第二届 eBPF开发者大会
www.ebpftravel.com

Pipy 与 eBPF
重塑系统级编程的新范式

张晓辉

个人介绍

张晓辉 Flomesh 架构师/技术布道师

资深程序员，CNCF Ambassador，LFAPAC 开源布道师，云原生社区管委会成员，微软 MVP，公众号“云原生指北”作者。多年的微服务和云原生实践经验，主要工作涉及微服务、容器、Kubernetes、DevOps 等。

一、eBPF 工作场景

eBPF 场景

将 eBPF 程序附加到跟踪点以及内核和用户应用探针点的能力，使得应用程序和系统本身的运行时行为具有前所未有的可见性。eBPF 不依赖于操作系统暴露的静态计数器和测量，而是实现了自定义指标的收集和内核内聚合，并基于广泛的可能来源生成可见性事件。

可观测

eBPF 场景

将 eBPF 程序附加到跟踪点以及内核和用户应用探针点的能力，使得应用程序和系统本身的运行时行为具有前所未有的可见性。eBPF 不依赖于操作系统暴露的静态计数器和测量，而是实现了自定义指标的收集和内核内聚合，并基于广泛的可能来源生成可见性事件。

可观测

看到和理解所有系统调用的基础上，将其与所有网络操作的数据包和套接字级视图相结合，可以采用革命性的新方法来确保系统的安全。通过检测和阻止恶意行为，如 DDoS攻击、网络钓鱼等，实施网络策略、可以增强系统的安全性和稳定性。

安全

eBPF 场景

将 eBPF 程序附加到跟踪点以及内核和用户应用探针点的能力，使得应用程序和系统本身的运行时行为具有前所未有的可见性。eBPF 不依赖于操作系统暴露的静态计数器和测量，而是实现了自定义指标的收集和内核内聚合，并基于广泛的可能来源生成可见性事件。

可观测

看到和理解所有系统调用的基础上，将其与所有网络操作的数据包和套接字级视图相结合，可以采用革命性的新方法来确保系统的安全。通过检测和阻止恶意行为，如 DDoS攻击、网络钓鱼等，实施网络策略、可以增强系统的安全性和稳定性。

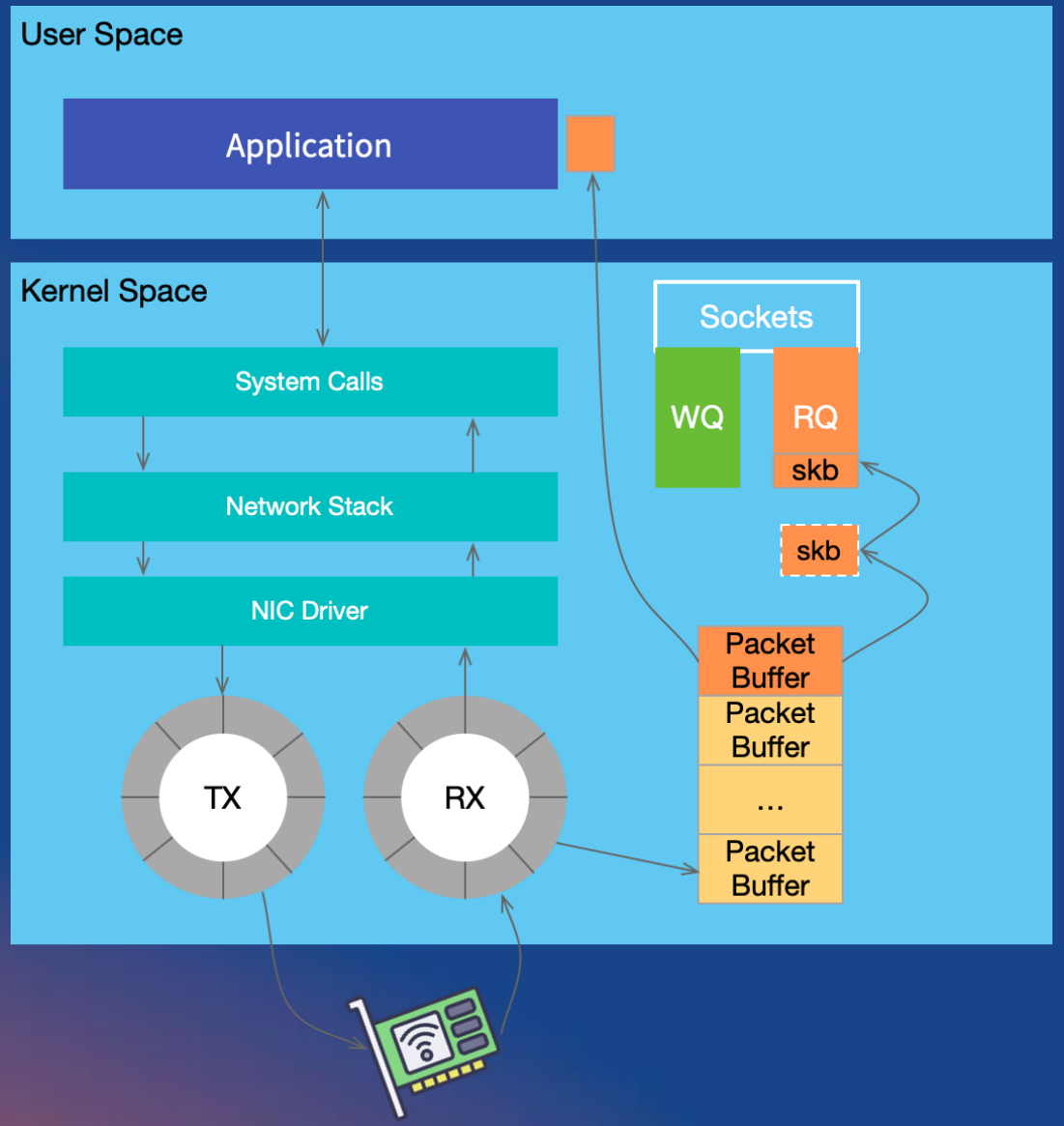
安全

eBPF 的可编程性使其能够在不离开 Linux 内核的包处理上下文的情况下，添加额外的协议解析器，并轻松编程任何转发逻辑以满足不断变化的需求。JIT 编译器提供的效率使其执行性能接近于本地编译的内核代码。

