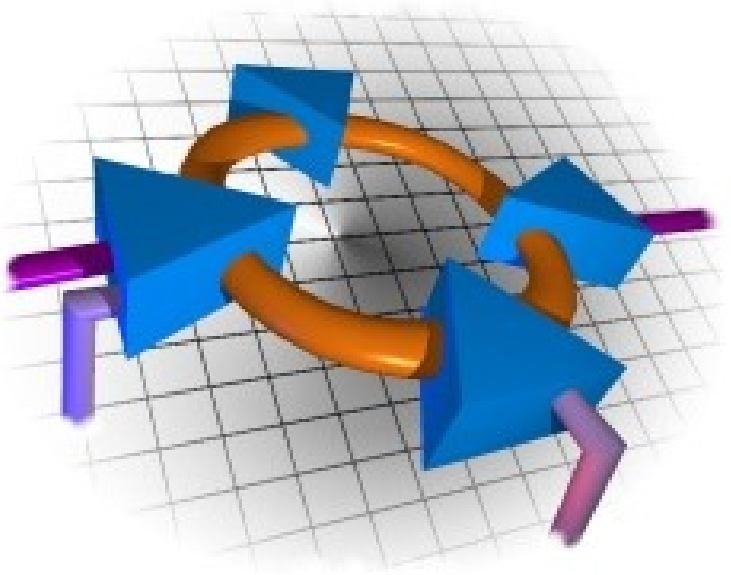
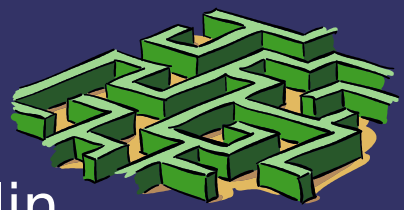
# muXTCP – Writing your own TCP/IP Stack – Ninja Style!!!



Paul Böhm

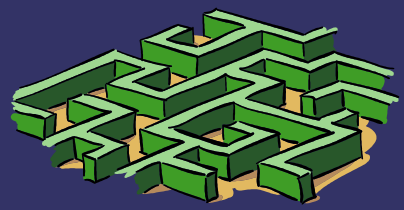
paul@boehm.org



Presentation at the 22C3, 27th-30th December 2005, Berlin

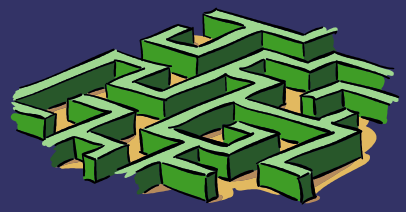
# *Who is talking?*

- Paul Böhm
- Professionally trained Ninja and Packet Juggler
- Member of TESO Security
- Quantum Cryptography Protocol hacking at Univie



# *You'll be hearing about*

- TCP/IP Hacking Tools
- Stateful Protocol Frameworks
- Framework Design
- muXTCP: Python Framework for writing stateful network hacking tools.



# *Stateless Network Hacking Tools*

- Applications:
  - Port Scanning
  - Active OS Fingerprinting
  - Passive OS Fingerprinting
  - ARP Spoofing
  - ...
- Mode of Operation:
  - Connectionless
  - Stimuli-Response Operation: Send a packet, receive a packet.

# *Limitations of stateless network Hacking Tools*

- Can only scratch the surface of TCP
- Misses vital Information:
  - E.g.: p0f can't reliably fingerprint any packets other than SYN
- Can't combine Network with Application Level Attacks

