Mirror Presence: Secure Web Identity Resolution and Call Control for WebRTC

Li Li
Huawei Shannon Lab
400 Crossing Blvd.
Bridgewater, NJ 08807, USA
+1 908-541-3500
li.nj.li@huawei.com

Wu Chou
Huawei Shannon Lab
400 Crossing Blvd.
Bridgewater, NJ 08807, USA
+1 908-541-3500
wu.chou@huawei.com

Tao Cai
Huawei Shannon Lab
Bantian, Longgang District,
Shenzhen 518129, P.R. China
+86 755-2897-6925
t.cai@huawei.com

Zhe Wang
Rutgers University
900 Davidson Rd.
Piscataway NJ 08854
+1 732-208-0718
zhewang@cs.rutgers.edu

Zhihong Qiu
Huawei Shannon Lab
Bantian, Longgang District,
Shenzhen 518129, P.R. China
+86 755-2842-4097
qiuzhihong@huawei.com

ABSTRACT

Identity resolution is a critical component in any real-time communication system, including WebRTC that relies on Web identities. The difficulty with Web identity resolution is that there is no uniform Web identity scheme and current Web identity schemes decouple Web identities and Web locations. To address this problem for WebRTC without introducing a central authority, we present a secure personal Web identity resolution framework, *mirror presence*, to dynamically map a user's Web identity to Web locations while the user can move between Web locations. The system is agnostic to Web identity schemes, and it consists of 3 layers of Web protocols and services: 1) bilateral authorization based on OAuth 2.0; 2) Web identity binding based on presence subscription and notification; 3) presence driven call routing and pickup protocols. Each layer of the system can be controlled by users based on security, privacy, and personal preferences. A prototype mirror presence system has been implemented, and experimental results indicated that the approach is feasible and the performance is satisfactory.

Categories and Subject Descriptors

H.3.5 [**Web-based services**]: Information Systems – INFORMATION STORAGE AND RETRIEVAL.

General Terms

Algorithms, Management, Performance, Design, Security, Standardization, Languages.

Keywords

mirror presence, presence subscription, call routing, call pickup, Web service, OAuth, WebRTC.

1. INTRODUCTION

WebRTC is an ongoing joint standard effort from W3C and IETF to specify JavaScript APIs, media codecs and network protocols to enable real-time communications between Web browsers. Although voice and video communication between Web browsers is possible through browser plug-ins, WebRTC is the first industrial effort to integrate rich multimedia real-time communication functions into HTML5 Web applications.

Although WebRTC adopts many protocols from VOIP, such as SDP [1], RTP [2] and ICE [3], its potential is beyond VOIP through the convergence of real-time communication and the Web which is the most scalable and active global information space. WebRTC allows Web developers, or even Web users, to rapidly create real-time Web applications that run on any operating systems and devices. Web browsers will become an open platform for delivering real-time communication and collaboration services, as part of Web based business transactions, online education, healthcare services and social interactions. Over 30 companies and organizations, including Google, Microsoft, Cisco, Mozilla, Huawei, and Ericsson are investing in WebRTC, with the hope to change the telecommunication industry and the Web ecosystem.

However, creating a global communication system over the Web faces some acute technical challenges, and many of them are from the mismatches between the requirements of telecommunication systems and the design of the Web.

One of these mismatches is identity resolution, a process that maps an identity to a location (address) for a call to that identity.

VOIP systems typically use SIP address [4] (URL similar to an email address) to identify users and use SIP Registry, SIP Location Service and DNS system to resolve a SIP URL to one or more SIP user agents. However, this identity resolution system has not been adopted by WebRTC since SIP is not an integral part of the Web.

Web identity resolution system faces several problems. First, there lacks a uniform method to identify a user who makes and receive calls, and instead, many methods are used depending on the Web systems, such as a local name, an email address, or a URI.

Second, unlike the SIP address scheme that always consists of a domain that recognizes the name, some Web identities are not allocated by the websites where they are used. Third, we should not introduce another central authority like DNS to the Web to solve the identity resolution problem.

To illustrate these problems, we briefly describe the two Web identity schemes: authoritative and federated, used by most websites to authenticate users.

In authoritative identity scheme, a websites authenticates a Web identity directly without any third-party entities. For example, a user can log in Yahoo with Yahoo email address, or a News websites with a local name at that site. This scheme ties a Web identity to the websites that uses it. To make a call to this identity, the calling party has to know the identity if it can be resolved to IP (e.g. email address) or the identity plus the websites address, if the identity is not resolvable to IP (e.g. local name).

In federated identity scheme, a websites authenticate a Web identity through a third-part identity provider (a Web server), such that the user can reuse the same identity on several websites that trust the identity provider. An identity provider may implement any of the Web identity protocols, such as OpenID [5], WebID [6], BrowserID [7], and OAuth [9]. This scheme decouples a Web identity from the websites that use it. Therefore, the Web identity cannot be resolved to the correct website. For example, user A may log on a website http://www.example.com with a federated identity a@identity.com. When user B calls this identity, the call should not be routed to identity.com, because it is neither the intended destination nor the site that knows A's current website www.example.com. To make a call to this identity, the calling party has to know the identity plus the websites address. Moreover, due to the decentralized nature of the Web, it is unlikely that the Web will adopt one uniform identity scheme in the short term.

Without a central Web identity system, most Web users maintain several Web identities on different websites, each for a particular communication service and social context. For example, a user may have a GMail identity for personal emails, a Twitter identity for fans, a Facebook identity for friends and a LinkedIn identity for professional relations. In other words, a seasoned Web user maintains *a circle of frequent websites* that constitutes the user's personal Web space.

