1. Wprowadzenie

Systemy Uniksowe były jednymi z pierwszych systemów operacyjnych, które posiadały zaimplementowaną obsługę sieci komputerowych. Obecnie są one często używane jako systemy dla serwerów. Linux jest najbardziej znaczącym przykładem takiego zastosowania. Ten materiał jest przeglądem podstawowych informacji na temat obsługi interfejsów sieciowych przez jądro Linuksa. Ze względu na stopień skomplikowania podsystemu sieciowego pominięta została większość szczegółów. Wiadomości zawarte w materiale zostały podzielone na trzy części: ogólny schemat obsługi sieci, skoncentrowany wokół ścieżek przetwarzania wysyłanych i odbieranych pakietów, schemat budowy sterowników obsługi urządzeń sieciowych, ze szczególnym uwzględnieniem NAPI, oraz budowę i działanie filtra sieciowego (ang. *netfilter*), służącego głównie do budowy zapór sieciowych.

2. Ogólny schemat

Rysunek 1 opisuje schematycznie przetwarzanie pakietów wychodzących i nadchodzących [1]. Wynika z niego, że jądro Linuksa wykonuje czynności związane z obsługą trzech warstw modelu ISO/OSI - warstwy łącza, sieci i transportowej. Wysyłanie danych przez proces użytkownika odbywa się za pomocą odpowiednich wywołań systemowych, które aktywują metodę `write()` z obiektu pliku związanego z gniazdem sieciowym procesu. Ta z kolei wywołuje, w zależności od użytego protokołu transportowego, funkcję `tcp_sendmsg()` lub `udp_sendmsg()`. Po zbudowaniu nagłówków właściwych dla odpowiedniego protokołu funkcje te wywołują odpowiednio `ip_queue_xmit()` lub `ip_push_pending_frames()` odpowiedzialne między innymi za utworzenie nagłówka protokołu IP. Pakiet, który otrzymał wszystkie wymagane nagłówki przekazywany jest do sterownika interfejsu sieciowego za pomocą funkcji `dev_queue_xmit()`. Zanim pakiet zostanie wysłany, jego trasa jest ustalana za pomocą funkcji `ip_route_output_key()`. Sprawdza ona pamięci podręczne lub, w razie konieczności, tablice routingu i w przypadku pakietów wysyłanych do innych komputerów w sieci każe je przetworzyć funkcji `ip_output()`. Kiedy urządzenie sieciowe odbiera ramkę danych, to na ogół generuje przerwanie. Istnieją wyjątkowe sytuacje, kiedy tego nie czynni. Będą one opisane w części poświęconej NAPI. Przerwanie takie generowane jest również wtedy, kiedy zakończy się transmisja ramki lub gdy pojawi się błąd transmisji (jest to zachowanie opcjonalne). Sterownik urządzenia po odebraniu przerwania alokuje pamięć na bufor pakietu i ustawia wskaźnik bufora na nagłówek IP. Taki pakiet jest przesyłany do funkcji `netif_rx()`, która umieszcza go w kolejce. Pakiety z kolejki są przetwarzane przez funkcję `ip_rcv()`. Ta z kolei, wywołuje funkcję `ip_local_deliver()`, a ona w zależności od protokołu pakietu uruchamia albo funkcję `tcp_v4_rcv()`, albo `udp_rcv()`. Następnie wywoływane są funkcje, które sygnalizują procesowi oczekującemu, że pakiet został odebrany. W przypadku protokołu TCP jest to `tcp_data_queue()`, a w przypadku UDP, `udp_queue_rcv_skb()`.

