Bistro

a Platform for Building Scalable Wide-Area *Upload* Applications

Prof. Leana Golubchik

Email: leana@cs.usc.edu
URL: http://cs.usc.edu/~leana
# Scalable Data Transfer Applications

End-system / Application-level

<table>
<thead>
<tr>
<th># of Receivers</th>
<th>One</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>ftp traditional apps ...</td>
<td>web downloads software distribution video-on-demand server push ...</td>
</tr>
<tr>
<td>Many</td>
<td><strong>Bistro!!</strong></td>
<td>chat rooms video conferencing multiplayer games ...</td>
</tr>
</tbody>
</table>

## # of Senders

- One
- Many

**Leana Golubchik**  
Internet Multimedia Lab  
CS / EE-S / IMSC / ISI  
University of Southern California

**USC Internet Multimedia Lab (IML)**
Who Is Working on Uploads?

To the best of our knowledge, there is no existing work on making \textit{many-to-one} communication at the \textit{application} layer \textit{scalable} and \textit{efficient}
What Are *Upload* Applications?

- **Hard deadlines**
  - IRS income tax submission
  - Paper submission
  - Real-life events

- **No hard deadlines**
  - Internet-based storage
  - Data warehousing
Why is *Upload* Different?

- many-to-one data transfer
- read vs. *write*
  - traditional solution such as replication of data (caching), replacement of data, etc. won’t help
  - fault tolerance, *security*
- contention for service rather than data
- data consumed later (*will exploit this*)

- replication of services and resources for a single event is expensive, inflexible, & not scalable
Traditional Approaches
(at the application layer)

- Increase capacity
- Spread the load ... over time, space, or both
- Change the workload

Examples
- data replication  ftp mirroring, web caching
- data replacement  multi-resolution images, video
- service replication  DNS lookup, NTP
- server push  news download, software distribution
Our Goals

- A single infrastructure (termed Bistro) for all *data collection* needs
  - good performance (for both service providers and users)
  - scalable (shares resources among all service providers)
  - secure (one service provider does not have to trust another)
Current State

- Independent data transfers over the Internet, i.e., TCP/IP
  - TCP/IP shares bandwidth fairly
  - Individual clients experience poor performance when number of clients is large (if transfer time is long enough to see other connections)
  - TCP/IP is here to stay

Not scalable!
Key Observations
(applications with deadlines)

→ Existence of hot spots in uploads is largely due to *approaching deadlines*

→ Exacerbated by *long transfer times*

→ Problem: too much data ... too little time ...
What is actually needed is an **assurance** that specific data was submitted before a specific time; then the transfer of that data needs to be done in a timely manner, but does *not* have to occur by the deadline.

*the data may not be consumed by the server right away*
Solution with *Bistro*

- **Before deadline:**

- **Traffic at/near *Destination Server***:
A Solution to *Upload with Deadlines*

Real-time timestamp

Low-latency upload to *any* intermediary (*commit*)

Timely transfer to final destination (large scale *data transfer*)
Advantages of Bistro

- Shares resources and a *single* infrastructure
- Replaces a traditionally *synchronized client push* solution with a *non-synchronized* combination of *client-push* and *server-pull*
- Eliminates hot spots by spreading most of the demand on the server *over time*, by making the actual data transfer *independent* of the deadline
- Deployable *today*, i.e., no change required inside the network
- *Gradual* deployment over a public, private, or mixed infrastructure of hosts
- More *dynamic* and therefore more *adaptive* to system and network conditions
Deployment Issues

Public vs. private infrastructure

- good Bistro
- malicious Bistro
Bistro

**Over public infrastructure trust issues (e.g., IRS)**

**Legend:**
- **EID**: Event ID
- **K\_pub**: Event Public Key
- **K\_priv**: Event Private Key
- **K\_pub^X**: Bistro X Public Key
- **K\_priv^X**: Bistro X Private Key
- **T**: Data to upload
- **h()**: Message Digest
- **σ**: Timestamp
- **ξ**: Ticket
- **R**: Receipt

---

**Event Owner** (IRS)

**Client** (a Taxpayer)

**Destination** *Bistro D*

**Any** *Bistro X*

**Event Creation**

\[ K\_pub, EID \]

\[ h(T), Email \]

\[ \xi = K\_priv(h(T), \sigma) \]

\[ K\_pub(K_{ses, \xi}), K\_ses(T), EID \]

\[ R = K\_X^{priv}(K\_pub(K_{ses, \xi})) \]

\[ R, K\_pub^X \]

\[ Retrieve(EID, R) \]

\[ K\_pub(K_{ses, \xi}), K\_ses(T), EID \]

**Deadline**

**Time**
Who is Trusted with What?

Event Owner
- trusts the Destination Bistro for this event

End User
- trusts its Client software
- trusts the Destination Bistro for this event

Destination Bistro
- generates \((K_{pub}^{Bistro} , K_{priv}^{Bistro})\) for this event

Any Bistro X
- generates \((K_{pub}^{X} , K_{priv}^{X})\) for any event (only once)
Some Issues

Mirroring

The Destination *bistro* can also be *Any bistro X*

Issues

- the usual *public key distribution* problem
- no client authentication (e.g., multiple submissions from the same user)
- single point of attack
- event owner doesn’t want to use the Destination *bistro*’s public key crypto system
Contributions Thus Far

- First effort to study many-to-one communication problem at the *application* layer & attempt at stating fundamental obstacles
- Proposed a reasonably general framework
- Proposed solutions to all parts of the problem
- Suggested some open problems
Related Work

- Akamai and other content distribution networks
- Napster
- A variety of server selection problems
- Internet security
Related Work (Cont...)

- Many-to-one communication at IP level & within Active network framework
  - Gathercast [Badrinath & Sudame 98]
  - Concat [Calvert et al. 00]

- Wide area applications
  - wide-area download applications: e.g., Akamai [Karger et al. 97]
  - Napster type systems, e.g., [Kong & Ghosal 99]
  - application layer multicast: e.g., [Chu et al. 00]

- Client-side server selection
  - statistical: e.g., [Seshnm et al. 97]
  - dynamic: e.g., [Carter & Crovella 97] [Sayal et al. 98] [Dykes et al. 00]
Related Work (Cont...)

- Application level re-routing
  - alternate paths [Savage et al. 99]
  - Detour [Savage et al. 99]
  - RON: resilient overlay network [Andersen et al. 01]

- Online batch-based digital signature schemes
  - modification on cryptographic algorithm [A. Fiat 89]
  - one-time signatures used in secret key system [Lamport 79, Merkle 88]
Vision

- A *bistro* in every administrative domain e.g., co-located with web servers
- Entire network of *bistros* collects data for one application one day and for another application the next day
- Use the *Bistro* infrastructure for other large scale data gathering, transfer, and storage needs
Participants

- Faculty Members:
  - Leana Golubchik
  - Samir Khuller (UMD)
  - Cheng-Fu Chou (NTU)

- Research Staff:
  - William C. Cheng

- Students:
  - Leslie Cheung
  - Yung-Chun Wan (UMD)
  - Yan Yang

Contact Information

Prof. Leana Golubchik
CS / EE-S / IMSC / ISI
University of Southern California
Los Angeles, CA 90089

Email: leana@cs.usc.edu
Voice: (213) 740-4524
Fax: (213) 740-7285
URL: http://cs.usc.edu/~leana

Project URL: http://bourbon.usc.edu/iml/bistro
Lab URL: http://bourbon.usc.edu/iml