Mobile Netw Appl
DOI 10.1007/s11036-008-0110-0

Linear Representation of Network Traffic

With Special Application to Wireless Workload Generation

Stefan Karpinski · Elizabeth M. Belding · Kevin C. Almeroth · John R. Gilbert

Abstract We propose a representation of wireless workload patterns as large, sparse matrices and provide a method for stochastically generating experimental workloads from a given matrix. The essential property of the algebraic representation is that the summation of vectors naturally yields a faithful description of the aggregate behavior of the corresponding flows. This deceptively simple property allows us to express many common concepts from traffic modeling succinctly in terms of a few linear transformations. The algebraic representation has many benefits: (1) it makes the meaning of generally understood but vague concepts, such as "uniform behavior," mathematically precise and unambiguous; (2) it allows us to see clearly, through the lens of linear algebra, the implications of common modeling assumptions; (3) the implementation of traffic models becomes unprecedentedly simple and orthogonal, requiring only a handful of high-level matrix operations, which can be freely composed; (4) the vast body of algebraic theory and highly optimized numerical software may immediately be applied to traffic modeling. We use the paired differential simulation methodology introduced by the authors in previous work to experimentally demonstrate that the general matrix model accurately reproduces realistic network performance (Karpinski et al. 2007a, b). We use the same experimental methodology to explore the implications of various assumptions and simplifications that are commonly made in traffic modeling.

Keywords traffic modeling · traffic analysis · workload generation · wireless networks · wireless simulation · linear algebra · nonnegative matrix factorization

S. Karpinski (✉) · E. M. Belding · K. C. Almeroth · J. R. Gilbert
Department of Computer Science,
University of California, Santa Barbara, USA
e-mail: sgk@cs.ucsb.edu

E. M. Belding
e-mail: ebelding@cs.ucsb.edu

K. C. Almeroth
e-mail: almeroth@cs.ucsb.edu

J. R. Gilbert
e-mail: gilbert@cs.ucsb.edu

1 Introduction

More than 20 years of research in analysis and modeling of network traffic have yielded tremendous advances in our understanding of the complex behaviors that emerge from the interaction of millions of humans and computers on the Internet and in local-area networks (LANs). Both traffic analysis and realistic workload generation, however, remain active areas of research with many unanswered questions. One of the major challenges of modeling whole-network traffic is the interdependence of packet, flow,[1] and node behaviors.

[1]We use the common definition of a *flow* as a sequence of packets sharing the same "5-tuple": IP protocol type, source and destination nodes, and TCP/UDP port numbers.

Three types of simplifying assumptions are commonly applied in both analysis and generation:

1. **Uniform:** that traffic properties are uniformly distributed. For example, assuming that all nodes in a network have equal probability of being the source or destination of each flow. Another common example is the assumption that all packet sizes and inter-packet intervals are the same; this is commonly known as the "constant bit-rate" (CBR) model.
2. **Marginal:** that traffic properties share a single, common *marginal*[2] distribution across the entire network, and individual values are drawn independently and unconditionally from this shared distribution. An example is assuming that a single distribution of packet sizes describes all flows in a network, as compared to equipping each flow with its own specific distribution of sizes.
3. **Conditional:** that traffic behaves marginally within equivalence classes defined by certain conditions. The assumption, for example, that all flows emanating from a common source node share the same packet size distribution is a conditional model, conditioned on the source nodes.

We would be arguing against a straw man if we implied that researchers who apply these assumptions actually believe them to be true. They clearly are not: different nodes initiate and receive drastically different numbers of packets and flows; flows have drastically different packet size distributions. Simply considering intuitive examples of common behavior illustrates this amply: BitTorrent users *vs.* occasional email checkers; downloading a large file *vs.* typing in an SSH session.

If no one actually believes that assumptions of uniformity and marginality accurately describe network behavior, then why are these assumptions so common? There are two reasons. The first is that these assumptions drastically simplify both traffic analysis and workload generation, making them tractable problems. The second is that there are no generally accepted alternatives for representing and simplifying traffic patterns that allow effective analysis and generation without assuming uniformity or marginality.

[2]The statistical term "marginal" refers to the margins of actuarial tables formerly used for statistical computations. The rows and columns of the table represent possible values of two properties. Each entry in a table contains a count of the number of events falling into the joint category for that row and column. The margins contain sums of the rows and columns, thus giving the unconditional distributions of each property.

Fundamentally, uniform, marginal and conditional models are applied so that network traffic behavior can be represented using a manageable collection of empirical or analytical statistical distributions. Reduction in representation size and complexity makes analysis and generation feasible. From this perspective, three questions immediately present themselves:

1. Do these assumptions distort the metrics we are trying to analyze, replicate, and predict?
2. Can we represent "unreduced" network behavior so that analysis and generation remain tractable?
3. Are there better ways to simplify and reduce the representation of network behavior?

This paper answers all three questions. We begin, however by answering the second problem and using our solution to address the other two questions.

To represent unreduced network behavior practically, we propose the concept of a *linear representation* of network traffic. A linear representation of traffic maps each flow to a vector, such that the following linearity condition is satisfied:

> The sum of the representations of two flows is a vector that represents the aggregate behavior of the two flows.

The complete behavior of a network is expressed as a matrix of such vectors, having one row per flow. Linear representations express full network behavior, without imposing assumptions of uniformity or marginality, yet still allow effective traffic analysis and workload generation. We present a specific linear representation, called the general matrix model (GMM), and demonstrate experimentally that this model generates wireless workload that accurately reproduces the performance characteristics of real wireless network traffic.

To address the other two questions, we begin with a simple observation. In linear representations of traffic, uniform, marginal and conditional assumptions are assertions of equality between the rows and columns of the traffic matrix. Moreover, real traffic matrices can be transformed into uniform, marginal and conditional forms through very simple matrix multiplications. These transformations, however, are extremely destructive: the matrices are degenerate with very low rank, so multiplication destroys almost all information in the original traffic matrix. We demonstrate experimentally that this loss of information is not benign: uniform and marginal traffic models severely distort crucial network performance metrics.

Once traffic is represented linearly, alternatives to assuming uniformity or marginality become apparent.

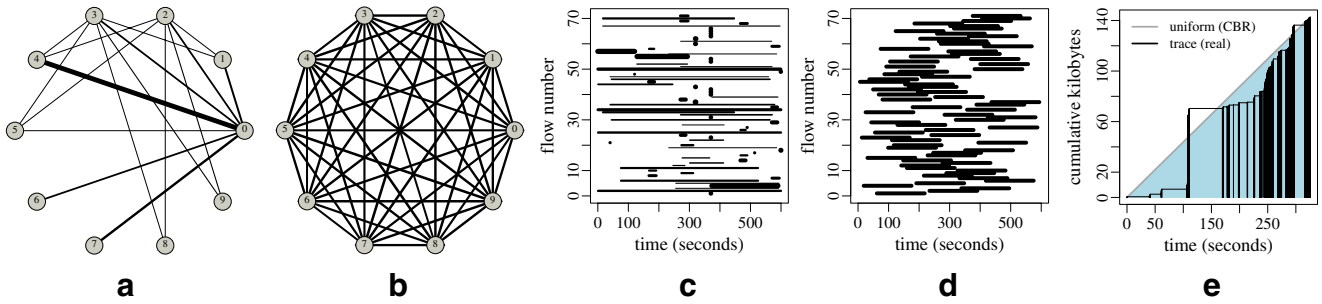


Figure 1 Examples illustrating real versus uniform behaviors at the node, flow and packet levels. **a**, **b** Example node behaviors. The width of each line is proportional to the logarithm of the number of flows between the nodes (zero is the Internet gateway). Uniform and trace flow behavior examples are plotted in **c** and **d**. The time axis indicates when flows start and end; the width of each flow line is proportional to the logarithm of its data rate. **e** Uniform (i.e. CBR) packet behavior with the trace of an actual flow. In the uniform model, the cumulative data sent increases smoothly over time, whereas in the actual packet trace, the transmissions are variable both in size and in inter-packet interval, leading to a "lumpy" cumulative data plot

We argue that matrix factorization, which is nondestructive and does not assume uniformity or marginality, is far superior to multiplication for this problem. To demonstrate this approach, we use nonnegative matrix factorization (NMF) [7] to find a low-dimensional approximation of the packet size and inter-packet interval matrices of a real traffic trace. This factorization extracts a small set of "basic behaviors" from the traffic matrix and simultaneously explains each flow's observed behavior as a mixture of these basic behaviors. Although the factorization is computed without examining port numbers, basic behaviors nevertheless correspond to intuitive expectations for protocols. We find basic behaviors corresponding to typical network activities, including file transfer, typing via SSH, web surfing, ping traffic, peer-to-peer, and others.

The rest of the paper is organized as follows. In Section 2 we discuss motivation and related work. The notion of linear representation is presented in Section 3 and the general matrix model is presented in Section 4. Our experimental and analytical methodology for evaluating traffic models is presented in Section 5, while the results of our experiments are explained and analyzed in Section 6. The ramifications of these results are discussed in Section 7. Finally, in Section 8, we conclude with a summary of this research and its impact on the future of wireless networking.

2 Motivation and related work

The interaction of network users and applications with the lower layers of the networking stack is characterized by where, when, how much, and to whom data is transmitted. The joint pattern of traffic generation and mobility through time and space completely determines the effect of network usage on the lower levels of the stack. This is due to the data-agnostic nature of the protocol stack: by design, IP networks treat all data in the same manner [2].[3] The credibility of conclusions derived from simulation or experimental deployment depends crucially on our confidence that the models used to generate traffic in experiments are sufficiently realistic. Figure 1 illustrates through examples how drastically different network traffic appears under assumptions of uniformity at different levels of behavior. These examples are take from a wireless trace recorded at an IETF meeting; details of the trace are given in Section 5.2. Marginality assumptions are not as visually striking, but as we will demonstrate, they distort the performance characteristics of networks almost as much, and sometimes more, than uniform models.

While our work is not inherently specific to wireless networks, since the modeling techniques developed are equally applicable to traffic in wired networks, the results are of special importance to wireless research. Realistic traffic models have a drastic impact on experimental results in wireless networks [5, 6]. This is primarily because wireless networks are locally resource constrained. Compare this with the typical situation in wired LANs, where gigabit Ethernet and modern routers can easily handle all but the most intense levels of traffic. If a wired network operator expects larger volumes of traffic on a conditional basis, the solution

[3]This is violated by some quality of service (QoS) schemes. However, we can simply add QoS metadata—such as traffic classes or urgency flags—to our models of user behavior and the rest of our arguments remain valid. The network is still disinterested in the exact content of the data being transported; only the QoS metadata is relevant.

is to upgrade the hardware, not to change the network protocols. In wireless networks, on the other hand, the physical medium cannot be upgraded; protocols must be improved to make better usage of the medium. The ability to perform experiments with realistic workload is a vital part of the ongoing process of improving wireless protocols. Without realistic workload models for experiments, performance comparisons may be misleading or even completely wrong [6].

