Achieving Human-Aware Seamless Handoff

David Nunes, Duarte Raposo, David Silva, Pedro Carmona and Jorge Sá Silva
Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
dsnunes@dei.uc.pt, {draposo, djsilva, pcarmona}@student.dei.uc.pt, sasilva@dei.uc.pt

***Abstract*—Smartphones have spawned a new generation of people-centric sensing applications, where their sensors gather information from users to achieve a wide range of objectives, from fitness monitoring to the urban-wide management of traffic. The connections between devices are of utmost importance for these systems; in fact, switching between the multitude of network interfaces available to smartphones (3G/4G, WiFi) can be beneficial to improve connectivity, the distribution of network traffic and save battery power. Current solutions in the literature for the management of network interfaces are limited in the sense that they only consider either application requirements or the system's status. In this paper, we propose a new model that considers not only these aspects but also the human-context, that is, the user's position and activity to intelligently manage which interfaces should be used. To support the switching between different interfaces without interrupting existing connections, we also discuss several handoff techniques. Finally, we present an experimental evaluation of our model based on Multipath TCP.**

Keywords—Network interfaces; Handover; Human factors; Context-aware services

I. Introduction

People-centric sensing applications allow for collaborative data gathering by individuals, facilitated by smartphone's sensors. These applications present dynamic levels of Quality of Service (QoS): some applications have sudden needs for bandwidth and speed to transmit large amounts of sensory data; others may only require the exchange of short messages, while prioritizing connectivity. Harvesting the flexibility of different connection interfaces in smartphones might allow the device to better adapt to the circumstances; for example, WiFi might be used to offload large quantities of data, cellular connections might benefit connectivity and even Bluetooth might be used to pass information in an opportunistic manner. On the other hand, the device's own status (e.g. battery level, available memory) should also influence management of connections. While both these aspects have been previously considered in research, there is another important factor that has yet to be considered. We define the concept of **human user's context** or **awareness** as every aspect of reality that directly relates to the human condition, such as human emotions, activity, logical location or even social networking contacts. These are factors that should be heavily considered in a network management's control-loop. For example, a people-centric sensing app may dynamically provide additional communication redundancy when the user enters areas with known interference issues, or use multiple interfaces simultaneously whenever the user is feeling anxious, in order to avoid further frustrations resultant from intermittent connectivity. In fact, the stability of connections is important, since if connectivity is intermittent the real-time component of the system is compromised [1].

With these issues in mind, this paper firstly discusses the problem of managing networking connections in mobile devices in section II. It then proposes a model for developing human-aware network-selection mechanism that considers the system, application and human contexts in section III. Section IV gives an overview of different technologies for seamless handoff that can be used with our model. In section V, we present our case-study, a people-centric sensing mobile application for nightlife entertainment. In this section, we also validate our model several experiments based on MultiPath TCP (MPTCP). Finally, we conclude our article on section VI with a discussion on the results and future work.

II. Management of Connection Interfaces

Networking needs are dynamic in people-centric sensing, since several different types of data are sent depending on the context. When processing and sending accelerometer data, for example, bandwidth and speed may be important since we are dealing with large amounts of data. However, for receiving notifications or sending location data, connectivity is more important, so that the user does not miss eventual events of interest. Thus, an intelligent network-selection mechanism can in managing networking performance, energy consumption and reduce intermittent connectivity, resulting in more reliable systems. It is also important to note that the overhead for acquiring sensing information may also be crucial in many people-centric sensing apps. However, the analysis of this overhead is out of the scope of this paper. On the other hand, the availability of wireless connections and their signal strength varies greatly as the user moves. In fact, the usage of multiple wireless mediums and network interfaces can not only contribute to a better distribution of network traffic, to also to the increase of Internet coverage and connectivity for mobile environments [2] and even to the energy efficiency of devices [3].

