Scalable and Secure Data Collection Using Bistro*

[Appeared in Proceedings of the 2nd National Conference on Digital Government Research, 2002]

W. C. Cheng†
TeleGIF
Marina del Rey, CA 90292
bill.cheng@acm.org

C.-F. Chou
Dept. of Computer Science
and UMIACS
University of Maryland
College Park, MD 20742
chengfu@cs.umd.edu

L. Golubchik†
Computer Science Dept.
and ISI
USC
Los Angeles, CA 90089
leana@cs.usc.edu

S. Khuller
Dept. of Computer Science
and UMIACS
University of Maryland
College Park, MD 20742
samir@cs.umd.edu

J. Y.C. Wan
Dept. of Computer Science
and UMIACS
University of Maryland
College Park, MD 20742
ycwan@cs.umd.edu

http://bourbon.usc.edu/iml/bistro

Abstract

Data collection (or *uploading*) is an inherent part of numerous digital government applications and is an important problem, in general. In this paper we describe our recent research results in the development of Bistro, a scalable and secure architecture designed for collection of data over the Internet for digital government applications.

1 Introduction

Hotspots are a major obstacle to achieving scalability in the Internet; they are usually caused by either high demand for some data or high demand for a certain service. At the application layer, hotspot problems have traditionally been dealt with using some combination of increasing capacity, spreading the load over time and/or space, and changing the workload. Some examples of these are data replication (web caching, ftp mirroring), data replacement (multi-resolution images, video), service replication (DNS lookup, Network Time Protocol), and server push (news or software distribution).

These classes of solutions have been studied in the context of applications using the following types of communication: (a) one-to-many (data travels primarily from a server to multiple clients, e.g., web download, software distribution, video-on-demand); (b) many-to-many (data travels between multiple clients, through either a centralized or a distributed server, e.g., chat rooms, video conferencing); and (c) one-to-one (data travels between two clients, e.g., e-mail, e-talk). However, to the best of our knowledge there is no existing work, except ours on making applications using *many-to-one* communication scalable and efficient; existing solutions, such as web based uploads, simply

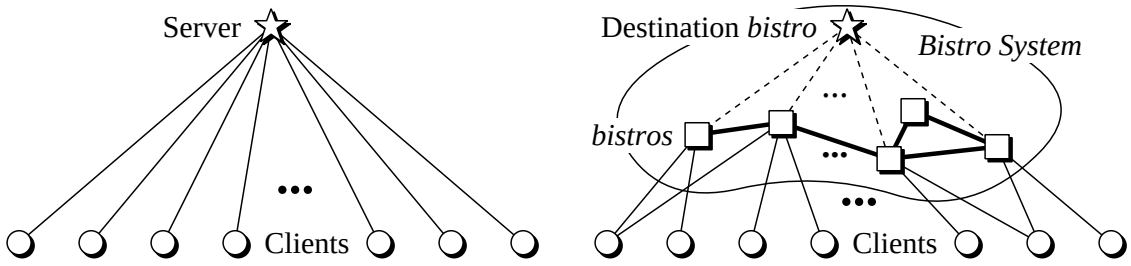
*This work is supported in part by the NSF Digital Government Grant EIA-0091474 and the NSF CCR-0113192 grant.

†This work was done in part while the authors were with the Department of Computer Science and UMIACS, University of Maryland at College Park.

use many independent one-to-one transfers. This corresponds to an important class of applications, whose examples include a large number digital government applications.

Specifically, government at all levels is a major *collector* and provider of data, and there are clear benefits to disseminating and collecting data over the Internet, given its existing large-scale infrastructure and wide-spread reach in commercial, private, and government domains. In this project, we focus on the *collection of data over the Internet*, and specifically, on the *scalability* and *security* issues which arise in the context of Internet-based massive data collection applications. By data collection, we mean applications such as the Internal Revenue Service (IRS) applications with respect to electronic submission of income tax forms. Briefly other such applications are as follows. The Integrated Justice Information Technology Initiative facilitates information sharing among state, local, and tribal justice components. An integrated (global) information sharing system involves collection, analysis, and dissemination of criminal data. Clearly, in order to facilitate such a system one must provide a *scalable* and *secure* infrastructure for collection of data. Furthermore, a number of government agencies (e.g., NSF, NIH) support research activities, where the funds are awarded through a grant proposal process, with deadlines imposed on submission dates. The entire process involves not only submission of proposals, which can involve fairly large data sizes, but also a review process, a reporting process (after the grant is awarded), and possibly a results dissemination process. All these processes involve a data collection step. Lastly, digital democracy applications, such as online voting during federal, state, or local elections, constitute another set of massive upload applications. Of course, there are numerous other examples of digital government applications with large-scale data collection needs.

