

Protocol Support for Bulk Transfer Architecture

Uendra Rathnayake^{†*}, Henrik Petander*, Maximilian Ott* and Aruna Seneviratne^{†*}

[†] School of Elec. Eng. & Telecomm's, UNSW, Sydney, Australia

* NICTA, Sydney, Australia

Email: [first name.second name]@nicta.com.au

Abstract—Today's mobile devices are increasingly containing multiple radios, enabling users on the move to take advantage of the heterogeneous wireless network environment. In addition, we observe that many bandwidth intensive services such as podcasts, software updates etc are essentially non-real-time and buffers in mobile devices are effectively unlimited. We therefore proposed EMUNE, a new transfer service architecture in our previous work which leverages these aspects and supports opportunistic bulk transfers in high bandwidth networks.

EMUNE uses multiple wireless network interfaces according to an optimal transfer schedule derived based on device power concerns, application requirements, user preferences of cost and quality and future network availability. In this paper, we explore the usability of network/transport protocols to achieve the use of multiple network interfaces with required functionalities including flow mobility and striping in mobile environments and propose a MONAMI [1] + R²CP [3] hybrid approach for the architecture.

Index Terms—Mobility Protocols, Network Interface Selection, Data Transfer Scheduling.

I. INTRODUCTION

The modern wireless mobile devices are increasingly equipped with multiple radios enabling them to connect to different access networks based on different technologies. As users move they will encounter networks with different capabilities and available capacity. Prior work, such as [4] has demonstrated that a user can optimize the utility by performing vertical handovers between different available networks.

The work on vertical handovers to date has generally focused on providing seamless connectivity. However, there is a large class of bandwidth intensive non-real-time applications such as software updates, podcasts, email and to some extent even video on demand (with HTTP streaming) where continuous connectivity is not mandatory. Specially since modern mobile devices have plenty of memory to buffer large amount of data in the device enabling a shift from streaming to bulk transferring with constraints, the data transfers of this class of applications can be scheduled taking into consideration a number of factors such as power usage, user preferences like willingness to wait and cost. As some networks like Wi-Fi are not ubiquitous, the knowledge of future availability of networks (and their bandwidth) will further improve the effectiveness of such a scheduling mechanism.

Therefore, we proposed EMUNE; a novel and a unique architecture which takes into consideration above factors to derive an optimal schedule to utilize heterogeneous wireless access networks for application data transfers (interested

readers are referred to [5]). The schedule allows use of multiple network interfaces simultaneously for application data transfers, suspending and resuming data flows on particular interfaces etc. In implementing and realizing these functionalities on a mobile device, we observe that not all of the existing protocol mechanisms support them and therefore, we investigate in detail the applicability and usability of such mechanisms in this paper. As there is no single protocol which is capable of fulfilling all of the requirements alone, we propose a MONAMI [1] + R²CP [3] hybrid cross layer solution.

The rest of the paper is organized as follows. In the next section we briefly describe our architecture and also formulate the required protocol functionalities it needs. Then, we look at several options available in section III. Section IV discusses their pros and cons in detail followed by introducing a hybrid approach in section V. Section VI provides the conclusion.

II. ARCHITECTURE AND PROTOCOL REQUIREMENTS

In order to exploit the delay tolerance of non/near-real-time applications for optimizing the cost and performance for a mobile user, we developed EMUNE, the Architecture for Effective Mobile Usage of heterogeneous networks. EMUNE predicts the availability of access networks and their available bandwidth in near future, for a set duration ahead. Then it finds an optimum way to use the *future* available networks in that duration considering several factors and risks including the uncertainty of predictions. Then it transfers data of applications accordingly in both uplinks and downlinks so that the anticipated objective of maximizing the benefits to the user is accomplished. After passing that time duration, EMUNE does this for another new duration for which, a new availability prediction is derived. This process is repeated by the architecture until there are data available to be transferred.

