­

White Paper

Smartphone VoIP performance on enterprise WLANs

Peter Thornycroft

Rev 1.3 October 2010

## Introduction

This paper examines how smartphones perform on enterprise WLANs, with special attention to the performance of voice over Wi-Fi (VoWi-Fi) and video applications that require uninterrupted, continuous media streams as they move around the building or campus.

Most VoWi-Fi implementations to date have focused on home or residential use. This was a significant first step towards enterprise voice, as it is more demanding than serving Web pages or ‘normal’ data traffic: voice frames must be delivered on-time, without interruptions or errors. But application developers have found that a simple implementation of a VoIP (voice over IP) client is sufficient for the home user on a single, isolated Wi-Fi access point.

As we move into an enterprise WLAN, however, the RF environment becomes more complex. First, there is more data ‘on the air’ as many users share the WLAN. This means quality of service (QoS) priority levels must be invoked. Second, a smartphone on-call while moving through a building will move between APs (access points) quite frequently, so the implementation of inter-AP handover must be very robust. Third, business users on a WLAN are likely to participate in conference calls and other long-duration calls, and to be significant consumers of data services such as email on their smartphones. It is important that protocols to extend battery life are implemented correctly. Of these separate issues, we focus here mostly on inter-AP handover, as it is the most complex and least understood aspect of VoWi-Fi, as well as the most noticeable if performance is lacking.

The underlying theme of this paper is that existing, industry-wide open standards can achieve superior VoWi-Fi performance. Some smartphone vendors already have robust implementations, as a result of earlier VoWi-Fi experience, while others are still in the early stages. By ensuring that they follow the standards, and considering multi-AP WLANs in their use-case models, smartphone silicon, operating system and applications developers can improve and optimize the VoWi-Fi experience.

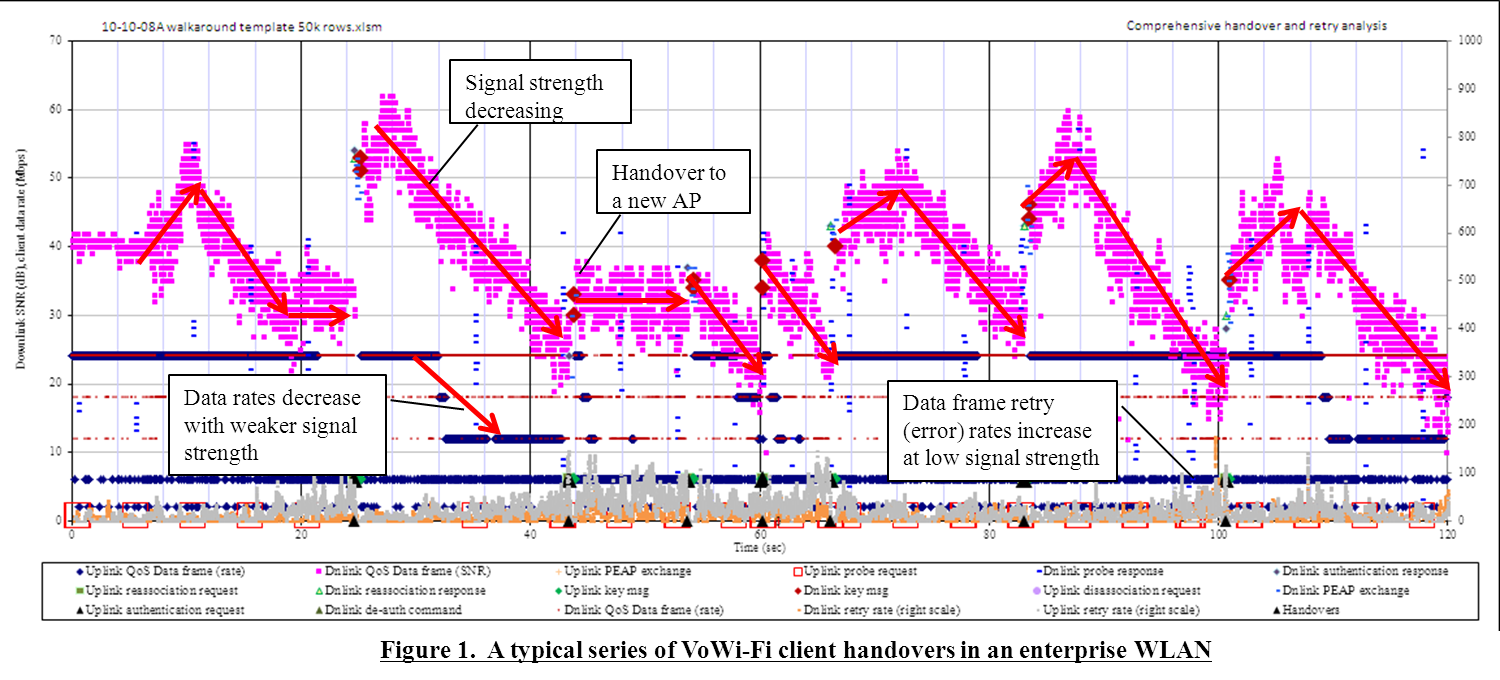
We hope that our analysis in this paper will help WLAN implementers in their assessment of VoWi-Fi technology, and developers in their implementation. VoWi-Fi technology is moving quickly: before this paper is published, it will have been overtaken by better smartphones, better software and performance, but our focus is on what’s available to the user today. The test devices we used are off-the-shelf smartphones with the latest publicly-available software, rather than lab models: the results here represent a snapshot in time. Similarly, there are no special settings or software in the WLAN infrastructure. These results reflect what would be seen, heard and experienced in practical, state-of-the-art networks.

This paper represents a subset of the extensive test and evaluation programs at Aruba Networks, as we continuously evaluate voice- and video-capable mobile devices. The particular examples used here were selected as they are typical of what might be expected in a ‘real’ network, and we have simplified our analysis in order to highlight the most important results – a detailed evaluation of a single device can take more space than we have available here.

## Anatomy of a handover

An on-call VoWi-Fi client device must support a continuous stream of media frames, usually one every 20 msec in each direction of the call. This is relatively easy to accomplish when the client is static, but enterprise WLANs consist of large numbers of coordinated APs, in the order of 20 meters apart. When the user is moving through the building, the device must shift its association from AP to AP to maintain a usable connection, with high signal strength.

