Vida e Morte de um Pacote de Rede no Kernel

Fábio Olivé
(fabio.olive@gmail.com)

livre
OeSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Objetivo

Compreender a interação entre os principais protocolos da família TCP/IP, a organização do código de redes do Linux, suas principais estruturas de dados, e principalmente aprender a usar algumas das principais ferramentas que podem ajudar na continuação deste estudo.

livre
eSC
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Tópicos

- Parte 1: Preparação
  - Obtendo e preparando o repositório git do Kernel
  - Ferramentas utilizadas: systemtap, crash, tags/cscope, ...
- Parte 2: Conceitos
  - Protocolos TCP, UDP, ICMP, IP, ARP e Ethernet
  - Estrutura de dados sk_buff (skb), net_device, ...
- Parte 3: Exploração
  - Aplicação dos conceitos e ferramentas
  - Exemplos de investigação do código dos protocolos

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Parte 1

Preparação

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Obtendo e Preparando o Código

- Instalar git, gitk e cscope
- Clonar o repositório linux-stable, que é o kernel usado pelas distribuições estáveis
- Rodar o cscope para criar as referências de símbolos para navegar facilmente no código
- Usar seu editor favorito pra ler o código

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Obtendo e Preparando o Código

```
$ sudo yum install git gitk cscope

$ git clone git://git.kernel.org/pub/scm/linux
              /kernel/git/stable/linux-stable.git

$ cd linux-stable; git checkout linux-3.9.y
  (ou git checkout v3.9.5, por exemplo)

$ make cscope
$ vim -t ip_rcv
```

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Ferramentas: systemtap

- Intercepta funções do Kernel em execução
- Linguagem simples para programar as probes
  - Foco em extração rápida de informações, backtraces, ...

```
# stap -e 'probe kernel.function(“ip_rcv”) {
  printf(“Pacote da %s\n”, kernel_string($dev->name));
}

# stap arquivo-de-probes.stp
# stap -L 'kernel.function(“ip_rcv”)'
```

livre eSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Ferramentas: systemtap

```
$ sudo yum install systemtap

$ sudo debuginfo-install kernel
```

- Debuginfo são pacotes imensos de informações geradas na compilação dos pacotes binários
  - Símbolos, estruturas de dados, offsets, endereços
- Systemtap é uma maneira fantástica e segura de aprender sobre o funcionamento do Kernel

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Ferramentas: crash

- Debugger de Kernel baseado no GDB
- Permite analisar crashes através de vmcores ou plugar no Kernel em execução (read-only)
- Ótimo para investigar estruturas de dados e outras informações mais "estáticas"
- Também precisa dos debuginfos

```
$ sudo yum install crash
```

livre
eSC
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Ferramentas: crash

```
# crash
...
crash> udp_protocol
udp_protocol = $9 = {
  early_demux = 0,
  handler = 0xffffffff815ba930 <udp_rcv>,
  err_handler = 0xffffffff815b9a50 <udp_err>,
  ...
}
crash> dis udp_rcv
0x...815ba930 <udp_rcv>:   nopl   0x0(%rax,%rax,1)
0x...815ba935 <udp_rcv+5>: push   %rbp
...
```

livre OeSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Sugestão de Uso das Ferramentas

- Estudar o código de algumas funções
- Criar um teste que exercite estas funções
- Criar systemtaps para extrair algumas informações úteis, como ponteiros para certas estruturas
- Investigar as estruturas interativamente no crash, usando os endereços obtidos com systemtap

