6WINDGate TCP

#SPEEDMATTERS For Serious Networks

6WINDGate Solution For High Performance TCP-Based Applications

6WIND

#SPEEDMATTERS For Serious Networks

The TCP Problem

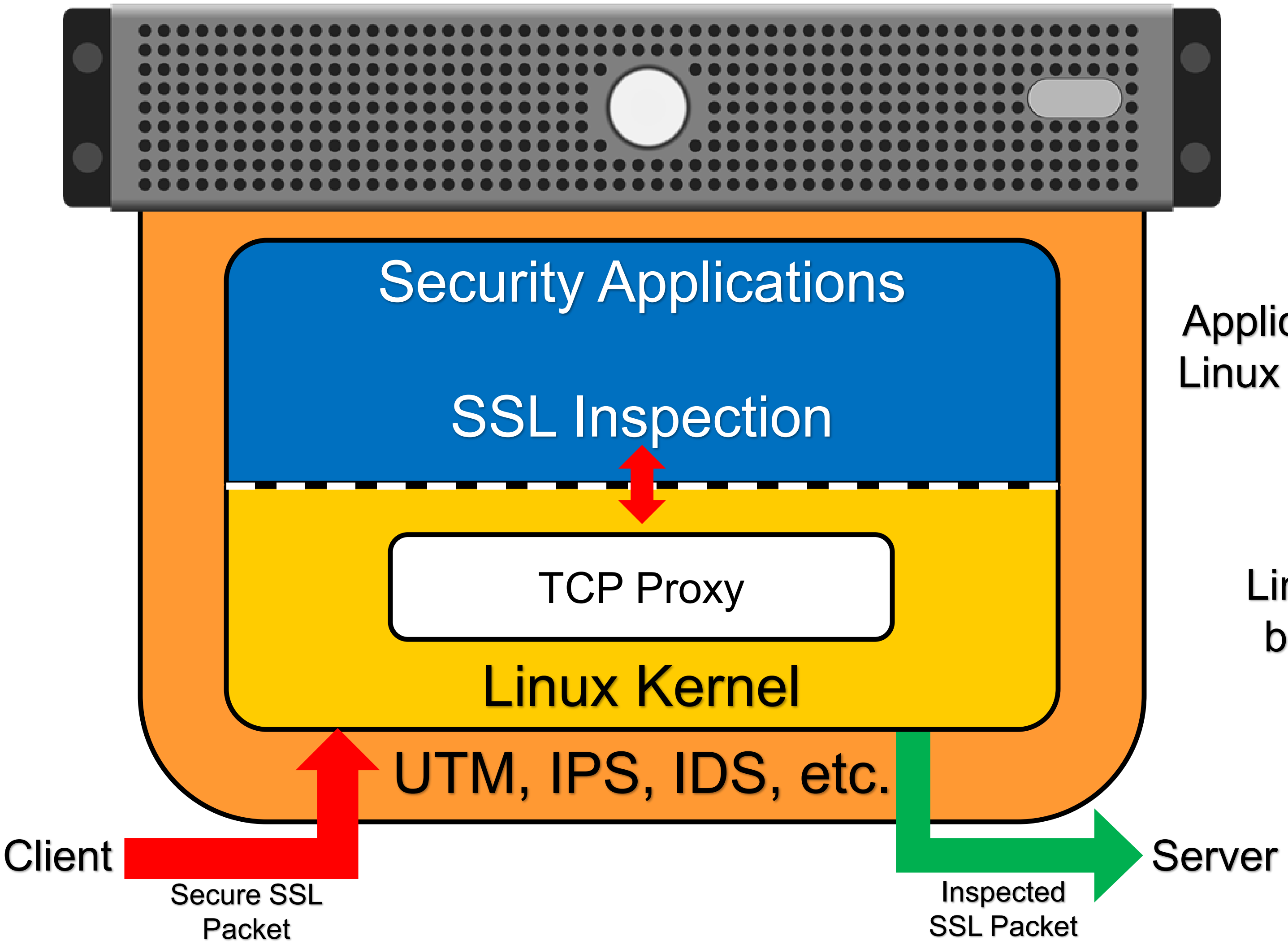
- **Performance of TCP applications is limited by Linux**

