



首届中国eBPF研讨会

[www.ebpftravel.com](http://www.ebpftravel.com)

# 基于eBPF的多路径 网络传输协议栈扩展

沈 典

东南大学计算机科学与工程学院  
江苏省网络与信息安全重点实验室  
[dshen@seu.edu.cn](mailto:dshen@seu.edu.cn)



01

# 背景





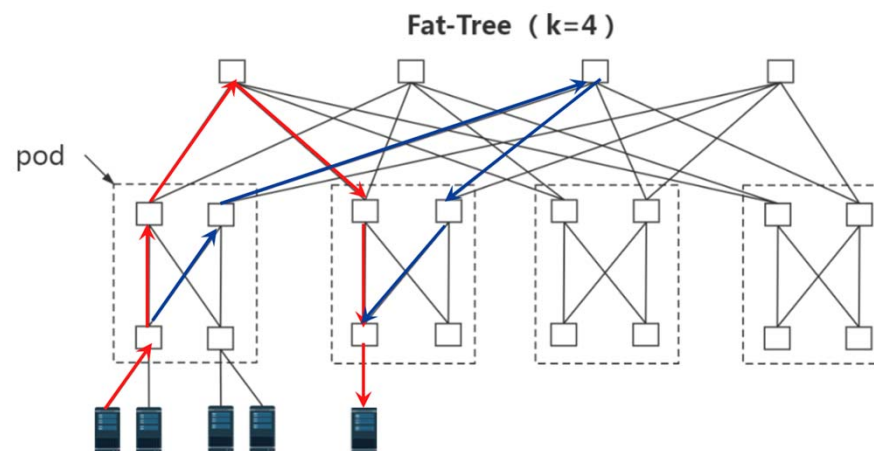
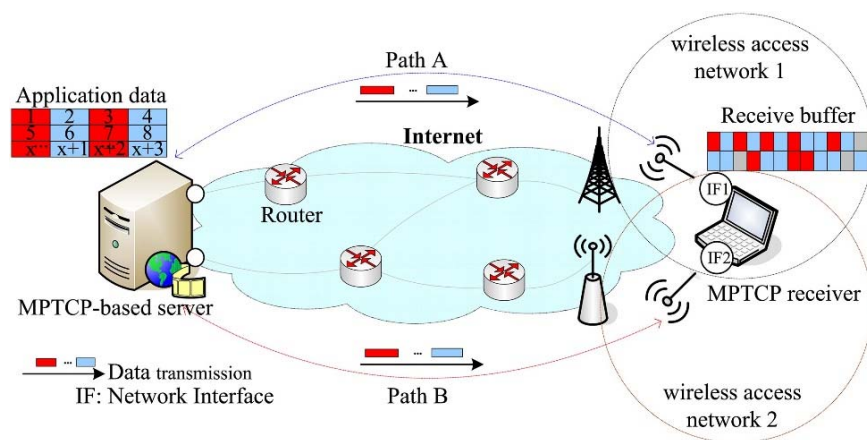
## 01 背景

### □ 当今的网络是多路径的

- 移动设备通常有多个无线的网络接口，例如wifi和蜂窝网络。
- 数据中心服务器有多张网卡，服务器之间存在多条并行的传输路径。



### 如何更好地利用多路径？





## 01 背景

首届中国eBPF研讨会

### □ Multipath TCP (MPTCP)



Olivier  
Bonaventure

**SIGCOMM 2010**

提出一种MPTCP的  
Linux实现



**RFC 8684 MPTCPV1**，在V0基  
础上完善了ADD\_ADDR，  
MP\_CAPABLE选项，增加了其它  
选项。

2013.1



**RFC 6824**。IETF第一次将  
MPTCP协议标准化，即  
**MPTCPV0**

2020.3

2020.3.29



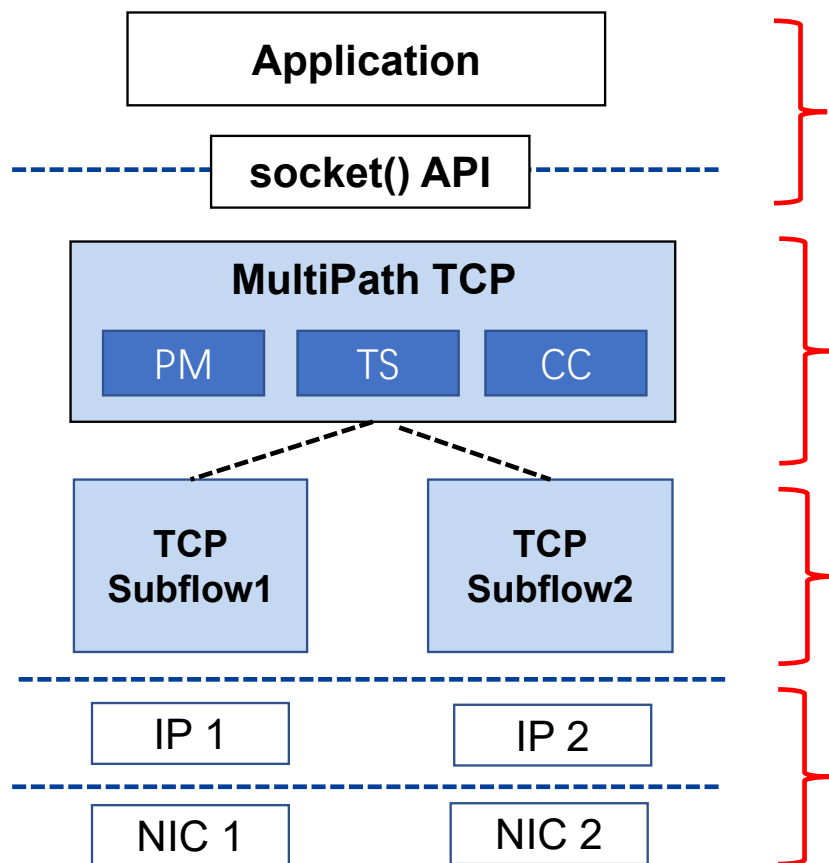
**Linux 5.6** 开始在内核中实  
现RFC 8684，即MPTCPV1