In order to analyze handover behavior, we use multi-channel test equipment to capture 802.11 frames on the air, both to and from the client device. With some post-capture analysis, we can construct a narrative for the client as it moved from AP to AP through the building. We are not able to see how the smartphone makes its decisions directly, although phone vendors themselves can use diagnostic code to track their algorithms, but we can infer a decision-making process from observing behavior in a variety of handover events. The graph below follows a smartphone as it moves through and enterprise WLAN.



There is a great deal of information in the graph, but the following are the main points of interest:

* The horizontal scale is in time (seconds), but this relates to distance, as the user moves at a constant 1.6 meters/second through the building. As the route turns corners and passes doorways, RF conditions can change very quickly.
* The magenta dots each represent a downlink data frame, from an AP to the client, on a vertical axis of signal strength (SNR). When the user walks towards an AP, the signal strength increases. At some point, the user’s path moves away from the AP, and signal strength weakens.
* Inter-AP handovers are identified by step-changes in received signal strength, and also by the black triangle markers, showing ‘authentication request’ frames from the client. We expect to see AP signal strength after the handover significantly higher than before, if the client made a good choice of target AP.
* Since the sniffer device is carried next to the smartphone, signal strength on the uplink always appears good. Uplink frames are represented by blue diamonds, sorted on the vertical axis by data rate. Generally, as signal strength weakens, data rates are reduced. Downlink data rates are plotted as red bars on the same scale.
* Retries occur when no ack was received for the original frame. The graph plots uplink and downlink retries separately, on the right-hand scale. Generally, when signal strengths decline, retry rates increase, although the effect is not always linear. We like to see a handover initiated before retry rates rise high enough to affect voice quality.

In the example above, the client executes six handovers over a two-minute circuit of the building. This is somewhat of an upper bound for practical handover requirements – not many people walk as far or as fast as this – but by no means unreasonable as a worst-case for design purposes.

Handover can be analyzed in three stages:

* The client must decide which is the best candidate AP for a handover;
* It must decide when to initiate an inter-AP handover;
* It then executes the re-association to the new AP.

Most lab testing regimes focus only on the third phase, but the information in this paper shows that the other stages are more important in determining performance. While re-authentication protocols must be closely co-ordinated between client and infrastructure, WPA2-enterprise is now widely understood, and implementations seldom diverge very far: once a re-authentication is initiated, it seldom fails because of protocol or state mismatch. Given reasonable conditions, the variation in re-authentication times is in the order of 10%. However, when they occur, errors and failures are usually due to a poor choice of target AP, or to holding the old AP too long before deciding to handover. While outages due to these reasons are generally infrequent – perhaps less than 15% of handovers for a bad case – they can result in media outages of several seconds, which will be noticeable to the listener and can sometimes result in calls dropping.

We can gain an insight into each stage of handover from the graphs generated.

## Deciding on the best candidate AP for handover

A voice client should always maintain a list of handover candidates, because RF conditions can change very rapidly and it may need to make a handover decision quickly, for instance when the user turns a corner or closes a door. Since APs operate on different RF channels, the smartphone must steal time between sending and receiving voice frames to scan both the current channel and other channels and identify APs with good signal strength.

Scanning can be passive or active. Since APs regularly broadcast beacons, it is possible to switch to other channels and monitor the beacons of APs on those channels. But beacons are relatively infrequent, usually repeating every 100 msec, so it can take a while to build up a list of candidate APs in this way. Most clients use active scanning, where they will switch to another channel, transmit a probe request, and remain on the channel for several milliseconds to receive probe responses from audible APs. This should reduce the time off-channel to perhaps 10 or 20 msec rather than the 100+ msec necessary for a passive scan.

The analysis tool cannot show passive scans, but every probe request is tracked, as are responses from nearby APs. By observing the pattern of probe requests we can learn about the client’s scanning algorithm. There are many other tricks to optimizing AP candidate lists, but the pattern of probe requests is a strong determinant of success.

Some smartphones broadcast probe requests every 5 seconds or so, while others don’t probe at all until signal strength falls towards 30dB SNR. There seems to be little difference in performance: it’s the probe pattern when signal strength falls that is important. Most algorithms increase the frequency of probe requests when SNR is below 30 dB, as they need to construct a candidate list: conditions might change very quickly and it is important to have a short list ready to go. A burst of 2 – 3 probe requests per channel every 3 – 5 seconds seems to be successful. The number of channels scanned should be limited to those where the SSID (ESSID) has been seen before, usually 1, 6 and 11 in the USA, to minimize the time spent off-channel, and the overall duration of scanning the environment. Also, probe requests should be directed to the specific SSID used by the enterprise – an ‘open’ request may elicit many unnecessary responses.

Even though active scanning reduces the time off-channel, voice frames can be lost unless the smartphone has an algorithm that times frame intervals. This is difficult to do, and anyway it can be defeated by jittered downlink frames. A better solution is to invoke Wi-Fi Multimedia –power save (WMM-PS), where downlink frames are buffered in the AP and delivered immediately following an uplink frame. Now, the client knows it will not lose downlink frames because it is off-channel when the AP needs to send them.

As we shall see, some clients have very economical probe request behavior, and these tend to be more successful in executing fast, reliable handovers.

As we cannot see the client’s candidate list, we must judge its effectiveness by results. This means looking at the choice of target AP in a handover. The simplest measure of performance is whether the target AP offers higher signal strength than the old AP, and this is readily seen from the graphs. A more nuanced view compares the chosen AP with other possible target APs: these are visible by their earlier probe responses, in answer to probe requests from the client. If the data frames after handover are at the level of the strongest probe response just beforehand, the choice was good. If a stronger signal existed, it may have been a better choice, although if clients broadcast with undirected probe requests, the registered responses may come from other SSIDs or networks that could not be used for handover.

## Deciding when to initiate a handover

Once a smartphone or other VoWi-Fi client has a short list of candidates for handover, it must still make the decision that the moment has arrived. This is more difficult than it seems, because as the graph above shows, normal RF fluctuations cause the signal strength of an AP’s frames to fluctuate over a range of at least 6dB, without adding any perturbations from closing doors or turning corners. VoWi-Fi client designers have found that they must take account of the current AP’s signal strength, sometimes with both short-term and longer-term average levels, as well as the signal from the target AP.

For instance, as a rule of thumb it is normally a good thing to initiate a handover when signal strength from the current AP falls below the 25-30dB level. But if the best candidate AP is only at 30dB, it may represent a worse choice than the current one. Similarly, even though the current signal may be good, a closer AP with an even stronger signal may justify a handover decision. Other parameters such as error rates may be usefully incorporated in the algorithm, to move away from a noisy channel.

