i.e.*, the Pareto optimal tradeoff curve, is the best. Points on the Pareto optimal tradeoff curve are not comparable, which point is better depends on the desired tradeoff between efficiency and fairness. Since the BEB protocol is a static protocol, it always provides the same efficiency (rate) and fairness regardless of the choice of utility functions. Hence, we cannot flexibly control the efficiency-fairness tradeoff in the BEB protocol. Algorithm 1 and its variant achieve higher network utility and wider dynamic range of rate-fairness tradeoff.

⁴The performance of the BEB protocol highly depends on the choice of maximum and minimum window sizes, W_l^{max} and W_l^{min} . It turns out that for the network in Figure 5, the average-performance parameters are: $W_l^{max} = 20$ and $W_l^{min} = 10$.

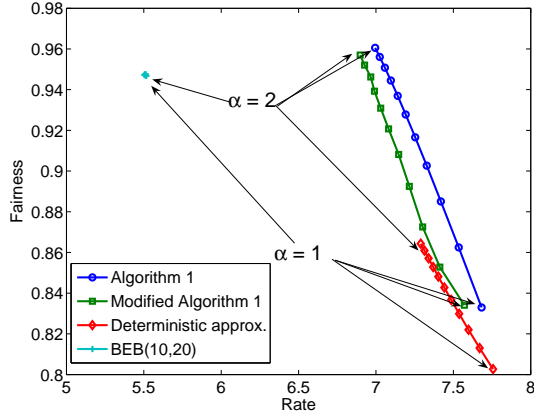


Figure 9: Comparison of rate-fairness tradeoff.

3 Vertical Decomposition

In this section, we turn to vertical decomposition across the protocol stack. Intuitively, allocating limited resources among competing users can be conducted through a variety of combinations. For example, to mitigate congestion at a bottleneck link, source rates can be reduced through congestion control, link capacity can be increased through power control, or alternative routes can be chosen. Four case studies from our recent publications [106, 35, 13, 55, 57, 11] are summarized in Section 3.1, with more details on the first two cases, which illustrate the applications to the analysis and design aspects, respectively. We will formulate generalized NUM problems to capture the interactions across such functional modules. It can be decomposed into subproblems each of which solved by a layer, and the interface between the layers represented by some function of the optimization variables. Then in Section 3.2, we will show that these case studies have only used a subset of alternative layering architectures.

3.1 Case Studies

3.1.1 Case 1: Jointly optimal congestion control and routing

Suppose that there are K^s acyclic paths from source s to its destination, represented by a $L \times K^s$ 0–1 matrix \mathbf{H}^s where

$$H_{lj}^s = \begin{cases} 1, & \text{if path } j \text{ of source } s \text{ uses link } l \\ 0, & \text{otherwise.} \end{cases}$$

Let \mathcal{H}^s be the set of all columns of \mathbf{H}^s that represents all the available paths to s . Define the $L \times K$ matrix \mathbf{H} as

$$\mathbf{H} = [\mathbf{H}^1 \dots \mathbf{H}^N]$$

where $K := \sum_s K^s$. \mathbf{H} defines the physical topology of the network.

Let \mathbf{w}^s be a $K^s \times 1$ vector where the j th entry represents the fraction of i 's flow on its j th path such that

$$w_j^s \geq 0 \quad \forall j \quad \text{and} \quad \mathbf{1}^T \mathbf{w}^s = 1$$

where $\mathbf{1}$ is a vector of an appropriate dimension with the value 1 in every entry. We require $w_j^s \in \{0, 1\}$ for single path routing, and allow $w_j^s \in [0, 1]$ for multipath routing. Collect the vectors \mathbf{w}^s , $s = 1, \dots, N$, into a $K \times N$ block-diagonal matrix \mathbf{W} . Let \mathcal{W}_n be the set of all such matrices corresponding to single path routing defined as

$$\{\mathbf{W} \mid \mathbf{W} = \text{diag}(\mathbf{w}^1, \dots, \mathbf{w}^N) \in \{0, 1\}^{K \times N}, \mathbf{1}^T \mathbf{w}^s = 1, \forall s.\}$$

Define the corresponding set \mathcal{W}_m for multipath routing as

$$\{\mathbf{W} \mid \mathbf{W} = \text{diag}(\mathbf{w}^1, \dots, \mathbf{w}^N) \in [0, 1]^{K \times N}, \mathbf{1}^T \mathbf{w}^s = 1, \forall s.\} \quad (47)$$

As mentioned above, \mathbf{H} defines the set of acyclic paths available to each source, and \mathbf{W} defines how the sources load balance across these paths. Their product defines a $L \times N$ routing matrix $\mathbf{R} = \mathbf{H}\mathbf{W}$ that specifies the fraction of s 's flow at each link l . The set of all single-path routing matrices is

$$\mathcal{R}_n = \{ \mathbf{R} \mid \mathbf{R} = \mathbf{H}\mathbf{W}, \mathbf{W} \in \mathcal{W}_n \} \quad (48)$$

and the set of all multipath routing matrices is

$$\mathcal{R}_m = \{ \mathbf{R} \mid \mathbf{R} = \mathbf{H}\mathbf{W}, \mathbf{W} \in \mathcal{W}_m \}. \quad (49)$$

The difference between single-path routing and multipath routing is the integer constraint on \mathbf{W} and \mathbf{R} . A single-path routing matrix in \mathcal{R}_n is an 0-1 matrix:

$$R_{ls} = \begin{cases} 1 & \text{if link } l \text{ is in the path of source } s \\ 0 & \text{otherwise} \end{cases}$$

A multipath routing matrix in \mathcal{R}_m is one whose entries are in the range $[0, 1]$:

$$R_{ls} \begin{cases} > 0, & \text{if link } l \text{ is in a path of source } s \\ = 0, & \text{otherwise.} \end{cases}$$

The path of source s is denoted by $\mathbf{r}^s = [R_{1s} \dots R_{Ls}]^T$, the s th column of the routing matrix \mathbf{R} . We now model the interaction of congestion control at the transport layer and shortest-path routing at the network layer.

We first consider the situation where TCP-AQM operates at a faster timescale than routing updates. We assume a *single* path is selected for each source-destination pair that minimizes the sum of the link costs in the path, for some appropriate definition of link cost. In particular, traffic is not split across multiple paths from the source to the destination even if they are available. This models, *e.g.*, IP routing within an Autonomous System. We focus

on the timescale of the route changes, and assume TCP–AQM is stable and converges instantly to equilibrium after a route change. As explained in the last section, we interpret the equilibria of various TCP and AQM algorithms as solutions of NUM and its dual.

Specifically, let $\mathbf{R}(t) \in \mathcal{R}_n$ be the (single-path) routing in period t . Let the equilibrium rates $\mathbf{x}(t) = \mathbf{x}(\mathbf{R}(t))$ and prices $\boldsymbol{\lambda}(t) = \boldsymbol{\lambda}(\mathbf{R}(t))$ generated by TCP–AQM in period t , respectively, be the optimal primal and dual solutions, *i.e.*,

$$\mathbf{x}(t) = \arg \max_{\mathbf{x} \geq 0} \sum_s U_s(x_s) \quad \text{s. t. } \mathbf{R}(t)\mathbf{x} \leq \mathbf{c} \quad (50)$$

$$\boldsymbol{\lambda}(t) = \arg \min_{\boldsymbol{\lambda} \geq 0} \sum_s \max_{x_s \geq 0} \left(U_s(x_s) - x_s \sum_l R_{ls}(t)\lambda_l \right) + \sum_l c_l \lambda_l \quad (51)$$

The link costs used in routing decision in period t are the congestion prices $\lambda_l(t)$. Each source computes its new route $\mathbf{r}^s(t+1) \in \mathcal{H}^s$ individually that minimizes the total cost on its path:

$$\mathbf{r}^s(t+1) = \arg \min_{\mathbf{r}^s \in \mathcal{H}^s} \sum_l \lambda_l(t) r_l^s \quad (52)$$

We say that $(\mathbf{R}^*, \mathbf{x}^*, \boldsymbol{\lambda}^*)$ is an *equilibrium of TCP/IP* if it is a fixed point of (50)–(52), *i.e.*, starting from routing \mathbf{R}^* and associated $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$, the above iterations yield $(\mathbf{R}^*, \mathbf{x}^*, \boldsymbol{\lambda}^*)$ in the subsequent periods.

We now characterize the condition under which TCP/IP as modeled by (50)–(52) has an equilibrium. Consider the following generalized NUM:

$$\max_{\mathbf{R} \in \mathcal{R}_n} \max_{\mathbf{x} \geq 0} \sum_s U_s(x_s) \quad \text{s. t. } \mathbf{R}\mathbf{x} \leq \mathbf{c}, \quad (53)$$

and its Lagrange dual problem:

$$\min_{\boldsymbol{\lambda} \geq 0} \sum_s \max_{x_s \geq 0} \left(U_s(x_s) - x_s \min_{\mathbf{r}^s \in \mathcal{H}^s} \sum_l R_{ls} \lambda_l \right) + \sum_l c_l \lambda_l, \quad (54)$$

where \mathbf{r}^s is the s th column of \mathbf{R} with $r_l^s = R_{ls}$. While (50) maximizes utility over source rates only, problem (53) maximizes utility over both rates and routes. While (50) is a convex optimization problem without duality gap, problem (53) is nonconvex because the variable \mathbf{R} is discrete, and generally has a duality gap.⁵ The interesting feature of the dual problem (54) is that the maximization over \mathbf{R} takes the form of minimum-cost routing with congestion prices $\boldsymbol{\lambda}$ generated by TCP–AQM as link costs. This suggests that TCP/IP might turn out to be a distributed algorithm that attempts to maximize utility, with a proper choice of link costs. This is indeed true – when an equilibrium of TCP/IP exists.

⁵The nonlinear constraint $\mathbf{R}\mathbf{x} \leq \mathbf{c}$ can be converted into a linear constraint (see proof of Theorem 8 in [106]), so the integer constraint on \mathbf{R} is the real source of difficulty.

Theorem 7. *An equilibrium $(\mathbf{R}^*, \mathbf{x}^*, \boldsymbol{\lambda}^*)$ of TCP/IP exists if and only if there is no duality gap between (53) and (54). In this case, the equilibrium $(\mathbf{R}^*, \mathbf{x}^*, \boldsymbol{\lambda}^*)$ is a solution of (53) and (54).*

Method 10. *Analyzing a given cross-layer interaction through generalized NUM.*

Hence, one can regard the layering of TCP and IP as a decomposition of the utility maximization problem over source rates and routes into a distributed and decentralized algorithm, carried out on two different timescales, in the sense that an equilibrium of the TCP/IP iteration (50)–(52), if it exists, solves (53) and (54). An equilibrium may not exist. Even if it does, it may not be stable [106].

The duality gap can be interpreted as a measure of “cost for not splitting”. To elaborate, consider the Lagrangian

$$L(\mathbf{R}, \mathbf{x}, \boldsymbol{\lambda}) = \sum_s \left(U_s(x_s) - x_s \sum_l R_{ls} \lambda_l \right) + \sum_l c_l \lambda_l.$$

The primal (53) and dual (54) can then be expressed respectively as:

$$\begin{aligned} V_{np} &= \max_{\mathbf{R} \in \mathcal{R}_n, \mathbf{x} \geq 0} \min_{\boldsymbol{\lambda} \geq 0} L(\mathbf{R}, \mathbf{x}, \boldsymbol{\lambda}), \\ V_{nd} &= \min_{\boldsymbol{\lambda} \geq 0} \max_{\mathbf{R} \in \mathcal{R}_n, \mathbf{x} \geq 0} L(\mathbf{R}, \mathbf{x}, \boldsymbol{\lambda}). \end{aligned}$$

If we allow sources to distribute their traffic among multiple paths available to them, then the corresponding problems for multi-path routing are

$$\begin{aligned} V_{mp} &= \max_{\mathbf{R} \in \mathcal{R}_m, \mathbf{x} \geq 0} \min_{\boldsymbol{\lambda} \geq 0} L(\mathbf{R}, \mathbf{x}, \boldsymbol{\lambda}), \\ V_{md} &= \min_{\boldsymbol{\lambda} \geq 0} \max_{\mathbf{R} \in \mathcal{R}_m, \mathbf{x} \geq 0} L(\mathbf{R}, \mathbf{x}, \boldsymbol{\lambda}). \end{aligned} \tag{55}$$

Since $\mathcal{R}_n \subseteq \mathcal{R}_m$, $V_{np} \leq V_{mp}$. The next result clarifies the relation among these four problems.

Theorem 8. $V_{sp} \leq V_{sd} = V_{mp} = V_{md}$.

According to Theorem 7, TCP/IP has an equilibrium exactly when there is no duality gap in the single-path utility maximization, *i.e.*, when $V_{np} = V_{nd}$. Theorem 8 then says that in this case, there is no penalty in not splitting the traffic, *i.e.*, single-path routing performs as well as multipath routing, $V_{np} = V_{mp}$. Multipath routing achieves a strictly higher utility V_{mp} precisely when TCP/IP has no equilibrium, in which case the TCP/IP iteration (50)–(52) cannot converge, let alone solving the single-path utility maximization problem (53) or (54). In this case the problem (53) and its dual (54) do not characterize TCP/IP, but their gap measures the loss in utility in restricting routing to single-path and is of independent interest.

Even though shortest-path routing is polynomial, the single-path utility maximization is NP-hard.

Theorem 9. *The primal problem (53) is NP-hard.*

Theorem 9 is proved [106] by reducing all instances of the integer partition problem to some instances of the primal problem (53). Theorem 8 however implies that the sub-class of the utility maximization problems with no duality gap are in P, since they are equivalent to multipath problems which are convex programs and hence polynomial-time solvable. It is a common phenomenon for sub-classes of NP-hard problems to have polynomial-time algorithms (*e.g.*, satisfiability is NP-hard, and yet 2-SAT is in P). Informally, the hard problems are those with nonzero duality gap.

Theorem 7 suggests using pure prices $\lambda(t)$ generated by TCP–AQM as link costs because in this case, an equilibrium of TCP/IP, when it exists, maximizes aggregate utility over both rates and routes. It is shown in [106] however that such an equilibrium can be unstable, and hence not attainable by TCP/IP. Routing can be stabilized by including a strictly positive traffic-insensitive component in the link cost, in addition to congestion price. Stabilization however reduces the achievable utility. There thus seems to be an inevitable tradeoff between achievable utility and routing stability, when link costs are fixed. If the link capacities are optimally provisioned, however, pure *static* routing, which is necessarily stable, is enough to maximize utility. Moreover, it is optimal even within the class of multipath routing: again, there is no penalty in not splitting traffic across multiple paths.

In [35], three alternative timescale separations are further considered for the joint congestion control and shortest-path routing dynamics based on congestion price. Analytic characterizations and simulation experiments demonstrate how the step size of the congestion-control algorithm affects the stability of the system models, and how the timescale of each control loop and homogeneity of link capacities affect system stability and optimality. In particular, the stringent conditions on capacity configuration for TCP/IP interaction to remain stable suggest that congestion price, on its own, would be a poor “layering price” for TCP and (dynamic routing based) IP in practice. Alternative traffic engineering methods should be considered instead [36].

3.1.2 Case 2: Jointly optimal congestion control and physical layer resource allocation

The physical layer provides the transmission pipes for end-to-end transport. Adaptive resource allocation in physical layer, such as power control and error correction coding considered in this subsection, produces intriguing interactions with transport layer.

First consider a wireless multihop network with an established logical topology represented by \mathbf{R} or equivalently $\{S(l)\}$, where some nodes are sources of transmission and some nodes act as relay nodes. Revisiting the basic NUM (3), for which TCP congestion control solves, we observe that in an interference-limited wireless network, data rates attainable on wireless links are not fixed numbers \mathbf{c} as in (3), and instead can be written as global and nonlinear functions of the transmit power vector \mathbf{P} and channel conditions:

$$c_l(\mathbf{P}) = \frac{1}{T} \log(1 + K \text{SIR}_l(\mathbf{P})), \forall l.$$

Here constant T is the symbol period, which will be assumed to be one unit without loss of generality, and constant $K = \frac{-\phi_1}{\log(\phi_2 \text{BER})}$ where ϕ_1, ϕ_2 are constants depending on the modulation and BER is the required bit error rate.

The signal to interference ratio for link l is defined as $\text{SIR}_l(\mathbf{P}) = \frac{P_l G_{ll}}{\sum_{k \neq l} P_k G_{lk} + \nu_l}$ for a given set of path losses G_{lk}

(from the transmitter on logical link k to the receiver on logical link l) and a given set of noises n_l (for the receiver on logical link l). The G_{lk} factors incorporate propagation loss, spreading gain, and other normalization constants. Notice that G_{ll} is the path gain on link l (from the transmitter on logical link l to the intended receiver on the same logical link). With reasonable spreading gain, G_{ll} is much larger than G_{lk} , $k \neq l$, and assuming that not too many close-by nodes transmit at the same time, $KSIR_l$ is much larger than 1. In this case, c_l can be approximated as $\log(KSIR_l(\mathbf{P}))$.