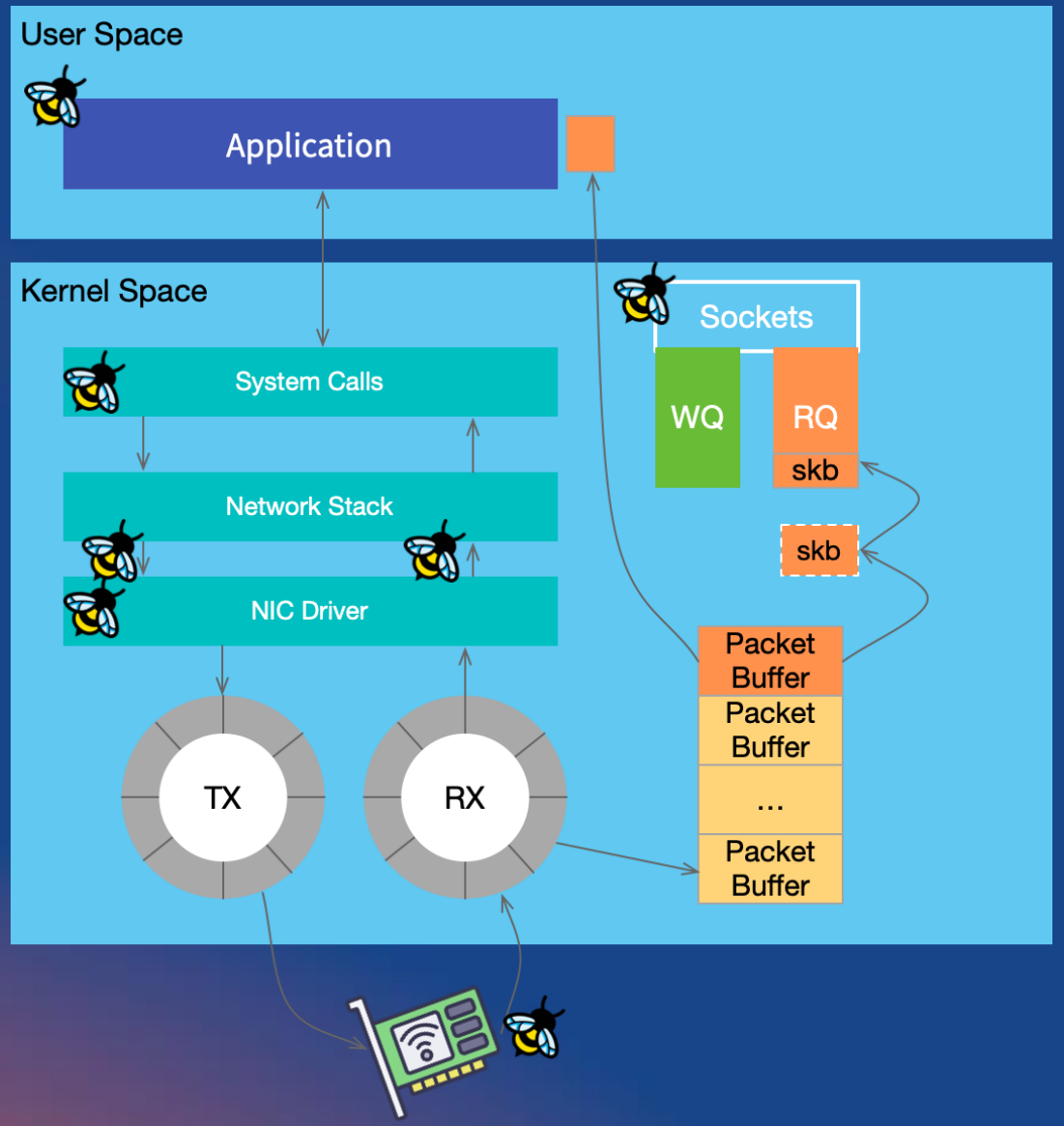
网络

二、eBPF 与网络

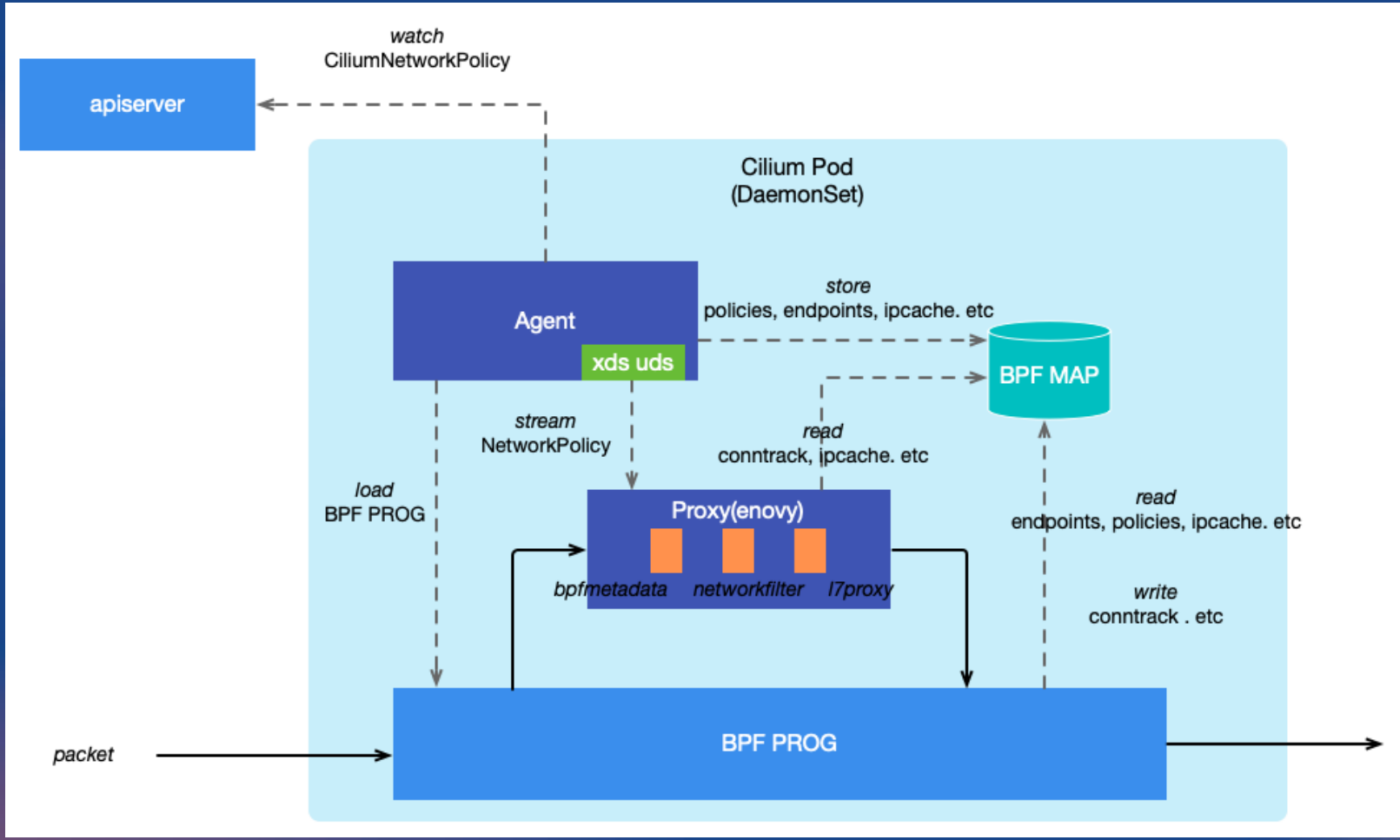
网络数据路径


User Space
Application
Kernel Space
System Calls
Network Stack
NIC Driver
TX
RX
Sockets
WQ
RQ
skb
skb
Packet Buffer
Packet Buffer
...
Packet Buffer

