Red Hat
Enterprise Linux

Idea of “networkingfy” Linux tracing

Or “how Linux Tracing can use net/ subsystem”

Alexander Aring

Red Hat

Why this talk idea?

- -ENOCODE yet - I am sorry!
- It’s just an idea - Moonshot
- Any kind of feedback is welcome!
- Helps me: “Can this work? Good idea?”
- Is it too “crazy”?

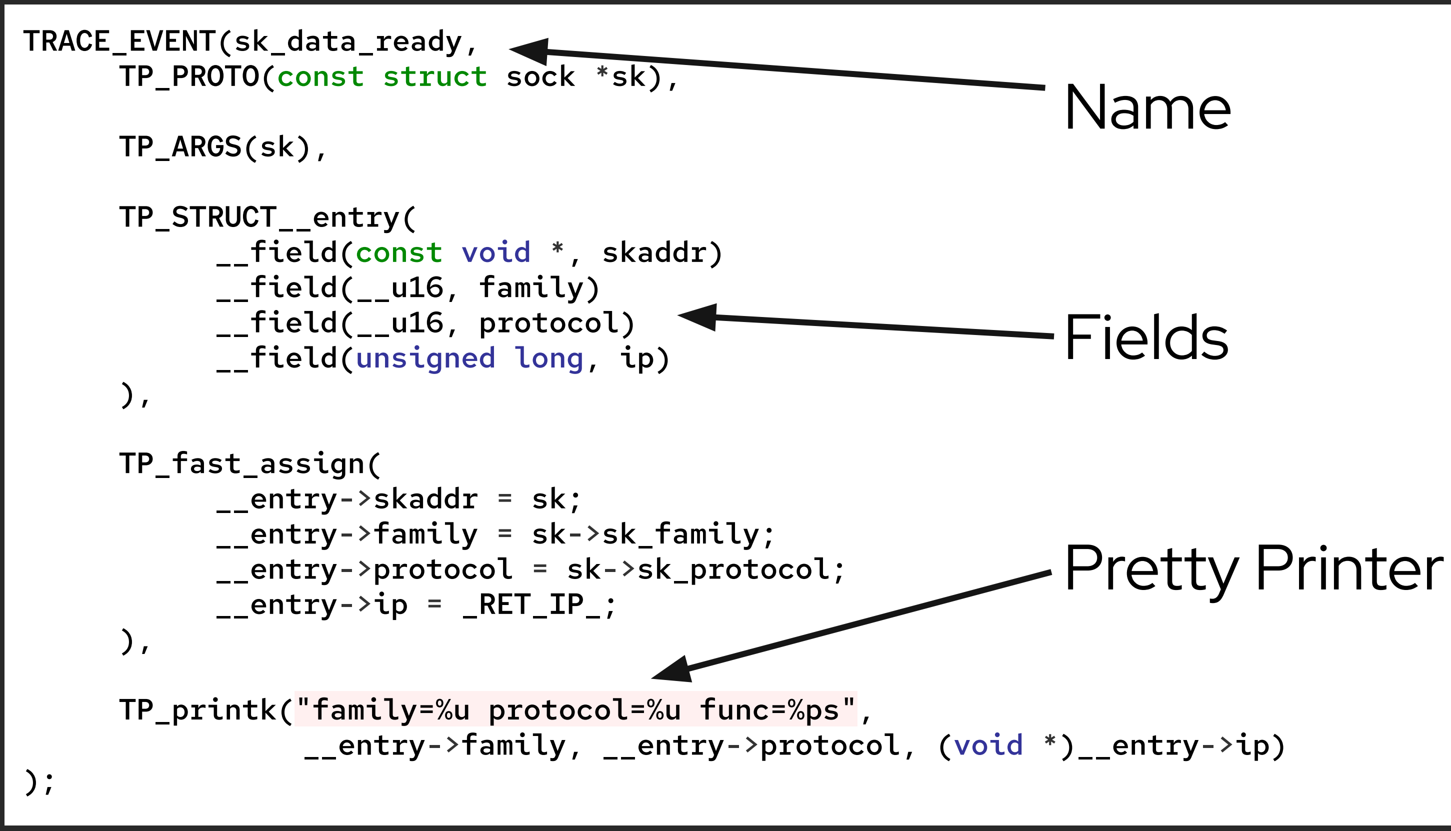
Tracing Basics

- Components
- Buffer Handling
- Common Filtering mechanism
- In reality everything is more complex