As depicted in Figure 1, EMUNE consists of an API, two major functional units: the prediction engine and the scheduling engine, and a transport service. The API enables application programmers to easily use the functionalities provided by EMUNE [5]. The prediction engine predicts availability (and their available bandwidth) of each network type supported by the mobile device using the mobile device's associated context information [6]. The scheduling engine uses above predictions together with application requirements and user concerns of quality and cost to find the optimal schedule, which probabilistically allocates the data transfers

of all application requests over the *future* available networks so as to maximize the user utilities over energy and dollar costs for non-real-time bulk data [7]. The transport service, the component that we are investigating about in this paper, controls application data transfers accordingly so that they adhere to the above schedule.

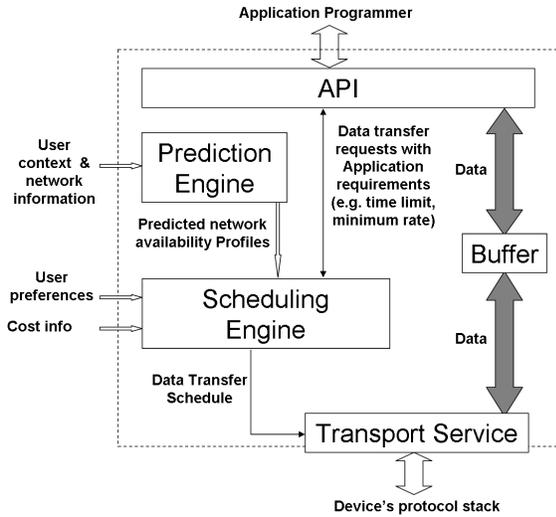


Fig. 1. EMUNE Architecture.

The transport service should possess the capabilities mentioned below for the architecture to attain the desired functionality.

A. Simultaneous Use of Multi-Interfaces

The utility of application flows vary, thus the scheduling policies for them will also be different. To implement the policies without conflicts, interfaces should be selected for each flow independently of the other flows¹. In other words, if several applications are transferring data, they should be able to use different interfaces independently.

B. Flow Mobility

When the network environment changes, the scheduler re-evaluates the policies and remaps the flows to the available interfaces. The transport service should be able to implement this remapping, and move each flow which has started on a particular interface to any other interface depending on the schedule, independently of other flows over the time.

C. Flow Striping

An application flow may need to use more than one interface simultaneously, which is commonly known as bandwidth aggregation [11]. For example, the scheduler may allocate a file download to use two network interfaces at the same time for maximizing the available bandwidth for the flow. Therefore the corresponding data flow should be split into several sub flows and sent over those interfaces and aggregated/reordered

¹The scheduler [7] ensures that each flow gets the right amount of bandwidth.

at the end point. The transport service thus should enable flow striping. In dynamic network conditions the portion of the flow scheduled over each interface should be adjustable to prevent reordering of packets due to a larger portion of packets being sent over a lower bandwidth interface.

D. Suspend/Resume flows

The scheduling engine may not allocate any time slot of an interface for a data flow in the immediate time frame and in those cases, the transport service should be able to suspend the flow and later to resume after getting a quota for an interface from the scheduling engine. That means the data flow session has to be persistent even when in suspended status as otherwise the data transfer would start from the beginning, jeopardizing the motive of the architecture. The suspension and resumption of a flow may be done due to loss of network coverage or policy decisions, e.g. not downloading large files over 2nd generation cellular networks. Therefore, the suspend/resume should be able to handle pausing of traffic for periods of time ranging from seconds to hours. Further, ideally the state should only be kept in the client, so that servers would not have "zombie" connections taking up limited resources.

E. Failure Recovery

Sometimes, the data transfer flows may abruptly be terminated due to reasons such as mobile device failures, flow terminations by the other end due to time expiration etc. Effective recovery from such instances is crucial, without the mobile device starting the data transfers from scratch which unnecessarily wastes network resources. Therefore, it is very important for the transport service to be able to transfer specific data blocks selectively.

