CS530
Scalable Wide-area Upload
[Bistro00]
Bill Cheng

http://merlot.usc.edu/cs530-s10
Bistro

a Platform for Building Scalable Wide-Area Upload Applications

I Can Hold Out For A Couple More Minutes!
## Scalable Data Transfer Applications

*End-system / Application-level*

<table>
<thead>
<tr>
<th># of Senders</th>
<th># of Receivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>Many</td>
<td>Many</td>
</tr>
</tbody>
</table>
## Scalable Data Transfer Applications

*End-system / Application-level*

<table>
<thead>
<tr>
<th># of Senders</th>
<th># of Receivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td>Many</td>
<td>Many</td>
</tr>
<tr>
<td>One</td>
<td>ftp traditional apps ...</td>
</tr>
<tr>
<td>Many</td>
<td></td>
</tr>
</tbody>
</table>
### Scalable Data Transfer Applications

**End-system / Application-level**

<table>
<thead>
<tr>
<th># of Senders</th>
<th># of Receivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td>One</td>
</tr>
<tr>
<td></td>
<td>ftp traditional_apps ...</td>
</tr>
<tr>
<td>Many</td>
<td>Many</td>
</tr>
<tr>
<td></td>
<td>web downloads software distribution video-on-demand server push ...</td>
</tr>
</tbody>
</table>

Copyright © William C. Cheng
## Scalable Data Transfer Applications

*End-system / Application-level*

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

Copyright © William C. Cheng
## Scalable Data Transfer Applications

*End-system / Application-level*

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

To the best of our knowledge, there is no existing work on making *many-to-one* communication at the *application* layer *scalable* and *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
Traditional Approaches (Cont...)

*Example: Akamai*

- Relieve web download hotspots through data replication (caching)
- Use their own network of servers, with strategic placement of servers around the world
- Clients include: Microsoft, Paramount, Wired, CBS Sports, Nike, BBC America, Apple, ...

- Why are there hotspots?
  - real-life events
  - availability of new data

- > 2700 servers
- > 45 countries
- > 150 networks
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 of Affairs for Uploading

- Independent data transfers over the Internet, i.e., TCP/IP
  - TCP/IP shares bandwidth fairly
  - Individual clients experience poor performance when the 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 ...

![Diagram](image-url)
What is actually needed is an assurance that specific data was submitted before a specific time

- i.e., we need a commitment of what and when a submission took place

Then the transfer of that data needs to be done in a timely manner, but does not have to occur by the deadline

- unlink downloads, the data may not be consumed at the server right away
- if a piece of data arrives after the deadline, we just need to guarantee that it’s exactly the same piece of data that was committed before the deadline
Solution with Bistro

- **Before deadline:**

- **Traffic at/near Destination Server:**

  with contemporary cryptography technology, hash size is constant (10s of bytes), no matter how big a document is.
A Solution to *Upload with Deadlines*

(a) upload without *Bistro*  
(b) upload with the *Bistro System* after *Bistro software* is installed on the Server

- A *bistro* is like an *e-Post Office*, built to handle *certified e-submissions*
- A bistro can be installed on an IRS server or a tax partner’s server
- Note:
  - Picture above is for a *single event*, *e.g.*, *2005 personal income tax submission*
  - Multiple events may be going on concurrently or overlapping, each with a different destination server
A Solution to *Upload with Deadlines*

**Step 1: Real-time fingerprinting & timestamp**
- A client generates a *fingerprint* for the document (tax return)
- Destination *bistro* issues a *timestamped* and *certified e-ticket*
A Solution to *Upload with Deadlines*

(a) upload without *Bistro*

(b) upload with the *Bistro System* after *Bistro* software is installed on the Server

**Real-time timestamp**

**Step 2: Low-latency upload to *any* intermediary (commit) *(client-push)*

- A client *verifies* the *digital signature* on the *e-ticket*, *encrypts* the document, and upload the *encrypted document* to any *bistro* (or a designated bistro for a tax partner)
A Solution to *Upload with Deadlines*

- **Step 1:** Real-time fingerprinting & timestamp
- **Step 2:** Low-latency upload to *any* intermediary (commit) *(client-push)*
- **Step 3:** Timely transfer to final destination *(large scale data transfer)* *(server pull)*
Bistro Protocol Summary [Cheng01a]

