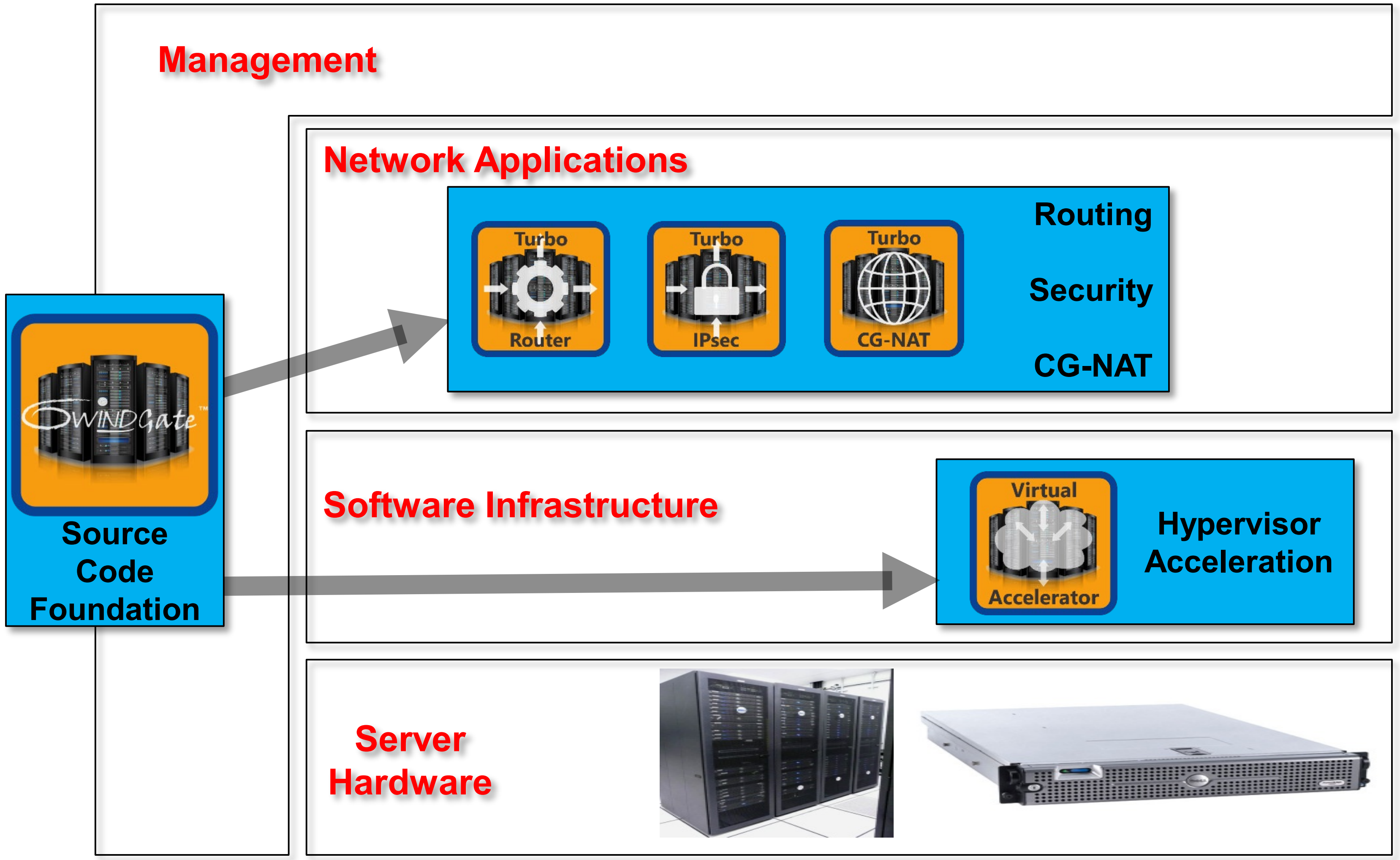
6WINDGate TCP

#SPEEDMATTERS For Serious Networks

6WIND Mission – Democratize Networking

Provide the **future of networking** with **software on white box servers**, giving customers **independence from expensive networking hardware at a fraction of the cost,** while delivering the **high performance** necessary for software to replace hardware

6WIND Software for High Performance Networking



- **Build-Your-Own vRouter**
  - 6WINDGate
  - Complete L2-4 networking stack
  - A la carte modules
- **vRouter for Network Appliances**
  - IP Routing
  - IPsec VPNs
  - CG-NAT
- **vRouter for Hypervisor**
  - Virtual Accelerator

6WINDGate Solution For High Performance TCP-Based Applications

#SPEEDMATTERS For Serious Networks

The TCP Problem

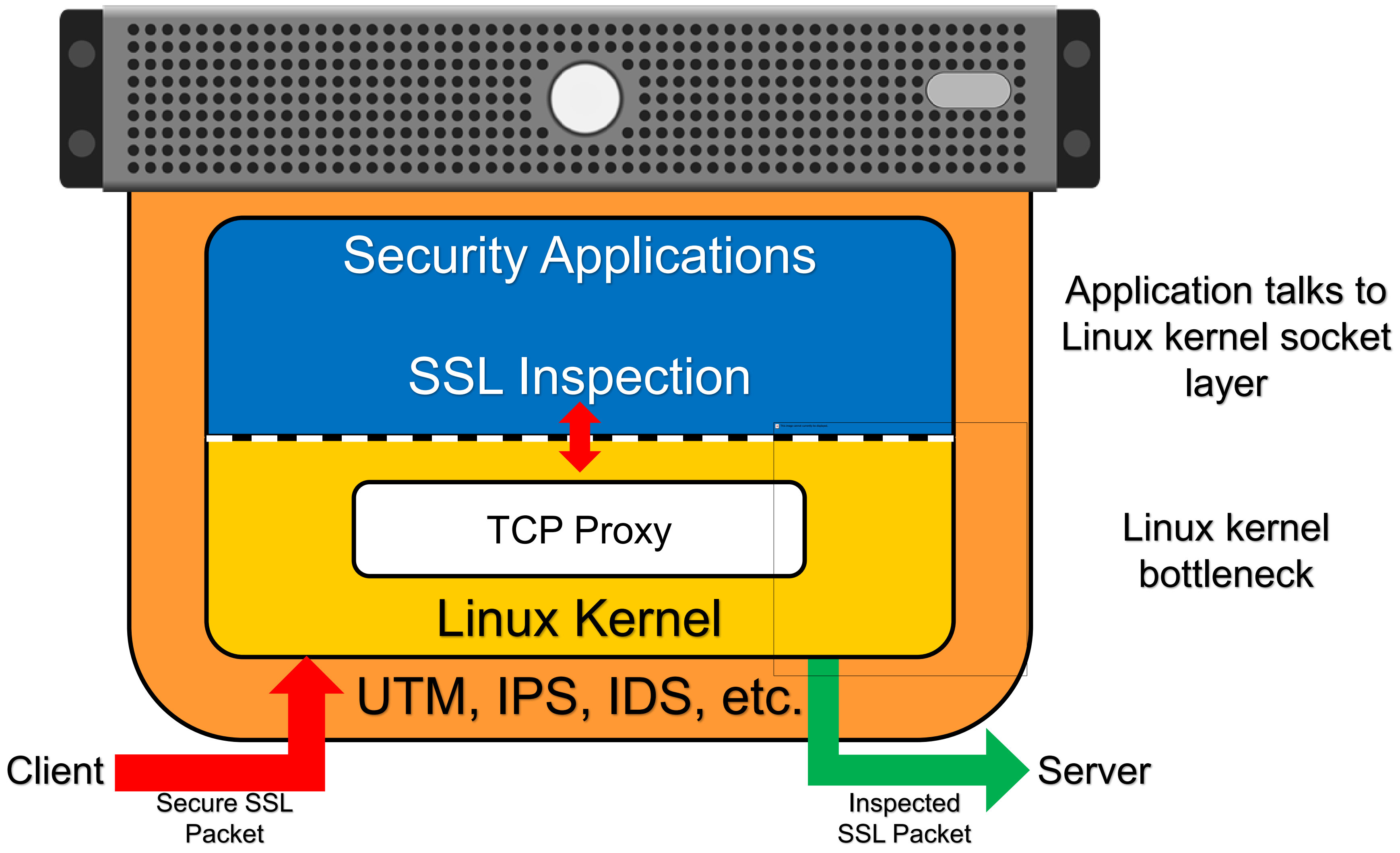
- **Performance of TCP applications is limited by Linux**

- **6WINDGate provides a high performance TCP stack relying on the 6WINDGate architecture to offload packet processing from Linux**
  - High throughput
  - Low latency
  - Large number of simultaneous sessions
  - Fast session establishment rate

- **Let’s take two examples using TCP Proxy and TCP Termination**

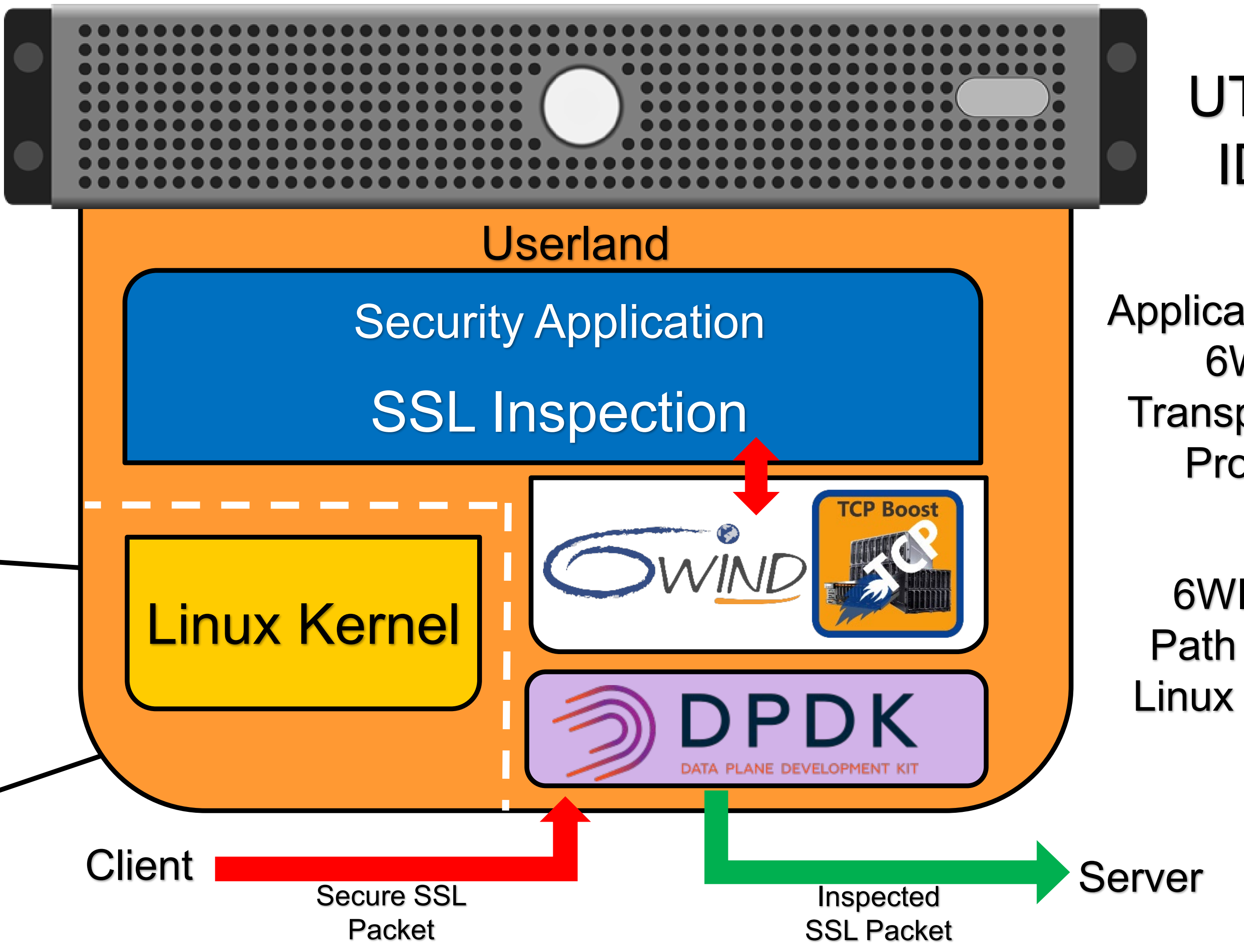
SSL Inspection for Cyber Threat Protection

- Cyber Threat Protection Devices (UTM, IPS, IDS etc.) include SSL Inspection solutions built on TCP proxies
- TCP proxy performance is limited by Linux kernel bottlenecks that cripple SSL Inspection speed
- High performance TCP proxy solutions are required to remove the Linux bottlenecks



