Fault-Tolerant Linux

- Goal
  - Transparent server fail-over for existing client/server applications
- Wish List
  - Transparent: Server failures do not disturb Client connections.
    - How can we preserve application state?
    - How can we preserve connection state (TCP, encryption?)
  - Universal: Support existing applications.
    - How can we make a legacy app fault tolerant without rewriting it?
  - Fault Tolerant: Server failures never lose or corrupt data
  - Cheap: Minimal cost and overhead

Target Applications

- Linux
  - Web: Apache (58.4% of all sites)
  - CGI: PHP, mod_perl(2.8M sites), Python
  - Database: MySQL, PostgreSQL
  - Teleconferencing: Roger Wilco
  - Games: Quake
- Windows
  - IIS, SQL Server, NetMeeting

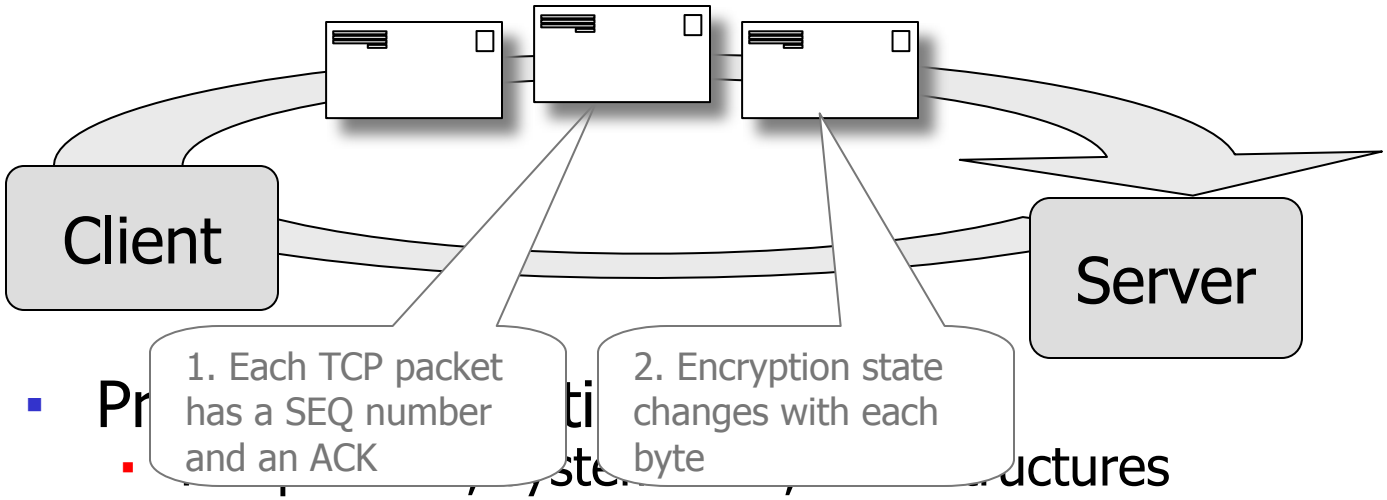
Quotes from http://www.NetCraft.com/survey, March 2002

Current Solutions

- Server Farms: F5 Networks BigIP, Cisco LocalDirector
  - Connections to failed server are broken
- Connection Migration
  - Client must be smart enough to detect failed server and go there
- Periodic Updates: periodic database dump/restore
  - Backup is always a little out-of-date
- Application Development Frameworks: E2EE, EJB
  - Incomplete support: fail-over support for "non-idempotent transactions" only.
- Application Fault Tolerance: Oracle,
  - Good, but single-product solution.
- Hardware: Compaq NonStop, Stratus ftServer
  - Was expensive, but low end has fallen to ~$10k, Win2k only.