Tracefs

- Special Filesystem
- UAPI to control Linux Tracing subsystem
- Low-Level controllable via coreutils
- Usually /sys/kernel/tracing

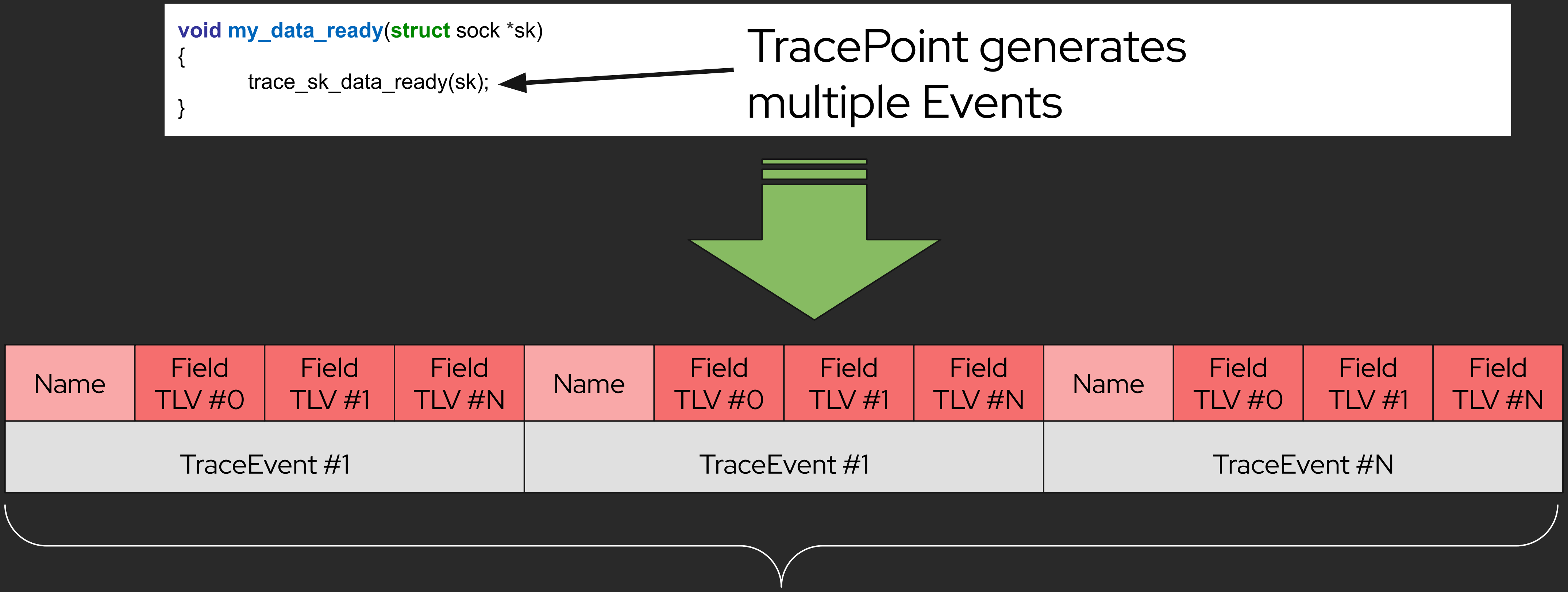
TracePoint Declaration



```
TRACE_EVENT(sk_data_ready,
        TP_PROTO(const struct sock *sk),

        TP_ARGS(sk),

        TP_STRUCT__entry(
                __field(const void *, skaddr)
                __field(__u16, family)
                __field(__u16, protocol)
                __field(unsigned long, ip)
        ),

        TP_fast_assign(
                __entry->skaddr = sk;
                __entry->family = sk->sk_family;
                __entry->protocol = sk->sk_protocol;
                __entry->ip = _RET_IP_;
        ),

        TP_printk("family=%u protocol=%u func=%ps",
                        __entry->family, __entry->protocol, (void *)__entry->ip)
);
```

