Recent Advances in Technology and Engineering (RATE-2017)
6th National Conference by TJIT, Bangalore

International Journal of Science, Engineering and Technology
An Open Access Journal

Involuntary Exclusion of Unnecessary Packets for Smart Terminals in Wireless LAN Environments

[1]Srinivasa H P, [2]Dr. Kamalesh V N

Abstract

This paper shows the problems with respect to the Linux-based smart terminals when they receive unnecessary packets from Wireless LAN environments. We recommend an algorithm to improve these problems. The unnecessary packets which are received by the smart terminals are stored into a socket buffer instead of dropping immediately. Then the smart terminals transport the packets to an upper layer and the packets are eliminated at the corresponding protocol layer. If the smart terminals receive a lot of unproductive packets, the processing time for these packets are unnecessary overhead. In order to reduce the unnecessary overhead, this paper introduces an automatic elimination algorithm for unproductive packets. In this algorithm, we create a new table which contains information of preventable packets in kernels, and the unproductive packets are filtered according to the tabled information in the premature stage of the MAC layer. Our intended algorithms are implemented by adaptation of the Linux kernel and the WLAN device driver source code.

Keywords: Wireless LAN, Smart terminals, Packet filtering, Network stack, Linux kernel

Introduction

Wireless Internet traffic using Wireless LAN is growing further and more, since the smart terminal's propagation is increased. Most of the smart terminal users use Wireless LAN communications. Wireless LAN offer's communication services through the AP's (Access Point) relations and smart terminals transmit numerous packets to the AP or receive many packets from the AP. The smart terminal receives the useful packets for Wireless LAN service and also many unnecessary packets will be received by them. The smart terminal drops the unnecessary packets at the relevant layer, not dropping them immediately upon delivery.

The operations for the preventable packets are continual and the operations of the smart terminals are useless for data processing. If the smart terminal receives abundant unnecessary packets, the processing time for the preventable packets could cause unnecessary overhead on the smart terminals. These problems can be improved by the smart terminals in avoiding the transmission of unnecessary packets to the relevant layer, and drop them at the MAC (Media Access Control) layer. In this paper, we propose an involuntary exclusion algorithm of unproductive packets to improve the smart terminal's inefficient processing. In this algorithm, the smart terminals drop the unsuccessful packets at the MAC layer of the Wireless LAN.

Analysis of Unsuccessful Packets in Wireless Lan

There are three suitable cases that the smart terminals obtain unsuccessful packets. The first case is getting the multicast packets or broadcast packets by the smart terminals. Multicast packets are transmitted for the precise group. The Linux kernel which receives the multicast packets decides to receive or decline the multicast packets by checking the inclusion of the multicast group address of the multicast packets. Within the same Wireless LAN broadcast packets are transmitted to all the systems with respect to the infrastructure network and then the host system receives the packet with this broadcast address as the destination address [1,2,14]. The received multicast or broadcast packet drops at the Network Layer or the Transport Layer by the smart terminals if unnecessary. The second case is for external TCP packets for unfeasible applications. The TCP packets which are received by the smart terminals check the relevant application process and drop them at the Transport Layer.

The third case is with respect to the receiving of routing protocol packets. The routing packets are transmitted by the routing protocol to create a route between the two ends. The routing packets are unnecessary among smart terminals. So these unproductive routing packets are dropped at the network layer by the smart terminals [3,4].

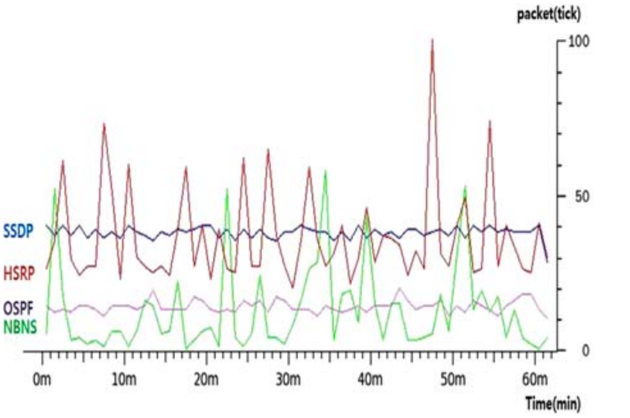

Fig. 1. The number of unproductive packets in Wireless LAN environments

Fig. 1 describes the number of unproductive packets that the smart terminal receives from the Access Point in which these packets are according to types such as SSDP (Simple Service Discovery Protocol), HSRP (Hot Standby Router Protocol), OSPF (Open Shortest Path First) and NBNS (Netbios Name Service) [12]. HSRP packets and OSPF packets are for direction-finding protocols, NBNS packets are relay packets and SSDP packets are multicast packets. These packets are the unproductive packets in the elegant terminals. The consequence analysis of unproductive packets can vary since each AP's constitution is different. In this paper, we use an involuntary elimination algorithm for the unproductive packets.

