Optimising Simultaneous Interface Usage In A Converged Multimode Terminal

Alessandra Pandolfi†, Abigail Surtees*, Robert Finking* and Stephen McCann*
† Siemens AG, Haidenauplatz 1, 81617 Munich, Germany
*Roke Manor Research Ltd, Old Salisbury Lane, Romsey, SO51 0ZN, United Kingdom

***Abstract*— Mobile devices are moving towards the use of multiple heterogeneous radio interfaces (multimode functionality). At present, multimode capabilities are limited in that they tend to use just one radio interface at a time due to an inability to select and handover individual user applications between different interfaces as they become available. By taking the radical step of *automatically* selecting applications to transfer between active interfaces based on application requirements and interface capabilities, we immediately increase the network resources available to the mobile device, and also allow individual applications to run on the interface or interfaces that best suit them. This paper presents in detail a general theory for assessing an interface's conformance to an application's requirements. A proof of concept demonstrator is discussed briefly and the paper concludes that the theory provides a practical solution for optimising the simultaneous usage of multiple interfaces.**

***Index Terms*— Artificial intelligence, Communication equipment, Communication system software, Mobile communication, Multimode Terminal, QoS, Wireless LAN**

I. Introduction

Over the past few years, the range of wireless network services and technologies has increased rapidly. Conventional single interface mobile terminals are now evolving into multimode terminals, which include more than one interface technology such as Cellular, WLAN and Bluetooth[1] [1,2].

Converged devices available within the public market are relatively primitive, in that although separate radio interfaces are present there is minimal interaction between these interfaces in terms of support for active applications using different interfaces [3]. Largely such devices are based on the legacy "handover" model, where the terminal uses just one interface at a time. The prototypes that *have* appeared which allow multiple interfaces to be used simultaneously, have focused on the radio aspects of the simultaneous usage and rely on the user to manually switch applications between interfaces [4]. However, manual switching does not make the best use of the available resources and is inconvenient for the user. Given a choice of different wireless technologies via which to communicate, the user needs the terminal to adapt automatically and dynamically to the performance and availability of the interfaces, whilst still being able to influence the automatic behaviour if desired. This has been referred to as smart terminal behaviour.

We have developed an architecture that allows a terminal to have this smart behaviour, for example, when a new interface becomes available, automatic assessment of which, if any, running applications may benefit from using the new interface. The next section of this paper gives a brief overview of a key function of the architecture which enables smart behaviour. Next, the artificial intelligence behind the core of this function (how to assess an interface's suitability for use by an application) is discussed in detail; this is the main focus of the paper. Following on we discuss some QoS issues which present an obstacle to employing the presented theory in a real terminal. We then briefly discuss a proof of concept demonstrator which was developed to evaluate the theory in a live wireless environment.

[1] The *Bluetooth* trademarks are owned by Bluetooth SIG, Inc.

II. Architecture

Over the past few years we have developed a reference architecture for use within mobile terminals. This architecture supports a number of functions beyond those needed in a conventional unimode terminal. One of the key functions required to support smart behaviour in a converged terminal is Session Transfer Management (STM).

STM monitors and assess the suitability of the interfaces available for a given application according to the application's QoS requirements and also the user's preferences (such as cost). The interface through which a user can provide their preferences is implementation specific, but should enable the user to change preferences easily whilst maintaining the sophistication of the algorithm. When STM detects that the application may not be on the most suitable interface, it can trigger the application to change interfaces. There are several possible triggers for this function, including a new interface becoming available that may better suit some of the active applications, an interface's QoS changing, or an interface becoming inactive.

Within STM we have defined an algorithm that compares how well each of the available interfaces suits the QoS (and other) requirements of applications. This is called the conformance calculation.

III. Conformance Calculation

The conformance calculation identifies the most suitable interface for a given application.

In order to calculate how well an interface conforms to the requirements of each application it must assess how well the interface conforms to each of the separate requirements that an application has. It must then combine each of those assessments of conformance to individual requirements, into a single assessment of conformance.

