Revolutionizing the ICT Landscape

Sebastiano Miano

eBPF Day, Rome, 13 March 2024

POLITECNICO
MILANO 1863

What is  eBPF ?

Accurate definition

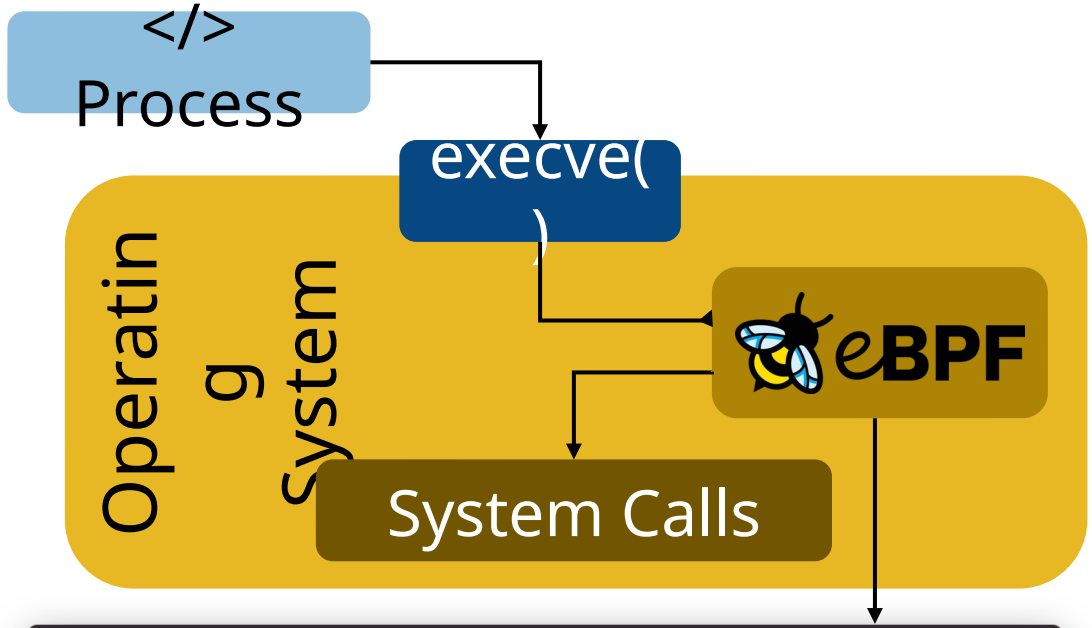
is a programming language & runtime to extend operating systems

Practical comparison

eBPF is like JavaScript/Lua

but for Kernel Developers


```
function hello() {
  alert('hello')
}
</script>
<form onsubmit="return
false;"> <input type="submit"
          name="hello"
          onclick="hello()">
</form>
```

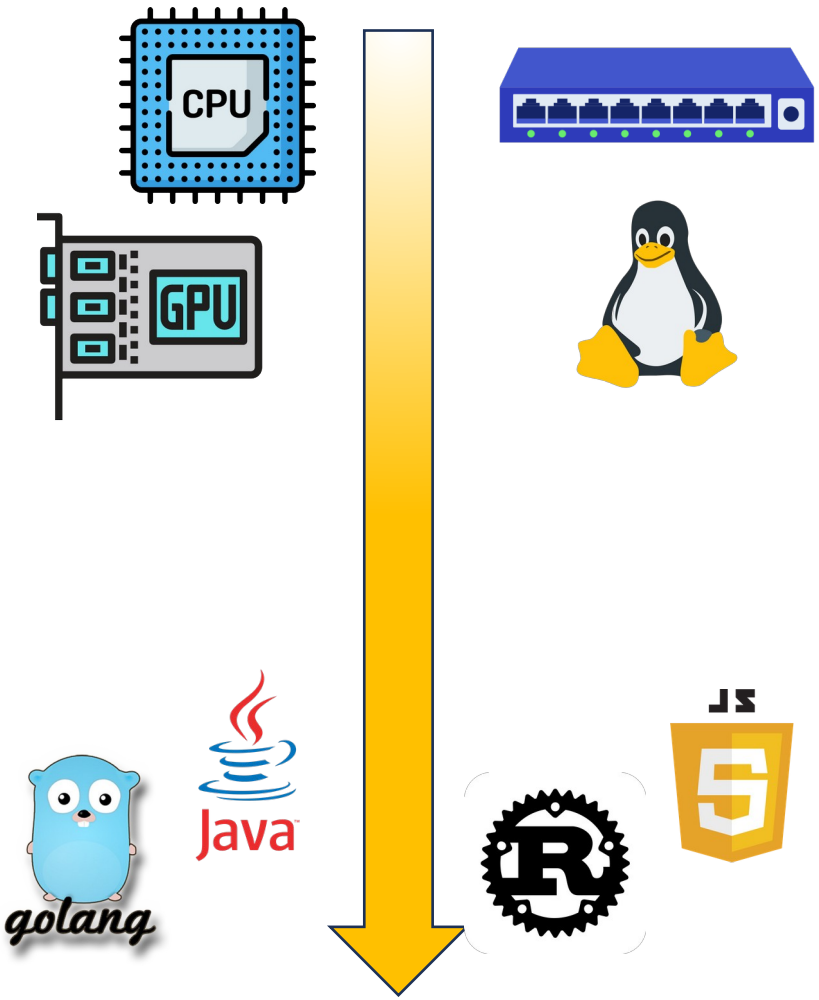



```
int syscall__ret_execve(struct pt_regs *ctx)
{
  struct comm_event event = {
    .pid = bpf_get_current_pid_tgid() >> 32,
    .type = TYPE_RETURN,
  }

  bpf_get_current_comm(&event.comm, sizeof(event.comm));
  comm_events.perfubmit(ctx, &event, sizeof(event));

  return 0
}
```

Why  eBPF ?

Operating Systems are like hardware, hard to change and with long innovation cycles

Long Innovation Cycle

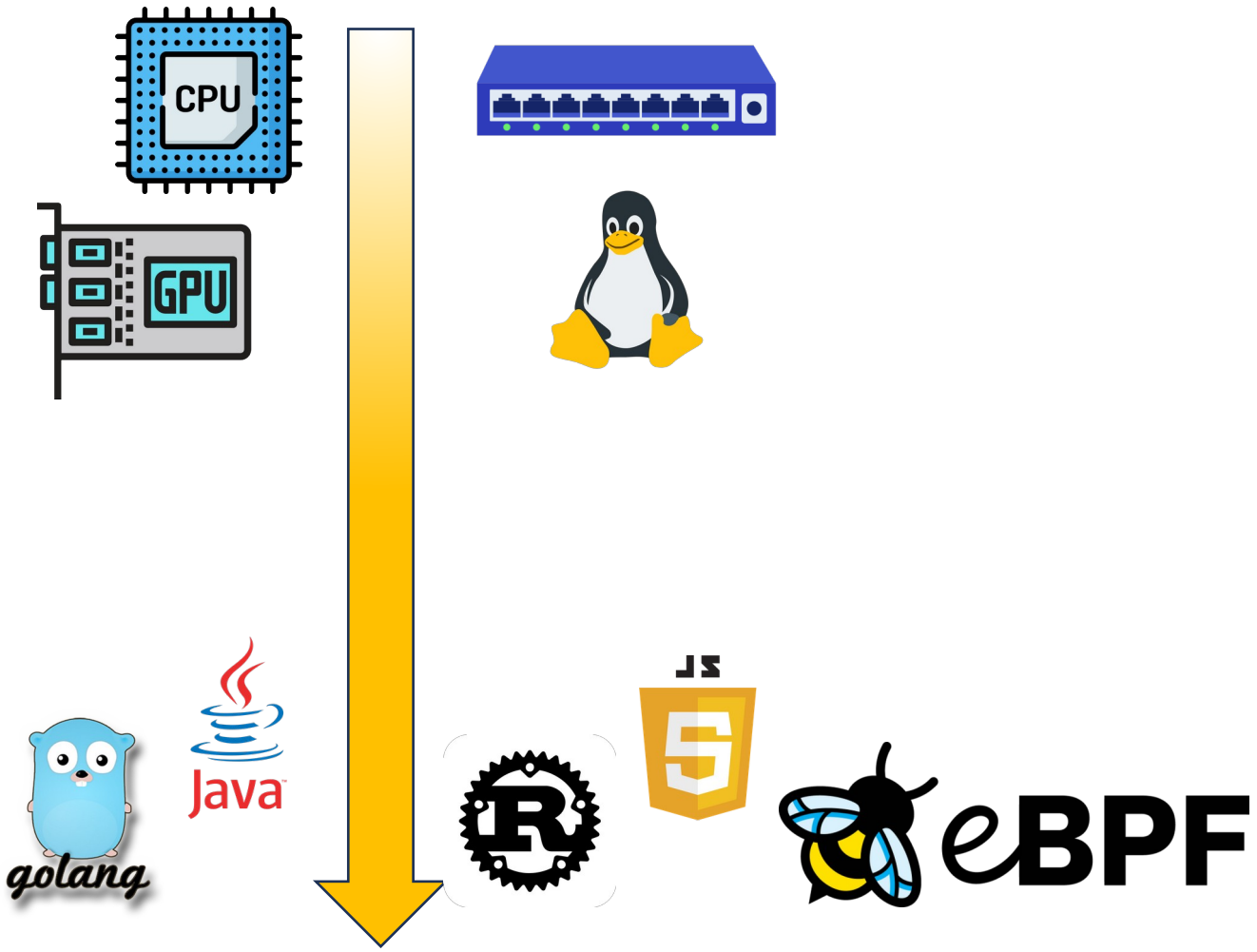


Rapid Innovation Cycle

Before eBPF

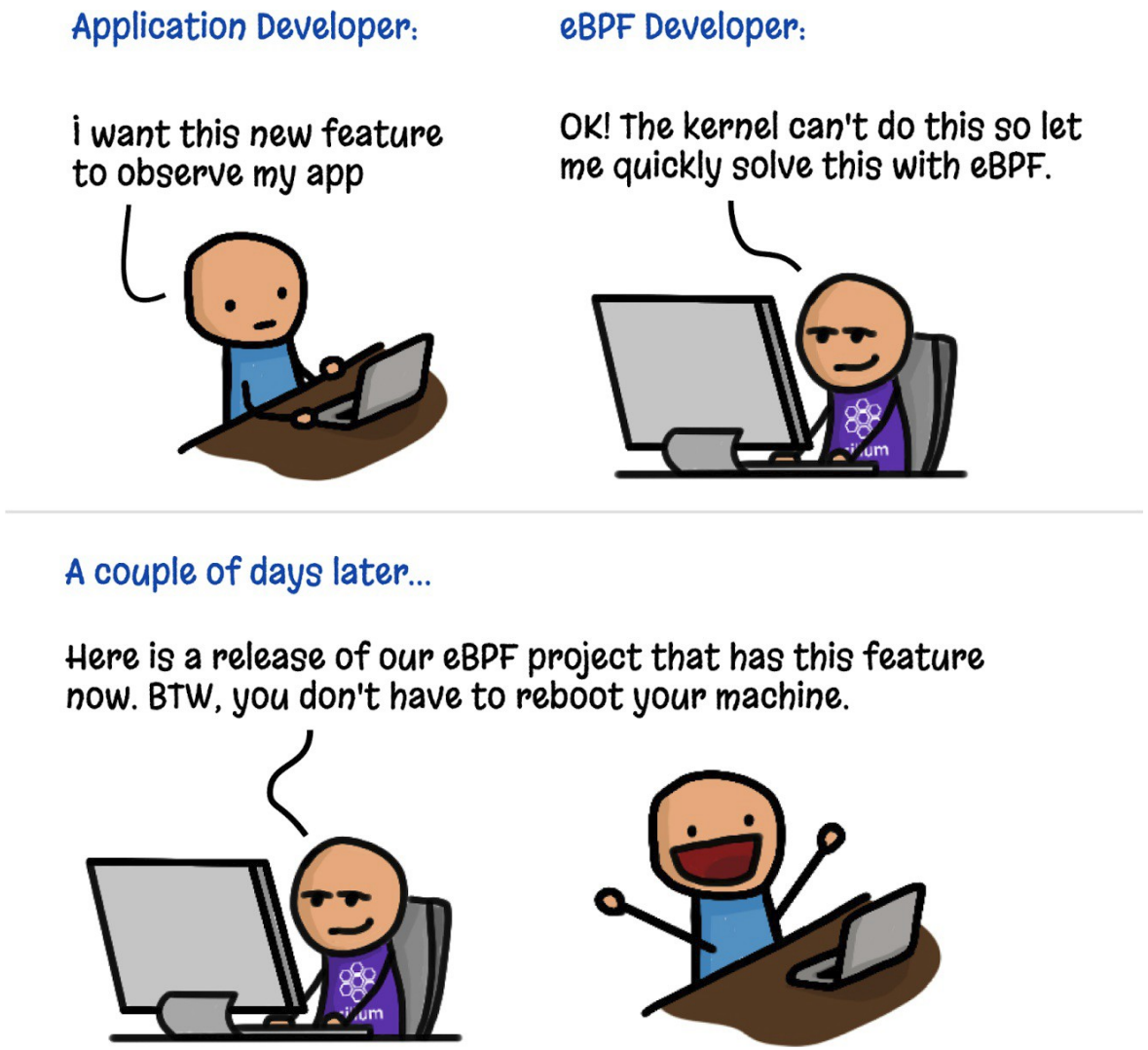

Application Developer:
I want this new feature to observe my app
Hey kernel developer! Please add this new feature to the Linux kernel
OK! Just give me a year to convince the entire community that this is good for everyone.
1 year later...
I'm done. The upstream kernel now supports this.
But I need this in my Linux distro
5 years later...
Good news. Our Linux distribution now ships a kernel with your required feature
OK but my requirements have changed since...