Implications

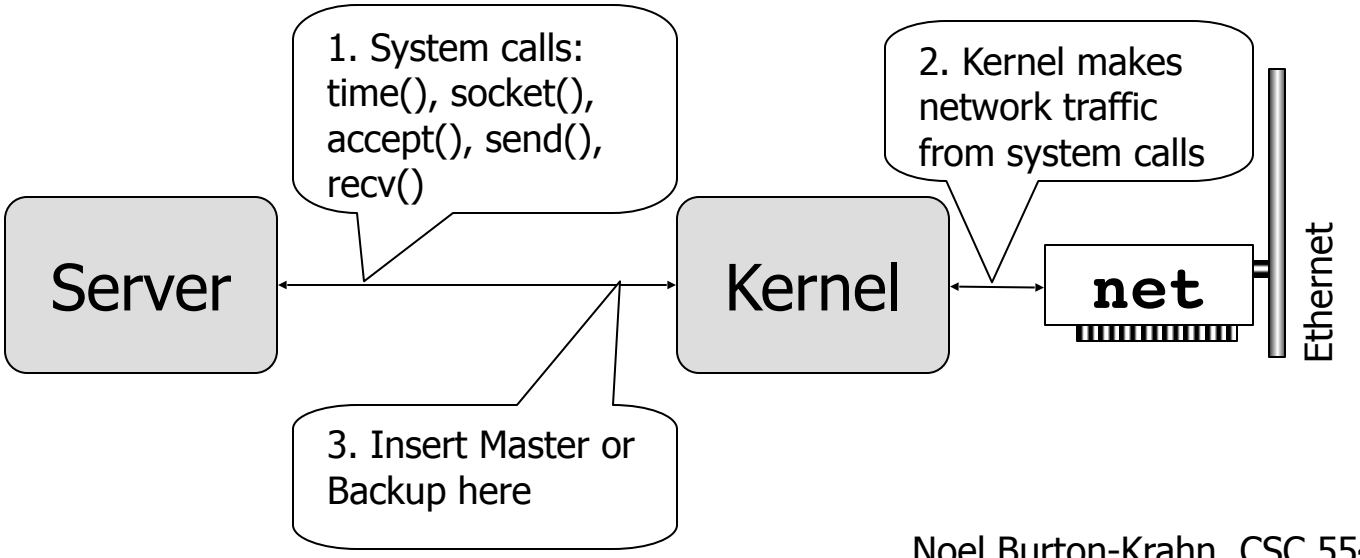
- Preserve Connection State
  - Must preserve TCP state: SYN, ACK, SEQ, WIN
  - Must preserve Encryption state: session keys, stream state



