Paper title: User reachability in multi-apps environment.
Speaker: Kundan Singh. Track: Applications II.

Hello everyone. My name is Kundan Singh. My presentation is on user reachability in multi-apps environment.

Today's communication systems often create silos or islands of communication where a user of one service cannot easily talk to that on another.

WebRTC or web real-time communication is an emerging technology that allows plugin-free browser-to-browser (peer-to-peer) audio/video flows. It basically enables a website to quickly become a standalone communication service provider.

This creates too many such islands, each controlled by a WebRTC capable website.

Many websites are already using WebRTC, and the number is growing continuously. These websites often require their users to use only the site specific apps, on web or mobile, and do not interoperate with other websites.

My paper presentation is about why we are in a multi-apps environment and how do we do user reachability. It is a systems paper with description of a software -- its architecture and implementation.

In the past, people have attempted to bridge the communication islands in many ways.
Iterate manually or use presence, e.g., to check if user is online on Gtalk or Yahoo before sending a message, or try Hangout video and if not picked up, dial out by phone.
One service can federate with another, e.g., collaboration between Yahoo and Microsoft could allow you to seamlessly send message to your Yahoo friends from your MSN client. Such pair-wise service federation works only for few popular services, and does not scale with growing number of services.
Projects such as hookflash, and matrix.org are creating a global location service for WebRTC. However, it is hard to convince popular websites to follow their protocol or to work with their systems.
Pidgin and Trillian are multi-protocol apps that allow you to login to and use many different instant messaging services. Scaling to hundreds of protocols is hard. Moreover, proprietary signaling used by the website makes it harder to include in such an app.

In fact, multi-protocols reachability is not the same as multi-apps reachability. And the latter is what is desired.
Fig.2 shows that users have installed multiple apps, and use the right app to reach the right person.

The paper is basically about this multi-apps reachability, and how to automate the process. And the lessons learned from practical experience.
First, we decouple the contacts from the communication apps. The contacts and apps are independently managed or installed by the user.
*We have developed a contacts-app called **Strata Top9**. It is a front-end to launch and interact with other communication apps to reach a user.*
The user independently installs the communication apps. And Strata determines the right app with automatic fallback, e.g., use the video app, and if fails, try a phone call.

Let me show a quick video demonstration of Strata Top9.

Strata is mobile-first application, but can be installed on desktop as well as mobile. The user interface looks the same on all platforms.

This user is signed in as Richard Henley, and has bunch of contacts populated in the top-9 list. User can click and hold to select how to reach the target, or to edit the contact item, e.g., the preferred mode.

I will initiate a default call, which will use video mode if possible, to user Green. This is a two party video call between this device and another tablet device I am logged in on. In the settings, you can change the devices used or mute them. It is possible to invite more people to make it a multi-party call.

Let me shown an incoming call. The notification can be clicked to answer or closed to decline. Or can be downgraded to other modes. I will answer using message mode which opens a text chat, with standard text chat features. You can also share attachments, e.g., pictures or camera recorded video. There is also a way to send one time multimedia message instead of a text chat session.

The user can edit her reachability information and ordered preference list. For example, I add a new video reachability on my enterprise system based on Avaya IP office product. Now if I can receive a call on IP office system, this app receives it. Also if I am not available on Strata, then an incoming Strata video call will automatically fall back to my IP office number.

Strata also has programmable user reachability policies for caller and receiver. For example, I can write a small JavaScript like code snippet to say that for dialing out to anyone in avaya.com, prefer to use the voice or phone mode. The programmable policies override the user reachability list in settings.

This user bhalla is not logged in on Strata, but has his reachability set up for his enterprise video conference number on Avaya Scopia. When I try to reach him on video, it launches that app's video client. When I try to reach him on voice, it dials in that conference bridge number including the conference access code.

In fact this app, engagement dialer, can be used as a generic phone dialer in Strata. I can create a phone device contact, and use it to dial phone numbers. We have integration with our corporate directory, so you can search users by name as well.

There is special type of dynamic contact I can add from my calendar, as the next calendar meeting. Once it connects to my microsoft exchange service, it gets my next meetings and shows them as clickable contacts. It shows two ongoing meetings at the time. The first is on Scopia video conference bridge. The second is a meeting on a different conference system. If I click on it it will join the appropriate bridge with the right mode.

Strata has integration with many other modules and communication systems shown here. I showed a few of them already. And new ones are very easy to integrate.

A user can also see which other devices he is logged in from. And can interact or transfer the calls between devices, e.g., I can click here to join the ongoing conversation of my other device on mobile.

This shows the same user logged in using the same client app, but using a different authentication token, which connects the app to a different database, and an entirely different app experience. This is how we implemented multi-tenancy in Strata, using database namespace.

Consider the scenario shown in Fig.1.

It shows about 10 users some of whom work at First Hospital. They are using three different communication services.

Richard on the left is on his social Strata app with its contacts. The preferred mode - voice, video or message - are shown in each contact. The mode preference can be from caller or receiver.

Alice on the right is on the Strata app downloaded from the First Hospital website, and customized for that business. Some of the top 9 boxes are pre-populated, e.g., with on-call nurse, or hospital billing department. Alice can further add or edit the boxes for more contacts, e.g., her child's pediatrician.

When Alice attempts to reach the on call nurse, the Strata tries preferred video mode call first using the hospital video conference service. If that fails to reach target, it dials out the phone number by launching the native phone device in her smart phone.

