CapAuth: A Capability-based Handover Scheme

Liang Cai
University of California, Davis
lngcai@ucdavis.edu

Sridhar Machiraju*
Google Inc.
machiraju@gmail.com

Hao Chen
University of California, Davis
hchen@cs.ucdavis.edu

***Abstract*—Existing handover schemes in wireless LANs, 3G/4G networks, and femtocells rely upon protocols involving centralized authentication servers and one or more access points. These protocols are invariably complex and use extensive signaling on the wireless backhaul since they aim to be be efficient (minimal handover latency) without sacrificing robustness. However, the mobile user has little involvement especially with the so-called *context transfer* stage; this stage involves the transfer of necessary state to the new access point as well as the enforcement of security goals such as user authentication and single point of access. We propose the incorporation of user *capabilities*, network-asserted proofs of user identity and access control, as a general mechanism to simplify the context transfer stage. To this end, we have designed *CapAuth*, a capability-based scheme that has reduced complexity, low overhead, high level of fault tolerance and is general enough to implement a range of security policies.**

I. Introduction

Highly reliable, available wireless connectivity requires the ability to handle high handover rates and fast handover. Handovers consist of three stages: the discovery of the new access point, the physical layer process of re-tuning to the new access point and the process of *context transfer* [6]. This last stage includes the transfer of state, mostly related to security, QoS, accounting and header compression from the existing (old) access point to the new access point as well as the enforcement of security goals such as authentication and access control.

We propose *CapAuth*, which allows users to play an enhanced role by acquiring *signed capabilities* [7], [17] — signed assertions of context — from access points. During handover, users ensure that their (new) access point has the necessary authenticated context by providing their capabilities. Capabilities and associated mechanisms endow *CapAuth* with qualitative and quantitative advantages over existing schemes.

Fast and simple context transfer: The user-assisted context transfer of *CapAuth* is both simple and fast since it involves a simple message exchange to transfer the capability from the user to the access point. This is in contrast to most existing schemes for context transfer, which have little role for mobile users (apart from being authenticated). Instead, they enable fast handover by relying on complex communication protocols between access points and (centralized) authentication servers (or proxies thereof). Such protocols are often performed proactively [11], [12] before the handover process and may employ sophisticated heuristic algorithms [10].

High tolerance to faults: Existing schemes rely on communication with centralized servers and require significant backhaul capacity as well as server resources. More importantly, when the connection between the access points and centralized servers is lost, handover cannot complete and wireless service is fully disrupted. Recent reports on large-scale wireless network outages [18], [19] have amply demonstrated the lack of robustness and fault tolerance in existing schemes. Moreover, recently deployed femtocells in 3G/4G networks rely on often-congested cable/DSL backhaul links. This makes seamless handover from/to femtocells quite challenging. Under similar failure conditions, *CapAuth* will not lose global availability because a new access point can retrieve the necessary state from the user's capability. Thus, *CapAuth* possesses high levels of tolerance to faults.

Flexibility to enforce a range of security policies: Handovers have two main security goals: user authentication and ensuring single point of access. The latter ensures that they are serviced only by one access point in the network. This is often ignored by systems providing free WiFi, but is the foundation of most commercial networks that require payment for usage. The capabilities used in *CapAuth* help us decouple these two security goals. Specifically, these signed capabilities allow (new) access points to perform user authentication (and secure establishment of state) while allowing for other mechanisms to ensure single point of access. Thus, they allow a broader range of tradeoffs between performance and access control goals. In particular, *CapAuth* allows an access point to employ *optimistic* policies that provide authenticated users service without verifying that they are being serviced by no other access point.

Robustness and security: To enforce a single point of access, *CapAuth* uses a provably secure and robust protocol between access points while minimizing reliance on communication with central servers. This protocol is not only simple (it has 4 messages in the common case) but also guards against malicious access points. In addition, the overhead of guarding against malicious access points can be configured for any desired trade-off with the extent of malicious behavior. In contrast, many existing schemes [10] achieve fast handovers by pre-distributing keys to several access points, at the expense of dismissing the underlying threat model in which access points are not trusted.

This work is supported by the National Science Foundation under grant 0644450 and by a generous gift from Sprint. We thank Jean Bolot, Hui Zang, and Jamie Schneider for their valuable comments and help.

