Rev. Econ. Design
DOI 10.1007/s10058-009-0089-4

ORIGINAL PAPER

Firms, queues, and coffee breaks: a flow model of corporate activity with delays

Benjamin Golub · R. Preston McAfee

Received: 18 May 2007 / Accepted: 30 July 2009

Abstract The multidivisional firm is modeled as a system of interconnected nodes that exchange continuous flows of projects of varying urgency and queue waiting tasks. The main innovation over existing models is that the rate at which waiting projects are taken into processing depends positively on both the availability of resources and the size of the queue, capturing a salient quality of human organizations. A transfer pricing scheme for decentralizing the system is presented, and conditions are given to determine which nodes can be operated autonomously. It is shown that a node can be managed separately from the rest of the system when all of the projects flowing through it are equally urgent.

Keywords Decentralization · Transfer pricing · Queuing

JEL Classification C44 · C62 · L23 · M11 · M21

1 Introduction

Since the foundational work of Marschak (1955) and Radner (1959), an active research program has focused on analyzing the firm as a network of interacting units whose

B. Golub
Graduate School of Business, Stanford University, 518 Memorial Way, Stanford, CA 94305, USA
e-mail: ben.golub@gmail.com

R. P. McAfee (✉)
Yahoo! Research, 3333 Empire Blvd., Burbank, CA 91504, USA
e-mail: preston@mcafee.cc

Published online: 19 September 2009

operations can be optimized.[1] This paper continues that program by adding certain human elements to standard queuing models—in particular, the tendency for some resources to be idle even when tasks remain undone, and for units to work more efficiently when they are busier.

The aim of this study is to examine resource allocation and decentralization in multicomponent enterprises that handle projects of varying urgency. To this end, we model the firm as a network of interlinked nodes which process tasks and queue pending jobs. This approach imposes only weak assumptions about the structure of the corporation. For example, no managerial division is modeled as being outside the structure of processing. Every operation is viewed as a node, from assembly lines to legal and human resources departments.

This work extends the value-chain model described by Porter (1985) and the value net of Brandenburger and Nalebuff (1996), in which products flow through a processing network. Our insight, which builds on the McAfee (2002) three-layer model, is that services and administrative activities can be viewed in a similar light as physical outputs and intermediates—that is, as flows of jobs or tasks within the firm. For example, production and sale of output generates customer returns, lawsuits, repeat purchases, and so forth. These tasks generate further tasks—e.g., lawsuits generate requests for information—which flow around the company. Our major assumption is that such demands scale linearly. For example, increasing the number of employees increases demands on the human resources department proportionally. Increasing sales increases customer returns proportionally. Since tasks are modeled in an abstract way, the model can accommodate virtually any task repeatedly performed within the company. Indeed, the only major task which is not modeled is the redesign of the system itself. Since the primitives in this approach are jobs—i.e., requests for service—flowing through a network of business components, the theory is consistent with on demand, service-oriented, component-based models of the firm which have recently appeared in the literature (Fisher et al. 1994; Crawford et al. 2005).

The main innovation of this paper concerns the modeling of the rate at which waiting tasks are taken into processing. In traditional models of queuing networks, processors are fully occupied, though there may be delays in switching from task to task (Kleinrock 1975; Takagi 1991). These models are poorly suited to many business operations, in which some processing resources may be idle even when there are remaining tasks. For instance, airplanes are not filled even when some passengers desire flights on the same route; employees on a team are temporarily idle while some tasks remain undone; goods sit in inventory while customer orders are slowly filled. In these cases, the uptake of projects into processing is delayed, not by a fixed interval of time, but by an amount that depends on both the scale of unused resources and the magnitude of the queue.

There are many reasons for this; we focus on two stylized facts. First, employees tend to be faster at moving waiting tasks to available capacity when queues are long than when they are short. That is, workers tend to take fewer coffee breaks when they are busy. Second, uptake of tasks in human organizations depends on matching

[1] See Marschak and Radner (1972) for a detailed review of the development of this literature.

them to appropriate resources, which is faster when there are more unoccupied processors checking for new assignments. An innovation of our approach over existing queuing models which have been used to study economic problems is that it captures these features of human uptake. Consequently, we are able to model the dependence of unused capacity on system demand and investigate the impact of this on transfer prices, decentralization, and other economically relevant questions. Throughout, our analysis focuses on the expectations of the variables involved (such as inflow rates, process sizes, and queue sizes) and thus we work with deterministic variables and abstract from the stochastic nature of arrivals.

The analysis of the model proceeds as follows. In Sect. 2, we examine the steady state configuration of the network and obtain stability results under the variable uptake hypothesis. It is shown that the system has a unique, locally stable steady state. We then consider the problem of resource allocation. Value is determined by the speed of task completion moderated by the importance, or urgency, of the task. This is modeled as follows: the firm is paid per completed project, minus a penalty that is proportional to the amount of time taken to complete the project. The constant of proportionality depends on where the project originates and is higher for more urgent projects. The corporation trades off capacity costs against queuing and delay costs. Increasing capacities is expensive, but it also shortens queues and allows projects to be processed faster, reducing the time-penalties. The goal of the firm is to manage this trade-off so as to maximize expected profit. In Sect. 3, it is shown that there is a unique set of profit-maximizing capacities, which are characterized with a formula. The formula depends in a straightforward way on urgency variables and global flows.

Next, we ask whether optimal capacities can be set, not via a top-down command system, but through a budget-balanced transfer mechanism in which individual, profit-maximizing nodes choose capacities and trade in the same way that the firm as a whole trades with the outside world. In Sect. 4, it is shown that this can be done. Unfortunately, there are situations in which the transfer prices at a particular node depend on what happens elsewhere in the system. Thus, some nodes must be managed together. Nevertheless, we show in Sect. 5 that if all the projects which flow through a node are equally urgent, then the transfer prices it faces depend only on local information, and hence the node may be managed autonomously. We note that there is a way of looking at this fact from the perspective of system design. If one wants to have nodes managed locally and autonomously, flows of projects should be divided by urgency, with each flow having its own dedicated nodes.

The importance of these results is that aggregation of several nodes under the same management is needed only if the situation is fairly complicated: certain components must handle projects of different urgencies. Business divisions consisting of multiple nodes may then be viewed as mechanisms for managing this complexity. It follows that managing multiple urgencies or priorities is a more difficult organizational task than managing similar priority tasks, as the latter type of management can be decentralized.

Finally, the analysis is applied to the problem of forecasting how changes in demands and prices affect optimal capacities and how quickly the firm should be prepared to react to such changes.

1.1 Related literature

We view the insights of this paper as complementary to several recent papers. Since Radner (1993), many studies have focused on how to design organizations which process general flows, such as flows of information or tasks requiring special skills or knowledge. Beggs (2001) considers a queuing model of task flow in an organization which is assumed to be organized hierarchically. Abstracting from incentive problems, this study derives optimal organizational structures and distributions of activities. We believe these insights are valuable, especially in organizing individual business units, where the particles in the model are people (who report to superiors in a traditional hierarchy). This model seems less applicable to the more complex organizational structure of modern, multicomponent corporations; here, relationships between different units are characterized by many interdependencies, and no clear top-down structure can be assumed. Thus, our paper is complementary to Beggs (2001) in that it considers optimal interaction on a larger scale.

Another related paper is Arenas et al. (2006). The corporation is modeled as an arbitrary network of interconnected nodes. Problems arrive from outside the network; each problem can be solved at exactly one node and must be routed there. The aim of the paper is to derive optimal network structures when delays are costly. Like Beggs (2001), this study abstracts from incentive problems, but, in contrast, it allows much more complex network relationships.

The focus of our paper is somewhat different. We do not assume that the corporation has only one stylized goal (such as routing a problem to a single solver, as above) in order to solve a system design problem. Instead, we start with a richer and more multifaceted notion of processing. The same project may need to go through many nodes before being discharged, and each node may need to do something to it. To capture this, we take the structure of processing as given. This seems reasonable since there is such a large variety of possible processing systems; the routing of projects or demands is often determined by technological constraints. While endogenizing these would be very interesting, it would depend heavily on the specific application considered, so we abstract away from these issues.

Taking the structure of processing as given permits us to consider in some depth the incentive, internal pricing, and management problems which are not analyzed in the papers discussed above. In particular, we take up the problem of transfer prices from a new perspective. Much of the work to date in this field, initiated by Hirshleifer (1956) has focused on the interaction between just two components of a firm.[2] We broaden the analysis to allow a much more complex network structure of interaction. The same is done in Adelman (2009), in which a transfer pricing system for a complex queuing network is developed. The difference between that paper and ours is that, by allowing rates of uptake to be smooth functions of available capacity and queue size, we greatly simplify the analysis and can more explicitly characterize many relationships in the system and their dependence on the primitives.

[2] See e.g., Holmstrom and Tirole (1991), Alles and Datar (1998), and Baldenius et al. (1999).

Finally, there are connections to the queuing literature. A very general model which allows for idle processors when projects are queued up is outlined in Kelly (1987, Chap. 3). Our paper extends that work by applying a tractable version of such a model to specific economic problems—especially those of transfer pricing and decentralization. There has also been work on optimal internal pricing for a single service facility (Dewan and Mendelson 1990; Ha 1998). This paper embeds the service centers in a network, allowing the study of more complex enterprises while focusing less on the details of individual queues.

2 The basic model

The model used is a variant of a common queuing model. There are N processing units, called *nodes*, whose indices compose the set $\mathcal{V} := \{1, \ldots, N\}$. The nodes process *projects*, an abstract name for whatever objects or tasks move through the system of nodes. In particular instances, these projects may be computers undergoing assembly, airline customers, administrative assignments in an office, etc. Node n has p_n projects in processing, q_n projects in its queue, and a capacity c_n, which is the largest number of projects that can be in processing simultaneously. Jobs are modeled continuously and flow through the system as a "fluid", an approach which has been used elsewhere in the queuing literature (Bramson 1996).

Certain nodes can accept projects from outside the system; this may be interpreted as taking customer orders, receiving regulatory requests for information, and so on. Let the set $\mathcal{R} \subseteq \mathcal{V}$ contain exactly those nodes which can accept inflows of projects, and call these the *receiving* nodes. The remaining nodes are called *internal*. Let r_n be the number of projects per unit time which enter the system at node n. If $n \notin \mathcal{R}$, then $r_n = 0$. Otherwise, r_n is nonnegative and is treated as a constant until Sect. 6.