2010



## 01 背景

### □ MPTCP内核协议栈



### 标准Socket API

MPTCP协议栈使用标准的Socket API, 对用户层透明可以像使用TCP一样使用MPTCP。

### 多路径控制

PM : 路径管理, 控制路径建立  
TS : 流量调度, 控制发包子流  
CC : 多路径联合拥塞控制

### TCP子流

一条MPTCP连接可以有多条子流, 每一条子流是单独的TCP连接。

### 多地址

同时使用多个IP地址/接口并行传输。



## 01 背景

首届中国eBPF研讨会

### □ MPTCP优势

#### 对应用层透明

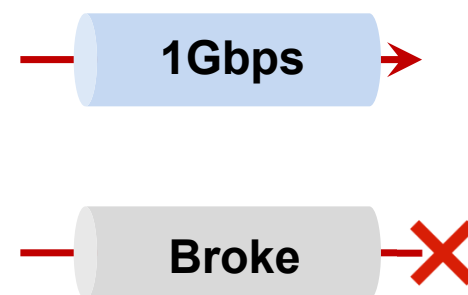
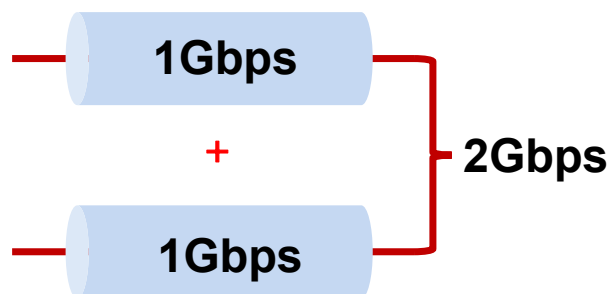
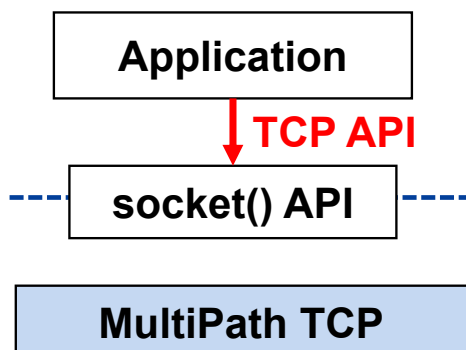
MPTCP使用**标准的socket API**，对用户层透明。让传统的TCP应用不需要修改代码即可使用MPTCP。

#### 提升带宽

MPTCP允许同时使用**多条TCP子流**并行传输数据，**使用多条底层链路**，有效提升聚合带宽。

#### 链路鲁棒性

MPTCP允许同时使用**多条TCP子流**，当其中一条子流失效，可以**使用其它子流**，无需重新建立连接。





## 01 背景

### □ MPTCP优化

为了**提升性能**和**扩展功能**，存在很多优化MPTCP的工作：

#### ■ 流量调度 (Traffic Scheduler)

Y.-s. Lim et al, “ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths,” CoNEXT '17.

#### ■ 路径管理 (Path Management)

M. Kheirkhah et al, “MMPTCP: A multipath transport protocol for data centers,” in IEEE INFOCOM 2016.

#### ■ 跨协议协同设计 (Network-application codeign)

F.LeandE.M.Nahum, “Experiences Implementing Live VM Migration over the WAN with Multi-Path TCP,” in IEEE INFOCOM 2019.

---

然而，当前MPTCP的设计和实现并不**有利于扩展**。



## 01 背景

首届中国eBPF研讨会

### □ MPTCP优化

为了**提升性能**和**扩展功能**，存在很多优化MPTCP的工作：

#### ■ 流量调度 (Traffic Scheduler)

Y.-s. Lim et al, "ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths," CoNEXT '17.

**问题：如何方便、安全地扩展多路径传输内核协议栈？**

F. LeandE.M. Nahum, "Experiences Implementing Live VM Migration over the WAN with Multi-Path TCP," in IEEE INFOCOM 2019.

然而，当前MPTCP的设计和实现并不**利于扩展**。

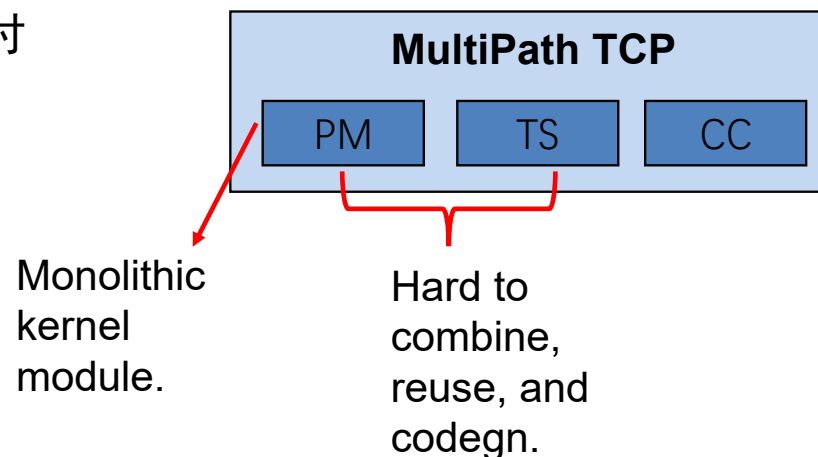




## 02 挑战

### ❑ 通过修改内核扩展MPTCP

- 正确地修改内核代码/实现内核模块需要付出**大量的时间和精力**。
- 难以针对变化的网络环境**动态地微调**模块 (例如流量调度模块)。
- 无法灵活地**复用、组合、协同设计**不同的协议模块。



---

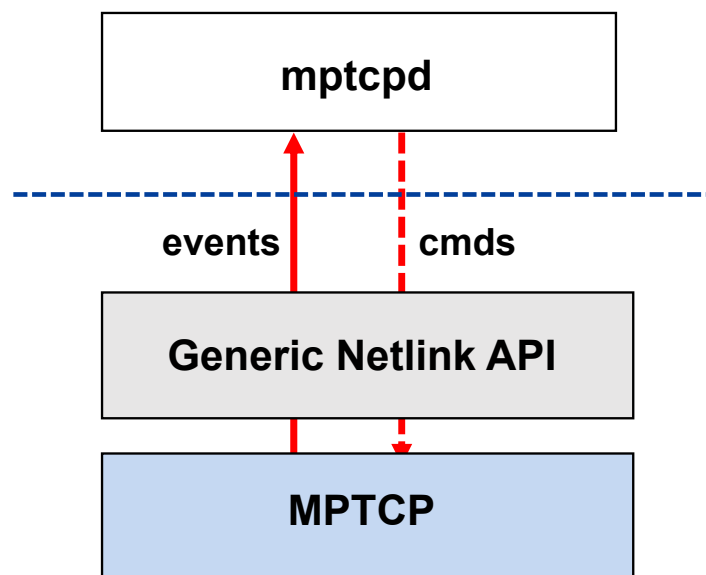
挑战1: **缺乏灵活性**



## 02 挑战

### □ 通过用户态控制模块优化MPTCP

- 功能性和可扩展性高度受限于MPTCP**协议栈暴露的接口** (Generic Netlink API)。
- 支持的MPTCP事件和命令依赖于协议栈实现 (例如 Linux 5.10 不支持MPTCP相关事件)。
- *mptcpd*直接和MPTCP协议栈交互，只能部署在终端 (无法部署在中间节点)。



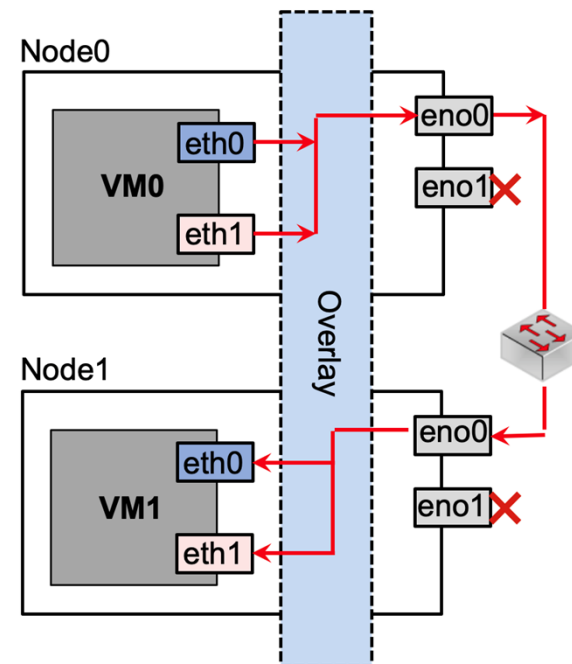
挑战2: **功能性受限。**



## 02 挑战

### ❑ 通过用户态控制模块优化MPTCP

- 目前，MPTCP及其扩展都是**基于终端** (end-host) 的解决方案。
- 受限的**底层网络信息**和控制能力。
- 应对新型场景，例如**多租户场景**、云原生场景的能力有限。



