

A Trading System for Fairly Scheduling Fixed-Sized Delay-Tolerant Jobs at a Shared Link

Jorge Londoño
jmlon@bu.edu
Computer Science Dept
Boston University, USA

Azer Bestavros
best@bu.edu
Computer Science Dept
Boston University, USA

Nikolaos Laoutaris
nikos@tid.es
Telefonica Research
Barcelona, Spain

Abstract—Scheduling of delay-tolerant jobs has been proposed as a mechanism to alleviate pressure on congested network resources. However, when multiple competing users share these resources, they may not be willing to reveal the flexibility of the schedule for their jobs. This work presents a trading system that enables the users to trade their finite allowances in a scenario where they have fixed-size atomic jobs. The trading system makes it possible for tasks with strict timing requirements to be completed on time, while rewarding customers who exhibit flexibility regarding the schedule of their workloads (by reducing their operating costs or assigning them a larger share of off-peak capacity). The trading system hereby presented thus provides the right incentives so that user agents schedule their delay-tolerant jobs in a way beneficial for the whole system. It is proven to always converge, and simulations on real traces show significant reductions on the peak-to-valley ratio on the link utilization.

I. INTRODUCTION

Traffic patterns follow diurnal patterns with large peak-to-valley ratios. This large variability poses serious challenges when provisioning resources, for example the capacity of the network links. On the one hand, the provider could use the trends of peak utilization to estimate the future demand and provision the resources accordingly. In doing so, it will ensure that the degradation of the Quality of Service (QoS) observed by end users stays within bounds, although this will most likely require huge investments in infrastructure. On the other hand, not doing so means the performance will suffer significant degradation during peak hours and in a competitive environment where customers may easily switch providers, customers will do so. Laoutaris and Rodriguez [1] propose a third alternative, based on the observation that some tasks are Delay-Tolerant (DT), *i.e.* they can be scheduled at some later time as they are not interactive and responsiveness is not required. The benefit of shifting DT workloads is twofold: For the provider, the installed capacity is better utilized over time and the peak demand is reduced, reducing the pressure for over-provisioning its infrastructure. For the users, the reduced demand during peak times translates to better performance for interactive applications.

J. Londoño is supported in part by the Universidad Pontificia Bolivariana and COLCIENCIAS–Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología “Francisco José de Caldas”.

This work is supported in part by NSF awards CCF-0820138, CSR-0720604, EFRI-0735974, CNS-0524477, CNS-0520166, and CNS-0952145.

In a recent work, Laoutaris et al [2] presented concrete scheduling policies for the transfer of Delay Tolerant Bulk traffic over wide area networks. The fundamental assumption in this construction is that there is a single controlling authority able to coordinate the execution of transfers according to a globally optimal schedule. This paper considers a different scenario where the users of the system compete for the shared resource and look to optimize their own utility regardless of the system-wide goals. This behavior is commonly called *selfish* and it is typically the case that selfish behavior leads to a suboptimal utilization of the system resources. The problem is exacerbated by the fact that commonly used pricing schemes do not provide incentives for the users to choose actions beneficial for the system. This is exemplified by the flat-rate and the proportional-usage pricing models. In the first case, the user pays a fixed amount (*e.g.* a monthly fee) independent of the actual usage. In the second case, the charges are proportional to the user’s utilization (for example when charged per MB transferred), but independent of the overall system utilization. Neither of these schemes provides any incentive to schedule DT tasks off the peak hours, as the cost for the user is the same. This is precisely the problem addressed in this paper: Create a mechanism that in the face of selfish users, incentives them to schedule their DT tasks as to better load-balance the utilization of the shared resource over time.

The mechanism presented in this paper also takes into consideration other practical challenges and requirements: 1) The users do not have exact valuations of their jobs. Various micro-economic mechanisms such as auctions and commodity markets rely on having precise valuations, but in practice they may not be known, may be subjective and are impractical to communicate to the system. Instead, our mechanism defines a variable cost scheme so that the cost of heavily demanded resources becomes high and gives the users the opportunity to re-schedule their jobs as to avoid these hot-spots. 2) The system should provide for some notion of fairness between the users. This is non-trivial given the lack of valuations for their jobs and the distinction of classes of jobs. Our mechanism provides a notion of fairness where the users of the same resource in the same period pay the same unit cost. 3) The actual trading and allocation of the resources should be implementable by software agents, in order for the system to operate autonomically and minimize the need for actual user

intervention.