6WINDGate High Performance TCP Proxy For SSL Inspection

- Number of concurrent sessions: 8 million
- Connection rate: 1 million CPS
- Transaction rate: 7.1 million TPS
- Throughput: 12 Gbps per core
- Latency: 24 µs
- Integrated with 6WINDGate L2-L3 protocol stacks including Linux synchronization
- Dedicated Transparent TCP Proxy APIs



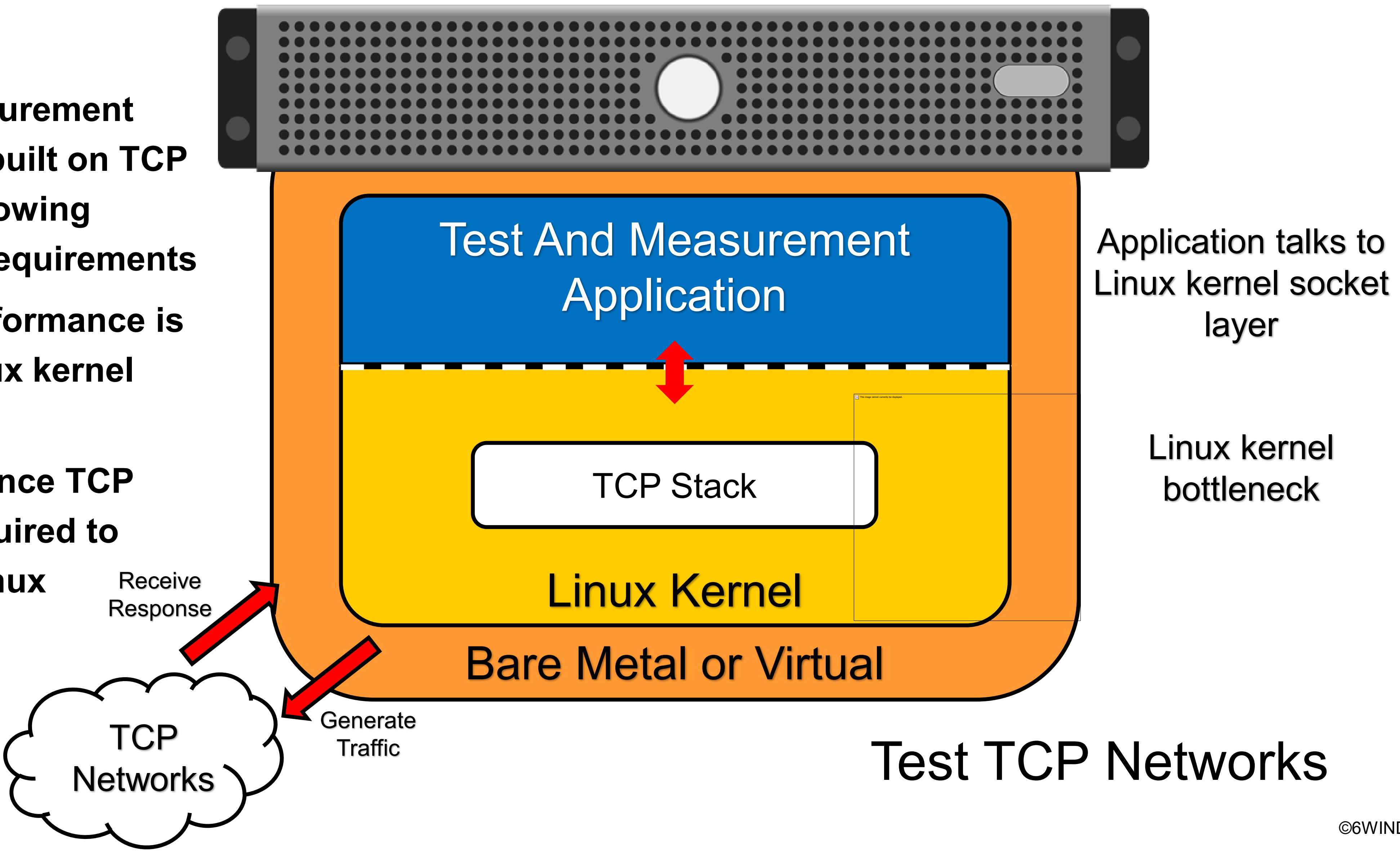
UTM, IPS, IDS, etc.

Application Talks to 6WIND's Transparent TCP Proxy APIs

6WIND Fast Path Removes Linux Bottleneck

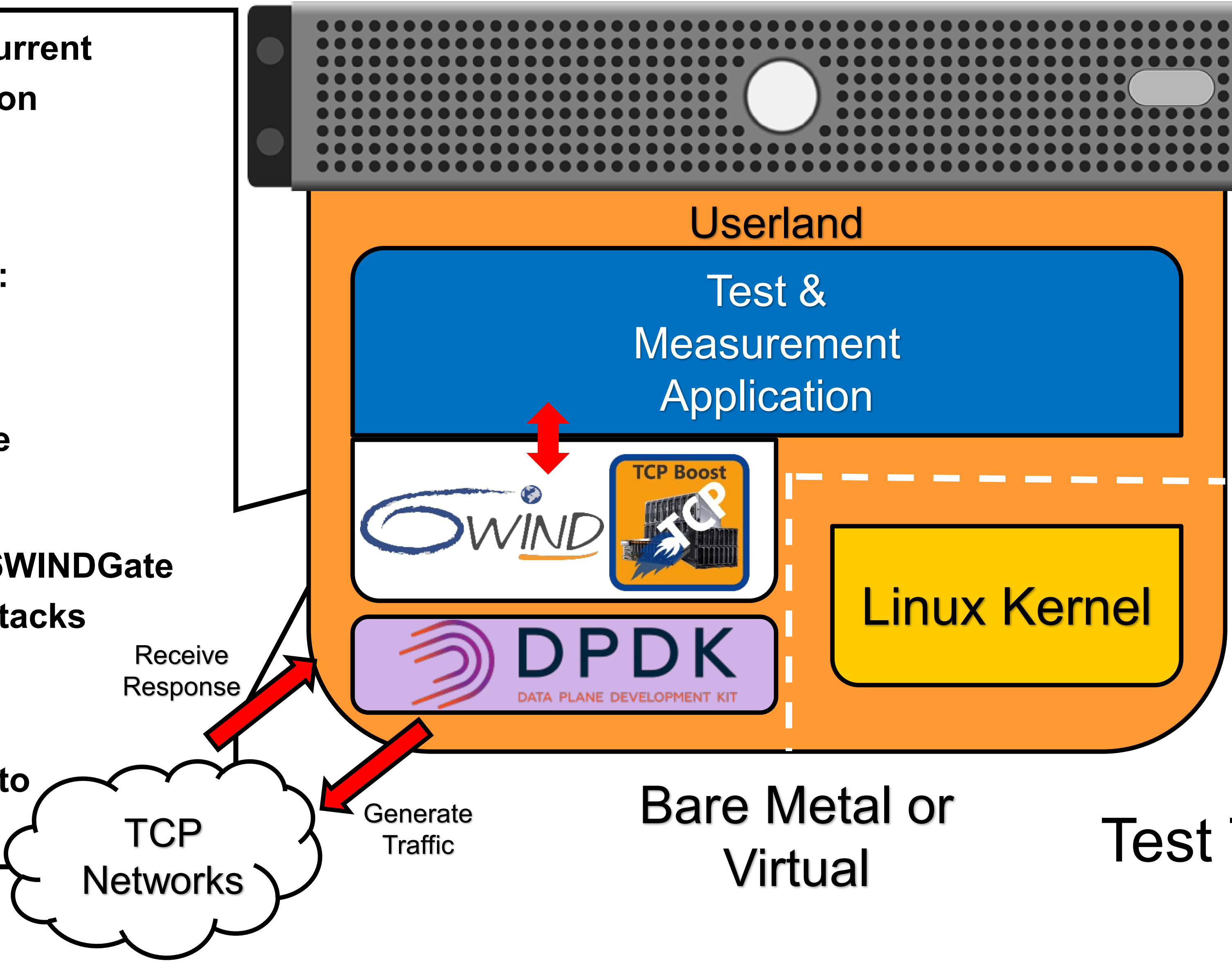
Test and Measurement Solutions for TCP Networks

- Test and Measurement solutions are built on TCP stacks with growing performance requirements
- TCP stack performance is limited by Linux kernel bottlenecks
- High performance TCP stacks are required to remove the Linux bottlenecks



6WINDGate High Performance TCP Stack For Test And Measurement Solutions

- Number of concurrent sessions: 6 million
- Connection rate: 1.4 million CPS
- Transaction rate: 7.1 million TPS
- Throughput: 12 Gbps per core
- Latency: 24 µs
- Integrated with 6WINDGate L2-L3 protocol stacks including Linux synchronization
- Extensible APIs to collect statistics



