Pricing the Cloud

Current cloud pricing schemes are fairly simple, but what's the future of cloud pricing? Here, the authors discuss the economic issues shaping the cloud marketplace and address the open questions these issues yield. They then explore what the current state of research in economics and computer science has to say about some of these questions and how this relates to the future evolution of cloud pricing.

Ian A. Kash and Peter B. Key
Microsoft Research

Robert Gossman informally defined clouds as "cluster[s] of distributed computers providing on-demand resources ... over the Internet" at scale, which still serves as a useful definition.[1] The scale here refers to datacenter-size units, where the largest datacenters might have hundreds of thousands of servers. This scale naturally yields savings, and an influential whitepaper from Microsoft argued the benefits of outsourcing IT infrastructure from in-house or private cloud to public cloud.[2] The shift from owned infrastructure to public cloud has accelerated over the past few years, helped also by simple interfaces and virtualizations. Moreover, cloud computing itself is becoming richer. Although the dominant cloud services still primarily sell computing instances (Amazon's Elastic Compute Cloud [EC2], Google's Compute Engine, and Microsoft's Azure Compute), now different types of resources (such as storage) are also being offered, platform-as-a-service (PaaS) offerings such as app services are appearing, as well as more sophisticated products, such as Azure ML.

In simple economic terms, current cloud providers form an oligopoly. There are certain natural constraints on capacity, which argues the equilibrium pricing for raw resources approximates a Cournot equilibrium. This equilibrium gives prices higher than the competitive price (where prices equal production costs) — however, with decreasing resource costs, this essentially becomes an almost frictionless commodity market. For example, Microsoft has committed to match Amazon's pricing for basic infrastructure. The natural reaction of the providers is to want to provide service differentiation, and create a richer set of services that have high added value to the users — hence, the desire to "move up the stack" from vanilla infrastructure as a service (IaaS) to rich PaaS offerings and beyond to software as a service (SaaS) and complete solutions. The benefits to end users are higher-value services. For providers, the potential profits from these richer services provide the needed incentive to build market share in the commodity market. Orthogonal to this is the use of variants of price discrimination that charge different amounts for (essentially the) same products as a way to capture more of the value created, thereby earning more revenue. We can interpret current forms of

 Published by the IEEE Computer Society 1089-7801/16/$33.00 © 2016 IEEE

tiered or menu pricing for compute instances as examples of this.

But what about cloud economics' and cloud pricing's future? With more computation occurring in the cloud, this is an important question. Here, we look at what the Econ-CS research field (by which we mean the cross-disciplinary field that lies at the intersection of computer science, game theory, operations research, and economics) has to say about this question.

Goods and Services

The jury is still out on what the fundamental "goods" are with respect to pricing the cloud. It's a messy place with different types of resources available (storage and bandwidth), different application types (server and batch), and different service types (IaaS and SaaS), all of which come with variations in attributes such as quality-of-service (QoS) measures and service-level agreements (SLAs). The model that has been most commonly adopted to date is a utility model: find something whose use can be measured and charge per unit, much like electricity and water markets work on the consumer-facing side. Thus, we have Amazon Web Services (AWS) and Azure offering prices for virtual machine (VM) hours, storage gigabytes, and external bandwidth. Even more sophisticated services such as load balancing and database use are metered. To some extent, this is reasonable because the amount of capacity used can be viewed as a proxy for both the value created for customers and the provider's costs, but in many cases it seems a crude one. Perhaps the biggest virtue of this approach is that it's simple, both to understand and implement.

Electricity is a good reference point in this respect; power companies charge residential customers for usage as a fixed rate because it's all their technology allows them to do. One of their motivations to moving toward smart meters is the eventual ability for finer-grained pricing that reflects the fact that their costs vary throughout the day with demand (as exists today for larger customers). In contrast, airline yield-management teams engage in exquisite price discrimination, where every customer on a flight might have paid a different price based on market segmentation, shifts in demand over time, and the full context in which that seat is purchased.