Long Innovation Cycle

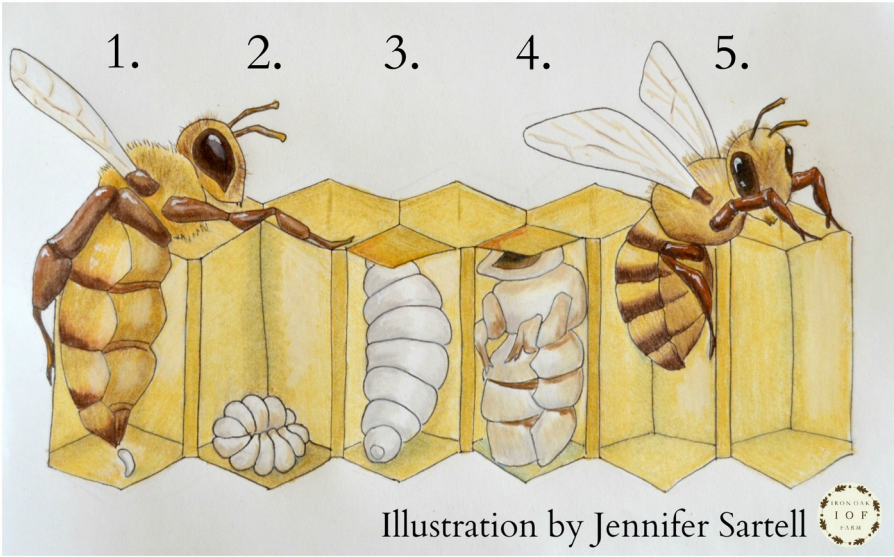


Rapid Innovation Cycle

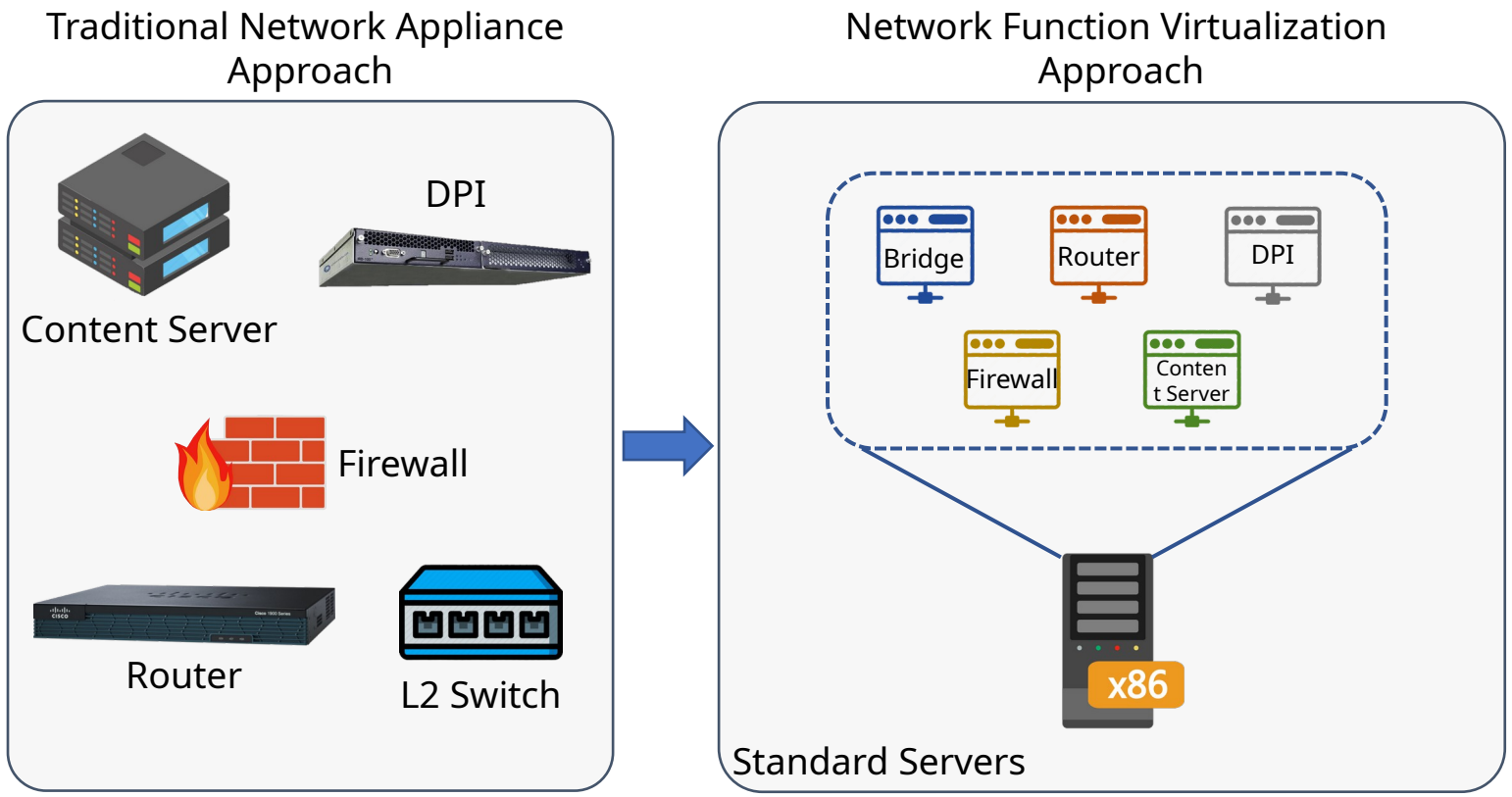
After eBPF


Application Developer:
i want this new feature to observe my app
eBPF Developer:
OK! The kernel can't do this so let me quickly solve this with eBPF.
A couple of days later...
Here is a release of our eBPF project that has this feature now. BTW, you don't have to reboot your machine.

How eBPF was born?