An interface's conformance is scored on a scale of 0.0 to

1.0. A score of 0.0 indicates that the interface does not conform to the applications requirements at all – the application cannot run on this interface. A score of 1.0 indicates that the interface is fully conformant to the application's requirements. To keep the combining function simple all the assessments of the conformance of the interface to the individual requirements are also scored from 0.0 to 1.0. We begin by looking at the end product and work backwards from there.

A. The Interface's Conformance

The interface's conformance is determined by combining together the assessments of the conformance of the interface to each requirement.

The initial consideration is whether to combine each of the individual scores into a set or to reduce them to a single overall score. Handling of sets is more processor and memory intensive, which is an issue in the low resource environment found in a typical multimode terminal (MT). More importantly however, using a set would require the calculation of the Pareto Set (a set of sets), which whilst giving a "correct" result may not be particularly useful; since the number of interfaces available is small, it is highly likely that the Pareto set will contain multiple "optimal" choices and may even contain all available interfaces. This would not aid the decision making process. Therefore we prefer a function which produces a single conformance value for the interface.

We will consider three possible combining functions, assessing whether they have the following two properties:

- to yield 0.0 if any essential requirement is not met, and
- to yield 1.0 if all the requirements are fully satisfied.

A technique often used for combining multiple parameters into a single figure is a weighted mean. However this does not have the desired properties outlined above. If an essential requirement is not met, the overall conformance would not necessarily be 0.0.

A very simple function, which does have the desired properties, is minimum. It is also an intuitive function to use in that an interface is no better than its least conformant aspect. However minimum suffers from only taking into account the worst attribute of the interface. Hence an interface scoring 0.6 on all requirements can not be distinguished from one which scores 0.6 on just one, but 1.0 on all the rest.

Multiplication has the desired properties, is simple and unlike minimum gives significance to all requirements, not just the least conformant one, which allows it to distinguish between cases which minimum can not. So, the assessments of the conformance of the interface to the individual requirements are multiplied together to give a conformance value (C) between 0.0 and 1.0:

$$C = \prod_{i=0}^{i=n} P_i \qquad (1)$$

where n is the number of requirements/preferences expressed by the application/user and P_i is the conformance of the interface to the ith requirement/preference.

B. Types of Requirement

In general applications have two kinds of requirement or preference, those which are a choice from a number of options (e.g. bearer type) and those which relate to some value (e.g. cost, bandwidth).

Requirements which relate to discrete choice will allow one option to be picked from a list (e.g. "GPRS", "UMTS", "WLAN", "default"). Each option has a score associated with it.

For some discrete requirements a "none" option will be needed. For example, the MT may not always have network connectivity, so it makes sense to give a score for interface "none" for each application. For most applications, this score would be zero. However, not all applications need network connectivity in order to be useful (e.g. e-mails can be written/read offline), hence they may have a non-zero score.

The conformance score for discrete requirements is therefore obtained from a simple lookup table.

In its simplest form, a requirement related to a value could be described by an acceptable lower limit, below which the conformance is 0.0 and above which it is 1.0. However, this makes poor use of the information available, effectively reducing the value to a single bit. An improvement is to specify an upper limit also, beyond which there is no appreciable increase in benefit. There is then a continuum of conformances between the two limits.

Note that higher values are not always better (e.g. the more bandwidth the better, the lower the cost the better), but for the ease of description within this document we will always treat bigger as better.

The lower limit is always a fixed value (e.g. "minimum bandwidth"). The upper limit either may be fixed (e.g. "maximum useful bandwidth"), or may be relative to the best value currently available on any interface (e.g. "best available bandwidth").

Because of the continuous nature of value requirements and the fact that there are two limits involved, calculating the conformance to the requirement is non-trivial.

C. Assessing Conformance to Numeric Requirements

The calculation used for numeric preferences is more complex than for discrete preferences, so we will build up the components of the formula in parts in order that the mathematics behind the formula may be understood.

The basis for the numeric preference calculation is a simple linear function. The function reflects the current value of the parameter in question against two values, the lowest acceptable value (the lower limit, l) and the largest useful value (the upper limit, u).