With the above assumptions, we have specified the following generalized NUM with “elastic” link capacities:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && \sum_{s \in S(l)} x_s \leq c_l(\mathbf{P}), \quad \forall l \\ & && \mathbf{x}, \mathbf{P} \geq 0 \end{aligned} \tag{56}$$

where the optimization variables are both source rates \mathbf{x} and transmit powers \mathbf{P} . The key difference from the standard utility maximization (3) is that each link capacity c_l is now a function of the new optimization variables: the transmit powers \mathbf{P} . The design space is enlarged from \mathbf{x} to both \mathbf{x} and \mathbf{P} , which are clearly coupled in (56). Linear flow constraints on \mathbf{x} become nonlinear constraints on (\mathbf{x}, \mathbf{P}) . In practice, problem (56) is also constrained by the maximum and minimum transmit powers allowed at each transmitter on link l : $P_{l,min} \leq P_l \leq P_{l,max}$, $\forall l$.

The major challenges are the two global dependencies in (56):

- Source rates \mathbf{x} and link capacities \mathbf{c} are globally coupled across the network, as reflected in the range of summation $\{s \in S(l)\}$ in the constraints in (56).
- Each link capacity $c_l(\mathbf{P})$, in terms of the attainable data rate under a given power vector, is a global function of all the interfering powers.

We present the following distributive algorithm and later prove that it converges to the global optimum of (56). To make the algorithm and its analysis concrete, we focus on delay-based price and TCP Vegas window update (as reflected in items 1 and 2 in the algorithm, respectively) and the corresponding logarithmic utility maximization over (\mathbf{x}, \mathbf{P}) , where α_s is a constant parameter in TCP Vegas:

$$\begin{aligned} & \text{maximize} && \sum_s \alpha_s \log x_s \\ & \text{subject to} && \sum_{s \in S(l)} x_s \leq c_l(\mathbf{P}), \quad \forall l \\ & && \mathbf{x}, \mathbf{P} \geq 0. \end{aligned} \tag{57}$$

Algorithm 2. Joint Congestion-control and Power-control Algorithm

During each time slot t , the following four updates are carried out simultaneously until convergence:

1. At each intermediate node, a weighted queuing delay λ_l is implicitly updated, where $\beta_1 > 0$ is a constant:

$$\lambda_l(t+1) = \left[\lambda_l(t) + \frac{\beta_1}{c_l(t)} \left(\sum_{s \in S(l)} x_s(t) - c_l(t) \right) \right]^+. \quad (58)$$

2. At each source, total delay D_s is measured and used to update the TCP window size w_s . Consequently, the source rate x_s is updated:

$$w_s(t+1) = \begin{cases} w_s(t) + \frac{1}{D_s(t)} & \text{if } \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} < \alpha_s \\ w_s(t) - \frac{1}{D_s(t)} & \text{if } \frac{w_s(t)}{d_s} - \frac{w_s(t)}{D_s(t)} > \alpha_s \\ w_s(t) & \text{else.} \end{cases} \quad (59)$$

$$x_s(t+1) = \frac{w_s(t+1)}{D_s(t)}.$$

3. Each transmitter j calculates a message $m_j(t) \in \mathbf{R}_+$ ⁶ based on locally measurable quantities, and passes the message to all other transmitters by a flooding protocol:

$$m_j(t) = \frac{\lambda_j(t) \text{SIR}_j(t)}{P_j(t) G_{jj}}.$$

4. Each transmitter updates its power based on locally measurable quantities and the received messages, where $\beta_2 > 0$ is a constant:

$$P_l(t+1) = P_l(t) + \frac{\beta_2 \lambda_l(t)}{P_l(t)} - \beta_2 \sum_{j \neq l} G_{lj} m_j(t). \quad (60)$$

With the minimum and maximum transmit power constraint $(P_{l,\min}, P_{l,\max})$ on each transmitter, the updated power is projected onto the interval $[P_{l,\min}, P_{l,\max}]$.

Item 2 is simply the TCP Vegas window update [8]. Item 1 is a modified version of queuing delay price update [64] (and the original update [8] is an approximation of item 1). Items 3 and 4 describe a new power control using message passing. Taking in the current values of $\frac{\lambda_j(t) \text{SIR}_j(t)}{P_j(t) G_{jj}}$ as the messages from other transmitters indexed by j , the transmitter on link l adjusts its power level in the next time slot in two ways: first increases power directly proportional to the current price (e.g., queuing delay in TCP Vegas) and inversely proportional to the current power level, then decreases power by a weighted sum of the messages from all other transmitters, where the weights are the path losses G_{lj} ⁷. Intuitively, if the local queuing delay is high, transmit power should increase, with a more

⁶Note that this does not denote price-mapping functions as in Subsection 2.1.4.

⁷This facilitates a graceful reduction of message passing scope since messages from far-away neighbors are weighted much less.

moderate increase when the current power level is already high. If queuing delays on other links are high, transmit power should decrease in order to reduce interference on those links.

To compute m_j , the values of queuing delay λ_j , signal-interference-ratio SIR_j , and received power level $P_j G_{jj}$ can be directly measured by node j locally. This algorithm only uses the resulting message m_j but *not* the individual values of λ_j , SIR_j , P_j and G_{jj} . Each message is a real number to be explicitly passed. To conduct the power update, G_{lj} factors are assumed to be estimated through training sequences.

It is important to note that there is no need to change the existing TCP congestion control and queue management algorithms. All that is needed to achieve the joint and global optimum of (57) is to utilize the values of weighted queuing delay in designing power control algorithm in the physical layer. The stability and optimality of this layering price can be stated through the following

Theorem 10. *For small enough constants β_1 and β_2 , Algorithm 2 (58,59,60) converges to the global optimum of the joint congestion control and power control problem (57).*

The key steps of vertical decomposition with congestion price as the layering price are again through dual decomposition. We first associate a Lagrange multiplier λ_l for each of the constraints $\sum_{s \in S(l)} x_s \leq c_l(\mathbf{P})$. Using the KKT optimality conditions [3, 7], solving problem (57) (or (56)) is equivalent to satisfying the complementary slackness condition and finding the stationary points of the Lagrangian.

Complementary slackness condition states that at optimality, the product of the dual variable and the associated primal constraint must be zero. This condition is satisfied since the equilibrium queuing delay must be zero if the total equilibrium ingress rate at a router is strictly smaller than the egress link capacity. We also need to find the stationary points of the Lagrangian: $L_{system}(\mathbf{x}, \mathbf{P}, \boldsymbol{\lambda}) = \sum_s U_s(x_s) - \sum_l \lambda_l \sum_{s \in S(l)} x_s + \sum_l \lambda_l c_l(\mathbf{P})$. By linearity of the differentiation operator, this can be decomposed into two separate maximization problems:

$$\begin{aligned} \text{maximize}_{\mathbf{x} \geq 0} \quad & L_{congestion}(\mathbf{x}, \boldsymbol{\lambda}) = \sum_s U_s(x_s) - \sum_s \sum_{l \in L(s)} \lambda_l x_s, \\ \text{maximize}_{\mathbf{P} \geq 0} \quad & L_{power}(\mathbf{P}, \boldsymbol{\lambda}) = \sum_l \lambda_l c_l(\mathbf{P}). \end{aligned}$$

The first maximization is already implicitly solved by the congestion control mechanism for different U_s (such as TCP Vegas for $U_s(x_s) = \alpha_s \log x_s$). But we still need to solve the second maximization, using the Lagrange multipliers $\boldsymbol{\lambda}$ as the shadow prices to allocate exactly the right power to each transmitter, thus increasing the link data rates and reducing congestion at the network bottlenecks. Although the data rate on each wireless link is a global function of all the transmit powers, distributed solution is still feasible through distributed gradient method with the help of message passing. Issues arising in practical implementation, such as asynchronous update and reduced message passing, and their impacts on convergence and optimality, are discussed in [13].

Method 11. *Dual decomposition for jointly optimal cross layer design.*

The logical topology and routes for four multi-hop connections are shown in Figure 10 for a numerical example. Sources at each of the four flows use TCP Vegas window updates. The path losses G_{ij} are determined by the relative physical distances d_{ij} , which we vary in different experiments, by $G_{ij} = d_{ij}^{-4}$. The target BER is 10^{-3} on each logical link.

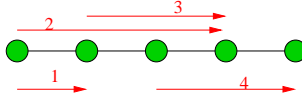


Figure 10: Logical topology and connections for a joint congestion control power control example.

Transmit powers, as regulated by the proposed distributed power control, and source rates, as regulated through TCP Vegas window update, are shown in Figure 11. The initial conditions of the graphs are based on the equilibrium states of TCP Vegas with fixed power levels of $2.5mW$. With power control, the transmit powers \mathbf{P} distributively adapt to induce a “smart” capacity \mathbf{c} and queuing delay λ configuration in the network, which in turn lead to increases in end-to-end throughput as indicated by the rise in all the allowed source rates. Notice that some link capacities actually decrease while the capacities on the bottleneck links rise to maximize the total network utility. This is achieved through a distributive adaptation of power, which lowers the power levels that cause most interference on the links that are becoming a bottleneck in the dynamic demand-supply matching process. Confirming our intuition, such a “smart” allocation of power tends to reduce the spread of queuing delays, thus preventing any link from becoming a bottleneck. Queuing delays on the four links do not become the same though, due to the asymmetry in traffic load on the links and different weights in the logarithmic utility objective functions. We indeed achieve the primary goal of this co-design across the transport and physical layers. The end-to-end throughput per watt of power transmitted is 82% higher with power control.

In the second half of this subsection, we discuss the interaction of per-hop adaptive channel coding with end-to-end congestion control. At the end hosts, the utility for each user depends on both transmission rate and signal quality, with an intrinsic tradeoff between the two. At the same time, each link may also provide a “fatter” (or “thinner”) transmission “pipe” by allowing a higher (or lower) decoding error probability.

In the basic NUM, the convexity and separability properties of the optimization problem readily lead to a distributed algorithm that converges to the globally optimal rate allocation. The generalized NUM problems for joint rate-reliability provisioning turn out to be non-separable and non-convex. We review a price-based distributed algorithm and its convergence to the globally optimal rate-reliability tradeoff under readily-verifiable sufficient conditions on link coding block lengths and user utility curvatures. In contrast to standard price-based rate control algorithms for the basic NUM, in which each link provides the same congestion price to each of its users and/or each user provides its willingness to pay for rate allocation to the network, in the joint rate-reliability algorithms each link provides a possibly different congestion price to each of its users and each user also provides its willingness to pay for its own reliability to the network.

On some communication links, physical layer’s adaptive channel coding (*i.e.*, error correction coding) can change the information “pipe” sizes and decoding error probabilities, *e.g.*, through adaptive channel coding in Digital Subscriber Loop (DSL) broadband access networks or adaptive diversity-multiplexing control in Multiple-Input-Multiple-Output (MIMO) wireless systems. Then each link capacity is a function of the signal quality (*i.e.*, decoding reliability) attained on that link. A higher throughput can be obtained on a link at the expense of lower decoding reliability, which in turn lowers the end-to-end signal quality for sources traversing the link and

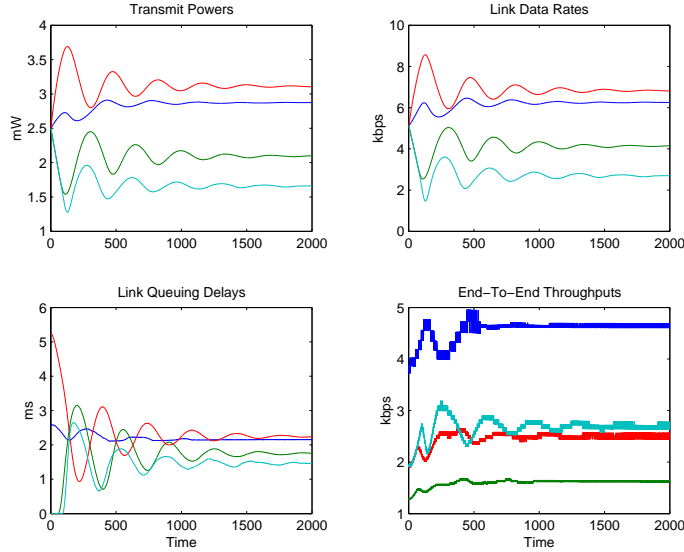


Figure 11: A typical numerical example of joint TCP Vegas congestion control and power control. The top left graph shows the primal variables \mathbf{P} . The lower left graph shows the dual variables λ . The lower right graph shows the primal variables \mathbf{x} , *i.e.*, the end-to-end throughput. In order of their y-axis values after convergence, the curves in the top left, top right, and bottom left graphs are indexed by the third, first, second, and fourth links in Figure 10. The curves in the bottom right graph are indexed by flows 1, 4, 3, 2.

reduces users' utilities, thus leading to an intrinsic tradeoff between rate and reliability. This tradeoff also provides an additional degree of freedom for improving each user's utility as well as system efficiency. For example, if we allow lower decoding reliability, thus higher information capacity, on the more congested links, and higher decoding reliability, thus lower information capacity, on the less congested links, we may improve the end-to-end rate and reliability performance of each user. Clearly, rate-reliability tradeoff is globally coupled across the links and users.

In the case where the rate-reliability tradeoff is controlled through the code rate of each source on each link, there are two possible policies: integrated dynamic reliability policy and differentiated dynamic reliability policy. In integrated policy, a link provides the same error probability (*i.e.*, the same code rate) to each of the sources traversing it. Since a link provides the same code rate to each of its sources, it must provide the lowest code rate that satisfies the requirement of the source with the highest reliability. This motivates a more general approach called differentiated policy to fully exploit the rate-reliability tradeoff when there exist multi-class sources (*i.e.*, sources with different reliability requirements) in the network. Under the differentiated dynamic reliability policy, a link can provide a different error probability (*i.e.*, a different code rate) to each of the sources using this link.

We assume that each source s has a utility function $U_s(x_s, \rho_s)$, where x_s is an information data rate and ρ_s is

reliability of source s . We assume that the utility function is a continuous, increasing, and strictly concave function of x_s and ρ_s . Each source s has a minimum reliability requirement ρ_s^{min} . The reliability of source s is defined as

$$\rho_s = 1 - p^s,$$

where p^s is the end-to-end error probability of source s . Each link l has its maximum transmission capacity C_l^{max} . After link l receives the data of source s from the upstream link, it first decodes it to extract the information data of the source and encodes it again with its own code rate, $r_{l,s}$, where the code rate is defined by the ratio of the information data rate x_s at the input of the encoder to the transmission data rate $t_{l,s}$ at the output of the encoder [31]. This allows a link to adjust the transmission rate and the error probability of the sources, since the transmission rate of source s at link l can be defined as

$$t_{l,s} = \frac{x_s}{r_{l,s}},$$

and the error probability of source s at link l can be defined as a function of $r_{l,s}$ by

$$p_{l,s} = E_l(r_{l,s}),$$

which is assumed to be an increasing function of $r_{l,s}$. Rarely there are analytic formula for $E_l(r_{l,s})$, and we will use various upper bounds on this function. The end-to-end error probability for each source s is

$$p^s = 1 - \prod_{l \in L(s)} (1 - p_{l,s}) = 1 - \prod_{l \in L(s)} (1 - E_l(r_{l,s})).$$

Assuming that the error probability of each link is small (*i.e.*, $p_{l,s} \ll 1$), we can approximate the end-to-end error probability of source s as

$$p^s \approx \sum_{l \in L(s)} p_{l,s} = \sum_{l \in L(s)} E_l(r_{l,s}).$$

Hence, the reliability of source s can be expressed as

$$\rho_s \approx 1 - \sum_{l \in L(s)} E_l(r_{l,s}).$$

Since each link l has a maximum transmission capacity C_l^{max} , the sum of transmission rates of sources that are traversing each link cannot exceed C_l^{max} :

$$\sum_{s \in S(l)} t_{l,s} = \sum_{s \in S(l)} \frac{x_s}{r_{l,s}} \leq C_l^{max}, \quad \forall l.$$

For (the more general) differentiated dynamic reliability policy in which a link may provide a different code rate to each of the sources traversing it, the associated generalized NUM becomes the following problem with variables $\mathbf{x}, \rho, \mathbf{r}$:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s, \rho_s) \\ & \text{subject to} && \rho_s \leq 1 - \sum_{l \in L(s)} E_l(r_{l,s}), \quad \forall s \\ & && \sum_{s \in S(l)} \frac{x_s}{r_{l,s}} \leq C_l^{max}, \quad \forall l \\ & && \rho_s^{min} \leq \rho_s \leq 1, \quad \forall s \\ & && 0 \leq r_{l,s} \leq 1, \quad \forall l, s \in S(l). \end{aligned} \tag{61}$$

There are two main difficulties in distributively and globally solving the above problem. The first one is the convexity of $E_l(r_{l,s})$. If random coding based on binary coded signals is used, a standard upper bound on the error probability is:

$$p_l < \frac{1}{2} 2^{-M(R_0 - r_l)},$$

where M is the block length and R_0 is the cutoff rate. In this case, $E_l(r_l) = \frac{1}{2} 2^{-M(R_0 - r_l)}$ is a convex function for given M and R_0 . A more general approach for discrete memoryless channel models is to use the random code ensemble error exponent [31] that upper bounds the decoding error probability:

$$p_l \leq \exp(-M E_r(r_l)),$$

where M is the codeword block length and $E_r(r_l)$ is the random coding exponent function, which is defined as [31]

$$E_r(r_l) = \max_{0 \leq \mu \leq 1} \max_{\mathbf{Q}} [E_o(\mu, \mathbf{Q}) - \mu r_l],$$

where

$$E_o(\mu, \mathbf{Q}) = -\log \sum_{j=0}^{J-1} \left[\sum_{k=0}^{K-1} Q_k \bar{P}_{jk}^{1/(1+\mu)} \right]^{1+\mu},$$

K is the size of input alphabet, J is the size of output alphabet, Q_k is the probability that input letter k is chosen, and \bar{P}_{jk} is the probability that output letter j is received given that input letter k is transmitted.