- **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

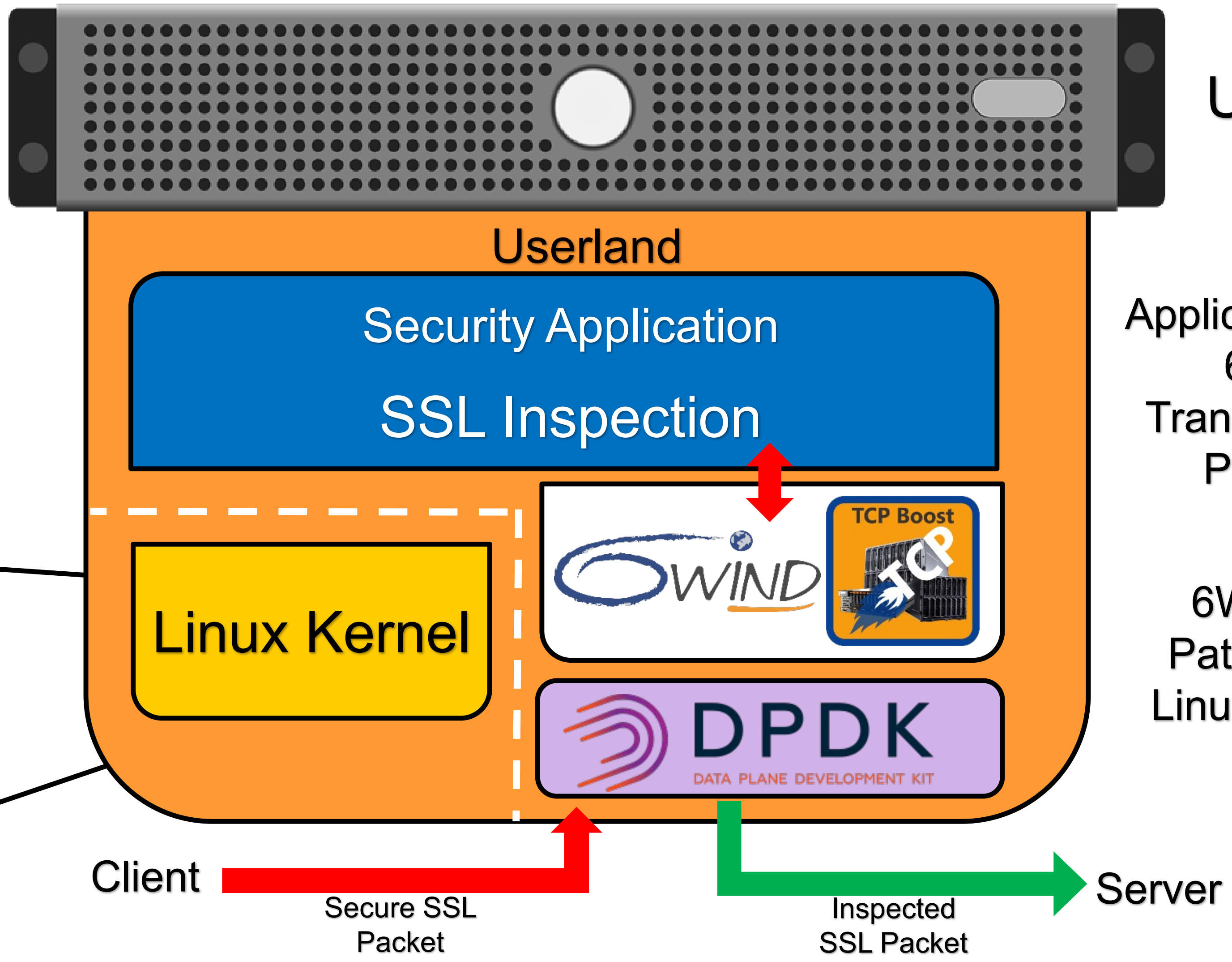


Application talks to Linux kernel socket layer

Linux kernel bottleneck

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**



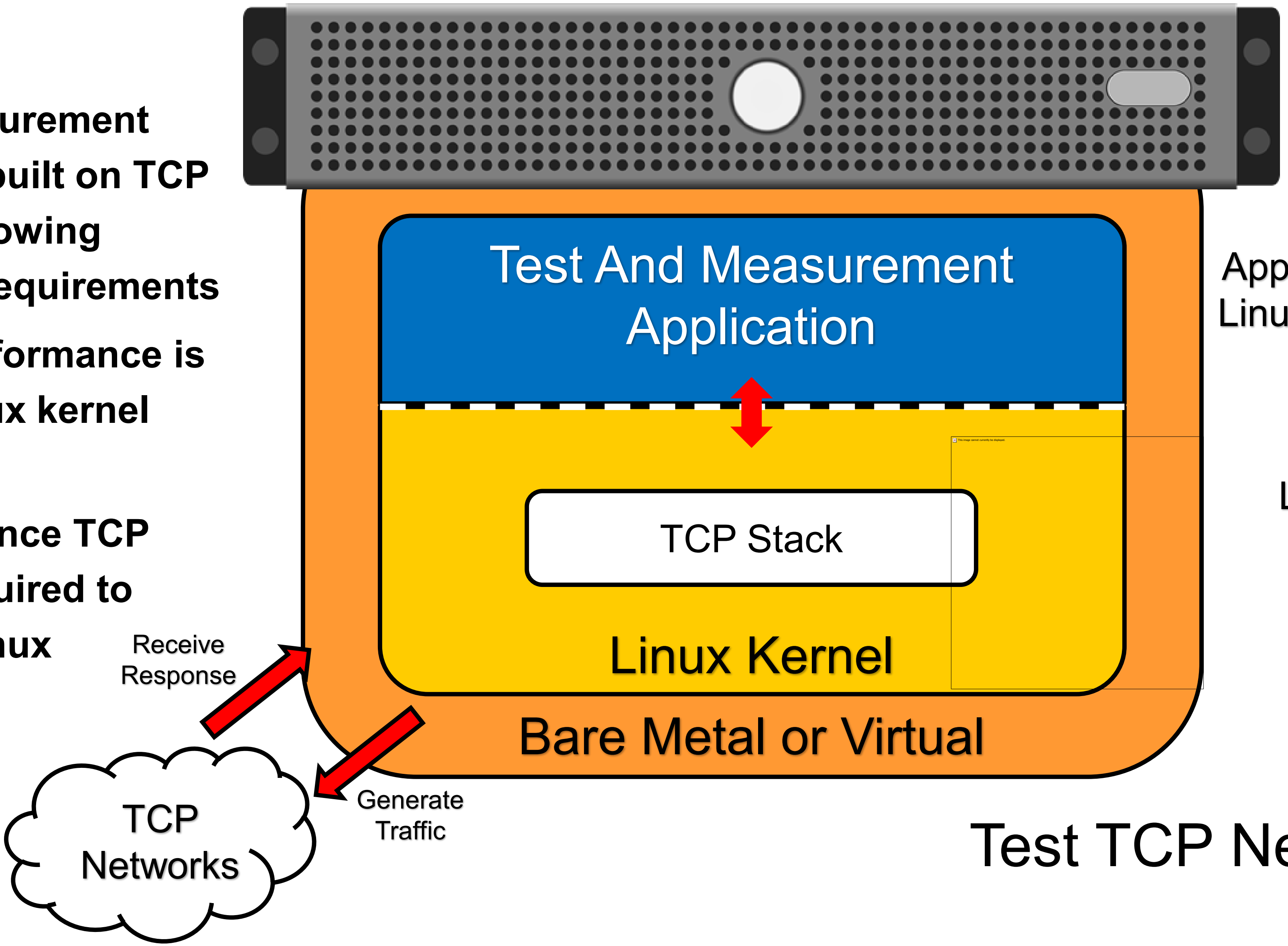
UTM, IPS, IDS, etc.

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



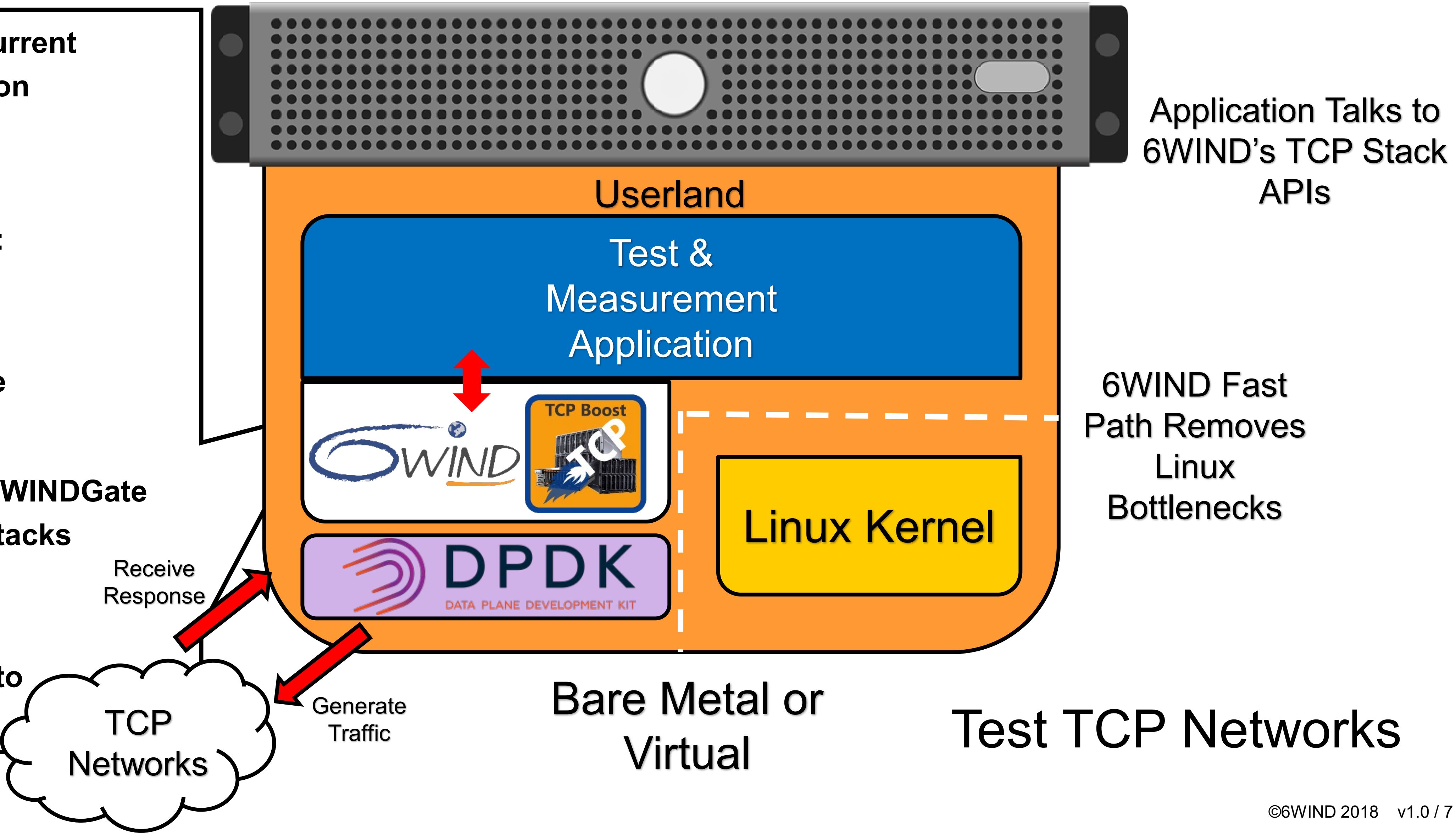
Application talks to Linux kernel socket layer

Linux kernel bottleneck

Test TCP Networks

6WINDGate High Performance TCP Stack For Test And Measurement Solutions

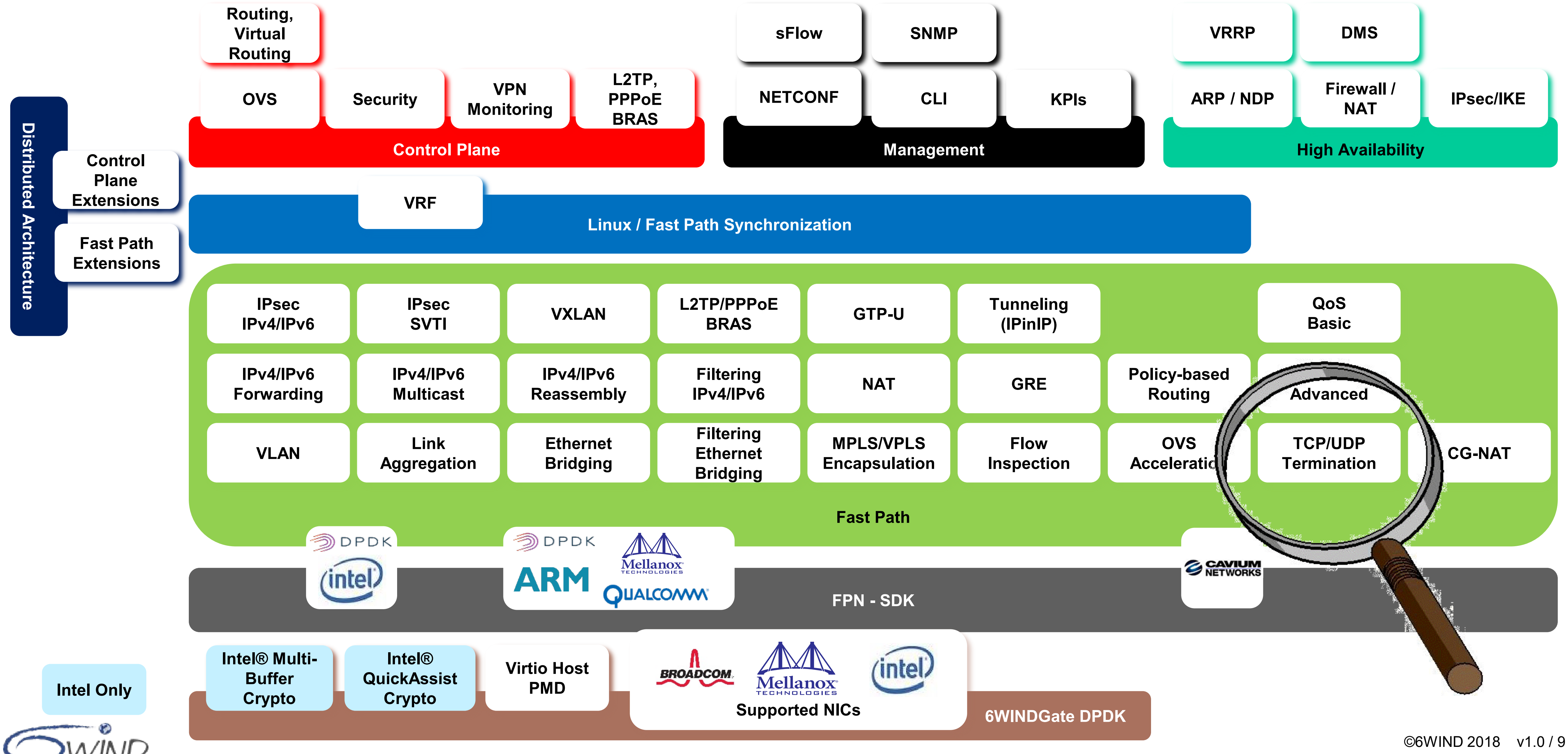
- Number of concurrent sessions: 6 million
- Connection rate: 1.4 million CPS
- Transaction rate: 7.1 million TPS
- Throughput: 12 Gbps per core
- Latency: 24 µs
- Integrated with 6WINDGate L2-L3 protocol stacks including Linux synchronization
- Extensible APIs to collect statistics