Application Talks to 6WIND's TCP Stack APIs

6WIND Fast Path Removes Linux Bottlenecks

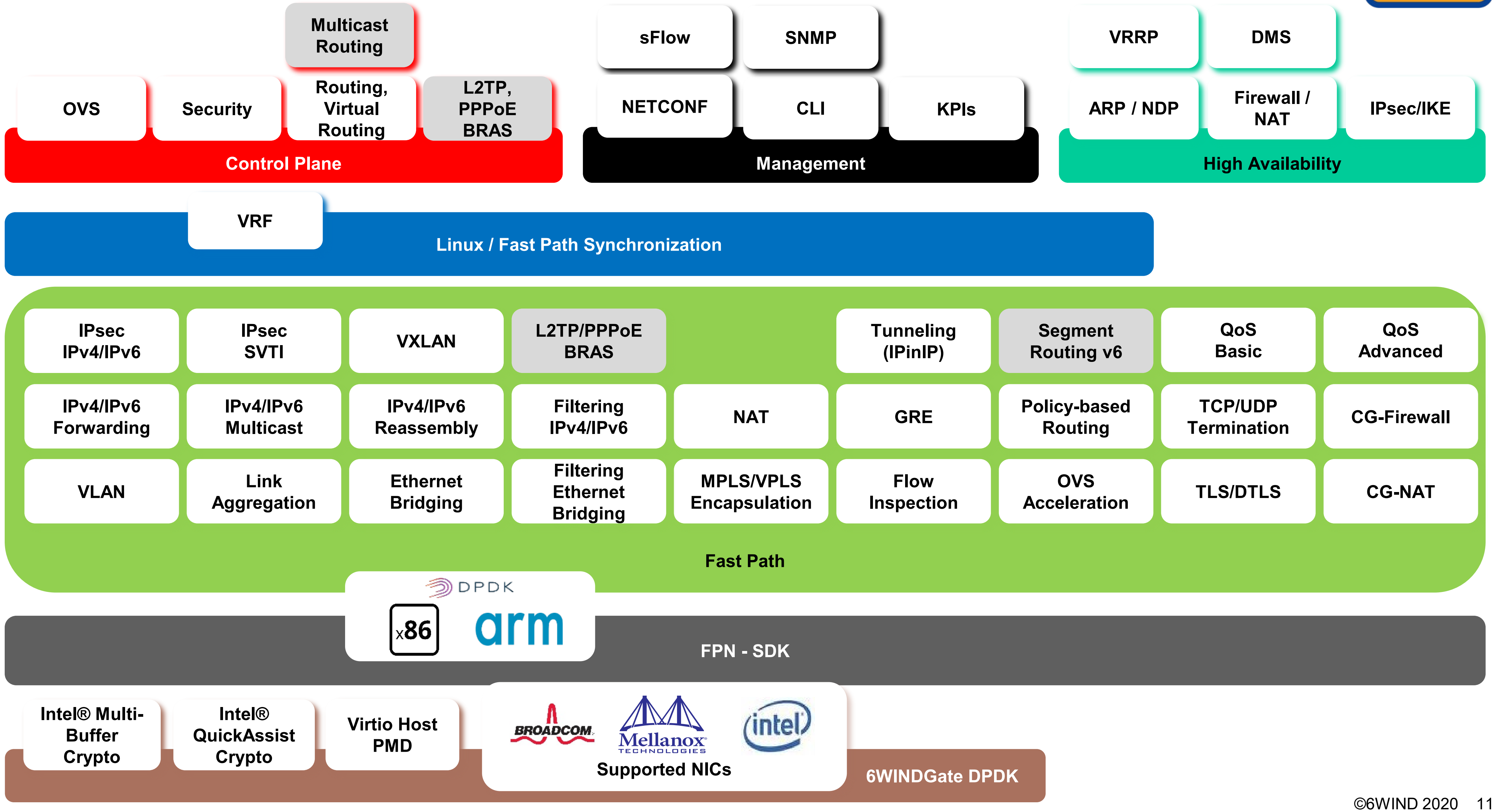
Bare Metal or Virtual

Test TCP Networks

6WINDGate TCP - Product Presentation

#SPEEDMATTERS For Serious Networks

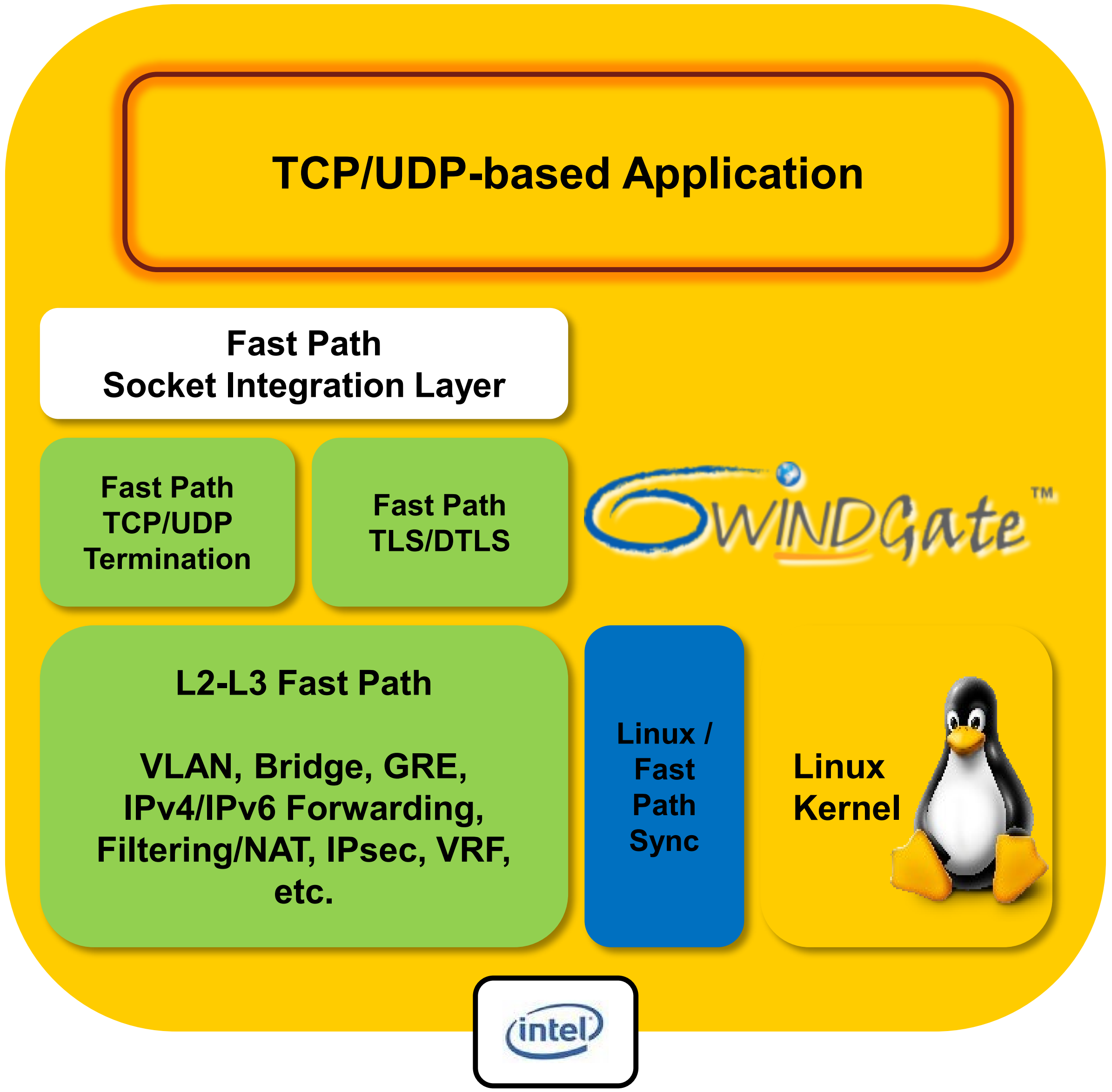
Accelerated Layer 2-4 Stacks Synchronized with Linux



Roadmap

6WINDGate TCP/UDP Termination

- **Software**
  - 6WINDGate source code license including the TCP modules and others 6WIND modules depending on the customer use case
- **Integrated with L2-L3 6WINDGate modules**
- **TCP stack configuration through dedicated CLI**
- **TCP/UDP-based application must be integrated with Fast Path Socket Integration Layer**

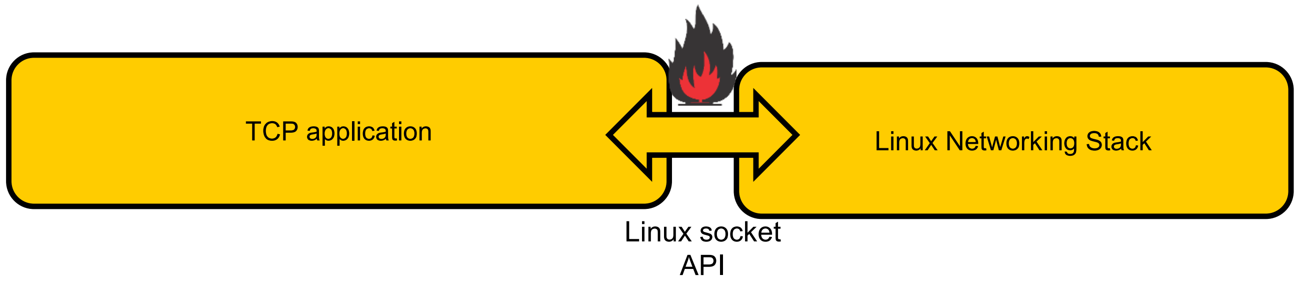

TCP/UDP-based Application
Fast Path Socket Integration Layer
Fast Path TCP/UDP Termination
Fast Path TLS/DTLS
6WINDGate™
L2-L3 Fast Path
VLAN, Bridge, GRE, IPv4/IPv6 Forwarding, Filtering/NAT, IPsec, VRF, etc.
Linux / Fast Path Sync
Linux Kernel
intel

6WINDGate TCP Implementation

- **Stack implementation**
  - Redesigned and fully optimized for multi-core execution environments
  - Highly scalable processing of parallel establishment / shutdown of TCP connections
  - High performance data exchanges on a huge number of TCP/UDP sockets on established TCP connections
  - Event driven notifications from the stack to the application
  - Plugin support for custom TCP/UDP applications including bridged and routed transparent proxy support with configurable IP bypass lists

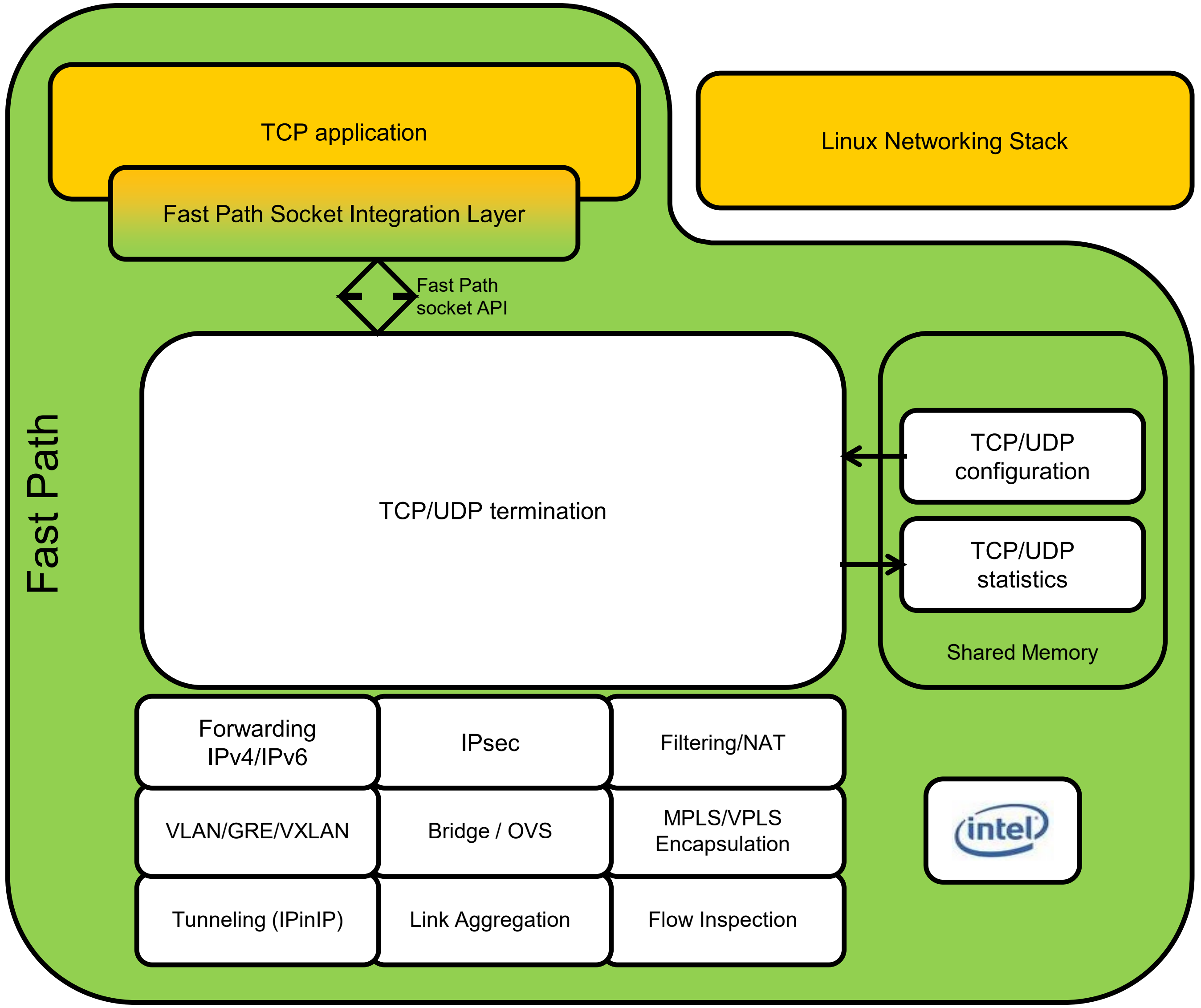
- **Socket API**
  - Full support of TCP and UDP sockets over IPv4 and IPv6
  - POSIX-compliant socket API and Zero-copy based socket APIs
  - VRF-aware sockets
  - Netstat like support to dump state and statistics of the sockets

Architecture



- **TCP application performance suffers from Linux networking stack bottlenecks**

Architecture



- **Fast Path TCP/UDP termination**
  - TCP/UDP protocols are processed in the Fast Path
  - Full featured TCP/UDP stack using BSD-like socket API
  - Timers are re-designed to get more scalability
  - Locks are removed
  - Memory footprint is reduced
- **Performance**
  - Scale: 8M active concurrent TCP sockets
  - Throughput: 40+ Gbps
  - CPS: 1.47M TCP connections per second
  - TPS: 7.1M TCP transactions per second
  - Latency TTFB: 24 µs
- **Optimized Fast Path TCP/UDP socket implementation**
  - Using event-based socket callbacks
  - Latency of socket calls is minimized

6WINDGate TCP Features Details

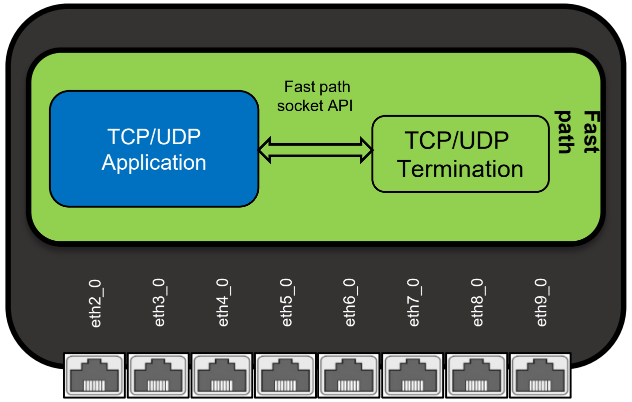
- **Available**
  - TCP_SACK and TCP_FACK
  - TCP_QUICKACK
  - Socket options to retrieve/Set TTL, MSS, TOS, DF bit
  - Reno, New Reno
  - ECN support (RFC 3168)
  - TCP Protection against wrapped sequence number
  - TCP Appropriate Byte Counting (RFC 3465)
  - TCP Segment Offload (TSO) support
  - Window Scaling
  - L2 bridge hook for transparent proxy
  - UDP transparent proxy
  - TLS v1.2/v1.3 support
  - DTLS v1.2/v1.3

- **20/Q1**
  - Per socket rate-limit (SO_MAX_PACING_RATE)
  - Initial congestion window per route (initcwnd)
  - Cubic congestion algorithm
  - TCP early retransmit (RFC 5827)
  - TCP Fast Open (RFC 7413)
  - Optimizations
    - Mbuf clone support: avoid copy on transmit side
    - Bulk API
  - L2 bridge plugin enhancement for Transparent proxy
    - Support L2 flow association with socket (ETH/VLAN)
  - TCP syn cookies

- **20/Q3 and next**
  - PATH MTU discovery and ICMP support
  - Duplicate SACK (RFC 3708), challenge ack limit
  - TCP fast RTO (RFC 5682)
  - Slow Start After Idle (RFC 5681)
  - L2 bridge plugin enhancement for Transparent proxy
    - QinQ
    - VxLAN
    - PPPoE
    - GTP-U
    - L2TPv2/v3
  - Full transparency: Socket creation in connected state