To address the Web identity resolution problems by taking advantage of personal Web spaces, this paper proposes a decentralized, efficient and secure identity resolution system for WebRTC - *mirror presence*. The mirror presence system dynamically maps a user's Web identity to Web locations based on user presence while he travels in the personal Web space. The system is agnostic to Web identity schemes, and it consists of 3 layers of Web protocols and services: 1) bilateral authorization based on OAuth 2.0; 2) identity binding protocols based on presence subscription and notification; 3) identity driven call routing and pickup. Since each layer can be controlled by users based on security, privacy and personal preferences, mirror presence can be regarded as a personal identity resolution system.

The rest of this paper is organized as follows. Section 2 provides a brief survey of related work. Section 3 describes the overall architecture of the system. Section 4 describes the Web bilateral authorization protocol. Section 5 describes the Web identity binding protocol. Section 6 describes the Web call routing and pickup protocols. Section 7 describes the prototype implementation and experimental results, and we conclude the paper with Section 8.

2. RELATED WORK

WebRTC WG at W3C [8] is working on the standard JavaScript API for Web applications to control and monitor local media (audio and video), data channel, and peer connections between HTML5 pages, while the connectivity protocol (ICE), media transport protocols (RTP), data channel protocols [10], and media codecs (opus, VP8, H.264) are developed by IETF RTCWeb WG [13]. WebRTC does not mandate any identity schemes or identity resolution architecture [11].

The key components and boundaries of WebRTC are illustrated in the following picture (Figure 1) that involves two websites A and B. It is possible that A and B are the same website.

A typical call flow in WebRTC happens as follows: when user A calls user B by providing user B's identity and site B address to page PA, PA will generate a call request message, which traverses websites A and websites B, and arrives at page PB to alert user B. If user B answers the call, a call response message will traverse all the way back to page PA. Once the two pages negotiate the media and transports, they instruct the browsers to exchange media streams directly.

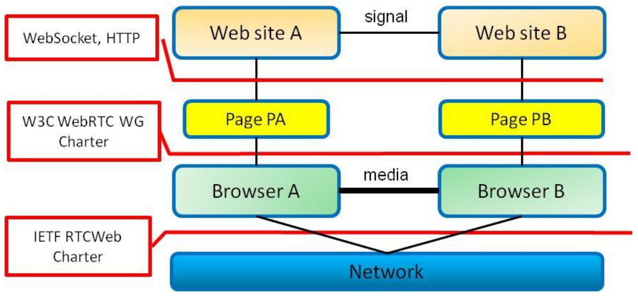


Figure 1: Basic WebRTC architecture

In order for this call flow to happen, user B must be on site B when the call from user A happens. However, user B may not be on site B but is visiting other websites. This imposes a serious constraint on the usability of WebRTC. The mirror presence framework of our approach will allow calls to be routed within a personal Web space based on presence, such that no call will be missed when the user is moving and browsing within the space.

Inter-domain presence federation is supported by SIP [14][15] and XMPP [16][17] systems based on the subscribe/notify design pattern. In these systems, a presence watcher subscribes to a presentity in order to receive presence events from the presentity. However, the SIP and XMPP systems do not support the concept of mirror presence. For instance, XMPP does not provide a mechanism to transfer the presence of a JID to a different JID, such that the second JID reflects the presence of the first one.

FindMe/FollowMe refers to two technologies that enable incoming phone calls to be received at different locations, on different phones [18]. This is achieved in some system by forwarding incoming calls to a virtual number to a set of designated numbers according to call preferences, or user location and presence. While both FindMe/FollowMe and Mirror Presence attempt to locate users, our approach is the opposite of FindMe/FollowMe. In FindMe/FollowMe, a single identity serves as the proxy for many physical addresses. In mirror presence, many identities serve as the proxies for a single Web location. In other words, mirror presence extends a user's presence to many websites at the same time.

Mirror presence concept may appear to be analogous to a 3G cellular network where the browser corresponds to mobile phones, the visitor sites correspond to VLR and the mirror sites correspond to HLR [19]. But there are some fundamental differences:

- In cellular networks, a roaming mobile phone maintains a unique ID (SIM) to the system whereas in mirror presence, a "roaming" browser has no unique ID to all websites.
- In cellular networks, a mobile phone can occur in one VLR and HLR at a time, whereas in our system, a browser can connect to many websites at the same time.
- In cellular networks, the system is configured by operators, whereas in our approach the system is configured by users.

Web tracking technologies can track user's movement between websites. A websites can embed third party Web beacons [20] in HTML pages so that the tracking sites can know when a user visits the site. However, Web tracking raises security and privacy concerns among many user groups [21]. In addition, there are some major technical issues and deficiencies:

- The beacons can tell the tracking sites when a user visits a site, but it cannot tell when the user leaves the site, for example when the browser exits or the network is down.
- Using tracking sites to resolve Web identities to websites requires the users to know each other's tracking site, in addition to each other's identities. However, the users usually do not know their tracking sites since the Web beacons are hidden to the users.
- Each beacon results in 2 messages between the browser and the tracking site, and adding a lot of beacons may become a problem for resource constrained mobile devices;

OAuth 2.0 (RFC6749) [9] is an IETF specification to enable cross-site authorizations sanctioned by users. An example use case is that a user asks an image processing service at site A to access his private online photos at site B, without giving site A his credentials on site B. OAuth 2.0 achieves this by using HTTP redirects between the two websites such that the user can request B to issue an access code to A after B authenticates the user. With the access code, A can get an access token to visit the authorized resources at B.

However, OAuth 2.0 has two fundamental limitations:

- It only supports unilateral authorization while mirror presence in particular, and call signaling in general, require two-way authorization between websites.
- The authorization scope is a predefined list of resources and it is difficult to extend for real-time communication and collaboration where resources are dynamically added and removed.