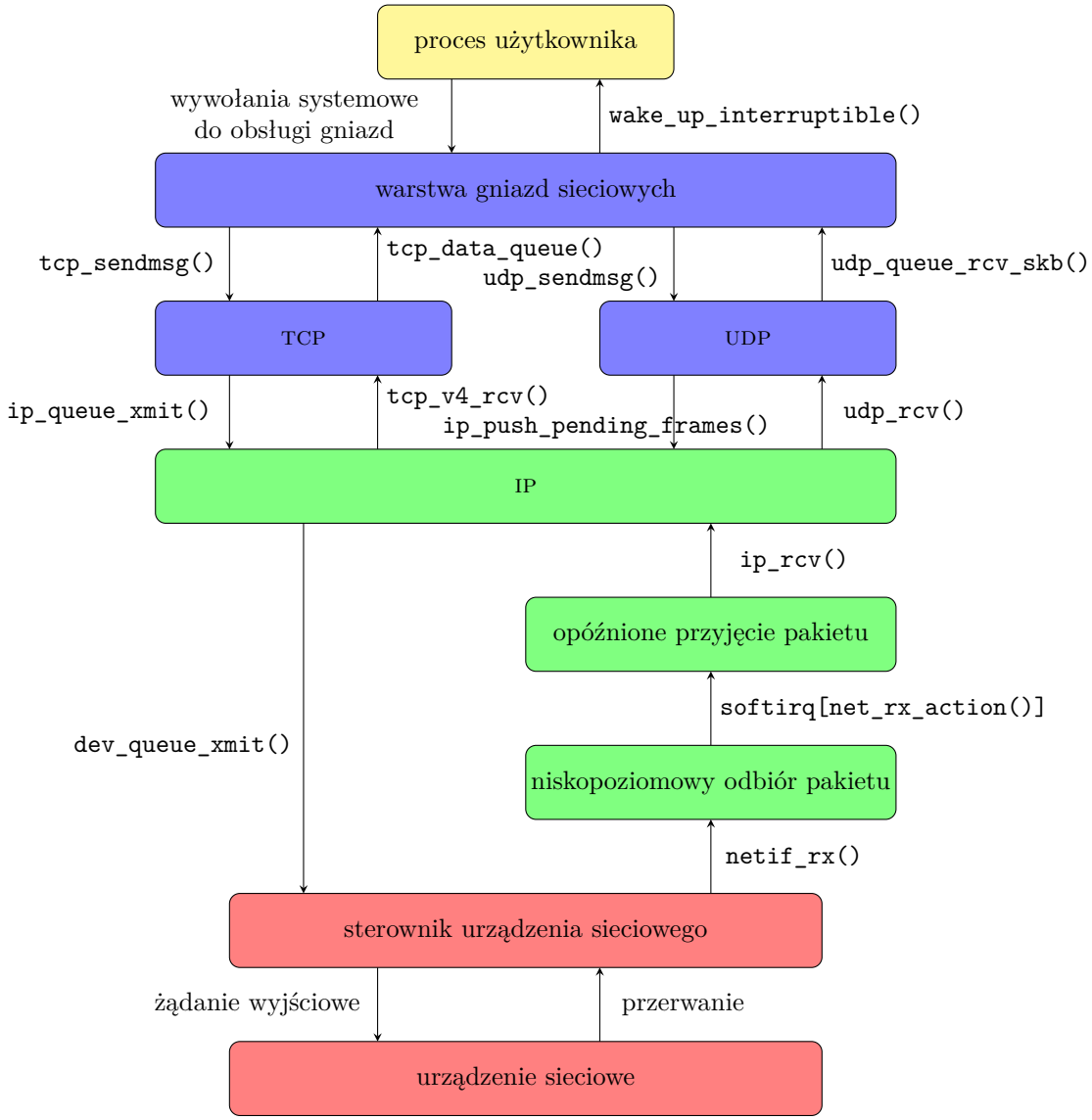
Główną strukturą danych używaną przez podsystem sieciowy jest bufor na pakiety o nazwie `sk_buff`, typu `struct sk_buff`. Struktura ta zawiera nie tylko dane pakietu, ale również metadane niezbędne do ich przetwarzania, umieszczone w nagłówku. Bufor pakietu został tak zaprojektowany, aby operacje przenoszenia go między kolejkami były wykonywane wydajnie. Jeśli zachodzi konieczność jego kopiowania, to kopiowany jest wyłącznie nagłówek. Zawiera on trzy pola, które wskazują z kolei na prywatne nagłówki dla każdej z warstw sieci z osobna, tzn. `transport_header` wskazuje na nagłówek warstwy transportowej, `network_header` na nagłówek warstwy sieciowej, a `mac_header` na nagłówek warstwy łącza. Bufory są powiązane w większą strukturę, która jest kolejką dwukierunkową.

3. Sterowniki urządzeń sieciowych

Główną strukturą danych używaną przez sterowniki urządzeń sieciowych jest `struct net_device`[2]. Reprezentuje ona dany interfejs w systemie. Do najważniejszych pól tej struktury należą `mtu`, które określa maksymalny rozmiar ramki, którą może obsłużyć urządzenie, `flags` określa stan urządzenia,

[1] Część napisana na podstawie: *network_overview* (`http://www.linuxfoundation.org/collaborate/workgroups/networking/network_overview`) oraz William Stallings, „Systemy operacyjne", PWN, Warszawa 2009.

[2] Część napisana na podstawie: *network_overview* (`http://www.linuxfoundation.org/collaborate/workgroups/networking/network_overview`) oraz *napi* (`http://www.linuxfoundation.org/collaborate/workgroups/networking/napi`)



Rysunek 1: Przetwarzanie pakietów wychodzących i nadchodzących w jądrze Linuksa

`dev_addr`, zawiera adres MAC, `promiscuity` liczba żądań ustawienia interfejsu sieciowego w trybie bezładu, `ip_ptr`, wskaźnik na dane specyficzne dla protokołu IP w wersji 4.