Most client designers have an in-built bias towards ‘stickiness’. Their algorithms don’t decide to handover until the situation is dire, signal strengths are low and error rates high. This may be acceptable for data-oriented clients, but with voice services it results in poor call quality, as error rates start to increase non-linearly as signal strengths drop below 20 dB SNR.

## Execute the handover

Just before initiating a handover, the client usually sends a probe request to the target AP to verify it is still available with good signal strength. Then it starts to re-authenticate to the new AP. Depending on the authentication protocol used, this can entail up to 50 or more frames over the air, and take several hundred milliseconds.

While the re-authentication phase of handover is the one that is usually emphasized in lab tests, as it is quite repeatable and test conditions can be closely controlled, it is sometimes a minor contributor to overall handover times.

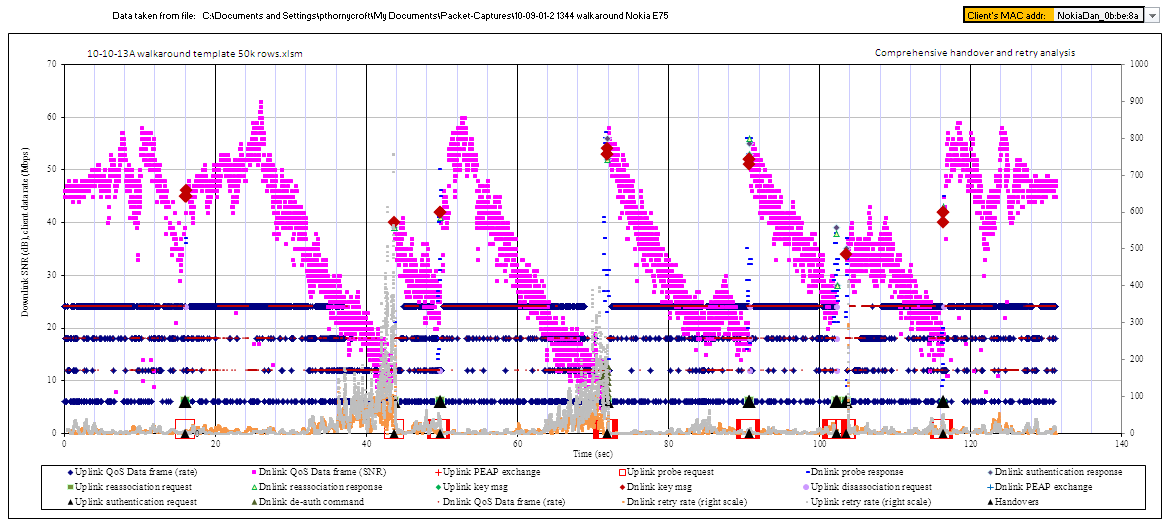
Following the handover, the client is associated with the target AP, and data frames should resume in both uplink and downlink directions.

## Measuring the outage

The simplest measure of handover time is the re-authentication phase: usually from the authentication request to the last key frame completing the protocol. This is the easiest figure to measure, and it can be easily automated in test tools. But the measure that matters to the user is how much of a gap is heard between the last frame received (or sent) on the old AP and the first frame on the new one. This is more difficult to measure, and it depends on higher-layer functions as well as the Wi-Fi protocol.

Indeed, to tell the complete story we should include estimates of how good voice quality was, before and after the handover. In this paper we offer figures for re-authentication protocol and media interruption, along with qualitative comments about retry rates around the handover.

## Nokia E75 handover performance



We start with the Nokia E75 because Nokia was very early to develop Wi-Fi enabled smartphones, and has spent considerable effort optimizing their handover performance in enterprise environments. We use the integral Symbian SIP voice client, configured for Aruba’s Avaya SIP server. The graph above shows excellent handover behavior.

The phone appears to rely on active scanning, but only scans immediately before a handover, and with a small number of directed probe requests. On the graph, probe requests are shown as red squares on the x-axis, and each set of probe requests has an authentication request in the center, showing an immediate handover initiation.

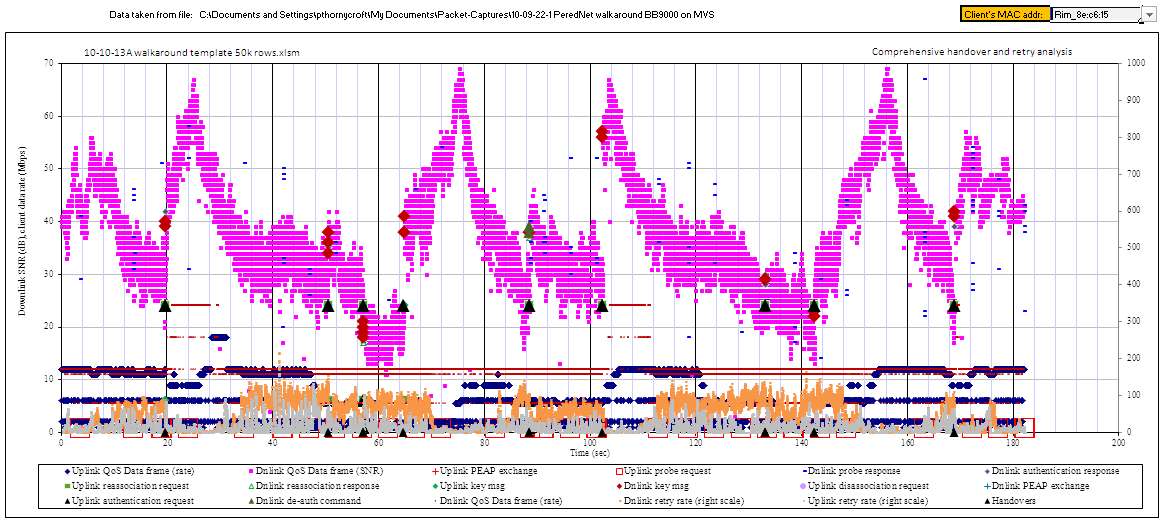
Handovers generally start in plenty of time before voice quality degrades. However, we can see that before the handovers at 44 sec and 72 sec, the client allows signal strength to fall to ~10 dB SNR, which is too low. The retry rates shown by the brown and grey graphs show a marked increase as signal strength drops below 20 dB, and it would have been better if the E75 had initiated handovers some seconds earlier in both cases.