Recently we proposed Bistro [BCC+00, CCG+01], a framework for building scalable and secure wide-area digital government *upload* applications. Briefly, the Bistro upload architecture works as follows. Given a large number of



(a) upload without the *Bistro System* (b) upload with the *Bistro System*

Figure 1: Upload Problem.

clients that need to upload their data by a given deadline to a given destination server (refer to Figure 1(a)), the Bistro architecture breaks the upload problem into three steps (refer to Figure 1(b)):

- *Step 1*, the timestamp step, which must be accomplished prior to the deadline for clients to submit their data to the destination server. In this step, each client sends to the server a message digest of their data and in return receives a secure timestamp ticket from the destination server as a receipt indicating that the client made the deadline for data submission. The purpose of this step is to ensure that the client makes the deadline *without* having to transfer their data which is significantly larger than a message digest and might take a long time to transfer during high loads which are bound to occur around the deadline time. It is also intended to ensure that the client (or an intermediate bistro used in Step 2 below) does not change their data after receiving the timestamp ticket (hence the sending of the message digest to the destination server). All other steps can occur before or after the deadline.
- *Step 2*, the transfer of data from clients to intermediate hosts, termed bistros. This results in a low data transfer response time for clients since (a) the load of many clients is distributed among multiple bistros and (b) a good

or near-by bistro can be selected for each client to improve data transfer performance. Since the bistros are not trusted entities (unlike the destination server), the data is encrypted by the client prior to the transfer.

- *Step 3*, the collection of data by the destination server from the bistros. The destination server determines when and how the data is collected in order to avoid hotspots around the destination server (i.e., the original problem of having many sources transfer their data to the same server around the same time). Once the destination server collects all the data, it can decrypt it, recompute message digests, and verify that no changes were made to a client's data (either by the client or by one of the intermediate bistros) after the timestamp ticket was issued.

A summary of main advantages of this architecture is as follows: (1) hotspots can be eliminated around the server because the transfer of data is decoupled from making of the deadline, (2) clients can receive good performance since they can be dispersed among many bistros and each one can be directed to the "best" bistro for that client, and (3) the destination server can minimize the amount of time it takes to collect all the data since now it is in control of when and how to do it (i.e., Bistro employs a server pull).

Our main research activities within the Bistro framework have been along the above described three steps. This paper focuses on recent research results in each of these steps. These results are described in the following section.

2 Overview of Results

In this section we give an overview of our current results for data collection over the Internet for digital government applications using the Bistro framework [CCG+01]. We organize these results using the three step approach described in Section 1.

2.1 Secure Timestamp (Step 1)

Provision of a secure timestamp in Bistro involves the use of digital signatures. A digital signature is an important type of authentication in a *public-key* (or asymmetric) cryptographic system, and it is in wide use [Sch96, Sta99]. If Alice would like to send an authenticated (but not encrypted) message M to Bob, Alice can compute a digital signature from M, concatenate M with the digital signature, and send it to Bob. Bob, with possession of Alice's public key, can verify the following important security properties of the message.

- *Integrity* – that not a single bit in the message has been altered.
- *Authentication* – that the message was truly sent by Alice.
- *Nonrepudiation* – that Alice cannot deny that she has sent the message.

Another important property of a digital signature is that it is *not* vulnerable to the so-called *man-in-the-middle* attack, i.e., no one (other than Alice) can change a single bit of either M or the digital signature of M without Bob noticing that the message has been altered.

In a client/server-based application, a server, which offers a set of services, can play the role of Alice and a client can play the role of Bob. Often, a client would like to obtain a receipt from the server, describing the service rendered, and signifying the completion of the prescribed transaction. A digital signature can act as such a receipt due to the nice security properties listed above. Although we focus our efforts on digital government applications, there are many client/server-based applications where a digital signature is desirable. For example, in a lottery ticket selling service (or a concert tickets purchasing priority numbers issuing service), a server can timestamp and digitally sign

each ticket it issues; in a pay-per-view stock tip service, a server can generate the latest report on a stock symbol from its database and digitally sign the report; in an income tax form collection service, a proposal collection service, a conference paper collection service, or a bid collection service for contract bidding, a server can generate a timestamp and send the digitally signed timestamp as proof that a client's submission has been received and that the client has made a deadline [Gol].

