A Performance Study of a Large-scale Data Collection Problem*

Cheng-Fu Chou†
ccf@csie.ntu.edu.tw

Yung-Chun (Justin) Wan‡
ycwan@cs.umd.edu

William C. Cheng§
bill.cheng@acm.org

Leana Golubchik¶
leana@cs.usc.edu

Samir Khuller†
samir@cs.umd.edu

Abstract

We consider the problem of moving a large amount of data from several source hosts to a destination host over a wide-area network, i.e., a large-scale data collection problem. This problem is important since improvements in data collection times are crucial to performance of many applications, such as wide-area uploads, high-performance computing, and data mining.

Existing approaches to the large-scale data collection problem are (a) transferring data directly from the source hosts to the destination host, using IP-prescribed routes (which we refer to as direct methods) or or (b) using "best"-path type application-level re-routing techniques, which we refer to as non-coordinated methods. However, we believe that in the case of large-scale data collection applications, it is important to *coordinate* data transfers from multiple sources. More specifically, our coordinated method takes into consideration the transfer demands of all source hosts and then schedules all data transfers in parallel, using multiple paths existing between the source hosts and the destination host. All this is done at the application layer.

In this paper, we present a performance and robustness study of the different data collection methods, namely the direct, the non-coordinated, and the coordinated methods. Our results show that coordinated methods can perform significantly better than non-coordinated and direct methods under various types of network congestion conditions. We also show that coordinated methods are more robust than non-coordinated methods under inaccuracies in network conditions information. Therefore, we believe that coordinated methods are a promising approach to large-scale data collection problems.

1 Introduction

The large-scale data collection problem we consider here is the problem of moving a large amount of data from a set of source hosts to a destination host over a wide-area network. Such problems correspond to a set of important applications. These applications include online submission of income tax forms, submission of papers to conferences, Internet-based storage, and many more. We refer to such applications as wide-area uploads. Upload applications are likely to become significant contributors to Internet traffic in the near future, as large-scale data transfer applications take advantage of the proliferation of the Internet in society and industry. For instance, consider the online submission of income tax forms, U.S. Congress has mandated that 80% of tax returns should be filed electroni-

*This work was supported in part by the NSF ITR CCR0113192 and the NSF Digital Government EIA0091474 grants.

†Dept. of Computer Science and Information Engineering, National Taiwan University. This work was partly done while the author was with the Department of Computer Science and UMIACS at the University of Maryland.

‡Department of Computer Science and UMIACS, University of Maryland at College Park.

§TeleGIF, Marina del Rey, California. This work was partly done while the author was with the Department of Computer Science and UMIACS at the University of Maryland.

¶Computer Science Department and IMSC and ISI, University of Southern California. This work was partly done while the author was with the Department of Computer Science and UMIACS at the University of Maryland.

cally by 2007. With (on the order) 100 million individual tax returns filed in U.S. yearly, where each return is on the order of 100KB [11], scalability and performance issues are a major concern.

In addition to wide-area upload applications, one could consider other applications where improvement of data collection times is an important problem. One example is high-performance computing applications where large amounts of data need to be transferred from one or more data repositories to one or more destinations, where computation on that data is performed. Another example is data mining applications where large amounts of data may need to be transferred to a particular server for analysis purposes.

Recently, we proposed a scalable and secure application-level architecture for wide-area upload applications, termed Bistro [4]. Given a large number of clients that need to upload their data by a given deadline to a given destination server, Bistro breaks the upload process into three steps: (1) a timestamp step which ensures that the data is submitted on-time *without* having to actually transfer the data, (2) a data transfer step, where clients *push* data to intermediate hosts (termed bistros) participating in the Bistro architecture, which ensures fast response time for the clients, and (3) a data collection step, where a destination server (termed destination bistro) *pulls* data from bistros, i.e., the destination server determines how and when the data is transferred from the bistros. During Step (2) receipts corresponding to clients' transfers are sent by the (intermediate) bistros to the destination bistro; hence the destination bistro knows where to find all the data which needs to be collected during Step (3). We consider Step (3) as a large-scale data collection problem, and the *performance of this step is the focus of our paper*.

Consequently, our *data collection problem* can be stated as follows:

> *Given*
>
> > a set of source hosts, the amount of data to be collected from each host, and a common destination host for the data
>
> *our goal is to*
>
> > construct a data transfer schedule which specifies on which path, in what order, and at what time should each "piece" of data be transferred to the destination host
>
> *where the objective is to*
>
> > minimize the time it takes to collect all data from the source hosts, usually referred to as *makespan*.

Our focus here is on application-level solutions. Hence, a path in the above stated problem is defined as a sequence of hosts, where the transfer of data between any pair of hosts is performed over TCP/IP.

One simple approach to solving the data collection problem is to transfer the data *directly* from source hosts to the destination host based on a transfer schedule computed by the destination host; hence the data would follow an IP-prescribed route. However, this might result in long transfer times, for example, due to network congestion. Given the current state of IP routing, congestion conditions alone may not necessarily cause a change of routes between a pair of hosts, even if alternate routes exist.

Another approach is to transfer the data using *application-level re-routing* techniques. Recent works such as Detour and RON [2, 16] consider frameworks for application-level re-routing around congestion and failures. Most such re-routing techniques use "best"-path type rerouting, where data from a source host is transferred over the "best" application level path to the destination host. This path is determined independently for each source host (refer to Section 3 for more details).