The era of Network Function Virtualization



Shift in the Networking Business

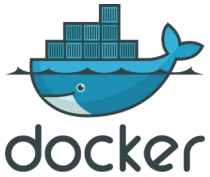
Hardware Business

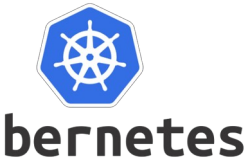
Software Business

Router

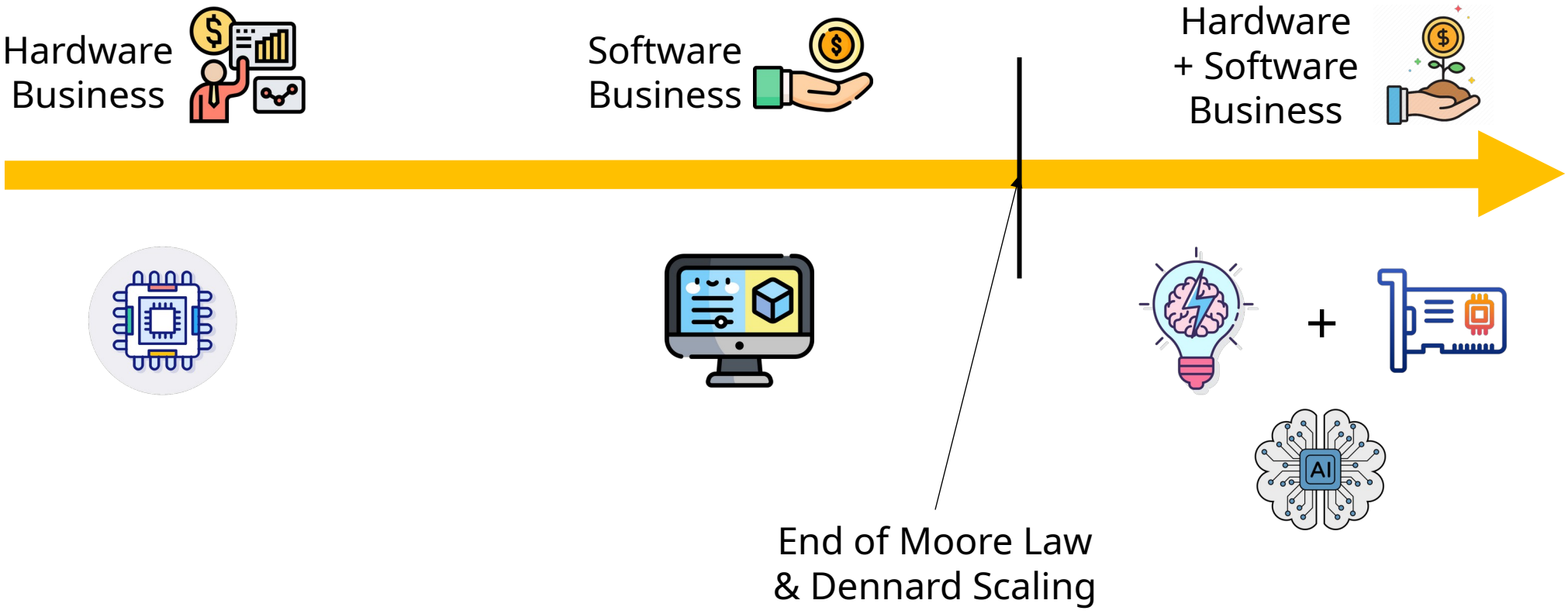
L2 Switch

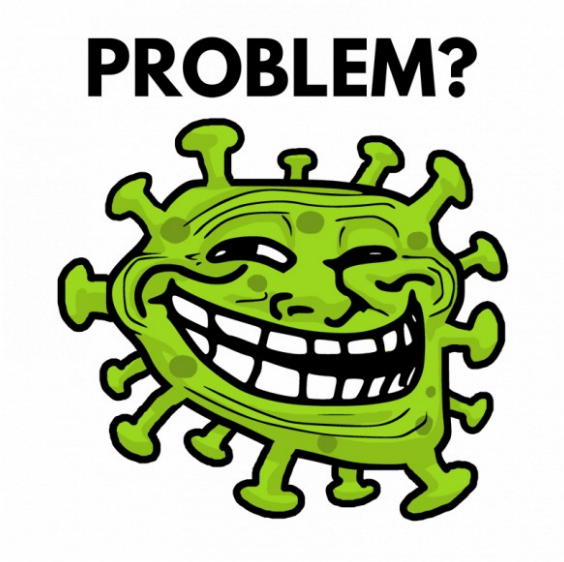
DPI





Shift in the **Networking Business** in **2024**

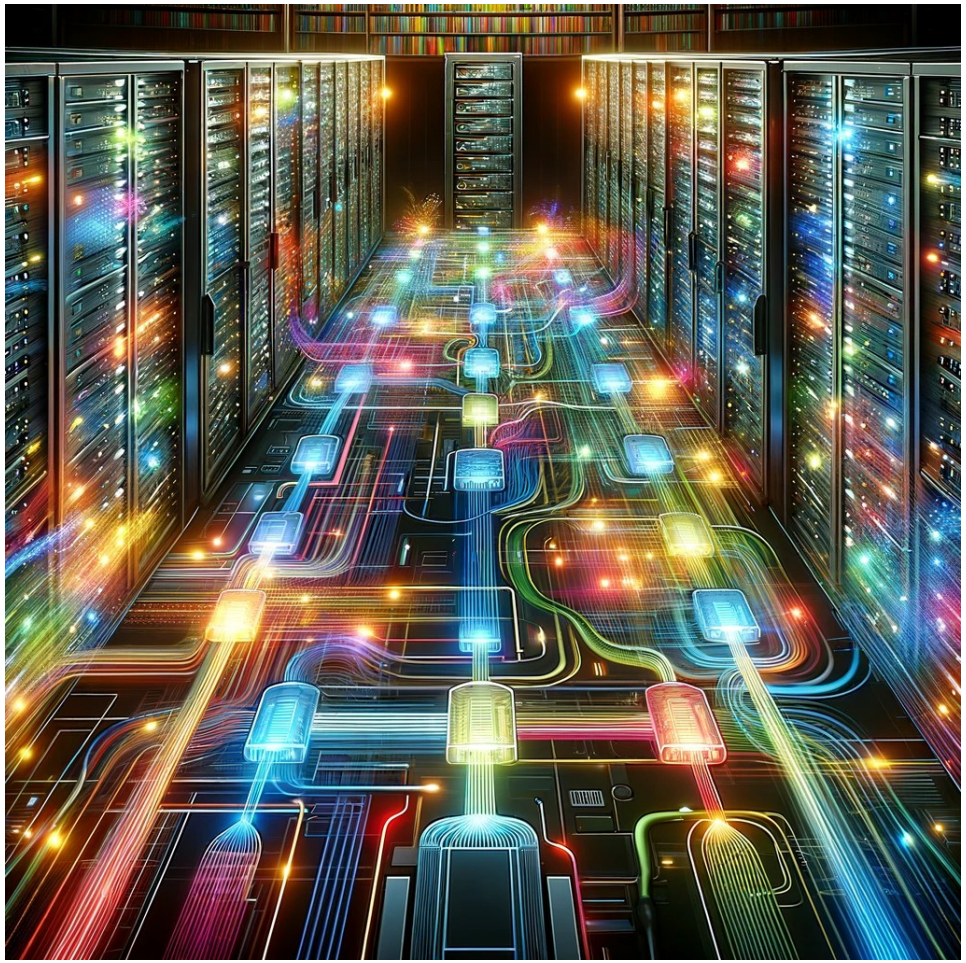




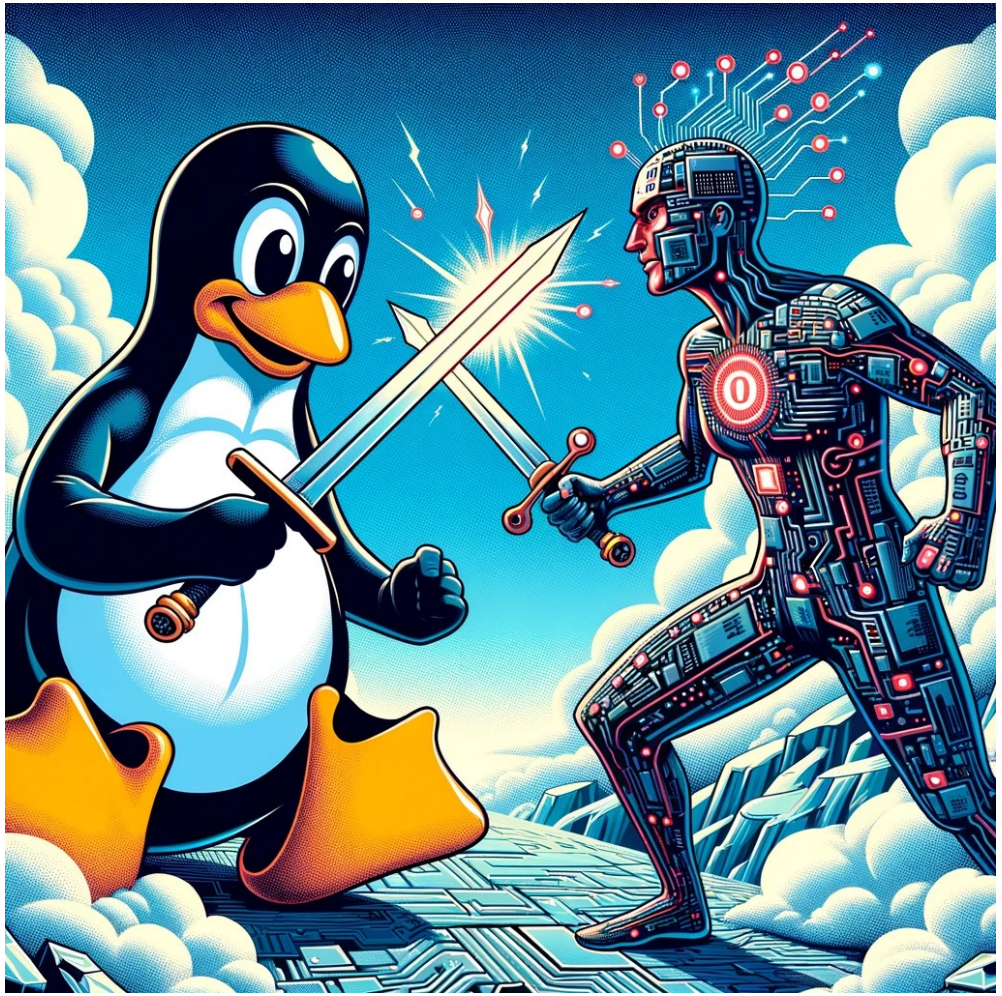
Which problem we were trying to solve?

Move packets efficiently across servers

- With NFV, the **single server** takes a **prominent** role in networking
- Most of the servers are Linux-based, however...
- ...at that time the Linux kernel was focusing only on bridging and routing



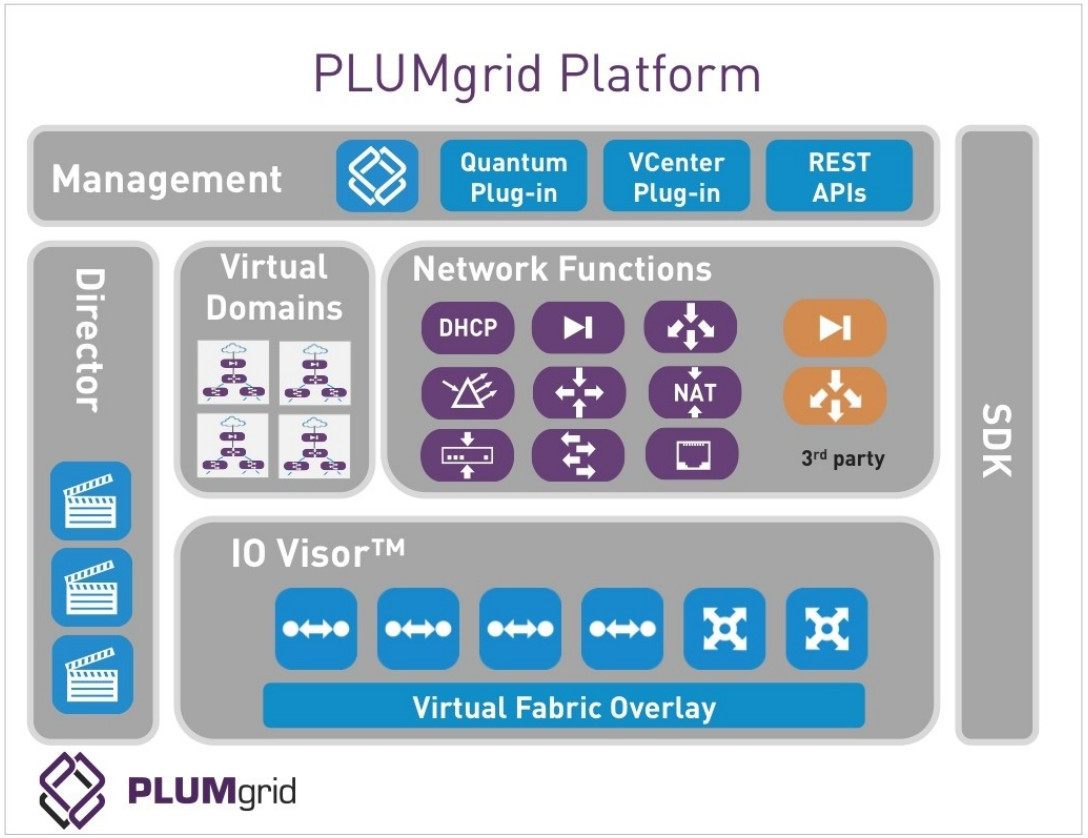
Linux **Tux**: “Innovation? Nah…”



eBPF Storyline

PLUMgrid: Where eBPF started

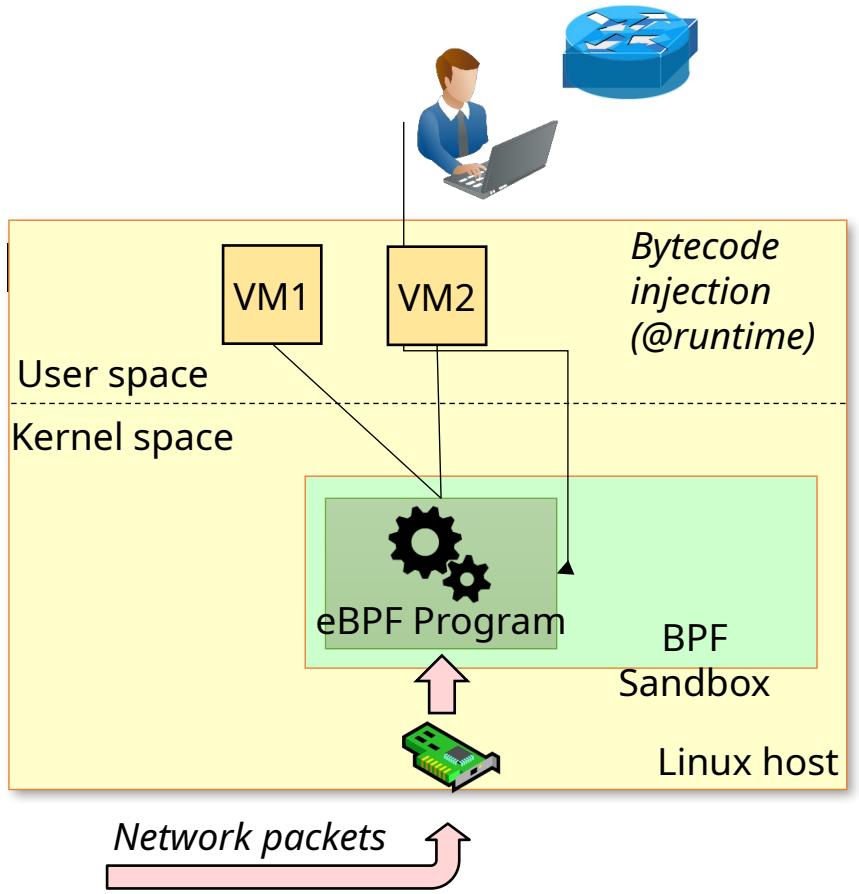
- The idea of PLUMgrid was to create a collection of “plumlets” connected to perform custom networking.
- Those plumlets can be loaded in the kernel
  - Enforce safety of loaded programs into the kernel using a customized language, compiler, like Rust



eBPF Key feature #1

#1: Runtime bytecode injection

- eBPF programs can be dynamically created and injected in the kernel at run-time
  - Vanilla Linux kernel, without any patch
  - No additional kernel module
  - Obviously, no need to recompile the kernel



Alexei Starovoitov: The God

- One day, Alexei hit a “bug” that was causing the host to crash after hours of networking traffic
- He recognized that we cannot trust the compiler, but we need something in the kernel that verifies the code itself

eBPF Key feature #2

#2: In-kernel verifier!