In general, $E_l(r_l) = \exp(-M E_r(r_l))$ may not be convex, even though it is known [31] that $E_r(r_l)$ is a convex function. However, the following lemma provides a sufficient condition for its convexity.

Lemma 1. *If the absolute value of the first derivatives of $E_r(r_l)$ is bounded away from 0 and absolute value of the second derivative of $E_r(r_l)$ is upper bounded, then for a large enough codeword block length M , $E_l(r_l)$ is a convex function.*

Method 12. *Computing conditions under which a general constraint set is convex.*

The second difficulty is the global coupling of constraints $\sum_{s \in S(l)} \frac{x_s}{r_{l,s}} \leq C_l^{max}$. This problem is tackled by first introducing auxiliary variables $c_{l,s}$, which can be interpreted as the allocated transmission capacity to source s at link l :

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s, \rho_s) \\ & \text{subject to} && \rho_s \leq 1 - \sum_{l \in L(s)} E_l(r_{l,s}), \quad \forall s \\ & && \frac{x_s}{r_{l,s}} \leq c_{l,s}, \quad \forall l, s \in S(l) \\ & && \sum_{s \in S(l)} c_{l,s} \leq C_l^{max}, \quad \forall l \\ & && \rho_s^{min} \leq \rho_s \leq 1, \quad \forall s \\ & && 0 \leq r_{l,s} \leq 1, \quad \forall l, s \in S(l) \\ & && 0 \leq c_{l,s} \leq C_l^{max}, \quad \forall l, s \in S(l). \end{aligned} \tag{62}$$

Note that effectively a new ‘‘layer’’ has been introduced into the problem: scheduling of flows by deciding bandwidth sharing on each link $\{c_{l,s}\}$.

Method 13. *Introducing a new “layer” to decouple a generalized NUM.*

A log change of variable $x'_s = \log x_s$ can be used to decouple the above problem for horizontal decomposition. Define a modified utility function $U'_s(x'_s, \rho_s) = U_s(e^{x'_s}, \rho_s)$, which needs to be concave in order for the transformed problem to remain a convex optimization problem, similar to the curvature condition on utility function in Subsection 2.2.2.

Define

$$\begin{aligned} g_s(x_s, \rho_s) &= \frac{\partial^2 U_s(x_s, \rho_s)}{\partial x_s^2} x_s + \frac{\partial U_s(x_s)}{\partial x_s}, \\ h_s(x_s, \rho_s) &= \left(\left(\frac{\partial^2 U_s(x_s, \rho_s)}{\partial x_s \partial \rho_s} \right)^2 - \frac{\partial^2 U_s(x_s, \rho_s)}{\partial x_s^2} \frac{\partial^2 U_s(x_s, \rho_s)}{\partial \rho_s^2} \right) x_s - \frac{\partial^2 U_s(x_s, \rho_s)}{\partial \rho_s^2} \frac{\partial U_s(x_s, \rho_s)}{\partial x_s}, \\ q_s(x_s, \rho_s) &= \frac{\partial^2 U_s(x_s, \rho_s)}{\partial \rho_s^2}. \end{aligned}$$

Lemma 2. *If $g_s(x_s, \rho_s) < 0$, $h_s(x_s, \rho_s) < 0$, and $q_s(x_s, \rho_s) < 0$, then $U'_s(x'_s, \rho_s)$ is a concave function of x'_s and ρ_s .*

Now the joint rate-reliability problem (61) can be solved distributively through dual decomposition.

Algorithm 3. *Differentiated Dynamic Reliability Policy Algorithm.*

In each iteration t , by solving (63) over (x'_s, ρ_s) , each source s determines its information data rate and requested reliability (i.e., $x'_s(t)$ or equivalently, $x_s(t) = e^{x'_s(t)}$, and $\rho_s(t)$) that maximize its net utility based on the prices in the current iteration. Furthermore, by price update equation (64), the source adjusts its offered price per unit reliability for the next iteration.

Source problem and reliability price update at source s :

- Source problem:

$$\begin{aligned} &\text{maximize} && U_s(x'_s, \rho_s) - \lambda^s(t)x'_s - \mu_s(t)\rho_s \\ &\text{subject to} && \rho_s^{\min} \leq \rho_s \leq 1, \end{aligned} \tag{63}$$

where $\lambda^s(t) = \sum_{l \in L(s)} \lambda_{l,s}(t)$ is the end-to-end congestion price at iteration t .

- Price update (where step size can be set to $\beta(t) = \frac{\beta_0}{t}$ for some $\beta_0 > 0$):

$$\mu_s(t+1) = [\mu_s(t) - \beta(t)(\rho^s(t) - \rho_s(t))]^+, \tag{64}$$

where $\rho^s(t) = 1 - \sum_{l \in L(s)} E_l(r_{l,s}(t))$ is the end-to-end reliability at iteration t .

Concurrently in each iteration t , by solving problem (65) over $(c_{l,s}, r_{l,s})$, $\forall s \in S(l)$, each link l determines the allocated transmission capacity $c_{l,s}(t)$ and the code rate $r_{l,s}(t)$ of each of the sources using the link, so as to maximize the “net revenue” of the network based on the prices in the current iteration. In addition, by price update equation (66), the link adjusts its congestion price per unit rate for source s during the next iteration.

Link problem and congestion price update at link l :

- Link problem:

$$\begin{aligned} & \text{maximize} && \sum_{s \in S(l)} \lambda_{l,s}(t) (\log c_{l,s} + \log r_{l,s}) - \mu_s(t) E_l(r_{l,s}) \\ & \text{subject to} && \sum_{s \in S(l)} c_{l,s} \leq C_l^{\max} \\ & && 0 \leq c_{l,s} \leq C_l^{\max}, \quad s \in S(l) \\ & && 0 \leq r_{l,s} \leq 1, \quad s \in S(l). \end{aligned} \quad (65)$$

- Price update (where step size can be set to $\beta(t) = \frac{\beta_0}{t}$ for some $\beta_0 > 0$):

$$\begin{aligned} \lambda_{l,s}(t+1) &= [\lambda_{l,s}(t) - \beta(t) (\log c_{l,s}(t) + \log r_{l,s}(t) - x'_s(t))]^+ \\ &= [\lambda_{l,s}(t) - \beta(t) (\log c_{l,s}(t) + \log r_{l,s}(t) - \log x_s(t))]^+, \quad s \in S(l). \end{aligned} \quad (66)$$

In the above algorithm, to solve problem (63), source s needs to know $\lambda^s(t)$, the sum of congestion prices $\lambda_{l,s}(t)$'s of links that are along its path $L(s)$. This can be obtained by the notification from the links, *e.g.*, through acknowledgment packets. To carry out price update (64), the source needs to know the sum of error probabilities of the links that are along its path (*i.e.*, its own reliability that is provided by the network, $\rho^s(t)$). This can be obtained by the notification either from the links that determine the code rate for the source (by solving problem (65)) or from the destination that can measure its end-to-end reliability. To solve the link problem (65), each link l needs to know $\mu_s(t)$ from each of sources using this link l . This can be obtained by the notification from these sources. To carry out price update (66), the link needs to know the information data rate of each of the sources that are using it (*i.e.*, $x_s(t)$). This can be obtained by measuring it by the link itself.

Method 14. End user generated pricing for distributed update of metrics in user utility objective function.

Theorem 11. For sufficiently concave utilities and sufficiently strong codes, the dual variables $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\mu}(t)$ converge to the optimal dual solutions $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ and the corresponding primal variables \mathbf{x}'^* , ρ^* , \mathbf{c}^* , and \mathbf{r}^* are the globally optimal primal solutions of the joint rate-reliability problem, *i.e.*, $\mathbf{x}^* = (e^{x'_s})_{\forall s}$, ρ^* , \mathbf{c}^* , and \mathbf{r}^* are the globally optimal primal solutions of problem (61).

We now present numerical examples for the proposed algorithms by considering a simple network, shown in Figure 12, with a linear topology consisting of four links and eight users. Utility function for user s is $U_s(x_s, \rho_s)$ in the following α -fair form, shifted such that $U_s(x_s^{\min}, \rho_s^{\min}) = 0$ and $U_s(x_s^{\max}, \rho_s^{\max}) = 1$, and with utility on rate and utility on reliability summed up with a given weight θ_s between rate and reliability utilities:

$$U_s(x_s, \rho_s) = \theta_s \frac{x_s^{1-\alpha} - x_s^{\min(1-\alpha)}}{x_s^{\max(1-\alpha)} - x_s^{\min(1-\alpha)}} + (1 - \theta_s) \frac{\rho_s^{(1-\alpha)} - \rho_s^{\min(1-\alpha)}}{\rho_s^{\max(1-\alpha)} - \rho_s^{\min(1-\alpha)}}.$$

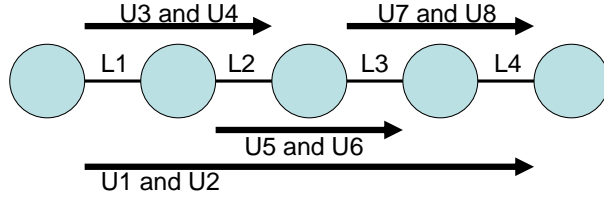


Figure 12: Network topology and flow routes for a rate-reliability tradeoff example.

Different weights θ_s are given to the eight users as follows:

$$\theta_s = \begin{cases} 0.5 - v, & \text{if } s \text{ is an odd number} \\ 0.5 + v, & \text{if } s \text{ is an even number} \end{cases} \quad (67)$$

and vary v from 0 to 0.5 in step size of 0.05.

The decoding error probability on each link l is assumed to be of the following form:

$$p_l = \frac{1}{2} \exp(-M(1 - r_l)).$$

We trace the globally optimal tradeoff curve between rate and reliability using differentiated and integrated dynamic reliability policies, and compare the network utility achieved by the following three schemes:

- Static reliability: each link provides a fixed error probability 0.025. Only rate control is performed to maximize the network utility.
- Integrated dynamic reliability: each link provides the same adjustable error probability to all its users.
- Differentiated dynamic reliability: each link provides a possibly different error probability to each of its users.

Figure 13 shows the globally optimal tradeoff curves between rate and reliability for a particular user, under the three policies of static reliability, integrated dynamic reliability, and differentiated dynamic reliability, respectively. The differentiated scheme shows a much larger dynamic range of tradeoff than both the integrated and static schemes. The gain in total network utility through joint rate and reliability control is shown in Figure 14.

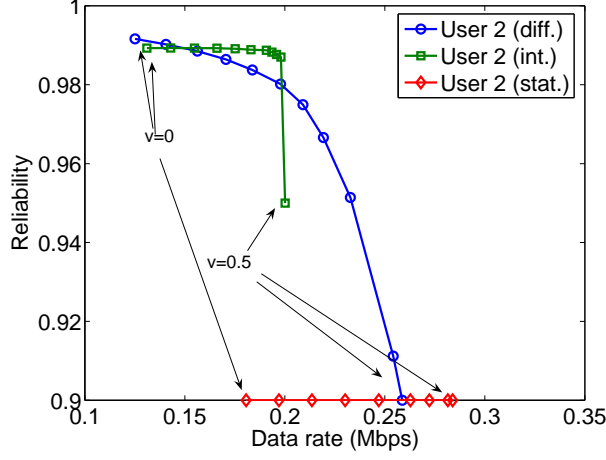


Figure 13: Comparison of data rate and reliability tradeoff in each policy for user 2, when θ_s are changed according to (67).

3.1.3 Case 3: Jointly optimal congestion and contention control

For joint end-to-end rate allocation and per-hop medium access control, the generalized NUM problem for random-access-based MAC and TCP can be formulated as follows:

$$\begin{aligned}
& \text{maximize} && \sum_s U_s(x_s) \\
& \text{subject to} && \sum_{s \in S(l)} x_s \leq c_l p_l \prod_{k \in N_{to}^I(l)} (1 - P^k), \quad \forall l \\
& && \sum_{l \in L_{out}(n)} p_l = P^n, \quad \forall n \\
& && x_s^{min} \leq x_s \leq x_s^{max}, \quad \forall s \\
& && 0 \leq P^n \leq 1, \quad \forall n \\
& && 0 \leq p_l \leq 1, \quad \forall l.
\end{aligned} \tag{68}$$

Similar to the discussions on MAC forward engineering and jointly optimal rate reliability control, for sufficiently concave utilities, problem (68) is a convex optimization after a log change of variables $\{p_l, P^k\}$. Its solution can now be distributively carried out using either the standard primal-based penalty function approach or the dual-based Lagrangian relaxation approach, both with standard convergence properties but now producing different implications to the timescale of TCP/MAC interaction, as shown in the rest of this subsection.

First the penalty function approach is pursued. We first define $h_l(\mathbf{p}, \mathbf{x}') = \log(\sum_{s \in S(l)} e^{x'_s}) - c'_l - \log p_l -$

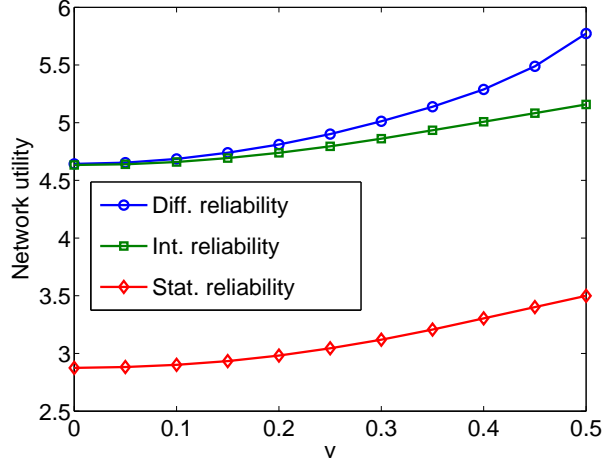


Figure 14: Comparison of the achieved network utility attained by the differentiated dynamic policy, the integrated dynamic policy, and the static policy, when θ_s are changed according to (67).

$\sum_{k \in N_{to}^I(l)} \log(1 - \sum_{m \in L_{out}(k)} p_m)$ and $w_n(\mathbf{p}) = \sum_{m \in L_{out}(n)} p_m - 1$. Then, problem (68) can be rewritten as

$$\begin{aligned}
& \text{maximize} && \sum_s U'_s(x'_s) \\
& \text{subject to} && h_l(\mathbf{p}, \mathbf{x}') \leq 0, \quad \forall l \\
& && w_n(\mathbf{p}) \leq 0, \quad \forall n \\
& && x_s'^{min} \leq x'_s \leq x_s'^{max}, \quad \forall s \\
& && 0 \leq p_l \leq 1, \quad \forall l.
\end{aligned} \tag{69}$$

Instead of solving problem (69) directly, we apply the penalty function method and consider the following problem:

$$\begin{aligned}
& \text{maximize} && V(\mathbf{p}, \mathbf{x}') \\
& \text{subject to} && x_s'^{min} \leq x'_s \leq x_s'^{max}, \quad \forall s \\
& && 0 \leq p_l \leq 1, \quad \forall l,
\end{aligned} \tag{70}$$

where $V(\mathbf{p}, \mathbf{x}') = \sum_s U'_s(x'_s) - \kappa \sum_l \max\{0, h_l(\mathbf{p}, \mathbf{x}')\} - \kappa \sum_n \max\{0, w_n(\mathbf{p})\}$ and κ is a positive constant.

Since the objective function of problem (70) is concave, problem (70) is convex optimization with simple, decoupled constraints, which can be solved by using a subgradient projection algorithm. We can easily show that

$$\frac{\partial V(\mathbf{p}, \mathbf{x}')}{\partial p_l} = \kappa \left(\frac{\epsilon_l}{p_l} - \frac{\sum_{k \in L_{from}^I(t_l)} \epsilon_k}{1 - \sum_{m \in L_{out}(t_l)} p_m} - \delta_{t_l} \right), \tag{71}$$

and

$$\frac{\partial V(\mathbf{p}, \mathbf{x}')}{\partial x'_s} = \frac{\partial U'_s(x'_s)}{\partial x'_s} - \kappa e^{x'_s} \sum_{l \in L(s)} \frac{\epsilon_l}{\sum_{k \in S(l)} e^{x'_k}}, \tag{72}$$

where

$$\epsilon_l = \begin{cases} 0, & \text{if } \sum_{n \in S(l)} e^{x'_n} \leq c_l p_l \prod_{k \in N_{to}^l(l)} (1 - \sum_{m \in L_{out}(k)} p_m) \\ 1, & \text{otherwise} \end{cases}$$

and

$$\delta_n = \begin{cases} 0, & \text{if } \sum_{m \in L_{out}(n)} p_m \leq 1 \\ 1, & \text{otherwise} \end{cases}.$$

Then, an iterative subgradient projection algorithm, with iterations indexed by t , that solves problem (70) is obtained as follows. On each logical link, transmission is decided to take place with persistence probability

$$p_l(t+1) = \left[p_l(t) + \alpha(t) \frac{\partial V(\mathbf{p}, \mathbf{x}')}{\partial p_l} \Big|_{\mathbf{p}=\mathbf{p}(t), \mathbf{x}'=\mathbf{x}'(t)} \right]_0^1, \quad \forall l, \quad (73)$$

and concurrently at each source, end-to-end rate is adjusted:

$$x'_s(t+1) = \left[x'_s(t) + \alpha(t) \frac{\partial V(\mathbf{p}, \mathbf{x}')}{\partial x'_s} \Big|_{\mathbf{p}=\mathbf{p}(t), \mathbf{x}'=\mathbf{x}'(t)} \right]_{x'_s^{min}}^{x'_s^{max}}, \quad \forall s, \quad (74)$$

where $[a]_c^b = \max\{\min\{a, b\}, c\}$.