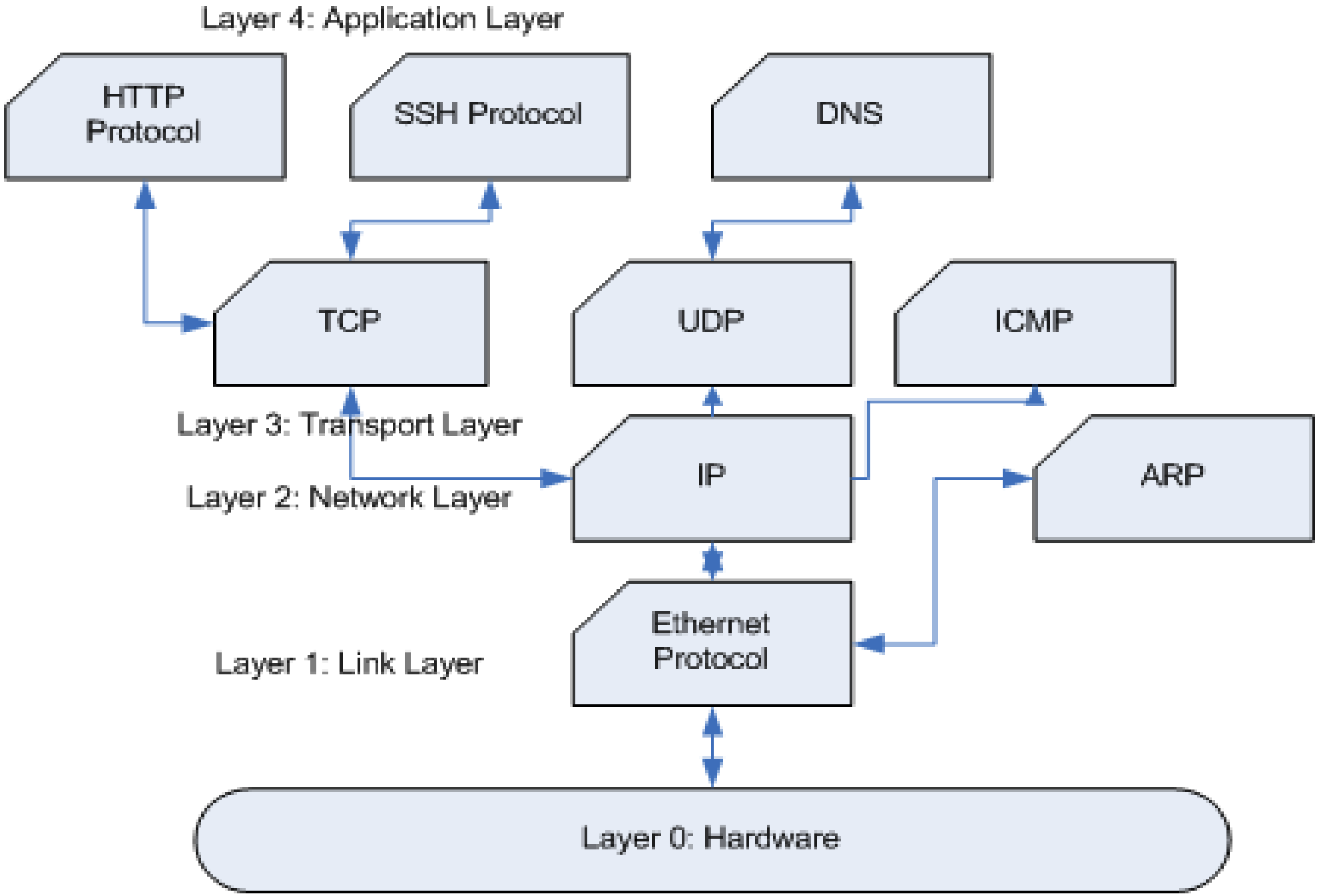
# *Motivation*

- There are awesome stateless TCP/IP Hacking Frameworks (Scapy). -> Drivers for Innovation
- No general purpose stateful network hacking Tools exist.
- Writing a new Stack for every tool is expensive. This stalls innovation in stateful hacking.
- Lessions learned here might also be applicable to other areas (stateful fuzzers, P2P Protocols).

# *How is it done?*

- Choose an IP Addr and/or Port for muXTCP to listen on.
- If IP-Addr is used by the Host-OS: Block the Port using the local Firewall
- If IP-Addr unused: Answer ARP Requests for the Addr
- Promiscuous sniffing on the Ethernet
- Answer subsequent Packets destined for the muXTCP IP-Addr or Port.