However, we believe that in the case of large-scale data collection applications, it is important to *coordinate* data transfers from multiple sources. More specifically, the difference as compared to an application level re-routing approach, which we refer to as a non-coordinated method, is that a coordinated method would (i) take into consideration the transfer demands of all source hosts, and (ii) potentially use all possible paths existing between the source hosts and the destination host, all at the application layer. We have been developing such a coordinated approach in the context of our Bistro system. We also note that in the case of Bistro, there is no need for a distributed algorithm for a coordinated approach to the large-scale data collection problem since in Bistro we employ a server pull approach (as stated above), with all information needed to solve the data collection problem available at the destination server.

The contribution and the focus of this paper are as follows. We present a study which compares the performance and robustness of the three potential approaches to the large-scale data collection problem, namely direct, non-coordinated, and coordinated approaches. (The specific performance metrics used in this comparison are defined in Section 4.) We show that coordinated methods can perform better than non-coordinated and direct methods under various degrees and types of network congestion. These improvements are achieved without significant effects on other network traffic throughput. In addition, we show that coordinated methods are more robust than non-coordinated methods under inaccuracies in network condition information.

The remainder of this paper is organized as follows. In Section 2 we briefly survey related work. Section 3 gives a brief overview of the direct, the non-coordinated, and the coordinated methods for the large-scale data collection problem. In Section 4 we give a quantitative evaluation and comparison of these methods. Section 5 gives our concluding remarks.

2 Related Work

As stated above, in this paper we focus on a performance and robustness study of different approaches to large-scale data transfers over wide-area networks, in the context of upload applications. In [4] an application-level *framework* for large-scale upload applications, termed Bistro, is proposed. To the best of our knowledge no other large-scale *upload* architecture exists to date. Hence, we do this work in the context of Bistro, although as noted above, other types of applications can benefit as well. Specifically, we focus on the performance of the *data collection step* as described in Section 1.

We note that some work exists on efficient design of multipoint-to-point aggregation mechanisms at the IP layer, e.g., [3, 5]. However, the solutions suggested there, in the context of IP routers and active networks, are focused on reduction of overheads due to small packets, such as *ACK*s, and aggregation of multiple small packets with the same destination into one large packet. It also requires the use of the active network framework which is not currently widely deployed over the public Internet.

We now briefly survey representative works on application level re-routing. Re-routing at the application level has been used to provide better end-to-end performance or efficient fault detection and recovery for wide-area applications. For instance, in [17] the authors perform a measurement-based study of comparing end-to-end quality of default routing vs alternate path routing (using metrics such as round-trip time, loss rate, and bandwidth). Their results show that in 30% to 80% of the cases considered, there is an alternate path with significantly superior quality. This work provides evidence for existence of alternate paths which can outperform default Internet paths.

Other frameworks or architectures which consider re-routing issues include Detour [16] and RON [2]. The Detour framework [16] is an informed transport protocol. It uses sharing of congestion information between hosts to provide a better "detour path" (via another node) for applications in order to improve the performance of each flow and the overall efficiency of the network. Detour routers are interconnected by using tunnels (i.e., a virtual point-to-point link); hence Detour is an in-kernel IP-in-IP packet encapsulation and routing architecture designed to support alternate-path routing. This work also provides evidence of potential long-term benefits of "detouring" packets via another node by comparing the long-term average properties of detoured paths against Internet-chosen paths.

The Resilient Overlay Network (RON) [2] is an architecture which allows distributed Internet applications to detect failure of paths (and periods of degraded performance) and recover fairly quickly by routing data through other (than source and destination) hosts. It also provides a framework for the implementation of expressive routing policies.

We note that the above mentioned re-routing works focus on architectures, protocols, and mechanisms for accomplishing application-level re-routing through the use of overlay networks. These works also provide evidence that such approaches can result in significant performance benefits.

In this work, we consider a similar environment (i.e., application-level techniques in an IP-type wide-area network). However, we consider an alternate approach, namely that of a *coordinated* data transfer. That is, in the above mentioned works all data transfers are treated independently, and hence each takes the "best" application-level route available. In contrast, we also consider coordination of multiple data transfers destined for the same host. We note that this coordination was initially motivated by large-scale data transfer problems which exist in the upload applications mentioned in Section 1. Furthermore, an important focus in this work and its main contribution is the performance and robustness study and the subsequent comparison of different approaches to large-scale transfers, namely direct, non-coordinated, and coordinated approaches.

3 Overview of Data Collection Approaches

In this section, we give a brief overview of the data collection methods evaluated in this paper.

Direct Method

A simple approach to the data collection problem defined in Section 1 is to devise a schedule for transferring the data *directly* from the source hosts to the destination host, where the data would follow an IP-prescribed route. For instance this includes:

1. transferring the data from all source hosts to the destination host in parallel, or
2. transferring the data from the source hosts to the destination host sequentially in some order, or
3. transferring the data in parallel from a constant number of source hosts at a time and possibly during a predetermined time slot,

as well as other variants.

Our extensive evaluation of a variety of direct methods showed that the first method given above performs as well as any of the other direct methods in terms of the makespan metric [6]. Hence, we use that method as a representative *direct method* in the remainder of the paper.

Non-Coordinated Method

We consider a non-coordinated approach in which each source host chooses a single application-level "best" path (in terms of available bandwidth) to transfer its data to the destination host. Specifically, this non-coordinated method is similar[1] to the current approaches used in RON [2] and Detour [16]. As suggested in [2, 16], in this non-coordinated approach, we only consider re-routing traffic through at most one intermediate host on the way to the destination host. For each source host, we first choose a path, p, with one intermediate host, and consider p as the current "best" path. We then consider another potential path, q, and choose q instead of p, if q's available bandwidth is more than 105% of p's available bandwidth, as in [2]. We continue in this manner, until all paths have been considered.