III. ANALYSIS OF EXISTING PROTOCOLS

By looking through the available options in the literature, we found few solutions that cater for some of the above mentioned needs and are described below.

A. MONAMI and Freeze TCP

MONAMI [1] is an extension to Mobile IP protocol, which allows mapping of flows to multiple interfaces by giving each flow its own identifier and mapping these identifiers to the IP addresses of the interfaces. MONAMI signaling creates tunnels between the mobile device and its *Home Agent*, a router in the fixed network which keeps track of the location of the mobile device and delivers packets to it. The MONAMI signaling extensions enable the Home Agent to deliver packets to the right interface on the mobile device.

Freeze TCP [2] refers to the mechanism of the receiver "freezing" a TCP connection by sending a zero window advertisement to the sender. This message indicates that the receiver cannot handle more data and it makes the sender pause the transmission. To prevent "zombie" connections from clogging up the sender, it sends keep alive messages to the receiver to make sure that the receiver socket is still alive.

The combination of Freeze TCP and MONAMI could be used to implement the scheduling with some limitations.

MONAMI enables simultaneous use of multiple interfaces and flow mobility. However, being a pure network layer solution, it does not support flow striping or flow suspension. MONAMI can be complemented with Freeze TCP which allows suspension and resumption of flows, but requires keep alive messages between the mobile device and its peer. This becomes problematic, when the connection is paused due to loss of connectivity, since the mobile device would not be able to respond to the keep alive messages. Further, the server needs to keep the TCP connection open for the duration of the suspension. For large servers serving numerous mobile devices this may become a problem. An additional mechanism or extensions to MONAMI would be required for striping of flows across multiple interfaces.

B. Radial RCP (R^2CP)

This introduced in [3] is a multi-state extension of the Reception Control Protocol (RCP) [3] which delegates all the responsibilities of flow control, congestion control, reliability etc to the receiver as opposed to the sender in TCP. The key idea behind RCP is that in wireless environments it is the receiver which has the first hand information of access network channel conditions and therefore it should take care of all the control of the data transfer sessions. RCP achieves this by delegating the responsibility to the receiver to determine how much data the sender can send (via congestion control and flow control) and which data the sender should send (via reliability). R^2CP controls several RCP flows and enables seamless handoffs, server migration and bandwidth aggregation.

The unique feature of RCP (and hence R^2CP) is that they have internal sequence numbers which allow them to ask to send some specific data blocks with the *pull* mode. Further, R^2CP can freeze RCP flows and later resume them with *FREEZE* and *RESUME* calls. Bandwidth aggregation is achieved by R^2CP opening several RCP connections over different interfaces to the same sender and then intelligently controlling them to use those interfaces simultaneously. Moreover, even though R^2CP does the flow control, it can instruct each RCP pipe to use different congestion control mechanisms if R^2CP is provided such instructions, making RCP connections more adaptive to the concerned access networks. Use of R^2CP enables the EMUNE architecture to achieve its objectives effectively in the download path.

However, when it comes to upstream traffic, it's the receiver (the remote server) who controls the RCP sessions and therefore it becomes challenging for EMUNE to achieve its functionalities with uploads. The authors in [3] suggest to use TCP in such situations but still, the functionalities like flow striping would be unavailable without special provisions. Further, R^2CP and RCP being new protocols poses an implementation challenge as changing the stacks in the legacy servers is not as easy as in the mobile devices.

C. OCMP

This session layer protocol introduced in [8] allows applications to use multiple network interfaces simultaneously

according to the policies provided to it. The design of the protocol is such that it does not require changes at the legacy servers' side. That means, a proxy placed close to the mobile device acts as an intermediate point to communicate with and serves on behalf of the mobile device using conventional protocols like TCP. But the proxy to mobile device path possibly over several interfaces is communicated with OCMP session protocol while any transport protocol (e.g. TCP, UDP with added reliability) acting as an underline communication method. The proxy identifies a mobile device uniquely by its Globally Unique Identifier (GUID) and that can be drawn from an existing namespace such as International Mobile Subscriber Identities (IMSI) of mobile phones or IPV6 addresses. It uses several other identifiers also to distinguish different applications, sessions etc.

