

Globally Optimal Bulk Data Transfers in Overlay Routing Networks

Andrei Agapi and Thilo Kielmann

aagapi@few.vu.nl, kielmann@cs.vu.nl

*Dept. of Computer Science – Vrije Universiteit Amsterdam
De Boelelaan 1083, 1081HV Amsterdam, The Netherlands*

Marcelo Pasin, Sébastien Soudan and Pascale Vicat-Blanc Primet

{marcelo.pasin,sebastien.soudan,pascale.primet}@ens-lyon.fr

INRIA RESO and Laboratoire d'Informatique du Parallélisme: LIP

Ecole Normale Supérieure

46, Allée d'Italie, 69364 LYON Cedex 07, France



CoreGRID Technical Report
Number TR-0157

August 13, 2008

Institute on Grid Systems, Tools and Environments

CoreGRID - Network of Excellence

URL: <http://www.coregrid.net>

Globally Optimal Bulk Data Transfers in Overlay Routing Networks

Andrei Agapi and Thilo Kielmann

aagapi@few.vu.nl, kielmann@cs.vu.nl

Dept. of Computer Science – Vrije Universiteit Amsterdam

De Boelelaan 1083, 1081HV Amsterdam, The Netherlands

Marcelo Pasin, Sébastien Soudan and Pascale Vicat-Blanc Primet

{marcelo.pasin, sebastien.soudan, pascale.primet}@ens-lyon.fr

INRIA RESO and Laboratoire d'Informatique du Parallélisme: LIP

Ecole Normale Supérieure

46, Allée d'Italie, 69364 LYON Cedex 07, France

CoreGRID TR-0157

August 13, 2008

Abstract

This report describes the results of a six-week research visit by the first author to the INRIA RESO research group at ENS Lyon. We address the problem of scheduling bulk data transfers in routing overlays running over public networks, in a globally optimal manner. One challenge encountered in this endeavor is to jointly address the global optimization of routes taken by transfers in the routing overlay, on the one hand, and the transfer schedules in the time domain, on the other, such that a single global objective is optimized (we currently choose minimizing global network congestion as our objective). In this direction, we explore two alternative approaches, one based on a linear programming optimization, and another on oblivious routing strategies. A second challenge in scheduling deadline-constrained transfers in a public network setting is dealing with uncertainties in future resource availability (e.g. available bandwidths) given by the presence of cross-traffic. For this, we propose using distributions rather than fixed values for describing network links and further deriving probabilistic guarantees for deadlines to be met based on the distributions of transfer paths chosen by our algorithm.

Our system is currently still in a development and testing phase. We plan evaluations based on trace-driven simulations, using traces collected on PlanetLab by us and traces we downloaded from the S3 measurement project. We will use S3 traces we have been collecting for about two months. Depending on the outcome of the simulation approach and on other practical factors, we might also evaluate the system by running a full-fledged implementation over PlanetLab.

Applications of our approach include p2p networks used for data transfers, QoS-enhancing routing overlays such as [5] [19] [12], SETI@home-like p2p computing, with deadline constraints, over large scale data (e.g. astronomic streams), as well as any scenario in which a distributed organization needs to transfer and process large amounts of data over public networks with deadline constraints.

1 Introduction

In the present work, we address the problem of scheduling bulk data transfers over routing overlays in a globally optimal manner. In the context of a best-effort Internet, which is unable to provide QoS guarantees for applications

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

using it, systems such as opportunistic routing overlays [5] [19] [12] have emerged over the past decade to provide a flexible and interesting way to get extra QoS out of public networks by only manipulating the application level, thus needing no cooperation from the underlying network. Such routing overlays typically work by routing around congestions and network bottlenecks in an attempt to improve QoS. Besides routing overlays, many other overlays and p2p systems, such as Bittorrent [1] or Tribler [15], whether in a structured or heuristic manner, as a design goal or as a side effect of their design, use various forms of application-level relaying or multicasting in an attempt to improve network performance for the peers using them. While this approach might be profitable for individual, selfish peers, a question arises on the efficiency of these approaches from the perspective of the network as a whole.

