

# Passively Monitoring Networks at Gigabit Speeds

Luca Deri <deri@ntop.org>

Yuri Francalacci <yuri@ntop.org>

**ntop.org**

# Presentation Overview

- Monitoring Issues at Wire Speed
- Traffic Filtering and Protocol Conversion
- Packet Capture and Classification
- Final Remarks

# Monitoring Issues at Wire Speed

- Monitoring low speed (100Mb) network is already available with common tools libpcap based
- Problem Statement: monitor high speed (10 GB and over) network with common PC's (64 bit 66MHz PCI bus)
- PCI Bus Limited Bandwidth (64 bit bus transfer limit 533 Mbit/s)

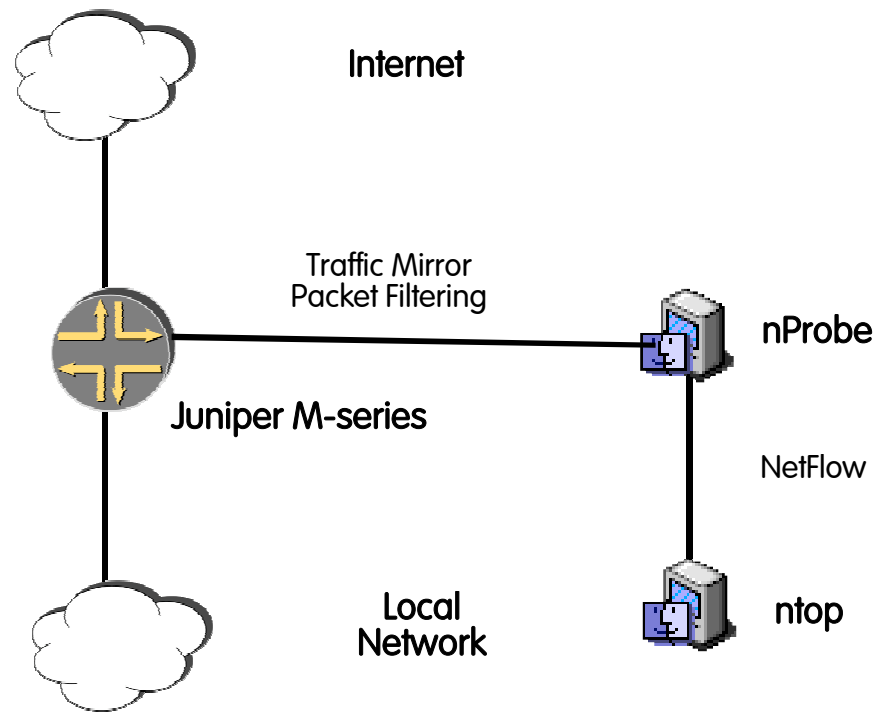
# Proposed Approach: Requirements

- Hardware and Software:
  - Intelligent routers (e.g. Juniper M-series): they are needed to run the network
  - x86-based PCs for capturing traffic
  - Linux/FreeBSD Operating System
  - Standard 64 bit PCI Gigabit NICs (Intel)

# Proposed Approach: Goals

- Passively monitor networks at Gbit speeds with no (or very little) packet loss
- Traffic information generated in a standard format (NetFlow/nFlow)
- Ability to monitor both IPv4/v6
- Provide accounting, performance information

# Architecture Overview



# Traffic Filtering and Protocol Conversion [1/3]

- Juniper routers provide:
  - a built-in traffic-filter (firewall configuration statement)
  - traffic mirroring (forwarding configuration statement)

# Traffic Filtering and Protocol Conversion [2/3]

- Traffic filter capabilities:
  - IPv4 and IPv6 filter types available
  - BPF-like filtering terms
  - Filter complexity as user request
- Traffic filter term counter
  - Possibility to define a counter for each term (could be used for accounting reason)
  - All counters could be read via SNMP

# Traffic Filtering and Protocol Conversion [3/3]

- Traffic mirroring advantages:
  - Interface type independency (router provides the protocol conversion)
  - Sampling capabilities (if link speed > monitoring NIC speed)
  - Multilink mirroring (on the monitoring link can be mirrored more than one line)