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Creator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation</td>
<td>Event Operator</td>
<td>(IRS)</td>
</tr>
<tr>
<td>Any</td>
<td>Bistro X</td>
<td>X</td>
</tr>
<tr>
<td>Destination</td>
<td>Bistro D</td>
<td></td>
</tr>
<tr>
<td>Retrieve(EID,R)</td>
<td>^K pub, EID</td>
<td></td>
</tr>
<tr>
<td>Legend:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EID : Event ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K pub : Event Public Key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K priv : Event Private Key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K pub : Bistro X Public Key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K priv : Bistro X Private Key</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T : Data to upload</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h() : Message Digest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>σ : Timestamp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ξ : Ticket</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R : Receipt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K[ ] : Encryption</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Client (a Taxpayer) → Any Bistro X → Destination Bistro D → Event Operator (IRS)

(1) Timestamp
h(T), Email
ξ ≡ K priv[h(T), Email, σ]

(2) Commit
K pub[K ses, ξ], K ses[T], EID
R ≡ K priv[K pub[K ses, ξ]], R, K pub

(3) Transfer
Retrieved(EID, R)
K pub[K ses, ξ], K ses[T], EID
Who is Trusted with What?

- **Event Operator (IRS)**
  - trusts the Destination **Bistro** for *this* event

- **End User (tax payer)**
  - trusts its Client software
  - trusts the Destination **Bistro** for *this* event

- **Destination **Bistro**
  - trusts the Bistro software to generate a pair of public and private keys \( (K_{pub}, K_{priv}) \) for *this* event

*analog to certified mail with untrusted post-office*
**Bistro FAQ**

Why do you need step (2)? Why can’t the destination server get the document directly from a client in step (3)?

- A client can be behind a firewall or a client’s machine can be turned off.
- A *bistro* is always on the public Internet, and may be subject to attacks. Therefore, all documents on a *bistro* must be encrypted.

Why did you show that step (2) is done before the deadline?

- Step (2) is the *commit* step, it does not need to be done before the deadline since the only transaction that is required to be completed before the deadline is step (1). However, to complete a client’s transaction (so that the client can leave or shutdown its PC), we must push the encrypted data out of the client’s PC.
- Since there can be many *bistros*, this will not cause a traffic jam. Also, most of the data transfers during this step are localized.
Bistro FAQ (Cont...)

Can a fingerprint be forged?

- SHA1 is the state-of-the-art electronic fingerprinting algorithm. It generates a 160-bit fingerprint for any-size document. If you modify a single bit in a document, the new document has a completely different fingerprint. There is no known algorithm that can forge a SHA1 fingerprint while maintaining the integrity of a document. (The Bistro system is not tied to a particular fingerprinting algorithm. Please see below.)

Can the destination server be under denial-of-service attack?

- Yes. That’s one weakness of the Internet. However, you can setup mirrors for the destination server by copying the credentials of the destination server onto alternative servers. Nevertheless, in the current Bistro system, this needs to be done ahead of time.

How secure is the encryption? Can it be cracked?

- The strength of encryption is usual a function of the algorithm and key size. The Bistro system is not tied to a particular algorithm or key size. It lets the event operator choose these at the time an event is setup. As new and more secure algorithms become available, the system will need to be upgraded to support them.
How big a server do we need in order to give out so many timestamped and certified e-tickets in a short period of time?

- To certify an e-ticket requires a digital signature, and signing digital signatures is a time consuming process. But, as it turns out, digital signatures can be **batched**. We have developed **batch signing schemes** (please see our publications) to remove this limitation. Now we can sign as many as it comes.

What about client authentication? Do we know, with certainty, who is submitting a tax return?

- As in the current system, you do not know who is submitting a tax return at the time of submission. Even with paper submission, it is very difficult to verify a signature. **Client authentication** is outside the scope of the **Bistro** system.

- If a tax payer uses a tax partner’s service to submit his/her tax return, it would be easy to authenticate a tax partner. Each tax partner can independently generate a pair of public and private keys (according to the specifications from IRS) and send the public key to IRS. Each submission can be digitally signed with the tax partner’s private key. IRS can verify the digital signature using the corresponding public key.
Opportunities to Speed up Data Transfers

- Host
- Router
- Network Link
- Congested Link
- IP Route
- Application Level Re-routing

Diagram: Three nodes labeled as $S_1$, $S_2$, and $S_3$ connected through network links. The diagram illustrates the concept of re-routing to speed up data transfers.
Opportunities to Speed up Data Transfers (Cont...)

Scenarios:
- X & Y send simultaneously to D -- 2 units of time
- X sends to D, then Y sends to D -- 2 units of time
- X & Y send simultaneously to Z then to D -- 3 units of time
- X sends to D // Y sends to Z then to D -- 1.5 units of time
- ??? -- 1.2 units of time
Advantages of Bistro

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

- A bistro in every administrative domain, e.g., co-located with web servers or mail servers.

- Entire network of bistros collects data for one application/agency one day and for another application/agency the next day.