Benchmarks Platform

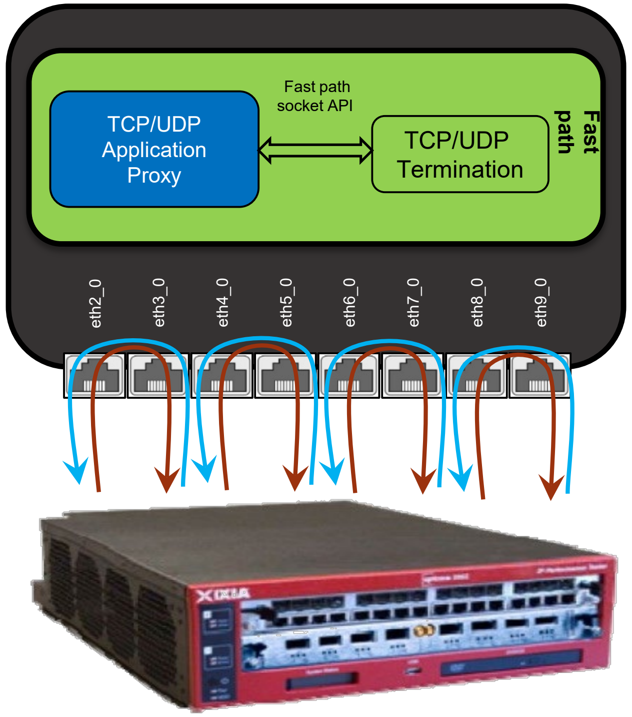
- **Tester is IXIA XT80 using IxLoad 8.01.106.3**
- **Node under test**
  - CPU: 2 x Intel(R) Xeon(R) Platinum 8170 CPU @ 2.10GHz
  - RAM: 48 GB DDR3
  - NICs: 4 x Intel X520 and 82599ES dual port 10G
- **Benchmarks**
  - Proxy
    - CPS
    - Bandwidth
  - Server
    - CPS
    - TPS
    - Bandwidth
    - Latency



Proxy Benchmarks

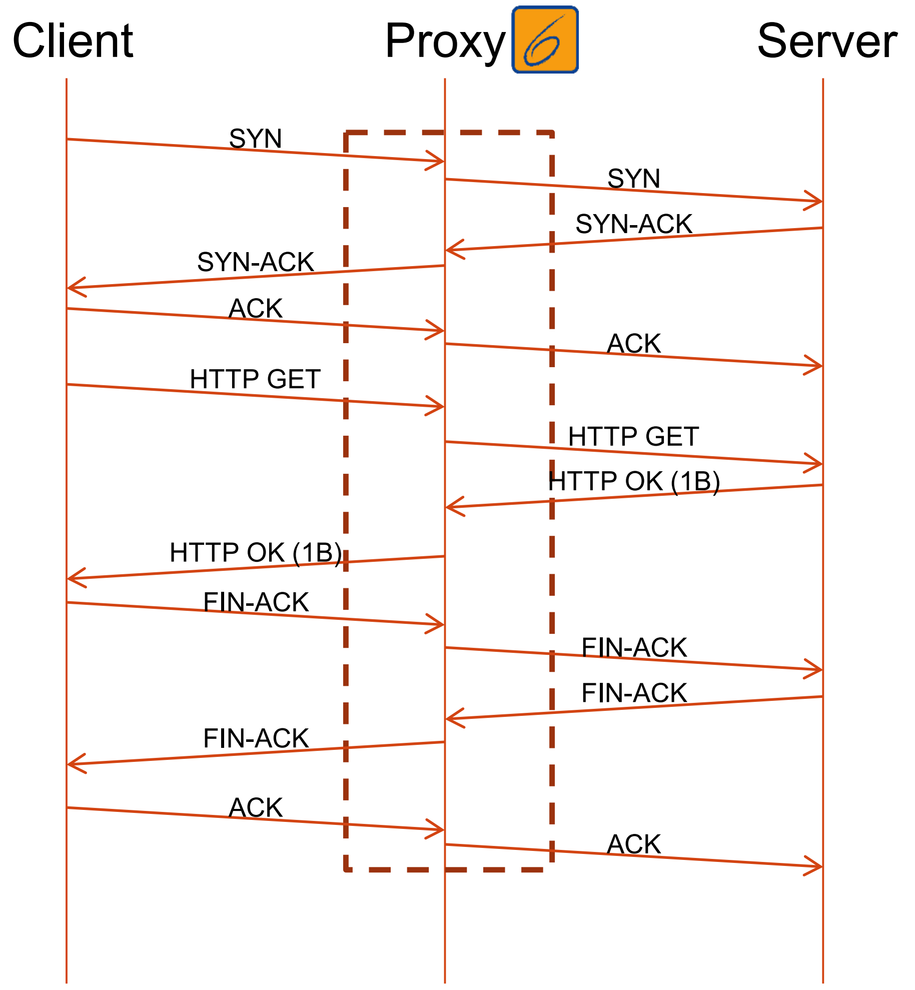
- **TCP connection rate**
  - Reported result is system TCP continuous socket open and close per second capacity
  - TCP sockets are opened first to reach socket objective (10K to 8M sockets)
  - Once objective is reached, sockets are continuously closed and opened at maximum rate
  - The connection rate reported is the average rate measured

- **TCP bandwidth**
  - Reported result is throughput received by IXIA
  - TCP sockets are opened first to reach socket objective (10K to 2M+ sockets)
  - New sockets opened are used for traffic emission and reception



Proxy Connection Rate Test

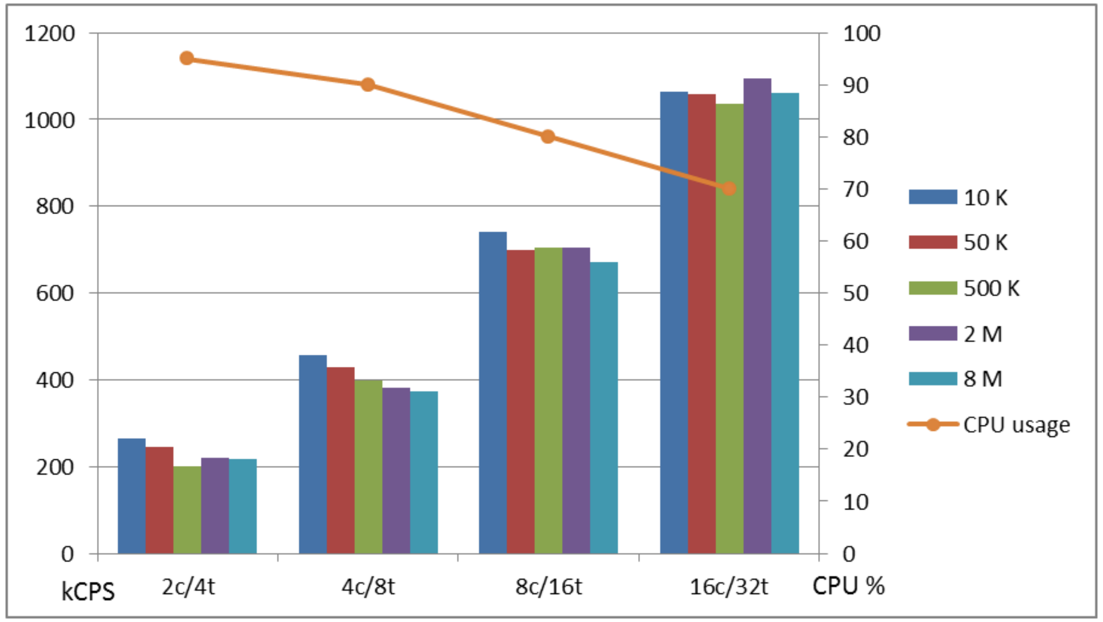
- **6WINDGate TCP proxy application running on node under test**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
- **Once done, sockets are continuously opened and closed**
- **IXIA measures maximum number of sockets per second**
- **The test is successful when all sockets are opened and closed correctly**



one connection = 2 sockets

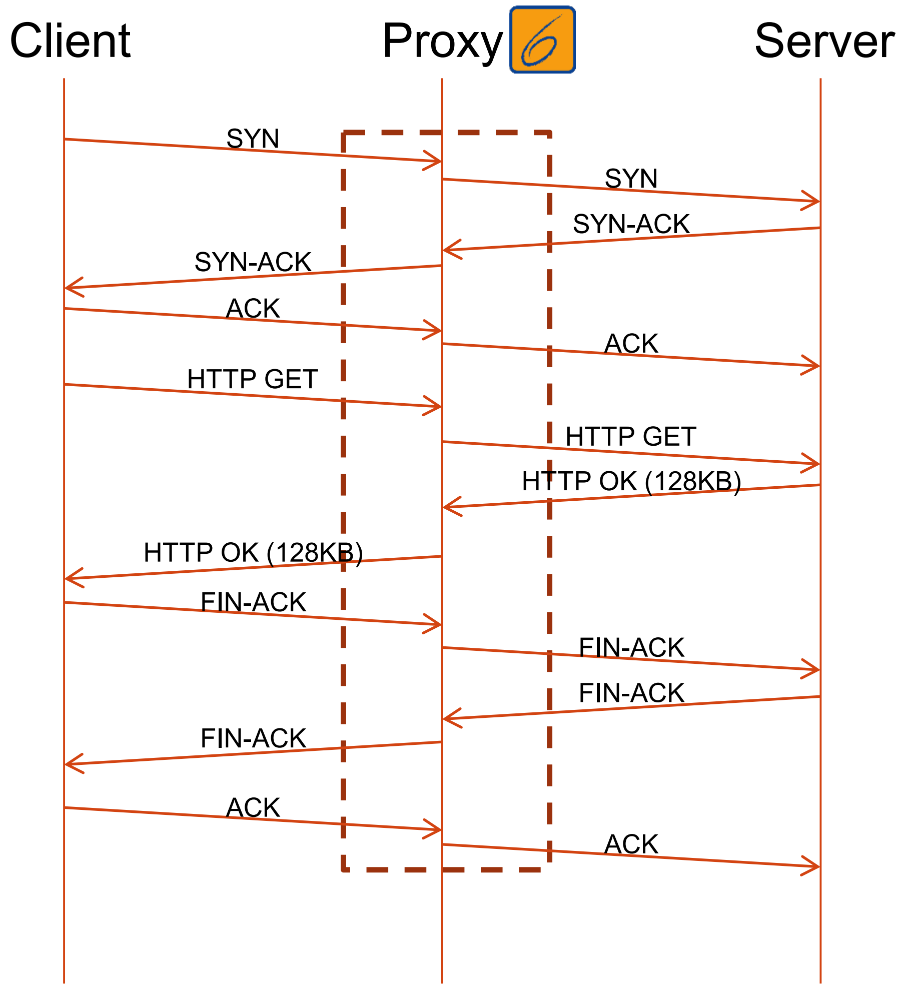
Proxy Connection Rate Results

- **Up to 1M socket per second using 16 cores and 8M concurrent sockets**
  - All connections are established properly
  - The number of concurrent connections impact is limited