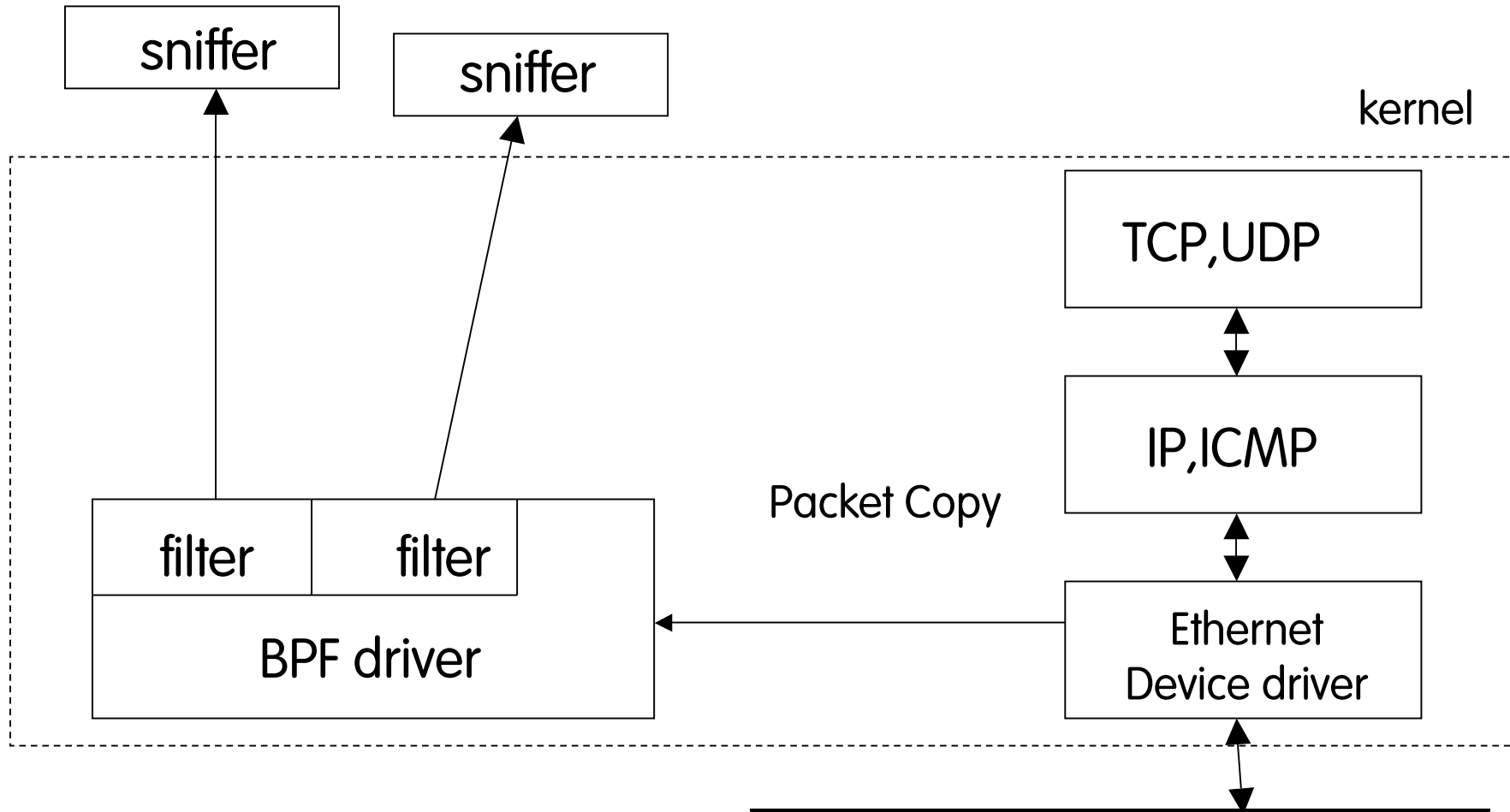
# Juniper Accounting

- NetFlow (v5/v8) support
- Flexible flow aggregation (AS, service, etc)
- Complex accounting (e.g. using ntop) using a PC connected on a mirror port

# Packet Capture and Classification: Issues

- Most Gbit network cards/OSs have not been designed for capturing thousand of packets per second in promiscuous mode
- Most NetFlow implementations (e.g. Juniper, Cisco, Extreme Networks) handle up to ~10k packet/sec and/or decrease dramatically switch performances
- Flow collector performance is often rather limited (load balancing)