A check is made if the value of the interface exceeds the upper limit, in which case the conformance is 1.0, or if the value of the interface is worse than the lower limit, then the score is 0.0.

If the value falls between the limits then a simple linear calculation is done. That is, if l is the lower acceptable limit, u is the upper target value and n is the value supplied by the

interface, the conformance of the network to this preference is:

$$P_i = \frac{(n_i - l_i)}{(u_i - l_i)} \quad (2)$$

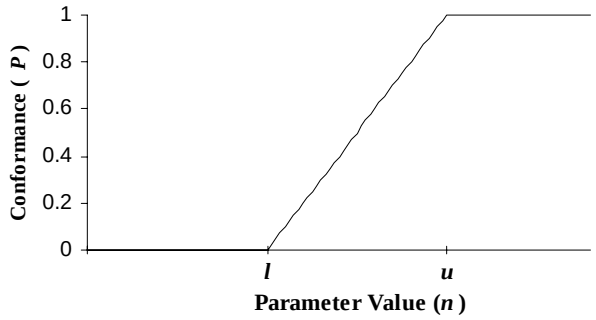

Fig. 1. Graph of requirement conformance function (2)

With (2), any given requirement can ultimately reduce the conformance of an interface to zero, since all individual requirements are combined into a product (see (1) above). It may be that a particular requirement is desirable, but not absolutely necessary e.g. low delay for file transfers. In that case, it should only be given an importance of say 25%, meaning that even in the worst case it would still have a conformance of 0.75 (75%). To reflect this in the formula it is necessary to add an importance weighting. The formula becomes:

$$P_i = 1 - w_i \left(\frac{(u_i - n_i)}{(u_i - l_i)} \right) \quad (3)$$

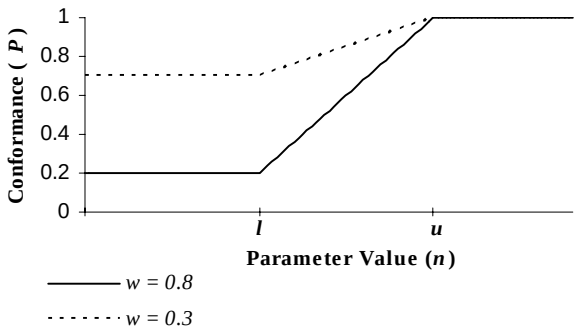

Fig. 2. Example graphs of requirement conformance function (3). If $w = 1$ the graph would be identical to Fig. 1.

Where w is the weighting of that particular requirement (0.25 in the above example). An importance weighting of 0 means "not at all important", it will not affect the conformance calculation. An importance weighting of 1 means "essential"; without meeting the acceptable limit for this preference, the interface isn't useable. With a weighting of 1, the formula is identical to (2).

This simple linear calculation is a good starting point, however in practice applications vary in how well they perform relative to the upper and lower limits. Consider bandwidth for example. Some applications don't perform very well until they get close to their target bandwidth, although they can manage with less. Other applications perform close to optimum as soon as their minimum acceptable bandwidth is available and only improve in performance slightly as they approach their target bandwidth. A simple way of reflecting this emphasis on one limit or the other (l or u), is to raise the linear calculation to a power, γ:

$$P_i = 1 - w_i \left(\frac{(u_i - n_i)}{(u_i - l_i)} \right)^{\gamma_i} \quad (4)$$

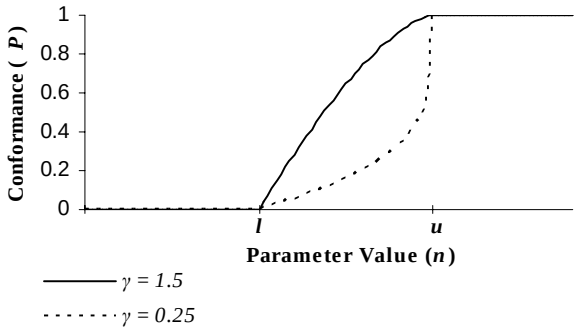

Fig. 3. Example graphs of requirement conformance function (4); $w = 1$. If $\gamma = 1$ the graph would be identical to Fig. 1.