Paxson and Floyd observed [8] that the interaction between endpoint behavior and the network conditions is inherently *closed-loop* in the sense that the emergent behavior is governed by a closed feedback loop. In the case of most non-TCP applications, this feedback does not occur: an open-loop model suffices. Hernández-Campos's dissertation [3] addresses in great depth how to model the feedback between the network and typical TCP applications. The author shows how to abstract an application's behavior from a TCP trace and replay it in a manner that accurately reflects how typical applications react in response to different network conditions. This research has subsequently been turned into a TCP flow generation tool called TMIX [12]. The acronym is somewhat misleading: TMIX does not provide models for application behaviors or for mixtures of applications in networks. Rather, it provides the ability to reproduce accurate closed-loop dynamics for a single TCP flow *given* an application behavior model as input. Our research complements this perfectly: we provide the means to analyze traffic and generate realistic whole-network application mixtures which can then serve as the necessary inputs to TMIX.

The two most prominent general traffic generation frameworks are Harpoon and D-ITG. Harpoon [11] uses a traffic trace for self-training, and can subsequently generate synthetic traffic with certain statistical properties based on the original trace. The properties reproduced are the empirical distributions of the following: "file size, inter-connection time, source and destination IP ranges, [and] number of active sessions." Harpoon exemplifies the uniform and marginal modeling approaches. Source and destination nodes are treated marginally: empirically derived marginal frequencies of sources and destinations are used to choose endpoints independently for each flow. Flow sizes (i.e. file sizes) are likewise treated marginally: each flow's total byte count is sampled from a marginal distribution of flow sizes. UDP flows are modeled as being constant bit-rate; TCP flows are all treated as file transfers, making them effectively uniform with bit-rate determined by TCP dynamics. Inter-connection times are sampled from a single empirically derived marginal distribution. While Harpoon may be sufficiently realistic for the intended purpose of generating Internet backbone traffic,[4] for generating local-area traffic in wireless experiments, it is not. Previous research has shown that uniform packet behavior models used in Harpoon drastically distort vital performance metrics at all levels of the protocol stack [6]. This paper shows, moreover, that marginal models, like those used in Harpoon, also severely misrepresent performance metrics in wireless networks.

Like Harpoon, D-ITG [1] exemplifies uniform and marginal modeling assumptions. However, unlike Harpoon, D-ITG does not derive models from real traffic. Packet behavior can be modeled using standard statistical distributions ("constant, uniform, normal, Cauchy, Pareto, exponential, etc."). Guidance is not provided, however, for choosing among these distributions, choosing parameter values, or selecting a mixture of distributions across a network. Flow and node behaviors are not specified at all, but rather left to the user's discretion. The focus of this project is primarily on the engineering challenge of generating very large volumes of synthetic traffic and injecting it into networks in a distributed fashion. While D-ITG provides the mechanism for generating large volumes of traffic, it does not provide any real guidance as to what traffic to generate.

Hernández-Campos et al. [4] have defined the state of the art in parametric modeling of wireless traffic patterns. They use rigorous statistical methods to fit parametric models to a number of properties of wireless traffic. Flow arrivals fit a non-stationary Poisson model; flow inter-arrival times fit a Lognormal distribution; the number of flows per session fits a BiPareto model; flow sizes also fit a BiPareto model. The statistical analysis fitting these parametric models is thorough and convincing. The models, however, are marginal: a single statistical model applies to all flows and nodes; packet behavior is not considered. Each node has a different number of flows assigned to it, but otherwise nodes are identical. Moreover, all flows, regardless of source node, are statistically identical: flow sizes are drawn from a single marginal distribution, and nothing else distinguishes them. Yet, intuitively it is clear that different nodes in real networks have drastically different tendencies for flow size, flow rates, and packet behavior. Such interdependent behaviors cannot be captured

[4] Even this is unclear: generated traffic is only compared with trace traffic using the very same metrics that are explicitly specified as part of the model. Unsurprisingly, the distributions of sampled metrics closely match the distributions those metrics were sampled from. Realism is not double-checked using other statistical metrics or actual performance metrics.

using marginal modeling. In this paper, we demonstrate that this inability is not harmless: treating all flows and nodes statistically identically severely distorts crucial wireless performance metrics in experiments. We also provide sorely needed methods of representing and analyzing traffic patterns without making uniform or marginal assumptions about network behavior.

This work uses the technique known as *paired differential simulation* to evaluate the realism of modified or synthetically generated wireless workloads. This technique was proposed and subsequently refined by Karpinski et al. [5, 6]. They introduce the notion of *sufficient realism*, and propose paired differential simulation as an experimental methodology based on the standard technique of pairing control and test subjects. We describe the technique in Section 5. In the first paper [5], they explore various ways of modifying trace traffic patterns, as a form of sensitivity analysis to determine which aspects of network behavior are essential and must be preserved in realistic models, and which aspects can be discarded as inessential. Of particular interest is their conclusion that detailed time-series packet behavior for flows is inessential: it suffices, for each flow, to have accurate unconditional distributions of packet sizes and inter-packet intervals; sufficiently realistic flow behavior can be reconstructed from these distributions by repeated independent sampling. In the second paper [6], Karpinski et al. explore the impact of a broad variety of uniformity assumptions at the packet, flow and node levels of behavior. They find that all uniformity assumptions are detrimental, inducing significant and often extreme misrepresentation of a broad variety of wireless performance metrics at every level of the protocol stack. Moreover, they show, using OLSR and AODV, that uniform traffic models can completely invert the relative performance of network protocols.

3 Linear representation of traffic

The traffic workload pattern in a network is a collection of IP packets between hosts in the network sent at certain times. It is standard in network analysis, however, to aggregate sequences of packets sharing the same "5-tuple": IP protocol, source and destination nodes, and source and destination TCP/UDP port numbers. Such a sequence of packets is called a *flow*. The traffic pattern of an entire network is simply the collected behaviors of all flows occurring in the network.

The behavior of a flow as it affects the network is characterized by the following properties: its IP protocol type (TCP, UDP, ICMP, etc.); its source and destination nodes; its start time; and the specific sizes of packets and intervals between their transmission. We begin our exposition of traffic representation by formalizing these properties mathematically as sets:

- types: $\mathsf{type} \in \mathcal{Y} = \{1, \ldots, 255\} \subset \mathbb{N}$
- nodes: $\mathsf{src}, \mathsf{dst} \in \mathcal{N} = \{1, \ldots, n\} \subset \mathbb{N}$
- times: $\mathsf{start} \in \mathcal{T} = \mathbb{R}$
- sizes: $\mathsf{size} \in \mathcal{Z} = \{0, \ldots, \mathrm{MTU}\} \subset \mathbb{N}$
- intervals: $\mathsf{ival} \in \mathcal{V} = \mathbb{R}^+ \cup \{\infty\}$.

There are 255 possible IP types (0 is reserved for IPv6). The number n is simply the total number of nodes in the network being modeled. MTU is the maximum transfer unit of the network (typically around 1,500 bytes). We formalize the packet behavior of a flow as a pair of infinite sequences of packet sizes and inter-packet intervals:

$$\mathsf{sizes} = \langle \mathsf{size}_1, \mathsf{size}_2, \ldots \rangle \in \mathcal{Z}^{\mathbb{N}} \tag{1}$$

$$\mathsf{ivals} = \langle \mathsf{ival}_1, \mathsf{ival}_2, \ldots \rangle \in \mathcal{V}^{\mathbb{N}}. \tag{2}$$

Since flows contain only a finite number of packets, there exists a number of packets, $\mathsf{pkts} \in \mathbb{N}$, for each flow. Since flows are finite, we require that the sequences sizes and ivals satisfy the following requirements:

$$\forall\, k\colon \quad \mathsf{size}_k > 0 \iff k \leq \mathsf{pkts} \tag{3}$$

$$\forall\, k\colon \quad \mathsf{ival}_k < \infty \iff k < \mathsf{pkts}. \tag{4}$$

Actual packet sizes are required to be positive because we are modeling application layer data transfer: empty data transmission requests are non-events. The send time of the kth packet can be expressed as

$$\mathsf{time}_k = \mathsf{start} + \sum_{i=1}^{k-1} \mathsf{ival}_i. \tag{5}$$

Equation 4 implies that this will be finite if and only if $k \leq \mathsf{pkts}$. Finally, we formalize the space of flows:

$$\mathcal{F} = \mathcal{Y} \times \mathcal{N}^2 \times \mathcal{T} \times \mathbb{N} \times \mathcal{Z}^{\mathbb{N}} \times \mathcal{V}^{\mathbb{N}}. \tag{6}$$

We write a generic element of $\mathcal{F}$ as

$$\mathsf{flow} = \langle \mathsf{type}, \mathsf{src}, \mathsf{dst}, \mathsf{start}, \mathsf{pkts}, \mathsf{sizes}, \mathsf{ivals} \rangle \in \mathcal{F}. \tag{7}$$

Let $2^{\mathcal{F}}$ denote the power set of $\mathcal{F}$, and $\mathcal{F}^* \subseteq 2^{\mathcal{F}}$ the set of all finite sets of flows:

$$\mathcal{F}^* = \left\{ X \in 2^{\mathcal{F}} : |X| < \infty \right\}. \tag{8}$$

A traffic sample is simply an element of $\mathcal{F}^*$. We will refer to elements of $\mathcal{F}^*$ interchangeably as "traffic samples" or "traffic instances."

Having chosen a mathematical formalization of network traffic, we may now define the notion of a linear representation of network traffic.

Definition 1 A *linear representation* of network traffic is a pair, $\langle V, \psi \rangle$, where V is a vector space and ψ is a function, $\psi : \mathcal{F}^* \to V$, such that

$$\forall\ T \in \mathcal{F}^* \colon \psi(T) = \sum_{\mathsf{flow} \in T} \psi(\mathsf{flow}). \tag{9}$$

In plain words, every finite collection of flows is represented by a vector that is the sum of the elements representing the individual flows in the collection. The careful reader will notice that *any* function $\mathcal{F} \to V$ uniquely determines a linear representation: Eq. 9 uniquely extends the function to all of $\mathcal{F}^*$. Representing flows by arbitrary vectors, however, is not particularly useful or interesting. We focus instead on meaningful representations, where collections of flows are represented by vectors that naturally describe the aggregate behavior in some manner. In the next section, we clarify these concepts through examples.