Nodes can also send projects to other nodes. Let T_{nm} be the rate at which projects in processing at node n spawn projects at node m. This rate may be zero or positive, and varies only in the long run for the purposes of this model. We will assume throughout that $T_{nn} = 0$ to simplify some later expressions, though allowing the nonzero case does not change any of the results substantially. A positive value of T_{nn} means that some projects are suspended and placed back in node n's queue. Note also that under this model, a single project at node n can simultaneously spawn multiple projects at many other nodes (because $\sum_m T_{nm}$ can exceed 1). For instance, a single lawsuit causes production of multiple documents throughout the entire corporation.

We emphasize that, in our view, it is best to interpret the inflows in this model as demands or assignments—from customers, partners, regulators, litigants, and other external entities—which drive further tasks in the firm and may generate revenue. This interpretation is distinct from the understanding of inflows in a traditional supply chain, where they are simply raw materials. In our model, raw materials would be procured after the initialization of a project by sending orders to other nodes.

In addition to transferring projects to other parts of the system, each node also terminates, or discharges, a certain fraction of its projects, causing them to exit the system. Let d_n denote the fraction of projects in processing at node n that are

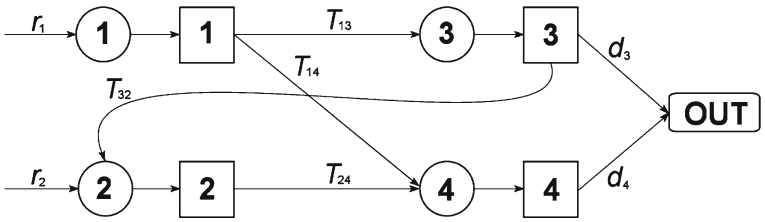

Fig. 1 A queuing network model. Queues are represented by *circles*, processors by *squares*. When there is no flow from one node to another, we assume the link between them in that direction does not exist

discharged per unit of time. Assume that $d_n > 0$ for each n. This will later guarantee that inflow does not accumulate in the system without bound. An example of the model is illustrated in Fig. 1.

This is a standard model of queuing so far.[3] What is unusual about the model is the transfer from queue to processing. In particular, this takes place at a rate that depends on the size of the queue q_n and the excess capacity $c_n - p_n$; the rate of transfer flow is

$$\frac{q_n(c_n - p_n)}{a_n q_n + b_n(c_n - p_n)},$$

where a_n and b_n are two positive constants.

This rate can be derived from the following two-stage description of the transfer mechanism. The first step is a preparation process that is required to go into processing: for instance, a customer being issued a ticket by an airline, a manager approving a programming project, or a shipping clerk signing off on a shipment. A tendency of human organizations is to become faster and more efficient at moving waiting projects to available capacity when lines are longer. At busy times, ticket agents work more quickly, appointments are scheduled more tightly, and so forth. We assume that the length of the preprocessing lag per project varies inversely with the length of the queue, i.e., it is equal to b_n/q_n, where b_n is some constant.[4] After this, processing occurs as soon as the appropriate type of service becomes available. For instance, a package being shipped remains in storage until the next pickup occurs; a customer referred to a particular kind of technical support technician must wait until the next one becomes available, etc. There are $c_n - p_n$ available servers. Assume they check for new projects at intervals having some distribution with mean a_n (the accommodation

[3] The network of processing nodes, each having its own queue, is standard in the queuing literature: see, for example, Kleinrock (1975, 147–160), Lemoine (1977, 466–467), or Bose (2001, 143–150). In each version of the standard model, a fixed transfer proportion is associated to each ordered pair of nodes; this number describes what fraction of projects in processing at the first node is transferred to the second per unit of time. In some cases, these transfer proportions are described as transfer probabilities instead, but this does not significantly alter the model. In all of the models mentioned above, the processor is assumed to be fully occupied, which makes the rate of uptake from the queue discontinuous.

[4] One explanation for this form is as follows: individual employees work under limited time horizons; for instance, their goal is to empty the projects that are queued up in the morning by the day's end, or face a costly reprimand by supervisors. Then, if the queue increases by a factor of k, it is necessary to reduce the preprocessing lag by a factor of k to achieve this goal.

delay). Then, by Little's Law (Little 1961), the average waiting time is $a_n/(c_n - p_n)$. Summing the delays and inverting to obtain the flow suffices to give the rate above, which is, of course, always less than the smaller of the preprocessing rate q_n/b_n and the accommodation rate $(c_n - p_n)/a_n$. In contrast to many other queuing models, it is smoothly differentiable in both endogenous variables q_n and p_n.

The transfer from queue to processing contrasts with existing models in which processors are fully occupied (Chandy et al. 1975; Lemoine 1977; Bitran and Dasu 1992; Bose 2001). Here, the uptake of projects into processing increases with available resources—excess capacity accelerates the movement of projects into processing, as do increases in queue size. Moreover, there is a real distinction between projects in processing and in queue. In contrast, models in which the processor is always fully occupied present no real distinction between being in processing and being in queue. In our view, excess processing capacity even when the queue is not empty is a characteristic of humans, but not of computer processors, so the model is a better model of firms (which is the intended application) than of machine networks.

The description of processing entails rates of change in the endogenous variables of

$$\dot{p}_n = -d_n p_n - p_n \sum_{m=1}^{N} T_{nm} + \frac{q_n(c_n - p_n)}{a_n q_n + b_n(c_n - p_n)}, \text{ and} \tag{1}$$

$$\dot{q}_n = r_n + \sum_{m=1}^{N} T_{mn} p_m - \frac{q_n(c_n - p_n)}{a_n q_n + b_n(c_n - p_n)}. \tag{2}$$

The flow into p_n consists exclusively of projects in the queue of node n, and by hypothesis this flow is the positive term in (1). The flow out, again by assumption, is proportional to p_n; per unit of time, the fraction d_n of the amount p_n leaves the system, and the fraction T_{nm} flows to the queue m. Similarly, projects from outside the system arrive into queue n at a rate r_n. The rest of the expression for $\dot{q}_n$ is just the flow from within the system, determined by consistency with the processor equations.

2.1 Steady state analysis

We define a steady state to be a specification, for each node, of a nonnegative process size (not greater than the capacity there) and nonnegative queue size so that the derivatives (1) and (2) are both zero.

Observe that the total rate of change in the number of projects at node n is

$$\dot{p}_n + \dot{q}_n = r_n + \sum_{m=1}^{N} T_{mn} p_m - d_n p_n - p_n \sum_{m=1}^{N} T_{nm}.$$

Thus, in the steady state, when $\dot{p}_n = \dot{q}_n = 0$, we have

$$p_n^* \left(d_n + \sum_{m=1}^{N} T_{nm} \right) = r_n + \sum_{m=1}^{N} T_{mn} p_m^*, \tag{3}$$

where p_n^* is the steady state process size at node n. For each node n, define a constant $\mu_n := d_n + \sum_{m=1}^{N} T_{nm}$. This shorthand transforms (3) into

$$\mu_n p_n^* = r_n + \sum_{m=1}^{N} T_{mn} p_m^*. \tag{4}$$

The quantity μ_n can be interpreted as the total rate of outflow from node n divided by the number of projects at node n. Defining $\mathbf{M}$ as the $N \times N$ diagonal matrix with μ_n in the (n, n) position for each n, the previous equation becomes[5]

$$\left(\mathbf{M} - \mathbf{T}^{\mathrm{T}}\right) \mathbf{p}^* = \mathbf{r}. \tag{5}$$

As the proof of Theorem 1 will show, the matrix $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$ is invertible, and the inverse has nonnegative entries, which gives a nonnegative solution for the steady state process sizes:

$$\mathbf{p}^* = \left(\mathbf{M} - \mathbf{T}^{\mathrm{T}}\right)^{-1} \mathbf{r}. \tag{6}$$

We will assume, without loss of generality, that $p_n^* \neq 0$ for each n; any nodes failing this condition may simply be neglected in the analysis, since they do not have any effect on the system.

An important simplification is embodied in this expression: the steady state processing levels are independent of the capacities, assuming the capacities are large enough for a steady state to exist at all. This may seem peculiar but is actually quite sensible. In the steady state, the inflow must equal the outflow at each node, and these are not affected by the capacities. If the capacities are less than the steady state process size values, then no steady state exists.

Note also that the matrix in (6) contains interpretable information about the system. Write $\mathbf{S} := \left(\mathbf{M} - \mathbf{T}^{\mathrm{T}}\right)^{-1}$ for brevity and let the entry of this matrix in the (n, m) position be denoted, as usual, S_{nm}. Then (6) entails that $S_{nm} = \partial p_n^* / \partial r_m$, i.e., the constant rate of change of the nth process size with respect to mth inflow. Additionally, we will define the *elasticity of the nth process size with respect to the mth inflow* as

$$\epsilon_{nm} := \frac{r_m}{p_n^*} \cdot \frac{\partial p_n^*}{\partial r_m} = \frac{r_m S_{nm}}{p_n^*}.$$

The elasticity will prove to be important in characterizing optimal capacities and transfer prices.

[5] Given an $N \times N$ matrix $\mathbf{X}$, we denote the entry in the (i, j) position by X_{ij}. Conversely, given an indexed set of N^2 values $\{X_{ij}\}_{1 \leq i,j \leq N}$, we denote by $\mathbf{X}$ the $N \times N$ matrix with X_{ij} in the (i, j) position. The notation is analogous for vectors, but only one index is used.

The steady state queue sizes $q_1^*, \ldots q_n^*$ can be readily determined using

$$0 = \dot{q}_n = r_n + \sum_{m=1}^{N} T_{mn} p_m^* - \frac{q_n^*(c_n - p_n^*)}{a_n q_n^* + b_n(c_n - p_n^*)}.$$

Using (4), this can be rewritten as

$$q_n^* = \frac{b_n \mu_n p_n^*(c_n - p_n^*)}{c_n - p_n^*(1 + a_n \mu_n)}. \tag{7}$$

Theorem 1 *There is a steady state if and only if* $c_n > p_n^*(1 + a_n \mu_n)$ *for each n. If so, the steady state occurs uniquely at process sizes* $\mathbf{p}^*$ *and queue sizes* $\mathbf{q}^*$ *and is locally stable.*

All proofs can be found in the Appendix. We will denote by $\mathcal{D}$ the set of N-tuples of capacities which satisfy the condition stated in the theorem, i.e., those which correspond to steady state configurations.