eBPF in Data Path



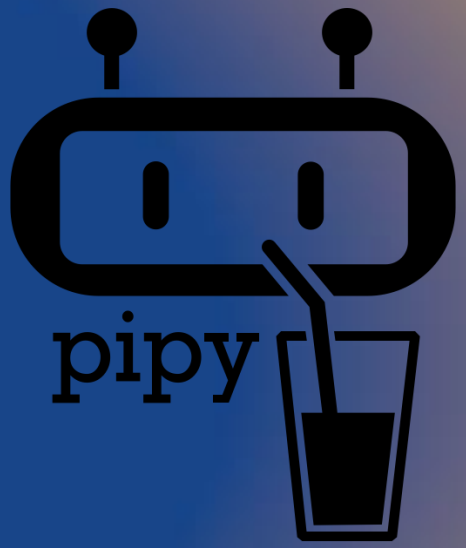
Cilium L3/4/7



多组件协同

三、eBPF 与 Pipy

可编程应用引擎 Pipy



◆灵活多变 提供了一系列可插拔的组件，又称过滤器，同时对组合的方式没有限制。

◆快 C++ 开发，使用了异步网络；分配的资源被池化并可复用；内部尽可能地使用指针传递数据，最大限度降低内存带宽的压力。

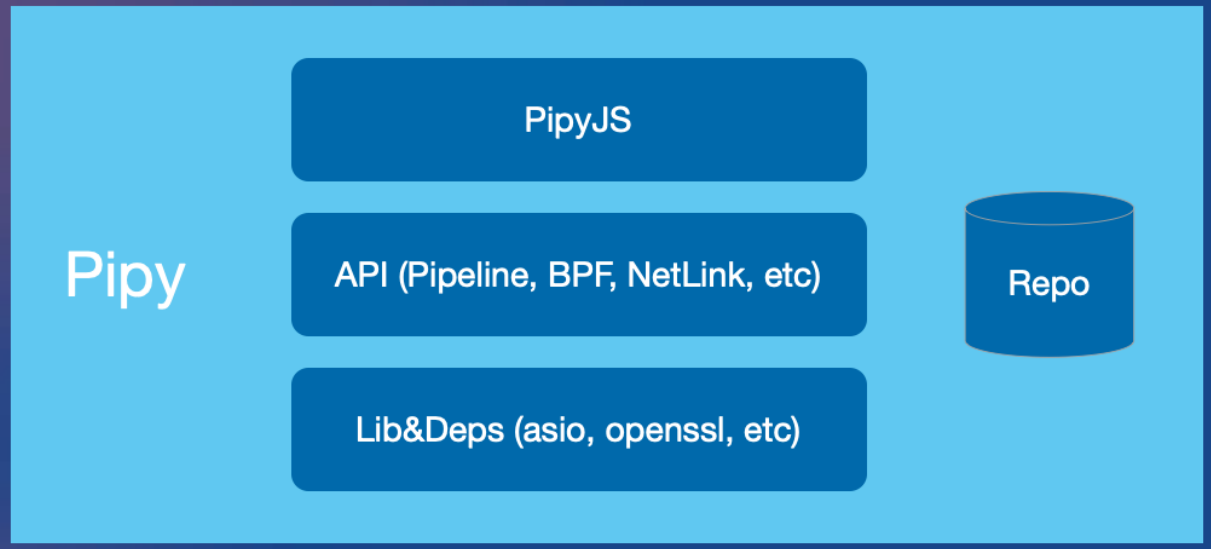
◆小 用于工作节点的可执行文件仅 15MB 左右，并且没有任何外部依赖。

◆可编程 运行 PipyJS（标准 JavaScript 的定制版本） 的脚本引擎。

◆开放 比开源更加开放，不隐藏任何细节，但使用无需了解。

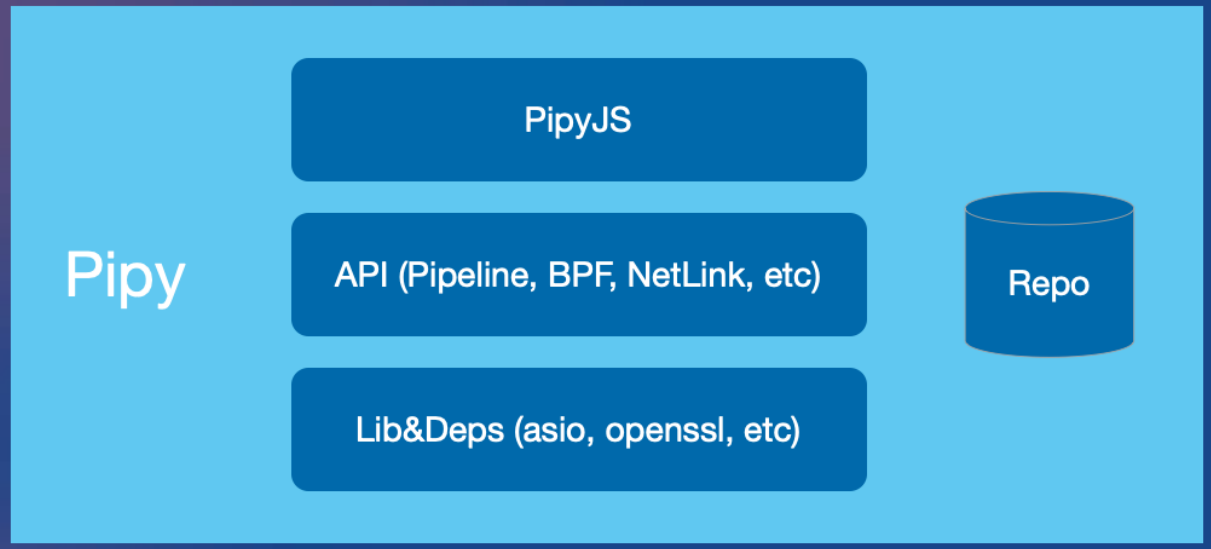
https://github.com/flomesh-io/pipy

Pipy 架构



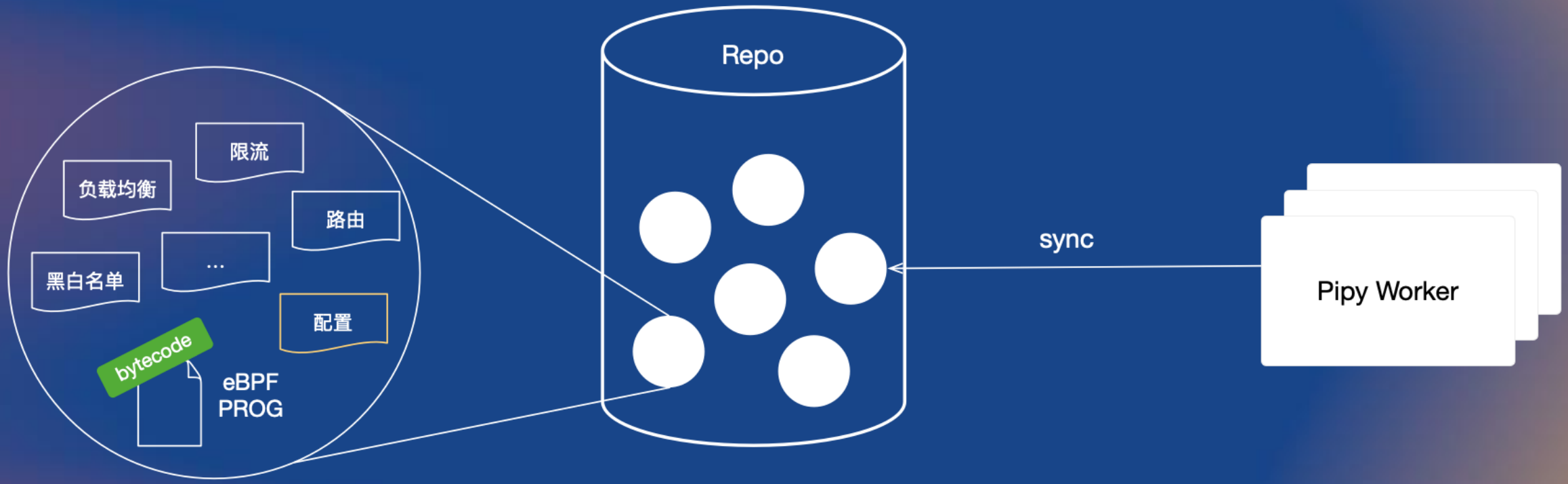
- 上层：使用核心层的能力，对流量进行编程
- 中层：核心层，提供流处理的 API、可插拔组件
