7

QoS Support for Mobile Computing

Steven Pope and Paul Webster.
Olivetti and Oracle Research Laboratory.
24a Trumpington Street,
Cambridge,
England.
{spope,pwebster}@orl.co.uk

Abstract

In recent years small, completely portable computers have become available on the marketplace. There is demand for such computers, termed *walkstations*, to access network services while retaining their mobility and to operate effectively whilst traversing both indoor Wireless LAN (WLAN) networks and the outdoor Mobile Radio Network (MRN) infra-structure.

This paper introduces the *traded handoff* where connections are rebuilt during a handoff to the most appropriate service, taking into account the properties required by the application and locally available, replicated or compatible services. Network level solutions can only redirect connections back to the original endpoint as the walkstation moves. This is the case even if the connection is to a locally replicated service.

Keywords

Mobile, QoS, Trader, Handoff.

1 QUALITY OF SERVICE SUPPORT

The trading-space visible to an application potentially contains the offers from a number of federated traders corresponding to different network types, providers, or gateways. When trading for offers in this environment, a client may wish to place additional *constraints* on the offer based upon the Quality of Service (QoS) obtainable. For example, to minimise the cost of communication. Such constraints do not relate to properties of the service exported by the server, but instead to properties of the service which would be received by an application given the current and expected behavior of the walkstation. It would therefore be impossible for an offer to contain these properties when exported.

Building QoS into Distributed Systems A. Campbell & K. Nahrstedt (Eds.)

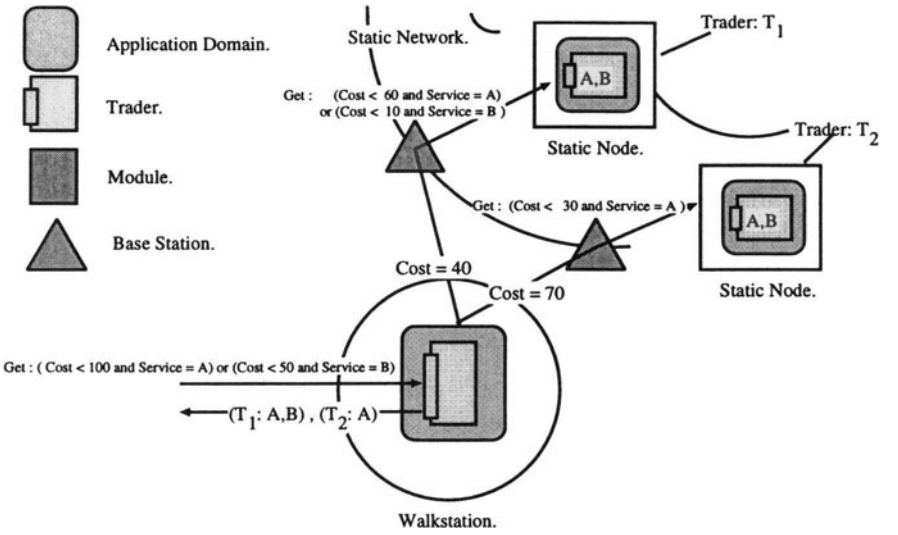

Figure 1 QoS Constraints.

Since it is not possible to determine the exact QoS available for a particular offer without direct negotiation with the exported server interface, a heuristic is used during trading: offers from federated traders are matched only if the QoS experienced between the federated traders matches the QoS constraints from the client. This may be illustrated by the trivial example shown in Figure 1. Here, the walkstation is federated with two traders, T_1 and T_2, over links with respective *Cost* of 40 and 70. If the walkstation's client queries its trader with the constraint:

$$(Cost < 100 \text{ and } Service = A) \text{ or } (Cost < 50 \text{ and } Service = B)$$

The walkstation's trader considers the QoS available to each of its federated traders T_1 and T_2 and sends a request to trader T_1 with constraint:

$$(Cost < 60 \text{ and } Service = A) \text{ or } (Cost < 10 \text{ and } Service = B)$$

and to trader T_2 with constraint:

$$(Cost < 30 \text{ and } Service = A)$$

The second **or** clause is dropped in the request to trader T_2, since it requires a link of $Cost < 50$ and the link to trader T_2 has a *Cost* of 70. These queries result in the list of returned offers: $((T_1 : A, B), (T_2 : A))$. The walkstation's

trader inserts the appropriate values for the *Cost* QoS property to each of these offer's property lists so that they are considered by the client.