Coordinated Method

To construct a coordinated data transfer schedule, we formulate the problem as a network flow problem. By using a min-cost flow algorithm on an appropriately defined time-expanded (network) graph, we can obtain an optimal data transfer schedule; [6] gives the details of our algorithm. The basic idea behind this approach is to devise a coordinated data transfer schedule while optimizing a given performance metric, such as makespan (i.e., the total time to transfer the data). This is done by transferring data in parallel over multiple paths with the goal of shortening the makespan metric and

[1] We are not able to use the exact re-routing algorithms in [2, 16] as the *details* of these algorithms are not stated there.

scheduling each transfer at some appropriate time, so as to avoid competition for bandwidth among source hosts. Moreover, this coordinated data transfer method is an application level approach and hence does not require any additional information beyond what the above mentioned non-coordinated approaches use.

Figure 1 gives a high-level example to illustrates the intuition behind the differences among these approaches. Assume that both source hosts S1 and S2 have 1 unit of data to upload to the destination host D. Normally, this would go through network N1. Let us further assume that there is a bottleneck link somewhere between N1 and D and this link has unit capacity. Therefore, if D pulls data simultaneously from both S1 and S2. It will take 2 units of time to complete the transfer.

Another host, denoted by K, can transfer data to D at twice the unit capacity. However, its connection to hosts S1 and S2 is limited at unit capacity. If both S1 and S2 choose to use K to reroute the data to D, it will take 2 units of time to transfer data to K and another unit of time to transfer data from K to D. The total transfer time is 3 in this case. Chances are, uncoordinated approaches will not choose this solution.

However, if we *coordinate* the data transfers in the following fashion, we can cut the total data transfer time to 1.5. During the first unit of time, S1 transfers its data directly to D and, *simultaneously*, S2 transfers its data to K. At the end of this time period, there is 1 unit of data left at K. It would take another 0.5 units of time to complete the data transfer. As far as the makespan metrics is concerned, the data transfer from S2 to K is *free* because it occurs *in parallel* with the data transfer from S1 to D. Also, it does not *compete* with the other data transfer. This parallelism cannot be exploited explicitly by the other approaches. We believe that in a large network, many such opportunities exist and can be exploited using our approach.

We note that the running time of our coordinated data collection algorithm is dominated by the running time of the min-cost flow algorithm (refer to [6]), which can be solved efficiently. In our experiments, the entire computation takes on the order of a few seconds on a Pentium III 650 MHz machine.

We also note that in our evaluation of non-coordinated and coordinated methods we assume that the initial estimates of available capacity are used in construction of data transfer schedules for the entire duration of each experiment. If this is not the case, we can always compute a new estimate of available bandwidth during the execution of the transfer schedule and re-compute a new transfer schedule for the remaining data. (Our algorithm itself is very fast, and so this does not cause a problem even if the current transfer is stopped, and the schedule is changed.) Such adaptation to changing network conditions is an ongoing effort.

4 Performance Evaluation

In this section we evaluate the performance of direct, non-coordinated, and coordinated approaches to illustrate the benefits of using coordinated approaches. This evaluation is done through simulation; all results are reported with at least $95\% \pm 5\%$ confidence intervals.

Experimental Setup

We use ns2 [9] for all simulation results reported below. Our network topology is generated by the Georgia Tech Internetwork Topology Generator (GT-ITM) [10]; this topology consists of 152 nodes and two transit domains. Each transit domain has, on the average, 4 transit nodes with there being an edge between each pair of nodes with a probability of 0.6. Each node in a transit domain has, on the average, 3 stub domains connected to it. Each stub domain, on the average, has 6 nodes with there being an edge between each pair of nodes with a probability of 0.2. Figure 2 shows the transit domains of our topology. The capacity of transit-stub links and stub-stub links is 2.5Mbps, while the capacity of inter-transit-domain links (peering points) and intra-transit-domain links are 5Mbps and 10Mbps, respectively. We assign lower capacity to the peering points as peering points are often cited as the cause of delays in wide-area transfers [12]. The destination host and 2 of the source hosts are connected to transit domain A, and the remain-

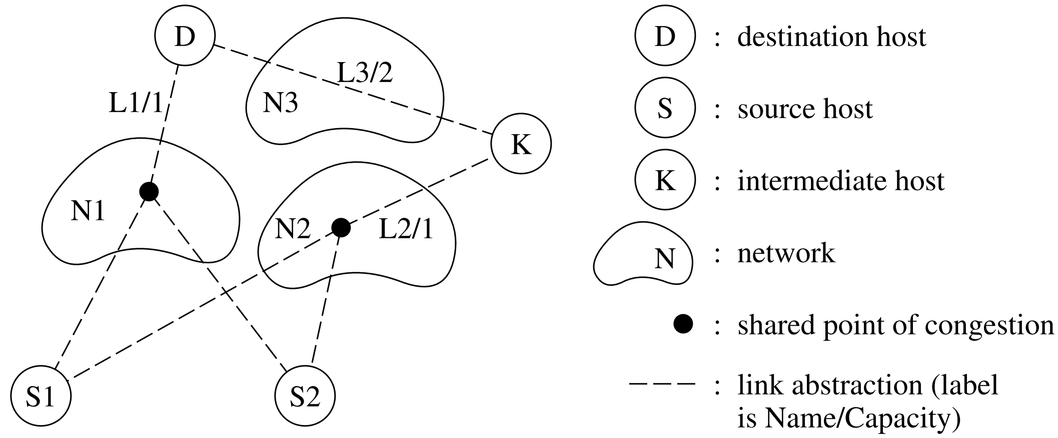


Figure 1: A high-level example.