Thus, several QoS and Quality-of-Experience (QoE)-aware interface selection mechanisms have been previously proposed in the literature. Some solutions promoted savings in 3G cellular traffic by trying to offload as much data as possible to WiFi. Through simple history-based predictors that estimated future WiFi throughput, the transmission of data was held until a WiFi network was available [2]. However, to do this, it is necessary to know the application's delay tolerance threshold and the QoS requirements. Two types of solutions were envisioned for this purpose. Firstly, inference mechanisms that use port information and binary names; e.g. packets corresponding to known email ports or belonging to “Outlook” can be delayed [2]. Secondly, APIs used by applications to transmit this QoS information [4]. Other works have expanded on this idea of offloading data to WiFi. For example, in [5] the utility and cost of network flows were modeled to determine when to use the 3G interface and an implementation that took advantage of both 3G and WiFi simultaneously, by extending SCTP, was also presented. On the other hand. alternative approaches focused on specific goals set by the user. Such was the case in MultiNets [6], a system that considered different switching policies (energy saving, data offloading, or performance). Regarding this, the authors in [4] offered a different perspective. They argued that network usage patterns may change quickly and it is not feasible to expect users to understand their system’s behavior and constantly update policies. In [3], the energy cost of mobile connections was reduced through schedulers based on energy models for different radio interfaces and the continued communications history of the user. While this is a step in the right direction for creating intelligent network-selection mechanisms, the approach was limited in the sense that it was only concerned with saving battery power. Additionally, it also only took into consideration the interface’s energy consumption and the user’s usage history, disregarding application’s requirements and user’s intentions.

III. A MODEL FOR CONTEXT-ORIENTED NETWORK-SELECTION MECHANISMS

Current state-of-the-art on networking interface management tends to focus exclusively on achieving a specific objective or leaves the choice of selecting the best high-level goal to the end-user, which is inefficient and undesirable in terms of usability. We believe the limiting factor is the lack of regard of the human context. Observing the development of people-centric-sensing systems, it is natural to consider that it is not only the device’s status (e.g. battery level) or the application’s QoS requirements are of importance to the problem at hand. The human himself should be a motivating factor for considering which network interfaces should be used. Unlike MultiNets [6], which merely considered user-defined policies, we intend to go a step forward and automate such policies in accordance to the human context. This idea may be applied to a human’s psychological state (e.g. emotions), physiological condition (e.g. physical activity) and real-world context (e.g. logical position), factors that can influence the control-loop of the the interface-selection mechanism. Based on the previous discussion there are three main aspects that need to be considered: **System-awareness** - the status of the device, such as the available system’s resources, its power status (charging / battery power / battery level) and link-layer information (e.g. signal strength), among other variables; **User-awareness** - user’s should be able to impose restrictions (e.g. limiting cellular traffic due to monetary costs), and define high-level goals (e.g. performance, independently of battery-life), although these should not be required for the system’s autonomy. Another aspect of user-awareness is the use of sensory information to infer the user’s real-world context. As an example, emotional distress may prompt the smartphone to improve networking performance, in order to promote a better mood; **Application-awareness** - as suggested in [2], [4] and [5], it is very important to consider the application’s requirements in terms of bandwidth and connectivity.

While previous research has proposed architectures for implementations of interface switching engines [6], these are bound to the limitations discussed above. With this in mind, we hereby propose a general architecture for intelligent network selection engines, of which we have developed an experimental implementation. As far as we know, this is the first attempt at defining a general model that combines in a logical and concise manner the major ideas present in the literature, while also innovating by considering the human-context. As we

