An Architecture for Internet Data Transfer

Niraj Tolia

Michael Kaminsky*, David G. Andersen, and Swapnil Patil

Carnegie Mellon University and *Intel Research Pittsburgh
Innovation in Data Transfer is Hard

Imagine: You have a novel data transfer technique

How do you deploy?
1. Update HTTP. Talk to IETF. Modify Apache, IIS, Firefox, Netscape, Opera, IE, Lynx, Wget, …
2. Update SMTP. Talk to IETF. Modify Sendmail, Postfix, Outlook…
3. Give up in frustration
Barriers to Innovation in Data Transfer

• Applications bundle:
  • **Content Negotiation**: What data to send
    – Naming (URLs, directories, …)
    – Languages
    – Identification
    – …
  • **Data Transfer**: Getting the bits across

• Both are tightly coupled (e.g., HTTP, SMTP)
• Hinders innovation and evolution of new services
Solution: A Data Transfer Service

- Decouple content negotiation from data transfer
- Applications perform negotiation as before
- But hand data objects to the Transfer Service
  - The Transfer Service is shared by applications
Extensible Transfer Architecture

**Plugins**

- Application-independent cache
- New network features
- Non-networked transfers
Transfer Service Benefits

- Apps. can reuse available transfer techniques
  - No reimplementation needed
- Easier deployment of new technologies
  - Applications need no modification
- Provides for cross-application sharing
  - Can interpose on all data transfers
- Handles transient disconnections
Outline

- Motivation
- Data Oriented Transfer (DOT) service
- Evaluation
- Open Issues and Future Work
- Conclusion
10,000 Foot View of Transfers using DOT

• How does the transfer service name data?
• How does the transfer service locate data?
DOT: Object Naming

- Application defined names are not portable
- Use content-naming for globally unique names
- Objects represented by an OID
  - Each OID corresponds to a list of descriptors
  - Descriptor lists allow for partial transfers

Objects are further sub-divided into “chunks”
DOT: Object Location

• Data transfers in DOT are receiver driven
  • Receiver has better idea of available resources

• Senders specify ‘hints’ - potential data locations
  – dot://sender.example.com:12000/
  – dht://opendht.org/
  – …
A Transfer using DOT

Sender

Request File X
OID, Hints

Receiver

put(X)

OID, Hints
get(OID, Hints)
read()
data

Xfer Service

Transfer Plugins

Xfer Service
DOT’s Modular Architecture

1. Application API

2. Transfer Plugin API

3. Storage Plugin API
Transfer Plugin API

• Simple API
  • get_descriptor_list( OID, hints )
  • get_chunks( descriptor_list, hints )
  • cancel_chunks( chunk_list )

• Transfer plugin chaining is easy
  • e.g., multipath plugin
Implementation

• In C++ using *libasync* event-driven library
• One storage plugin:
  – In-memory hash tables, disk backed.
• Three transfer plugins:
  • Default Xfer-Xfer plugin
  • Portable Storage plugin
  • Multipath plugin

• Applications
  • gcp, an scp-like tool for file transfers
  • A DOT-enabled Postfix email server
    – Included a socket-like adapter library
Current DOT Prototype

Plugins

✓ Application-independent cache
✓ Multipath and Mirror support
✓ Non-networked transfers
Outline

• Motivation
• Data Oriented Transfer (DOT) service
• Evaluation
• Open Issues and Future Work
• Conclusion
Evaluation

- Standard file transfer
- Portable Storage
- Multi-Path
- Case Study: Postfix Email Server
  - Capture and analysis of email trace
  - Evaluation of DOT-enabled SMTP server
  - Integration effort
Standard File Transfer Setup

Two DOT-enabled machines
Network Emulator
  • Evaluate various b/w + delay combinations
Use $gcp$ for the file transfers
Used 40MB, 4MB, 400KB, 40KB, 4KB files
  • Presenting 40MB here
Standard File Transfer