6WINDGate TCP - Product Presentation

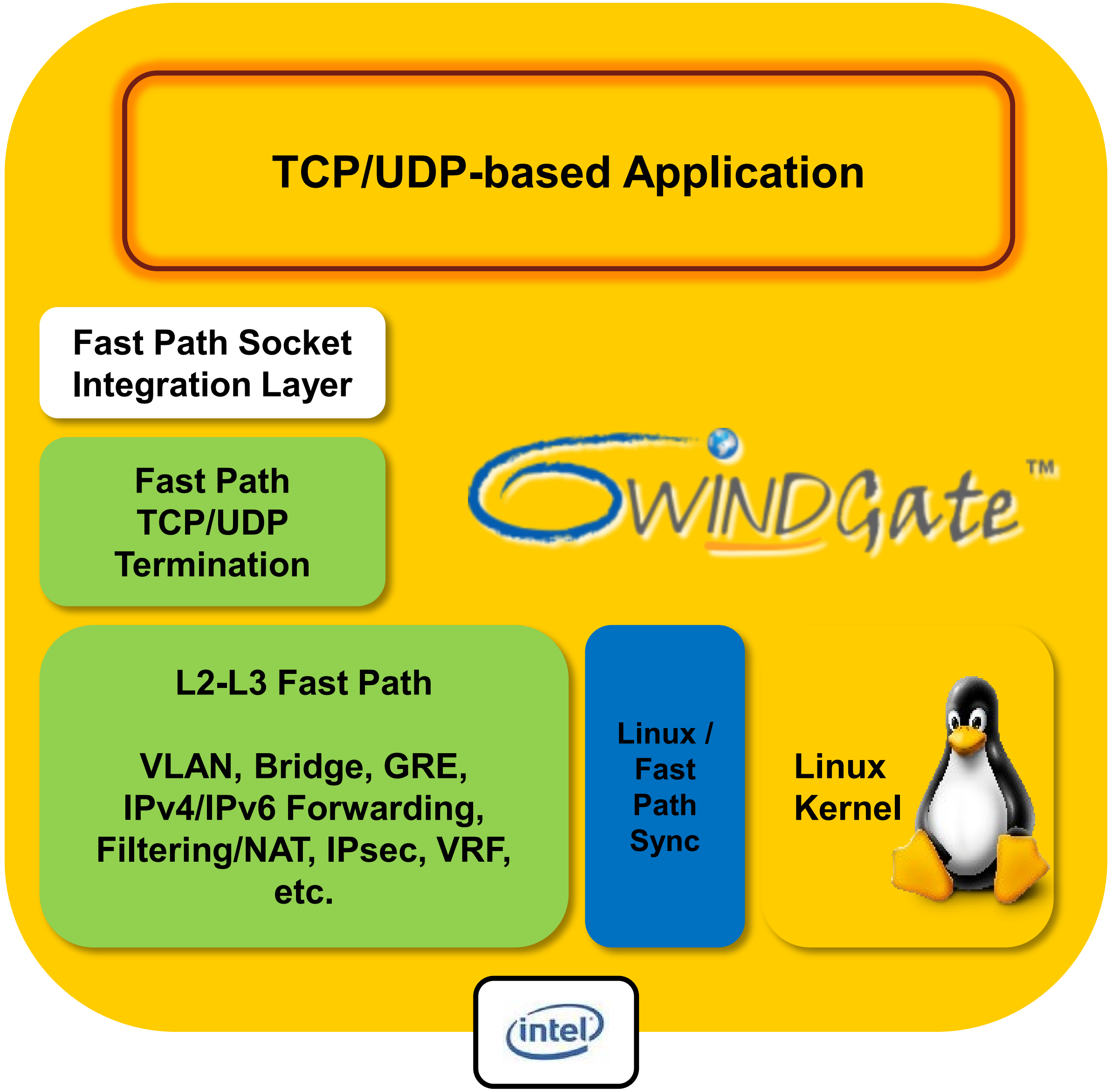
#SPEEDMATTERS For Serious Networks

6WINDGate Accelerated Layer 2-4 Networking Stacks

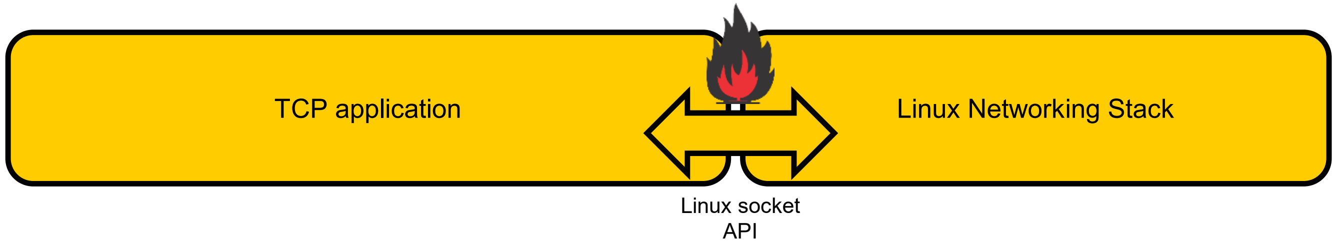


6WINDGate TCP/UDP Termination

- **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**

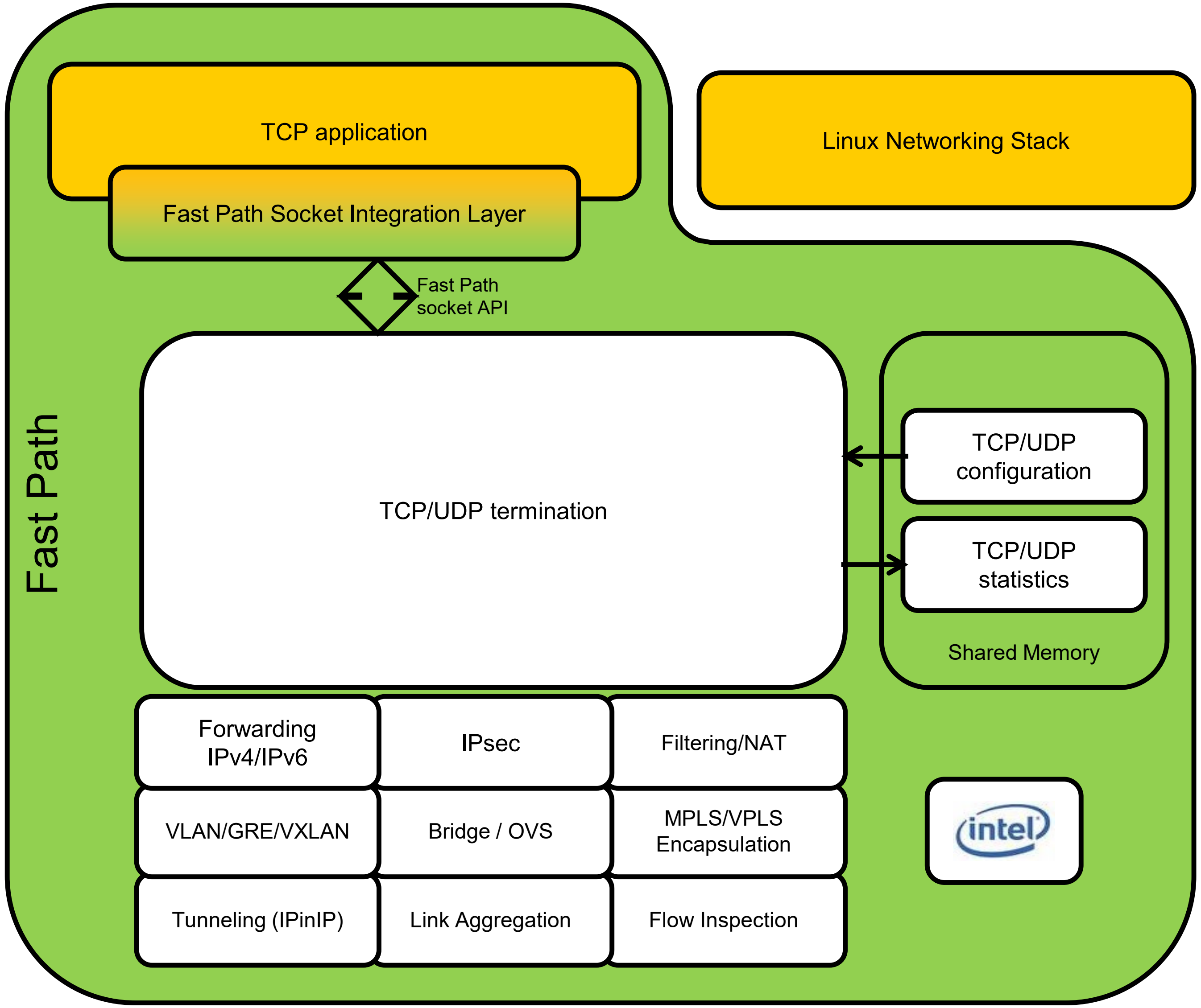


Architecture



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

Architecture



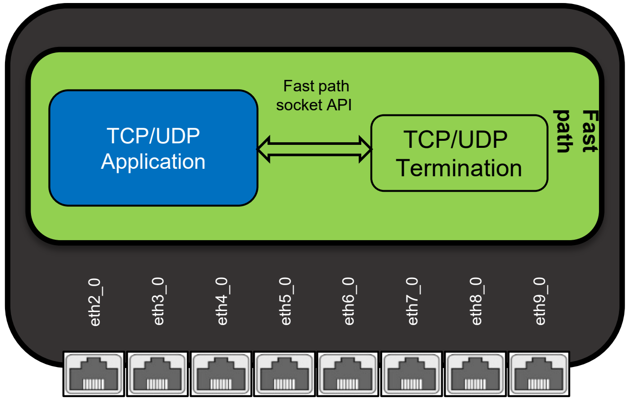
- **Fast Path TCP/UDP termination**
  - TCP/UDP protocols are processed in the Fast Path
  - Full featured TCP/UDP stack using BSD-like socket API
  - Timers are re-designed to get more scalability
  - Locks are removed
  - Memory footprint is reduced
- **Performance**
  - Scale: 8M active concurrent 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