With a value of 1 for γ the formula is identical to (3) and so gives equal emphasis to both limits. Values of γ between 0 and 1 (exclusive) emphasize achieving the lower limit, and values greater than 1 emphasize reaching the target limit or failing that, getting as close to the target value as possible.

A further refinement addresses the issue of cut-off at the upper and lower limits. So far, we have forced the conformance to be either 1 or 0 outside those limits. For some applications the limits are "hard", e.g. real time applications like voice over IP are simply unusable below a minimum bandwidth limit. For other applications, the limits are a good guideline, but performance might be possible below the lower limit or some benefit may be had beyond the upper limit. For example web browsing falls into this category. Therefore a mechanism is needed which reflects how "hard" the limits are. A simple way to do this is to calculate a weighted mean of two functions: the "hard" function described so far, and a "soft" function. The weighting assigned to each, reflects how "hard" or "soft" the behaviour is required to be.

The "soft" function chosen for this application is the sigmoid [10] (sometimes called the "S-curve" or "standard logistic function"), since it is simple, has a similar basic shape to the limited linear functions we have discussed thus far, and has no hard limits. The sigmoid is illustrated in Fig. 4 below as a dotted line.

The sigmoid we use must have the same gradient at its turning point as the limited linear functions ((2) to (4)). A shallower gradient than this does not represent the limits well, a steeper gradient than this conflicts with the basic limited linear function, because in a sense, a steeper gradient is

"harder". Moreover, keeping to the same gradient opens up the possibility of specifying two separate hardnesses, one for the upper half of the function and one for the lower half, without introducing discontinuities. For example a streaming video player may have a hard lower limit for bandwidth below which it can't operate at all, but may have a flexible upper limit. The target bandwidth may be enough to guarantee no frame loss in high motion sequences, but the player may be able to increase the picture quality beyond the target level if it receives above target bandwidth, which will be of interest to the user.

The basic form of the sigmoid function is:

$$f(x) = \frac{1}{1+e^{-Hx}} \tag{5}$$

Where H is the steepness of the function. The sigmoid has a turning point at $x = 0$ (i.e. the point at which the function is linear). For the gradient of the sigmoid to match the gradient of the linear function at the sigmoid's turning point, we require the gradient at the turning point to be 1 (since this is the derivative of x, the basic linear function):

$$\frac{d}{dx}\left[\frac{1}{1+e^{-Hx}}\right] = \frac{d}{dx}[x] \quad \Rightarrow$$

$$-\left[\frac{1}{1+e^{-Hx}}\right]^2\left(-He^{-Hx}\right)=1$$

at the turning point where $x = 0$, we have:

$$-\left[\frac{1}{2}\right]^2(-H)=1 \quad \Rightarrow \quad H = 4 \tag{6}$$

Combining the "soft" function (6) with the "hard" function (4) in a weighted mean gives our final formula:

$$P_i = \begin{cases} 1-w_i\left(\dfrac{1-g_i}{1+e^{-4\left(\frac{u_i-n_i}{u_i-l_i}-0.5\right)}}\right)^{\gamma_i} & for \quad 0 \geq \dfrac{u_i-n_i}{u_i-l_i}, \\ 1-w_i\left(\dfrac{1-g_i}{1+e^{-4\left(\frac{u_i-n_i}{u_i-l_i}-0.5\right)}}+\dfrac{g_i(u_i-n_i)}{u_i-l_i}\right)^{\gamma_i} & for \quad 0.5 \geq \dfrac{u_i-n_i}{u_i-l_i} \geq 0, \\ 1-w_i\left(\dfrac{1-h_i}{1+e^{-4\left(\frac{u_i-n_i}{u_i-l_i}-0.5\right)}}+\dfrac{h_i(u_i-n_i)}{u_i-l_i}\right)^{\gamma_i} & for \quad 1 \geq \dfrac{u_i-n_i}{u_i-l_i} \geq 0.5, \\ 1-w_i\left(\dfrac{1-h_i}{1+e^{-4\left(\frac{u_i-n_i}{u_i-l_i}-0.5\right)}}+h_i\right)^{\gamma_i} & for \quad \dfrac{u_i-n_i}{u_i-l_i} \geq 1. \end{cases} \tag{7}$$

Where the hardnesses of the upper (g) and lower (h) limits, are between 0 and 1, with 0 indicating soft limits and 1 indicating hard limits. Note that the time and space complexity for evaluating this formula are both O(1).