*The author was at Sprint when this work was done.

II. Motivation

A. The Handover Process

The handover architecture and protocols used in WiMax networks and 3G networks are quite similar. Access points are connected to an upstream aggregator (ASN-GW or PDSN). While the access points manage the wireless link with the mobile device, the ASN-GW performs radio-independent functions including that of authentication and handover. The ASN-GW communicates with an AAA server that maintains user and QoS information specifying what the user can access.

Handovers typically consist of three stages. In the first stage of **discovery**, the mobile device receives information about neighboring access points from the serving access point. Then, the mobile device synchronizes with the neighboring access points and measures their signal strength, timing, etc. The second stage is that of **context transfer**. Before a hard handover, mobile devices use a handover algorithm to decide target access points for handover and send this list to the serving access point. The latter informs the target access points about a possible handover which can then retrieve necessary state such as service authorization and keys from the ASN-GW. In the third **reconnection** stage, the mobile device chooses a target access point, sends disconnection message and reconnects to the chosen target access point. Note that, if the latter does not have any context, it will have to retrieve it. It also sets up a new data path from the ASN-GW.

There are two main requirements for context transfer in handovers. The first requirement is that of **state** establishment. To provide service to the user, the new access point needs to be able to access the necessary context, which may include authentication, authorization and QoS information. MAC-layer scheduling information may also be part of the context. The second requirement is that of **security**. The context being transferred includes information that the provider must use to authenticate the user as well to verify if and how much service to provide. The security imperatives of context transfer are information privacy, transfer legitimacy and security preservation [6]. In particular, the provider needs to preserve a single point of access per user.

Existing mechanisms for context transfer in cellular networks have three key disadvantages. First, they rely on **complex** state-transfer and negotiation protocols involving the access points and ASN-GW. Second, they are **not fault-tolerant** since they cannot function if the ASN-GW or AAA server fails. Third, the hard and soft handovers can prepare multiple access points for handover. To our knowledge, the network does **not guarantee a single point of access** and instead relies on the difficulties inherent in duplicating (or cloning) the authentication hardware such as SIM cards.

In open Wi-Fi networks, it has been found [9] that Wi-Fi handover latency is dominated by the discovery process since it involves scanning the 802.11 frequencies for access points. There have been many proposals to eliminate the scanning time (for example, see [13]). Even with fast scanning, repeated disconnection and reconnection using strong EAP-based protocols induces high latencies [3]. These latencies are exacerbated as the round trip times to the authentication servers increases and/or the bandwidth decreases.

B. Goals and Assumptions

Threat Model: We assume the network provider trusts the centralized server(s) for user authorization. The centralized server(s) possess a master keypair with a public key that is well-known to users and access points. Without lack of generality, we assume that each access point has a keypair certified by the provider using its master key, although other setups, such as PKI, suffice as well. We do not trust mobile devices.

Security and Robustness Requirements (1) Unauthorized mobile users should not be serviced by uncompromised access points. (2) No single user should be able to acquire service from multiple uncompromised access points simultaneously. (3) Malicious users attempting to violate any of the above goals must be detected as quickly as possible. Note that a compromised access point can provide or deny service to any user - honest or malicious. In fact, a compromised access point can eavesdrop on the messages sent from/to honest users associated with it unless those users employ end-to-end authentication. As is typically the case with 802.11i, preventing this is not our goal.

We also aim to make the design simple and fault-tolerant. We are particularly interested in eliminating the reliance on remote servers, such as AAA, and the associated backhaul signaling. Under such failure scenarios, the handover process should be successful and the above goals fulfilled.

III. Capability-based Architecture

In most existing schemes, the mobile user has mostly been uninvolved in the context transfer stage of handovers even though the user participates in the discovery and reconnection stages. Given that the user is directly connected to both the old and new access points during a handover, the user may be able to facilitate context transfer significantly. This observation motivates us to explore how the mobile user can assist the context transfer. Here and in the rest of this paper, we assume a canonical handover scenario where a user U hands over from an access point A to an access point B.