©6WIND 2018

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

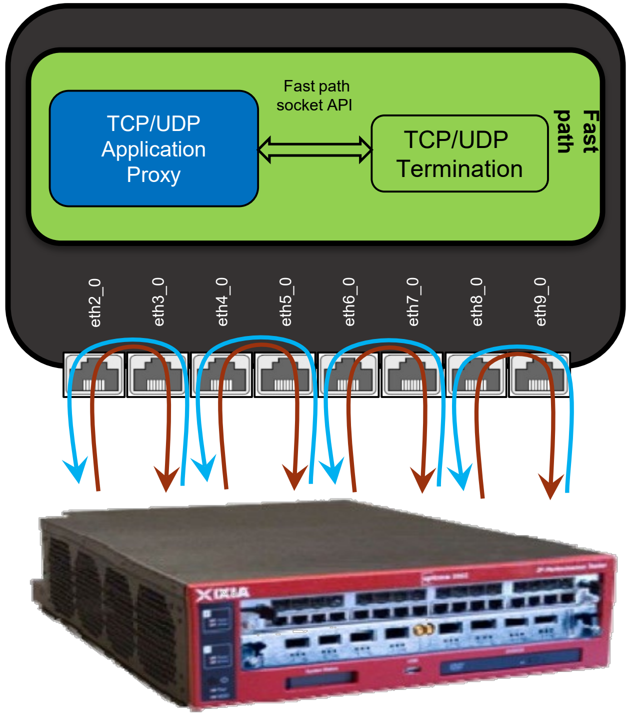


- **Benchmarks**
  - Proxy
    - CPS
    - Bandwidth
  - Server
    - CPS
    - TPS
    - Bandwidth
    - Latency

Proxy Benchmarks

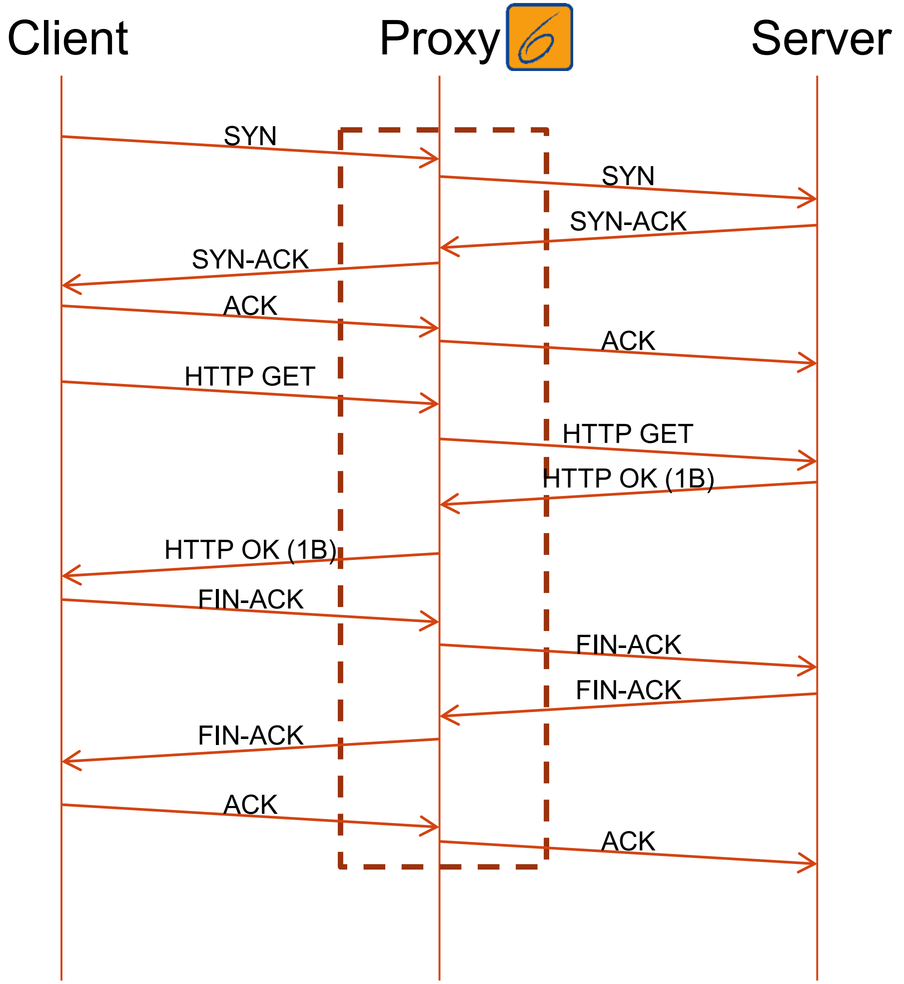
- **TCP connection rate**
  - Reported result is system TCP continuous socket open and close per second capacity
  - TCP sockets are opened first to reach socket objective (10K to 8M sockets)
  - Once objective is reached, sockets are continuously closed and opened at maximum rate
  - The connection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

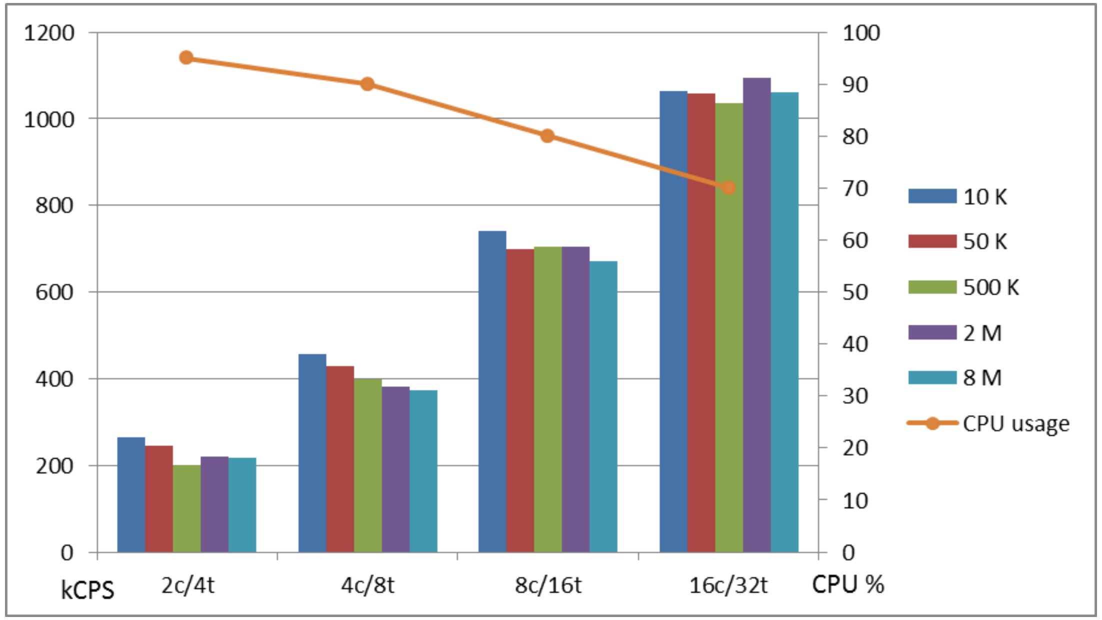
- **6WINDGate TCP proxy application running on node under test**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
  - One done, sockets are continuously opened and closed
- **IXIA measures maximum number of sockets per second**
  - The test is successful when all sockets are opened and closed correctly
- **A connection includes the following operations on each proxy**
  - Open a socket on client side
  - Open a socket on server side
  - Proxy a HTTP 1.1 Get request (one packet)
  - Proxy the HTTP 1.1 Response (one packet)
    - Page requested is 1 Byte
  - Close the socket on client side