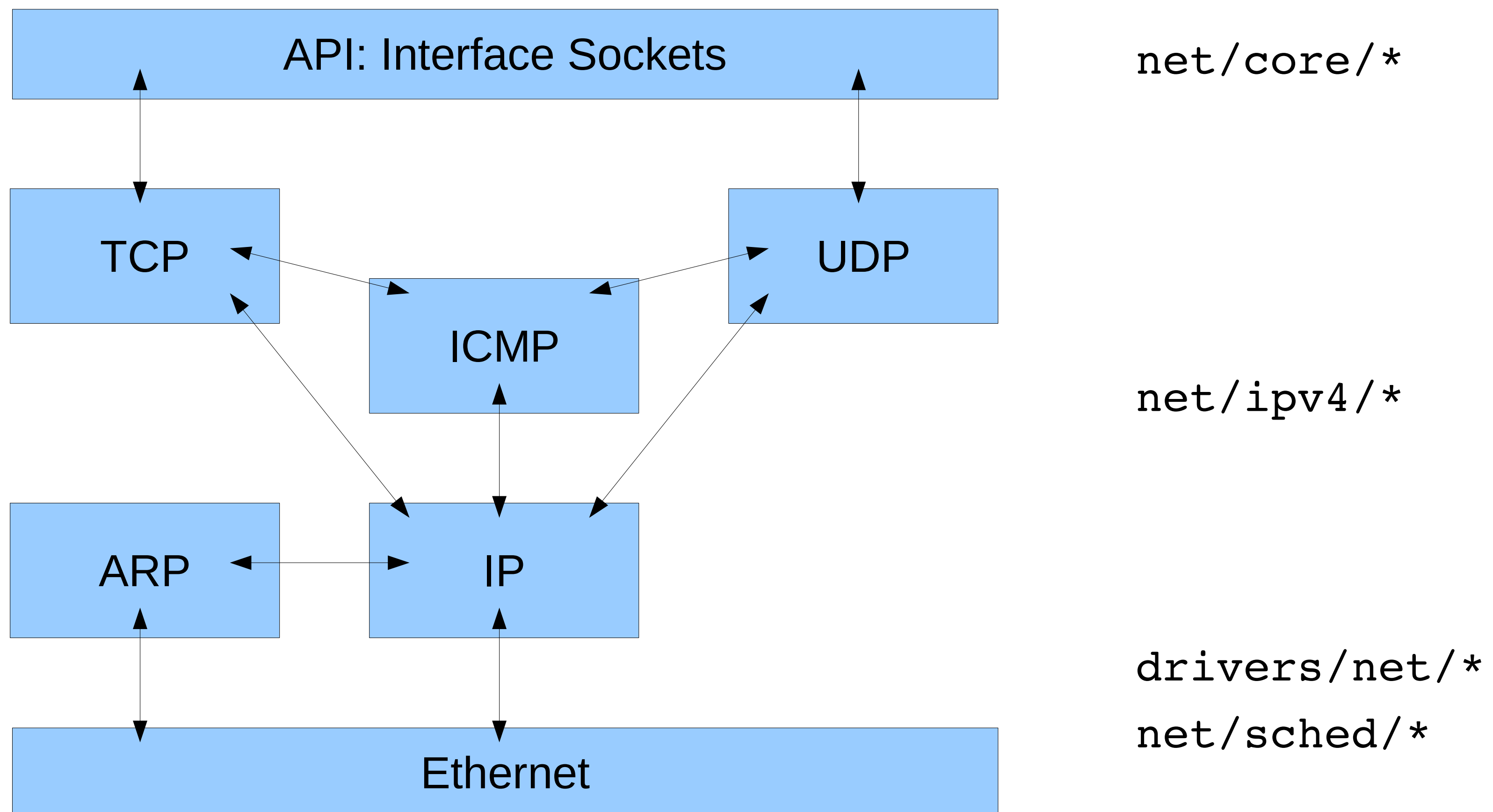
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Parte 2

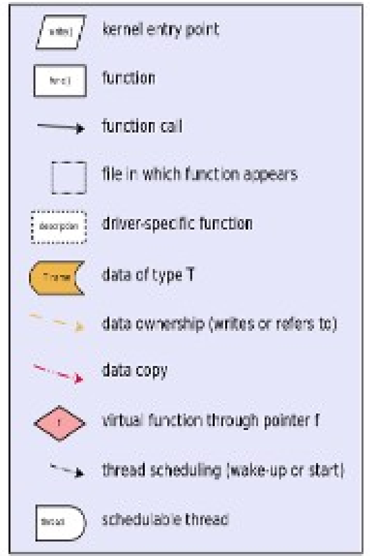
Conceitos

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

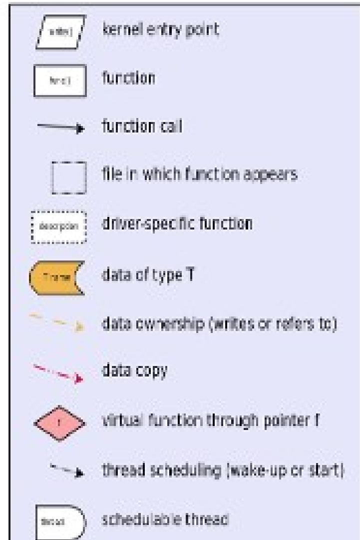
Protocolos da Família TCP/IP



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.



kernel entry point
function
function call
file in which function appears
driver-specific function
data of type T
data ownership (writes or refers to)
data copy
virtual function through pointer f
thread scheduling (wake-up or start)
schedulable thread



VAMOS COM CALMA! :-)