3. MIRROR PRESENCE

Without losing generality, we assume that each user is identified on the Web as a point in a 2-dimensional identity coordinate system: (ID, IS), where ID is a string that uniquely identifies a user at website IS. The "distance" between the points depends on the relations between them. The *equivalence* relation links two identity points that identify the same user. The *location* relation links first identity point to second one if the first point can locate the second point. Mirror presence is a way to establish the location relations between identity points based on the equivalence relations.

Similarly, we define the Web location of a user as a point in a 2-dimensional location coordinate system: (VS, VP), where VP is a hypertext page from websites VS the user is visiting. VS is necessary to forward call request messages to VP, when Web browsers do not act as Web servers. Here a natural distance metric between two location points is the minimum "click" distance between the two pages.

Under these definitions, the function of Web identity resolution is to map an identity point (ID, IS) to a location point (VS, VP) while the user is moving in the location coordinate system by pointing his Web browser at different URI.

When VS=IS, mapping (ID, IS)→(VS, VP) reduces to trivial mapping ID→VP on the same site.

Mirror presence system addresses the general situations where VS≠IS. The basic components and protocols of mirror presence are illustrated in Figure 2, where the user has two websites in his circle: site A with identity (Id1, A) and site B with identity (Id2, B), and his current location point is (A, PA).

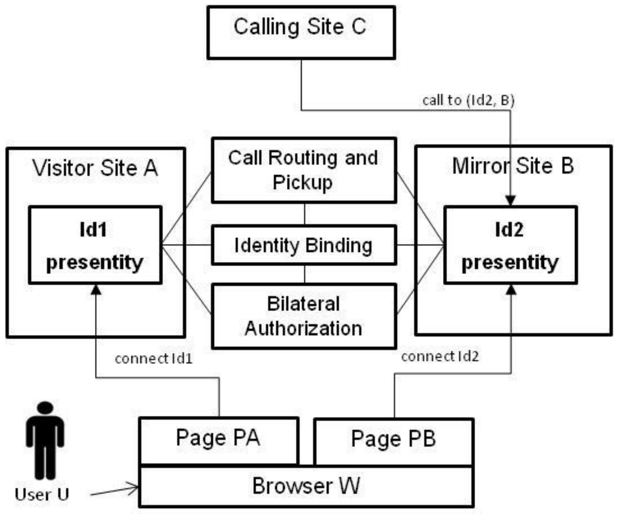


Figure 2: Basic mirror presence architecture

The basic idea is to bind distinct identity points in user's personal Web space through presence such that one identity point becomes the mirror image of another identity point. As the result, calls arrive at an identity point (ID, IS) without a location point can be automatically routed to the closest location point (VS, VP) through the bound identity points.

In Figure 2, the call request from site C to (Id2, B) will follow the route C→B→A→PA when Id1 and Id2 are bound by mirror presence, and the user only connects to site A. However, the same call will follow the route C→B→PB when the user connects to site B directly.

To realize this idea, mirror presence employs 3 layers of services: 1) bilateral resource authorization based on OAuth 2.0; 2) identity binding based on presence subscription and notification; and 3) presence driven call routing and pickup. The following sections will describe each layer in more details.

4. CROSS-SITE BILATERAL AUTHORIZATION

To secure presence subscription between Web identities, such that a user's Web location does not leak to unauthorized entities, we develop a bilateral authorization protocol by extending OAuth 2.0 [9]. With bilateral OAuth 2.0, a user can grant two websites to access each other's resources using the same number of messages as the unilateral OAuth 2.0. Furthermore, a user can start the bilateral authorization process from either website A or B and the end result for the sites will be the same.

The high level steps of bilateral OAuth 2.0 are illustrated below, where we denote the user by U and the browser by W:

1. U→W: user fills a HTML form to create mirror presence from identity Id1 on site A to identity Id2 on site B;
2. W→A: browser submits the form to site A;
3. A→A: site A generates an Authorization Request containing the following fields (new fields in bold font), where the optional **bi_code** field indicates that A wishes to engage bilateral authorization with B:
    a. response_type=code
    b. client_id=A
    c. redirect_uri={URI_A}
    d. bi_code={code_a}
    e. scope = session
    f. session={session_a}
4. A→W: site A redirects the browser to site B with the Authorization Request;
5. W→B: browser sends the Authorization Request to site B;
6. B→W: site B returns a page to ask the user to grant the Authorization Request;
7. U→W: user enters his credentials and grants the request;
8. W→B: browser sends the user grant to site B;
9. B→B: site B generates an Authorization Response containing the following fields (new fields in bold font), where the optional **bi_code** field indicates that B agrees to use bilateral authorization with A:
    a. code = {code_b}
    b. bi_code={code_a}
    c. session={session_b}
10. B→W: site B redirects the browser to site A with the Authorization Response;
11. W→A: browser sends the Authorization Response to site A;
12. A→B: site A sends an Access Request containing the following fields (new fields are in bold font):
    a. grant_type = authorization_code
    b. code = {code_b}
    c. redirect_uri = {URI_A}
    d. client_id = {Id1}
    e. token_type=session
    f. access_token={token_a1}
    g. expires_in={xsd:duration|xsd:dateTime}
    h. refresh_token={token_a2}
13. B→A: site B returns an Access Response containing the following fields (new fields in bold font):
    a. access_token = {token_b1}
    b. token_type = **session**
    c. expires_in = {**xsd:duration|xsd:dateTime**}
    d. refresh_token = {token_b2}
    e. scope=**session**

Although the process seems complicated as it involves 4 entities and two HTTP redirects (step 4 and step 10), the user is only involved twice at step 1 and step 7. At the end, site A obtains the following states:

1. session URIs between A and B: {session_a}, {session_b};
2. access tokens to B: {token_a1}, {token_a2};
3. access tokens from B: {token_b1}, {token_b2};

Similarly, site B obtains the following states:

1. session URIs between A and B: {session_a}, {session_b};
2. access tokens to A: {token_b1}, {token_b2};
3. access tokens from A: {token_a1}, {token_a2};

With these states in hands, sites A and B can put the appropriate tokens in the exchanged messages and they can authorize each other without user involvement. If these tokens expire, each site can use the refresh tokens to extend the mutual authorization relations.