Taking such a perspective might be interesting especially from the point of view of novel emerging applications, such as p2p computing, cloud computing [3], distributed stream processing [8] and, in general, for any distributed application in which distant peers have common, rather than selfish goals. Examples include commercial distributed organizations that need to transfer and process, in real-time, data produced at geographically distributed locations or soft real-time, volunteer-based, processing of large scale astronomic data, as produced by new-generation distributed radio telescopes such as LOFAR [2] or SKA [4], or, in the case of Very Long Baseline Interferometry, by multiple existing telescopes found at very distant locations. For such applications, cooperatively transferring and processing the data in a globally optimal manner, such that all of the deadlines are met, is a much more appropriate approach than acting as a collection of selfish peers. Even in the case of applications where peers have selfish goals (such as file downloads), using a global optimization might, in some instances, yield better results than the current approaches, whereby each peer generally uses the system in a greedy and selfish manner.

We do the global optimization of bulk transfers in an overlay in two dimensions: the time dimension and the routing dimension. The time dimension refers to scheduling each requested transfer in the time domain such that the deadlines for all requests are met, as well as a global objective (such as network congestion) is optimized. Basically, in each time interval, the bandwidth that each transfer can use is rate-limited such that the global objective is cooperatively achieved. In previous work [7], some of the present authors have addressed scheduling in the time domain by defining the Bulk Data Transfer Scheduling (BDTS) problem and proposing a linear programming based global optimization solution for it. Solving BDTS can globally optimize the scheduling in the time domain of a set of bulk data transfers over a predefined set of paths in a network. We further detail BDTS, as well as the defined paradigm it uses in section 2.

While optimally scheduling transfers in the time domain, BDTS does not handle routing, but instead takes the network routes that transfers should use as an input. This is partly because the initial focus of the BDTS solver was on native networks, where application-level control over routing is anyway limited, unlike in the routing overlay setting. In the present work, for the purpose of addressing overlays running over public networks, such as the Internet, we propose to use a routing dimension in the global optimization. This refers to the overlay-level routes taken by data transfers in the overlay. Basically, the problem we address, Bulk Data Routing and Transfer (BDRT), is to jointly optimize the routing of the requests within the overlay, as well as the time-domain transfer schedules for these requests, such that deadlines are met and the global objective is optimized. Currently, for reasons we will further develop, we use minimizing global network congestion (for now, at the overlay, not native network level) as our objective.

This report is structured as follows. Section 2 defines the model and formulates the BDRT problem we are trying to solve. We describe our approach in section 3. Specifically, we reference a linear programming optimization approach in subsection 3.1, an approach based on oblivious routing algorithms in subsection 3.2 and we describe our approach in dealing with network resource variability in subsection 3.3. Section 4 outlines ongoing and future work, performance evaluation, as well as possible applications of our system. Section 5 describes the relation of this work with the roadmaps of the relevant CoreGRID institutes.

2 Paradigm and problem definition

The paradigm we work in is that previously used by us in [7], and also by other previous work such as [10]. In this setting, bulk transfers are characterized, besides the data volume $niur$ to be transmitted, by an arrival time $etar$ (at which data is available at the source) and a deadline fir (by which all data has to be transferred to the destination). We define the interval $omegar=[niur, fir]$ as the active window of a request. [7] does not address routing, therefore also takes as a problem input $bigfir$, a finite set of network paths associated with each transfer request (representing the paths the request can use to transfer data, as found by a separate algorithm). Then, formally, a bulk transfer request r is defined as a triple $(niur, omegar, bigfir)$, where members mean the associated data volume, active window (arrival

time to deadline) and set of paths, respectively. In the offline scheduling case, the simplest case, we assume that all the information on future transfer requests is available from the beginning.

We have thus defined the Bulk Data Transfer Scheduling (BDTS) problem, as the following: given as a problem input a network graph and a finite set of transfer requests as described before, find a bandwidth allocation profile λ for each request r such that all deadlines are met and overall network congestion factor is minimized. A bandwidth allocation profile is defined as a function $\lambda_r(t) : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, specifying the rate at which request r is entitled to transfer at time t . The network congestion factor is defined as the maximum link utilization over all links in the network, over all intervals.