3.1 Examples of linear representations

Our first example is the packets-bytes-duration (PBD) representation. The representation is $\mathsf{pbd} : \mathcal{F}^* \to \mathbb{R}^3$, mapping $\mathsf{flow} \mapsto \langle p, b, d \rangle \in \mathbb{R}^3$, where

$$p = \mathsf{pkts}, \quad b = \sum_{i=1}^{p} \mathsf{size}_i, \quad d = \sum_{i=1}^{p-1} \mathsf{ival}_i. \tag{10}$$

This representation encapsulates the information necessary to reproduce a CBR version of each flow: packets have payloads of $\lfloor b/p \rfloor$ bytes, and are sent every d/p seconds.[5] Given two flows with behavior vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, their sum as vectors,

$$\mathbf{x}_1 + \mathbf{x}_2 = \langle p_1 + p_2, b_1 + b_2, d_1 + d_2 \rangle, \tag{11}$$

provides an appropriate description of the aggregate behavior of the two flows together. Similarly, their average, $\frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_2)$, has the average number of packets, bytes, and duration, which is precisely what the "average behavior" of the two flows should intuitively be. In contrast, a "packets-payload-interval" representation, expressing the CBR parameters directly, does not have this property: the sum of average payloads is not the overall average of payloads; likewise for intervals. In the PBD representation, the linearity property leads to natural descriptions of aggregate behavior through vector addition, whereas in the packets-payload-interval representation, vector addition is "unnatural."

The PBD representation is highly lossy: it discards most of the information detailing the behavior of each flow. At the other extreme, we can give a "perfect" representation of flows, that allows us to completely reconstruct each flow from its representation. With care, the sum of representations of flows can still naturally describe their aggregate behavior. Let the representation be $\Phi : \mathcal{F}^* \to \mathcal{X}$, where

$$\mathcal{X} = \mathbb{R}^{255} \times \mathbb{R}^n \times \mathbb{R}^n \times L^2[\mathbb{R}] \times L^2[\mathbb{R}]. \tag{12}$$

Here $L^2[\mathbb{R}]$ is the vector space of real functions, $\mathbb{R}^{\mathbb{R}} = \{f : \mathbb{R} \to \mathbb{R}\}$, with the usual norm [10]. We define

$$\Phi(\mathsf{flow}) = \langle \mathsf{e}_{\mathsf{type}}, \mathsf{e}_{\mathsf{src}}, \mathsf{e}_{\mathsf{dst}}, f, g \rangle, \tag{13}$$

where e_k is the kth standard basis vector, and $f, g \in L^2[\mathbb{R}]$ are functions defined by

$$f(t) = \sum_{k=1}^{\mathsf{pkts}} \begin{cases} 1 & \text{if } t \geq \mathsf{time}_k \\ 0 & \text{if } t < \mathsf{time}_k, \end{cases} \tag{14}$$

$$g(t) = \sum_{k=1}^{\mathsf{pkts}} \begin{cases} \mathsf{size}_k & \text{if } t \geq \mathsf{time}_k \\ 0 & \text{if } t < \mathsf{time}_k. \end{cases} \tag{15}$$

Thus, f and g are both integer-valued, monotonically increasing step functions, giving the cumulative number of packets and total bytes sent, respectively, at time t. From the representation, $\Phi(\mathsf{flow})$, of a single flow, we can readily recover its type, source, destination, and the transmission times and sizes of packets.

Why is Φ natural in the same sense that the PBD representation is natural? Suppose we have a traffic sample, $T \in \mathcal{F}^*$. The first component of $\Phi(T)$ is a 255-dimensional vector, whose coordinates are a histogram of the number of times each flow type occurs in T. Similarly, the second and third components of $\Phi(T)$ are n-dimensional histograms of the number of times each node occurs as source and destination. The last components are step functions, like f and g. Vector addition in $L^2[\mathbb{R}]$ dictates that these functions give the total number of cumulative packets and bytes sent, respectively, over all flows in T. Thus, summation of vectors under the representation Φ yields a natural and useful description of the aggregate behavior of flows.

Consider, for contrast, if we had represented flow type simply by $\mathsf{type} \in \mathbb{N}$ instead of $\mathsf{e}_{\mathsf{type}} \in \mathbb{R}^{255}$. The natural numbers containing type can readily be embedded into $\mathbb{R}^1$, which is a vector space. However, the addition of vectors is highly unnatural. The sum of protocol

[5] There are many schemes to correct the discrepancy in bytes when p does not divide b evenly. The simplest is to ignore it; we leave more complex schemes to the reader.

numbers is meaningless: two TCP flows plus five ICMP flows does not equal a single UDP flow, despite the fact that $2 \cdot 6 + 5 \cdot 1 = 17$. The summation of vectors in the representation Φ, on the other hand is $5\,\mathsf{e}_1 + 2\,\mathsf{e}_6$, indicating the frequency with which each type occurred.

Both of the example representations discussed here can be applied in practice. The PBD representation naturally allows us to compute aggregate and average CBR behaviors of collections of flows. With a suitable machine encoding of step-functions, Φ can be implemented and used in practice. It provides complete information about individual flows and highly detailed (though not complete) information about collections of flows. Representations that preserve even more information about traffic instances can readily be constructed. We turn now, however, in a different direction, and introduce a specific linear representation that strikes a practical balance in fidelity between pbd and Φ.

4 The general matrix model

The PBD representation of flows uses three dimensions to encode the behavior of each flow, while Φ uses $255 + 2n$ dimensions, plus the continuum of dimensions of two Banach spaces (i.e. the two copies of $L^2[\mathbb{R}]$). In this section, we define a representation, ϕ, that splits the difference and uses a large but finite number of dimensions to represent each flow. Once the vector representation of each flow is defined, we may construct matrices of representation vectors, with a row for each flow in a traffic sample. We call the matrix representation of network traffic the *general matrix model* (GMM). Having expressed network behavior in terms of matrices, we can very simply and efficiently compute a broad variety of network behavior properties using standard matrix operations. Furthermore, we can express uniformity and marginality assumptions about behavior as equality constraints between the rows and columns of the traffic matrix. Moreover, we can transform unconstrained traffic instances into uniform, marginal, or conditional approximations via right or left multiplications by easily computed matrices.

The general strategy for defining ϕ is to divide the descriptive properties of each flow into bins (one per value for properties that are discrete already and require no compression), and map values in each bin to a standard basis element of $\mathbb{R}^d$, where d is the number of bins. In this scheme, representation vectors are effectively histograms, with coordinate values in each dimension counting items falling into the corresponding bin. Table 1 lists the properties that are used to define ϕ, together with the quantization and dequantization functions. Each quantization function takes a possible property value as input and maps it to an index value in $\{1, \ldots, d\}$, where d is the number of bins, i.e. dimension, for the property in question.

The histogram-vector approach produces some representations of properties that are not immediately intuitive, and may seem to use an excessive number of representation dimensions. However, this approach automatically satisfies the "naturalness" criterion for linear representations. That is, the representation, $\phi(T)$, naturally describes the aggregate behavior of flows in T. Perhaps even more importantly, arbitrary linear combinations of vectors can be meaningfully interpreted as describing some realizable, hypothetical traffic instance. In Section 4.5 we will describe how to generate flows from arbitrary composite representation vectors.

4.1 Model specification

The quantization functions specified in Table 1 allow us to define the representation $\phi : \mathcal{F}^* \to \mathbb{R}^N$:

$$\phi(\mathsf{flow}) = \mathbf{f} = \langle \mathbf{t}, \mathbf{s}, \mathbf{d}, \mathbf{a}, \mathbf{b}, \mathbf{z}, \mathbf{v} \rangle, \tag{16}$$

Table 1 Quantization and dequantization functions for properties of flows

Property (prop)	Domain	Quantization ($\mathcal{Q}_{\mathsf{prop}}$)	Dim.	Dequantization ($\mathcal{Q}^{-1}_{\mathsf{prop}}$)
IP type	$\mathsf{type} \in \{1, \ldots, 255\}$	$y \mapsto y$	255	$x \mapsto \lceil x \rceil$
Source node	$\mathsf{src} \in \{1, \ldots, n\}$	$s \mapsto s$	n	$x \mapsto \lceil x \rceil$
Destination node	$\mathsf{dst} \in \{1, \ldots, n\}$	$d \mapsto d$	n	$x \mapsto \lceil x \rceil$
Start time	$\mathsf{start} \in [0, t_{\max}]$	$t \mapsto \max\left\{1, \left\lceil d_a \left(\frac{t}{t_{\max}}\right) \right\rceil\right\}$	d_a	$x \mapsto (t_{\max})\left(\frac{x}{d_a}\right)$
Bytes (flow size)	$\mathsf{bytes} \in \{1, \ldots, b_{\max}\}$	$b \mapsto 1 + \left\lceil (d_b - 1)\left(\frac{b-1}{b_{\max}-1}\right)^{\beta} \right\rceil$	d_b	$x \mapsto \max\left\{1, \left\lceil (b_{\max} - 1)\left(\frac{x}{d_b}\right)^{\frac{1}{\beta}} \right\rceil\right\}$
Packet size	$\mathsf{size} \in \{1, \ldots, \mathrm{MTU}\}$	$z \mapsto z$	d_z	$x \mapsto \lceil x \rceil$
Inter-packet interval	$\mathsf{ival} \in [0, v_{\max}]$	$v \mapsto \max\left\{1, \left\lceil d_v \left(\frac{v - v_{\min}}{v_{\max} - v_{\min}}\right)^{\gamma} \right\rceil\right\}$	d_v	$x \mapsto v_{\min} + (v_{\max} - v_{\min})\left(\frac{x}{d_v}\right)^{\frac{1}{\gamma}}$

The domain specifies the set of possible input values to the quantization function. Quantized values are elements of $\{1, \ldots, d\}$, where d is the dimension specified. The dequantization function maps a real value, $x \in [0, d]$, back into the original domain.

where $N = 255 + 2n + d_a + d_b + d_z + d_v$ and vectors are

$$\mathbf{t} = \mathsf{e}_k \quad k = \mathcal{Q}_{\mathsf{type}}(\mathsf{type})$$

$$\mathbf{s} = \mathsf{e}_k \quad k = \mathcal{Q}_{\mathsf{src}}(\mathsf{src})$$

$$\mathbf{d} = \mathsf{e}_k \quad k = \mathcal{Q}_{\mathsf{dst}}(\mathsf{dst})$$

$$\mathbf{a} = \mathsf{e}_k \quad k = \mathcal{Q}_{\mathsf{start}}(\mathsf{start})$$

$$\mathbf{b} = \mathsf{e}_k \quad k = \mathcal{Q}_{\mathsf{bytes}}(\mathsf{bytes})$$

$$\mathbf{z} = \sum \mathsf{e}_k \quad k \in \{\mathcal{Q}_{\mathsf{size}}(\mathsf{size}_i) : i \leq \mathsf{pkts}\}$$

$$\mathbf{v} = \sum \mathsf{e}_k \quad k \in \{\mathcal{Q}_{\mathsf{ival}}(\mathsf{ival}_i) : i < \mathsf{pkts}\}.$$

The functions $\mathcal{Q}_{\mathsf{prop}}$ are the respective quantization functions for each property. If there are f flows in a traffic instance, then the collective behavior can be described with seven matrices—one for each property; the rows of each matrix are the behavior vectors of the flows, indexed in some chosen order. We use the following notations for the matrices and indexed flow vectors:

- types: $\mathbf{T} = (\mathbf{t}_i) = (t_{ij}) \in \mathbb{R}^{f \times 255}$
- sources: $\mathbf{S} = (\mathbf{s}_i) = (s_{ij}) \in \mathbb{R}^{f \times n}$
- destinations: $\mathbf{D} = (\mathbf{d}_i) = (d_{ij}) \in \mathbb{R}^{f \times n}$
- start times: $\mathbf{A} = (\mathbf{a}_i) = (a_{ij}) \in \mathbb{R}^{f \times d_a}$
- sizes in bytes: $\mathbf{B} = (\mathbf{b}_i) = (b_{ij}) \in \mathbb{R}^{f \times d_b}$
- packet sizes: $\mathbf{Z} = (\mathbf{z}_i) = (z_{ij}) \in \mathbb{R}^{f \times d_z}$
- inter-packet intervals: $\mathbf{V} = (\mathbf{v}_i) = (v_{ij}) \in \mathbb{R}^{f \times d_v}$.