# What Protocols are handled?



# *What is possible? 1/2*

- New Tools: e.g. Traceroute over existing TCP Connections
- Monkey-in-the-Middle Attack Suites
- Intrusion Detection Systems
- P2P Stacks
- Honeynets

# *What is possible? 2/2*

- IDS Evasion (IP Fragmentation)
- Stateful Scanning (heavy-duty scanner with more than 65535 simultaneous connections)
- Fooling OS Detection (passive and active)
  - We can emulate the behavior of any OS, both for nmap (active), and p0f (passive)
- Improving OS Detection
  - p0f (passive) only handles SYN (stateless) packets reliably.
  - Active Fingerprinting (nmap) could also go much deeper.

# *How to get, send and process the Data: Libraries and Programming Paradigms*

- Asynchronous Programming
- Twisted
- Scapy
- Layered Design
- OOP Design Patterns

# *Asynchronous Programming 1/2*

- Non-Blocking Sockets
  - read(fd) and other socket calls don't block
  - select() notifies if new data has arrived on a socket.

```
def mainloop(self):
    while True:
        rlist, wlist, xlist = select(self.connections, [], [])
        for sock in rlist:
            data = sock.recv(2048)
            print data,
```

# *Asynchronous Programming 2/2*

- Callbacks abstract Control-Flow

```
# This is just about the simplest possible protocol
class Echo(Protocol):
    def connectionMade(self):
        print "Connection Established."
    def dataReceived(self, data):
        """As soon as any data is received, write it back."""
        self.transport.write(data)
```

```
from twisted.protocols import basic

from twisted.internet import protocol, reactor
from twisted.application import internet

class MyChat(basic.LineReceiver):
    def connectionMade(self):
        print "Got new client!"
        self.factory.clients.append(self)

    def connectionLost(self, err):
        print "Lost a client:", err
        self.factory.clients.remove(self)

    def lineReceived(self, line):
        print "received", repr(line)
        for c in self.factory.clients:
            c.message(line)

    def message(self, message):
        self.transport.write(message + '\n')

factory = protocol.ServerFactory()
factory.protocol = MyChat
factory.clients = []

reactor.listenTCP(1025, factory)
reactor.run()
```

# *Twisted*

- Generic Mainloop (Reactor Pattern)
- Basic Protocol Infrastructure
- Lots of implemented Protocols
- Integrates well with other Mainloops (GUIs etc.)

# *Scapy*

- Creating and sending Packets
- Sniffing Packets
- Traceroute Demo
- p0f Demo
- ...