Source:
Linux Kernel "include/trace/events/sock.h"

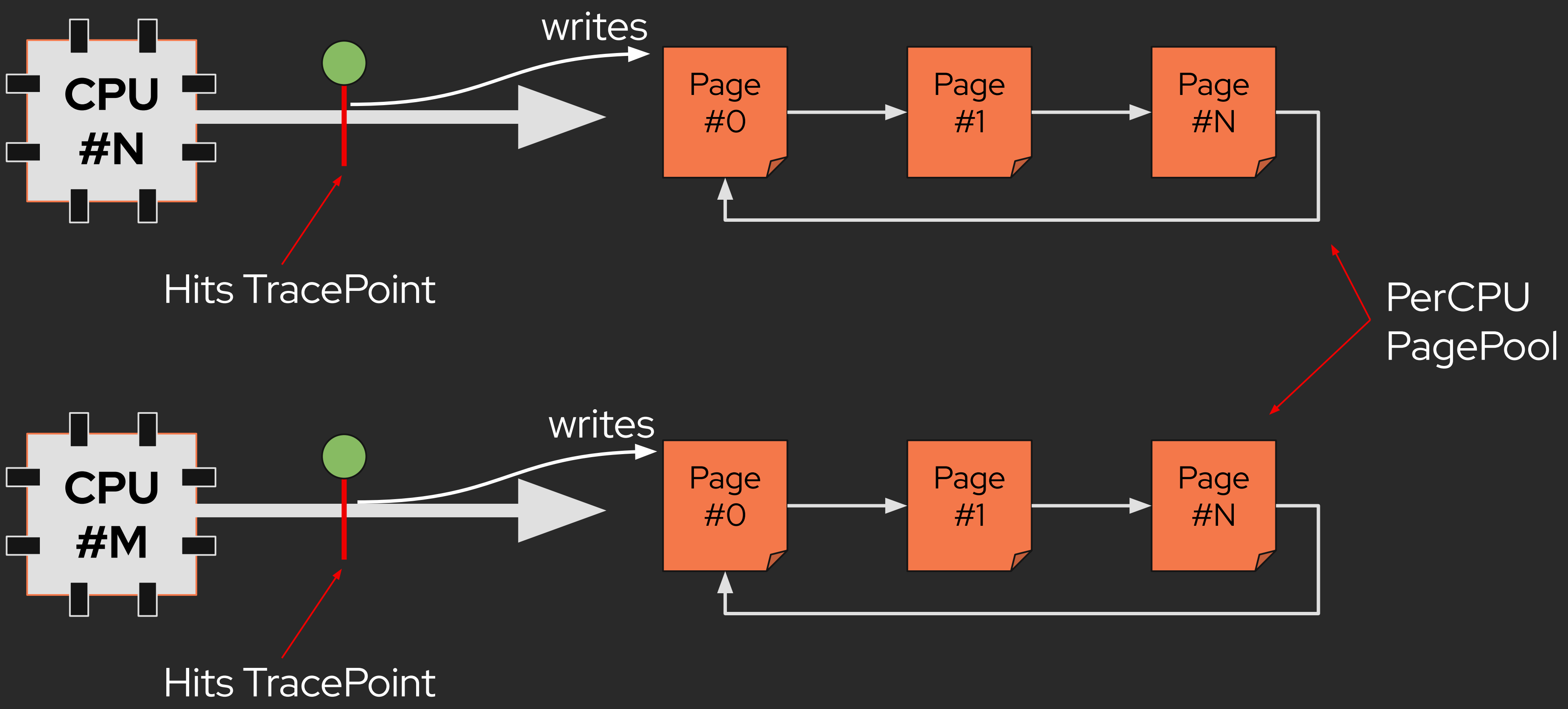
TraceEvent