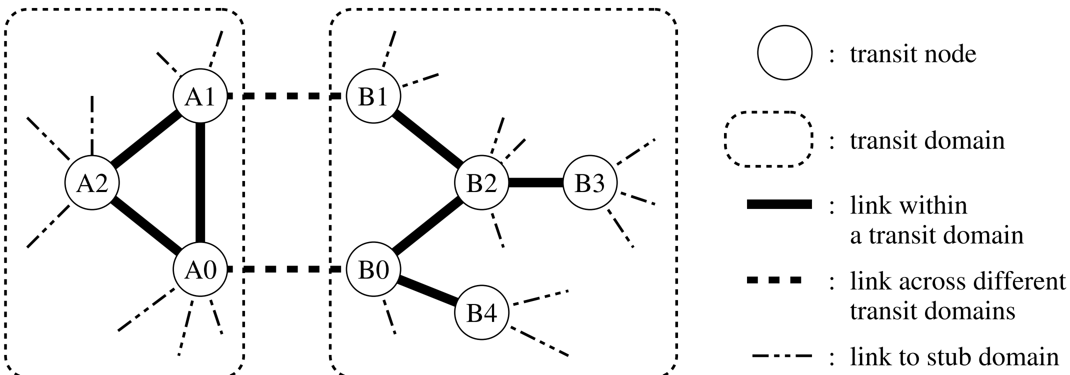


Figure 2: The simulation topology.

ing source hosts are connected to transit domain B. Each source host holds a total amount of data which is uniformly distributed between 25MB to 75MB with an additional constraint that the total amount of data in all 7 source hosts is 350MB. We note that in the case of our Bistro upload architecture, the amount of data in each source host is much larger than the size of the data transfer corresponding to an individual client because each source host receives data from a large number of clients in step(2) of the upload process (refer to Section 1).

We setup 0 to 60 background traffic flows from nodes attached to transit domain B to nodes attached to transit domain A. We investigate how the methods behave under different levels of congestion, which we create through the use of background traffic. Specifically, we control the ratio of the number of background flows in peering point (B0,A0) to the number of background flows in peering point (B1,A1), which in our experiments are 1:1 (symmetric), 2:1, and 1:3 (asymmetric). The background traffic is similar to that in [15]. Each background flow is an infinite FTP with a probability of 0.75. Otherwise, it is an on-off CBR UDP flow. The average on-times and off-times are chosen uniformly between 0.2 and 3 seconds. The average rate (including the off periods) is chosen so that the expected total volume of UDP traffic through a peering point takes up 0%, 5%, or 20% of the capacity of that point. To measure the available bandwidth of a path, we send (several times) a 5MB file between all pairs of source and destination hosts. We do this as a separate experiment, one pair at a time, and measure the corresponding throughput, before transferring any real data. Note that the method of estimating available bandwidth is a separate issue from the coordinated and non-coordinated approaches. We use this method to emulate the use of data collected during previous transfers in estimating bandwidth for future transfers. Nevertheless, we can replace the sending of a large file approach with any other bandwidth estimation approach, e.g., as in [7, 1, 13]. However, most such methods are currently fairly inaccurate. Hence, we perform a sensitivity to bandwidth estimation study to evaluate the effects of bandwidth estimation inaccuracies on the data transfer methods (see below). We expect that in the future such information will be provided by other Internet services, e.g., such as those proposed in SONAR [14], Internet Distance Map Service (IDMaps) [8], Network Weather Service (NWS) [18], and so on. Since these services will be provided for many applications, we do *not consider bandwidth estimation as an overhead* of *our* application but rather something that can be amortized over many applications.

Performance Study

The performance metrics used in the remainder of this section are (a) makespan, i.e., the time needed to complete transferring all data from all source hosts to the destination host, (b) maximum storage requirements averaged over source hosts (not including the destination host since it must collect all the data), and (c) mean throughput of background traffic during the data collection process, i.e., we also consider the effect of upload traffic on other network traffic. We believe that these metrics reflect the quality-of-service characteristics that would be of interest to large-scale data collection applications.

Figures 3(c) and 3(d) illustrate the effect of symmetric and asymmetric congestion, under the makespan metric, with varying amounts of the peering points' capacities utilized by the UDP traffic. The coordinated method performs better than other methods, especially for a large number of background flows, because it coordinates all transfers, and utilizes available bandwidth better. The improvement in the symmetric case is more than 100% over other methods. In the asymmetric case, the makespan of the coordinated method results in 156% and 64% better performance than that of the direct method and the non-coordinated method, respectively. Note that under the symmetric condition, the direct method performs as well as the non-coordinated method. This occurs because in the direct method, the IP-prescribed route of approximately half of the data passes through one peering point, while the IP-prescribed route of the other half of the data passes through the second peering point. This happens to be a good schedule in this case. Under the asymmetric condition, the non-coordinated method is better than the direct method since it re-routes data around the more congested path. Lastly, we note that when the available bandwidth of one path dominates, by a great amount, the available bandwidth of other paths, the performances of the coordinated method and the non-coordinated method should converge since both of them will use that extremely good path to transfer their data.

We also ran experiments with ratios of loading on peering points of 1:3, 2:1, and 3:1 (asymmetric). Qualitatively the results are similar, i.e., the coordinated method performs significantly better than the non-coordinated method under various types of network congestion conditions.

