

Computer Communications - CSCI 551

2

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

*Bistro*

Copyright © William C. Cheng

Computer Communications - CSCI 551

4

**Scalable Data Transfer Applications**  
End-system / Application-level

|                |      |     |      |
|----------------|------|-----|------|
| # of Receivers |      | One | Many |
|                |      | One | Many |
| # of Senders   | One  | One | Many |
|                | Many | One | Many |

ftp traditional apps

Copyright © William C. Cheng

Computer Communications - CSCI 551

6

**Scalable Data Transfer Applications**  
End-system / Application-level

|                |      |     |      |
|----------------|------|-----|------|
| # of Receivers |      | One | Many |
|                |      | One | Many |
| # of Senders   | One  | One | Many |
|                | Many | One | Many |

web downloads  
video-on-demand  
server push

chat rooms  
video conferencing  
multiplayer games

Copyright © William C. Cheng

Computer Communications - CSCI 551

1

**CS551**

**Scalable Wide-area Upload [Bistro00]**

**Bill Cheng**

<http://merlot.usc.edu/cs551-f12>

Copyright © William C. Cheng

Computer Communications - CSCI 551

3

**Scalable Data Transfer Applications**  
End-system / Application-level

|                |      |     |      |
|----------------|------|-----|------|
| # of Receivers |      | One | Many |
|                |      | One | Many |
| # of Senders   | One  | One | Many |
|                | Many | One | Many |

Copyright © William C. Cheng

Computer Communications - CSCI 551

5

**Scalable Data Transfer Applications**  
End-system / Application-level

|                |      |     |      |
|----------------|------|-----|------|
| # of Receivers |      | One | Many |
|                |      | One | Many |
| # of Senders   | One  | One | Many |
|                | Many | One | Many |

web downloads  
video-on-demand  
server push

ftp traditional apps

Copyright © William C. Cheng

Copyright © William C. Cheng

### 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
  - > 2700 servers
  - > 45 countries
  - > 150 networks
- Clients include: Microsoft, Paramount, Wired, CBS Sports, Nike, BBC America, Apple, ...

**Why are there hotspots?**

- real-life events
- availability of new data

12

Copyright © William C. Cheng

### 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

10

Copyright © William C. Cheng

### 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*

8

Copyright © William C. Cheng

### 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

11

Copyright © William C. Cheng

### What Are Upload Applications?

- Hard deadlines: IRS income tax submission
- paper submission
- real-life events: Digital Democracy, Distance Education
- No hard deadlines: Internet-based storage, Data warehousing

Internet-based Storage, Internet-based Computing, E-commerce, Digital Government, Digital Education, Digital Democracy, Data Warehousing

9

Copyright © William C. Cheng

### Scalable Data Transfer Applications

End-system / Application-level

| # of Receivers | # of Senders  |          |   |
|----------------|---|----------|---|
|                | One   | Many     |   |
| Many           | One   | Many     |   |
| Many           | Many  | Bistro!! |   |
| ...            | web downloads, traditional software distribution, video-on-demand server push | ...      | chat rooms, video conferencing, multiplayer games |

7

**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)

Copyright © William C. Cheng

**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 number of clients is large (if transfer time is long enough to see other connections)
- ↳ TCP/IP is here to stay

**Not scalable!**

Copyright © William C. Cheng

**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 ...

Copyright © William C. Cheng

**Key Observations (Cont..)**

(applications with deadlines)

- ↳ 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

Copyright © William C. Cheng

**Solution with *Bistro***

Before deadline:

- ↳ Client 1 (fingerprint) Hash
- ↳ Client N
- ↳ Destination Server
- ↳ Internet bottleneck
- ↳ with contemporary cryptography technology, hash size is constant (10s of bytes), matter how big a document is

Traffic at/near Destination Server:

Copyright © William C. Cheng

**A Solution to Upload with Deadlines**

Destination *bistro* (i.e., Server)

- ↳ upload without *Bistro*
- ↳ upload with the *Bistro* System after *Bistro* software is installed on the Server