one connection = 2 sockets

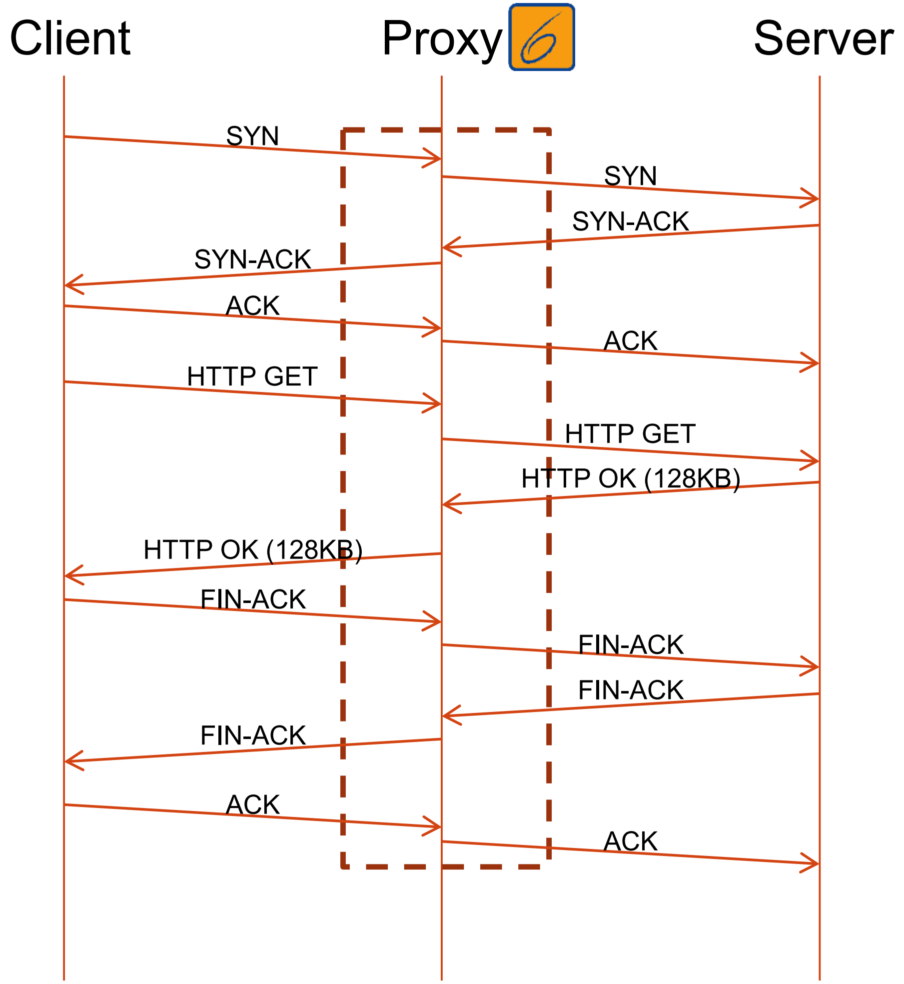
Proxy Connection Rate Results

- **Up to 1M sockets 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

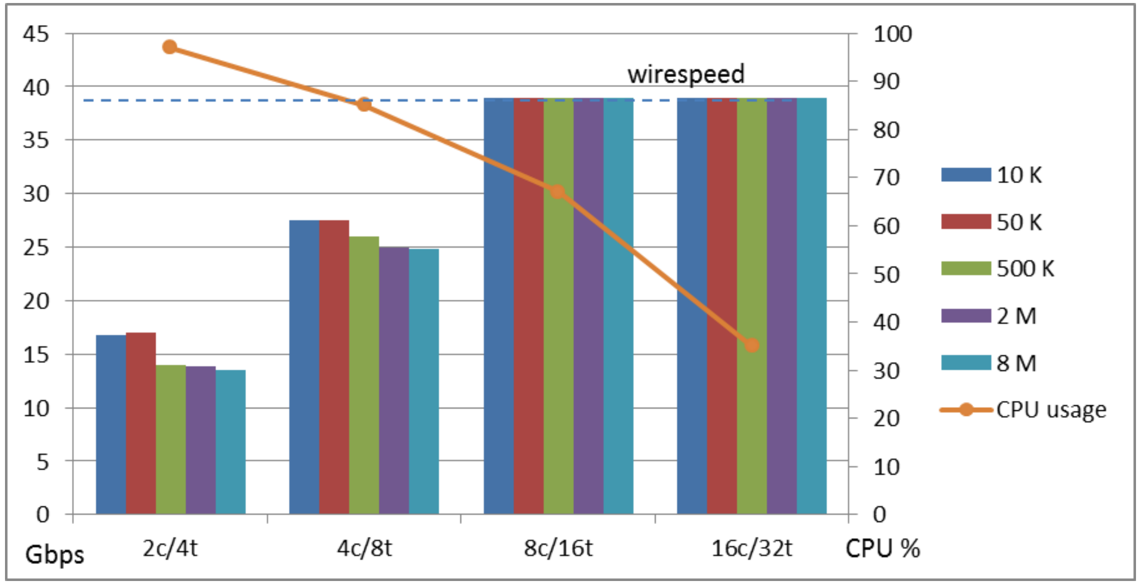
- **6WINDGate is running TCP proxy application**
  - Single port 80 is used (worst case)
- **IXIA establishes connections until concurrent socket objective is reached**
  - One done, sockets are continuously opened and closed
- **IXIA measures the bandwidth of the proxy**
  - This includes packet reception and emission
  - New connections are used to create load
  - The page size is 128 KB



one connection = 2 sockets

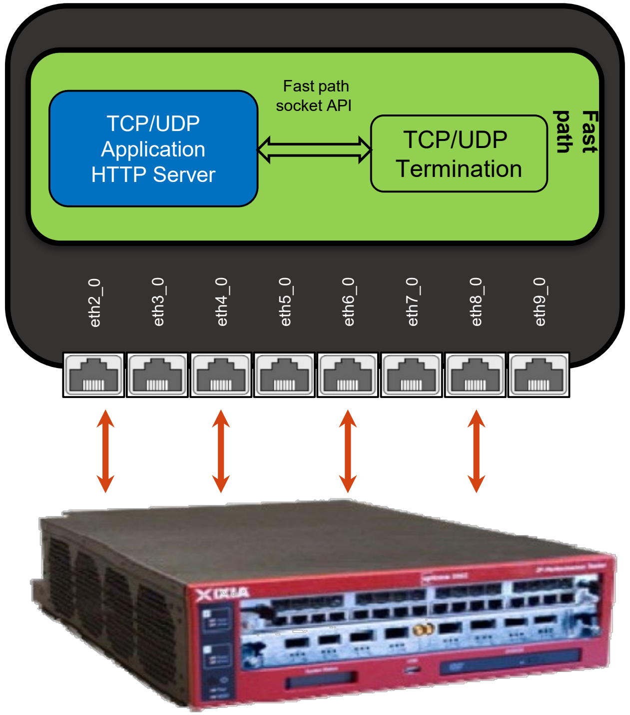
Proxy Bandwidth Results

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



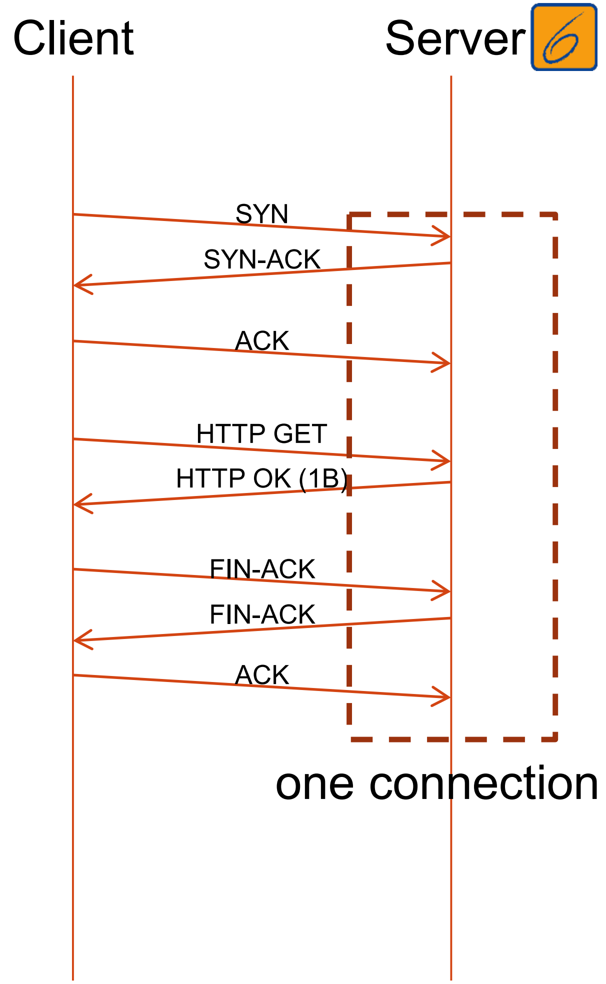
Server Benchmarks

- **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 request 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)



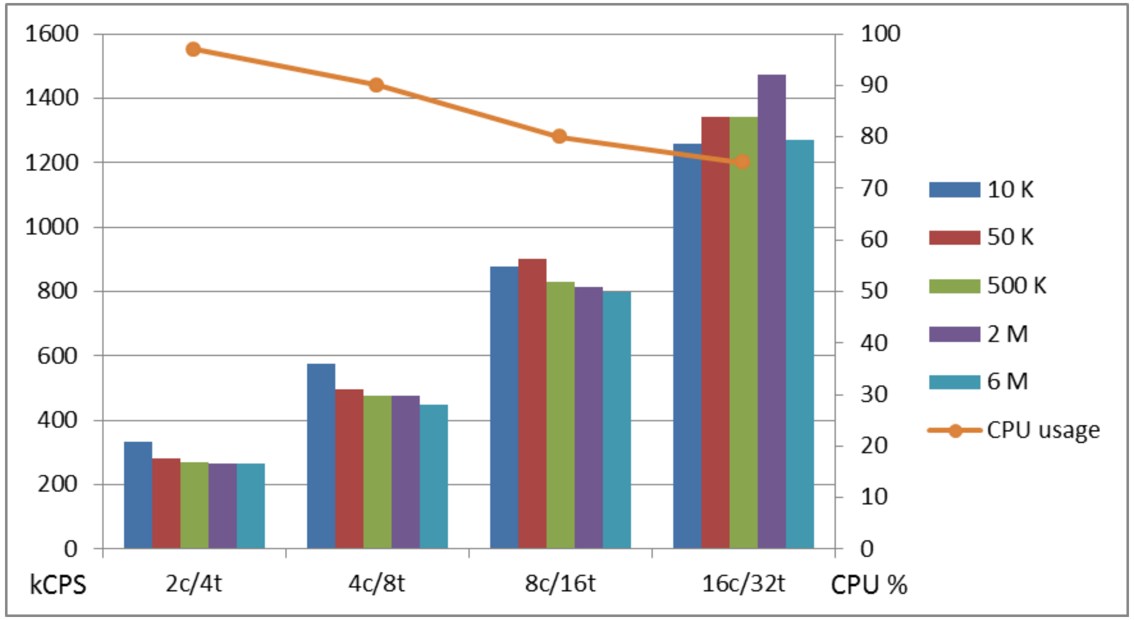
Server Connection Rate Test

- **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**
  - One done, sockets are continuously opened and closed
- **IXIA measures the maximum number of sockets per second**
  - The test is successful when all sockets are opened and closed correctly
- **A connection includes the following operations on the server**
  - Open a socket on client side
  - Process a HTTP 1.1 Get request (one packet)
    - Page requested is 1 Byte
  - Close the socket on client side