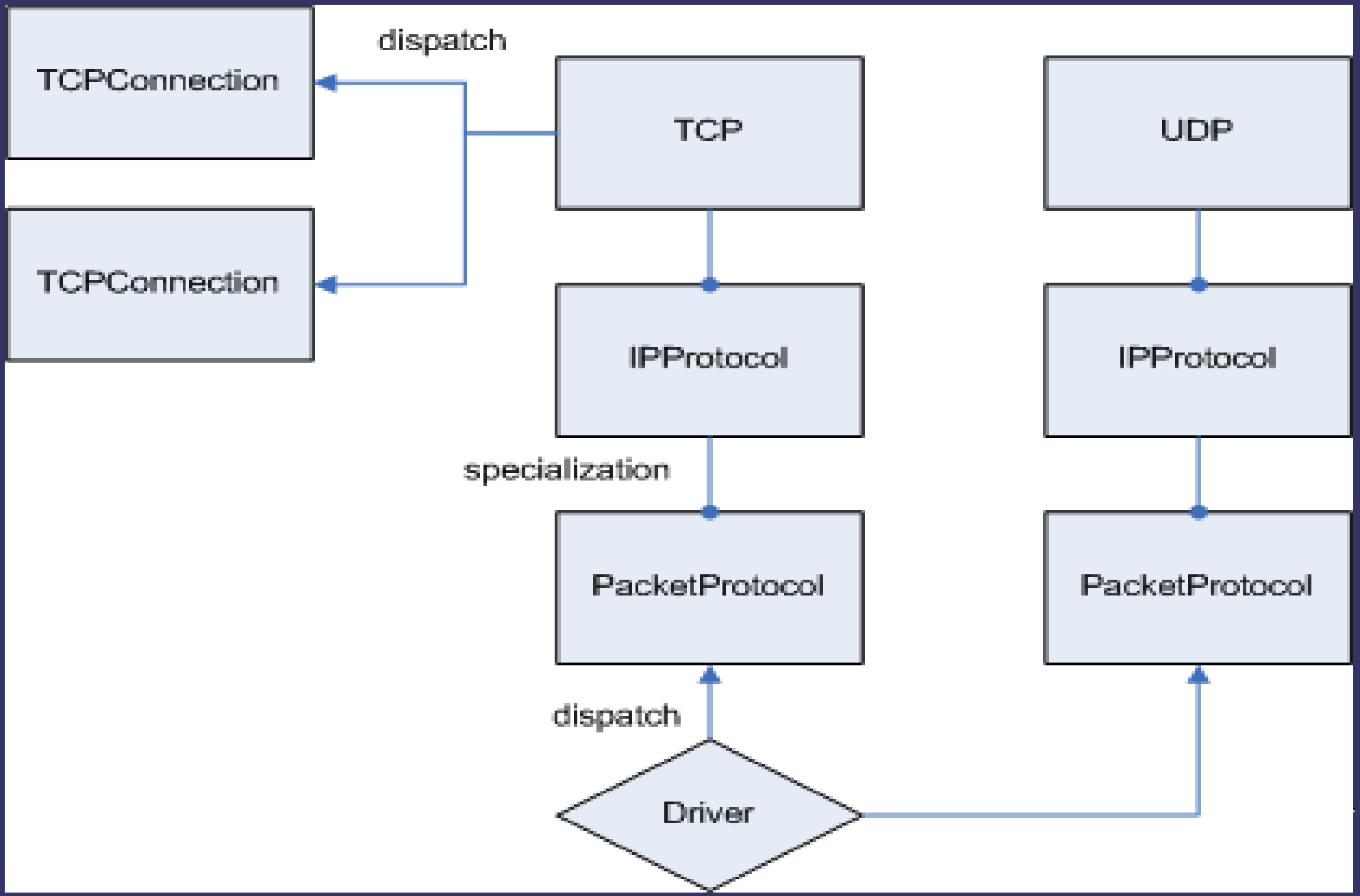
# *Implementation Strategies*

- Inheritance

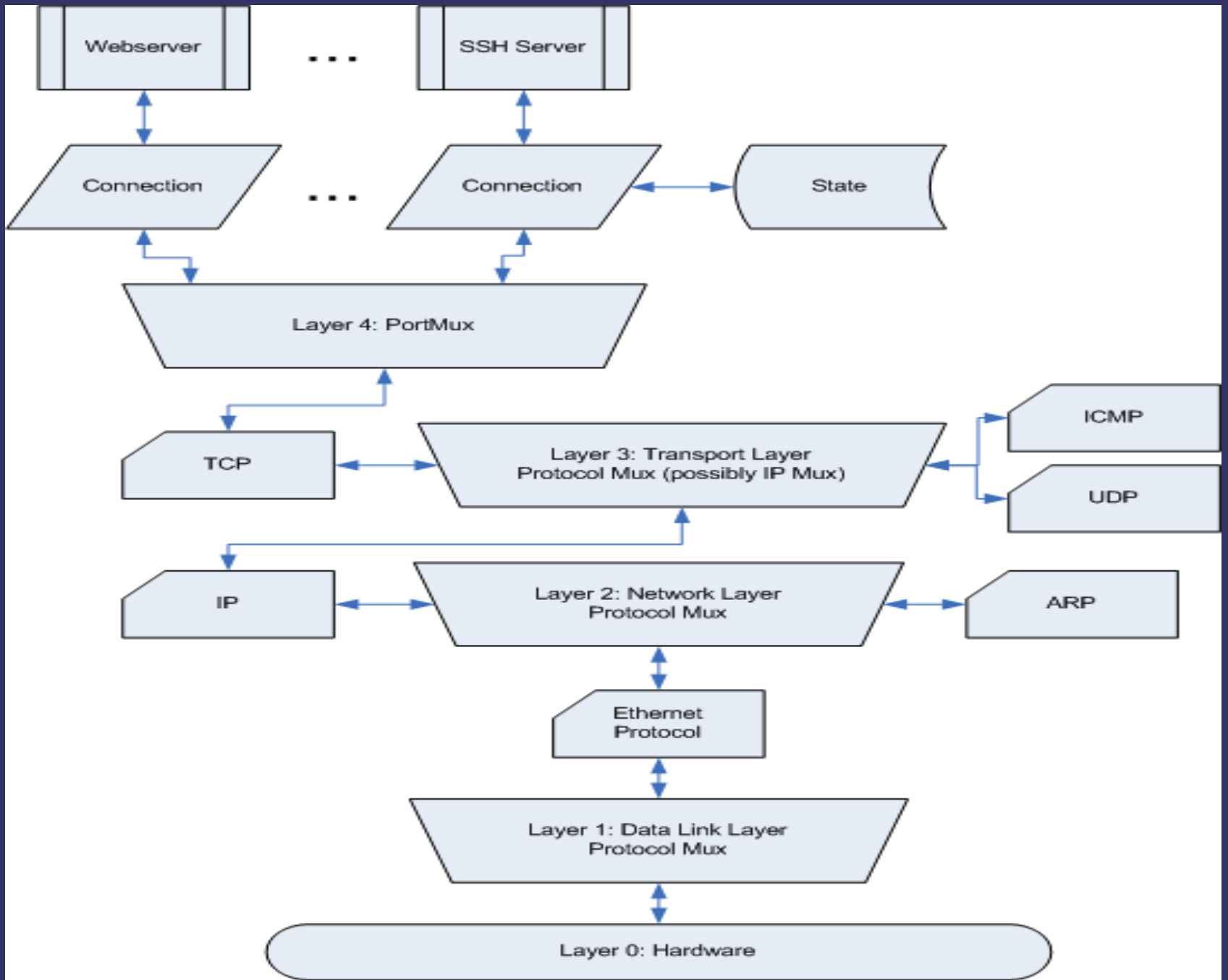
- Composition

- Hybrid

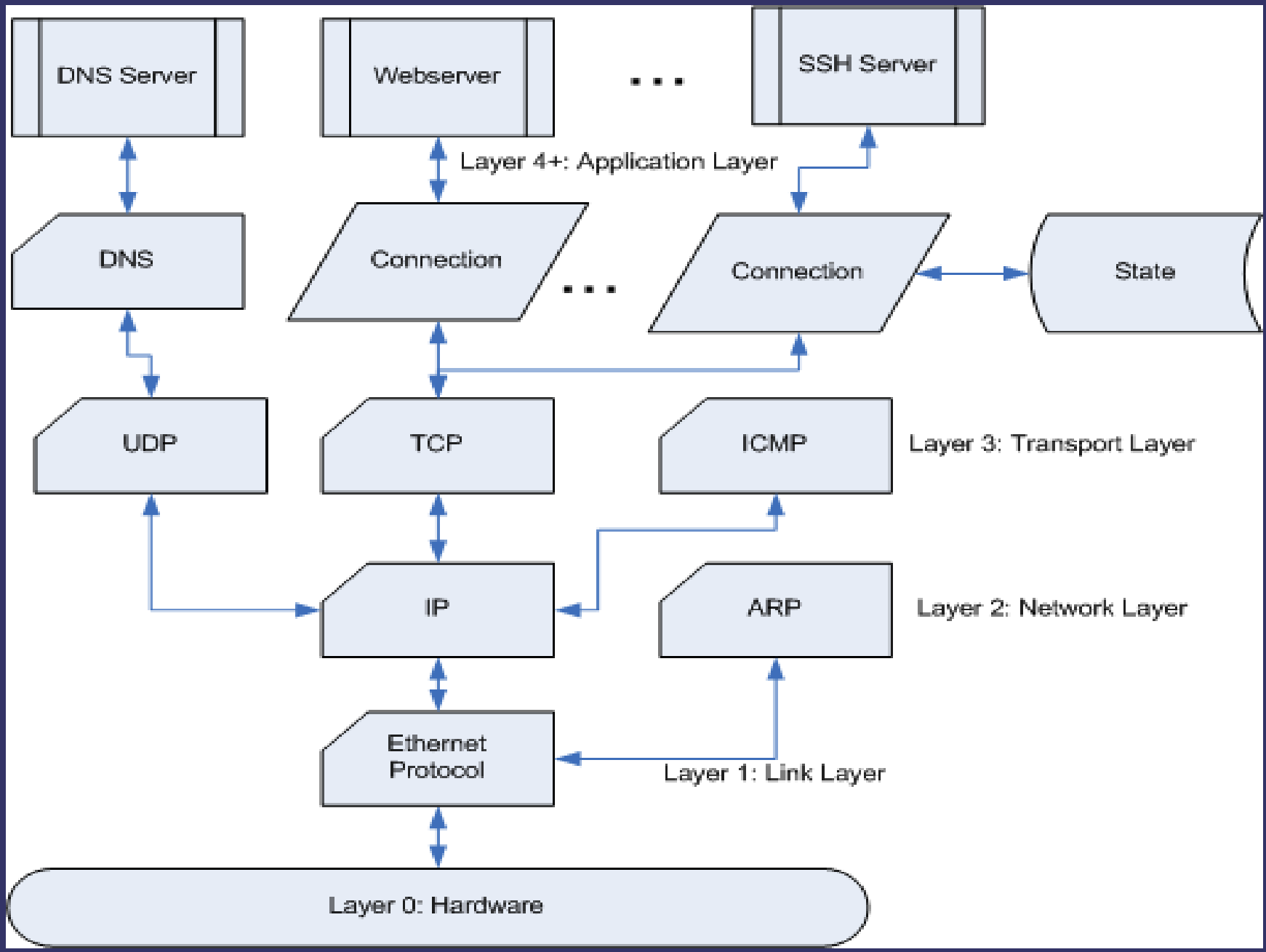
# Inheritance



# Composition



# Composition Model Simplified



# *Hybrid*

- Accepting that things in the real world are chaotic.
- Singleton Pattern: Make specific Protocols globally accessible by name.
- Protocols communicate in any way they want, and pass around meta-information and requests to build Packets.
- Packets are processed by the Protocol Chain.
  - Strategy Pattern (externalized States, discussed later) allows for enough flexibility
- Packets are sent by the layer generating them.