- 底层：依赖极少，轻量、高效

可编程



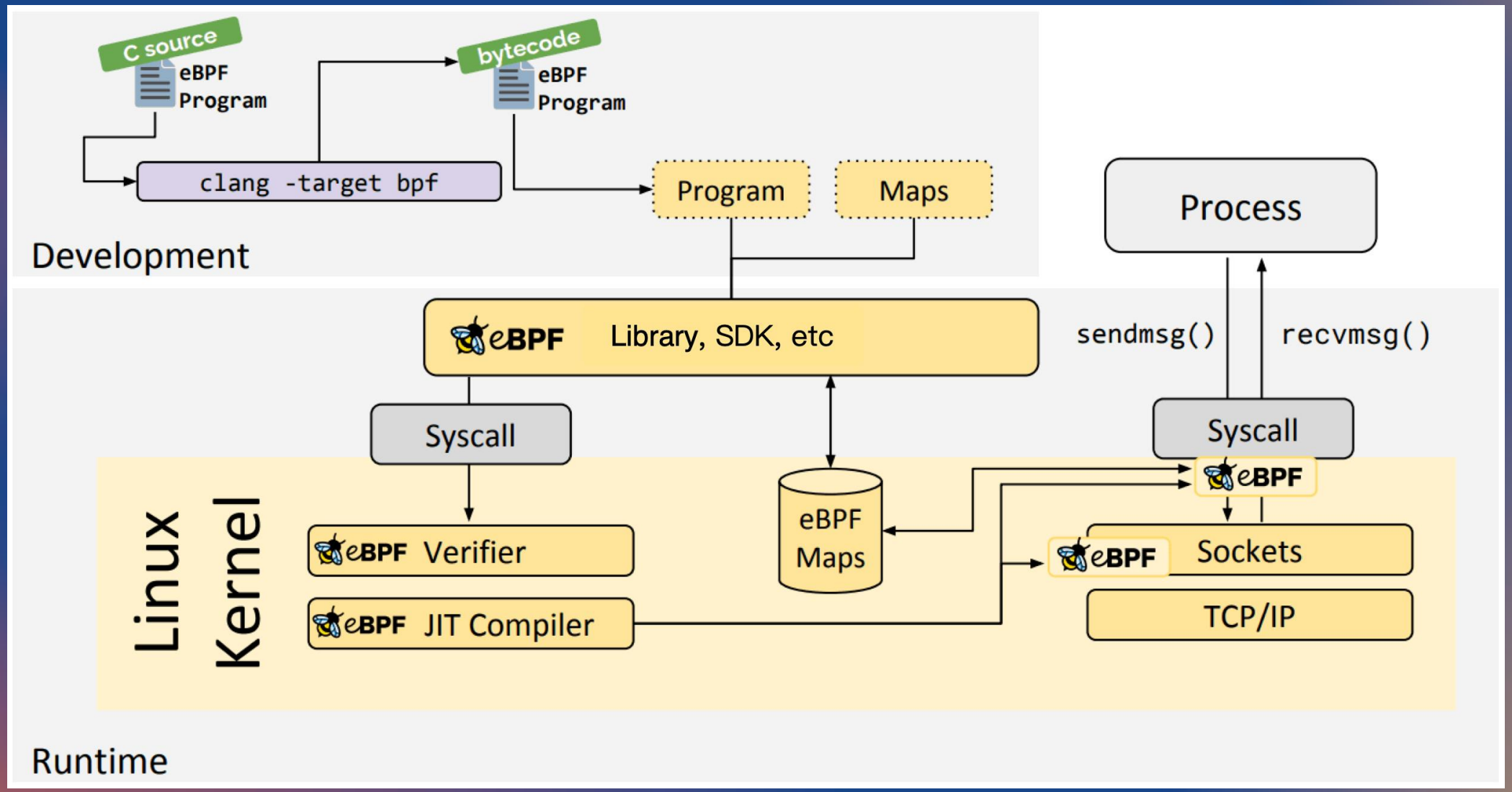

```
var router = new algo.URLRouter({
  '/': new algo.LoadBalancer(
    ['127.0.0.1:8080', '127.0.0.1:8082'],
    { algorithm: 'round-robin' }
    )
})

var $target
pipy.listen(8000, $ => $
  .demuxHTTP().to($ => $
    .handleMessageStart(
      function (msg) {
        $target = router.find(
          msg.head.headers.hosts,
          msg.head.path,
        ).allocate().target
            }
        )
    .muxHTTP(() => new Object).to($ => $
      .connect(() => $target)
        )
    )
)
```