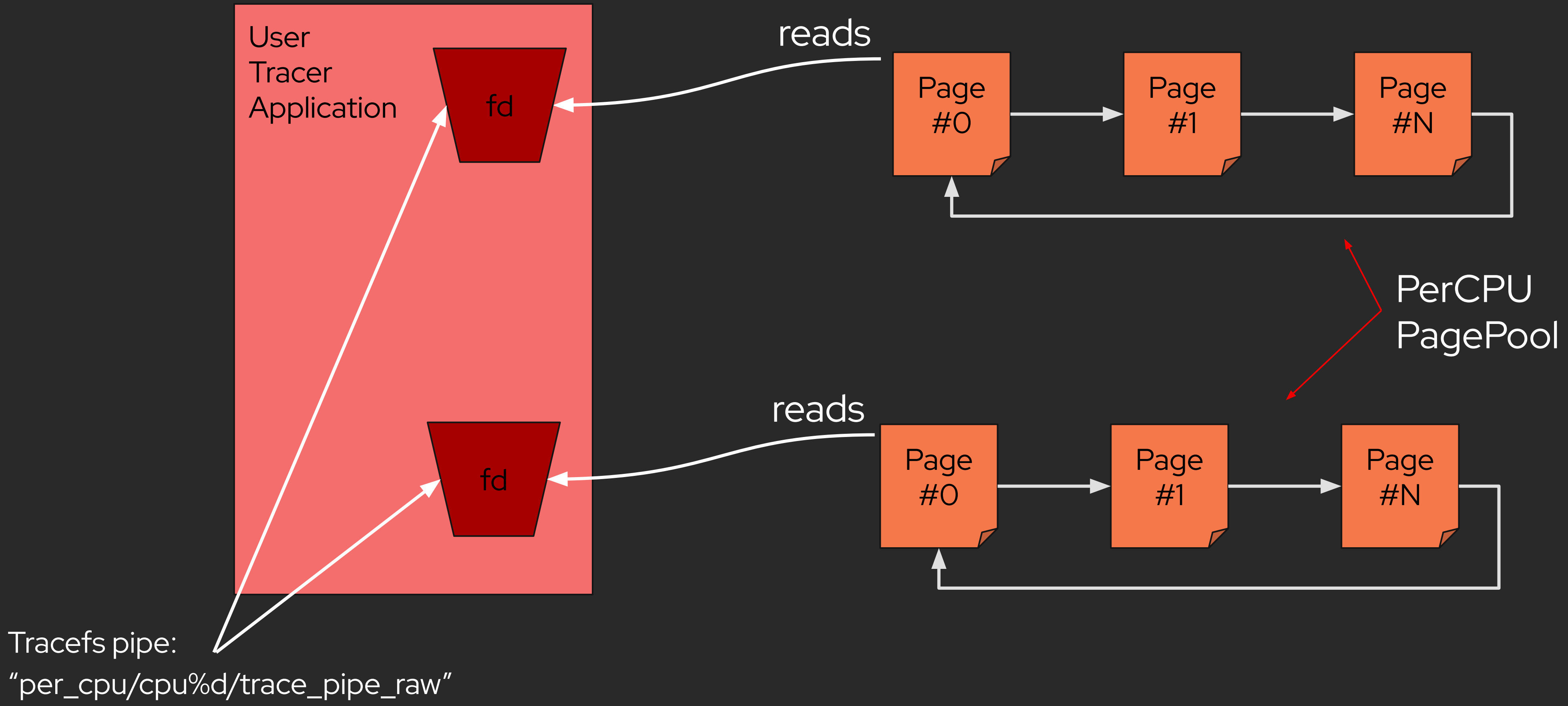
```
void my_data_ready(struct sock *sk)
{
        trace_sk_data_ready(sk);
}
```