Our analysis in the remainder of the paper is in steady state (an assumption which we will, for brevity, not state explicitly in every result). In this, we follow the standard approach used in, e.g., Beggs (2001) and Arenas et al. (2006). Since the steady state is locally stable, we can expect it to persist once established, and so the system will spend a large fraction of the time in this regime. This justifies the attention the steady state receives in our analysis.

Taking the partial derivative of q_n^* with respect to the capacity c_n shows that steady state queue size is a strictly decreasing function of capacity if the steady state condition given in the theorem is met.

Throughout the paper, we will be interested in the question of which aspects of the corporation's operations can be managed *locally*—that is, when the optimal values of choice variables pertaining to node n can be computed using only variables labeled n and constants. Of course, what is a variable and what is a constant is a matter of definition. We take the inflows, process sizes, queue sizes, uptake parameters, and capacities to be variables, viewing them as quantities that can be adjusted in the short run. Other quantities, such as the network structure parameters T_{nm}, are taken to be constants in our analysis. This, of course, does not mean that they do not change, but that we do not endogenize their variation in this model.

In this vein, it is interesting to note that the expression for the minimal required capacity is purely local as long as the steady state process size p_n^* is known. In a stable system, this steady state process size can be directly observed. From this point, one only needs the product $a_n \mu_n$ at node n in order to compute the minimal capacity near which the queue at n becomes huge and the steady state of the entire system breaks down. The relatively modest number of inputs for this calculation is convenient in situations where capacity must be quickly reduced (perhaps due to an emergency or an unexpected increase in costs). An operator of a node—even one with information about only that node—can determine completely locally and fairly simply how low

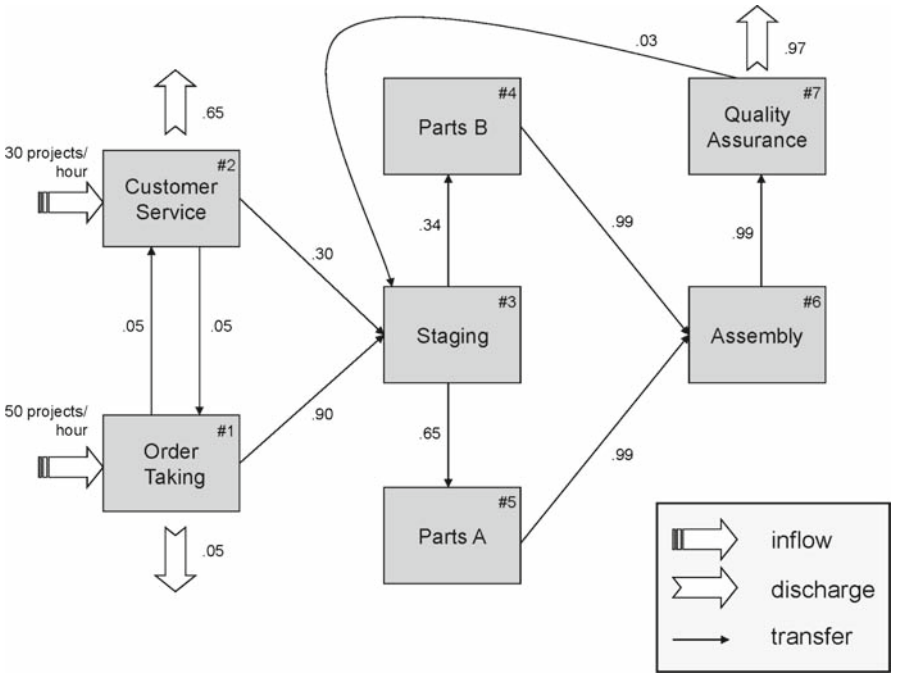

Fig. 2 A particular network which will be used to illustrate the analysis. This example corresponds to a simple version of a build-on-demand firm producing computers, which serves individual consumers. *Arrow* labels without units are the proportions of projects at the originating node flowing out per unit of time (here, the time units are hours)

the capacity can get without destroying the equilibrium for the entire system. More details on estimation procedures are in Sect. 7.

Example In Fig. 2, we present a particular example of a network which will be used to illustrate the analysis. The example represents a simple version of a firm that builds customized computers on demand and then provides customer service for those who buy them. The projects in this context are customer requests: either for new computers or for service on existing ones.

There are two receiving nodes, Order Taking and Customer Service. The former receives orders for new computers and the latter takes requests for repairs and other assistance. Of the projects that come through Customer Service, 5% are mistakes, and are actually intended for Order Taking. Similarly, 5% of the projects going through Order Taking per unit time are properly routed to the Customer Service department. Additionally, 5% of customers in processing at Order Taking decide not to buy and are discharged.

Customer Service discharges 65% of its projects after handling them. Another 30% of projects per unit time are routed to the Staging department, which performs more extensive service, as discussed below.

The projects arriving at Order Taking which are not misrouted Customer Service requests go to Staging (where they join Customer Service requests that require repairs).

Staging orders components from two Parts Procurement nodes, which fulfill the orders and route the projects to Assembly. Per unit time, 34% of the projects in staging go to Node 5, and 65% go to Node 4. Finally, Assembly sends projects to Quality Assurance. There, 97% of them are discharged, and 3% of computers are defective and are sent back to Staging for repair.

For technical reasons, all nodes that are not explicitly shown discharging projects nevertheless discharge 1% of their projects per unit time (which corresponds to projects being cancelled at some intermediate point with small probability).

The transfer matrix associated to the firm is

$$\mathbf{T} = \begin{bmatrix} 0 & 0.05 & 0.90 & 0 & 0 & 0 & 0 \\ 0.05 & 0 & 0.30 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.34 & 0.65 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.99 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.99 \\ 0 & 0 & 0.03 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The inflows are $r_1 = 50$, $r_2 = 30$, with the rest set to zero. The discharge rates are

$$\mathbf{d} = (.05, .65, \epsilon, \epsilon, \epsilon, \epsilon, .97)^{\mathrm{T}},$$

where $\epsilon = .01$ is a small parameter which is nonzero for technical reasons as noted above. Using (6), we compute

$$\mathbf{p}^* = (52, 33, 58, 20, 38, 57, 56)^{\mathrm{T}}.$$

These are the steady state process sizes.

We will return to this example again.

2.2 Completion times

While the number of projects in processing is invariant to capacity in the steady state, the size of the queue is not—increases in capacity reduce queue length and hence reduce the time to completion. Let t_n be the average total amount of time necessary to process a project that enters at node n, and let the column vector of all these values be denoted $\mathbf{t}$. In steady state, $\dot{q}_n = 0$ means that the outflow of queue n is $r_n + \sum_{m=1}^{N} T_{mn} p_m^* = \mu_n p_n^*$, where the last equality follows by (4). The amount of time a project spends in queue n is the size of the queue divided by the outflow, that is,

$$\frac{q_n^*}{\mu_n p_n^*} = \frac{b_n(c_n - p_n^*)}{c_n - p_n^*(1 + a_n \mu_n)},$$

which is obtained using (7). The amount of time a project spends in processing at node n is

$$\frac{p_n^*}{\mu_n p_n^*} = \frac{1}{\mu_n}.$$

Thus, the time spent at node n is

$$\begin{aligned}\tau_n &:= \frac{1}{\mu_n}\left(\frac{q_n^*}{p_n^*} + 1\right) \\ &= \frac{1}{\mu_n}\left[\frac{b_n\mu_n(c_n - p_n^*)}{c_n - p_n^*(1 + a_n\mu_n)} + 1\right].\end{aligned} \tag{8}$$

When processing is finished, a given project may be sent to where its processing time is assumed to be independent of what happened to it previously.

At this point we will make the assumption, to be maintained from this point forward, that the total rate of outflow per unit time at any node does not exceed the number of units in processing—that is, $\mu_n \leq 1$. This places some substantive restrictions on the processing structure. The model could also be solved without this assumption, but the formulas for completion times that we are about to derive would be more complex.

In view of the assumption, the contribution to the average total time associated with the projects sent to node m is t_m multiplied by the probability that a random project is sent there conditional on exiting, namely T_{nm}/μ_n. This gives a recursive equation for the amount of time required to process a project:

$$\begin{aligned}t_n &= \frac{1}{\mu_n}\left(\frac{q_n^*}{p_n^*} + 1\right) + \sum_{m=1}^{N}\frac{T_{nm}}{\mu_n}t_m \\ &= \frac{1}{\mu_n}\left[\frac{b_n\mu_n(c_n - p_n^*)}{c_n - p_n^*(1 + a_n\mu_n)} + 1\right] + \sum_{m=1}^{N}\frac{T_{nm}}{\mu_n}t_m.\end{aligned} \tag{9}$$

Define the column vector $\mathbf{z}$ so that, for each n,

$$z_n := \frac{b_n\mu_n(c_n - p_n^*)}{c_n - p_n^*(1 + a_n\mu_n)} + 1.$$

Then equation (9) assumes the matrix form

$$(\mathbf{M} - \mathbf{T})\mathbf{t} = \mathbf{z}.$$

By the proof of Theorem 1, $(\mathbf{M} - \mathbf{T}^{\mathrm{T}})^{-1}$ is invertible and has strictly nonnegative entries. Note that $(\mathbf{M} - \mathbf{T}^{\mathrm{T}})^{-1} = \left[(\mathbf{M} - \mathbf{T})^{\mathrm{T}}\right]^{-1} = \left[(\mathbf{M} - \mathbf{T})^{-1}\right]^{\mathrm{T}}$. Taking transposes, $(\mathbf{M}-\mathbf{T})^{-1}$ also exists and has nonnegative entries. Thus, there is a well-defined, nonnegative solution for the completion times, given by

$$\mathbf{t} = (\mathbf{M} - \mathbf{T})^{-1}\mathbf{z}. \tag{10}$$

We note a simple consequence of the formulas that is relevant when considering a merger of two organizations with the same processing structure (i.e., the same **T**). Since steady-state process sizes are linear in inflows and z_n is strictly concave in p_n^*, such a merger will always result in weak economies of scale; it will result in strict economies of scale if the ratios of process size to capacity in the constituent firms are unequal at any node, as can be verified by a direct computation. This is a quite general fact about queuing models. Intuitively, it holds because it is possible for one processor to be free while the other is occupied; merging analogous units utilizes capacity more efficiently.[6]

