FreeSurf: Application-Centric Wireless Access with SDN

Zhen Cao, Jürgen Fitschen, Panagiotis Papadimitriou
Institute of Communications Technology, Leibniz Universität Hannover, Germany
{zhen.cao,panagiotis.papadimitriou}@ikt.uni-hannover.de, mail@jfitschen.de

1. INTRODUCTION

The increasing need for ubiquitous Internet connectivity has led to the deployment of wireless access infrastructure, including public WiFi networks used for mobile data offloading. According to recent predictions [1], there will be nearly 53 million hotspots by the end of 2018, compared to 22 million in 2013. Despite the density of public WiFi hotspots especially in residential areas, these networks are severely underutilized, with an average of *ten percent* of active users and an average of *one connection per user per month* [2]. Since the revenue from the public WiFi infrastructure is much lower than the cellular counterpart, the operators seek opportunities to monetize their WiFi networks.

One such opportunity emerges from the intention of mobile application vendors to provide near-ubiquitous Internet access for their clients. In fact, the revenue and any potential business opportunities for mobile application vendors are highly dependent on the users' connection time. Therefore, both network operators and mobile application vendors have incentives to march together towards a ubiquitous WiFi for their clients. However, the traditional wireless architecture hinders any such opportunity, since it requires clients to subscribe and authenticate to the *operator* before establishing Internet access.

To overcome this limitation, we propose *application-centric wireless access*, at which Service Providers (SPs), such as Amazon, can authenticate and connect their clients through public WiFi networks, free of charge. Application-centric wireless access requires the delegation of user authentication and access from the operator to the Service Provider (SP) in a secure and auditable manner. In particular, user authentication requests should be redirected to authentication servers deployed by SPs, while access control should be enabled such that only the traffic associated with the network service should be permitted.

In this paper, we present *FreeSurf*, an architecture that enables an application-centric wireless access service over existing public WiFi networks. FreeSurf leverages on software-defined networking (SDN) to provide open APIs for authentication request routing and efficient access control. This enables SPs to install user authentication and data-flow forwarding policies on access points (APs), so that the SP: (i) can steer user authentication requests to its own authentication servers, bypassing the ones provided by the operator, and (ii) can permit the forwarding of the authenticated data flows in a flexible and efficient way. For authentication request routing, we present *AuthFlow*, the first application of SDN to control authentication flows. For policy-based forwarding in the AP, we present a Bloom Filter [3] based OpenFlow table lookup method. To the best of our knowledge, this is the first implementation of BF with OpenFlow.

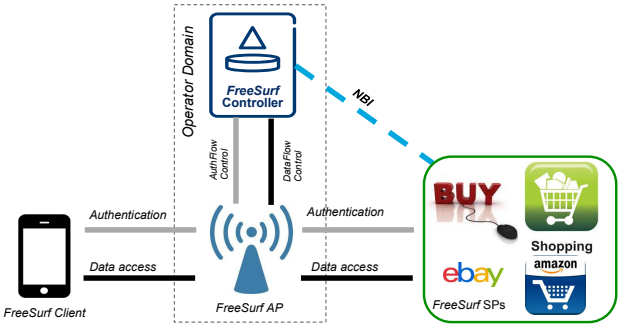

Figure 1: FreeSurf overview.

2. FREESURF OVERVIEW

The major players involved in FreeSurf include the network operator, the SP, and the user. The network operator controls and manages the access points, their backhaul to the Internet, and the FreeSurf controller. The SP deploys the application and authentication servers. FreeSurf requires the following components:

FreeSurf Controller. The FreeSurf controller is responsible for the installation of authentication flow policies (*AuthFlow Control*, grey line in Fig.1) and data forwarding policies (*DataFlow Control*, dark line in Fig.1) on the AP. In addition, the controller provides a northbound interface for the SP to opt in.

FreeSurf Access Point. FreeSurf APs expose APIs to the controller for authentication and access control. The APs are configured to forward authentication requests to the SP's servers and provide Internet access only for the flows that have been authenticated.

FreeSurf Client. The FreeSurf client is authenticated and establishes Internet access to run the application offered by the SP.

In the following, we discuss the main functions pertaining to application-centric wireless access with FreeSurf.

2.1 Authentication

In order to keep the client unmodified, FreeSurf authentication must be compatible with the EAP standard. Fig. 2 illustrates the authentication steps in FreeSurf. If the client wants to authenticate to a SP (*e.g.,* Amazon), FreeSurf specifies an anonymous id (*e.g., anon@amazon.com*) in `msg 2`. This message is examined by the AP. The realm portion (*@amazon.com*) of this id is used to perform a lookup in the *authentication flow table*. Based on the lookup result, the authentication request is forwarded to the FreeSurf SP's Authentication, Authorization, and Accounting (AAA) server, initiating full authentication (*e.g.,* in Fig. 2 the Amazon AAA initiates

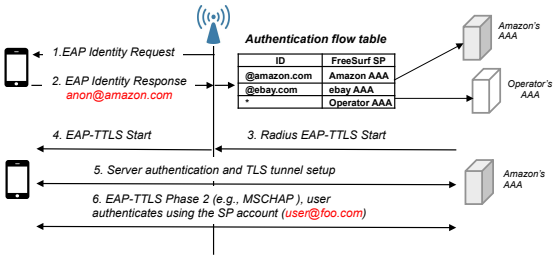

Figure 2: FreeSurf authentication.