We propose to involve the mobile user in the context transfer stage using *capabilities* [7]. Capabilities are “tickets” that give the user rights to access objects. We now describe how we use signed capabilities [17] for wireless authentication. Consider, for instance, that user U has a network identity id and QoS profile Q, which are all the context that the access point requires to provide service. When a user signs up for service with the network provider, a central server constructs as the user capability a signed assertion verifying the user’s identity and QoS profile. Specifically, the provider uses his master public-private keypair (K, K^{-1}), where K is the public key and K^{-1} is the private key, to generate a capability $C_A = [id, Q]_{K^{-1}}$ ($[x]_{K^{-1}}$ denotes x signed by K^{-1}).

The provider also uses the master keypair to sign the individual keypairs of access points. These signatures are time-constrained to limit exposure in case of access point compromise. Thus, access point B with a keypair (K_B, K_B^{-1}) would be provided with a signature $[K_B, T]_{K^{-1}}$ where T is the expiration time of the signature. The system provides users with the master public key K to allow them to authenticate the access points themselves.

When a user U associates with an access point B, U provides this capability as context using the following protocol:

1) $U \rightarrow B$: **UserREQ**(m).
   where m is a nonce generated by the user to prevent replay attacks.
2) $B \rightarrow U$: **AuthREQ**(n, $[m]_{K_B^{-1}}$, $[K_B, T]_{K^{-1}}$).
   where n is a nonce generated by B. U can authenticate B by verifying the signature $[m]_{K_B^{-1}}$
3) $U \rightarrow B$: **AuthRESP**
   $(\{C_A, n\}_K)$ $(\{x\}_K$ denotes x encrypted by K).
4) $B \rightarrow U$: **AuthACK**
   after which user is provided service.

Here we assume that all access points have access to the single private key K^{-1}. This makes it possible for a single access point vulnerability to compromise the entire network. We can solve this problem by a hierarchical trust system.

So far, a user needs only a valid capability C_A to authenticate to an access point. However, once an access point becomes compromised, it can masquerade as any user whose capability the access point possesses. Hence, certain networks, including GSM-based ones, limit such vulnerability by using *authentication vectors*, which allow users to be authenticated without revealing their secret. We can incorporate such mechanism into our protocol, too. One way is to use a user's public key as his id (and only the user knows the corresponding private key id^{-1}). To successfully authenticate, the user needs not only provide his capability but also demonstrate the knowledge of his private key, as in the modified step 3 below:

3) $U \rightarrow B$: **AuthRESP**(id, $[n]_{id^{-1}}, \{C_A, k\}_{K_B}$).
   where k is the session key generated by U.

The above protocol is our first-cut design of *CapAuth*. It results in very short handover time. The new access point provides U with service immediately after the above four messages since the necessary context is readily available to B and the access point has authenticated the user without any communication with other network entities (such as AAA servers). The above protocol also ensures the security goal that only authorized users are serviced by uncompromised access points even when other network entities fail.

There are two problems with the above protocol. The first problem arises because a malicious user may clone his capability and then obtain services simultaneously at multiple access points. Therefore, we need a mechanism by which each capability can be used only once. The second problem with the above protocol is that of *revocation*: a user who is assigned a capability can use it forever. In the next section, we discuss how we improve *CapAuth* to solve both of these problems.

IV. Ensuring Single Point of Access

A. Optimistic Access

Our optimistic approach is motivated by prior work on optimistic concurrency control in databases. Unlike locking-based schemes, which prevent conflicts at the cost of high algorithmic overhead and delay, optimistic concurrency control schemes optimistically provide concurrent access; when conflicts do occur, they roll back conflicting transactions. We argue that the risk-performance tradeoff enabled by optimistic approaches is well-suited for wireless handover especially with capabilities. To see why, consider an access point B that begins to service a user after the simple capability-based protocol is completed and *before* a central registry is contacted. Clearly, this ensures that user-perceived latencies depend only on the 4 messages of the simple protocol. If and when the central registry reports a violation of the single point of access, user service can be terminated. Thus, an optimistic scheme can trade off performance (in terms of handover delay) for the potential risk (of temporarily providing gratuitous access at multiple points to a single user).