3 Optimal capacities

The objective of the corporation is to maximize the average profit accumulated per unit of time. Revenues consist of payments for the completion of projects that are ordered at each node; the price paid to the corporation for projects that originate at node k is ρ_k per project. Costs are of three types. The first two are costs of capacity and of queues; we assume capacity at node k has a constant marginal cost $\omega_k > 0$, and the queue has a constant marginal cost γ_k, both in units of money per project per unit of time. The third type of cost arises from delays in project completion. Assume that if a project originating at node k takes θ_k units of time to complete, the corporation pays a time premium $f_k(\theta_k)$, in units of money per project. This description entails the following profit function for the firm:

$$\pi = \sum_{k=1}^{N} \left[\rho_k r_k - \omega_k c_k - \gamma_k q_k^* - r_k \mathbb{E}(f_k(\theta_k)) \right]. \tag{11}$$

When inflows and transfer rates are fixed, the only endogenous variables in the profit function are capacity, queue size, and time to completion, so the profit may be maximized by choosing the capacities **c** to minimize the variable cost

$$K(\mathbf{c}) = \sum_{k=1}^{N} \left[\omega_k c_k + \gamma_k q_k^* + r_k \mathbb{E}(f_k(\theta_k)) \right]. \tag{12}$$

In this model, the cost of delays is linear in their length. This is relevant in the short and medium run, when immediate factors, as opposed to long term discounting, impose the most significant costs. For example, machines elsewhere sit idle while the firm produces a custom-made input or component, imposing on the corporation a per-day opportunity cost arising from the foregone output. Clients requiring a particular product must rent the next best alternative until the product is delivered. To the first order, we model these costs as a linear function of the delay time, so that $f_k(\theta_k) = \beta_k \theta_k$ for a constant β_k, which we call the *urgency* of projects originating at

[6] We are grateful to two anonymous referees for pointing out to us this simplification of our earlier arguments.

node k (or the urgency at node k for short). Then, given any distribution on completion times with mean t_k, we have $\mathbb{E}(f_k(\theta_k)) = \beta_k t_k$.

The main result of this section entails that the optimization problem facing the firm always has a unique solution with an interpretable characterization.

Lemma 1 *The global minimum of the total cost K as a function of the capacities* $\mathbf{c}$ *over the domain $\mathcal{D}$ of possible steady-state configurations occurs uniquely at a vector of capacities* $\mathbf{c}^*$*, which can be calculated explicitly from the primitives of the system. These capacities are*

$$c_n^* = p_n^*(1 + a_n\mu_n) + \mu_n\left[\frac{a_n b_n p_n^*}{\omega_n}\left(\gamma_n p_n^* + \sum_{k=1}^{N} S_{nk}\beta_k r_k\right)\right]^{1/2}. \tag{13}$$

Note that the cost-minimizing set of capacities explicitly exceeds the corresponding minimal steady state capacity as characterized in Theorem 1.

Rewriting the formula for the optimal capacities $\mathbf{c}^*$ in (13) yields an interpretable version of the condition for optimality. Using the fact that $S_{nk} = \partial p_n^*/\partial r_k$ and equation (7) we find that the capacities are optimal exactly when, for each n,

$$\omega_n = \frac{a_n}{b_n} \cdot (q_n^*)^2 \cdot \left(\gamma_n + \sum_{k=1}^{N} \epsilon_{nk}\beta_k\right).$$

That is, optimal capacities occur when, at each node n, the marginal cost of capacity is equated to the product on the right hand side. The first factor of this product is the ratio of accommodation delay to preprocessing delay—or, if the two-stage interpretation from Sect. 2 does not apply, merely a constant parameter pertaining to node n. The second factor is the queue size squared. The third factor is the marginal cost of the queue at node n plus the elasticity-weighted sum of the urgencies at all the receiving nodes. This is an attractive equation because its components are readily estimated, especially with the aid of enterprise resource planning software. Further discussion of estimation procedures can be found in Sect. 7.

The lemma entails that the optimal capacity at a given node does not depend on others' actual capacities. This has two convenient consequences. First, there are no complications arising from the fact that all the nodes are adjusting capacities simultaneously. Second, even if some nodes are constrained to be operating away from the optimum, the optimal behavior of the rest of the nodes does not change. In particular, the transfer prices that we use to decentralize the system in Sect. 5, and which induce the nodes to operate at the capacities derived above, will allow the system to achieve the second-best outcome in this constrained situation.

Example We continue the example introduced in Sect. 2. First, we specify the uptake parameters $\mathbf{a}$ and $\mathbf{b}$, as well as costs. After discussing what these quantities mean in the context of the example, we use them to calculate optimal capacities.

We set $\mathbf{a}$ and $\mathbf{b}$ as shown in Table 1. Recall that a_n is the time it takes takes a typical server at node n to check for a new project. This time is short for Order Taking and

Table 1 The accommodation and preprocessing delays

Node	Accommodation delay, **a**	Preprocessing delay, **b**
Order taking	0.01	0.01
Customer service	0.01	2
Staging	0.10	0.01
Parts A	0.10	0.01
Parts B	0.10	0.01
Assembly	0.15	5
Quality assurance	0.15	5

Table 2 The capacity and queuing costs

Node	Capacity cost, $\boldsymbol{\omega}$	Queuing cost, $\boldsymbol{\gamma}$
Order taking	1	1000
Customer service	10	1000
Staging	20	0.1
Parts A	20	0.1
Parts B	20	0.1
Assembly	50	0.1
Quality assurance	35	0.1

Customer Service (under a minute), and 6 min at Staging and Parts Procurement. It is longer at Assembly and Quality Assurance—specialized workers check for new assignments about every 10 min when idle.

The parameters b_n correspond to preprocessing delay. When b_n is nonnegligible, we set it so that the average preprocessing delay when the queue is on the order of 100 projects is on the order of minutes. Preprocessing is fairly fast at Customer Service (a little over a minute per project when there are 100 projects queued). It is longer at Assembly and Quality Assurance (3 min per project when 100 projects are queued). Preprocessing is essentially instantaneous at Order Taking, Staging, and Parts Procurement, where it is automated.

We briefly discuss what preprocessing means in the context of the nodes in this example. At Customer Service, it consists of identifying the type of problem a customer is having and routing the customer to a specialist on that kind of problem within the Customer Service node. At Assembly, preprocessing consists of reviewing the customer's order to verify that it makes sense and assigning it to an appropriate assembly location. At Quality Assurance, preprocessing consists of moving a computer to an expert who will test it.

Next, we define the revenues per project. Set $\rho_1 = 1500$ and $\rho_2 = 0$. That is, customers pay \$1500 for the average new computer, and customer service is free. Urgencies are $\beta_1 = 2$ and $\beta_2 = 3$—it costs the corporation \$48 per day in lost future business to delay a computer delivery and \$72 per day to delay a customer service request.

The costs of capacity and queuing are set as shown in Table 2. The entries of the vector $\boldsymbol{\omega}$ are the hourly costs of employing one sever at each node. A computer which

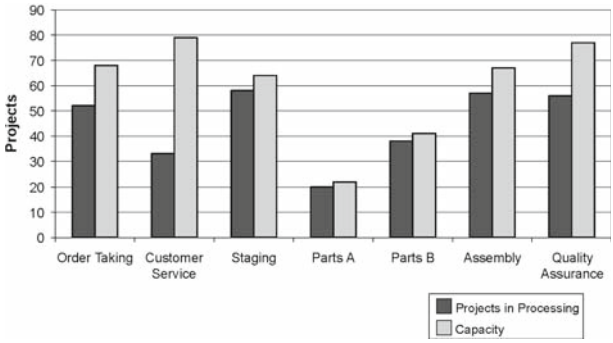

Fig. 3 The process sizes and optimal capacities in the example

Table 3 The process sizes, optimal capacities, and queue sizes in the example

Node	Steady state process size	Optimal capacity	Percentage of capacity unused (%)	Steady state queue
Order taking	52	68	24	1
Customer service	33	79	59	66
Staging	58	64	10	6
Parts A	20	22	10	2
Parts B	38	41	10	4
Assembly	57	67	25	514
Quality assurance	56	77	27	472

takes orders costs \$1 per hour per project, whereas a skilled assembly worker has wages of \$50 per hour.

The queuing costs γ_1 and γ_2 correspond to the costs in dollars (per project per hour) of keeping customers waiting in queues at Order Taking and Customer Service. We posit that keeping a customer waiting on the telephone for a fraction x of an hour will result in \$1000x lost future profits for the firm. The other queuing costs are quite low, corresponding to the cost (per project per hour) of maintaining warehouses where partially assembled computers wait. Note that queuing costs can be of two types—opportunity costs to the firm of making customers wait in a particular line, or physical maintenance costs when queues are just warehoused merchandise. The reason that the first type of cost is not included in the urgency parameters is that it is not a cost of delaying *project completion*, but the cost of making a customer wait in a *particular queue*. That is, a customer may not mind if the project takes 3 days to finish and ship, but may object to being put on hold on the telephone for 30 min.

We can now compute $\mathbf{c}^*$ using (13). The steady state process sizes (computed in Sect. 2) and capacities are plotted in Fig. 3 and recorded in Table 3, along with the queue sizes.

Note that the capacities can be significantly larger than the process sizes at the same nodes. This may seem odd in light of the fact that capacity is often quite expensive. The

explanation is that idle processors do not check for new projects instantly. To avoid bottlenecks, there must be enough available processors at any given time to ensure a sufficiently high rate of uptake from the queue.

Finally, we note that the system in this example is amenable to a simple rule of thumb. Suppose that the percent excess capacity is held equal at all nodes: that is, the ratio c_n/p_n is constrained to be the same for all n. Optimizing numerically over this ratio, we find that profit is maximized when each node has 40% of its capacity unused at any given moment. The variable cost per day—a quantity we denoted by $K(\mathbf{c})$ in the proof of Lemma 1—is then only 2.0% higher than what it is when all capacities are exactly optimal.