Proxy Bandwidth Test

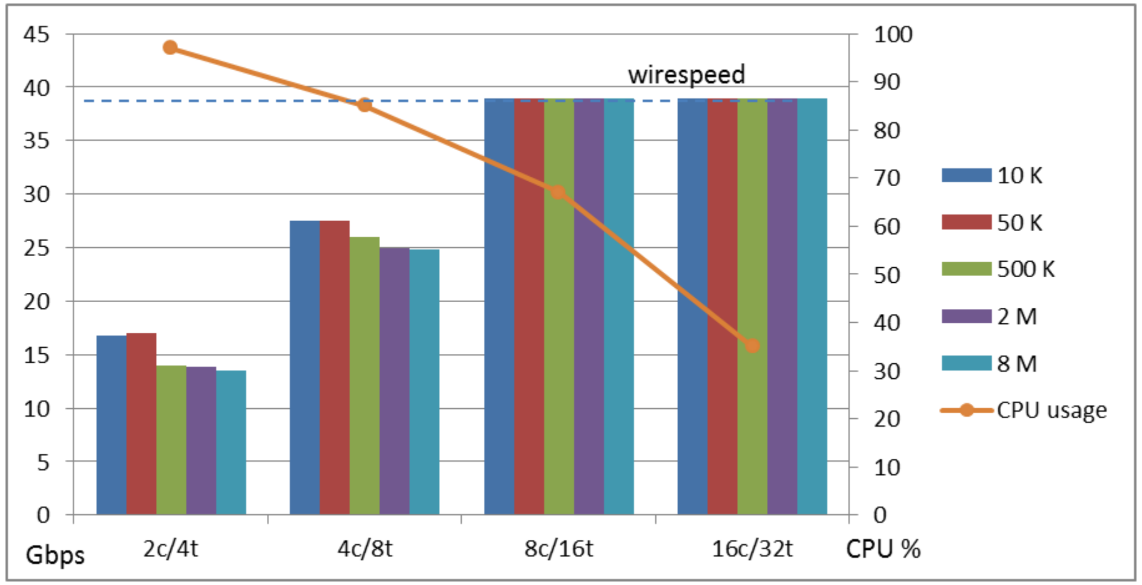
- **6WINDGate is running TCP proxy application**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
- **Once done, connections are continuously opened and closed to create load**
  - The page size is 128 KB
- **IXIA measures the throughput of the proxy**



one connection = 2 sockets

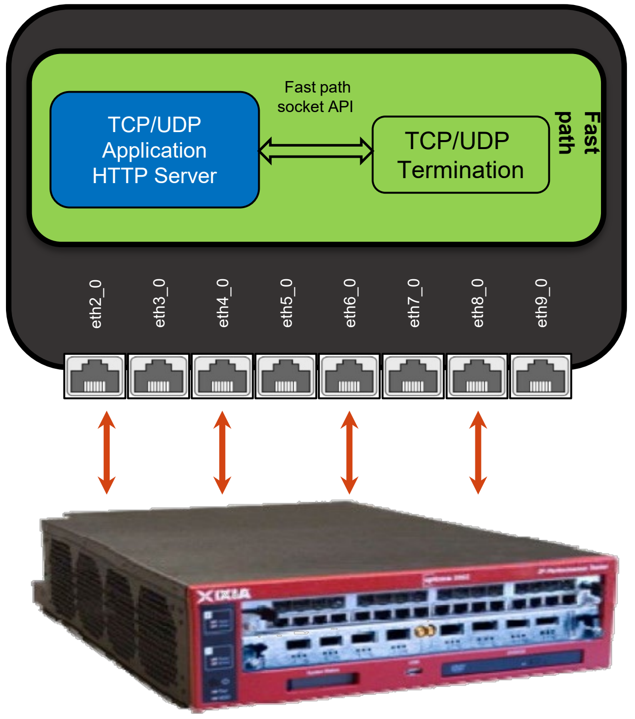
Proxy Bandwidth Results

- **Bandwidth performance remains stable with 8M active concurrent sockets**
- **Performance is limited by IXIA maximum capacity 40Gbps**
- **CPU usage decreases as more CPU resources are allocated**
  - Leaving more CPU resources available for application processing



Server Benchmarks

- **TCP socket rate**
  - Reported result is system TCP continuous socket open and close per second capacity
  - TCP sockets are opened first to reach socket objective (10K to 8M sockets)
  - Once objective is reached, sockets are continuously closed and opened at maximum rate
  - The socket rate reported is the average rate measured
- **TCP bandwidth**
  - Reported result is throughput received by IXIA
  - TCP sockets are opened first to reach socket objective (10K to 2M+ sockets)
  - New sockets opened are used for traffic emission and reception
- **TCP transaction rate**
  - Reported result is system HTTP requests served per second
  - TCP sockets are opened first to reach socket objective (10K to 2M+ sockets)
  - Several requests (10) are processed by connection opened
- **TCP latency**
  - Reported result is system latency to serve request first byte
  - TCP sockets are opened first to reach socket objective (10k)
  - Request rate is set to different values (1K to 1M TPS)



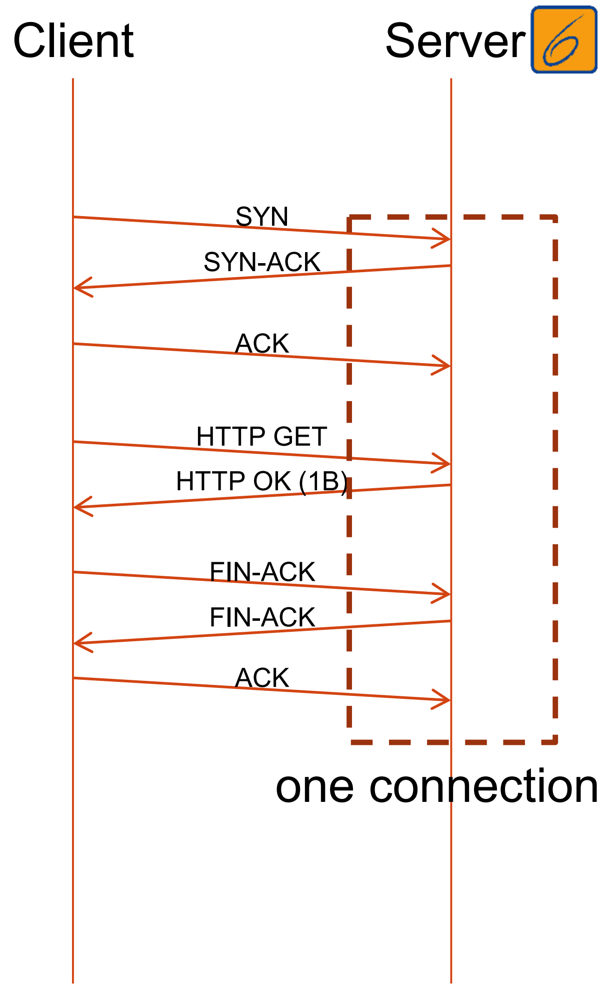
Server Connection Rate Test

- **6WINDGate TCP HTTP 1.1 server application running on node under test**
  - Single port 80 is used (worst case)

- **IXIA establishes connections until concurrent socket objective is reached**

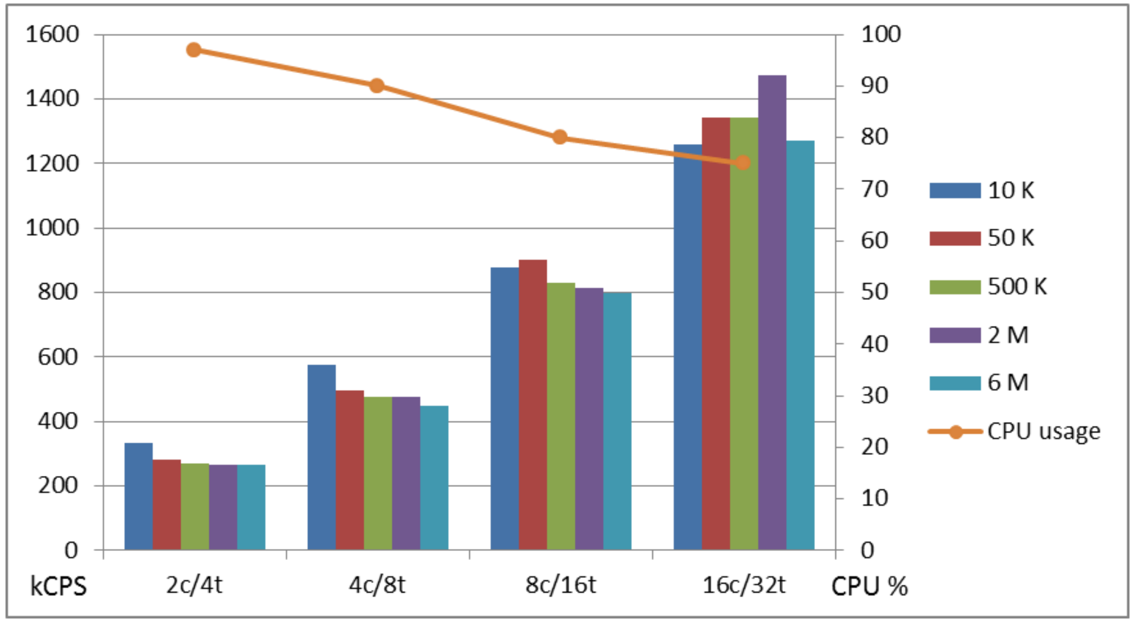
- **Once done, sockets are continuously opened and closed**

- **IXIA measures the maximum number of sockets per second**



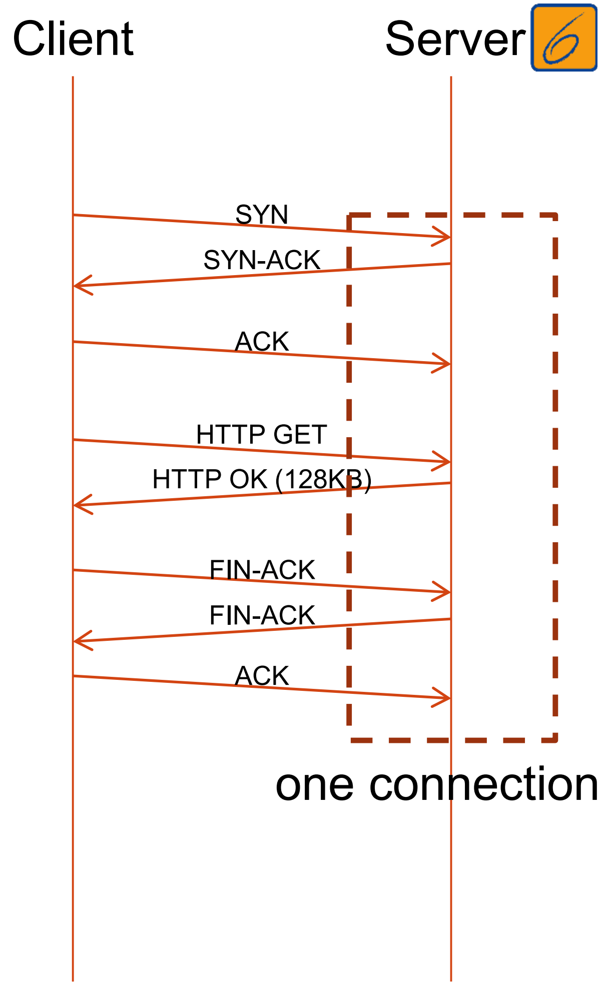
Server Connection Rate Results

- **Up to 1.47M sockets per second using 16 cores and 6M concurrent sockets**
  - All connections are established properly
  - The number of concurrent connections impact is limited



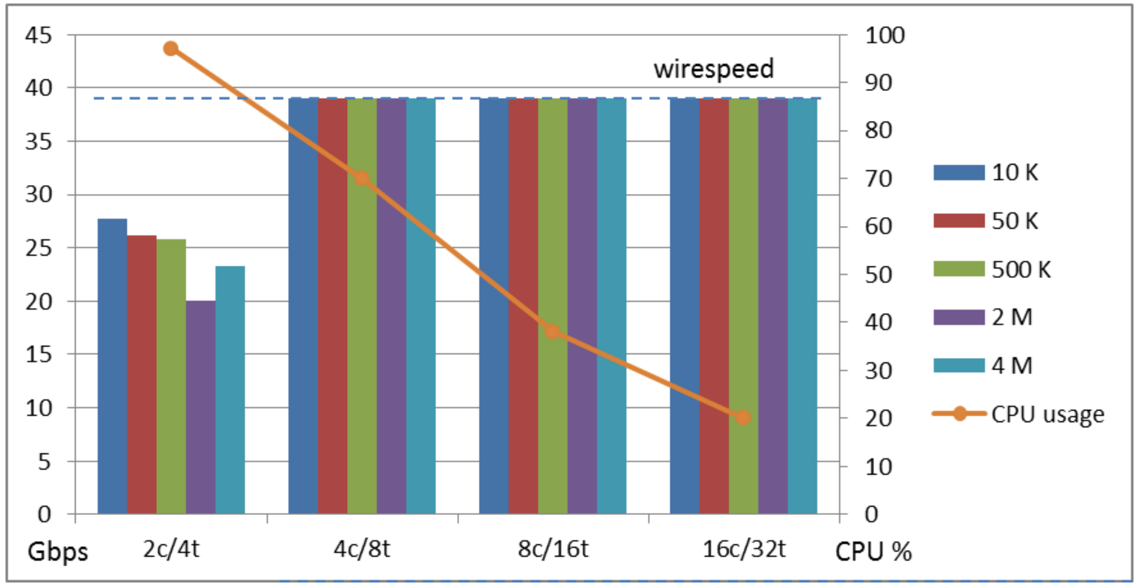
Server Bandwidth Test

- **6WINDGate is running TCP HTTP 1.1 server application**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
- **Once done, sockets are continuously opened and closed to create load**
  - The page size is 128 KB
- **IXIA measures the throughput of the server**



Server Bandwidth Results

- **Bandwidth performance remains stable with 4M active concurrent sockets**
- **Performance is limited by IXIA maximum capacity 40Gbps**
- **CPU usage decreases as more CPU resources are allocated**
  - Leaving more CPU resources available for application processing