B. Exploiting Locality to Implement Revocation

So far, we have considered capabilities as being assigned once to every user and never expiring. This implies that capabilities cannot be revoked; this is contrary to the authentication process. A simple solution is to include an expiration timestamp in the capability and have the centralized servers be responsible for re-issuing capabilities. A malicious user could obtain (optimistic) service until his capability expires. Thus, there is an obvious tradeoff between the re-issuing overhead (communication with centralized servers) and the amount of revoked access users can obtain.

The above tradeoff can be eliminated if the capability-issuing authority is delegated to access points (APs). With *AP-issued capabilities*, there are three advantages:

- There is minimal communication overhead for re-issuing since only the wireless link is utilized.
- The lifetimes of capabilities can be relatively short and also include dynamic state (e.g., QoS reservations and MAC-layer scheduling state) at the access points.
- We can exploit locality to quickly enforce single point of access since an access point needs to contact a nearby capability-issuing access point. The decreased reliance on remote servers also improves fault tolerance and the load on those servers. Indeed, prior handover protocols [10], [16] have used different levels of access point intelligence to reduce the involvement of centralized servers.

To modify our scheme to use AP-issued capabilities, we first extend the capability to include the expiration and signing authority's identity. Thus, our capability contains: (1) User identity, e.g., MAC ID. (2) Context, including the service(s) that the user has signed up for, e.g., bandwidth limitation and QoS requirements. (3) Signing information, including the identity of the signing authority, proof of such identity (possibly a certificate signed by the master key of the network provider) and its physical address. (4) Timestamp, storing the validity duration of the capability.

When a user joins the network at access point A, the capability held by the user may have expired, in which case A must verify the single point of access. In addition, the centralized server also delegates A to be the capability-issuing authority for user U. This also means that A is considered to be the single point of access for U. When U hands over to access point B, B can use the (short-term) capability to optimistically provide access and verify that A still has the delegation.

C. Transfer of Capability-Issuing Authority

We now develop a protocol to transfer the capability-issuing authority securely from access point A to access point B. Our primary security goal is to ensure that the transfer is secure and that only one access point possesses the authority at any point in time. Unlike previously proposed (proactive) protocols between access points [11], [10], which can execute more transfers of context than necessary, we can execute only one transfer per handover. We start by observing that the capability in our earlier 4-step protocol includes the user's identity and authenticates the user to the access point B. Hence, there is no need to add any additional authentication step.

1) Need for Intermediate States: Consider the state of each access point with respect to a user U. Before the handover, A is delegated the authority and B is not. After the handover, B should have the delegated authority and A have none. This implies that there should be a message m_1 from A whose successful receipt by B will cause it to accept the delegation.

There are two possibilities. The first is that A deletes its delegation before sending m_1. In this case, if m_1 is lost, no delegated authority exists in the network. The second possibility is that A needs to receive an additional acknowledgement message m_2 from B after m_1. In this case, if m_2 is lost, there will be two delegations in the network, which is not desirable.

Hence, A and/or B cannot directly go from having no delegation about user U to having the delegation. Note that this is an application of the famous Byzantine Two-Generals problem [15], [1], which essentially shows that no amount of acknowledgements can ensure that two communicating parties can both reach consensus and be sure that the other party has reached the same consensus. Our transfer protocol is also similar to a typical 2-phase commit transaction in databases.

2) Final Protocol: It turns out that a “minimal” protocol in which a user U caches at most 2 capabilities (and the identities of the delegated access points that issued them) and each access point has just one intermediate state is possible. We describe this protocol next. We start by adapting our simple scheme for capability verification. After the capability provided by the user is verified, the delegation authority is transferred from A to B. As shown above, to transfer the delegation authority from A to B, both access points are in one of three states each:

- *Authority*: The access point is the delegation authority.

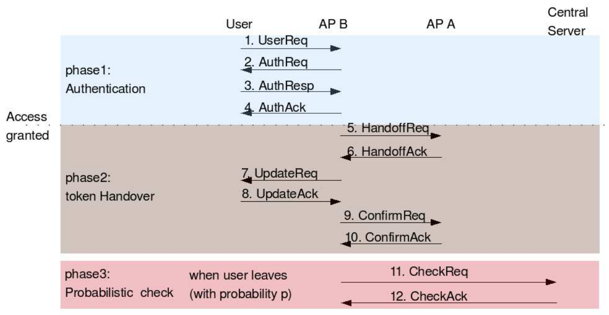

Fig. 1. An Overview of our protocol to transfer delegation authority from access point A to B after handover. The first 4 steps involve the secure exchange of the capability.