We wczesnych wersjach sterowników urządzeń sieciowych odebranie każdego pakietu było sygnalizowane przerwaniem. Prowadziło to do dużego obciążenia systemu w przypadku dużego ruchu sieciowego. Dlatego w wersjach 2.5/2.6 jądra wprowadzono nowe API dla sterowników takich urządzeń, które określono mianem NAPI (New API). Umożliwia ono przełączenie urządzenia w tryb przeglądania (ang. *polling*), co pozwala mu zakumulować większą liczbę pakietów, które w późniejszym terminie zostaną przetworzone przez jądro. Dzięki temu spada liczba generowanych przez nie przerwań i tym samym obciążenie systemu. To rozwiązanie pozwala także na odrzucanie pakietów zanim dotrą one do jądra (tzw. dławienie pakietów). Aby można było użyć NAPI konieczne jest wsparcie sprzętowe ze strony urządzenia w postaci tzw. cyklicznego bufora dla transmisji DMA (ang. DMA *ring*) lub odpowiednio dużego miejsca w RAM komputera na bufory dla bezpośrednich transmisji do pamięci operacyjnej.

4. Filtr sieciowy

Filtr sieciowy (ang. *netfilter*) jest (w uproszczeniu) zestawem wskaźników na funkcję (uchwytów) rozmieszczonych w strategicznych miejscach stosu sieciowego, które umożliwiają implementację zapór sieciowych (ang. *firewall*) oraz rozwiązań typu NAT (ang. *Network Addresses Translation*). Funkcje te są na ogół dostarczane w modułach jądra i umożliwiają tworzenie własnych zapór sieciowych[3]. Jest pięć punktów w stosie sieciowym (patrz rysunek na początku), gdzie mogą zostać „podłączone" takie funkcje:

NF_IP_PRE_ROUTING
: funkcja skojarzona z tym uchwytem jest wywoływana zaraz po odebraniu pakietu,

NF_IP_LOCAL_IN
: funkcja skojarzona z tym uchwytem przetwarza pakiety, które przeznaczone są do odbioru,

NF_IP_FORWARD
: funkcja skojarzona z tym uchwytem przetwarza pakiety, które mają być przesłane do innego komputera,

NF_IP_POST_ROUTING
: funkcja skojarzona z tym uchwytem przetwarza pakiety, dla których została określona trasa i które mają zostać wysłane,

NF_IP_LOCAL_OUT
: funkcja skojarzona z tym uchwytem przetwarza pakiety, które zostały wysłane lokalnie.

Każda z funkcji może wykonać dowolną operację na pakiecie i jego zawartości, ale musi zwrócić na koniec jedną z następujących wartości: NF_ACCEPT - pakiet został zaakceptowany do dalszego przetwarzania, NF_DROP - pakiet został odrzucony, NF_REPEAT - należy powtórzyć działanie funkcji dla tego pakietu, NF_STOLEN - funkcja, „wykrada" pakiet, co oznacza, że będzie on przetwarzany w inny sposób niż pozostałe pakiety, NF_QUEUE - pakiet jest umieszczany w kolejce do przestrzeni użytkownika, NF_STOP - dalsze przetwarzanie pakietu jest wstrzymywane. Funkcje przetwarzające pakiety są reprezentowane za pomocą struktury zdefiniowanej następująco:

```
struct nf_hook_ops
{
        struct list_head        list;
        nf_hookfn               *hook;
        struct net_device       *dev;
        void                    *priv;
        u_int8_t                pf;
        unsigned int            hooknum;
        int                     priority;
};
```

Pole `list` służy do łączenia takich struktur w listę, co umożliwia skojarzenie z jednym uchwytem kilku funkcji przetwarzających, pole `hook` jest wskaźnikiem na funkcję przetwarzającą, pole `dev` jest wskaźnikiem na strukturę urządzenia sieciowego, pole `priv` jest wskaźnikiem na obszar pamięci zawierający dane prywatne dla funkcji przetwarzającej, pole `pf` zawiera identyfikator rodziny protokołów, których pakiety będą przetwarzane, pole `hooknum` zawiera numer uchwytu, a `priority` priorytet, decydujący o kolejności uruchomienia (np. NF_IP_PRI_FIRST - funkcja jest wywoływana jako pierwsza). Struktury te rejestruje się w systemie za pomocą wywołań funkcji `nf_register_hook()`, a wyrejestrowuje za pomocą funkcji `nf_unregister_hook()`. Każda z funkcji przetwarzających musi mieć odpowiedni prototyp: zwracać wartość typu `unsigned int` i pobierać trzy argumenty: adres danych prywatnych (typ `void *`), adres bufora pakietów (typ `struct sk_buff`) oraz adres struktury opisującej stan uchwytu (typ `struct nf_hook_state`).

[3] zobacz: `http://www.paulkiddie.com/2009/11/creating-a-netfilter-kernel-module-which-filters-udp-packets/`