We will return to this example again.

4 Transfer prices

The modern corporation is an aggregation of semi-autonomous business units, which trade with each other as well as with the outside world. The problem of transfer pricing is to identify prices for the internal node-to-node transactions so that the decisions made by the business units are efficient. In the context of the present model, we ask when a balanced transfer pricing scheme exists so that each node, if operated as an autonomous, profit-maximizing business unit, chooses efficient capacities. By a balanced scheme, we mean a mechanism such that if node m pays an amount α for some task performed by node n, then node n receives α, without any corporate division absorbing surpluses or compensating for shortfalls. It turns out that such a pricing system always exists, and, surprisingly, the transfer prices are independent of the uptake parameters at the individual nodes.

Let node m pay node n a price λ_{mn} per project sent from node m to node n, minus a penalty for delays at node n in processing the assignment. This penalty accumulates at the rate χ_n per unit of time per project, which we will call *the internal price of time* at node n. Node n pays for its own capacity and queue. Moreover, it receives the price ρ_n per project that enters at node n from the outside, minus the cost of the delay to the corporation: namely, $\beta_n\theta_n$ for a project that takes θ_n to exit the system entirely. Since the mean time to completion for such projects is t_n, the expected time premium is $\beta_n t_n$ per project. For the purpose of discussing internal transfer prices at node n, we can clearly assume that there is some m such that $T_{mn} \neq 0$. If not, then no internal transfer prices are needed at node n, since it receives no inflow from within the system.

Each node is assumed to maximize the expected local profit per unit of time. The description above entails that this profit is

$$\pi_n = \rho_n r_n - \beta_n r_n t_n - \omega_n c_n - \gamma_n q_n^* + \sum_{m=1}^{N} \left(\lambda_{mn} T_{mn} p_m^* - \chi_n \tau_n T_{mn} p_m^*\right) - \sum_{m=1}^{N} \left(\lambda_{nm} T_{nm} p_n^* - \chi_m \tau_m T_{nm} p_n^*\right). \tag{14}$$

Since node n has control only over the capacity c_n, maximizing π_n is equivalent to minimizing the sum of the costs at node n that depend on c_n, which we will call K_n. Note that

$$\begin{aligned} K_n(c_n) &= \omega_n c_n + \gamma_n q_n^* + \beta_n r_n t_n + \sum_{m=1}^{N} \chi_n \tau_n T_{mn} p_m^* \\ &= \omega_n c_n + \gamma_n q_n^* + \beta_n r_n t_n + \chi_n \left(\mu_n p_n^* - r_n\right) \tau_n, \end{aligned} \tag{15}$$

where we have used the fact that $T_{nn} = 0$ and (4).

Optimal internal prices of time χ_n would equate the capacity $c_n^\dagger$ which minimizes the local cost K_n (if such a capacity exists) to c_n^*, the capacity at node n in the solution to the global cost minimization problem, which we found in Sect. 3. It turns out that there is indeed a unique capacity $c_n^\dagger$ which minimizes local cost, and there are prices that make it equal to the globally optimal capacity c_n^*. This is the content of the following result.

Theorem 2 *Globally optimal capacities are locally optimal if and only if*

$$\chi_n = \frac{\sum_{k \neq n} \epsilon_{nk} \beta_k}{1 - r_n/(\mu_n p_n^*)}. \tag{16}$$

These prices create a balanced transfer mechanism within the corporation.

The formula for the prices on time immediately yields the following.

Corollary 1 *The prices on time which induce optimal capacity choice are always invariant to the uptake parameters* **a** *and* **b**, *as well as to all the capacities.*

As a consequence, individual operators can restore optimal capacities in response to changes in uptake speeds at their nodes without any outside interference, and upper management can ignore uptake rates altogether. This is felicitous because it corroborates a commonly cited reason for granting autonomy to managers of business units: their ability to deal independently and efficiently with local adjustments.

The formula in Theorem 2 for the optimal internal price of time at node n is interpretable. The numerator of the right hand side of (16) is the elasticity-weighted sum of the urgencies at nodes other than n. To interpret the denominator, recall from (4) that $\mu_n p_n^*$ is the total rate of flow through node n, so that $r_n/(\mu_n p_n^*)$ is the fraction of inflow that enters node n from outside the system. Therefore, the denominator is simply the fraction of inflow coming into node n from inside the system.

The quantities involved in the formula for transfer prices are often quite accessible. Elasticities and urgencies are observable, especially using enterprise resource planning software, as is the fraction of tasks coming into a certain node from inside the system. Thus, the transfer prices are a particularly implementable result of this analysis.

On the other hand, it is important to note that these quantities must be observed near steady state, and should be available to the authority setting the transfer prices. While this is often possible, it is a drawback of the mechanism described here if the

optimal transfer prices depend on many local variables.[7] In the next section, we consider the case where setting the prices requires minimal information to be available to the central authority.

5 Decentralization

In this section, we will be working in the regime of the short or medium term, in which the transfer rates $\mathbf{T}$ and urgencies remain fixed, but the inflows $\mathbf{r}$ can vary. At any given node, the optimal transfer prices obtained above depend on the entire vector $\mathbf{r}$ of inputs. Consequently, the setting and updating of transfer prices cannot generally be accomplished locally, but instead depends on global information. In this section, we consider when the transfer prices facing a particular node depend only on local information. In such cases, we will say that the node can be managed locally.

Another definition will help to streamline the statement of a sufficient condition for decentralization. We say node m *influences* node n if $\partial p_n^*/\partial r_m = S_{nm} \neq 0$, i.e., if changes in the inflow at n affect the process size at m. There is an easy way to decide whether one node influences another.

Lemma 2 *Node m influences node n (such that $n \neq m$) if and only if there is a sequence of nodes*

$$n_1 = m, n_2, n_3, \ldots, n_\ell = n$$

such that $T_{n_j n_{j+1}} > 0$ for all $1 \leq j < \ell$. That is, node m influences node n if and only if, in the weighted directed graph corresponding to $\mathbf{T}$, there is a directed path with positive edge weights from node m to node n.

We can now state the main result:

Theorem 3 *A node can be managed locally if all the receiving nodes that influence it have the same urgency.*

If all the receiving nodes which influence node n have the urgency $\widehat{\beta}_n$, then the proof of the theorem shows that

$$\chi_n = \frac{\widehat{\beta}_n - \epsilon_{nn}\beta_n}{1 - r_n/(\mu_n p_n^*)}.$$

If, moreover, node n is internal, so that $r_n = 0$, it follows that $\epsilon_{nn} = 0$ and the above expression takes on the particularly simple form $\chi_n = \widehat{\beta}_n$. That is, the internal price of time at an internal node is just the urgency at the nodes that influence it.

This suggests that when the internal price of time at node n differs from the urgency of the nodes that influence it, the difference exists to account for the additional incentives due to the projects entering node n from outside the system. In particular, recall

[7] For a detailed treatment of incentive compatibility as it relates to the transfer pricing problem, see Groves and Loeb (1979).

that when a project enters the system at node n, the node gets paid for the completed project, minus a penalty that depends on how long the project takes to exit the system entirely. This penalty provides an incentive for node n to reduce its own contribution to the total time. This *external* incentive to work faster, arising from the penalty charged by the outside customer, reduces the *internal* time costs which must be imposed to get the node to work at the correct speed.

More broadly, these results yield insights about aggregating multiple nodes into business divisions. In particular, Theorem 3 shows that such aggregation is needed only if inflows of projects with different urgencies influence the same node—that is, if a single supply chain handles tasks of varying urgency. In this case, managing many nodes together may be necessary, because this allows the operator to adjust the price of time at each node in response to changes elsewhere throughout the system. On the other hand, when all the receiving nodes that influence a given node have the same urgency, all the information needed to set the internal price of time is transmitted through the local process size—and for internal nodes, even this is not needed.

There is a way of looking at this result from the perspective of system design. If one wants to have nodes managed locally, flows of projects should be divided by priority; projects with a given urgency should go through nodes that handle only projects with the same urgency.

We should note that the steady-state process sizes and other local variables must still be verifiable (at least occasionally) by the corporation's management to ensure that divisions are using transfer prices set in accordance with the policy. However, as mentioned in the introduction, our main concern is with informational efficiency—when does optimal pricing require minimal information aggregation? This is the question that the theorem answers.

Theorem 3 can also be interpreted to say that managing companies engaged in tasks of differing urgencies is harder than managing companies with a single urgency level, for management can be decentralized in the latter but not generally in the former. This, in turn, provides a formal basis for Michael Porter's (1985) admonition against companies becoming "stuck in the middle" by striving to serve both high-value and cost-sensitive customers. Serving cost-sensitive customers entails relentless pressure to minimize cost, while providing high value will often require fast response to customer inquiries and needs. In the context of the theory, these customer segments represent different urgencies and hence are difficult to manage together.

Lastly, we note that if urgencies are viewed as variables in the short run, then even when a node processes only one urgency, it cannot be managed completely locally. The urgency of the nodes that influence it must still be transmitted to it. In this case, the theorem shows that nodes processing only one urgency require less external information than those managing multiple urgencies—one needs to send them only information about the relevant urgency, but not about system-wide inflows.

Example Continuing the previous example, we compute the transfer prices and delays at the nodes in the system. These are shown in Table 4. Note that the prices on time are the same at all internal nodes, but vary when there is some inflow. This fits with the discussion of the incentive role of the local price of time in this section—in

Table 4 The local price of time and the delays in the system

Node	Local price of time	Time spent in queue	Time to completion
Order taking	2.77	0.01	21
Customer service	1.94	2.01	11
Assembly	2.18	0.10	–
Parts A	2.18	0.10	–
Parts B	2.18	0.10	–
Assembly	2.18	9.05	–
Quality assurance	2.18	8.39	–

particular, how it changes when the node is incentivized by time penalties from outside customers.

Mean time to completion is about a day for projects originating at Order Taking. It is a weighted average of a day and a few hours at Customer Service—recall that some projects are discharged immediately after being processed by Customer Service, while others go into the Staging node.

Since $\beta_1 \neq \beta_2$, and both Order Taking and Customer Service clearly influence all internal nodes (by Lemma 2), it is not possible to decentralize the system. Different urgencies at Order Taking and Customer Service cause externalities to each other.