A. Related work

Laoutaris and Rodriguez [1] identify the lack of mechanisms for handling DT jobs on the network and proposed two mechanisms for doing so. The first one is to give the users *higher-than-the-purchased* access rate during off-peak hours. The second one is the introduction of “internet post offices” for scheduling and performing DT transfers during off-peak periods. Our mechanism fits in the first category, noting that our proposal gives a concrete mechanism for providing incentives in a scenario with selfish/rational users, provides a notion of fairness, and can be implemented by autonomous software agents. Other schemes, that rely on computing an optimal schedule and executing the DT tasks according to this schedule have been explored by Laoutaris et al [2]. In this case, the fundamental assumption is that either all the tasks are under the control of a single authority, or all the users cooperate by following the globally optimal schedule. Our scheme on the other hand assumes non-cooperative users who act as maximizers of their own utility.

Many works have studied the use of micro-economic mechanisms such as auctions and commodity markets for resource management. AuYoung [3] provides a review of two systems, *Bellagio* and *Mirage*, that use combinatorial auctions as their allocation mechanism. In particular several difficulties are pointed out, among which are: 1) the need of mechanisms for the user to elicit honest and comparable utility functions, 2) mechanisms such as the Vickrey-Clarke-Grooves (VCG) auction have the properties of making truthful bidding a dominant strategy and maximizing the social-value, but this properties do not hold when using approximate solutions for the subjacent winner determination problem, or when the auction is not static, but dynamic. They are also known to be susceptible to various attacks, for example collusion attacks. In G-Commerce [4], the authors compare both, auctions and commodities markets for allocation of grid resources. Our setting is different in that the auction/market is run per time-slot to allocate all the available resources to the highest bidders.

In [5] the authors highlight some of the potential benefits and the challenges when using market mechanisms for resource allocation. In this context, our mechanism uses as a social-goal that of load-balancing the utilization of the resource, which makes sense when the total demand is less than the capacity of the resource, but the cost of the resource depends on its utilization. Load-balancing is also beneficial from the standpoint of improving some QoS metrics, as for example the response time for interactive applications. Our mechanism also addresses the difficulty associated with expressing valuations, replacing the need for the users to elicit their valuations, by the problem of minimizing cost on behalf of the user, thus eliminating the need for additional/unpractical interfaces needed if the users were to communicate their valuations.

Another related issue is the definition of *fairness*. In the context of congestion management, definitions such as *max-min fairness* and *proportional-fairness* have been extensively considered, but their adequacy has also been questioned [6]. In this regard, our work seeks to offer a mechanism that implements a form of long-term cost fairness as suggested by Briscoe [6].

II. THE MULTI-USER SCHEDULING PROBLEM

A. Definitions

Each user or application¹ has a task. Let A_i be the task of user i . For the purposes of the analysis, it is assumed that time is quantized into intervals of length Δt called time-slots² and j is the slot number. Each task is characterized by the demand on the resource during each time-slot, therefore the tasks themselves are described by vectors of the form $A_i = (\dots, a_{ij}, \dots)$, where the values $a_{ij} \geq 0$ are positive real constants, indicating the task’s resource requirement during time-slot j . For this reason, the tasks are said to be *fixed-size*. Observe that even though in practice the demand of each task with a time-slot may be variable, our model may still be used taking a_{ij} to be the maximum demand per slot.

In addition, each task has a *slack* parameter s that captures its delay-tolerance. So for example, if $s = 0$ the task cannot be scheduled at any other time-slot, if $s = 1$ the task can be shifted one slot back or forward in the schedule and so forth.³ Fig.1 illustrates these definitions.