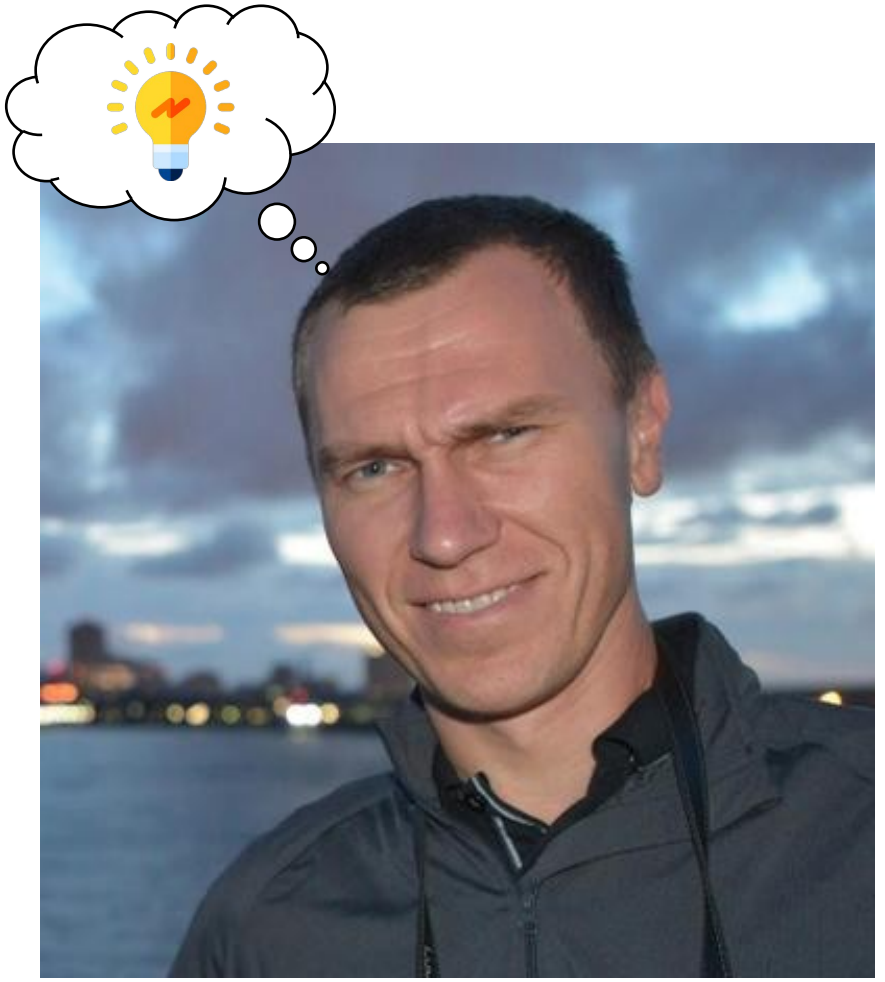
- Linux kernel must be protected from erroneous or malicious injected programs
- Achieved with a **sandbox** that prevents possible critical conditions at run-time
- **A verifier** checks the code and refuse to inject it in the sandbox
  - No invalid memory accesses
  - Bounded program size
  - Bounded max number of instructions

- Consequence: eBPF does not support completely arbitrary programs
  - Even if the eBPF language **is** Turing complete

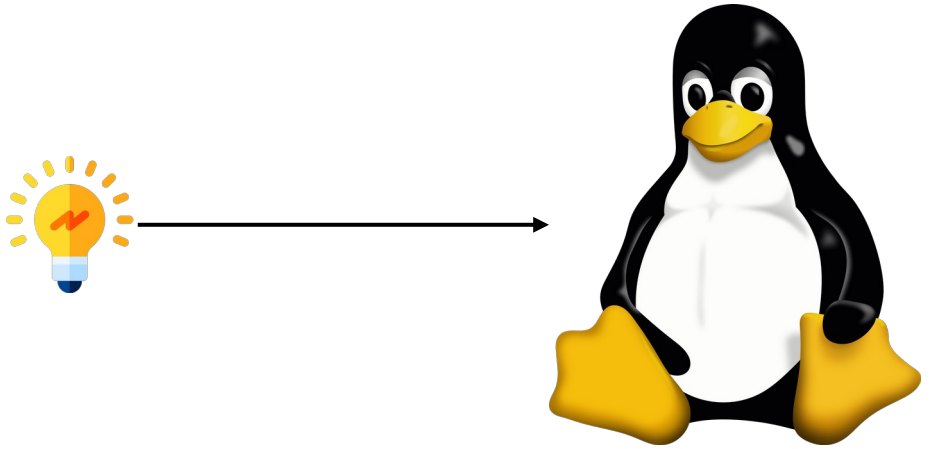
eBPF Day

In-kernel verifier! Yeah...but how to do it?

- Since he had to change the compiler to support this new language, he invented its own instruction set!

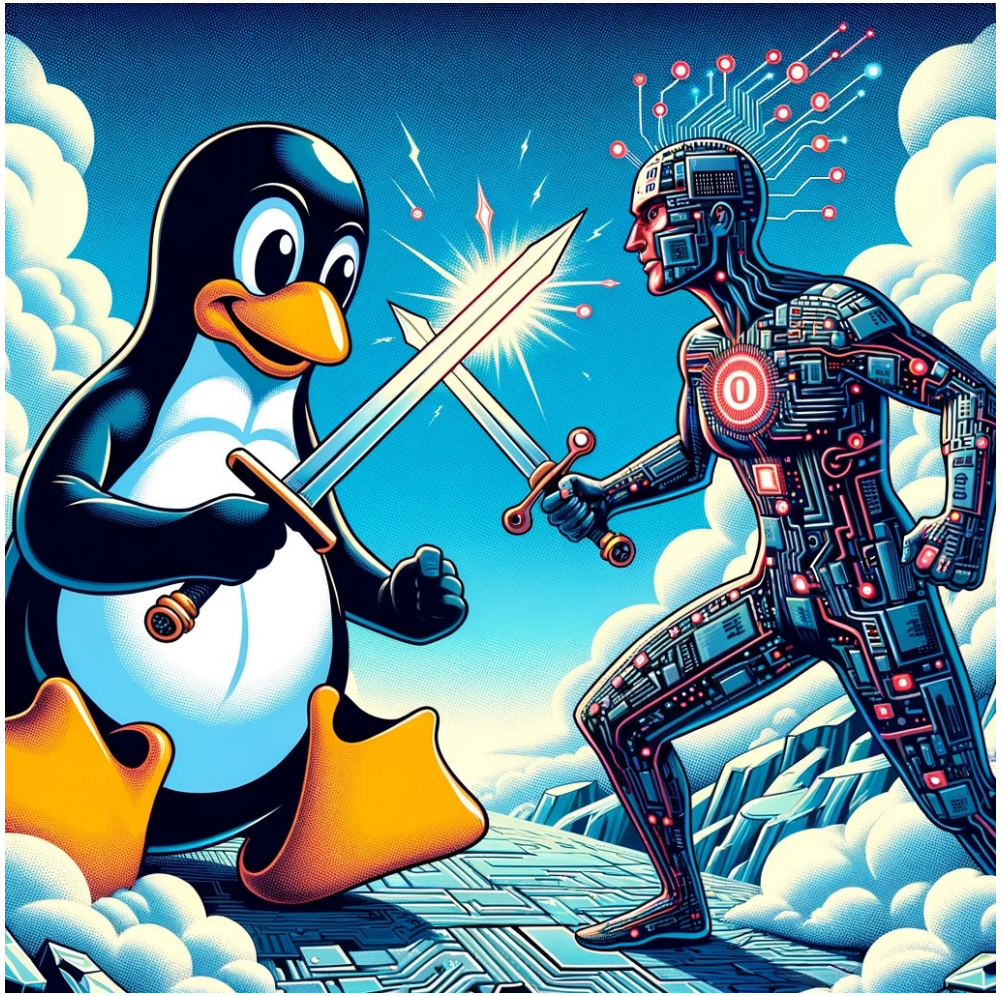


In-kernel verifier! Yeah...but how to do it?



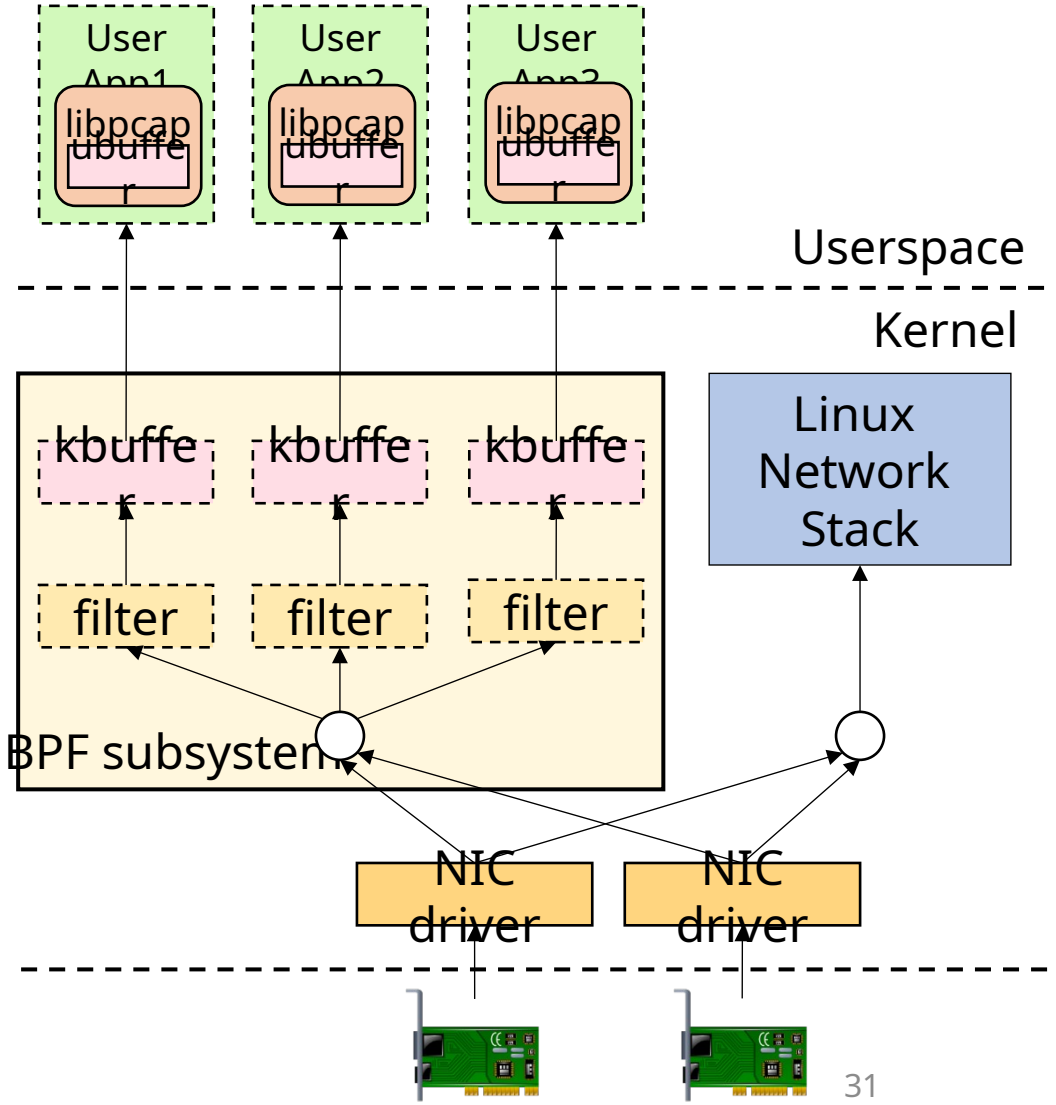
eBPF Storyline

Linux **Tux**: “Innovation? Nah…”



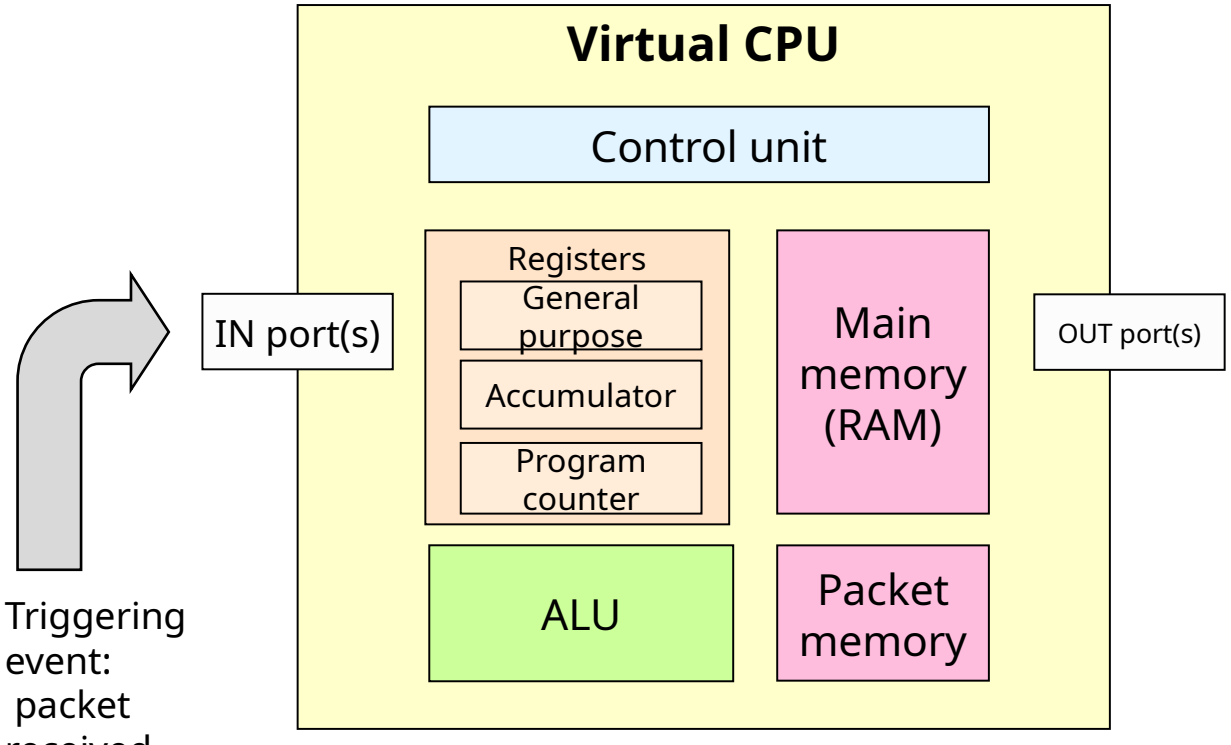
The Berkeley Packet Filter (BPF)