multi-tenant难以感知底层链路

挑战2: **功能性受限。**

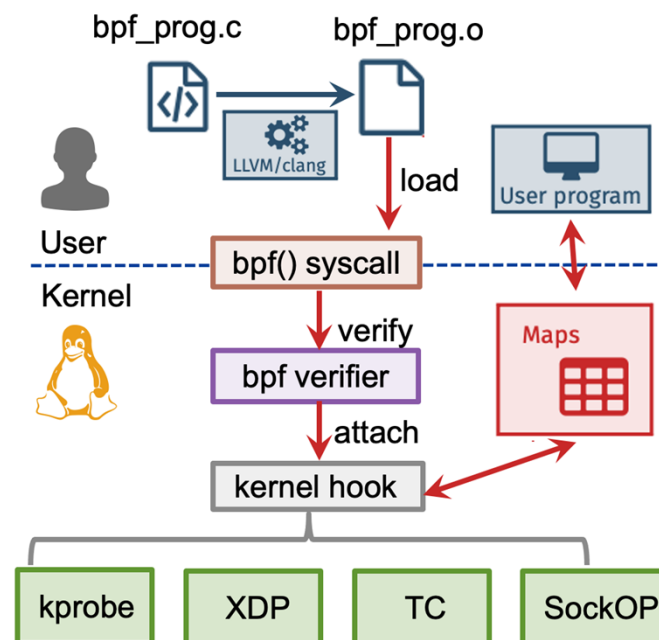


## 02 挑战

### □ 通过eBPF优化MPTCP

- eBPF允许在在**内核态运行用户态**程序。
- **eBPF验证器**对加载的程序进行验证以确保安全。
- 在实践中，通过eBPF验证器验证**并不容易**。  
(针对字节码进行验证)

```
from 4 to 6: R1=ctx(id=0,off=0,imm=0) R2=inv1 R4=inv2 R10=fp0,call_-1
6: (b7) r2 = 2
7: (05) goto pc+6
14: (67) r2 <= 2
15: (0f) r1 += r2
16: (61) r3 = *(u32 *)(r1 +48)
dereference of modified ctx ptr R1 off=8 disallowed
-- END PROG LOAD LOG --
```



挑战3: **难以同时确保安全性和易用性。**



03

# eMPTCP设计



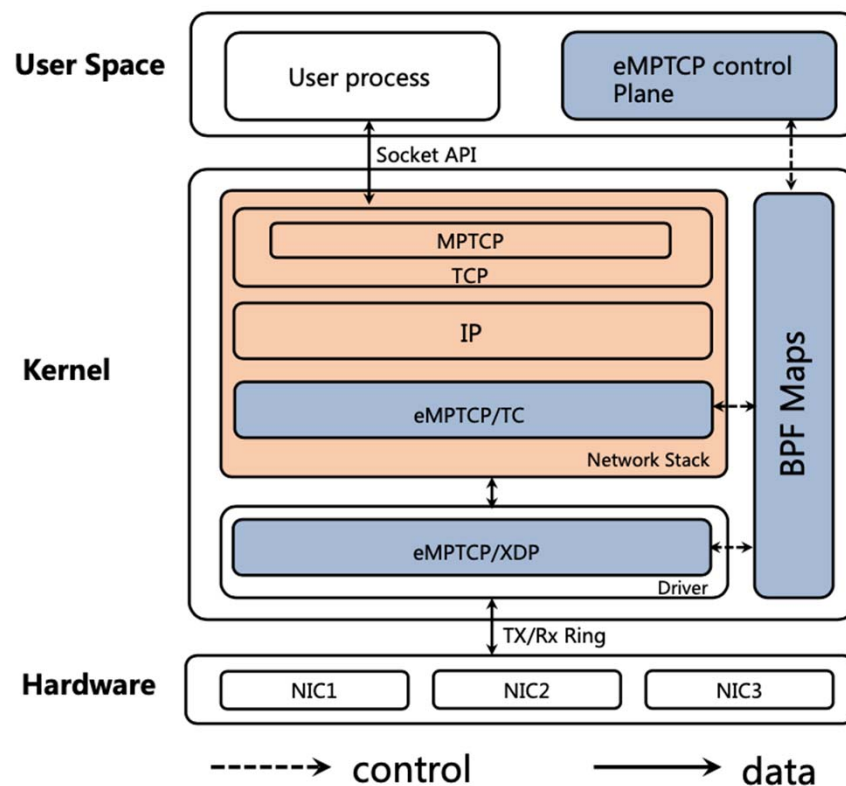


## 03 eMPTCP设计

### □ eMPTCP设计目标

我们针对上述挑战提出**eMPTCP**，一种基于eBPF技术的MPTCP控制框架。

- eMPTCP支持使用**模块化和插件化**的方式扩展MPTCP。
- eMPTCP**支持多种MPTCP操作**并扩展了MPTCP的功能使其更好地适应新兴环境。
- eMPTCP对用户友好。





## 03 eMPTCP设计

首届中国eBPF研讨会

### □ eMPTCP整体框架

#### 策略链

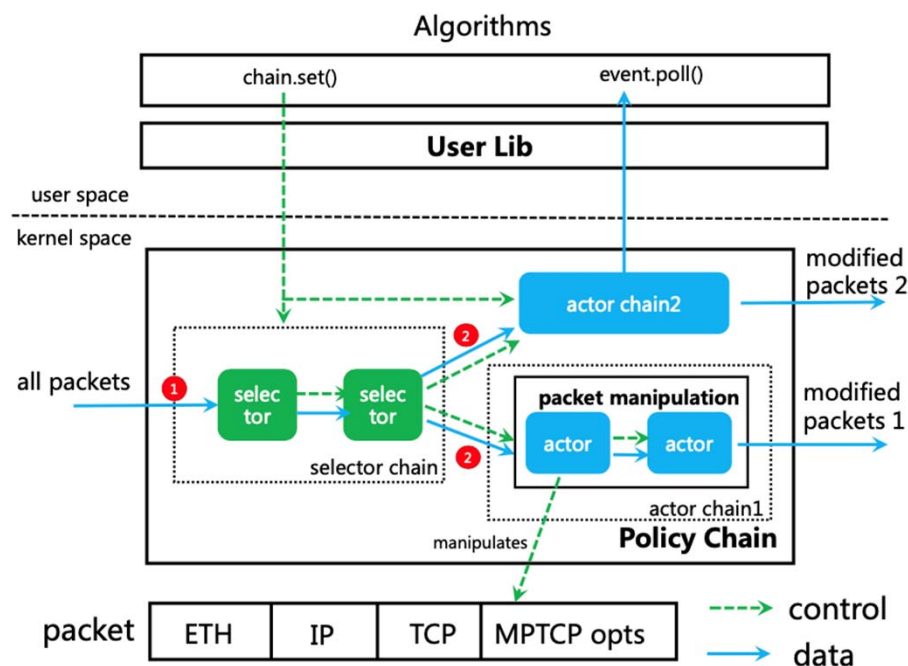
实现**selector-actor**风格的复杂策略链，基于策略链实现MPTCP扩展。**提升灵活性。**