- Use the Bistro infrastructure for other large scale data gathering, transfer, and storage needs.
CS530
Bistro Improvements
Bill Cheng

http://merlot.usc.edu/cs530-s10
Some Research Problems

- Resource location and discovery
- Placement and assignment
- Security
- Large scale data transfer

Diagram:
- Bistro System
- Destination *bistro* (i.e., Server)
- *bistros*
- Data Flow
- Clients
Bistro

<table>
<thead>
<tr>
<th>Event Owner (IRS)</th>
<th>Client (a Taxpayer)</th>
<th>Destination Bistro D</th>
<th>Any Bistro X</th>
</tr>
</thead>
<tbody>
<tr>
<td>h(T), Email</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\xi = K_{\text{priv}}(h(T),\sigma))</td>
<td>R = K^X_{\text{priv}}(K_{\text{pub}}(K_{\text{ses}}(\xi)), K_{\text{ses}}(T), EID)</td>
<td>K_{\text{pub}}(K_{\text{ses}}(\xi)), K_{\text{ses}}(T), EID</td>
<td>K_{\text{pub}}(K_{\text{ses}}(\xi)), K_{\text{ses}}(T), EID</td>
</tr>
</tbody>
</table>

**Legend:**
- **EID**: Event ID
- **K_{\text{pub}}**: Event Public Key
- **K_{\text{priv}}**: Event Private Key
- **K^X_{\text{pub}}**: Bistro X Public Key
- **K^X_{\text{priv}}**: Bistro X Private Key
- **T**: Data to upload
- **h()**: Message Digest
- **\sigma**: Timestamp
- **\xi**: Ticket
- **R**: Receipt

**Deadline**
Online Digital Signatures

Why digital signature?

- integrity
- authentication
- nonrepudiation
Real-Time Timestamp

Using digital signature to generate real-time timestamp

(a) system

(b) basic service

(c) reply message sent to client j

high cost of modular of arithmetic
Our Approach

No batching scheme

Documents

Signing Server

Send $I_j + DS[I_j]$ to each Client$_j$ for $1 \leq j \leq B$

Simple batching scheme

Requests are queued behind the Gate

Batch Signing Server

Gate

$D = H(I_1) + H(I_2) + ... + H(I_B)$

Send $D + DS[D] + I_j$ to each Client$_j$ for $1 \leq j \leq B$

extra information to be sent to clients
Our Approach (Cont...)

Tree-based batching scheme

\[ R = H(H(H(I_1)+H(I_2))+H(H(I_3)+H(I_4))) \]

- Send to C_1:
  - R
  - DS[R]
  - H(I_2)
  - H(H(I_3)+H(I_4))
  - I_1

- Send to C_2:
  - R
  - DS[R]
  - H(I_1)
  - H(H(I_3)+H(I_4))
  - I_2

- Send to C_3:
  - R
  - DS[R]
  - H(I_4)
  - H(H(I_1)+H(I_2))
  - I_3

- Send to C_4:
  - R
  - DS[R]
  - H(I_3)
  - H(H(I_1)+H(I_2))
  - I_4
Batch-based schemes do reduce a server’s CPU load (where hashing is not the dominant factor)
Performance Evaluation

- Batching schemes have considerable advantage but cost relatively little (where hashing is not the dominant factor)

- avg document size: 1K
- simple batching

![Graph showing mean response time and network overhead factor](image)

- Document size: 1K

- simple batching
- tree-based batching
**Legend:**
- **EID**: Event ID
- **K_{pub}**: Event Public Key
- **K_{priv}**: Event Private Key
- **K^X_{pub}**: Bistro X Public Key
- **K^X_{priv}**: Bistro X Private Key
- **T**: Data to upload
- **h()**: Message Digest
- **σ**: Timestamp
- **ξ**: Ticket
- **R**: Receipt

---

**Event Owner (IRS)**

**Client (a Taxpayer)**

**Deadline**

**Destination Bistro D**

**Any Bistro X**

**Commit**

- \( h(T), \text{Email} \) → \( K_{priv}(h(T), \sigma) \)
- \( R \equiv K^X_{priv}(K_{pub}(K_{ses}(\xi))) \), \( K^X_{pub}(K_{ses}(T), \text{EID}) \)
- \( K_{pub}(K_{ses}(\xi)), K_{ses}(T), \text{EID} \)
- \( K_{pub}(K_{ses}(\xi)), K_{ses}(T), \text{EID} \)
- \( \text{Retrieve}(\text{EID}, R) \)