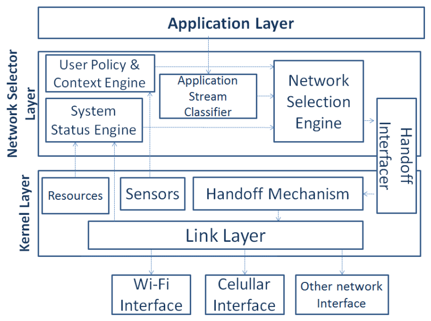


Fig. 1: Network Selection Engine’s Architecture

can see in Figure 1, our architecture considers three layers, the Application Layer, the Network Selector Layer and the Kernel Layer. Applications exist at the **Application Layer** and initiate connections with remote entities through the **Application Stream Classifier**, which evaluates the application’s requirements either through direct methods, such as APIs, as proposed in [4] or indirect methods, such as through mathematical models based on network throughout [7], or both. The **Kernel Layer** includes the system’s lower-level functions that provide information about the system’s resources (e.g. battery-level, available memory), link-layer information (e.g. wireless connection strength) and sensory data. This layer is usually accessed through abstraction APIs provided by the device’s OS for higher-level layers. Handoff mechanisms responsible for switching between the different network interfaces exist at this layer. Some of these mechanisms will be discussed in section IV. The network selection and evaluation process happens at the **Network Selector Layer**, supported by the **Application Stream Classifier** module and the **User Policy & Context Engine** and **System Status Engine** modules. The User Policy & Context Engine processes the policies defined by the user (restrictions in network interfaces, desire for high-performance or power savings) as well as processing contextual information coming from the device’s sensors (e.g. emotional or activity inference). The System Status Engine is responsible for acquiring and processing relevant information from the Kernel Layer about the current condition of the system. These three modules influence the Network Selection Engine, which makes decisions about which networks and interfaces should the system connect to. These decisions are translated to the Kernel Layer through the Handoff Interfacer, which is responsible for interacting with the handoff mechanism in place. We believe this three-layer design fully conceptualizes the key problem in a way that previous designs have not achieved. As evidenced by the state-of-the-art presented in section II, system-aware and application-aware designs have already been extensively considered. Our design builds on top of these concepts, while also introducing the user as a third major factor. Thus, we will now focus our discussion on this human element.

Actual implementations of a User Policy & Context Engine would have to consider techniques for modeling user context. As previously defined in section I, the concept of **human context** can encompass every aspect of reality relating to the human condition, such as human emotions, activity, location, social networking contacts, among others. The work in [7] has previously expressed how much a user values the throughput of a network flow as $U(x)$, a function of throughput x. Modeling $U(x)$ as a logarithmic function for elastic flows (such as TCP downloads) has been common in networking literature, since this function has a diminishing marginal rate of increase with increasing throughput [5]. While our approach does not depend on the nature of this utility function, we do consider a component that specifically translates the human context, $H(k_1, k_2, ...k_n)$, where k_n represents a factor that is relevant to the human (e.g. body temperature, number of social networking contacts, etc.). Assuming a logarithmic behavior, we can write the utility function as:

$$U(x) = K_1 \log(x) + H(k_1, k_2, ...k_n)$$

Considering now the cost for using each networking interface as a function $C(x)$ and assuming that x_w and x_g are the instantaneous throughputs of the WiFi and 3G interfaces of the smartphone, respectively [5], an implementation of our model could maximize:

$$A(x_w, x_g) = U(x_w + x_g) - C_w(x_w) - C_g(x_g)$$

which can be rewritten as follows, to evidence the human factor:

$$A(x_w, x_g) = K_1 \log(x) + H(k_1, k_2, ...k_n) - C_w(x_w) - C_g(x_g)$$

The design of utility functions for specific applications is outside of the scope of this paper. In particular, solutions for the human context function $H(k_1, k_2, ...k_n)$ have extensively been considered in the literature, often requiring advanced mathematical models or machine learning techniques. Nevertheless, we present a simple implementation of these concepts in section V. We would also like to note that user policies can also be represented by adaptive weights in the cost function $C(x)$. For example, we can assume $C_g(x)$ to be a linear function:

$$C_g(x) = K_2 + M_1 x$$

where M_1 is a weight representing the monetary cost established by the cellular provider for using 3G traffic.