Analysis of the network stack in Linux kernel

The packet delivery flow of Data Link Layer:

Fig. 2 describes the flow of packets in the Data Link Layer. The inward packets are saved in a socket buffer and send to the next higher layer by using Data Link Layer. When the NIC (Network Interface Card) receives packets, then system interruption occurs and net_interrupt() function is enacted. The inward packets are saved into the socket buffer by interrupt handler and informs the netif_rx() function and in turn informs the do_softirq() function to deliver the packets to the higher layer.

The net_rx_action() function proceeds to cause a software interruption in the queue of network drivers. The netif_receive_skb() function then transmit the packets to the handler function parallel to the higher layer of the packets [3,4,13].

The packet delivery flow of the Network Layer:

The Network Layer finds the routing information from the socket buffer delivered from the Data Link Layer. Fig. 3 describes the Flow of packets in the Network Layer. The ip_rcv() function is called to find the length of the packet, the header and checksum for IPv4 packet. The ip_rcv_finish() function checks the routing information. The ip_local_deliver() function joins the fragmented packet and so on. The Transport Layer receives packets from ip_local_deliver_finish() function.

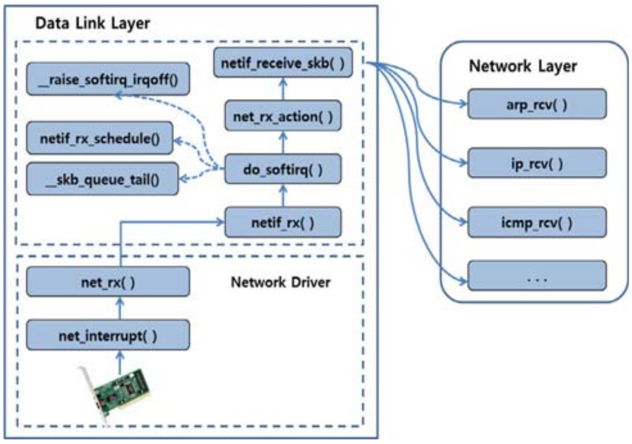

Fig. 2. The packet delivery flow in Data Link Layer.

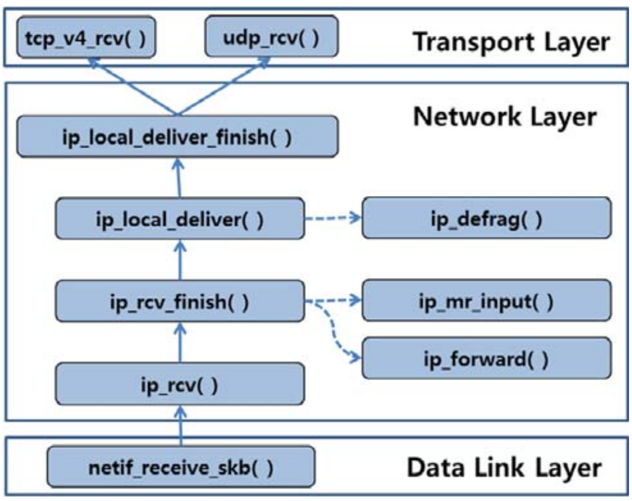

Fig. 3. The flow of IP packets in Network Layer.

The multicast packet is deleted in the ip_rcv_finish() regardless of the host's group information and also the packet of routing protocol[3,4,13]. These packets are needless in the smart terminals. Therefore, we added the filter table algorithm to remove unproductive packets from source parts.

The packet delivery flow of the Transport Layer:

Using Flow control the Transport layer detects an error of the TCP packets in socket buffer and deliveries them to the application layer. Fig. 4 shows the Transport Layer TCP packet flow.