We have shown in [7] that, by dividing the timeline in $2|R| - 1$ intervals generated by all transfer arrival times and deadlines, where $|R|$ is the number of requests (similarly to [14]), BDTS is in P and it possesses an optimal solution with the bandwidth allocation profile for each task in the form of a step function with $O(|R|)$ intervals. This means that, within each interval, the rate at which the transfer bandwidth is limited, for any task, is a constant. We have shown this by writing the problem in a Linear Programming (LP) form (thus effectively solvable in practice, e.g. by the simplex method) and by further showing it is equivalent with the Maximum Concurrent Flow Problem (MCFP), which is in P. For further details, please see [7].

In the current work, besides scheduling in the time domain, we jointly address the routing of transfers within the routing overlay network. Therefore, the new problem, Bulk Data Routing and Transfer (BDRT), is defined as: given as a problem input a network graph and a finite set of transfer requests only described as a tuple (μ_r, ω_r) , i.e. transfer volume and active window, with no predefined paths for transfers, find the network paths to schedule each transfer on, as well as the bandwidth allocation profile for each of these requests and each of these paths such that all deadlines are met and overall network congestion factor is minimized. In section 3, we will briefly describe or reference our solutions to this problem.

3 Explored approaches

Briefly, we find that the problem can be written in an LP form, inspired by Wang's work in [20], done in the context of Internet core traffic engineering. BDRT enjoys a linear formulation, thus is effectively solvable for the multipath case (that is, requests are "splittable" over many paths in the network and there is no restriction on the number of paths to use). For the single path or max K paths case, the problem becomes an Integer Linear Programming (ILP) one, and we believe it to be NP, equivalent to the K-disjoint paths problem. However, as in [20], there might exist efficient heuristics for these cases, for instance based on rerouting flows starting from the optimal multipath solution.

We also explore an approach based on oblivious routing algorithms, further detailed in subsection 3.2. Basically, in this approach, we use an oblivious routing algorithm that minimizes congestion for choosing paths for each request, then use the BDTS solver to find the bandwidth allocation profiles in the time domain for these paths. A third approach is to simply use congestion-minimizing oblivious routing, with no time-domain optimization to schedule requests. A fourth approach is to use a simpler, greedy, selfish heuristic to find paths (such as shortest path or the BARON heuristic proposed in [13], based on widest paths), alone or together with time-domain optimization.

The reason we have explored the last, heuristic approaches is that, although we expect them to yield sub-optimal schedules (actually, oblivious routing algorithms have been proven to be suboptimal in [16]), we find them to be an interesting term of comparison to the optimal solution. If for no other reason, because, albeit suboptimal, oblivious algorithms are much easier to distribute than a global linear optimization and we find it interesting to quantitatively explore the performance penalty they incur. We further briefly explain and reference each of the approaches taken. Since this is currently work in progress, we defer further details to a later publication.

3.1 The linear programming approach

We wrote BDRT in a linear form, inspired by Wang's work in [20]. Done in the context of Internet core traffic engineering, [20] solves the routing problem for the case in which transfer requests arrive simultaneously and there are no deadlines for transfers, but rather each request is treated as a continuous bandwidth reservation request, typical for Internet core flows. Requests have thus variable bandwidth demands, but no specific arrival time and are basically infinite flows. Inspired by this method, we propose a linear optimization method for multipath routing for the BDRT problem, in which requests are characterized by variable data volumes, arrival times and deadlines and there is an extra time dimension. Being work in progress, we defer further details of our method until further evaluation.

The LP form is only valid for the multipath routing case, i.e. when there is no bound on the number of network paths any transfer can use. In the single path or maximum K path case, the problem has an ILP formulation and becomes NP, similarly to the more particular problem solved in [20]. As shown there, it is equivalent with the K-disjoint paths problem, which is NP. Finding efficient heuristics to efficiently approximate BDRT for these cases is subject of our current ongoing work.

3.2 The oblivious routing approach

Oblivious routing algorithms are algorithms that route each packet with no information about the routes taken by any other packets in the network. Because they need no global information, they can be easily distributed and are quite commonly used.