6 Response to external changes

An important question facing modern enterprises is how to respond flexibly and efficiently to frequent changes in demands and prices (Fisher et al. 1994; Crawford et al. 2005; Cherbakov et al. 2005). Indeed, this challenge has driven new, service-oriented, on demand approaches to production, resulting in firms well approximated by our model. Thus, it is natural to consider how a system responds to external changes in the context of the theory.

There is a way to analyze this question using the results we have derived above. In Sect. 4, we described a mechanism to translate the global incentives facing the firm into local transfer prices. Once these prices are known, they determine each node's profit-maximizing behavior—in particular, the investments it makes in capacity. Consequently, to describe the corporation's response to external changes, it suffices to characterize how they affect the local transfer prices. Such a characterization would capture how demand shocks and price shifts alter the priorities, and hence the behavior, of each part of the firm.

Theorem 4 *The marginal change in the internal price of time at node n caused by a change in inflow at a node $m \neq n$ is*:

$$\frac{\partial \chi_n}{\partial r_m} = \frac{1}{p_n^*} \cdot \frac{\partial p_n^*}{\partial r_m} \cdot \frac{\beta_m - \chi_n}{1 - r_n/(\mu_n p_n^*)}.$$

This rate of change is directly proportional to $\beta_m - \chi_n$, which is the amount by which the urgency at node m exceeds the current internal price of time at node n.

The marginal change in the internal price of time at node n caused by a change in inflow at node n is given by

$$\frac{\partial \chi_n}{\partial r_n} = \frac{1}{p_n^*} \cdot \frac{\chi_n (1/\mu_n - \partial p_n^*/\partial r_n)}{1 - r_n/(\mu_n p_n^*)}.$$

The internal price of time at node n is nonincreasing in inflow at node n.

First, this theorem shows that the internal prices on time obtained in Sect. 4 are quite natural. Increasing the inflow of projects (at a location other than node n) that have a higher urgency than the current internal price of time at node n increases the price of time at node n. On the other hand, increasing the inflow of projects that have a lower urgency than the current internal price of time at node n decreases the price of time at this node. Moreover, the magnitude of the change in each case is proportional to the magnitude of the difference between the current internal price of time and the urgency of the increasing flow.

The reason for the second part of the theorem—that the internal price of time at node n is nonincreasing in inflow at node n—is similar to the explanation after Theorem 3. Increasing the inflow at node n increases the external incentives to work quickly (since the node has to pay the corporation's delay costs for a larger share of projects), which reduces the need for high internal prices on delays.

Theorem 5 *The marginal change in the internal price of time at node n caused by a change in urgency at a node $m \neq n$ is*

$$\frac{\partial \chi_n}{\partial \beta_m} = \frac{\epsilon_{nm}}{1 - r_n/(\mu_n p_n^*)}.$$

The marginal change in the internal price of time at node n caused by a change in urgency at node n is given by

$$\frac{\partial \chi_n}{\partial \beta_n} = 0.$$

These formulas are interpretable. The quantity $\partial \chi_n / \partial \beta_m$ is just the elasticity of the nth process size with respect to the mth inflow divided by the fraction of the projects that enter node n from within the system. It is also easy to see why χ_n should be invariant to β_n. Since all the extra delay costs arising from increases in the urgency at node n are already paid by node n to the outside customer ordering the project, it is not necessary for this extra incentive to be conveyed through internal transfer prices.

7 Remarks on implementation of the analysis

To test and apply this model empirically, it is necessary to obtain estimates for the vectors of input parameters: **r**, **d**, **c**, the matrix **T** of transfer rates, and the uptake

parameters **a** and **b**. Fortunately, all of these can be observed, albeit with varying degrees of difficulty.

To start the analysis, it is necessary to identify nodes. This would probably be done using a company's enterprise resource planning software, by examining the unit level of that software. These component units can, in turn, be delineated using approaches such as those outlined in Parnas (1972); Hagel and Singer (1999) and Cherbakov et al. (2005). The rate parameters, namely **r**, **d**, and the transfer proportions per unit time in the matrix **T**, can be observed simply and directly. For instance, to find the inflow rate r_n, one simply picks a convenient measuring interval and monitors how many projects flow into a given node during that time, dividing by the interval length to find the rate. To measure T_{nm}, one similarly tracks the number of projects per unit time that flow from node n to node m, dividing by the total number of projects in processing at node n. All such measurements are done many times and averaged. The capacities **c** can also be directly observed by noting the maximum number of projects that could be handled simultaneously at each node. If the values of $\mathbf{p}^*$ or $\mathbf{q}^*$ are desired as a means of testing the theory, they can often be observed by direct inspection of the number of projects being processed and waiting in line, respectively. Alternatively, $\mathbf{p}^*$, $\mathbf{q}^*$, and **c** can be estimated by fitting the model to the data generated by the flow of tasks and their resolution—i.e., processing times, profits, etc.

At several points in the analysis, we made use of the elasticities of process sizes with respect to inflows. Recall that these were defined as

$$\epsilon_{nm} := \frac{r_k}{p_n^*} \cdot \frac{\partial p_n^*}{\partial r_m}.$$

To find this quantity, it is necessary to measure or calculate $\partial p_n^*/\partial r_m$. This can be done by taking many measurements of $\Delta p_n^*/\Delta r_m$—that is, the change in the nth process size brought about by a change in tne mth inflow—for small changes in r_m. Alternatively, we can recall that $S_{nm} = \partial p_n^*/\partial r_m$, and that **S** can be obtained by inverting $\mathbf{M} - \mathbf{T}$, whose entries we have already seen how to measure.

The moderately challenging quantities to observe are the uptake parameters **a** and **b**. There are two approaches to this problem. First, consider the situation in which the description of processing given in Sect. 2 is reasonable in the context of the particular system, and the lag times can be observed. Then b_n can be found as the slope of the line which best approximates the plot of preprocessing delay versus $1/q_n$ for different configurations of the system. The parameter a_n can be determined analogously by fitting a line to the plot of accommodation delays versus $1/(c_n - p_n)$.

On the other hand, the lag times may be difficult to measure. In that case, more general fitting procedures can be used. In particular, one would collect readings of c_n, p_n, q_n, and the uptake rate in various configurations of node n, and then use standard statistical tools to find the values of a_n and b_n so that the uptake rate is well approximated by

$$\frac{q_n(c_n - p_n)}{a_n q_n + b_n(c_n - p_n)}.$$

Finally, it may be desirable to measure completion times **t** to find out whether the model produces reasonable predictions. This can be done by tracking projects and recording where and when they enter and exit the system, via ticketing or timestamping, depending on what is moving through the system.

8 Conclusion

In order to model some aspects of the problem of optimally allocating resources in a firm, we have presented a flow model of corporations based on queuing of tasks and delayed uptake. The perspective of the model is that most of the tasks performed by companies can be treated in a similar way: as generating activity of processing nodes or workstations, after some time spent in queue. Delayed uptake from the queues, because it makes the key rates smooth in the endogenous variables, actually simplifies standard queuing models dramatically.

We derived the optimal capacities in the context of the model. These satisfy an interpretable elasticity condition. The fact that the quantities in the formula are tracked using enterprise resource planning software used by corporations makes the formula a realistic starting point for testing the model empirically or for optimizing capacities.

We investigated two large questions about the model. First, when do transfer prices exist that induce autonomous node operators to choose efficient levels of investment? The surprising answer is: always. Moreover these transfer prices are independent of many of the system primitives and are characterized with a simple formula. Second, we considered when these prices depend only on local information; that is, when can the system be decentralized? Decentralization is possible when a node shares the same urgency as other nodes processing the same flow of projects. The sharing of urgency is clearly necessary for decentralization since different urgencies will cause externalities to each other. The remarkable part of the theorem is that this condition is also sufficient for decentralization.

There are several natural extensions of this project. First, we would like to consider more general forms for the flow rate from the queue to the processor. In this paper, we assumed that the flow takes a special (though fairly natural) form. This demonstrated the simplification arising from making the flow rate a smooth function of the endogenous variables and allowed several questions to be answered precisely. We think many of the benefits of this smoothing approach would carry over to a more general setting.

A second direction we would like to explore is allowing several queues to feed into the same processor, or several processors to accept flows from the same queue, or both. This would permit the analysis of system design questions—such as those tackled in Beggs (2001) and Arenas et al. (2006)—while also allowing a general, network-based specification of the processing procedure. For example, one could ask what happens when multiple internal suppliers compete to sell the same input to a node that needs it. We view this paper as a first step toward that goal, since it illustrates the solution of several optimization and incentive problems in a special case of that more general model.

Acknowledgements We are grateful to Roy Radner, seminar participants at Caltech, Berkeley, and the South-West Economic Theory Conference, and two anonymous referees for helpful comments. Golub is grateful for support from the Larson SURF Fellowship.

Appendix: Proofs of Theorems and Lemmas

Proof of Theorem 1 To show the "only if", suppose there is a steady state. We will show that the steady state process sizes are uniquely determined and nonnegative. To this end, we first claim that $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$ has a strictly dominant diagonal. That is, there is a diagonal matrix $\mathbf{D}$ (in this case the identity matrix) such that the magnitude of each diagonal entry of $\mathbf{D}\left(\mathbf{M} - \mathbf{T}^{\mathrm{T}}\right)$ exceeds the sum of the magnitudes of the other entries in its column. This is seen as follows. For each n, we have $\mu_n = d_n + \sum_{m=1}^{N} T_{nm}$, so that

$$\mu_n - T_{nn} = d_n + \sum_{m \neq n} T_{nm} > \sum_{m \neq n} T_{nm},$$

since $d_n > 0$ by hypothesis. This establishes the dominance of the diagonal. Consequently, $\left(\mathbf{M} - \mathbf{T}^{\mathrm{T}}\right)^{-1}$ exists (McKenzie 1960, Theorem 1). In addition, $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$ is a Z-matrix, i.e., a matrix all of whose off-diagonal entries are non-positive. It follows that its inverse has no negative entries (Berman and Plemmons 1979, Theorem 2.3). Along with the fact that all the inflows $\mathbf{r}$ are nonnegative, this implies that the vector on the right side of (6) is well-defined and all of its entries are nonnegative. The fact that this is the only possible steady state was argued before the statement of equation (6).