To process the packets ip_local_deliver_finish() function transmits corresponding protocol to the handler function. The packet's type, header length and checksum will be known though Tcp_v4_rcv() function. The session state of TCP is TCP_ESTABLISHED, then Tcp_v4_do_rcv() transmits to the tcp_rcv_established() function[3,4].

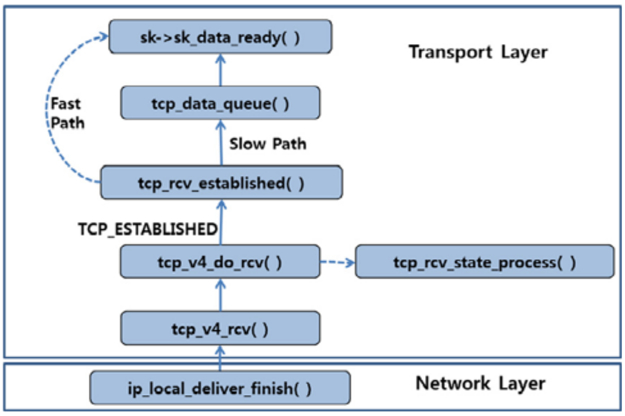

Fig. 4. The Transport Layer TCP packets delivery flow

To find the socket port of the related application layer the tcp_v4_rcv() transmits to the inet_lookup_skb(). The packet will be deleted by using inet_lookup_skb() function if there is no corresponding socket port. Those deleted packets are unpredictable in the smart terminals. Therefore, we added the filter table algorithm to remove unproductive packets from source parts.

Fig. 5 shows the UDP packet flow in the Transport Layer. If, UDP packet is received by the smart terminals, the ip_local_deliver_finish() function transmits to the udp_rcv() function. The checksum errors will be detected by udp4_lib_rcv () function. to find the connected to The Application Layer socket path will be found using udp4_lib_rcv() function by transmitting packets to the udp4_lib_lookup_skb() function.

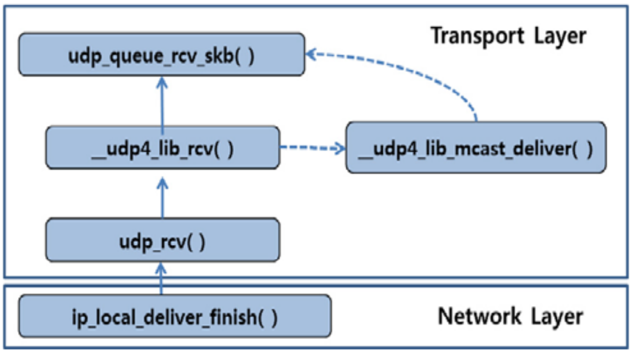

Fig. 5. The UDP packets delivery flow in Transport Layer.

If the __udp4_lib_lookup_skb() function finds the socket for the received UDP packet, it transmits it to the udp_quue_rcv_skb() to save the packet in the socket's reception queue [3,4]. If the __udp4_lib_rcv() function and the __udp4_lib_mcast_deliver() function do not find the relevant socket port, the UDP packets are deleted[3,4,13]. These deleted packets are unnecessary in the smart terminals. Therefore, we added the filter table algorithm to the source parts and deleted the ineffective packets.

Analysis of the IEEE 802.11 MAC device driver:

In this we used a wireless network tool with a RT3070wireless LAN chipset. In Fig. 2, we state that the NIC receives a packet and then executes the Interruption and handler function. The Wireless LAN NIC conducts a system interruption and transmits it to the STARxDone InterruptHandle() function after receiving a packet.

Fig. 6 shows the packet delivery flow driver using RT3070 wireless LAN chipset. The STARxDoneInterruptHandle() function saves the packet from NIC device to the memory buffer and if the packet have to be delivered to a higher layer, this function is enacted. The STAHandleRxDataFrame() function coordinate with the _Legacy_Packet() to convert the IEEE 802.11 packet to an IEEE 802.3 packet [4,15]. After this operation is completed, the announce_ 802_3_packet () function is called to save the packet to the socket buffer and then the packet in the socket buffer is then delivered to the netif_rx() function [3–6]. The operation of this function is similar to the operation of Data Link Layer's packet delivery flow.

