Network Programming TDC 561

Lecture # 8: Techniques for Inter-network Programming

Dr. Ehab S. Al-Shaer

School of Computer Science & Telecommunication
DePaul University
Chicago, IL

1

Programming Techniques for Reliable Network Systems

- Making a background processes
  - fork and parent exits
  - `init` inherits the child
- Closing all inherited file descriptors
- Deattach a (daemon) sever form TTY
  - to avoid signals from the terminal (SIGHUP):

```
fd=open("/dev/tty",O_RDWR);
ioctl(fd, TIOCNOTTY,0);
close(fd);
```

- Run server only from standard directory

2

Programming Techniques for Reliable Network Systems

- Use umask(027) in servers
  - to avoid creating accessible files by others
- Independent groups for a server
  - to avoid signals sent to the parent's group
  - setpgrp(0, getpid());
- Avoiding running multiple copies of a server
  - bind()
  - lock files (P.442)

3

Programming Techniques for Reliable Network Systems

- Avoiding zombie state (defunct process)
  - if server exits, no problem (init takes over)
  - if a child exits, parent must wait.
- Ignoring Signals
  - `(void) signal(SIG_IGN, SIGHUP)`
- Get the msgsize before reading the message
- Use multi-threaded for I/O and CPU mixed programs
- Always read incoming messages even if they are not important!

Deadlocks in Network Programming

- What is a deadlock?
  - Circular dependencies
  - Deadlock in Client/Server systems
  - UDP without select()
- Deadlock detection is very hard!
- Deadlock Avoidance
  - In a single client/server interaction:
    - the protocol is ambiguous and synchronization is defined
    - using timers for unreliable communications

Avoid Client Starvation

- In iterative servers
  - limit number of requests
  - client connect but never sends!
    - Solution: <you must know it>
  - Client sends requests but does not read replies
- Denial Service Attack of concurrent servers

```
for (;;)
    connectTCP(machine, service);
```

  - Solution: check IP+ port from messages

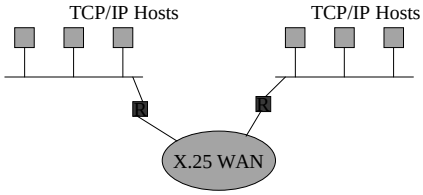
General Concepts in Internetworks Programming

- Heterogeneous networks
  - Multiple vendors protocols
    - XTP, SNA, DECNet, XNS, X.25
  - Old organizations' networks
  - Lack of WAN support (X.25 used to be in WAN more)
- IP over X.25 (well, not common any more)
  - dedicated links are expensive
  - Tunneling is required
  - N*(N-1)/2 VC is needed for N sites!
    - Dynamic circuit allocation
  - Now IP is used to link proprietary protocols

General Concepts in Internetworks Programming

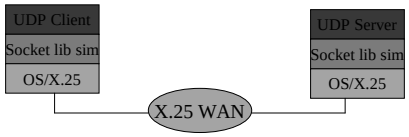
- Protocol Encapsulation
  - IP inside Ethernet frame
  - generally, envelop messages
  - Encapsulation and Decapsulation (in layers)
- Protocol Tunneling
  - IP inside other network/transport services
  - Needs Kernel/OS system support
  - E.g., IP over X.25, SNA over IP, …

IP over X.25



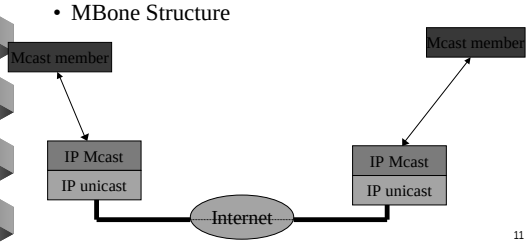
Application-level Tunneling

- Provides a communication path between client and server
- Motivation
  - no OS or hardware support
  - flexibility
- Again, IP over X.25



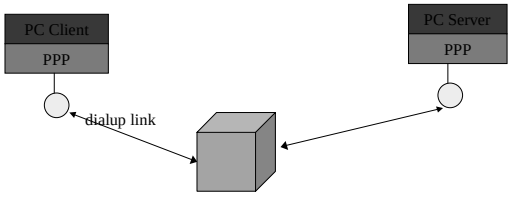
Application-level Tunneling

- IP Multicast Tunneling
  - No multicast-enables routers/hosts
  - MBone Structure



Application-level Tunneling

- SLIP/PPP for Dial up Service
  - Is it “encapsulation” or “tunneling”?



Application Level Gateways

- Intermediate program to relay information between the client and the server
- Motivation
  - The same as tunneling
- Examples
  - Mail gateway: UUCP and Internet Mail
- Advantages
  - Application level (no change is needed in OS)
  - Transparent
- Disadvantages
  - Separate gateway for each additional service
  - Requires extra HW and SW resources (CPU or memory intensive)

Application-level Gateways

- SLIRP PPP Gateway
  - avoids a "real" valid IP address (use temp addr)
  - impersonates remote TCP/IP hosts