The joint control algorithm (73,74) can be implemented as follows. Each link l (or its transmission node t_l) updates its persistence probability $p_l(t)$ using (73), and concurrently, each source updates its data rate $x_s(t)$ using (74). To calculate the subgradient in (71), each link needs information only from link k , $k \in L_{from}^l(t_l)$, *i.e.*, from links whose transmissions are interfered from the transmission of link l , and those links are in the neighborhood of link l . To calculate the subgradient in (72), each source needs information only from link l , $l \in L(s)$, *i.e.*, from links on its routing path. Hence, to perform the algorithm, each source and link need only local information through limited message passing and the algorithm can be implemented in a distributed way. In particular, note that δ_n is calculated at the transmitter node of each link to update the persistence probability of that link, and does not need to be passed among the nodes. There is no need to explicitly pass around the values of persistence probabilities, since their effects are included in $\{\epsilon_l\}$. Quantities such as $\sum_{m \in L_{out}(t_l)} p_m$ and $\sum_{k \in S(l)} \exp(x'_k)$ can be measured locally by each node and each link.

To implement a dual-based algorithm instead, we can decompose problem (68) into two problems, using a standard technique of dual decomposition also used in [10, 105]:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && \sum_{s \in S(l)} x_s \leq y_l, \forall l \\ & && x_s^{min} \leq x_s \leq x_s^{max}, \forall s, \end{aligned} \quad (75)$$

where y_l is the average data rate of link l , and

$$\begin{aligned} & \text{maximize} && \hat{U}(\mathbf{p}) \\ & \text{subject to} && \sum_{m \in L_{out}(n)} p_m \leq 1, \quad \forall n \\ & && 0 \leq p_l \leq 1, \quad \forall l, \end{aligned} \quad (76)$$

where

$$\begin{aligned} \hat{U}(\mathbf{p}) &= \max \left\{ \sum_s U_s(x_s) \mid \sum_{s \in S(l)} x_s \leq y_l(\mathbf{p}), \forall l, \right. \\ y_l(\mathbf{p}) &= c_l p_l \prod_{k \in N_{to}^I(l)} \left(1 - \sum_{m \in L_{out}(k)} p_m \right), \forall l, \\ \left. x_s^{min} \leq x_s \leq x_s^{max}, \forall s \right\}. \end{aligned}$$

For a given \mathbf{y} , problem (75) can be solved by dual decomposition and distributed subgradient method just as before.

We now solve problem (76). To this end, we first add a penalty function to the objective function of the problem as:

$$\begin{aligned} &\text{maximize } \hat{V}(\mathbf{p}) \\ &\text{subject to } 0 \leq p_l \leq 1, \quad \forall l, \end{aligned} \quad (77)$$

where $\hat{V}(\mathbf{p}) = \hat{U}(\mathbf{p}) - \kappa \max\{0, \sum_n (1 - \sum_{m \in L_{out}(n)} p_m)\}$ and κ is a positive constant. As in the previous subsection, since problem (77) is a convex problem with simple constraints, we can solve it by using a subgradient projection algorithm as:

$$p_l(t+1) = \left[p(t) + \beta(t) \frac{\partial \hat{V}(\mathbf{p})}{\partial p_l} \Big|_{\mathbf{p}=\mathbf{p}(t)} \right]_0^1, \quad \forall l, \quad (78)$$

where $\frac{\partial \hat{V}(\mathbf{p})}{\partial p_l}$ is a subgradient of $\hat{V}(\mathbf{p})$ with respect to p_l . It can be readily verified that $\frac{\partial \hat{V}(\mathbf{p})}{\partial p_l}$ is obtained as:

$$\begin{aligned} \frac{\partial \hat{V}(\mathbf{p})}{\partial p_l} &= \lambda_l^*(t) c_l \prod_{k \in N_{to}^I(l)} \left(1 - \sum_{m \in L_{out}(k)} p_m \right) \\ &\quad - \sum_{n \in L_{from}^I(t_l)} \lambda_n^*(t) c_n p_n \prod_{k \in N_{to}^I(n), k \neq t_l} \left(1 - \sum_{m \in L_{out}(k)} p_m \right) - \kappa \delta_{t_l}, \end{aligned} \quad (79)$$

where

$$\delta_n = \begin{cases} 0, & \text{if } \sum_{m \in L_{out}(n)} p_m \leq 1 \\ 1, & \text{otherwise} \end{cases}$$

and $\lambda^*(t)$ is the optimal dual solution to dual problem of (75) with $\mathbf{y} = \mathbf{y}(\mathbf{p}(t))$.

This dual-based algorithm can also be implemented in a distributed way. In each time-slot, each link determines its persistence probability by solving (78) with the help of local message passing to obtain the expression in (19). Then, within the time-slot, based on $\mathbf{y}(\mathbf{p}(t))$, each source and link use standard dual-based algorithm to solve (75) and determine the data rate of each source in the time-slot.

Unlike with the primal-based algorithm, this dual-based algorithm clearly decomposes TCP and MAC layers through the vertical decomposition (75) and (76). However, it needs an embedded loop of iterations (*i.e.*, the convergence of a distributed subgradient algorithm to solve (75) in each time-slot). Hence, it usually requires much longer convergence time than the primal-based algorithm.

Method 15. *Providing different timescales of protocol stack interactions through different decomposition methods.*

3.1.4 Case 4: Jointly optimal congestion control, routing, and scheduling

A generalized NUM is formulated in [94, 73, 23, 59, 11] where the key additional feature is the optimization over not just source rates but also scheduling of medium access and the incorporation of scheduling constraint. The standard dual decomposition decomposes it vertically into subproblems that can be solved through TCP, routing and scheduling.

Consider an ad hoc wireless network with a set N of nodes and a set L of logical links. We assume some form of power control so that each logical link l has a fixed capacity c_l when it is active. The feasible rate region at the link layer is the convex hull of the corresponding rate vectors of independent sets of the conflict graph. Let Π denote the feasible rate region. Let x_i^k be the flow rate generated at node i for destination k . We assume there is a queue for each destination k at each link (i, j) . Let f_{ij}^k be the amount of capacity of link (i, j) allocated to the flows on that link for final destination k . Consider the following generalized NUM in variables $x_s \geq 0, f_{ij}^k \geq 0$:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && x_i^k \leq \sum_{j:(i,j) \in L} f_{ij}^k - \sum_{j:(j,i) \in L} f_{ji}^k, \quad \forall i, j, k \\ & && \in \Pi \end{aligned} \quad (80)$$

where x_s is a shorthand for x_i^k . The first constraint is flow balance equation: the flow originated from node i for final destination k plus total capacity allocated for transit flows through node i for final destination k should be no more than the total capacity going out of node i for final destination k . The second constraint is on schedulability. The dual problem of (80) decomposes into the following two subproblems:

$$\max_{\lambda \geq 0} D_1(\lambda) := \max_{x_s \geq 0} \sum_s (U_s(x_s) - x_s \lambda_s) \quad (81)$$

$$\max_{\lambda \geq 0} D_2(\lambda) := \max_{f_{ij}^k \geq 0} \sum_{i,k} \lambda_i^k \sum_j (f_{ij}^k - f_{ji}^k) \quad \text{s. t.} \quad bf \in \Pi \quad (82)$$

The first subproblem is congestion control where λ_s is the congestion price locally at source $s = (i, k)$. The second subproblem corresponds to a joint problem of multi-path routing and allocation of link capacities. Thus, by dual decomposition, the flow optimization problem decomposes into separate local optimization problems of transport, network and physical layers which interact through congestion prices.

The congestion control problem (81) admits a unique maximizer $x_s(\lambda) = U_s'^{-1}(\lambda_s)$. The joint routing and

scheduling problem (82) is equivalent to

$$\max_{\boldsymbol{\lambda} \geq 0} \sum_{i,j} \sum_k \max_{f_{ij}^k \geq 0} f_{ij}^k (\lambda_i^k - \lambda_j^k) \quad \text{s. t.} \quad \boldsymbol{\lambda} \in \boldsymbol{\Pi}$$

Hence an optimal schedule is to have $f_{ij}^k = c_{ij}$ if k maximizes $(\lambda_i^k - \lambda_j^k)$ and 0 otherwise. This motivates the following joint congestion control, scheduling and routing algorithm:

1. Congestion control: the source of flow s sets its rate as $x_s(\boldsymbol{\lambda}) = U_s'^{-1}(\lambda_s)$.
2. Scheduling:
 - For each link (i, j) , find destination k^* such that $k^* \in \arg \max_k (\lambda_i^k - \lambda_j^k)$ and define $w_{ij}^* := \lambda_i^{k^*} - \lambda_j^{k^*}$.
 - Choose an $\tilde{\boldsymbol{\lambda}} \in \arg \max_{\boldsymbol{\lambda} \in \boldsymbol{\Pi}} \sum_{(i,j) \in L} w_{ij}^* f_{ij}$ such that $\tilde{\boldsymbol{\lambda}}$ is an extreme point. Those links (i, j) with $\tilde{f}_{ij} > 0$ will transmit and other links (i, j) (with $\tilde{f}_{ij} = 0$) will not.
3. Routing: over link $(i, j) \in L$ with $\tilde{f}_{ij} > 0$, send data for destination k^* at full link capacity c_{ij} .
4. Price update: each node i updates the price on the queue for destination k according to:

$$\lambda_i^k(t+1) = \left[\lambda_i^k(t) + \beta \left(x_i^k(\boldsymbol{\lambda}(t)) - \sum_{j:(i,j) \in L} f_{ij}^k(\boldsymbol{\lambda}(t)) + \sum_{j:(j,i) \in L} f_{ji}^k(\boldsymbol{\lambda}(t)) \right) \right]^+ \quad (83)$$

The w_{ij}^* values represent the maximum differential congestion price of destination k packets between nodes i and j , and was introduced in [102]. The above algorithm uses back pressure to perform optimal scheduling and hop-by-hop routing.

Method 16. *Maximum differential congestion pricing for node-based back-pressure scheduling.*

Method 17. *Architectural implication due to dual decomposition: absorb routing functionality into congestion control and scheduling.*

It is easy to show that the generalized NUM has no duality gap. It is shown in [11] that the algorithm converges statistically to a neighborhood of the optimal point using constant stepsize, in the sense that the time averages tend to the optimal values arbitrarily closely. Specifically, let the primal function (the total achieved network utility) be $P(\mathbf{x})$ and let \mathbf{x}^* be the optimum. Let $\bar{\mathbf{x}}(t) := \frac{1}{t} \sum_{\tau=0}^t \mathbf{x}(\tau)$ be the running average rates. Similarly, recall that $D(\boldsymbol{\lambda})$ is the dual objective function. Let $\boldsymbol{\lambda}^*$ be an optimal value of the dual variable and $\bar{\boldsymbol{\lambda}}(t) := \frac{1}{t} \sum_{\tau=1}^t \boldsymbol{\lambda}(\tau)$ be the running average prices.

Theorem 12. *Consider the dual of (80) and suppose the subgradient of the dual objective function is uniformly bounded. Then for any $\delta > 0$ there exist a sufficiently small stepsize β in (83) such that*

$$\begin{aligned} \liminf_{t \rightarrow \infty} P(\bar{\mathbf{x}}(t)) &\geq P(\mathbf{x}^*) - \delta \\ \limsup_{t \rightarrow \infty} D(\bar{\boldsymbol{\lambda}}(t)) &\leq D(\boldsymbol{\lambda}^*) + \delta \end{aligned}$$

Instead of using dual algorithm, as in [73, 11], where the congestion control part (Step 1 above) is static, the algorithms in [94, 23] are primal-dual where the source congestion control algorithm can be interpreted as an ascent algorithm for the primal problem.

The most difficult step in the above algorithm is scheduling. Solving it exactly requires a centralized computation which is clearly impractical in large scale networks. Various scheduling algorithms and heuristics have been proposed in the context of joint rate allocation, routing, and scheduling. The effects of imperfect scheduling on cross-layer design has recently been characterized in [59], for both the case when the number of users in the system is fixed and the case with dynamic arrivals and departures of the users.

3.2 Decomposition Methods

3.2.1 Decoupling coupled constraints

The basic idea of a decomposition is to decompose the original large problem into subproblems which are then coordinated by a master problem by means of some kind of signalling [4]. Many of the existing decomposition techniques can be classified into *primal decomposition* and *dual decomposition* methods. The former (also called partitioning of variables, decomposition by right-hand side allocation, or decomposition with respect to variables) is based on decomposing the original primal problem, whereas the latter (also termed Lagrangian relaxation of the coupling constraints or decomposition with respect to constraints) is based on decomposing the dual of the problem. As illustrated in Figure 15, primal decomposition methods have the interpretation that the master problem directly gives each subproblem an amount of resources that it can use; the role of the master problem is then to properly allocate the existing resources. In dual decomposition methods, the master problem sets the price for the resources to each subproblem which has to decide the amount of resources to be used depending on the price; the role of the master problem is then to obtain the best pricing strategy. Roughly speaking, the engineering mechanism realizing dual decomposition is *pricing feedback* while that realizing primal decomposition is *adaptive slicing*.

Note that the terminology of “primal-dual” has a number of different meanings. For example, “primal-dual interior-point method” is a class of algorithms for centralized computation of an optimum for convex optimization, “primal-dual distributed algorithm” is sometimes used to describe any algorithm that solves the primal and dual problems simultaneously, and “primal-driven penalty function approach” and “dual-driven pricing approach” have been used in the earlier discussion on congestion control and TCP/MAC joint design. In this subsection, primal and dual decompositions have a different meaning of decomposing coupling constraints through direct resource allocation and indirect pricing control, respectively.

We first illustrate how the dual decomposition approach can be applied to the basic NUM problem to produce the standard dual-based distributed algorithm. The Lagrange dual problem of (3) is readily derived. We first form the Lagrangian:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \sum_s U_s(x_s) + \sum_l \lambda_l \left(c_l - \sum_{s \in S(l)} x_s \right)$$

where $\lambda_l \geq 0$ is the Lagrange multiplier (*i.e.*, link price) associated with the linear flow constraint on link l .

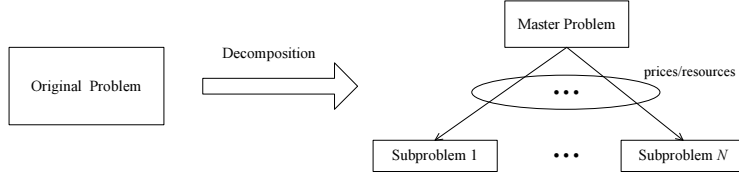


Figure 15: A schematic illustrating problem decomposition.

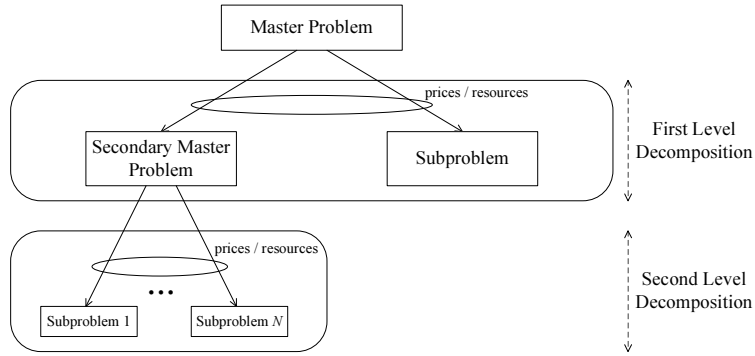


Figure 16: A schematic illustrating multi-level decomposition.

Additivity of total utility and linearity of flow constraints lead to a Lagrangian dual decomposition into individual source terms:

$$\begin{aligned}
 L(\mathbf{x}, \boldsymbol{\lambda}) &= \sum_s \left[U_s(x_s) - \left(\sum_{l \in L(s)} \lambda_l \right) x_s \right] + \sum_l c_l \lambda_l \\
 &= \sum_s L_s(x_s, q_s) + \sum_l c_l \lambda_l
 \end{aligned}$$

where $q_s = \sum_{l \in L(s)} \lambda_l$. For each source s , $L_s(x_s, q_s) = U_s(x_s) - q_s x_s$ only depends on local rate x_s and the path price q_s (*i.e.*, sum of λ_l on links used by source s).