IV. Handoff techniques for mobile devices

One important aspect of our proposed architecture rests on the applied handoff mechanism. "Brute-force" handoffs, where one network is simply disabled and another enabled, cause periods of interruption of connectivity, which lead to losses of data and latency. While UDP is the protocol traditionally used for real-time applications, its lack of ordering and error correction schemes make not ideal for people-centric sensing, where robustness depends on the quality of sensed data. Unfortunately, TCP does not play well with handoffs, since acknowledgments from the mobile host may not be delivered. TCP misinterprets the loss of data as a congestion problem, reduces its sending window size and attempts to retransmit packets, while increasing the time between each unsuccessful retransmission exponentially (exponential backoff). This results in long delays and loss of data, even after connection has been reestablished. Since 99.7% of all mobile traffic is TCP [8] this is a serious problem. MobileIPv6 attempted to solve this problem; however, its handoff performance is usually affected by large delays and high data loss rates, making it an unfit protocol for real-time people-centric sensing. While there are some extensions for its improving handoff performance[1], these features come at the cost of modifying the standard to a great extent. Additionally, MobileIPv6 is highly dependent on the existence of proxies and gateways, which can become bottlenecks, single-points of failure and would require an expensive investment in terms of additional hardware, software agents and other changes to currently deployed systems [8]. Handoff techniques based on the manipulation of the device's routing tables have also been proposed [8] [6]. The solution proposed in these works is interesting, since it does not depend on protocol changes or additional infrastructure. Nevertheless, these handoff approaches fail for long-lived TCP flows whose transmission outlives the device's connection to the original network. This may be quite a common occurrence, particularly in cases where links are very transient (e.g. the user is using public transportation) or for complex applications that support real-time sensing data.

MPTCP is a modified version of TCP that naturally allows for make-before-break handoffs. It implements multipath transport by pooling multiple TCP paths from disjoint network interfaces within a single transport connection, transparently to the application[2]. The protocol maintains backwards compatibility with traditional TCP and was designed to flow freely through existing middleboxes. Previous research has used MPTCP to support mobility and reduce energy consumption [9] [3]. These works show that, due to energy constraints, it is often not feasible to use all available network interfaces and that mobile devices often benefit from having just a single primary connection active. Thus, interface selection mechanisms are important to better manage energy consumption and performance. The ability for multipath transport had previously been introduced by modifications to SCTP [5]. However, most middleboxes, highly pervasive in home and enterprise networks, require a deep knowledge of the transport layer to be able to manipulate ports, addresses and keep track of connection states and their unawareness of SCTP leads to the protocol being consequently blocked. SCTP is also fundamentally different from TCP in the sense that it is a message-oriented protocol, rather than a stream of bytes, meaning that applications would need to be modified to support it.

V. Human-aware Handoff Validation

Previous studies have presented simulation experiments with other protocols, namely comparing several mechanisms for improving MobileIP handoff performance [10]. These show that standard MobileIPv6 presents a handoff delay of approximately 814ms, with various of its improvement architectures achieving delays of around 270-450ms. In particular, S-MIP, a seamless handoff architecture for MobileIP, achieves a delay, as perceived by the sender, of a mere 100ms. However, other

[1]IETF RFC7411 - https://tools.ietf.org/html/rfc7411
[2]IETF RFC6182 - https://tools.ietf.org/html/rfc6182

studies based on real test-beds show that Round-Trip Time for standard MobileIP handoff goes as high as 8 seconds [11]. This sort of performance evaluation cannot be applied to techniques such as the manipulation of routing tables proposed in [8], since these are not, in essence, true handoff operations. In this section, we will present the results of an evaluation study that serves as a proof-of-concept for our human-aware model, previously proposed in section III. In this study we implemented a simple User Policy & Context Engine based on the user's movement and position. At the same time, this section further validates the usage of MPTCP as a handoff protocol, in the same vein as the works discussed in section IV.