- Overhead: hashing, extra RTT
- No noticeable overheads with latency
Portable Storage Experiment

- 255 MB transfer over emulated DSL
  - Based on Virtual Machine transfers at Carnegie Mellon
  - DOT preemptively copies data onto Flash drive
- Wait 5 minutes, plug flash drive into receiver
- Two drive speeds
  - 8MB/s - 1GB
  - 20MB/s - 2GB
Portrayal of Storage Results

.. 1126s
(~ 19 min)

Portable Storage - 20 MB/s
Portable Storage - 8 MB/s
scp

Device Inserted
Multipath Plugin: Load Balancing

- Varied capacity + delay of experimental links
  - Compare fastest link alone with multipath plugin on both links; what speedup?
- Transferred 40MB file
  - 128 KB socket buffer sizes
Multipath Plugin is Effective

<table>
<thead>
<tr>
<th>Link 1</th>
<th>Link 2</th>
<th>Single</th>
<th>Multipath</th>
<th>Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/0</td>
<td>100/0</td>
<td>3.59</td>
<td>1.90</td>
<td>47%</td>
</tr>
<tr>
<td>10/0</td>
<td>10/0</td>
<td>3.54</td>
<td>1.4%</td>
<td></td>
</tr>
</tbody>
</table>

-40 MB @ 100Mbit/s ideal: 3.2 seconds

- Multipath plugin nearly doubles throughput

- TCP effects dominate. Pipe not full.

- Multipath plugin doubles by adding second stream. Actual capacity irrelevant.
Postfix Email Trace Replay

• Generated 10,000 email messages from trace
  • Random data matched to chunk hash data
  • Preserves some similarity between messages
  • Replayed through Postfix to a single local server

<table>
<thead>
<tr>
<th>Program</th>
<th>Seconds</th>
<th>Bytes Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postfix</td>
<td>468</td>
<td>172 MB</td>
</tr>
<tr>
<td>Postfix + DOT</td>
<td>468</td>
<td>117 MB (68%)</td>
</tr>
</tbody>
</table>

• Postfix disk bound… DOT CPU overhead negligible
• Savings due to duplication within emails
Postfix Integration

• Integrated DOT with the Postfix mail server

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC</th>
<th>Added LoC</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTC Lib</td>
<td>--</td>
<td>421</td>
<td>2.1%</td>
</tr>
<tr>
<td>Postfix</td>
<td>70,824</td>
<td>184</td>
<td>0.3%</td>
</tr>
<tr>
<td>smtpd</td>
<td>6,413</td>
<td>107</td>
<td>1.7%</td>
</tr>
<tr>
<td>smtp</td>
<td>3,378</td>
<td>71</td>
<td>2.1%</td>
</tr>
</tbody>
</table>

• 1 part-time week, 1 student new to Postfix
  • Includes time to write generic adapter library
Discussion on Deployment

• Application Resilience
  • DOT is a service - it’s outside the control of the application.
  • Our Postfix falls back to normal SMTP if
    – No Transfer Service contact
    – Transfer keeps failing
  • In the short term, a simple fallback is encouraged. However, this could interfere with some functions
    – DOT-based virus scanner…
  • In the long term, DOT would be a part of a system’s core infrastructure
Future Work

• Security
  • Application encrypts before DOT
    - No block-based caching, reuse, mirroring, …
  • No encryption
    - Resembles the status quo
• In progress: Convergent encryption
  – Requires integration with DOT chunking

• Application Preferences
  • Encryption, QoS, priorities, …
    – DOT might benefit from application input
  • Need an extensible way to express these
Conclusion

• DOT separates app. logic from data transfer
  • Makes it easier to extend both

• Architecture works well
  • Overhead low (especially in wide-area)
  • Major benefits
    • Caching
    • Flexibility to implement new transfer techniques

• Source code available on request

http://www.cs.cmu.edu/~dga/dot/