Session is another extension in our approach to OAuth 2.0 in order to allow websites to authorize new resource access without going through the entire process. A websites can grant access to a resource by adding it to an authorized session and revoke the access by removing it from the session. Resource authorization thus becomes a local decision on whether a user can add or remove resources to a local session. Session can also enforce the resources to have the same access policy. For example, when a session is removed, all resources within the session become inaccessible. A user can also revoke the access token of a session to make the session inaccessible to other sites without removing the session.

Besides authorization code, the bilateral extensions presented here can also be used to transform three other OAuth 2.0 unilateral authorization modes into bilateral modes: 1) implicit grant; 2) owner password; and 3) client credentials. Since the protocol modifications are very similar, we will not discuss the details.

5. CROSS-SITE IDENTITY BINDING

Binding identity point (Id1, A) to identity point (Id2, B) is carried out automatically by the sites after successful bilateral authorization.

As illustrated in Figure 2, the user may connect to both site A and site B at the same time but using different pages PA and PB, site B needs to aggregate the *mirror presence* states from PA with the *direct presence* states from PB. For example, the use may set PA presence to "busy" and PB presence to "available." To facilitate this process, we create three presentity REST resources for each bound presentity on site B as shown in Figure 3.

The mirror presence resource is responsible to subscribe and receive mirror presence from PA (through site A). The direct presence resource is responsible to manage the user presence from page PB. The presentity resource is responsible to aggregate the presence states from the mirror presence and direct presence resources.

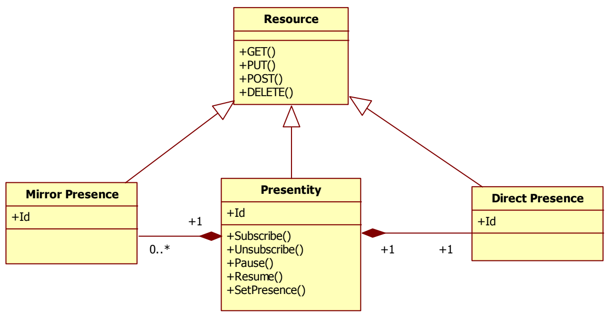


Figure 3: Presence aggregation architecture

With these resources, the mirror presence subscription proceeds automatically after successful bilateral authorization between Id1 and Id2 presentities:

1. B→A: the mirror presence resource at site B sends a HTTP POST subscription message containing the following fields:
    a. access_token={token_a1}
    b. sink_uri={URI_mirror_presentity}
    c. Web_beacon={URI_BA}
    d. source_id={Id1}
    e. sink_id={Id2}
2. A→B: site A returns a HTTP response message containing the following fields:
    a. access_token={token_b1}
    b. subscription_uri={URI_A}
    c. presentity_id={Id1}
    d. presence_state={state}

At this point, presence events from Id1 presentity on site A can flow to Id2 mirror presentity on site B. The user can cancel the subscription anytime by deleting the subscription resource. Users can set their presence at any time explicitly. But there is also a need for a website to detect a user's presence, since many events can change his presence, such as powering off the device or network outage. There are many ways for a website to automatically determine a user's presence, with different tradeoff of detection accuracy and resource requirement.

One way for site A to detect user presence automatically is to monitor WebSocket [22] connection. Whenever a Web socket is connected to site A, the Id1 presentity is considered "available" and whenever it is disconnected, the Id1 presentity is considered "unavailable."

Another way for presence detection is to combine Web beacon with WebSocket. In this approach, site B provides a Web beacon {URI_BA} to site A in the subscription request (step 1 field c). Site A then includes this beacon in the page returned to the user after he logs in. When the browser retrieves the beacon from site B, B knows that the user is available at site A.

Yet another way to detect user presence is to use timeout, which is a technique employed by many secure sites. In this approach, the site starts a timer when the user logs in. Whenever the user interacts with the site within the timeout, the timer is reset. If no interaction occurs during the timeout period, the site determines the user is absent.

Between the "on" and "off" presence states, the user can set Id1 presentity to any presence state permitted by site A. These presence events will be propagated by HTTP POST to the mirror presence resource at site B. The presence event includes the access token {token_b1} so that the message can be authorized by site B.

Mirror presence prefers direct presence over mirror presence in order to reduce network traffic from site A to site B and find the shortest call route:

1. B→A: whenever the direct presence resource detects user presence is available, it notifies the Id2 presentity, which instructs the mirror presence to pause the subscription to Id1 presentity on site A. Upon success, the Id2 presentity state will be set to the direct presence state.
2. B→A: whenever the direct presence resource detects user presence is unavailable, it notifies the Id2 presentity, which instructs the mirror presence to resume the subscription to Id1 presentity on site A and the positive response will include latest Id1 presentity state. Upon success, the Id2 presentity state will be set to the mirror presence state;

Following this method, when the user connects to both site A and site B, the mirror presence is suppressed, and a call to the user from site C will take the shortest route C→B→PB as expected.

6. CROSS-SITE CALL ROUTING AND PICKUP

The identity binding process can be applied to the personal Web space to create an overlay network whose nodes represent Web identity points of a user, and whose edges represent identity binding relations (Figure 4). In this network, any node can receive a call, and the node will route the call to the closest Web location following the reverse direction of the binding relations.