- Generic **in-kernel, even-based virtual CPU**
  - Introduced in 1993 paper from Lawrence Berkeley National Laboratory
  - Available in Linux kernel 2.1.75 (1997)
  - Initially used as packet filter by packet capture tool **tcpdump** (via libpcap)
- In-kernel
  - No syscall overhead, kernel/user context switching



Special purpose Virtual CPU

- Ad-hoc execution environment specially crafted for packet filtering purposes
  - E.g., specific memory for packets (separated from the main RAM)
- vCPU interpreter




```
Example of BPF injected code

Filter: “ip” (with simple Ethernet frames)

(000) ldh   [12]
(001) jeq   #0x800   jt 2   jf 3
(002) ret   #96
(003) ret   #0
```

eBPF Storyline

tracing: accelerate tracing filters with BPF

From: Alexei Starovoitov <ast@plumgrid.com>
To: Ingo Molnar <mingo@kernel.org>
Subject: [PATCH RFC net-next] tracing: accelerate tracing filters with BPF
Date: Tue, 13 May 2014 19:55:11 -0700
Message-ID: <1400036111-7803-1-git-send-email-ast@plumgrid.com>
Cc: "David S. Miller" <davem@davemloft.net>, Eric Dumazet <edumazet@google.com>, Daniel Borkmann <dborkman@redhat.com>, Steven Rostedt <rostedt@goodmis.org>, Peter Zijlstra <a.p.zijlstra@chello.nl>, Arnaldo Carvalho de Melo <acme@infradead.org>, Jiri Olsa <jolsa@redhat.com>, Thomas Gleixner <tglx@linutronix.de>, "H. Peter Anvin" <hpa@zytor.com>, netdev@vger.kernel.org, linux-kernel@vger.kernel.org
Archive-link: Article

2012-2014

Alexei working at PLUMgrid

2014

First eBPF patch set.

author	Alexei Starovoitov <ast@plumgrid.com> 2014-03-28 18:58:25 +0100
committer	David S. Miller <davem@davemloft.net> 2014-03-31 00:45:09 -0400
commit	bd4cf0ed331a275e9bf5a49e6d0fd55dffc551b8 (patch)
tree	6ffb15296ce4cdc1f272e31bd43a5804b8da588c
parent	77e0114ae9ae08685c503772a57af21d299c6701 (diff)
download	linux-bd4cf0ed331a275e9bf5a49e6d0fd55dffc551b8.tar.gz

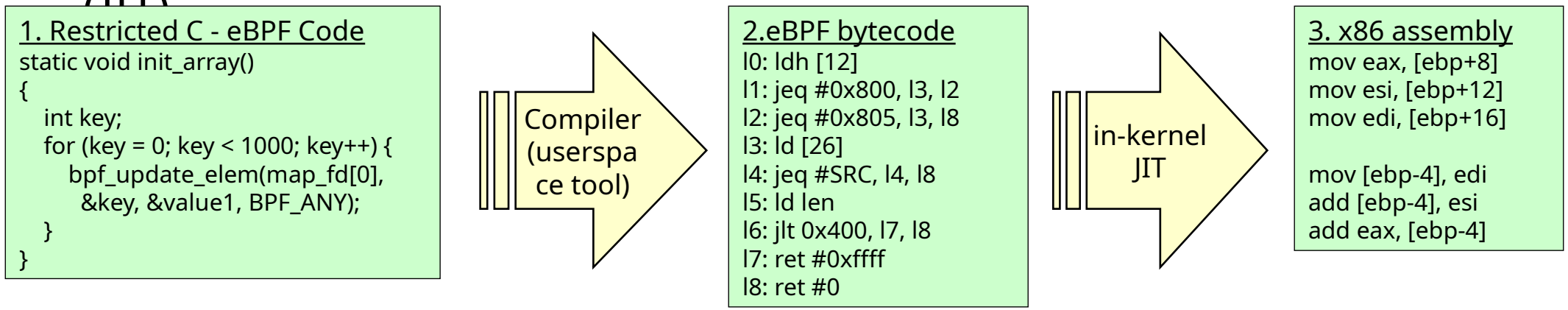
net: filter: rework/optimize internal BPF interpreter's instruction set

```
This patch replaces/reworks the kernel-internal BPF interpreter with
an optimized BPF instruction set format that is modelled closer to
mimic native instruction sets and is designed to be JITed with one to
one mapping. Thus, the new interpreter is noticeably faster than the
current implementation of sk_run_filter(); mainly for two reasons:
```

eBPF Key feature #3

#3: Efficiency with JIT compilation

- eBPF programs consumes a little amount of resources
- They executes in kernel space, potentially close to when packets are received (no need to copy packets as when we move them from kernel to user)
- BPF bytecode is either
  - interpreted
  - translated into native assembly code with a Just-in-time translator (JIT)

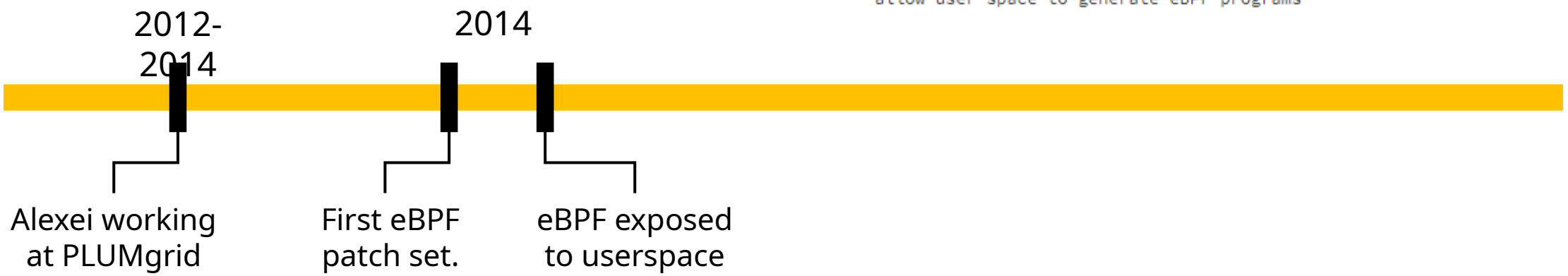


eBPF Storyline

author	Alexei Starovoitov <ast@plumgrid.com>	2014-09-04 22:17:18 -0700
committer	David S. Miller <davem@davemloft.net>	2014-09-09 10:26:47 -0700
commit	daedfb22451dd02b35c0549566cbb7cc06bdd53b (patch)	
tree	f990840a7b9e6afe48ea73a5fdafe1cdc50f936d	
parent	02ab695bb37ee9ad515df0d0790d5977505dd04a (diff)	
download	linux-daedfb22451dd02b35c0549566cbb7cc06bdd53b.tar.gz	

net: filter: split filter.h and expose eBPF to user space

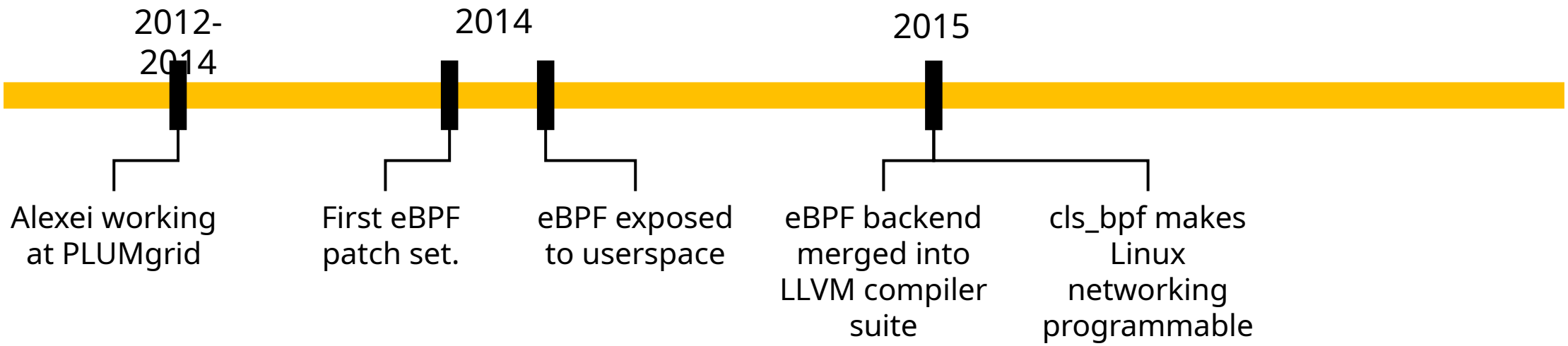
allow user space to generate eBPF programs



The **extended** Berkeley Packet Filter (**e**BPF)

- Officially part of the Linux kernel since 3.15
  - In practice, kernel 4.x are required to take advantages of the more advanced features
  - Continuously evolving platform
- Naming:
  - Initially, new eBPF was identified with “eBPF”, and old BPF called “Classic BPF” or “cBPF”
  - Recently (2018), people tend to refer to this technology simply as “BPF”
    - The “cBPF” has been now replaced and it is converted to eBPF in newer kernels

eBPF Storyline



is just for

Networking?

A new use case eBPF

eBPF Day

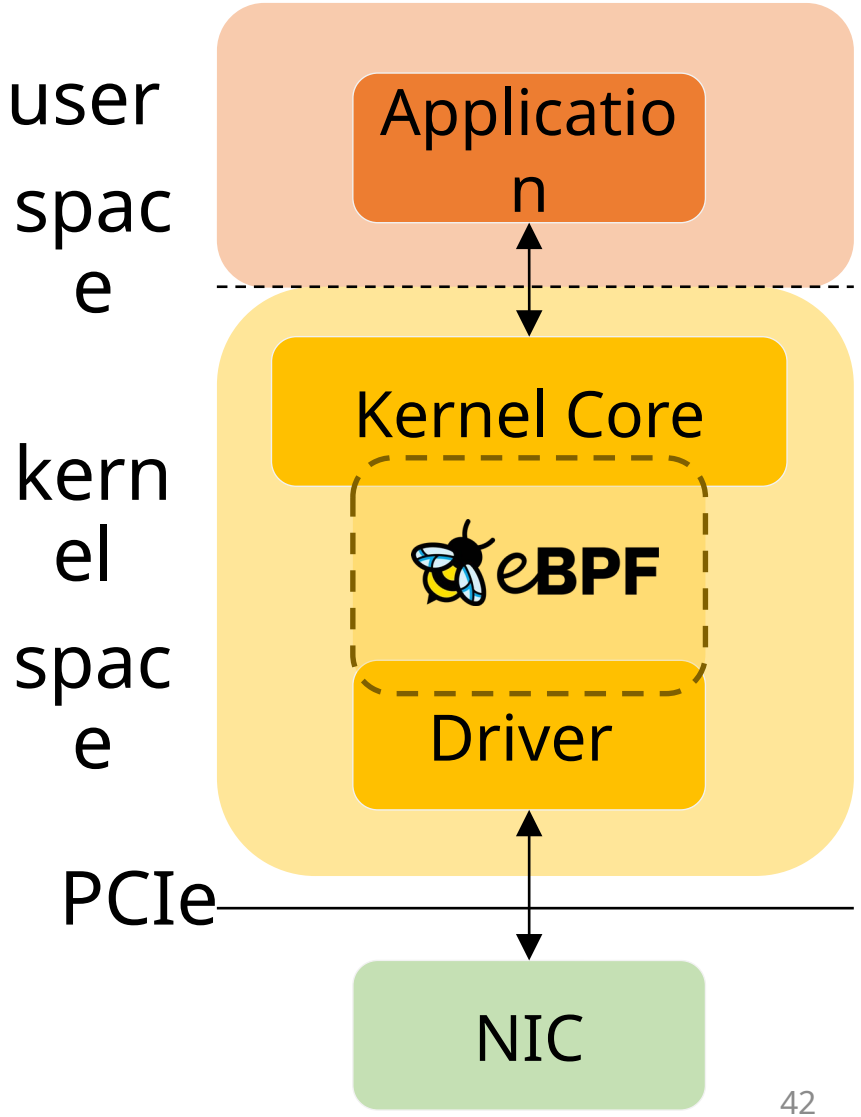
NETFLIX

- Brendan was looking for tools and approaches to perform **kernel tracing**
  - **Lots of existing tools were not usable and half finished**
  - **None of them was able to do what they needed**