A typical application is illustrated in Figure 2. In Figure 2(a), a server is shown to provide documents to a large number of clients spread across the network, such as the Internet. (This can be generalized to multiple services and multiple/mirrored servers). Each client j sends a request for a document, I_j, to the server. The server digitally signs document I_j, to produce DS[I_j] and sends the document together with the digital signature to client j (refer to Figures 2(b) and (c)).

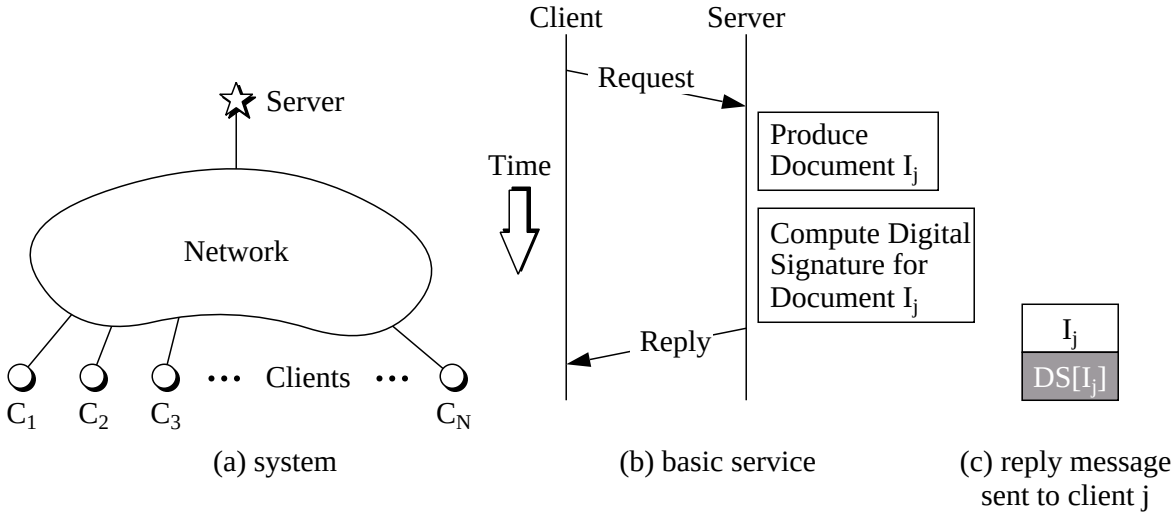


Figure 2: Digital Signatures Problem.

The main problem here is that the digital signature process is very *computationally expensive*. Therefore, if the particular service is very popular, under high loads, the response time for a client can be very large. Each of the applications mentioned above corresponds to a *real-life event* which may experience high demands at certain times. For example, submission of income tax forms are often done close to the deadline.

The main focus of this part of our work is to examine ways of *reducing a client's response time when the load on the server is high*. And, specifically, to reduce the computational needs of a server due to digital signatures under high workloads. We explore the use of batching schemes and show that, using standard cryptographic techniques, we can significantly improve the performance of a server under high load. That is, even under *high load* (near 100% utilization), the server can *keep up with the demands without sacrificing security* (while keeping computational and networking overhead at a minimal). We demonstrate the effectiveness of these schemes by developing an analytical model, validating this model against emulation and simulation studies, and comparing the performance of the batching schemes against a non-batched system, using the analytical model. We also establish stability conditions for the batching schemes. From the stability conditions, it is fairly easy to see that as the document sizes grow, the performance of the server will be limited by hash functions calculations, and the benefit of batching will diminish. Furthermore, we have shown that significant computational benefits can be obtained from batching schemes without significant increases in the amount of additional information that needs to be sent to the clients. In summary, for applications such as the ones mentioned in [CCGK01] which require secure timestamps, batch signing can relieve the CPU bottleneck at the server.

2.2 Transfer of Data from Clients to Intermediaries (Step 2)

In the Bistro framework we employ the use of intermediaries, termed bistros, for improving the efficiency and scalability of uploads. Consequently, appropriate assignment of clients to bistros has a significant effect on the performance of upload applications and thus constitutes an important research problem. Therefore, in this part of our work we focus on the assignment of clients to bistros problem and on a performance study which demonstrates the potential performance gains of the Bistro framework and gives insight into the general upload problem.