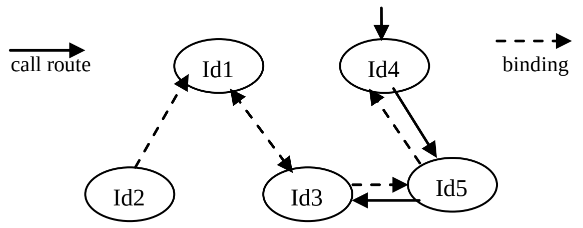


Figure 4: Presence overlay based on mirror presence

When a call is routed through more than one websites, it would be useful if the user can pick up the call at any site on the route to select the best call service. For example, if a call takes the C→B→A→PA route, the user can answer the call at either site A or B. However, this flexibility faces the following issues:

1. Identity authentication: because the call request is aimed at Web identity (Id2, B), the user has to prove to the site that his identity is Id2. If the user picks up the call at site B, this is not a problem because the user has Id2 on site B. If the user picks up the call at site A, this is a problem if A only authenticates user's identity Id1, and Id2≠Id1. However, the user can safely pick up the call at site A if both sites accept the same identity issued by a third-party Identity Provider;
2. User interface: The user interface may be different because each site offers different call control functions (e.g. transfer) and call history or the same functions and history through different GUI;
3. User privacy: The user's privacy context may be different because answering a call may reveal which site the user is currently visiting. For example, if the user picks up a call from route C→B→A→PA at site A, site C learns that user is at site A since the call response message will come from site A. However, if the user picks up the call at site B, no additional information is leaked to site C, since site C already knew the user is at site B;

To cope with these issues, mirror presence lets the user pick up the call at the call origin site, which is the first identity node in a call route. This process is described as follows:

1. C→B: site C sends a call request message from Id0 for identity (Id2,B) to site B;
2. B→B: Id2 presentity saves the call (offer SDP) and generates a new call request message with a pickup URI that identifies the saved call:
    a. from_id=Id0
    b. to_id=Id2
    c. to_site=B
    d. pickup_uri={URI_pickup}
3. B→A: Id2 presentity sends the new call request message to site A;
4. A→PA: site A forwards the call request message to HTML5 page PA over the Web socket and the page will alert the user the incoming call from Id0;
5. U→PA: the user answers the call;
6. PA→A|W: The page PA will take two actions:
    a. PA→A: the page sends a presence event to site A indicating the user is answering the call, to avoid the calling party hang-up or redial while the user tries to pick up the call;
    b. PA→W: the page PA instructs the browser to visit URI_pickup;
7. W→B: the browser visits URI_pickup;
8. B→W: site B returns a HTML5 page in a new window asking the user to authenticate to site B as Id2;
9. U→W: the user enters user name and password to the page;
10. W→B: the browser sends the user credential with URI_pickup to site B;
11. B→W: site B returns a HTML5 page that contains JavaScript code to complete the call signaling process with site C and the negotiated media will flow between the user browser and the calling party's browser;

If the user has been authenticated to site B, and the authentication cookie is saved by the browser, step 8 can be skipped and there will be no user involvement during call pickup. Site B may also embed a one-time authentication token in URI_pickup such that the browser is automatically authenticated and authorized to site B (the browser jumps to step 10 from step 7).

7. PROTOTYPE SYSTEM AND EXPERIMENTS

We implemented a prototype mirror presence system with the described protocols and services in a LAN environment. We simulated two independent websites New Jersey (NJ) and Shen Zhen (SZ) with two Tomcat 7.0.27 servers and used Chrome browsers (version 26.0.1410.42 m) to visit the Web servers and make audio/video calls.

We modified the open source Apache Oltu OAuth 2.0 Java Library [23] to implement the bilateral authorization module described in Section 4.

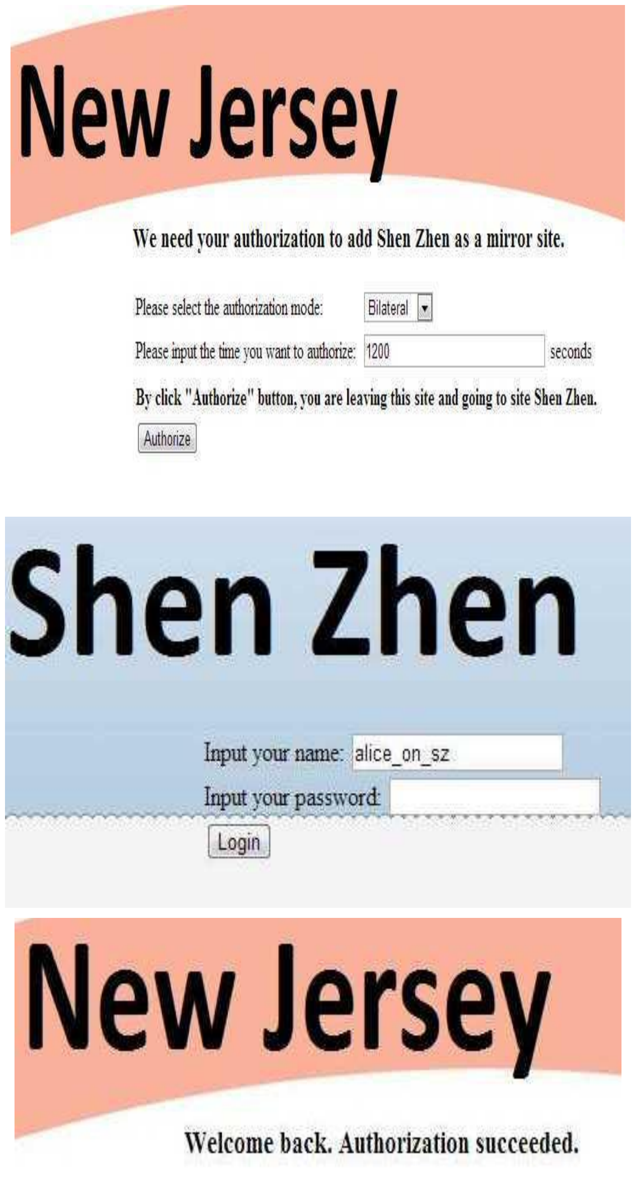
We implemented the presence subscription protocol in Sections 5 and the call pickup protocol in Section 6 in Java based on the Tomcat Servlet framework.