Racke has found the remarkable result [16] that, for generic graphs, there exist oblivious routing algorithms that achieve a polylogarithmic competitive ratio w.r.t. congestion, that is, can achieve congestion that is within a polylogarithmic factor ($O(\log^3(n))$) from optimal. He bases his proof on producing a tree decomposition of the network, and then showing that routing on the tree is almost as good as routing in the original network. Later Bienkowski et al [6] and, independently, Harrelson et al. [11] have proposed polynomial-time oblivious algorithms that are $O(\log^4(n))$, and $O(\log^2(n)\log(\log(n)))$ -competitive, respectively.

As a first approach, we use a simple, randomized, oblivious routing algorithm to route requests. Although it is less competitive than the tree decomposition-based algorithms presented in [11] or [6] (since, as will be explained, it only considers per-path, rather than per-link characteristics), it has the advantage that it is quite simple, lightweight, distributed, and can be implemented in logarithmic time. The general idea is that, for a given routing request, a path is chosen by a weighted random pick: the probability that any of the feasible paths is chosen is variable and proportional to the path's capacity (which is the minimum of the capacities of component links). For instance, assume there are 3 feasible paths between a source and a destination: one path of 40 Mbps and 2 of 30 Mbps each. Then, the first path has 40% probability of getting chosen, and the other 2 each have a 30% probability. This way, all routing requests coming from each source are uniformly distributed over the network resources and thus congestion is reduced. When considering the multipath routing case, each request is split in multiple, small-sized packets and one "routing request" is issued for each of these packets.

Note that each node needs to locally keep evidence of the feasible paths between itself and all destinations in the network, as well as the corresponding capacities. Instead of capacity, a metric such as available bandwidth or an expected value derived from a distribution of available bandwidths can be used (as detailed in subsection 3.3), but, for simplicity of exposition, I will call this information "capacity" in the following.

For the single path case, when only 1 path can be chosen for any bulk transfer, the heuristic is as following:

1) When a node needs to route a request, it first calculates the minimum bandwidth needed by that request, based on the request volume V , start time s and deadline d . This is the minimum bandwidth needed such that the deadline is made, in a continuous transfer: $V/(d-s)$.

2) As explained, the node keeps evidence, locally, of all feasible paths (either direct or through at most K intermediate relays) from itself to all possible data sinks. If we have R relays and S sinks, for 1 hop there are $S(R+1)$ such paths. In general, for K hops, there are $S(R^K + R^{K-1} + \dots + R + 1)$ such paths. If any node in the network can be a relay or a sink, this means $N^2 + N$, or $N^{K+1} + N^K + \dots + N^2 + N$ paths, respectively. Between each source-destination pair, there are at most $N^K + \dots + N^2 + N + 1$ alternate paths. As explained, the node is also locally aware of the capacities of these paths.

Given this information, the node does the following to choose the path:

3) From all alternate paths feasible between source and destination, it filters out all paths that have a capacity lower than the minimum bandwidth needed by the request, as calculated at step 1. This is because we assume we don't do multipath streaming, thus we have an "unsplittable flow" setting, therefore paths with lower capacity than b cannot be used.

4) As in the multipath case, among remaining feasible paths, it does a random pick weighted on capacity to pick the path to route the request on.

This routing algorithm can be implemented in logarithmic time in the size of the number of alternate paths maintained for each destination, as shown in Figure 1, since operations at all steps take logarithmic time if the table is maintained as a heap, ordered by path capacity. The filtering of feasible paths (step 3) is obviously a logarithmic lookup in the heap. This step yields path P_{lim} , such that all paths below it are unusable (have capacity $< b$) and all above, inclusively, are usable. For step 4, the random pick, for each path P in the table, we precompute and maintain $SumP$,

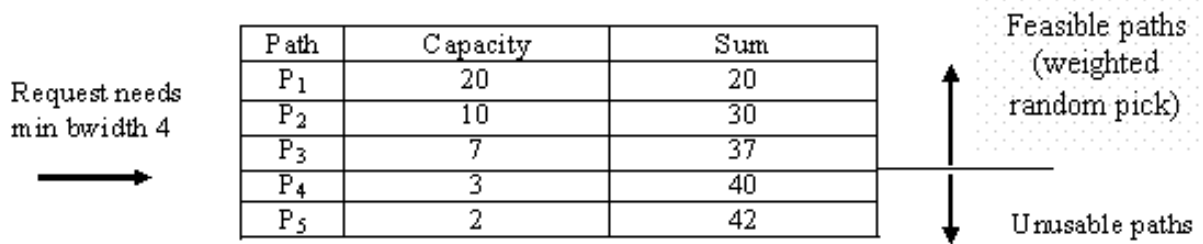


Figure 1: Routing table for single path randomized oblivious routing.

