Station Disassociaciation Problem in Hosted Network

Artyom Shal
Software Engineering department
The Higher School of Economics
Moscow, Russia
artiom.shal@gmail.com

Abstract — **hosted network technology gives an opportunity to create the virtual access point. However, client stations are forced to disassociate from AP due to poor configuration. This paper proposes solution to the issue of station disassociation in the hosted networks.**

Keywords — hosted network, Wi-Fi, TCP/IP.

I. INTRODUCTION

Microsoft hosted network is not a new technology, however, it is not examined enough yet. Although it gives the opportunity to arrange fully qualified access point with no additional hardware required, users face connectivity problems too often. Virtual access point drops the connection with users' PC frequently for no apparent reason (at first sight). This action is very disturbing and disruptive. This paper is aimed to uncover possible issues and pitfalls of the powerful technology.

II. FIRST INVESTIGATION

The first apparent reason for station disassociating from virtual AP is that TCP and Wi-Fi are not perfectly combined. The nature of the IEEE 802.11 technology causes packet delay and loss rate, which triggers TCP congestion control mechanism [1]. This may lead to performance degradation. However, connection breaks were not reported on such scenarios. Possible reason for connection breaks could be specific features of Microsoft TCP/IP stack. Virtual access point may behave differently compared to real devices, indeed. Responsibility for smooth operation of the hosted network technology is on NIC manufacturers. Hence, we assume that virtual AP is fully compliant to 802.11 set of standards. What is the reason for such behavior?

A. Testing

A clear behavior pattern was discovered after performing tests. The testing involved *Microsoft Network Monitor* tool for capturing and analyzing wireless network traffic. The monitoring showed that the connection was fine when the heavy data transfer existed, e.g. video stream. On the other hand, when there was no network activity for more than 10 seconds connection was breaking. This is a rather infrequent situation for ordinary users, as many network services (like NetBIOS) on client stations usually communicate with each other. Yet, it is common enough for corporate environment, where security policies prohibit using many services. This may lead to total blackout in network activity and thus to frequent connection breaks.

To measure the issue we used Windows API on the side of virtual AP. To monitor Wi-Fi network events, we registered system notifications from miniport driver. For this, we used *WlanRegisterNotification* function. The function was called in the following way:

```
DWORD prevNotif = 0;
DWORD lastError = WlanRegisterNotification(
    handle(),
    WLAN_NOTIFICATION_SOURCE_ALL,
    TRUE, //Ignore duplicate
    (WLAN_NOTIFICATION_CALLBACK)handleNotification,
    NULL,
    NULL,
    &prevNotif
    );
```

To get the state of Wi-Fi NIC we handled specific message type in *handleNotification* callback function.

```
VOID handleNotification(WLAN_NOTIFICATION_DATA
*wlanNotifData, VOID *p)
{
switch(wlanNotifData->NotificationSource) {
case WLAN_NOTIFICATION_SOURCE_HNWK:
   switch(wlanNotifData->NotificationCode){
   case wlan_hosted_network_peer_state_change:
   ...
   }
...
}
}
```

Three devices were used for testing:

- Dell Latitude E5420 laptop with NIC Intel Centrino Advanced-N 6205
- Samsung Galaxy S3 smartphone with Samsung Exynos 4 Quad system on chip
- Macbook Air 13 laptop with NIC Realtek RTL8188CU Wireless LAN 802.11n

The results for Samsung device were the worst. It couldn't connect to access point at all. Dell device with Windows 7 OS on board showed good results. Virtually no disassociations were detected. Macbook Air laptop was attempting to reconnect the access point every 10-15 seconds (Fig.1).

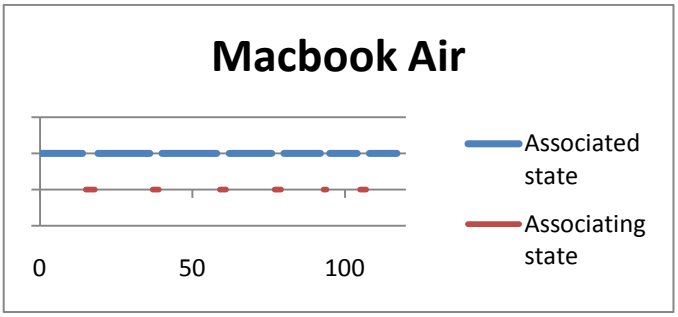

Fig 1. Macbook Air connection state pattern

B. DHCP server integration

The easiest way to fix this issue is to send stub packets. One candidate is ICMP packets used by ping utility. However, the obstacle is that we need to know the IP address of every connected client. The only way to know this at application layer is to allocate IP addresses dynamically through DHCP server.

Starting the wireless Hosted Network typically involves the launch of Internet Connection Sharing (ICS) service in standalone mode. This, in turn, leads to DHCPv4 server to begin providing private IPv4 addresses to connected devices. In this mode, only the DHCPv4 server is operating. This is a special operation mode for ICS and is only made available through the wireless Hosted Network. A user or application are not able to directly start and stop standalone ICS through public ICS APIs or *netsh* commands. Moreover, there are no ways to manage DHCP server operation. Therefore we had to stop ICS manually and to use some open source alternative.