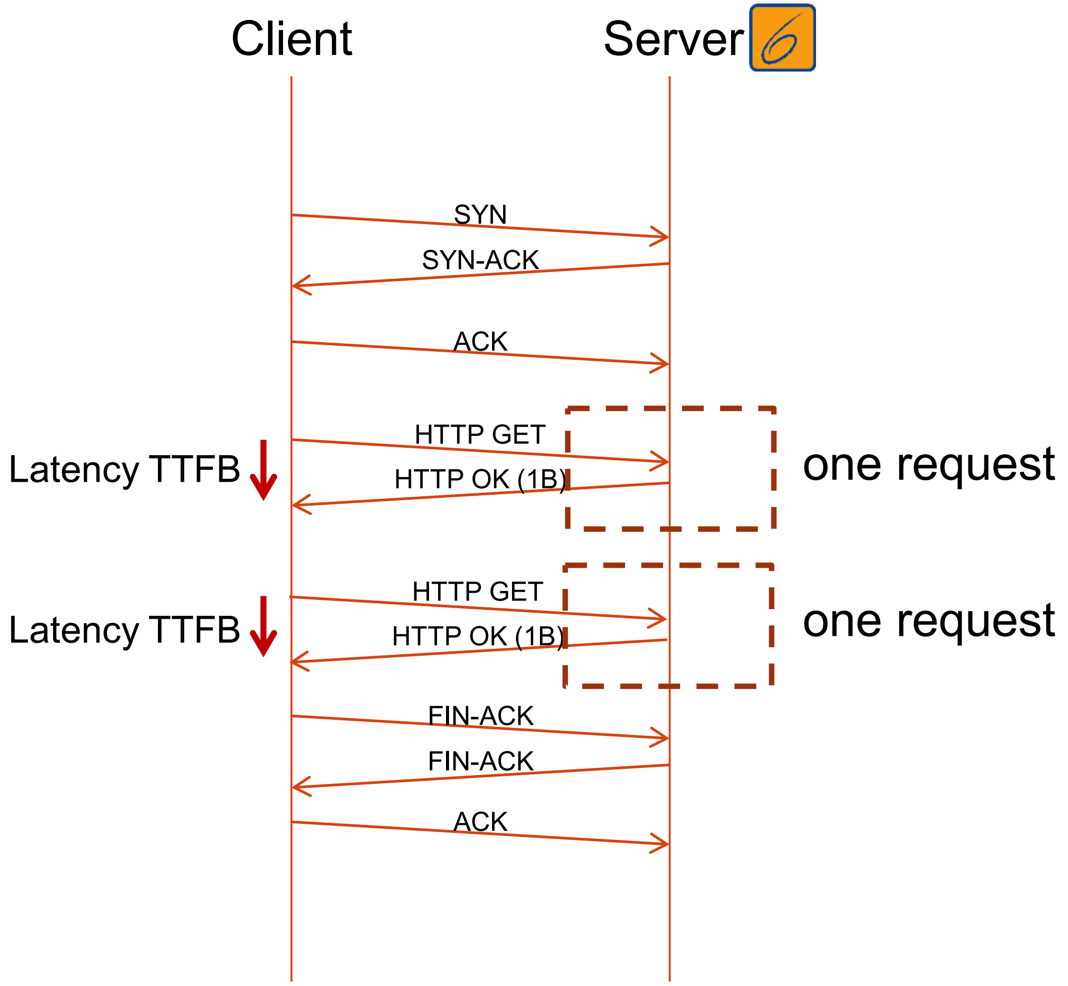


Server Transaction Rate and Latency

- **6WINDGate TCP HTTP 1.1 server application running on node under test**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
- **Once done, multiple requests are sent by client to server**
- **IXIA measures the maximum number of transactions per second and the time to first byte (TTFB)**
  - The test is successful when all requests are served correctly
  - Latency is measured by IXIA between the emission of HTTP GET and the first byte reception of the response

Maximum TPS measured is 7.1 M

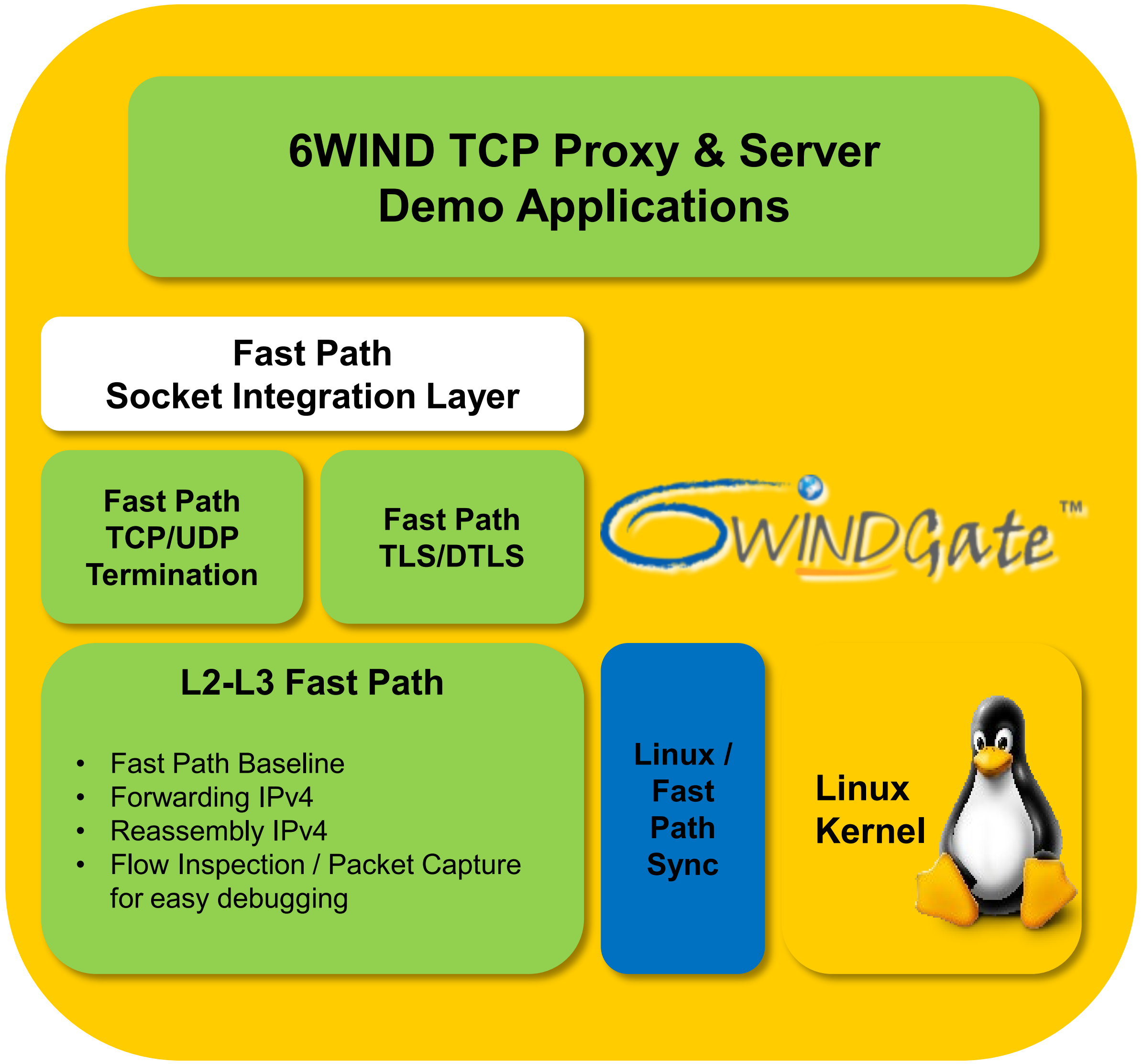
Minimum measured latency is 24 µs



TCP Boost Evaluation Package

- Evaluation package available to replay benchmarks

- TCP Boost Evaluation Package documentation
- 6WINDGate modules in binary code
- 6WIND Proxy and Server demo applications in source code


6WIND TCP Proxy & Server Demo Applications
Fast Path Socket Integration Layer
Fast Path TCP/UDP Termination
Fast Path TLS/DTLS
6WINDGate™
L2-L3 Fast Path
• Fast Path Baseline
• Forwarding IPv4
• Reassembly IPv4
• Flow Inspection / Packet Capture for easy debugging
Linux / Fast Path Sync
Linux Kernel

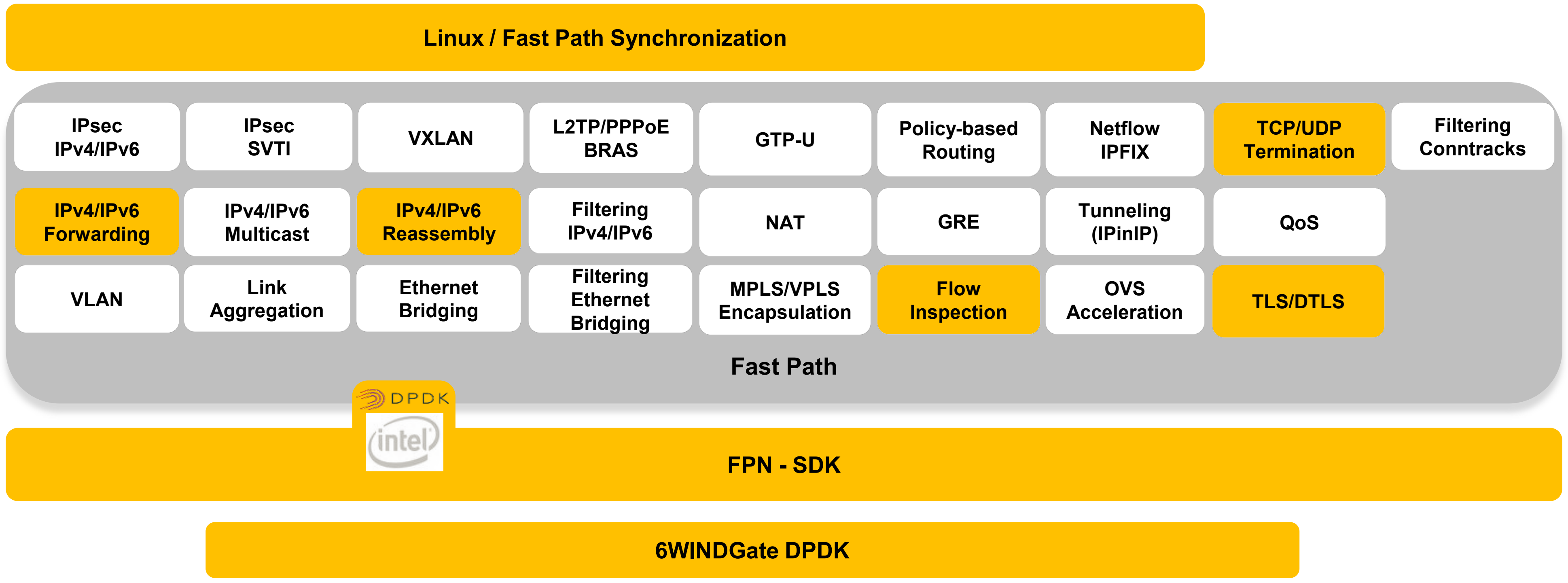
6WINDGate TCP Offering

6WIND

#SPEEDMATTERS For Serious Networks

6WINDGate TCP Offering

TCP Boost Software Package



- **TCP Boost Software Package**
  - Ready-to-use set of 6WINDGate modules bundled with consulting services

- **A la carte 6WINDGate modules**

TCP Boost Software Packages

- **6WINDGate UDP/TCP Termination IPv4 (License)**
  - 6WINDGate DPDK and FPN-SDK for Intel
  - Fast Path Baseline
  - Forwarding IPv4
  - Reassembly IPv4
  - Flow Inspection / Packet Capture for easy debugging
  - Linux - Fast Path Synchronization
  - TCP/UDP Termination IPv4
  - TCP/UDP Termination TLS/DTLS add-on

- **Add-On License for UDP/TCP Termination IPv6**
  - Forwarding IPv6
  - Reassembly IPv6
  - TCP/UDP Termination add-on IPv6

- **Services**
  - Training (3 days)
  - Engineering Consulting Services (10 days)
  - One Year of Maintenance included in License

