COMPASS: DECENTRALIZED MANAGEMENT AND ACCESS CONTROL FOR WLANS

ARTUR HECKER

hecker@wave-storm.com,
Wavestorm SARL
37-39 rue Dareau, 75014 Paris, France

ERIK-OLIVER BLASS

erik@erik-blass.de
Karlsruhe, Germany

HOUDA LABIOD

labiod@enst.fr,
GET-Télécom Paris, LTCI-UMR 5141 CNRS, ENST
46 rue Barrault, 75013 Paris, France

In this paper, we propose COMPASS, a new decentralized access control architecture for modern WLANs. As traditional centralized access control systems like AAA do not scale well, we propose the use of P2P technologies for the distribution of management data directly between the deployed WLAN access points. Our system COMPASS does not require any additional equipment or central entities. Using auto-organization and fault recovery mechanisms of modern P2P systems, it is robust and easier to maintain. Standard 802.1X mechanisms on the user link guarantee compatibility to the existing user equipment.

1. Introduction

Wireless local area networks as defined by the IEEE 802.11 standard [1] experience a tremendous popularity. However, some deployment hesitation has been observed in the industry which is believed to be due to the security and management issues with such installations.

Modern 802.11 security is based on the IEEE 802.1X standard [2]. 802.1X defines an extensible authentication framework using IETF's Authentication, Authorization and Accounting (AAA) protocols [3][4]. Thus, 802.1X is typically used with a central AAA server. 802.1X is the chosen base technology for the expected IEEE 802.11i standard [5].

The scalability, robustness, and cost of centralized solutions like AAA become an issue in the broad installations of WLANs ranging from small private networks to big, multi-site WLANs. Indeed, the AAA approach results in a centralized server farm which is either over or under-dimensioned for the most network sizes. In that sense, it does not support an natural evolutionary growth of the installed network. Also, from the robustness perspective, it introduces a single point of failure (SPF). These two points can be partly resolved by introducing redundant servers. This however results in a more complicated, costly and difficult to manage AAA infrastructure. Besides, given the low access point (AP) prices in the 802.11 segment, the cost of the central server and its maintenance can hardly be amortized in a typical WLAN (up to 30 APs).

However, 802.1X does not oblige the use of AAA. In this paper, we propose to replace the central AAA server through a distributed P2P based access control architecture in the wired network built directly by the access points. We thus present our **Co**nfiguration **M**anagement **P**2P-based **A**ccess **S**ecurity **S**ystem (COMPASS). It integrates P2P technology with the 802.1X access control and does not require any central entity[*]. It relies on modern P2P technologies which provide highly scalable, efficient and fault-tolerant distributed data retrieval mechanisms [6][7][8].

The rest of the paper is organized as follows. In the next section, we present our distributed access control architecture and discuss several important points like self-organization and user management. Then, we give a qualitative comparison and an outlook to the future work.

2. COMPASS: Our New Proposed Architecture

2.1. *Main Idea*

To reduce costs; we propose to use the access points directly for the storage of data relevant to access control and network management. However, since the resources of access points are limited, the idea is to distribute the administrative load over all access points. Thus, in our proposal every AP holds only a part of the whole management database.

The difficult part is to provide a scalable and robust mechanism for distributed data retrieval. Herein, the problem is not the data transfer but locating the data [6]. Distributed Hash Tables (DHTs) [6][7][8] have been designed to overcome these difficulties.

[*] While a *radius* connects a point on the circumference to the center, a *compass* directly interconnects the circle points.

A DHT is a hash table divided into multiple parts (called *zones*) and distributed over several nodes. Famous DHT examples are file-sharing/P2P networks such as EDonkey2000 or Kazaa. We have to choose a P2P system tailored to the restricted resources of an ordinary AP. We namely compare CAN [6], Chord [7] and Pastry [8]. Without going into structural or design details of these networks, here we present a short overview of properties that seem crucial for our work. Table 1 shows properties of common DHTs for a network with *n* nodes. "#Hops for lookup/store" is the expected number of nodes a request for a lookup or store has to pass. The second criterion is the number of elements each node has to store in its neighbor or routing table. Both properties are expected values for a well-balanced DHT network.

Table 1: Properties of common DHTs

Property	**CAN**	**Chord**	**Pastry**
#Hops for lookup/store	$O(n^{1/d})$	$O(\ln n)$	$O(\log_{2^b} n)$
#Elements in routing table	$2d$	$\ln n$	$B*\ln n$
Used in	Secure Service Directory	CFS-file-system	Oceanstore, Scribe

Chord and Pastry seem to perform better regarding the average path length. This means less communication overhead and shorter latencies. On the other hand, CAN's advantage is a constant *O(1)* memory consumption which is known at node setup time, prior to network use. For that reason, we choose CAN.

2.2. *Basic System Architecture*

Based on standard TCP/IP networking in the core network, our P2P management network is formed of the deployed 802.11 access points. This is illustrated in Figure 1. Every access point acts as a P2P node building a logical overlay network over the physical core network. This overlay stores different logical databases, primarily user and management databases (DB). The user DB stores AAA-like user profiles. The management DB helps the administrator manage all connected APs and stores AP settings expressed in the respective syntax (e.g. 802.11 MIB variables, proprietary manufacturer settings, etc).