Pricing of Internet or network services also provides a natural comparison for cloud pricing. Pricing telephony evolved towards non-linear, time-varying, usage-sensitive pricing for long-distance, coupled with a flat fee for connection and local traffic; as the provisioning cost of telephony has declined, so has the usage-sensitive component of the pricing for fixed network telecoms. The primary good for Internet pricing is bandwidth, which is provided through congestible resources. Thus, it looked as though it was a natural candidate for non-linear, usage-sensitive pricing.[3] But instead, flat-rate pricing became almost universal. The current practice is to offer end-customers limited menu pricing, with a flat fee plus a fixed additional fee, dependent on the maximum bandwidth rate and maximum bandwidth transferred, which is similar to mobile telephony pricing.

These flat-fee, unlimited-use prices appear to fly in the face of economic theory, which argues that usage-sensitive pricing is optimal. Andrew Odlyzko[4] argues that declining provisioning costs mean that customer's demand for simplicity override the benefits of usage-sensitive charging, while Arun Sundararajan[5] argues from a theoretical model that any positive transaction cost associated with implementing a usage-based charging scheme makes it optimal for sellers of information goods to offer customers a combination of usage-based pricing and unlimited-use fixed fees. An information good is defined as having zero variable production costs, which doesn't exactly hold for bandwidth provisioning, and is quite far from holding for cloud providers. Thus, we expect cloud pricing to take a somewhat different route.

Compute Capability or Computational Assets?

Often customers want to run batch tasks on the cloud, such as MapReduce jobs. What's the object of value that should be priced here, the cluster and software associated with running the job, or a service that takes the job and runs it for you? Both business models exist, even within the same company (for example, Google offers Cloud SQL to let customers provision their own SQL databases as well as BigQuery to let customers simply submit individual queries). In such instances, pricing needs to interact accordingly, but how should this be done in practice?

Constraints and Desires

In some types of cloud systems, there's no choice about when to do work: when a user makes a request of a Web-facing system, it needs to be

responded to immediately. In others, there's some flexibility about when work is scheduled; for example, a job that must be run sometime between the close of business one day and the start of business the next. How should cloud services allow customers to express this flexibility? Provide no guarantees and leave it up to the customer to decide when to do the work each day? Provide a reservation service? Accept jobs with hard constraints about when they're run and an SLA about meeting those constraints? Allow a more expressive utility function that explains how value changes depending on when the work is completed? Similarly, how should providers share the information that they have? For example, what historical data should Amazon provide about bids in its spot markets? Such combined scheduling and pricing models are an active area of research in a variety of contexts, and the right answer remains unclear.[6]

Unidimensional or Multidimensional?

What are the fundamental elements involved in computation? Should they all be bundled together and sold as a unit as a notional VM with associated connectivity, bandwidth, and storage, or sold as a flexible computational entity where all these aspects can be customized? We call the former *unidimensional*, because the only decision a customer need make is how many of these units are required (of course, in practice, there might be a small number of different options rather than literally a single type of VM), and the latter *multidimensional* because the customer must express preferences along a number of axes. From an economic perspective, these two models lead to highly different techniques, with those for a unidimensional world much better understood. In the remainder of this article, we examine how research has begun to address some of the questions we raised, roughly splitting it using this lens of unidimensional versus multidimensional approaches.

Pricing Unidimensional Offerings

The simplest type of cloud pricing is for unidimensional resources, such as basic IaaS offerings or raw compute power. Because resources are continually being sold, this argues for a stochastic demand model, where resources behave as a queuing system. Within this framework, customers can value jobs differently, and also have different valuations for response time.

A Hybrid Market: Pay as You Go and a Spot Market

In our work with Vineet Abhishek,[7] we model customers as having different values for jobs by assuming that customers belong to different classes, where each class has a different value v_i for job completion, a different distribution for waiting time "costs" $c \sim F_i(c)$, and a different arrival rate. The market is a hybrid one, comprising a pay-as-you-go (PAYG) market offering a fixed priced p per unit time, and a spot market with a variable clearing price.

A job pays an amount m for using the resource, and if it spends a time w in the system, then the payoff to a class i customer with waiting time cost c is $v_i - cw - m$, the difference between the value to the customer and the cost incurred. Hence, the expected payoff to a job entering the PAYG market is $v - (c + p)$, where we have assumed the expected service time is normalized to one and the PAYG market is assumed to have sufficient capacity to serve all demands with negligible queueing time. The spot market has finite capacity and runs an auction mechanism, eliciting bids from customers and giving priority to those jobs that bid more, preempting jobs as necessary, where preempted jobs rejoin the queue. Hence, the expected cost to a job in the spot market is $\hat{w} + \hat{m}$, where the expected waiting time is $\hat{w} \geq 1$, because the job incurs possible additional delay while waiting or preemption occurs.