# *Strategy Pattern*

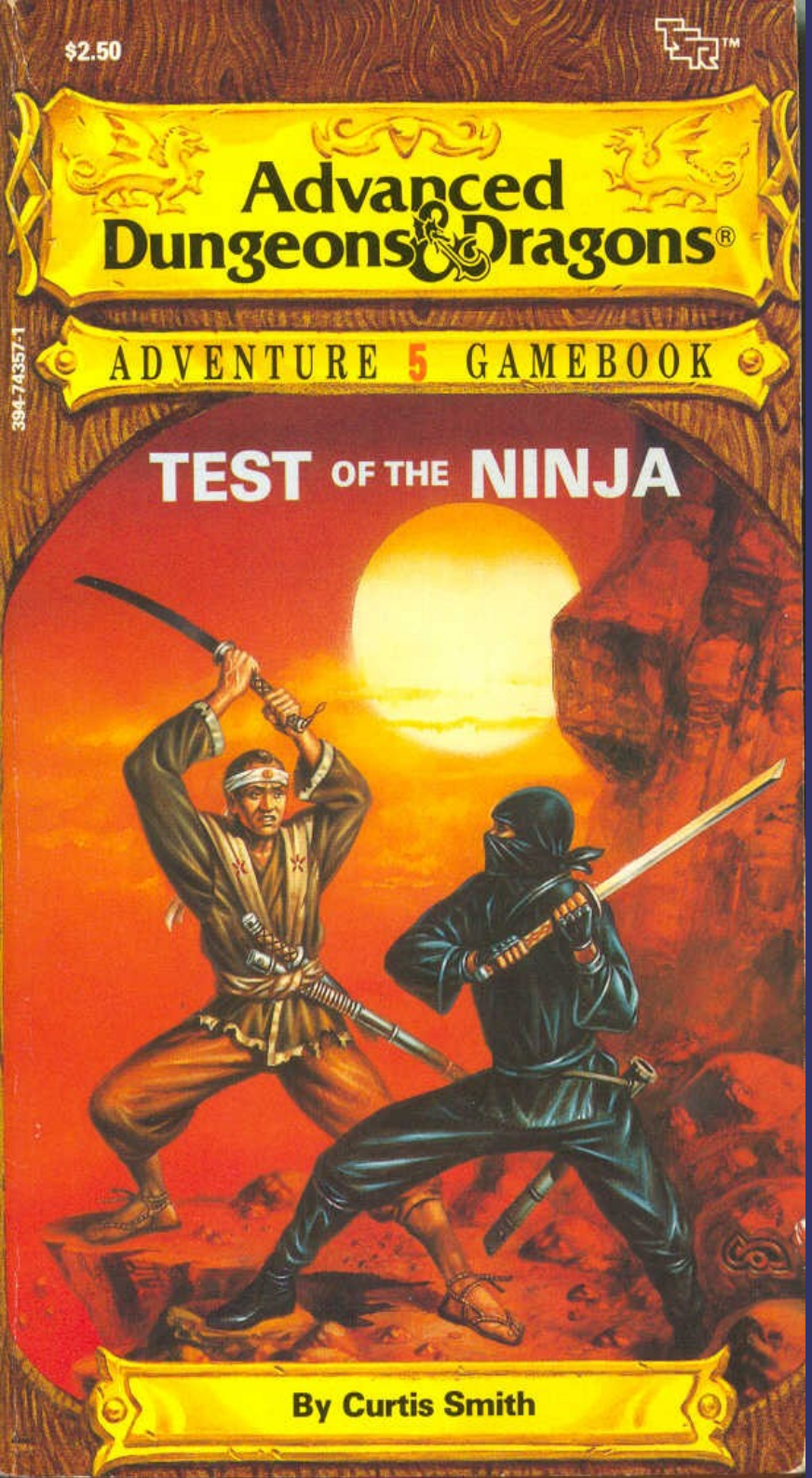
- Decouples Algorithm from Protocol. Makes it easy to change behavior at runtime.
- All TCP States are Strategy-Objects that implement Packethandling behavior.
- States don't have local variables, but use the Protocol for variable Storage.

# *What doesn't work yet?*

- IP Fragmentation
- Most advanced IP and TCP Options
- ICMP
- Simulaneous Open (Two-Way-SYN) and other Border-Cases
- DNS
- Full ARP

# *What works?*

- Routing Packets through the whole Protocol Chain
- Basic ARP (95.0% done)
- UDP (but no ICMP Errors handled yet)
- Basic TCP State Transitions for Listeners and active connect.
- Rough Twisted Protocol Integration




# Demo

- Webserver running on muxTCP
- SSH Server running on muxTCP
- All running in the same Process and same thread together with the Stack
- Runs on full muXTCP Stack including Ethernet, ARP, IP, and TCP

# *What will work soon? (think weeks)*

- ICMP
- ICMP Error Propagation to other Protocols
- More IP Support
- More TCP Support
- Traceroute over existing Connections
- OS Spoofing

# *Future Goals*

- C Shared Library Interface
- Registering new Protocols at Runtime (Server Mode)
- GUI Protocol Builder
- Asynchronous DNS
- TCP-over-UDP for P2P Apps
- Full IP Support
- Lots of Applications

# *The End*

*Questions?*

Applications Places System 600 MHz 100% Sat Dec 24, 7:49 AM

Shirt Ninja - Mozilla Firefox

File Edit View Go Bookmarks Tools Help del.icio.us

http://www.entertheninja.com/ninja_fun/shirt_ninja.shtml

Go

pgr rr kyblog li KybWeb blglns wetter Studium gr flckr nyt Letzte Änderungen -...

Oral Presentation Advice | typoninja.png (PNG Image, 1024... | Shirt Ninja


Find: minut Find Next Find Previous Highlight Match case

Done PR:5 Adblock

Firefo... enki... Inbox... XMMS muxt... muxt... eth0:... Ether... Ether... cond...