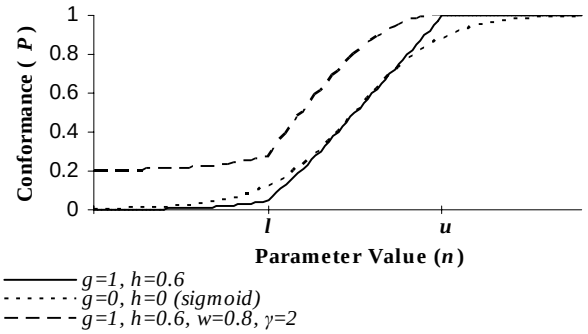

Fig. 4. Example graphs of requirement conformance function (7); $w = 1$ and $\gamma = 1$ unless otherwise stated. If all parameters are set to 1 the graph would be identical to Fig. 1.

IV. QoS Harmonization

When it comes to implementing any algorithm that deals with QoS on a real mobile terminal [5] there is an immediate difficulty: there is no one standard scheme used to describe QoS. The conformance calculation is no exception. Users and applications do not use the same scheme to describe their required QoS, and the interface QoS capabilities are described in terms that are technology specific.

The QoS parameters used by different QoS schemes vary widely. One scheme may specify just a single parameter related to bandwidth (e.g. mean bandwidth) another may specify eight or ten whole groups of parameters. Even those concepts which are common between two schemes may not be specified in the same way. For example mean available bandwidth may be specified by one scheme in bits per second, and by another scheme in terms of packets per second, with the mean packet size also being specified (in bytes). Without some harmonization between the schemes the conformance calculation will be dealing with different sets of parameters for different interfaces, making them impossible to compare.

The issue of QoS harmonization has been addressed previously, and is known to be a difficult problem to solve. Work has been done in this area by research projects such as BRAIN [6], and also within the IETF [7,8].

There are three main approaches to solving the QoS harmonization problem:

1. Instead of using QoS parameters directly, Use a high level categorization leading to a number of classes (like those specified in UMTS - Conversational, Streaming, Interactive and Background), which are defined by a variety of QoS parameters (e.g. bandwidth, latency, jitter) such that all QoS requirements fit into one category or another. Some parameters will have to be considered more important than others in order to choose the category in all cases. This approach is simple but inflexible and wastes information.
2. Employ an internal QoS model that is an

approximated subset of existing and proposed QoS facilities. This allows the Multimode Terminal (MT) to perform a direct comparison between networks by describing them in a common way.

3. Allow each application and network to describe its QoS parameters exactly using its native QoS scheme. This heterogeneous scheme requires STM to manage the differences in the way the applications specify their QoS requirements and the ways the different available networks describe them, but does not discard information unnecessarily. However it is highly complex and impacts the entire MT architecture.

Because of the limitations of QoS classes the first approach is rejected. Because of the difficulties inherent in the heterogeneous solution and because even for that solution at least a minimal internal QoS model is required, it is recommended that the second approach be taken.

This leaves the issue of parameters that an application specifies required values for, that are not described by a particular interface. It is necessary for there to be a mechanism for estimating unspecified values. There are a number of approaches which may be taken:

- Assume the worst case,
- Use user specified defaults for the application,
- Use configured defaults for the interface,
- Use calculated default values, based on current values of parameters that are available, or
- Use measured statistics.

The simplest and perhaps safest approach to take is to assume the worst about the missing parameter. However, this may stop a perfectly good interface being used simply because that interface does not advertise a particular QoS parameter.