#### 基于包处理

基于包处理实现selector和actor，支持多种现有MPTCP操作扩展MPTCP功能。**确保功能性。**

#### 面向需求抽象

封装了一系列**用户态API**和eBPF相关帮助函数。同时**确保安全性和易用性。**

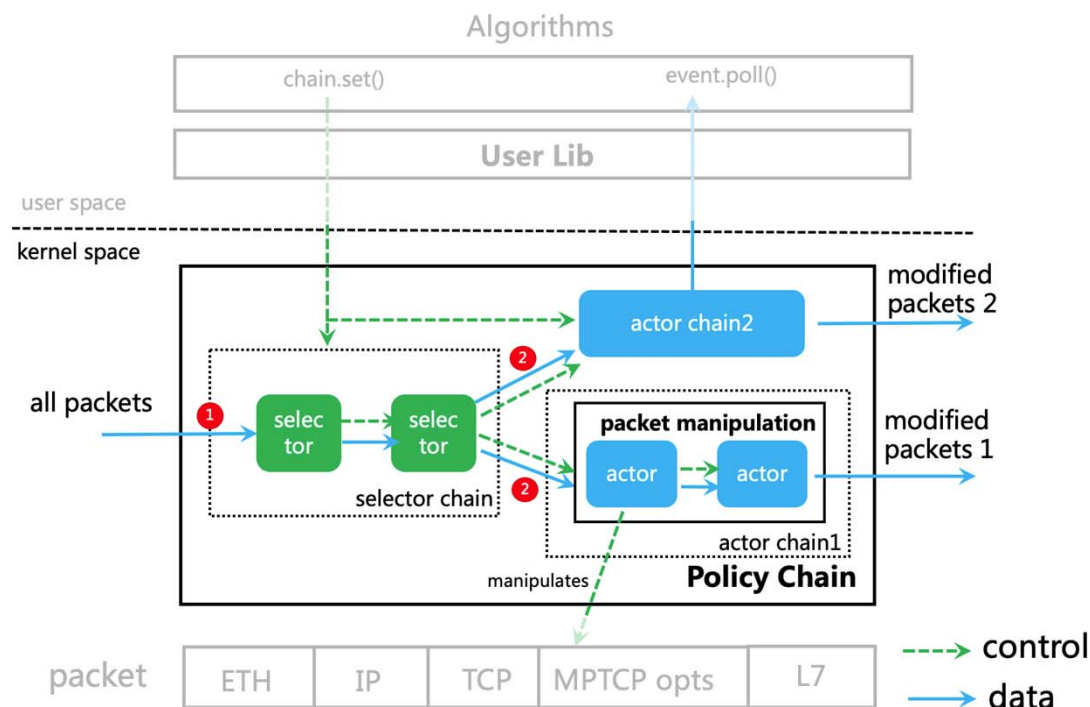




## 03 eMPTCP设计

### □ 基于选择器-执行器的策略链

- 选择器选择报文，动作器对选择的报文执行对应的操作。
- 多个选择/执行器可以以任意长度和顺序组成复杂的策略链。
- 支持根据应用需求添加新的选择/执行器。



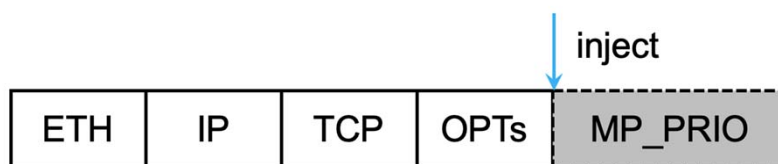




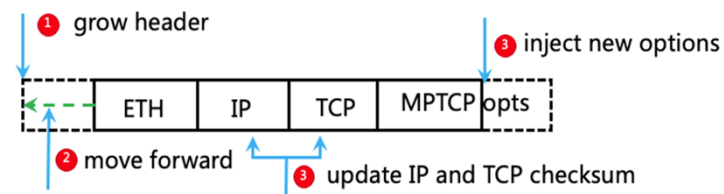
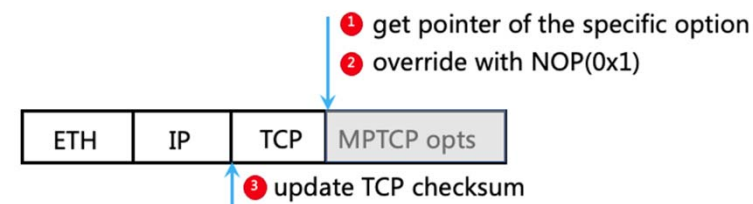
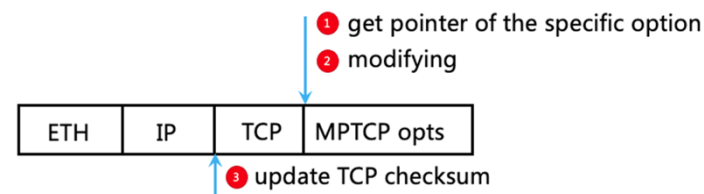
## 03 eMPTCP设计

### □ 基于包处理的跨协议多路径控制

- 修改、添加、删除TCP选项控制MPTCP行为。
- eMPTCP可以部署在传输的中间节点。
- 修改L2-L7的报文实现多协议层的协同控制。
- 以`set_backup`修改MPTCP子流优先级为例：



*set\_backup actor 实现*





## 03

# eMPTCP设计

### □ 选择器示例：

Selector name	Functionality
subflow	Filter packets by the TCP 4-tuple.
ip_pair	Filter packets by a (src, dst) pair.
src/dst	Filter packets by source or destination IP address.
sequence	Filter packets by Data Sequence Number or Subflow Sequence Number
packet_type	Filter packets by type, e.g., MPTCP SYN, Data ACK, etc.,

### □ 动作器示例：

Actor name	Parameters	Description
rate_limit	Rate	Update the recv_win of ACKs of a subflow to control the sending rate.
set_backup	Priority	Add MP_PRIO option to packet to set or remove the current subflow
blk_subflow	N/A	Remove and store MP_ADD_ADDR to avoid creation of subflows.
add_subflow	N/A	Add MP_ADD_ADDR to packet and enable creation of subflows.
get_connect	N/A	Parse MP_CAPABLE option to send MPTCP keys to event queue.
get_subflow	N/A	Parse MP_JOIN option to send subflow token to event queue.
record	Metric	Record specific metrics of selected packets such as RTT, flow size and etc.,