The Lagrange dual function $D(\boldsymbol{\lambda})$ is defined as the maximized $L(\mathbf{x}, \boldsymbol{\lambda})$ over \mathbf{x} for a given $\boldsymbol{\lambda}$. This ‘net utility’ maximization obviously can be conducted distributively by the each source:

$$x_s^*(q_s) = \operatorname{argmax} [U_s(x_s) - q_s x_s], \quad \forall s. \quad (84)$$

Such Lagrangian maximizer $\mathbf{x}^*(\boldsymbol{\lambda})$ will be referred to as price-based rate allocation (for a given price $\boldsymbol{\lambda}$). The

Lagrange dual problem of (3) is

$$\begin{aligned} & \text{minimize} && D(\boldsymbol{\lambda}) = L(\mathbf{x}^*(\boldsymbol{\lambda}), \boldsymbol{\lambda}) \\ & \text{subject to} && \boldsymbol{\lambda} \succeq 0 \end{aligned} \quad (85)$$

where the optimization variable is $\boldsymbol{\lambda}$. Since $D(\boldsymbol{\lambda})$ is the pointwise supremum of a family of affine functions in $\boldsymbol{\lambda}$, it is convex and (85) is a convex minimization problem. Since $g(\boldsymbol{\lambda})$ may be non-differentiable, an iterative *subgradient* method can be used to update the dual variables $\boldsymbol{\lambda}$ to solve the dual problem (85):

$$\lambda_l(t+1) = \left[\lambda_l(t) - \beta(t) \left(c_l - \sum_{s \in S(l)} x_s(q_s(t)) \right) \right]^+, \quad \forall l \quad (86)$$

where $c_l - \sum_{s \in S(l)} x_s(q_s(t))$ is the l th component of a subgradient vector of $D(\boldsymbol{\lambda})$, t is the iteration number, and $\beta(t) > 0$ are step sizes. Certain choices of step sizes, such as $\beta(t) = \beta_0/t$, $\beta > 0$, guarantee that the sequence of dual variables $\boldsymbol{\lambda}(t)$ converges to the dual optimal $\boldsymbol{\lambda}^*$ as $t \rightarrow \infty$. It can be shown that the primal variable $\mathbf{x}^*(\boldsymbol{\lambda}(t))$ also converges to the primal optimal variable \mathbf{x}^* . For a primal problem that is a convex optimization, the convergence is towards a global optimum.

In summary, the sequence of source and link algorithms (84,86) forms a standard dual-based distributed algorithm that globally solves NUM (3) and the dual problem (85), and computes an optimal rate vector \mathbf{x}^* and optimal link price vector $\boldsymbol{\lambda}^*$. Note that no explicit signaling is needed.

The general methodology of primal and dual decompositions is now presented. A more comprehensive tutorial can be found in [75]. It turns out that primal and dual decompositions are also interchangeable through alternative representation of the optimization problem.

A primal decomposition is appropriate when the problem has a coupling variable such that, when fixed to some value, the rest of the optimization problem decouples into several subproblems. Consider, for example, the following problem over \mathbf{y} , $\{\mathbf{x}_i\}$:

$$\begin{aligned} & \text{maximize} && \sum_i f_i(\mathbf{x}_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i, \quad \forall i \quad \forall i \\ & && \mathbf{A}_i \mathbf{x}_i \leq \mathbf{y}, \quad \forall i \\ & && \mathbf{y} \in \mathcal{Y}. \end{aligned} \quad (87)$$

If variable \mathbf{y} were fixed, then the problem would decouple. This suggests separating the optimization in (87) into two levels of optimization. At the lower level, we have the subproblems, one for each i over \mathbf{x}_i , in which (87) decouples when \mathbf{y} is fixed:

$$\begin{aligned} & \text{maximize} && f_i(\mathbf{x}_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i \\ & && \mathbf{A}_i \mathbf{x}_i \leq \mathbf{y}. \end{aligned} \quad (88)$$

At the higher level, we have the master problem in charge of updating the coupling variable \mathbf{y} by solving:

$$\begin{aligned} & \text{maximize} && \sum_i f_i^*(\mathbf{y}) \\ & \text{subject to} && \mathbf{y} \in \mathcal{Y} \end{aligned} \quad (89)$$

where $f_i^*(\mathbf{y})$ is the optimal objective value of problem (88) for a given \mathbf{y} .

A subgradient for each $f_i^*(\mathbf{y})$ is given by

$$\mathbf{s}_i(\mathbf{y}) = \boldsymbol{\lambda}_i^*(\mathbf{y}), \quad (90)$$

where $\boldsymbol{\lambda}_i^*(\mathbf{y})$ is the optimal Lagrange multiplier corresponding to the constraint $\mathbf{A}_i \mathbf{x}_i \leq \mathbf{y}$ in problem (88). The global subgradient is then $\mathbf{s}(\mathbf{y}) = \sum_i \mathbf{s}_i(\mathbf{y}) = \sum_i \boldsymbol{\lambda}_i^*(\mathbf{y})$. The subproblems in (88) can be locally and independently solved with the knowledge of \mathbf{y} .

A dual decomposition is appropriate when the problem has a coupling constraint such that, when relaxed, the optimization problem decouples into several subproblems. Consider, for example, the following problem:

$$\begin{aligned} & \text{maximize} && \sum_i f_i(\mathbf{x}_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i \quad \forall i \\ & && \sum_i \mathbf{h}_i(\mathbf{x}_i) \leq \mathbf{c}. \end{aligned} \quad (91)$$

If the constraint $\sum_i \mathbf{h}_i(\mathbf{x}_i) \leq \mathbf{c}$ were absent, then the problem would decouple. This suggests relaxing the coupling constraint in (91) as

$$\begin{aligned} & \text{maximize} && \sum_i f_i(\mathbf{x}_i) - \boldsymbol{\lambda}^T (\sum_i \mathbf{h}_i(\mathbf{x}_i) - \mathbf{c}) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i \quad \forall i \end{aligned} \quad (92)$$

such that the optimization separates into two levels of optimization. At the lower level, we have the subproblems, one for each i over \mathbf{x}_i , in which (92) decouples:

$$\begin{aligned} & \text{maximize} && f_i(\mathbf{x}_i) - \boldsymbol{\lambda}^T \mathbf{h}_i(\mathbf{x}_i) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i. \end{aligned} \quad (93)$$

At the higher level, we have the master dual problem in charge of updating the dual variable $\boldsymbol{\lambda}$ by solving the dual problem:

$$\begin{aligned} & \text{minimize} && g(\boldsymbol{\lambda}) = \sum_i g_i(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \mathbf{c} \\ & \text{subject to} && \boldsymbol{\lambda} \geq \mathbf{0} \end{aligned} \quad (94)$$

where $g_i(\boldsymbol{\lambda})$ is the dual function obtained as the maximum value of the Lagrangian solved in (93) for a given $\boldsymbol{\lambda}$. This approach is in fact solving the dual problem instead of the original primal one. Hence, it will only give appropriate results if strong duality holds.

A subgradient for each $g_i(\boldsymbol{\lambda})$ is given by

$$\mathbf{s}_i(\boldsymbol{\lambda}) = -\mathbf{h}_i(\mathbf{x}_i^*(\boldsymbol{\lambda})), \quad (95)$$

where $\mathbf{x}_i^*(\boldsymbol{\lambda})$ is the optimal solution of problem (93) for a given $\boldsymbol{\lambda}$. The global subgradient is then $\mathbf{s}(\boldsymbol{\lambda}) = \sum_i \mathbf{s}_i(\boldsymbol{\lambda}) + \mathbf{c} = \mathbf{c} - \sum_i \mathbf{h}_i(\mathbf{x}_i^*(\boldsymbol{\lambda}))$. The subproblems in (93) can be locally and independently solved with knowledge of $\boldsymbol{\lambda}$.

Method 18. *Primal and dual decomposition for coupling constraints.*

Not all coupling constraints can be readily decomposed through primal or dual decompositions. For example, the feasibility set of SIR in wireless cellular network power control problems is coupled in a way with no obvious decomposability structure. A re-parametrization of the constraint set is required before dual decomposition can be applied [34].

3.2.2 Decoupling coupled objective

Examining the dual decomposition of the basic NUM reveals that the following are the underlying reasons why distributed and end-to-end algorithms solves (3):

1. Separability in objective function: The network utility is a sum of individual source utilities.
2. Additivity in constraint functions: The linear flow constraints are summing over the individual flows.
3. Interchangeability of summation index: $\sum_l \lambda_l \sum_{s \in S(l)} x_s = \sum_s x_s \sum_{l \in L(s)} \lambda_l$.
4. Zero duality gap.

For cases where property 1 fails, recent progress on coupled utility formulations has been made [95, 96]. In many communication systems, utilities are indeed coupled. An example of cooperation model can be found in networks where some nodes form a cluster and the utility obtained by each of them depends on the rate allocated to others in the same cluster (this can be interpreted as a hybrid model of selfish and non-selfish utilities). An example of competition model is in wireless power control and DSL spectrum management, where the utilities are functions of SIRs that are dependent on the transmit powers of other users.

The generalized NUM problem considered in this subsection is

$$\begin{aligned} & \text{maximize} && \sum_{k=1}^K U_k(\mathbf{x}_k, \{\mathbf{x}_l\}_{l \in \mathcal{L}(k)}) \\ & \text{subject to} && \mathbf{x}_k \in \mathcal{X}_k \quad \forall k, \\ & && \sum_{k=1}^K \mathbf{g}_k(\mathbf{x}_k) \leq \mathbf{c} \end{aligned} \tag{96}$$

where the (strictly concave) utilities U_k depend on a vector local variable \mathbf{x}_k and on variables of other utilities \mathbf{x}_l for $l \in \mathcal{L}(k)$ (i.e., coupled utilities), $\mathcal{L}(k)$ is the set of nodes coupled with the k th utility, the sets \mathcal{X}_k are arbitrary convex sets, and the coupling constraining function $\sum_k \mathbf{g}_k(\mathbf{x}_k)$ is not necessarily linear, but still convex. Note that this model has two types of coupling: coupling constraints and coupled utilities.

The key idea to tackle the coupling problem in the utilities is to introduce auxiliary variables and additional equality constraints, thus transferring the coupling in the objective function to coupling in the constraints, which can be decoupled by dual decomposition and solved by introducing additional *consistency pricing*. It is reasonable to assume that if two nodes have their individual utilities dependent on each other's local variables, then there must be some communication channels in which they can locally exchange pricing messages. It turns out that the

global link congestion price update of the standard dual-based distributed algorithm is not affected by the local consistency price updates, which can be conducted via these local communication channels among the nodes.

The first step is to introduce in problem (96) auxiliary variables \mathbf{x}_{kl} for the coupled arguments in the utility functions and additional equality constraints to enforce consistency:

$$\begin{aligned} & \text{maximize} && \sum_k U_k(\mathbf{x}_k, \{\mathbf{x}_{kl}\}_{l \in \mathcal{L}(k)}) \\ & \text{subject to} && \mathbf{x}_k \in \mathcal{X}_k \quad \forall k, \\ & && \sum_k \mathbf{g}_k(\mathbf{x}_k) \leq \mathbf{c}, \\ & && \mathbf{x}_{kl} = \mathbf{x}_l, \quad \forall k, l \in \mathcal{L}(k). \end{aligned} \quad (97)$$

Next, to obtain a distributed algorithm, we take a dual decomposition approach by relaxing all the coupling constraints in problem (97):

$$\begin{aligned} & \text{maximize} && \sum_k U_k(\mathbf{x}_k, \{\mathbf{x}_{kl}\}_{l \in \mathcal{L}(k)}) + \boldsymbol{\lambda}^T (\mathbf{c} - \sum_k \mathbf{g}_k(\mathbf{x}_k)) + \sum_{k,l \in \mathcal{L}(k)} \gamma_{kl}^T (\mathbf{x}_l - \mathbf{x}_{kl}) \\ & \text{subject to} && \mathbf{x}_k \in \mathcal{X}_k \quad \forall k, \\ & && \mathbf{x}_{kl} \in \mathcal{X}_l \quad \forall k, l \in \mathcal{L}(k) \end{aligned} \quad (98)$$

where $\boldsymbol{\lambda}$ are the *link prices* and the γ_{kl} 's are the *consistency prices*. By exploiting the *additivity* structure of the Lagrangian, the Lagrangian is separated into many subproblems where maximization is done using local variables (the k th subproblem uses *only* variables with the first subscript index k). The optimal value of (98) for a given set of γ_{kl} 's and $\boldsymbol{\lambda}$ defines the dual function $g(\{\gamma_{kl}\}, \boldsymbol{\lambda})$. The dual problem is then

$$\begin{aligned} & \text{minimize}_{\{\gamma_{kl}\}, \boldsymbol{\lambda}} && g(\{\gamma_{kl}\}, \boldsymbol{\lambda}) \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \quad (99)$$

It is worthwhile noting that (99) is equivalent to

$$\begin{aligned} & \text{minimize}_{\boldsymbol{\lambda}} && \left(\text{minimize}_{\{\gamma_{kl}\}} g(\{\gamma_{kl}\}, \boldsymbol{\lambda}) \right) \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \quad (100)$$

Problem (99) is easily solved by simultaneously updating the prices (both the link prices and the consistency prices) using a subgradient algorithm. In problem (100), however, the inner minimization is fully performed (by repeatedly updating the $\{\gamma_{kl}\}$) for each update of $\boldsymbol{\lambda}$. This latter approach implies two timescales: a fast timescale in which each cluster updates the corresponding consistency prices and a slow timescale in which the network updates the link prices; whereas the former approach has just one timescale. The formulation with two timescales is interesting from a practical perspective since consistency prices can be exchanged very quickly over local communication channels only by nodes that are coupled together.

Therefore, problem (96), where the utilities U_k are strictly concave, the sets \mathcal{X}_k are arbitrary convex sets, and the constraining functions \mathbf{g}_k are convex, can be optimally solved by the following distributed algorithm:

- the l th link updates the congestion price as

$$\boldsymbol{\lambda}(t+1) = \left[\boldsymbol{\lambda}(t) - \beta_1 \left(\mathbf{c} - \sum_k \mathbf{g}_k(\mathbf{x}_k) \right) \right]^+ \quad (101)$$

where β_1 is the stepsize, and then broadcast them to the nodes,

- the k th node, for all k , updates the consistency prices (at a faster or same timescale as the update of $\lambda(t)$) as

$$\gamma_{kl}(t+1) = \gamma_{kl}(t) - \beta_2 (\mathbf{x}_l(t) - \mathbf{x}_{kl}(t)), l \in L(k) \quad (102)$$

where β_2 is the stepsize, and then broadcast them to the coupled nodes within the cluster, and

- the k th node, for all k , locally solves the problem

$$\begin{aligned} & \underset{\mathbf{x}_k, \{\mathbf{x}_{kl}\}_r}{\text{maximize}} && U_k(\mathbf{x}_k, \{\mathbf{x}_{kl}\}_{l \in \mathcal{L}(k)}) - \lambda^T \sum_k \mathbf{g}_k(\mathbf{x}_k) + \left(\sum_{l: k \in \mathcal{L}(l)} \gamma_{lk} \right)^T \mathbf{x}_k - \sum_{l \in \mathcal{L}(k)} \gamma_{kl}^T \mathbf{x}_{kl} \\ & \text{subject to} && \mathbf{x}_k \in \mathcal{X}_k \\ & && \mathbf{x}_{kl} \in \mathcal{X}_l \quad \forall l \in \mathcal{L}(k) \end{aligned} \quad (103)$$

where $\{\mathbf{x}_{kl}\}_{l \in \mathcal{L}(k)}$ are auxiliary local variables for the k th node.

Summarizing, all the links must advertise their local variables \mathbf{x}_k (not the auxiliary ones \mathbf{x}_{kl}); then a central unit can update and signal λ to all the links; each link can update the corresponding γ_{kl} 's (with knowledge of the variables \mathbf{x}_k of the coupled links) and signal it to the coupled links; each link can update the local variable \mathbf{x}_k as well as the auxiliary ones \mathbf{x}_{kl} . The only additional price due to the coupled utilities is limited signaling between the coupled links within each cluster.

Method 19. *Using consistency pricing to decouple coupled utility objective functions.*

3.2.3 Alternative decompositions

Decomposition of a generalized NUM has significant implications to network protocol design along two directions: vertical (functional) decomposition into layers and horizontal (geographical) decomposition into distributed computation by network elements. There are many ways to decompose a given NUM formulation along both directions, each of which provides a different structure to possible solution to the problem through a different layering scheme. A systematic exploration of alternative decompositions is more than just an intellectual curiosity, it also leads to different network architectures with a wide range of choices of communication overhead, computation load, and convergence behavior, as illustrated through some case studies in Section 3.1. There is no “universally” best decomposition scheme, the choice depends on the specific application context.

An important technique that leads to alternatives of distributed architectures is to apply primal/dual decompositions recursively. The basic decompositions are repeatedly applied to a problem to obtain smaller and smaller subproblems. For example, consider the following problem over $\mathbf{y}, \{\mathbf{x}_i\}$ which includes both a coupling variable and a coupling constraint:

$$\begin{aligned} & \text{maximize} && \sum_i f_i(\mathbf{x}_i, \mathbf{y}) \\ & \text{subject to} && \mathbf{x}_i \in \mathcal{X}_i, \quad \forall i && \forall i \\ & && \sum_i \mathbf{h}_i(\mathbf{x}_i) \leq \mathbf{c} \\ & && \mathbf{A}_i \mathbf{x}_i \leq \mathbf{y}, \quad \forall i \\ & && \mathbf{y} \in \mathcal{Y}. \end{aligned} \quad (104)$$

One way to decouple this problem is by first taking a primal decomposition with respect to the coupling variable \mathbf{y} and then a dual decomposition with respect to the coupling constraint $\sum_i \mathbf{h}_i(\mathbf{x}_i) \leq \mathbf{c}$. This would produce a two-level optimization decomposition: a master primal problem, a secondary master dual problem, and the subproblems. An alternative approach would be to first take a dual decomposition and then a primal one.