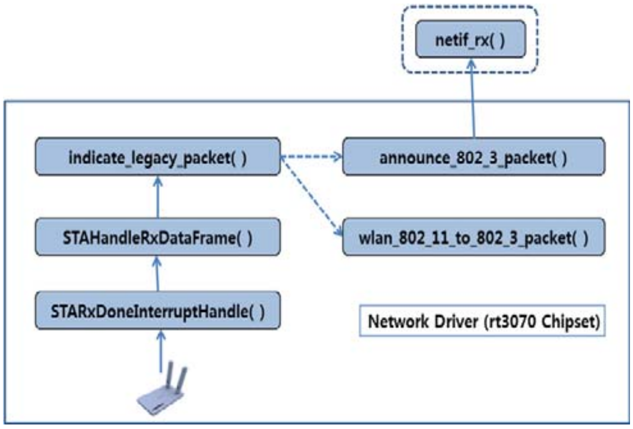

Fig. 6. The packet delivery flow driver using the RT3070 in the wireless network chipset

The automatic elimination algorithm of unnecessary packets

The algorithm implementation:

Fig. 7 describes the automatic elimination of unnecessary packets.

The proposed algorithm creates the filter table to separate the unnecessary packets. The network driver refers to this filter table and checks whether the received packet's socket port information is included in the filter table. If the socket port information exists in the filter table, the packet is sent to the network driver as a MAC layer. Otherwise, the packet is processed normally. The input operations of the filer table are implemented where the ineffective packets are deleted in the Linux kernel. The filtering operation using the filter table is implemented using the network device driver before the ineffective packet is saved in the socket buffer.

The implementation of the Filter Table Algorithm in the Linux kernel

The automatic elimination algorithm of unnecessary packets saves the socket port information of the ineffective packets at higher Layers and refers them to the list of the saved socket ports at lower layers (MAC Layer) as a network driver.

Fig. 8 indicates the structure of a filter table and management process. The filter data structure is composed of three member variables. The list member field is used to connect links between the filter_list structures. The protocol member and dport fields are the variables to save the destination port numbers and protocol numbers of ineffective packets.

In this study, the unpacket_filter_table() function is implemented to manage ineffective packet in the filter table of the. The figure shows the flow chart of the unpacket_filter_table() function.

The unpacket_filter_table() function is called at the source line in which the unnecessary packets are deleted. The unpacket_filter_table() function is requested when the ineffective packet is receive dand the parameter of the function is a socket buffer structure.

The unpacket_filter_table() function first receives the IP header or TCP header in the socket buffer to refer the protocol number and destination port number. Next, the function checks the head of the filter table. It operates memory allocation for the head of filter table in case the head value is zero which means there is no filter table list. If the head value of the filter table already exists, the protocol number and destination port number is saved in the filter_list structure.Finally, the unpacket_filter_table() function transmits to the list_add_tail() function. The list_add_tail() function proceeds to add the filter_list structure including the ineffective packet information to the filter table list [7–10]. The EXPORT_SYMBOL() is used for exporting the filter table of the Kernel Network Layer area to the network device driver [5,6,11].

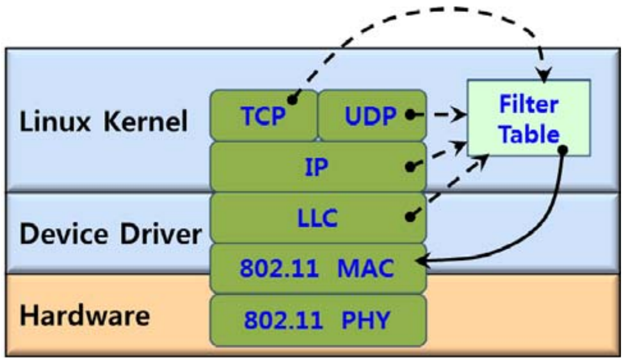

Fig. 7. The Involuntary elimination algorithm of unnecessary packets.

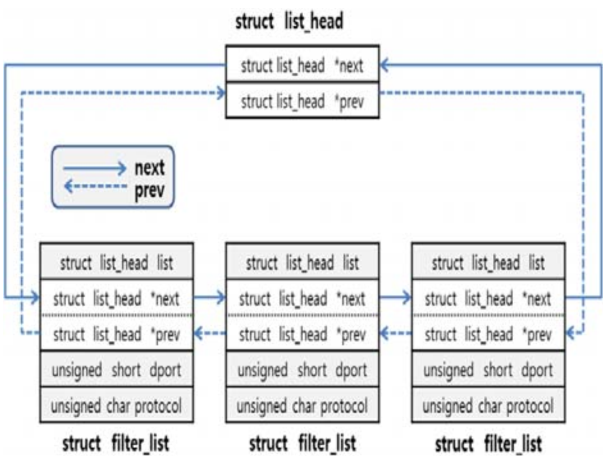

Fig. 8. The structure of Filter Table.