Now we show that $c_n > p_n^*(1 - a_n\mu_n)$. The only possible solution for the steady state queue sizes was deduced above in (7). Suppose, to get a contradiction, that $c_n \leq p_n^*(1 - a_n\mu_n)$ for some n. Then the denominator in the expression of (7) for q_n^* is zero or negative. In the case $p_n^* < c_n$, the expression for q_n^* is undefined or negative, which cannot happen in a steady state by definition. In the case $p_n^* > c_n$, no steady state can occur because the processors cannot accommodate the steady state process sizes. Finally, in the case $p_n = c_n$, (7) gives that q_n^* is either undefined (so we are done) or zero. But if $q_n^* = 0$, then (1) indicates that a steady state is impossible unless all the process sizes are zero, which contradicts (6), using the assumption that $\mathbf{r} \neq \mathbf{0}$ and the fact that $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$ is invertible. So $c_n \leq p_n^*(1 - a_n\mu_n)$ is impossible and the forward implication is done.

To prove the "if": if $c_n > p_n^*(1 - a_n\mu_n)$ then by inspection of (7) we have that q_n^* is well-defined and nonnegative; direct computation shows that this choice of q_n^*, along with the p_n^* of (6), results in $\dot{p}_n = \dot{q}_n = 0$ for each n, i.e., the steady state.

To prove that the steady state is locally stable, we will linearize the system by finding the Jacobian matrix of $(\dot{p}_1, \ldots, \dot{p}_n, \dot{q}_1, \ldots, \dot{q}_n)^{\mathrm{T}}$ at the steady state configuration. Defining

$$k_n = a_n \left[\frac{q_n^*}{a_n q_n^* + b_n(c_n - p_n^*)} \right]^2 \quad \text{and} \quad \ell_n = b_n \left[\frac{c_n - p_n^*}{a_n q_n^* + b_n(c_n - p_n^*)} \right]^2,$$

the Jacobian matrix is, by direct computation,

$$\mathbf{J} = \left[\begin{array}{cccc|cccc} -k_1-\mu_1 & & & & \ell_1 & & & \\ & -k_2-\mu_2 & & & & \ell_2 & & \\ & & \ddots & & & & \ddots & \\ & & & -k_N-\mu_N & & & & \ell_N \\ \hline k_1+T_{11} & T_{21} & \cdots & T_{N1} & -\ell_1 & & & \\ T_{12} & k_2+T_{22} & \cdots & T_{N2} & & -\ell_2 & & \\ \vdots & \vdots & \ddots & \vdots & & & \ddots & \\ T_{1N} & T_{2N} & \cdots & k_N+T_{NN} & & & & -\ell_N \end{array}\right],$$

where the blank regions are filled with zeros. We will show the matrix has a strictly dominant diagonal. To this end, we claim there exist constants $\xi_1, \ldots, \xi_{2N}$ such that the following inequality holds:

$$\xi_m |J_{mm}| > \sum_{n=1}^{2N} \xi_n |J_{nm}| \quad \text{for each } m \in \{1, \ldots 2N\}. \tag{17}$$

First, observe that for each $m \in \mathcal{V}$, we have $k_m + \mu_m > k_m + \sum_{n=1}^{N} T_{mn}$ because, by definition, $\mu_m = d_m + \sum_{n=1}^{N} T_{mn} > \sum_{n=1}^{N} T_{mn}$; the last inequality uses the assumption $d_m > 0$. Then there are constants $\xi_1, \ldots, \xi_N < 1$ such that, for each $m \in \mathcal{V}$, $\xi_m(k_m + \mu_m) > k_m + \sum_{n=1}^{N} T_{mn}$. Setting $\xi_{N+1} = \cdots = \xi_{2N} = 1$ satisfies (17). Consequently, $\mathbf{J}$ is negative definite (McKenzie 1960, Theorem 2). This implies that the system is locally stable around the steady state (Jordan and Smith 1999, Theorem 8.15). □

Proof of Lemma 1 Expanding (12) using (7) and (10) yields

$$K(\mathbf{c}) = \sum_{k=1}^{N} \left(\omega_k c_k + \gamma_k \frac{b_k \mu_k p_k^* (c_k - p_k^*)}{c_k - p_k^*(1 + a_k \mu_k)} + \beta_k r_k \sum_{m=1}^{N} S_{mk} \left[\frac{b_m \mu_m (c_m - p_m^*)}{c_m - p_m^*(1 + a_m \mu_m)} + 1 \right] \right). \tag{18}$$

Optimal capacities $c_1^*, \ldots, c_n^*$ which minimize K, if they exist, must satisfy for each n the first order condition:

$$0 = \frac{\partial K}{\partial c_n} = \omega_n - \frac{a_n b_n \gamma_n \left(\mu_n p_n^*\right)^2}{\left[c_n - p_n^*(1 + a_n \mu_n)\right]^2} - \frac{a_n b_n \mu_n^2 p_n^*}{\left[c_n - p_n^*(1 + a_n \mu_n)\right]^2} \sum_{k=1}^{N} S_{nk} \beta_k r_k.$$

The equation reflects the fact that p_n^* is determined independently of capacity. We solve these equations to obtain

$$c_n^* = p_n^*(1 + a_n\mu_n) + \mu_n\sqrt{\frac{a_n b_n p_n^*}{\omega_n}\left(\gamma_n p_n^* + \sum_{k=1}^{N} S_{nk}\beta_k r_k\right)}.$$

Let $\mathbf{c}^*$ denote the column vector of these capacities.

Note that the expression whose square root is taken above is obviously a nonnegative number and so c_n^* is well-defined. Write $\hat{c}_n := p_n^*(1 + a_n\mu_n)$. Pick an arbitrary $\mathbf{x} \in \mathcal{D}$. Define the compact set

$$\Omega_{\delta,\Delta} := \left[\hat{c}_1 + \delta, \hat{c}_1 + \Delta\right] \times \cdots \times \left[\hat{c}_N + \delta, \hat{c}_N + \Delta\right].$$

We claim $\delta > 0$ can be chosen sufficiently small and Δ can be chosen sufficiently large so that all the following conditions hold:

1. $\mathbf{x} \in \Omega_{\delta,\Delta}$;
2. $\mathbf{c}^* \in \Omega_{\delta,\Delta}$;
3. $K(\mathbf{c}) > K(\mathbf{c}^*)$ for all $\mathbf{c} \in \partial\Omega_{\delta,\Delta}$.

It is obvious how the first two conditions can be simultaneously satisfied. To get the third, note that for any point $\mathbf{c} \in \partial\Omega_{\delta,\Delta}$, there exists an n such that $c_n = \hat{c}_n + \Delta$ or $c_n = \hat{c}_n + \delta$. In the former case, Δ can be chosen sufficiently large so that the $\omega_n c_n$ term in (18) is larger than $K(\mathbf{c}^*)$. In the latter case, δ can be chosen small enough to make the denominator $c_n - p_n^*(1 + a_n\mu_n)$ in (18) arbitrarily small, so that the associated $\beta_n r_n t_n$ term of the total cost K is arbitrarily large, and in particular larger than $K(\mathbf{c}^*)$. Indeed, we can ensure these things for all n simultaneously. Since all the terms being summed in (18) are nonnegative, this suffices to show that condition 3 can be satisfied. Finally, observe that the condition can be satisfied without breaking either of its predecessors by simply taking the smallest δ and the largest Δ required by any of the three conditions.

Since the cost K is a continuously differentiable function of the capacities $\mathbf{c}$, the Weierstrass extreme value theorem guarantees that all global minima of K over the compact set $\Omega_{\delta,\Delta}$ occur either at critical points of K that lie inside $\Omega_{\delta,\Delta}$, or otherwise on the boundary $\partial\Omega_{\delta,\Delta}$. By the first paragraph of this proof, and condition 2 above, there is exactly one critical point in $\Omega_{\delta,\Delta}$, namely $\mathbf{c}^*$. Condition 3 ensures that the global minimum of K on $\Omega_{\delta,\Delta}$ does not occur on the boundary, so it must occur at $\mathbf{c}^*$. Hence, by definition of the global maximum, $K(\mathbf{c}^*) \leq K(\mathbf{x})$. The argument works for an arbitrary fixed $\mathbf{x} \in \mathcal{D}$, so a global minimum of K over $\mathcal{D}$ occurs at $\mathbf{c}^*$. The first paragraph of the proof entails that at all other possible steady state capacities, $\partial K/\partial c_n$ is nonzero for some n, so none of these points can be even local minima, and this establishes that the global minimum occurs at exactly one point in $\mathcal{D}$, namely at $\mathbf{c}^*$. □

Proof of Theorem 2 Expanding q_n^*, t_n, and τ_n in (15) using (7), (8), and (10) gives

$$K_n(c_n) = \omega_n c_n + \gamma_n \frac{b_n \mu_n p_n^*(c_n - p_n^*)}{c_n - p_n^*(1 + a_n \mu_n)} + \beta_n r_n \sum_{m=1}^{N} S_{mn} \left[\frac{b_m \mu_m (c_m - p_m^*)}{c_m - p_m^*(1 + a_m \mu_m)} + 1 \right]$$
$$+ \chi_n (\mu_n p_n^* - r_n) \cdot \frac{1}{\mu_n} \left[\frac{b_n \mu_n (c_n - p_n^*)}{c_n - p_n^*(1 + a_n \mu_n)} + 1 \right].$$

The first-order condition for cost minimization is

$$0 = \frac{\partial K_n}{\partial c_n} = \omega_n - \frac{a_n b_n \mu_n p_n^* \left[\gamma_n \mu_n p_n^* + \beta_n r_n S_{nn} \mu_n + \chi_n (\mu_n p_n^* - r_n) \right]}{\left[c_n - p_n^*(1 + a_n \mu_n) \right]^2}.$$

From this, we find that the capacity $c_n^\dagger$ which solves the first order conditions is

$$c_n^\dagger = p_n^*(1 + a_n \mu_n) + \sqrt{\frac{a_n b_n \mu_n p_n^* \left[(\gamma_n + \chi_n) \mu_n p_n^* + \beta_n \mu_n r_n S_{nn} - \chi_n r_n \right]}{\omega_n}}. \quad (19)$$

The one-dimensional case of the proof of Lemma 1 establishes that this is the capacity corresponding to the unique global minimum of the local cost function K_n.