Also, at 102 sec, there are two handover attempts in quick succession. The first attempt is to a heavily-loaded AP, and it rejects the association request to force the phone to load-balance to a neighbor AP (an Aruba feature). The additional handover delay is justified by the better QoS resulting from balanced load across the network. Similar mechanisms can be used for call admissions control on the WLAN.

The choice of target AP is good. All new APs have a signal strength > 30 dB, and usually much higher. (This is possible because the WLAN provides ubiquitous, overlapping coverage from adjacent APs). Comparing the signal strength of probe responses (blue bars) with the actual strength of the new AP indicates that the target chosen is almost always the AP with the strongest signal.

Because Nokia has implemented the 802.11 OKC (opportunistic key caching) standard, all handovers are executed very quickly. The average handover time (authentication to key frame) in this sequence is 23 msec, in a range from 16 to 32 msec. Media-media handover times average 234 msec, with a range from 28 msec to 1.307 sec (for the handover at 102 sec). Without the long handover, the average would be 57 sec.

## RIM BlackBerry 9000 handover performance



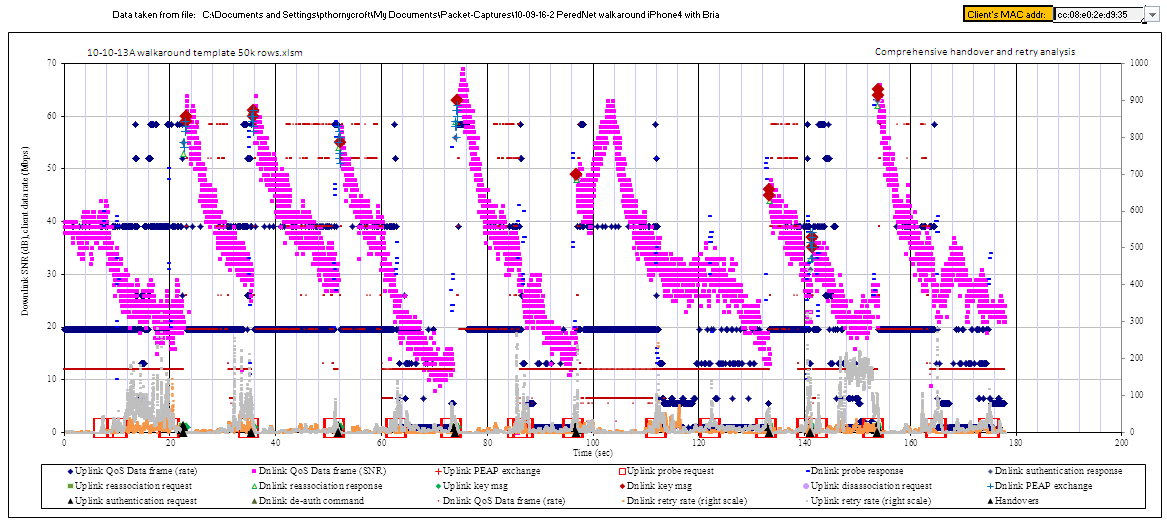
RIM has been involved in VoWi-Fi for some years, notably with T-Mobile’s ‘Wi-Fi calling’ UMA service, and more recently with Mobile Voice System (MVS) which delivers VoWi-Fi to a SIP server via the RIM MVS server. When configuring a BlackBerry smartphone for enterprise WLAN use, it is important to check the ‘inter-AP handover’ option.

The graph shows a pattern of probe requests every 5-10 seconds, curtailed when signal strengths are very high. There appears to be no acceleration of scanning at low signal strengths. More probe requests are seen here than with Nokia, but results are good – handovers are generally timely and accurate. The uplink retry rates never rise very far, although downlink retries are high.

The handovers all start as signal strength drops to 25 dB SNR, a little higher than for the Nokia E75. The exceptions are at 56 sec, which is a poor choice of target AP (although it is not certain that a better AP replied to the coincident probe requests), and 142 sec, where signals are a little weaker. But the BlackBerry smartphone makes very good handover timing decisions.

RIM does not implement OKC, but it offers a more limited variant, PMKc (PMK caching), where a client can return to an AP it has previously been associated with. In this case it was the second run of the day, so the BlackBerry had already cached credentials for all APs in this network and was able to use PMKc everywhere – an unusual situation. Protocol handover times averaged 40 msec, in a range from 33 to 48 msec. Media-media handover delays average 58 seconds, in a range from 46 to 77 msec.

## iPhone4 (iOS-Apple) handover performance



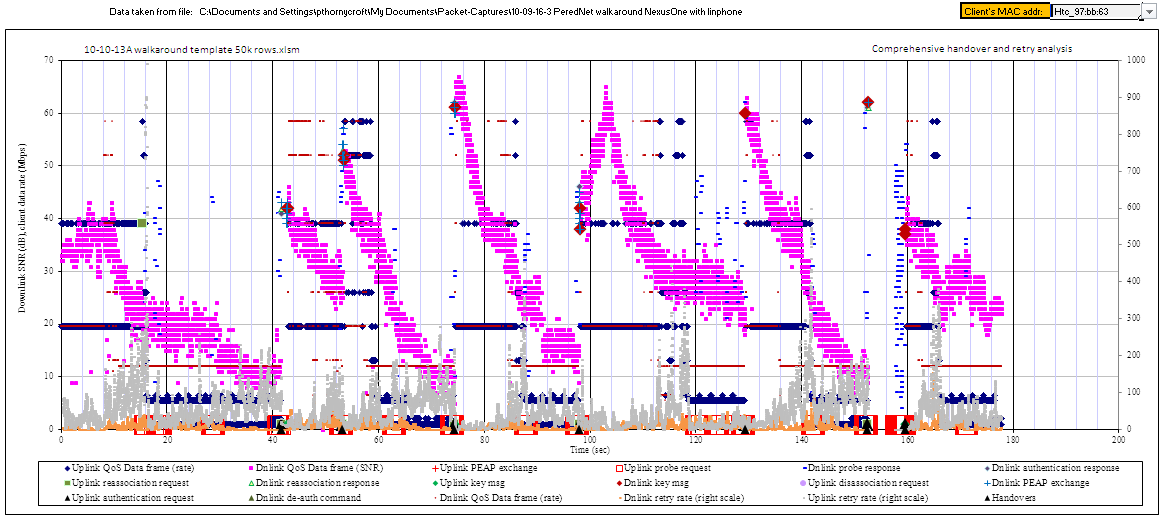
Apple iOS and Android devices present a rather different environment for VoWi-Fi than Nokia or RIM, because the VoIP application is written by an independent developer, rather than being tested as part of the phone-OS-app combination. This loose coupling means the choice of app, in part, determines the VoWi-Fi behavior. For this test we used the Bria client from CounterPath Corporation.