We also wrote the HTML5 pages and some JavaScript based on WebRTC JavaScript API and WebSocket API to allow two users Alice and Bob, who are online friends, to establish mutual mirror presence between the two sites.

To show the system is usable for average users, the following section illustrates the main user interfaces and interactions with screenshots.

7.1 Screenshots

When Alice logs in the NJ site and Bob logs in the SZ site, Alice appears on SZ as "unavailable" because there is no presence binding from identity alice_on_nj at NJ to identity alice_on_sz at SZ. The following screenshots (Figure 5) show the pages for Alice to bind her identity (alice_on_nj) at NJ site to the one (alice_on_sz) at SZ site using bilateral authorization. Alice first fills a form at NJ site to specify type of authorization and duration of the identity binding (first). After she clicks the "authorize" button, Alice is redirected to SZ site, whose page asks if she grants the presence biding (second). After Alice clicks the "login" button, she authorizes the binding and is redirected back to NJ site (third). After three pages, Alice's mirrors her presence from NJ to SZ.


New Jersey

We need your authorization to add Shen Zhen as a mirror site.

Please select the authorization mode: Bilateral

Please input the time you want to authorize: 1200 seconds

By click "Authorize" button, you are leaving this site and going to site Shen Zhen.

Authorize

Shen Zhen

Input your name: alice_on_sz

Input your password:

Login

New Jersey

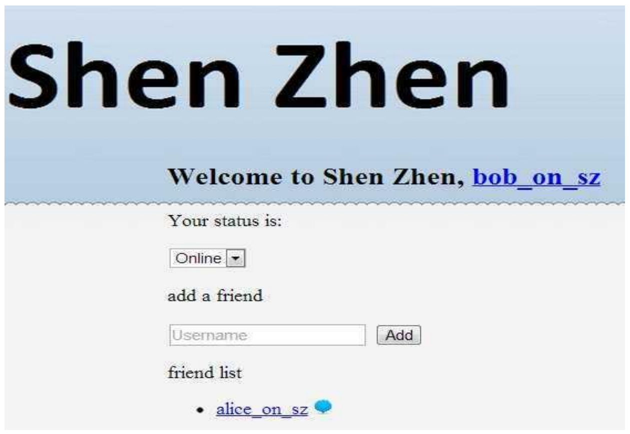
Welcome back. Authorization succeeded.

Figure 5: Alice creates mirror presence

Figure 6 shows that Bob at SZ now sees Alice's mirror presence (online) at the SZ site although Alice is actually on NJ site.

When Alice decides to visit SZ site and the system will automatically use this connection as Alice's direct presence at SZ. Figure 7 shows that after Alice connects to SZ site directly, Bob only sees her direct presence (busy) at SZ, as her mirror presence (available) from NJ site is paused.

Figure 8 shows Alice at NJ picks up a call coming from SZ. After Alice disconnects from SZ but still connects to NJ, her mirror presence will resume for SZ automatically and she appears as online for Bob at SZ. When Bob at SZ calls Alice, the call is presented to Alice as a hyperlink (top). To answer the call, Alice clicks the link, which opens a page at SZ site, asking her to log in (second). After she logs in, the video call is established (bottom).


Shen Zhen

Welcome to Shen Zhen, bob_on_sz

Your status is:

Online

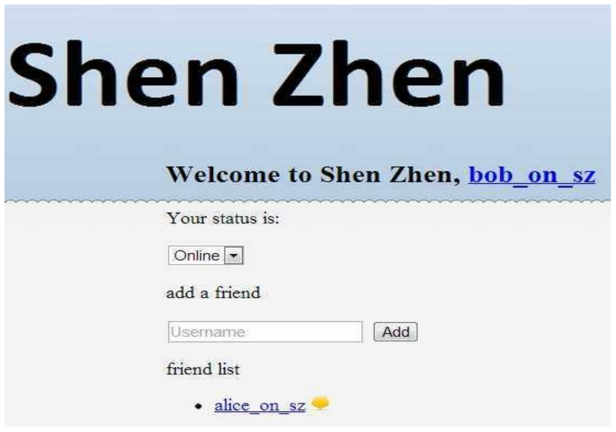
add a friend

Username Add

friend list

- alice_on_sz

Figure 6: presence states after mirror presence


Shen Zhen

Welcome to Shen Zhen, bob_on_sz

Your status is:

Online

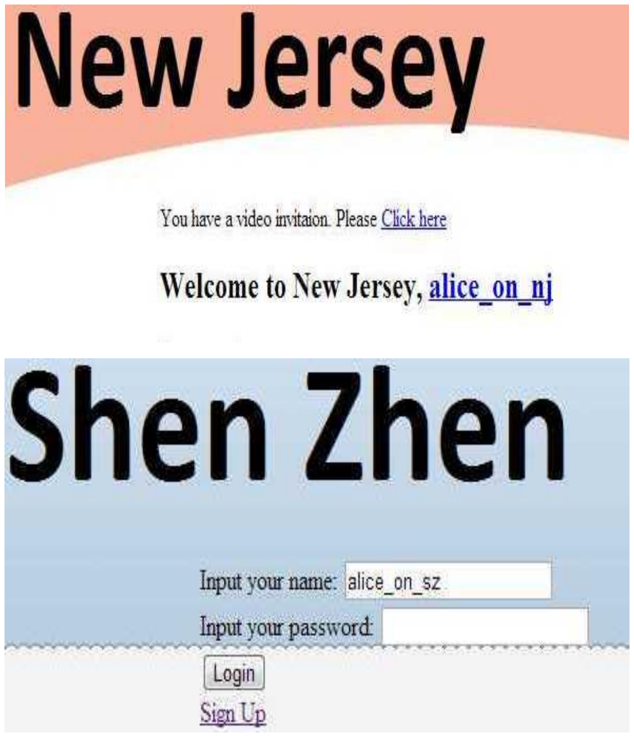
add a friend

Username Add

friend list

- alice_on_sz

Figure 7: presence states after direct connection


New Jersey

You have a video invitaion. Please Click here

Welcome to New Jersey, alice_on_nj

Shen Zhen

Input your name: alice_on_sz

Input your password:

Login

Sign Up

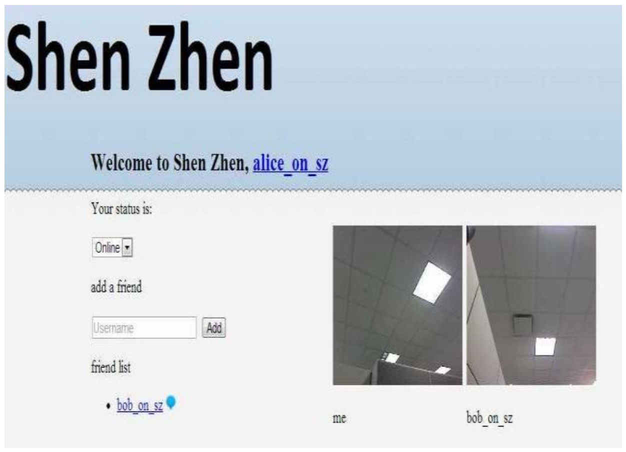


Figure 8: Alice at NJ picks up call from Bob at SZ

7.2 Performance Evaluations

Since mirror presence subscriptions are carried by users manually before making calls, their performance does not impact call process. Therefore, we focus on the latency of presence messages and call request messages in the following experiments.

In the first experiments, we measure the end-to-end and server processing delays of mirror presence events following the route: Alice's browser (WA) → New Jersey site (NJ) → Shenzhen site (SZ) → Bob's browser (WB). To measure the delay from WA to WB accurately, we put the two browsers on one machine while the two websites run on two different machines. The deployment of 3 networked machines is depicted in Figure 9.

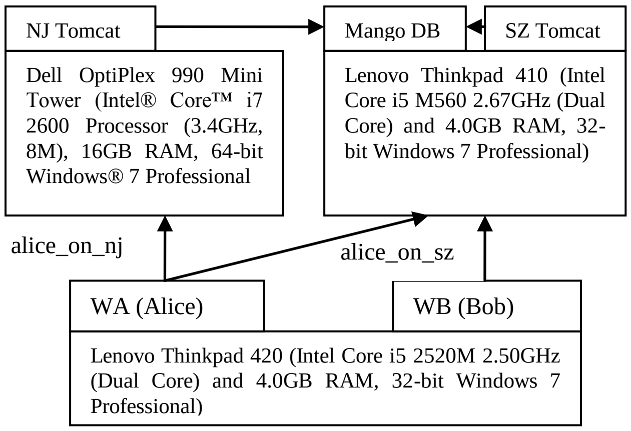

NJ Tomcat
Dell OptiPlex 990 Mini Tower (Intel® Core™ i7 2600 Processor (3.4GHz, 8M), 16GB RAM, 64-bit Windows® 7 Professional
Mango DB
SZ Tomcat
Lenovo Thinkpad 410 (Intel Core i5 M560 2.67GHz (Dual Core) and 4.0GB RAM, 32-bit Windows 7 Professional)
alice_on_nj
alice_on_sz
WA (Alice)
WB (Bob)
Lenovo Thinkpad 420 (Intel Core i5 2520M 2.50GHz (Dual Core) and 4.0GB RAM, 32-bit Windows 7 Professional)

Figure 9: Experiment System

Table 1 summarizes the end-to-end and server processing delays (in millisecond) for presence events averaged over 20 test runs.

Table 1: End-to-end and processing delays of presence events

hop	mean	std
NJ site	8.08	9.11
SZ site	3.92	0.64
WA→WB	21.56	4.25

The NJ performance is slower than the SZ performance despite the more powerful NJ machine. We believe this is because the NJ server accesses the database remotely while the SZ accesses it locally.

In the second experiments, we measure the call request delays using the same system for the presence delay experiment, but the call request message traverse the reverse direction of presence events. Table 2 summarizes the delays (in millisecond) averaged over 20 test runs.

Table 2: End-to-end and processing delays of call requests

hop	mean	std
SZ site	5.24	1.88
NJ site	0.64	3.20
WB→WA	14.44	5.37

These test results show the performance of the system is satisfactory as the end-to-end presence delay is about 21 ms and the end-to-end call request delay is about 14 ms.

Although our experiments did not test large scale deployment, we believe the approach is scalable for the following reasons. First, the state and computation for each user are very simple. For each user, a site needs to store a triple: (id, uri, state) and update it whenever the user location and state change. If a user connects to multiple mirror sites, the site needs to aggregate the presence, which can be carried out efficiently. Second, the presence states for the users on a site can be managed independently and processed in parallel. Third, the presence and call initiation messages are stateless and take small network bandwidth because they are typically very small (~1 KB). However, a mirror presence system needs to prevent spurious presence messages from entering the network, as these messages may create unnecessary network traffic and waste memory and processing cycles.