Simplified: “Stream of Events”

TraceBuffer – PagePool Producer



TraceBuffer - PagePool Consumer



Red Hat

In-Kernel Event Filtering

Common solution

In-Kernel Interpreter (yes there is one!)

“kernel/trace/trace_events_filter.c”

Mostly work on TLVs (numbers, strings)

Kernel Tracer vs User Tracer

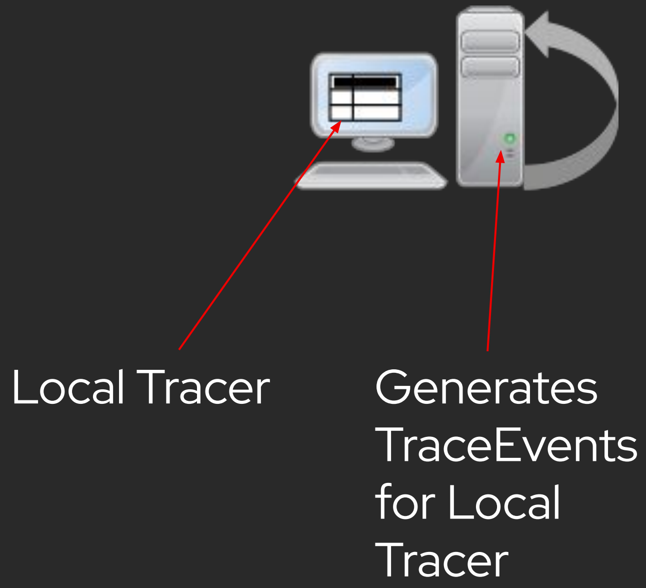
- Kernel Tracer
  - Lives in the Kernel
  - E.g. ftrace “kernel/trace/trace_functions.c”
- User Tracer
  - Lives in the Userspace “trace-pipe-raw”
  - Kernelshark 1), trace-cmd 2)