- *NoAuthority*: The access point has no delegation authority for user U.
- *TerminatingAuthority*: The access point is transiting from *Authority* to *NoAuthority*.
- *InitiatingAuthority*: The access point is transiting from *NoAuthority* to *Authority*.

Using the above states, the following protocol can transfer the delegation authority, assuming benign users and no message failures:

5) **State**(B, U): *NoAuthority*.
State(A, U): *Authority*.
$B \rightarrow A$: **HandoffREQ**(id, C_A).
Message is periodically repeated until reply received from A.
State(A, U): *Authority* $\rightarrow$ *TerminatingAuthority*.
6) $A \rightarrow B$: **HandoffACK**($\{id, C'_A, n\}_K$).
C'_A may be a context more recent than user-cached C_A.
7) $B \rightarrow U$: **UpdateREQ**(C'_B).
Message is repeated until next message received.
Here, C'_B is a capability that is signed by B and contains the context from C'_A. The user U overwrites any capability other than C_A with C'_B.
8) $U \rightarrow B$: **UpdateACK**().
State(B, U): *Authority* $\rightarrow$ *InitiatingAuthority*.
9) $B \rightarrow A$: **ConfirmREQ**().
Message is repeated until reply is received from A.
State(A, U): *TerminatingAuthority* $\rightarrow$ *NoAuthority*.
10) $A \rightarrow B$: **ConfirmACK**.
State(B, U): *InitiatingAuthority* $\rightarrow$ *Authority*.

Our protocol is designed to handle message losses. Specifically, access point B periodically resends messages until A or U replies as appropriate. We show a schematic of our protocol in Figure 1. Notice that steps $1-4$ involve the initial capability negotiation whereas the remaining steps involve transfer of capability-issuing authority. The communications between the user U and B are secured using the session key from step 3 whereas the communications between A and B are secured since both have keypairs signed by the master key K^{-1}.

In some cases, the access point B may not receive the message 8 from U indicating that it received C'_B. This may happen if U moves away, for instance. In this case, B cancels its delegation transfer.

9) $B \rightarrow A$: **CancelREQ**().
   Message is repeated until reply received from A.
   State(A, U): *TerminatingAuthority* $\rightarrow$ *Authority*.
10) $A \rightarrow B$: **CancelACK**.
   State(B, U): *InitiatingAuthority* $\rightarrow$ *NoAuthority*.

Note that message 7 causes U to store C'_B. Thus, if previous delegation transfers fail, then U can have multiple capabilities. For example, message 8 of a previous transfer to access point Z may fail causing Z to cancel the transfer. If U provides C_Z to access point B, then B will receive an error notification as message 6. In such a case, B asks U to delete the invalid capability C_Z and will try the next capability provided by U.

6) $A \rightarrow B$: **ErrHandoffREQ**().
7) $B \rightarrow U$: **DeleteCAP**(C_Z).
   Message is repeated until next message received.
8) $B \rightarrow U$: **UpdateACK**().
   Message is repeated until next message received.

V. Related Work

Pre-authentication: The IEEE (802.11i) pre-authentication scheme allows the AAA server to pre-distribute keys to multiple access points. These can be chosen as those likely to be visited by the user (see [11], [12]). IEEE 802.11r is an effort to enable fast handovers by establishing security and QoS state at the new access point before roaming. The old access point is used as a conduit to the other (potentially future) access points (Fast BSS transition). As with soft handovers in cellular networks, it is unclear how these protocols enforce the single point of access. Moreover, they continue to heavily depend on a central server and hence suffer from fault tolerane issues.

Proactive Key Distribution: Proactive schemes typically use the Inter-Access Point Protocol [16] of IEEE 802.11 standard. Prior work [10] has proposed using predictive mobility models for proactive context transfer as part of IAPP. This work enforces the single point of service by moving context from the current access point to the new access point; however, upon full authentication, it requires the access point to explicitly remove the stale state at other access points. Duong et al. [4] proposed a scheme to estimate the best time to transfer context for low latency handovers. Intelligent caching [8] has also been proposed as another solution. Pro-active schemes either use heuristics such as mobility prediction that may not always work or have high overhead since they set up context at more access points than required. Moreover, these schemes [10] transfer authentication keys (Pairwise Master Key or PMK in 802.11i) between access points and thus assume that all access points are fully trusted.