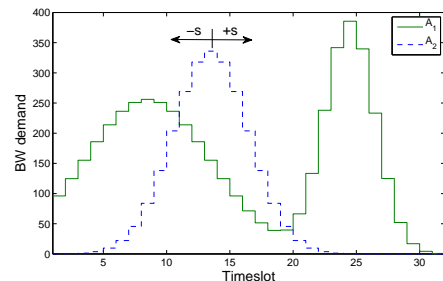


Fig. 1. Example tasks descriptions for two users

Under the model just described, when a single authority controls all the tasks, the problem is that of finding an optimal schedule subject to capacity and slack constraints. The optimality goal may be application dependent, but for illustration, our analysis assumes that the goal is load-balancing the utilization of the resource over time. For example, this minimizes the system’s cost when the 95/5-rule⁴ is used for pricing. When many users share the system, the problem becomes more challenging, as explained next.

¹For the purposes of this paper we will use both terms interchangeably
²Time-slots are large compared to packet transmission times.
³Alternatively, a deadline-based description can be shown to be equivalent.
⁴The cost of the service is determined by the 95-percentile of the 5-min traffic aggregates during a pre-defined period

B. Problem statement

When there are many competing users in the system, their choice of actions (the schedule for their task) usually follows their own private goals (for example minimizing their individual cost), but ignoring the overall system goal. For this reason, this kind of actions is usually referred to as *selfish*. Selfish behavior is detrimental for the system because it makes it very difficult and expensive to handle the diurnal patterns of activity. The typical solutions are either to provision the resources to the peak utilization at a very large cost for the provider, or to let the resources saturate sacrificing the QoS perceived by the users.

It follows that the problem is to design a mechanism so that selfish users get the right incentive to schedule their delay-tolerant task during periods of low utilization. The problem is challenging for several reasons: Selfish users may not be truthful, meaning that they may not reveal the true slack associated with their tasks; the system should provide some measure of *fairness*, understood as charging the same to users with the same requirements; the system should be “neutral”, meaning that the valuation associated with the execution of a task within its feasible interval is set by its owner, and not set ad-hoc or “dictated” by some central authority; the valuations themselves may not be known, as it is commonly the case in computing systems and it would be impractical having the actual user communicate them to the system even if known.

III. SYSTEM ARCHITECTURE

A. The Demand-Based Trading System

Fig.2 illustrates the operation of the trading system. Here the marketplace is run by the owner of the resource, as this is the party interested in load-balancing its utilization over time. The owner of the resource has no control over the preferences or constraints of the users. A per-user agent bids for allocations within each one of the time-slots. A bid is a task vector A_i . Once the marketplace has received all the bids, it reports back a vector of prices-per-slot, and a vector of total demands per time-slot. The key idea is that the unit-price is variable and dependent on the total demand on a given slot, effectively making the slot price oscillate according to the supply and demand. In response, user agents may submit new bids, corresponding to vectors A'_i where some of the components may be shifted in time. What components get shifted and by how much is the result of minimizing the cost of the task, subject to the tasks’ timing constraints.⁵ The marketplace recomputes the prices and the system iterates until it finds a stable solution. Two key issues for the operation of the system are: 1) how to set the prices, and 2) ensure that the system converges to a stable, mutually satisfactory solution. The remainder of this section considers the first one, and the second is analyzed in §IV.

Regarding the assignment of prices, we adopt an idea inspired from the concept of *exchange markets*: the price is

⁵A dynamic programming algorithm for solving this problem is presented in full version of this work [7]

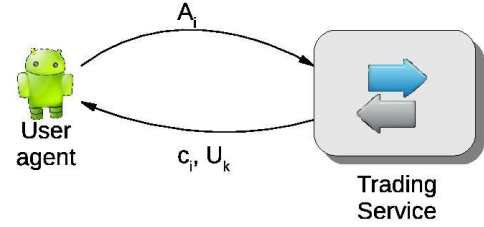


Fig. 2. Interaction with the trading system

variable and driven by the supply and demand of the goods. However, in an exchange market the utility function is known, making it possible to establish a market-clearing solution. The alternative we propose is to make use of a demand-dependent cost function, such that the unit-cost reflects the demand on a given resource (time-slot). Highly demanded resources become more expensive and this gives the right incentive for delay-tolerant tasks to be re-scheduled during periods of lower demand. The use of a cost function also eliminates the need for private valuations, the users’ selfish goal is satisfied by minimizing their costs. To accomplish this, we define the cost of a task allocating a_{ij} units of the j^{th} time-slot to be