Another example that shows flexibility in terms of different decompositions is the following problem with two sets of constraints:

$$\begin{aligned} & \text{maximize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \leq 0, \quad \forall i \\ & && h_i(\mathbf{x}) \leq 0, \quad \forall i. \end{aligned} \tag{105}$$

One way to deal with this problem is via the dual problem with a full relaxation of both sets of constraints to obtain the dual function $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$. At this point, instead of minimizing g directly with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, it can be minimized over only one set of Lagrange multipliers first and then over the remaining one: $\min_{\boldsymbol{\lambda}} \min_{\boldsymbol{\mu}} g(\boldsymbol{\lambda}, \boldsymbol{\mu})$. This approach corresponds to first applying a full dual decomposition and then a primal one on the dual problem. The following lemma [76] characterizes the subgradient of the master problem at the top level.

Lemma 3. *Consider the following partial minimization of the dual function*

$$g(\boldsymbol{\lambda}) = \inf_{\boldsymbol{\mu}} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{106}$$

where $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is the dual function defined as

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) \triangleq \sup_{\mathbf{x} \in \mathcal{X}} \left\{ f_0(\mathbf{x}) - \sum_i \lambda_i f_i(\mathbf{x}) - \sum_i \mu_i h_i(\mathbf{x}) \right\}. \tag{107}$$

Then, $g(\boldsymbol{\lambda})$ is convex and a subgradient, denoted by $\mathbf{s}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda})$, is given by

$$s_{\lambda_i}(\boldsymbol{\lambda}) = -f_i(\mathbf{x}^*(\boldsymbol{\lambda}, \boldsymbol{\mu}^*(\boldsymbol{\lambda}))) \tag{108}$$

where $\mathbf{x}^*(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is the value of \mathbf{x} that achieves the supremum in (107) for a given $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, and $\boldsymbol{\mu}^*(\boldsymbol{\lambda})$ is the value of $\boldsymbol{\mu}$ that achieves the infimum in (106).

Alternatively, problem (105) can be approached via the dual but with a partial relaxation of only one set of constraint, say $f_i(\mathbf{x}) \leq 0, \forall i$, obtaining the dual function $g(\boldsymbol{\lambda})$ to be minimized by the master problem. Observe now that in order to compute $g(\boldsymbol{\lambda})$ for a given $\boldsymbol{\lambda}$, the partial Lagrangian has to be maximized subject to the remaining constraints $g_i(\mathbf{x}) \leq 0, \forall i$, for which yet another relaxation can be used. This approach corresponds to first applying a partial dual decomposition and then, for the subproblem, another dual decomposition.

Note that there can be different orderings of update, including the choice of parallel (Jacobi) or sequential (Gauss-Siedel) updates [4]. When there are more than one level of decomposition, and all levels conduct some type of iterative algorithms, such as the subgradient method, convergence and stability are guaranteed if the lower level master problem is solved on a faster timescale than the higher level master problem, so that at each iteration

of a master problem all the problems at a lower level have already converged. If the updates of the different subproblems operate on similar timescales, convergence of the overall system can still be guaranteed under certain technical conditions [86, 4].

Method 20. *Partial and hierarchical decompositions for architectural alternatives of the protocol stack.*

As another example, consider the following special case of NUM in variables (\mathbf{x}, \mathbf{y}) :

$$\begin{aligned} & \text{maximize} && \sum_i U_i(x_i) \\ & \text{subject to} && f_i(x_i, y_i) = 0, \quad \forall i \\ & && y_i \in \mathcal{Y}_i, \quad \forall i \\ & && \sum_i g_i(x_i, y_i) \leq 1 \end{aligned} \tag{109}$$

where \mathbf{x} models the performance metrics that users utilities depend on and \mathbf{y} models some resources that are globally coupled and have local impacts on performance. This problem has applications in distributed waterfilling algorithms in DSL spectrum management and distributed power control algorithms in wireless cellular networks, and can be decomposed in at least seven different ways following three general approaches below. Each decomposition results in a new possibility in striking the most appropriate tradeoff between computation and communication.

1. *A primal decomposition approach.* Problem (109) decouples if the y_i 's are fixed. We can decompose the original problem into the master problem over \mathbf{y} :

$$\begin{aligned} & \text{maximize} && \sum_i \tilde{U}_i(y_i) \\ & \text{subject to} && y_i \in \mathcal{Y}_i \quad \forall i \\ & && \sum_i g_i(y_i) \leq 0 \end{aligned} \tag{110}$$

where each $\tilde{U}_i(y_i)$ is the optimal objective value of the subproblem over x_i :

$$\begin{aligned} & \text{maximize} && U_i(x_i) \\ & \text{subject to} && x_i \in \mathcal{X}_i \\ & && f_i(x_i, y_i) \leq 0. \end{aligned} \tag{111}$$

Each of the subproblems can be solved in parallel and only needs to know its local information (i.e., the local functions U_i , f_i and the local set \mathcal{X}_i) and the corresponding y_i (given by the master problem). Once each subproblem is solved, the optimal value $\tilde{U}_i(y_i)$ and (possibly) a subgradient can be communicated to the master problem. In this case, the master problem needs to communicate to each of the subproblems the available amount of resources y_i allocated.

2. *A full dual decomposition approach* with respect to all coupling constraints $f_i(x_i, y_i) \leq 0$ and $\sum_i g_i(y_i) \leq 0$. The master dual problem is to

$$\text{minimize } g(\boldsymbol{\lambda}, \boldsymbol{\gamma}) \tag{112}$$

over $\lambda, \gamma \geq 0$ where $g(\lambda, \gamma)$ is given by the sum of the optimal objective values of the following subproblems over (x_i, y_i) for each i :

$$\begin{aligned} & \text{maximize} && U_i(x_i) - \lambda_i f_i(x_i, y_i) - \gamma g_i(y_i) \\ & \text{subject to} && x_i \in \mathcal{X}_i. \end{aligned} \tag{113}$$

Each of the subproblems can be solved in parallel and only needs to know its local information and the Lagrange multipliers λ_i and γ (given by the master problem). Once each subproblem is solved, the optimal value and (possibly) a subgradient (given by $-f_i(x_i, y_i)$ and $-g_i(y_i)$) can be communicated to the master problem. In this case, the master dual problem needs to communicate to each of the subproblems the private price λ_i and the common price γ .

3. A *partial dual decomposition approach* only with respect to the global coupling constraint $\sum_i g_i(y_i) \leq 0$. The master dual problem over $\gamma \geq 0$:

$$\text{minimize} \quad g(\gamma) \tag{114}$$

where $g(\gamma)$ is given by the sum of the optimal objective values of the following subproblems for all i :

$$\begin{aligned} & \text{maximize} && U_i(x_i) - \gamma g_i(y_i) \\ & \text{subject to} && x_i \in \mathcal{X}_i \\ & && f_i(x_i, y_i) \leq 0. \end{aligned} \tag{115}$$

Each of the subproblems can be solved in parallel and only needs to know its local information and the Lagrange multiplier γ (given by the master problem). Once each subproblem is solved, the optimal value and (possibly) a subgradient, given by $-g_i(y_i)$, can be communicated to the master problem. In this case, the master dual problem needs to communicate to each of the subproblems simply the common price γ .

The amount of signalling of the three decomposition methods is summarized in Table 8. In this particular type of generalized NUM, method 1 is the worst since it requires the largest amount of signalling in both directions. Method 3 is the best, requiring a single common price from the master problem to the subproblems and a single number from each subproblem to the master problem. Method 2 is intermediate, requiring the same amount as method 3 plus an additional individual price from the master problem to each subproblem.

For a particular instance of problem (109), a numerical example for the convergence results of various horizontal decomposition possibilities is shown in Figure 17.

There are also many possibilities for vertical decomposition. For example, in [10] it is shown that the joint TCP and MAC design problem may be formulated as maximizing network utility subject to the constraint that $\mathbf{FRx} \preceq \mathbf{c}$, where \mathbf{F} is a contention matrix and \mathbf{R} is the routing matrix. Then depending whether we first group \mathbf{Rx} or \mathbf{FR} in the constraint, the Lagrange dual variables we introduce are different, corresponding to either drawing a division between TCP and MAC or not.

Implicit in all decompositions is a choice of particular *representation* of the constraints. Since every individual constraint (e.g., capacity of a link) gives rise to a corresponding Lagrange multiplier, we have the surprising consequence that the specific dual problems will be different depending on how the primal constraints are written. Even redundant constraints that may change the dual problem properties.

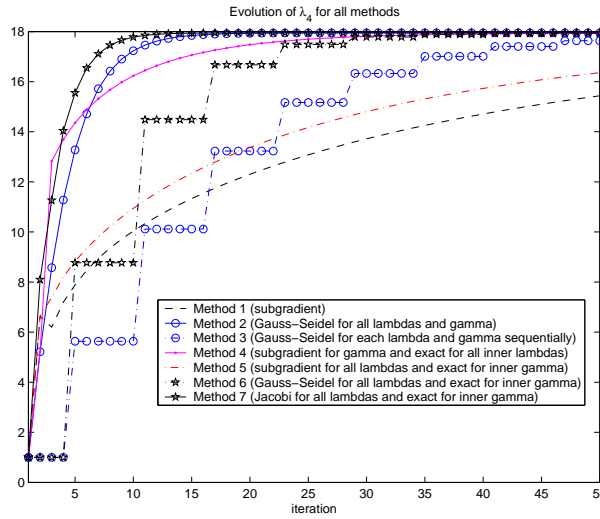


Figure 17: Different speeds of convergence for seven different alternatives of horizontal decomposition in a particular numerical example of (109). Each curve corresponds to a different decomposition structure.

3.3 Related Work

Following is a *non-exhaustive* list of some of the recent publications using “Layering as Optimization Decomposition” for vertical decomposition⁸. Some of the key results presented in these papers have been explained or briefly outlined in Section 3.1. Almost all of the following papers start with some generalized NUM formulations, and use either dual decomposition or primal penalty function approach to modularize and distribute the solution algorithm followed by proofs of optimality, stability, and fairness. The individual modules in the holistic solution range from adaptive routing and distributed matching to information-theoretic source coding and video signal processing, coupled through implicit or explicit message passing of functions of appropriate “layering price”.

- Jointly optimal congestion control and adaptive coding or power control [13, 55]
- Jointly optimal congestion and contention control [10, 42, 57, 105, 120, 121]
- Jointly optimal congestion control and scheduling [23, 1, 68]
- Jointly optimal routing and power control [74, 115]
- Jointly optimal congestion control, routing, and scheduling [11, 59]
- Jointly optimal routing, scheduling, and power control [18, 114]

⁸We apologize in advance for any references we may have missed and would appreciate any information about other citations.

- Jointly optimal routing, resource allocation, and source coding [119]
- TCP/IP interactions [106, 35] and jointly optimal congestion control and routing [33, 43, 47, 51, 77]
- Network lifetime maximization [71]

4 Future Research Directions

Despite the progress made over the last several years, there are still a variety of open issues in the area of “Layering as Optimization Decomposition”. Some of the main challenges and recent progress are outlined in this section.

4.1 Modeling and Complexity Challenges

First of all, there are semantic functionalities, such as session initiation and packet reordering, that we do not explicitly model. Border Gateway Protocol (BGP) in IP protocol and a variety of wireless ad hoc network routing protocols are yet to be fully incorporated in network utility maximization framework. Much further work also remains to be done to model utility functions in specific applications, especially inelastic, real-time applications such as VoIP and streaming media. In a more refined physical/link layer model, the option of forwarding rather than re-encoding at intermediate nodes must be considered, as well as layer 2 retransmission schemes through ARQ.

Several important modules commonly encountered in many cases of “Layering as Optimization Decomposition” still do not have simple, distributed solutions. An important example is the need for distributed and fast scheduling algorithms. Similarly, most of the cross-layer design has focused on the optimal message passing across layers and theoretical investigation on parameters such as stepsize. A systematic study on suboptimal message passing heuristics and practical guidelines in choosing algorithmic parameters would help characterize the optimal tradeoff between complexity and suboptimality gap.

4.2 Exploration of Alternative Decompositions

As discussed in Subsection 3.2.3, while different representations of a given generalized NUM problem do not change the optimum (the performance benchmark of resource allocation), they may lead to different decomposability structures, which in turn provide alternative decompositions and distributed network architectures. Further research is needed for systematically comparing vertical decomposition alternatives and generating alternative decompositions from different problem representations. While it is impossible to determine a universally “best” decomposition, certain principles such as convergence speed, robustness with respect to stochastic variations in the network, and stability under delay are some metrics that can be used to compare among alternative architectures.

4.3 Research Issues Involving “Time”

Different functions in each layer operate in timescales that may be several orders of magnitude different. For example, the application layer timescale is often determined by the user behavior, the transport layer timescale by the RTT in traversing the network, the physical layer timescale by the physics of the transmission medium. Iterative algorithms themselves also have a timescale of operation determined by their rate of convergence, which is often difficult to bound tightly.

Added to the complexity of timescale issue is that the transient behaviors of optimization models in many layers are not well understood, even without crossing the layers. In some application scenarios such as wireless network power control, if the allocated resource such as SIR drops below a certain threshold, that connection may be dropped, turning “equilibrium” into an irrelevant concept. Bounding the transient behavior of “Layering as Optimization Decomposition” algorithms is a challenging topic.

Utility functions are often modeled as functions of equilibrium rates. For applications involving real-time control or multi-media communication through resource allocations provided by solutions to generalized NUM, utility should instead be a function of latency or even the entire vector of rate allocation through the transients. How to maximize such utility functions remains an under-explored topic.

4.4 Stochastic NUM

Stochastic theory of communication networks has a long and rich history. However, many key problems in this area remain open despite decades of effort. Since late 1990s, various researchers have used deterministic fluid model to remove packet level details and microscopic queuing dynamics, followed by an optimization/game/control-theoretic approach. By assuming deterministic fluids with infinite backlog, we avoid the difficulty of coupling across links in a general queuing network and are often able to prove optimal design in network resource allocation. It is important, however, to incorporate the dynamics at session, packet, channel, and topology level to generalized NUM formulations. In such a unifying framework, service rates of queues are determined by distributed solution to NUM while parameters of NUM formulations are stochastically varying.

Stochastic models arise due to several reasons:

- Flow level (also referred to as session level, connection level, or end-user level): flows arrive and depart with finite workload, rather than holding infinite backlog and staying in the network forever.
- Packet level: packets of each flow arrive in bursts and at microscopic level goes through probabilistic marking and interact with uncontrolled flows such as those running UDP.
- Channel level: network transmission conditions is time-varying rather than fixed.
- Topology level: nodes may disappear and re-appear (based on mobility and battery consumption models) rather than fixed.

A combination of stochastic network control and optimization-based resource allocation raises challenging new questions, including stochastic stability, average case performance, outage performance, and, eventually, the distribution of attained utility as induced by the distributions of the stochastic models at various levels. Among these questions, stochastic stability is the most basic and important one: under what conditions will a certain distributed algorithm of a NUM problem remain stochastically stable, in the sense that the number of flows and the total queue length in the network remain finite?

4.4.1 Session level stochastic

Consider flow level dynamics characterized by the random arrivals and departures of flows. For each type r , flows arrive and depart according to a Poisson process with intensity λ_r , and the size of the flows to be transmitted is exponentially distributed with mean $1/\mu_r$. The traffic load is $\rho_r = \lambda_r/\mu_r$. Let N_r be the number of ongoing flows, *i.e.*, the number of type r flows in the network. It is a Markov process with transition rates:

- $N_r(t) \rightarrow N_r(t) + 1$, with rate λ_r
- $N_r(t) \rightarrow N_r(t) - 1$, with rate $\mu_r x_r(t) N_r(t)$

For the basic NUM, it is shown [5, 69, 20, 116] that the stochastic stability region is the interior of feasibility rate region formed by the fixed link capacities, *i.e.*, the following condition is sufficient to guarantee stochastic stability of the dual-based solution to the basic NUM for many utility functions:

$$\mathbf{R}\rho < \mathbf{c}. \tag{116}$$

These results assumed time scale separation. However, in many practical networks, flow level stochastic operates on a fast timescale, with the arrive-and-depart process of flows varying constantly. Hence, instantaneous convergence of the rate allocation algorithm may not hold. [60, 92] extend the above result to the case without time-scale separation assumption: [60] studies α -fair utilities using a discrete-time model and shows that there is an upper bound on the step size that would guarantee stochastic stability, and [92] shows similar results for α -fair utilities using the fluid limit model.

Another branch of the stability study is based on the fluid limit model where large capacity scaling is used. [19] has established the stability of the Markov chain by checking the stability of its corresponding fluid limits. Using this technique, [117] relaxes the assumption of Poisson arrivals, by studying a general stationary and a bursty network model respectively. It is shown that under the natural stability condition (116), many bandwidth allocation policies ensure network model stability for the stationary arrival case. Recently [50] studies a model which features flows of two types, file transfers and streaming traffic, and generalizes the congestion control problem with convex constraints. Also recently [6] correlates the utility maximization to classical queueing theory, and studies several typical utility functions and the stability condition of each. All these work implicitly assume timescale separation. Stochastic stability for any strictly concave maximization over general convex constraints without timescale separation is recently reported in [61].

Table 9 summarizes the existing results on flow level stochastic, where “t.s.s” means timescale separation; “ Θ_o ” denotes the interior of the feasible rate region.

4.4.2 Packet level stochastic

Randomness at packet level may be a result of probabilistic marking of certain AQM schemes. It can also model “mice” traffic which is not captured in the standard utility maximization model. In [2], a detailed stochastic model is presented to model N TCP Reno sources sharing a single bottleneck link with capacity Nc implementing RED. They show that as the number of sources and the link capacity both increase linearly, the queue process converges (in appropriate sense) to a deterministic process described by differential equations as usually assumed in the literature. Even though these results are proved only for a single bottleneck node, they provide a justification for the popular deterministic fluid model by suggesting that the deterministic process is the limit of a scaled stochastic process as the number of flows tends to infinity.