- Pr ti
  - uctures
- Support Existing Applications
  - Can't rewrite or relink

Proposal

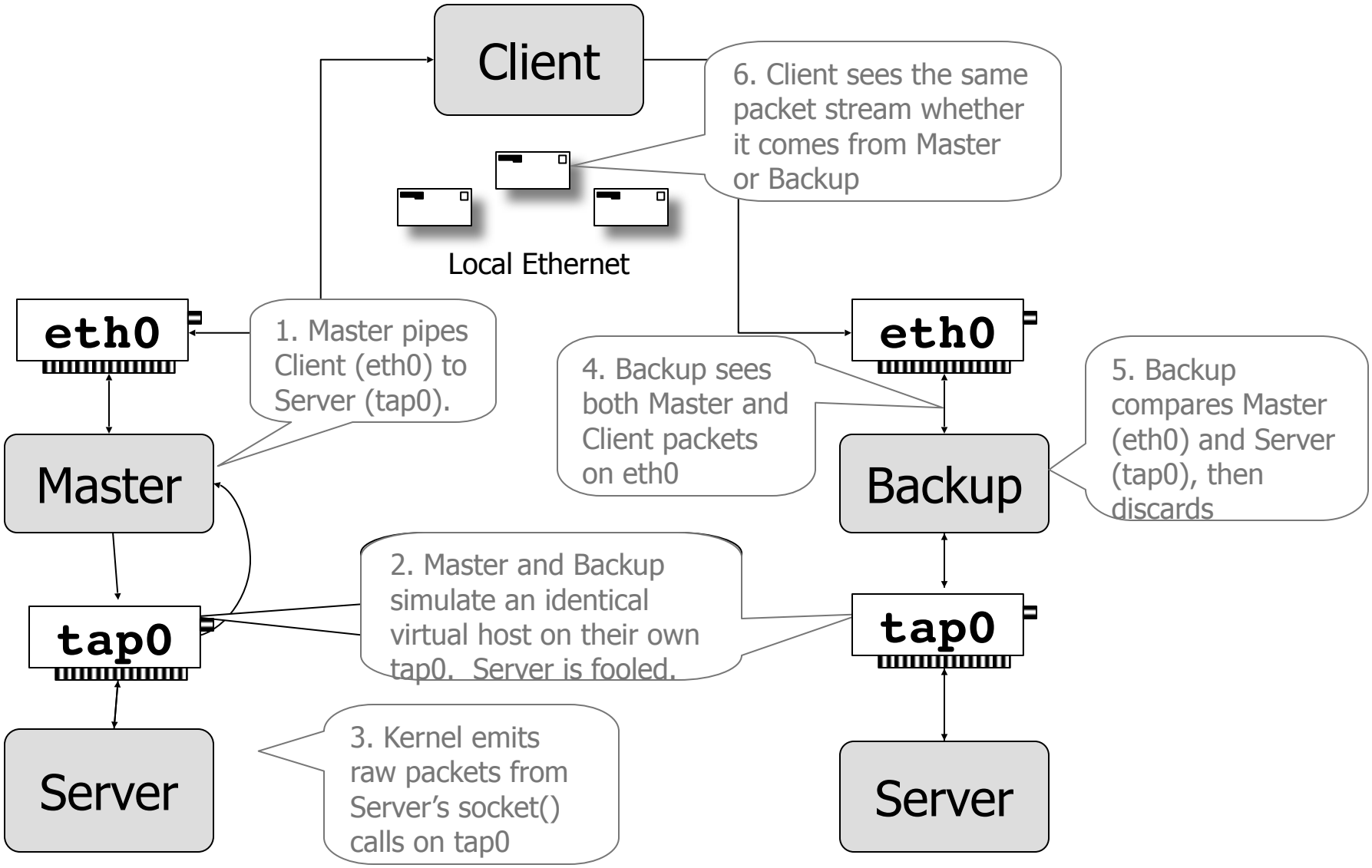
- Master and Backup hosts run duplicate Servers on isolated hosts.
- Assume programs are deterministic: if they get the same input, they will produce the same output.
- Backup carefully maintains identical input with the Master:
  - Copies all network traffic from Client
  - Same IP and MAC addresses as Master
  - Duplicate system calls to time(), /dev/urandom, etc



Cloning Packet Streams: TCP Review

- Recall how a TCP connection works:
  1. Client chooses SEQ, sends SYN to server
  2. Router on Server does ARP to find server's ether address
  3. Server Chooses SEQ, sends SYN,ACK
  4. Server does ARP to find Router's ether address
  5. Client and server exchange packets with ACK
  6. Encryption (optional): Client and Server choose random session keys. Decryption state changes with each byte