$$c_{ij} = a_{ij} \cdot U_j \quad (1)$$

where $U_j = \frac{1}{C} \sum_i a_{ij}$ is the total demand due to all the users on the j^{th} time-slot, and C is a positive constant.⁶ The total cost for user i is simply the sum of the costs throughout all the time-slots $c_i = \sum_j c_{ij}$. It is worth observing two properties of this cost function: 1) Every user pays the same unit-price, for this reason it is said to be fair; and 2) The users who contribute the most to the congested time-slots are also the most heavily penalized.

B. Mechanism and Policy

The trading system just described provides a measure of cost to be enforced by the service provider. The simplest implementation would just convert this value into real currency. Currency lets the users express how much the service is worth and also gives the service provider the incentive to upgrade the infrastructure to keep up with the demand. The use of real currency is not always possible, though. For example, many services operate under flat-rate pricing models, which effectively disconnect pricing and utilization. One possible solution is the use of *virtual currency*. For virtual-currency to be effective, it must be worth to its users. In fact, this can be achieved by having a limited supply of it. If it is not wisely used, the user may run out of virtual currency and be unable to complete its jobs. On the other hand the mechanism makes it possible for the agents to exchange flexibility for volume. Given that highly demanded slots are more expensive, the same allowance translates into a larger volume if used in less-demanded slots. If virtual-currency is to be used, the allocation of virtual currency should be determined by some external policy. For example, all the users could receive

⁶When C is the capacity of the resource, U_j is the utilization

identical amounts, and what slots they elect to spend their budgets on is up to them. Another alternative is to have classes, for example gold/silver/bronze with allowances established per class. What is important is that policies like these can be easily implemented and the mechanism empowers the users to trade their allowances as they better fit their needs.

C. Notes about implementation

The user himself needs not to be directly involved in the trading process. As a matter of fact, the whole process can be easily automated to be performed by a software agent doing the trading on the user’s behalf. All that is needed is for the agent to have a characterization of the tasks of the users, which could be performed by a software agent as well, say from historical trace analysis. This characterization would provide the sets of tasks and their respective slacks to the trading agent. Exploring techniques to accomplish this characterization is left for future work.

IV. ANALYTICAL RESULTS

The bidding process described in the previous section can be analyzed as a pure-strategies game, where each bid is the action of a player in response to the other players actions. This bidding process is guaranteed to terminate thanks to the following theorem

Theorem 1: When the per-player cost is $c_i = \sum_j a_{ij} \cdot U_j$, the pure strategies game in which users adopt better/best responses to allocate atomic units work on each time-slot converges to a Nash-Equilibrium (NE).

The proof of this theorem is given in the Appendix.⁷

An interesting observation from the proof is the fact that the ordering of the task components is not important as long as each atomic component is scheduled into mutually exclusive time-slots. This allows generalizing the model for handling several subtasks per user with different slacks per subtask.

An important concern in game-theory analysis is the Price of Anarchy (PoA), defined as the ratio between the worst-case NE social cost and the globally optimal solution. The following theorem establishes the PoA for the our trading system

Theorem 2: The PoA for the pure-strategies game modeling the market where n agents bid for the allocation of the resources for their tasks, and the tasks are described as finite ordered sequences of resource demands, is n .⁸

Although this upper bound is high, it refers to a worst-case which in practice is very difficult to encounter. In fact, Fig.3-left illustrates simulation-based results for randomly generated sets of workloads, where this ratio is always below two. We observe that as the number of tasks and slots increases this ratio becomes very close to one, which bodes well for the target applications of our system.