The total behavior matrix is the horizontal concatenation of these component matrices:

$$\mathbf{F} = \left[\mathbf{T}\,\mathbf{S}\,\mathbf{D}\,\mathbf{A}\,\mathbf{B}\,\mathbf{Z}\,\mathbf{V}\right] \in \mathbb{R}^{f \times N}. \tag{17}$$

We call this total representation the *general matrix model*. While it is by no means the only possible representation of traffic in terms of matrices, as we demonstrate in the following sections, many common traffic modeling assumptions can be expressed succinctly and precisely in terms of transformations or conditions of the GMM and its components. Furthermore, realistic workload can be generated from matrix instances.

4.2 Choosing constants

Before continuing with less mundane topics, we must briefly address our choices of constants appearing in Table 1. The externally predetermined constants are: $d_z = \text{MTU} = 1,500$, determined by the wireless network medium; $b_{\max} \approx 109$ MB, determined by the size of the largest flow in our trace data; $t_{\max} = v_{\max} = 600$, determined by our experimental methodology which slices traces into 10-min scenarios; and $v_{\min} = 10^{-6}$, determined by the time resolution of our trace. For lack of a better choice, we have simply chosen to use the same number of quantization bins as packet size for the unfixed dimension constants: $d_a = d_b = d_v = 1,500$. Arguments can certainly be made for other choices, but this choice appears to work adequately in practice.

Two other unspecified constants appear in Table 1: β and γ. These parameters are non-linear scaling exponents that allows the quantization to smoothly shift from fine-grained resolution at the low end of the spectrum to coarse-grained resolution at the high end. We have chosen $\beta = 1/2.7$ and $\gamma = 1/3$ because these values yield desirable dynamic ranges: from single bytes at the lower end to megabytes at the high end for flow sizes; from microseconds to seconds for intervals.

4.3 Computation of common properties

Having defined a specific matrix representation of network behavior, we turn now to deriving useful expressions for commonly used network properties in terms of these algebraic building blocks.

4.3.1 Packet size and inter-packet interval statistics

From the matrices $\mathbf{Z}$ and $\mathbf{V}$ we can compute a variety of statistics about packet sizes and inter-packet intervals. The row vectors in the matrices may represent individual flows or composites of many flows. The computations require knowing the values that each matrix column represents, whether size or interval. The vectors of representative values for $\mathbf{Z}$ and $\mathbf{V}$ respectively are:

$$\boldsymbol{\sigma} = \langle 1, 2, 3, \ldots, d_z \rangle \tag{18}$$

$$\boldsymbol{\tau} = \left\langle \mathcal{Q}_{\mathsf{ival}}^{-1}\left(k - \tfrac{1}{2}\right)\right\rangle_{k=1}^{d_v}. \tag{19}$$

The packet count, total byte and total duration vectors are computed as

$$\boldsymbol{\Sigma}_{\mathsf{pkts}} = \mathbf{Z}\mathbf{1} = \mathbf{V}\mathbf{1} \in \mathbb{R}^f \tag{20}$$

$$\boldsymbol{\Sigma}_{\mathsf{bytes}} = \mathbf{Z}\boldsymbol{\sigma}^T \in \mathbb{R}^f \tag{21}$$

$$\boldsymbol{\Sigma}_{\mathsf{ivals}} = \mathbf{V}\boldsymbol{\tau}^T \in \mathbb{R}^f. \tag{22}$$

Here $\mathbf{1}$ is a row vector of all ones.[6] Packet count and total bytes values are exact; the total durations are only approximate because information is irretrievably lost in the quantization process. Using these vectors of totals,

[6] Through out this paper, $\mathbf{1}$ denotes a matrix of all ones; dimensions of the matrix are inferred by context. Where necessary, we disambiguate the dimensions by explicitly specifying the dimensions of the final matrix product, as we do here.

we may readily compute the average packet sizes and inter-packet intervals:

$$\bar{\mathbf{z}} = \boldsymbol{\Sigma}_{\text{bytes}} ./ \boldsymbol{\Sigma}_{\text{pkts}} = \mathbf{Z}\boldsymbol{\sigma}^T ./ \mathbf{Z1} \tag{23}$$

$$\bar{\mathbf{v}} = \boldsymbol{\Sigma}_{\text{ivals}} ./ \boldsymbol{\Sigma}_{\text{pkts}} = \mathbf{V}\boldsymbol{\tau}^T ./ \mathbf{V1}. \tag{24}$$

The symbol ./ here represents element-wise division of vectors, as it does in Matlab.

4.3.2 Inter-node communication graphs

The total number of source flows for each node is given by the vector $\mathbf{S}^T\mathbf{1} \in \mathbb{R}^n$, and the total number of destination flows is given by $\mathbf{D}^T\mathbf{1} \in \mathbb{R}^n$. The number of flows between each pair of nodes in the network is expressed by the matrix product $\mathbf{S}^T\mathbf{D} \in \mathbb{R}^{n\times n}$. That is $(\mathbf{S}^T\mathbf{D})_{ij}$ is the number of flows from node i to node j. Interpreting this sparse matrix as a weighted graph immediately yields the source-destination flow communication graph. Figure 1 gives two visual examples of such graphs. Similar graphs counting other quantities can readily be computed by inserting the appropriate weighting matrix. For example, we may compute the inter-node packet transmission graph:

$$\mathbf{S}^T \text{diag}(\boldsymbol{\Sigma}_{\text{pkts}})\mathbf{D} \in \mathbb{R}^{n\times n}. \tag{25}$$

Here $\boldsymbol{\Sigma}_{\text{pkts}}$ is the byte vector defined in Eq. 20, and the "diag" operator gives the square, diagonal matrix with the given vector of values on its diagonal. Similar graphs for total bytes and total duration of flows may be computed using Σ_{bytes} or Σ_{ival} in place of Σ_{pkts}.

4.4 Expression of common modeling concepts

The simplifying assumptions made by most traffic models can be classified into three categories. We use the terms *uniform*, *marginal*, and *conditional* to describe these classes of assumptions. They are all assumptions about the "sameness" of aspects of network behaviors; they differ, however, in whether they make stipulations about uniformity within each flow, across all flows, or within groups of flows. In the following sections, we explore how various modeling concepts can be expressed in the framework of the general matrix model. Models are expressed as transformations that convert arbitrary traffic matrix instances into similar ones that satisfy the model's assumptions. These transformations later allow us to evaluate the effect of each model's simplifications on the realism of generated workloads.

4.4.1 Uniform modeling

The concept of uniformity, as used here, entails that certain properties of each flow are assumed to be statistically or deterministically uniform. Examples include: assuming that each flow's packets have the same size and inter-packet interval (i.e. the CBR flow model); assuming that each node is equally likely to be the source of a given flow; assuming that all start times for a flow are equally likely.

Since uniform behaviors entail homogeneity of properties with respect to each flow individually, a trace matrix can be transformed to a matrix satisfying the model's assumptions via right matrix multiplication. Right multiplication mixes the elements within each row, and if the mixing weights are uniform, the result is uniform behavior. For example, to make the roles of the nodes with respect to each flow statistically identical, we transform **S** and **D**:

$$\mathbf{S}' = \text{rs}(\mathbf{S1}) = \tfrac{1}{n}\mathbf{S1} \in \mathbb{R}^{f\times n} \tag{26}$$

$$\mathbf{D}' = \text{rs}(\mathbf{D1}) = \tfrac{1}{n}\mathbf{D1} \in \mathbb{R}^{f\times n}. \tag{27}$$

The "rs" operator scales each row to sum to unity, thus making the entire matrix row-stochastic; in this case, this is equivalent to scaling by $1/n$. These transformations of **S** and **D** make all the values in each flow's row identically the average of the original n values. As another example, the constant bit-rate (CBR) flow model, in its strictest sense, is an intra-flow model, which stipulates that all the packets in each flow behave identically, having identical sizes and intervals; thus each flow's bit-rate is constant, as the name implies. The strict CBR model can be easily expressed with the packets-bytes-duration (PBD) model used as an example in Section 3. The pair of matrices [**Z V**] is transformed to PBD form using the vectors defined in Section 4.3.1: [**Z V**] $\Gamma \in \mathbb{R}^{f\times 3}$, where

$$\Gamma = \begin{pmatrix} \mathbf{1} & \boldsymbol{\sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\tau} \end{pmatrix} \in \mathbb{R}^{(d_v+d_z)\times 3}. \tag{28}$$

Figure 1 graphically depicts examples of uniform behaviors as compared to realistic behaviors from traces.

4.4.2 Marginal modeling

Marginality is an orthogonal concept to uniformity: the distribution of some property is identical *across* all flows. For example, if we determine a distribution of flow sizes across the entire network and assume that this distribution applies independent of other flow properties, this is a marginal model for flow size. Marginal behavior is effected by left multiplication by a

matrix with uniform columns. This specific example is generated by left-uniformizing the flow size matrix: $\mathbf{B}' = (1/f)\,\mathbf{1B}$. The traffic models proposed by Hernández-Campos et al. are all marginal: they propose certain network-wide parametric distributions for start times, flow sizes and the number of flows per node. These models do not account for non-independence between these and other flow properties. We discuss this limitation and its implications when we analyze our experimental results.