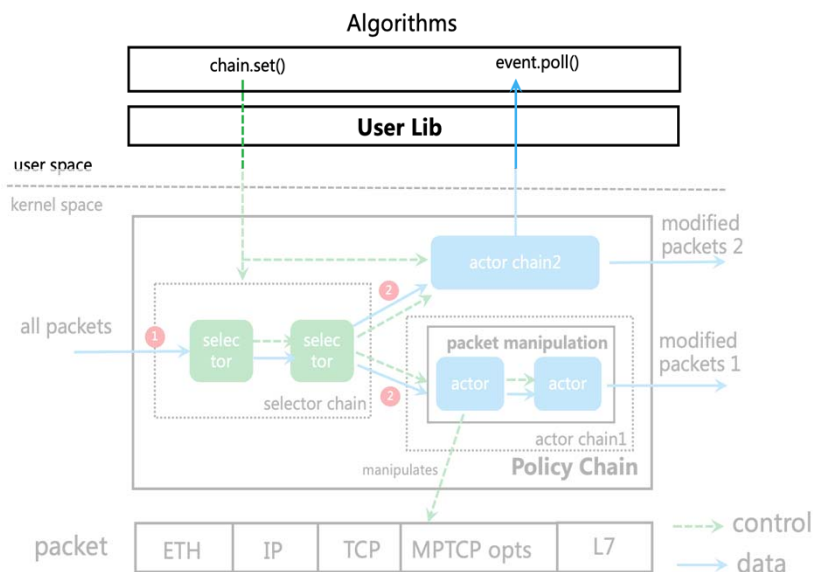


## 03 eMPTCP设计

首届中国eBPF研讨会

### □ 面向需求的API抽象

- 从使用角度：一系列 **intent-based** 的 python API。
- 从扩展角度：经过 eBPF 安全验证的 helper functions。



### ■ 用户态链式API

```
1 #create selector chain
2 sc = SelectorChain()
3 sc.add("ip_pair", SELECTOR_AND).\
4   add("port", SELECTOR_AND)
5 #create actor chain
6 ac = ActionChain()
7 ac.add("add_subflow").add("record")
8 #create policy chain
9 pc = PolicyChain(sc, ac)
10 #apply policy chain
11 pc.select(0, local_addr = "10.200.0.2", \
12   remote_addr = "10.200.1.2").set(1, 8080)
```

声明选择链

声明策略链

### ■ 自定义选择器/动作器

```
1 #include "emptcp_utils.h"
2 #include "emptcp_common.h"
3 SEC("xdp")
4 int your_own_policy(struct xdp_md *ctx)
5 {
6     SELECTOR_PRE_SEC
7     /*your own codes*/
8     SELECTOR_POST_SEC
9 }
10
11 }
```

使用eMPTCP提供的 helper function



04

# eMPTCP实现





## 04 eMPTCP实现

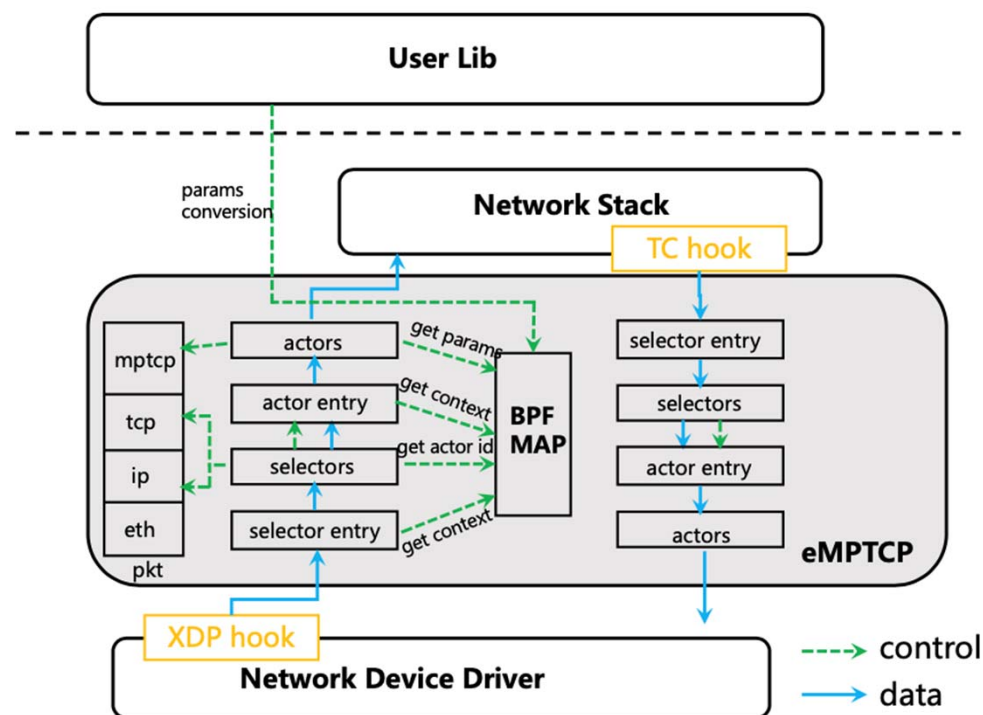
❑ 基于eBPF TC和XDP程序类型。

❑ eMPTCP策略链实现

- 基于eBPF尾调用的策略链。
- 基于 eBPF MAP 和 XDP/TC meta 的 actor 和 selector的**参数传递**。

❑ 包处理实现

- Direct Packet Access (修改包)。
- 增长包空间。
- Header-only helper functions。





## 04 eMPTCP实现：策略链

首届中国eBPF研讨会

### □ eBPF尾调用

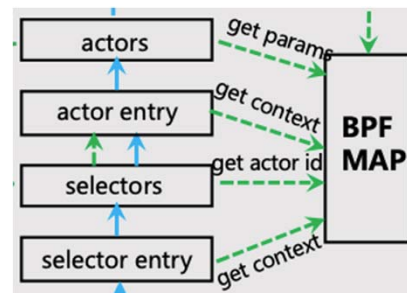
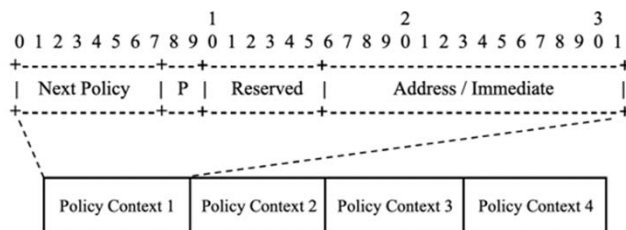
- eBPF尾调用解决单个eBPF程序**最大长度限制**。
- 执行尾调用要求caller和callee的程序类型相同，并且跳转后**不会返回caller的执行流**。

挑战

- 如何**动态地决定一条eBPF程序尾调用链（尾调用特性）**？如何知道下一个要调用的eBPF程序？
- 如何解决尾调用过程中可能出现的并发问题？

### □ Policy Context

- 用Context数组保存尾调用链所需要的状态信息。(next policy, param等)
- Context保存在per-packet的meta-data (通过指针直接访问) 解决并发问题。(XDP **data\_mata 32bytes**/TC **cb array**(20bytes))
- *entry*程序从BPF\_MAP**读取context保存到meta data**。调用链上的eBPF程序从context array中**获取下一个要调用的程序**以及参数。