We adopt a static equilibrium model and assume customers are rational agents and will choose the option that maximizes their expected payoff, whereas the market provider chooses the price p to maximize expected revenue. Under mild assumptions, we show that behavior can be characterized regardless of the precise auction mechanism used, and results are insensitive both to the distribution of arrival times and to the service time distribution. We show that, for each class i there is a cutoff c_i below which jobs participate in the spot market, that the payment function $m(c)$ must be increasing in c, and that there's a unique vector of cutoffs $c(p)$ for a given price p.

More surprisingly, typically PAYG raises more revenue than the hybrid mechanism: indeed, it's always the case when all classes participate in PAYG — that is, when for the optimal hybrid price p^h, $p^h \leq v_i - c_i(p^h)$ for all i. Figure 1 shows an example when not all classes participate in the

spot market (for a high price), and still PAYG raises more revenue. Why does this happen? In any hybrid mechanism there's no way to prevent high-value, low-waiting-time-cost jobs choosing the spot market when they would have been willing to pay a higher PAYG price; in other words, "rich" jobs can choose the "cheaper" class, with a consequent loss of revenue. This assumes that there are no extra costs associated with pre-emption. If such costs are sufficiently high, they would allow the spot market to avoid cannibalizing the primary market. Of course, one way to avoid the cannibalization effect would be using a damaged goods approach, where delay is deliberately introduced into the secondary market as a way of ensuring that significant costs are felt. Alternatively, a spot market could be run for reasons other than revenue optimality, such as to gain market share by attracting new small-scale customers.

Their findings are consistent with anecdotal evidence that Amazon makes it difficult to operate in their spot market at scale, and with the findings of Orna Agmon Ben-Yehuda and her colleagues,[8] who found that Amazon controls reserve prices and causes them to spike, suggesting that Amazon might be making its spot market behave like a menu-priced market. Even in cases where a spot market would be beneficial from a revenue perspective, it might be desirable not to run one to avoid the complexity it creates for customers.

Adding Time

Even when a single good is being priced, there's room for considerable richness beyond simply asking about willingness to pay (whether through a fixed price or an auction). A simple extension involves adding time, specifically to deal with the scheduling and pricing of batch jobs (such as MapReduce, DryadLINQ, or SCOPE), which extends the idea of admission control. For many such jobs, it's critical that they're completed by a particular deadline. For others, there might not be a hard deadline, but the value of the job depends on how soon they can be completed. A line of work looks at this scheduling problem, using algorithms based on linear programming approaches.[6] While the primary focus is on deciding which jobs to schedule and when, this work has also looked at how to charge prices such that it's optimal for job owners to truthfully reveal how their value for the job changes depending on when it's done.

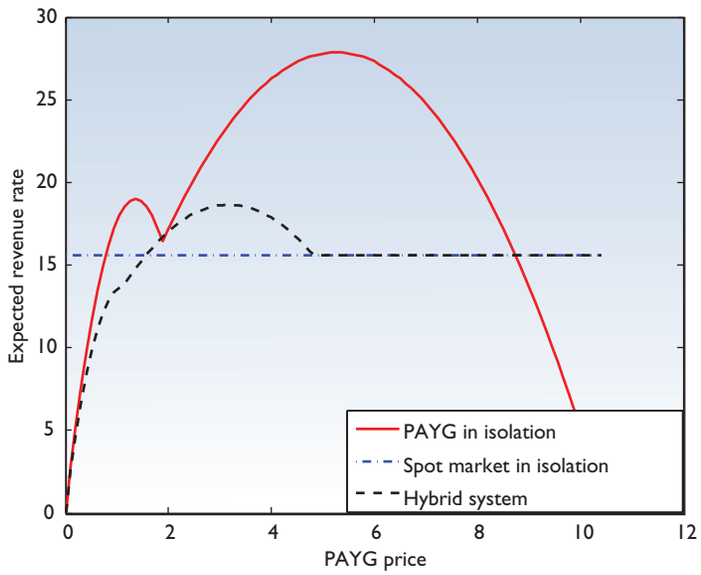

Figure 1. Fixed pricing, spot pricing, and a hybrid market. (PAYG stands for pay as you go.)