4.4.3 Conditional modeling

The final form of homogenization we consider is conditionality. This is a restricted form of marginality: we may assume that marginal distributions apply to all the flows in certain groupings. For example, we can provide a distribution of flow sizes on a per-source basis. That is, each source node can have its *own* distribution, and the sizes of its flows are drawn from that distribution, independent of their other properties. This is simply accomplished by left-multiplication by a non-uniform matrix. For example, to make flow behavior uniform per source, consider the "source-conditioning" matrix: $\mathbf{S}^T\mathbf{S} \in \mathbb{R}^{n\times n}$. This matrix has $(\mathbf{S}^T\mathbf{S})_{ij} = 1$ if and only if flows i and j have the same source node. To condition flow behavior by source, we left-multiply by the stochasticized source-conditioning matrix:

$$\mathbf{F}' = \mathrm{rs}(\mathbf{S}^T\mathbf{S})\mathbf{F}. \tag{29}$$

In this case, rows do not have the same sums, so the "rs" operator is not equivalent to scalar multiplication. Per destination uniformity can be achieved similarly, replacing $\mathbf{S}$ with $\mathbf{D}$.

The GMM has the benefit of making the conditioning of behaviors completely explicit. It also allows us to easily define intermediate choices. If we wish, for example, to make the behavior of each flow a weighted average of the composite behaviors of its source and destination nodes, we can express this easily:

$$\mathbf{F}' = ((1-\alpha)\,\mathrm{rs}(\mathbf{S}^T\mathbf{S}) + (\alpha)\,\mathrm{rs}(\mathbf{D}^T\mathbf{D}))\mathbf{F}. \tag{30}$$

The parameter α allows us to smoothly vary the weighting of the source and destination behaviors. Source-only behavior is given by $\alpha = 0$; destination-only by $\alpha = 1$; to weight source and destination equally, take $\alpha = 1/2$.

4.5 Traffic generation

Given an instance of the general matrix model, how do we produce an actual workload from it? Each row of the model is used to randomly generate the behavior of a single flow. To do this, we use the "dequantization" functions listed in Table 1. The dequantization procedure works as follows. Let $\mathbf{f}$ be a row of the traffic matrix, and let prop be a property. Write $\mathbf{f}_{\mathsf{prop}}$ for the corresponding component vector of $\mathbf{f}$. Chose an index value $k \in \{1, \ldots, d_{\mathsf{prop}}\}$, randomly, with weight given by the coefficients of $\mathbf{f}_{\mathsf{prop}}$. Choose $u \in [0, 1]$ uniformly at random. Then $x = k - u \in [0, d]$, with probability of lying in the interval $[i-1, i]$ proportional to the coefficient of $\mathbf{e}_i$ in $\mathbf{f}_{\mathsf{prop}}$. The dequantization function for prop, namely $\mathcal{Q}^{-1}_{\mathsf{prop}}$, will map the value x back into the appropriate range of property values for prop.

The procedure above allows us to sample random values for each property in accordance with the behavioral description implied by $\mathbf{f}$. This allows us to stochastically generate a sequence of packets in accordance with a row from an arbitrary GMM matrix instance using the following procedure:

1. Sample random source and destination node pairs. Since a flow must have distinct source and destination nodes, the sampling of endpoints must be done jointly, avoiding collisions. The simplest technique is the rejection method: if the same node is chosen for both source and destination, reject that pair and begin the selection process again.
2. Next, randomly sample the number of bytes.
3. The packet size and inter-packet interval vectors, $\mathbf{z}$ and $\mathbf{v}$, allow us to compute the expected average packet size and inter-packet intervals for the flow (see Section 4.3.1); call them $\bar{z}$ and $\bar{v}$. From these, together with the flow size in bytes, we estimate the expected duration of the flow: $\bar{d} = b\,\bar{v}/\bar{z}$.
4. Randomly sample a start time for the flow. Ensure that the start time plus the expected duration, $\bar{d}$, does not exceed the maximum simulation time for the workflow by using the rejection method.
5. The first packet is sent at the chosen start time. The size of each packet is determined by sampling a random size value. The interval until the next packet is determined by sampling a random interval value. The packet stream ends when the number of bytes allocated to the flow have been emitted or when the packet emission time exceeds $t_{\max}$. The last packet is only the size of the remaining number of bytes.

Some remarks about the procedure:

– For some vectors in $\mathbb{R}^{f\times N}$, it may be impossible to choose distinct source and destination nodes or flow sizes and start times satisfying the procedure. However, for any vector that is a sum of vectors

representing valid flows—i.e. ones having distinct source and destination nodes and feasible combinations of start time, duration, and packet behaviors—the procedure will terminate.
– Because the representation ϕ models the seven properties independently of each other, it is impossible to do better than to sample the properties independently, modulo the requirements of the procedure. In the case of a row which is simply a representation of an actual flow, this makes no difference, except for the generation of packets (which are modeled independently). In the case of composite rows, however, this implies that flow properties are modeled independently of each other.

Despite these remarks, we will show in our experimental evaluation that this procedure generates *sufficiently realistic* wireless workloads [5, 6].

5 Experimental methodology

To evaluate whether traffic models are realistic or not, we use the method of *paired differential simulation* introduced in our previous work on traffic modeling [5, 6]. Paired differential simulation is based on the standard scientific technique of controlled experimentation. The hypothesis we are testing is that a given synthetic model accurately reproduces the performance metrics exhibited by real network traffic. To test this hypothesis, we conduct a series of paired experiments and compare the results for each test subject with a corresponding control subject. In this case, the experiments are wireless network simulations, and the subjects are real or synthetic workload instances. The control group is the set of simulations where the workload is an actual traffic trace, recorded as described in Section 5.2. The test group is a set of simulations where workload is synthetically generated using a simplified traffic model. The experiments are paired: for each control simulation, there is a synthetic workload, with behavior as similar to the control as model will allow, against which the results are compared. The output of the series of experiments are paired values of performance metrics for each simulated scenario: one from the control, one from the test model.

5.1 Traffic models

The first model we evaluate is the general matrix model itself. We derive the matrix representation directly from the trace data and use it to generate workload as described in Section 4.5. This serves to test whether the GMM is sufficiently general: if it accurately reproduces all performance metrics across the collection of scenarios simulated, this provides strong evidence that GMM captures enough of the detail of trace behavior to be considered a sufficiently general representation of network traffic patterns.

After validating the GMM, we explore the effects of various uniformity, marginality, and conditioning assumptions on the realism of traffic models. The uniform models are the most common, being the simplest both to conceptualize and to implement: simply make no distinction between start times, source or destination nodes. Choose them at random, using no information from reality other than the number of nodes and the possible range of start times. We do not evaluate uniform packet behavior because it has been sufficiently discredited in our previous work.

In more sophisticated modeling work, where uniformity is discarded as too simple a model for behavior, marginality is often applied instead. It is assumed—often implicitly—that marginal models are adequate to capture the essential aspects of network behavior. We put that implicit assumption to the test here. How well do perfect marginal models represent real behavior? When we say that our marginal models are "perfect," we mean that they are nonparametric and calculated directly from trace behavior. These are the marginal models that most accurately describe the trace traffic, because they are directly derived from it and make no parametric approximations. This provides an upper bound on the quality of marginal models in general, since parametric marginal models are derived, in turn, as simplifications of these nonparametric marginal behaviors. Table 2 provides a complete list of all the traffic models we compare in our evaluation.

5.2 Trace data

Our general methodology is to compare performance metrics in simulations using real traffic patterns from traces to the same metrics in simulations using a variety of trace-based models and synthetic traffic models defined using the general matrix model. We use a 24-h trace recorded in an infrastructured 802.11 g wireless LAN with 18 access points, deployed at the 60th Internet Engineering Task Force meeting (IETF60), held in San Diego during August of 2004. The traffic trace was captured using `tcpdump` at a single router, through which all wireless traffic for the meeting was routed, including traffic between wireless nodes. The snap length of the capture for each packet was 100 bytes, allowing IP, ICMP, UDP and TCP headers to be analyzed. We limit our work to the 24-h sub-trace recorded on

Table 2 Matrix-based traffic models used in paired differential simulation experiments

Model	Transformation	Description
GMM	—	The general matrix model
Time uniform	$\mathbf{A}' = (1/n)\,\mathbf{A1}$	All start times are equally likely
Node uniform	$\mathbf{S}' = (1/n)\,\mathbf{S1}$ $\mathbf{D}' = (1/n)\,\mathbf{D1}$	All nodes are equally likely as source or destination
Full uniform	*All uniform transformations*	All start times, sources and destinations are equally likely
Time marginal	$\mathbf{A}' = (1/f)\,\mathbf{1A}$	Start times chosen from marginal distribution of start times
Size marginal	$\mathbf{B}' = (1/f)\,\mathbf{1B}$	Flow size (bytes) is chosen from marginal distribution of sizes
Node marginal	$\mathbf{S}' = (1/f)\,\mathbf{1S}$ $\mathbf{D}' = (1/f)\,\mathbf{1D}$	Source and dest. chosen marginally, independent of other properties
Packet marginal	$\mathbf{Z}' = (1/f)\,\mathbf{1Z}$ $\mathbf{V}' = (1/f)\,\mathbf{1V}$	Packet behavior is variable bit-rate, marginal across all flows
Full marginal	*All marginal transformations*	Applies all the marginality assumptions of the above four models
Source conditional	$\mathbf{F}' = \mathrm{rs}(\mathbf{S}^T\mathbf{S})\,\mathbf{F}$	Flow behaviors are aggregated on a per source-node basis
Dest. conditional	$\mathbf{F}' = \mathrm{rs}(\mathbf{D}^T\mathbf{D})\,\mathbf{F}$	Flow behaviors are aggregated on a per destination-node basis

Wednesday, August 4th. This trace contains a broad variety of behaviors and entails a very large volume of traffic: 2.1 million flows, 58 million packets, and 52 billion bytes.

We do not assume or claim that the traffic found at IETF60 is representative of conference settings in general. The observed behaviors are also unlikely to resemble those found in a typical commercial or residential setting. We have chosen this trace, however, because within it can be found behaviors resembling many different types of wireless usage cases. Figure 2 shows the wide variations in the number of active flows and nodes over the course of the trace. In the night and morning hours, the traffic patterns are similar to those one might find in a moderately trafficked business or residential area. During working group sessions, we see highly concentrated, heavy usage patterns. At the zenith of activity, over 800 users, 33 thousand flows, and 1 million packets are seen in a single 10-min trace segment. At the nadir, a lone node sent only a single 61-byte packet over the course of 10 min. All levels of activity between these extremes are represented. Moreover, the mix of traffic types observed changes dramatically over the course of the day, providing a wide representation of possible blends of behavior. This heterogeneity and extreme range of behaviors makes the IETF data set ideal for this evaluation. The variety of activity gives us greater confidence that success or failure of traffic models is not tied to any specific network condition, but is broadly and generally applicable.

