

The Network Stack (1)

Lecture 5, Part 1: Network Stacks

Prof. Robert N. M. Watson

2023-2024

Introduction to Network Stacks

Rapid tour across hardware and software:

- Networking and the sockets API
- Network-stack design principles
- Memory flow across hardware + software
- Network-stack work flows
- Recent network-stack research



Lecture 5, Part 1



Lecture 5, Part 2



Lecture 5, Part 3

Networking: A key OS function (1)

- Communication between computer systems
 - **Local-Area Networks (LANs)**
 - **Wide-Area Networks (WANs)**
 - **Inter-VM communication on a single host**
- A network stack provides:
 - Sockets API and extensions
 - Interoperable, feature-rich, high-performance protocol implementations (e.g., IPv4, IPv6, ICMP, UDP, TCP, SCTP, 802.1, 802.11, ...)
 - Security functions (e.g., cryptographic tunneling, firewalls...)
 - Device drivers for Network Interface Cards (NICs)
 - Monitoring and management interfaces (BPF, `ioctl`)
 - Plethora of support libraries (e.g., DNS)

Networking: A key OS function (2)

- Dramatic changes over 30 years:
 - 1980s: Early packet-switched networks, UDP+TCP/IP, Ethernet
 - 1990s: Large-scale migration to IP; Ethernet VLANs
 - 2000s: 1-Gigabit, then 10-Gigabit Ethernet; 802.11; GSM data
 - 2010s: Deployment of IPv6; 40/100-Gbps Ethernet; 3G to 5G;
... billions → trillions of devices?
- Vanishing technologies
 - UUCP, IPX/SPX, ATM, token ring, SLIP, ...

The Berkeley Sockets API (1983)

```
close()  
read()  
write()  
...  
  
accept()  
bind()  
connect()  
getsockopt()  
listen()  
recv()  
select()  
send()  
setsockopt()  
socket()  
...
```

- **The Design and Implementation of the 4.3BSD Operating System**
 - (but APIs/code first appeared in 4.2BSD)
- Now universal TCP/IP (POSIX, Windows)
- Kernel-resident network stack serves networking applications via system calls
- Reuses file-descriptor abstraction
 - Same API for local and distributed IPC
 - Simple, synchronous, copying semantics
 - Blocking/non-blocking I/O, `select()`
- Multi-protocol (e.g., IPv4, IPv6, ISO, ...)
 - TCP-focused but not TCP-specific
 - Cross-protocol abstractions and libraries
 - Protocol-specific implementations
 - “Portable” applications

BSD network-stack principles (1980s-1990s)

Multi-protocol, packet-oriented network research framework:

- **Object-oriented:** multiple protocols, socket types, but one API
 - **Protocol-independent:** streams vs. datagrams, sockets, socket buffers, socket addresses, network interfaces, routing table, packets
 - **Protocol-specific:** connection lists, address/routing specialization, routing, transport protocol itself – encapsulation, decapsulation, etc.
- **Packet-oriented:**
 - Packets and packet queueing as fundamental primitives
 - Best effort: If there is a failure (overload, corruption), drop the packet
 - Work hard to maintain packet source ordering
 - Differentiate ‘receive’ from ‘deliver’ and ‘send’ from ‘transmit’
 - Heavy focus on TCP functionality and performance
 - Middle-node (forwarding), not just edge-node (I/O), functionality
 - High-performance packet capture: Berkeley Packet Filter (BPF)

FreeBSD network-stack principles (1990s-2010s)

All of the 1980s features and also ...

- **Hardware:**

- Multi-processor scalability
- NIC offload features (checksums, TSO/LRO, full TCP)
- Multi-queue network cards with load balancing/flow direction
- Performance to 10s or 100s of Gigabit/s
- Wireless networking

- **Protocols:**

- Dual IPv4/IPv6
- Pluggable congestion control, delay-based congestion control (BBR)
- Security/privacy: firewalls, IPsec, ...

- **Software model:**

- Flexible memory model integrates with VM for zero-copy
- Fine-grained locking and lockless algorithms (e.g., RCU)
- Network-stack virtualisation
- Userspace networking via netmap