1) https://kernelshark.org/
2) https://trace-cmd.org/

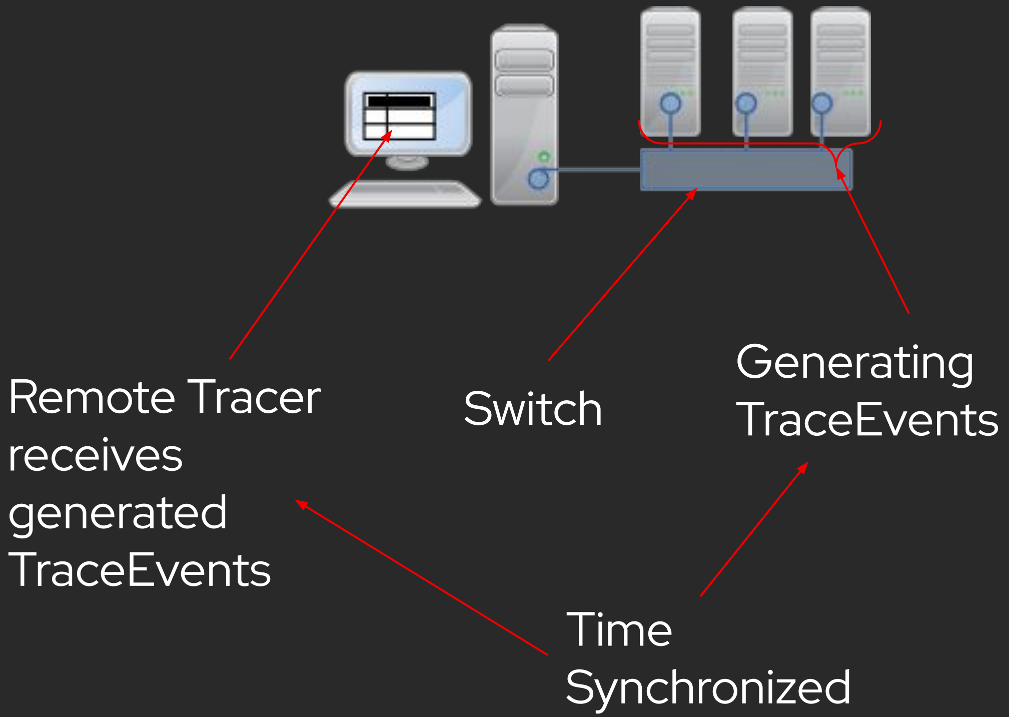
Red Hat

Local vs Remote Tracing

Local



Remote



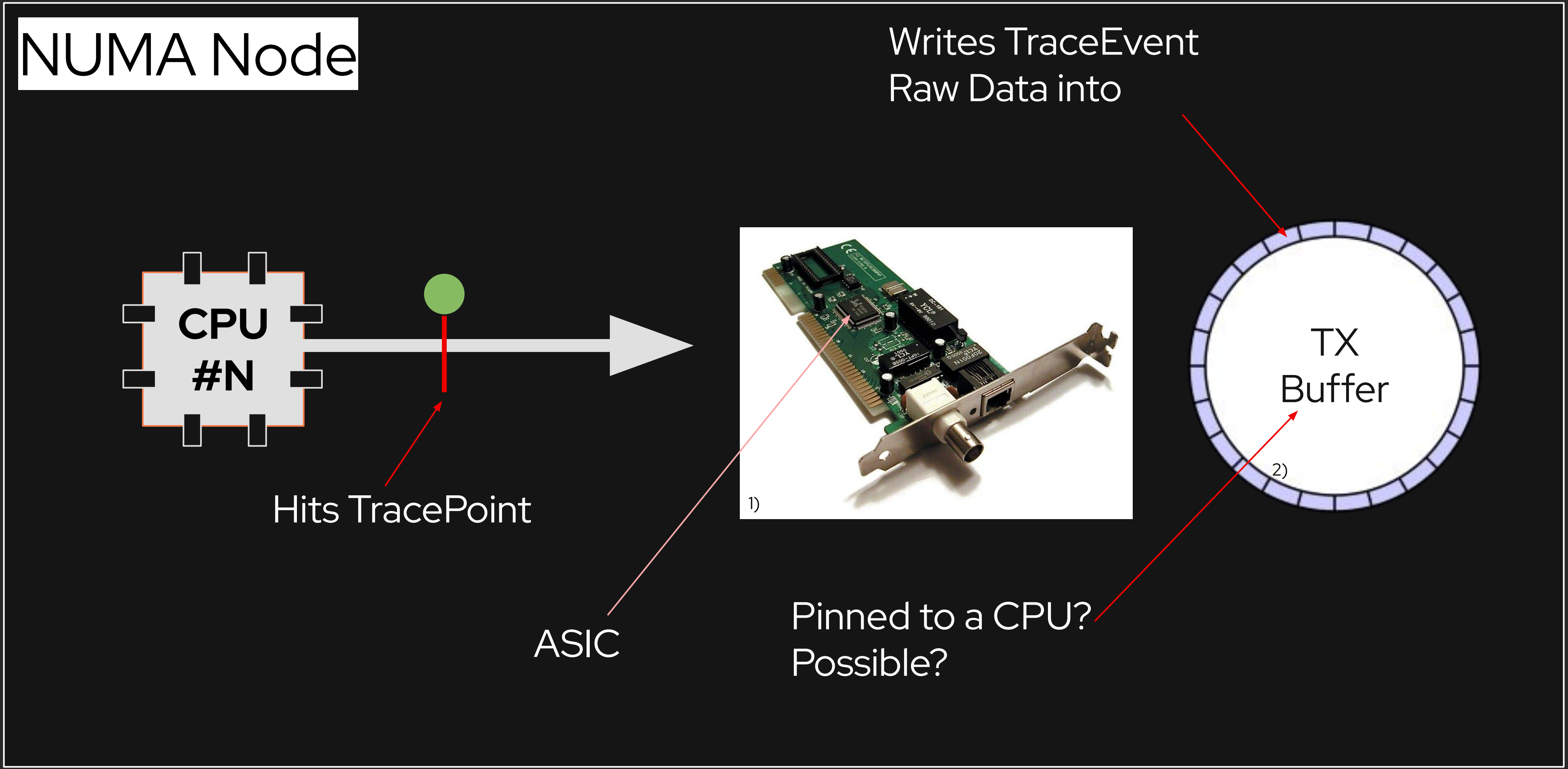
How we can adapt this to net/ ?