The contacts can be dynamic, e.g., the on call nurse changes as the staff changes shift. Furthermore, Alice can add "billing for specific insurance plan" to reach the right person instead of navigating through voice prompts, or can reach the available nurse or doctor who can deal with "natural pregnancy" by keyword search.

The paper also summarizes the important system requirements to support the scenarios I described.

In particular, the communication apps should be independent of each other and that of the contacts app. Both the caller and the receiver should be able to influence the reachability decisions. And number 5, there should be a minimum reachability in the form of phone number and email address.

The rest of paper presentation shows more details of the system architecture and implementation.

Fig.3 to 6 present some basic concepts in the system architecture.

We separate the contacts and reachability apps. A contact item may be prepopulated by the service, e.g., on call nurse, or managed by the user, or it could be obtained from the underlying device's contact list or from external context such as web browsing or users in the local area network.

There are two types of communication apps as shown here. A dialer only does outbound request to reach a user, and once launched it does not need to return to the contacts app. A communicator can return to the contacts app for intermediate decisions or to apply policy, e.g., on an incoming call request, user can downgrade or pick a different app to connect.

There are three types of handoffs considered shown in Fig.4. (a) device handoff, e.g., move a video call from mobile to desktop using the same app/service, (b) app handoff, e.g., change a video call on one app to a voice call on another on the same device, or (c) call component handoff, e.g., add a mobile touch-input white-board to a desktop call or add a desktop screenshare to a mobile video call.

Strata and many of our communication apps are based on resource based software architecture shown in Fig.5. It enables loosely coupled endpoint driven apps that use a light weight resource server, and mash up at the data level using shared data. And all the application logic for contact list, reachability or call control, are implemented in the endpoint. Further details are referred in the paper.

In Fig.6, the left side shows proactive user presence, where the user is logged-in with all the services, and the caller knows which one to use before a call attempt. On the other hand, we prefer to use on-demand reachability shown on the right, where the apps are tried one by one to reach the target user, and the first one to succeed is connected.

There are many reasons detailed in the paper for our design decision, but in summary proactive presence does not work well with diverse multi-apps systems.

User's reachability consists of one or more reachability items. A reachability item is defined as a triplet shown here. The first is mode, which is one of phone, video or message. Phone and voice identical, and indicate a voice communication. The second is the app, and the third is the reachability information specific for that app, e.g., a phone number or user ID.

Fig.7 in the paper describes how the user's reachability is derived based on various parameters such as dynamic contact, selected mode or caller or receiver policies.

The caller and receiver policies to manipulate the reachability are illustrated here.

(a) To reach my colleagues, prefer video and avoid my personal instant messenger.	if (receiver.email =~ "*@office.com") {
	prefer("video");
	exclude("message", "AIM");
	}

(b) Always call my cell after office hours irrespective of caller's preferred mode, and stop further policy lines.	if (now.hh >= 17) {
	choose("phone","Phone","+1212123456");
	break;
	}

(c) When I am traveling, only receive message mode; and fallback to my personal messenger service.	if (location.address.country != "India") {
	include("message");
	deprecate("message", "AIM", "alice");
	}

The policy is currently specified in a JavaScript-like language with limited constructs. It allows regular expression matching, as well as simple if-else control. Some data objects such as receiver, caller, current time "now" or location can be used as shown above. Some functions such as prefer, deprecate, include, exclude, and choose are available to manipulate the reachability list. The details are in the paper.

Besides the contacts app, we have implemented several communication apps to cover a wide range of enterprise scenarios. Fig.8 shows only the apps that deal with WebRTC, but there are more integrated.

The first app named Vclick is the default communicator app in Strata. There will be another presentation on Vclick tomorrow in Systems and Architecture track.

These apps in Strata cover a wide range of scenarios, e.g., client-server media path vs. peer-to-peer media flows, or dialers vs. communicators, or experimental vs. commercial systems, or thin-client vs thin-server.

All these apps including the Strata contacts app are implemented using Chrome Cordova Apps framework - which allows me to write the client application once in HTML5, JavaScript and CSS, and be able to run in range of platforms. For example, as a web application, or as an installed application on desktop as well as mobile. Some details and references are in the paper.

There are many other details in the paper. I will highlight some problems here, and leave it to interested readers to read the paper.

What's more in the paper?

1. Why did we choose only top "9"?
2. What is received contacts page?
3. What is person vs non-person contact?
4. Can same contact fill multiple slots?
5. How is an external app launched on web, desktop vs. mobile?
6. What is mode fallback vs device fallback?
7. What are the problems with proactive presence?
8. What responsibilities lie with contact list vs. individual app?
9. How is location data obtained?
10. What methods are supported in policy script?
11. Why is policy not applied in active call?
12. Why do we support only sequential fallback? not parallel?
13. How do we create cross-platform apps?
14. What are the specific apps that are integrated in Strata?
15. How can data-mining of past conversations be used to populate contacts?
16. How does web browser context populate dynamic contacts?
17. When are endpoint driven apps useful?
18. How are external apps launched?
19. What kind of contextual input can be used?
20. What is data-level mashup of apps?

In summary, focus of our work is on user driven reachability and policy decisions, unlike a global location service or pair-wise federations, and that makes our system useful in practice for emerging WebRTC apps.

To conclude, WebRTC promotes a multi-apps environment. And we present a system where users decide how they want to be reached. Our architecture separates the user contacts from the individual reachability apps, supports user and endpoint driven reachability policies, and has real-world implementation.

That concludes my presentation, on why we are in multi-apps enviroment and how do we do user reachability.