Бесшовная миграция (роуминг) Wi-Fi для клиентов Linux и Android

С wpa_supplicant из GIT теперь можно настроить в Linux прозрачную миграцию, позволяющую переключиться на другую точку доступа без разрыва соединений приложений, в том числе продолжается вещание мультикастовоего IPTV, а выполнение iperf в большинстве случаев не умирает, а лишь деградирует по скорости в момент миграции. При этом даже корректно отрабатывает вариант с несколькими AP в разных поддиапазонах 5ГГц.

Для настройки роуминга потребуется (на примере Mageia Linux 5):

1) wpa_supplicant из git. Можно взять подготовленный автором src.rpm

```
wget http://wive-ng.sf.net/downloads/wpa_supplicant-2.6-1.mga5.src.rpm
```

и собрать командой

```
rpmbuild --rebuild wpa_supplicant-2.6-1.mga5.src.rpm
```

2) переключить его на использование nl80211 вместо wext, драйвер вашего wifi-модуля должен полностью поддерживать nl80211 (при использовании wext всегда будут долгие периоды реконнекта с потерей соединений пользовательских приложений) так как требуется, чтобы SME был на уровне wpa_supplicant.

Большинство дистрибутивов по умолчанию использует WEXT, т.е. до сих пор далеко не все драйверы wifi, даже входящие в состав ядра, умеют nl80211 (iwlwifi точно умеют, ath*k тоже должны, в rt28xxx по факту даже сканирование не работает). Проверить, что используется на данный момент, можно, посмотрев, с какими опциями запущен supplicant

```
ps ax | grep wpa_supplicant
```

должен выдать что-то типа

```
25201 ? Ss 0:01 /usr/sbin/wpa_supplicant -B -i wlan0 -c
/etc/wpa_supplicant.conf -D nl80211 -f /var/log/wpa_supplicant.log
```

Для этого редактируем /etc/sysconfig/network-scripts/ifcfg-имя_интерфейса

Важно установить:

```
WIRELESS_WPA_DRIVER=nl80211
WIRELESS_WPA_REASSOCIATE=no
MII_NOT_SUPPORTED=yes
USERCTL=yes
```

В /etc/wpa_supplicant.conf везде отключаем рандомизацию MAC. Включаем фоновое сканирование:

```
bgscan="simple:30:60:200"
```

Т.е. при уровне ниже -60 будем начинать время от времени "щупать эфир" на предмет наличия кандидатов для миграции и по возможности мигрировать. Вместо фонового сканирования можно было бы использовать данные из RRM, но этой логики в wpa_supplicant пока нет.

Также следует обратить внимание на используемый DHCP-клиент. Важно, чтобы он не сбрасывал адрес на интерфейсе (иначе сбросится состояние соединений приложений) по сигналу от ifplugd. Я использую штатный "
"Internet Systems Consortium DHCP Client 4.3.3-P1". Ну и естественно, точки доступа и опорная сеть должны нормально отрабатывать ситуацию с перемещением клиентов.

В прошивке Wive-NG для этого сделано абсолютно всё, что возможно. Больше ничего не требуется. Проверено на адаптерах Intel I3160/I7260. Выпинывать их принудительно не надо - оно само инициирует переход без всяких проблем. Под Windows эти карты ведут себя аналогично, т.е. без проблем мигрируют с сохранением пользовательских соединений, достаточно лишь в настройках драйвера увеличить агрессивность роуминга.

Настройка миграции Android-устройств с беспроводными чипами от Atheros/Qualcomm.

Достаточно давно atheros/qualcomm добавили в свои драйверы вполне внятную логику handover (миграции внутри плоской L2 сети, с точки зрения клиента, с множественными AP, на которых установлен один SSID). Собственно, тот самый роуминг, да ещё и бесшовный (ну если используемые AP не совсем плохи и умеют моментально передавать в опорную сеть критичные данные, например, ARP-ы для обновления ARP-таблиц на устройствах в сети + ещё ряд условий на тему кривости).

Теперь о проблеме. Handover есть, сеть с нормальными AP есть, но чтобы клиент мигрировал, по-прежнему требуется пинок, иначе висит до последнего... Что делать и кто виноват?

Для продолжения нужно само устройство, обязательно рутованное, обязательно использующее радио на чипе qualcomm (например yota phone 2).

Перемонтируем разделы в RW, идём в /system/etc/wifi, видим там файл WCNSS_qcom_cfg.ini - собственно, это основной конфигурационный файл, читаемый драйвером wifi.

Драйверы QCA сами реализуют слои SME/MLME, не экспортируя эти функции на плечи wpa_supplicant. И вся логика управления подключениями и миграцией возложена на них. Wpa_supplicant в Android собран с NO_ROAMING, а значит можно ожидать, что это сделано так же у всех абсолютно чипмэйкеров для Android (файлы конфигурации, естественно, разные, или же, как у Broadcom, интересующие нас параметры к исправлению задаются при компиляции, и изменить их на лету невозможно).