The transfer of data from clients to intermediaries problem, for the purposes of this work, can be reduced to (a) assignment of clients to bistros participating in a particular upload event and possibly (b) placement, i.e., choosing which bistros should participate in a particular upload event, if such a choice is possible. Both problems are NP-complete [BCC+00]. Given that the assignment problem is sufficiently difficult, thus far, for the most part, we focused on a quantitative study of the *assignment* problem, and only briefly investigated potential benefits of better placement. Our work illustrates that both have a significant effect on the system's performance. Thus, the main contribution of this work thus far is that it is the first performance study of a scalable and efficient solution to the deadline-driven upload problem. Other contributions of this work are as follows: (a) a quantitative performance study of this problem; (b) development of an approximation to a lower bound on this problem (for comparison purposes); and (c) insight into the general upload problem and characterization of potential performance gains of the Bistro framework.

2.3 Data Collection (Step 3)

Performance of the data collection step is the focus of this part of our work. Specifically, we focus on the collection of reasonably large amounts of data, such as in the online tax submission example given above which can easily result in approximately 10 Terabytes of data corresponding to individual tax forms alone (business tax returns can be significantly larger). In such applications, long transfer times between one or more of the hosts (holding this data) and the destination server can significantly prolong the amount of time it takes to complete the data collection process. Such long transfer times can be the result of poor connectivity between a pair of hosts, or it can be due to wide-area network congestion conditions, e.g., due to having to transfer data over one or more (so-called) peering points whose congestion is often cited as cause of delay in wide-area data transfers [Lei01]. Given the current state of IP routing, congestion conditions may not necessarily result in a change of routes between a pair of hosts, even if alternate routes exist.

Thus, we consider *application-level* approaches to improving performance of large-scale data collection. We do this in the context of the digital government applications and the Bistro upload framework. However, one could consider other applications where such improvements in data transfer 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 [FK98]. Another example is data mining applications where large amounts of data may need to be transferred to a particular server for analysis purposes.

Consequently, in this work we consider large-scale data collection from a set of source hosts (bistros) to the destination host (destination bistro) which we term the *data collection problem*. The data collection problem is a non-trivial one because the issue is not only to avoid congested link(s), but to devise a *coordinated* transfer schedule which would afford maximum possible utilization of available network resources between multiple sources and the destination.

In this work, we show that "indirect" and furthermore "coordinated" methods which re-route data through other hosts in a coordinated fashion can result in a significant performance improvement as compared to "direct" methods (which send the data directly to its final destination). Consequently, our focus in this work is on development of

algorithms for *indirect coordinated* transfer methods for the data collection problem.

The contributions of this work thus far are as follows. We proposed novel algorithms for the large-scale data collection problem, intended for an IP-type network. The main benefit of these methods is application-level coordinated re-routing of large-scale data transfers around congestion spots or poor connectivity between a source of data and its final destination. We evaluated the performance of these algorithms and showed that the indirect methods perform significantly better than direct methods. Specifically we showed *one to two orders of magnitude improvement* under high congestion conditions (without losses in performance under no congestion conditions). These improvements are achieved under low storage requirement overheads and without significant detrimental effects on other network traffic.

3 Conclusions

Ideas and results described in Section 2 illustrate that significant performance improvements are possible in collection of data over the Internet for digital government applications. We believe that much work remains to be done in bringing these ideas to their full potential as well as in investigating their applicability to a broader class of digital government applications.

References

[BCC+00] 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.

[CCG+01] W.C. Cheng, C.F. Chou, L. Golubchik, S. Khuller, and H. Samet. Scalable data collection for internet-based digital government applications. In *1st National Conference on Digital Government Research*, pages 108–113, Los Angeles, CA, May 2001.

[CCGK01] W. C. Cheng, C.-F. Chou, L. Golubchik, and S. Khuller. A secure and scalable wide-area upload service. In *Proceedings of the 2nd International Conference on Internet Computing, Volume 2*, pages 733–739, June 2001.

[FK98] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[Gol] L. Golubchik. Scalable data collection for Internet-based Digital Government applications. *Advances in Digital Government: Systems, Human Factors, and Policy*.

[Lei01] T. Leighton. *The Challenges of Delivering Content on the Internet*. Keynote address at the ACM SIGMETRICS 2001 Conference, Cambridge, Massachusetts, June 2001.

[Sch96] B. Schneier. *Applied Cryptography, Second Edition*. Wiley, 1996.

[Sta99] W. Stallings. *Cryptography and Network Security: Principles and Practice, 2nd Edition*. Prentice Hall, 1999.