as the sum of all path capacities above it, inclusively (see example table in the figure). Then, once Plim is found, the node generates a random number between 0 and Sumlim. The request is routed on the path at position result in the table, such that the random number is minimally contained in $[0, \text{Sumresult}]$ i.e. it is contained in $[0, \text{Sumresult}]$, but not in $[0, \text{Sumresult}-1]$. The size of the table shown in Figure 1 is the number of alternate paths maintained per destination, which, as explained above, is $O(N^K)$. Therefore, complexity is $O(\log(N^K))$.

As previously explained, we use this heuristic to obliviously assign requests to either multiple or single paths in the network. In the multiple path case, traffic originating from any single source to any destination is uniformly spread over available network paths, thus congestion is likely to be minimized. Of course, the algorithm is sub-optimal for several reasons, including the fact that it only looks at end-to-end paths, not individual links (like [11] and [6]), thus requests for individual links might be skewed, depending on the network graph structure. Another reason is that, for the practical purpose of limiting the size of the routing tables, the number of alternate paths maintained might be limited, as previously explained. In the single path case, the heuristic has the additional, potentially serious, problem that it uniformly distributes variable-sized requests, rather than equally-sized packets, to paths, therefore a high variance in the distribution of transfer volume sizes can seriously affect network congestion.

The heuristic can be used on its own, in which case we try to solve the entire BDRT problem in a totally oblivious way, or, alternatively, a time-domain optimization (solving BDTS) can be applied to the output of the heuristic. We plan to compare both these approaches with the linear optimization approach. We also plan to compare all of these with selfish routing heuristics such as BARON [13] and perhaps near-optimal oblivious algorithms such as [11] or [6].

3.3 Dealing with cross traffic

Our optimizer targets overlays that might be running over wide area, public networks, where cross-traffic is present and available bandwidth of links is variable. As a first step, we consider overlay-level links. Extending the system to native layer links is subject of future work.

To address resource variability, our system is stochastic. For this, instead of single values for the per-link available bandwidths, we rather maintain distributions of values, representing the distributions of available bandwidths historically observed on overlay level links. To monitor the links and build these distributions, we currently use active measurements with a certain frequency. Individual probes are available bandwidth probes obtained with the pathchirp [17] and spruce [18] tools, as well as capacity estimation probes, using the pathrate [9] tool. As discussed in section 4, for evaluation, we also leverage public measurement traces produced within the S3 project [21].

To find appropriate paths and transfer schedules by our algorithms, we use these distributions. Specifically, we characterize each distribution by a Complementary Cumulative Distribution Function (CCDF), which gives us the probability $P(X)$ that the available bandwidth on a given link will be greater than a certain value X . Then, in the current version of our algorithm, by fixing a certain overall probability margin M that we are willing to accept (for instance 95%), we derive, for each link, the respective bandwidth value X such that $\text{CCDF}(X)=M$. This basically gives us an expected value over which the available bandwidth on that link is likely to be, with probability M . These likely expected values are then used as link weights by our global optimization algorithms.

Since the global optimization objective of our algorithms is to minimize network congestion, the maximum link utilization is minimized. This means basically that we try to minimize the ratio of available bandwidth we require from network links. In the context of CCDFs of available bandwidth distributions, an interesting side effect of minimizing the available bandwidth required from links is that we basically maximize the probability that the required value is available (since, on any link, it is more likely that at least 200 Kbps are available than 1 Mbps). Therefore, the final

probability we output that the schedules are met is typically higher than the initial margin M . This is one of the main reasons we focus on minimizing network congestion as a global optimization objective in this work.