When transferring data, the mobile device's OCMP client informs the proxy's *application plugin* about the data request and it then downloads and stores it in its persistent storage. Then the proxy transfers the data to the mobile over opportunistic links, of course according to the schedule given by the mobile device. The simultaneous use of network interfaces can be achieved as the OCMP plugin on the server side can forward data of different applications through different interfaces. Not only that, it can stop using one interface and move a flow to another interface upon instructions. Further, a single flow can be stripped and sent over different interfaces with proper coordination, for the delay tolerant applications where re-sequencing buffers can be assumed of having infinite size on the persistent storage. OCMP persists the application sessions across disconnections due to device switch offs, connectivity loses etc. However it may not be possible to recover in instances such as device failures as the sessions cannot be stored when the device is failing.

The deployment of a proxy which introduces some extra effort has to be placed close to the mobile device, which becomes a challenging task when different networks of different ownerships are concerned. Further, the flow striping is not possible for real-time/near-real-time applications as authors have not provisioned for the head-of-line blocking in re-sequencing buffers which is available in R^2CP . This requires these applications to be separated and communicated using conventional protocols and hence the execution of the fine-grained data transfer schedule may become problematic. Moreover, OCMP can induce a substantial protocol overhead to the resource constrained mobile device as it introduces a complete session layer.

D. mSCTP

SCTP [9] is a multi-streamed, multi-homed reliable transport protocol originally developed for carrying signaling traffic in telecommunication networks. Initially, the use of multiple IP addresses was meant for reliability purposes as non primary paths were only used for redundancy. But the new amendments have enabled dynamic addition/deletion of address to ongoing SCTP sessions [10] making it mobility capable so that applications can use multiple interfaces simultaneously. [11]

proposed an enhanced version of SCTP which can do striping over multi-interfaces. Further, freezing and resuming is also proposed for SCTP which is named as "Standstill SCTP" in [12].

However, it is not clear whether the above functionalities can be controlled by a separate module (scheduling unit) externally in upward and downward data transfer directions and the research on SCTP is still ongoing. Another hurdle is the deployability concern as this protocol has to be introduced in both ends of a communication. Moreover, failure recovery capabilities (ability to transfer data blocks selectively) of mSCTP are unclear.

E. Other Options

There are other candidates like pTCP[13], HIP extensions [14] etc. which may support some of the required protocol functionalities. However, they do not address the entire set of requirements mentioned in section II (especially, transferring only specific data blocks in recovering from failures) and also controlling them according to the policies given by the scheduling engine seems unclear.

IV. DISCUSSION

An ideal solution would be able to suspend flows so the transport was frozen or the state of a session saved and a new transport flow could continue from where the old flow ended. This could be achieved most optimally as a part of the transport layer or with some extra overhead from restarting flows on top of it. Striping, on the other hand would benefit from being on top of transport layer allowing use of independent flow control over different links. From this point of view an ideal solution would split a flow into multiple transport streams, so that the congestion on each path could be monitored separately. Thus, a pure network layer approach such as MONAMI would not be sufficient and could not be easily extended either, for implementing the needed functionalities. This leaves two alternatives, a transport or session layer approach and alternatively a cross layer solution combining transport or session layer mechanisms for striping and flow suspension with lower layer mechanisms for flow mobility.

The pure transport/session layer approaches have their own limitations. For example, the OCOMP protocol introduces a complete session layer which may consume substantial amount of resources of a resource limited mobile device. Further, it also has not provisioned for real-time applications. R²CP, a pure transport layer approach, can not alone handle all of the problems caused by device mobility, namely it does not provide reachability for the mobile device to deal with new flows for which the mobile device is not the originator. Therefore, a suitable locating mechanism such as Dynamic DNS would have to be deployed unless a cross layer solution was not used.