Первая настройка, которая нас будет интересовать в файле конфигурации, это:

```
   # default value of this parameter is zero to enable dynamic threshold
allocation
   # to set static roming threshold uncomment below parameter and set vaule

   gNeighborLookupThreshold=78
```

Эта настройка напрямую отвечает за то, когда клиент начнёт пытаться искать кандидата (форсирует сканирование, или запросит список ближайших AP по RRM и проведёт короткую процедуру скана для подтверждения). И значение у нас выставлено -78дБ...

И вроде бы всё хорошо, но... Мы как обычно забыли, что то, что слышит клиент и то, что слышит AP, может отличаться по уровню на десятки дБ. Обычно в android-клиентах передатчик в 20МГц полосе может выдать 16дБм, в 40МГц в лучшем случае 14дБм. Тогда как со стороны AP обычно имеем как минимум 20дБм дури даже в 40МГц полосе (обычно определяется законодательными ограничениями конкретной страны, для РФ 20дБм в 2.4ГГц и 23дБм в 5ГГц независимо от ширины, хоть в 80МГц эти 23дБм, хоть в 20МГц). Из опыта, при таком раскладе в среднем перекос по уровням в прямой видимости уже в 10 метрах будет составлять около 10-15 Дб, а если есть преграды, то запросто перевалит и за 30 Дб.

Но возьмём 10дБ для простоты. Т.е. когда клиент видит AP с уровнем в -78дБ, AP видит клиента уже с уровнем около -88дБ. Стоит говорить, что нормальной работы при таком раскладе уже не будет (особенно в 2.4ГГц в зашумленном эфире офиса)? TCP, требующий подтверждения доставки, просто встанет колом, голос начнёт квакать и т.д.

Что бы этого избежать (плюс приземлить побольше клиентов, ведь для этого надо заставить всех работать на самой ближней AP, желательно с максимальными рэйтами и без повторов передачи), админ в сети наверняка на AP настроил handoff с уровнем эдак в -75дБ. Т.е. при -75дБ RSSI handoff со стороны AP застрелит такого клиента (при этом на клиенте уровень от AP будет аж -65дБ, т.е. далеко до заветных -78дБ, когда он решит подумать мигрировать). Т.е. будет link beat, сброс стэйтов и обрыв соединений на пользовательской стороне.

Ещё пример - когда AP видит клиента с уровнем в -75дБ, клиент видит AP с уровнем -65дБ!! Или, когда на клиенте видим -78дБ, то со стороны AP клиент будет слышен лишь с уровнем -88дБ.

Т.е. что бы обеспечить нормальный сервис, мы должны отстреливать клиента сильно раньше, чем он сам подумает мигрировать (имеется в виду, с вышеозначенными умолчаниями в драйвере).

Или же нам придётся снижать мощность на AP что бы "уравнять" шансы, что в условиях грязного эфира может привести к катастрофическому падению SNR и как следствие бульканью, хрипам и прочему.

А вот чтобы этого не происходило, достаточно выше обозначенную переменную поправить, выставив значение в диапазоне -65 ~ -70дБ.

Побочный эффект - чуть упадёт скорость у клиента, так как чаще будет выполняться фоновые сканирования, плюс незначительно вырастет энергопотребление. Зато у него "будет повод" начать смотреть, кто есть вокруг, и попытаться мигрировать, не дожидаясь, пока его выпнет точка доступа.

Следующий блок:

```
# CCX Support and fast transition

CcxEnabled=0
FastTransitionEnabled=1
```

CcxEnabled - это поддержка rrm ccx location-measurement, т.е. определения местоположения. Зачастую бесполезна и лишь будет расходовать аккумулятор.

FastTransitionEnabled - поддержка 802.11R, однако в старых драйверах не полная, но хотя бы умеет учитывать MDIE при миграции (работает всегда, если переменная в значении 1). Заметим, что FT, а именно ускорение фазы аутентификации просто при взводе значения переменной в 1 работать не начнёт, так как требуется ещё пропатчить Supplicant и обвязку андроидную (как это делает, например, Samsung в своих топовых моделях). Однако, учёт MDIE - уже польза. Включаем.

Не забываем также отключить 802.11d, иначе встретим много чудес в 5ГГц с DFS каналами. Пусть использование или не использование оных лежит на совести админа сети.

```
# 802.11d support

   g11dSupportEnabled=0
```

Далее блок:

```
   # Legacy (non-CCX, non-802.11r) Fast Roaming Support
   # To enable, set FastRoamEnabled=1
   # To disable, set FastRoamEnabled=0

   FastRoamEnabled=1
```

Включает возможность миграции в любых сетях. Иначе, логика handover будет активироваться, только если клиент видит, что AP вещает поддержку 802.11R в IECAP. И/или используется WPA*-Enterprise.