- Producer and Consumer
  - Operate directly on NIC DMA Rings
  - Abstract a Tracing Interface/Sockets?
  - Use net/core/page_pool, AF_XDP?
- Offload Filtering
  - Use existing Filtering Infrastructure
  - TC, eBPF Action, even P4?

How we can adapt this to net/ ?

- Encapsulate LinkLayer e.g. Ethernet around raw data?
- Even TCP/IP based? If necessary?
- Local Tracing
  - Internal Loopback to RX Buffer?
  - Loopback cable?
- Remote Tracing
  - Send the data directly to remote Machine
  - Time Synchronized Tracing

Tracing - Producer



1) https://commons.wikimedia.org/wiki/File:Network_card.jpg
2) https://commons.wikimedia.org/wiki/File:Circular_buffer.svg

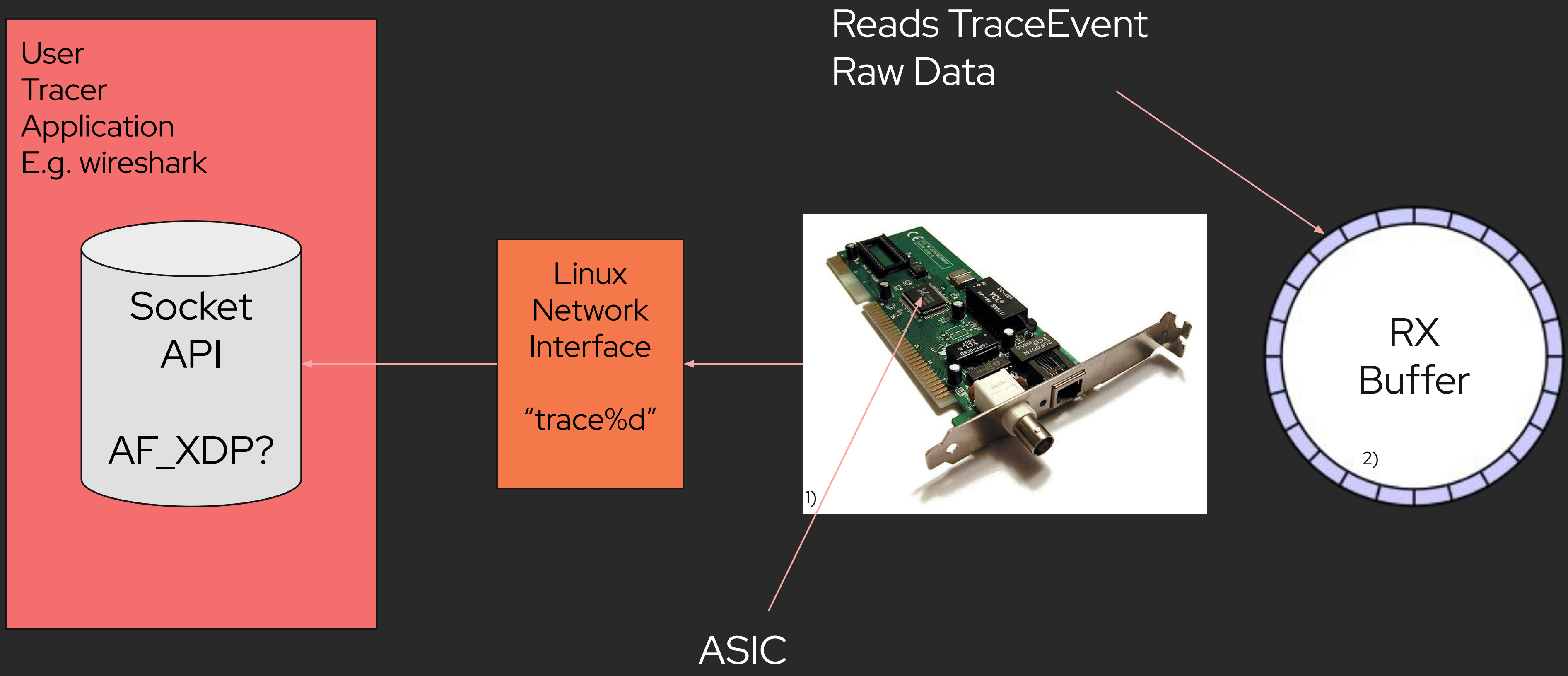
DMA Ring as Tracing Buffer

- Pin DMA Ring to CPU? (Our Page Pool)
- NUMA Node requirement
    - CPU and NIC on same NUMA Node
    - Avoid Traversing over internal Bus
    - See other Netdev Talk 1)

1) https://netdevconf.info/0x18/sessions/talk/multi-pf-single-netdev.html

Red Hat

Tracing - Consumer



1) https://commons.wikimedia.org/wiki/File:Network_card.jpg
2) https://commons.wikimedia.org/wiki/File:Circular_buffer.svg

Tracing Filtering and net/

- Kernel knows the Metadata!
- Can be “discovered” by Tracefs
- Userspace Application only (Control Plane)
    - Easy to use Key-Value (Events) pairs
    - u32 (offload?), eBPF -> Action DROP