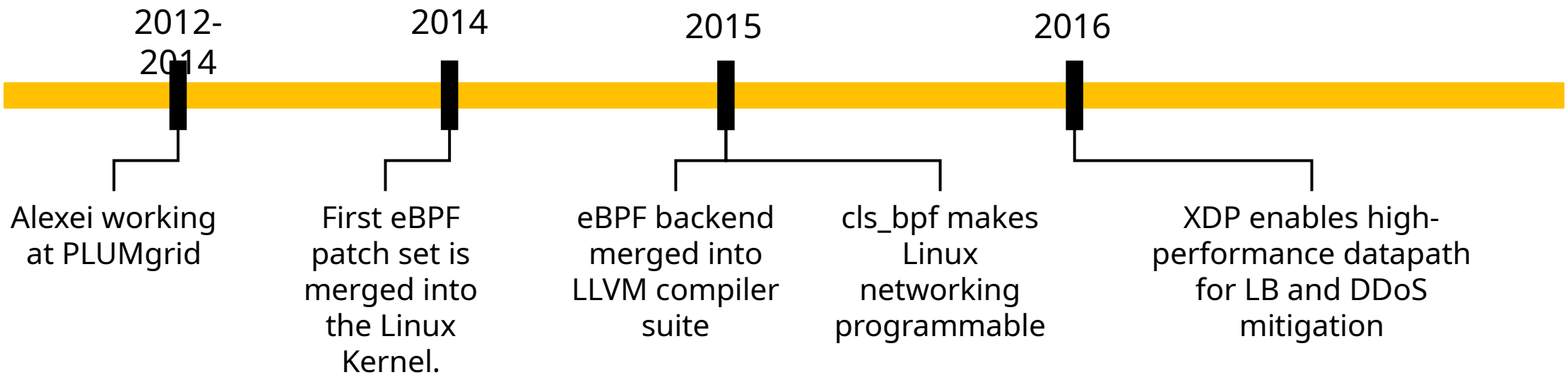
eBPF Key feature #4

#4: React to generic kernel events

- eBPF code is hooked to a kernel event
  - When fired, your code (associated to an *event handler*) is executed
- Some possible events:
  - Network packet received
  - Message (socket-layer) received
  - Data written to disk
  - Page fault in memory
  - File in /etc folder being modified
- In general, any kernel event can be potentially intercepted



eBPF Storyline



eBPF Storylin

Sebastiano started working with eBPF!

2012-2014: Alexei working at PLUMgrid

2014: First eBPF patch set is merged into the Linux Kernel.

2015: eBPF backend merged into LLVM compiler suite

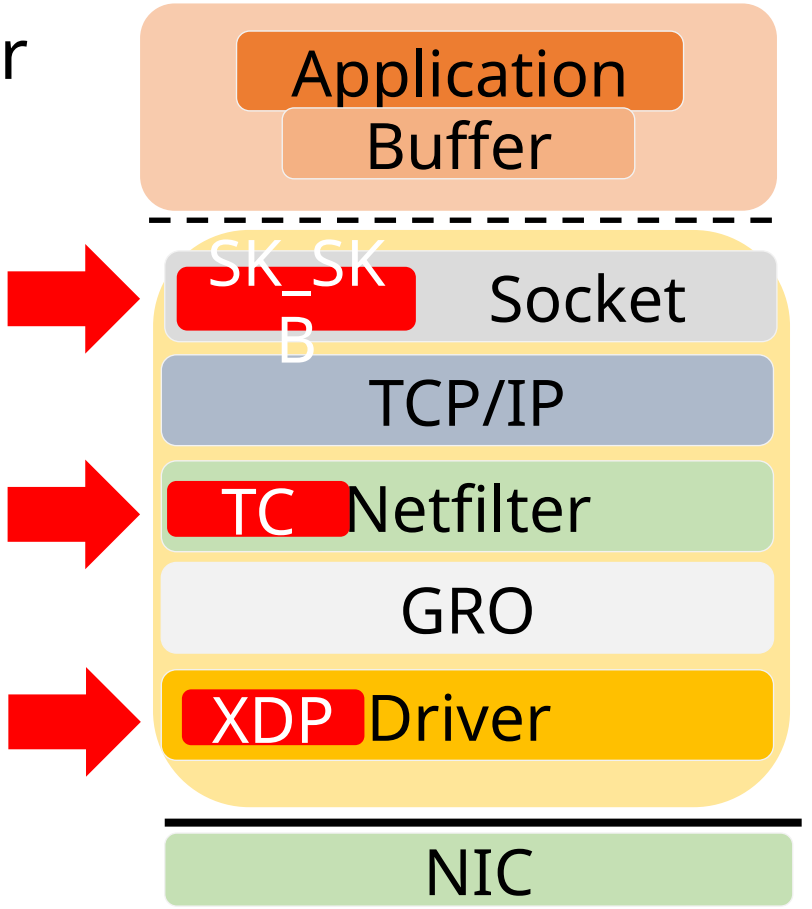
2015: cls_bpf makes Linux networking programmable

2016: XDP enables high-performance datapath for LB and DDoS mitigation

Key feature #5

#5: eBPF Hook points

- Several hook points (a.k.a. kernel events) for networking:
  - Located at different levels of the stack
  - Opens the possibility to implement packet processing programs at different layers of the stack
- Some of interest:
  - eXpress Data Path (XDP)
  - Traffic Control (TC)
  - Socket SKB (SK_SKB)
  - There are many more…



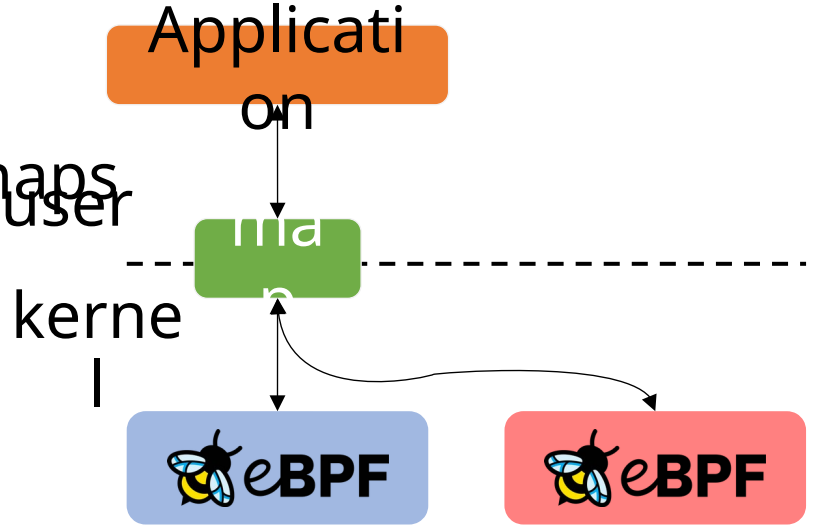
Key features #N

I want more features!

- #6: eBPF **Helpers**
  - Functions that are implemented natively in the Linux kernel, which are available as an assembly call

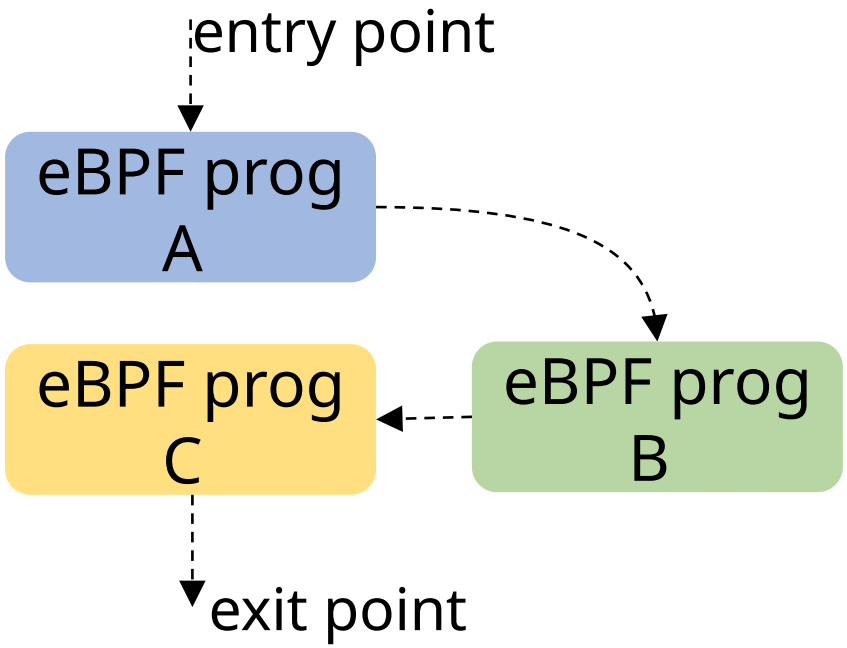
I want more features!

- #6: eBPF **Helpers**
  - Functions that are implemented natively in the Linux kernel, which are available as an assembly call
- #7: **Persistent** storage
  - Data access arbitrated by structures called maps
    - Key/value storage of different types
    - Array, HashMap, LRUMap..



I want more features!

- #6: eBPF **Helpers**
  - Functions that are implemented natively in the Linux kernel, which are available as an assembly call
- #7: **Persistent** storage
  - Data access arbitrated by structures called maps
    - Key/value storage of different types
    - Array, HashMap, LRUMap..
- #8: **Service chains**
  - eBPF programs can be chained together to create complex (and modular) services
    - Enable to split a complex function in multiple components



More details in the next talks!

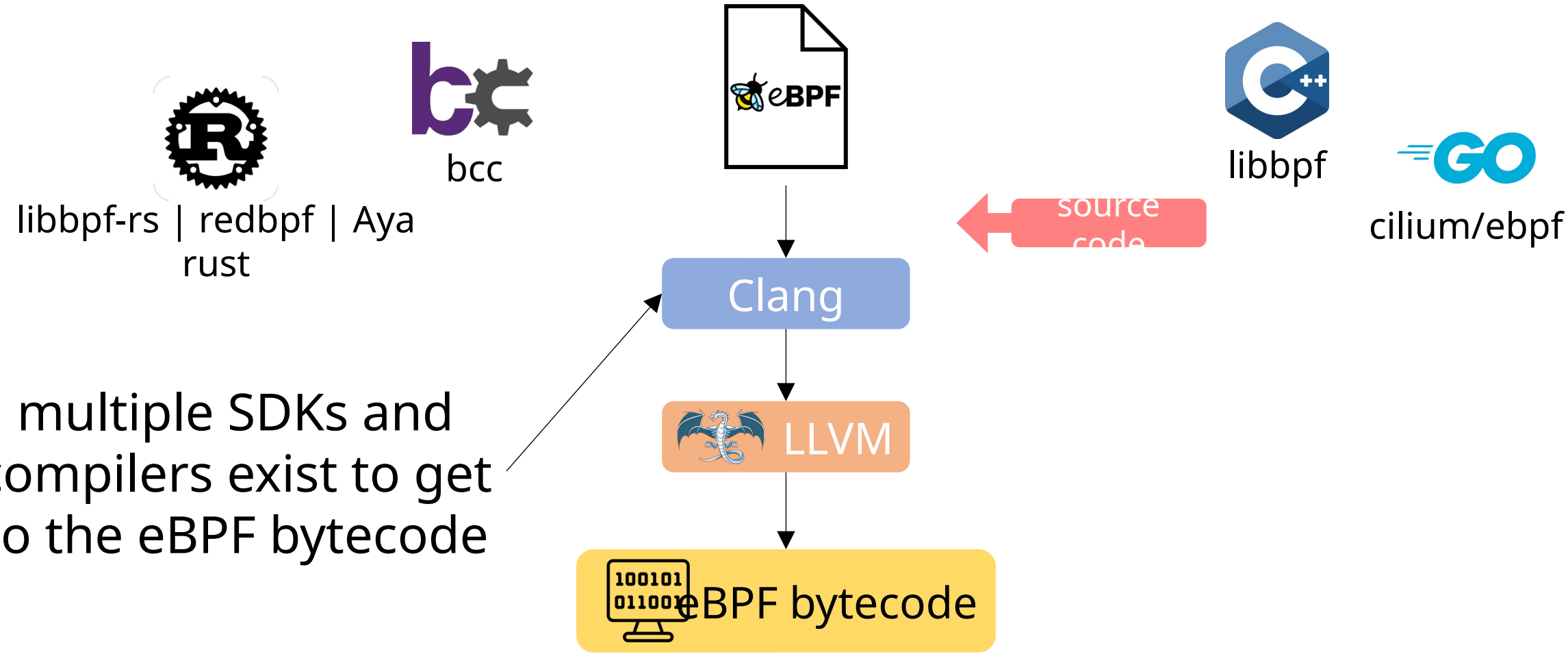
How does  eBPF work?