⁷The analysis presented here does not consider external constraints, such as the capacity of the link. We have shown that the system still converges in these case, for further details see [7]

⁸Proof omitted due to space constraints. Please refer to [7]

V. EXPERIMENTAL EVALUATION

To further illustrate the benefits obtainable by using the schedule trading system, we evaluate the system using network traces as a readily available source of traces. Needless to say that the system could be used in scheduling other types of workloads, as for example batch jobs to be executed in a server. Publicly available WAN traces [8] were used to conduct the experimental evaluation of the system. These traces provide packet-level information on a high capacity WAN link. For all the experiments, the traffic was aggregated in $5min$ time-slots, a sample of a $24hr$ period was used, thus giving 288 time-slots. Some pre-processing steps were done to identify user sessions (traffic belonging to a single application) and the sessions were used as subtasks with independent slack, according to extended model of §IV.

We conducted an experiment using these sessions to evaluate the effect on the 95th percentile of the link’s 5-minute traffic volume (the 95% traffic envelop). Figure 3-(center,right) shows the outcome after the market reaches an equilibrium. The curve labelled $slack = 0$ is the original schedule as present in the traces. For brevity, it is assumed that all users adopt the same $slack$ value for all their sessions, so the curves $slack=3,6,12$ correspond to an slack of $15min$, $30min$, and $1hr$ respectively. The center is the actual traffic aggregate on per time slot and shows that as the flexibility of tasks increases, the peak-to-valley ratios reduce. In fact a very large peak of 139MB at time-slot 235 was reduced to 58MB just by having a $slack$ of 3. The figure on the right shows the distribution of the traffic per time-slot. The distribution for different values of $slack$ illustrate the load-balancing effect of the market: Slots the utilization increases for slots with low demand and the decreases for highly demanded slots. Of particular attention is the 95% value of these curves, as this is the basis for pricing when using the 95/5 rule. Table I shows the values of the 95% traffic envelop. These results underscore that selfishly scheduling subtasks yields an equilibrium with *significant* reduction in the 95% traffic envelop – up to 31% reduction when slack is 1 hour. Even for a small slack of 15 minutes, the savings amount to 16%.

Slack	95%(MB)	Reduction%
0	36.3	0.0
3	30.6	15.6
6	27.4	24.4
12	24.9	31.4

TABLE I
95% UTILIZATION RESULTING FROM BANDWIDTH TRADING.

VI. CONCLUSION

The schedule trading system presented in this work enables self-interested agents to collectively converge on what *they* perceive to be an equitable allocation, based on their individual, private valuation of system utility. The system incentivizes otherwise non-cooperative users to schedule their delay-tolerant tasks, to achieve a smoother aggregate utilization of the resource. Doing so reduces the cost/improves the utility of

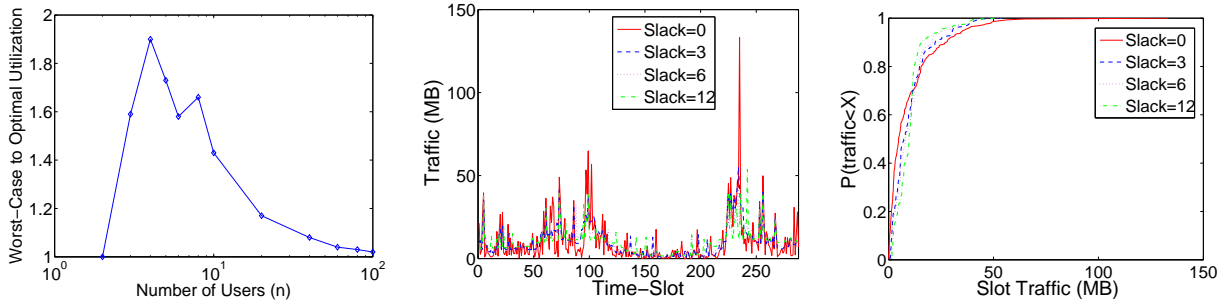


Fig. 3. Simulation results – Left: worst-case to optimal ratio, center: utilization over time, right: CDF of slots traffic

delay-tolerant tasks and makes it possible for tasks with more stringent constraints to be completed on time within the finite constraints of existing resources. It also benefits the provider by yielding smoother aggregate utilization, which represents important savings (particularly when their cost are driven by peak utilization, as for example when using the 95/5 pricing rule), or alternatively a better utilization of the otherwise idle periods of its infrastructure thus reducing the pressure for infrastructure upgrades.