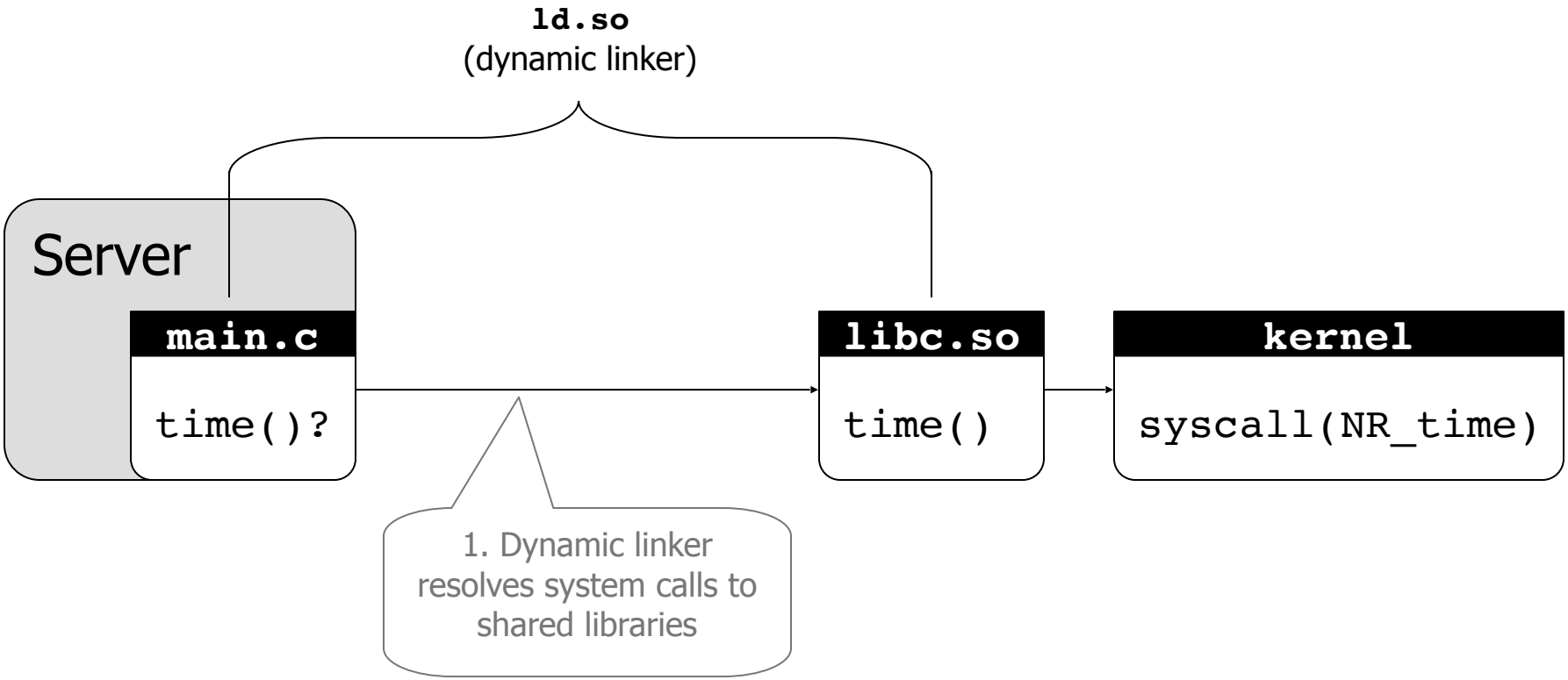
Cloning Packet Streams: tap Devices



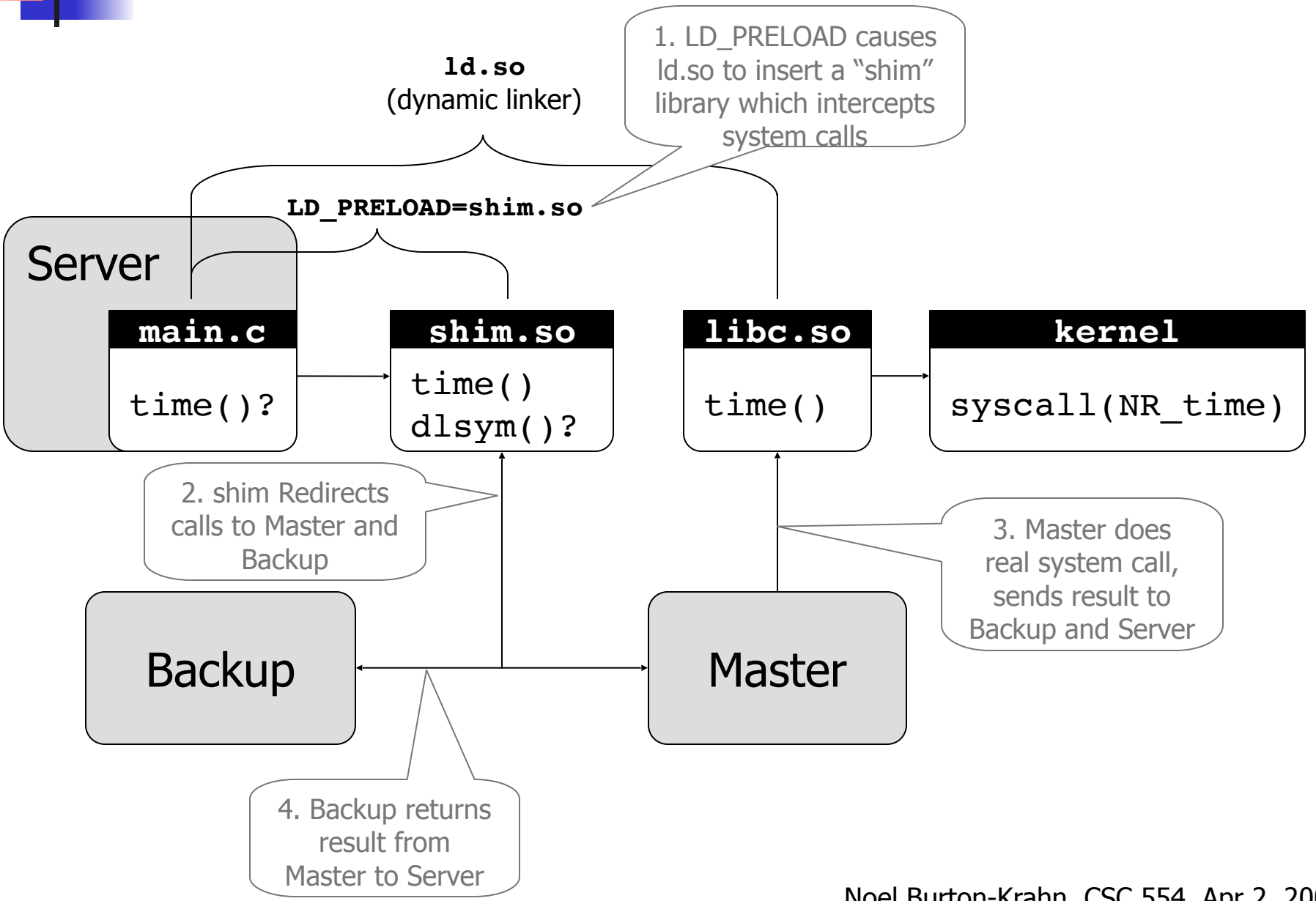
Cloning System Calls: Options

- Kernel
  - All Linux system calls go through a jump table. A kernel module can alter this, but it's ugly.
- External Debugger
  - Linux provides a ptrace() system call to write your own debugger, but context switching is slow
- Code Patching
  - You can write over a process' function pointer table or the function bodies themselves. Evil!
- LD_PRELOAD
  - The best solution: let the dynamic linker redirect function calls to your own library.

Cloning System Calls #1: Dynamic Linking



Cloning System Calls #2: LD_PRELOAD



Test Results

- A test system was build to simulate a Master and Backup Web server. Same host, different tap devices.
- The Backup compared every packet to the Master, and they were identical, except:
  - Backup and Master did ARP independently, but ok.
  - Backup and Master chose different TCP SEQ numbers, but ok because the difference was constant
  - Backup and Master chose different fragment IDs, but they're arbitrary anyway.
- LD_PRELOAD worked.
- Did not test high load or many fork()s
- Did not test flow control (Master and Backup run at different speeds)

Conclusions

- Benefits:
  - Transparent fail-over looks possible for existing servers.
  - Even servers with very dynamic in-memory state can be Backed up.