    - In Software - May faster than Interpreter?

Traceevent Dropping and net/

- TraceEvent Dropping
    - It is a topic in Linux Tracing (avoid buffer bloating)
    - Unreliable e.g. currently PagePool is Full
    - Reliable Protocols for Tracing Data?
    - Which event to drop?
- Qdisc for Linux Tracing?

Next steps? Proof of Concept?

- Ignore the DMA Ring-Buffer for now
- Focus on the virtual Tracing networking Interface
- Local Tracing only (put Events in a skb)
- Avoid recursion tracing cases
- Wireshark as Linux Tracer (AF_PACKET)
    - Shows Traceevent TLVs
    - Dissector configurable during runtime?

Future steps? Try to Filtering!

- Create user space app to configure Tracing Filter
  - Operates on Tracefs
  - Reads Metadata configure existing Networking Filtering techniques to apply filtering
  - Observe Wireshark Tracer
- Simulate “Remote Tracing” over veth?

Future steps? Look for Performance?

- Try to use real hardware
- Use DMA TX/RX Rings (AF_XDP/"net/core/page_pool.c")?
- Try to offload Filtering on NICs ASIC

Future steps? Time synchronization?

- SO_TIMESTAMPING 1) ? Willem de Bruijn
- Additional metadata TLV required?
- Causality requirement (Events in Order as they appeared in the Network)

1) https://netdevconf.org/0x17/sessions/talk/so_timestamping-powering-fleetwide-rpc-monitoring.html

Thank you

Red Hat