## 04 eMPTCP实现：参数传递

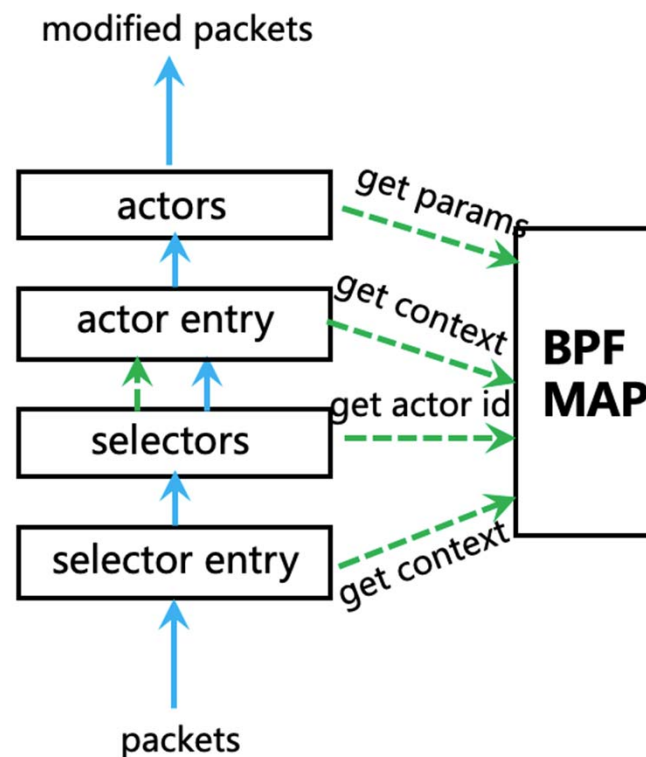
首届中国eBPF研讨会

### □ 子策略之间传递和共享参数

- 对于小参数 ( $\leq 2\text{Bytes}$ ) 参数直接内联地保存在 **meta-data里的context array**中 (直接通过指针访问)。
- 对于大参数 ( $> 2\text{Bytes}$ )从context array中获取offset, 从BPF\_MAP中读取参数。
- 通过context array传递参数。

### □ 编程技巧

- 不同的 **.o** 文件里的BPF Prog要共享MAP, 可以通过libbpf的 `bpf_map__reuse_fd` API。
- 将BPF\_MAP pin (通过**`bpf_obj_pin`** API)到虚拟文件系统来控制BPF\_MAP生命周期。



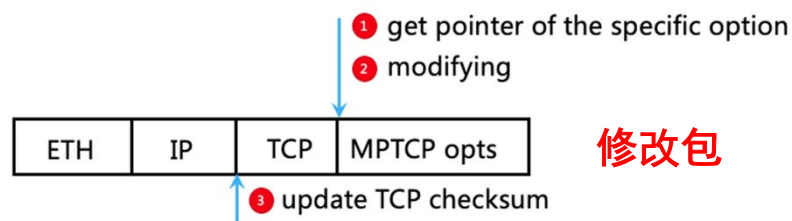




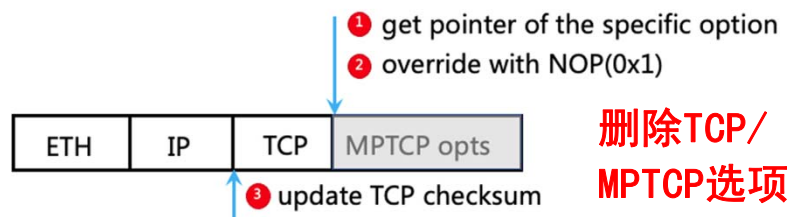
## 04 eMPTCP实现：包处理

首届中国eBPF研讨会

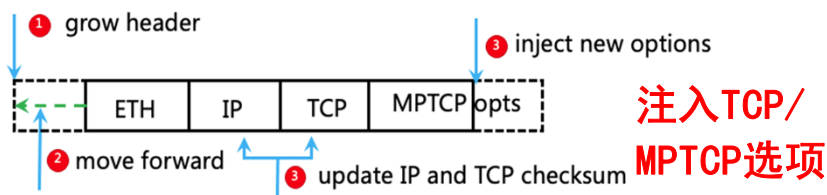
### □ eBPF for packet processing



Directly Packet Access (DPA)。



将TCP选项用NOP代替，问题转化为包修改。



增长packet space，修改增长的space。

eMPTCP header-only helper functions





## 04 eMPTCP实现：包处理

首届中国eBPF研讨会

### □ Direct Packet Access

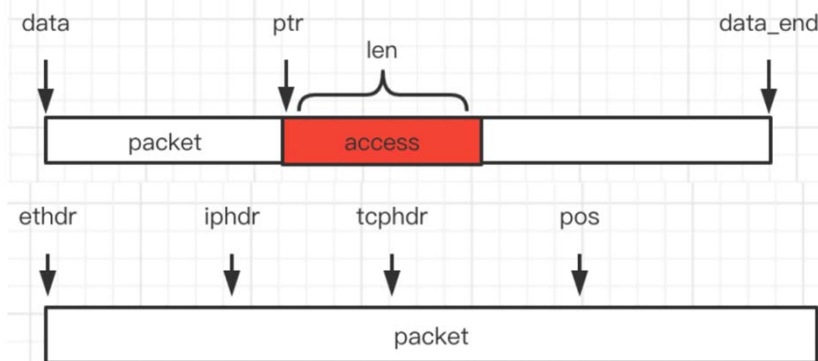
- DPA的含义是直接通过指针来访问和修改Packet内容。
- DPA可以访问packet(XDP), sk\_buff线性区(TC)和meta\_data

### □ 使用Direct Packet Access

- 要点在于，对指针解引用之前必须验证指针的有效性。
- $ptr \in [data, data\_end)$  (对于packet内容)
- $ptr \in [data\_meta, data)$  (对于meta)

```
1 struct xdp_md {
2     __u32 data;
3     __u32 data_end;
4     __u32 data_meta;
5     /* Below access go through struct xdp_rxq_info */
6     __u32 ingress_ifindex; /* rxq->dev->ifindex */
7     __u32 rx_queue_index; /* rxq->queue_index */
8 };
```

```
1 if ((void*)ptr + len > data_end) {
2     //如果ptr不是有效的指针
3     return;
4 }
5 // access ptr here
```



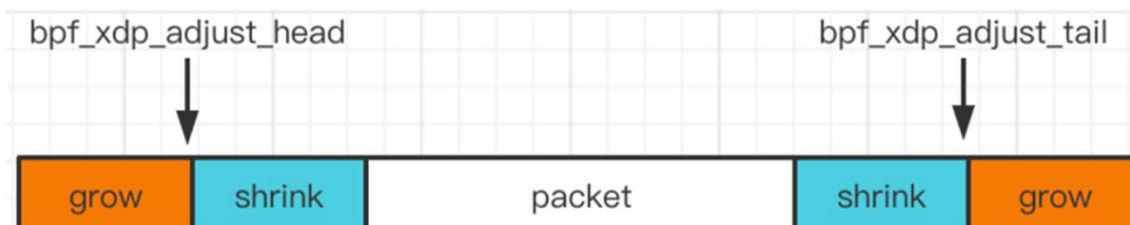


## 04 eMPTCP实现：包处理

首届中国eBPF研讨会

### □ 修改 (grow/shrink) 包空间

- 使用 adjust room 相关的 eBPF 帮助函数 (*bpf\_xdp\_adjust\_head*, *bpf\_xdp\_adjust\_tail*)。
- adjust包空间之后，所有的用于DPA的指针必须重新验证。



### □ Header-only helper functions

```
1 // add MP_PRIO 4bytes
2 xdp_grow_tcp_header(ctx, &nh, tcp_opt_len, sizeof(struct mp_prio), &modified); //1
3 is_tcp_packet(&new_nh, data_end, &eth, &iph, &tcph); //2
4 add_tcp_opts(&nh, data_end, &prio_opt, sizeof(struct mp_prio)); //3
5 update_tcphlen_csum(iph, tcph, sizeof(struct mp_prio)); //4
6 //recompute checksum , mp_prio 4 bytes
7 add_tcpopt_csum(&tcph->check, &prio_opt, sizeof(struct mp_prio)); //5
```