eBPF language

+

eBPF runtime

eBPF language

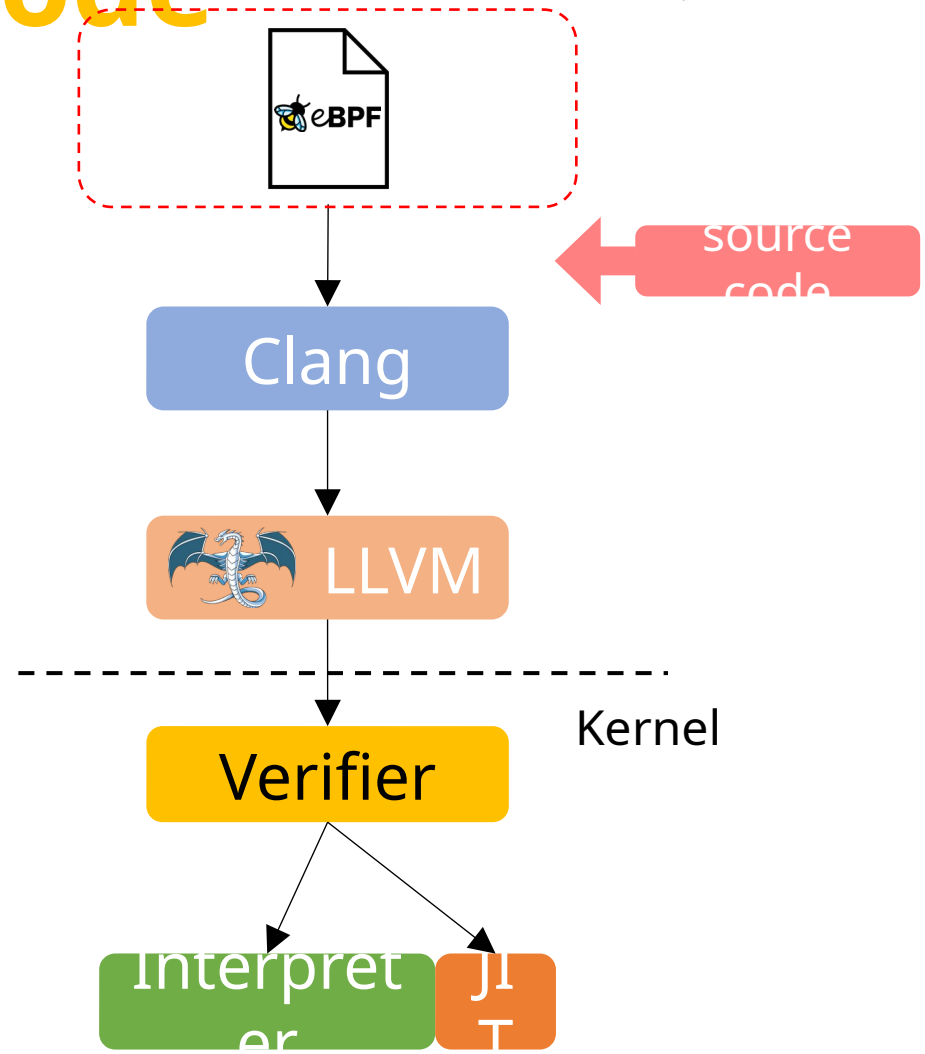


multiple SDKs and compilers exist to get to the eBPF bytecode

eBPF toolchain – Source code

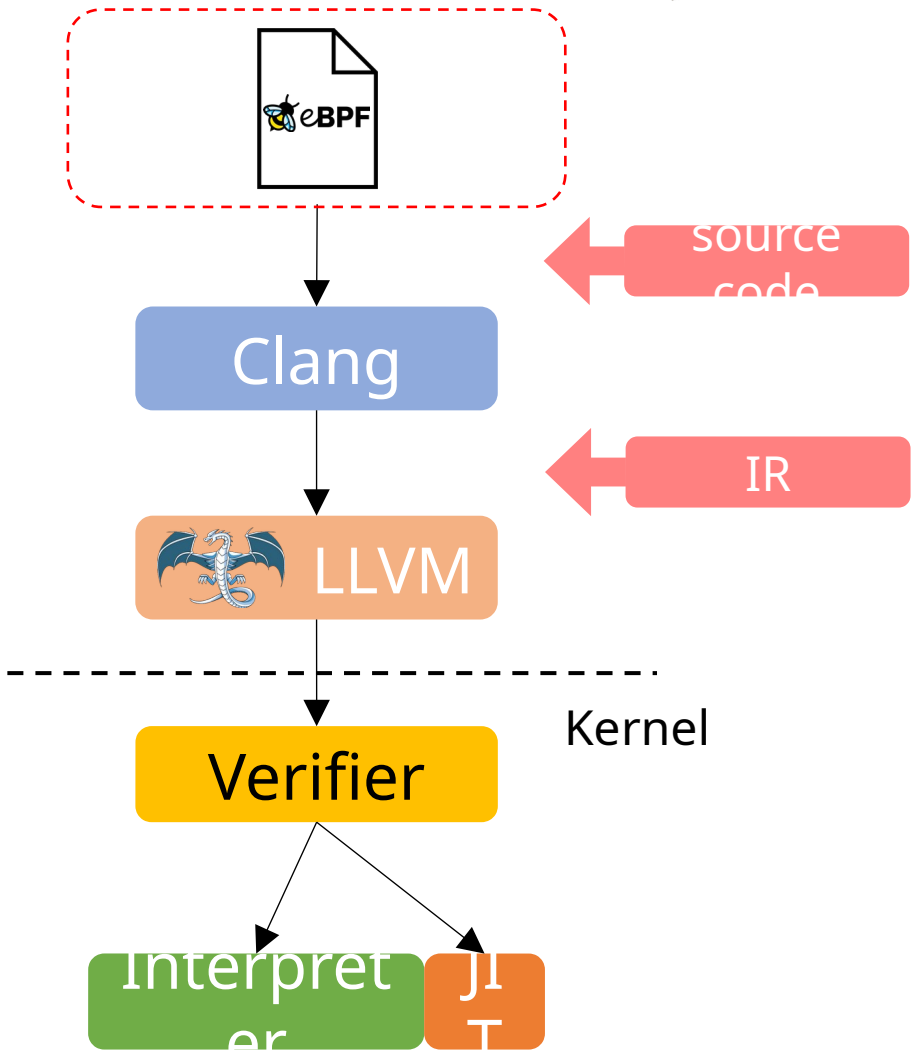
- eBPF code is written in restricted C (compilers exist for other languages)

```c
static __always_inline int parse_ethhdr(void *data, void *data_end,
                                        __u16 *nh_off,
                                        struct ethhdr **ethhdr) {
    struct ethhdr *eth = (struct ethhdr *)data;
    int hdr_size = sizeof(*eth);

    /* Byte-count bounds check; check if current pointer + size of header
      * is after data_end.
      */
    if ((void *)eth + hdr_size > data_end)
        return -1;

    *nh_off += hdr_size
    *ethhdr = eth;

    return eth->h_proto; /* network-byte-order */
}
```



eBPF toolchain – LLVM IR

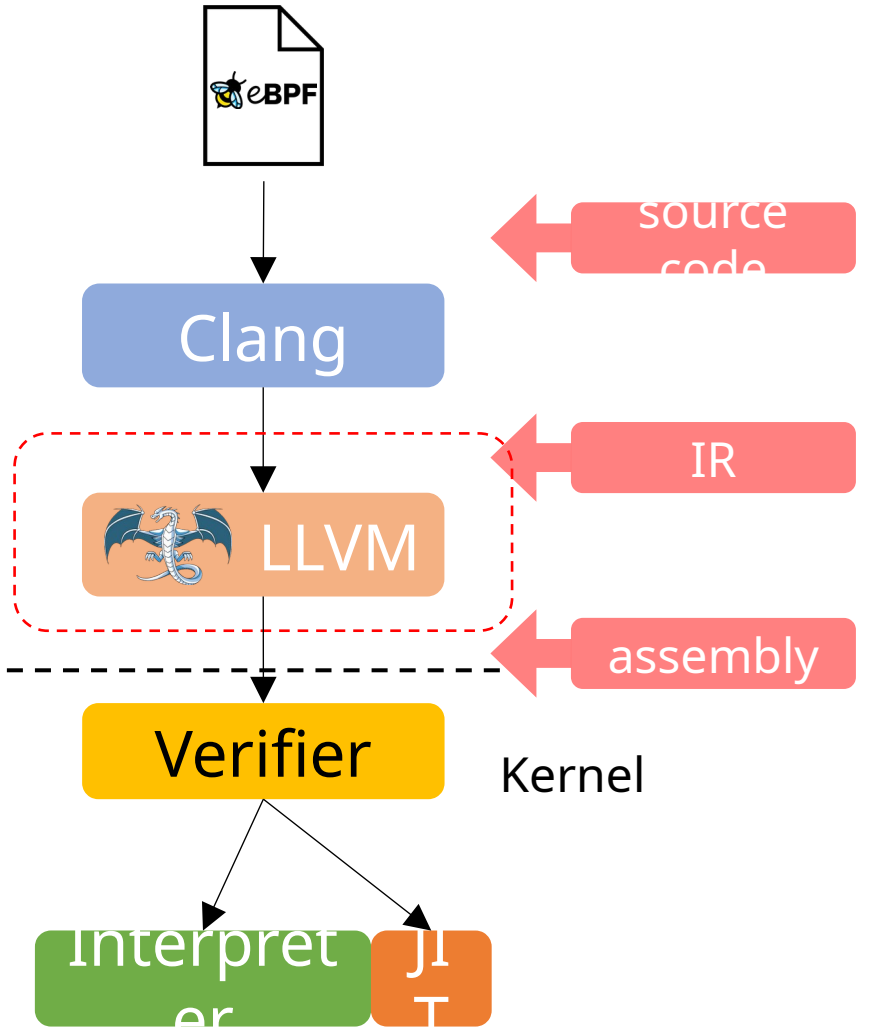
- The code gets processed by Clang, a compiler front end for C-style programming languages
  - There should also be support for eBPF on GCC, but Clang is the most used



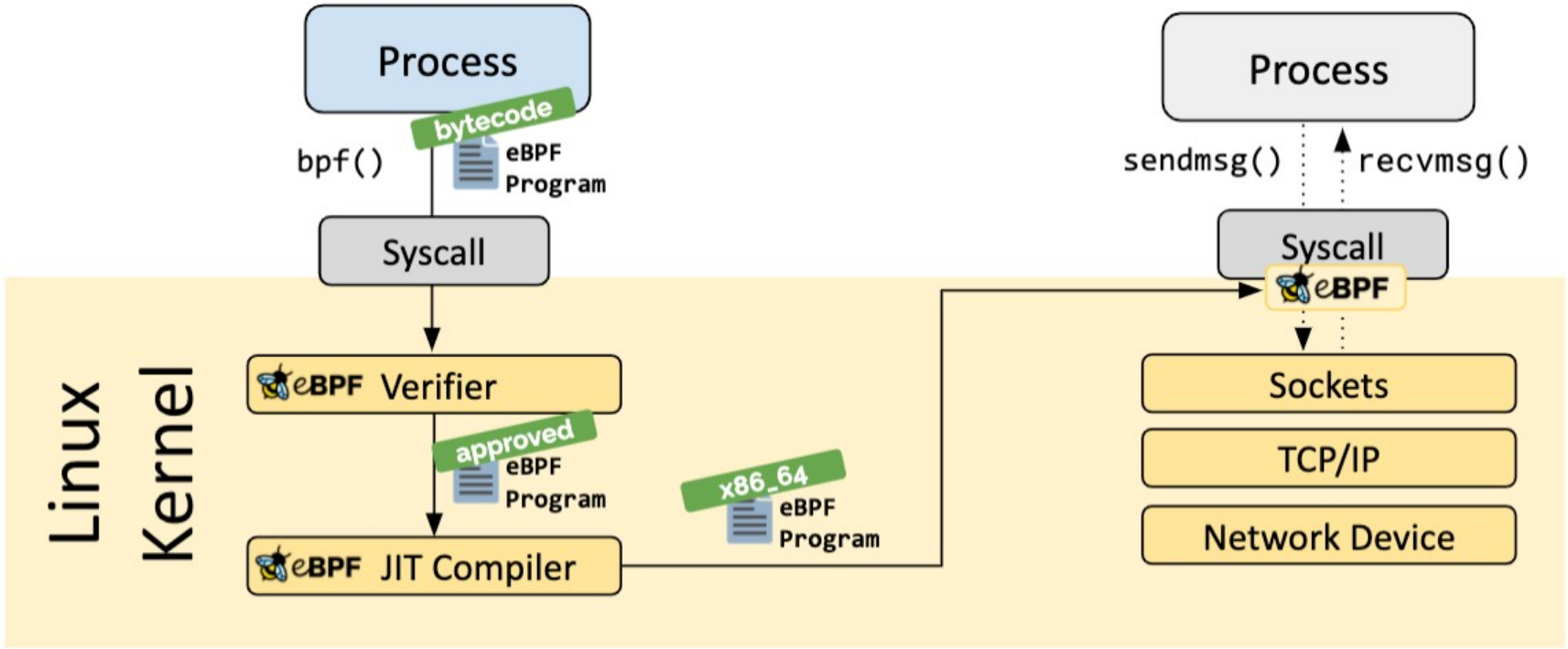
eBPF toolchain – LLVM IR

- The code gets processed by Clang, a compiler front end for C-style programming languages
  - There should also be support for eBPF on GCC, but Clang is the most used

- LLVM converts the C code into an Intermediate Representation (IR)
  - Performs several optimizations
  - Generates the final eBPF assembly



eBPF runtime



The runtime **accepts** bytecode, **verifies** it, **just-in-time compiles** it, and **runs** it at the requested hook point

Who controls  eBPF ?

eBPF Foundation

Founding Members

FACEBOOK

Google

ISOVALENT™

Microsoft

NETFLIX

Where is  eBPF used today?