Protocolo TCP

- Obviamente é o que tem código mais complexo
- Diferentes estados, filas, timers
- Não é o melhor protocolo pra iniciar o estudo :-)
- Pontos interessantes de pesquisa:
  - `tcp_sendmsg, tcp_transmit_skb, tcp_retransmit_skb`
  - `tcp_v4_rcv, tcp_v4_do_rcv, tcp_rcv_established`

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolo UDP

- Código simples e “fácil” de compreender
- Fácil de exercitar via pequenos testes e programas
- Quase todo contido em `net/ipv4/udp.c`
- Pontos interessantes de pesquisa:
  - `udp_sendmsg`, `udp_send_skb`, `udp_push_pending_frames`
  - `udp_rcv`, `udp_queue_rcv_skb`

livre OeSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolo ICMP

- Lida apenas com pacotes simples, fácil de entender
- Só não é tão fácil de exercitar com programas, por ser um protocolo “interno”
- Todo contido em `net/ipv4/icmp.c`
- Pontos interessantes de pesquisa:
  - `icmp_send`, `icmp_reply`
  - `icmp_rcv`, `icmp_echo`

livre
eSC
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolo IP

- Precisa lidar com fragmentação, roteamento, e também aplica os “ganchos” do iptables
- Implementação separada em alguns arquivos:
  - `ip_input.c, ip_output.c, ip_forward.c, ...`
- Pontos interessantes de pesquisa:
  - `ip_rcv[_finish], ip_local_deliver[_finish]`
  - `ip_queue_xmit, ip_local_out, ip_output`
  - `ip_forward, ip_forward_finish`

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolo ARP

- Ainda mais fácil que o ICMP, por ser bem limitado
  - Pacotes simples direto no Ethernet
  - Interação com o Neighbour Cache complica um pouco
- Todo contido em `net/ipv4/arp.c`
- Pontos interessantes de pesquisa:
  - `arp_create, arp_xmit, arp_send`
  - `arp_rcv, arp_process`

livre OeSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Camada Ethernet

- Implementada pelos drivers e pelo packet scheduler
- Recepção:
  - Driver encapsula os frames recebidos em `sk_buffs` (skb)
  - Chama `netif_receive_skb` para "entrar na pilha"
- Transmissão:
  - Protocolos enviam frames via `dev_queue_xmit`
  - O packet scheduler aplica a "disciplina de fila" (QoS, etc)
  - Para transmitir chama `hard_start_xmit` no driver

livre OeSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Mas onde estão os pacotes???

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

A Estrutura `sk_buff`

- Representa um pacote de rede no Kernel
- Definida em `include/linux/skbuff.h`
- Precisa ser extremamente eficiente
  - Pensem num roteador processando milhões de pacotes/s
- Concentra vários ponteiros para a área de dados contendo o frame, e informações de protocolos de vários níveis e também do driver de rede

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

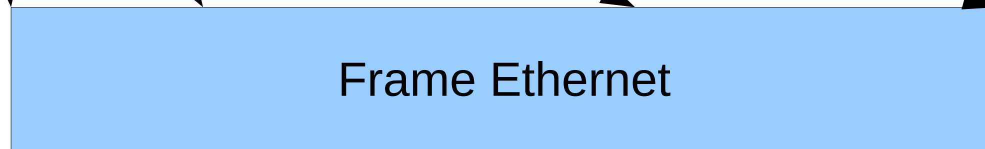
A Estrutura `sk_buff`

```
struct sk_buff {
	struct sk_buff *next, *prev;

	struct sock *sk;
	struct net_device *dev;

	unsigned int len, data_len;
	unsiged char *head;
	unsiged char *data;
	unsiged char *tail;
	unsiged char *end;
}
```

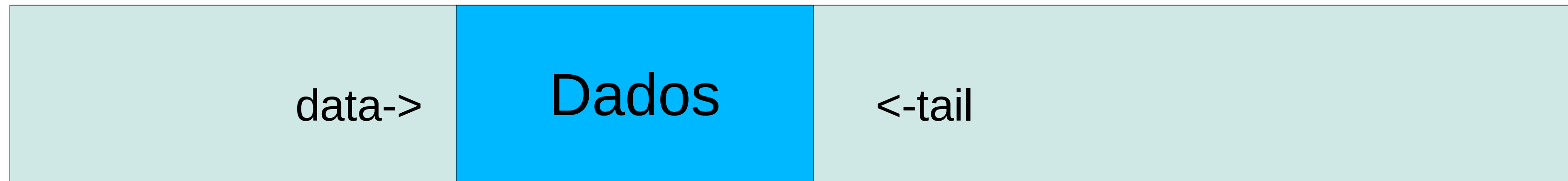
```
struct net_device {
	char name[];
	unsigned int flags;
	unsigned int mtu;
}
```