When building the distributions, we need to address the complementary problem of tuning the measurement framework such that the historical data we use to infer expected values is representative for current values. In the context of the previously observed self-similar nature of cross traffic over several timescales (e.g. typically periodic pattern observed over day-night timeframes) and of other practical factors, such as keeping the sampling traffic reasonably low, this is a non-trivial problem. We defer a more detailed description of how to address the sampling tuning and distribution building problem to a later publication.

4 Future work and objectives, evaluation and possible applications

Our system is currently in a development, testing and evaluation phase. Our current Java-based implementation leverages software previously written for the BDTS solver framework, as well as the Gnu Linear Programming Kit (GLPK). The final outcome of our implementation should be in the form of a global scheduler for bulk data transfers, to be used by applications using overlays that do application-level routing over public networks. The type of applications targeted are mainly those mentioned in the introduction section, primarily including novel cloud computing or streaming applications in which peers need to cooperatively reach a common goal, such as jointly transferring and processing sets of distributed data.

For evaluation, we plan a trace-driven simulation using real available bandwidth and capacity traces collected by us on Planetlab. We also leverage public Planetlab traces we downloaded from the S3 measurement project. These traces span about 2 months worth of measurements. Depending on the outcome of the simulation approach and on other practical factors, we might also evaluate the system by running a full-fledged implementation over PlanetLab.

5 Relation to the roadmap of CoreGRID institutes

This work has been performed in the context of CoreGRID's institute on Grid systems, tools, and environments (STE). A key objective of the STE institute is to build a generic platform for efficiently executing applications in Grid environments. An important aspect here is the ability to adapt an application's behavior to fluctuations in (network) resource capacities at run time. Whereas this topic has been intensively studied within the STE institute, the aspect of optimizing application-level bulk transfers had been unaddressed so far. The work presented in this report is closing this gap.

References

- [1] Bittorrent, www.bittorrent.com.
- [2] Lofar, www.lofar.org.
- [3] Seti@home, <http://setiathome.berkeley.edu>.
- [4] Ska, www.skatelescope.org.
- [5] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of SOSP*, October 2001.
- [6] M. Bienkowski, M. Korzeniowski, and H. Racke. A practical algorithm for constructing oblivious routing schemes. In *Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures*, 2003.
- [7] Binbin Chen and Pascale Vicat-Blanc Primet. Scheduling bulk data transfers in grid networks. 2007. IEEE CCGRID2007.
- [8] F. Douglis, M. Branson, K. Hildrum, B. Rong, and F. Ye. Multi-site cooperative data stream analysis. In *SIGOPS Oper. Syst. Rev.*, 2006.
- [9] C. Dovrolis and R. Prasad. Pathrate: A measurement tool for the capacity of network paths.

- [10] R. Guerin and A. Orda. Networks with advance reservations: The routing perspective. In *Proceedings of IEEE INFOCOM*, 2000.
- [11] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures*, 2003.
- [12] P. Karbhari, M. Ammar, and E. Zegura. Optimizing end-to-end throughput for data transfers on an overlay-tcp path. In *Proceedings of IFIP Networking Conference*, 2005.
- [13] Sung-Ju Lee, Sujata Banerjee, Puneet Sharma, Praveen Yalagandula, and Sujoy Basu. Bandwidth-aware routing in overlay networks. In *Proceedings of INFOCOM*, 2008.
- [14] C. Martel. Preemptive scheduling with release times, deadlines, and due times. In *Journal of ACM*, volume 29, pages 812–829, July 1982.
- [15] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M. van Steen, and H.J. Sips. Tribler: A social-based peer-to-peer system. In *5th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.
- [16] H. Racke. Minimizing congestion in general networks. In *Proceedings of FOCS 43*, 2002.
- [17] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *In Passive and Active Measurement Workshop (2003)*., 2003.
- [18] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *In Internet Measurement Conference (2003)*., 2003.
- [19] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. Overqos: An overlay based architecture for enhancing internet qos. In *Proceedings of NSDI*, 2004.
- [20] Y. Wang and Z. Wang. Explicit routing algorithms for internet traffic engineering. In *Proceedings of ICCCN*, September 1999.
- [21] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. S3: a scalable sensing service for monitoring large networked systems. In *Proceedings of the 2006 SIGCOMM workshop on Internet network management*, 2006.