The filtering algorithm implementation in the Wireless LAN network driver

The Packet is necessary or not is decided by network driver according to the filter table and it deletes the packet if it's deemed not effective. In this study, we implemented the unpacket_filter() function in the network driver to eliminate the ineffective packets. The _RX_BLK structure is saved by the network driver of the RT3070 chipset for the received packet before saving the packet in the socket buffer. To get the protocol number and destination port number of the packet the unpacket_filter() function will extracted from the IP header and the TCP header of the packet from the _RX_BLK structure.

The filter table list will be compared with the protocol number and destination port number of the packet. The list mapping is done using the received packet's protocol number and destination port number along with the first filter list of filter table's head if it has value. In the filter table the respective protocol number and port number are found, the received packet is unnecessary. Thus, the ineffective packet is deleted using unpacket_filter() function using the REALEASE_NDIS_PACKET() macro. If the unpacket_filter() function comes to end if it does not find the corresponding information in the filter table.

Conclusion

The unnecessary packets received by the existing smart terminal are dropped at the higher layer after saving them in the socket buffer. Unnecessary overhead may cause on the smart terminal, if the smart terminal receives many unnecessary packets [16,17]. The existing inefficient operations are removed by applying our proposed algorithm is applied in the Linux kernel of smart terminals and the operating cost of the smart terminal could be reduced. In this paper, the ineffective packets are filtered at the lowest layer with a network driver. Thus, for each wireless LAN network driver, a filtering algorithm is applied. The filtering algorithm is applied at LLC layer, which is the most closed and highest layer of the network driver. The packets are filtered using filtering algorithm at the LLC layer without editing the packets. In the future, further study in depth in the network drivers and LLC layer for the efficiency of ineffective packets filtered.

References

[1] LydiaParziale, David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, TCP/IP

[2] Behrouz A. Forouzan, TCP/IP Protocol Suite, fourth ed., McGraw Hill, 2009.

[3] Klaus Wehrle, Frank Pählke, The Linux Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel, Prentice Hall, 2004.

[4] Christian Benvenuti, Understanding Linux Network Internals, O'Reilly, 2005.

[5] Jonathan Corbet, Greg Kroah-Hartman, Linux Device Drivers, third ed., O'Reilly, 2005.

[6] SreekrishnanVenkateswaran, Essential Linux Device Drivers, Prentice Hall, 2008.

[7] Robert Love, Linux Kernel Development, second ed., Sams Publishing, 2005.

[8] Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel, third ed., O'Reilly, 2005.

[9] Wolfgang Mauerer, Professional Linux Kernel Architecture, Wiley Publishing, 2008.

[10] Robert Love, Linux Kernel Development, third ed., Addison-Wesley, 2010.

[11] Wolfgang Mauerer, Linux¢ç Kernel Architecture, Wiley Publishing, 2008.

[12] William Stallings, Data and Computer Communications, ninth ed., Pearson/Prentice Hall, 2010.

[13] Mohamed DafirEch-Cherif El Kettani, Brahim En-Nasry, MIdM: an open architecture for mobile identity management, JoC 2 (2) (2011) 25–32.

[14] Fan-Hsun Tseng, Li-Der Chou, Han-Chieh Chao, A survey of black hole attacks in wireless mobile ad hoc networks, in: HCIS 2011, 2011.

[15] D. Javale, Performance evaluation of wireless image transmission: an approach using embedded system, in: Hybrid Intelligent Systems (HIS), 12th International Conference on, 2012.

[16] Shao-Cheng Wang, Brahim En-Nasry, BEWARE: background traffic-aware rate adaptation for IEEE 802.11, IEEE/ACM Transactions onNetworking 19 (2011).

Author's details

[1]Associate Professor, Dept. of C S E, T. John Institute of Technology, Bangalore, Karnataka, srinihp@yahoo.co.in.

[2]Professor & Dean, Dept. of C S E, T. John Institute of Technology, Bangalore, Karnataka