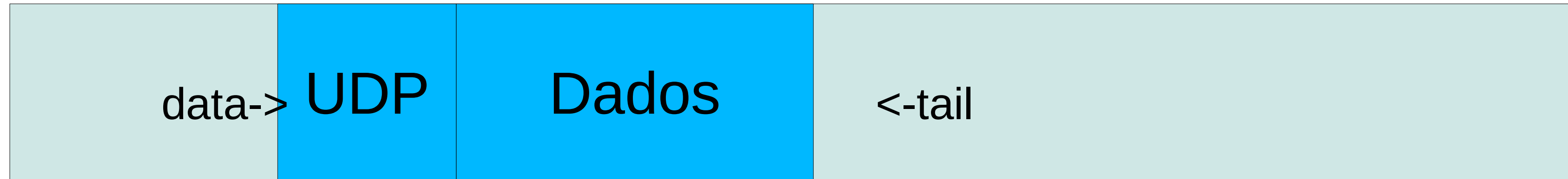


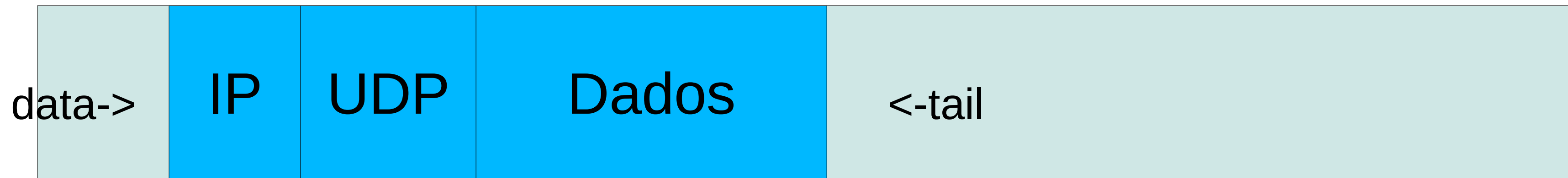
This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

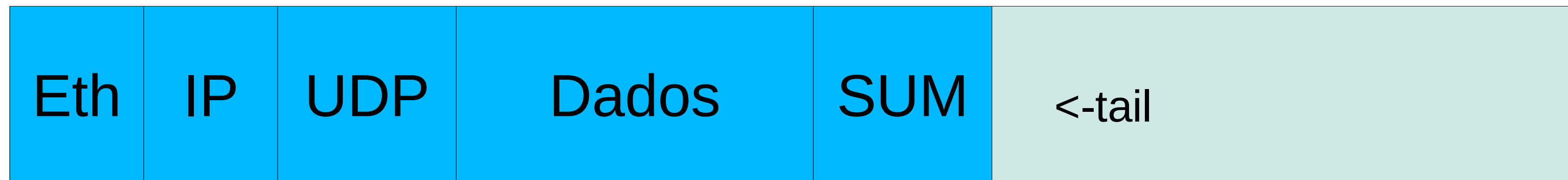
A Estrutura `sk_buff`

head end









livre eSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Parte 3

Exploração

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolos Simples: ICMP

- Observar um ping com systemtap
- O comando ping passa por icmp_send?
- Que outras funções são chamadas?

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Protocolos Simples: ARP

- Observar processamento de ARP com systemtap

```
# stap -ve '
probe kernel.function(*@net/ipv4/arp.c)
{
  println(probefunc())
}'
```

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Rastreando SKBs

- Guardar o endereço de um SKB quando ele entra e rastrear por quais outras funções ele passou
- Mostrar o tamanho do frame em vários pontos da pilha de protocolos, e os ponteiros data e tail
- Mostrar quem aloca e quem libera SKBs na pilha, e quando isso ocorre

livre
eSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Sockets UDP

- Verificar os pacotes sendo transmitidos e recebidos
- Monitorar o tamanho da fila de recepção

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Sockets TCP

- Tentar observar o TCP Segmentation Offload
- Tentar causar e observar uma retransmissão

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Investigação Generalizada

Que outros pontos querem olhar?

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Dúvidas?

Perguntem!

Experimentem!

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

Referências Mínimas

- Kernel Networking Flow
  - http://www.linuxfoundation.org/collaborate/workgroups/networking/kernelflow
- How SKBs Work
  - http://vger.kernel.org/~davem/skb.html
- Systemtap Wiki
  - http://sourceware.org/systemtap/wiki
- Whitepaper on Crash
  - http://people.redhat.com/anderson/crash_whitepaper/
- Muitas horas estudando e brincando com stap ;-)

livre eSC

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.