Active scanning probe requests appear at approximately 10 second intervals, usually just before a handover attempt, although sometimes (e.g. at 64 sec) the phone should have started a handover, but didn’t. On this occasion, and again at 90 sec and 133 sec, the phone holds its old AP too long, and signal strength declines, with retries rising, particularly from 12 – 20 seconds, even though suitable target APs exist. On a detailed view, it is clear the iPhone generates many more probe requests than either Nokia or RIM. Also, the scanning pattern includes ‘open’ probe requests, which generate many unneeded replies, reducing network capacity and potentially concealing other responses through co-channel interference.

The choice of target AP is generally very good, with associations picking up at the 40 – 60 dB SNR level. Few probe responses are seen with better signal strength than the chosen AP.

Of the 8 handovers on this graph, three successfully use PMKc (or OKC, it’s impossible to tell from the trace) while the remaining five follow full PEAP re-authentication. For PMKc, the average protocol time is 33msec (31 - 33 msec) and the media-media delay averages 48 msec (38 - 60 msec). Full PEAP averages 438 msec authentication – key frame (422 – 454 msec), and 480 msec (440 – 540 msec) for media-media delay.

## Nexus One (Android-HTC) handover performance



The Android OS does not include a SIP client, so to exercise the phone we used the Linphone open-source application configured as a client on our Avaya server. This app is somewhat rudimentary for enterprises: for instance, it does not use QoS, but we do not believe it has any influence on handover behavior.

The active scanning is even more frenzied than for the iPhone. While there is a basic 10 second pattern, there are several episodes of almost continuous probe requests for a second or more. This is not helpful to the phone – it only takes a few probe requests to discover all neighboring APs – and furthermore, many of the probe requests are of the ‘open’ type, which is wasteful of data capacity.

Also, this phone holds onto its old AP much longer than the others. Anywhere the signal strength is below 30 dB, it should be looking to handover to APs with stronger signal strength, and these APs exist, as shown by the probe response points, but the Nexus One does not initiate handovers until very late, in several cases when signal strength has fallen to 10 dB SNR. The retry rate is high for a large part of the graph, reflecting this attempt to continue despite the low signal strength.

However, when a handover decision is made, the choice of target AP is good. In all cases here, it is over 40 dB SNR.

Most of the handovers are executed smartly. There are two PMKc handovers, at 55 and 83 msec for the protocol, and 141 msec for media-media. Also, five full PEAP handovers, ranging from 238 – 1385 msec. If one removes the last, the average is 277 msec for the re-authentication protocol and 323 msec media – media.

However, there are two episodes in the test that would be very noticeable to the user. The long handover (1.385 sec) at 41 sec is apparently because the phone paused in its authentication sequence. It sent null frames for around 600 msec before resuming the protocol. And at 152 sec, it attempted a handover but then abandoned it, sending a de-authentication frame to the AP and then going into a long sequence of ‘open’ probe requests before sending targeted probe requests at 158 sec, resulting in a good handover at 159 sec. This represents a media outage of 7.259 seconds, unacceptable for voice communications.

## Notes on Quality of Service and other features for voice

In this section we examine how well the smarphones deliver high-quality voice streams. Voice quality depends primarily on delay, jitter and packet loss. Long network delay from talker to listener eventually makes conversations difficult, as co-ordination suffers, but the WLAN contributes very little to this effect, perhaps 10 msec in a budget of 150 – 200 msec. Jitter can be damaging if the receiver’s jitter buffer is small: when the buffer under- or over-runs there is a gap in speech either because packets are lost, or while waiting for the next packet. And frames can be lost over-the-air, but Wi-Fi has a retry mechanism that attempts to re-transmit a frame several times before giving up, so irretrievably lost frames are unusual under normal conditions, and retries add to jitter but not more than a few milliseconds. Finally, congestion on the air can make it difficult for a phone to ‘seize the medium’ for a transmit opportunity: this can be a source of lost packets under heavy-load conditions, particularly without QoS priorities.

In this section we will discuss the significant standards and techniques necessary for good voice quality over Wi-Fi. We can’t deal with all of the nuances of good VoWi-Fi design in a short document, but luckily most of the benefits accrue from two key features:

* Compliance with the Wi-Fi Multimedia (WMM) standard; and
* Compliance with the WMM-power save (WMM-PS) standard.

And we can observe some of the facets of good voice traffic over the air:

* Uplink jitter performance; and
* Selection of transmission rates; and
* Resulting retry (error) rates over the air,

## WMM priority for Quality of Service

Much can be written on QoS for VoWi-Fi, but most of the benefits depend on implementing just one aspect of the standards. WMM is a Wi-Fi Alliance certification that gives voice frames priority over-the-air compared with other forms of traffic. If a client (or AP) has high-priority traffic queued to transmit, it has an opportunity to send it before other, lower priority traffic is allowed on the air. WMM is quite simple to implement, and very effective. It only really makes a difference when networks become congested with traffic, which is why many VoWi-Fi applications aimed at home use still do not implement it, but it should be de rigueur for enterprise users, as momentary congestion can appear quite frequently in enterprise WLANs.

Both Nokia and RIM apply WMM to their organic VoWi-Fi clients, and it appear sto work well. For both the iOS and Android operating systems, WMM is available but the application that must invoke WMM priority or the lowest priority level will be assigned. Most SIP client apps for these operating systems do not currently apply WMM, the Bria client for iPhone mentioned earlier being an honorable exception. But we expect these shortcomings to be rectified over time.

## WMM-PS for downlink traffic triggering and battery life

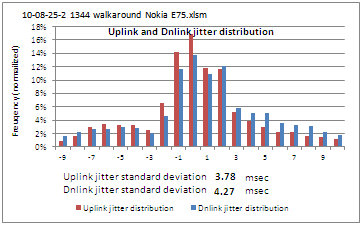
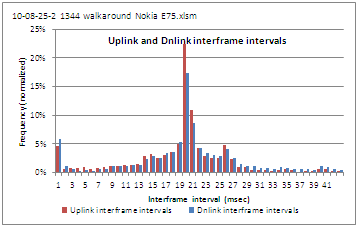
Battery life is not strictly a QoS feature for smartphones, but it is such a key determinant of user satisfaction that we include a reference here. The Wi-Fi standards of several years ago were quite power-hungry, but much progress has been made by the silicon designers in bringing down the power draw when Wi-Fi is turned on. The remaining significant gain comes from implementing WMM-PS, a Wi-Fi Alliance certification that allows the phone to ‘sleep’ between frames when on-call. To avoid losing downlink traffic while the client is sleeping, the AP must buffer frames for the client until it sees an uplink transmission, when it replies immediately with its downlink frame.