8. CONCLUSIONS

This paper describes a secure Web identity resolution and call control system, mirror presence, for WebRTC, with three contributions: 1) cross-site bilateral Web authorization; 2) cross-site Web identity binding based on presence subscription and notification; 3) cross-site Web call routing and call pickup driven by presence. We implemented a prototype system in a LAN environment and the preliminary experiments showed the approach is feasible.

The advantages of this approach for the Web users are:

- Presence based call routing through late binding of Web identity to Web locations can reduce missed calls while the users travelling and browsing between websites;
- A user can reduce the number of concurrent connections to his personal Web space without losing calls when he moves in the space;
- A user can control all layers of the system, such as add and remove mirror sites, and switch browsers without having to reestablish mirror presence relations;
- Distributed mirror sites and multiple mirror presence relations eliminate single point of failure for call routing;

The advantages for the websites are:

- Increased Stickiness
  - A mirror site can keep its users as the users see their contacts are present on the same site;
  - A visitor site can keep its users as the users will not miss any calls;
- Reduced Network Traffic
  - Small presence messages prevent large and repetitive call messages when the target users are not present;
  - Mirror presence is paused/resumed automatically to reduce unnecessary presence notifications;
- Reduced Development Cost
  - Mirror presence can be easily implemented within current Web architectures and protocols without introducing any new centralized system;
  - Mirror presence is agnostic to identity protocols, and the websites have freedom to select which ones to implement;
  - The added cost of serving mirror presence messages is compensated by the increased number of users a site retains;

For the future work, we plan to improve the call routing aspect of the system to seamlessly integrate call control and identity authentication across websites for the users.

9. REFERENCES

[1] M. Handley et al (ed): SDP: Session Description Protocol, April 1998, http://tools.ietf.org/html/rfc2327 (last visit: October 2, 2013)

[2] H. Schulzrinne et al (ed): RTP: A Transport Protocol for Real-Time Applications, July 2003, http://tools.ietf.org/html/rfc3550 (last visit: October 2, 2013)

[3] J. Rosenberg (ed): Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols, April 2010, http://tools.ietf.org/html/rfc5245 (last visit: October 2, 2013)

[4] J. Rosenberg et al (ed): SIP: Session Initiation Protocol, June 2002, http://www.ietf.org/rfc/rfc3261.txt (last visit: October 2, 2013)

[5] OpenID Foundation: http://openid.net/wg/ (last visit: October 2, 2013)

[6] Henry Story (ed): WebID-TLS, WebID Authentication over TSL, W3C Editor's Draft, 08 July 2013, https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/tls-respec.html (last visit: October 2, 2013)

[7] Mozilla Persona: https://developer.mozilla.org/en-US/docs/Mozilla/Persona?redirectlocale=en-US&redirectslug=Persona (last visit: October 2, 2013)

[8] http://www.w3.org/2011/04/Webrtc-charter.html (last visit: October 2, 2013)

[9] D. Hardt (ed): The OAuth 2.0 Authorization Framework, October 2012, http://tools.ietf.org/html/rfc6749 (last visit: October 2, 2013)

[10] R. Jesup et al (ed): RTCWeb Data Channels, February 25, 2013, http://tools.ietf.org/html/draft-ietf-rtcWeb-data-channel-04 (last visit: October 2, 2013)

[11] E. Rescorla (ed): RTCWEB Security Architecture, January 22, 2013, http://tools.ietf.org/html/draft-ietf-rtcWeb-security-arch-06 (last visit: October 2, 2013)

[12] Adam Bergkvist et al (ed): WebRTC 1.0: Real-time Communication Between Browsers, W3C Editor's Draft, 03 June 2013, http://dev.w3.org/2011/Webrtc/editor/Webrtc.html (last visit: October 2, 2013)

[13] IETF RTCWeb Charter: http://tools.ietf.org/wg/rtcWeb/charters (last visit: October 2, 2013)

[14] J. Rosenberg (ed): A Presence Event Package for the Session Initiation Protocol (SIP), August 2004, http://tools.ietf.org/html/rfc3856 (last visit: October 2, 2013)

[15] A. Houri et al (ed): Presence Interdomain Scaling Analysis for SIP/SIMPLE, August 27, 2009, http://tools.ietf.org/html/draft-ietf-simple-interdomain-scaling-analysis-08 (last visit: October 2, 2013)

[16] P. Saint-Andre (ed): Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, October 2004, http://xmpp.org/rfcs/rfc3921.html (last visit: October 2, 2013)

[17] P. Saint-Andre (ed): Interdomain Presence Scaling Analysis for the Extensible Messaging and Presence Protocol (XMPP), January 16, 2008, http://tools.ietf.org/html/draft-saintandre-xmpp-presence-analysis-03 (last visit: October 2, 2013)

[18] http://en.wikipedia.org/wiki/Find_Me/Follow_Me (last visit: October 2, 2013)

[19] Frédéric Firmi, The Evolved Packet Core, http://www.3gpp.org/The-Evolved-Packet-Core (last visit: October 2, 2013)

[20] http://en.wikipedia.org/wiki/Web_bug (last visit: October 2, 2013)

[21] http://www.w3.org/2011/tracking-protection/ (last visit: October 2, 2013)

[22] Ian Hickson (ed): The WebSocket API, W3C Candidate Recommendation, 20 September 2012, http://www.w3.org/TR/Websockets/ (last visit: October 2, 2013)

[23] Apache Oltu: http://oltu.apache.org/ (last visit: October 2, 2013)