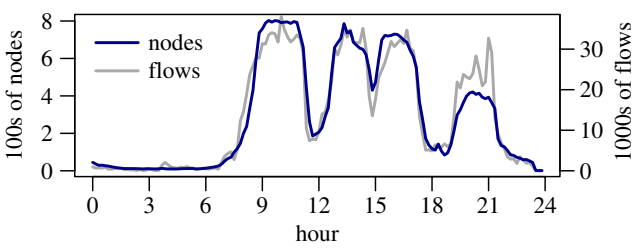


Figure 2 The number of active nodes and flows over time

Before using the traces, it is necessary to extract application-level behavior from the trace header data. First, we split the trace into individual packet flows. A flow is a series of packets sharing the following five attributes: IP and transport protocols (raw IP, ICMP, TCP, UDP); source and destination IP addresses and TCP/UDP port numbers. Next, the quantity of application-initiated data contained in each packet is calculated. For non-TCP packets, this quantity is simply the size of the transport-layer payload, but for TCP the calculation is more complicated: only new data transfers, explicitly initiated by the application, are counted. Data retransmitted by TCP is disregarded, and empty ACKs are ignored. SYN and FIN flags in packets (even empty ones) are counted as a single byte each, since they are explicitly signaled by the application.

5.3 Simulations

We use the Qualnet wireless network simulator (version 4.0.1) to perform our experiments. We simulate a stationary multi-hop 802.11 g network using the Ad hoc On-demand Distance Vector (AODV) routing protocol [9], with nodes placed randomly in a square field with sides of 150 m but a radio range of only 20 m—both in accordance with an indoor network environment. In addition to the active nodes corresponding to trace IPs, half as many passive "infrastructure" nodes are added to each simulation: these nodes initiate no data and simply serve as additional network relays. Our simulations resemble multi-hop mesh networks of the kind that are increasingly studied and deployed for delivery of broadband access in residential, corporate and conference settings. We do not attempt to reproduce the physical environment of the original wireless network, nor do we simulate mobility. The only aspect of the

original network's behavior that is reproduced is the total pattern of network-wide traffic.

There are a number of potential objections to this approach. We use single-hop trace data to drive multi-hop simulations; the physical environment, node mobility, handover behavior, and closed-loop dynamics of the original wireless setting are not faithfully reproduced. One must keep in mind, however, that the goal of this research is *not* to understand the conditions of the original network. Rather, we are using the traffic behaviors observed as examples to help us better understand how different types of workload can affect performance metrics. In particular, we aim to understand how real workload compares with common synthetic traffic models. Of course, the reason for such objections is that networking researchers understand that the many aspects of behavior interact with each other in a complex and nearly inextricable manner. However, before we can hope to understand the interaction between workload and other features affecting network behavior, we must study traffic patterns alone, and learn to model them with reasonable accuracy in the absence of additional complicating factors. Accordingly, in this study, we detach application level traffic patterns from the other factors influencing network conditions, and study them in isolation.

The 24-h trace is split into 144 10-min segments, each of which serves as the basis for a set of simulations using different traffic models. The traffic models range from a completely realistic trace-driven model, to a standard CBR traffic model. Various partially synthetic intermediate models, described in Section 5.1, are simulated to study the impact of different aspects of traffic behavior on network performance. To preserve the fairness of the performance comparison, we keep as many features as possible constant across different traffic models. The traffic generated by each synthetic model preserves as many characteristics from the original trace as possible, within the constraints of the model. Moreover, the following features are preserved across all models: the number of wireless nodes, the number of flows, the number of application-initiated data units sent, the total bytes of application data sent, and the average flow duration (and therefore the average data rate).

In our previous work, we approximated TCP with a pair of half-duplex UDP flows [5, 6]. For this work, we have implemented a new Qualnet application driver that allows trace-driven full-duplex TCP flows. This allows us to accurately reproduce the full dynamics of TCP feedback. Moreover, raw IP, UDP and TCP flows are implemented using the same code, guaranteeing that all traffic is simulated uniformly and that all performance metrics are aggregated in the same manner.

5.4 Performance metrics

We have selected six performance metrics to present here. They are commonly used as indicators of network performance at the application, network, and link layers of the protocol stack:

1. **Application:** average end-to-end delay, packet delivery ratio, received throughput.
2. **Network:** AODV control overhead (RREQ/RREP/RERR), packets dropped in routing queues.
3. **Link:** 802.11 control overhead (RTS/CTS/ACK).

These metrics are commonly used to evaluate wireless protocols. We have examined a broad variety of additional wireless network performance metrics, and although we do not have room to present or discuss them here, the results shown are representative of the overall realism of the traffic models.

6 Results

Our simulation results are summarized in Fig. 3. Each subfigure shows a single performance metric. The distribution of log-ratio error values for each traffic model is visualized with a box-and-whisker plot. The box indicates the range from the 25th to 75th percentiles of values, while the "whiskers" indicate the full range, excluding outliners (which are shown as isolated points). These plots allow immediate assessment of realism: a good traffic model should have error values that are tightly clustered around the center, with a small, evenly balanced box, and relatively small whiskers. Additionally, the mean and median markers should be close to the center. It should be noted that the error scale is logarithmic, so even a slight increase in spread or deviation from the center indicates a disproportionately large decrease in model quality. Non-logarithmic scale markers are shown above each plot; the values indicate the factor of over- or under-representation for the performance metric.

The primary result is that the GMM accurately reflects the performance of the trace data it is based on. It has small, centered error bars for every performance metric presented here, as well as for those we have looked at otherwise. This is an essential result, since otherwise, GMM is at best a shaky foundation for further modeling work. With these results, we can be assured that if we can approximate the matrix for a given example of trace traffic, then we have also approximated the original trace behavior. In a sense, this result is a reduction of the general traffic

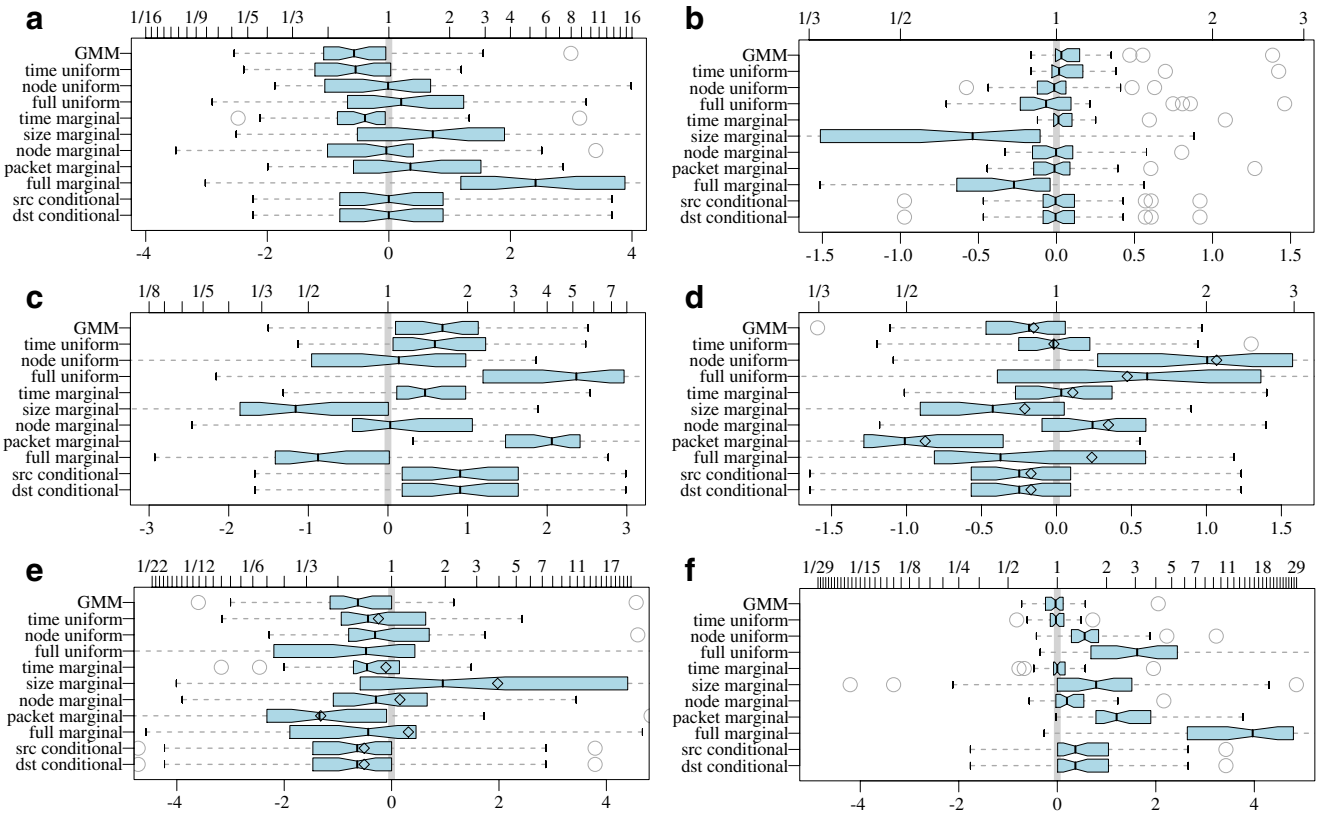


Figure 3 Box-and-whisker plots of log-ratio error values for all metrics and traffic models. The *lower axis* indicates the log-ratio, while the *upper axis* shows raw ratio values. *Each box* contains the central majority of log-ratio values: the *left and right bounds* are at the 25th and 75th percentiles. The *dark middle line* indicates the median value, while the *diamond* marks the mean. The *whiskers (dotted lines)* extend to the furthest non-outlier values, while the points beyond that are outliers. The *notch in the middle of each bar* indicates a 95% confidence interval for the true underlying median value; if two notches do not overlap, they are very unlikely to have the same median

modeling problem to a more tractable matrix modeling problem.

It is worth noting that GMM has competition from a few of the time-simplified models: the time marginal and time uniform model both do approximately as well, and in some cases better, for the metrics examined. Hernández-Campos et al. found strong evidence that session arrivals followed a time-varying Poisson arrival process, meaning that other than large scale time-varying arrival rate and very small scale clustering of flows within sessions, the overall flow arrival rate is fairly smooth. We suspect that on the scale of minutes to tens of minutes in which our simulations exist, it is sufficient to model both flow and session arrivals uniformly. This result indicates that the precise temporal placement of flows is not highly sensitive: they tend to be relatively evenly spaced out at short time-scales, and network performance is not sensitive to changes in start-time on that scale.

Not all marginal models have such a benign effect on the accuracy of performance metrics. The worst offender by far is the size marginal model. This model distorts behavior at every level, by more than a factor 20 in 25% of scenarios in the case of routing queue packet drops. This is a highly significant result because modeling flow size using a network-wide marginal distribution is precisely what Hernández-Campos et al., for example, attempt. This result informs us that this approach is doomed to failure: flow size cannot be assigned without considering its relation to other flow properties; source and destination nodes, and packet behavior at the very least.