Effect of UDP traffic

We investigate the effect of UDP traffic on the makespan metric. In Figures 3, we depict the makespan metric under 0%, 5%, and 20% of UDP traffic. Figures 3(a), 3(c), and 3(e) show the symmetric case (1:1) and Figures 3(b), 3(d), and 3(f) show the asymmetric case (2:1). As we can see, Figures 3(a), 3(c), and 3(e) illustrate very similar results. The same can be said of Figures 3(b), 3(d), and 3(f). Hence, the observations we made above, in comparing direct, non-coordinated, and coordinated methods, still hold under different volumes of UDP traffic, i.e., similar trends persist.

Robustness Study

The motivations for doing a robustness study of the different data collection methods are (i) it is not easy to get a good estimation of available bandwidth of a path and (ii) network congestion conditions might change over time, and hence it is important to see how sensitive the results are to the accuracy of available bandwidth information. To this end we use perturbed values of available bandwidth in computing the data transfer schedule for both non-coordinated and coordinated methods. That is, we deviate available bandwidth values, used in construction of transfer schedules, from the actual estimates to emulate inaccuracies in available bandwidth estimates. Note that, the direct method does not use available bandwidth estimation information, and is therefore unaffected by the errors in the estimation.

Since more interesting scenarios correspond to inaccuracies in bandwidth estimation of congested points, (B0,A0) and (B1,A1), we consider the following two cases: (a) over-estimating the available bandwidth of paths passing through link (B0,A0) while under-estimating the available bandwidth of paths passing through link (B1,A1), and (b) under-estimating the available bandwidth of paths passing through link (B0,A0) while over-estimating the available bandwidth of paths passing through link (B1,A1). We ran a number of experiments corresponding to different loadings on peering points and different volumes of UDP traffic, and the results were qualitatively similar. Therefore, we only present an experiment with the ratio of traffic loads between peering point (B0,A0) and peering point(B1,A1) being 1:3. Figure 4 shows a series of results from over-estimating the available bandwidth of paths passing through the more congested link by 70% in Figure 4(a), 50% in Figure 4(b), and 30% in Figure 4(c), to accurate estimation in Figure 4(d), to under-estimating the available bandwidth of paths passing through the more congested link by 50% in Figure 4(e), and 70% in Figure 4(f). Although the coordinated method is affected by estimation errors in all paths, since it uses multiple paths to perform transfers, the effect is relatively small. For instance, even when the available bandwidth of the paths passing through the more congested link is over-estimated by 70%, as in Figure 4(a), the makespan is increased by 46% when comparing to the same method with accurate bandwidth estimation. If we under-estimate the available bandwidth by 70%, as in Figure 4(f), the makespan is increased by 24%. However, note that in all cases, the non-coordinated method performs worse than the coordinated method, anywhere from 37% to 142%. Furthermore, for the non-coordinated method, a mis-estimation can be more dangerous; e.g., a high over-estimation of the available bandwidth of a more congested link, as in Figures 4(a) and 4(b), can result in most of the traffic being routed through the more congested path. Thus, the resulting performance would degrade sharply and can be even worse than that of the direct method. The makespan of the non-coordinated method in Figure 4(a) is more than double of the makespan of the same method with accurate bandwidth estimation. On the other hand, since the non-coordinated method chooses the "best" path (in terms of available bandwidth), this over-estimation needs to be sufficiently high in order to force the wrong choice of a "best" path. For instance, in Figure 4(c), the mis-estimation does not affect the makespan metric. We note that, under-estimating the available bandwidth of paths passing through a highly congested link does not affect the non-coordinated method since that method does not use such paths.

Storage and Throughput Metrics

Figure 5 illustrates the *normalized* maximum per source host storage requirements (averaged over all source hosts) of the coordinated method and the non-coordinated method, under the symmetric and asymmetric congestion conditions. (These storage requirements are normalized by those of the direct method.) The direct method does not require additional storage since data is sent to the destination directly. The increase of storage space in the coordinated method is fairly small. In these experiments, it is at most 4%. The storage requirements of the non-coordinated method are significantly larger. Multiple data flows from different source hosts may be re-routed to the same intermediate host at the same time, which can result in large increases in storage requirements on that host (as large as 73% in these experiments). One advantage of the coordinated method is that it can schedule multiple data flows to be re-routed at different times.

Figure 6 shows how different methods affect the throughput of other network traffic[2]. Both the non-coordinated and the direct methods have a smaller impact on the throughput of the other network traf-

[2] We only consider TCP traffic since UDP traffic does not adjust its throughput when congestion occurs.

fic than the coordinated method. This is not surprising since the coordinated method is more aggressive than other methods in taking advantage of bandwidth available in the network. However, the difference between the direct method and the coordinated method in our experiments is only 2% to 9%.

5 Conclusions

In this work, we studied the performance and robustness characteristics of different approaches to the large-scale data collection problem, under various network congestion conditions and inaccuracies in estimation of bandwidth availability. We showed that coordinated methods perform significantly better than the non-coordinated and direct methods under the makespan metric. These improvements are achieved under low storage space overheads and without significant detrimental effects on other network traffic throughput. Moreover, we have showed that under various available bandwidth mis-estimation scenarios, our coordinated method still performs better than the non-coordinated method. Under high degree of mis-estimation, the performance of the coordinated method is less sensitive to such mis-estimations than the performance of the non-coordinated method. Therefore, we believe that coordinated methods are a promising approach to large-scale data collection problems.