The trading system presented here acts as a facilitator for a community of users so that they can share a resource and reach a globally desirable state, that would be unobtainable if it were to rely on uncooperative users. In doing so, it is neutral in the sense of leaving the valuation of tasks to their owners and not imposing some ad-hoc policies.

In this work we presented the case where jobs are fixed size. Another interesting question arises when considering *elastic traffic*, *i.e.* jobs whose allocation per time-slot need not be fixed. In a related paper [9], we have explored this case as well as the problem of combining fixed-size and elastic traffic.

REFERENCES

- [1] N. Laoutaris and P. Rodriguez, “Good Things Come to Those Who (Can) Wait or how to handle Delay Tolerant traffic and make peace on the Internet,” in *HotNets’08*, 2008.
- [2] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, “Delay tolerant bulk data transfers on the internet,” in *SIGMETRICS*. New York, NY, USA: ACM, 2009, pp. 229–238.
- [3] A. AuYoung, P. Buonadonna, B. N. Chun, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, *Market Oriented Grid and Utility Computing*. Wiley, 2009, ch. Two Auction-Based Resource Allocation Environments: Design and Experience.
- [4] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, “G-commerce: Market formulations controlling resource allocation on the computational grid,” in *IPDPS’01*, Washington, DC, USA, 2001, p. 46.
- [5] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. Chun, “Why markets could (but don’t currently) solve resource allocation problems in systems,” in *HOTOS’05*, Berkeley, CA, USA, 2005.
- [6] B. Briscoe, “Flow rate fairness: Dismantling a religion,” *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 63–74, April 2007.
- [7] J. Londoño, “Embedding Games: Distributed Resource Management with Selfish Users,” Ph.D Thesis, Boston University, Boston, MA, May 2010.
- [8] MAWI Working Group, “Traffic archive,” 2009. [Online]. Available: <http://tracer.csl.sony.co.jp/mawi/>
- [9] J. Londoño, A. Bestavros, and N. Laoutaris, “Trade & Cap: A Customer-Managed, Market-Based System for Trading Bandwidth Allowances at a Shared Link,” in *NetEcon’10*, Vancouver, Canada, Oct. 2010.

APPENDIX

PROOF OF THEOREM 1

Proof: Define the following function:

$$\Phi = \sum_{i=1}^n \sum_{j=1}^T c_{ij} = C \sum_{j=1}^T U_j^2$$

where n is the number of users and T the number of time-slots. When a player makes a cost-reducing move, $\Delta c_i < 0$,

$$\sum_j (a'_{ij} U'_j - a_{ij} U_j) < 0 \quad (2)$$

where a'_{ij}, U'_j denote the user allocation and the total utilization after the execution of the move. Notice that for any other player $k \neq i$, its utilization of interval j does not change, but the change in the total utilization affects its cost as follows

$$\Delta c_k = \sum_j a_{kj} (U'_j - U_j)$$

Adding the changes of the players other than i

$$\begin{aligned} \sum_{k \neq i} \Delta c_k &= \sum_{k \neq i} \left(\sum_j a_{kj} (U'_j - U_j) \right) \\ &= \sum_j \left((U'_j - U_j) \sum_{k \neq i} a_{kj} \right) \\ &= \frac{1}{C} \sum_j \left((a'_{ij} - a_{ij}) \sum_{k \neq i} a_{kj} \right) \quad (3) \end{aligned}$$

where the last step uses the fact that $U'_j - U_j = \frac{1}{C} (a'_{ij} - a_{ij})$ because players other than i did not change their allocations. The atomicity of the components implies that the vector (a'_{ij}, \dots) is a permutation of the elements of the vector (a_{ij}, \dots) , therefore $\sum_j a'_{ij} = \sum_j a_{ij}$. Then expression (2) can be re-arranged as follows

$$\sum_j (a'_{ij} U'_j - a_{ij} U_j) = \sum_j \left((a'_{ij} - a_{ij}) \sum_{k \neq i} a_{kj} \right) < 0$$

The last expression and the fact that $C > 0$ allows to conclude that expression (3) is also negative, thus letting us conclude that $\Delta \Phi < 0$, *i.e.* Φ is a strictly decreasing potential function for the game. ■