Pipy Repo

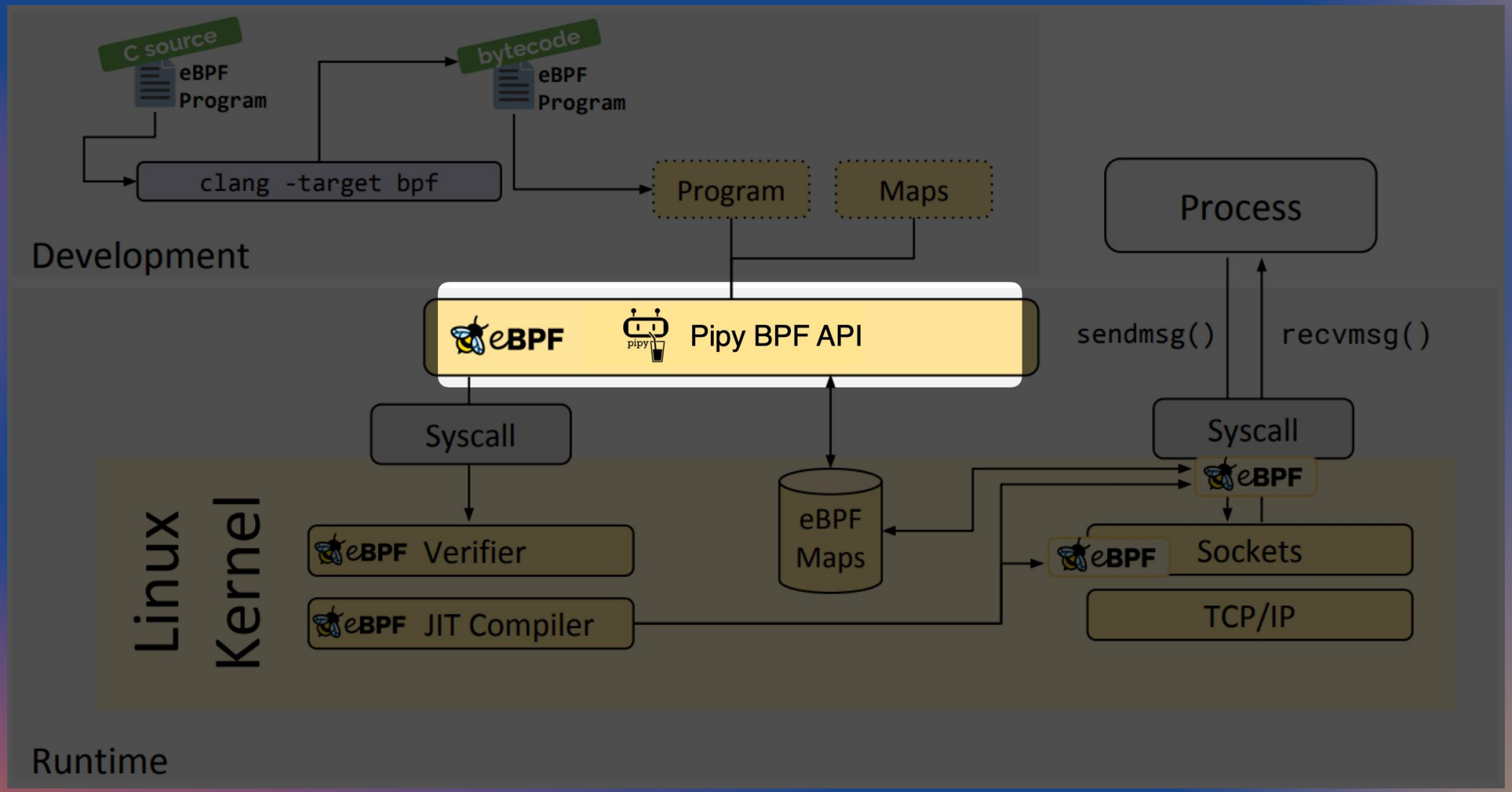


类似 GitOps

使用 eBPF


C source
eBPF
Program
bytecode
eBPF
Program
clang -target bpf
Program
Maps
Development
Process
eBPF Library, SDK, etc
sendmsg()
recvmsg()
Syscall
Syscall
Linux
Kernel
eBPF
eBPF Verifier
eBPF
Maps
eBPF
eBPF Sockets
eBPF JIT Compiler
TCP/IP
Runtime

使用 eBPF


C source
eBPF Program
bytecode
eBPF Program
clang -target bpf
Program
Maps
Development
Process
eBPF
Pipy BPF API
sendmsg()
recvmsg()
Syscall
Syscall
eBPF
eBPF Maps
eBPF Verifier
eBPF
eBPF Sockets
eBPF JIT Compiler
TCP/IP
Linux Kernel
Runtime

透明代理

```
var obj = bpf.object(pipy.load('transparent-proxy.o'))
var progCgConnect4 = obj.programs.find(p => p.name === 'cg_connect4').load('BPF_PROG_TYPE_CGROUP_SOCK_ADDR', 'BPF_CGROUP_INET4_CONNECT')
var progCgSockOps = obj.programs.find(p => p.name === 'cg_sock_ops').load('BPF_PROG_TYPE_SOCK_OPS')
var progCgSockOpt = obj.programs.find(p => p.name === 'cg_sock_opt').load('BPF_PROG_TYPE_CGROUP_SOCKOPT', 'BPF_CGROUP_GETSOCKOPT')
```

```
obj.maps.find(m => m.name === 'map_config').update(
  { i: 0 }, {
    proxy_port: PROXY_PORT,
    pipy_cgroup_id: bpf.cgroup(CGRP_PIPY)
     }
)
```