The packet marginal model is another offender, albeit not as egregiously. The direction of the misrepresentations in this case, however, is more dangerous: received throughput is overestimated by more than three times, 75% of the time. On the other hand, AODV control overhead is underestimated by half on

average. Such serious misrepresentations are especially noteworthy given the apparently innocuous assumption applied by this model: the packet marginal model is essentially a standard variable bit-rate (VBR) packet behavior model, using the empirical network-wide distributions of packet sizes and inter-packet intervals. Otherwise the model matches the original trace behavior exactly. And yet these drastic distortions occur. In our previous work, we have shown similarly drastic misrepresentations to occur when CBR packet behavior is assumed [6]. This result indicates that to reproduce accurate network performance, we must provide packet behaviors on a more granular level: each node or even each flow must be assigned a "custom" packet behavior by some means.

These strong negative results for two fundamental and very common marginal models provide severe limits on the realism that can be achieved through marginal modeling. Traffic modeling research *must* move beyond simply finding parametric models for marginal distributions of network-wide properties. Realistic behavior cannot be generated using these distributions alone. Something must be captured about the interaction between the elements of flow behavior.

Finally, we turn to the conditional models. These demonstrate neither exceptionally good nor exceptionally bad behavior. That the conditional models should be similar to the full marginal model but somewhat more realistic is entirely expected: conditioning is a restricted form of marginal modeling, where each group of flows has its own marginal distribution. In these cases, the flows are grouped by source or by destination. Neither source nor destination conditioning appears to out-perform the other: they perform identically in each case; neither is perfect, but both are better in general than any marginal or uniform model. The source/dest. conditional models tend to err in the same direction as the full marginal model, but in a few cases where the aspects of marginality lead to errors in different directions, the outcome follows one of the aspects more than others. In the case of received throughput, for example, the packet marginal tendency to overestimate error overshadows the tendency of the other marginal aspects to underestimate. Overall, we see that the conditional models are more realistic than other models, but this is effect occurs only as the number of nodes—i.e. the number of conditioning classes—approaches the full rank of the original traffic matrix. In our discussion, we present and demonstrate a far more compelling method of reducing and simplifying the representation of the behavior of large collections of flows.

7 Discussion

Although we have shown it to be sufficiently realistic for driving wireless simulations, the general matrix model is not intended to be a final model used by networking researchers. Nor is it intended to be the ultimate perfect model of network behavior. Rather it serves as both a research tool and a stepping stone to better understanding of network traffic. Because it captures so much detail about traffic sample—without making assumptions of uniformity or marginality of flow behavior—it serves as a basis for deriving more advanced models that preserve more of the original behavior of networks than any uniform or marginal model ever can. The first important role it plays is in allowing us to succinctly mathematically express common models. All of the traffic models included in this paper are implemented in only 19 lines of Matlab code, applying simple matrix transformations to the GMM representation of traffic. Through the lens of linear algebra, we see why uniform, marginal and even conditional models fail to adequately capture the gestalt behavior of networks: these simplifying assumptions force us to ignore most of the information present in a trace. In what follows, we address limitations of the results presented in this paper, but conclude with an exciting and promising new technique for traffic analysis that amply demonstrates the potential of linear representation of network traffic.

7.1 Generality of results

A significant limitation on the generality of our results is that they are based on a single data set from IETF60—albeit a large and varied one. It is possible that traffic in this trace happens to produce network performance that is unusually dissimilar to standard traffic behavior. This data set, however, represents a highly heterogeneous collection of network usage behaviors, from slow and steady off-peak usage, to extremely heavy peak usage: over 800 users, 33 thousand flows, and 1 million packets in a single 10-min trace segment. Despite the broad variety of behaviors, the results are consistent: in all types of usage scenarios, uniform and marginal models, and to a lesser degree conditional ones, systematically skew important performance measurements at all levels of the network. While the precise results for other data sets might differ, it is very unlikely that these traffic models will happen to accurately reproduce realistic performance in other experiments. This paper provides strong evidence that better traffic models are needed for performance evaluations. Thus, while it is necessarily limited, our

evaluation points us strongly in the direction away from uniform and marginal models.

Another potential concern about generality is the relatively limited set of performance metrics evaluated. As mentioned in Section 5.4, we have evaluated a broad variety of performance metrics—28 in all. The metrics evaluated here were selected both because they are among the most commonly used in evaluations, and because they are representative of the results for other metrics. In general, metrics at the same level of the protocol stack tend to behave in related ways: they may be positively or negatively correlated, but if one metric is severely distorted, it is unlikely that others are left intact. The direction of the distortion of metrics is highly sensitive to the experimental setup; the presence of error, however, is not. In other words, models that are sufficiently realistic in one experimental setup tend to be sufficiently realistic in other experiments; models that misrepresent performance metrics in certain cases tend to do so in other cases as well. The direction of the distortion may change, but the presence or absence of accuracy does not. See previous work using the

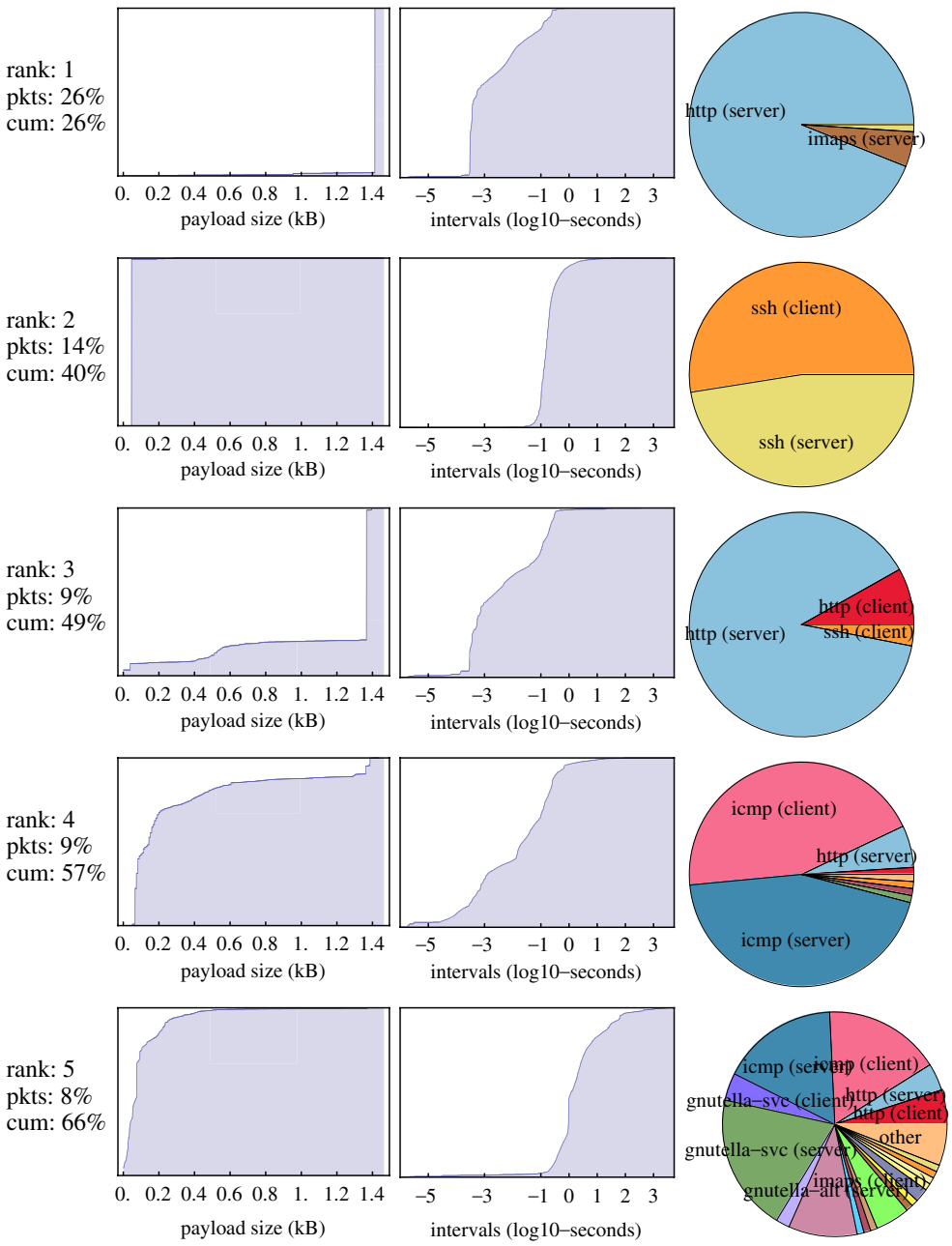


Figure 4 The ten most prevalent components of flow behavior as pairs of packet payload size and of inter-packet interval distributions. The distributions are shown as cumulative distribution functions: the *x*-axis represents payload size (in kilobytes) or inter-packet interval duration (in log-seconds, base 10), respectively; the *y*-axis indicates the probability a value occurring, less than or equal to the *x* value. The behaviors are ordered in descending rank of how many packets they explain; *to the left*, each behavior's rank is indicated, together with the percentage of total packets associated with it, and the cumulative percentage of packets explained. *To the right* of each behavior is a *pie chart* showing the breakdown of associated packets by protocol, as determined by well-known port numbers. *Pie slices* are only labeled if they constitute at least 3% of associated packets