The above example has illustrated a case where the QoS properties are cumulative over a route. The sum of the cost over each link must be no greater than the total specified in the constraint. Other QoS properties, such as *Bandwidth* are not. Non-cumulative properties are passed unmodified over each link.

If the same server instance is available through a number of different routes, the corresponding offers returned to the client must be distinguishable. In some circumstances, this may require the modification of the returned interface references by the walkstation's trader.

By propagating the QoS constraints in the queries to federated traders, each trader in a chain of federated traders is able to consider the constraints and the QoS available between itself and the next trader, and so perform further restrictions on the query.

What the trader cannot do is to negotiate precise QoS parameters for a particular service. These will vary over a particular network and over time at the same base station, dependent upon factors such as signal quality and the number of walkstations simultaneously using a particular base station. An application must negotiate its own QoS at the time a binding is established.

2 THE TRADED HANDOFF INTERFACE

A program module which is to explicitly use the traded handoff mechanism is expected to provide an implementation of the Handoff Interface (HDI), and its three constituent methods: *Open, Unmount* and *Mount.* The *Open* method requires the module to create and stash all its required bindings to servers. The *Unmount* method requires a module to close down its bindings and block its threads at one of a set of well defined synchronisation points. The *Mount* method requires the module to initiate a new session with the server using the stashed bindings, transferring appropriate client state. The module's threads (if blocked) may at this point continue. The example shown in Figure 2 illustrates how the traded handoff is performed for a client.

To utilise the full potential of the traded handoff, it is essential that the end-application is given the opportunity to transfer it's session level state. For example, a video server application might be initialised with the title of a video to be played together with a frame offset. Assuming that a more appropriate video server is visible in the trader after a handoff, the new video server should be initialised with the required title and offset as part of the *Mount* phase.

However, where the application wishes to use a default behavior or where the application is unaware of traded handoffs, the Remote Procedure Call (RPC) package provides an implementation of the HDI and directly responds to traded handoff requests. Applications which are aware of the traded handoff

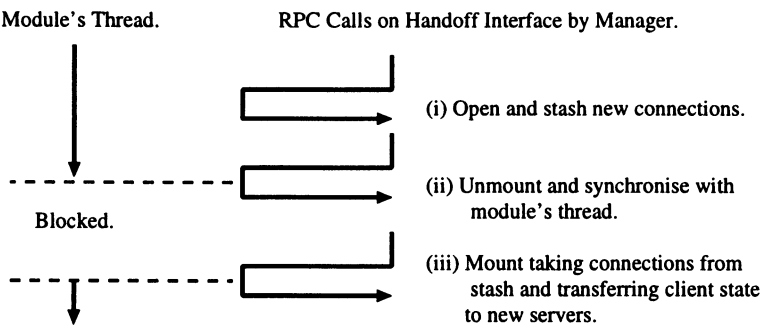

Figure 2 Blocking During a Traded Handoff.

may specify a default behavior of either re-establishing bindings to the original server interface or to another server which matches a given constraint. For applications which are unaware of mobility, the RPC package responds by re-establishing bindings only to the original interface. This enables the use of legacy code in an environment which relies on traded handoffs.

3 CONCLUSION

This paper has introduced the traded handoff, enabling widely replicated services and heterogeneous networks to be utilised as a walkstation is carried over large distances. The handoff protocol described here has been implemented and integrated with an object based RPC package [2] and trader which implements a subset of [1]. Experience in the local area has demonstrated the traded handoff for a number of applications including a network based video server. In this case, the disruption experienced by the end viewer (13ms) was much less that the play out time for a frame of video and indicate that use of the traded handoff in the wide area is feasible.

Future directions for this work would involve the use of a mobile infrastructure using walkstations over multiple wireless network types and in the wide area.

REFERENCES

[1] ISO/IEC. *ODP Reference Model: Trading Function*, October 1994. Committee Draft ISO/IEC/JTC1/SC21 N807.

[2] S. Pope. *Application Support for Mobile Computing.* PhD thesis, University of Cambridge Computer Laboratory, 1996. Technical Report 415.