---

**Bistro**
Commit Problem

Extreme Cases

- Final destination is the only bistro
- All hosts are bistros
- Each organization has a local bistro (same granularity as NNTP servers, DNS servers, etc.); in this case commit problem still non-trivial if the local bistro is not part of the public Internet
Commit Problem

*Middle Ground*

**Assignment problem**

- bistros are fixed & the difficulty is in assigning clients to the bistros

**Placement or selection (plus assignment) problem**

- location of bistros is flexible
- choose $M$ out of $N$ bistros as well as assign clients to chosen bistros

*Why is this different from downloads?*
Performance Study

Simulation setup (using ns2 & GT-ITM)

- transit-stub graph with 152 nodes
- 2 transit domains, with avg 4 nodes each, edge between pair of nodes with prob 0.6 & each node having 3 stub domains connected
- stub domains have on avg 6 nodes each, edge between pair of nodes with prob 0.2
- capacity of transit-transit edge is 1 Mbit/s
- capacity of transit-stub or stub-stub edge is 256 Kbits/s
- 96 simultaneous uploads with files unif. distr. between 100 KBytes & 2 MBytes
- low background load (30%); high background load (70%)
Performance Study

⇒ Note: *seq.* uploads to *single* server should be approx 3000 sec, and avg. transfer time of one client should be approx 33 sec

⇒ Note: *simultaneous* uploads to *single* server takes approx 3000 sec, but avg. transfer time of one client takes approx 2000 sec

⇒ Performance metrics used
  - mean transfer time over all clients
  - total (or maximum) transfer time

⇒ Policies
  - random, ping-v, ping-m
  - unrealistic heuristic (approx. lower bound)
Performance Results

- 30% background load
- identical fixed placement (determined by our heuristic algorithm)

- 30% background load
- identical random placement

Performance gains mainly due to parallelism
Performance Results

(a) 70% background load
identical fixed placement (determined by our heuristic algorithm)

(b) 70% background load
identical fixed placement (determined by our heuristic algorithm)

(c) 70% background load
identical random placement

(d) 70% background load
identical random placement
Performance Results

- 30% background load
- each scheme uses its own placement and assignment policies

- 70% background load
- each scheme uses its own placement and assignment policies

30% background load each scheme uses its own placement and assignment policies

70% background load each scheme uses its own placement and assignment policies
Bistro

Event Owner (IRS)  Client (a Taxpayer)  Destination Bistro D  Any Bistro X

Legend:
- **EID**: Event ID
- **K_{pub}**: Event Public Key
- **K_{priv}**: Event Private Key
- **K^X_{pub}**: Bistro X Public Key
- **K^X_{priv}**: Bistro X Private Key
- **T**: Data to upload
- **h()**: Message Digest
- **σ**: Timestamp
- **ξ**: Ticket
- **R**: Receipt

Date Transfer

\[ R \equiv K^X_{priv}(K^X_{pub}(K_{ses}(ξ))), \]
\[ K^X_{pub}(K_{ses}(ξ)) \]
\[ K_{pub}(K_{ses}(ξ)), K_{ses}(T), EID \]
\[ K_{pub}(K_{ses}(ξ)), K_{ses}(T), EID \]
\[ \text{Retrieve}(EID,R) \]

**Deadline**
Large-scale Data Collection

Destination server needs to collect data from all other bistros but how?

Several simple approaches

- one-by-one poor resource utilization due to non-shared bottleneck link
- all-at-once longer transfer time
- spread-in-time-GT
- concurrent-G network congestion

application level re-routing

- avoid congested links
- devise a coordinated transfer schedule
Opportunities

- Host
- Router
- Network Link
- Congested Link
- IP Route
- Application Level Re-routing
Opportunities (Cont...)

Scenarios:
- X & Y send simultaneously to D -- 2 units of time
- X sends to D, then Y sends to D -- 2 units of time
- X & Y send simultaneously to Z then to D -- 3 units of time
- X sends to D // Y sends to Z then to D -- 1.5 units of time
- ??? -- 1.2 units of time
Transfer To Destination

Simulation setup (using ns2 & GT-ITM)

- 7 other bistros, each with a total amount of data unif. distr. between 25 MBytes & 75 MBytes and the total amount of data in bistros is 350 MBytes.

Performance improvement due to rerouting around congestion
Storage Space Requirements

- needs fairly little additional storage space

pathsyc < 4%, pathmerge and pathdelay < 41%
Effect on Other Traffic

- no significant effect on throughput of other traffic
  
  ($< 17\%$)
Contributions Thus Far

- First effort to study many-to-one communication problem at the \textit{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]
  - Concast [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:
  - Yung-Chun Wan (UMD)

Contact Information

Prof. Leana Golubchik
Computer Science Department
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