WMM-PS is useful because it allows the phone to switch off its Wi-Fi radio for inter-frame intervals when on-call, increasing talk-time in the order of 300% in phones Aruba has tested. But it also ensures that downlink and uplink frames are grouped together, so the phone knows it can scan off-channel, or perform other tasks between its transmissions without the risk of losing downlink traffic.

Having read this far, one might expect that the Nokia and RIM smartphones would have implemented WMM-PS, while HTC and Apple have not. This is indeed the current state of play, but hopefully not for much longer.

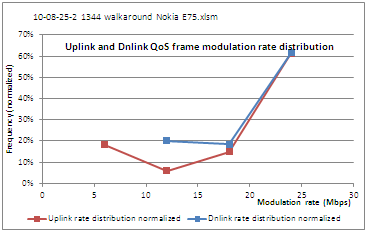
## Voice Quality analysis graphs

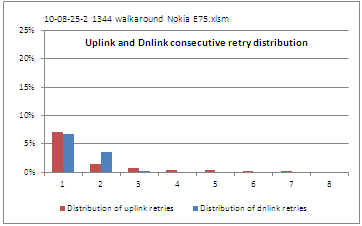
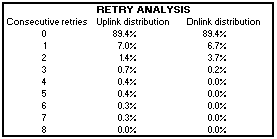
The first dimension we analyze is jitter. In all these graphs, the red histograms or lines are for the uplink, from the smartphone to the AP, while blue is for the downlink: it can be useful to compare the two. The test calls were made to the voicemail server, so we believe frames were generated with good timing integrity for the downlink, while uplink timing depends on the phone and over-the-air conditions.



The first graph (on the left) is an aggregation of inter-frame arrival times. Simply put, we subtract the timestamps of consecutive frames. For most codecs, including G.711 which is used in all these tests, we expect frames to be generated at regular 20 msec intervals, and the graph indeed peaks at 20 msec. Since we are including retry frames, there is a small peak at 1 msec for immediate retries, also some cases where two frames are buffered and sent consecutively, with WMM-PS. Sometimes we see a local peak at 40 msec, where the sniffer missed the odd frame. Where WMM-PS is used, we expect to see up- and downlink plots match, as the downlink is triggered by uplink frames.

The right-hand graph uses very similar data, the frame arrival timestamps, but divides them mod 20 msec to see how stable the internal clocks are. It is sorted for center-weighting, so we always see the peak at 0 msec, and the distribution reflects how jitter, clock wander and step-changes affected transmit and receive frame timing. We will see that some smartphones keep very accurate clocks, while others vary considerably, which could have consequences for voice quality by increasing overall jitter.

The second area we analyze is the over-the-air data rates used by both the smartphone and the AP. In Wi-Fi, each device sets the data rate (modulation rate) it uses for transmissions, usually based on how many acks it sees. If all frames are successfully ack’d, the rate is increased, if acks are missed, the rate is reduced: for a given SNR, a higher data rate will be associated with a higher error rate. As network engineers we like to see high data rates, as this maximizes overall WLAN data capacity, but for voice it is often better to reduce the rate to minimize retries – every retried frame increases air occupancy and jitter.

In the graph above, the uplink (red) frames are mostly at 24 Mbps, with some at 6, 12 and 18 Mbps. These are 802.11g rates. We would prefer to see 802.11n phones, as most of those tested here are, because they have slightly higher data rates (they are only single-spatial stream, so 58.5 Mbps is usually the top rate) and support a number of options introduced with 802.11n, but for voice services a good 802.11g smartphone is almost equivalent to a good 802.11n device.

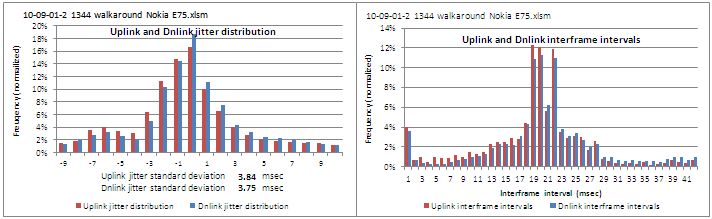
Lastly we analyze retried frames. Whenever a transmission is not ack’d (which can be because it was not received by the far end, or because the ack was lost), the frame is queued for immediate re-transmission, subject to air occupancy. A low rate of retries is always expected, as wireless is an inherently uncertain medium, but large numbers of retries increase jitter, and when a frame has been re-transmitted a number of times (usually from 4 – 8, depending on device settings), it is dropped, and the far end will never receive it.

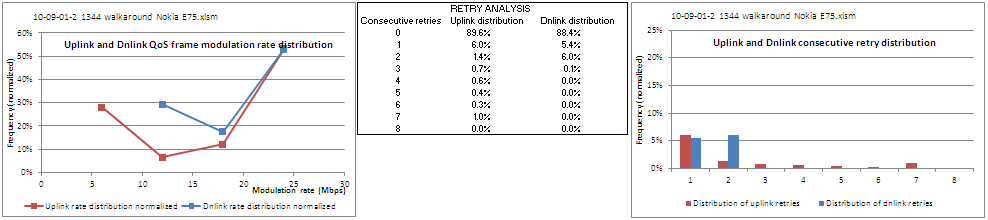
Our analysis shows how many frames have one, two, three… retries. For instance, in the graph above (the table gives the figures depicted in the histograms) 89% of frames are ack’d the first time, without re-transmission. Of the remaining 11%, most have just one retry, and the numbers fall off rapidly till only 0.3% of frames are re-transmitted 7 times which indicates in this case that they are dropped. The downlink shows a similar pattern, 89% of frames are ack’d immediately, but nearly all are corrected by one or two re-transmissions, and it appears none are lost. It takes further analysis to determine whether the smartphone or the AP is missing more frames or acks on the air, but this graph shows us whether the situation is normal, or we need to investigate.

In the following pages, we report these results for each of our four smartphones.

## Nokia E75 call quality measures

The graphs for Nokia jitter and interframe intervals show a strong tendency to 20 msec spacing, which indicates good timekeeping. The downlink pattern closely matches the uplink, which is consistent with the WMM-PS standard.