В новых драйверах, поставляемых в SDK для Android 6.0.1, добавлена ещё одна прекрасная опция:

```
   # Handoff Enable(1) Disable(0)

   gEnableHandoff=0
```

Эта опция (если выставить в 1) позволяет обрабатывать handoff с пинком со стороны AP как сигнал к миграции, т.е. банально не генерируется LinkBeat при kickout со стороны AP, а сразу выполняется попытка перескочить на следующую подходящую выбранную сканом AP, и только если не получилось сгенерировать Link Beat системе (т.е. о том, что его выпнули (kickout), знает только драйвер, и сразу пытается мигрировать, если не получается, то тогда уже сообщает системе, что всё, связи с этой сетью больше нет, надо выбрать другой SSID из тех, которые знает Android).

Пользовательские соединения при этом останутся целыми (ну разве что iperf пострадает, но мессенджеры и голос останутся работать, правда в случае голоса будет квак).

Это слегка нарушает "стандарт" 802.11, но в случае миграции очень полезная штука. После её включения, устройство прозрачно (с точки зрения запущенных приложений) сможет мигрировать даже в сетях, где весь роуминг построен исключительно на отстрелах клиента по уровню.

А вот ещё одна крайне полезная настройка:

```
   #Check if the AP to which we are roaming is better than current AP in terms
of RSSI.
   #Checking is disabled if set to Zero.Otherwise it will use this value as to
how better
   #the RSSI of the new/roamable AP should be for roaming

   RoamRssiDiff=5
```

Дельта уровней между AP для выбора кандидата при миграции. Больше значение - меньше вероятность прилететь назад на ту же AP, но более отложенный старт миграции. Установленных по умолчанию 5дБ маловато. Нужно иметь дельту около 8-10дБ. Для начала выставим 8.

```
   RoamRssiDiff=8
```

Ещё интересная штука:

```
   #Beacon Early Termination (1 = enable the BET feature, 0 = disable)
enableBeaconEarlyTermination=1

   beaconEarlyTerminationWakeInterval=11
```

Эта опция откидывает все маяки, если в этих фрэймах не взведён бит TIM, а клиент спит. Т.е. во сне клиент не может вести пассивный сбор данных о соседних AP. Хорошо для аккумулятора - вероятно, плохо для миграции (надо глубже копать драйвер).

```
   gActiveMaxChannelTime=60
   gActiveMinChannelTime=30
   gActiveMaxChannelTimeConc=60
   gActiveMinChannelTimeConc=30
```

Настройки времени прослушивания эфира при активном сканировании поканально, в мс. Больше значения - больше времени уйдёт на сканирование всего диапазона. Меньше время - быстрее просканируем, но можем половину не услышать.

RRM включается так:

```
   # 802.11K support
   gRrmEnable=1
   gRrmOperChanMax=8
   gRrmNonOperChanMax=8
   gRrmRandIntvl=100
```

Правда, я на доступных мне железках (в смысле, на тех, в которых было выключено и включил, на SGS A5 2017 запросы есть, но там и так всё с миграцией более-менее) так и не дождался ни одного запроса с использованием RRM от Yota Phone. Видимо, отключена поддержка при сборке драйвера.

Это основное, что касается миграции.

Да и ещё:

```
   # 1: Enable standby, 2: Enable Deep sleep, 3: Enable Mcast/Bcast Filter
   gEnableSuspend=3
```

По умолчанию обычно 0. В таком режиме клиент при переходе в режим сна (часто просто при выключении экрана) полностью тушит RF часть, временно просыпаясь только для того, чтобы AP его по таймауту не выпнула. При этом, реализация такова, что и роумингу зачастую становится плохо. Следует установить его в 3, т.е. фильтровать броадкасты и мультикасты во сне, и только.

В файле конфигурации ещё много интересных параметов. Например, куча настроек для энергосбережения, если поотключать которые, миграция становиться более весёлой и точной, но батарея...

Ну и на закуску:

```
   #Channel Bonding
   gChannelBondingMode24GHz=1
   gChannelBondingMode5GHz=1
   gShortGI20Mhz=1
   gShortGI40Mhz=1
```

Поддержка широких каналов (>20МГц) + поддержка SGI. Зачастую для 2.4ГГц поддержка 40МГц отключена, т.е. gChannelBondingMode24GHz установлен в 0. Что, во-первых, не позволяет работать на максимальных рэйтах (150Мбит/с для 1T1R в 2.4ГГц при 40МГц, будет только 72). Увы, проблема в том, что видимо для первого соединения значение перекрывается откуда-то ещё, и даже выставив gChannelBondingMode24GHz=1, сразу мы 150Мбит/с не видит. Но после первой же миграции драйвер плюёт на всех и взлетает в 40МГц полосе.

Также установка gChannelBondingMode24GHz решает проблему совместимости с некоторыми 2.4ГГц AP на Xiaomi и One+ (как обычно, намутили с анонсами в зависимости от региона, в итоге AP и клиент не могут договориться, в какой полосе разговаривать).

Не забываем после правки сохранить, перемонтировать назад системный раздел в RO и перезагрузиться.