Capabilities: Capabilities are widely used in access control [17]. While schemes such as MS Live Passport use capabilities for delegated authentication, their main goal is single sign-on. By contrast, the main purpose of *CapAuth* is to achieve fast handover. To our knowledge, we are the first to propose using strong, secure capabilities for wireless authentication (the scheme proposed in [2] uses a weaker, less-functional construct). Our work can be viewed as extending user involvement in the handover process, which has so far been restricted to the discovery stage [14]. Prior work has focussed on reducing the handover overhead in terms of implementation cost, e.g., [5], or latency costs [20]. Such decisions are not the focus of this paper.

VI. Conclusion

We designed *CapAuth*, a capability-based scheme for high-performance handovers. *CapAuth* is characterized by two key ideas - the use of capabilities to achieve user-assisted transfer of context to new access points, and the decoupling of user authentication from the enforcement of single point of access. *CapAuth* enables low latency handovers with minimal communication overhead (especially on the backhaul) and high degree of fault tolerance.

References

[1] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. *SIGOPS Oper. Syst. Rev.*, 9(5), 1975.

[2] T. Aura and M. Roe. Reducing Reauthentication Delay in Wireless Networks. In *Proc. of SecureComm*, 2005.

[3] A. Bohk, L. Buttyn, and L. Dra. An Authentication Scheme for Fast Handover between WiFi Access Points. In *Proc. of ACM Wireless Internet Conference (WICON)*, 2007.

[4] Ha Duong, Arek Dadej, and Steven Gordon. Proactive context transfer and forced handover in ieee 802.11 wireless lan based access networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(3), 2005.

[5] B. Hamadaoui and P. Ramanathan. A network-layer soft handoff approach for mobile wireless IP-based systems. *IEEE Journal on Selected Areas in Communications*, 22(4), May 2004.

[6] J. Kempf. Problem Description: Reasons For Performing Context Transfers Between Nodes in an IP Access Network, 2002. RFC 3374.

[7] Henry M. Levy. *Capability-Based Computer Systems*. Butterworth-Heinemann, 1984. http://www.cs.washington.edu/homes/levy/capabook.

[8] W. Liang and W. Wang. A Quantitative Study of Authentication and QoS in Wireless IP Networks. In *Proc. of IEEE INFOCOM*, 2005.

[9] A. Mishra, M. H. Shin, and W. A. Arbaugh. An Analysis of the Layer 2 Handoff costs in Wireless Local Area Networks. *ACM Computer Communications Review*, 33(2), 2003.

[10] Arunesh Mishra, Min ho Shin, and William A. Arbaugh. Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network. In *Proc. of IEEE INFOCOM*, 2004.

[11] Sangheon Pack and Yanghee Choi. Fast Inter-AP Handoff Using Predictive Authentication Scheme in a Public Wireless LAN. *IEEE Networks*, August 2002.

[12] Sangheon Pack and Yanghee Choi. Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model. In *IFIP Personal Wireless Communications*, 2002.

[13] I. Ramani and S. Savage. SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks. In *Proc. of IEEE INFOCOM*, 2005.

[14] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. 1996.

[15] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. John Wiley and Sons, 2004.

[16] IEEE 802.11 Standard. Draft Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation, 2003.

[17] A. Tanenbaum, S. Mullender, and R. v. Renesse. Using Sparse Capabilities in a Distributed System. In *Proc. of 6th Intl. Conference on Distributed Computing*, 1986.

[18] http://telephonyonline.com/access/news/Sprint_service_outage_011006/. Dual fiber cut causes sprint outage, 2006.

[19] http://www.informationweek.com/story/showArticle.jhtml?articleID= 170700333. Wireless services returning after massive hurricane disruption, 2005.

[20] Haitao Wu, Kun Tan, Yongguang Zhang, and Qian Zhang. Proactive Scan: Fast Handoff with Smart Triggers for 802.11 Wireless LAN. In *Proc. of IEEE INFOCOM*, 2007.