Pricing Storage

For many cloud offerings, however, control resides with the customer, and the provider is left guessing how the resources will be used. A natural alternative is to ask the customer for information about future resource usage and prices accordingly. Sofia Ceppi and Ian Kash[9] explore pricing for storage, which asks in advance for predictions about how much will be used each month, in contrast to current pricing that charges per month based on the total that was used. While complicated probabilistic information could in principle be requested, to keep things simple from the perspective of the customer, Ceppi and Kash assume this consists solely of lower and upper bounds on usage over a time interval, with the provider using its own models to understand how that will affect usage in each period. For the provider, this is helpful information to make capacity-planning decisions, and in particular allows for more efficient operation by reducing both the amount of storage that must be kept free to allow for future use and the frequency with which a customer exceeds the amount of storage locally available and ends up getting its data split to, for example, another rack.

Although this information is useful, getting it presents a significant pricing challenge for the cloud. Because tighter estimates are more useful to the provider, these should be rewarded with lower prices. At the same time, this information

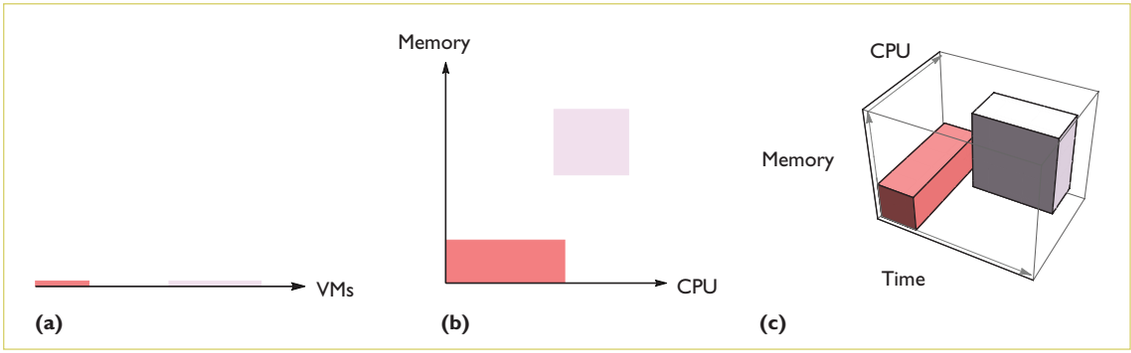

Figure 2. Resource demand profiles: (a) unidimensional and (b and c) multidimensional. (VM stands for virtual machines.)

is only useful if it's accurate, so the prices need to be such that customers aren't incentivized to report inaccurate estimates. They provide a pricing scheme that provides these incentives, as well as ensuring that the provider covers its costs. The main idea is to quote customers a price per gigabyte per month, just as is done today, except that these prices are personalized based on the customer's report. If the report proves accurate (that is, the customer doesn't violate the lower or upper bound), that's all the customer pays. If the bound is violated, the customer pays an additional penalty charge based on how badly it was violated, and these penalty charges are carefully calculated to provide correct incentives.

While they focus on pricing one particular aspect of cloud services and eliciting one particular piece of information about future usage, this is an area ripe for further exploration. Better understanding of customer plans more broadly could lead to substantially higher use of resources, and thus substantially lower costs.

Marketplace Competition

Competition is important, but there are few existing models of competition in the cloud. Jonatha Anselmi and his colleagues[10] looked at a stylized tiered model of the cloud, where users seek service from service providers (SaaS), who themselves buy resources from providers (IaaS or PaaS), and consider both congestion and pricing. In this vertical market structure, under their model the profits of the IaaS or PaaS providers decrease as competition intensifies, whereas that of the service providers doesn't; in effect, the SaaS providers maintain their market power.

