Systemy Operacyjne 2

Obsługa sieci w Linuksie

Arkadiusz Chrobot

Katedra Systemów Informatycznych

20 czerwca 2024

Plan

Wprowadzenie

Unix jest jednym z pierwszych systemów operacyjnych, które oferowały implementację komunikacji sieciowej. Współcześnie większość serwerów w Internecie pracuje pod kontrolą Linuksa, systemu operacyjnego, który jest kompatybilny z Uniksem. Ten wykład jest krótkim przeglądem budowy i działania podsystemu jądra Linuksa odpowiedzialnego za obsługę sieci. Tematyka ta jest złożona, dlatego tylko najważniejsze zagadnienia są tutaj zaprezentowane. Treść wykładu jest podzielona na trzy części:

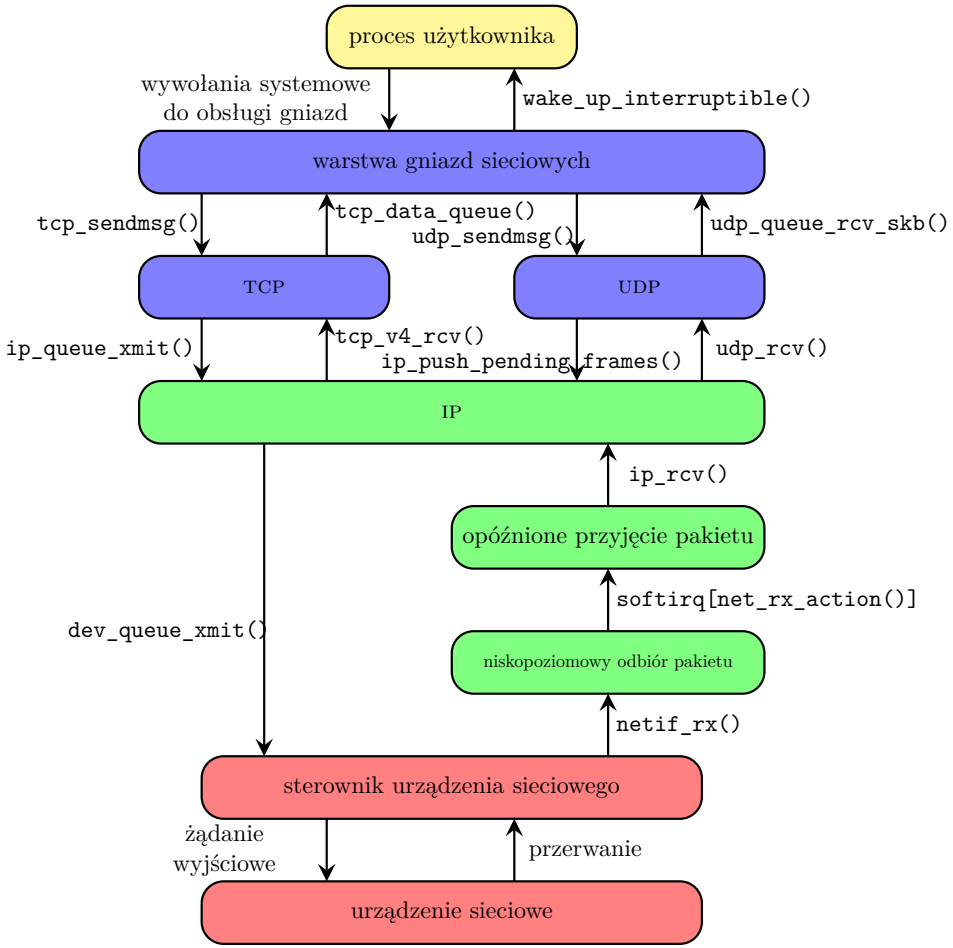
- przetwarzanie pakietu na poziomie jądra,
- sterowniki urządzeń sieciowych,
- implementacja filtra sieciowego.

Stos TCP/IP

Część jądra odpowiedzialna za przetwarzanie nadchodzących i wychodzących pakietów (ang. *packets*) sieciowych jest nazywana stosem TCP/IP. Rysunek 1 przedstawia jej schemat [1]. Podsystem jądra Linuksa odpowiedzialny za komunikację sieciową składa się z trzech części, które odpowiadają trzem warstwom modelu ISO/OSI — warstwie łącza danych, warstwie sieciowej i warstwie transportowej. Aby wysłać dane przez sieć proces użytkownika wywołuje odpowiednie wywołanie systemowe, które aktywuje metodę `write()` obiektu pliku związanego z gniazdem sieciowym tego procesu. W zależności od użytego protokołu transportowego ta metoda wywołuje funkcję jądra `tcp_sendmsg()` lub `udp_sendmsg()`. Są one odpowiedzialne za utworzenie nagłówka odpowiedniego protokołu.

