TCP Splice Benefits for Web Proxy Servers

Marcel C. Rosu
Daniela Rosu
IBM T.J. Watson
Server ‘in-the-middle’

- Web proxies, CDN nodes, Edge Servers...
- Act as caches for Web content
  - Hit rates are 50% or lower
- Relay data between end nodes
  - Process small fraction of data (headers)
  - Handle a very large number of connections

Our target
- Reduce overheads of data relay
Our Approach

- Use a General-Purpose Platform
  - Large servers vs. dedicated appliances
- Improve the data-forwarding path
  - Lower CPU overheads and packet latencies
- Restrict OS & app changes to a minimum
  - Improves chances of being deployed

- In-kernel connection splicing
TCP Connection Splicing

User Space
---
Kernel Space

Proxy App.
---
Socket Layer
---
TCP Layer
---
IP Layer
---
Network Interface
---
Client Link

---
Server Link

App.-level Splicing

Socket-level Splicing

IP-level Splicing
Related Work

- **IP-level Splice [Maltz et al., Spatcheck et al.]**
  - For firewalls, mobile gateways
  - Restricts splicing to connections with identical characteristics

- **Socket-level Splice [Balakrishnan et al.]**
  - Evaluated for throughput implications
  - Mobile gateways

- **Our work**
  - Use socket-level splice for Web Proxies
  - Evaluate for overhead reductions
Outline

- Implementation
- Experimental Testbed
- Experimental Evaluation
  - Forwarding overheads and latencies
    - GET requests and SSL Tunnels
  - Interaction with serving from proxy cache
- Conclusions and Future Work
Implementation

- New system call in AIX
  - Integrated with the TCP stack
- Data forwarding path
  - ≈100 lines C code
  - Executes in interrupt context
Experimental Testbed

- Platforms
  - AIX 5.10 on RS/6000s and Linux/Pentium
- Clients
  - s-client: generates concurrent request streams
  - best-effort workload
- Custom proxy
  - event-driven, minimal header processing
- HTTP server emulator
  - SSL handshake
- WAN emulation
  - enhanced Nistnet
Forwarding Overheads: GET

Proxy utiliz./req
- 25-50% reductions
- Proxy overloaded for 140 clients, app-level splicing

WAN conditions
- C-P: 10ms, loss 0.1%
- P-S: 90ms, loss 1.0%
Forwarding Latency

Latency
- Significant reductions
  - 5k+ files: 5-25%
- Small increases (< 5%)
  - small files, many clients
- Most important contribution:
  - Congestion window opens faster
SSL Tunneling

Proxy utiliz./req

- 25-50% reductions

SSL Handshake (full)
- Client: 98 bytes
- Server: 2239 bytes
- Client: 73 bytes
- Server: 6 bytes
- Client: 67 bytes
- Server: 61 bytes
Mixed Traffic: Cache & Server

- **Workload mix:**
  - 40 clients to cache
  - 40/80/120 to server

- **Performance**
  - Rates – similar for app- and socket-level splicing
  - Latencies – higher for socket-level splicing
Comparing to IP-level Splice

Faster loss recovery
- Independent loss recovery on the two TCP connections
- Lower RTTs and loss rates

WAN conditions
- C-P: 50ms, loss 0.1%, 56k modem
- P-S: 90ms, loss 1-2%
Conclusions

- Socket-level Splicing in proxy servers
  - Enables substantial overhead reductions for medium and large data transfers
  - Requires small/few kernel & app changes

- Future Work
  - Extend splicing interface
    - HTTP/1.1, handle cacheable content
  - Control resource allocation (memory, CPU)
    - kernel vs. application