Looking back at the (much simpler) Internet pricing literature is salutary: simple modes offering different levels of QoS looked appealing initially, but then looked fragile in the face of competition. This is similar to the case of the so-called "Paris Metro Pricing" proposal,[11] where better QoS was provided solely by charging more for a better service — taking its name from pricing used at one time in the Paris metro, where first- and second-class carriages were identical, but tickets cost more for first-class carriages, and hence were likely to be less crowded. Richard Gibbens and his colleagues[12] showed that such differential pricing wasn't sustainable under competition, because an operator offering, say, two levels of service by splitting capacity with differential charging, would lose out to a competitor offering a single price.

Problems of Multidimensional Goods

Multidimensional approaches explicitly grapple with the fact that VMs aren't monolithic entities — instead, they're bundles that bring together resources such as CPU, memory, and bandwidth (see Figure 2), where each demand has a minimum and maximum requirement for a resource. Because different applications have different needs for these resources, gains from trade are possible. But realizing those trades requires confronting issues at a variety of levels: implementation, information, and mechanism design (for example, how resource allocation and pricing decisions are made).

Delivering Flexibility

For implementation, actually providing such flexible bundles isn't a trivial task. We can handle some resources, such as processor time or memory, by extending standard paradigms for scheduling and resource allocation on a single

machine to datacenter settings. But what about resources such as bandwidth, where providing guarantees requires bringing together network design, VM placement, routing, and congestion control? The systems community has seen a large body of work on performance isolation for network, middle-box, and storage resources, as well as their combination into a virtual datacenter.[13] As the ideas from this research percolate into datacenters, cloud service providers will begin to be able to address a problem with current pricing schemes: costs depend on factors outside the control of customers. For example, if a customer's VMs are located such that there's significant network contention, jobs will take longer to run, and, because VMs are charged based on how long they're used, they'll cost more.[14]

Once resources can be delivered, the cloud provider needs sufficient information to determine which resources to deliver. From a customer perspective, this requires understanding both how a job or system will perform with different resource bundles and how to express this knowledge to the cloud provider. Ideally, such job-performance profiling would be largely automatic. Virajith Jalaparti and his colleagues[15] studied this problem with applications from three domains: data analytics, Web facing, and high-performance computing (HPC). They found that they could achieve similar performance in all three settings with several remarkably different bundles of resources. For MapReduce jobs, they were able to provide reasonable predictions for what performance would be with different bundles composed of network and compute resources. However, such batch jobs are perhaps the easiest setting in which to make such predictions, and much more work is needed in this area.

Fair Division

Once customers know what they want and how to explain this to the cloud provider, and the cloud provider is able to deliver the resources it promises, there remain significant challenges to determining how to operate the market that determines who gets what resources. While mechanism design is well understood in unidimensional settings, much less is known in multidimensional settings. However, one line of work has managed to bring together good properties in terms of both fairness and incentives. It assumes that the decision on whether to admit or schedule a request has already been made, and focuses on the consequent allocation. Further, it assumes that the system is work-conserving in the sense that it attempts to deliver as much of the available resources to each job as it can while being fair, as opposed to simply delivering the amount required to meet an SLA.

Ali Ghodsi and his colleagues[16] extended earlier ideas about max-min fairness to multiple resources, under the assumption that people's preferences for them are *Leontief*, which means they only want them in some fixed ratio. A good example is hot dogs and buns, where I only want a hot dog if I have a bun to go with it and vice versa. In a cloud setting, this makes sense when someone needs, for example, a particular amount of compute and memory to create a useful VM, but we then can create a large number of copies of VMs that can do useful work. In this setting, instead of maximizing the minimum total amount of resources a person gets, their algorithm maximizes the minimum amount of the resource that the person requires the most of (relatively speaking). Thus, Ghodsi and his colleagues call this approach *dominant resource fairness* (DRF). They extend this deal to customers who have paid different amounts by weighting them appropriately. This allows systems with a number of key guarantees:

- nobody is worse off than if all the resources had just been divided evenly,
- nobody prefers the bundle of resources that someone else received,
- any leftover resources aren't usable by anyone, and
- nobody has an incentive to misreport which bundles of resources they need.