Copyright © William C. Cheng

**A Solution to Upload with Deadlines**

Computer Communications - CSC1 551

- ↳ 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

Copyright © William C. Cheng

**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 ...

Copyright © William C. Cheng

**Key Observations (Cont..)**

(applications with deadlines)

- ↳ 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

Copyright © William C. Cheng

Copyright © William C. Cheng

19

### A Solution to Upload with Deadlines

- Step 1: Real-time fingerprinting & timestamp
  - Client generates a fingerprint for the document (tax return)
  - Destination *bistro* issues a timestamp and certified e-ticket
- Step 2: Low-latency upload to *any* intermediary (commit)
  - Real-time timestamp
  - (client-push)
- A client verifies the digital signature on the e-ticket, encrypts the document, and uploads the encrypted document to any *bistro* (or a designated *bistro* for a tax partner)

Computer Communications - CSC1 551

Copyright © William C. Cheng

21

### A Solution to Upload with Deadlines

- Step 1: Real-time fingerprinting & timestamp
  - is installed on the Server after *Bistro* software is uploaded with the *Bistro* System
- Step 2: Low-latency upload to *any* intermediary (commit)
  - (client-push)
- Step 3: Timely transfer to final destination (large scale data transfer) (server pull)

### A Solution to Upload with Deadlines

Computer Communications - CSC1 551

Copyright © William C. Cheng

23

### Who is Trusted with What?

- Event Operator (IRS) trusts the Destination *Bistro* for this event
- End User (tax payer) trusts its Client software
- Bistro* for this event trusts the Destination software
- Destination *Bistro* trusts the *Bistro* software to generate a pair of public and private keys ( $K_{pub}, K_{priv}$ ) for this event

Computer Communications - CSC1 551

Copyright © William C. Cheng

22

### Bistro Protocol Summary [Cheng01a]

**Legend:**

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

Computer Communications - CSC1 551

Copyright © William C. Cheng

24

### 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.
  - 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.

Computer Communications - CSC1 551

Copyright © William C. Cheng

### 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

30

Copyright © William C. Cheng

### 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 -- 3 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

Legend:

- : Host
- : Network
- : Shared point of congestion
- : Link abstraction (label is Name/Capacity)

28

Copyright © William C. Cheng

### Bistro FAQ (Cont...)

- 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 taxpayer 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.

26

Copyright © William C. Cheng

### 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

29

Copyright © William C. Cheng

### Opportunities to Speed up Data Transfers

Legend:

- : Host
- : Router
- ▬ : Network Link
- ▬▬▬ : Congested Link
- ← : IP Route
- ← : Re-routing

27

Copyright © William C. Cheng

### 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 an 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.
- 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.

25

32 *Eun*

### Some Research Problems

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

Destination bistro (i.e., Server)  
bistros  
Clients  
Data Flow

MIT University of Southern California

34 *Eun*

### Online Digital Signatures

Why digital signature?

- integrity
- authentication
- nonrepudiation

**digitally signing**

Alice

**verifying**

Bob

Yes/No

MIT University of Southern California

36 *Eun*

### Our Approach

No batching scheme

Documents  $I_1, \dots, I_{j-1}, I_j, \dots$

Send  $I_j + DS[I_j]$  to each  $Client_j$ ,  $1 \leq j \leq B$

Signing Server

Simple batching scheme

Requests are queued behind the Gate

Batch Signing Gate Server

Send  $D + DS[D] + I_j$  to each  $Client_j$ ,  $1 \leq j \leq B$

Requests are queued behind the Gate

Batch Signing Gate Server

$D = H(I_1) + H(I_2) + \dots + H(I_B)$

extra information to be sent to clients

MIT University of Southern California

31 *Eun*

# Bistro Improvements

## Bill Cheng

<http://merlot.usc.edu/cs551-f12>

Computer Communications - CS51

Copyright © William C. Cheng

33 *Eun*

### Bistro

Event Owner (IRS)

Client (a Taxpayer)