On user demand, the solicited node retrieves the correspondent profile by using overlay's lookup method. Using the retrieved profile the serving AP follows the usual 802.1X procedure acting as Authenticator with a local Authentication Server [2].

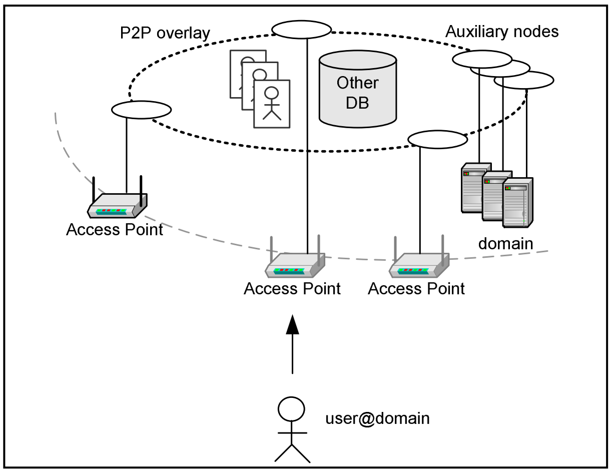

Figure 1 Main entities in COMPASS

2.3. *Preliminary AP Configuration*

Each AP needs a minimum configuration prior to its deployment in the network. This is necessary for a secure management access to this AP, the overlay discovery and the classical 802.11 settings. The trust relationship with the AP is expressed by the installation of a signed certificate on every AP. In addition to the usual 802.11 parameters, the administrator supplies the bootstrap-address of the overlay network and deploys the AP in the desired location.

2.4. *Bootstrapping (AP join)*

The original CAN proposal [6] makes use of a bootstrap method based on a well-known DNS address. In CAN this method guarantees a uniform partitioning of the index space. However, it also means that a physical neighborhood does not result in CAN neighborhood.

We want to be able to tie the overlay to the network topology. This can potentially shorten the handoff-delay since the new AP can rapidly retrieve the profile data from the old AP using the overlay (since the old AP is also the overlay neighbor). The other reason for reflecting the physical topology in the overlay is a transparent load balancing: we suppose that the administrator installs an additional AP in the neighborhood of every AP which suffers a heavy traffic. If the APs are CAN neighbors, they also share the administrative load. Otherwise they only share the 802.11 traffic load. CAN's landmark-ordering method to reflect the physical network topology to the overlay explicitly targets the IP layer topology. We define a 802.11-adapted landmark-ordering mechanism:

1. Booting up, a preconfigured AP searches the 802.11 environment for 802.11 neighbor APs. The necessary mechanisms are defined in the 802.11 standard and include an active and passive discovery of neighboring APs of the same SSID [1]. The joining AP compiles a list of (wireless) MAC addresses of all neighboring APs configured with the same SSID.
2. The joining AP now sends a discovery request to the predefined DNS address of the overlay. The request contains the MAC address list (step 1). If it is not empty, the solicited bootstrapping node chooses from it the node whose zone is the biggest. It is essential to provide a mechanism resolving a wireless link MAC address into the management IP address without global network knowledge. We achieve this by storing these data into the overlay itself, but other mechanisms could be applicable as well. Thus, the solicited node executes a lookup in the overlay for every listed MAC address. The returned value is a pair `(IP_address, zonesize)`. The solicited node chooses the pair with the biggest zone size. If the received wireless MAC address list is empty, the bootstrapping node proceeds as in CAN randomly choosing a node. In either case it replies with the IP address of the chosen node.
3. The join procedure itself is like in CAN. After the joining procedure, the new AP executes a store command in the overlay, posting its own wireless MAC address, the management IP address and the zone size. Each AP is responsible for the validity of that entry.

Following this scheme, the new installed AP automatically becomes an overlay neighbor of one of its 802.11 neighbors. The advantages of this scheme are an equal pre-configuration of all APs and the requested binding of the overlay to the physical topology. Our method does not affect the scalability since the preconfigured overlay address can correspond to several APs. This can be achieved with a round robin DNS or with a multicast address (e.g. "all overlay APs"). Moreover, the join events are expected to be much rarer than the operational procedures like user access. During an initial deployment (or after a complete system failure), no node is available under the bootstrap address. This case is rarer than join events and needs a special treatment. It can be resolved by weakening the equality of the AP configuration (choose some nodes for join requests) or by dynamically updating the round robin DNS on new node joins and departures. Such mechanisms are however out of scope of this document.

2.5. *AP Leave*

AP leave events can occur because of an AP failure (e.g. power down) or because of a admin shutdown. If an AP is shut down correctly, it proceeds like a CAN node. In a case of a power down, the zone databases held by this AP are lost. CAN redundancy mechanisms are used in that case.