As a case-study we are developing an Android people-centric sensing app named **HappyHour**. People seek different environments when socializing, sometimes feeling more cheerful and like partying in a crowded place, others just wanting to relax and pass the time while listening to some music and talking with friends. Finding a place that fits our mood is a traditionally trial and error process that involves visiting several places in our surroundings, or a more predictable approach of visiting regular places. Thus, people either spend a lot of time trying to find an adequate place that fits their social group's desires or get bored of always going to the same old places. Our application allows users to find the nightlife establishment in a directed way, by allowing them to know what is happening around them in near real-time. Through a map, as shown in Figure 2, users can check the establishments of interest in their surroundings. The App aggregates the points of interest (POIs) in clusters depending on the zoom level, for easy viewing. By clicking on a POI, a description of the establishment and its currently scheduled events (e.g.,"80's night") is shown. By periodically aggregating the users' GPS positioning and accelerometer readings, the App can also show on the map an estimation of which areas are the most populated and those where people's movements are more pronounced (e.g. due to dancing). This allows users to pick areas with a more vibrant party vibe (more populated and agitated) or calmer areas that are more prone to soothing environments (less populated and agitated). In addition, users can also be informed about promotions and offerings (e.g. drinks 50% off) that are occurring in their surroundings through a notifications system that allows nightclub managers to send notifications to users within 700 meters if their establishment. This system is based on a management web interface (accessed through Facebook) where nightclub / bar owners can create and edit events and schedule notifications to be sent at a certain time. Thus, we offer a new business model that gives establishments the opportunity to reach customers in a more direct way. Figure3

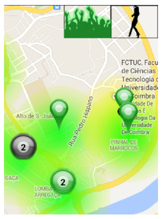


Fig. 2: HappyHour Map screen

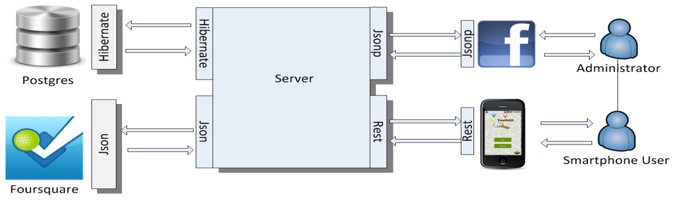


Fig. 3: HappyHour's Architecture

shows the application's architecture. The **Backend server** is responsible for management of users, establishments, events and notifications. The server offers its services in the form of RESTful web services, both to the Android App and to the Management web interface. It also communicates with a database where records of users, POI locations, descriptions and events are kept. The **Android App** is responsible for the interaction with the end-user. It displays a map where the relevant points of interest are shown. It is also responsible for acquiring, processing and sending GPS positions and accelerometer data. The **Management Web Interface** is restricted to establishment owners allowing them to manage events and notifications.

It is often difficult to communicate meeting locations to friends in crowded nightlife enviroments, due to the constant background chatter and movement of nearby people. **FindMe** is a new HappyHour feature that intends to facilitate the process of meeting friends in the vicinity. Having previously registered his friends on HappyHour, a user can enable the FindMe feature, triggering the backend server to periodically check for the presence of friends in the surroundings. Whenever this is the case, the users' respective positions and activity are shared on each other's maps, in near real-time. For this experimental proof-of-concept, we have implemented a simple activity classifier that performs a Fast Fourier Transformation (FFT) on the accelerometer's data, measuring the amount of movement. Depending on this amount, the user's state is classified as either resting, walking or dancing. Each of these activities are represented in the map interface of the user's friends as specific types of markers, placed at the user's current geographical position, as shown in figure 4. However, even a simple classification process such as this can be somewhat taxing on mobile hardware [1]. Therefore, FindMe automatically begins a connection, sending the user's GPS coordinates and accelerometer data in near-real time, leaving the classification process to the remote backend server. Empirically, we estimated that network disconnections would often disrupt our application, particularly when users moved between different WiFi networks. These disconnection periods could be as high as 10 seconds when losing WiFi connectivity due to the reestablishment of TCP connections on the 3G interface. To maintain an accurate evaluation on position and activity, the QoS requirements for FindMe require these disconnection periods to be as low as possible. In order to minimize losses of data resultant from disconnections during the measuring process, we devised a simple User Policy & Context Engine based on MPTCP that implements a $H(k_f, k_a, k_u)$ function with the following factors:

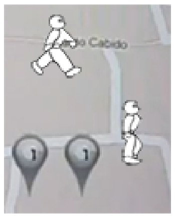


Fig. 4: FindMe

The variable k_f represents a boolean that translates whether or not the user is in the vicinity of a friend. We define "vicinity" in terms of the user's last known GPS position, pos_u, which should be within 500 meters of the last known position of another user in his list of friends, that is:

$$k_f = \begin{cases} 1 & \text{if } pos_u \leq 500m \\ 0 & otherwise \end{cases}$$