From the available options, it is clear that a hybrid network-transport or network-session solution would be the best for

dealing with the issues discussed above. As there is no apparent failure recovery mechanism in mSCTP compared to R²CP where downloading only specific data blocks is possible with *pull* mode, and the OCOMP protocol is incapable of handling head of line blocking in real-time/near-real-time applications' data flows, we consider the combination of MONAMI and R²CP as the best mechanism for tackling the requirements of our architecture.

V. PROPOSED SOLUTION

To deal with legacy servers and peers which do not support R²CP, we borrow the proxy concept from OCOMP. In MONAMI, all flows go through the Home Agent (HA) which makes it an ideal selection for an application layer proxy server allowing it to translate TCP or UDP flows from legacy peers to R²CP flows. The Mobile Node (MN) can then communicate with the HA using R²CP utilizing its capabilities to execute the data transfer schedule in the download path, as depicted in Figure 2.

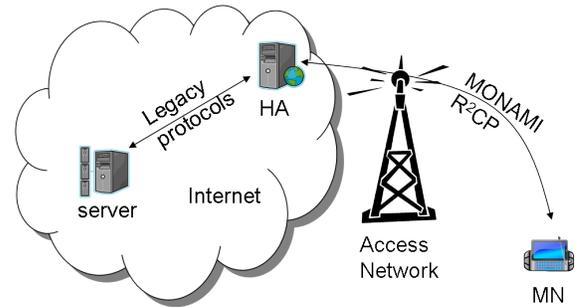


Fig. 2. MONAMI and R²CP Hybrid Approach.

Integration of R²CP and MONAMI requires cross layer information on distribution of the R²CP flows since with MONAMI all flows are bound to the same IP address, the home address of the MN. MONAMI distributes transport flows by mapping them to care-of addresses located on different network interfaces. To this end, MONAMI uses flow rules which describe the flow, so the packets belonging to it can be recognized both in MN and HA and routed via the correct care-of address. The routing is done by connecting the flows to *flow ids* using flow rules, so that all flows under the same id are treated in the same way. The flow ids can then be bound to a specific care-of address using a MONAMI *Binding Update* message which will cause incoming traffic from HA to MN to be routed to that care-of address. For outgoing traffic, MN can use for example policy based routing or firewall rules to send packets belonging to a flow over the correct interface.

The R²CP flow distribution would need to be integrated with the MONAMI flow rule management for proper use of both protocols together. The integration could be achieved by using a flow rule description language, such as the one proposed by Larsson et. al [15] over a TCP control connection. In the flow graph shown in Figure 3, we illustrate the interaction between MONAMI and R²CP. Upon starting a new transport, the scheduler would divide it into R²CP transport flows. These

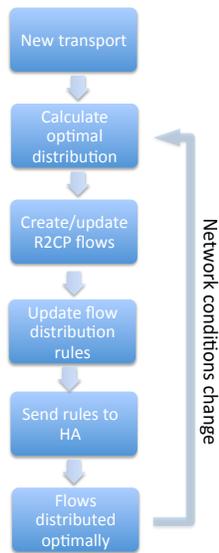


Fig. 3. Interaction of MONAMI and R²CP.

flows would then be distributed over the available interfaces by adjusting the existing MONAMI flow rules, mapping the new flows to the care-of addresses and communicating the new flow rules and bindings to the HA. After this, the flows would be optimally adjusted, until an existing transport session ended, a new one started or the network conditions changed.

This proposed cross layer solution would have the additional benefit of MN being able to distribute (stripe) both uplink and downlink traffic across all available interfaces, whereas in R²CP, only downlink traffic could be distributed. Therefore, the use of MONAMI and R²CP together with proper cross layer design would allow EMUNE to achieve its desired data transfer scheduling functionality both in uplink and downlink directions.