References

[1] M. Allman. Measuring end-to-end bulk transfer capacity. *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *18th ACM SOSP Conference*, Canada, October 2001. ACM.

[3] B. R. Badrinath and P. Sudame. Gathercast: An efficient mechanism for multi-point to point aggregation in ip networks. Technical Report DCS-TR-362, Computer Science Department, Rutgers University, July 1998.

[4] S. Bhattacharjee, W. C. Cheng, C.-F. Chou, L. Golubchik, and S. Khuller. Bistro: a platform for building scalable wide-area upload applications. *ACM SIGMETRICS Performance Evaluation Review (also presented at the Workshop on Performance and Architecture of Web Servers (PAWS) in June 2000)*, 28(2):29–35, September 2000.

[5] K. Calvert, J. Griffioen, B. Mullins, A. Sehgal, and S. Wen. Concast: Design and Implementation of an Active Network Service. *IEEE Trans. on Networking*, 2000.

[6] W. C. Cheng, C.-F. Chou, L. Golubchik, S. Khuller, and Y. C. Wan. On a graph-theoretic approach to scheduling large-scale data transfers. Technical Report CS-TR-4322, University of Maryland, Jan 2002.

[7] C. Dovrolis and M. Jain. Pathload: A measurement tool for end-to-end available bandwidth. *Proceedings of the 3rd Passive and Active Measurements Workshop*, 2002.

[8] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, and Y. Jin. *Internet Distance Map Service*. http://irl.eecs.umich.edu/jamin/, 1999.

[9] http://www-mash.cs.berkeley.edu/ns/. *UCB/LBNL/VINT Network Simulator - ns (version 2)*.

[10] http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html. *Georgia Tech Internetwork Topology Generator*.

[11] IRS. *Fill-in Forms*. http://www.irs.ustreas.gov/prod/forms_pubs/fillin.html, 2001.

[12] T. Leighton. *The Challenges of Delivering Content on the Internet*. Keynote address at the ACM SIGMETRICS 2001 Conference, Cambridge, Massachusetts, June 2001.

[13] M. Mathis and M. Allman. A framework for defining empirical bulk transfer capacity metrics. *RFC 3148*, 2001.

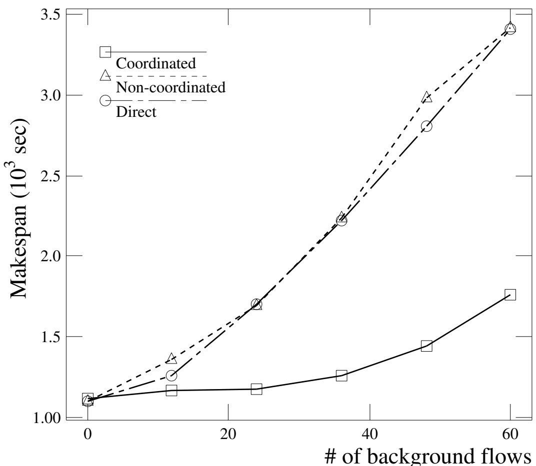
[14] K. Moore, J. Cox, and S.Green. Sonar - a network proximity service. *IETF Internet-Draft*, 1996.

[15] Dan Rubenstein, James F. Kurose, and Donald F. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Proceedings of 2000 ACM SIGMETRICS Conf.*, pages 145–155, 2000.

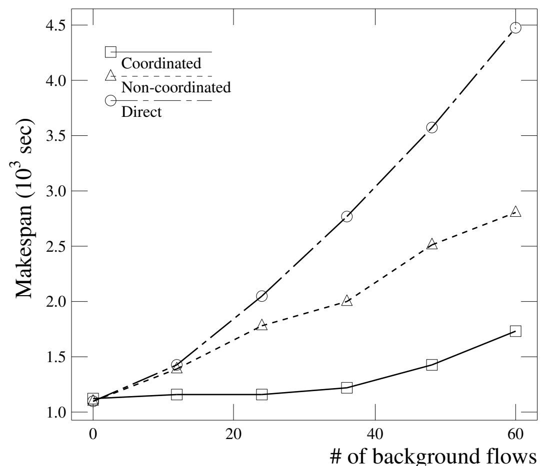
[16] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: a case for informed internet routing and transport. In *IEEE Micro*, volume 19, pages 50–59. IEEE, January 1999.

[17] Stefan Savage, Andy Collins, and Eric Hoffman. The end-to-end effects of internet path selection. In *ACM SIGCOMM '99 conference on Applications, technologies, architectures, and protocols for computer communication*, 1999.

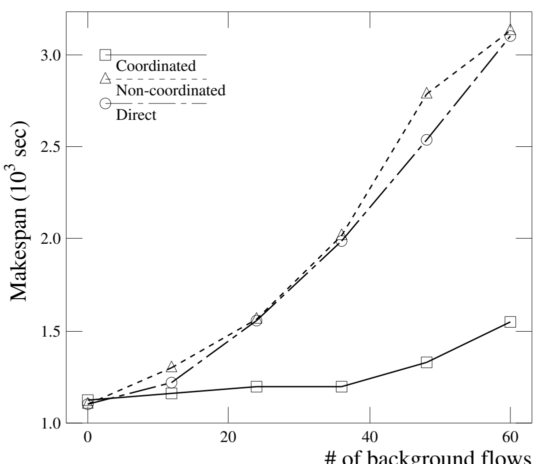
[18] R. Wolski and M. Swany. *Network Weather Service*. http://nws.cs.utk.edu/, 1999.