As mentioned in the text, the local operators will maintain optimality if and only if the local prices χ_n are set so that the locally optimal capacity $c_n^\dagger$ matches the globally optimal capacity c_n^*. Solving $c_n^\dagger = c_n^*$ using expressions (13) and (19) yields the formula for χ_n.

The system of transfer prices is balanced by the description of payments among nodes: the amount sent by node m for node n's services is exactly the amount received by node n.

Finally, we verify that the expression for χ_n is well defined. It suffices to verify that the denominator in (16) is nonzero. Note that, using (4),

$$1 - \frac{r_n}{\mu_n p_n^*} = \frac{\sum_{m=1}^{N} T_{mn} p_m^*}{r_n + \sum_{m=1}^{N} T_{mn} p_m^*}.$$

By assumptions stated in Sect. 2, the process size p_m^* is nonzero for each m, so the numerator is nonzero. By an assumption stated at the beginning of Sect. 4, T_{mn} is nonzero for some m, so that the denominator is nonzero. □

Proof of Lemma 2 Recall from Sect. 2 that $S_{nm} = \partial p_n^* / \partial r_m$, where $\mathbf{S} = \left(\mathbf{M} - \mathbf{T}^{\mathrm{T}} \right)^{-1}$. We will show that $S_{nm} > 0$ if and only if there is a sequence of nodes

$$n_1 = m, n_2, n_3, \ldots, n_\ell = n$$

such that $T_{n_j n_{j+1}} > 0$ for all $1 \leq j < \ell$.

First, we can find an $s > 0$ and a nonnegative matrix $\mathbf{B}$ such that

$$\mathbf{M} - \mathbf{T}^{\mathrm{T}} = s\mathbf{I} - \mathbf{B}.$$

Moreover, all off-diagonal entries of $\mathbf{B}$ match those of $\mathbf{T}^{\mathrm{T}}$, and we can choose s and $\mathbf{B}$ so that $\mathbf{B}$ has only positive entries on its diagonal. By Berman and Plemmons (1979, p. 138), the diagonal dominance of $\mathbf{M} - \mathbf{T}$ ensures that the spectral radius of $\mathbf{B}$ is at most s. Then by a standard result on the Neumann series (Meyer 2000, p. 618), we may write

$$\mathbf{S} = \frac{1}{s} \cdot \sum_{k=0}^{\infty} (\mathbf{B}/s)^k . \tag{20}$$

Denote by $B_{nm}^{(k)}$ the entry in the (n, m) position of $\mathbf{B}^k$. By (20), S_{nm} is nonzero if and only if $B_{nm}^{(k)}$ is nonzero for some k.

By a simple induction argument,

$$B_{nm}^{(k)} = \sum_{n_2,\ldots,n_{k-1} \in \mathcal{V}} B_{nn_2} B_{n_2 n_3} \cdots B_{n_{k-1} m}.$$

Since the off-diagonal entries of $\mathbf{B}$ match those of $\mathbf{T}^{\mathrm{T}}$ (and by our conditions on $\mathbf{B}$), this sum is nonzero if and only if

$$\sum_{n_2,\ldots,n_{k-1} \in \mathcal{V}} T_{m n_{k-1}} T_{n_{k-2} n_{k-3}} \cdots T_{n_2 n}$$

is nonzero, where we have reversed the order of indices (because of transposition) and the order of the factors in each summand. This sum is nonzero for some k if and only if the condition in the statement of the lemma holds. □

Proof of Theorem 3 Suppose the hypothesis holds. Consider formula (16), which we rewrite as

$$\chi_n = \frac{\left(\sum_{k=1}^{N} S_{nk} \beta_k r_k\right)/p_n^* - \epsilon_{nn} \beta_n}{1 - r_n/(\mu_n p_n^*)} \tag{21}$$

Whenever $r_k \neq 0$ and $S_{nk} \neq 0$, β_k is equal to the common urgency of the nodes that influence node n; we will call this common urgency $\widehat{\beta}_n$. Then

$$\sum_{k=1}^{N} S_{nk} \beta_k r_k = \widehat{\beta}_n \sum_{k=1}^{N} S_{nk} r_k = \widehat{\beta}_n p_n^*,$$

where the last equality follows from the defintion of $\mathbf{S}$ and from (6). Then, simplifying (21), we obtain

$$\chi_n = \frac{\widehat{\beta}_n - \epsilon_{nn}\beta_n}{1 - r_n/(\mu_n p_n^*)}.$$

Since only constants and local parameters appear on the right, node n can be managed separately from the rest of the system. □

Proof of Theorem 4 A tedious but straightforward calculation shows that if $m \neq n$, then

$$\frac{\partial \chi_n}{\partial r_m} = \frac{1}{p_n^*} \cdot \frac{\partial p_n^*}{\partial r_m} \cdot \frac{\beta_m - \chi_n}{1 - r_n/(\mu_n p_n^*)}.$$

From this, the claims about proportionality relationships follow immediately.

Moreover,

$$\frac{\partial \chi_n}{\partial r_n} = \frac{1}{p_n^*} \cdot \frac{\chi_n(1/\mu_n - \partial p_n^*/\partial r_n)}{1 - r_n/(\mu_n p_n^*)}.$$

To show this quantity is not positive, it suffices to establish that $\frac{1}{\mu_n} - \frac{\partial p_n^*}{\partial r_n} \leq 0$ or, equivalently, $\frac{\partial p_n^*}{\partial r_n} \cdot \mu_n = S_{nn}\mu_n \geq 1$. Note that μ_n is the entry in the (n, n) position of the matrix $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$, using the assumption that $T_{nn} = 0$. Since $\mathbf{S}^{\mathrm{T}}$ and $\mathbf{M} - \mathbf{T}^{\mathrm{T}}$ are inverses, it follows that their product is the identity matrix. In particular, writing out explicitly the expression for the entry in the (n, n) position of their product, we get $1 = S_{nn}\mu_n - \sum_{m \neq n} S_{nm}T_{mn}$. Since $\mathbf{S}$ has no negative entries by the proof of Theorem 1, it follows that the summation is nonnegative and so $S_{nn}\mu_n \geq 1$, as desired. □

References

Adelman D (2009) Price-directed control of a closed logistics queueing network. Oper Res (to appear)

Alles M, Datar S (1998) Strategic transfer pricing. Manage Sci 44(4):451–461

Arenas A, Cabrales A, Danon L, Díaz-Guilera A, Guimerá R, Vega-Redondo F (2006) Optimal information transmission in organizations: search and congestion. Forthcoming, available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=433861

Baldenius T, Reichelstein S, Sahay S (1999) On the economics of transfer pricing. Rev Account Stud 4(2):67–91

Beggs AW (2001) Queues and hierarchies. Rev Econ Stud 68:297–322

Berman A, Plemmons RJ (1979) Nonnegative matrices in the mathematical sciences. Academic Press, New York

Bitran GR, Dasu S (1992) A review of open queueing network models of manufacturing systems. Queueing Syst 12(1–2):95–134

Bose SK (2001) An introduction to queueing systems. Kluwer/Plenum, New York

Bramson M (1996) Convergence to equilibria of certain fifo and processor sharing queuing networks. In: Kelly FP, Zachary S, Ziedins I (eds) Stochastic networks: theory and applications. Oxford University Press, Oxford, pp 1–18

Brandenburger AM, Nalebuff BJ (1996) Co-opetition. Doubleday, New York

Chandy KM, Herzog U, Woo LS (1975) Approximate analysis of general queuing networks. IBM J Res Dev 19(1):43–49
Cherbakov L, Galambos G, Harishankar R, Kalyana S, Rackham G (2005) Impact of service orientation at the business level. IBM Syst J 44(4):653–668
Crawford CH, Bate GP, Cherbakov L, Holley K, Tsocanos C (2005) Toward an on demand service-oriented architecture. IBM Syst J 44(1):81–108
Dewan Sanjeev, Mendelson Haim (1990) User delay costs and internal pricing for a service facility. Manage Sci 36(12):1502–1517
Fisher ML, Hammond JH, Obermeyer WR, Raman A (1994) Making supply meet demand in an uncertain world. Harv Bus Rev 72(3):83–93
Groves T, Loeb M (1979) Incentives in a divisionalized firm. Manage Sci 25(3):221–230
Ha Albert Y. (1998) Incentive-compatible pricing for a service facility with joint production and congestion externalities. Manage Sci 44(12):1623–1636
Hagel J, Singer M (1999) Unbundling the corporation. Harv Bus Rev 77(2):133–141
Hirshleifer J (1956) On the economics of transfer pricing. J Bus 29(3):172–184
Holmstrom B, Tirole Jean (1991) Transfer pricing and organizational form. J Law, Econ, Organ 7(2):201–228
Jordan DW, Smith P (1999) Nonlinear ordinary differential equations: an introduction to dynamical systems. Oxford University Press, Oxford
Kelly FP (1987) Reversibility and stochastic networks. Wiley, New York
Kleinrock L (1975) Queueing systems, volume I: theory. Wiley, New York
Lemoine AJ (1977) Networks of queues—a survey of equilibrium analysis. Manage Sci 24(4):464–481
Little JDC (1961) A proof of the queuing formula: $l = \lambda w$. Oper Res 9:383–387
Marschak J (1955) Elements for a theory of teams. Manage Sci 1(2):127–137
Marschak J, Radner Roy (1972) Economic theory of teams. Yale University Press, New Haven
McAfee RP (2002) Competitive solutions. Princeton University Press, Princeton
McKenzie L (1960) Matrices with dominant diagonals in economic theory. In: Arrow KJ, Karlin S, Suppes P (eds) Proceedings of the first Stanford symposium on mathematical methods in the social sciences. Stanford University Press, Stanford, CA, pp 47–62
Meyer CD (2000) Matrix analysis and applied linear algebra. SIAM, Philadelphia
Parnas DL (1972) On the criteria to be used in decomposing systems into modules. Commun ACM 15(12): 1053–1058
Porter M (1985) Competitive advantage: creating and sustaining superior performance. Free Press, New York
Radner R (1959) The application of linear programming to team decision problems. Manage Sci 5(2):143–150
Radner R (1993) The organization of decentralized information processing. Econometrica 61:1109–1146
Takagi H (1991) Queueing analysis, vol 1. Elsevier North Holland, Amsterdam