6WINDGate TCP – Implementation

6WIND

#SPEEDMATTERS For Serious Networks

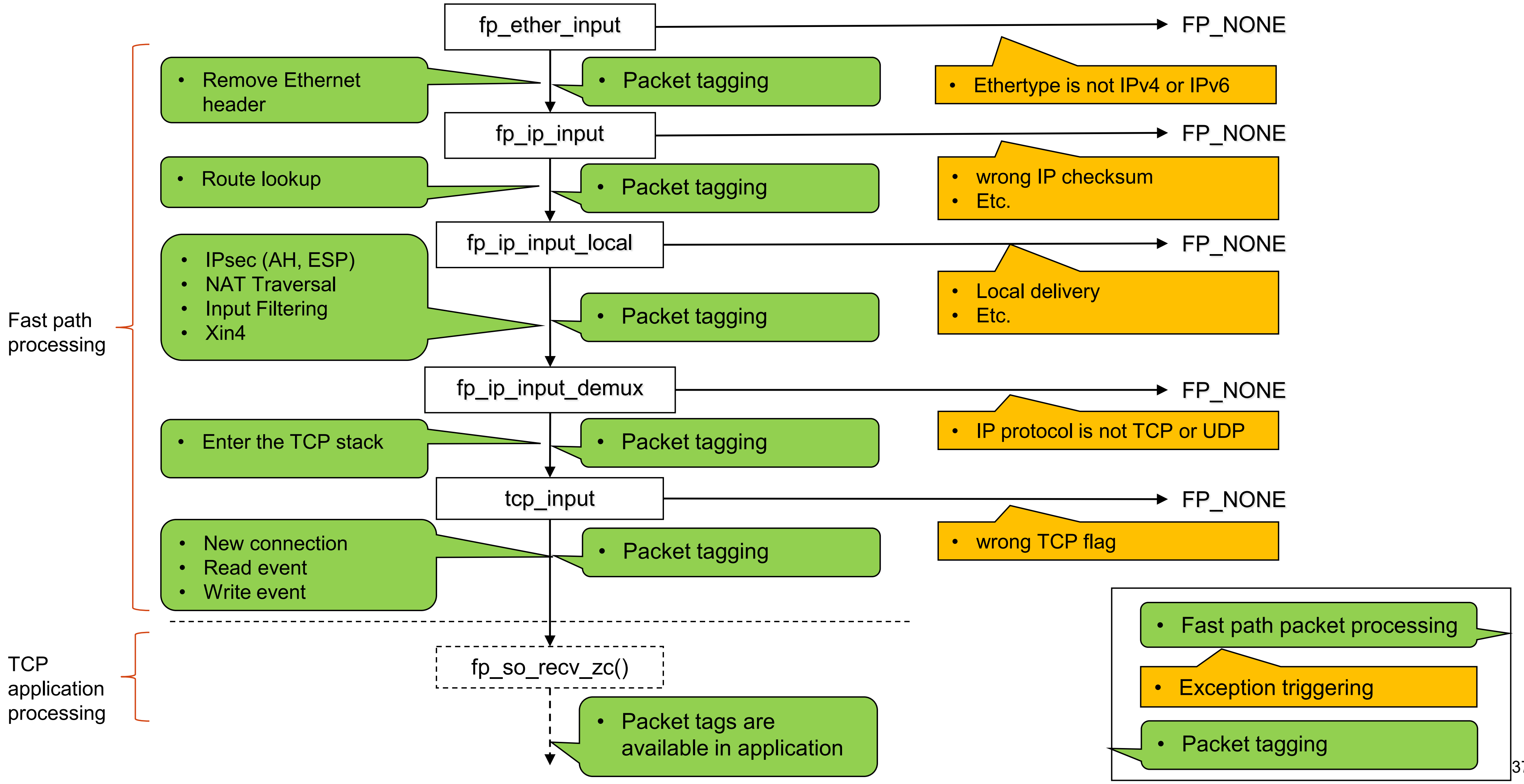
6WINDGate TCP/UDP Termination - Stack Design

- **It is not a NetBSD port on fast path (like Rump)**
  - Would be reusing BSD lock based data structures
  - Suffers from design bottlenecks
    - accept() would be running on one core and would limit CPS performance

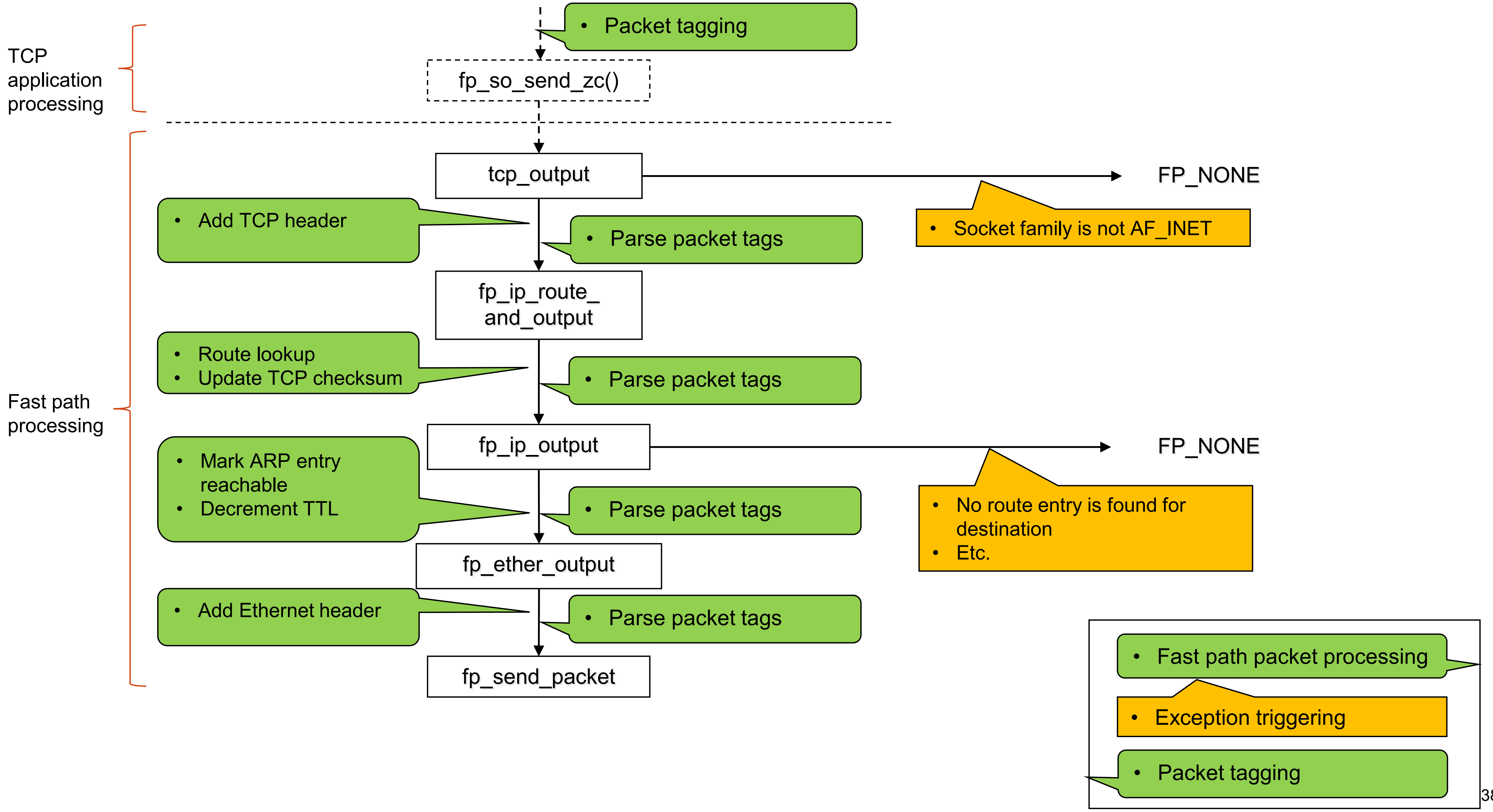
- **Not written from scratch**

- **6WIND reused *BSD design flow but rewrote code to scale with multicore architectures to**
  - Get benefits from a widely deployed TCP stack
  - Minimize CPU cycles per packet
  - Remove blocking calls, it is event callback based
  - Remove BSD bottlenecks
    - accept() is running on multiple cores

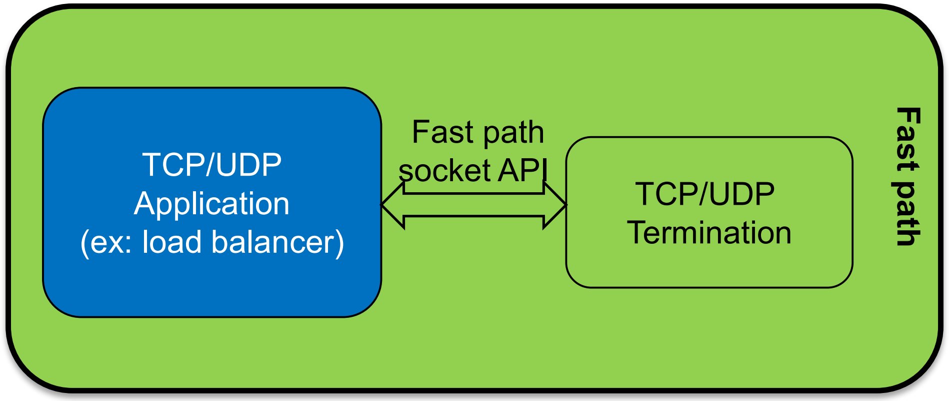
6WINDGate TCP/UDP Termination - Ingress



6WINDGate TCP/UDP Termination - Egress



6WINDGate TCP/UDP Termination - Fast Path Socket



- **Application is running on the dedicated fast path cores**
- **Data is provided to the application through an optimized fast path TCP/UDP socket implementation**
  - minimal TCP/UDP applications re-design using fast path socket API
- **Performance is maximized**
  - Zero-copy can be used to share buffers between fast path and TCP application
  - Latency of socket calls is minimized

6WINDGate TCP/UDP Termination - Fast Path Socket

- **System calls vs. callback functions**
  - For scalability, 6WINDGate TCP termination introduces callback functions in case of:
  - New connections
  - Read events
  - Write events

- **Those callbacks replace the following system calls:**
  - select()
  - poll()
  - epoll_wait()

6WINDGate TCP/UDP Termination - Fast Path Socket

- **Fast path socket API is similar to standard Linux socket API:**
  - fp_so_socket()
  - fp_so_close()
  - fp_so_send() / fp_so_send_zc() / fp_so_sendto() / fp_so_sendto_zc()
  - fp_so_recv() / fp_so_recvfrom() / fp_so_recv_zc() / fp_so_recvfrom_zc()
  - fp_so_bind()
  - fp_so_connect()
  - fp_so_listen()
  - fp_so_accept()
  - fp_so_getpeername()
  - fp_so_getsockname()
  - fp_so_getsockopt()
  - fp_so_setsockopt()

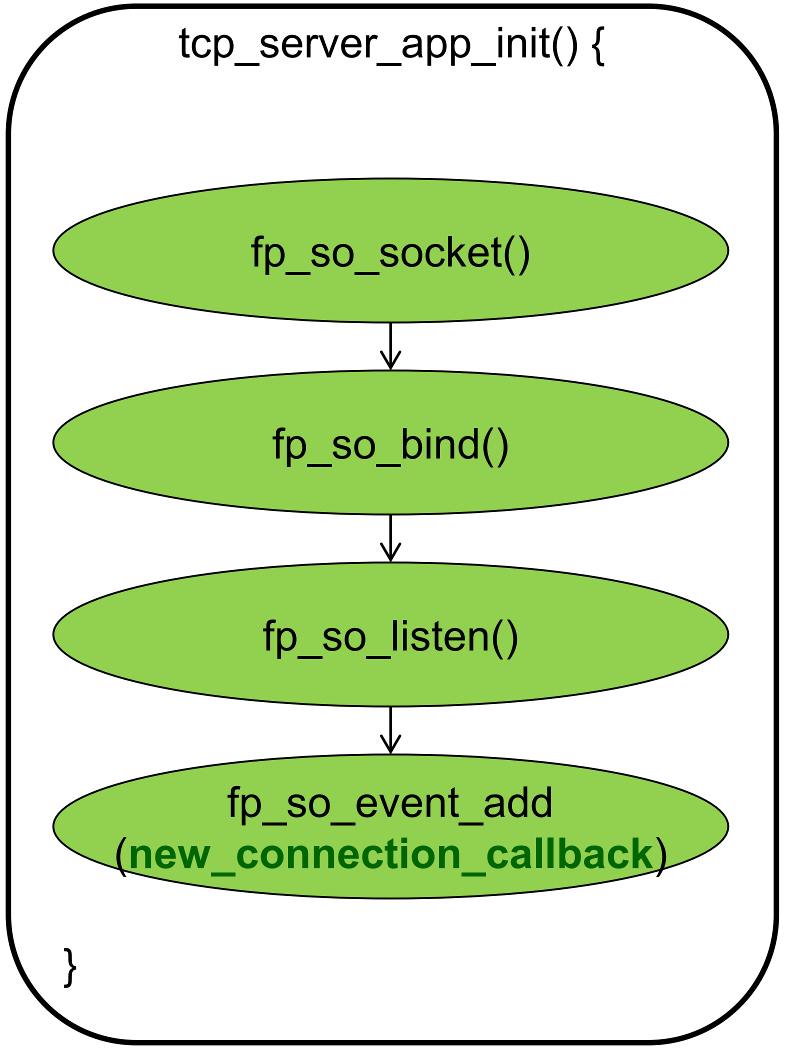
6WINDGate TCP/UDP Termination - Fast Path Socket Buffer Handling

- **6WINDGate TCP/UDP termination provides copy and zero-copy APIs**

- **Read/Write copy mode**
  - Returns a message buffer structure
  - Linear buffers
  - send() / recv() API
- **Read/Write zero-copy mode**
  - Returns a pointer to the message buffer structure
  - Scatter/Gather buffers : one vectored I/O read or write can replace many ordinary reads or writes
  - Faster than copy mode
  - send_zc() / recv_zc() API