This idea also provides the inspiration for Mesos, a thin management layer now in Apache, which allows different applications (such as Hadoop and Spark) to share the same underlying pool of resources.[17]

Fairness and Time: Dynamic Fairness

Systems like Mesos take a static view of the world, and try their best to maintain their fairness guarantees at each point in time. In other work, Ian Kash and his colleagues[18] study a version of the problem where not all applications necessarily exist at the same time, and showed that these techniques extend to this setting. However, if resources cannot easily be taken away from an application there is an inherent tension between efficiency (putting resources to work now) and

fairness (saving resources for later arrivals). They give two different algorithms, each of which relaxes one of these two guarantees from DRF while preserving the other.

Fairness with Multiple Entities

Cloud platforms have a large ecosystem of applications and services running on them. This includes both services provided by the cloud provider and services built by customers and then sold onward to other customers. When such services communicate, there are three different economic relationships involved: two between the services and the cloud provider and one between the services. How should these multiple economic relationships affect the allocation of bandwidth? Hitesh Ballani and his colleagues[19] demonstrate that this scenario is already common and propose an answer to this question using a notion of *upper-bound proportionality*, which limits the bandwidth that a service can acquire regardless of the amount paid by those with which it communicates. This prevents services that communicate widely from claiming a disproportionate share of the network.

Fairness in a Cooperative Game Setting

Customers' willingness to pay for resources is handled in the above fairness settings by using weighted allocations — effectively conflating a proportionally fair principle with the chosen fairness method (such as DRF). An alternative approach is adopted by Gideon Blocq and his colleagues,[20] who introduce the Shared Assignment Game, borrowing ideas from cooperative game theory to discuss both allocation and pricing in a static context. In the Shared Assignment Game, sellers have multidimensional resources, and buyers need bundles of resources to execute their jobs. Buyers have values for their jobs, and the game is a cooperative game, where buyers and sellers are the agents, and the objective is to find the coalition of agents that maximizes (say) social welfare — the aggregate welfare of the coalition. Clearly, a coalition that doesn't have both sellers and buyers has zero value, and the value of a coalition is defined as the maximum welfare achievable from a feasible assignment, where a feasible assignment matches jobs to resources that respects capacity constraints.

As a simple example, the sellers could be individual servers with multidimensional attributes (CPU, memory, and bandwidth), and the buyers each have a number of jobs with associated values. Pricing is performed using the Shapley value as an instrument for revenue sharing: this takes the optimal welfare from the grand coalition of all agents, and apportions it by calculating what each buyer or seller contributes, by looking at their contribution to each subcoalition, randomizing over the way subcoalitions are formed. Such an apportionment is a fair division, in that it rewards those who contribute the most, and can be derived from an axiomatic approach to fairness. Calculating the allocation and the Shapley value are both computationally hard, so approximation methods are needed. Although such pricing is not strategy-proof, it's reasonably resistant to natural manipulations, such as *splits* where buyers split their goods, or *bluffs* where fake goods are declared. Simulations suggest that using the Shapley value as a basis for pricing could improve both welfare and revenue. While this presents an interesting viewpoint, implementing it would introduce a number of difficulties. Nevertheless, it provides a useful perspective to inform pricing decisions.

In describing some of the pricing issues inherent in pricing the cloud, along with related research, we deliberately focused on the simplest settings, such as pricing resources or jobs in an IaaS or PaaS setting. However, the approaches taken and issues raised apply much more broadly to SaaS offerings and beyond — for example, the fundamental dichotomy between unidimensional or multidimensional service specification and allocation applies generally. Within multidimensional settings, time behaves as a dimension, bringing its own unique issues. As services evolve, the type of resources might become even richer. Presently, the understanding of multidimensional pricing is embryonic: the work on fairness gives a handle on allocation, but even this is partial. The fairness framework doesn't account for the future effect on demand that a fair-allocation might have (demand externality), and isn't well integrated with temporal requirements. This is an important and fruitful area for research. Multidimensional scheduling and pricing offers greater potential for increasing both customer satisfaction and revenue; but militating against increasing complexity in pricing and scheduling is the customers' need for simplicity: pricing schemes must be understandable by a user or

their agent. A complementary strand of research and innovation is needed to understand how best to capture and reflect user requirements.

More broadly, it's clear that pricing is in its infancy in the cloud. The research frontier is moving rapidly, and we expect that in the coming years the approaches used by cloud providers will do so as well.