# 05 实验





## 05 实验

### □ 实验环境

CPU	2 Intel(R) Xeon(R) E5-2630 v4 2.2Ghz CPUs (12 cores)
Network	3 10 Gbps Broadcom Network Interface Card
Memory	128GB

**MPTCP**设置: eMPTCP对MPTCP**实现版本并不敏感**, 支持MPTCP V0和V1。在Linux4.19.126, Linux5.10, Linux5.15, Linux5.19下均测试过。

---

### □ 实验目标

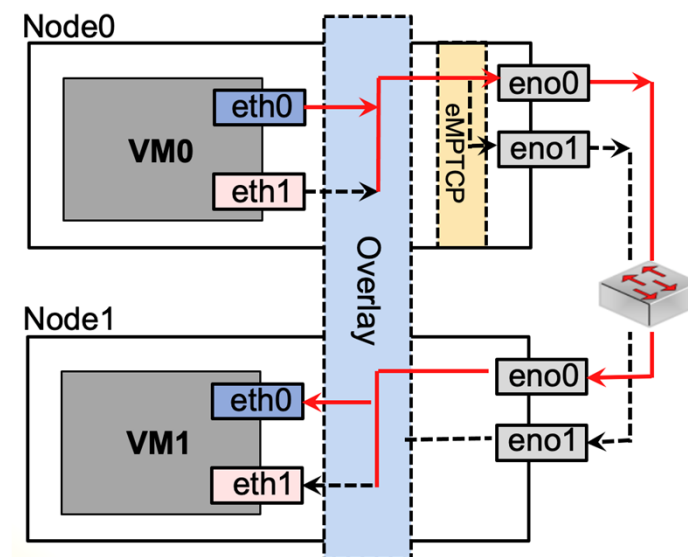
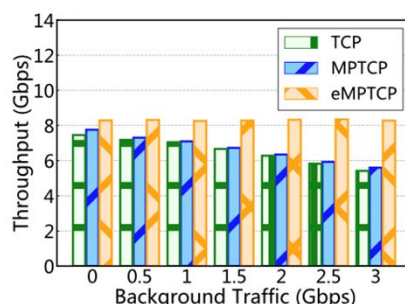
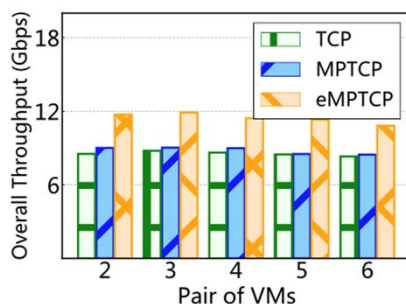
- 验证eMPTCP在若干不同**use case**下的效果。
- 测试eMPTCP对**性能**造成的影响。



## 05 实验：用例1

### ❑ 多租户 (multi-tenant) 环境和MPTCP

- 把eMPTCP部署在hypervisors上。
- 基于eMPTCP实现一个简单的策略，将不同子流的流量**重定向**到不同的物理链路上。



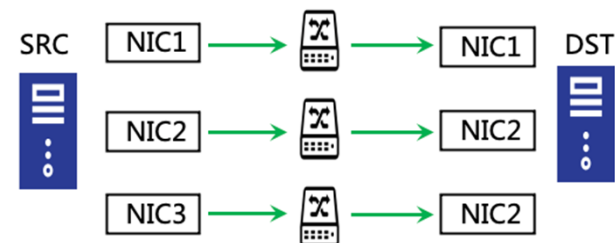
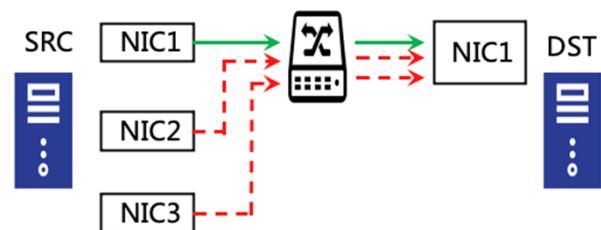
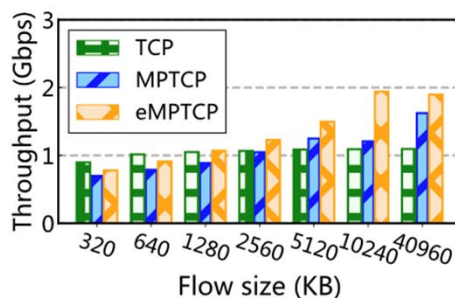
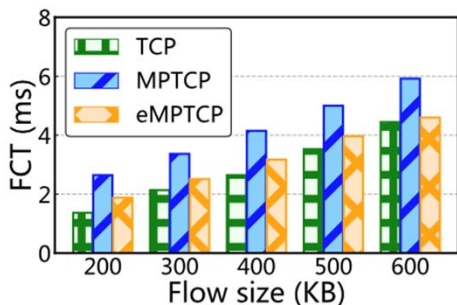
- 允许不同的子流能够通过底层**不同的物理链路**发送。
- eMPTCP可以有效提升VMs之间的通信带宽，在没有背景流平均提升 **23.03%**，在有背景流下最多提升 **32.3%**。



## 05 实验：用例2

### □ MPTCP路径管理

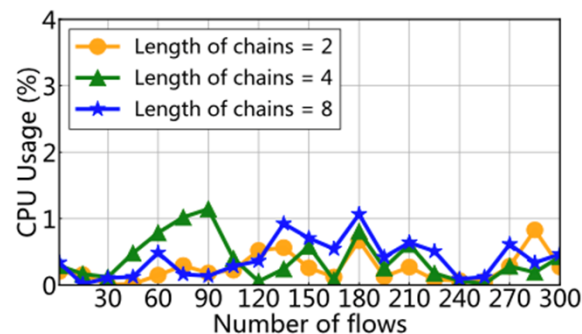
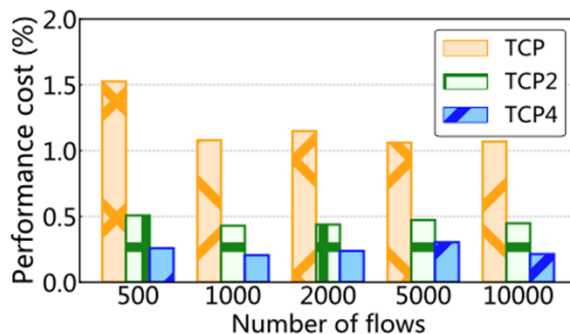
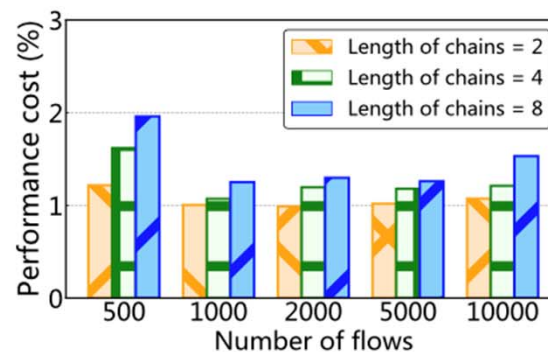
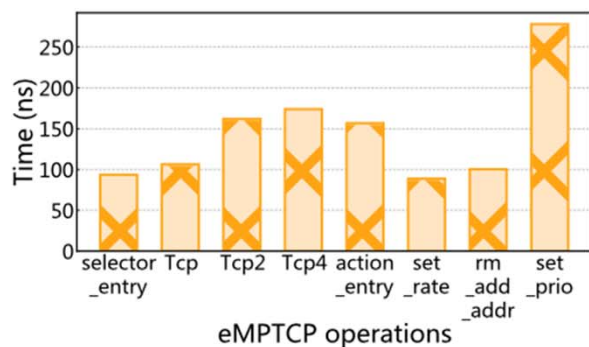
- 在连接之初**禁止子流**建立，只使用单路径传输。
- 发送一定量数据之后，恢复子流建立。并保证新建的子流和现有子流**不存在共享链路**。