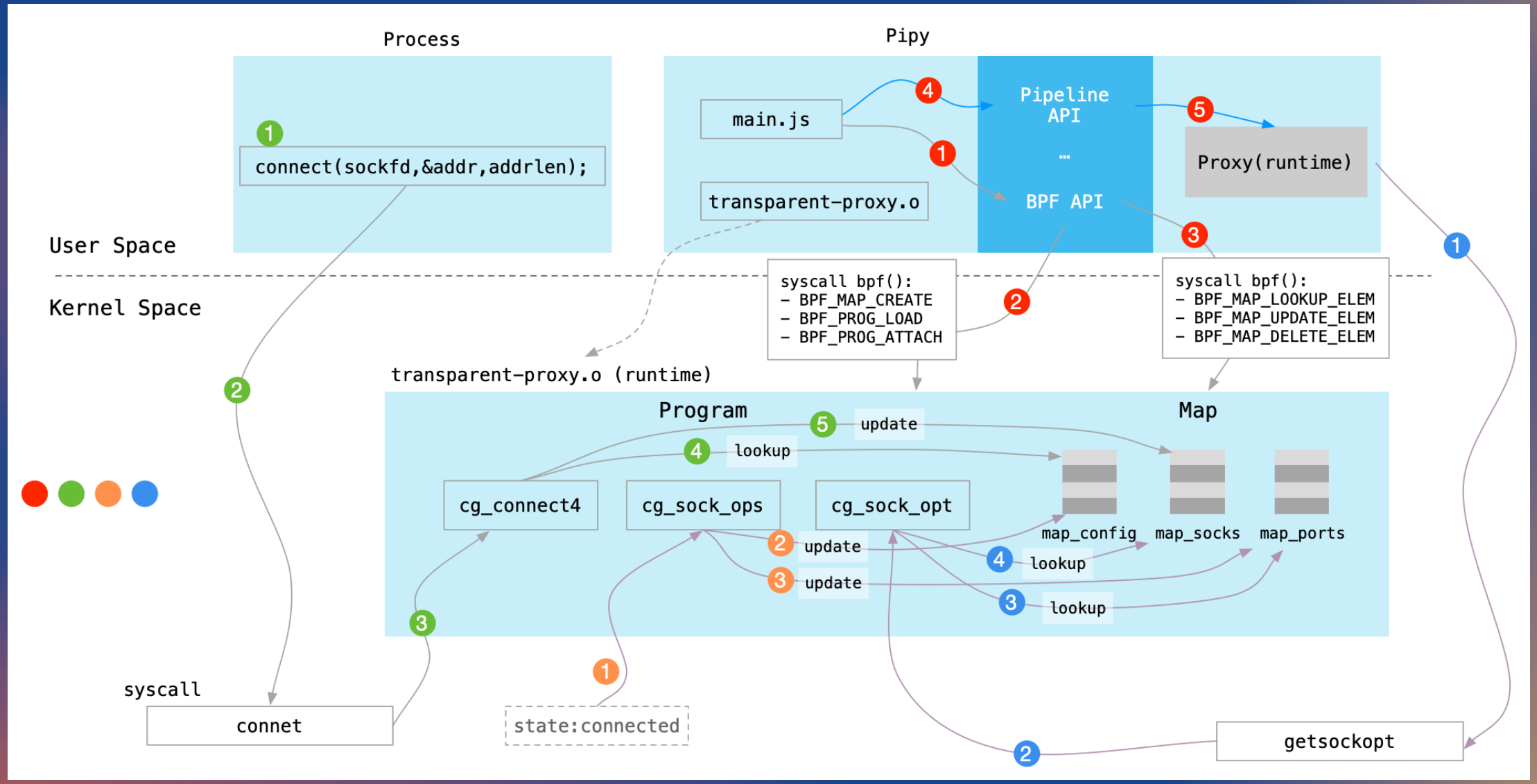
```
bpf.attach('BPF_CGROUP_INET4_CONNECT', progCgConnect4.fd, CGRP)
bpf.attach('BPF_CGROUP_SOCK_OPS', progCgSockOps.fd, CGRP)
bpf.attach('BPF_CGROUP_GETSOCKOPT', progCgSockOpt.fd, CGRP)
```

```
pipy.exit(
  function() {
    bpf.detach('BPF_CGROUP_INET4_CONNECT', progCgConnect4.fd, CGRP)
    bpf.detach('BPF_CGROUP_SOCK_OPS', progCgSockOps.fd, CGRP)
    bpf.detach('BPF_CGROUP_GETSOCKOPT', progCgSockOpt.fd, CGRP)
    os.write(`${CGRP}/cgroup.procs`, pipy.pid.toString())
    os.rmdir(CGRP_PIPY)
     }
)
```

```
pipy.listen(PROXY_PORT, $⇒$
  .onStart(
    function (ib) {
      var od = new Data
      ib.socket.getRawOption(SOL_IP, SO_ORIGINAL_DST, od)
      var sa = sockaddr_in.decode(od)
      var addr = sa.sin_addr
      var port = sa.sin_port
      $targetAddr = addr.join('.')
      $targetPort = (port[0] << 8) | port[1]
        }
    )
  .fork().to($⇒$
    .decodeHTTPRequest()
    .handleMessageStart(
      ({ head }) ⇒ println(head.method, head.path, head.headers.host)
        )
    )
  .connect(() ⇒ `${$targetAddr}:${$targetPort}`)
  .fork().to($⇒$
    .decodeHTTPResponse()
    .handleMessageStart(
      ({ head }) ⇒ println(' ', head.status, head.statusText)
        )
    )
)
```

```
#!/usr/bin/env pipy

var obj = bpf.object(pipy.load('transparent-proxy.o'))
var progCgConnect4 = obj.programs.find(p ⇒ p.name === 'cg_connect4').load('BPF_PROG_TYPE_CGROUP_SOCK_ADDR', 'BPF_CGROUP_INET4_CONNECT')
var progCgSockOps = obj.programs.find(p ⇒ p.name === 'cg_sock_ops').load('BPF_PROG_TYPE_SOCK_OPS')
var progCgSockOpt = obj.programs.find(p ⇒ p.name === 'cg_sock_opt').load('BPF_PROG_TYPE_CGROUP_SOCKOPT', 'BPF_CGROUP_GETSOCKOPT')

var PROXY_PORT = 18000
var CGRP = '/sys/fs/cgroup'
var CGRP_PIPY = `${CGRP}/pipy`
var SOL_IP = 0
var SO_ORIGINAL_DST = 80

var sockaddr_in = new CStruct({
  sin_family: 'uint16',
  sin_port: 'uint8[2]',
  sin_addr: 'uint8[4]',
})

os.mkdir(CGRP_PIPY)
os.write(`${CGRP_PIPY}/cgroup.procs`, pipy.pid.toString())

obj.maps.find(m ⇒ m.name === 'map_config').update(
  { i: 0 }, {
    proxy_port: PROXY_PORT,
    pipy_cgroup_id: bpf.cgroup(CGRP_PIPY)
     }
 )

bpf.attach('BPF_CGROUP_INET4_CONNECT', progCgConnect4.fd, CGRP)
bpf.attach('BPF_CGROUP_SOCK_OPS', progCgSockOps.fd, CGRP)
bpf.attach('BPF_CGROUP_GETSOCKOPT', progCgSockOpt.fd, CGRP)

pipy.exit(
  function() {
    bpf.detach('BPF_CGROUP_INET4_CONNECT', progCgConnect4.fd, CGRP)
    bpf.detach('BPF_CGROUP_SOCK_OPS', progCgSockOps.fd, CGRP)
    bpf.detach('BPF_CGROUP_GETSOCKOPT', progCgSockOpt.fd, CGRP)
    os.write(`${CGRP}/cgroup.procs`, pipy.pid.toString())
    os.rmdir(CGRP_PIPY)
     }
 )

var $targetAddr
var $targetPort

pipy.listen(PROXY_PORT, $⇒$
  .onStart(
    function (ib) {
      var od = new Data
      ib.socket.getRawOption(SOL_IP, SO_ORIGINAL_DST, od)
      var sa = sockaddr_in.decode(od)
      var addr = sa.sin_addr
      var port = sa.sin_port
      $targetAddr = addr.join('.')
      $targetPort = (port[0] << 8) | port[1]
        }
    )
  .fork().to($⇒$
    .decodeHTTPRequest()
    .handleMessageStart(
      ({ head }) ⇒ println(head.method, head.path, head.headers.host)
        )
    )
  .connect(() ⇒ `${$targetAddr}:${$targetPort}`)
  .fork().to($⇒$
    .decodeHTTPResponse()
    .handleMessageStart(
      ({ head }) ⇒ println(' ', head.status, head.statusText)
        )
    )
)
```