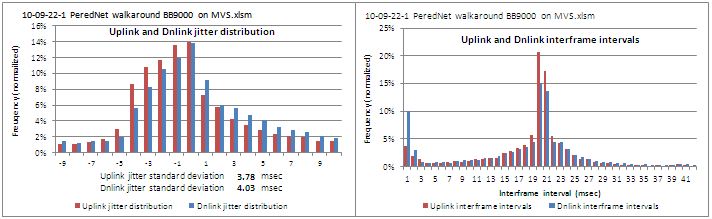


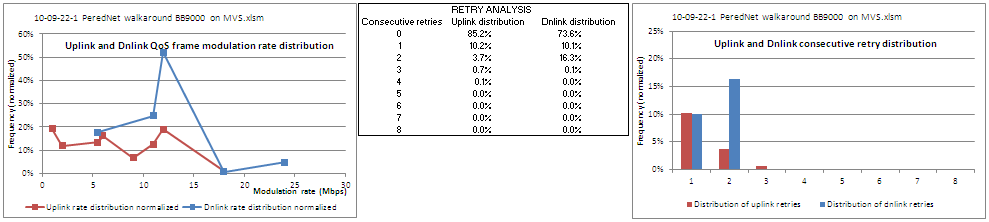
Meanwhile, most transmissions are at 24 Mbps, both on the uplink and downlink. This is consistent with the E75 as an 802.11g device (no 802.11n rates) but also shows a tendency to lower rates than allowed by the environment: 54 Mbps would normally be the maximum for 802.11g. Lower rates should imply lower retry rates but higher air occupancy.

The retry rates are generally low – 89% of frames are successful on the first attempt – and there are very low occurrences of multiple retries. It appears that about 1% of uplink frames fail after 7 retries, which is quite high, while the downlink has a higher incidence of frames with 2 retries (6%), but very apparently a zero frame-loss rate.

Overall these graphs indicate very good voice call quality. Despite our obsession with the WMM standard, its impact (or absence) is not seen in most of these tests, because the local WLAN traffic is not great enough to cause congestion. In the absence of bandwidth bottlenecks, WMM is not necessary.

## RIM BlackBerry 9000 call quality measures

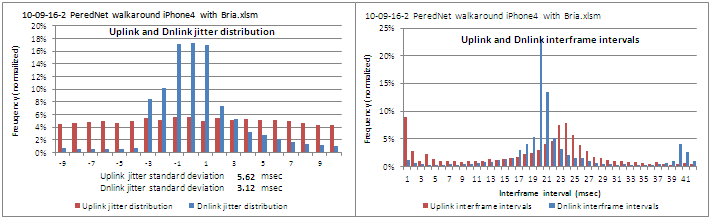
The BlackBerry smartphone also has a very strong tendency to 20 msec frame intervals, and it also uses WMM-PS, so downlink behavior mirrors the uplink. The standard deviation of jitter is similar to the Nokia E75.

Transmissions are also limited in data rate, with most uplink and downlink frames using the 12 Mbps rate, even lower than the E75. This is an 802.11g (a/g) client, and the highest rate attained is 24 Mbps for the downlink and 18 Mbps on the uplink, although very few frames use this rate.

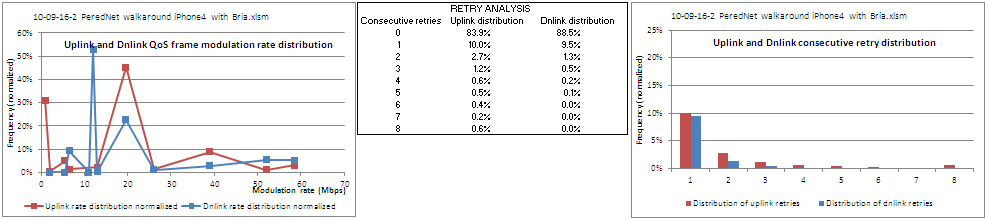
The retry pattern is also similar to the Nokia, with a slightly lower incidence of first-time success, but it appears that no frames in either direction are lost completely. More analysis is required to see whether the high occurrence of 2-retry downlink frames is due to a particular mechanism, but assuming these frames are ack’d, they should not contribute significantly to jitter.

## Apple iPhone4 call quality measures

The iPhone shows a much greater spread of uplink jitter than the two previous smartphones. While the blue histograms, for downlink frames, are broadly similar to Nokia and RIM – as one would expect, since the iPhone does not use WMM-PS – the uplink clocking seems very loose. This may be due to the lack of real-time handling of the software path on the smartphone, where the voice application is not able to get appropriate quality of service as it executes and sends packets down the stack to the Wi-Fi driver.



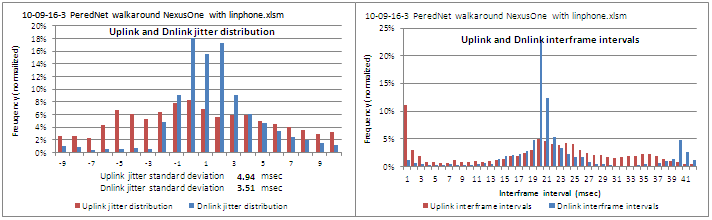
The modulation rates used by the iPhone extend to 58.5 Mbps, a maximum rate for 802.11n. But almost 50% of the uplink frames are at 19.5 Mbps. The downlink rates are very close to the uplink profile, indicating the algorithm used by the phone mimics that on the AP, generally a good sign.



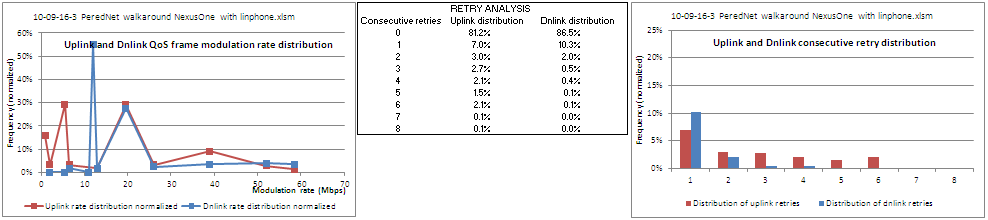
When we examine error rates, they are not very different from the earlier smartphones. First-time success is somewhat lower than Nokia, at 84% for the uplink, but the downlink at 89% is comparable. This may indicate that the downlink rate adaptation algorithm is a little more sophisticated, but it would take further analysis to determine why. About 10% of downlink frames are successful with just one retry, and after that there are very few stragglers. The uplink has higher retry rates at every step, with about 0.6% of frames seeing 8 retries and presumably timing out.