  - Low overhead: Backup and Master observe the same network traffic. A few extra packets to send system call results from Master to Backup
- Drawbacks:
  - Backup is totally dedicated. There's no load sharing with Backups.
  - Does not protect against software faults in Server.

References

TMR for Off-the-Shelf Unix Systems; *Eric Daniel and Gwan S. Choi, Texas A&M University, USA* http://www.crhc.uiuc.edu/FTCS-29/pdfs/daniele2.pdf

Fine-Grained Failover Using Connection Migration*; Alex C. Snoeren, David G. Andersen, and Hari Balakrishnan* *http://nms.lcs.mit.edu/papers/migrate-failover.html*

Client-Transparent Fault-Tolerant Web Service *http://citeseer.nj.nec.com/aghdaie01clienttransparent.html*

Transparent Process Migration: Design Alternatives and the Sprite Implementation*; Fred Douglis and John K. Ousterhout,; Software - Practice and Experience; vol 21, no 8, pp 757-785; 1991;* *citeseer.nj.nec.com/douglis91transparent.html*

Understanding Replication in Databases and Distributed Systems*; M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso:. In: 20th International Conference on Distributed Computing Systems (ICDCS), Taipei, Taiwan, Republic of China, April 2000.* *http://www.inf.ethz.ch/department/IS/iks/publications/wpska00.html*

Patch-free User-level Link-time intercepting of system calls and interposing on library functions *http://okmij.org/ftp/syscall-interpose.html*