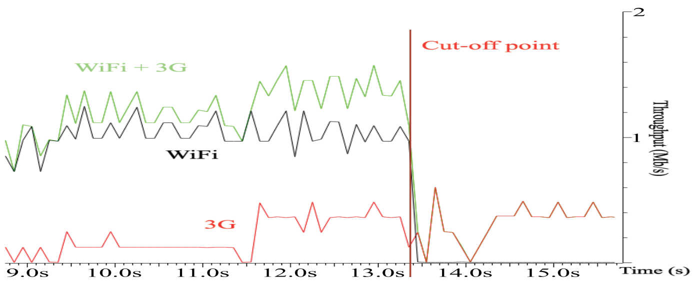
On the other hand, k_a is a boolean that represents whether or not the FindMe function is currently active in the application's settings. Finally, k_u translates a user-defined policy to primarily use either WiFi, 3G or both interfaces simultaneously, whenever sharing his data in near-real time. Thus, the $H(k_f, k_a, k_u)$ function is based on the following strategy:

```
if $k_f = 0$ or $k_a = 0$ then return "the user's policy or context are not within functionality parameters"
else
    if $k_u = Both$ then
        use $Both$ interfaces whenever possible
    else if $k_u = WiFi$ then
        use $WiFi$ as the primary interface
    else
        use $3G$ as the primary interface
    end if
end if
```

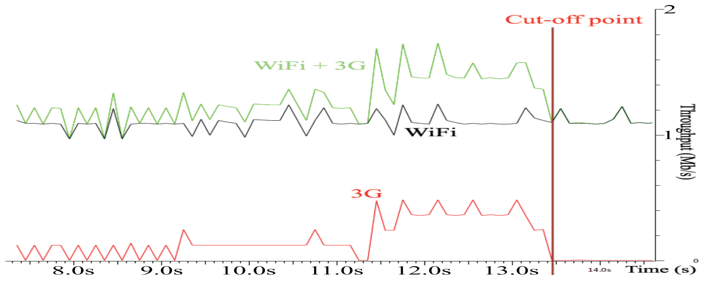
Despite its simplicity, this implementation allows us to consider several ideas previously discussed in section III. The human's context is herein represented by his geographic positioning and activity. Human desires are also taken into consideration through the enabling or disabling of the FindMe functionality, as well as through the selected policy. While using both interfaces achieves greater performance, it also results in additional battery consumption. On the other hand, using primarily the 3G interface can provide greater connectivity, but may induce additional monetary costs. Finally, using primarly WiFi offers good performance at a low cost, but this interface is prone to a limited connectivity range.

While we implemented this experimental User Policy & Context Engine on a Nexus 5 Android phone with a MPTCP-enabled kernel, we decided to perform the handoff tests using UCLouvain's Linux Kernel implementation[3] and a laptop equipped with a WiFi antenna and a 3G USB Dongle. The reason for this was simplicity, as the Linux Kernel implementation allowed greater control over testing conditions. The laptop hosts a virtual machine running an MPTCP-enabled Debian Squeeze and connects to a remote host located on our faculty's hosting farm running the same OS. The connection is made through a private WiFi network supported by a Cable Internet backhaul, offering a bandwidth of around 1 Mbps upstream and 18 Mbps downstream, and a public 3G network, offering a bandwidth between 0.2 and 0.4 Mbps. The laptop connects to the remote host through a standard socket. Initially, we intended to understand how the use of multiple interfaces can affect the handoff and throughput of HappyHour, that is, when $k_u = Both$. This setup focuses on performance and achieves the highest throughput, being desirable whenever the application is directly sending large amounts of accelerometer data (such as in the case of our FindMe scenario). It is also useful whenever the user is moving in a urban environment, due to the periodic connections and disconnections to WiFi hotspots; for example, whenever the user is entering or leaving a nightlife establishment bolstering its own wireless network. After the connection is initiated on both interfaces, a disconnection event was simulated by disabling one of them. This allowed us to study how the connection was maintained by the remaining interface. Figures 5a and 5b show the throughput of the TCP connection on the remote host when both interfaces are active at the same time. As we can see in both cases, when one interface is cut