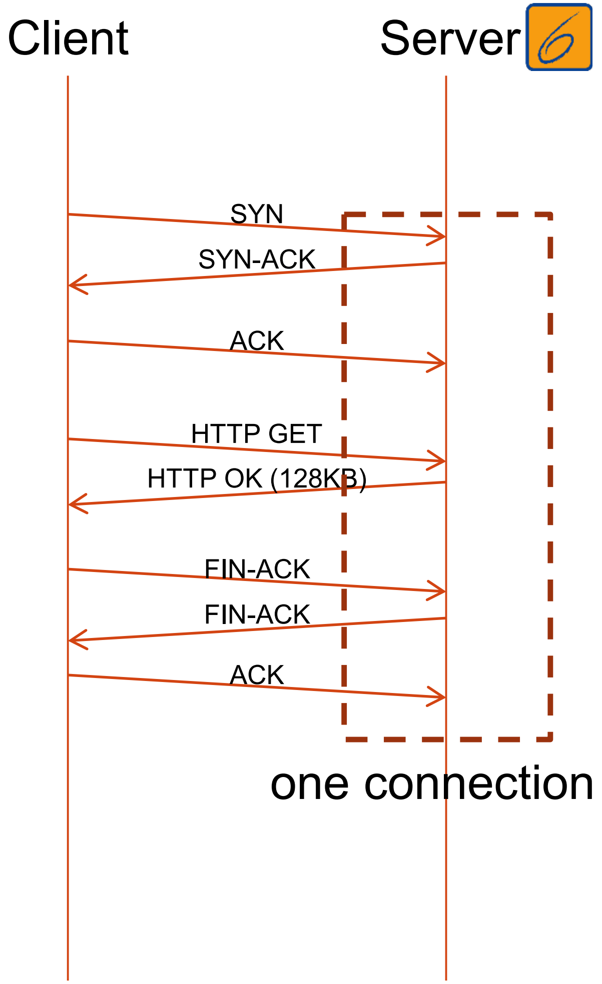
Server Connection Rate Results

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



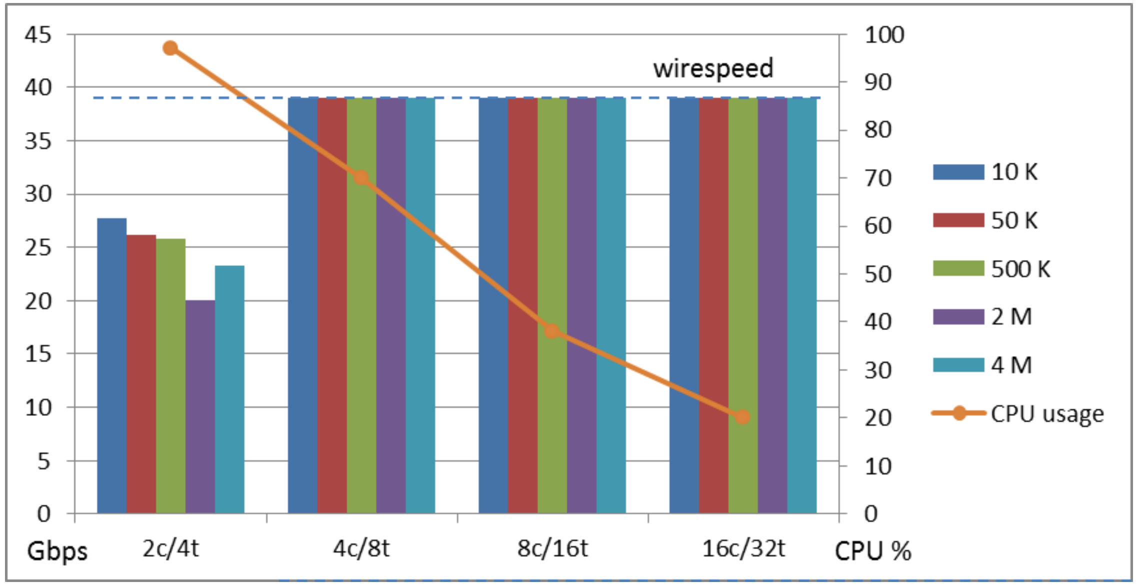
Server Bandwidth Test

- **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**
  - One done, sockets are continuously opened and closed
- **A connection includes the following operations on the server**
  - Open a socket on client side
  - Process a HTTP 1.1 Get request (one packet)
    - Page requested is 128 KByte
  - Close the socket on client side



Server Bandwidth Results

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



6WINDGate TCP – Implementation

#SPEEDMATTERS For Serious Networks

6WINDGate TCP/UDP Termination - Stack Design

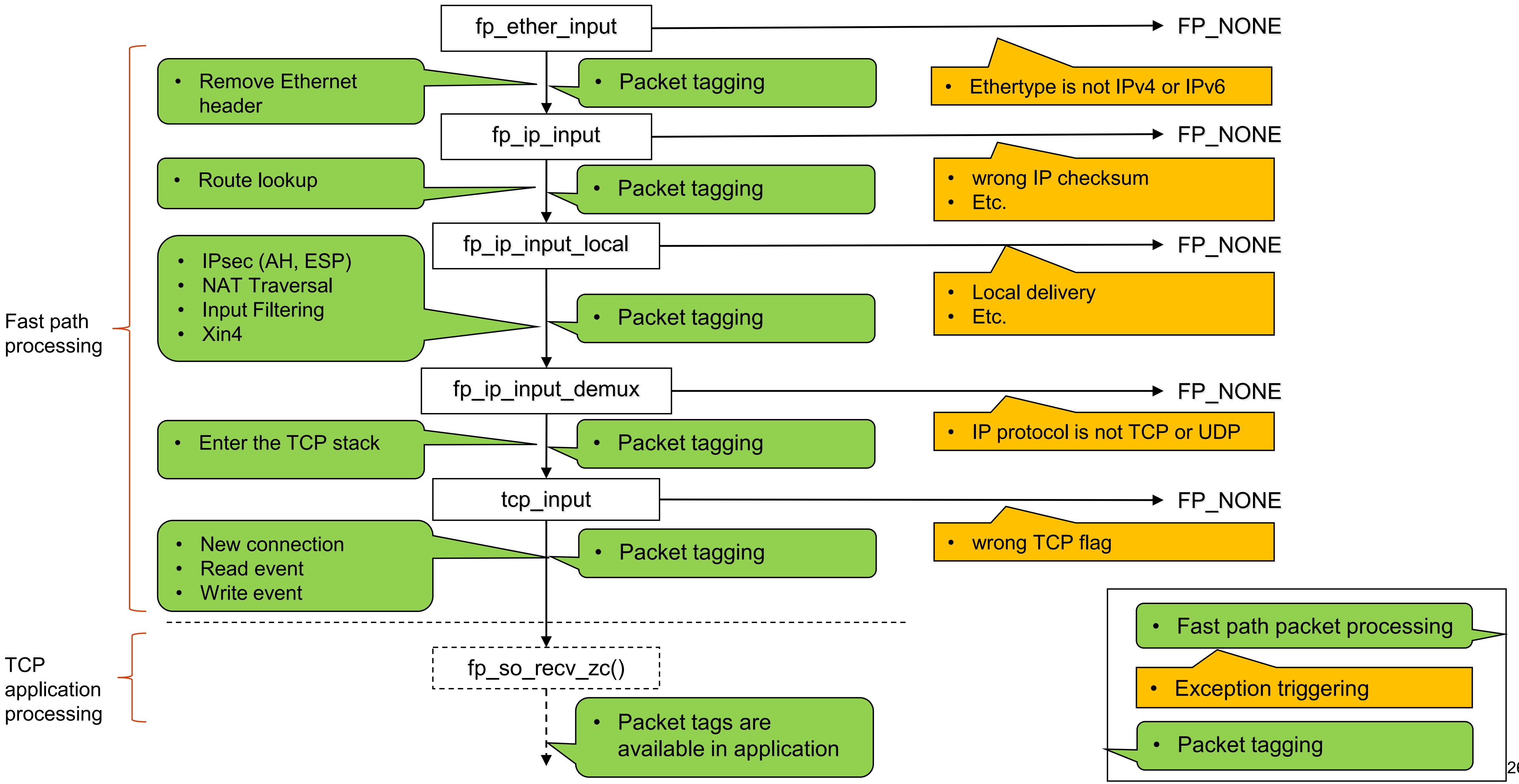
- **It is not a NetBSD port on fast path (like Rump)**
  - Would be reusing BSD lock based data structures
  - Suffers from design bottlenecks
    - accept() would be running on one core and would limit CPS performance