Destination D

Any Bistro X

Deadline

**Timestamp**

$h(T)$ , Email  
 $K_{pub}^{(K_{priv}^{(h(T), \sigma)})}$   
 $K_{priv}^{(K_{pub}^{(h(T), \sigma)})}$   
 $K_{pub}^{(K_{priv}^{(h(T), \sigma)})}$   
 $R = K_{priv}^{(K_{pub}^{(K_{priv}^{(h(T), \sigma)})})}$   
 $R_i, K_{pub}^{(K_{priv}^{(h(T), \sigma)})}$   
 $K_{pub}^{(K_{priv}^{(h(T), \sigma)})}$   
 $K_{priv}^{(K_{pub}^{(h(T), \sigma)})}$   
 $T$ : Data to upload  
 $H$ : Message Digest  
 $\sigma$ : Timestamp  
 $\xi$ : Ticket  
 $R$ : Receipt

Legend:

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

MIT University of Southern California

35 *Eun*

### Real-Time Timestamp

Using digital signature to generate real-time timestamp

(a) system

Client  $C_1, \dots, C_n$

Network

Server

Time

Request

Produce  $I_j$

Document  $I_j$

Compute Digital Signature for Document  $I_j$

Reply

basic service

(c) reply message sent to client  $j$

high cost of modular arithmetic

MIT University of Southern California

**Commit Problem**  
Middle Ground

- Assignment problem
  - bistros are fixed & the difficulty is in assigning clients to the bistros
  - NP-complete for several useful objective functions
- 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

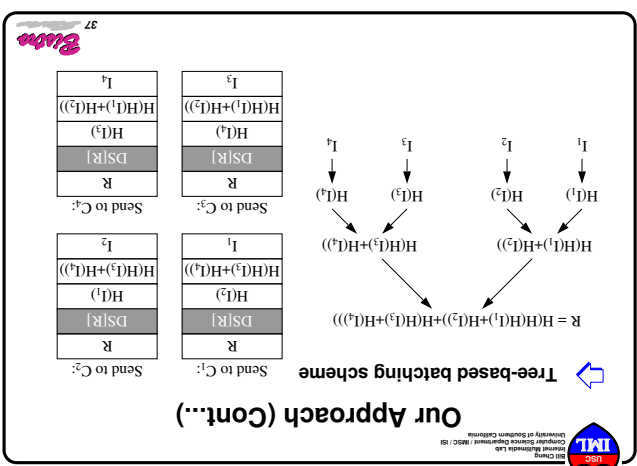
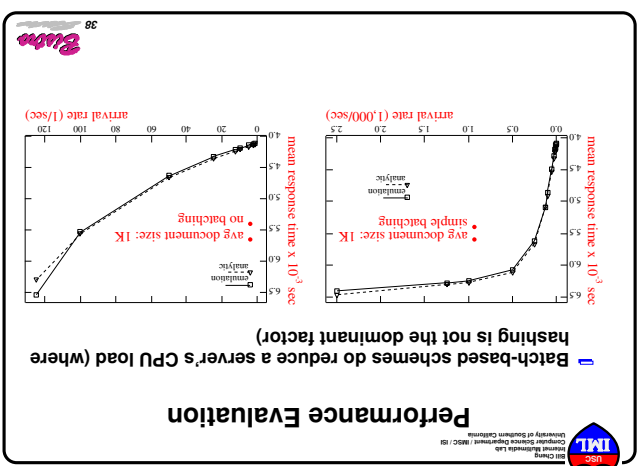
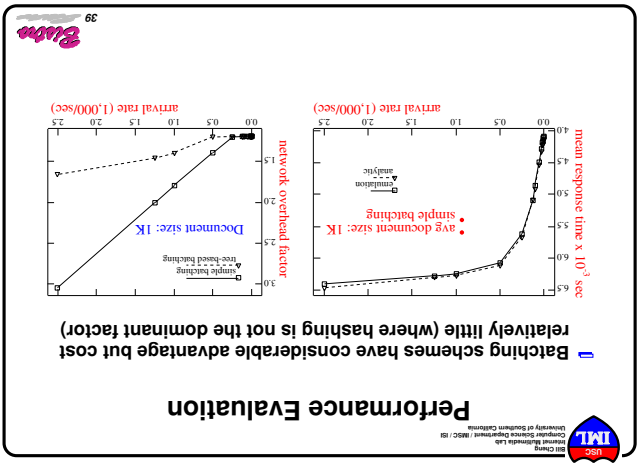
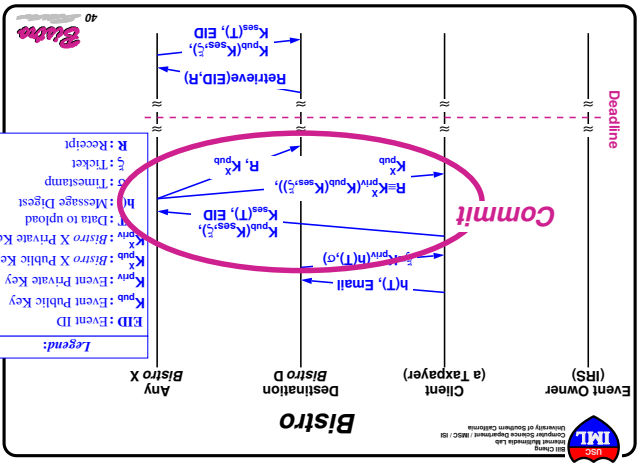
Why is this different from downloads?