[3]http://mptcp.info.ucl.ac.be/

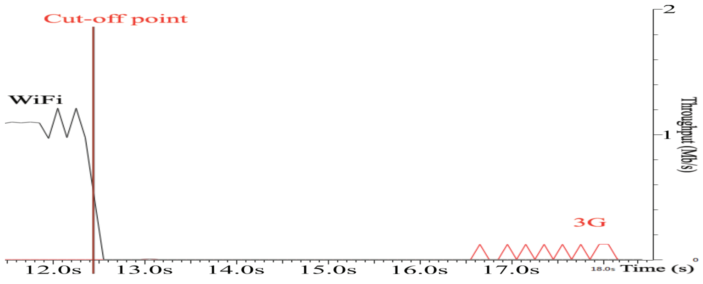


(a) WiFi being cut off

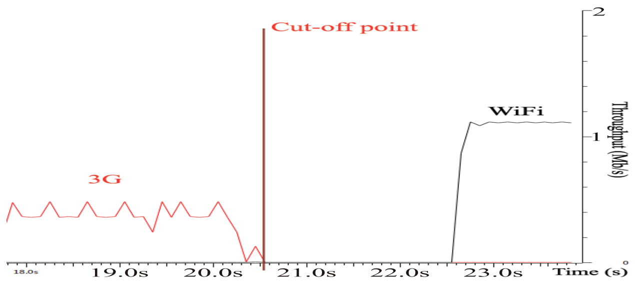


(b) 3G being cut off

Fig. 5: WiFi and 3G used simultaneously



(a) WiFi as primary, 3G as backup



(b) 3G as primary, WiFi as backup

Fig. 6: WiFi and 3G used simultaneously

off the connection effortlessly continues through the remaining interface. There is, of course, a drop on the general throughput, but the dropped packets are quickly resent and the information continues to flow without any major problems. It interesting to note that the maximum achieved throughput is less than the sum of the throughput of both interfaces (achieved around 91% of this theoretical sum).

We also performed a second series of tests for $k_u = WiFi$ and $k_u = 3G$. In these configurations, one interface was configured as backup and the primary interface was disabled mid-test. We imagine this to be a fairly popular setup amongst general users, since WiFi usually offers better performance and energy efficiency at a lower financial cost. Returning to our

HappyHour use case, the user has now defined $k_u = WiFi$ and desires to visit a different establishment. Since the WiFi's range is limited to the establishment, the WiFi interface eventually experiences a connection loss. MPTCP is responsible for rerouting traffic through 3G, as shown in Figure 6a. As the graphic shows, the connection to the server takes about 4 seconds to recover. While the connection has been already established through the 3G interface (the initial three-way handshake and MP_JOIN messages have been exchanged), the 3G subflow's congestion window still has its initial value and needs some time to adapt to the sudden increase in demand. In our tests, the WiFi loss occurs not long after the initial connection has been established, but in cases where a long time has passed since 3G was used, it may be possible that the 3G radio entered in an idle state and, thus, requires some time to wake up. Nevertheless, the connection is not broken and, therefore, it is not necessary to establish a new one under a different IP, as it would happen if only regular TCP was being used. Figure 6b shows a situation where $k_u = 3G$ and thus, 3G has been defined as the primary interface while WiFi has been relegated to backup. Such a configuration, together with using both interfaces simultaneously, is particularly useful setups for vehicular scenarios. Going back to our HappyHour use case, the user is given a ride by a friend and seats on the passenger seat, still using the application to decide where they should go next. When moving at vehicular speed (e.g. 60 km/h), connectivity for WiFi networks becomes very transient and it is difficult to obtain connections that last more than a few seconds. This being the case, using 3G as a main interface or using both 3G and WiFi at the same time is a good strategy for increasing the odds of maintaining stable connections. As we can see, WiFi recovers the connection in about 2,5 seconds, which is faster than 3G. This is expected, since WiFi has a much greater downlink capacity the congestion window increases much faster.