- **Not written from scratch**

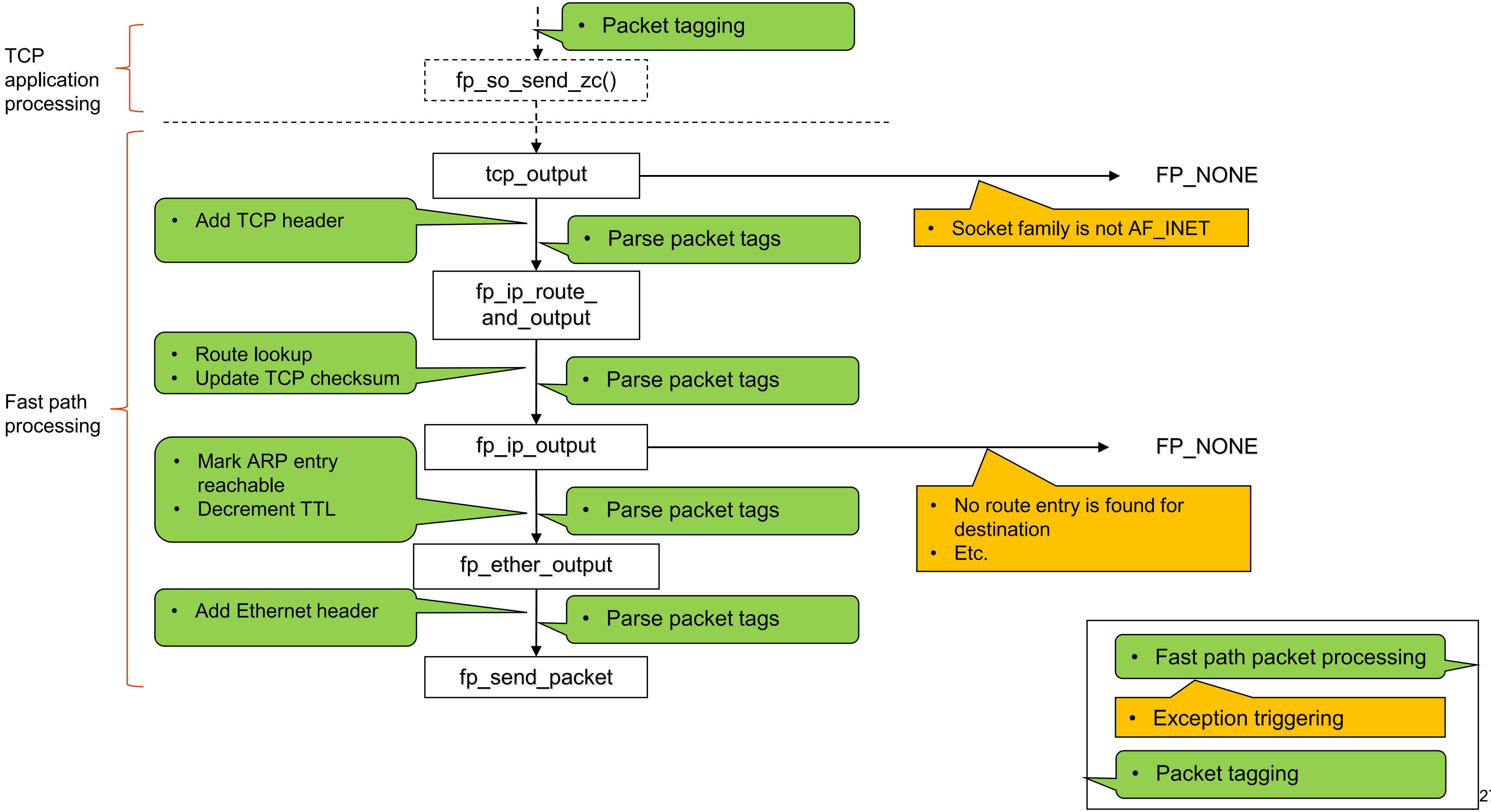
- **6WIND reused *BSD design flow but rewrote code to scale with multicore architectures to**
  - Get benefits from a widely deployed TCP stack
  - Minimize CPU cycles per packet
  - Remove blocking calls, it is event callback based
  - Remove BSD bottlenecks
    - accept() is running on multiple cores

©6WIND 2018

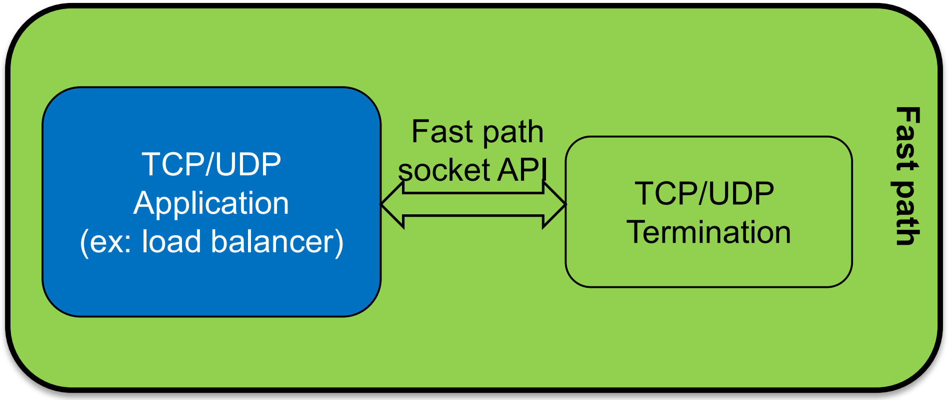
6WINDGate TCP/UDP Termination - Ingress



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

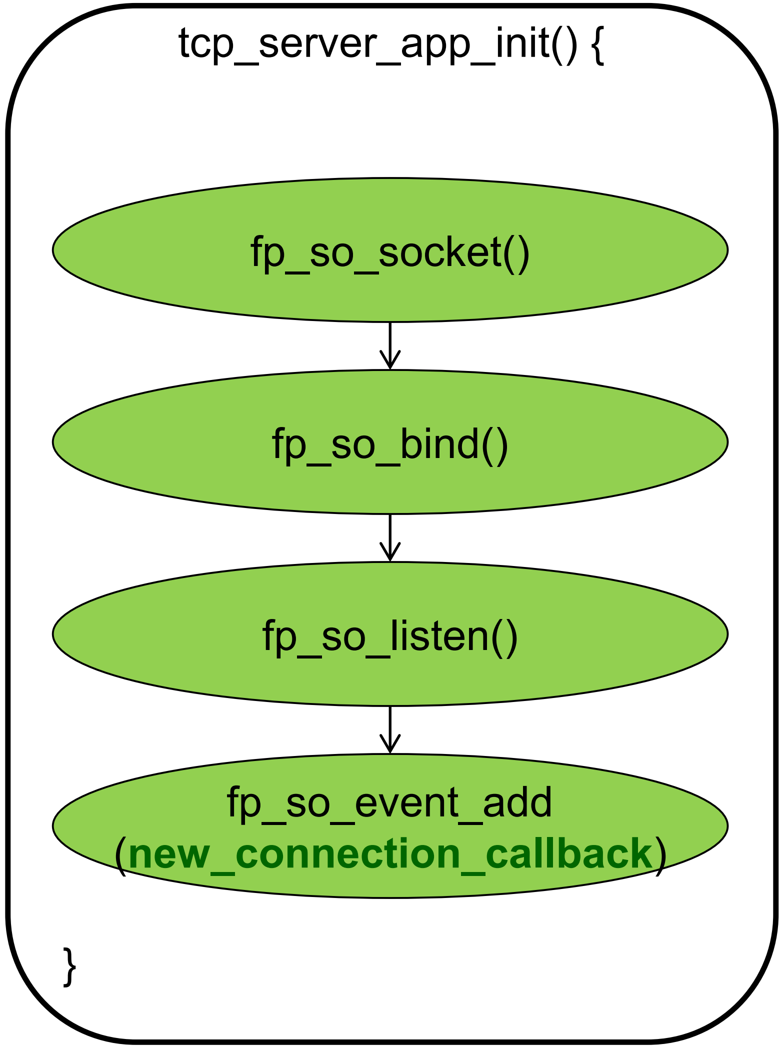
- **6WINDGate TCP/UDP termination provides copy and zero-copy APIs**

- **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.