- 对于小流，eMPTCP提供**接近TCP**的性能。
- 对于大流，eMPTCP将MPTCP吞吐量平均**提升23.1%**。



## 05 实验：性能测试



- eMPTCP算子处理时间在**纳秒级**。
- eMPTCP对流传输时间的影响**小于%2**。
- 平均CPU占用率为**0.35%**。



## 06 总结







## 06

## 总结

□ 提出了eMPTCP，基于eBPF技术的MPTCP控制框架

■ 插件化和模块化

- 基于selector和actor的策略链，采用链式的方式开发MPTCP扩展。
- 运行时修改策略链，无需中断服务。

■ 扩展了MPTCP的功能

- 支持多种MPTCP操作。
- 扩展了MPTCP的功能。(多租户场景)

■ 加快MPTCP发展步伐

- 对用户态友好的API。
- MPTCP相关的eBPF header-only 帮助函数。



06

## 总结

首届中国eBPF研讨会

### □ 相关论文发表



## IEEE ICNP 2022

The 30th IEEE International Conference on Network Protocols

Lexington, Kentucky, USA, October 30-November 2, 2022 Follow @IEEE\_ICNP 



- Towards the Full Extensibility of Multipath TCP with eMPTCP [C]//2022 IEEE 30th International Conference on Network Protocols (ICNP).



## IEEE CLUSTER 2022

Heidelberg, Germany • 6–9 September



- Last-mile Matters: Mitigating the Tail Latency of Virtualized Networks with Multipath Data Plane [C]//2022 IEEE International Conference on Cluster Computing (CLUSTER).

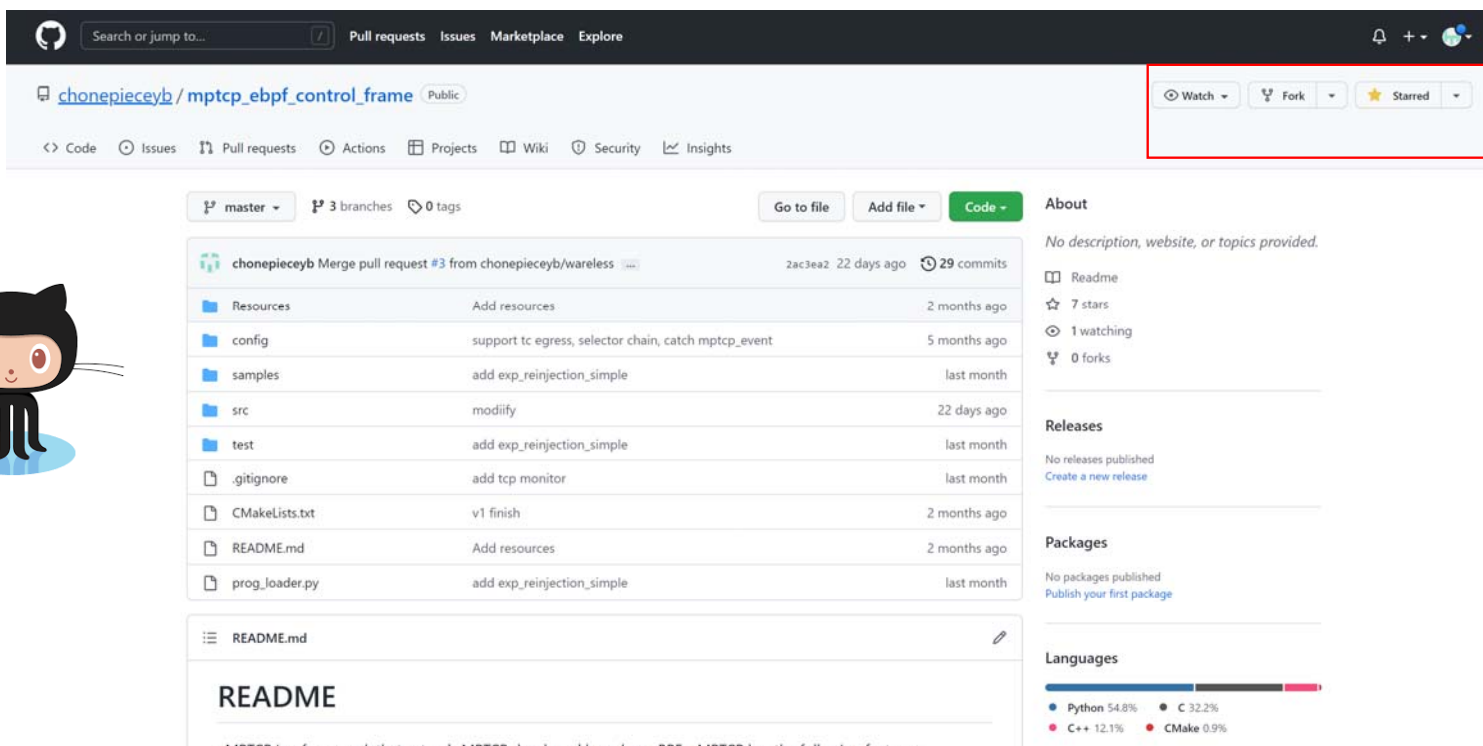


06

## 总结

首届中国eBPF研讨会

### □ 开源项目



The image shows a screenshot of a GitHub repository page for `chonepieceyb/mptcp_ebpf_control_frame`. The repository is public and has 7 stars, 1 watching, and 0 forks. The page includes a table of recent commits, a list of files, and a sidebar with repository statistics.

File	Commit Message	Time
Resources	Add resources	2 months ago
config	support tc egress, selector chain, catch mptcp_event	5 months ago
samples	add exp_reinjection_simple	last month
src	modify	22 days ago
test	add exp_reinjection_simple	last month
.gitignore	add tcp monitor	last month
CMakeLists.txt	v1 finish	2 months ago
README.md	Add resources	2 months ago
prog_loader.py	add exp_reinjection_simple	last month

**README**

mptcp is a framework that extends MPTCP developed based on eBPF. mptcp has the following features:

**Languages**

- Python 54.8%
- C 32.2%
- C++ 12.1%
- CMake 0.9%

[https://github.com/chonepieceyb/mptcp\\_ebpf\\_control\\_frame.git](https://github.com/chonepieceyb/mptcp_ebpf_control_frame.git)



◎ 首届中国eBPF研讨会



東南大學  
SOUTHEAST UNIVERSITY

# 感谢聆听!

会后有任何问题，欢迎交流指正!

联系方式: [dshen@seu.edu.cn](mailto:dshen@seu.edu.cn)