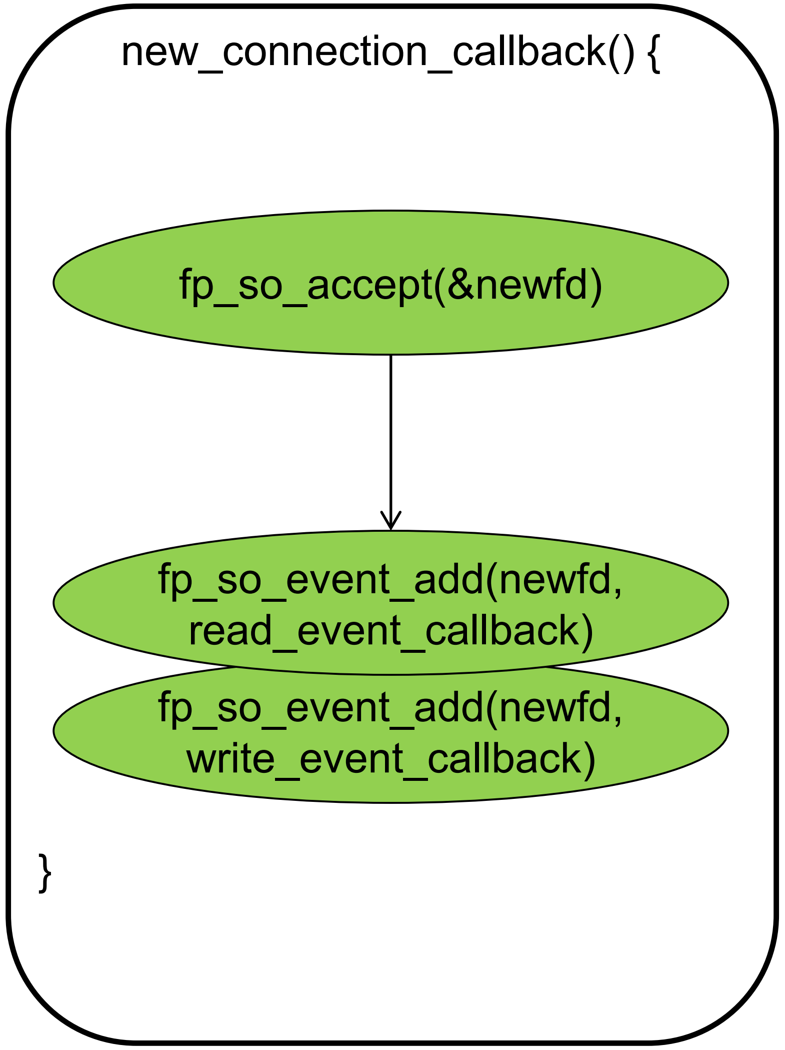
- **Why should we use copy mode?**
  - It allows to reuse existing malloc/free existing code with no modification.

6WINDGate TCP/UDP Termination - Fast Path Socket - Server



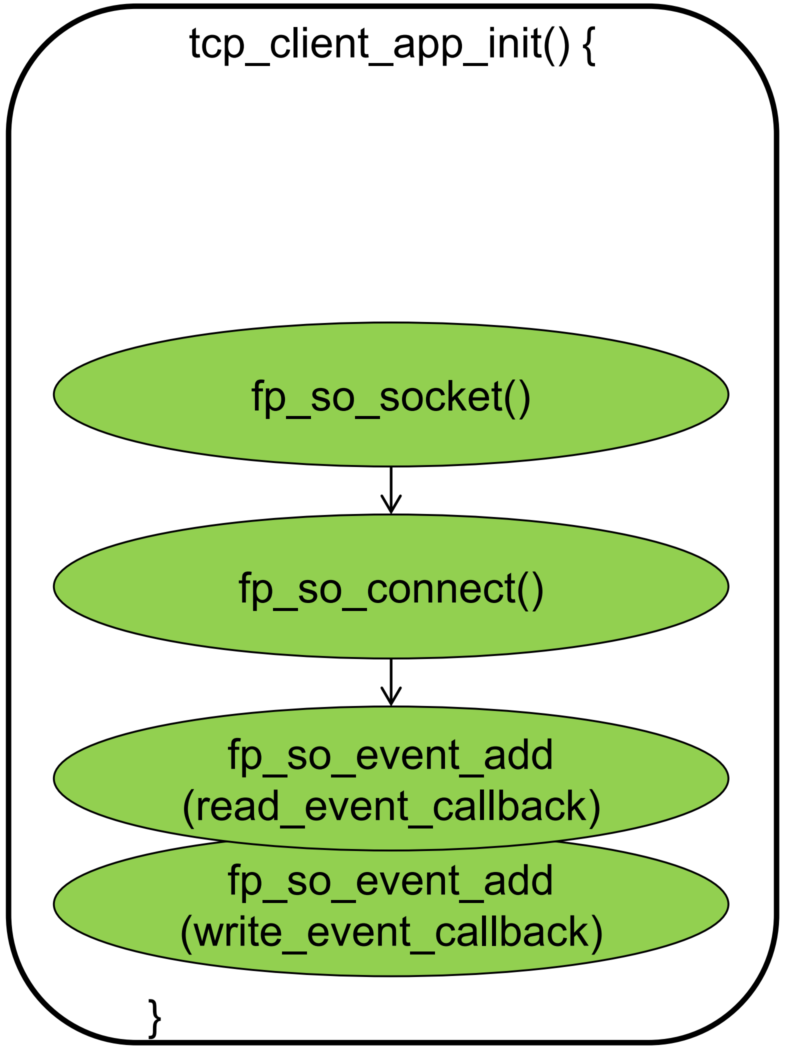
- **TCP application:**
  server socket initialization
- **Create fast path TCP socket**
- **Listen on a socket for a given IP address and port**
- **Register a callback function, called by the TCP termination module when receiving a new connection request**
  - Replace a blocking accept() call

6WINDGate TCP/UDP Termination - Fast Path Socket - Server


new_connection_callback() {
fp_so_accept(&newfd)
fp_so_event_add(newfd, read_event_callback)
fp_so_event_add(newfd, write_event_callback)
}

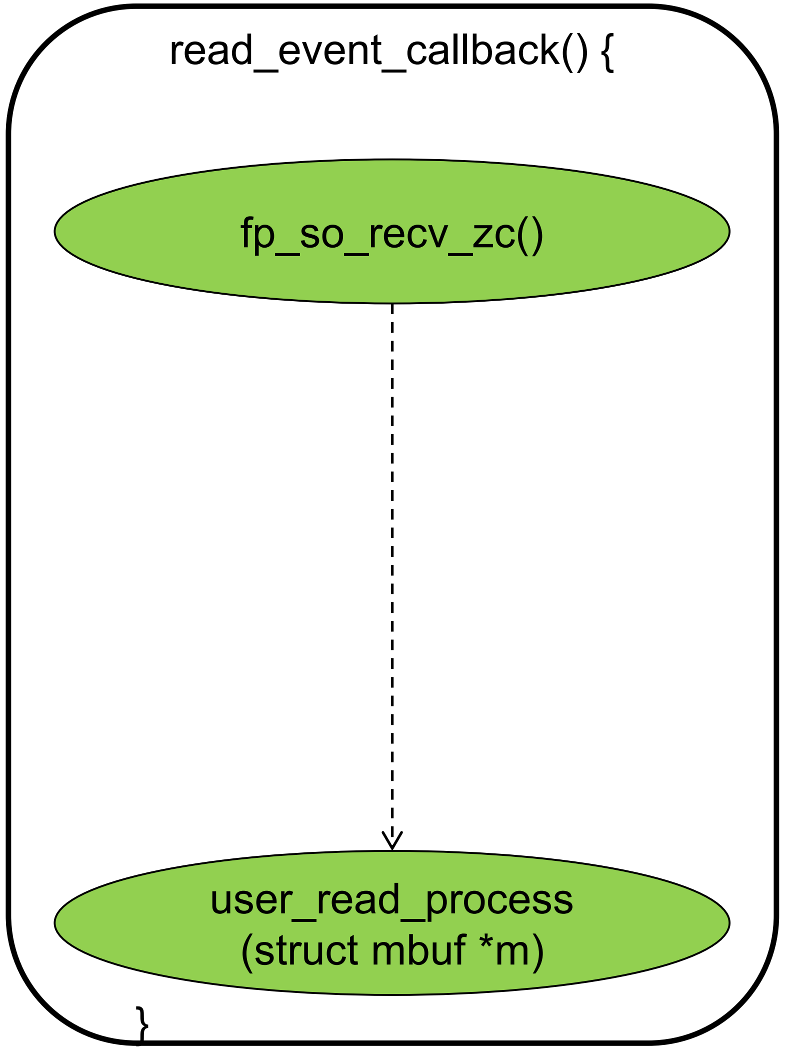
- **Called by the TCP termination module when a new connection occurs**
- **Called as soon as the syn/syn_ack/ack is done**
- **Accept connection**
- **Register callback functions for read and write events:**
  - Read event callback called when data is available to be read on the socket
  - Write event callback called when data can be sent on the socket
  - No select()/poll()/epoll()
    - replaced by r/w callbacks comparable to libevent

6WINDGate TCP/UDP Termination - Fast Path Socket - Client


tcp_client_app_init() {
fp_so_socket()
fp_so_connect()
fp_so_event_add (read_event_callback)
fp_so_event_add (write_event_callback)
}

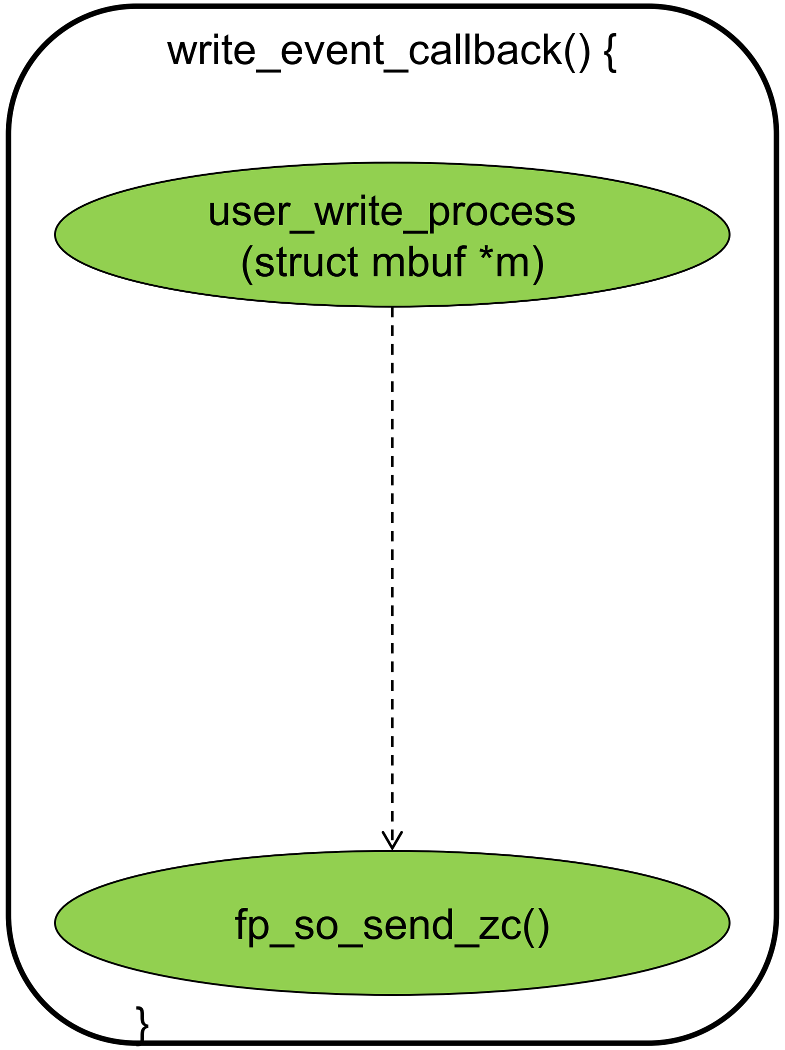
- **TCP application: client socket initialization**
- **Create fast path TCP socket**
- **Connect to a socket with a given IP address and port**
- **Register callback functions for read and write events**
  - Read event callback called when data is available to be read on the socket
  - Write event callback called when data can be sent on the socket
  - No select()/poll()/epoll(): replaced by r/w callbacks

6WINDGate TCP/UDP Termination - Fast Path Socket - Client and Server



- **TCP application: socket read event**
- **Called as soon as there is data on a socket**
- **Read data on the socket**
- **Return a pointer on a mbuf containing the data**

6WINDGate TCP/UDP Termination - Fast Path Socket - Client and Server



- **TCP application: socket write event**
- **Send data (as a mbuf) on the socket**
- **Called as soon as data can be sent on a socket.**

Thank You

#SPEEDMATTERS For Serious Networks