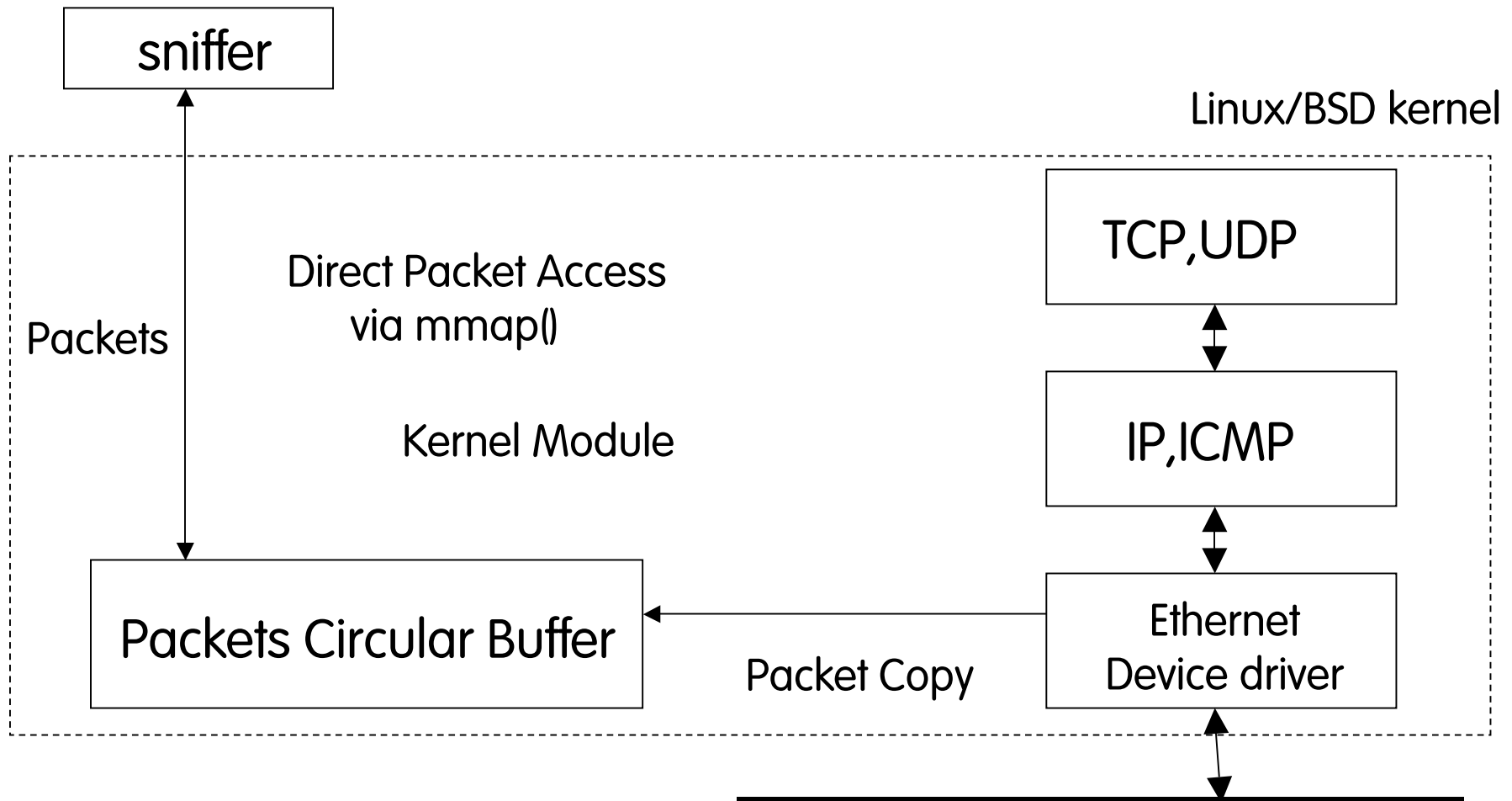
# Userland Packet Capture: libpcap



# Libpcap Limitations

- Multiple packet copies.
- Costly data exchange from kernel to user space via system calls
- Severe packet loss if userland applications cannot cope with packet/kernel speed

# Solution 1: Kernel Packet Capture



# Kernel Packet Capture: Code

```
packetBuffer = mmap(fd);  
while(1) {  
    if(select(fd)) { /* There's a Packet to read */  
        packet = packetBuffer[slotId];  
        /* Handle packet here */  
        slotId = (slotId + 1) % numSlots;  
    } /* select */  
} /* while */
```

# Kernel Packet Capture: Limitations [1/2]

- Little (~10%) performance improvement over pcap due to select() call (test performed on a 10/100 MBit/sec link).
- Possible workarounds:
  - Smart Select: as soon select() returns 1, keep on reading. When there's nothing to read call select() again.
  - Active polling: infinite loop until there's something to read on packetBuffer[slotId]