透明代理


Process
Pipy
connect(sockfd,&addr,addrlen);
main.js
Pipeline API
...
BPF API
Proxy(runtime)
transparent-proxy.o
User Space
Kernel Space
syscall bpf():
- BPF_MAP_CREATE
- BPF_PROG_LOAD
- BPF_PROG_ATTACH
syscall bpf():
- BPF_MAP_LOOKUP_ELEM
- BPF_MAP_UPDATE_ELEM
- BPF_MAP_DELETE_ELEM
transparent-proxy.o (runtime)
Program
Map
update
lookup
cg_connect4
cg_sock_ops
cg_sock_opt
map_config
map_socks
map_ports
update
update
lookup
lookup
syscall
connet
state:connected
getsockopt

更多示例

- 网络包统计：https://github.com/flomesh-io/pipy/tree/main/samples/bpf/packet-counter
- 四层负载均衡：https://github.com/flomesh-io/pipy/tree/main/samples/bpf/load-balancer
- 端口拦截：https://github.com/flomesh-io/pipy/tree/main/samples/bpf/port-interceptor
- 透明代理：https://github.com/flomesh-io/pipy/tree/main/samples/bpf/transparent-proxy

Netlink 与 eBPF

```
ip link set dev eth0 xdp obj bpf_prog.o sec xdp_section
```

```
netlinkRequests.enqueue(
  new Message(
          {
      type: 19, // RTM_SETLINK
      flags: 0x01 | 0x04, // NLM_F_REQUEST | NLM_F_ACK
    },
    encodeLink({
      index,
      attrs: {
        [43]: { // IFLA_XDP
          [1]: i32Struct.encode({ i: fd }), // IFLA_XDP_FD
          [3]: i32Struct.encode({ i: 1 << 1 }), // IFLA_XDP_FLAGS: XDP_FLAGS_SKB_MODE
        },
             }
    })
     )
)
```

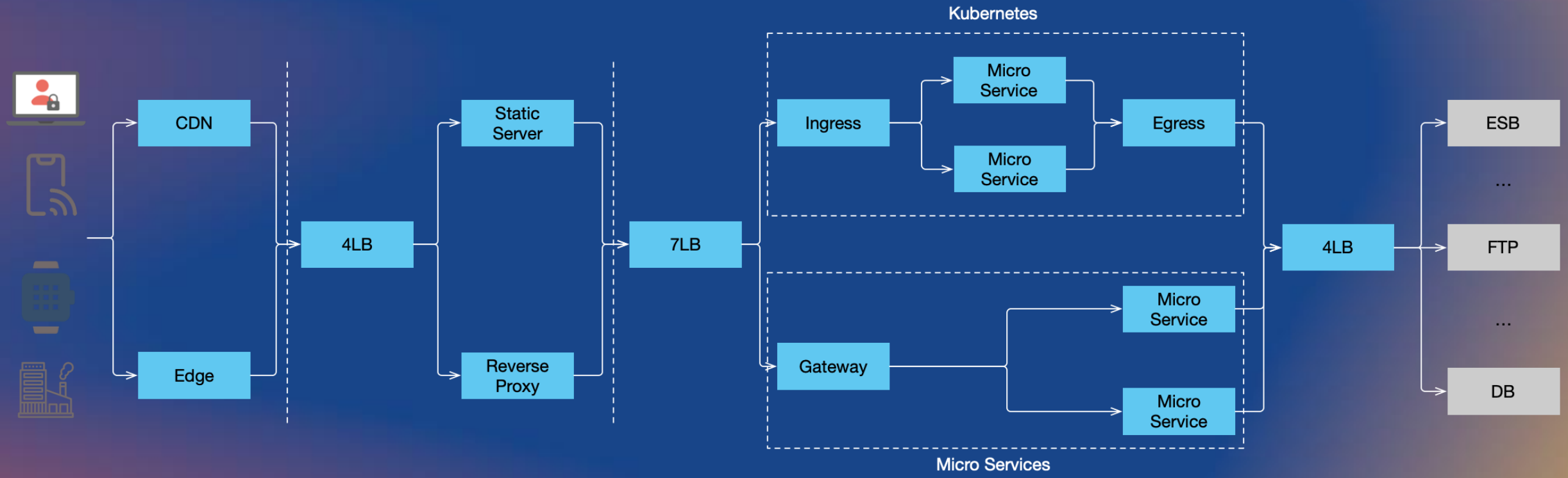
四、更多

Why eBPF and Pipy?