[1] Wykład został przygotowany na podstawie strony: https://docs.kernel.org/networking/index.html i książki Williama Stallingsa „Systemy operacyjne”, PWN, Warszawa, 2009

Stos TCP/IP



Rysunek 1: Przetwarzanie pakietów wychodzących i nadchodzących w jądrze

Stos TCP/IP

Nagłówek protokołu transportowego jest dołączany do danych przekazanych przez proces użytkownika. Następnie jest wywoływana funkcja odpowiedzialna za utworzenie i dodanie do pakietu nagłówka protokołu IP. Dla pakietu UDP jest to `ip_push_pending_frames()`, a w przypadku TCP, funkcja `ip_queue_xmit()`. Pakiet, ze wszystkimi wymaganymi nagłówkami jest przekazywany do sterownika urządzenia sieciowego przez funkcję `dev_queue_xmit()`. Jednakże, zanim zostanie on wysłany, jego trasa jest ustalana przez funkcję `ip_route_output()`, która sprawdza pamięci podręczne lub (jeśli jest to konieczne) tablice trasowania (ang. *routing tables*), by ją określić. Jeśli pakiet ma być wysłany do innego urządzenia w sieci, to następnie jest przetwarzany przez funkcję `ip_output()`.
Jeśli urządzenie sieciowe otrzyma nadchodzący pakiet, to zapisuje go w buforze i zazwyczaj generuje przerwanie. Opcjonalnie dwa inne przerwania mogą być przez nie używane: sygnalizujące zakończenie transmisji pakietu i sygnalizujące pojawienie się wyjątku transmisji.

Stos TCP/IP

Są przypadki, kiedy urządzenie sieciowe nie zgłasza przerwania po otrzymaniu pakietu. Ich opis znajduje się w dalszej części wykładu, poświęconej NAPI. Sterownik urządzenia przekazuje pakiet w buforze sprzętowym do funkcji `netif_rx()`, która przydziela pamięć na inny bufor, do którego zapisuje kopię otrzymanego pakietu, ustawia wskaźnik w sterowniku na nagłówek IP i dodaje pakiet do kolejki. Wszystkie pakiety z tej kolejki są przetwarzane przez funkcję `ip_rcv()`, która wywołuje `ip_local_deliver()`. Ta ostania uruchamia z kolei `tcp_v4_rcv()` dla pakietów TCP lub `udp_rcv()` dla pakietów UDP. Następnie, wywoływane są funkcje informujące proces użytkownika, że pakiet został odebrany. W przypadku pakietu TCP jest to funkcja `tcp_data_queue()`, a w przypadku pakietu UDP wywoływana jest `udp_queue_rcv_skb()`.

Stos TCP/IP

Główną strukturą danych używaną przez sieciowy podsystem jądra jest bufor pakietu nazywany `sk_buff`. Typem danych tego bufora jest `struct sk_buff`. Ta struktura przechowuje nie tylko odbierane lub nadawane dane, ale również metadane koniecznie do przetwarzania pakietu. Te ostatnie są zgromadzone w nagłówku pakietu. Bufor pakietu jest tak zaprojektowany, aby mógł być efektywnie przenoszony między kolejkami. Jeśli zachodzi potrzeba skopiowania go, to wystarczy tylko powielić jego nagłówek, który ma trzy pola wskazujące na prywatne nagłówki przechowujące metadane związane z trzema warstwami modelu ISO/OSI. Pole `transport_header` wskazuje na nagłówek warstwy transportowej. Nagłówek warstwy sieciowej jest wskazywany przez pole `network_header`. W końcu pole `mac_header` wskazuje na nagłówek warstwy łącza danych. Wszystkie bufory pakietów są zorganizowane w kolejkę zaimplementowaną jako lista dwukierunkowa.

Sterowniki urządzeń sieciowych

Główną strukturą danych używaną przez sterownik urządzenia sieciowego jest struktura typu `struct net_device`. Najważniejszymi polami tej struktury są: `mtu` — określa maksymalny rozmiar pakietu jaki urządzenie może obsłużyć, `flags` — określa stan urządzenia, `dev_addr` — wskaźnik na adres MAC, `promiscuity` — licznik, który przechowuje liczbę żądań przełączenia urządzenia w tryb bezładny (ang. *promiscuous mode*), `ip_ptr` — wskazuje na nagłówek przechowujący dane specyficzne dla protokołu IP w wersji 4, `netdev_ops` — wskazuje na strukturę wskaźników na funkcje, które wykonują takie operacje, jak np. wysyłanie pakietów, `rx_handler` — wskazuje na procedurę obsługi przerwania nadajnika.

Wcześniejsze implementacje sterowników urządzeń sieciowych wymagały, aby to urządzenia potwierdzało odebranie każdego pakietu przerwaniem. Prowadziło to do przeciążenia systemu w przypadku dużego ruchu sieciowego. W serii 2.6 jądra dodano nowe API sterowników urządzeń sieciowych, nazwane NAPI.