2.6. *User Add/ User Delete*

To add a new user record to the system, the network administrator executes the commands `store(username, profile)` or `delete(username)` on one of the nodes. Herein, the profile is a list of authorizations. Principally, such profile could be in an arbitrary suitable format (e.g. attribute value pairs). The profile defines the authentication method, the restrictions and session parameters A typical profile hardly ever exceeds 10kB.

2.7. *User Network Access*

When a user accesses a COMPASS network, user's mobile station (MS) and the solicited AP start the typical 802.1X authentication process. Within this process, at some point of time MS sends an `EAP Response/Identity` message containing the identity string. The solicited AP retrieves the corresponding user profile from the overlay by invoking an overlay lookup for this string as a key. On the receipt of the profile, the AP can continue the EAP conversation as defined in the profile acting as a 802.1X authenticator with a local 802.1X authentication server (AS). The identity used by the AP is an abstract identity of the whole overlay which acts as one logical entity.

2.8. *Failure Management and Optimizations*

We distinguish two major possible failures: the path failure (i.e. some of the nodes in the lookup/store path fail) and the end node failure (i.e. the zone database is not available). CAN provides mechanisms to counter the impacts of such failures [6]. In our particular application the path failure can be countered by increasing the number of CAN's dimensions *d*. That increases the number of tried paths. If the overlay stores only one DB copy, the failures of nodes holding a zone database result in associated data not being available. CAN provides data replication methods. By using multiple realities [6] or multiple different hash functions, the same zone database can exist on multiple nodes. We encourage the use of such mechanisms for user DB.

We demonstrate this at an example of CAN with *k* realities [6]. Let *f* of *n* nodes fail. Assuming that at least one overlay path still works, the probability that a stored pair can be retrieved is:

$$p = 1 - \left(\frac{f}{n}\right)^k$$

Example: in a CAN with 4 realities, if ¼ of all nodes fail without path failures, there is still 99.6% probability that an authorized user can use the service.

3. Discussion

CAN technology has been recently used in sensor networks [9]. Compared to sensors, the WLAN APs are powerful machines with about 16-32MB RAM and a CPU of about 150MHz. Recent APs with an embedded Linux OS show that the available resource safety margin is sufficiently large for additional tasks.

Table 2 Comparative chart of user management methods in modern WLANs

Access Control	User mobility	Admin. effort	Network extension		Partial system failure impact
			Effort	Scaling	
Single central AS	Full	Average	Easy	Bad	Fatal (SPF at AS)
Mult. ASs with user DBs	Full	Highest	Hard	Good	No access for some user groups
Mult. ASs with one DB	Full	High	Average	Bad	Fatal (SPF at DB)
APs with local user accounts	No	Low	Hard, limited	-	No access for locally stored users
AP with user accounts	Full	Average	Easy	Very bad	Fatal (SPF at AP)
COMPASS	Full	Low	Auto	Good	Adjustable over *d* & *k*

As mentioned, CAN has constant memory requirements. The management database stores settings valid for every AP. Its size is thus independent of the number of APs. Given a typical profile size of about 1-10kB, 30-100 user profiles per AP can principally be stored without any impact on AP performance. User profile size can be further reduced by using group management. Redundancy mechanisms can not be reasonably used when the overall number of APs is very low (say for up to 5-10 APs). When using redundancy mechanisms, the overall database size has to be multiplied by the redundancy factor *k*. For instance, in a network with 10 APs, 100 user profiles and the redundancy factor *k*=2, the zone database on each AP is about 5kB*100*2 / 10 = 1MB. These values seem to be realistic requirements for modern APs.

In Table 2, we compare different access control architectures in terms of user mobility, administration effort, network extensibility and robustness (expressed by the worst case impact of a partial system failure).

4. Conclusion

In this paper, we propose to integrate P2P technology with access control to provide a system supporting a natural scaling of the management infrastructure. As a P2P architecture, COMPASS inherits the scalability and fault tolerance of the existing P2P technologies. COMPASS is backwards compatible by using standard access methods on the user link and AAA-like user management in the core. It features an easy network extensibility by defining automatic node join. By storing the necessary management settings in the overlay, COMPASS can also be used as network management infrastructure.

References

[1] IEEE Standard 802.11, “Wireless LAN medium access control (MAC) and physical layer (PHY) specifications,” 1999.

[2] IEEE Standard 802.1X, “Port-based network access control,” June 2001.

[3] C. Rigney, S. Willens, A. Rubens, W. Simpson, “Remote Authentication Dial-In User Service (RADIUS),” RFC 2865, IETF, June 2000.

[4] P. Calhoun, J. Loughney, E. Guttman, G. Zon, J. Arkko, “Diameter Base Protocol”, RFC 3588, IETF, September 2003.

[5] IEEE Draft 802.11i, “Draft supplement to IEEE Std 802.11. part 11: specifications for enhanced security”, work in progress.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, “A Scalable Content-Addressable Network , Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.

[7] I. Stoica et al., “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”, Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.

[8] A. Rowstron, P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”, Lecture Notes in Computer Science, 2001.

[9] H.-J. Hof, E.-O. Blaß, T. Fuhrmann and M. Zitterbart, “Design of a Secure Distributed Service Directory for Wireless Sensornetworks”, EWSN 2004, Berlin.