In order to better ascertain the handoff delays for each interface, we performed 10 additional measurements and computed the mean recovery time. We considered the recovery time to be the delay between the last packet sent by the primary interface and the first packet sent by the backup one. The results in

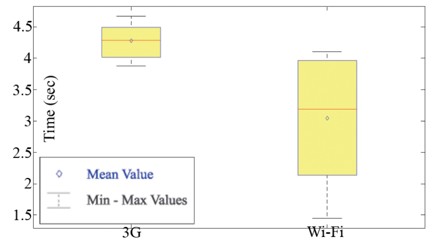


Fig. 7: Recovery times for each interface

Figure 7 show that, while the 3G connection suffers from a larger mean recovery time (4.27 seconds), this delay is also more stable since the values do not deviate too much from the 4 seconds mark. On the other hand, the WiFi connection tends to recover faster (mean of 3.04 seconds) but this recovery also fluctuates a lot more, with several experiments resulting in delays of around 4 seconds.

VI. Discussion and Future Work

We believe that our experimental proof-of-concept serves as ground for the continued development of human-aware network-selection mechanisms. Considering how our own "brute-force" evaluations resulted in disconnection periods as high as 10 seconds and how previous research had presented 8 second handover delays for standard MobileIP [11], we deem MPTCP's handover delays of around 3-4 seconds acceptable. Nevertheless, we believe that better handoff performance might still be achieved through the tweaking of several settings, such as the congestion window initial values. For future work, we are currently exploring the architecture herein presented and applying it to an emotionally-aware network-selection mechanism based on smartphone sensory data. This emotionality-aware system will allow users to select the best networking interfaces to be used, depending on their mood. We expect to publish new results based this more complex human-aware system in the near future.

Acknowledgments

The work presented in this paper was partially financed by the iCIS project (grant CENTRO-07-ST24-FEDER-002003), as well as by Fundação para a Ciência e a Tecnologia and POPH/FSE.

References

[1] M. Musolesi, E. Miluzzo, N. D. Lane, S. B. Eisenman, T. Choudhury, and A. T. Campbell, "The second life of a sensor - integrating real-world experience in virtual worlds using mobile phones," in *In Proc. of HotEmNets '08*, 2008.

[2] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 209–222.

[3] C. Pluntke, L. Eggert, and N. Kiukkonen, "Saving mobile device energy with multipath tcp," in *Proceedings of the sixth international workshop on MobiArch*, ser. MobiArch '11. New York, NY, USA: ACM, 2011, pp. 1–6.

[4] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson, "Intentional networking: opportunistic exploitation of mobile network diversity," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 73–84.

[5] X. Hou, P. Deshpande, and S. R. Das, "Moving bits from 3g to metro-scale wifi for vehicular network access: An integrated transport layer solution," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*. IEEE, 2011, pp. 353–362.

[6] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic, "Multinets: Policy oriented real-time switching of wireless interfaces on mobile devices," *Real-Time and Embedded Technology and Applications Symposium, IEEE*, vol. 0, pp. 251–260, 2012.

[7] S. Shenker, "Fundamental design issues for the future internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 7, pp. 1176–1188, 1995.

[8] A. Rahmati, C. Shepard, C. Tossell, A. Nicoara, L. Zhong, P. T. Kortum, and J. P. Singh, "Seamless flow migration on smartphones without network support," *CoRR*, vol. abs/1012.3071, 2010.

[9] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure, "Exploring mobile/wifi handover with multipath tcp," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, ser. CellNet '12. New York, NY, USA: ACM, 2012, pp. 31–36.

[10] R. Hsieh and A. Seneviratne, "A comparison of mechanisms for improving mobile ip handoff latency for end-to-end tcp," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, pp. 29–41.

[11] S. K. Sivagurunathan, J. Jones, M. Atiquzzaman, S. Fu, and Y.-J. Lee, "Experimental comparison of handoff performance of sigma and mobile ip," in *High Performance Switching and Routing, 2005. HPSR. 2005 Workshop on*. IEEE, 2005, pp. 366–370.