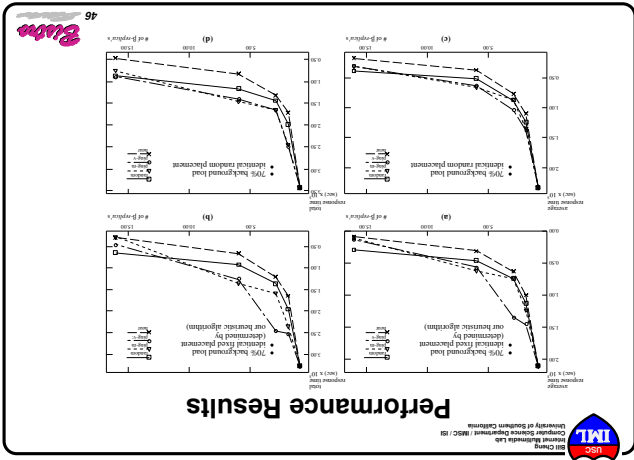
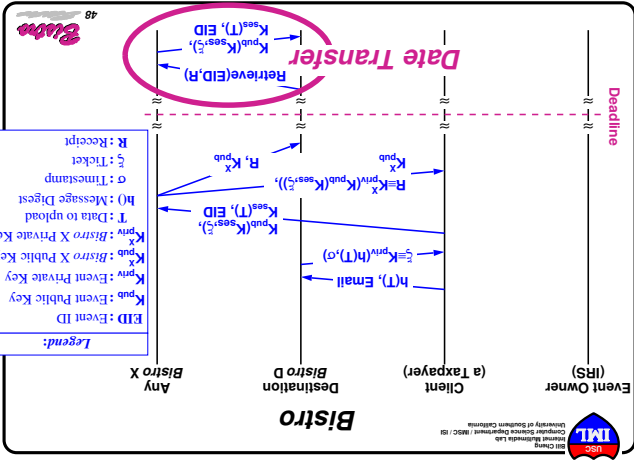
*Eidm*