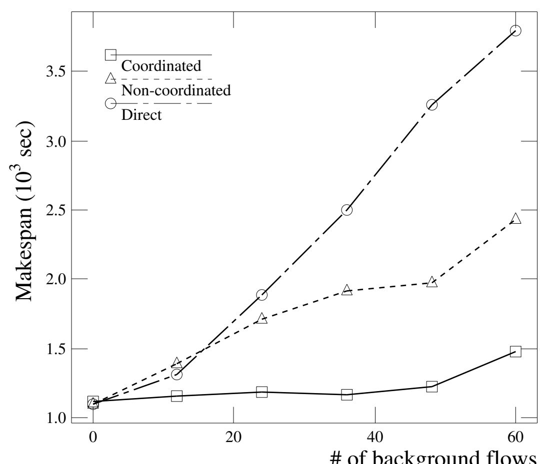
(a) 1:1, 0% UDP



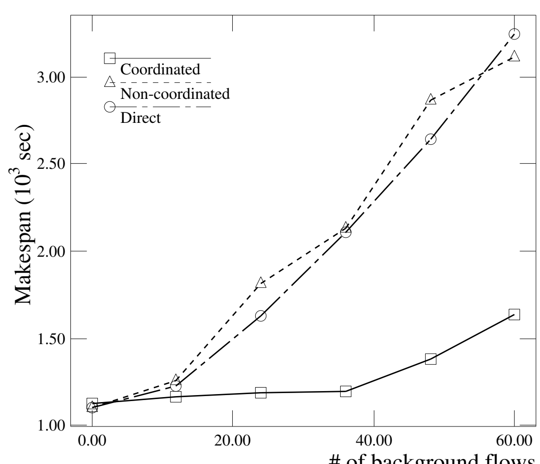
(b) 2:1, 0% UDP



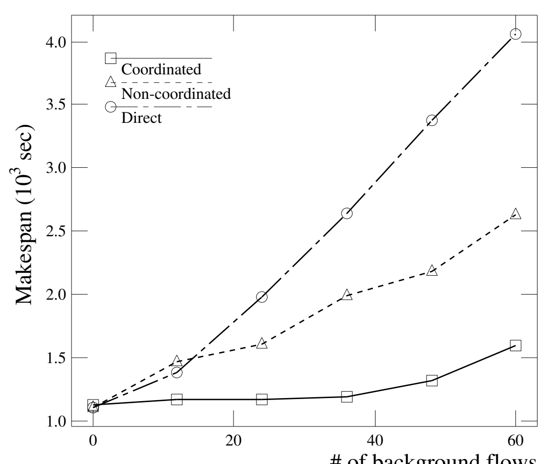
(c) 1:1, 5% UDP



(d) 2:1, 5% UDP



(e) 1:1, 20% UDP



(f) 2:1, 20% UDP

Figure 3: Makespan comparison under symmetric (1:1) and asymmetric (2:1) congestion with varying amounts of UDP traffic.