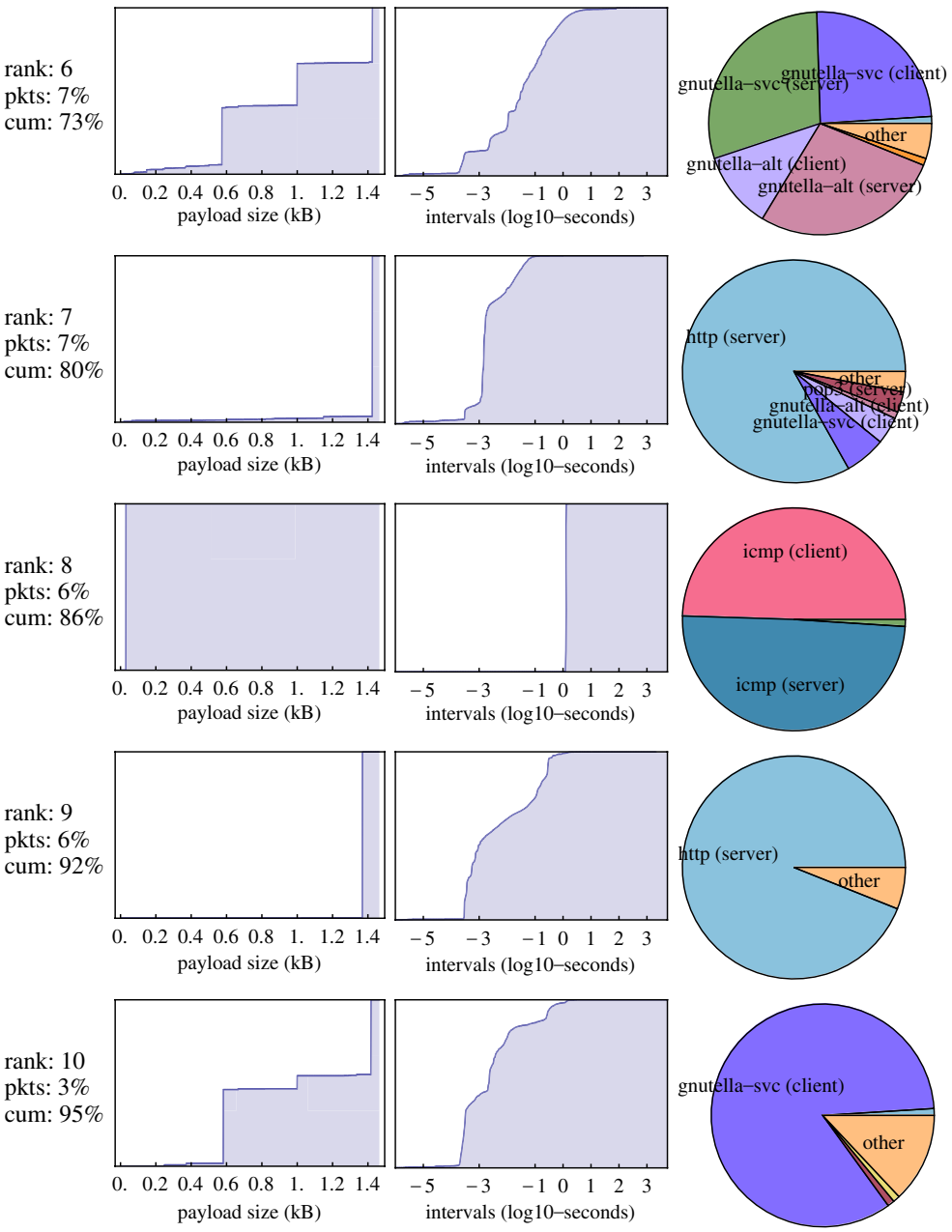
same technique for evaluating realism for further examples with different experimental conditions and more performance metrics considered [5, 6].

7.2 Computational complexity and overhead

Another potential concern about the approach of linear representation is the additional computational overhead it may induce in experimental settings. Large matrix representations of traffic models are certainly more computationally costly than simpler models. Currently, however, realistic local-area traffic modeling is an unsolved problem. The purpose of this work is to establish how traffic can be represented and generated realistically at all, not how it can be done most efficiently. We observe, however, that generating application traffic for a simulation takes orders of magnitude less CPU-time than running the simulation itself. Therefore, traffic generation complexity is considered a non-issue.

A related issue is the significant simulation time incurred when experiments use realistic volumes of traffic. Indeed, this is a problem: we have run our simulations on a 64-node Itanium cluster, and the experiments still take weeks to complete. Given the current state of the art in simulators and CPU power, however, this problem is unavoidable. Including model

Figure 4 Continued



preparation times, running simulations with uniform or marginal models takes negligibly less time than running simulations using realistic models like the GMM. One of the possible effects of this research, however, may be the ability to classify and catalog a realistic set of "exemplary" traffic scenarios for various applications. Such a catalog is far from existing, but its existence would allow researchers to run far fewer simulations to achieve incomparably more general and reliable results.

7.3 Nonnegative factorization of packet behavior

It remains to show how we can simplify and reduce the representation of network behaviors without making unwarranted and manifestly false assumptions of uniformity or marginality of network behaviors. In Section 4.4, we observed that application of uniformity and marginality assumptions to traffic instances is equivalent, in the GMM, to right and left multiplication, respectively, by matrices of rank one. These transformations destroy almost all of the information in the original traffic matrix. Conditional models are equivalent to left-multiplication by somewhat less degenerate matrices, but they achieve realism only by approaching the GMM itself. In this section we describe a matrix factorization technique that can provide an approximate low-rank representation of the original traffic without assuming uniformity or marginality. This technique not only allows us to simplify and analyze traffic without destroying the information content, but also leads us to a deep analytical understanding of network behaviors that matches our intuitive understanding of typical modes of network behavior.

We present here preliminary results from applying nonnegative matrix factorization (NMF) to the packet behaviors of real flows. Recall from Section 4 that the matrices $\mathbf{Z}$ and $\mathbf{V}$ represent the packet size and inter-packet interval histograms of a collection of flows. We apply NMF to the concatenation of these matrices, $\mathbf{P} = [\mathbf{Z}\ \mathbf{V}] \in \mathbb{R}^{f\times(d_z+d_v)}$. NMF attempts to approximate $\mathbf{P}$ as the product of nonnegative matrices:

$$\mathbf{P} \approx \mathbf{W}\Sigma\mathbf{H}, \tag{31}$$

$\mathbf{W} \in \mathbb{R}^{f\times b}$ is column-stochastic, $\Sigma \in \mathbb{R}^{b\times b}$ is diagonal with descending diagonal entries, and $\mathbf{H} \in \mathbb{R}^{b\times(d_z+d_v)}$ is row-stochastic. The inner dimension, $b \in \mathbb{N}$, must be determined as well. What is the interpretation of such a factorization? The rows of $\mathbf{H}$ are *basic behaviors*—pairs of packet size and inter-packet interval distributions, such that each flow's packet behavior can be approximated as a linear combination of these basic behaviors. Specifically, the matrix $\mathbf{M}\Sigma$ gives the mixing coefficients for each flow. The diagonal entries of Σ have a special meaning as well: they are the number of packets associated with each basic behavior. Thus the basic behaviors are sorted in descending order of how much traffic they explain.

Table 3 Components of flow packet behavior, ranked in order of prevalence, with percentage of packets explained

Rank		Description	Well-known ports
1	26%	Download	HTTP, IMAPS, SSH
2	14%	SSH Typing	SSH
3	9%	Web surfing	HTTP, SSH
4	9%	???	ICMP, HTTP, Gnutella
5	8%	???	Everything
6	7%	Gnutella control	Gnutella, SSH, HTTP, other
7	7%	Download	HTTP, Gnutella, other
8	6%	Ping	ICMP, Gnutella
9	6%	Download	HTTP, other
10	3%	Gnutella server	Gnutella, other

Also shown are our interpretation of each behavior and well-known port numbers associated with each.

The striking results of applying nonnegative matrix factorization to $\mathbf{P}$ are shown in Fig. 4. Twelve basic behaviors suffice to explain all the traffic in a "toy" sample of 4138 flows. The pairs of packet size and inter-packet interval distributions are almost immediately recognizable as resembling intuitive modes of network behavior. The first basic behavior, for example, has all packet sizes near the MTU, while intervals are smoothly distributed around a fraction of a millisecond. This is what we would expect from downloading large files; and indeed, the breakdown of packets by port number bears out this interpretation—almost all of the traffic is HTTP server traffic, with small slivers of IMAP and SSH server traffic. The second basic behavior is equally intuitive: tiny packets with frequency distributed around 0.1 s—this is all interactive SSH traffic, as indicated by the associate pie chart. The other behaviors are listed in Table 3 with interpretations. The true significance of this result lies in the fact that NMF succeeds in classifying flows by network behavior completely automatically, not only without using port numbers, but also without *any* information about the number or kind of plausible interpretations. The algorithm not only agrees with our intuition about network behavior, but exceeds it by discovering real but unknown behaviors awaiting interpretation.

8 Conclusions

We have presented a fundamentally new, algebraic way of representing traffic patterns in local area networks.

We call this representation the general matrix model. Each flow of a traffic collection is represented as a single high-dimensional vector with components representing the IP protocol type, source and destination nodes, start time, flow size, and packet behavior. The essential property of the algebraic representation is that standard vector operations compute natural and useful descriptions of aggregate behavior over the collection of flows represented. Our experimental validation demonstrates that the general matrix model accurately reproduces performance characteristics of real traffic.

The benefits of a performance-preserving algebraic representation are multifold. The algebraic forms of the simplifying assumptions made by many common modeling approaches provide unprecedented clarity and coherence to a complex and confusing subject. For example, assumptions that various aspects of flow behavior are stochastically or deterministically uniform all take the same form algebraically: left matrix multiplication. The other major class of common simplifying assumptions made by traffic models is to apply some marginal behavior across the entire collection of flows or to disjoint subsets of the flows. Such simplifications correspond to right matrix multiplication in GMM. Because of the simplicity and clarity that the algebraic structure brings to the subject, we can see that both common types of simplifying assumptions are naïve and simplistic. Our experimental results bolster this conclusion: both uniform and marginal models significantly misrepresent many important network performance metrics. With the general matrix model, however, we stand poised to use the powerful tools of modern linear algebra to develop general, elegant and effective models of network behavior.

References

1. Avallone S, Emma D, Pescap A, Ventre G (2004) A distributed multiplatform architecture for traffic generation. In: International symposium on performance evaluation of computer and telecommunication systems, San Jose, CA, USA, July
2. Clark D (1988) The design philosophy of the DARPA internet protocols. In: ACM Sigcomm. ACM, New York
3. Hernández-Campos F (2006) Generation and validation of empirically-derived TCP application workloads. PhD thesis, Univ. of North Carolina, Chapel Hill
4. Hernández-Campos F, Karaliopoulos M, Papadopouli M, Shen H (2006) Spatio-temporal modeling of traffic workload in a campus WLAN. In: 2nd annual international workshop on wireless internet, Boston, MA, USA, August
5. Karpinski S, Belding E, Almeroth K (2007) Towards realistic models of wireless workload. In: 3rd workshop on wireless network neasurements (WinMee), Limassol, Cyprus, April
6. Karpinski S, Belding E, Almeroth K (2007) Wireless traffic: the failure of CBR modeling. In: 4th international conference on broadband communications, networks, and systems (Broadnets). Raleigh, NC, USA, September
7. Lee D, Seung H (2001) Algorithms for non-negative matrix factorization. Adv Neural Inform Process 13:556–562
8. Paxson V, Floyd S (1995) Wide-area traffic: the failure of poisson modeling. IEEE/ACM Trans Networking 3(3):226–244, June
9. Perkins C, Belding-Royer E, Das R (2003) Ad hoc on-demand distance vector (AODV) routing. Request for comments (Experimental) 3561, Internet Engineering Task Force, July
10. Rudin W (1987) Real and complex analysis, 3rd edn. McGraw Hill, New York, NY
11. Sommers J, Barford P (2004) Self-configuring network traffic generation. In: ACM/USENIX internet measurement conference, Taormina, Sicily, Italy, pp 68–81 October
12. Weigle M, Adurthi R, Hernández-Campos F, Jeffay K, Smith F (2006) Tmix: a tool for generating tcp application workloads in ns-2. In: ACM/SIGCOMM computer communication review, Pisa, Italy, July