Similar convergence results are shown in [21, 87]: the deterministic delay differential equation model with noise replaced by its mean value is accurate asymptotically in time and the number of flows. Because such convergence is shown asymptotically in time (except in the special case of log utility [87] where it is shown for each time), the trajectory of the stochastic system does not converge to that of the deterministic system in the many-flow regime [21]. However, [21] shows that the global stability criterion for a single flow is also that for the stochastic system with many flows, thus validating parameter design in the deterministic model even when realistic systems have packet level stochastic dynamics

Stochastic stability of greedy primal-dual algorithm, a combination of utility maximization and maximum weight matching, is shown in [94] for dynamic networks where traffic sources and routers are randomly time-varying, interdependent, and limited by instantaneously available transmission and service rates.

Besides packet-level stochastic dynamics, there is also randomness at the application level. The paper [9] considers its effect on the TCP layer. It shows that the utility maximization at the TCP layer induces a utility maximization at the application layer, *i.e.*, an objective at the application layer is implemented in the TCP layer. Specifically, consider a single link with capacity Nc (bits) shared by N HTTP-like flows. Each flow alternates between *think times* and *transfer times*. During the period of a think time, a flow does not require any bandwidth from the link. Immediately after a period of think time, the source starts to transmit a random amount of data by a TCP connection. The transfer time depends on the amount of transfer and the bandwidth allocation to this flow by TCP. The number of active flows is random but at any time, the active flows share the link capacity according to TCP, *i.e.*, their throughputs maximize aggregate utility subject to capacity constraints. Assume there are a fixed number of flow types. Then it is shown in [9] that the average throughput, *i.e.*, the throughput aggregated over active flows of each type normalized by the total number of flows of that type, also solves a utility maximization problem with different utility functions as the TCP utility functions.

4.4.3 Channel level stochastic

Models in [94, 59, 11] consider random channel fluctuations. In [11], for instance, a channel is assumed to be fixed within a discrete time slot but changes randomly and independently across slots. Let $\mathbf{h}(t)$ denote the channel state in time slot t . Corresponding to the channel state \mathbf{h} , the capacity of link l is $c_l(\mathbf{h})$ when active and the feasible rate region at the link layer is $\Pi(\mathbf{h})$. We further assume that the channel state is a finite state process with identical distribution $q(\mathbf{h})$ in each time slot and define the mean feasible rate region as

$$\bar{\Pi} = \{\bar{\mathbf{r}} : \bar{\mathbf{r}} = \sum_{\mathbf{h}} q(\mathbf{h})\mathbf{r}(\mathbf{h}), \mathbf{r}(\mathbf{h}) \in \Pi(\mathbf{h})\}. \quad (117)$$

The joint TCP/routing/scheduling algorithm discussed in Section 3.1.4 can be directly applied with the schedulable region Π in Step 2 replaced by the current feasible rate region $\Pi(\mathbf{h}(t))$. It is proved in [11] that the prices $\lambda(t)$ form a stable Markov process, by appealing to the generalized NUM (80) with the rate region Π replaced by the mean rate region $\bar{\Pi}$:

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && x_i^k \leq \sum_{j:(i,j) \in L} f_{ij}^k - \sum_{j:(j,i) \in L} f_{ji}^k, \quad \forall i, j, k \\ & && \in \bar{\Pi} \end{aligned} \quad (118)$$

Moreover the primal and dual values along the trajectory converge arbitrarily close to their optimal values, with respect to (118), as the stepsize in the algorithm tends to zero.

For generalized NUM problems, [11] establishes the *stability* and *optimality* of dual algorithms under channel-level stochastic for any convex optimization where the constraint set has the following structure: a subset of the variables lie in a polytope and other variables lie in a convex set that vary according to an irreducible, finite-state Markov chain. Algorithms developed from the deterministic NUM formulation and requiring only the knowledge of current network state (such as channel state and queue-lengths), remain stochastically stable and optimal with respect to an optimization problem whose constraint is replaced by the average constraint set under the given channel variations.

4.4.4 Topology level stochastic

Topology of wireless networks can change due to mobility of nodes, sleep mode, and battery power depletion. Solving generalized NUM problems over networks with randomly varying topology remains an under-explored area with little known results or methodologies. The problem is particularly challenging when the topology level stochastic dynamics is determined by battery usage, which is in turn determined by the solution of the NUM problem itself.

4.5 Nonconvex NUM

Nonconvex problem formulation of NUM and non-zero duality gaps may arise due to a variety of reasons: integer constraints (*e.g.*, in single path routing, admission control, scheduling, algebraic coding, constellation size),

nonconcave utilities (*e.g.*, power efficiency or some empirically verified utility curves), and constraints describing nonconvex sets. A nonzero duality gap means that the standard dual-based distributed subgradient algorithm, and in general dual decomposition approaches, may lead to suboptimal and even infeasible primal solutions and instability in cross layer interactions. This very difficult problem can be tackled through a combination of well-established and more recent optimization techniques (*e.g.*, sum-of-squares programming [82] and geometric-signomial programming [12]). For example, there have been three recent approaches to solve nonconcave utility maximization over linear constraints:

1. [58] proposes a distributed, suboptimal heuristics (for sigmoidal utilities) called “self-regulating” method, to be adopted by end users with sigmoidal utilities and is shown to avoid link congestion caused by sigmoidal utilities. It attains the optimal rate allocation in the asymptotic case when the proportion of sources with nonconcave utilities vanishes.
2. [16] determines optimality conditions for the dual-based distributed algorithm to converge globally (for all nonlinear utilities). The engineering implication is that appropriate overprovisioning of link capacities will ensure global convergence of the dual-based distributed algorithm even when user utility functions are nonconcave.
3. [27] develops an efficient but centralized method to compute the global optimum (for a wide class of utilities that can be transformed into polynomial utilities), using the sum-of-squares method to show that duality gap can be closed by deploying polynomial, rather than constant, congestion pricing.

4.6 Network X-ities

Protocol design and layering architecture are not just for maximizing the efficiency of performance metrics, such as throughput, latency, distortion, but also robustness metrics, such as evolvability, scalability, and manageability. These non-performance metrics ending with “ity” are referred to as “network X-ities”. Interactions among layers introduce the risks of losing robustness against unforeseen demands arising over time or significant growth over space. Indeed, intuition suggests that under-specification may be better than over-optimization. Optimization is only as good a design approach as the choice on objective function, and performance-based utility function may not always be the right objective to maximize.

Despite the importance in practical network operations, these network X-ities remain as important yet fuzzy notions, and a quantified foundation for them is long overdue [15]. As an example, intuition suggests that “design by decomposition” enhances scalability and evolvability, but may present risks to manageability such as diagnosability and optimizability unless the decomposition structure is appropriate. As another example, we may consider continuing the shift in protocol design mentality from “forward engineering” (solve a given problem) to “reverse engineering” (discover the problem implicitly solved by a given protocol) further to “design for optimizability” (design the optimization problem to be easily solvable). Quantifying network X-ities and trading-off network X-ities with performance metrics in layered protocol stack design is a long-term, challenging direction.

5 Conclusion

We provided a survey of the recent efforts to establish “Layering as Optimization Decomposition” as a common “language” for systematic network design. “Layering as Optimization Decomposition” is a unifying framework for understanding and designing distributed control and cross-layer resource allocation in wired and wireless networks. It has been developed by various research groups over the last several years, and is now emerging to provide a mathematically rigorous and practically relevant approach to quantify the risks and opportunities of modifying existing layered network architecture. It shows that existing network protocols in layers 2, 3, and 4 can be reverse-engineered as implicitly solving some optimization-theoretic or game-theoretic problems. By distributively solving generalized NUM formulations through decomposed subproblems, we can also systematically generate layered protocol stacks. There are many alternatives for both horizontal decomposition into disparate network elements and vertical decomposition into functional modules (*i.e.*, layers). While queuing delay or buffer occupancy is often used as the “layering price”, it may sometimes lead to unstable interactions. A variety of techniques to tackle coupling and nonconvexity issues has become available, which enables developments of distributed algorithms and proofs of global optimality, respectively. Many of such techniques are becoming standard methods readily to be invoked by researchers.

The two cornerstones for the intellectual simplicity in this paper are “networks as optimizers” and “layering as decomposition”. Together they provide a promising framework to understand not just “what” works in the current layered protocol stacks, but also “why” it works, what may not work, and what alternatives network designers have.

Acknowledgements

We would like to gratefully acknowledge the collaborations and interactions on this topic with many colleagues, including Stephen Boyd, Lijun Chen, Neil Gershfeld, Prashanth Hande, Jiayue He, Sanjay Hegde, Maryam Fazel, Cheng Jin, Frank Kelly, Koushik Kar, Jang-Won Lee, Lun Li, Jiaping Liu, Zhen Liu, Asuman Ozdaglar, Daniel Palomar, Pablo Parrilo, Jennifer Rexford, Devavrat Shah, Ness Shroff, R. Srikant, Chee Wei Tan, Ao Tang, Jiantao Wang, David Wei, Jeff Wieselthier, Bartek Wydrowski, Lin Xiao, Edmund Yeh, and Junshan Zhang.

This work has been supported by NSF grants CCF-0440043, CCF-0448012, CNS-0417607, CNS-0427677, CNS-0430487, CNS-0519880, DARPA grant HR0011-06-1-0008, and Cisco grant GH072605.

References

- [1] M. Andrews, "Joint optimization of scheduling and congestion control in communication networks", *Proc. CISS*, March 2006.
- [2] F. Baccelli, D. R. McDonald and J. Reynier, "A mean-field model for multiple TCP connections through a buffer implementing RED," Tech. Rep. RR-4449, INRIA, April 2002.
- [3] D. P. Bertsekas, *Nonlinear Programming, 2nd Ed.*, Athena Scientific, 1999.
- [4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, 1989.
- [5] T. Bonald and L. Massoulié, "Impact of fairness on Internet performance", *Proc. ACM Sigmetrics*, pp. 82-91, 2001.
- [6] T. Bonald, L. Masoulié, A. Proutiere, J. Virtamo, "A Queueing Analysis of Max-Min Fairness, Proportional Fairness and Balanced Fairness", to appear in *Queueing Systems: Theory and Applications*, 2006.
- [7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
- [8] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE J. Selected Areas in Comm.*, vol. 13, no. 8, October 1995.
- [9] C. S. Chang and Z. Liu, "A bandwidth sharing theory for a large number of HTTP-like connections," *IEEE/ACM Trans. on Networking*, vol. 12, no. 5, Oct. 2004.
- [10] L. Chen, S. H. Low, and J. C. Doyle, "Joint TCP congestion control and medium access control," *Proc. IEEE INFOCOM*, Miami, FL, March 2005.
- [11] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Joint optimal congestion control, routing, and scheduling in wireless ad hoc networks," *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.
- [12] M. Chiang, "Geometric programming for communication systems," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 1, pp. 1-156, August 2005.
- [13] M. Chiang, "Balancing transport and physical layer in wireless multihop networks: Jointly optimal congestion control and power control," *IEEE J. Sel. Area Comm.*, vol. 23, no. 1, pp. 104-116, Jan. 2005.
- [14] M. Chiang and J. Bell, "Balancing supply and demand of bandwidth in wireless cellular networks: utility maximization over powers and rates," *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.
- [15] M. Chiang and M. Yang, "Towards X-ities from a topological point of view: Evolvability and scalability", *Proc. Allerton Conf.* Oct. 2004.

- [16] M. Chiang, S. Zhang, and P. Hande “Distributed rate allocation for inelastic flows”, *Proc. IEEE INFOCOM*, March 2005.
- [17] J. Y. Choi, K. Koo, D. X. Wei, J. S. Lee and S. H. Low, “Global stability of FAST TCP,” *submitted for publication*, 2006
- [18] R. L. Cruz and A. Santhanam: “Optimal routing, link Scheduling, and power control in multihop wireless networks,” *Proc. IEEE INFOCOM*, April 2003.
- [19] J. G. Dai, “On Positive Harris Recurrence of Multiclass Queueing Networks: A Unified Approach Via Fluid Limit Models”, *Annals of Applied Probability*, vol. 5, pp. 49-77, 1995.
- [20] G. de Veciana, T. J. Lee, and T. Konstantopoulos, “Stability and performance analysis of network supporting elastic services”, *IEEE/ACM Trans. on Networking*, vol. 9, no. 1, pp. 2-14, Feb. 2001.
- [21] S. Deb, S. Shakkottai, and R. Srikant, “Asymptotic behavior of Internet congestion controllers in a many-flow regime”, *Math. Operations Research*, 2005.
- [22] N. Dukkupati, M. Kobayashi, R. Zhang-Shen and N. McKeown “Processor sharing flows in the Internet,” *Proc. of the Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005.
- [23] A. Eryilmaz and R. Srikant, “Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control,” *Proc. IEEE INFOCOM*, March 2005.
- [24] FAST TCP Project netlab.caltech.edu.
- [25] FAST Copper Project www.princeton.edu/fastcopper.
- [26] Z. Fang and B. Bensaou, “Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad-hoc networks,” *Proc. IEEE INFOCOM*, vol. 2, 2004, pp. 1284–1295.
- [27] M. Fazel and M. Chiang, “Globally optimal rate allocation through nonconcave utility maximization,” *Proc. IEEE Control and Decision Conference*, Dec. 2005.
- [28] S. Floyd. “HighSpeed TCP for large congestion windows,” Internet draft `draft-floyd-tcp-highspeed-02.txt`, work in progress, <http://www.icir.org/floyd/hstcp.html>, February 2003.
- [29] S. Floyd and V. Jacobson. “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.
- [30] V. Jacobson. “Congestion avoidance and control,” *Proc. SIGCOMM’88, ACM*, August 1988. An updated version is available via <ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z>.
- [31] R. G. Gallager, *Information Theory and Reliable Communication*. Wiley, 1968.

- [32] T. G. Griffin, F. B. Shepherd, and G. Wilfong, “The stable path problem and interdomain routing,” *IEEE/ACM Trans. on Networking*, vol. 10, no. 2, pp. 232-243, April 2002.
- [33] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, “Overlay TCP for Multi-Path Routing and Congestion Control,” in *Proc. IMA Workshop on Measurement and Modeling of the Internet*, January 2004.
- [34] P. Hande, S. Rangan, and M. Chiang, “Distributed algorithms for optimal QoS assignment in cellular data networks”, *Proc. IEEE INFOCOM*, April 2006.
- [35] J. He, M. Chiang, and J. Rexford, “TCP/IP interaction based on congestion prices: Stability and optimality”, *Proc. IEEE ICC*, June 2006.
- [36] J. He, M. Chiang, and J. Rexford, “DATE: Dynamic adaptive traffic engineering,” *IEEE INFOCOM Poster Session*, April 2006.
- [37] J. Huang, Z. Li, M. Chiang, and A. K. Katsaggelos, “Pricing-based rate control and joint packet scheduling for multi-user wireless uplink video streaming”, *Proc. IEEE Packet Video Workshop*, April, 2006.
- [38] IEEE, *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, Jun. 1999, IEEE Standard 802.11.
- [39] Internet 0 Meeting Materials: cba.mit.edu/events/04.09.I0/.
- [40] C. Jin, D. X. Wei, and S. H. Low, “FAST TCP: Motivation, architecture, algorithms, and performance”, *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.
- [41] C. Jin, D. X. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh. “FAST TCP: From theory to experiments”. *IEEE Network*, 2005.
- [42] K. Kar, S. Sarkar, and L. Tassiulas, “Achieving proportional fairness using local information in Aloha networks,” *IEEE Trans. on Automatic Control*, vol. 49, no. 10, pp. 1858–1862, October 2004.
- [43] K. Kar, S. Sarkar, L. Tassiulas, “Optimization based rate control for multipath sessions,” *Proc. International Teletraffic Congress*, December 2001.
- [44] D. Katabi, M. Handley and C. Rohrs, “Congestion control for high-bandwidth delay product networks,” *Proc. ACM Sigcomm*, August 2002.
- [45] F. P. Kelly. “Fairness and stability of end-to-end congestion control,” *European Journal of Control*, vol. 9, pp. 159–176, 2003.
- [46] F. P. Kelly, A. Maulloo, and D. Tan, “Rate control for communication networks: shadow prices, proportional fairness and stability,” *Journal of Operations Research Society*, vol. 49, no. 3, pp.237-252, March 1998.