**Commit Problem**  
Extreme Cases

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

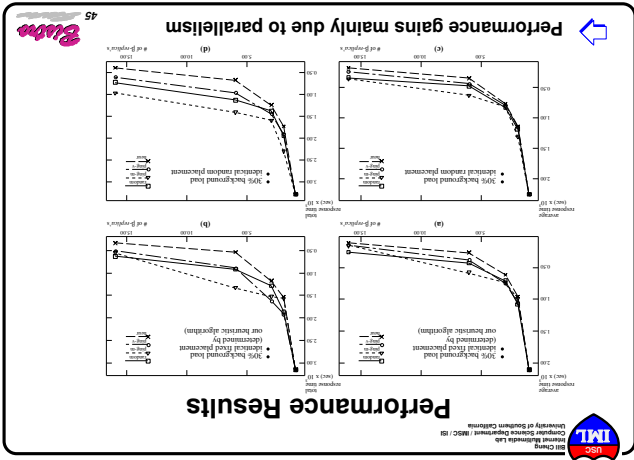
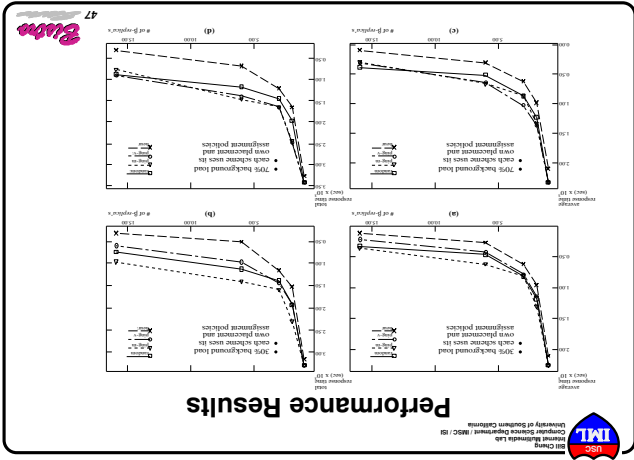
*Eidm*





**Performance Study**

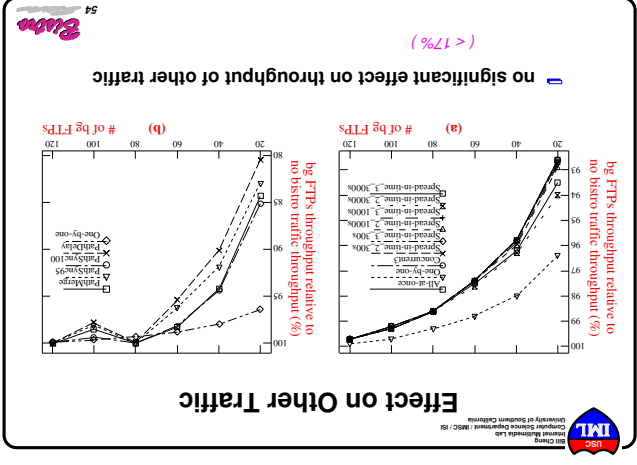
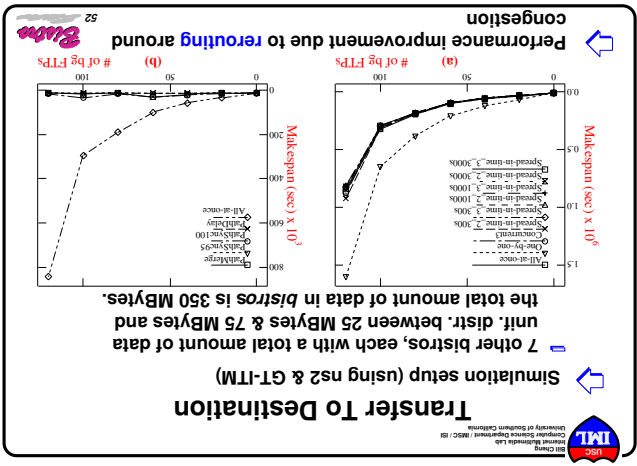
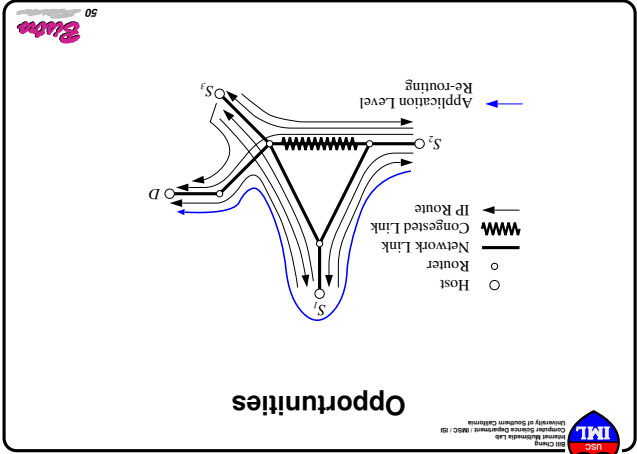
- Note: seq. uploads to single server should be approx 3000 sec, and avg. transfer time of one client should be approx 33 sec
- Note: simultaneous uploads to single server takes approx 3000 sec, but avg. transfer time of one client takes approx 2000 sec
- Performance metrics used
- mean transfer time over all clients
- total (or maximum) transfer time
- Polices
- random, ping-v, ping-m
- unrealistic heuristic (approx. lower bound)