To stop ICS in standalone mode we used simple workaround: the connected key in Windows registry was deleted. The OpenDHCP server is a good alternative to ICS DHCP server. After small modifications, we obtained the following result. The server was receiving the acknowledgement message and starting the ping utility. This simulated activity was enough to keep client stations connected.

C. Results

The tests of the modified DHCP server showed a much more steady connection for many devices. Still, the results were disappointing, as connection was still breaking. Deeper testing with wider variety of devices and different usage scenarios revealed that problem is not at transport or network layer [2] of the OSI model. It is somewhere at the underlying layers.

III. DEEPER EXPLORATION

To uncover the issue of station disassociation at data link layer we had to use special hardware and software. In particular, Proxim ORINOCO wireless network interface card was used to capture WLAN frames. This NIC can work in promiscuous mode, which makes the controller pass all received traffic to the central processing unit (instead of only passing the frames that the controller intended to receive). To monitor network activity the CommView software was used. The application has WLAN-specific features, such as displaying and decoding of management and control frames.

A. Monitoring

Tests showed that the disassociation frame (sent from virtual access point) was the reason for dropping connection. The reason field in that frame was "disassociated due to inactivity". This indicates that either the station's NIC is not sending probe frames frequent enough or that software-based AP cannot see them. We think that the latter is more likely due to specific feature of SoftAP—it shares the common processing unit with virtual station adapter. If proper buffering on NIC is not present, this may lead to a situation when AP is halted to process station operations and cannot process its own frames. This might be fixed by proper configuration during the process of association and initial handshake [3]. The virtual access point should notify the stations that more frequent probe requests are needed, as the listen intervals decreased.

B. Solution

As Microsoft doesn't provide any public API to configure hosted network we can manage it only through NDIS driver stack. NDIS stack has several types of drivers: protocol, miniport and filter (intermediate). Miniport driver is the prerogative of NIC manufacturer, so the filter driver is an appropriate tool to interact and affect the adapter.

The miniport driver notifies the filter driver on every event including virtual AP events. Using lightweight filter driver we modified the listen interval in beacon frames. This frames set short packet buffer, which forced the client stations to make probe requests more frequently. This prevents the AP from sending disassociation requests.

To access the configuration of the beacon frames we used the OID_DOT11_BEACON_PERIOD object type, which requests the miniport driver to set specified value of the IEEE 802.11 at *dot11BeaconPeriod management information base (MIB) object* . This object is used by the 802.11 station for scheduling the transmission of 802.11 beacon frames. It also represents the *Beacon Interval* field of the 802.11 Beacon and Probe Response frames sent by the station.

The data type for OID_DOT11_BEACON_PERIOD is a ULONG value that specifies the beacon period in 802.11 time units (TU). One TU is 1024 microseconds. The dot11BeaconPeriod MIB object has a value from 1 through 65535.

C. *Results*

The tests showed a stable connection for all reference devices. devices operating in power saving mode (Samsung Galaxy S3) can go asleep in ATIM window if there are no announcements [4]. However, right configured beacon intervals fixed this issue.

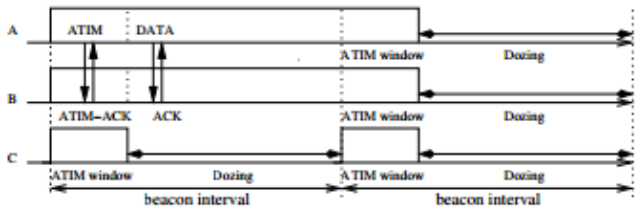

Fig 2. ATIM messages announcement

Another specific case is channel switch. The device that has the ability to scan networks in background may switch channels for short periods. Devices with this feature (Macbook Air) may miss beacon announcement. NDIS configuration fixes this issue as well.

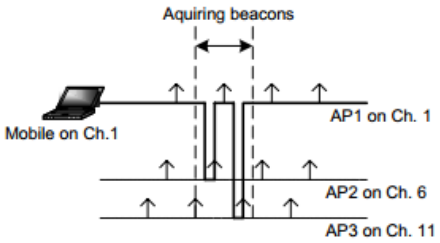

Fig 3. Channel switch for network scan

IV. CONCLUSION

It is clear that hosted network have some pitfalls. It is advisable for NIC manufacturers to provide not only standard-compliant devices, but also drivers that can avoid many pitfalls of the 802.11 protocol stack. The solution for station disassociation issue was given in this paper. It may find application in hot spot software.

REFERENCES

[1] M. Franceschinis, M. Mellia, M. Meo, M. MunafoMeasuring “TCP over WiFi: A Real Case,” 1st workshop on Wireless Network Measurements (Winmee), Riva Del Garda

[2] V. P. Kemerlis , E. C. Stefanis , G. Xylomenos , G. C. Polyzos “Throughput Unfairness in TCP over WiFi” Proc. 3rd Annual Conference on Wireless On demand Network Systems and Services (WONS 2006)

[3] V. Gupta, M. K. Rohil, “Information Embedding in IEEE 802.11 Beacon Frame,” Proc. National Conference on Communication Technologies & its impact on Next Generation Computing CTNGC 2012

[4] H. Coskun, I. Schieferdecker, Y. Al-Hazmi “Virtual WLAN: Going beyond Virtual Access Points” Electronic Communications of the EASST, Volume 17, 2009

[5] P. Bahl “Enhancing the Windows Network Device Interface Specification for Wireless Networking”, Microsoft Research