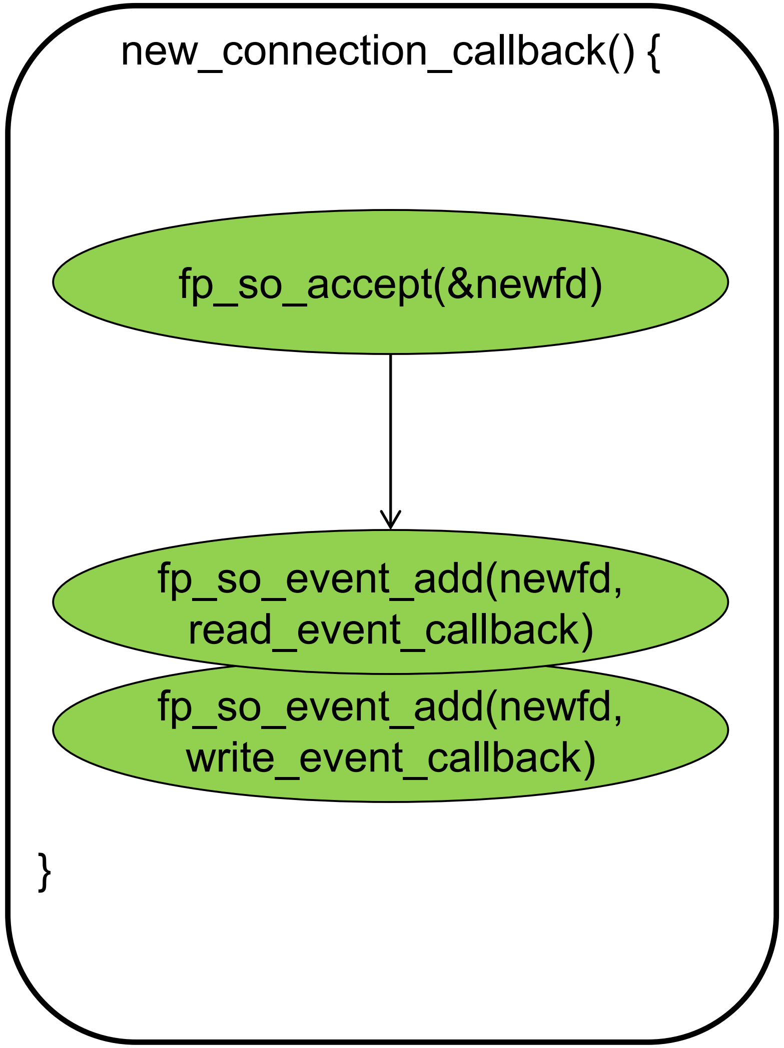
©6WIND 2018

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



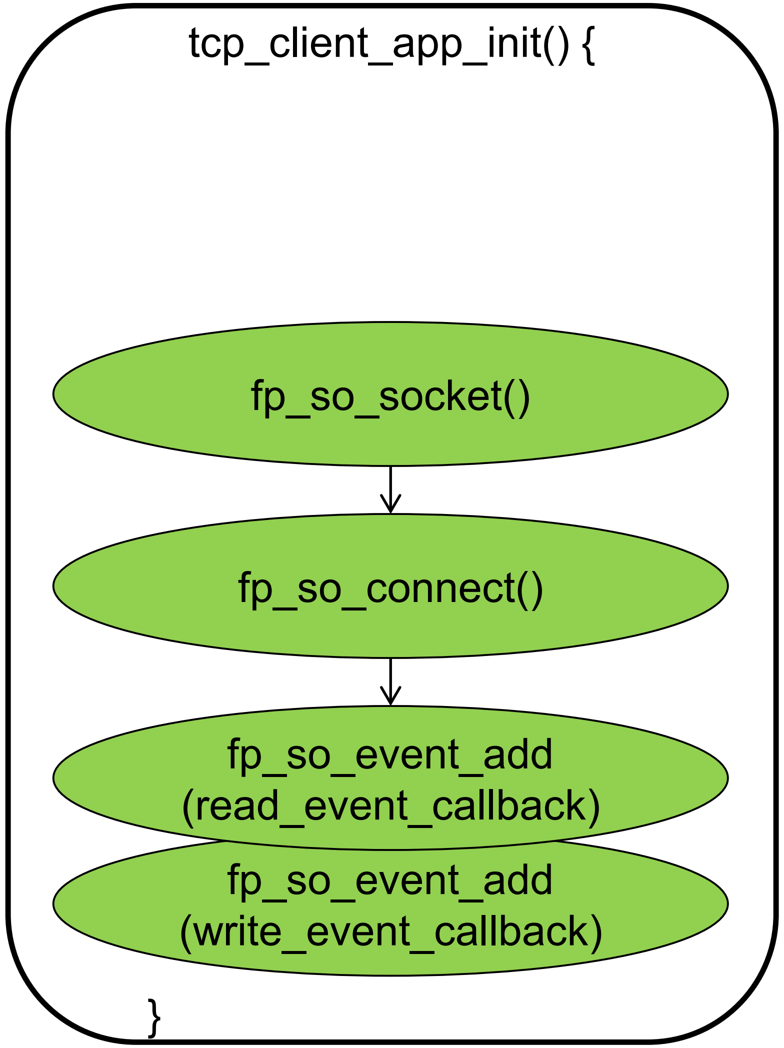
- **TCP application:**
  server socket initialization
- **Create fast path TCP socket**
- **Listen on a socket for a given IP address and port**
- **Register a callback function, called by the TCP termination module when receiving a new connection request**
  - Replace a blocking accept() call

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



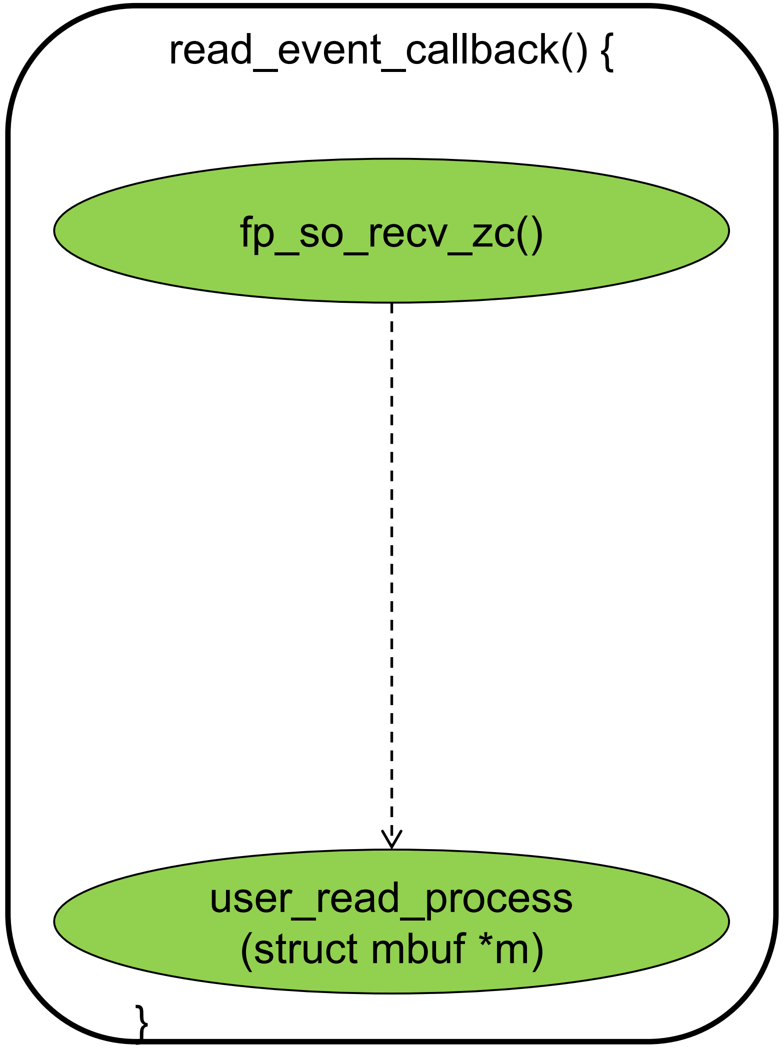
- **Called by the TCP termination module when a new connection occurs**
- **Called as soon as the syn/syn_ack/ack is done**
- **Accept connection**
- **Register callback functions for read and write events:**
  - Read event callback called when data is available to be read on the socket
  - Write event 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



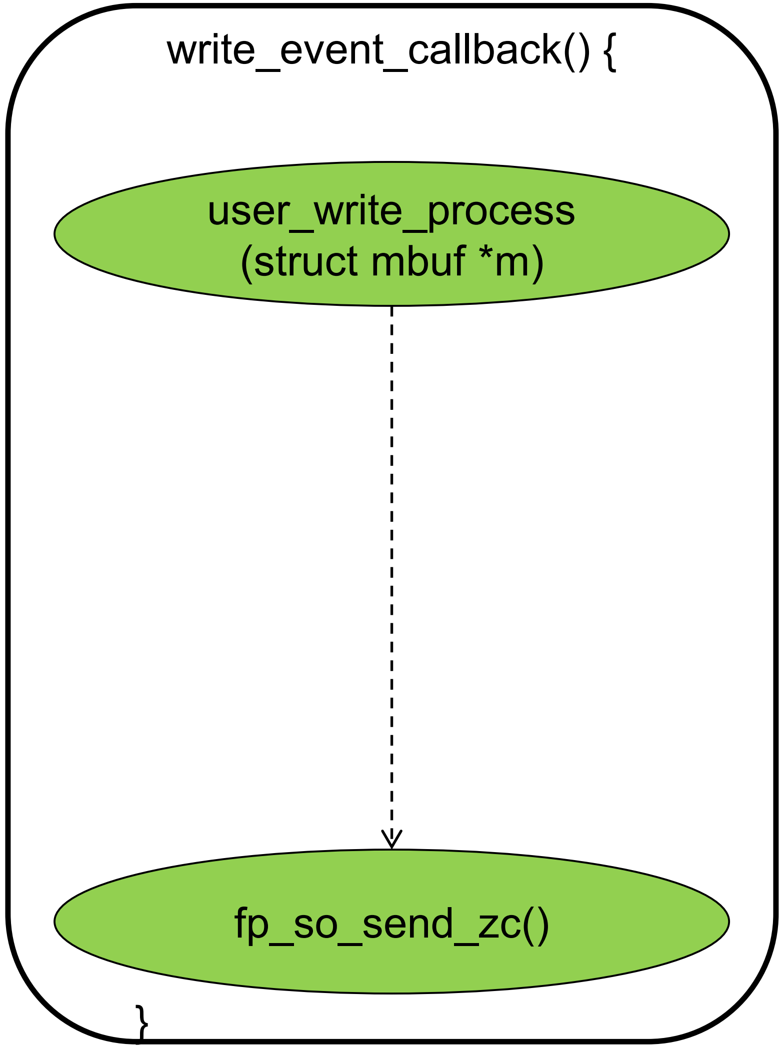
- **TCP application: client socket initialization**
- **Create fast path TCP socket**
- **Connect to a socket with a given IP address and port**
- **Register callback functions for read and write events**
  - Read event callback called when data is available to be read on the socket
  - Write event callback 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

6WIND.com

#SPEEDMATTERS For Serious Networks