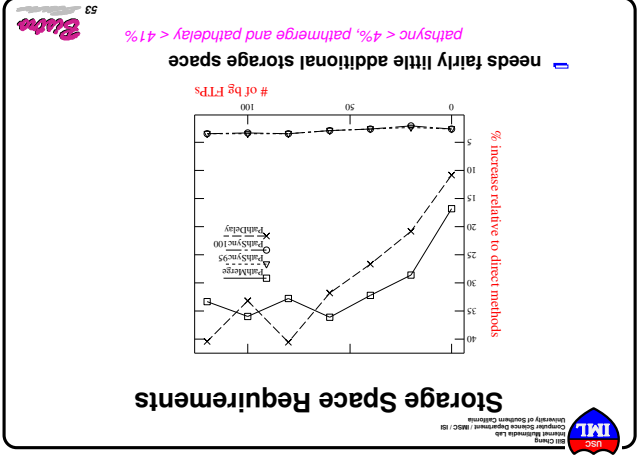
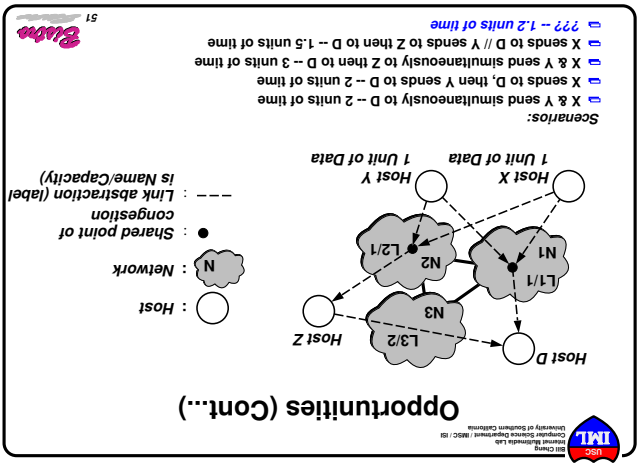
**Performance Study**

- Simulation setup (using ns2 & GT-ITM)
- transit-stub graph with 152 nodes
- 2 transit domains, with avg 4 nodes each, & each node having 3 stub domains connected
- stub domains have on avg 6 nodes each, edge capacity of transit-transit edge is 1 Mbit/s
- between pair of nodes with prob 0.2
- capacity of transit-stub or stub-stub edge is 256 Kbit/s
- 96 simultaneous uploads with files unif. distr. between 100 KBytes & 2 MBytes
- low background load (30%); high background load (70%)





- Large-scale Data Collection**
- Destination server needs to collect data from all other distros but how?
  - Several simple approaches
    - one-by-one
    - poor resource utilization due to non-shared bottleneck link
    - longer transfer time
    - all-at-once
    - spread-in-time-GT
    - concurrent-G
  - network congestion
  - application level re-routing
  - avoid congested links
  - devise a *coordinated* transfer schedule




**56**  
*Euro*

**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




UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455

**57**  
*Euro*

**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




UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455

**58**  
*Euro*

**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]




UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455

**57**  
*Euro*

**Related Work (Cont...)**

- Many-to-one communication at IP level & within Active network framework
- Gathercast [Badrath & 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., [Seshm et al. 97]
  - dynamic: e.g., [Carter & Crovelia 97] [Sayal et al. 98] [Dykes et al. 00]




UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455

**56**  
*Euro*

**Related Work**

- Akamai and other content distribution networks
- Napster
- A variety of server selection problems
- Internet security




UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455

**55**  
*Euro*

**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



UMI  
University of Minnesota  
Department of Computer Science  
1505 S. West  
Minneapolis, MN 55455