References

1. R.L. Grossman, "The Case for Cloud Computing," *IEEE IT Professional*, vol. 11, no. 2, 2009, pp. 23–27.
2. R. Harms and M. Yamartino, *The Economics of the Cloud*, tech. report, Microsoft, 2010.
3. J.K. MacKie-Mason and H.R. Varian, "Pricing Congestible Network Resources," *IEEE J. Selected Areas in Comm.*, vol. 13, no. 7, 1995, pp. 1141–1149.
4. A. Odlyzko, "Internet Pricing and the History of Communications," *Computer Networks*, vol. 36, no. 5, 2001, pp. 493–517.
5. A. Sundararajan, "Nonlinear Pricing of Information Goods," *Management Science*, vol. 50, no. 12, 2004, pp. 1660–1673.
6. N. Jain et al., "A Truthful Mechanism for Value-Based Scheduling in Cloud Computing," *Theory of Computing Systems*, vol. 54, no. 3, 2014, pp. 388–406.
7. V. Abhishek, I.A. Kash, and P. Key, "Fixed and Market Pricing for Cloud Services," working paper; http://arxiv.org/abs/1201.5621.
8. O.A. Ben-Yehuda et al., "Deconstructing Amazon EC2 Spot Instance Pricing," *ACM Trans. Economics and Comput*ation, vol. 1, no. 3, 2013, pp. 16:1–16:20.
9. S. Ceppi and I.A. Kash, "Personalized Payments for Storage-as-a-Service," *Proc. 10th Workshop the Economics of Networks, Systems, and Computation*, 2015; http://netecon.eurecom.fr/NetEcon2015/Final_for_website/Paper6-short.pdf.
10. J. Anselmi et al., "The Economics of the Cloud: Price Competition and Congestion," *ACM SIGecom Exchanges*, vol. 13, no. 1, 2014, pp. 58–63.
11. A. Odlyzko, "Paris Metro Pricing for the Internet," *Proc. 1st ACM Conf. Electronic Commerce*, 1999, pp. 140–147.
12. R. Gibbens, R. Mason, and R. Steinberg, "Internet Service Classes under Competition," *IEEE J. Selected Areas in Comm.*, vol. 18, no. 12, 2000, pp. 2490–2498.
13. S. Angel et al., "End-to-End Performance Isolation through Virtual Datacenters," *Proc. 11th Usenix Conf. Operating Systems Design and Implementation*, 2014, pp. 233–248.
14. J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance," *Proc. VLDB Endowment*, vol. 3, nos. 1–2, 2010, pp. 460–471.
15. V. Jalaparti et al., "Bridging the Tenant-Provider Gap in Cloud Services," *Proc. 3rd ACM Symp. Cloud Computing*, 2012, article no. 10.
16. A. Ghodsi et al., "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types," *Proc. Usenix Conf. Networked Systems Design and Implementation*, vol. 11, 2011, p. 323–336.
17. B. Hindman et al., "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center," *Proc. Usenix Conf. Networked Systems Design and Implementation*, vol. 11, 2011, pp. 295–308.
18. I. Kash, A.D. Procaccia, and N. Shah, "No Agent Left Behind: Dynamic Fair Division of Multiple Resources," *J. Artificial Intelligence Research*, 2014, pp. 579–603.
19. H. Ballani et al., "Chatty Tenants and the Cloud Network Sharing Problem," *Proc. 10th Usenix Conf. Networked Systems Design and Implementation*, 2013, pp. 171–184.
20. G. Blocq, Y. Bachrach, and P. Key, "The Shared Assignment Game and Applications to Pricing in Cloud Computing," *Proc. 2014 Int'l Conf. Autonomous Agents and Multi-Agent Systems*, 2014, pp. 605–612.

Ian A. Kash is a researcher at Microsoft Research. He works at the interface between computer science and economics. Kash has a PhD in computer science from Cornell University. Contact him at iankash@microsoft.com.

Peter B. Key is a principal researcher at Microsoft Research. His research interests include networks, economics, and algorithms. Key has a PhD in statistics from London University. He's a fellow of ACM, IEEE, the Institute of Engineering and Technology, and the Institute of Mathematics and its Applications.. Contact him at peter.key@microsoft.com.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*