VI. CONCLUSION

In our earlier work, we have proposed an architecture, EMUNE, which optimizes multiple network interface usage of a mobile device in a heterogeneous networking environment considering several factors such as user preferences, network costs and application requirements. In this paper, we explored the usability of network/transport protocols for our architecture and proposed to use MONAMI and R²CP together with cross layer information sharing so that the desired functionalities can properly be achieved both in the downlink and uplink paths of a mobile device.

ACKNOWLEDGMENT

NICTA, a research organization, is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. Further, this work has been performed in the context of NICTA's CAMP project, which is funded by Ericsson.

REFERENCES

- [1] "Analysis of Multihoming in Mobile IPv6", <http://www.nautilus6.org/ietf/monami6/ietf63/monami6-ietf63-bof.html>
- [2] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-tcp: A true end-to-end tcp enhancement mechanism for mobile environments", in IEEE INFOCOM, 2000.
- [3] H.Y. Hsieh, K.H. Kim, Y. Zhu, and R. Sivakumar., "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces", In Proc. of ACM MOBICOM'03, Sept. 2003.
- [4] M. Stenm and R. H. Katz, "Vertical handoffs in wireless overlay networks", ACM Mobile Networks and Applications, 3(4): 335-350, Dec. 1998
- [5] Upendra Rathnayake, Henrik Petander, Max Ott and Aruna Seneviratne, "EMUNE: Architecture for Effective Mobile Usage of Heterogeneous Networks", NICTA Technical Report July, 2009.
- [6] Upendra Rathnayake, Max Ott and Aruna Seneviratne, "A DBN Approach for Network Availability Prediction", in Proceedings of MSWiM'09, Canary Islands, Spain, Oct 2009.
- [7] Upendra Rathnayake, Mohsin Iftikhar, Max Ott and Aruna Seneviratne, "Mobile Data Transfer Scheduling with Uncertainty", In Press, IEEE ICC'10, Cape Town, South Africa, May 2010.
- [8] A. Seth, S. Bhattacharyya, and S. Keshav, "Application Support for Opportunistic Communication on Multiple Wireless Networks" (<http://blizzard.cs.uwaterloo.ca/keshav/home/Papers/data/05/ocmp.pdf>), November 2005.
- [9] R. Stewart. "Stream Control Transmission Protocol". RFC2960, November 2000.
- [10] Wei Xing, Holger Karl, Adam Wolisz, "M-SCTP: Design and Prototypical Implementation of an End-to-End Mobility Concept", Proc. of 5th Intl. Workshop on the Internet Challenge: Technology and Applications, Berlin, Germany, October 2002.
- [11] Tom Goff, Dhananjay S. Phatak: "Unified transport layer support for data striping and host mobility". IEEE Journal on Selected Areas in Communications 22(4): 737-746 (2004)
- [12] Jin-Woo Jung, Doug Montgomery, Hyun-Kook Kahng, "Mobile IP-Based SCTP Mobility over Wireless Access Networks", Proceedings of Cellular and Intelligent Communications(CIC) 2003, Oct. 2003
- [13] H.Y. Hsieh and R. Sivakumar., "a transport layer approach for achieving aggregate bandwidth on multi-homed mobile hosts", In Proc. of ACM MOBICOM'02,Atlanta, Georgia, USA. Sept. 2002.
- [14] Laszlo Bokor, Szabolcs Novczki, Laszlo Zeke, Jeney Gabor. "Design and Evaluation of Host Identity Protocol (HIP) Simulation Framework for INET/OMNeT++ ", MSWiM'09, Tenerife, Spain, 2009
- [15] C. Larsson, M. Eriksson, K. Mitsuya, K. Tasaka, R. Kuntz, "Flow Distribution Rule Language for Multi-Access Nodes", IETF draft, draft-larsson-mext-flow-distribution-rules-02, Feb 2009.