The second approach is a slightly better way of handling unspecified parameters. Setting a default value gives the user control. However, specifying a default for an application means that the same default is applied to every interface; the sensible default value may vary widely between interfaces.

An improvement is to have a default setting for each interface. A combination of interface default values and user configured application default values gives a fairly good, but easy to implement way of dealing with missing parameters.

Taking a step beyond static default values, for some parameters it is possible to make an intelligent guess at their value, based on the values of the parameters that are available.

Finally perhaps the most accurate approach is to measure the actual performance of the interface. However this is a large and complex topic and is beyond the scope of this paper.

V. Proof Of Concept

The conformance calculation was tested initially by building a mathematical model in a spreadsheet. This gave the confidence needed that the algorithm was capable of making "smart" decisions on behalf of the user, and conversely that the user could influence the algorithm's behaviour in an intuitive fashion by modifying the parameters u_i, l_i, w_i, γ_i, g_i and h_i.

A more thorough test was conducted by building a proof of concept demonstrator system, which was deployed and tested on both an ordinary laptop and also an iPaq running Familiar Linux [11]. The processing requirements of the conformance calculation were easily satisfied by the iPaq. The test platform used two air interfaces, WLAN and GPRS, and ran two applications, a web browser and a streaming video client. Application requirements were entered for cost and bearer type. As part of the test harness, there was also a monitoring application which displayed graphically which application was using which interface. User preferences are modified through a configuration file, but a Gui is under development.

It was demonstrated that as the cost of each bearer varied, the applications were moved between the two interfaces in an intuitive and intelligent fashion. It was also demonstrated that an application was automatically transferred from one bearer to another if the bearer it was on became unavailable.

We conclude that the theory provides a practical solution for optimising the simultaneous usage of multiple interfaces.

VI. Future Steps

Looking towards the future, it is hoped that the kind of algorithm presented above will be implemented and adopted within multimode terminals. Although the information presented here is by no means definitive, it should become a valuable aid for designers in designing smart software for future terminals.

References

[1] R. Janowski, "Software Radio Concept", http://3gsolutions.members.easyspace.com/soft_radio1.html , 2002

[2] S. McCann, W. Gröting, A. Pandolfi. E. Hepworth, "Next Generation Multimode Terminals, 3G2004 Conference, October 2004

[3] G. Leijonhufvud, "Multi access networks and Always Best Connected, ABC", Ericsson Research, http://jungla.dit.upm.es/~ist-mind/publications/abc-sld.pdf

[4] K Strohmenger et al, "Re-configurable Multi-mode Radio Architectures for enhanced 3G Terminals", EU IST-MuMoR, http://www.nokia.com/library/files/docs/Re-configurable_ Multi-mode_Radio_Architectuers_for_enhanced_3G_Terminals.pdf

[5] Dan Chalmers and Morris Sloman, "A Survey of Quality of Service in Mobile Computing Environments", IEEE Communications Surveys & Tutorials, Volume 2, Number 2, 1999, http://www.comsoc.org/livepubs/surveys/public/2q99issue/sloman.html

[6] Information Society Technologies, IST-1999-10050 BRAIN D2.2, BRAIN architecture specifications and models, BRAIN functionality and protocol specification, March 2001, http://jungla.dit.upm.es/~ist-brain/deliverables/BRAIN%20Del%202.2.pdf

[7] Jerry Ash, Attila Bader, Cornelia Kappler, "QoS-NSLP QSPEC Template", December 2004, http://ietf.org/internet-drafts/draft-ietf-nsis-qspec-02.txt

[8] J. Wroclawski, RFC 2210, "The Use of RSVP with IETF Integrated Services", September 1997, http://www.ietf.org/rfc/rfc2210.txt

[9] Kai Lehmann, "Exploiting QoS Characteristics of Heterogeneous Wireless Networks", August 2004, Department of Electronic Engineering, School of Electronics and Physical Sciences, University of Surrey, Guildford, Surrey, GU2 7XH, UK

[10] Eric W. Weisstein. "Sigmoid Function." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/SigmoidFunction.html

[11] The Familiar Project, http://familiar.handhelds.org/