## HTC Nexus One call quality measures

The Nexus One’s behavior is similar to the iPhone4. Uplink histograms diverge from the downlink because WMM-PS is not used, and while the uplink jitter standard deviation is somewhat lower than for the iPhone, the general pattern is similar.



Modulation rates again extend to the maximum 802.11n rates for single-stream traffic, but most frames are at 20 Mbps or below. Uplink and downlink rates are broadly similar.



Error rates are the highest of any smartphone tested, with only 81% of uplink frames successful without retries. The retry rates stay high, with 2% of frames requiring 6 retries (this would account for more than 12% of over-the-air uplink traffic, an indication of the capacity impact of multiple retries).

## Summary of results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | Nokia E75 | RIM BlackBerry 9000 | Apple iPhone4 | HTC Nexus One | Notes |
| Operating system | Symbian S60 3rd edition, fp 2 | RIM OS Rls 5.0 | iOS 4.1 | Android 2.2 | Current release at time of testing |
| Protocol handover time (min, median, max) PMKc/OKC | 16-18-32 msec | 33-41-48 msec | 31-33-33 msec | 55-83-83 msec | From authentication to last key frame |
| Protocol handover time (min, median, max) PEAP | Always uses OKC after initial auth | ~ 245 msec(from other tests) | 422-435-454 msec | 217-315-1271 msec | From authentication to last key frame |
| Media handover time (min, median, max) | 27-56-1307 msec | 46-64-107 msec | 42-429-546 msec | 147-427-7259 msec | Between QoS data frames |
| Uplink jitter (standard deviation) | 3.8 msec | 3.8 msec | 5.6 msec | 4.9 msec | Target should be < 4 msec |
| Uplink first-time frame success rate | 90% | 85% | 84% | 81% | 90% first-time success is achievable |
| Voice client | Integral SIP client | MVS client | Counterpath Bria SIP client | Linphone SIP client | Few enterprise-focused voice apps in app stores |
| Over-the-air QoS mechanism | WMM | WMM | WMM (due to Bria) | None | Required standard |
| On-call power save | WMM-PS | WMM-PS | None | None | Required standard |

## Conclusion

As Wi-Fi-equipped mobile devices proliferate in the enterprise, whether corporate-supplied or employee-owned, there is an opportunity to connect these devices to real-time streaming media services such as voice and video. This paper analyzed how well four of today’s popular smartphones perform on Wi-Fi in an enterprise WLAN environment, with a focus on voice over Wi-Fi applications.

Using Aruba-developed analysis tools, we were able to demonstrate that these devices are already quite capable of voice services, meaning they can be used with carriers’ voice over Wi-Fi offerings, or configured to connect directly to the corporate PBX when in range of the WLAN.

The complicating factor for VoWi-Fi in an enterprise WLAN stems from the large number of densely-deployed access points used to provide service. The mobile device must be able to recognize these APs, to continuously calculate the best AP to connect to, and to execute accurate and speedy handovers when the user moves and a new AP becomes a better choice for connection. These problems can be solved by state-of-the-art software incorporating standards and suitable software algorithms targeted at the WLAN environment.

We showed that Nokia and RIM (BlackBerry) devices offer excellent performance for VoWi-Fi. In part, this stems from several years’ investment in development and testing by these vendors, in co-operation with Aruba and other WLAN providers. It is also somewhat easier for these Nokia and RIM to deliver seamless voice services, as they control the hardware, operating system and applications on their devices.

The other smartphones tested were not as capable. They usually perform well, sufficient to satisfy some users, perhaps most: but are they prone to occasional long gaps in the speech signal, and their jitter and retry characteristics are not as good. The business models used by Apple and Android are partly responsible. Both operating systems (iOS and Android) are capable of WMM, but the applications must indicate priority for their voice streams, and at present most do not. This can be solved by better co-ordination between developer groups: all smartphone services depend on the chain from Wi-Fi chip and driver through the operating system to the application, and while RIM and Nokia can ensure co-operation between these groups, it is not very difficult to forge agreement on a few basic features in the more loosely-coupled ecosystems.

A forthcoming standard, the Wi-Fi Alliance ‘Voice-Enterprise’ certification (expected in 2011) will include advanced features to make it much easier to develop high-performance mobile devices for the enterprise. Information exchange at the Wi-Fi layer will allow client devices to learn about the RF channels used by the WLAN, as well as neighbor AP information that is useful in scanning and choosing handover targets. Voice-Enterprise also includes a mechanism for the WLAN to prompt clients to hand over to a new AP and a fast re-authentication protocol. All these features should make inter-AP handovers more accurate, quicker and more reliable. But it will take several years before all WLANs and clients are Voice-Enterprise compliant, so smartphone designers should ensure their existing algorithms are improved, where necessary, to perform well in the absence of Voice-Enterprise.

Aruba continues to develop features for higher-quality, more effective voice and video over Wi-Fi services, and we are confident that future mobile devices will offer ever-higher performance. The explanations in this paper only scratch the surface of our investigation and understanding of VoWi-Fi mechanisms and features.

About Aruba Networks, Inc.

People move. Networks must follow. Aruba securely delivers networks to users, wherever they work or roam. Our mobility solutions enable the Follow-Me Enterprise that moves in lock-step with users:

* Adaptive 802.11a/b/g/n Wi-Fi networks optimize themselves to ensure that users are always within reach of mission-critical information;
* Identity-based security assigns access policies to users, enforcing those policies whenever and wherever a network is accessed;
* Remote networking solutions ensure uninterrupted access to applications as users move;
* Multi-vendor network management provides a single point of control while managing both legacy and new wireless networks from Aruba and its competitors.

The cost, convenience, and security benefits of our secure mobility solutions are fundamentally changing how and where we work. Listed on the NASDAQ and Russell 2000® Index, Aruba is based in Sunnyvale, California, and has operations throughout the Americas, Europe, Middle East, and Asia Pacific regions. To learn more, visit Aruba at <http://www.arubanetworks.com>.

H

© 2009 Aruba Networks, Inc.  *AirWave*®, *Aruba Networks*®, *Aruba Mobility Management System*®,*Bluescanner*, *For Wireless That Works*®, *Mobile Edge Architecture*, *People Move. Networks Must Follow*., *RFProtect*, *Green Island, The All-Wireless Workplace is Now Open for Business*, and *The Mobile Edge Company*®are trademarks of Aruba Networks, Inc.  All rights reserved.  All other trademarks are the property of their respective owners.