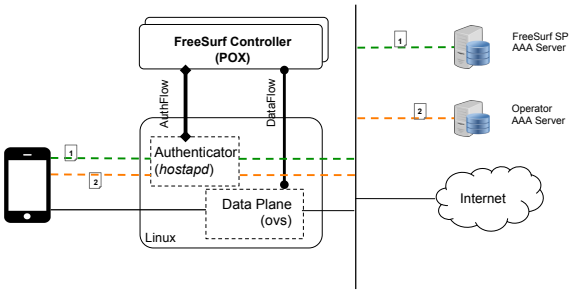

Figure 3: FreeSurf prototype implementation.

an EAP-TTLS authentication (`msg 3-4`)). The server authenticates to the user first with a certificate (`msg 5`), and subsequently the user authenticates to the SP using its associated account name *alice@foo.com* and password (`msg 6`).

Since most applications allow users to choose any valid email address as account names (which cannot be used for the authentication server IP address resolution), FreeSurf authentication requires both an anonymous id and a full SP account name. The use of two authentication identities are supported by EAP-TTLS/PEAP/FAST standards and most mobile devices OSes. Therefore, FreeSurf authentication does not require any client device modifications.

The *authentication flow table* is configured by FreeSurf controllers, such that authentication requests are forwarded to the corresponding AAA server. A low-priority flow entry that matches every request message allows non-FreeSurf users to be authenticated to the Operator's AAA, as shown in Fig. 2. EAP-based forwarding requires payload matching, a feature which is currently not supported by OpenFlow. However, there exist proposals to overcome this limitation (*e.g.,* [4]). In fact, payload matching may be supported by future specifications given the trend for enhancing the versatility of OpenFlow.

2.2 Policy-Based Forwarding

After a successful authentication, the SP forwards to the controller the allowed set of destination addresses/ports and the controller, subsequently, installs the required entries to the AP's flow table. This ensures that only authenticated users will be able to access the service.

FreeSurf relies on Openflow to enforce SP's access control policy. We leverage on bloom filters (BF) [3] to accelerate the table lookup on the FreeSurf AP. BF brings two significant benefits to FreeSurf APs: (i) flow table compression, since the BF occupies K bits irrespective of the table size, and (ii) lookup performance does not degrade with the increase in the flow table size.

BF raises two issues, *i.e.,* flow entry deletion and false positives. We address them as follows. A flow entry can be removed using the counting BF (CBF), which stores a counter value rather than a bit in each slot. The counter is decreased when a flow is removed. As long as the counter is greater than zero, CBF will report a match. On the other hand, false positives can lead to wrong forwarding decisions. However, false positives do not comprise a serious implication for FreeSurf, since they merely permit forwarding traffic from authenticated devices to unallowed IP addresses.

2.3 Operation

Having discussed the main functional components, we hereby exemplify the operation of FreeSurf. Initially, the FreeSurf controller installs the authentication flow entries in the AP. The client discovers the FreeSurf AP and starts the EAP authentication, using *e.g., anon@amazon.com* as the user name. The FreeSurf AP forwards the authentication request to the SP's authentication server by performing a lookup in the *AuthFlow* table. Subsequently, the client is authenticated to the SP using his SP account.

After authentication has been completed, the SP forwards its data forwarding policy (*i.e.,* server IP addresses and ports) to the controller. This initiates the accounting session. The controller then installs the required entries to the AP's flow table.

When the client initiates a connection with the SP's application servers, the AP performs policy-based forwarding, allowing the client to access the service. In this respect, any client's traffic that is not associated with the SP's application is filtered by the AP.

3. PROTOTYPE AND EVALUATION

Fig. 3 illustrates the main components of our prototype implementation (available at [5]). The FreeSurf AP is built atop a Linux laptop with a USB wireless card. The AuthFlow API is implemented via a Python SSH file I/O. The Controller invokes this API to change a local EAP user configuration file of *hostapd* (an open authenticator software) on the AP. This file contains the mapping between FreeSurf SP's realm name and its associated AAA address. On the arrival of client authentication requests, the AP examines the configuration file and forwards the requests to the respective AAA server (`flow 1` for FreeSurf clients and `flow 2` for non-FreeSurf clients).

We rely on OpenVSwitch for the OpenFlow data path implementation on the AP, while POX is used as the controller. In this respect, BF is implemented as a POX module. When OpenFlow table lookup does not yield a match, the AP sends a *packet-in* message to POX. Subsequently, POX uses the BF module to examine if the packet belongs to the access list.

Our experimental results show that FreeSurf incurs a minimal inflation in the delay during authentication compared to non-FreeSurf APs (*i.e.,* 1.7% and 2.4% additional delay with EAP-TTLS and EAP-PEAP, respectively). Furthermore, BF-assisted table lookup reduces RTT by 2%–25% (compared to a non-BF POX module) with flow table size ranging from 1K to 9K entries. As our preliminary evaluation results indicate the feasibility of FreeSurf, future work will be focused on assessing the scalability of the system.

4. REFERENCES

[1] "Visual networking index (vni)," http://bit.ly/1ojjtoE, 2014.

[2] "Public wi-fi survey," http://bit.ly/1bkTAoD, 2013.

[3] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2003.

[4] H. Mekky, F. Hao, S. Mukherjee, Z. Zhang, and T.V. Lakshman, "Application-aware data plane processing in sdn," in *Proceedings of HotSDN*, 2014.

[5] "FreeSurf evaluation code," https://www.github.com/rangzhi/freesurf.