Cloud-native eBPF landscape

Application Observability

pixie

bc

Inspektor Gadget

Networking & Service Mesh

cilium

PROJECT CALICO

Security

Tetragon

Tracee

Falco

Hyperscalers using eBPF

Meta

katran

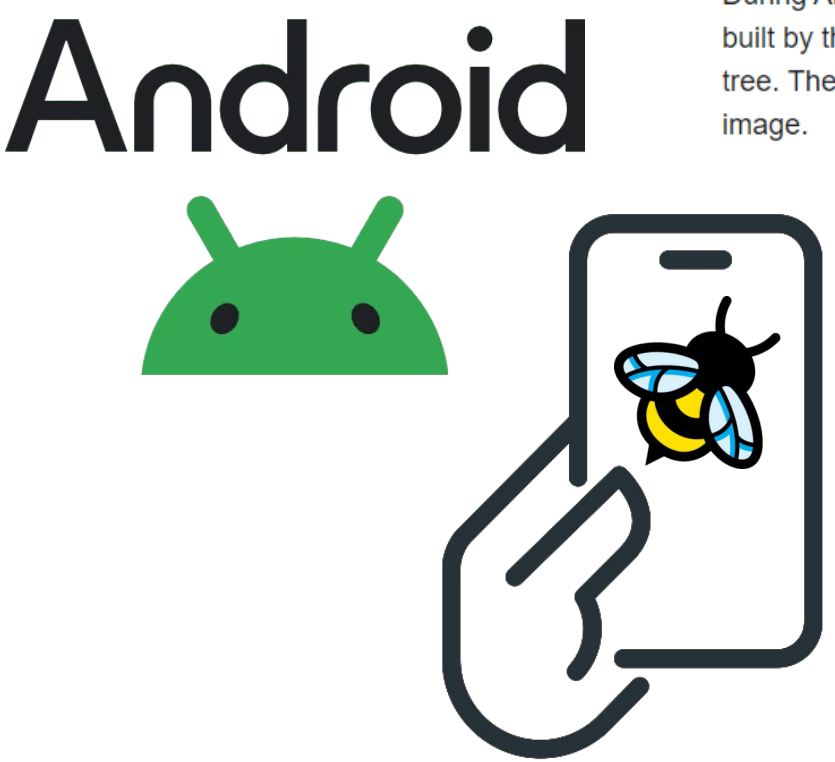
https://github.com/facebookincubator/katran

Fast, eBPF-based L4 load-balancer used at Facebook/Meta

Cloud providers using eBPF

All major cloud providers have picked eBPF-based Networking & Security for their Kubernetes platforms

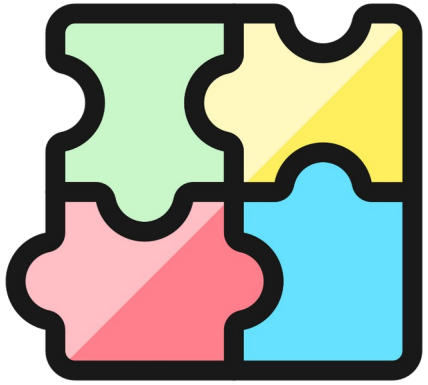
Smartphones with eBPF



Android BPF loader

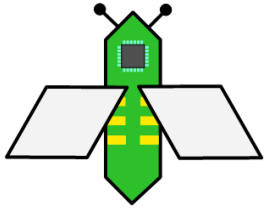
During Android boot, all eBPF programs located at `/system/etc/bpf/` are loaded. These programs are binary objects built by the Android build system from C programs and are accompanied by `Android.bp` files in the Android source tree. The build system stores the generated objects at `/system/etc/bpf`, and those objects become part of the system image.

*
https://source.android.com/docs/core/architecture/kern

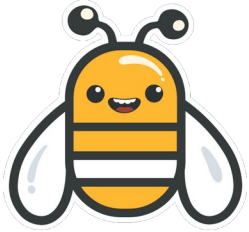


For windows

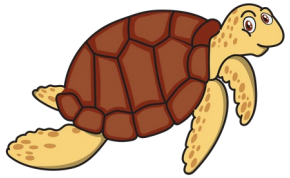
More eBPF Projects



hBPF:
eBPF in hardware



BumbleBee
OCI
compliant
eBPF tooling



Caretta
eBPF based Kubernetes
service map

Layer 4 Kubernetes LB

A system daemon and
Kubernetes operator
for managing eBPF
programs

DeepFlow
Highly Automated
Observability Platform
powered by eBPF

eunomia-bpf
eBPF programs in a
WASM module or
JSON

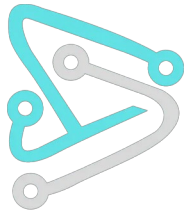
eCapture
SSL/TLS capture tool
using eBPF

Kindling
eBPF-based Cloud
Native Monitoring
& Profiling Tool

More eBPF Projects

KubeArmor
Container-aware Runtime Security Enforcement System

L3AF
Complete lifecycle management of eBPF programs



LoxiLB
eBPF based cloud-native load-balancer for 5G Edge

Pulsar
A modular runtime security framework for the IoT

pwru
eBPF-based Linux kernel network packet tracer

Merbridge
Use eBPF to speed up your Service Mesh like crossing an Einstein-Rosen Bridge

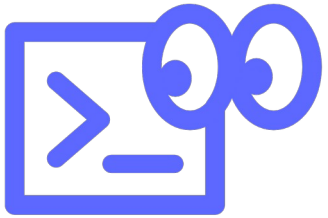
Parca
Continuous Profiling Platform

ply
A dynamic tracer for Linux

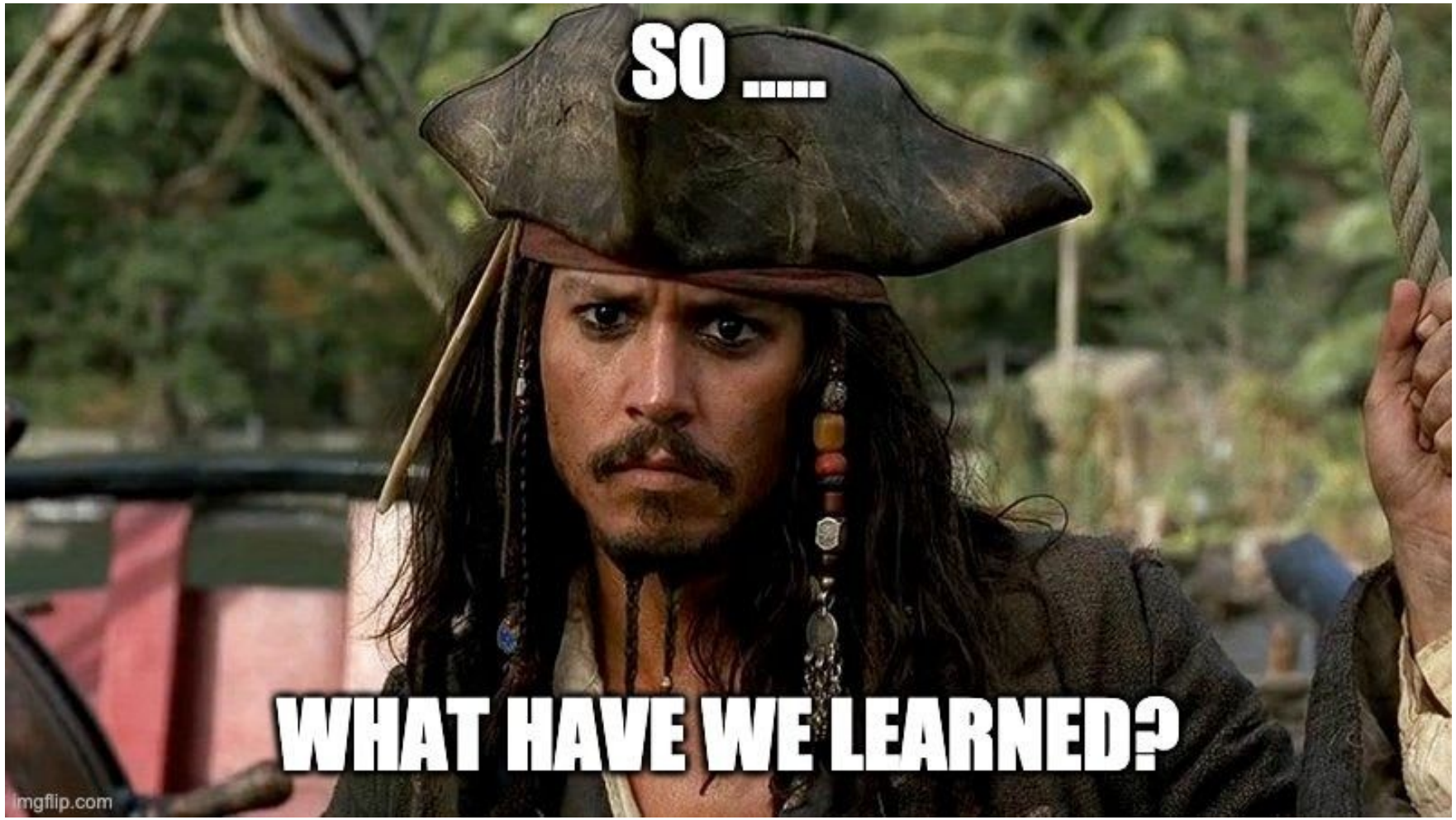
More eBPF Projects

Pyroscope
Continuous Profiling Platform

wachy
UI for interactive eBPF-based userspace performance debugging



SSHLog
eBPF SSH session monitoring


SO .....
WHAT HAVE WE LEARNED?
imgflip.com

eBPF is a cool technology

It allowed us to do something that was unthinkable before.

Operating Systems can now be programmed, at runtime!

is not just Networking

...but also...

Observabil
ity

Securi
ty

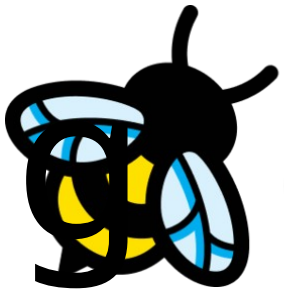
Stora
ge

Scheduli
ng

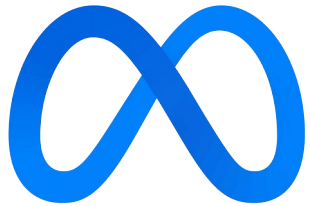
TETRIS

- https://github.com/mmisono/bpftrace-tetris

Who is using eBPF



Hyperscalers

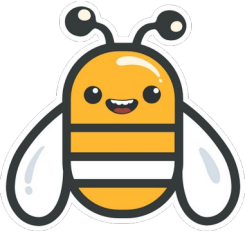


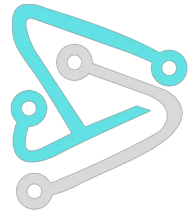
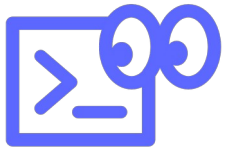
Cloud-providers

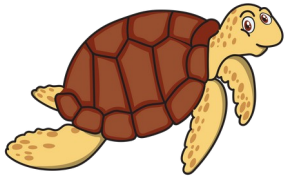
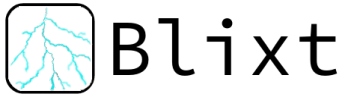
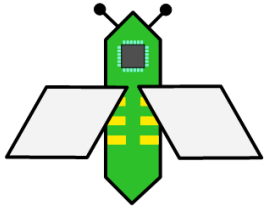
Projects & Startups

eBPF Day





L3AF





Blixt


parca
ply
/* a dynamic tracer for Linux */

bpfd




EUNOMIA

...but the most important thing is that...

...although Sebastiano started working with eBPF from the beginning...


I AM THE ONE WHO IS POOR
amc