Sterowniki urządzeń sieciowych

NAPI umożliwia sterownikowi przełączenie urządzenia sieciowego w tryb przeglądania (ang. *polling*) pozwalający mu na zgromadzenie odpowiedniej liczby przychodzących pakietów, które są przetwarzane później przez jądro. To redukuje liczbę zgłaszanych przez urządzenie przerwań i w konsekwencji zmniejsza obciążenie jądra ich obsługą. Niektóre z przychodzących pakietów mogą nawet być odrzucone zanim zostaną przekazane jądru do przetwarzania. To rozwiązanie nazywa się *dławieniem pakietów* (ang. *packet throttling*). Aby użycie NAPI było możliwe wymagane jest wsparcie sprzętowe w postaci tzw. cyklicznego bufora dla transmisji DMA (ang. DMA *ring*) lub odpowiednio duża wolna przestrzeń w RAM, aby można było ją przydzielić dla buforów dla tych transmisji.

Filtr sieciowy

Filtr sieciowy (ang. *netfilter*) jest zbiorem wskaźników na funkcje, nazywanych *uchwytami* (ang. *hooks*), rozlokowanych w strategicznych miejscach stosu TCP/IP. Te wskaźniki mogą być użyte do zaimplementowania zapory sieciowej (ang. *firewall*) lub takich rozwiązań jak NAT (ang. *Network Address Translation*). Funkcje wskazywane przez uchwyty są zazwyczaj definiowane w modułach jądra[2]. W podsystemie sieciowym jądra znajduje się pięć takich uchwytów:

`NF_IP_PRE_ROUTING` funkcje związane z tym uchwytem są wywoływane, gdy odbierany jest pakiet,

`NF_IP_LOCAL_IN` funkcje związane z tym uchwytem przeprowadzają wstępne przetwarzanie pakietów dostarczonych do urządzenia sieciowego (ang. *host*),

`NF_IP_FORWARD` funkcje związane z tym uchwytem przeprowadzają przetwarzanie pakietów, które powinny być przekazane do innych urządzeń sieciowych,

[2] http://www.paulkiddie.com/creating-a-netfilter-kernel-module-which-filters-udp-packets

Filtr sieciowy

`NF_IP_POST_ROUTING` funkcje związane z tym uchwytem przetwarzają pakiety z ustalonymi trasami, które mają zostać wysłane,

`NF_IP_LOCAL_OUT` funkcje związane z tym uchwytem przetwarzają pakiety, które zostały wysłane lokalnie.

Każda funkcja związana z dowolnym z tych uchwytów może przeprowadzać dowolną operację na pakiecie, jaka jest konieczna, ale musi ostatecznie zwrócić jedną z następujących wartości:

`NF_ACCEPT` pakiet zaakceptowany do dalszego przetwarzania,

`NF_DROP` pakiet został odrzucony,

`NF_REPEAT` należy ponownie wywołać funkcję dla tego pakietu,

`NF_STOLEN` funkcja „wykrada” pakiet, co oznacza, że będzie on przetwarzany w inny sposób niż pozostałe pakiety,

Filtr sieciowy

`NF_QUEUE` pakiet jest umieszczony w kolejce, skąd zostanie przekazany do przestrzeni użytkownika,

`NF_STOP` przetwarzanie pakietu zostało wstrzymane.

Pojedyncza funkcja związana z uchwytem jest reprezentowana przez strukturę typu `struct nf_hook_ops`. Definicja tego typu jest zamieszczona w Listingu nr 1. Pole `list` pozwala przechowywać te struktury w liście. Pole `hook` jest wskaźnikiem na funkcję przetwarzającą pakiet. Pole `dev` jest wskaźnikiem na strukturę reprezentującą urządzenie sieciowe. Składowa `priv` jest wskaźnikiem na obszar pamięci przechowujący dane prywatne funkcji przetwarzającej pakiety. Pole `pf` przechowuje identyfikator rodziny protokołów, której pakiety będą przetwarzane przez funkcję. Pole `hooknum` przechowuje numer uchwytu, a `priority` priorytet funkcji, który określa porządek w jakim funkcje związane z uchwytem są wykonywane (np. stała `NF_IP_PRI_FIRST` oznacza najwyższy priorytet).

Filtr sieciowy

```
1  struct nf_hook_ops
2  {
3          struct list_head          list;
4          nf_hookfn                 *hook;
5          struct net_device         *dev;
6          void                      *priv;
7          u_int8_t                  pf;
8          unsigned int              hooknum;
9          int                       priority;
10 };
```

Listing 1: Definicja typu `struct nf_hook_ops`

Filtr sieciowy

Struktury typu `struct nf_hook_ops` są rejestrowane przy użyciu funkcji `nf_register_net_hook()`, a wyrejestrowywane przy pomocy funkcji `nf_unregister_net_hook()`. Typ wartości zwracanej przez funkcję przetwarzającą pakiet to `unsigned int`[3]. Przyjmuje ona trzy argumenty: adres obszaru pamięci przechowującego jej prywatne dane (jest przekazywany przez parametr typu `void *`), adres bufora pakietu (bufor jest typu `struct sk_buff`) i adres struktury, która przechowuje stan uchwytu. Ta struktura jest typu `struct nf_hook_state`.

[3] Na wcześniejszych slajdach opisano możliwe wartości do zwrócenia.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!