- [47] F. P. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", *Computer Communications Review*, vol. 35, no. 2, pp. 5-12, 2005.
- [48] F. P. Kelly and R. J. Williams, "Fluid model for a network operating under a fair bandwidth-sharing policy", *Annals of Applied Probability*, vol. 14, no. pp. 1055-1083, 2004.
- [49] T. Kelly. "Scalable TCP: Improving performance in highspeed wide area networks," *Computer Communication Review*, vol. 32, no. 2, April 2003.
- [50] P. Key and L. Massoulié, "Fluid models of integrated traffic and multipath routing", To appear in *Queueing Systems*, 2006.
- [51] P. Key, L. Massoulié, and D. Towsley, "Combining multipath routing and congestion control for robustness", *Proc. CISS*, March 2006.
- [52] S. Kunniyur and R. Srikant. "A time-scale decomposition approach to adaptive ECN marking," *IEEE Trans. Auto. Control*, June 2002.
- [53] S. Kunniyur and R. Srikant. "End-to-end congestion control: utility functions, random losses and ECN marks," *IEEE/ACM Trans. on Networking*, 2003.
- [54] R. J. La and V. Anantharam, "Utility-based rate control in the Internet for elastic traffic," *IEEE/ACM Trans. on Networking*, vol. 10, no. 2, pp. 272-286, April 2002.
- [55] J. W. Lee, M. Chiang, and R. A. Calderbank, "Price-based distributed algorithm for optimal rate-reliability tradeoff in network utility maximization", To appear *IEEE Journal of Selected Areas in Communications*, 2006.
- [56] J. W. Lee, M. Chiang, and R. A. Calderbank, "Utility-optimal medium access control: reverse and forward engineering," *Proc. IEEE INFOCOM*, April 2006.
- [57] J. W. Lee, M. Chiang, and R. A. Calderbank, "Jointly optimal congestion and contention control in wireless ad hoc networks", *IEEE Communication Letters*, vol. 10, no. 3, pp. 216-218, March 2006.
- [58] J. W. Lee, R. R. Mazumdar, and N. Shroff, "Non-convex optimization and rate control for multi-class services in the Internet," *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.
- [59] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," *Proc. IEEE INFOCOM*, March 2005.
- [60] X. Lin and N. B. Shroff, "On the stability region of congestion control", *Proc. Allerton Conference*, 2004.
- [61] J. Liu, M. Chiang, and H. V. Poor, "Stability of distributed dual algorithm for convex optimization", Submitted to *IEEE Conf. Control and Decision*, 2006.

- [62] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Tran. on Networking*, vol. 11, no. 4, pp. 525-536, Aug. 2003.
- [63] S. H. Low, J. Doyle, and F. Paganini, "Internet congestion control," *IEEE Control Systems Magazine*, Feb. 2002.
- [64] S. H. Low, L. L. Perterson, and L. Wang, "Understanding Vegas: a duality model," *Journal of the ACM*, vol. 49, no. 2, pp.207-235, March 2002.
- [65] S. H. Low and D. E. Lapsley, "Optimization flow control, I: basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, December 1999.
- [66] S. H. Low and R. Srikant. "A mathematical framework for designing a low-loss, low-delay internet," *Networks and Spatial Economics, special issue on "Crossovers between transportation planning and telecommunications"*, E. Altman and L. Wynter, vol. 4, pp. 75–101, March 2004.
- [67] D. G. Luenberger. *Linear and Nonlinear Programming, 2nd Ed.* Addison-Wesley Publishing Company, 1984.
- [68] P. Marbach and Y. Lu, "Active queue management and scheduling for wireless networks: The single cell case", *Proc. CISS*, March 2006.
- [69] L. Massoulié and J. W. Roberts, "Bandwidth sharing and admission control for elastic traffic", *Telecommunication Systems*, vol. 15, pp. 185-201, March 2000.
- [70] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. on Networking*, vol. 8, no. 5, pp. 556–567, October 2000.
- [71] H. Nama, M. Chiang, and N. Mandayam, "Utility lifetime tradeoff in self regulating wireless sensor networks: A cross-layer design approach", *Proc. IEEE ICC*, June 2006.
- [72] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC layer fairness in wireless packet networks," *Proc. ACM Mobicom*, Boston, USA, Aug. 2000.
- [73] M. J. Neely, E. Modiano, and C. P. Li, "Fairness and optimal stochastic control for heterogeneous networks", *Proc. IEEE Infocom*, March 2005.
- [74] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing time varying wireless networks", *IEEE J. Sel. Area Comm.*, vol. 23, no. 1, pp. 89-103, Jan. 2005.
- [75] D. Palomar and M. Chiang, "A short tutorial on decomposition methods and distributed architectures in network resouue allocation", *IEEE J. Sel. Area Comm.* vol. 24, Aug. 2006.
- [76] D. Palomar and M. Chiang, "Alternative decompositions for distributed maximization of network utility: Framework and applications", *Proc. IEEE INFOCOM*, April 2006.

- [77] F. Paganini, “Congestion control with adaptive multipath routing based on optimization”, *Proc. CISS*, March 2006.
- [78] F. Paganini, J. C. Doyle, and S. H. Low, “Scalable laws for stable network congestion control,” In *Proceedings of IEEE Conference on Decision and Control*, December 2001.
- [79] F. Paganini, Z. Wang, J. C. Doyle, and S. H. Low, “Congestion control for high performance, stability and fairness in general networks,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, February 2005.
- [80] A. Papachristodoulou, “Global stability analysis of a TCP/AQM protocol for arbitrary networks with delay,” *Proc. IEEE Conference on Decision and Control*, December 2004.
- [81] A. Papachristodoulou, L. Li and J. C. Doyle, “Methodological frameworks for large-scale network analysis and design,” *Computer Communication Review*, July 2004.
- [82] P. A. Parrilo, “Semidefinite programming relaxations for semialgebraic problems,” *Mathematical Programming Series B*, vol. 96, no. 2, pp. 293-320, 2003.
- [83] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, “SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB User’s Guide”, June 2004.
- [84] P. Ranjan, R. J. La, and E. H. Abed, “Characterization of global stability conditions with an arbitrary communication delay,” *IEEE/ACM Trans. on Networking*, April 2006.
- [85] R. T. Rockafellar, *Network Flows and Monotropic Programming*. Athena Scientific, 1998.
- [86] R. T. Rockafellar, “Saddle-points and convex analysis,” in *Differential Games and Related Topics*, H. W. Kuhn and G. P. Szego, Eds. North-Holland, 1971.
- [87] S. Shakkottai and R. Srikant, “Mean FDE models for Internet congestion control under a many-flows regime”, *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 1050-1072, June 2004.
- [88] S. Shenker, “Fundamental design issues for the future Internet,” *IEEE J. Sel. Area Comm.*, vol. 13, no. 7, pp. 1176-1188, Sept. 1995.
- [89] R. N. Shorten and D. J. Leith, “H-TCP: TCP for high-speed and long-distance networks,” *Proc. PFLDnet*, Argonne, 2004.
- [90] A. Simsek, A. Ozdaglar and D. Acemoglu, “Generalized Poincare-Hopf Theorem for compact nonsmooth regions,” submitted for publication, 2005.
- [91] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.
- [92] R. Srikant, “On the positive recurrence of a Markov chain describing file arrivals and departures in a congestion-controlled network”, *IEEE Computer Communications Workshop*, 2004.

- [93] W. Stevens. *TCP/IP illustrated: the protocols*, volume 1. Addison–Wesley, 1999. 15th printing.
- [94] A. L. Stolyar, “Maximizing queueing network utility subject to stability: greedy primal-dual algorithm”, *Queueing Systems*, vol. 50, no. 4, pp. 401-457, 2005.
- [95] C. W. Tan, D. Palomar, and M. Chiang, “Distributed optimization of coupled systems with applications to network utility maximization”, *Proc. IEEE ICASSP*, May 2006.
- [96] C. W. Tan, D. Palomar, and M. Chiang, “Solving nonconvex power control problems in wireless networks: Low SIR regime and distributed algorithms,” *Proc. IEEE Globecom*, Nov. 2005.
- [97] A. Tang, J. W. Lee, J. Huang, M. Chiang, and R. A. Calderbank, “Reverse engineering MAC”, *Proc. IEEE WiOpt*, April 2006.
- [98] A. Tang, J. Wang and S. H. Low, “Counter-intuitive throughput behavior in networks under end-to-end control,” *IEEE/ACM Transactions on Networking*, April 2006.
- [99] A. Tang, J. Wang, S. Hegde, and S. H. Low, “Equilibrium and fairness of networks shared by TCP Reno and FAST,” *Telecommunications Systems – special issue on High Speed Transport Protocols*, 2005.
- [100] A. Tang, J. Wang, and S. H. Low, “Understanding CHOCkE: throughput and spatial characteristics,” *IEEE/ACM Trans. on Networking*, August 2004.
- [101] A. Tang, J. Wang, S. H. Low, and M. Chiang, “Network equilibrium of heterogeneous congestion control protocols,” *Proc. IEEE INFOCOM*, Miami, FL, March 2005.
- [102] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Trans. on Automatic Control*, 36:1936–1948, December 1992.
- [103] G. Vinnicombe, “On the stability of end-to-end congestion control for the Internet,” Technical report, Cambridge University, CUED/F-INFENG/TR.398, December 2000.
- [104] G. Vinnicombe, “On the stability of networks operating TCP-like congestion control,” *Proc. of IFAC World Congress*, 2002.
- [105] X. Wang and K. Kar, “Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access,” *Proc. ACM Mobihoc*, 2005.
- [106] J. Wang, L. Li, S. H. Low and J. C. Doyle, “Cross-layer Optimization in TCP/IP Networks,” *IEEE/ACM Trans. on Networking*, 13(3):582-268, June 2005
- [107] J. Wang, D. X. Wei, J-Y. Choi and S. H. Low, “Modeling and stability of FAST TCP,” *IMA Volumes in Mathematics and its Applications*, Volume 143: Wireless Communications, Prathima Agrawal, Matthew Andrews, Philip J. Fleming, George Yin, and Lisa Zhang (Eds.) Springer Science, 2006.

- [108] J. Wang, D. X. Wei and S. H. Low, "Modeling and stability of FAST TCP," *Proc. of IEEE INFOCOM*, March 2005.
- [109] D. X. Wei, C. Jin, S. H. Low and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, and performance," To appear *IEEE/ACM Trans. on Networking*, 2007.
- [110] J.T. Wen and M. Arcak, "A unifying passivity framework for network flow control," *IEEE Transaction on Automatic Control*, vol. 49, no. 2, Feb. 2004.
- [111] B. Wydrowski and M. Zukerman, "MaxNet: a congestion control architecture for maxmin fairness," *IEEE Communications Letters*, 6:512–514, November 2002.
- [112] B. Wydrowski, L. L. H. Andrew and M. Zukerman, "MaxNet: a congestion control architecture for scalable networks," *IEEE Communications Letters*, 7:511–513, October 2003.
- [113] L. Xu, K. Harfoush and I. Rhee, "Binary increase congestion control for fast long distance networks," *IEEE Proc. of INFOCOM*, March 2004.
- [114] Y. Xi and E. Yeh, "Node-based distributed optimal control of wireless networks", *Proc. CISS*, March 2006.
- [115] L. Xiao, M. Johansson and S. Boyd, "Joint routing and resource allocation via dual decomposition," *IEEE Trans. Comm.*, vol. 52, no. 7, pp. 1136-1144, July 2004.
- [116] H. Ye, "Stability of data networks under optimization-based bandwidth allocation", *IEEE Trans. Auto. Control*, vo.. 48, no. 7, pp. 1238-1242, July 2003.
- [117] H. Ye, J. Qu, and X. Yuan, "Stability of Data Networks: Starionary and Bursty Models", *Operations Research*, vol. 53, pp. 107-125, 2005.
- [118] H. Yäiche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing of elastic connections in broadband networks: theory and algorithms," *IEEE/ACM Trans. on Networking*, vol. 8, no. 5, pp. 667–678, October 2000.
- [119] W. Yu and J. Yuan, "Joint source coding, routing, and resource allocation for wireless sensor networks," *Proc. IEEE ICC*, May 2005.
- [120] C. Yuen and P. Marbach, "Price-based rate control in random access networks," *IEEE/ACM Trans. on Networking*, vol. 13, no. 5, pp. 1027–1040, December 2005.
- [121] J. Zhang and D. Zheng, "A stochastic primal-dual algorithm for joint flow control and MAC design in multihop wireless networks", *Proc. CISS*, 2006.

<i>Symbol</i>	<i>Meaning</i>
t	Time index
α	Parameter of α fair utility functions
$\beta(t)$	Step size at time t
β_0	Constant step size
θ	Weighted combination's weights
L, l	Set of links and index for links
N, s	Set of sources and index for sources
c_l	Capacity on link l
$L(s)$	Set of links used by source s
$S(l)$	Set of sources traversing link l
\mathbf{R}	Routing matrix
λ_l	Congestion price on link l
x_s	Source s rate
y_l	Link l load
U_s	Source s utility function
W_s	Source s window size
T_s	Round trip time (total delay) for source s
d_s	Total propagation delay for source s
q_s	Total queuing delay for source s
$b_l(t)$	Queue length at time t
r_l	Average queue length
ω_l	RED weighting factor
$\rho_1, \rho_2, M_l, \bar{b}_l, \underline{b}_l$	RED parameters
α_s	TCP Vegas parameter
γ_s, γ_l	Gain factors in FAST and AVQ
F_s, \mathbf{H}_l, G_l	General congestion control functions
f_s	Function mapping from source rate to total queuing delay
κ_s	Primal congestion control algorithm's gain
g_l	Function mapping from link load (and link capacity) to congestion price
$L()$	Lagrangian
$D()$	Dual function
\tilde{c}_l	Virtual capacity on link l
$(\mathbf{z}, \mathbf{u}, \mathbf{v})$	Passivity system
$V(\mathbf{z})$	Lyapunov or storage function
N^j	Number of sources using type j congestion control protocol
m_l^j	Price mapping function on link l for type j protocol

Table 4: Summary of main notation for Section 2.1.

<i>Symbol</i>	<i>Meaning</i>
W_l	Backoff window size
CW_l	Current backoff window size
W_l^{max}	Maximum backoff window size
W_l^{min}	Minimum backoff window size
p_l	Persistence probability of link l
q_l	Conditional persistence probability of link l
p_l^{max}	Maximum persistence probability
p_l^{min}	Minimum persistence probability
T_l	The event that link l transmits
C_l	The event that the transmission by link l is collided
β_l	Backoff multiplier
$S(\mathbf{p})$	Probability of successful transmission
$F(\mathbf{p})$	Probability of failed transmission
$R(p_l)$	Reward for successful transmission
$C(p_l)$	Cost of failed transmission
K	Maximum number of contending links
r_l	Receiving node of link l
t_l	Transmitting node of link l
$L_{out}(n)$	Set of egress links from node n
$L_{in}(n)$	Set of ingress links to node n
$L_{from}^I(n)$	Set of links interfered by node n
$N_{to}^I(l)$	Set of nodes whose transmission interfere link l
P^n	Persistence probability of node n
C_{CL}	Capacity of clique

Table 5: Summary of main notation for Section 2.2.

<i>Symbol</i>	<i>Meaning</i>
\mathbf{H}^s, \mathbf{H}	Physical connectivity matrices
\mathbf{W}^s, \mathbf{W}	Load balancing matrices
\mathcal{W}_n	Set of load balancing matrices with single-path routing
\mathcal{W}_m	Set of load balancing matrices with multi-path routing
\mathcal{R}_n	Set of routing matrices with single-path routing
\mathcal{R}_m	Set of routing matrices with multi-path routing
r^s	Set of routes available to source s
V_{np}	Optimized primal value of TCP/IP NUM with single-path routing
V_{nd}	Optimized dual value of TCP/IP NUM with single-path routing
V_{mp}	Optimized primal value of TCP/IP NUM with multi-path routing
V_{md}	Optimized dual value of TCP/IP NUM with multi-path routing
P_j	Transmit powers
m_j	Message generated by node j
SIR_j	Signal to Interference Ratio
G_{ij}	Channel gain from transmitter on link j to receiver on link i
ρ_s	Reliability at source s
$\rho^s(t)$	Offered reliability at source s at time t
p^s	Probability of decoding error at source s
$p_{l,s}$	Probability of decoding error on link l for source s
$t_{l,s}$	Transmission rate on link l for source s
$r_{l,s}$	Code rate on link l for source s
C_l^{max}	Maximum capacity on link l
M	Codeword length
E_l	Error function mapping code rate to decoding error probability
μ_s	Signal quality price generated by source s
$c_{l,s}$	Capacity on link l allocated to source s
x'_s	Log transformed x_s
U'_s	Utility function of log transformed x_s
v	Parameter weighting preference between rate and reliability
κ	Penalty function weight
ϵ, δ	Message passing for joint congestion and contention control
Π	Schedulability constraint
x_i^k	Rate for source destination pair (i, k)
f_{ij}^k	Flow on link (i, j) for destination k
w_{ij}	Weighting for maximum weight matching
\tilde{U}	Utility function of subproblems

Table 6: Summary of main notation for Section 3.

<i>Symbol</i>	<i>Meaning</i>
N_r	Number of active flows of type r
λ_r	The Poission arrival rate of flow of type r
μ_r	The mean of exponential file size in flow of type r
ρ_r	The load of flow of type r
\mathbf{h}	Channel state

Table 7: Summary of main notation for Section 4.

	From master problem to ith subproblem	From ith subproblem to master problem
Method 1 (primal decomp.)	amount of resources y_i	subgrad. of $\tilde{U}_i(y_i)$
Method 2 (full dual decomp.)	prices λ_i and γ	subgrads. $-f_i(x_i, y_i)$ and $-g_i(y_i)$
Method 3 (partial dual decomp.)	price γ	subgrad. $-g_i(y_i)$

Table 8: Summary of signalling between the master problem and the subproblems of the three considered decompositions.

Table 9: Summary of results on flow level stochastic NUM.

utility type	constraint type	t.s.s	stability region	ref.
concave U	affine	yes	Θ_o	[116]
α -fair	affine	no	Θ_o	[60, 92]
proportional	general convex	yes	Θ_o	[6]
max-min	general convex	yes	Θ_o	[6]
balanced fair	general convex	yes	Θ_o	[6]
concave U	general convex	yes	other condition	[50]
concave U	general convex	no	Θ_o	[61]