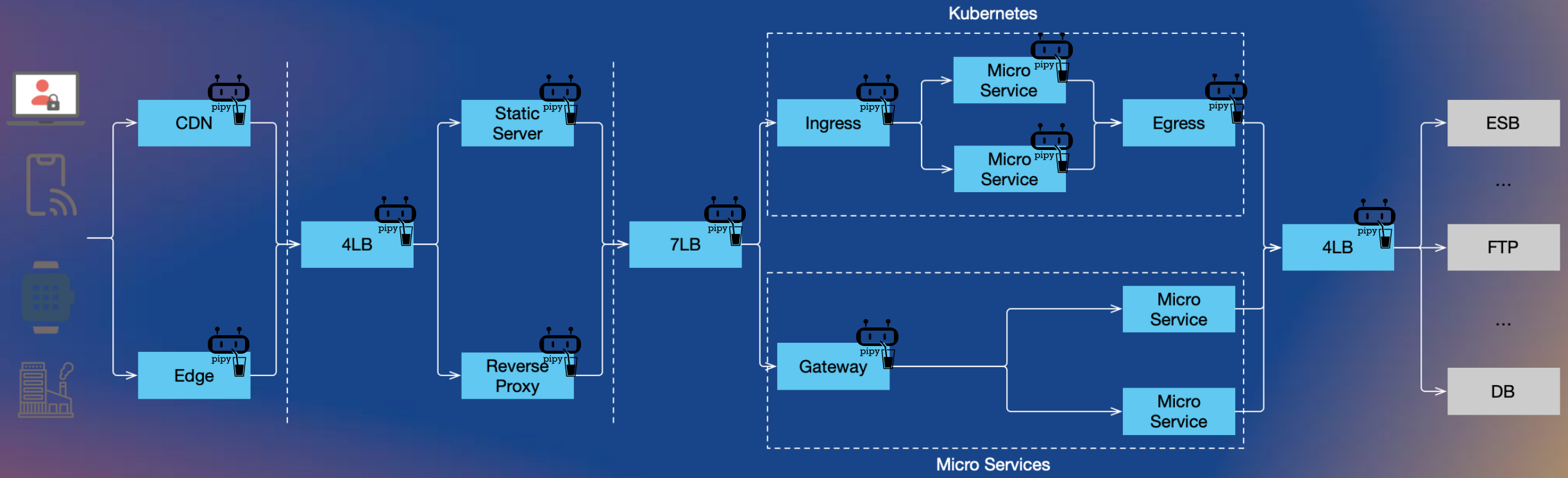
Dynamically program the kernel for efficient networking, observability, tracing, and security.

High performance **programmable** ~~proxy~~ engine for the cloud, edge and IoT.

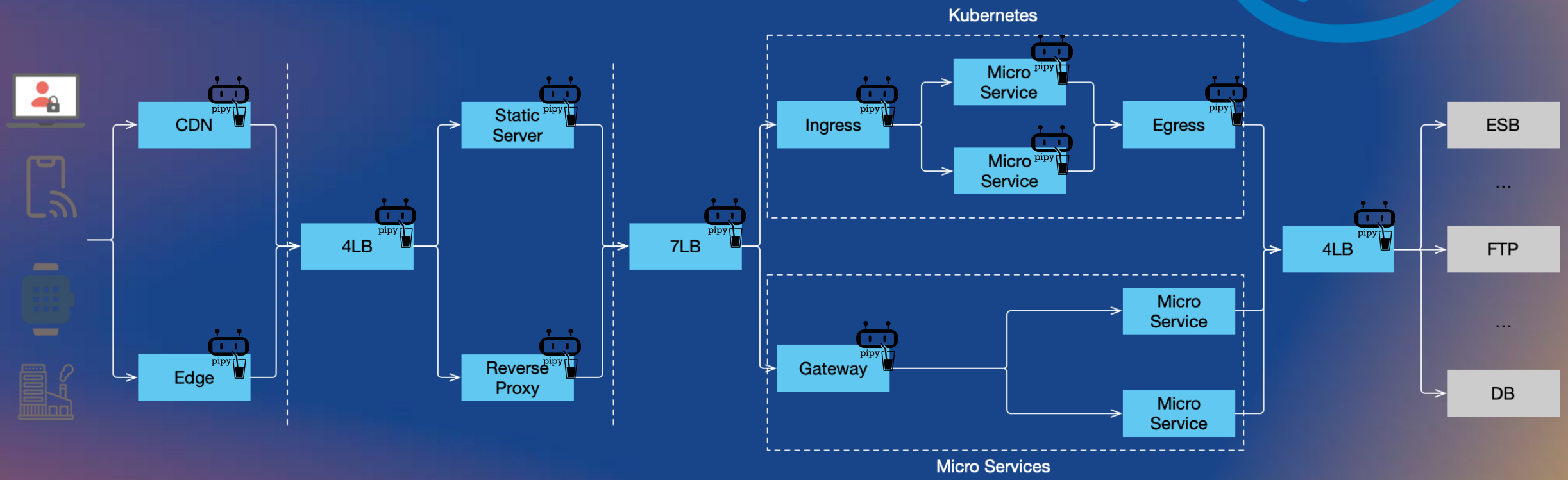
Pipy Inside


Kubernetes
CDN
Edge
4LB
Static Server
Reverse Proxy
7LB
Ingress
Micro Service
Micro Service
Egress
Gateway
Micro Service
Micro Service
Micro Services
4LB
ESB
...
FTP
...
DB

Pipy Inside


Kubernetes
CDN
Edge
4LB
Static Server
Reverse Proxy
7LB
Ingress
Micro Service
Micro Service
Egress
Gateway
Micro Service
Micro Service
Micro Services
4LB
ESB
...
FTP
...
DB

Pipy Inside


Kubernetes
CDN
Edge
4LB
Static Server
Reverse Proxy
7LB
Ingress
Micro Service
Micro Service
Egress
Gateway
Micro Service
Micro Service
Micro Services
4LB
ESB
...
FTP
...
DB

Flomesh flomesh.io

github.com/flomesh-io

flomesh-io.slack.com