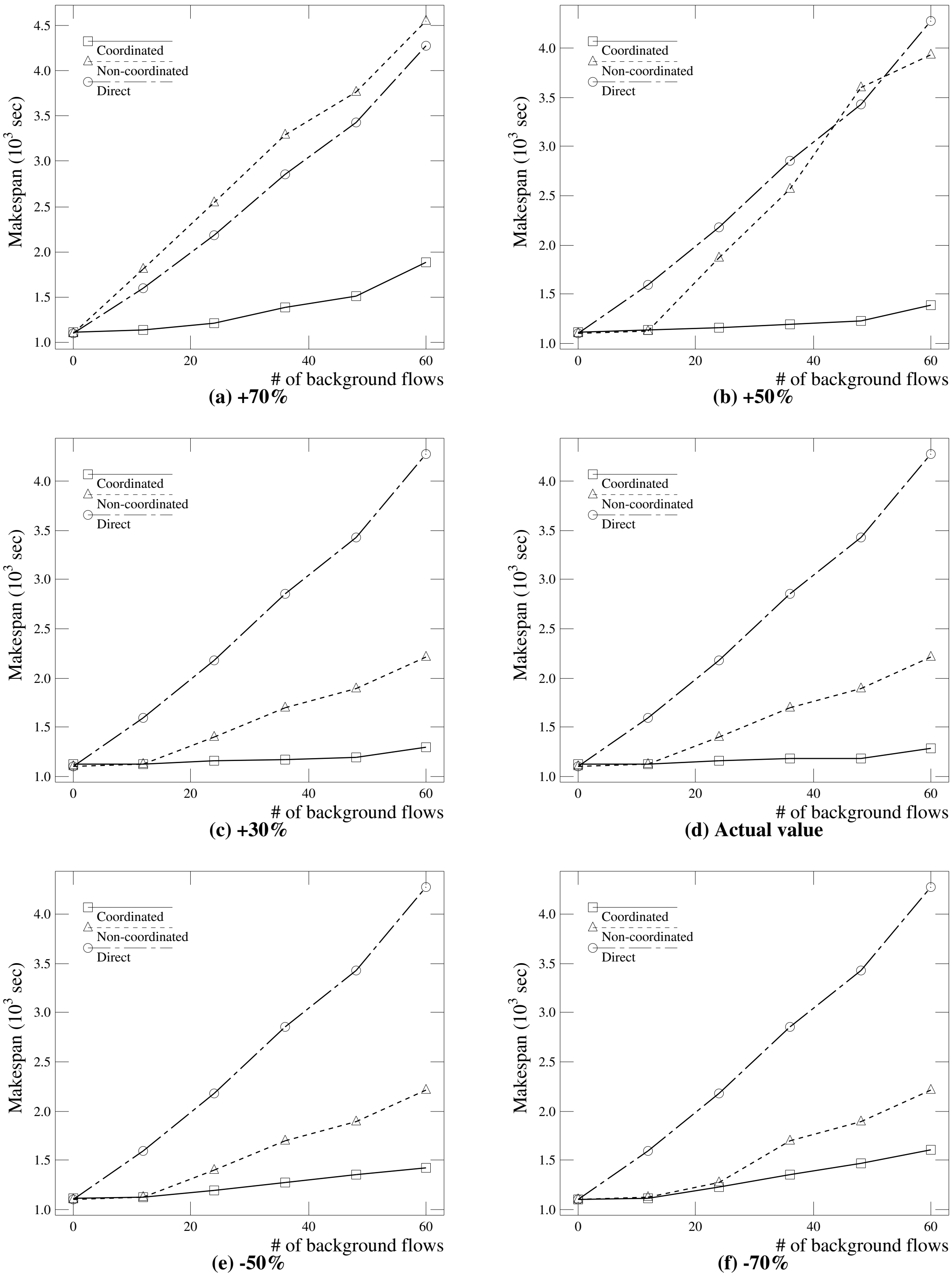


Figure 4: Sensitivity to bandwidth estimation under 5% UDP traffic and ratio of loads on peering points of 1:3, under the Makespan Metric. Individual figure captions indicate the percentage of mis-estimation of paths going through (B1,A1).

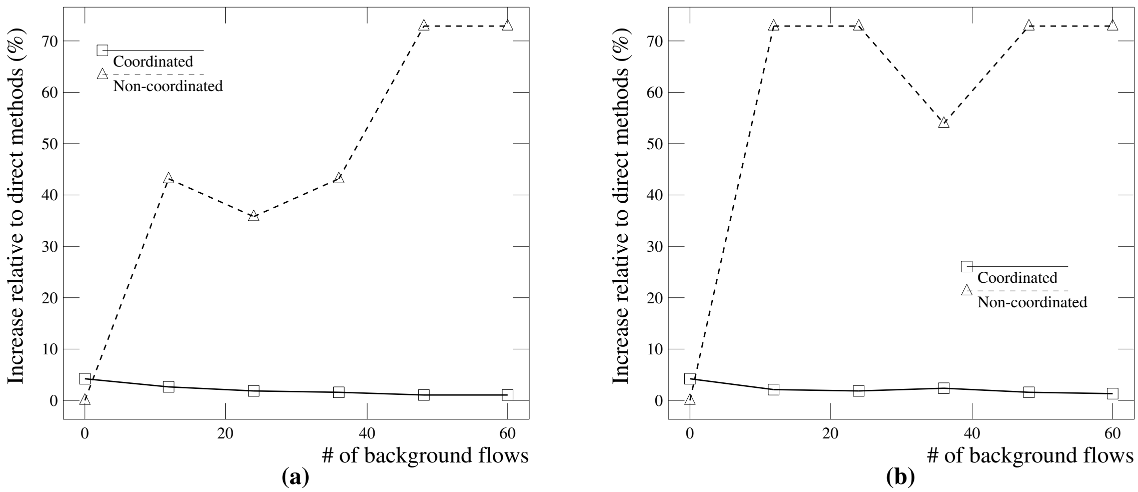


Figure 5: The effects of (a) symmetric (1:1) and (b) asymmetric (2:1) congestion, under the Storage Metric.

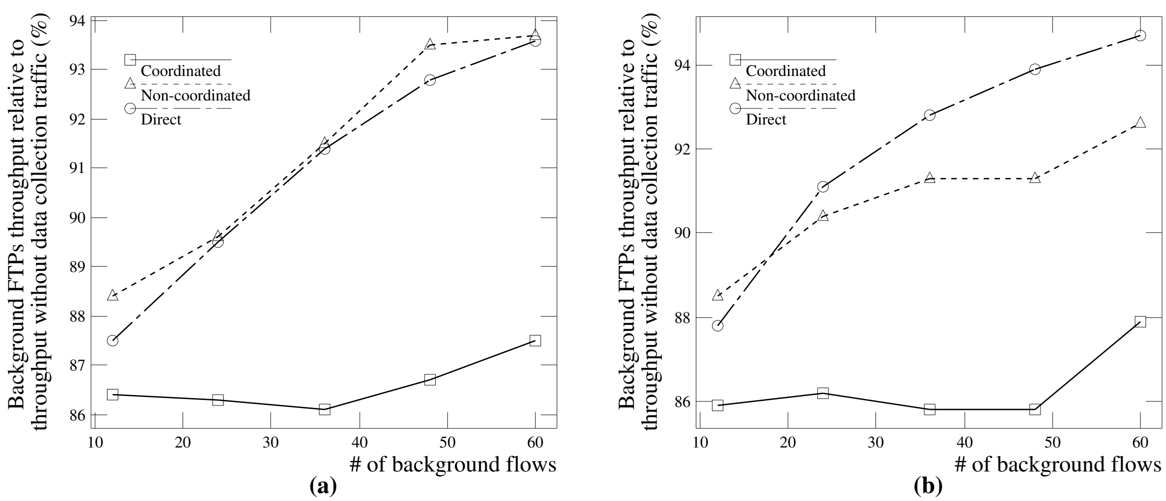


Figure 6: Throughput comparsion under the effects of (a) symmetric (1:1) and (b) asymmetric (2:1) congestion.