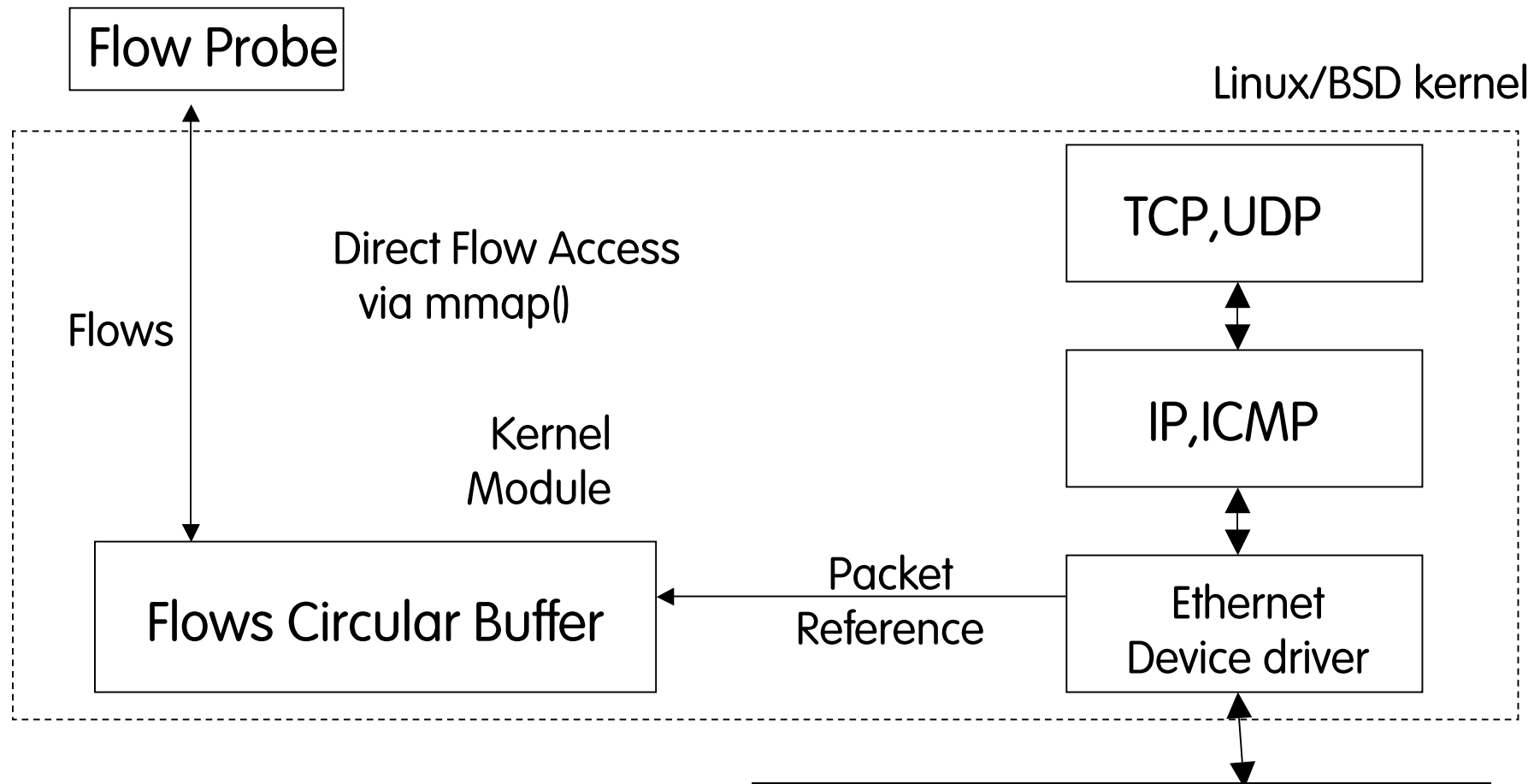
# Kernel Packet Capture: Limitations [2/2]

- Both workarounds do not improve performance significantly.
  - Smart Select: some select() calls are avoided.
  - Active polling: user time vs. kernel time increases significantly. At very high speeds (probability that there's something to read is high) it's better than smart select (see L. Rizzo).
- Drawback: user time increases causing packet loss.

# Solution 2: Kernel Packet Classification

- Principles:
  - Handle packets only inside the kernel (i.e. they are not passed to userland applications).
  - Pass flows, not packets, (flows << packets) to userland applications.

# Kernel Packet Classification: Architecture



# Kernel Packet Classification: Features

- Strong performance improvement over pcap due to full in-kernel packet processing.
- No NIC (DMA)->kernel->userland packet copy
- No packet loss
- Speed limited by the CPU speed (ability to handle interrupts)
- Simple userland NetFlow probe implementation

# nFlow (<http://www.nflow.org>)

- New flow definition based on NetFlow
- Major features:
  - Support for both IPv4 and IPv6
  - Added VLAN tagging/MPLS label support
  - Added (network and application) performance and (passive) fingerprinting information
  - Flow compression (gzip), non ripudiation (MD5)

# Final Remarks

- Packet filtering and protocol conversion in hardware (Juniper).
- External accounting application based on a PC with in-kernel NetFlow flow generation.
- Kernel-based nProbe (alpha-code) runs at kernel/interrupt speed (pcap-based version handles  $\leq 250k$  pkt/sec)